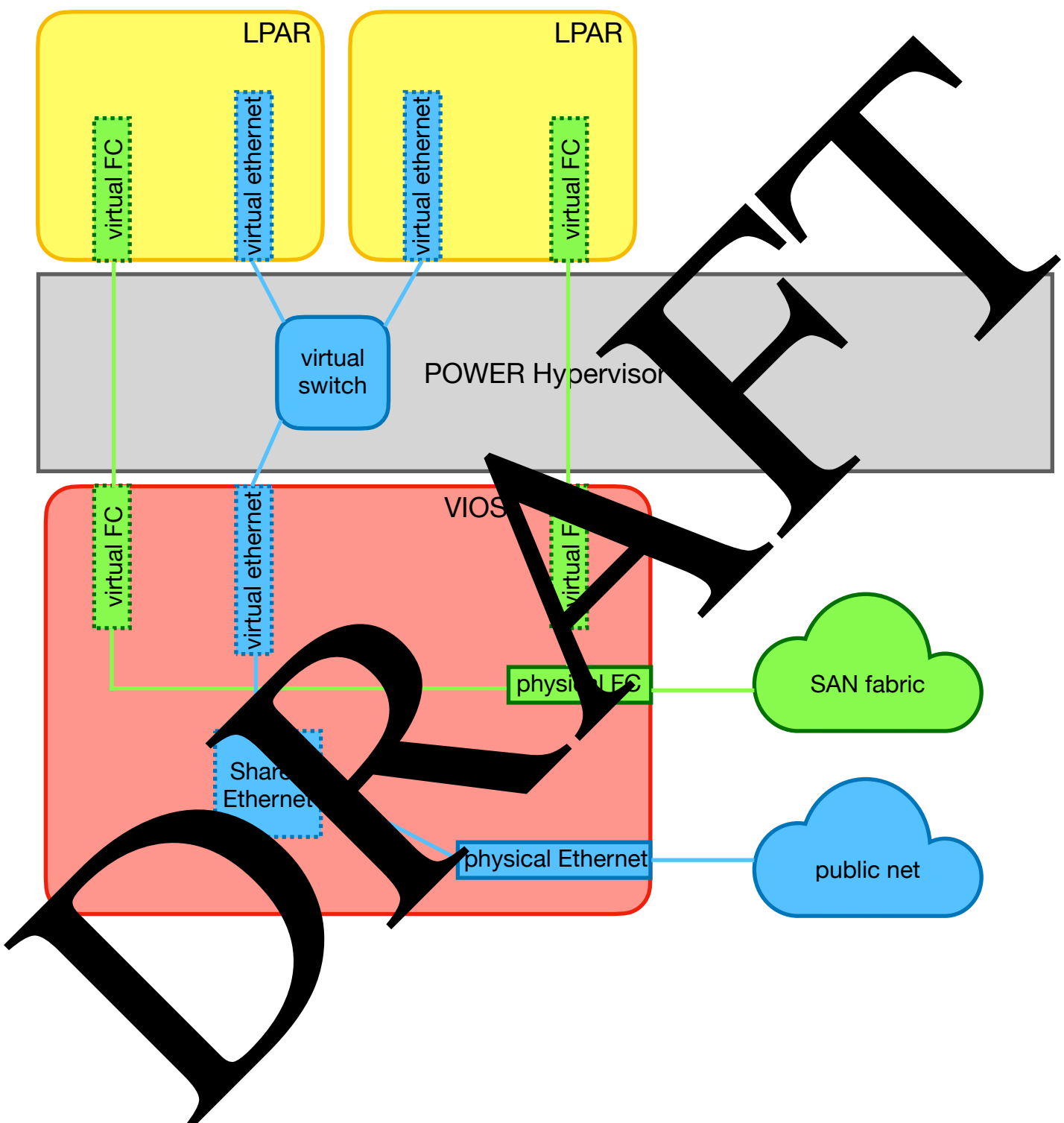
Virtualisierung mit PowerVM

Praktische Einführung



Armin Schmidt

https://www.powercampus.de

https://www.powercampus01.com

Zweite Edition (Februar 2023)

Diese Version des Dokuments bezieht sich auf die Version 1.7.1.0 des LPAR-Tools.

Vorwort

Einführung

IBM PowerVM ist eine Virtualisierungslösung von IBM für IBM Power Systeme. Das *LPAR-Tool* ist ein kommandozeilenbasiertes Tool von PowerCampus 01 zur einfachen Administration von IBM PowerVM Umgebungen.

Ziel des Handbuchs ist es die Virtualisierungskonzepte von IBM PowerVM vorzustellen und zu erläutern, sowie die Konfiguration und Administration von IBM PowerVM zu zeigen. Dabei wird an allen Stellen das *LPAR-Tool* verwendet, womit das Handbuch gleichzeitig als Einführung für das *LPAR-Tool* dienen kann. Der Schwerpunkt liegt allerdings auf der PowerVM Virtualisierung.

Da es sich um eine Einführung in die Virtualisierung mit PowerVM handelt, sind die Themen nicht bis in die letzten Details dargestellt. Zur Vertiefung sollte hier die IBM Dokumentation zu Rate gezogen werden.

Das Handbuch setzt folgendes voraus:

- grundlegende Kenntnis zum Arbeiten auf der Kommandozeile eines UNIX Systems
- grundlegendes Verständnis von Virtualisierung ist hilfreich, wird aber nicht vorausgesetzt

Dieses Handbuch kann über den Download-Bereich auf der PowerCampus^{01} Webseite heruntergeladen werden:

- *https://www.powercampus.de* oder *https://www.powercampus01.com*

Weitere Informationen

Weitere Informationen zum *LPAR-Tool* sind im Produkt-Bereich auf der PowerCampus^{01} Webseite verfügbar:

- *https://www.powercampus.de* oder *https://www.powercampus01.com*

Weitere Informationen zu IBM PowerVM findet man auf der Website von IBM:

- *https://www.ibm.com*

1. Einführung in IBM PowerVM

IBM PowerVM ist eine Virtualisierungslösung von IBM speziell für Power Systeme. Sie besteht aus einer Kombination von spezieller Hardware und Software. IBM PowerVM besteht im wesentlichen aus den folgenden Komponenten:

- Spezielle Power Hardware mit Firmware (POWER Hypervisor), welche das Teilen von Ressourcen und weitere Features ermöglicht.

- Den Virtual-I/O-Server, als spezielle Service Partition, welche das Teilen von Disk- und Netzwerk-Ressourcen ermöglicht. Für erhöhte Redundanz können mehrere Virtual-I/O-Server auf einem Power System konfiguriert werden.

Die durch PowerVM erzeugten Gast-Systeme werden gerne als LPARs (Logical PARtition) bezeichnet. Neuerdings wird aber auch hier häufig von VMs (Virtual Machine) gesprochen. Wir haben in diesem Dokument den Begriff LPAR anstelle von VM beibehalten. Als Gast-Betriebssysteme können AIX, IBM i und Linux verwendet werden, wobei es sich bei Linux dann um ein POWER-basiertes Linux handelt.

1.1. PowerVM Features

Im Folgenden sollen einige der Technologien und Features von PowerVM kurz aufgeführt werden.

PowerVM Hypervisor (PHYP): Diese Funktionalität wird durch die Hardware-Plattform in Kombination mit System Firmware für die Power Server zu Verfügung gestellt. Der Hypervisor ist letztlich die Grundlage für jegliche Virtualisierung auf einem Power System.

Logical Partition (LPAR): LPARs werden über den Hypervisor bereitgestellt. Ursprünglich konnten einer LPAR nur dedizierte Hardware-Komponenten und komplette Prozessoren zugeteilt werden, nur der Speicher war geteilt. Im Laufe der Power Systems Generationen wurden die Möglichkeiten aber immer weiter erweitert (Micro-Partition, Dynamic Logical Partition), wobei der Begriff LPAR aber beibehalten wurde.

Micro Partition: Die Micro Partition erlaubt das Teilen eines Prozessors zwischen verschiedenen Partitionen. Die Micro Partitionen bekommen dabei Anteile eines Prozessors zugeordnet, man spricht auch von sogenannten Shared Prozessor Partitionen.

Dynamic Logical Partition (DLPAR): Virtuelle Ressourcen (CPU, Memory, physikalische Adapter und virtuelle Adapter) können zur Laufzeit der Partition hinzugefügt oder weggenommen werden (Support durch das entsprechende Betriebssystem vorausgesetzt). Damit können Ressourcen dynamisch an den Bedarf einer Partition angepasst werden.

Shared Prozessor Pools (SPP): Partitionen können Shared Prozessor Pools zugewiesen werden, damit kann man den Verbrauch von Prozessor Ressourcen von Partitionen auf die im Pool verfügbaren Ressourcen begrenzen.

Virtual-I/O-Server (VIOS): Es handelt sich hierbei um eine spezielle Service Partition mit einem auf AIX basierenden speziell erweiterten Betriebssystem für die Unterstützung einer Reihe von Virtualisierungsfunktionen. Über Virtual-I/O-Server können Netzwerk-Adapter (Virtual Ethernet) und I/O-Adapter virtualisiert werden (Virtual SCSI und Virtual FC).

Virtual Ethernet (VETH): Client Partitionen können mit Hilfe von virtuellen Ethernet Adaptern im Netzwerk kommunizieren, ohne dafür eigene physikalische Ethernet Adapter zu besitzen.

Virtual SCSI (VSCSI): Mit Hilfe der Virtual-I/O-Server können Client Partitionen über einen virtuellen SCSI-Adapter auf Disks zugreifen, ohne hierfür einen eigenen physikalischen I/O-Adapter zu besitzen. Die notwendigen physikalischen Adapter gehören den Virtual-I/O-Servern und können damit geteilt von vielen Partitionen gemeinsam genutzt werden. Die Disks müssen den virtuellen SCSI Adaptern zugeordnet werden.

Virtual FC (VFC): Im Unterschied zu Virtual SCSI erlaubt Virtual FC das Zuordnen eines virtuellen FC Adapters direkt auf einen physikalischen FC Adapter. Es müssen nicht mehr wie bei VSCSI die einzelnen Disks den virtuellen Adaptern zugeordnet werden, was die Administration deutlich erleichtert.

Live Partition Mobility (LPM): Dieses Feature erlaubt die online Verschiebung einer aktiven Partition von einem Power System auf ein anderes Power System. Alle Applikationen und das Betriebssystem laufen während der online Verschiebung einfach weiter. Aus Sicht der Applikationen ist die Verschiebung transparent.

Active Memory Expansion (AME): Durch die Komprimierung von Hauptspeicher kann zusätzlicher verfügbarer Hauptspeicher gewonnen werden. Die gewünschte Komprimierung kann dabei vorgegeben werden. Damit kann man z.B. aus 32 GB physikalischem Hauptspeicher und einem Kompressionsfaktor (AME-Faktor) von 1.5 48 GB Hauptspeicher für eine Partition gewinnen. Das Betriebssystem und alle Applikationen sehen 48 GB verfügbaren Hauptspeicher.

Single Root I/O Virtualization (SR-IOV): Bei dieser Art der Virtualisierung wird kein Virtual-I/O-Server mehr benötigt. Die Virtualisierung findet in Hardware direkt auf dem physikalischen Adapter statt. Bei PowerVM ist dies aktuell auf SR-IOV fähige Netzwerk-Adapter beschränkt. Die Bandbreite der SR-IOV Ethernet-Ports lässt sich auf die einzelnen Partitionen aufteilen.

Virtual Network Interface Controller (vNIC): Erlaubt bei Ausfall eines SR-IOV Ethernet-Ports einen automatischen Failover auf einen anderen SR-IOV Ethernet-Port. Hierzu wird aber wieder die Unterstützung von Virtual-I/O-Servern benötigt.

Einige weitere Features von PowerVM wurden in dieser Übersicht nicht berücksichtigt.

1.2. Der POWER Hypervisor

Die meisten Virtualisierungsfunktionen sind nur durch den POWER Hypervisor möglich. Der Hypervisor bietet die Möglichkeit physikalische Ressourcen aufzuteilen und den Partitionen zur Verfügung zu stellen. Dabei stellt der Hypervisor sicher, das jede Partition nur Zugriff auf die eigenen Ressourcen hat, er isoliert die einzelnen Partitionen voneinander. Jede der Partition kann dabei eine eigene Instanz eines Betriebssystems verwenden, die unterstützten Betriebssysteme sind AIX, IBM i und Linux, oder Virtual-I/O-Server.

In Bild 1.1 ist die Virtualisierung mittels Hypervisor skizziert. Die Hardware Schicht, mit allen physikalischen Ressourcen, wie CPUs, Speicher und physikalischen Slots, liegt unterhalb des Hypervisors. Partitionen haben keinen direkten Zugriff auf die Hardware Schicht. Ein Zugriff ist nur über den Hypervisor möglich. Der Hypervisor kann dabei Ressourcen dediziert zur exklusiven Nutzung durch eine LPAR zuweisen, z.B. dedizierte CPUs oder physikalische Slots oder physikalische Speicherbereiche, er kann aber je nach Art der Ressource auch eine gemeinsame Nutzung durch mehrere LPARs erlauben, z.B. Shared Prozessoren oder Active Memory Sharing. Bei

Shared Prozessoren, bekommt eine LPAR nur einen vorgegebenen Anteil an einem Prozessor, sie kann den Prozessor dann nur zeitweise nutzen. Die Aufteilung auf die verschiedenen LPARs erfolgt durch den Hypervisor in einem Zeitscheiben-Verfahren.

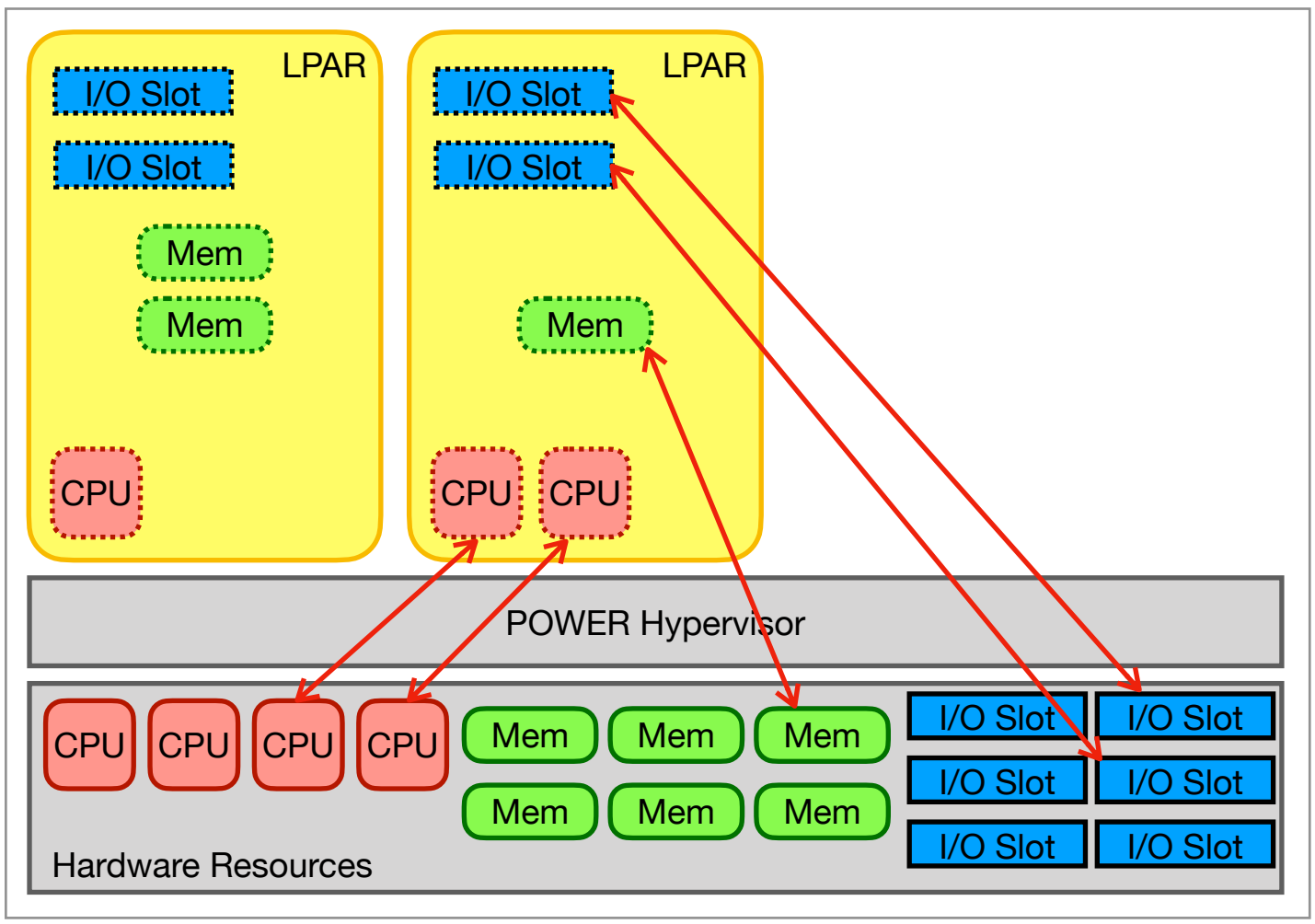


Bild 1.1: Virtualisierung mit PowerVM Hypervisor

Dem POWER Hypervisor sind keine speziellen Prozessor Ressourcen zugewiesen. LPARs kommunizieren mit dem POWER Hypervisor über Hypervisor calls (*hcalls*). Unter AIX lässt sich dies mit dem Kommando *lparstat* (Option „*-h*“ oder „*-H*“) genauer überwachen.

1.3. PowerVM Editionen

Es gibt 3 verschiedene Editionen von PowerVM. Diese unterscheiden sich im Umfang der unterstützten Virtualisierungsfunktionen.

1.3.1. PowerVM Standard Edition

Die PowerVM Standard Edition unterstützt die folgenden Features:

- Micro Partitionierung
- Virtual-I/O-Server (VIOS)
- Shared Prozessor Pools
- N-Port ID Virtualisierung (NPIV)
- Shared Storage Pools
- Single Root I/O Virtualisierung (SR-IOV)
- Virtual Network Interface Controller Adapter (vNIC)

- Thin Provisioning
- Suspend und Resume auf POWER8 (oder höher) Systemen
- PowerVM NovaLink

1.3.2. PowerVM Enterprise Edition

Die PowerVM Enterprise Edition unterstützt die folgenden Features:

- Live Partition Mobility (LPM)
- Active Memory Sharing (AMS)
- Remote Restart
- Power Virtualization Performance (PowerVP)
- Micro Partitionierung
- Virtual-I/O-Server (VIOS)
- Shared Prozessor Pools
- N-Port ID Virtualisierung (NPIV)
- Shared Storage Pools
- Single Root I/O Virtualisierung (SR-IOV)
- Virtual Network Interface Controller Adapter (vNIC)
- Thin Provisioning
- Suspend und Resume auf POWER8 (oder höher) Systemen
- PowerVM NovaLink

1.3.3. IBM PowerVM, Linux Edition

Die IBM PowerVM Linux Edition unterstützt die folgenden Features:

- Active Memory Sharing (AMS)
- Live Partition Mobility (LPM)
- Remote Restart
- Power Virtualization Performance (PowerVP)
- Micro Partitionierung
- Virtual-I/O-Server (VIOS)
- Shared Prozessor Pools
- N-Port ID Virtualisierung (NPIV)
- Shared Storage Pools
- Single Root I/O Virtualisierung (SR-IOV)
- Virtual Network Interface Controller Adapter (vNIC)
- Thin Provisioning
- Suspend und Resume auf POWER8 (oder höher) Systemen

- PowerVM NovaLink

1.3.4. PowerVM Express Edition

Die früher angebotene PowerVM Express Edition mit einer minimalen Zahl an Features wird nicht mehr angeboten.

2. Einführung in das *LPAR-Tool*

Das *LPAR-Tool* ist ein Kommandozeilen basiertes Tool zur einfachen Administration von PowerVM Umgebungen. Es können beliebig große PowerVM Umgebungen mit mehreren HMCs und beliebig vielen Managed Systems verwaltet werden. Das *LPAR-Tool* wurde mit den folgenden Zielen entwickelt:

- Leichte Benutzbarkeit: Das *LPAR-Tool* soll leicht und intuitiv in der Benutzung sein. Bei allen Kommandos sind sinnvolle Defaults hinterlegt, so daß nicht jedes Detail auf der Kommandozeile angegeben werden muß. Damit sollte auch ein nicht so versierter Benutzer PowerVM ohne Probleme administrieren können.

- Hohe Geschwindigkeit: Die meisten Kommandos benötigen weniger als eine Sekunde für die Ausführung. Damit gibt es kein lästiges langwieriges Warten auf die Beendigung eines Kommandos.

- Online Dokumentation: Alle Funktionen des *LPAR-Tool*s sind online dokumentiert. Nichts ist bei der täglichen Arbeit lästiger als ständiges Nachschlagen von Informationen in externer Dokumentation. (Es gibt natürlich trotzdem eine ausführliche Dokumentation, falls die Online Dokumentation einmal nicht ausreichen sollte.)

- Feature für den PowerVM-Experten: Mit dem *LPAR-Tool* kann der Experte bis in die Tiefen von PowerVM vordringen.

- Wählbares Ausgabeformat: Bei der Ausgabe von Informationen kann neben der Standard-Ausgabe auch eine Ausgabe im JSON-, YAML- oder Stanza-Format gewählt werden.

- Verwendung in Skripten: Die Verwendung in Skripten für die Automation von Abläufen ist sehr einfach möglich.

Das *LPAR-Tool* kann auf einem oder mehrere Systemen installiert werden. Da das *LPAR-Tool* für die Kommunikation mit den *HMC*s *OpenSSH* verwendet, muß von diesen Systemen eine *SSH*-Verbindung zu den *HMC*s möglich sein. Das *LPAR-Tool* steht für die folgenden Betriebssysteme zur Verfügung:

- AIX
- Linux x86
- MacOS

Es stehen die folgenden Handbücher über den Download-Bereich https://powercampus.de/download zur Verfügung:

- LPAR-Tool 1.7.0.x Neuerungen (PDF)

- LPAR-Tool 1.7.0.x Erste Schritte (PDF)

- LPAR-Tool 1.7.0.x Benutzerhandbuch (PDF)

(Alle Handbücher sind im Download-Bereich auch in englischer Sprache verfügbar.)

2.1. Voraussetzungen

Das *LPAR-Tool* benutzt OpenSSH für die Verbindung zu den HMCs, d.h. eine funktionierende Version von OpenSSH muss auf dem Ziel-System installiert sein.

Damit mit dem *LPAR-Tool* PowerVM administriert werden kann, müssen die zu verwaltenden HMCs vom Ziel-System per OpenSSH erreichbar sein. D.h. falls eine Firewall zwischen Ziel-System und HMC verwendet wird, muss OpenSSH vom Ziel-System zur HMC frei geschaltet sein.

Für die Nutzung des *LPAR-Tool*s ist ein gültiger Account auf den HMCs notwendig. Wenn gewünscht, kann auch der *hscroot* Account genutzt werden. Die Empfehlung ist aber das jeder Nutzer einen eigenen HMC-Account verwendet.

Sollen DLPAR Operationen oder Live Partition Mobility verwendet werden, müssen die LPARs eine aktive RMC-Verbindung zu den HMCs besitzen. Ohne diese Voraussetzung kann PowerVM diese Operationen nicht durchführen (dies ist eine Beschränkung von PowerVM).

2.2. Download des *LPAR-Tool*s

Das LPAR-Tool kann über den Download-Bereich der Website https://powercampus.de bzw. https://powercampus01.com heruntergeladen werden. Je nach Betriebssystem gibt es eine oder zwei Versionen des LPAR-Tools.

AIX: Für das AIX Betriebssystem kann das *LPAR-Tool* entweder als *tar*-Datei oder als *BFF*-Fileset heruntergeladen werden. Möchte man das LPAR-Tool nicht dauerhaft installieren, sondern nur einmal ausprobieren, empfiehlt es sich die Version als *tar*-Datei herunterzuladen. Diese kann an beliebiger Stelle (z.B. im eigenen Home-Verzeichnis) ausgepackt werden.

Linux (x86): Für Linux gibt es ebenfalls eine Version als *tar*-Datei und ein *RPM*-Paket. Auch hier empfiehlt sich, für eine Test-Installation die *tar*-Datei zu verwenden.

MacOS: Für den Mac gibt es aktuell nur eine Version als *tar*-Datei, die an belieber Stelle entpackt werden kann.

Wird das *BFF*-Fileset oder das *RPM*-Paket installiert, dann findet man das *LPAR-Tool* im Verzeichnis */opt/pwrcmps/bin* auf dem Ziel-System.

2.3. Lizenz für das *LPAR-Tool*

Das *LPAR-Tool* benötigt als kommerzielles Produkt eine Lizenz, die auf dem Ziel-System in eine Datei eingetragen werden muß (*/opt/pwrcmps/etc/lpar.lic* oder *~/.lpar.lic*). Das *LPAR-Tool* darf produktiv nicht ohne eine gültige Subskription (enthält eine Lizenz) verwendet werden.

Für Test- und Evaluierung-Zwecke darf das LPAR-Tool auch ohne gültige Subskription verwendet werden. Damit dies möglichst einfach ist, enthält die jeweils aktuellste Version eine zeitlich beschränkte Test-Lizenz. Damit kann die Software unmittelbar nach dem Download und der Installation ausprobiert werden, ohne eine Lizenz anfordern zu müssen. Die integrierte Test-Lizenz hat eine beschränkte Gültigkeit von in der Regel 2 Monaten ab dem Build-Datum der Version. Auf der Download-Seite ist jeweils vermerkt bis wann die integrierte Test-Lizenz gültig ist.

Die integrierte Test-Lizenz unterstützt bis zu 10 HMCs, bis zu 100 Managed Systems und bis zu 1000 LPARs.

Für weitergehende Tests, ist es möglich eine Trial Lizenz anzufragen.

2.4. Die Kommandos des *LPAR-Tool*s

Das *LPAR-Tool* beinhaltet 4 Kommandos, welche nach den Komponenten einer PowerVM Umgebung benannt wurden: *hmc*, *ms*, *lpar* und *vios*. Jedes dieser 4 Kommandos unterstützt eine Vielzahl von Funktionen, insgesamt mehr als 200. Damit werden alle Bereiche der PowerVM Administration abgedeckt.

2.4.1. Das Kommando *hmc*

Mit dem Kommando hmc können die verfügbaren HMCs verwaltet werden. Es werden die Verwaltung von Benutzern auf den HMCs, sowie die Verwaltung von Rollen (Task- und Resource-Rollen) abgedeckt. Zusätzlich lassen sich eine Vielzahl von Informationen der HMCs anzeigen. Eine vollständige Liste der Funktionen erhält man durch Starten des Kommandos ohne Argumente:

```
$ hmc
USAGE:
   hmc [<option> ...] <keyword> [<option> ...] [<argument> ...]
   hmc -L|-V

Recognized keywords:
   add - Register HMC(s) with LPAR tool
   addauthkeys - Add an authorized SSH key
   chhmcfs - Free space in HMC file systems
   chhmcusr - Change attributes of HMC user account
   chresourcerole - Change managed resource role
   chtaskrole - Change managed task role
   chsvcevent - Update serviceable event on HMC
   cmd - Execute command on HMC
   connections - Display state of ssh master connections
   disconnect - Exit a ssh master connection
   flrt - Show FLRT report
   history - Show state or configuration changes
   list - List names of specified HMC(s)
   lsauthkeys - List authorized SSH keys
   lshmcfs - List HMC file system information
   lshmcusr - Show HMC user accounts
   lslic - List LIC levels available in hard disk repository
   lslogon - Show logon informations for HMC
   lslparmigr - List partiton mobility capabilities
   lsnet - Display HMC network configuration
   lsresource - Show all managed resource objects on the specified HMC
   lsresourcerole - Show managed resource roles on the specified HMC
   lssvcevents - Displays console or serviceable events
   lssysconn - List all systems connected to the specified HMC
   lstaskrole - Show managed task roles on the specified HMC
   mkhmcusr - Create HMC user account
   mkresourcerole - Create managed resource role
   mktaskrole - Create managed task role
   passwd - Change password
   remove - Unregister HMC(s) from LPAR tool
   rescan - Rereads cached informations from HMC(s)
   rmauthkeys - Remove an authorized SSH key
   rmhmcusr - Remove HMC user account
   rmresourcerole - Remove managed resource role
   rmtaskrole - Remove managed task role
```

```
   show - Show specified HMC(s)
   shutdown - Shut down or reboot
   termtask - Terminate a task
   version - Shows the HMC software version
   help [{<category>|<keyword>|usage}]
$
```

2.4.2. Das Kommando *ms*

Das Kommando *ms* (*managed system*) erlaubt die Administration von Managed Systems. Dabei werden unter anderem die folgenden Bereiche der Administration abgedeckt:

- Verwaltung von virtuellen Netzwerken und Switches
- Administration von SR-IOV
- Eintragen von Update Access Keys (UAK)
- Status von Managed Systems
- Power-On und Power-Off von Managed Systems
- Dynamic Plattform Optimisation
- Profile Backups
- Anzeigen von Prozessor, Speicher und I/O Informationen
- …

Eine vollständige Liste der unterstützten Funktionen wird angezeigt, wenn das Kommando *ms* ohne Argumente gestartet wird:

```
$ ms
USAGE:
   ms [<option> ...] <keyword> [<option> ...] [<argument> ...]
   ms -L|-V

Recognized keywords:
   addvnetwork - Add virtual network
   addvswitch - Add virtual switch
   bkprofdata - Back up profile data
   chattr - Change attributes
   cheth - Changes virtual ethernet attributes of a managed system
   chled - Change state of an LED
   chlparmigr - Changes partition migration attributes of managed system
   chlparutil - Change sample rate for utilization data collection
   chmem - Change memory attributes
   chprocpool - Change shared processor pool attributes
   chsriov - Switch SR-IOV adapter either to dedicated or shared mode
   flrt - List FLRT
   chvnetwork - Change attributes of a  virtual network
   chvswitch - Change attributes of a virtual switch
   enteruak - Enter new Update Access Key (UAC)
   history - Show state or configuration changes
   initprofdata - Initialize profile data
   list - Show names for specified managed systems
```

```
   lsattr - List attributes
   lseth - Display virtual ethernet attributes of managed system
   lsfc - List virtual FC informations
   lsled - Display state of system attention LEDs
   lslic - List LIC levels
   lslparmigr - Display partition migration attributes of managed system
   lslparutil - Show sample rate for utilization data collection
   lsmem - Display informations about memory
   lsmemopt - Show memory affinity scores
   lsproc - Show processing resources
   lsprocpool - Show shared processor pools
   lsprofdata - Lists profile data backups
   lspwrmgmt - Show power management settings
   lsrefcode - List reference codes for a service processor
   lsslot - List physical I/O slots
   lssriov - List informations for SR-IOV adapters, physical ports or logical ports
   lssysprof - List system profiles
   lsunit - List physical I/O units
   lsvnetwork - List virtual networks
   lsvswitch - List virtual switches
   mksysprof - Create system profile
   procstat - Show physical processor pool utilization data
   poweroff - Power off managed system
   poweron - Power on managed system
   rename - Rename managed system
   rmprofdata - Remove profile data backup
   rmsysprof - Remove system profile
   rmvnetwork - Remove virtual network
   rmvswitch - Remove virtual switch
   rstprofdata - Restore profile data
   show - Show basic informations
   startmemopt - Start DPO or mirrored memory defragmentation
   stopmemopt - Stop DPO or mirrored memory defragmentation
   status - Display status of managed system
   help [{<category>|<keyword>|usage}]
$
```

2.4.3. Das Kommando *lpar*

Alle PowerVM Funktionen die eine LPAR betreffen, können mit dem Kommando *lpar* durchgeführt werden. Dabei deckt das *LPAR-Tool* unter anderem die folgenden Bereiche der LPAR-Administration ab:

- Aktivieren und Deaktivieren von LPARs
- Zugriff auf die Konsole einer LPAR
- Administration von virtuellen Adaptern (seriell, Ethernet, FC, SCSI, SR-IOV, vNIC)
- Administration von Speicher und Prozessoren
- Anlegen und Löschen von LPARs
- Anzeigen von Performance Daten
- Live Partition Mobility (Validierung und Migration)
- Verwalten von Partitions Profilen

- ...

Eine Übersicht über die verfügbaren Funktionen wird beim Starten des Kommandos *lpar* ohne Argumente angezeigt:

```
$ lpar
USAGE:
   lpar [<option> ...] <keyword> [<option> ...] [<argument> ...]
   lpar -L|-V

Recognized keywords:
   activate - Activate AIX, Linux, IBM i or virtual I/O server partition
   actvnicbkdev - Make virtual NIC backing device active
   addeth - Add virtual ethernet adapter
   addfc - Add virtual FC client adapter
   addmem - Add memory
   addprocs - Add dedicated or shared processors
   addprocunits - Add processing units
   addscsi - Add virtual SCSI client adapter
   addserial - Add virtual serial adapter
   addslot - Add physical I/O slot
   addsriov - Add SR-IOV logical port
   addvlan - Add VLAN to virtual ethernet adapter
   addvnic - Add vNIC adapter
   addvnicbkdev - Add a backing device to a virtual NIC adapter of an LPAR
   applyprof - Apply partition profile
   backup - Create backup
   chattr - Change attributes
   cheth - Change attributes of virtual ethernet adapter
   chfc - Change virtual FC adapter
   chled - Change state of LED
   chmem - Change memory attributes
   chproc - Change processor attributes
   chprocpool - Change shared processor pool
   chscsi - Change virtual SCSI adapter
   chserial - Change virtual serial adapter
   chsriov - Change attributes of SR-IOV logical port
   chvnic - Change a vNIC adapter assigned to an LPAR.
   chvnicbkdev - Change attributes of a backing device for a virtual NIC
   clearvnicbkdev - Clear virtual NIC backing device error
   clone - Create clone
   console - Open virtual console session for AIX, Linux or virtual I/O server partition
   create - Create AIX, Linux, IBM i or virtual I/O server partition
   delete - Remove partition
   disableeth - Disable virtual ethernet adapter
   display - Display detailed informations
   dumprestart - Dump and restart a partition
   enableeth - Enable virtual ethernet adapter
   flrt - List FLRT
   history - Show state or configuration changes
   list - Show names of specific LPARs
   lsattr - List partition or profile attributes
   lseth - Show virtual ethernet adapters
   lsfc - Show virtual FC adapters
   lsled - Display state of system attention LED
   lslparmigr - List partition migration information
   lsmem - Show memory
   lsmemopt - Show DPO memory affinity scores
   lsproc - Show processing resources
   lsprof - List partition profiles
   lsrefcode - List reference codes for an LPAR
   lsscsi - Show virtual SCSI adapters
   lsserial - Show virtual serial adapters
   lsslot - Show physical I/O slots
   lssriov - Show SR-IOV logical ports
```

```
   lsvnic - Show vNIC adapters
   lsvslot - Show virtual slots
   migrate - Perform partition migration
   mkblueprint - Create blueprint
   osshutdown - Issue OS shutdown command to shut down a partition
   procstat - Show utilization data
   rename - Rename LPAR
   rmconsole - Close virtual console session for AIX, Linux or virtual I/O server partition
   rmeth - Remove virtual ethernet adapter
   rmfc - Remove virtual FC client adapter
   rmmem - Remove memory
   rmprocs - Remove dedicated or shared processors
   rmprocunits - Remove processing units
   rmprof - Remove partition profile
   rmscsi - Remove virtual SCSI client adapter
   rmserial - Remove virtual serial adapter
   rmslot - Remove physical I/O slot
   rmsriov - Remove SR-IOV logical port
   rmvlan - Remove VLAN from virtual ethernet adapter
   rmvnic - Remove a vNIC adapter from an LPAR.
   rmvnicbkdev - Removes a backing device from a virtual NIC adapter of an LPAR
   savecurrcfg - Save current configuration of an LPAR to profile
   show - Show basic informations
   shutdown - Shutdown AIX, Linux, IBM i or virtual I/O server partition
   status - Display current status
   stopmigr - Stop partition migration
   validate - Perform partition migration validation
   help [{<category>|<keyword>|usage}]
$
```

2.4.4. Das Kommando *vios*

Mit dem Kommando *vios* (*virtual I/O server*) können die Virtualisierungs-Funktionen der Virtual-I/O-Server administriert werden. Dazu wird kein direkter Login auf den Virtual-I/O-Servern benötigt. Eine aktive RMC-Verbindung der Virtual-I/O-Server zu den zugehörigen HMCs ist ausreichend.

Es können insbesondere die folgenden PowerVM Funktionalitäten eines Virtual-I/O-Servers administriert werden:

- Shared Ethernet Adapter (SEA)
- NPIV Mappings
- SCSI Mappings
- Virtual Optical Media Repository
- Storage Pools
- Link Aggregationen
- Geräte-Verwaltung (Anzeigen und Setzen von Attributen)

Auch hier erhält man eine vollständige Übersicht über die unterstützten Funktionen durch Starten des Kommandos *vios* ohne Argumente:

```
$ vios
```

```
USAGE:
   vios [<option> ...] <keyword> [<option> ...] [<argument> ...]
   vios -L|-V

Recognized keywords:
   addfc - Add virtual FC server adapter
   addlnaggadapter - Add adapter to Link Aggregation
   addscsi - Add virtual SCSI server adapter
   addsppv - Add physical volume to storage pool
   cfgdev - Configures devices in the Virtual I/O Server
   chdev - Changes the characteristics of a device
   chkdev - Check devices for virtual device provisioning capability
   chlnagg - Change Link Aggregation attributes
   chmedia - Change virtual optical media
   chrep - Change Virtual Media Repository
   chbdsp - Change attributes of backing device
   chsp - Change storage pool
   cmd - Execute ioscli command
   errlog - Display error log
   fcstat - Show FC statistics
   failoverlnagg - Failover Link Aggregation
   ioslevel - Display the Virtual I/O Server level
   list - Show names of specific virtual I/O servers
   loadopt - Load virtual optical media into virtual optical device
   lsattr - Show device attributes
   lsbdsp - Show backing devices
   lsdev - Show devices
   lslnagg - Show Link Aggregation adapters
   lsmedia - Show virtual optical media
   lsnpiv - Show NPIV mappings
   lsnports - List available NPIV capable ports
   lspv - Display physical volume(s)
   lsrep - Show Virtual Media Repository
   lssea - Show shared Ethernet adapters
   lssp - Show storage pool
   lstcpip - Display TCP/IP settings and parameters
   lsvopt - Show virtual optical devices
   lsvscsi - Show VSCSI mappings
   map - Map backing device to virtual SCSI server adapter
   mkbdsp - Create and/or map backing device from storage pool
   mklnagg - Create Link Aggregation adapter
   mkmedia - Create virtual optical media disk
   mkrep - Create Virtual Media Repository
   mksea - Create Shared Ethernet Adapter
   mksp - Create storage pool
   mkvopt - Create virtual optical device
   rmbdsp - Remove backing device
   rmdev - Remove device from the system
   rmfc - Remove virtual FC server adapter
   rmlnagg - Remove Link Aggregation adapter
   rmlnaggadapter - Remove adapter from Link Aggregation
   rmmedia - Remove virtual optical media
   rmrep - Remove Virtual Media Repository
   rmscsi - Remove virtual SCSI server adapter
   rmsea - Remove shared Ethernet adapter
   rmsp - Remove file storage pool
   rmsppv - Remove physical volume from storage pool
   rmvopt - Remove virtual optical device
   seastat - Show or clear SEA client statistics
   show - Show basic informations
   unloadopt - Remove virtual optical media from virtual optical device
   unmap - Unmap backing device from virtual SCSI server adapter
   vfcmap - Map virtual FC adapter to physical FC port
   vfcunmap - Unmap virtual FC adapter from physical FC port
   help [{<category>|<keyword>|usage}]
```

```
$
```

2.4.5. Die Online Dokumentation

Alle 4 Kommandos bieten eine ausführliche Online Hilfe. Alle Funktionen sind in Kategorien organisiert, wie z.B. SCSI, SEA, Memory usw. Welche Kategorien für eines der 4 Kommandos verfügbar sind, kann durch Verwenden des Arguments (Keyword) „*help*“ angezeigt werden, z.B.:

```
$ lpar help
Help is available for the following categories:

   lpar help blueprint eth fc io led lic lpm
   lpar help mem memory power proc processor prof profile
   lpar help scsi serial sriov vnic

Specific help is available for each of the supported keywords:

   lpar help <keyword>

For a complete list of all keywords try:

   lpar help usage
$
```

Alle Funktionen (Keywords) die zu einer Kategorie gehören, lassen sich anzeigen, indem man nach dem Keyword „*help*“ eine der verfügbaren Kategorien angibt, hier z.B. die Kategorie „*mem*“ beim Kommando *lpar*:

```
$ lpar help mem
USAGE: lpar [<option> ...] <keyword> [<option> ...] [<argument> ...]

Recognized keywords for topic 'mem' are:
   [-h <hmc>] [-m <ms>] [-p <profile>] addmem [-d] [-f] [-l <detail_level>] [-w
<wait_time>] [-v] <lpar> <mem>[k|K|m|M|g|G|t|T]
   [-h <hmc>] [-m <ms>] [-p <profile>] chmem [-d] [-f] [-l <detail_level>] [-w <wait_time>]
[-v] <lpar> <attributes> ...
   [-h <hmc>] [-m <ms>] [-p <profile>|-H] lsmem [{-o <format>|-f|-j|-y}] [-F <fields>] [-s
<selections>] [-v] [<lpar> ...]
   [-h <hmc>] [-m <ms>] [-p <profile>] lsmemopt [{-o <format>|-f|-j|-y}] [-F <fields>] [-s
<selections>] [-v] [<lpar> ...]
   [-h <hmc>] [-m <ms>] [-p <profile>] rmmem [-d] [-f] [-l <detail_level>] [-w <wait_time>]
[-v] <lpar> <mem>[k|K|m|M|g|G|t|T]
$
```

Detaillierte Informationen zu einer konkreten Funktion (Keyword) erhält man durch „*help*“ und die Angabe des Keywords, hier z.B. für das Keyword „*chmem*“:

```
$ lpar help chmem
USAGE:
   lpar [-h <hmc>] [-m <ms>] [-p <profile>] chmem [-d] [-f] [-l <detail_level>] [-w
<wait_time>] [-v] <lpar> <attributes> ...

DESCRIPTION

Change memory attributes for an LPAR. If no option '-d' or '-p' is
specified, the command tries to change the attributes by a DLPAR
operation and by changing the current profile.
Since most of the attributes are either valid only for DLPAR or
the profile, but not both, this makes changing memory related attributes
somewhat complicated. To ease usage, the following filtering is applied:
- If a specified attribute can not be changed by a DLPAR operation, the
```

```
  attribute is skipped from the DLPAR operation.
- If a specified attribute can not be changed in a profile, the attribute
  is skipped from the operation.
When skipping attibutes, a hint is shown to the user!

   -d : only DLPAR operation, don't update profile
   -f : force operation, even if there is no RMC connection
   -l : level of detail (1-4)
   -p : update the specified profile only
   -w : time (minutes) to wait for operation to finish

valid attributes for DLPAR and profile:
   mem_weight
   mem_expansion
      0 - disable memory expansion (profile only)
      1.00 to 10.00 - memory expansion factor

valid attributes for DLPAR only:
   hardware_mem_encryption
      0 - disable hardware-accelerated encryption
      1 - enable hardware-accelerated encryption
   hardware_mem_expansion
      0 - disable hardware-accelerated memory expansion
      1 - enable hardware-accelerated memory expansion

valid attributes for profile only:
   mem_mode : memory sharing mode
      ded - dedicated memory
      shared -shared memory
   min_mem : minimum number of megabytes
   desired_mem : desired number of megabytes
   max_mem : maximum number of megabytes
   min_num_huge_pages : minimum number of huge pages (only AIX and Linux)
   desired_num_huge_pages : desired number of huge pages (only AIX and Linux)
   max_num_huge_pages : maximum number of huge pages (only AIX and Linux)

NOTES

When the attribute 'mem_expansion' is set to '0', to disable active memory
expansion, the value of '0' is replaced by '1.0' in a DLPAR operation, since
active memory expansion can only be disabled in the profile.
Although the attribute 'desired_mem' is valid only in a profile, it is
interpreted as meaning the current amount of memory in a DLPAR operation.
The values for 'min_mem', 'desired_mem' and 'max_mem' can be specified with
a suffix indicating KB, MB, GB or TB. See 'lpar help addmem' for an overview
of the possibles suffixes.

EXAMPLES

Change memory expansion factor of LPAR lpar01 to 1.25:,
   lpar chmem lpar01 mem_expansion=1.25

Turn off memory expansion for LPAR lpar01:,
   lpar chmem lpar01 mem_expansion=0
(Memory expansion is turned off in the current profile, and is set to 1.0
by the DLPAR operation.)

Turn on hardware-accelerated memory expansion dynamically on for LPAR lpar03:
   lpar chmem -d lpar03 hardware_mem_expansion=1   # or
   lpar chmem lpar03 hardware_mem_expansion=1
(Since 'hardware_mem_expansion' is only valid for a DLPAR operation, the update
of the current profile is simply skipped in the second case!)

Change in profile 'standard' of lpar01 the memory sharing mode to 'shared',
and the maximum memory size to 16 gigabytes:
```

```
   lpar -p standard chmem lpar01 mem_mode=shared max_mem=16G

$
```

Neben einer Übersicht zur Syntax, gibt es für jedes Keyword eine kurze Beschreibung der Funktionalität. Eine Auflistung und Beschreibung der verfügbaren Optionen und Attribute des Kommandos, sowie eine Reihe von Beispielen der Verwendung des Keywords.

2.4.6. Der „*verbose*“ Mode

Das *LPAR-Tool* erlaubt über die Verwendung der Option „*-v*“ das für eine Funktion erst einmal angezeigt wird, was diese Funktion genau an Kommandos auf einer oder mehreren HMCs durchführen würde. Es werden die Kommandos angezeigt, aber es werden keine Kommandos ausgeführt, welche eine Änderung durchführen. Z.B.:

```
$ lpar -v status aixnim
hmc02: lssyscfg -r lpar -m s822
hmc02: lshwres -r mem -m s822 --level lpar
hmc02: lshwres -r proc -m s822 --level lpar
$
```

2.4.7. Logging der Kommando-Aufrufe

Das *LPAR-Tool* loggt standardmäßig alle Kommando-Aufrufe in der Datei *lpar.log* im Home-Verzeichnis des Benutzers mit. Dabei wird die Zeit festgehalten, die Version des *LPAR-Tool*s, das Kommando das gestartet wurde, sowie alle Kommandos auf den HMCs die durch den Kommando-Aufruf des *LPAR-Tool*s gestartet wurden:

```
$ cat ~/lpar.log
…
[12.09.20 11:15:56]
Version: 1.5.1.0 (20200801)
Command: lpar activate -p standard aix03
hmc02: chsysstate -m s822 -r lpar -o on -n aix03 -f standard

[12.09.20 11:16:05]
Version: 1.5.1.0 (20200801)
Command: lpar status
hmc02: lssyscfg -r lpar -m s822
hmc02: lshwres -r mem -m s822 --level lpar
hmc02: lshwres -r proc -m s822 --level lpar

$
```

Tritt ein Fehler bei der Ausführung eines Kommandos auf, wird die Fehlermeldung zusätzlich in der Log-Datei mitprotokolliert.

Sollte die Datei zu groß werden, kann sie jederzeit gelöscht werden. Die Log-Datei dient ausschließlich dem Zweck für den Benutzer selber festzuhalten welche Tätigkeiten er mit dem *LPAR-Tool* durchgeführt hat.

3. Komponenten einer PowerVM Umgebung

Eine PowerVM Umgebung besitzt typischerweise neben den Power Systemen auch sogenannte Hardware Management Consoles (HMC). Die Hardware Management Console (HMC) ist dabei eine dedizierte Hardware Appliance, kann aber neuerdings auch eine virtuelle Appliance sein. Jedes Power System kann an eine oder zwei HMCs angebunden werden. Über die HMC erfolgt dann die Administration der Power Systeme und LPARs. Mehrere Power Systeme können an eine HMC angebunden werden. Damit kann die Administration von vielen Power Systemen über einige wenige HMCs zentralisiert werden.

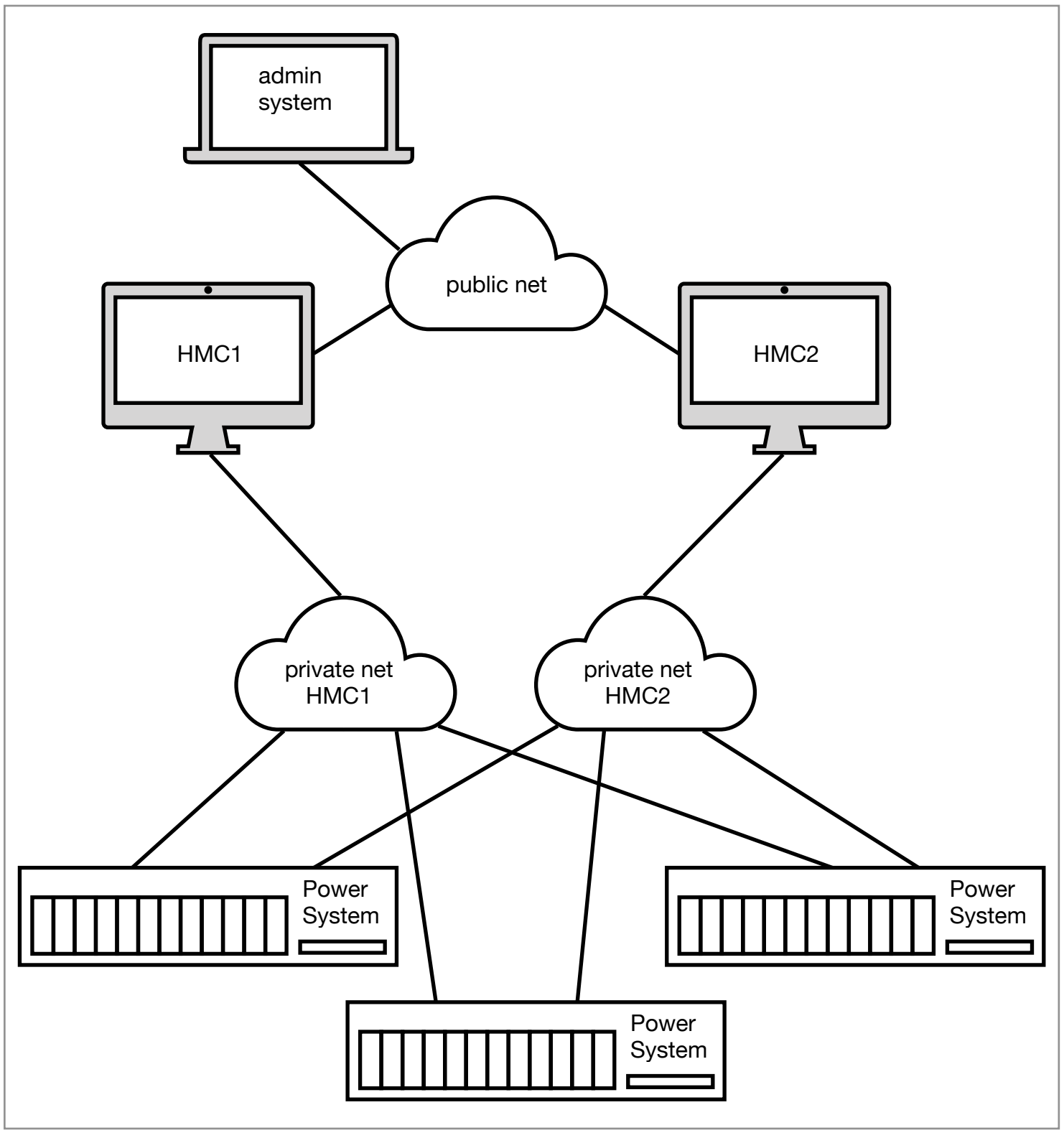


Bild 1.2: PowerVM Umgebung

Jedes Power System besitzt 2 spezielle HMC-Ports für die Anbindung an bis zu 2 HMCs. In Bild 1.2 sind 3 Power Systeme skizziert, welche jeweils an beide HMCs angebunden sind. HMC und Power System werden dabei über private Netzwerke miteinander verbunden. Der Zugriff von Außen auf die HMCs erfolgt über ein public Netzwerk.

3.1. Hardware Management Console (HMC)

Über die HMC können die angeschlossenen Power Systeme komplett verwaltet werden. Alle Möglichkeiten der PowerVM Virtualisierung können über die angeschlossenen HMCs konfiguriert und administriert werden. An eine HMC angebundene Power Systeme werden auch als sogenannte Managed Systems bezeichnet, da die HMC diese Power Systeme ‚*managed*'. Wir werden im Folgenden meistens den Begriff Managed System anstelle von Power System verwenden, in beiden Fällen ist letztlich der physikalische Server gemeint.

Die HMC bietet zwei verschiedene Möglichkeiten der Administration an:

- Ein Web-basiertes GUI: der Zugriff erfolgt über einen Browser.
- Und ein CLI auf der HMC: der Zugriff erfolgt typischerweise über einen Login mittels OpenSSH.

Typischerweise wird von den meisten Benutzern das Web-GUI verwendet. Es ist intuitiv in der Benutzung und es lassen sich alle Tätigkeiten mit Hilfe des GUIs ausführen.

Das CLI wird nur von wenigen Benutzern verwendet. Die Syntax der Kommandos ist nicht immer ganz einfach und selbst bei kleineren Tätigkeiten kommen schnell sehr lange Kommandos mit vielen Optionen zum Einsatz. Dies erfordert in der Regel häufiges Nachschauen in der Dokumentation, da man sich viele Optionen nur sehr schwer merken kann. Meist hat aber jeder POWER Administrator einen kleinen Satz von CLI Kommandos die er regelmäßig verwendet, da dies häufig schneller ist, als die GUI zu benutzen.

Das *LPAR-Tool* verwendet OpenSSH und die CLI der HMC. In der Regel muß man keine Optionen angeben, da das *LPAR-Tool* die notwendigen Optionen automatisch angibt.

Auf einer HMC gibt es per Default den folgenden Account:

- *hscroot* mit dem Passwort ‚*abc123*'.

Dieser Account besitzt die Rolle *hmcsuperadmin* und hat damit das Maximum (fast) an Rechten auf einer HMC. Für Benutzer können (und sollten auch) zusätzliche Accounts angelegt werden. Wir gehen später auf das Thema HMC Accounts und Rollen genauer ein, für den Moment gehen wir davon aus, das entweder schon ein Account für den Benutzer existiert, oder der Account *hscroot* verwendet wird.

Die dem *LPAR-Tool* bekannten HMCs kann man mit dem Kommando „*hmc show*" auflisten:

```
$ hmc show
NAME   SERIAL_NUM  TYPE_MODEL
$
```

Wurde noch keine HMC bekannt gemacht (registriert), dann ist die Ausgabe leer, wie in der Beispiel-Ausgabe.

Jede HMC muß für die Verwendung mit dem *LPAR-Tool* einmalig registriert werden. Dies erfolgt mit dem Kommando „*hmc add*" und der Angabe einer oder mehrerer HMCs. In unserem Beispiel heisst die HMC *hmc01*:

```
$ hmc add hmc01
hmc01:
   ms01
    > lpar01
```

```
    > ms01-vio1
    > ms01-vio2
   ms02
    > aix01
    > aix03
    > aixnim
    > ms02-vio1
    > ms02-vio2
   ms03
    > aix02
    > aix04
    > ms03-vio1
    > ms03-vio2
$
```

Das Registrieren einer HMC dauert in der Regel nur 1 oder 2 Sekunden. Wie man der Beispiel-Ausgabe entnehmen kann, werden dabei die angebundenen Managed Systems, sowie die LPARs auf den Managed Systems erkannt und ebenfalls registriert. An unsere HMC *hmc01* sind die 3 Managed Systems *ms01*, *ms02* und *ms03* angebunden.

Wird bei „*hmc add*“ nur der Name der HMC angegeben, geht das *LPAR-Tool* davon aus, das der Account auf der HMC dem lokalen Benutzernamen entspricht. Sollte dies nicht der Fall sein, kann der gewünschte Account aber auch explizit angegeben werden, so wie im folgenden Beispiel:

```
$ hmc add hscroot@hmc01
hmc01:
   ms01
    > lpar01
    > ms01-vio1
    > ms01-vio2
   ms02
    > aix01
    > aix03
    > aixnim
    > ms02-vio1
    > ms02-vio2
   ms03
    > aix02
    > aix04
    > ms03-vio1
    > ms03-vio2
$
```

Damit wird für die angegebene HMC immer der Account *hscroot* verwendet werden.

Auf die gleiche Weise sollten dann alle weiteren HMCs einmalig registriert werden.

Nachdem nun mindestens eine HMC registriert wurde, kann man diese mittels „*hmc show*“ anzeigen:

```
$ hmc show
HMC   SERIAL MODEL
hmc01 12BEEF 7042-CR9
$
```

Unsere HMC hat die Seriennummer *12BEEF* und ist vom Typ *7042-CR9* (das ist das letzte auf x86 basierende Modell).

Soll eine HMC, samt angebundener Managed Systems und deren LPARs, nicht mehr durch das *LPAR-Tool* verwaltet werden, dann kann die HMC ganz einfach deregistriert werden. Dazu gibt es das Kommando „*hmc remove*“:

```
$ hmc remove hmc01
$
```

Dies ändert natürlich nichts an der HMC, einem der angebundenen Managed Systems oder einer der LPARs! Es bedeutet lediglich das die HMC dem *LPAR-Tool* nicht mehr bekannt ist. Der Versuch die HMC trotzdem mit dem *LPAR-Tool* anzusprechen resultiert dann einfach in einer entsprechenden Fehlermeldung:

```
$ hmc lssysconn hmc01
ERROR: hmc01 is not a known HMC
USAGE:
   hmc lssysconn [-o <format>] [{-f|-j|-y}] [-F <fields>] [-s <selections>] [-v] <hmc>
$
```

Natürlich kann die HMC jederzeit wieder mittels „*hmc add*“ registriert werden und kann dann auch wieder über das *LPAR-Tool* administriert werden.

3.2. Managed Systems

Welche Managed Systems an die HMC angebunden sind, kann man mit dem Kommando „*hmc lssysconn*“ abfragen, hier eine Beispiel-Ausgabe von *hmc01*:

```
$ hmc lssysconn hmc01
TYPE_MODEL_SERIAL_NUM  STATE          RESOURCE_TYPE  IPADDR         ALT_IPADDR
SP           SP_PHYS_LOC
8286-41A*32C4D4W       Connected      sys            172.16.100.13  unavailable
primary      U78C9.001.VYR0UCG-P1
8284-22A*326E12W       Connected      sys            172.16.100.12  unavailable
primary      U78CB.001.VYR0AI4-P1
8284-22A*326E13W       Connected      sys            172.16.100.15  unavailable
primary      U78CB.001.VYR0DM6-P1
$
```

Es handelt sich hier um 2 Managed Systems vom Typ 8284-22A (S822) und einmal 8286-41A (S814). Da die Managed Systems vom *LPAR-Tool* registriert wurden und damit bekannt sind, kann man sich diese aber auch einfacher mittels „*ms show*“ anzeigen lassen:

```
$ ms show
NAME  SERIAL_NUM  TYPE_MODEL  HMCS   MODEL_NAME
ms01  326E12W     8284-22A    hmc01  IBM Power System S822
ms02  326E13W     8284-22A    hmc01  IBM Power System S822
ms03  32C4D4W     8286-41A    hmc01  IBM Power System S814
$
```

In dieser Ausgabe wird dann auch der Modell-Name mit angezeigt.

3.3. PowerVC

PowerVC ist eine erweitert Virtualisierungs- und Cloud-Management-Lösung. PowerVC basiert auf OpenStack und soll das Management und die Bereitstellung von AIX, IBM i und Linux auf Power Systemen vereinfachen.

Das LPAR-Tool ist nicht vollständig kompatibel zu PowerVC (zumindest in der aktuellen Version). Das LPAR-Tool kann zwar auch auf Systemen eingesetzt werden, die von PowerVC verwaltet werden, es gibt aber einige Einschränkungen:

- Einige Kommandos können nicht aufgerufen werden bzw. liefern einen Fehler beim Aufruf.
- Einige Kommandos können nicht mit allen Optionen und Argumenten aufgerufen werden, oder müssen mit bestimmten Optionen und Argumenten aufgerufen werden.

Dies liegt darin das auf HMCs die von PowerVC verwaltet werden, ein sogenannter Co-Management Modus verwendet wird, der eine Reihe von Einschränkungen bei den Kommandos auf der HMC mit sich bringt.

Unabhängig davon kann aber der Einsatz des LPAR-Tools auch in PowerVC Umgebungen sinnvoll sein, da das LPAR-Tool viele Informationen schneller und leichter liefern kann.

4. Überblick über eine PowerVM Umgebung

In diesem Kapitel soll gezeigt werden, wie man sich einen Überblick über eine PowerVM Umgebung verschaffen kann. Es wird gezeigt wie sich einige Fragen beantworten lassen, wie z.B.:

- Wieviele und welche LPARs gibt es in der PowerVM Umgebung.
- Wie ist der Status der LPARs und der zugehörigen Managed Systems.
- Welche Firmware Versionen sind installiert.
- Wie ist die Hardware Ausstattung eines Managed Systems.

Außerdem werden einige administrative Tätigkeiten exemplarisch vorgestellt:

- Aktivieren einer LPAR.
- Herunterfahren einer LPAR.
- Konsole für eine LPAR.

4.1. Welche LPARs gibt es?

Eine Liste aller LPARs einer PowerVM Umgebung lässt sich mit dem Kommando „*lpar show*“ ausgeben:

```
$ lpar show
NAME   ID  SERIAL   LPAR_ENV  MS    HMCS
aix01  3   12AF033  aixlinux  ms03  hmc01,hmc02
aix02  3   12BF033  aixlinux  ms04  hmc01,hmc02
aix03  11  12AF03B  aixlinux  ms03  hmc01,hmc02
aix04  19  12BF03J  aixlinux  ms04  hmc01,hmc02
aix05  4   7830W44  aixlinux  ms07  hmc03
…
$
```

Für jede LPAR werden dabei allgemeine Informationen, wie Name, LPAR-ID, Seriennummer, Typ der LPAR, zugehörige Managed System und die zugehörige(n) HMC(s) angezeigt. Diese Informationen werden vom *LPAR-Tool* lokal in einem Cache abgespeichert (Default: *~/lpar.list*). Der Cache kann mit Hilfe des Kommandos „*hmc rescan*“ aktualisiert werden, dabei werden alle bekannten HMCs abgefragt und eine Liste aller Managed Systems und LPARs erstellt und lokal abgespeichert.

```
$ hmc rescan
hmc01:
   ms03
    > aix01
    > aix03
    > lpar17
    > ms03-vio2
```

```
    > ms03-vio1
   ms04
    > aix02
    > aix04
    > lpar18
...
$
```

Je nach Größe der Umgebung kann dies einige Sekunden dauern, da eine Vielzahl von Kommandos auf den HMCs abgesetzt werden muß.

Benötigt man Informationen nur über bestimmte LPARs, können die betreffenden LPARs als Argumente angegeben werden:

```
$ lpar show aix01 aix05
NAME   ID  SERIAL   LPAR_ENV  MS    HMCS
aix01  3   12AF033  aixlinux  ms03  hmc01,hmc02
aix05  4   7830W44  aixlinux  ms07  hmc03
$
```

Sehr nützlich ist die Möglichkeit Wildcards zu verwenden, damit lassen sich z.B. leicht alle LPARs auflisten, deren Name mit „*aix*“ beginnt:

```
$ lpar show „aix*“
NAME   ID  SERIAL   LPAR_ENV  MS    HMCS
aix01  3   12AF033  aixlinux  ms03  hmc01,hmc02
aix02  3   12BF033  aixlinux  ms04  hmc01,hmc02
aix03  11  12AF03B  aixlinux  ms03  hmc01,hmc02
aix04  19  12BF03J  aixlinux  ms04  hmc01,hmc02
aix05  4   7830W44  aixlinux  ms07  hmc03
…
$
```

(Die Anführungszeichen können auch weggelassen werden, wenn es im aktuellen Verzeichnis keine Dateien gibt, die auf den Wildcard matchen. Ohne Anführungszeichen wird ein Wildcard von der Shell interpretiert und durch das Resultat des Matchings ersetzt.)

Häufig interessieren nur die LPARs auf einem bestimmten Managed System, hierzu bietet das Kommando „*lpar show*“ die Möglichkeit an, mit der Option „*-m*“ das interessierende Managed System anzugeben:

```
$ lpar show -m ms04
NAME       ID  SERIAL   LPAR_ENV   MS    HMCS
aix02      3   12BF033  aixlinux   ms04  hmc01,hmc02
aix04      19  12BF03J  aixlinux   ms04  hmc01,hmc02
ms04-vio1  1   12BF031  vioserver  ms04  hmc01,hmc02
ms04-vio2  2   12BF032  vioserver  ms04  hmc01,hmc02
$
```

4.2. Status von LPARs

Ob eine LPAR gerade aktiv ist, oder ausgeschaltet ist, kann durch Ausgeben des Status einer LPAR sehr leicht festgestellt werden. Mit dem Kommando „*lpar status*“ kann dabei der Status einer einzelnen LPAR oder der Status mehrerer LPARs (oder auch aller LPARs) ausgegeben werden:

```
$ lpar status aix\*
NAME   LPAR_ID  LPAR_ENV  STATE          PROFILE   SYNC  RMC       PROCS  PROC_UNITS
MEM    OS_VERSION
aix01  3        aixlinux  Running        standard  0     active    4      0.4
33792  AIX 7.1 7100-04-05-1720
aix02  3        aixlinux  Running        standard  0     inactive  1      0.2
16384  AIX 7.1 7100-04-05-1720
aix03  11       aixlinux  Running        standard  0     active    2      0.3
32768  AIX 7.1 7100-04-05-1720
aix04  19       aixlinux  Not Activated  standard  0     inactive  4      0.4
33792  AIX 7.1 7100-04-05-1720
…
$
```

Der Spalte *STATE* kann man entnehmen das die LPARs *aix01*, *aix02* und *aix03* aktiv sind, die LPAR *aix04* hingegen ausgeschaltet ist (*Not Activated*). Die LPARs *aix01* und *aix03* haben eine aktive RMC-Verbindung zu mindestens einer der HMCs, die beiden LPARs *aix02* und *aix04* haben keine aktive RMC-Verbindung. Bei der LPAR *aix04* ist der Grund klar - sie ist ausgeschaltet und kann daher keine aktive RMC-Verbindung haben. Bei LPAR *aix02* scheint es hingegen ein Problem zu geben, da RMC nicht aktiv ist, obwohl die LPAR aktiv ist.

Neben dem Status werden auch noch Informationen zur Anzahl Prozessoren (sowie Entitlement bei Shared-Prozessor LPARs) und der Hauptspeicher-Größe angezeigt. Als letztes wird das verwendete Betriebssystem, sowie die Version des Betriebssystems angezeigt. Diese Information ist aber nur verfügbar, wenn es eine aktiv RMC-Verbindung gibt oder irgendwann einmal gegeben hat.

4.3. Aktivieren einer LPAR

Ist eine LPAR ausgeschaltet (Status *Not Activated*), kann die LPAR mit dem Kommando „*lpar activate*“ aktiviert werden. Wird eine LPAR nach dem Anlegen das erste Mal aktiviert, so muß zwingend die Option „*-p*“ und ein Profilname angegeben werden:

```
$ lpar -p standard activate aix04
$
```

Die LPAR wird dann aktiviert und es wird versucht das Betriebssystem zu booten.

Der größte Teil der Konfiguration einer LPAR ist in Profilen hinterlegt. Eine LPAR kann bei Bedarf mehrere Profile haben, sie kann aber zu einem Zeitpunkt immer nur eines ihrer Profile verwenden. Ein Profil einer LPAR ist immer fest der LPAR zugeordnet und kann nicht für eine andere LPAR verwendet werden. Haben zwei LPARs Profile mit dem Namen „*myprofile*“, dann handelt es sich hier um 2 verschiedene Profile, auch wenn der Name der gleiche ist. Jedes der beiden Profile ist eindeutig genau einer der LPARs zugeordnet und kann auch nur von dieser LPAR genutzt

werden. Welche Profile für eine LPAR angelegt wurden, lässt sich mit dem Kommando „*lpar lsprof*“ (*list profiles*) anzeigen:

```
$ lpar lsprof aix04
NAME                      MEM_MODE  MEM    PROC_MODE  PROCS  PROC_COMPAT
last*valid*configuration  ded       33792  shared     4      default
ostandard                 ded       16384  shared     2      default
standard                  ded       33792  shared     4      default
$
```

Das Profil *last*valid*configuration* wird von PowerVM verwaltet. In diesem Profil steht immer die aktuelle Konfiguration (falls die LPAR gerade aktiv ist), oder die letzte aktive Konfiguration (falls die LPAR nicht aktiviert ist). Die Existenz dieses besonderen Profils ist der Hinweis darauf das die LPAR entweder aktiv ist, oder schon einmal aktiv war. Die LPAR kann dann ohne Angabe eines Profils aktiviert werden, es wird dann die Konfiguration des Profils *last*valid*configuration* verwendet. Für eine neu angelegte LPAR gibt es vor der ersten Aktivierung das Profil *last*valid*configuration* nicht.

4.4. Herunterfahren einer LPAR

Möchte man eine laufende LPAR herunterfahren, sollte man die LPAR über das verwendete Betriebssystem herunterfahren, indem man sich einloggt und als Administrator das Betriebssystem herunterfährt. Bei AIX kann dies z.B. mit dem Kommando *shutdown* erfolgen:

```
aix02 # shutdown

SHUTDOWN PROGRAM
Mon May  3 11:57:07 CEST 2021


Broadcast message from root@aix02 (tty) at 11:57:07 ...

PLEASE LOG OFF NOW ! ! !
All processes will be killed in 1 minute.



Broadcast message from root@aix02 (vty0) at 11:58:07 ...

! ! ! SYSTEM BEING BROUGHT DOWN NOW ! ! !

Stopping IBM.ConfigRM...
0513-044 The IBM.ConfigRM Subsystem was requested to stop.
…
Unmounting the file systems...
Unmounting the file systems...

Bringing down network interfaces: en0 lo0



open: No such file or directory

May  3 11:58:35 portmap: terminating on signal.
....Halt completed….
```

Eventuell auftretende Fehler beim Herunterfahren der LPAR werden dabei angezeigt und bleiben nicht unbeobachtet.

Es ist aber natürlich trotzdem möglich das Betriebssystem auch mit PowerVM herunterzufahren. Hierzu gibt es das Kommando „*lpar osshutdown*“ (*shutdown operating system*):

```
$ lpar osshutdown aix02
$
```

Dabei führt PowerVM auf der LPAR das folgende Kommando aus:

- AIX: es wird das Kommando *shutdown* ausgeführt.
- Linux: es wird das Kommando „*shutdown -h +1*“ ausgeführt.
- VIOS: es wird das Kommando *shutdown* (als Benutzer *padmin*) ausgeführt.

Die LPAR ist anschließend im Zustand *Not Activated*:

```
$ lpar status aix04
NAME   LPAR_ID  LPAR_ENV  STATE          PROFILE   SYNC  RMC       PROCS  PROC_UNITS
MEM    OS_VERSION
aix02  3        aixlinux  Not Activated  standard  0     inactive  1      0.2
16384  AIX 7.1 7100-04-05-1720

$
```

Für einen Reboot der LPAR kann die Option „*-r*“ bei „*lpar osshutdown*“ verwendet werden:

```
$ lpar osshutdown -r aix02
$
```

Hinweis: Bei einem Reboot bleibt die LPAR aktiviert, sie wird nicht deaktiviert und damit auch nicht mit einem ihrer Profile neu aktiviert. D.h. insbesondere das eine Änderung in einem der Profile keine Änderung in der Konfiguration der LPAR bewirkt. Dazu muß die LPAR heruntergefahren (deaktiviert) werden und mit Angabe des geänderten Profils neu aktiviert werden!

4.5. Konsole für eine LPAR

Gelegentlich benötigt man als Administrator eine Konsole für eine LPAR, da z.B. ein Netzwerk-Problem existiert, man Meldungen beim Booten der LPAR verfolgen möchte oder man die Meldungen beim Herunterfahren einer LPAR anschauen möchte. In all diesen Fällen kann mit dem Kommando „*lpar console*“ sofort eine Konsole für eine beliebige LPAR gestartet werden:

```
$ lpar console aix04

 Open in progress
```

```
  Open Completed.





AIX Version 7
Copyright IBM Corporation, 1982, 2020.
Console login: root
root's Password:
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 7.1!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************

aix04:/root>
aix04:/root>
```

Wird die Konsole nicht mehr benötigt, sollte man sich zunächst beim Betriebssystem abmelden (z.B. mittels Kommando *exit*). Anschließend kann die Konsolen-Sitzung mit der Tasten-Kombination „~.“ beendet werden.

4.6. Status der Managed Systems

Der Status eines oder mehrerer Managed Systems kann mit dem Kommando „*ms status*“ angezeigt werden. Wird kein Managed System als Argument angegeben, dann wird der Status von allen Managed Systems gezeigt:

```
$ ms status
NAME  STATE      DETAILS  IPADDR          SECONDARY       KEYLOCK      SERVICE_LPAR_ID
ms01  Operating  None     10.0.0.245      -               norm         none
ms03  Operating  None     10.0.0.244      -               norm         none
ms04  Operating  None     10.0.0.241      -               norm         none
ms05  Operating  None     10.0.0.242      -               norm         none
ms06  Operating  None     10.0.0.240      10.0.0.239      norm         none
ms07  Operating  None     10.0.0.234      -               norm         none
ms08  Operating  None     10.0.0.252      -               norm         none
$
```

In der Spalte *STATE* wird der aktuelle Zustand des Managed Systems angezeigt, hier bei allen Managed Systems *Operating*. Außerdem werden die IP-Adressen der Service-Prozessoren, sowie weitere Informationen angezeigt.

4.7. System Firmware Version der Managed Systems

Der POWER Hypervisor ist zum Teil in Form von System Firmware auf den Managed Systems implementiert. Von Zeit zu Zeit veröffentlicht IBM neue Versionen der System Firmware mit Fehlerbehebungen und neuen PowerVM-Merkmalen. Welche System Firmware auf einem Managed System installiert ist, IBM spricht hier vom sogenannten Licensed Internal Code (*LIC*), lässt sich mit dem Kommando „*ms lslic*“ anzeigen.

```
$ ms lslic
NAME  LIC_TYPE        ECNUMBER  INSTALLED  ACTIVATED  ACCEPTED  DEFERRED  IPL  UAK
ms01  Managed System  01AL770   126        126        116       None      126  -
ms03  Managed System  01SV860   215        215        138       None      215  08/07/2021
ms04  Managed System  01SV860   215        215        138       None      215  07/24/2021
ms05  Managed System  01SV860   215        215        138       None      215  08/14/2021
ms06  Managed System  01SV860   215        215        138       None      215  10/03/2021
ms07  Managed System  01VL940   50         50         50        None      50   11/05/2021
ms08  Managed System  01VL940   50         50         50        None      50   01/27/2022
$
```

Auch hier können Managed Systems explizit angegeben werden, ansonsten werden alle Managed Systems aufgelistet.

Die Ausgabe zeigt neben der installierten System Firmware, z.B. *SV860_215* bei dem Managed System *ms04*, auch an, welche Version aktiviert ist (*ACCEPTED*). Für POWER8 oder höher wird zusätzlich das Ablauf-Datum des Update Access Keys (*UAK*) in der letzten Spalte angezeigt.

4.8. Hardware Ausstattung von Managed Systems

Jedes Power System besitzt neben Prozessoren und Speicher eine Reihe von I/O Slots für PCI-Adapter. Wieviele I/O Slots verfügbar sind, hängt dabei von dem konkreten Modell des Power Systems ab. In der Regel werden Power Systeme schon beim Kauf mit einer Reihe von FC- und Ethernet-Adaptern gekauft. Welche PCI-Adapter ein konkretes Managed System besitzt, lässt sich mit dem Kommando „*ms lsslot*“ (*list slots*) auflisten:

```
$ ms lsslot ms07
DRC_NAME                  DRC_INDEX  IOPOOL  LPAR_NAME  DESCRIPTION
U78C7.001.LJD4A99-P1-C10  21010010   none    -          PCIe2 16Gb 2-Port Fibre Channel
Adapter
U78C7.001.LJD4A99-P3-R2   21010011   none    ms07-vio2  RAID Controller
U78C7.001.LJD4A99-P1-C11  21010018   none    -          PCIe2 4-port (10Gb FCoE & 1GbE)
SR&RJ45 Adapter
U78C7.001.LJD4A99-P1-C12  21010019   none    ms07-vio1  PCIe2 16Gb 2-Port Fibre Channel
Adapter
U78C7.001.LJD4A99-P1-C8   21010020   none    ms07-vio2  PCIe2 16Gb 2-Port Fibre Channel
Adapter
U78C7.001.LJD4A99-P3-R1   21010021   none    ms07-vio1  RAID Controller
U78C7.001.LJD4A99-P1-C9   21010028   none    -          PCIe2 4-port (10Gb FCoE & 1GbE)
SR&RJ45 Adapter
U78C7.001.LJD4A99-P1-T1   2101002B   none    -          Universal Serial Bus UHC Spec
U78C7.001.LJD4A99-P1-C6   2102002C   none    -          1 Gigabit Ethernet (UTP) 4 Port
Adapter PCIE-4x/Short
U78C7.001.LJD4A99-P1-C7   2103002D   none    -          PCIe2 16Gb 2-Port Fibre Channel
Adapter
U78C7.001.LJD4A99-P1-C3   21010030   none    -          PCIe3 4-port 10GbE SR Adapter
```

```
U78C7.001.KIC3988-P1-C4   21010038   none    ms07-vio1  Quad 8 Gigabit Fibre Channel LP
Adapter
U78C7.001.LJD4A99-P1-C1   21010040   none    -          PCIe3 4-port 10GbE SR Adapter
U78C7.001.LJD4A99-P1-C2   21010048   none    ms07-vio2  Quad 8 Gigabit Fibre Channel LP
Adapter
$
```

Neben dem Physical Location Code (Spalte *DRC_NAME*), wird der sogenannte *DRC_INDEX* angezeigt. Letzterer wird verwendet um einen I/O Adapter einer LPAR zuzuweisen. In der Regel sind physikalische I/O Adapter einem Virtual-I/O-Server zugewiesen, in der Beispiel-Ausgabe ist ein Teil der I/O Adapter auf die beiden Virtual-I/O-Server *ms07-vio1* und *ms07-vio2* aufgeteilt. In der letzten Spalte findet sich eine Beschreibung des I/O Adapters.

Prozessor- bzw. Speicher-Ausstattung eines Managed Systems kann mit „*ms lsproc*“ (*list processors*) bzw. „*ms lsmem*“ (*list memory*) angezeigt werden:

```
$ ms lsproc ms07
NAME  INSTALLED  CONFIGURABLE  AVAIL  MAX_SHARED_PROC_POOLS
ms07  48.0       48.0          1.15   64
$
$
$ ms lsmem ms07
NAME  INSTALLED  FIRMWARE  CONFIGURABLE  AVAIL   MEM_REGION_SIZE
ms07  2097152    43776     2097152       859392  256
$
```

Bei beiden Ausgaben wird jeweils auch angezeigt wieviel von der Resource noch (z.B. für weitere LPARs) verfügbar ist.

5. Prozessor Virtualisierung

PowerVM bietet eine Reihe von verschiedenen Möglichkeiten Prozessor-Ressourcen in LPARs zu verwenden. Im einfachsten Fall werden einer LPAR dedizierte Prozessoren zugeordnet. Die LPAR kann die zugeordneten dedizierten Prozessoren dann zu 100% nutzen und muß nicht mit anderen LPARs um Prozessor-Ressourcen konkurrieren. In den meisten Fällen werden Prozessoren (bzw. Prozessor-Cores) zwischen LPARs geteilt, um eine bessere Auslastung der Prozessoren zu erzielen. Man spricht dann von sogenannten Shared Prozessoren, im Unterschied zu den dedizierten Prozessoren. Die Aufteilung eines Shared Prozessors in Anteile, welche dann den LPARs zugeordneten werden, wird als Micro-Partitioning bezeichnet. Dabei teilt der POWER Hypervisor die Shared Prozessoren über ein Zeitscheiben-Verfahren auf die zugehörigen LPARs auf.

Bei beiden Varianten gibt es weitere Möglichkeiten der Konfiguration. Insbesondere können Pools von Shared Prozessoren konfiguriert werden, und LPARs können dann einem der Pools zugewiesen werden.

5.1. Dedizierte Prozessoren

Eine LPAR kann jeweils entweder ausschließlich dedizierte Prozessoren oder ausschließlich geteilte Prozessoren verwenden, aber nicht beides gleichzeitig. Werden dedizierte Prozessoren verwendet, dann stehen diese ausschließlich einer LPAR zur Verfügung (Ausnahmen später) und können nicht von anderen LPARs verwendet werden.

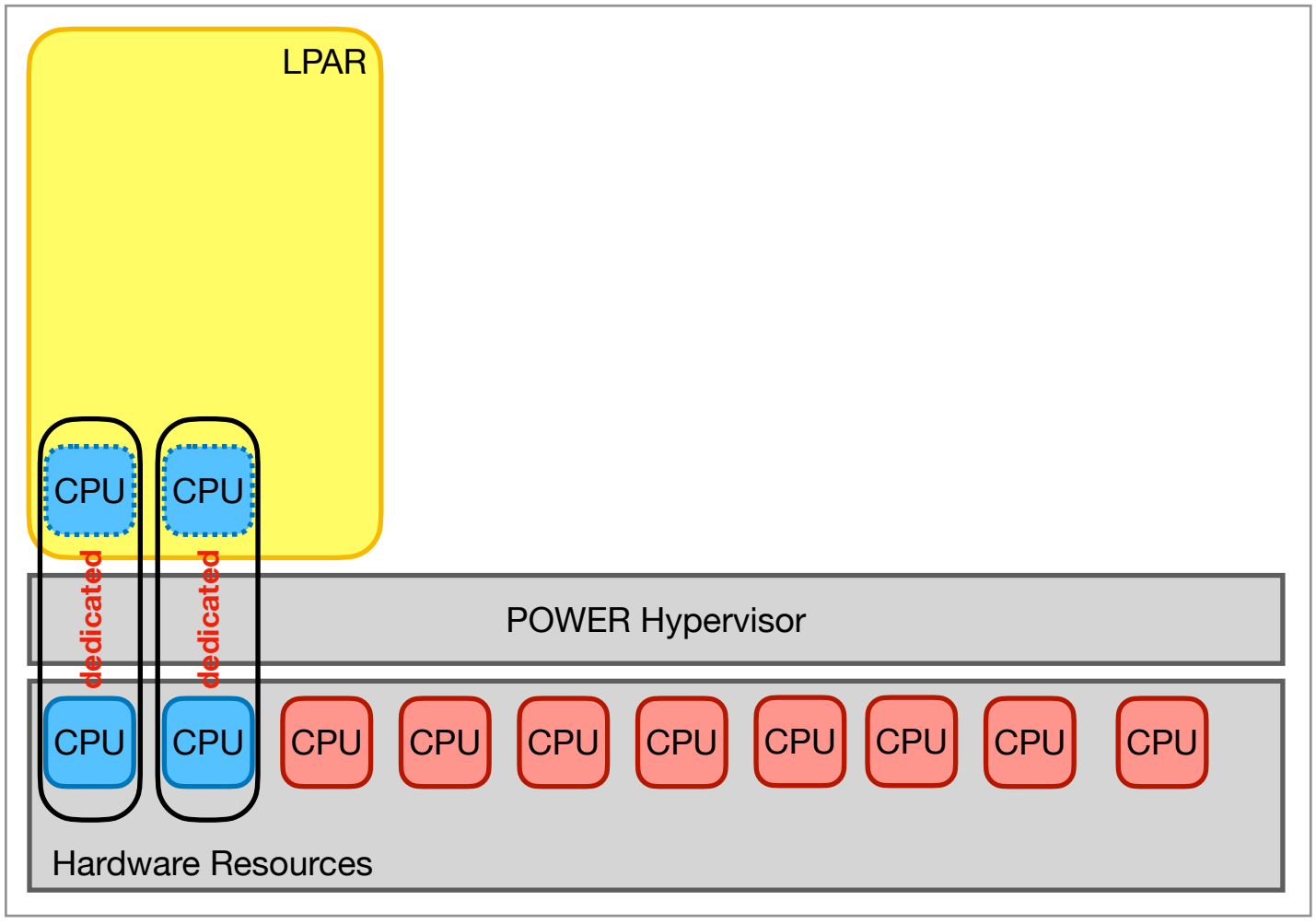


Bild 5.1: Partitionierung mit dedizierten Prozessoren

In Bild 5.1 ist eine LPAR mit 2 dedizierten Prozessoren (blau) gezeigt. Beim Erzeugen einer neuen LPAR kann durch Angabe des Attributs *proc_mode* angegeben werden, ob dedizierte Prozessoren verwendet werden sollen, oder Shared Prozessoren:

```
   proc_mode : processor mode
      ded - dedicated processors
      shared - shared processors
```

Eine neue LPAR kann mit dem Kommando „*lpar create*“ erzeugt werden. Um dedizierte Prozessoren zu verwenden, wird das Attribute *proc_mode* mit dem Wert *ded* auf der Kommandozeile angegeben:

```
$ lpar -m ms02 create lpar1 proc_mode=ded
.
    > lpar1
$
```

Die Option „*-m*“ mit dem Ziel Managed System, muss zwingend angegeben werden, sie legt fest auf welchem der Managed Systems die LPAR angelegt werden soll. Der LPAR-Name, hier *lpar1*, ist optional. Wird kein Name angegeben, generiert das *LPAR-Tool* einen eindeutigen Namen.

Eine neu erzeugte LPAR ist zunächst nicht aktiviert. Der größere Teil der LPAR-Konfiguration ist in einem Profil abgelegt, das beim Erzeugen der LPAR angelegt wird. Per Default wird der Name *standard* für das Profil verwendet, es lässt sich aber auch ein anderer Default hinterlegen. Ein Blick in das Profil *standard* zeigt das die LPAR mit einem dedizierten Prozessor konfiguriert wurde:

```
$ lpar -p standard lsproc lpar1
            PROC        PROCS            PROC_UNITS                        UNCAP   PROC
LPAR_NAME   MODE  MIN  DESIRED  MAX  MIN  DESIRED  MAX  SHARING_MODE      WEIGHT  POOL
lpar1       ded   1    1        1    -    -        -    keep_idle_procs   -       -
$
```

Soll eine LPAR mehr als einen dedizierten Prozessor bekommen, dann kann die gewünschte Anzahl an dedizierten Prozessoren über das Attribut *desired_procs* angegeben werden:

```
$ lpar create -m ms02 lpar2 proc_mode=ded desired_procs=2
    > lpar2
$
```

Im Profil *standard* der LPAR *lpar2* sind *2* dedizierte Prozessoren als *desired* (gewünscht) angegeben:

```
$ lpar -p standard lsproc lpar2
            PROC        PROCS            PROC_UNITS                        UNCAP   PROC
LPAR_NAME   MODE  MIN  DESIRED  MAX  MIN  DESIRED  MAX  SHARING_MODE      WEIGHT  POOL
lpar2       ded   1    2        2    -    -        -    keep_idle_procs   -       -
$
```

Neben dem Attribut *desired_procs* für die gewünschte Anzahl an dedizierten Prozessoren, gibt es noch 2 weitere Attribute und zwar *min_procs* und *max_procs*. Genauso, wie *desired_procs*, können auch diese Attribute auf der Kommandozeile beim Erzeugen einer LPAR angegeben werden. Der Wert von *min_procs* muss kleiner oder gleich dem Wert *desired_procs* sein, welcher wiederum kleiner oder gleich dem Wert *max_procs* sein muss:

```
   min_procs <= desired_procs <= max_procs
```

Der Wert von *min_procs* kommt mindestens in den folgenden beiden Situationen zum tragen:

- Eine LPAR wird aktiviert, es stehen aber nicht soviele Prozessoren wie über *desired_procs* gefordert zur Verfügung. In diesem Fall reduziert PowerVM die Anzahl der Prozessoren, welche der LPAR zugewiesen werden auf einen kleineren Wert. Allerdings darf dabei der Wert von *min_procs* nicht unterschritten werden.

- Bei einer aktiven LPAR mit laufendem Betriebssystem können dynamisch Prozessoren hinzugefügt oder weggenommen werden, ohne das Betriebssystem oder Applikationen stoppen zu müssen. Dabei kann die Anzahl der Prozessoren maximal auf den Wert von *max_procs* erhöht werden bzw. höchstens auf den Wert von *min_procs* reduziert werden.

Der Wert von *max_procs* wird, wie gerade beschrieben, bei der dynamischen Erhöhung von Prozessoren berücksichtigt.

Welche Attribute angegeben werden können und welche möglichen Werte diese Attribute haben, kann in der Online Hilfe nachgeschaut werden:

```
$ lpar help create
USAGE:
   lpar [-h <hmc>] [-m <ms>] [-p <profile>] create [{-b <blueprint>|-s <source_lpar>}] [-v]
[<lpar>] [<attributes> ...]

DESCRIPTION

Create a new LPAR on a managed system.

   -b : blueprint to use for creation'
   -s : source LPAR to use as blueprint'

Valid attributes:
   name : name for the LPAR
   lpar_id : the ID of the LPAR
   profile_name : name of the default profile
   lpar_env : type of LPAR
      aixlinux - AIX or Linux (default)
      os400 - IBM i
      vioserver - virtual I/O server
   min_mem : minimum amount of memory in MB
   desired_mem : desired amount of memory in MB
   max_mem : maximum amount of memory in MB
   mem_expansion : Active Memory Expansion
      0 - disable AME
      1.00-10.00 - expansion factor
…
$
```

5.1.1. Hinzufügen von dedizierten Prozessoren

Hat eine dedizierte LPAR eine aktive RMC-Verbindung zu den HMCs, dann können im laufenden Betrieb dedizierte Prozessoren hinzugefügt (und natürlich auch weggenommen) werden. Dazu müssen 2 Bedingungen erfüllt sein:

1. Die aktuelle Anzahl an dedizierten Prozessoren muß kleiner sein, als die maximale erlaubte Anzahl an dedizierten Prozessoren für die LPAR.

2. Die gewünschte Anzahl an Prozessoren muß auch noch verfügbar sein.

Wieviele dedizierte Prozessoren aktuell einer LPAR zugeordnet sind und wieviele maximal möglich sind, lässt sich mittels „*lpar lsproc*“ leicht bestimmen:

```
$ lpar lsproc aix02
            PROC        PROCS           PROC_UNITS                          UNCAP   PROC
LPAR_NAME  MODE  MIN  DESIRED  MAX  MIN  DESIRED  MAX  CURR_SHARING_MODE  WEIGHT  POOL
aix02      ded   1    2        4    -    -        -    keep_idle_procs    -       -
$
```

Die LPAR *aix02* besitzt aktuell 2 dedizierte Prozessoren und kann auf bis zu 4 dedizierte Prozessoren erweitert werden. Die Anzahl der Prozessoren ist natürlich auch vom Betriebssystem der LPAR aus feststellbar:

```
aix02 # lsdev -l proc*
proc0 Available 00-00 Processor
proc8 Available 00-08 Processor
aix02 #
```

Auf dem unterliegenden Managed System (*ms06*) sind aktuell noch *10.15* Prozessoren verfügbar:

```
$ ms lsproc ms06
NAME  INSTALLED  CONFIGURABLE  AVAIL  MAX_SHARED_PROC_POOLS
ms06  20.0       20.0          10.15  64
$
```

Damit können ohne Probleme 1 oder 2 dedizierte Prozessoren zur LPAR *aix02* hinzugefügt werden. Das Kommando hierfür lautet „*lpar addprocs*“ (*add processors*):

```
$ lpar addprocs aix02 1
$
```

Im Beispiel wurde um einen dedizierten Prozessor erweitert. Der Prozessor steht dem Betriebssystem sofort zur Verfügung, es ist keine weitere Konfiguration auf der AIX-LPAR notwendig:

```
aix02 # lsdev -l proc*
proc0  Available 00-00 Processor
proc8  Available 00-08 Processor
proc16 Available 00-16 Processor
aix02 #
```

Ist eine LPAR ausgeschaltet oder hat keine aktive RMC-Verbindung zu den HMCs, dann kann die Anzahl der dedizierten Prozessoren nur im Profil der LPAR geändert werden:

```
$ lpar -p standard addprocs aix02 1
$
```

Die Änderung wirkt sich allerdings erst aus, wenn die LPAR neu aktiviert wird (unter Angabe des geänderten Profils).

5.1.2. Wegnehmen von dedizierten Prozessoren

Auch für das Wegnehmen von dedizierten Prozessoren im laufenden Betrieb ist eine aktive RMC-Verbindung zu den HMCs erforderlich. Es können nur Prozessoren weggenommen werden, wenn die verbleibende Anzahl der dedizierten Prozessoren größer oder gleich der minimalen Anzahl von dedizierten Prozessoren ist.

Das Wegnehmen von Prozessoren geschieht, analog dem Hinzufügen, mit dem entsprechenden Kommando „*lpar rmprocs*“ (*remove processors*). Allerdings muß das Wegnehmen von Prozessoren nicht in allen Fällen funktionieren. Um dies zu demonstrieren, haben wir auf der AIX-LPAR *aix02* aus dem letzten Beispiel einige Prozesse besonders präpariert. Der Versuch einen dedizierten Prozessor wegzunehmen scheitert mit einer langen Fehlermeldung (1298 Zeilen!):

```
$ lpar rmprocs aix02 1
hmc01: chhwres -m ms06 -r proc  -o r -p aix02 --procs 1
ERROR: remote HMC command returned an error (1)
StdErr: HSCL294F The dynamic removal of processing resources failed: The operation to
remove processing has failed on partition aix02. The requested amount of processing to be
removed is 1 and the completed amount is 0.
StdErr: The OS standard output is:
…
StdErr: Invoking update_odm_smt
StdErr: Updated ODM smt_enabled for proc16 with state=true rc: 0
StdErr: smt_failures= 0
StdErr: remove_a_cpu failed! rc=1
StdErr:
StdErr: DR_TOTAL_RESOURCES=0
StdErr: ..cleanup_and_exit. rc=1
StdErr:
StdErr: The OS standard error is:
StdErr:
StdErr: 0930-033 Resource 0x10 is busy and cannot be released.
StdErr:
StdErr:
StdErr:
StdErr:
StdErr: The OS return code is 1.Please issue the lshwres command to list the processing
resources of the partition and to determine whether or not its pending and runtime
processing values match. If they do not match, problems with future processing-related
operations on the managed system may occur, and it is recommended that the rsthwres command
to restore processing resources be issued on the partition to synchronize its pending
processing value with its runtime processing value.
$
```

(Die Fehlermeldung ist hier nur in Ausschnitten abgebildet.)

Der Fehlermeldung ist zu entnehmen das der Versuch den Prozessor *proc16* wegzunehmen gescheitert ist, da der Prozessor beschäftigt (*busy*) ist: „*0930-033 Resource 0x10 is busy and cannot be released*.“. Das Problem ist an dieser Stelle, das es Prozesse gibt, die an einen festen Prozessor gebunden wurden:

```
aix02 # ps -eo THREAD |grep -vw -- -1
    USER      PID     PPID    TID S  CP PRI SC    WCHAN        F      TT    BND COMMAND
root 15597746 17104922      - A   0  60  1 f1000a01508a3bb0   200001  pts/0  16 sleep 3600
root 20709576 17104922      - A   0  60  1 f1000a0150a523b0   200001  pts/0   8 sleep 3600
root 22741088 17104922      - A   0  60  1 f1000a01508a9fb0   200001  pts/0   0 sleep 3600
aix02 #
```

Die Ausgabe des *ps*-Kommandos zeigt, das der Prozeß *15597746* an den Prozessor *16* (Spalte *BND* für *bind processor* in der Ausgabe) gebunden ist. D.h. der Prozeß darf nur auf diesem Prozessor laufen, was ein Wegnehmen des Prozessors verhindert!

Die Bindung lässt sich aber sehr leicht mit dem AIX-Kommando *bindprocessor* wieder aufheben:

```
aix02 # bindprocessor -u 15597746
aix02 #
aix02 # ps -eo THREAD |grep -vw -- -1
    USER      PID     PPID    TID S  CP PRI SC    WCHAN        F      TT    BND COMMAND
root 20709576 17104922      - A   0  60  1 f1000a0150a523b0   200001  pts/0   8 sleep 3600
root 22741088 17104922      - A   0  60  1 f1000a01508a9fb0   200001  pts/0   0 sleep 3600
aix02 #
```

Hinweis: Die Bindung wurde mit dem Kommando *bindprocessor* wie folgt eingerichtet:

```
aix02 # bindprocessor 15597746 16
aix02 #
```

Das erste Argumente ist dabei die *PID* des zu bindenden Prozesses und das zweite Argument die Nummer des Prozessors. Wobei die Prozessor-Nummer hier die Nummer des sogenannten logischen Prozessors ist (siehe Simultaneous Multi-Threading *SMT* später).

Damit kann der Prozessor nun ohne weitere Probleme weggenommen werden:

```
$ lpar rmprocs aix02 1
$
```

Damit sind anschließend wieder nur noch 2 dedizierte Prozessoren in der LPAR verfügbar:

```
aix02 # lsdev -l proc*
proc0 Available 00-00 Processor
proc8 Available 00-08 Processor
aix02 #
```

5.2. Shared Prozessoren (Micro-Partitioning)

Wird ein dedizierter Prozessor nur wenig ausgelastet, kann die nicht verwendete Rechenleistung nicht genutzt werden. Daher wurde mit Einführung von POWER5 Systemen von IBM die Micro Partitionierung eingeführt. Dabei werden ein oder mehrere Prozessoren von mehreren LPARs gemeinsam benutzt. Damit wird eine höhere Auslastung der Prozessoren erreicht. Die Rechenleistung eines physikalischen Prozessors wird dabei in 100 Anteile aufgeteilt. Jeder Anteil entspricht dabei 0.01 Processing Units (1 hundertstel eines Prozessors bzw. 1% eines Prozessors). Den LPARs können dann Processing Units mit einer Granularität von 0.01 Processing Units zugewiesen werden.

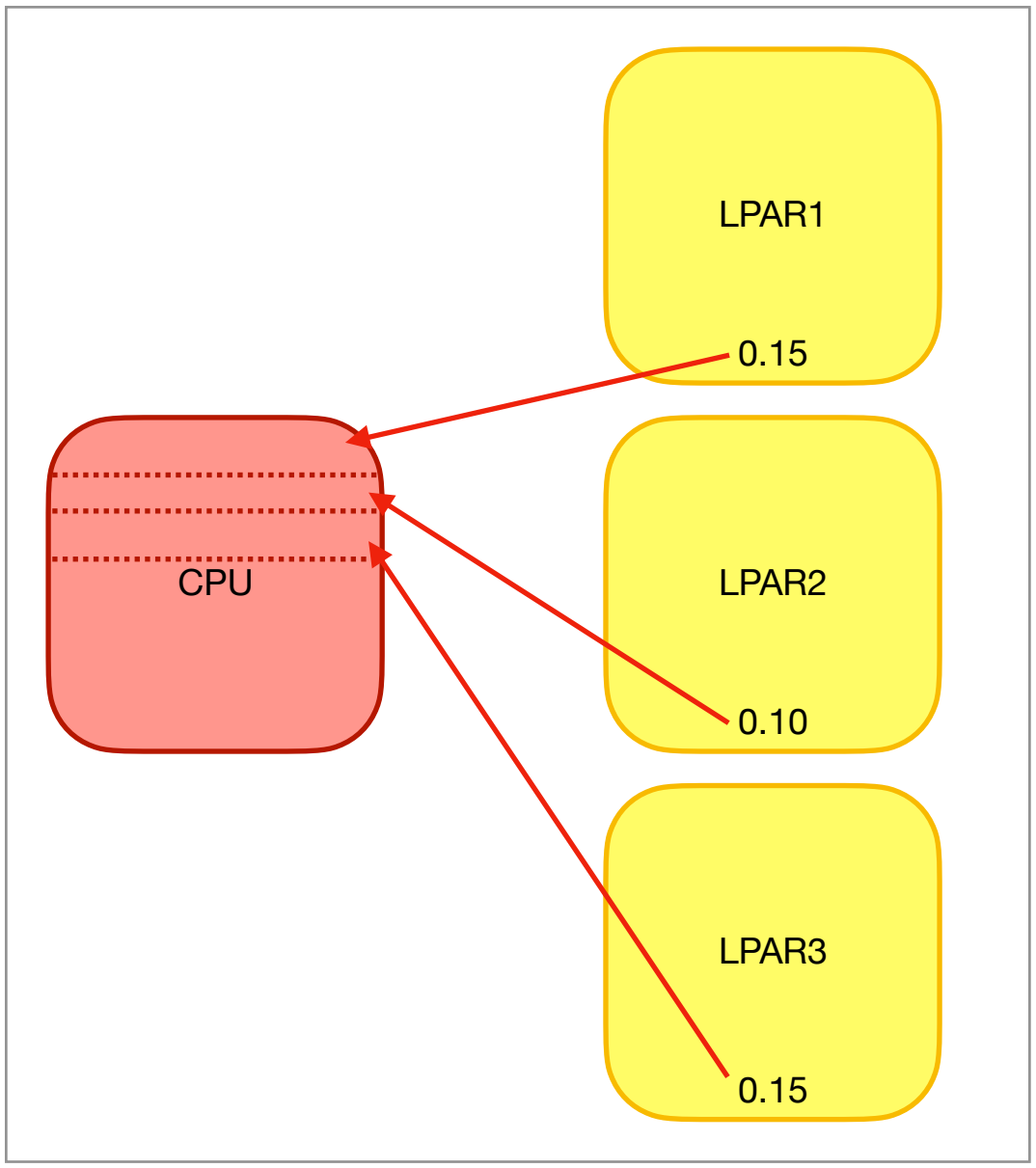


Bild 5.2: Aufteilen von Prozessor Anteilen auf 3 LPARs

In Bild 5.2 ist die Aufteilung der Prozessor Anteile eines physikalischen Prozessors auf 3 LPARs dargestellt. Dabei bekommt LPAR1 *0.15* Processing Units (*15%*), LPAR2 bekommt *0.10* Processing Units (*10%*) und LPAR3 bekommt *0.15* Processing Units (*15%*) der Rechenleistung des Prozessors. In der Summe sind das *0.40* Processing Units (*40%*), womit noch *0.60* Processing Units (*60%*) für weitere LPARs zur Verfügung stehen würden. Der Begriff Micro-Partitionierung kommt daher das hier eine Ressource in weitere Anteil aufgeteilt wird. Bei den dedizierten Prozessoren, wird als kleinste Einheit immer ein ganzer Prozessor einer LPAR zugeordnet, bei der Micro-Partitionierung wird ein Prozessor weiter aufgeteilt (in 100 Anteile) und diese kleineren Anteile können verschiedenen LPARs zugeordnet werden.

Die zugewiesenen Processing Units einer LPAR werden als *Entitlement* (Anspruch) oder *Entitled Capacity* bezeichnet. Der Hypervisor garantiert das eine LPAR die zugeordnete *Entitled Capacity* unter allen Umständen auch zugewiesen bekommt, unabhängig von der aktuellen Auslastung der unterliegenden Hardware.

Die Aufteilung der Anteile eines physikalischen Prozessors auf die zugehörigen LPARs erfolgt in einem Zeitscheiben-Verfahren. Dabei weist der Hypervisor in jedem *10ms* Zeitintervall den LPARs einen Anteil an der Prozessor-Zeit gemäß den konfigurierten Anteilen zu. Ein physikalischer Prozessor wird damit von mehreren LPARs gemeinsam genutzt, daher auch der Begriff Shared Prozessor für den gemeinsam genutzten physikalischen Prozessor, sowie Shared Prozessor LPAR für eine LPAR welche Shared Prozessoren verwendet. Wie in Bild 5.3 gezeigt ist, bekommt während jedes *10ms* Zeitintervalls jede LPAR den Shared Prozessor für ein kurzes Zeitintervall (entsprechend den konfigurierten Anteilen der LPAR) zugewiesen. Die gelbe LPAR aus dem Bild mit *0.15* Processing Units bekommt entsprechend einen Anteil von *1.5ms* in jedem *10ms* Zeitintervall zugewiesen.

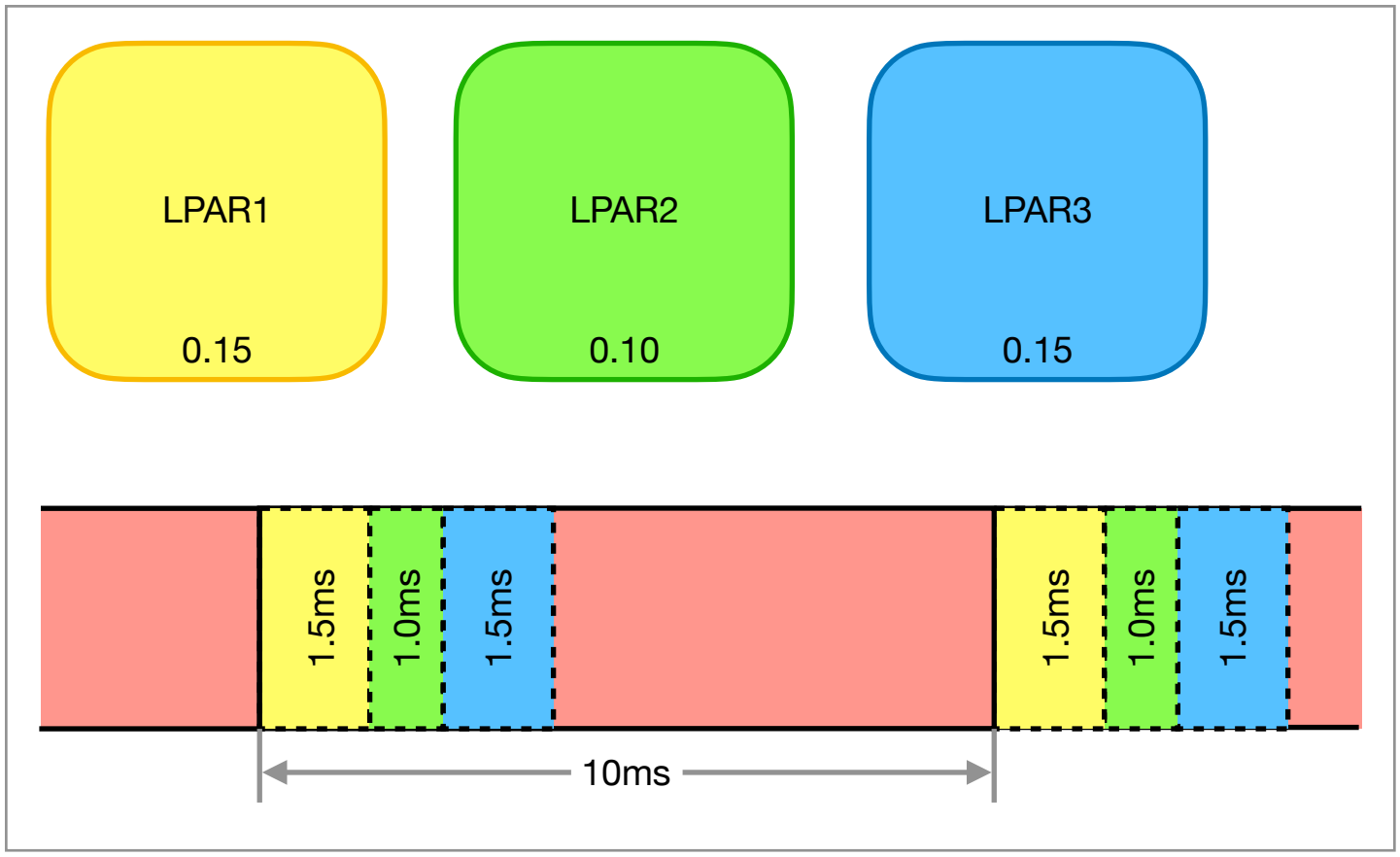


Bild 5.3: Zeitscheiben-Verfahren des Hypervisors

Wie bei LPARs mit dedizierten Prozessoren, lässt sich auch beim Anlegen von Shared Prozessor LPARs die Anzahl der gewünschten Prozessoren angeben. Es werden aber keine physikalischen Prozessoren zugeordnet, sondern sogenannte virtuelle Prozessoren. Ein virtueller Prozessor in einer LPAR erlaubt es Anteile an einem physikalischen Prozessor gemäß dem oben angesprochenen Zeitscheiben-Verfahren zu bekommen. Wird eine LPAR z.B. mit 8 virtuellen Prozessoren konfiguriert, dann kann sie bis zu 8 physikalische Prozessoren parallel nutzen. Wieviele Anteile die LPAR an den physikalischen Prozessoren bekommt, wird wie gehabt über das zugeordnete *Entitlement* (oder *Entitled Capacity*) der LPAR bestimmt.

Das *LPAR-Tool* erzeugt, wenn nicht anders angegeben, per Default Shared Prozessor LPARs. Die gewünschte Anzahl an virtuellen Prozessoren kann hierbei analog wie bei den dedizierten LPARs über das Attribut *desired_procs* angegeben werden:

```
$ lpar create -m ms02 lpar10 desired_procs=2
.
    > lpar10
$
```

Die schon von den dedizierten LPARs bekannten Attribute *min_procs* und *max_procs* behalten ihre Bedeutung. Die beiden Attribute beziehen sich jetzt aber nicht auf dedizierte Prozessoren, sondern auf virtuelle Prozessoren. Eine LPAR kann nur aktiviert werden, wenn mindestens die durch *min_procs* angegebene Anzahl an virtuellen Prozessoren zugeordnet werden kann. Die Anzahl der virtuellen Prozessoren kann online in den von *min_procs* und *max_procs* vorgegebenen Grenzen dynamisch geändert werden.

Die im Beispiel oben neu erzeugte LPAR *lpar10* hat die folgende Konfiguration:

```
$ lpar -p standard lsproc lpar10
            PROC          PROCS            PROC_UNITS                        UNCAP      PROC
LPAR_NAME   MODE    MIN   DESIRED   MAX   MIN   DESIRED   MAX   SHARING_MODE   WEIGHT   POOL
lpar10      shared  1     2         2     0.2   0.2       0.2   uncap          100      DefaultPool
$
```

Der Prozessor-Mode (*proc_mode*) ist *shared*. Die minimale Anzahl an virtuellen Prozessoren (*min_procs*) ist 1, die gewünschte Anzahl an virtuellen Prozessoren (*desired_procs*) ist 2 und die maximale Anzahl an virtuellen Prozessoren (*max_procs*) ist ebenfalls 2.

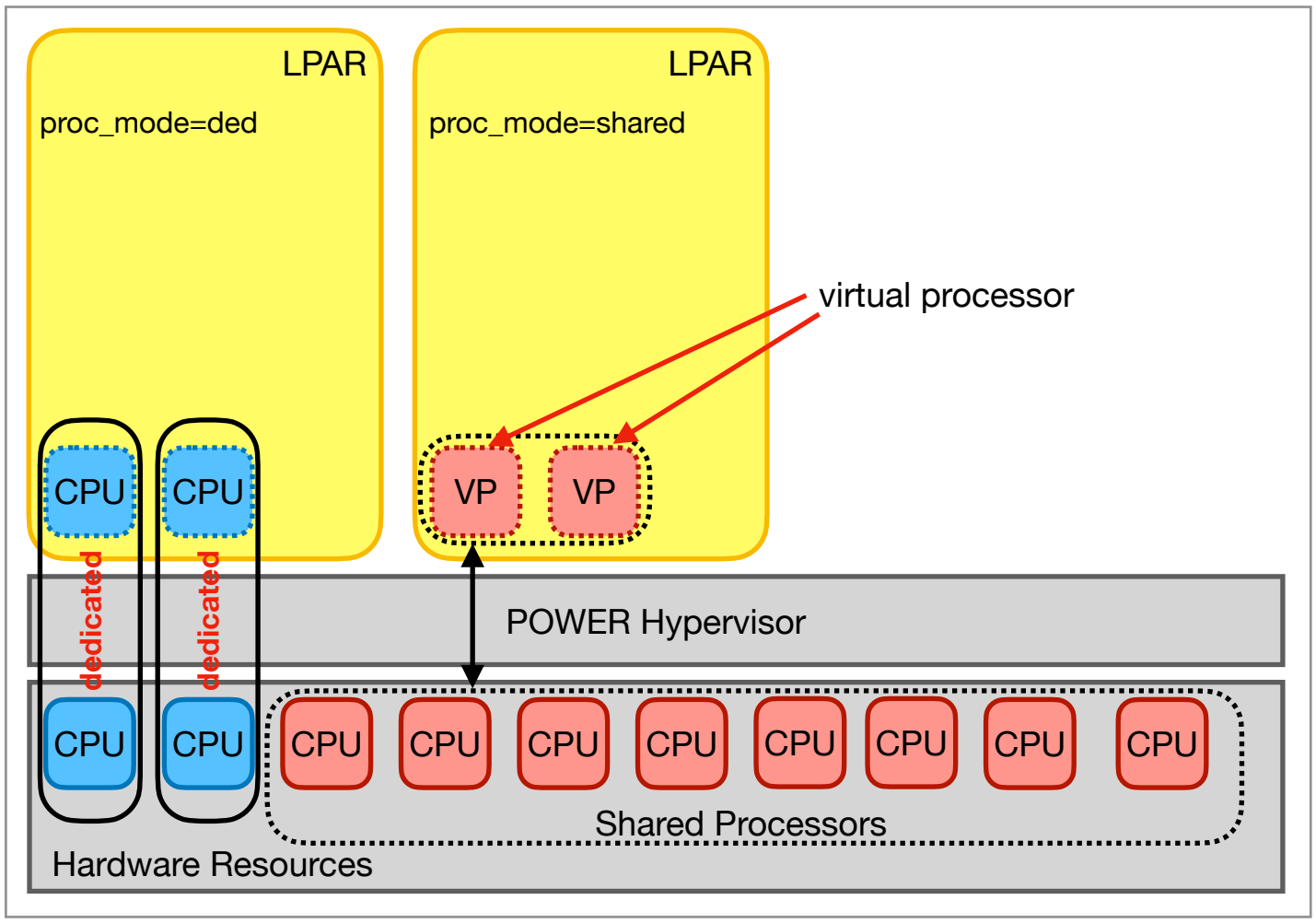


Bild 5.4: Partitionierung mit dedizierten und shared Prozessoren

Alle physikalischen Prozessoren die nicht als dedizierte Prozessoren verwendet werden, sind automatisch Shared Prozessoren und bilden den sogenannten Physical Shared Prozessor Pool. In Bild 5.4 sind 2 Prozessoren (blau) dedicated Prozessoren und die restlichen 8 Prozessoren (rot) sind damit Shared Prozessoren. Die virtuellen Prozessoren von Shared Prozessor LPARs bekommen ihr Entitlement aus dem Pool der Shared Prozessoren zugewiesen. Dabei muss nicht notwendigerweise in jedem *10ms* Zeitintervall der gleiche physikalische Prozessor zugewiesen werden, die Zuordnung kann auch wechseln, wenn die Ressource-Verteilung durch den Hypervisor dies notwendig macht. Die beiden virtuellen Prozessoren der gezeigten Shared Prozessor LPAR können 2 verschiedenen physikalischen Prozessoren in einem Zeitintervall zugewiesen werden, es ist aber durchaus auch möglich das beiden virtuellen Prozessoren der gleiche physikalische Prozessor zugewiesen wird (natürlich nicht zur gleichen Zeit). Die Zuordnung von virtuellen Prozessoren zu physikalischen Prozessoren ist also nicht fest, sondern kann sich dynamisch je nach Erfordernissen ändern.

Das gewünschte *Entitlement* (*Entitled Capacity*) kann beim Erzeugen einer LPAR über das Attribut *desired_proc_units* angegeben werden. Pro virtuellem Prozessor müssen mindestens *0.05* Processing Units konfiguriert werden (bei POWER7 oder älter *0.10* Processing Units). Bei 2 virtuellen Prozessoren (*desired_procs*=2) müssen damit für *desired_proc_units* mindestens 2 * *0.05* = *0.10* Processing Units angegeben werden. Auch beim *Entitlement* können ein minimaler Wert (*min_proc_units*) und ein maximaler Wert (*max_proc_units*) angegeben werden. Der Wert von *min_proc_units* bestimmt die minimal benötigten Processing Units um eine LPAR aktivieren zu können. Die Werte von *min_proc_units* und *max_proc_units* geben den Bereich an, innerhalb dessen das *Entitlement* zur Laufzeit einer LPAR dynamisch geändert werden kann. Der Wert von *min_proc_units* muss kleiner

oder gleich dem Wert *desired_proc_units* sein, welcher wiederum kleiner oder gleich dem Wert *max_proc_units* sein muss:

```
min_proc_units <= desired_proc_units <= max_proc_units
```

Das folgende Beispiel demonstriert, wie diese Attribute beim Anlegen einer neuen LPAR angegeben werden können:

```
$ lpar create -m ms02 lpar11 desired_procs=2 max_procs=4 desired_proc_units=0.8
max_proc_units=1.6
.
    > lpar11
$
```

Die LPAR *lpar11* wurde mit 2 gewünschten virtuellen Prozessoren und maximal 4 virtuellen Prozessoren angelegt. Das gewünschte *Entitlement* (*desired_proc_units*) ist *0.8* Processing Units und als Maximum sind *1.6* Processing Units angegeben. Die nicht explizit angegebenen Attribute *min_procs* und *min_proc_units* werden vom *LPAR-Tool* automatisch mit Werten belegt. Im Profil *standard* lassen sich alle Attribute nachschauen:

```
$ lpar -p standard lsproc lpar11
            PROC          PROCS            PROC_UNITS                      UNCAP      PROC
LPAR_NAME  MODE    MIN  DESIRED  MAX  MIN  DESIRED  MAX  SHARING_MODE  WEIGHT  POOL
lpar11     shared  1    2        4    0.1  0.8      1.6  uncap         100     DefaultPool
$
```

Die *0.8* Processing Units werden dabei nach Möglichkeit auf 2 physikalische Prozessoren aufgeteilt, die LPAR bekommt dabei in jedem *10ms* Zeitintervall *4ms* (entspricht *0.4* Processing Units) von beiden physikalischen Prozessoren zugeteilt. Die Zuteilung kann dabei wie in Bild 5.5 gezeigt gleichzeitig auf beide physikalische Prozessoren erfolgen, es ist aber auch möglich das die Zuteilung zu unterschiedlichen Zeitpunkten stattfindet, z.B. auf dem ersten physikalischen Prozessor erfolgt die Zuteilung sofort am Anfang des *10ms* Zeitintervalls und auf dem zweiten physikalischen Prozessor erst einige Millisekunden später. Prinzipiell kann dies auch in jedem *10ms* Zeitintervall unterschiedlich sein. Es ist lediglich garantiert das die LPAR ihr Entitlement von *0.8* Processing Units in jedem *10ms* Zeitintervall bekommt.

Sollten keine 2 physikalischen Prozessoren verfügbar sein, wird beiden virtuellen Prozessoren der gleiche physikalische Prozessor zugeordnet. Beide virtuellen Prozessoren bekommen dann eine Zuteilung von jeweils *4ms* des gleichen physikalischen Prozessors, natürlich nacheinander. Das sollte aber für eine gute Performance vermieden werden.

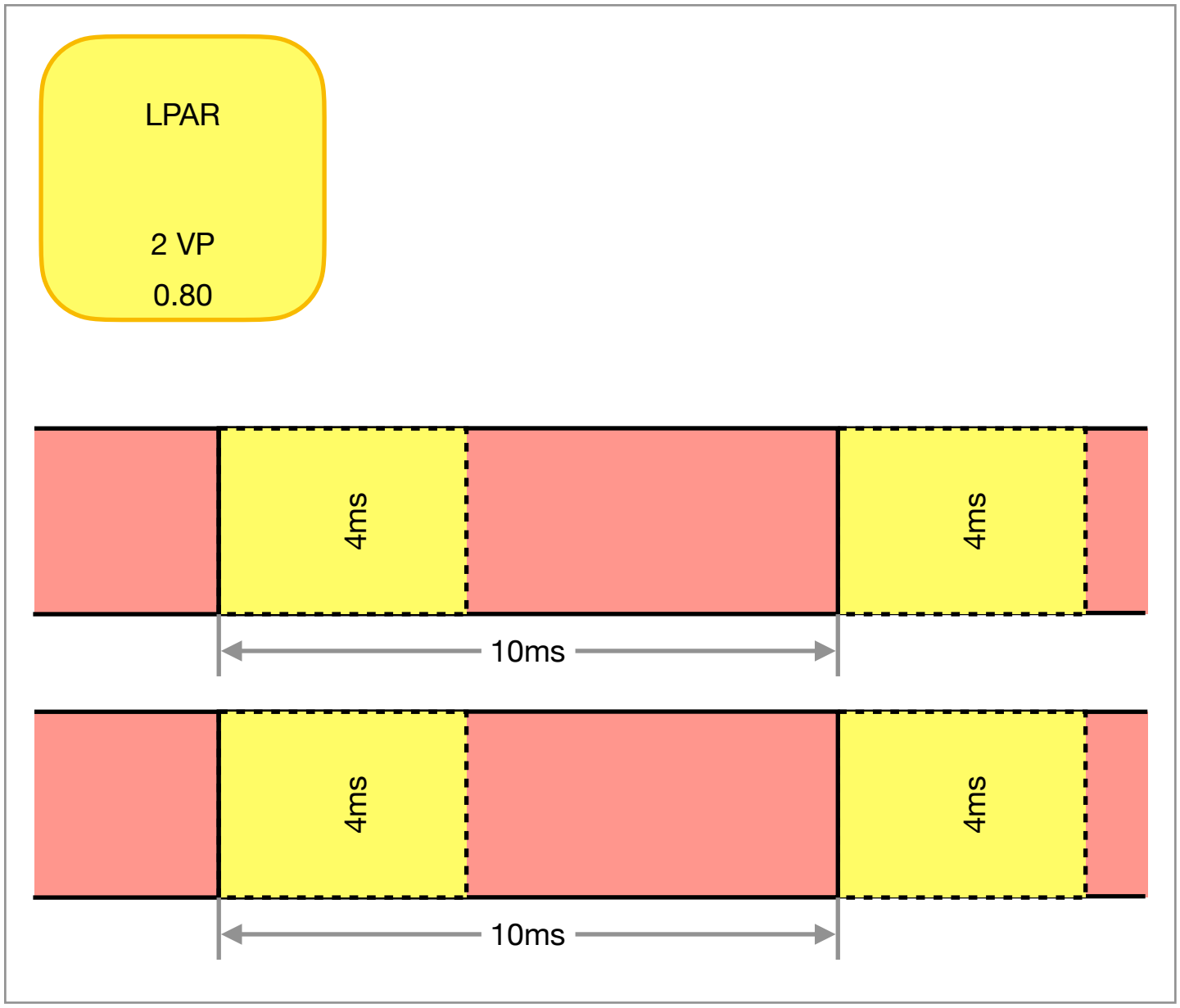


Bild 5.5: Entitlement bei 2 virtuellen Prozessoren wird auf 2 physikalische Prozessoren gleichmäßig aufgeteilt.

Das maximal mögliche Entitlement sind *1.0* Processing Units pro virtuellem Prozessor! Zusammen mit der Bedingung das pro virtuellem Prozessor ein Entitlement von mindestens *0.05* konfiguriert werden muß, erlaubt dies noch einen weitern Bereich von möglichen Werten. In der nachfolgenden Tabelle ist bei vorgegebener Anzahl von virtuellen Prozessoren der erlaubte Bereich des Entitlements aufgelistet:

Anzahl virtuelle Prozessoren	Minimum Entitlement	Maximales Entitlement
1	0.05	1.00
2	0.10	2.00
3	0.15	3.00
4	0.20	4.00
5	0.25	5.00
N	N * 0.05	N * 1.00

Wieviele Konfiguration-Möglichkeiten es gibt, lässt sich noch besser sehen, wenn man das Entitlement vorgibt und sich die minimal und maximale mögliche Anzahl von virtuellen Prozessoren anschaut, die mit dem Entitlement möglich sind:

Entitlement	Minimale Anzahl virtuelle Prozessoren	Maximale Anzahl virtuelle Prozessoren
0.05	1	1
0.10	1	2
0.20	1	4
0.40	1	8
0.80	1	16
1.00	1	20
1.01	2	20
2.01	3	40
EC	ceil(EC / 1.00)	floor(EC / 0.05)

Ein Entitlement von *0.40* beispielsweise, kann mit 1, 2 oder bis zu 8 virtuellen Prozessoren realisiert werden.

5.2.1. Prozessor Sharing-Mode

Beim Zeitscheiben-Verfahren des POWER Hypervisors ist jeder LPAR ihr Entitlement garantiert. Hat man z.B. auf einem Managed System mit nur einem physikalischen Prozessor, lediglich 3 LPARs, *LPAR1* mit einem Entitlement von *0.15*, *LPAR2* mit einem Entitlement von *0.10* und *LPAR3* mit einem Entitlement von *0.15*, dann ist dies in der Summe ein Entitlement von *0.15 + 0.10 + 0.15 = 0.40*. Damit sind in jedem *10ms* Zeitintervall *4ms* für die 3 LPARs garantiert, es bleiben aber damit noch *6ms*, die keiner LPAR garantiert sind, wie in Bild 5.6 dargestellt ist.

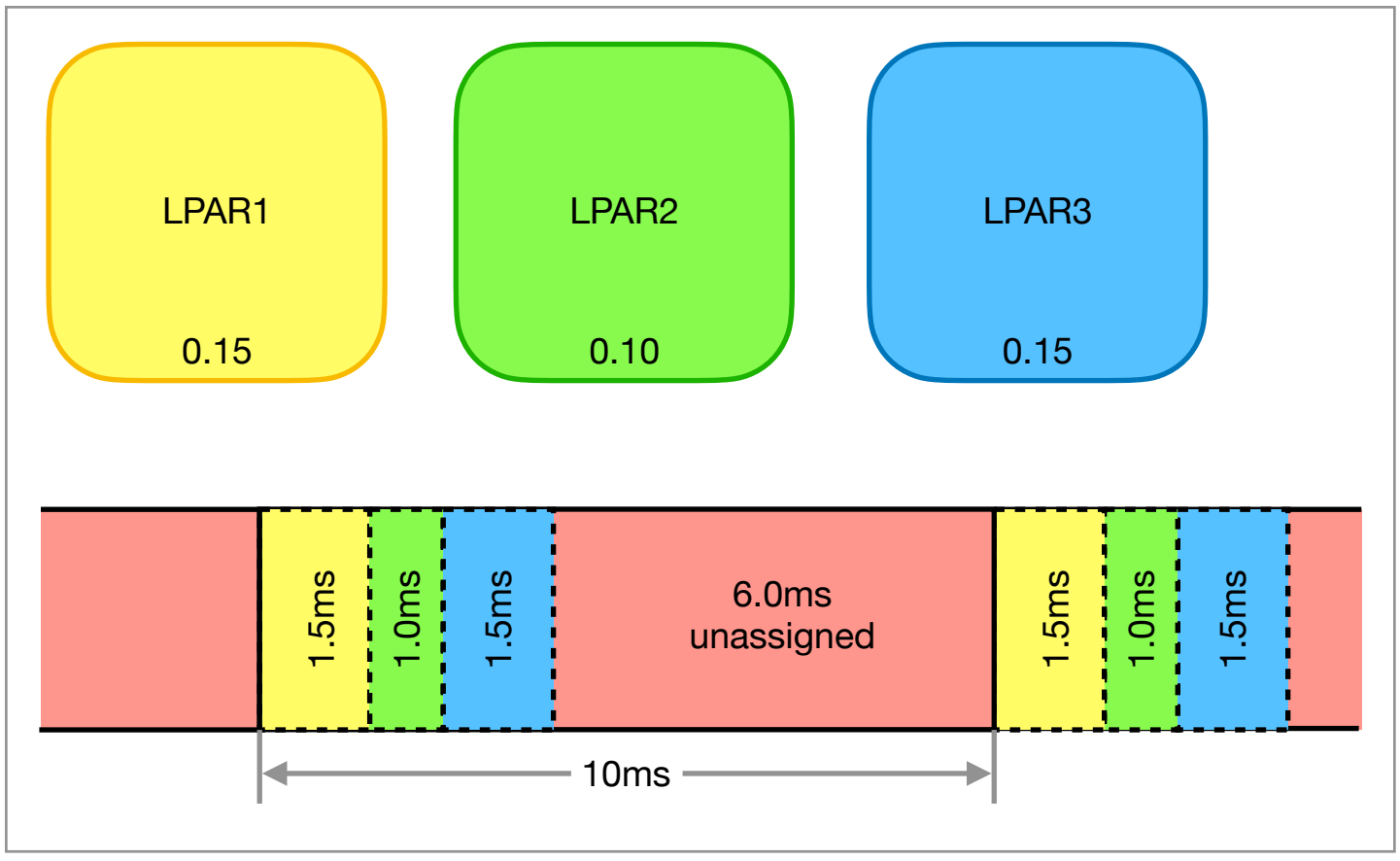


Bild 5.6: Nicht zugewiesene Zeit beim Zeitscheiben-Verfahren des Hypervisors

Diese nicht zugeteilte CPU-Zeit könnte unter den 3 aktiven LPARs aufgeteilt werden, um eine bessere Performance zu erzielen. Das würde einem Zuwachs an CPU-Zeit von 150% entsprechen!

Ob eine LPAR Anteile an nicht zugewiesener bzw. ungenutzter CPU-Zeit bekommen darf, kann über den Prozessor Sharing-Mode (Attribute *sharing_mode*) konfiguriert werden. Für Shared-Prozessor LPARs sind die folgenden beiden Werte möglich:

- *cap*: Die CPU-Zeit ist auf das garantierte Entitlement beschränkt. Der LPAR werden keine zusätzlichen Anteile bei Verfügbarkeit zugewiesen.
- *uncap*: Über das garantierte Entitlement hinaus, kann die LPAR zusätzliche CPU-Anteile an ungenutzten Prozessor-Ressourcen bekommen. Dies ist der Default für Shared-Prozessor LPARs.

Der Sharing-Mode kann beim Anlegen einer LPAR angegeben werden:

```
$ lpar create -m ms02 lpar12 desired_procs=2 desired_proc_units=0.8 sharing_mode=cap
.
    > lpar12
$
```

Der Sharing-Mode kann nur im Profil geändert werden, man kann also nicht zur Laufzeit einer LPAR vom Sharing-Mode *cap* auf *uncap* oder umgekehrt wechseln. Im Profil lässt sich der Sharing-Mode am einfachsten mit dem Kommando „*lpar chproc*“ unter Angabe der Option „*-p*“ und des Profilnamens ändern:

```
$ lpar -p standard chproc lpar11 sharing_mode=cap
$
```

Damit sich die Änderung auswirkt, muß die LPAR heruntergefahren werden, und unter Angabe des geänderten Profils neu aktiviert werden.

5.2.2. Sharing-Mode *cap*

Wurde für eine LPAR der Sharing-Mode *cap* konfiguriert, kann die LPAR maximal den ihr garantierten Anteil (*desired_proc_units*) an den Prozessoren bekommen. Zusätzliche CPU-Zeit wird einer solchen LPAR nicht zugewiesen. Garantierte CPU-Anteile welche von der LPAR nicht gebraucht werden, werden an den Shared Prozessor Pool zurück gegeben und können von anderen LPARs mit Sharing-Mode *uncap* verwendet werden.

In Bild 5.7 ist die Prozessor-Auslastung einer *capped* LPAR gezeigt. Das Entitlement ist *0.6* (gelbe Linie) und es wurde ein virtueller Prozessor konfiguriert (rote Linie). Verbraucht die LPAR ihr garantiertes Entitlement nicht, oder nur teilweise, dann wird der nicht genutzte Anteil anderen LPARs überlassen (*ceded capacity* - hell gelb). Diese Anteile gehen zurück an den Hypervisor (bzw. den Shared Prozessor Pool) und werden auf andere LPARs aufgeteilt. Der von der LPAR genutzte Anteil des garantierten Entitlement ist in der Graphik grün eingetragen. Die Summe von verbrauchtem garantierten Anteil (grün) und dem nicht genutzten Anteil (hell gelb) ist gleich dem garantierten Entitlement, hier *0.6*. Deutlich ist in der Graphik zu erkennen, daß die LPAR keine Prozessor-Anteile über ihr garantiertes Entitlement hinaus bekommt.

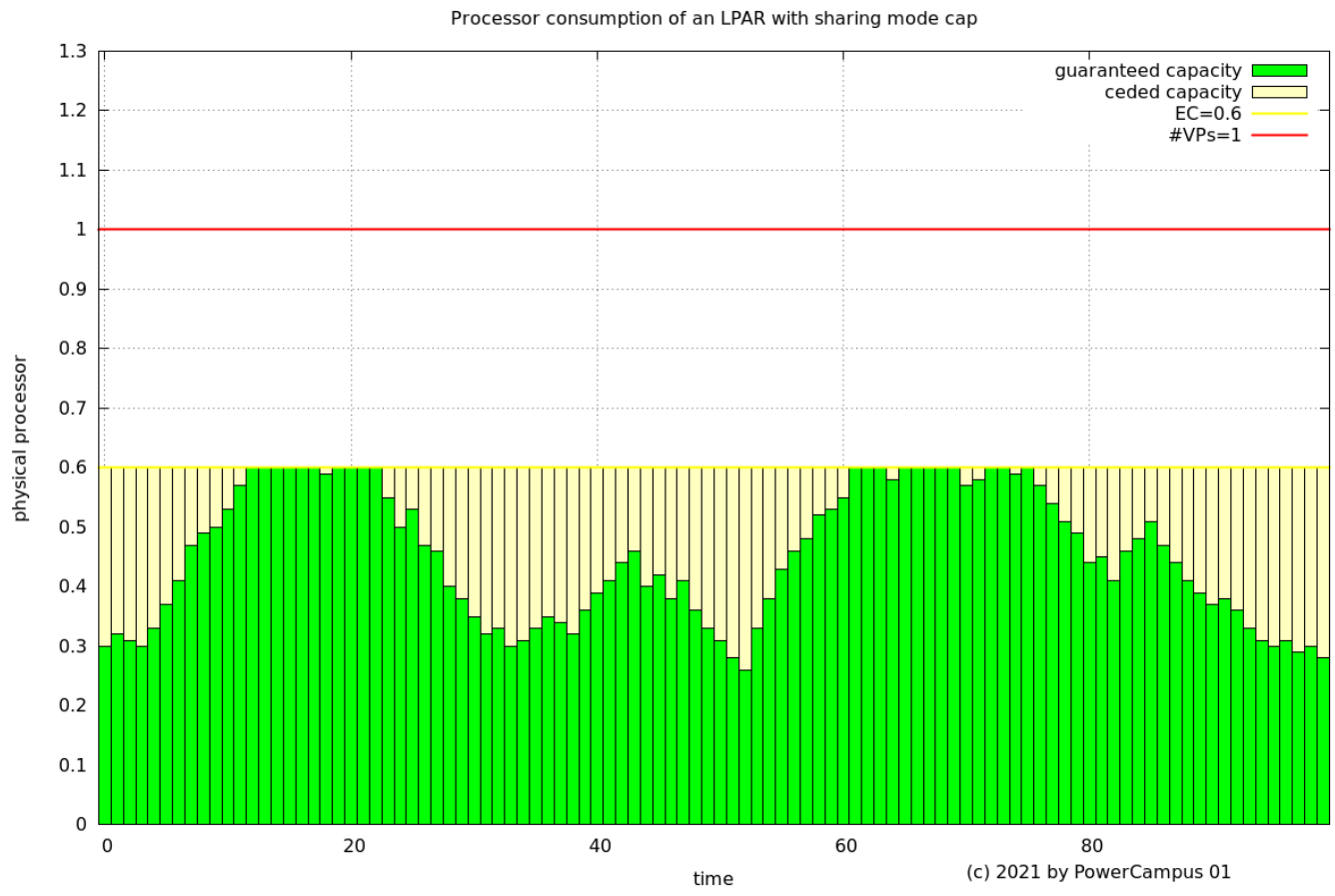


Bild 5.7: Prozessor Auslastung einer LPAR mit Sharing-Mode *cap*

5.2.3. Sharing-Mode *uncap* und *uncapped_weight*

Shared-Prozessor LPARs, die den Sharing-Mode *uncap* verwenden, konkurrieren miteinander um zusätzliche CPU-Zeit. Wieviel zusätzliche CPU-Zeit eine solche LPAR bekommen kann, hängt von drei Faktoren ab:

- Menge an zusätzlicher verfügbarer CPU-Zeit, die keiner LPAR zugewiesen ist.
- Der Gewichtung der LPAR, diese kann für jede uncapped Shared-Prozessor LPAR über das Attribut *uncapped_weight* auf einen Wert zwischen 0 und 255 gesetzt werden (Default ist 128). Je größer der Wert desto höher ist der Anteil den die LPAR bekommt.
- Wieviele weitere LPARs gerade um zusätzliche CPU-Zeit konkurrieren und deren Gewichtung. Eine LPAR die *idle* ist, konkurriert nicht um zusätzliche CPU-Zeit.

Die Ermittlung der zusätzlichen CPU-Anteile erfolgt dann wie folgt:

1. Die Gewichtungen aller LPARs die um zusätzliche Zeit konkurrieren werden addiert.
2. Der Anteile einer individuellen LPAR ergibt sich durch den Quotienten aus der eigenen Gewichtung und der Summe der Gewichtungen der konkurrierenden LPARs.
3. Der Anteil wird mit der zusätzlich zur Verfügung stehenden CPU-Zeit multipliziert.

In Bild 5.8 ist dies für 3 LPARs mit den Gewichtungen *LPAR1 150*, *LPAR2 50* und *LPAR3 100* gezeigt. Alle 3 LPARs konkurrieren um die nicht zugewiesenen 6ms des physikalischen Prozessors.

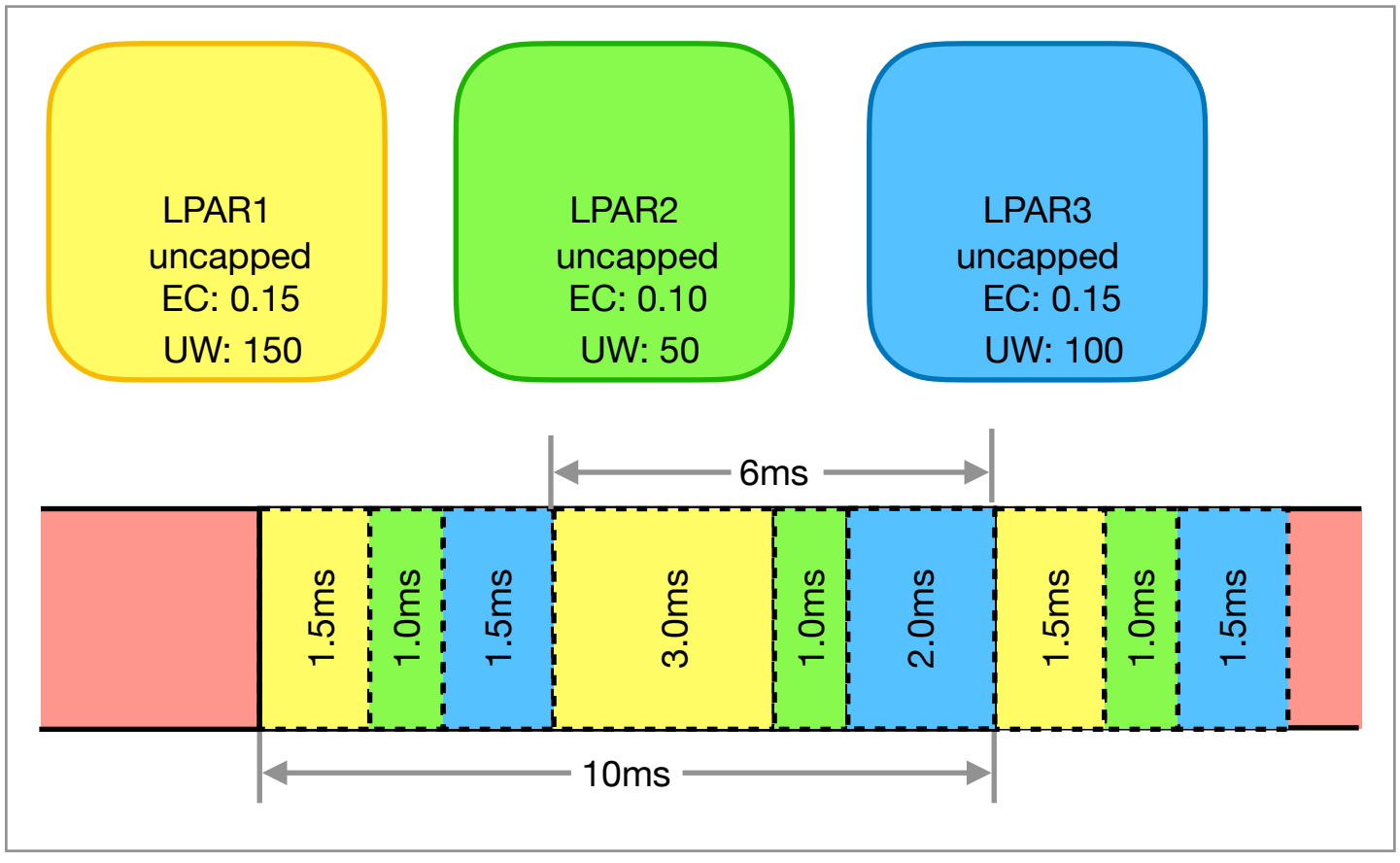


Bild 5.8: Aufteilung der zusätzlichen Zeit gemäß den Gewichtungen

Die Ermittlung der Anteile für die einzelnen LPARs erfolgt dabei wie oben beschrieben:

1. Bildung der Summe der Gewichtungen der konkurrierenden LPARs: *150 + 50 + 100 = 300*.

2. Ermittlung des Anteils eine LPAR durch Quotienten-Bildung:

   - LPAR1: *150 / 300 = 1/2*

   - LPAR2: *50 / 300 = 1/6*

   - LPAR3: *100 / 300 = 2/6*

3. Multipliziert mit dem zur Verfügung stehenden CPU-Anteil von 6ms ergibt sich dann:

   - LPAR1: *6.0 ms * 1/2 = 3.0 ms*

   - LPAR2: *6.0 ms * 1/6 = 1.0 ms*

   - LPAR3: *6.0 ms * 2/6 = 2.0 ms*

Die zusätzlichen Anteile an CPU-Zeit sind allerdings nicht garantiert und können sich auch in jedem 10ms Zeitintervall ändern! Z.B. könnte eine weitere LPAR aktiviert werden, welche dann natürlich ihr Entitlement auch garantiert bekommt. Oder das Entitlement einer laufenden LPAR wird erhöht, womit dann weniger nicht zugewiesene Kapazität zur Verfügung steht.

Benötigt eine LPAR die ihr garantierten Processing Units in einem Zeitintervall nicht, dann gehen die nicht genutzten Processing Units an den Hypervisor zurück, der diese dann nach dem obigen Schema auf andere LPARs aufteilt.

Die Gewichtung einer LPAR (*uncapped_weight*) kann dynamisch zur Laufzeit geändert werden. Dazu kann das Kommando „*lpar chproc*“ (*change processors*) verwendet werden. Hierbei wird neben der LPAR einfach das Attribut *uncapped_weight* mit einem neuen Wert zwischen *0* und *255* angegeben:

```
$ lpar chproc aix05 uncapped_weight=100
$
```

Wird für *uncapped_weight* der Wert *0* angegeben, dann kann die LPAR keine zusätzlichen CPU-Anteile über ihr Entitlement hinaus bekommen. Sie verhält sich dann genauso wie eine LPAR mit dem Sharing-Mode *cap*!

Soll die Gewichtung in einem Profil geändert werden, muß das Kommando mit der Option „*-p*“ und dem Profilnamen gestartet werden:

```
$ lpar -p standard chproc aix05 uncapped_weight=100
$
```

In Bild 5.9 ist die Prozessor-Auslastung einer LPAR mit Sharing-Mode *uncap* gezeigt. Die LPAR wurde mit einem Entitlement von *0.6* (gelbe Linie) und einem virtuellen Prozessor konfiguriert.

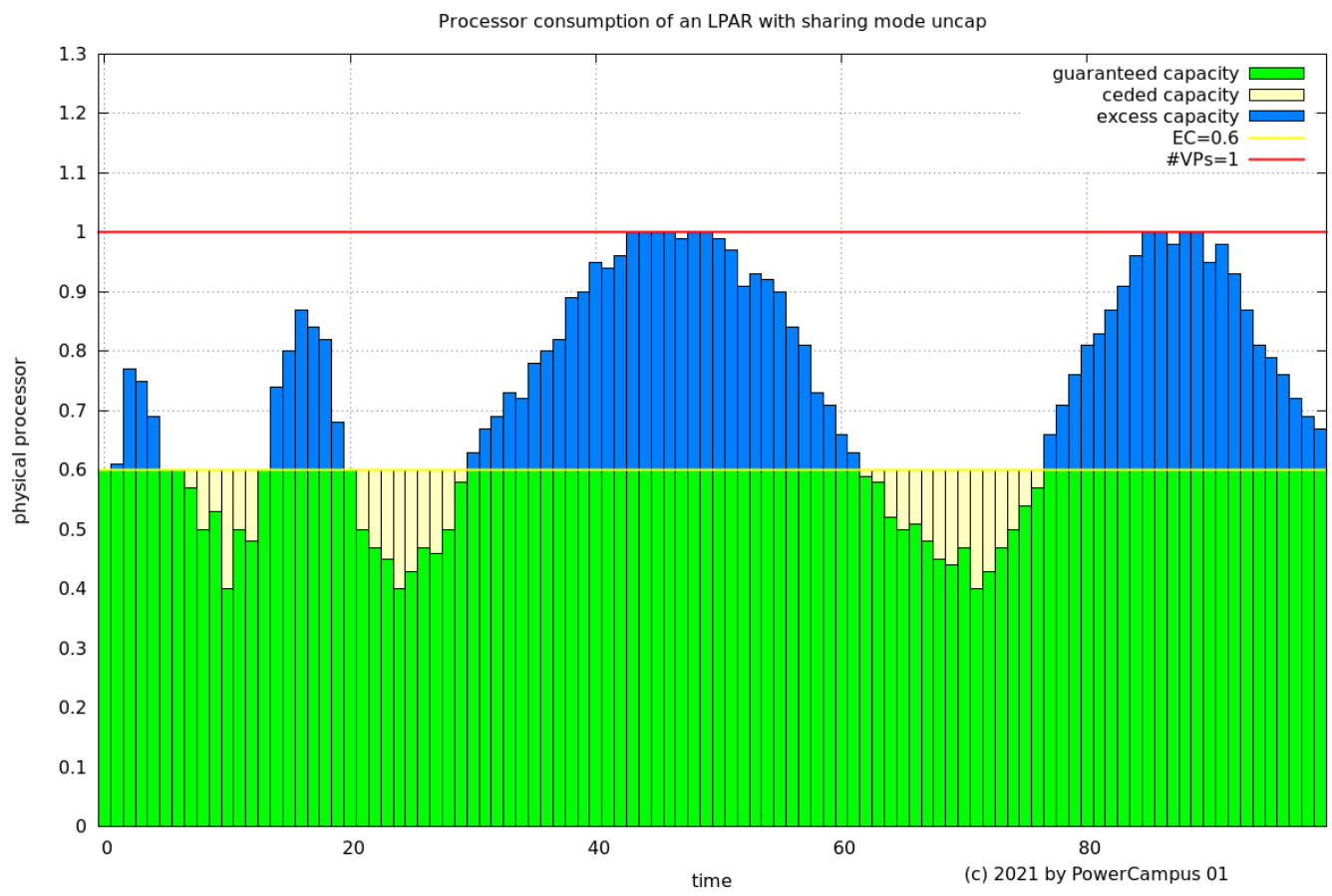


Bild 5.9: Prozessor Auslastung einer LPAR mit Sharing-Mode *uncap*

Mit nur einem virtuellen Prozessor kann maximal nur ein physikalischer Prozessor zu 100% verwendet werden, die maximal mögliche Prozessor-Auslastung ist daher *1.0* (rote Linie). Verbraucht eine LPAR in einem Zeitintervall ihr garantiertes Entitlement nicht, dann wird der nicht verwendete Teil des garantierten Anteils anderen LPARs überlassen (*ceded capacity* - hell gelb). Der verbrauchte Teil des garantierten Anteils ist in grün eingezeichnet. Es gilt: verbrauchter Anteil des garantierten Entitlement plus *ceded* Kapazität ist gleich dem garantierten Entitlement.

Das ist in der Graphik auch sehr gut erkennbar. Da die LPAR *uncapped* ist, kann sie bei Bedarf über den garantierten Anteil hinaus weitere Prozessor-Anteile bekommen, abhängig von der Menge an verfügbaren Anteilen und der Zuteilung an andere LPARs. Die zusätzlichen Prozessor-Anteile sind im Bild in blau eingezeichnet. Der grüne (garantierte) Anteil plus der blaue (zusätzliche) Anteil kann jedoch in Summe den Wert *1.0* nicht übersteigen. In der Graphik ist zu sehen, das die verbrauchten Prozessor-Anteile zweimal bei *1.0* „anstoßen".

Hinweis: Die maximal mögliche Prozessor-Auslastung hängt von der Anzahl der virtuellen Prozessoren ab. Insbesondere gibt der Wert von *max_proc_units* nicht die maximal mögliche Anzahl von Processing-Units an, die eine LPAR bekommen kann! Der Wert *max_proc_units* gibt an, bis zu welchem Wert das garantierte Entitlement angehoben werden kann.

5.2.4. Hinzufügen von virtuellen Prozessoren und Processing Units

Auch bei Shared-Prozessor LPARs können Prozessoren (virtuelle Prozessoren) dynamisch zur Laufzeit hinzugefügt werden. Die Anzahl der resultierenden virtuellen Prozessoren darf dabei die Anzahl der maximalen virtuellen Prozessoren für die LPAR dabei nicht übersteigen. Beim Hinzufügen von virtuellen Prozessoren muß das aktuelle Entitlement berücksichtigt werden. Ist das aktuelle Entitlement beispielsweise *0.20* und hat die LPAR schon *4* virtuelle Prozessoren, dann kann kein weiterer virtueller Prozessor hinzugefügt werden, ohne vorher das Entitlement zu erhöhen!

Als Beispiel betrachten wir die LPAR *aix05*, mit dem Ziel die Anzahl der virtuellen Prozessoren um *2* zu erhöhen. Aktuell besitzt die LPAR die folgenden Prozessor-Resourcen:

```
$ lpar lsproc aix05
            PROC          PROCS            PROC_UNITS                          UNCAP   PROC
LPAR_NAME  MODE    MIN  DESIRED  MAX  MIN  DESIRED  MAX  CURR_SHARING_MODE  WEIGHT  POOL
aix05      shared  1    1        4    0.1  0.1      2.0  uncap              100
DefaultPool
$
```

Aktuell hat die LPAR nur einen virtuellen Prozessor und ein Entitlement von *0.1*. Für eine Erhöhung um *2* virtuelle Prozessoren, auf insgesamt *3* virtuelle Prozessoren, wird ein Entitlement von mindestens *3 * 0.05* gleich *0.15* benötigt. Dementsprechend schlägt der Versuch auch fehl, die Anzahl der virtuellen Prozessoren um *2* zu erhöhen:

```
$ lpar addprocs aix05 2
hmc01: chhwres -m ms06 -r proc  -o a -p aix05 --procs 2
ERROR: remote HMC command returned an error (1)
StdErr: HSCL1566 The operation failed because the ratio of assigned processing units to
assigned virtual processors for partition 5 would have been less than the minimum ratio
supported by the operating system on the partition.
$
```

Um die Anzahl der Processing-Units (Entitlement) zu erhöhen, kann das Kommando „*lpar addprocunits*" (*add processing units*) verwendet werden. Wir erhöhen das Entitlement um *0.05*:

```
$ lpar addprocunits aix05 0.05
$
```

Nachdem die LPAR nun ein Entitlement von insgesamt *0.15* hat, kann auch die Erhöhung um *2* virtuelle Prozessoren erfolgreich durchgeführt werden:

```
$ lpar addprocs aix05 2
$
```

Soll die Änderung nur in einem Profil durchgeführt werden, dann ist die Option „*-p*“ mit dem Profilnamen zu verwenden:

```
$ lpar -p standard addprocunits aix05 0.05
$ lpar -p standard addprocs aix05 2
$
```

5.2.5. Wegnehmen von virtuellen Prozessoren und Processing Units

Das Wegnehmen von Prozessor Ressourcen funktioniert analog dem Hinzufügen. Es gibt hierfür die beiden Kommandos „*lpar rmprocs*“ (*remove processors*) und „*lpar rmprocunits*“ (*remove processing units*). Auch beim Wegnehmen von Prozessor Ressourcen, muß man darauf achten, das das Verhältnis zwischen virtuellen Prozessoren und Processing-Units im erlaubten Bereich bleibt.

Wie beim Wegnehmen von dedizierten Prozessoren beschrieben, kann auch ein Wegnehmen von virtuellen Prozessoren fehlschlagen, wenn es Prozesse gibt, die eine feste Bindung zu einem virtuellen Prozessor haben.

5.3. Prozessoren, Prozessor-Cores und Simultaneous Multi-Threading

Prozessoren haben mittlerweile eine lange Entwicklung hinter sich. War zu Beginn auf einem Prozessor-Chip genau eine datenverarbeitende Logik-Einheit, häufig als CPU bezeichnet, hat sich dies mittlerweile grundlegend geändert. Die meisten Prozessor-Chips beheimaten mittlerweile mehrere datenverarbeitende Logik-Einheiten. Man spricht dann von Prozessor-Kernen (oder Cores). Die Bezeichnung Prozessor wurde früher häufig als Synonym für die datenverarbeitende Logik-Einheit oder CPU verwendet, obwohl es sich bei einem Prozessor zunächst einmal um ein elektronisches Bauteil handelt. Mittlerweile unterscheidet man begrifflich den Prozessor als elektronisches Bauteil von den datenverarbeitenden Logik-Einheiten (Prozessor-Kernen), von denen gleich mehrere auf einem Prozessor untergebracht sein können.

In vielen Kontexten wird aber häufig weiterhin von Prozessoren gesprochen, obwohl eigentlich die datenverarbeitende Logik-Einheit und damit der Prozessor-Kern gemeint ist. Der dedizierte Prozessor in den vorausgegangenen Kapiteln ist als streng genommen in der Regel gar kein physikalischer Prozessor, sondern nur einer der Prozessor-Kerne auf einem heutigen Prozessor.

Ursprünglich konnte auf einer CPU (heute Prozessor-Kern) zu einem Zeitpunkt nur eine Instruktions-Sequence (Thread) zu einem Zeitpunkt ablaufen. Mittlerweile unterstützen aber viele Prozessor-Kerne die parallele

Abarbeitung mehrere Instruktions-Sequenzen (Threads). Dies wird als Simultaneous Multi-Threading oder kurz *SMT* bezeichnet. Unter AIX als Betriebssystem werden diese als logische Prozessoren bezeichnet, damit können auf einem Prozessor-Kern teilweise bis zu 8 verschiedene Threads zeitgleich abgearbeitet werden.

5.3.1. POWER-Prozessoren

In diesem Kapitel sollen exemplarisch einige der POWER-Prozessoren kurz erwähnt werden, um die Entwicklungsstationen der POWER-Architektur aufzuzeigen. Es ist allerdings nicht Ziel eine vollständige Beschreibung aller POWER Generationen zu liefern. Im Internet findet man zu dem Thema jede Menge an guter Literatur.

Bis einschließlich POWER3 Prozessor besaßen alle POWER-Prozessoren nur eine datenverarbeitende Logik-Einheit (aus heutiger Sicht 1 Prozessor-Kern). Mit dem POWER4 Prozessor erschien 2001 der erste POWER-Prozessor mit 2 Prozessor-Kernen. Mit der Einführung der POWER5 Prozessoren 2004 stand erstmalig SMT zur Verfügung. Die Anzahl der Prozessor-Kerne auf einem Prozessor wurde mit den nächsten Generationen von POWER-Prozessoren immer weiter gesteigert auf bis zu 24 Prozessor-Kernen bei POWER9 Prozessoren. Gleichzeitig wurde die Anzahl der SMT-Threads von ursprünglich 2 bei POWER4 Prozessoren auf bis zu 8 SMT-Threads bei POWER8 und POWER9 gesteigert.

Die neue POWER10 Generation wird bis zu 30 Prozessor-Kerne unterstützen. Erste Systeme sollen in 2021 ausgeliefert werden.

5.3.2. Simultaneous Multi-Threading

Heutige Prozessor-Kerne haben eine Vielzahl von Rechenwerken, welche parallel genutzt werden können, unter Anderem:

- 4 ALU Einheiten (Arithmetic-Logic-Unit)
- 4 FP Einheiten (Floating-Point)
- 8 Fetch Einheiten (Laden von Instruktionen)
- 6 Decode Einheiten (Dekodieren von Instruktionen)
- …

Es ist nicht möglich all diese Einheiten mit nur einem Thread auszulasten. Ein Thread kann immer nur einige wenige dieser Einheiten zu einem Zeitpunkt verwenden. Um eine höhere Auslastung der verfügbaren Recheneinheiten zu erreichen, müssen mehrere Threads parallel ausgeführt werden. Mit neuen Generationen von POWER-Prozessoren wurde die Anzahl der Recheneinheiten immer weiter erhöht, um letztlich SMT8 (Ausführung von bis zu 8 Threads parallel) zu ermöglichen. Die Prozessor-Kerne können dabei jederzeit die Anzahle der SMT-Threads variieren. Ist nur ein Thread verfügbar, kann kurzfristig von SMT8 auf die Ausführung von nur einem Thread gewechselt werden.

Unter dem Betriebssystem AIX kann die SMT-Einstellung mit dem Kommando *smtctl* abgefragt werden:

```
aix05 # smtctl

This system is SMT capable.
This system supports up to 8 SMT threads per processor.
SMT is currently enabled.
SMT boot mode is not set.
SMT threads are bound to the same physical processor.

proc0 has 8 SMT threads.
Bind processor 0 is bound with proc0
Bind processor 1 is bound with proc0
Bind processor 2 is bound with proc0
Bind processor 3 is bound with proc0
Bind processor 4 is bound with proc0
Bind processor 5 is bound with proc0
Bind processor 6 is bound with proc0
Bind processor 7 is bound with proc0


proc8 has 8 SMT threads.
Bind processor 8 is bound with proc8
Bind processor 9 is bound with proc8
Bind processor 10 is bound with proc8
Bind processor 11 is bound with proc8
Bind processor 12 is bound with proc8
Bind processor 13 is bound with proc8
Bind processor 14 is bound with proc8
Bind processor 15 is bound with proc8

aix05 #
```

Die angezeigten Prozessoren 0 bis 15 sind sogenannte logische Prozessoren. Jeweils 8 dieser logischen Prozessoren gehören zu einem Prozessor-Core. Die logischen Prozessoren sind die SMT-Threads. Soll die Anzahl der SMT-Threads pro Core auf 4 reduziert werden, geht dies ebenfalls mit dem Kommando *smtctl* und der Option „*-t*“:

```
aix05 # smtctl -t 4
smtctl: SMT is now enabled. It will persist across reboots if
        you run the bosboot command before the next reboot.
aix05 #
```

Damit auch nach dem nächsten Reboot *SMT4* verwendet wird, muß einmal das Kommando *bosboot* gestartet werden, ansonsten gilt nach dem nächsten Reboot wieder die ursprüngliche Konfiguration mit *SMT8*!

5.4. Shared Dedicated Kapazität

Bei Shared Prozessor LPARs werden nicht genutzte Prozessor Anteile des garantierten Entitlement zurück an den Shared Prozessor Pool (Hypervisor) gegeben. Dieser kann diese zusätzlich verfügbaren Prozessor Anteile dann anderen Shared Prozessor LPARs zuordnen, wenn diese einen Bedarf an zusätzlichen Prozessor Anteilen haben. Es wird also keine Prozessor Kapazität verschwendet.

Bei LPARs mit dedizierten Prozessoren werden die Prozessoren standardmäßig exklusiv zugeordnet, nicht genutzte Prozessor Anteile werden dann einfach nicht genutzt. Wie die Graphik in Bild 5.10 zeigt, können diese nicht genutzten Prozessor Anteile einen erheblichen Umfang haben.

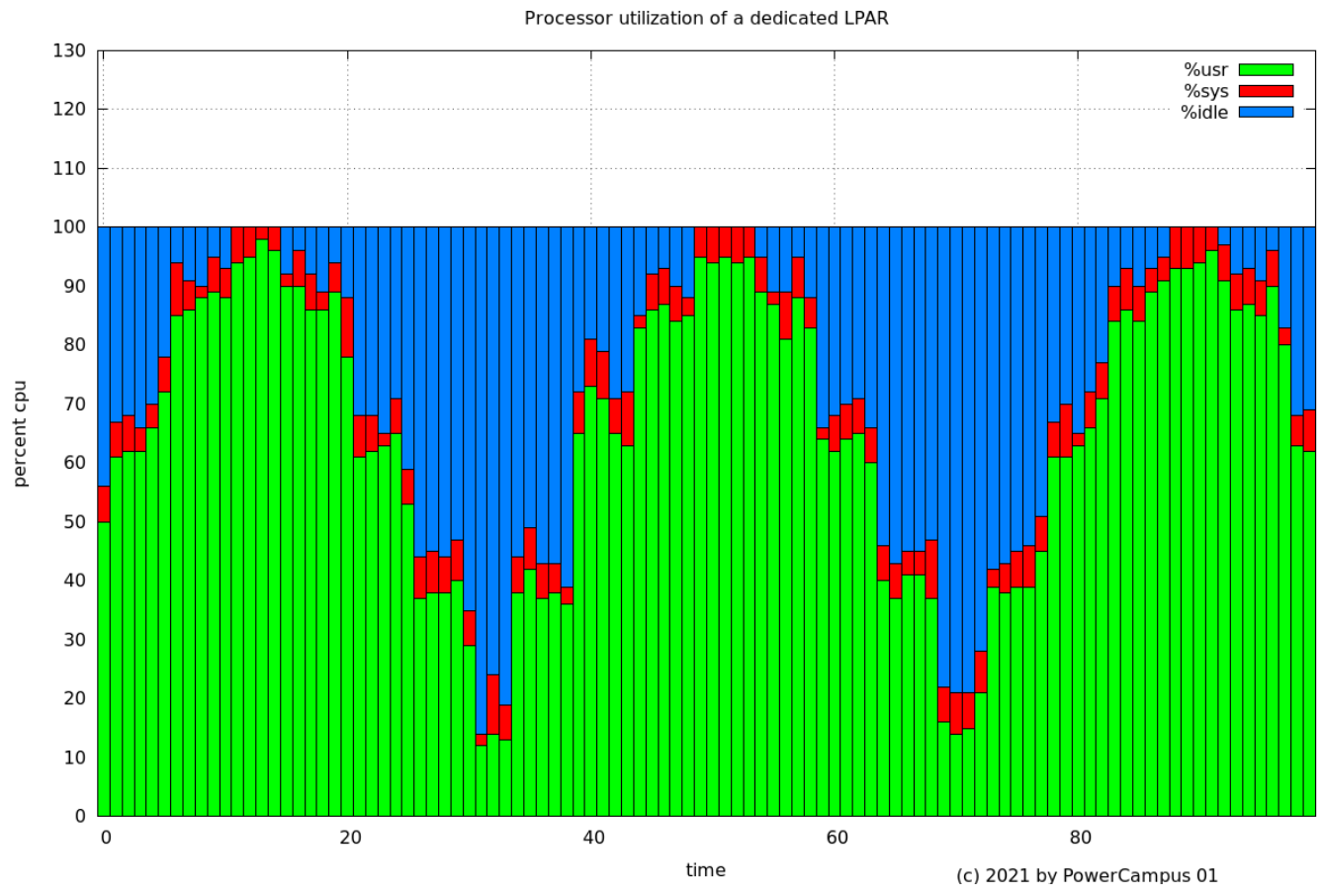


Bild 5.10: Prozessor Auslastung einer LPAR mit *dedicated* Prozessoren.

Die Graphik zeigt die CPU-Auslastung einer dedizierten LPAR. Dabei sind in grün (*%usr*) und rot (*%sys*) die Auslastungen gezeigt, bei denen die LPAR die CPUs auch verwendet, entweder im User-Modus oder im Kernel-Modus. In blau ist der Prozentsatz gezeigt, zu dem die CPUs nichts zu tun haben (*%idle*). Im Durchschnitt ist dies für die gezeigte Zeitspanne ca *28%*. D.h. *28%* der verfügbaren Prozessor-Kapazität der dedizierten Prozessoren der LPAR verrichten keine Arbeit.

Damit nicht genutzte Prozessor-Kapazitäten von dedizierten LPARs durch Shared Prozessor LPARs genutzt werden können, kann auch für LPARs mit dedizierten Prozessoren (*proc_mode=ded*) der Sharing-Mode konfiguriert werden. Das Attribut *sharing_mode* kann für dedizierte LPARs einen der folgenden 4 Werte haben:

keep_idle_procs - Nicht genutzte Prozessor Anteil werden nicht zur Verfügung gestellt. Die LPAR behält Prozessoren die „*idle*“ sind.

share_idle_procs - Nicht genutzte Prozessor Anteile werden nur zur Verfügung gestellt, wenn die dedizierte LPAR inaktiv ist. Wenn die dedizierte LPAR aktiv ist, werden keine Anteile zur Verfügung gestellt.

share_idle_procs_active - Nicht genutzte Prozessor Anteile werden zur Verfügung gestellt, wenn die dedizierte LPAR aktiv ist. Ist die dedizierte LPAR inaktiv, werden keine Anteile zur Verfügung gestellt!

share_idle_procs_always - Nicht genutzte Prozessor Anteile werden immer zur Verfügung gestellt, unabhängig davon ob die dedizierte LPAR aktiv oder inaktiv ist.

Der Sharing-Mode kann jederzeit mit dem Kommando „*lpar chproc*“ geändert werden:

```
$ lpar chproc aix02 sharing_mode=share_idle_procs_always
$
```

Hinweis: Teilt eine dedizierte LPAR nicht genutzte Anteile, kann dies negative Auswirkungen auf die Performance der LPAR haben. Sie bekommt zwar jederzeit alle Prozessor-Anteile, wenn sie diese braucht und muß nicht mit anderen LPARs um diese Prozessor-Anteile konkurrieren. Aber andere LPARs verwenden natürlich die verschiedenen Caches und z.B. den *TLB* (Translation Lookaside Buffer). Damit wird die dedizierte LPAR mehr Cache-Misses produzieren, was eine Auswirkung auf die Performance haben kann.

5.5. Multiple Shared Prozessor Pools

Eine Shared Prozessor LPAR mit Sharing-Mode *uncap* bekommt garantiert immer das konfigurierte Entitlement (*desired_proc_units*). Darüber hinaus kann eine solche LPAR, bei Verfügbarkeit, zusätzliche Prozessor-Anteile gemäß ihrer Gewichtung (*uncapped_weight*) bekommen. Es ist damit sicher gestellt das die LPAR bei Bedarf als Minimum ihr Entitlement zugewiesen bekommt, meistens aber auch mehr. Eine Begrenzung nach oben, also wieviele Prozessor-Anteile eine LPAR maximal bekommen kann, gibt es nur indirekt. Die Anzahl der virtuellen Prozessoren begrenzt zwar die Menge an Prozessor-Anteilen nach oben, die Begrenzung kann aber sehr hoch ausfallen, wenn eine LPAR z.B. mit einem Entitlement von *0.8* und *16* virtuellen Prozessoren konfiguriert wurde. Die LPAR kann in diesem Fall unter günstigen Umständen bis zu *16* komplette physikalische Prozessor-Cores in Beschlag nehmen (natürlich nur wenn entsprechend viele Prozessor-Anteile zur Verfügung stehen). Gibt es wenig Konkurrenz auf einem Managed System, kann eine LPAR auch mit einem niedrigen Wert bei der Gewichtung einen beträchtlichen Anteil an Prozessor-Anteilen bekommen, wie das Beispiel in Bild 5.11 zeigt.

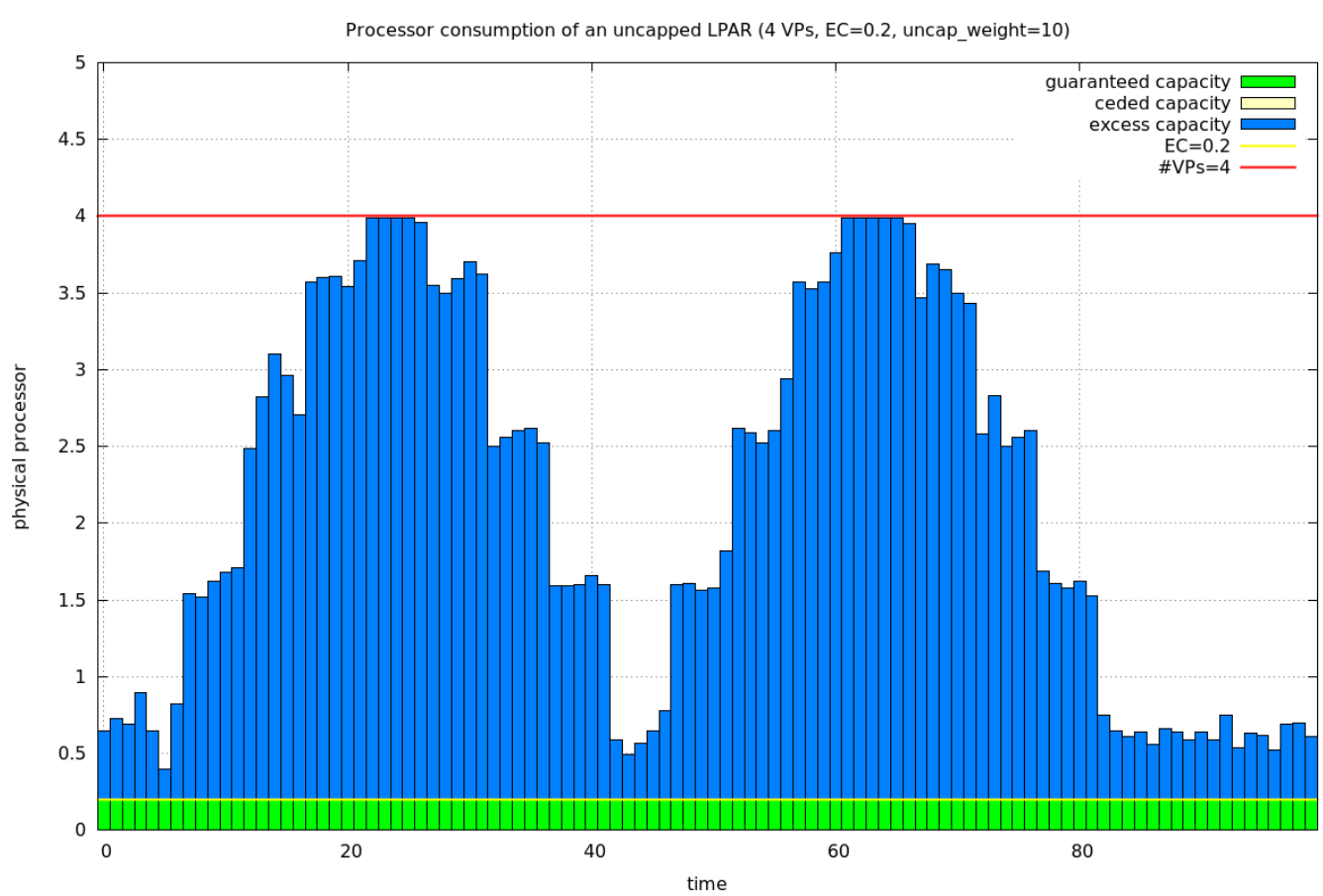


Bild 5.11: Massive Prozessor-Auslastung einer *uncapped* LPAR.

Obwohl die LPAR nur ein Entitlement von *0.2* besitzt und einen sehr niedrigen Wert für die Gewichtung (*uncap_weight=10*) belegt sie in der Spitze bis zu *4* komplette Prozessor-Cores! Mehr ist bei einer Konfiguration mit 4 virtuellen Prozessoren nicht möglich.

Natürlich kann die LPAR nur soviel Prozessor-Anteile bekommen, da genügend Prozessor-Ressourcen verfügbar stehen und offensichtlich alle LPARs mit einer höheren Gewichtung ebenfalls große Mengen an Prozessor-Anteilen bekommen haben. Insofern könnte man sich auf den Standpunkt stellen das es in diesem Falle gut ist eine solche Menge an Prozessor-Ressourcen zuzuteilen, da diese ja ansonsten nicht genutzt würden. Allerdings sollte man bedenken das die Performance der dargestellten LPAR extrem unterschiedlich ausfallen kann. Zu Zeiten wo keine zusätzlichen Prozessor-Anteile verfügbar sind, bekommt die LPAR ihr garantiertes Entitlement von *0.2* (grüner Bereich im Bild). Zu anderen Zeiten, wenn sehr viele Prozessor-Ressourcen verfügbar sind, verwendet sie bis zu *4.0* physikalische Prozessor-Cores. Das ist *20* mal mehr als das garantierte Entitlement. Die Performance der LPAR kann dann auch auf das *20* fache ansteigen. Das heißt die Performance der LPAR ist extrem variabel, mal ist die Performance niedrig (nur das Entitlement steht zur Verfügung), mal ist die Performance *20* fach höher (*4* komplette physikalische Prozessor-Cores stehen zur Verfügung). Dieses Verhalten ist in vielen Fällen nicht gewünscht.

Mit der Einführung von Multiple Shared Prozessor Pools (MSPP) gibt es die Möglichkeit die Verwendung von Prozessor-Anteilen für LPARs zu begrenzen. Hierfür können bis zu 64 Shared Prozessor Pools konfiguriert werden, wobei jedem Pool eine maximale Pool-Kapazität zugewiesen werden kann. Den Prozessor Pools können dann Shared Prozessor LPARs zugewiesen werden, welche dann in der Summe maximal die konfigurierte maximale Pool-Kapazität verbrauchen können. Dies ist aber nur eine der Möglichkeiten die der Einsatz von Multiple Shared Prozessor Pools bietet.

5.5.1. Physical Shared Prozessor Pool

Jedes Managed System besitzt einen sogenannten Physical Shared Prozessor Pool, siehe Bild 5.12 unten. Dabei besteht der Physical Shared Prozessor Pool aus allen Prozessoren (Cores), die nicht als dedizierte Prozessoren (Cores) verwendet werden. Inaktive *CoD*-Prozessoren oder dekonfigurierte Prozessoren gehören ebenfalls nicht zum Physical Shared Prozessor Pool.

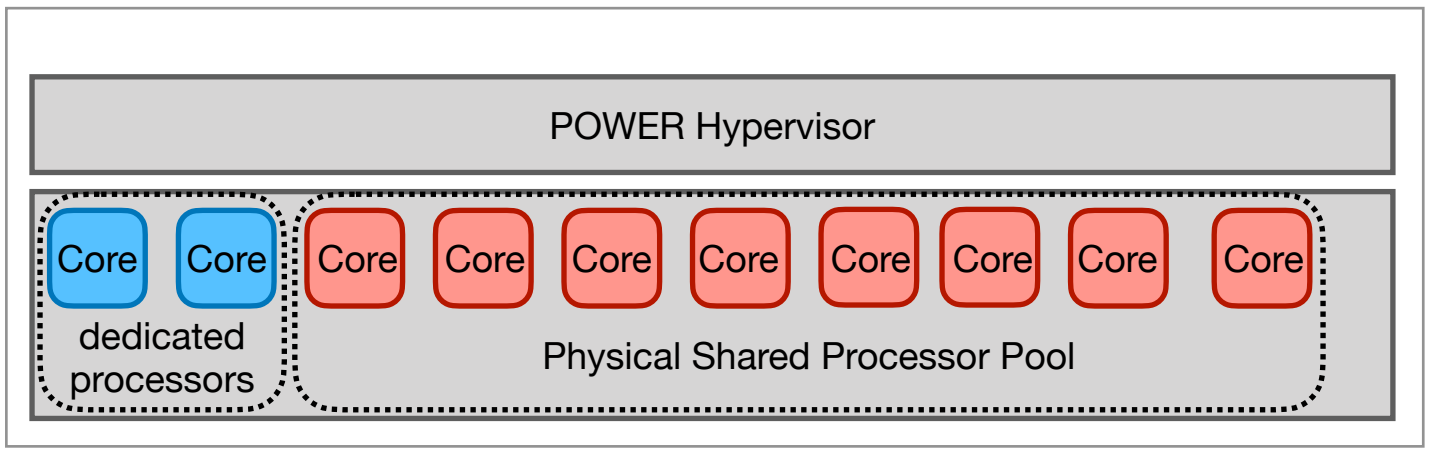


Bild 5.12: Dedizierte Prozessoren und Physical Shared Prozessor Pool

Alle Shared Prozessor LPARs beziehen ihre Prozessor-Anteile aus dem Physical Shared Prozessor Pool. Die Anzahl der Prozessoren (Cores) in diesem Pool ist die maximale Pool-Kapazität.

5.5.2. Multiple Shared Prozessor Pools

Auf jedem Managed System gibt es 64 Shared Prozessor Pools. Jeder der 64 Shared Prozessor Pools hat eine eindeutige Pool-ID (*0* bis *63*) und einen Namen. Der Pool mit der Pool-ID *0* ist der sogenannte *DefaultPool*. LPARs werden per Default diesem Pool zugeordnet, wenn kein expliziter Pool zugeordnet wurde. Neben dem Default-Pool gibt es 63 weitere Shared Prozessor Pools, welche die Namen *SharedPool01*, *SharedPool02* usw haben. Es können keine weiteren Shared Prozessor Pools angelegt werden, und es kann auch keiner der 64 Shared Prozessor Pools gelöscht werden! Shared Prozessor Pools können lediglich umkonfiguriert werden. Neben dem Namen und der eindeutigen Pool-ID besitzt jeder Shared Prozessor Pool 2 konfigurierbare Attribute:

- *max_pool_proc_units* - Mit diesem Attribut wird die sogenannte maximale Pool-Kapazität (Maximum Pool Capacity - *MPC*) festgelegt. Diese gibt an, wieviele Processing-Units maximal während eines *10ms* Intervalls an den Shared Prozessor Pool (und damit die zugeordneten LPARs) vergeben werden. Der Wert muß ganzzahlig sein.

- *reserved_pool_proc_units* - Mit diesem Attribut kann ein zusätzliches Entitlement für einen Shared Prozessor Pool reserviert werden, welches dann von den zugeordneten LPARs zusätzlich zu ihrem Entitlement verbraucht werden kann. Details hierzu folgen später.

Die aktuell benutzten Shared Prozessor Pools auf einem Managed System können mit „*ms lsprocpool*“ (*list processor pools*) angezeigt werden:

```
$ ms lsprocpool ms11
MS_NAME   PROCPOOL     ID  EC_LPARS  RESERVED  PENDING  ENTITLED  MAX
ms11      DefaultPool  0   1.20      -         -        1.20      -
$
```

Standardmäßig werden neben dem default Pool nur Shared Prozessor Pools mit einer maximalen Pool-Kapazität größer 0. Nur diesen können LPARs zugewiesen werden. Sollen alle Shared Prozessor Pools angezeigt werden, kann die Option „*-a*“ (*all*) verwendet werden:

```
$ ms lsprocpool -a ms11
MS_NAME   PROCPOOL      ID  EC_LPARS  RESERVED  PENDING  ENTITLED  MAX
ms11      DefaultPool   0   1.20      -         -        1.20      -
ms11      SharedPool01  1   0.00      0.00      0.00     0.00      0.00
ms11      SharedPool02  2   0.00      0.00      0.00     0.00      0.00
ms11      SharedPool03  3   0.00      0.00      0.00     0.00      0.00
ms11      SharedPool04  4   0.00      0.00      0.00     0.00      0.00
…
ms11      SharedPool62  62  0.00      0.00      0.00     0.00      0.00
ms11      SharedPool63  63  0.00      0.00      0.00     0.00      0.00
$
```

Shared Prozessor Pools sind keine Aufteilung der Prozessoren im Physical Shared Prozessor Pool. Shared Prozessor Pools sollten eher als eine Gruppierung von LPARs gesehen werden. Jede Gruppe von LPARs (Shared Prozessor Pool) hat eine maximale Prozessor-Kapazität (Maximum Pool Capacity), welche festlegt wieviele Prozessor-Anteile alle LPARs der Gruppe zusammen maximal bekommen können.

Eine weitere Eigenschaft von Shared Prozessor Pools ist die Aufteilung der garantierten Prozessor-Anteile der LPARs in einem Pool. Von einer LPAR nicht benötigte Anteile des garantierten Entitlement werden zuerst auf LPARs im gleichen Shared Prozessor Pool aufgeteilt. Nur wenn im gleichen Shared Prozessor Pool diese Anteil nicht benötigt

werden, gehen diese zurück an den Hypervisor und können auf alle LPARs (auch in anderen Shared Prozessor Pools) verteilt werden.

5.5.3. Konfigurieren eines Shared Prozessor Pools (Maximale Pool-Kapazität)

Damit ein Shared Prozessor Pool verwendet werden kann, muß er zunächst konfiguriert werden. Dabei muß mindestens die maximale Pool-Kapazität (*max_pool_proc_units*) auf einen ganzzahligen Wert größer Null gesetzt werden. Ein Shared Prozessor Pool lässt sich mit dem Kommando „*ms chprocpool*“ (*change processor pool*) konfigurieren:

```
$ ms chprocpool ms11 SharedPool01 max_pool_proc_units=2
$
```

(Anstelle des Pool-Namens kann auch die Pool-ID angegeben werden, das ist häufig kürzer.)

Der generische Pool-Name *SharedPool01* ist nicht sehr aussagekräftig und sollte daher geändert werden. Am Besten sollte ein Name gewählt werden, der den Zweck des Pools beschreibt. Wir benennen den Pool daher in *testpool* um:

```
$ ms chprocpool ms11 SharedPool01 new_name=testpool
$
```

Durch die Angabe der maximalen Pool-Kapazität werden keine Prozessoren oder Prozessor-Anteile reserviert! Der Wert von *max_pool_proc_units* kann ohne weiteres größer sein als die Anzahl der zur Verfügung stehenden Prozessoren (maximal möglich ist der Wert *256*), allerdings verliert das Attribut dann seinen Sinn, da alle LPARs zusammen höchstens die Anzahl der verfügbaren Prozessoren komplett auslasten können, aber nicht mehr. Die Begrenzung durch die konfigurierte maximale Pool-Kapazität kann dann in der Praxis nie erreicht werden, der Shared Prozessor Pool wäre damit praktisch unbeschränkt. Daher sollte *max_pool_proc_units* immer kleiner sein als die Anzahl der zur Verfügung stehenden Prozessoren.

Der Default Shared Prozessor Pool *DefaultPool* ist dabei besonders zu betrachten. Die Attribute *max_pool_proc_units* und *reserved_pool_proc_units* können für diesen nicht gesetzt werden! Auch der Name *DefaultPool* kann nicht geändert werden.

5.5.4. Zuweisen eines Shared Prozessor Pools

Um die Begrenzung der Prozessor-Auslastung durch einen Shared Prozessor Pool zu demonstrieren haben wir auf 2 LPARs ein kleines Benchmark-Programm parallel gestartet, siehe Bild 5.13. Beide LPARs sind mit *4* virtuellen Prozessoren, einem Entitlement von *0.4* und einer Gewichtung von *100* konfiguriert:

```
$ lpar lsproc lpar1 lpar2
             PROC          PROCS            PROC_UNITS                          UNCAP   PROC
LPAR_NAME  MODE    MIN  DESIRED  MAX  MIN  DESIRED  MAX  CURR_SHARING_MODE WEIGHT  POOL
```

```
lpar1       shared  1     4         8     0.1   0.4       2.0  uncap              100  DefaultPool
lpar2       shared  1     4         8     0.1   0.4       2.0  uncap              100  DefaultPool
$
```

Auf beiden LPARs wurde in etwa zur gleichen Zeit das gleiche Benchmark-Programm gestartet. Der Graphik in Bild 5.13 kann man entnehmen das beide LPARs zusammen bis zu 8 komplette Prozessoren (Cores) auslasten.

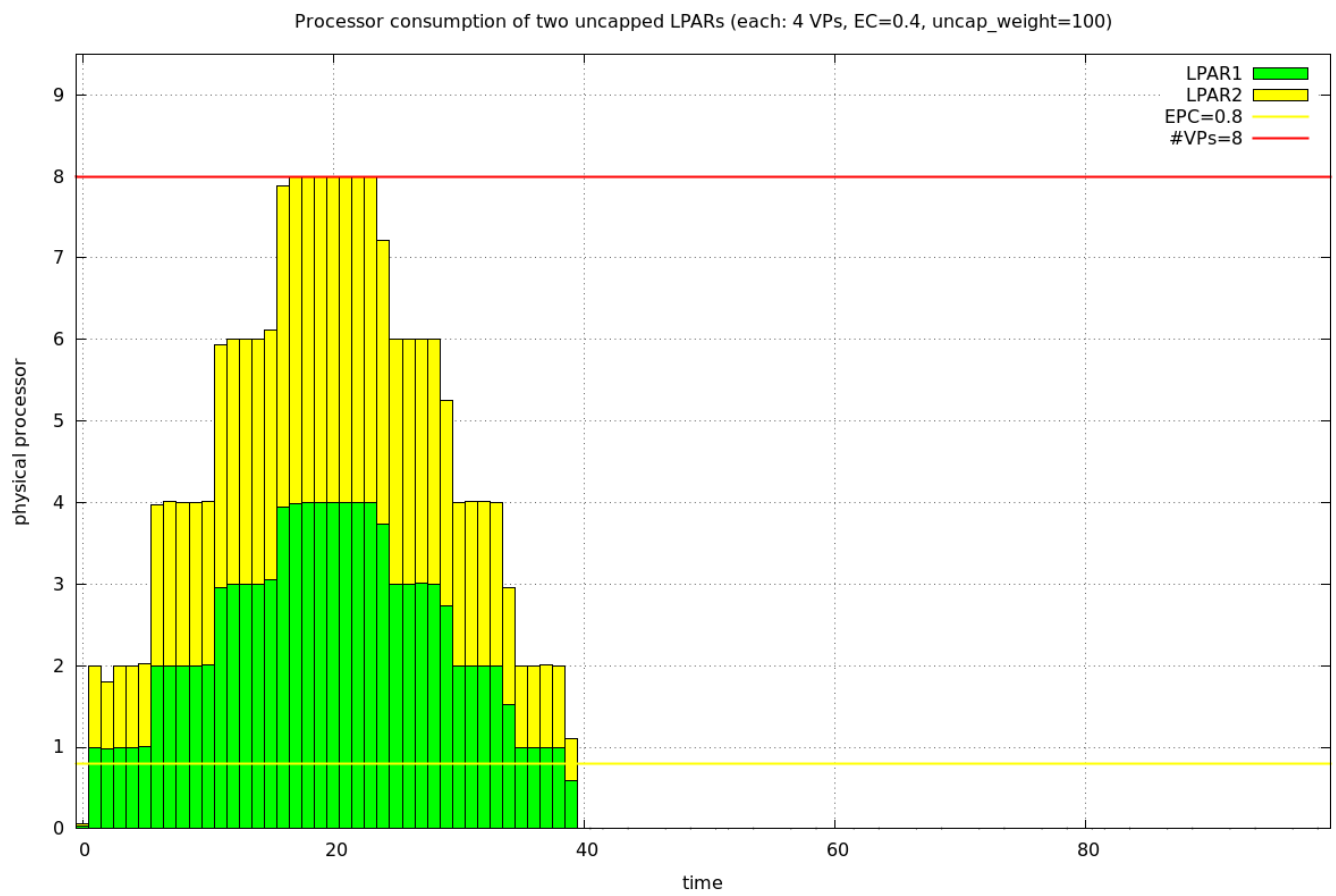


Bild 5.13: Prozessor-Auslastung durch 2 LPARs ohne Shared Prozessor Pool

Durch Verwendung eines Shared Prozessor Pools soll die Prozessor-Auslastung der beiden LPARs auf maximal *2* Prozessoren (Cores) beschränkt werden. Hierzu konfigurieren wir den bisher nicht benutzten Pool *SharedPool02* mit einer maximalen Pool-Kapazität von 2 Prozessoren (Cores) und benennen den Pool gleichzeitig um in *benchmark*:

```
$ ms chprocpool ms11 SharedPool02 new_name=benchmark max_pool_proc_units=2
$
```

Insgesamt haben wir damit die folgenden Shared Prozessor Pools:

```
$ ms lsprocpool ms11
MS_NAME   PROCPOOL     ID  EC_LPARS  RESERVED  PENDING  ENTITLED  MAX
ms11      DefaultPool  0   1.20      -         -        1.20      -
ms11      testpool     1   0.00      0.00      0.00     0.00      2.00
ms11      benchmark    2   0.00      0.00      0.00     0.00      2.00
$
```

Als nächstes weisen wir den beiden LPARs von oben den Shared Prozessor Pool *benchmark* zu. Das Kommando hierfür ist „*lpar chprocpool*“ (*change processor pool*):

```
$ lpar chprocpool lpar1 benchmark
$ lpar chprocpool lpar2 benchmark
$
```

Bei dem Kommando „*ms lsprocpool*“ können durch Verwendung der Option „*-l*“ (*list LPARs*) die einem Pool zugewiesenen LPARs angezeigt werden:

```
$ ms lsprocpool -l ms11
                PROCPOOL            LPAR                  PROCPOOL
MS_NAME   NAME          ID   NAME    ID   EC    RESERVED   ENTITLED   MAX
ms11      DefaultPool   0    lpar3   5    0.40  -          -          -
ms11      DefaultPool   0    -       -    0.40  -          0.40       -
ms11      testpool      1    -       -    0.00  0.00       0.00       2.00
ms11      benchmark     2    lpar1   3    0.40  -          -          -
ms11      benchmark     2    lpar2   4    0.40  -          -          -
ms11      benchmark     2    -       -    0.80  0.00       0.80       2.00
$
```

In der Ausgabe für den Shared Prozessor Pool *benchmark* kann man sehen das es 2 zugeordnete LPARs gibt, *lpar1* und *lpar2*, und das diese jeweils ein Entitlement von *0.4*0 besitzen, was in der Summe ein Entitlement von *0.80* macht. Der Shared Prozessor Pool hat eine maximale Pool-Kapazität von *2.00* (Spalte *MAX*).

Wir wiederholen den Benchmark-Lauf auf beiden LPARs. Die resultierende Prozessor-Auslastung ist in Bild 5.14 gezeigt:

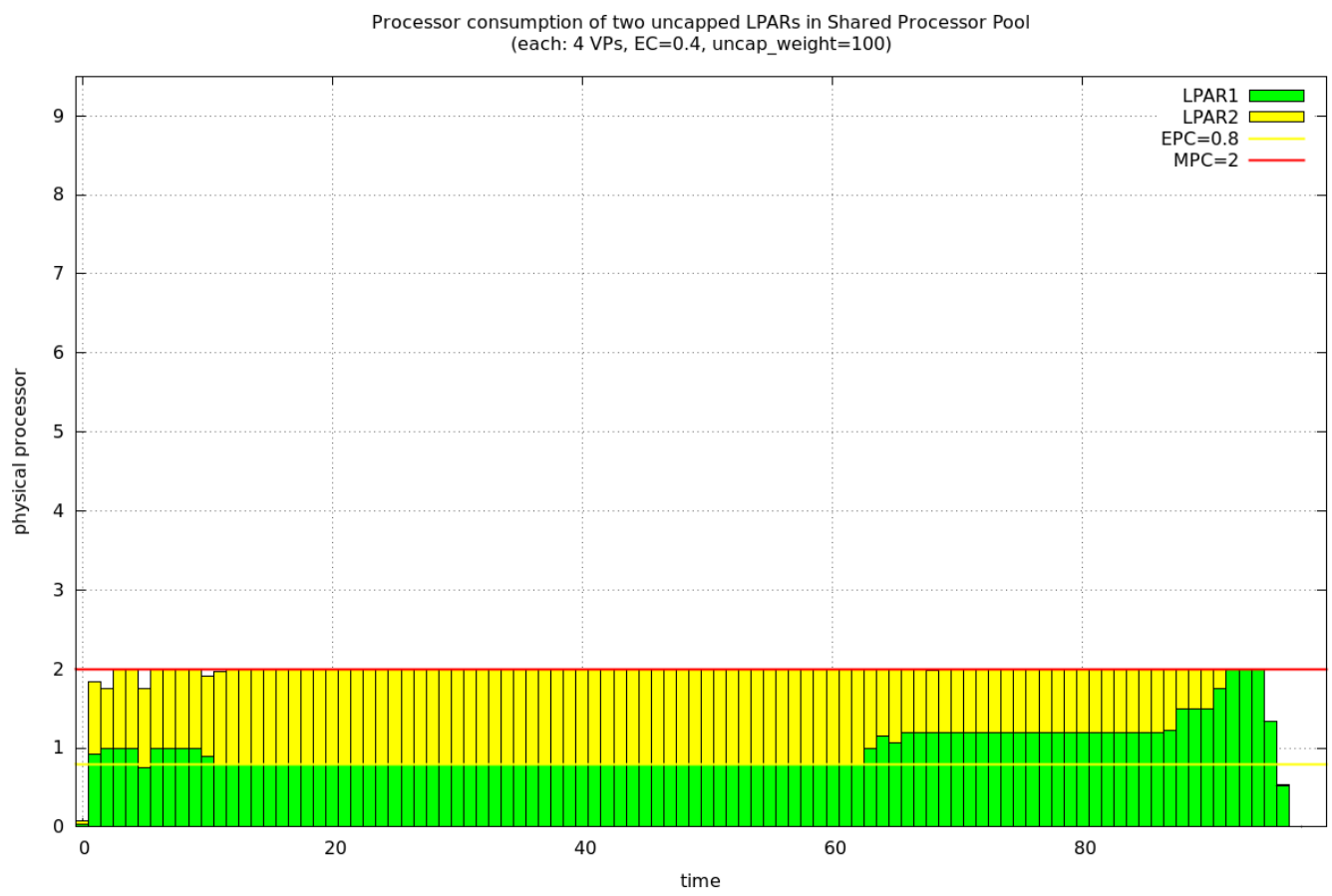


Bild 5.14: Prozessor-Auslastung durch 2 LPARs mit Shared Prozessor Pool

Man sieht deutlich das die maximale Pool-Kapazität (rote Linie) nicht überschritten wird. Die beiden LPARs bekommen jeweils im Durchschnitt ca *1.0* Processing-Units. Deutlich zu sehen ist auch das die Laufzeit des Benchmarks jetzt deutlich zugenommen hat.

Hinweis: Die Zuordnung einer LPAR zu einem Shared Prozessor Pool läßt sich auch ohne aktive RMC-Verbindung der LPAR zu den HMCs ändern.

5.5.5. Entitled Pool-Kapazität (EPC)

Eine wichtige Änderung bei Verwendung von Shared Prozessor Pools betrifft die Verteilung ungenutzter Prozessor-Anteile der LPARs. Ohne Shared Prozessor Pools werden ungenutzte Prozessor-Anteile an alle *uncapped* LPARs gemäß ihrer Gewichtung aufgeteilt. Sobald Shared Prozessor Pools verwendet werden, erfolgt die Verteilung zweistufig. Ungenutzte Prozessor-Anteile werden zuerst auf *uncapped* LPARs im gleichen Shared Prozessor Pool verteilt. Nur die ungenutzten Prozessor-Anteile, die von keiner anderen LPAR im gleichen Shared Prozessor Pool benötigt werden, werden auf LPARs in anderen Shared Prozessor Pools aufgeteilt.

Jeder Shared Prozessor Pool besitzt eine sogenannte Entitled Pool-Kapazität (*Entitled Pool Capacity EPC*). Diese setzt sich zusammen aus der Summe der garantierten Entitlements der zugewiesenen LPARs und der reservierten Pool-Kapazität (*Reserved Pool Capacity RPC*). Die reservierte Pool-Kapazität kann über das Attribut *reserved_pool_proc_units* des Shared Prozessor Pools konfiguriert werden und hat per Default den Wert *0*. So wie bei einer Shared Prozessor LPAR das Entitlement garantiert ist, ist für einen Shared Prozessor Pool die Zuweisung der Entitled Pool-Kapazität garantiert, unabhängig davon, wie diese dann auf die zugehörigen LPARs im Shared Prozessor Pool aufgeteilt wird. In Bild 5.15 sind *Reserved*, *Entitled* und *Maximum* Pool-Kapazitäten für einen Shared Prozessor Pool gezeigt.

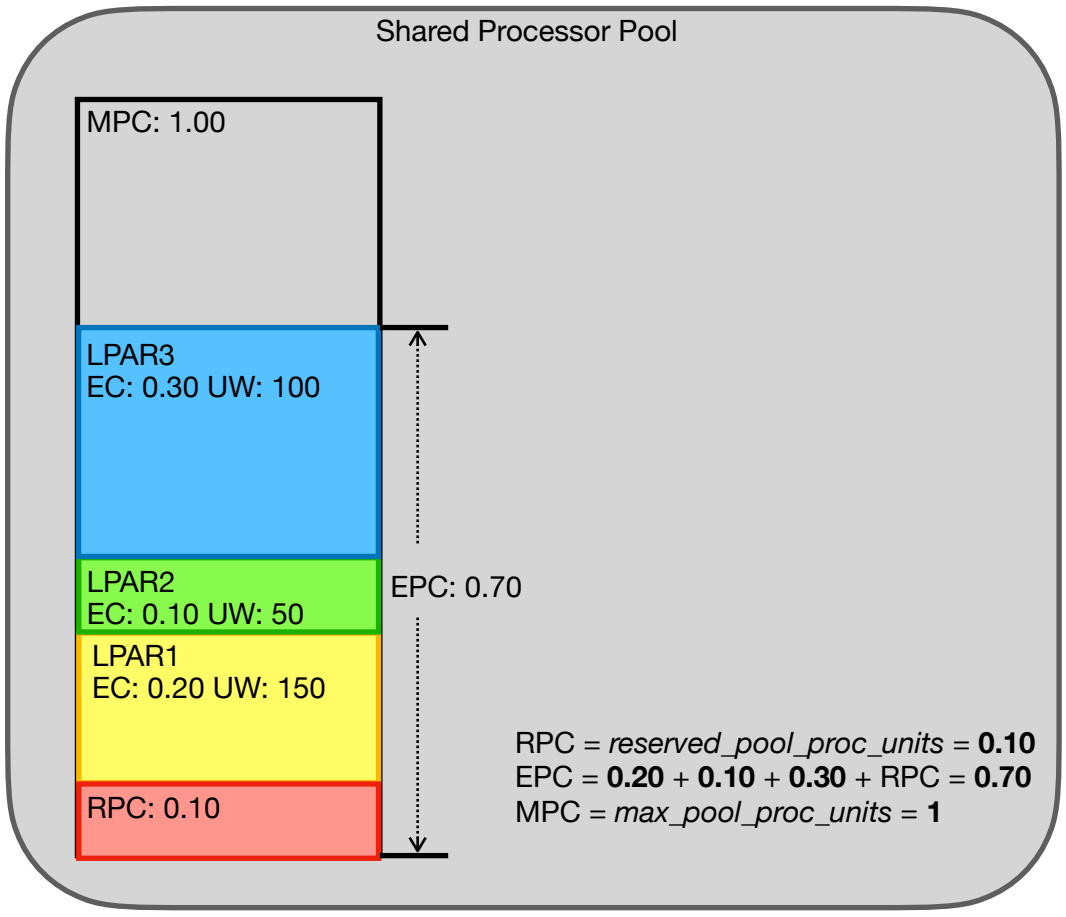


Bild 5.15: *RPC*, *EPC* und *MPC* für einen Shared Prozessor Pool

Dabei muß für die Pool-Kapazitäten immer folgende Bedingung erfüllt sein:

```
Reserved Pool Capacity <= Entitled Pool Capacity <= Maximum Pool Capacity
```

Die Pool-Kapazitäten werden in der Ausgabe von „*ms lsprocpool*“ immer mit angezeigt:

```
$ ms lsprocpool ms06
```

```
MS_NAME   PROCPOOL       ID  EC_LPARS  RESERVED  PENDING  ENTITLED  MAX
ms06      DefaultPool    0  7.90      -         -        7.90      -
ms06      SharedPool01   1  0.60      0.10      0.10     0.70      1.00
$
```

In der Spalte *EC_LPARS* sind die garantierten Entitlements der zugewiesenen LPARs aufaddiert, hier *0.60* für den Pool *SharedPool01*, in der Spalte *RESERVED* findet sich die reservierte Pool-Kapazität (*0.10* für *SharedPool01*), in der Spalte *ENTITLED* dann die Entitled Pool-Kapazität und schließlich in der Spalte *MAX* die maximale Pool-Kapazität. (Der *SharedPool01* ist der Shared Prozessor Pool aus Bild 5.15.)

Wie die Aufteilung von Prozessor-Anteilen in Anwesenheit von mehreren Shared Prozessor Pools funktioniert, ist in Bild 5.16 gezeigt.

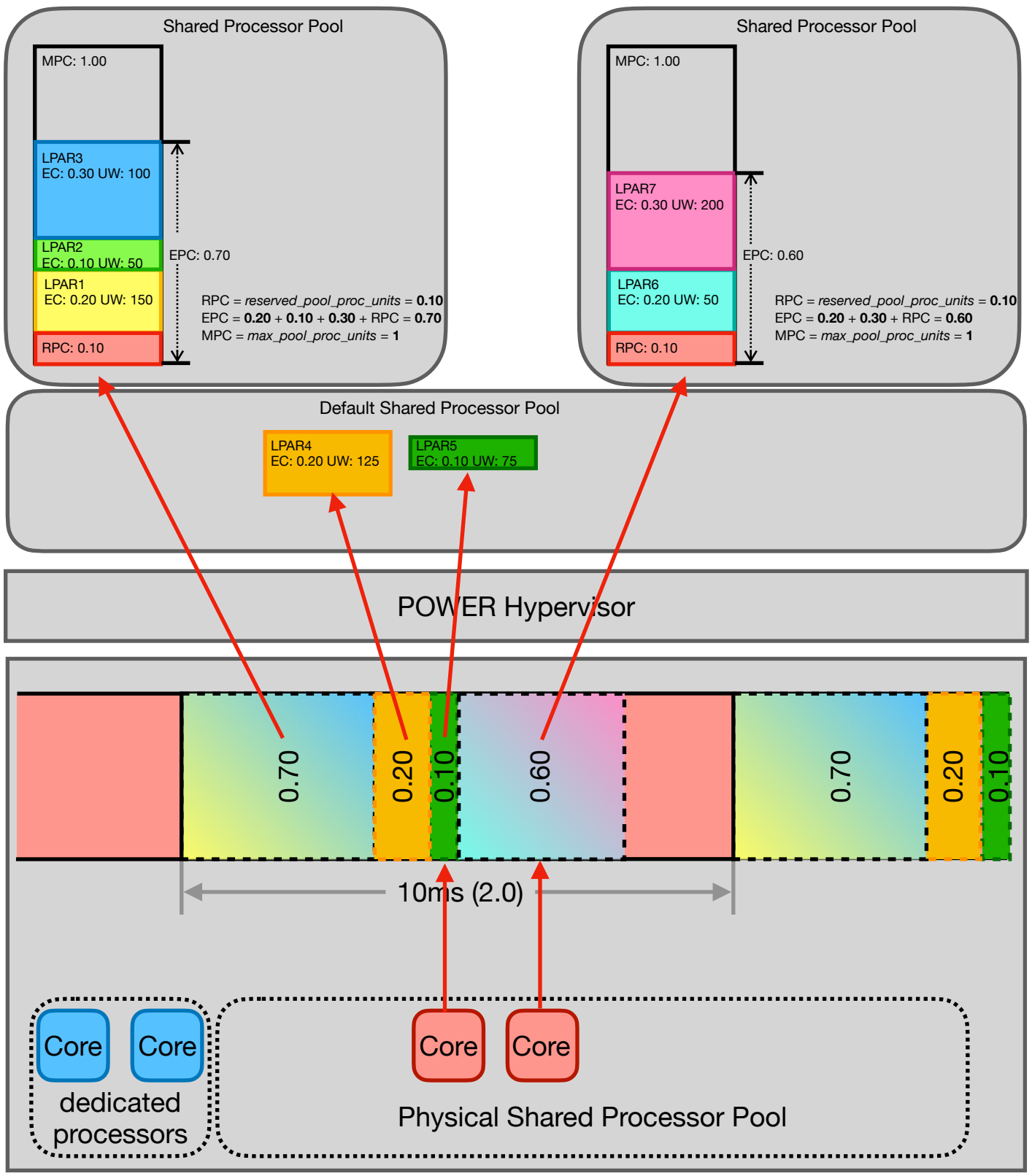


Bild 5.16: Verteilung von Prozessor-Anteilen an Shared Prozessor Pools und LPARs im Default Shared Prozessor Pool gemäß *EPC* bzw. *EC*.

Jeder Shared Prozessor Pool bekommt einen Anteil an den Prozessoren (Cores) gemäß seiner Entitled Pool-Kapazität. Shared Prozessor LPARs im Default Shared Prozessor Pool bekommen Prozessor-Anteile gemäß ihrem Entitlement. Die nicht zugewiesenen Prozessor-Anteile werden auf alle LPARs, unabhängig von Shared Prozessor Pools, gemäß ihrer Gewichtung aufgeteilt (das ist in der Graphik nicht gezeigt).

Die jedem Shared Prozessor Pool zugewiesenen Prozessor-Anteile (gemäß Entitled Pool-Kapazität) werden dann innerhalb des Shared Prozessor Pools auf die zugehörigen LPARs gemäß ihrem Entitlement aufgeteilt. D.h. insbesondere das auch jede LPAR in einem Shared Prozessor Pool weiterhin ihr garantiertes Entitlement bekommt!

Verbraucht eine LPAR in einem Shared Prozessor Pool ihr Entitlement nicht, dann werden diese ungenutzten Prozessor-Anteile zunächst innerhalb des Shared Prozessor Pools an andere LPARs verteilt, welche einen Bedarf an zusätzlichen Prozessor-Anteilen haben. Die Verteilung erfolgt dann wie gehabt unter Berücksichtigung der Gewichtung der LPARs. Ungenutzte Prozessor-Anteile werden also innerhalb eines Shared Prozessor Pools sozusagen „*recycled*“. Sollten auf diesem Wege nicht alle ungenutzten Prozessor-Anteile im Shared Prozessor Pool verbraucht werden, dann werden diese über den Hypervisor an alle (LPARs mit Bedarf an zusätzlichen Prozessor-Anteilen) LPARs aufgeteilt unabhängig vom zugehörigen Shared Prozessor-Pool.

Diese zweistufige Verteilung von Prozessor-Anteilen lässt sich in einem kleinen Versuch sehr gut beobachten. Dazu haben wir bei den 3 LPARs (*lpar1*, *lpar2* und *lpar3*) das garantierte Entitlement auf *0.8* erhöht:

```
$ lpar addprocunits lpar1 0.4
$ lpar addprocunits lpar2 0.4
$ lpar addprocunits lpar3 0.4
$
```

Die Zuordnung zu den Shared Prozessor Pools bleibt weiterhin *lpar1* und *lpar2* sind dem Shared Prozessor Pool *benchmark* zugeordnet und die *lpar3* bleibt in *DefaultPool*:

```
$ lpar -m ms11 lsproc
            PROC          PROCS            PROC_UNITS                        UNCAP   PROC
LPAR_NAME   MODE    MIN   DESIRED   MAX   MIN   DESIRED   MAX   SHARING_MODE   WEIGHT  POOL
lpar1       shared  1     4         8     0.1   0.8       2.0   uncap          100     benchmark
lpar2       shared  1     4         8     0.1   0.8       2.0   uncap          100     benchmark
lpar3       shared  1     4         8     0.1   0.8       2.0   uncap          100     DefaultPool
ms11-vio1   ded     1     7         8     -     -         -     keep_idle_procs    -       -
ms11-vio2   ded     1     6         8     -     -         -     keep_idle_procs    -       -
$
```

Im Shared Prozessor Pool *benchmark* ergibt sich dann die Entitled Pool-Kapazität von *2 * 0.8 + 0.0 = 1.6* (die reservierte Pool-Kapazität ist *0.0*). Die Entitled Pool-Kapazität des Default Shared Prozessor Pool mit nur einer LPAR ist *0.8*.

```
$ ms lsprocpool ms11
MS_NAME   PROCPOOL     ID  EC_LPARS  RESERVED  PENDING  ENTITLED  MAX
ms11      DefaultPool  0   0.80      -         -        0.80      -
ms11      testpool     1   0.00      0.00      0.00     0.00      2.00
ms11      benchmark    2   1.60      0.00      0.00     1.60      2.00
$
```

Wir starten wieder den Benchmark, dieses Mal auf *lpar1* (Shared Prozessor Pool *benchmark*) und *lpar3* (Shared Prozessor Pool *DefaultPool*) parallel. Auf *lpar2* (Shared Prozessor Pool *benchmark*) wird keine Auslastung

produziert, die LPAR liegt während des Benchmarks bei einer Auslastung von ca *0.00 - 0.01*. Damit steht die garantierte Entitled Pool-Kapazität von *1.6* exklusiv für *lpar1* zur Verfügung! Das garantierte Entitlement von *lpar2* im Default Pool ist nur *0.8*. Von den 3 physikalischen Prozessoren (Cores) im Physical Shared Prozessor Pool bleibt damit nur noch ein Entitlement von *3.0 - 1.6 - 0.8 = 0.6*, welches auf LPARs mit zusätzlichem Bedarf an Prozessor-Anteilen verteilt werden kann. Da *lpar1* und *lpar3* beide die gleiche Gewichtung (*uncap_weight=100*) haben, bekommen beide jeweils zusätzlich *0.3* Processing Units. Das macht dann für *lpar1*: *1.6 + 0.3 = 1.9*. Und für *lpar3*: *0.8 + 0.3 = 1.1*. In den Graphiken zur Prozessor-Auslastung (Bild 5.17) ist dies sehr schön zu sehen. Kurze Zeit nach dem Start des Benchmarks auf *lpar1* werden dort ca *1.9* physikalische Prozessoren (Cores) verbraucht, bei *lpar3* sind es ca *1.1*. Aufgrund der größeren Prozessor-Anteile wird der Benchmark auf *lpar1* schneller fertig, womit die Prozessor-Auslastung dort herunter geht. Damit steht aber dann *lpar3* mehr an Prozessor-Anteilen zur Verfügung und es werden von *lpar3* dann am Ende in der Spitze fast die *3* verfügbaren Prozessoren komplett vereinnahmt.

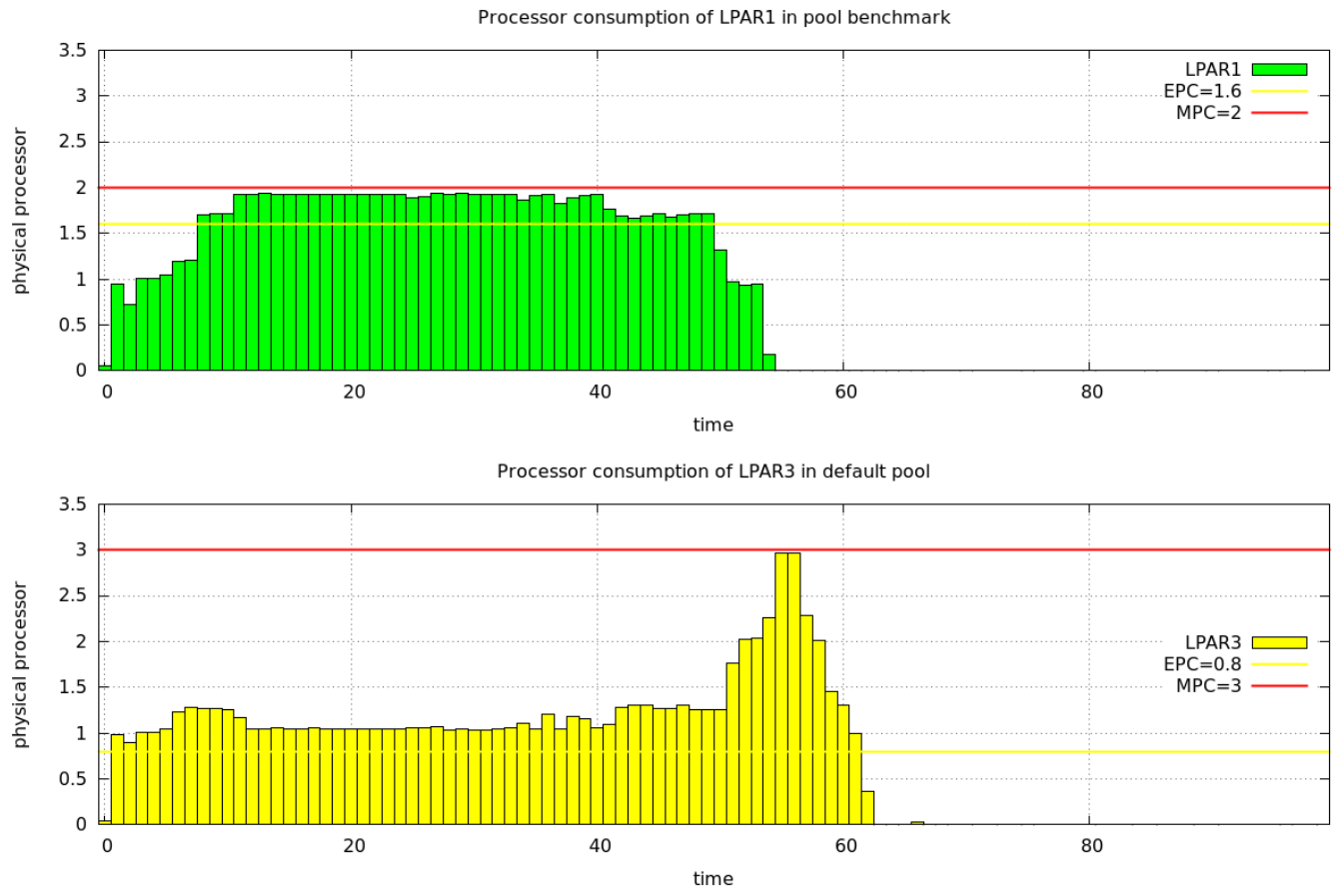


Bild 5.17: Prozessor-Auslastung durch 2 LPARs in verschiedenen Shared Prozessor Pools

Ohne zusätzliche Shared Prozessor Pools profitieren alle *uncapped* LPARs von ungenutzten Prozessor-Anteilen die eine LPAR nicht verbraucht. Da potentiell alle LPARs Teile dieser ungenutzten Prozessor-Anteile bekommen, ist der Anteil für eine individuelle LPAR nicht so groß. Werden zusätzliche Shared Prozessor Pools verwendet, dann profitieren in erster Linie *uncapped* LPARs im gleichen Shared Prozessor Pool von ungenutzten Prozessor-Anteilen einer LPAR. Das sind weniger LPARs und damit ist der Anteil an zusätzlicher Prozessor-Kapazität pro LPAR auch höher.

5.5.6. Reservierte Pool-Kapazität (RPC)

Eine weiteres Merkmal von Shared Prozessor Pools ist die Möglichkeit zusätzliche Prozessor-Anteile zu reservieren. Diese reservierte Pool-Kapazität kann über das Attribut *reserved_pool_proc_units* mit dem Kommando „*ms chprocpool*“ konfiguriert werden:

```
$ ms chprocpool ms11 testpool reserved_pool_proc_units=0.4
$
```

Diese zusätzlichen Prozessor-Anteile stehen den LPARs in einem Shared Prozessor Pool zusätzlichem zu ihrem Entitlement garantiert zur Verfügung. Die Anteile werden gemäß der Gewichtung der LPARs innerhalb des Shared Prozessor Pools aufgeteilt.

5.5.7. Deaktivieren eines Shared Prozessor Pools

Ein Shared Prozessor Pool kann nicht gelöscht werden. Allerdings kann er deaktiviert werden, indem man die maximale Pool-Kapazität zurück auf 0 setzt. Dabei dürfen jedoch keine LPARs dem Shared Prozessor Pool zugewiesen sein:

```
$ ms chprocpool ms11 benchmark max_pool_proc_units=0
hmc01: chhwres -r procpool -m ms11 -o s --poolname benchmark -a 'max_pool_proc_units=0'
ERROR: remote HMC command returned an error (1)
StdErr: HSCL3614 The shared processor pool cannot be unconfigured since at least one
partition is using the shared processor pool.
$
```

Die dem Shared Prozessor Pool zugewiesenen LPARs müssen zunächst einem anderen Shared Prozessor Pool zugewiesen werden.

```
$ lpar chprocpool lpar1 DefaultPool
$ lpar chprocpool lpar2 DefaultPool
$
```

Anschließend kann der Shared Prozessor Pool problemlos deaktiviert werden:

```
$ ms chprocpool ms11 benchmark max_pool_proc_units=0
$
```

Und wird dann standardmäßig auch nicht mehr angezeigt:

```
$ ms lsprocpool ms11
MS_NAME  PROCPOOL     ID  EC_LPARS  RESERVED  PENDING  ENTITLED  MAX
ms11     DefaultPool  0   0.80      -         -        0.80      -
ms11     testpool     1   0.00      0.40      0.40     0.40      2.00
$
```

6. Memory Virtualisierung

Auch im Bereich Speicher Virtualisierung bietet PowerVM verschiedene Möglichkeiten der Konfiguration an. Die am häufigsten genutzte Variante ist die Verwendung von dediziertem Speicher, dabei werden physikalische Speicherbereiche fest einer LPAR zugewiesen. Auf die zugewiesenen Speicherbereiche kann dann auch nur diese LPAR zugreifen. PowerVM unterstützt aber auch die Möglichkeit Speicher gemeinsam zu nutzen. Dabei teilen sich eine Reihe von LPARs den physikalischen Speicher in einem Speicher-Pool. Zu einem Zeitpunkt kann natürlich trotzdem nur eine LPAR einen physikalischen Speicherbereich aus dem Pool verwenden, allerdings wird von einer LPAR ungenutzter Speicher automatisch an den Speicher-Pool zurückgegeben und kann von einer anderen LPAR mit aktuell höherem Hauptspeicherbedarf verwendet werden. Durch das Teilen (*Sharing*) des physikalischen Speichers wird dieser effizienter ausgenutzt, was in der Regel dazu führt, das alle LPARs zusammen insgesamt weniger Hauptspeicher benötigen. Das Teilen von Speicher wird als Active Memory Sharing (*AMS*) bezeichnet.

Eine Möglichkeit physikalischen Hauptspeicher zu sparen, besteht in der Nutzung von Active Memory Expansion (*AME*). Dabei wird einem System eine bestimmte Menge an physikalischem Hauptspeicher zugeordnet, zusammen mit einem Speichererweiterungs-Faktor. Aus Sicht der LPAR steht dann mehr Hauptspeicher zur Verfügung, als physikalisch tatsächlich zugewiesen wurde. Dies wird durch Unterstützung des Betriebssystems erreicht, das den zusätzlichen Speicher durch Komprimierung von aktuell nicht genutzten Speicher-Seiten gewinnt. Eine LPAR kann damit z.B. mit einem zugewiesenen physikalischen Speicher von *32 GB* und einem AME-Faktor von *2.0* auf einen Speicher von *2.0 * 32 GB = 64 GB* zugreifen. Der zusätzliche Speicher, der durch die Komprimierung verfügbar wird, muß aber letztlich durch den Einsatz von mehr Prozessor-Ressourcen für die Komprimierung und Ent-Komprimierung bezahlt werden. Es gibt aber eine Reihe von Einsatzgebieten, wo dies sehr gut funktioniert.

6.1. Dedizierter Speicher

Eine LPAR kann entweder mit dediziertem Speicher oder geteiltem Speicher betrieben werden, aber nicht beides gleichzeitig. Wird dedizierter Speicher verwendet, dann steht dieser ausschließlich einer LPAR zur Verfügung und kann nicht von anderen LPARs verwendet werden.

Beim Zuweisen von dediziertem Speicher werden jeweils physikalische Speicherbereiche zugewiesen. Die Größe eines solchen physikalischen Speicherbereichs muß ein Vielfaches der sogenannten Logical Memory Block (*LMB*) Größe sein. Die *LMB*-Größe wird durch das Managed System vorgegeben und kann mit dem Kommando „*ms lsmem*“ (*list memory resources*) angezeigt werden. Ohne weitere Argumente werden alle vorhandenen Managed Systems aufgelistet:

```
$ ms lsmem
NAME      INSTALLED  FIRMWARE  CONFIGURABLE  AVAIL   MEM_REGION_SIZE
ms01      524288     6656      524288        498176  256
ms03      524288     14848     524288        26112   256
ms04      524288     12544     524288        61184   256
ms05      1048576    27392     1048576       81664   256
ms06      1048576    25600     1048576       0       256
ms07      2097152    43776     2097152       859392  256
ms08      2097152    50176     2097152       740352  256
$
```

Die *LMB*-Größe eines Managed Systems kann in der Spalte *MEM_REGION_SIZE* abgelesen werden, die Angabe ist in *MB*.

In Bild 6.1 ist die Zuordnung des Hauptspeichers einiger LPARs zum physikalischen Speicher des Managed Systems gezeigt. Der Hauptspeicher einer LPAR kann sich ohne weiteres aus mehreren physikalischen Speicherbereichen zusammen setzen!

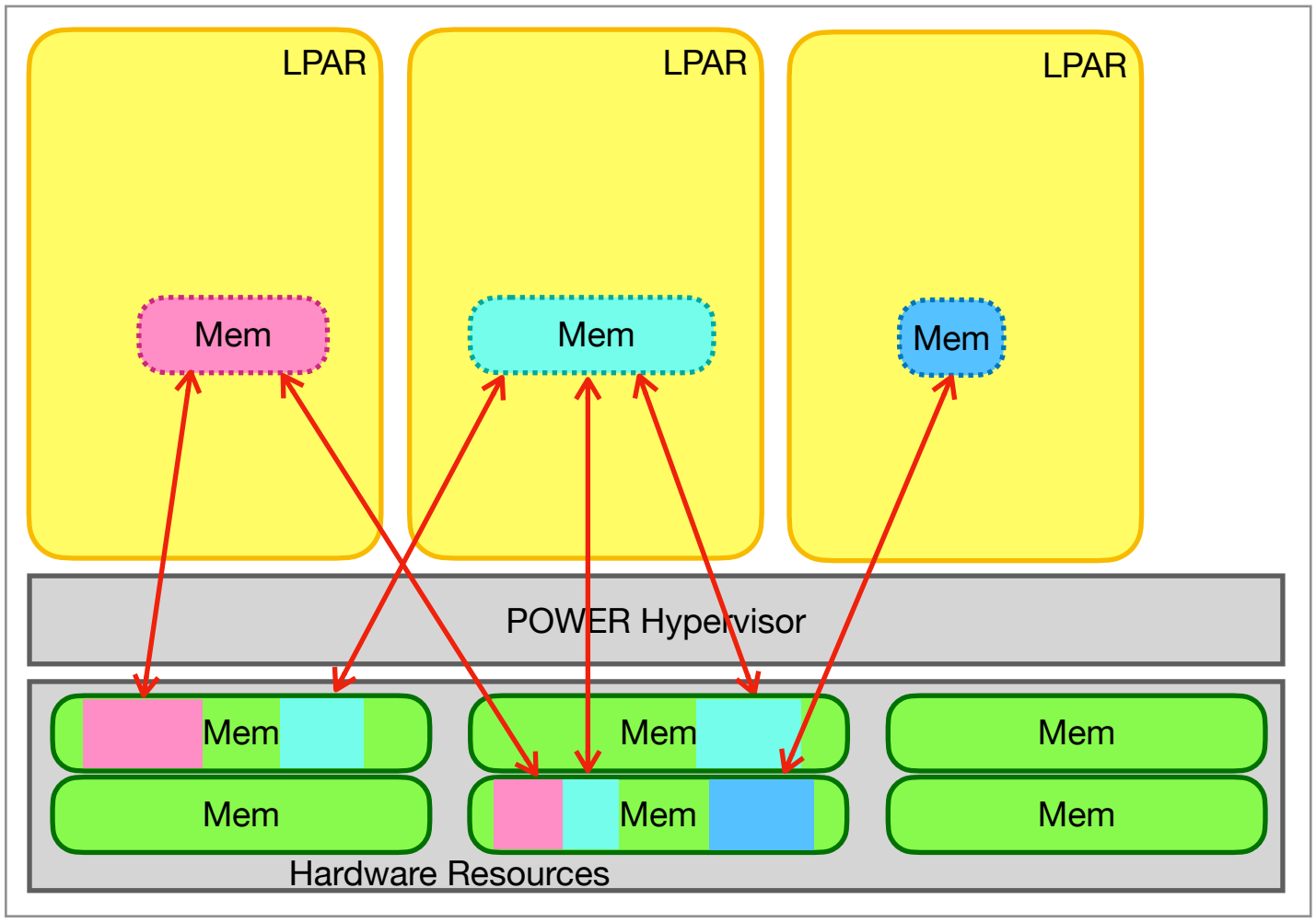


Bild 6.1: Dedizierter Speicher einiger LPARs

Beim Erzeugen einer neuen LPAR kann durch Angabe des Attributs *mem_mode* angegeben werden, ob dedizierter Speicher verwendet werden sollen, oder geteilter Speicher (*AMS*):

```
mem_mode : memory sharing mode
   ded - dedicated memory
   shared - shared memory
```

Eine neue LPAR kann mit dem Kommando „*lpar create*“ erzeugt werden. Standardmäßig werden LPARs mit dediziertem Speicher angelegt:

```
$ lpar -m ms02 create lpar3
.
    > lpar3
$
```

Die Option „*-m*“ mit dem Ziel Managed System, muss zwingend angegeben werden, sie legt fest auf welchem der Managed Systems die LPAR angelegt werden soll. Der LPAR-Name, hier *lpar3*, ist optional. Wird kein Name angegeben, generiert das *LPAR-Tool* einen eindeutigen Namen.

Eine neu erzeugte LPAR ist zunächst nicht aktiviert. Der größere Teil der LPAR-Konfiguration ist in einem Profil abgelegt, das beim Erzeugen der LPAR angelegt wird. Per Default wird der Name *standard* für das Profil verwendet, es lässt sich aber auch ein anderer Default hinterlegen. Ein Blick in das Profil *standard* zeigt das die LPAR mit *1024 MB* dediziertem Speicher (Spalte *MEMORY DESIRED*) konfiguriert wurde:

```
$ lpar -p standard lsmem lpar3
            MEMORY          MEMORY              HUGE_PAGES
LPAR_NAME  MODE  AME  MIN   DESIRED  MAX   MIN   DESIRED  MAX
lpar3      ded   0.0  1024  1024     1024  null  null     null
$
```

Soll eine LPAR mehr als *1024 MB* Speicher bekommen, dann kann die gewünschte Speichergröße über das Attribut *desired_mem* angegeben werden:

```
$ lpar create -m ms02 lpar4 desired_mem=4096
    > lpar4
$
```

Die angegebene Speichergröße muß dabei ein ganzzahliges Vielfaches der *LMB*-Größe sein. Im Profil *standard* der LPAR *lpar4* ist jetzt *4096 MB* als *desired* (gewünscht) hinterlegt:

```
$ lpar -p standard lsmem lpar4
            MEMORY          MEMORY              HUGE_PAGES
LPAR_NAME  MODE  AME  MIN   DESIRED  MAX   MIN   DESIRED  MAX
lpar4      ded   0.0  1024  4096     4096  null  null     null
$
```

Neben dem Attribut *desired_mem* für die gewünschte Speichergröße, gibt es noch 2 weitere Attribute und zwar *min_mem* und *max_mem*. Genauso, wie *desired_mem*, können auch diese Attribute auf der Kommandozeile beim Erzeugen einer LPAR angegeben werden. Der Wert von *min_mem* muss kleiner oder gleich dem Wert *desired_mem* sein, welcher wiederum kleiner oder gleich dem Wert *max_mem* sein muss:

```
min_mem <= desired_mem <= max_mem
```

Der Wert von *min_mem* kommt mindestens in den folgenden beiden Situationen zum tragen:

- Eine LPAR wird aktiviert, es steht aber nicht soviel physikalischer Hauptspeicher wie über *desired_mem* gefordert zur Verfügung. In diesem Fall reduziert PowerVM die Speichergröße, welche der LPAR zugewiesen wird, auf einen kleineren Wert. Allerdings darf dabei der Wert von *min_mem* nicht unterschritten werden.

- Bei einer aktiven LPAR mit laufendem Betriebssystem kann dynamisch Speicher hinzugefügt oder weggenommen werden, ohne das Betriebssystem oder Applikationen stoppen zu müssen. Dabei kann die Hauptspeichergröße maximal auf den Wert von *max_mem* erhöht werden bzw. höchstens auf den Wert von *min_mem* reduziert werden.

Der Wert von *max_mem* wird, wie gerade beschrieben, bei der dynamischen Erhöhung des Hauptspeichers berücksichtigt.

Dies ist analog zu den Attributen *min_procs*, *desired_procs* und *max_procs* bei Prozessoren.

Welche Attribute angegeben werden können und welche möglichen Werte diese Attribute haben, kann in der Online Hilfe nachgeschaut werden:

```
$ lpar help create
USAGE:
   lpar [-h <hmc>] [-m <ms>] [-p <profile>] create [{-b <blueprint>|-s <source_lpar>}] [-v]
[<lpar>] [<attributes> ...]

DESCRIPTION

Create a new LPAR on a managed system.

   -b : blueprint to use for creation'
   -s : source LPAR to use as blueprint'

Valid attributes:
   name : name for the LPAR
   lpar_id : the ID of the LPAR
   profile_name : name of the default profile
   lpar_env : type of LPAR
      aixlinux - AIX or Linux (default)
      os400 - IBM i
      vioserver - virtual I/O server
   min_mem : minimum amount of memory in MB
   desired_mem : desired amount of memory in MB
   max_mem : maximum amount of memory in MB
   mem_expansion : Active Memory Expansion
      0 - disable AME
      1.00-10.00 - expansion factor
…
$
```

6.1.1. Hinzufügen von Speicher

Hat eine LPAR eine aktive RMC-Verbindung zu den HMCs, dann kann im laufenden Betrieb Speicher hinzugefügt (und natürlich auch weggenommen) werden. Dazu müssen 2 Bedingungen erfüllt sein:

1. Die aktuelle Hauptspeichergröße muß kleiner sein, als die maximale erlaubte Hauptspeichergröße (*max_mem*) für die LPAR.

2. Die gewünschte zusätzliche Speichermenge muß auch noch verfügbar sein.

Wieviel Speicher eine LPAR aktuell besitzt und wieviele maximal möglich sind, lässt sich mittels „*lpar lsmem*“ (*list memory*) leicht bestimmen:

```
$ lpar lsmem aix22
            MEMORY          MEMORY          HUGE_PAGES
LPAR_NAME  MODE  AME  MIN   CURR  MAX   MIN  CURR  MAX
aix22      ded   0.0  1024  4096  8192  0    0     0
$
```

Die LPAR *aix22* besitzt aktuell *4096 MB* Hauptspeicher und kann auf bis zu *8192 MB* erweitert werden. Die Hauptspeichergröße ist natürlich auch vom Betriebssystem der LPAR aus feststellbar:

```
aix22 # lsattr -El mem0
```

```
ent_mem_cap          I/O memory entitlement in Kbytes           False
goodsize       4096  Amount of usable physical memory in Mbytes False
mem_exp_factor       Memory expansion factor                    False
size           4096  Total amount of physical memory in Mbytes  False
var_mem_weight       Variable memory capacity weight            False
aix22 #
```

Auf dem unterliegenden Managed System (*ms06*) sind aktuell noch *834.816 MB* (ca *815 GB*) Speicher verfügbar (Spalte *AVAIL*):

```
$ ms lsmem ms06
NAME  INSTALLED  FIRMWARE  CONFIGURABLE  AVAIL   MEM_REGION_SIZE
ms06  1048576    12032     1048576       834816  256
$
```

Damit können ohne weiteres 2 GB zur LPAR *aix22* hinzugefügt werden. Das Kommando hierfür lautet „*lpar addmem*“ (*add memory*):

```
$ lpar addmem aix22 2G
$
```

Die hinzuzufügende Speichermenge kann in KB (*K* oder *KB*), MB (*M* oder *MB*), GB (*G* oder *GB*) oder TB (*T* oder *TB*) angegeben werden. Wird keine Größeneinheit angegeben, wird der Wert als *MB* interpretiert. Der zusätzliche Speicher steht dem Betriebssystem sofort zur Verfügung, es ist keine weitere Konfiguration auf der AIX-LPAR notwendig:

```
aix22 # lsattr -El mem0
ent_mem_cap          I/O memory entitlement in Kbytes           False
goodsize       6144  Amount of usable physical memory in Mbytes False
mem_exp_factor       Memory expansion factor                    False
size           6144  Total amount of physical memory in Mbytes  False
var_mem_weight       Variable memory capacity weight            False
aix22 #
```

Ist eine LPAR ausgeschaltet oder hat keine aktive RMC-Verbindung zu den HMCs, dann kann der Speicher nur im Profil der LPAR geändert werden:

```
$ lpar -p standard addmem aix22 2GB
$
```

Die Änderung wirkt sich dann allerdings erst aus, wenn die LPAR neu aktiviert wird (unter Angabe des geänderten Profils).

6.1.2. Wegnehmen von Speicher

Auch für das Wegnehmen von Speicher im laufenden Betrieb ist eine aktive RMC-Verbindung zu den HMCs erforderlich. Es kann nur Speicher weggenommen werden, wenn die verbleibende Menge an Speicher größer oder gleich der minimalen Hauptspeichermenge (*min_mem*) ist.

Das Wegnehmen von Speicher erfolgt, analog dem Hinzufügen, mit dem entsprechenden Kommando „*lpar rmmem*“ (*remove memory*):

```
$ lpar rmmem aix22 2G
$
```

Voraussetzung hierfür ist allerdings das im Betriebssystem 2 GB entweder nicht in Verwendung sind, oder frei gemacht werden können, indem Speicherseiten auf den Paging-Space ausgelagert werden.

Damit sind anschließend wieder nur noch 4 GB Hauptspeicher in der LPAR verfügbar:

```
aix22 # lsattr -El mem0
ent_mem_cap         I/O memory entitlement in Kbytes           False
goodsize       4096 Amount of usable physical memory in Mbytes False
mem_exp_factor      Memory expansion factor                    False
size           4096 Total amount of physical memory in Mbytes  False
var_mem_weight      Variable memory capacity weight            False
aix22 #
```

6.2. Active Memory Sharing (AMS)

Ähnlich wie im Falle von Shared Prozessoren, besteht auch beim physikalischen Speicher die Möglichkeit diesen mit mehreren LPARs zu teilen. Dabei werden den LPARs zu unterschiedlichen Zeiten unterschiedlich viel physikalischer Speicher zugewiesen, je nach Bedarf. Die Menge des physikalischen Speichers einer LPAR kann dabei im Laufe der Zeit variieren.

In der Regel wird die Hauptspeicher-Größe einer LPAR vom Administrator ausreichend und mit einem mehr oder weniger großen Sicherheitspuffer für Spitzen der Auslastung gewählt. Das sogenannte Working-Set (Speicher-Seiten die tatsächlich verwendet werden), ist in der Regel kleiner. Andernfalls ist erhöhtes Paging die Folge. Bei PowerHA-Systemen muß für den Fall einer Übernahme ausreichend viel Speicher verfügbar sein, ansonsten könnte eine Übernahme im Fehlerfalle scheitern. Dies bedeutet aber, das insbesondere bei Cluster-Systemen der Unterschied zwischen dem aktuell tatsächlich benötigten Speicher und dem konfigurierten Speicher leicht die halbe Hauptspeicher-Größe erreichen kann. Solange keine Übernahme erfolgt, wird der hierfür konfigurierte Speicher nicht verwendet (abgesehen vom Filesystem-Caching).

In Bild 6.2 ist die tatsächliche Speicherauslastung einiger LPARs über den Verlauf eines Tages gezeigt. Die rote Linie zeigt die Summe der konfigurierten Hauptspeicher-Größen der LPARs an (248 GB). Die LPARs bleiben im Verlauf des Tages in Summe allerdings teilweise deutlich unter diesem Wert. Die dargestellten Werte stammen von realen Produktionssystemen. Man sieht zwar hier nicht die in diesem Zusammenhang gern dargestellten Benutzungs-Berge einer LPAR, die auf ein Benutzungs-Tal einer anderen LPAR treffen, es werden aber trotzdem zwischen ca. 60 und 100 GB an zugewiesenem Speicher von den LPARs nicht verwendet. Mit diesem Speicher könnten zum Beispiel weitere LPARs auf dem Managed System betrieben werden, oder es könnte einzelnen LPARs mehr Speicher zugewiesen werden.

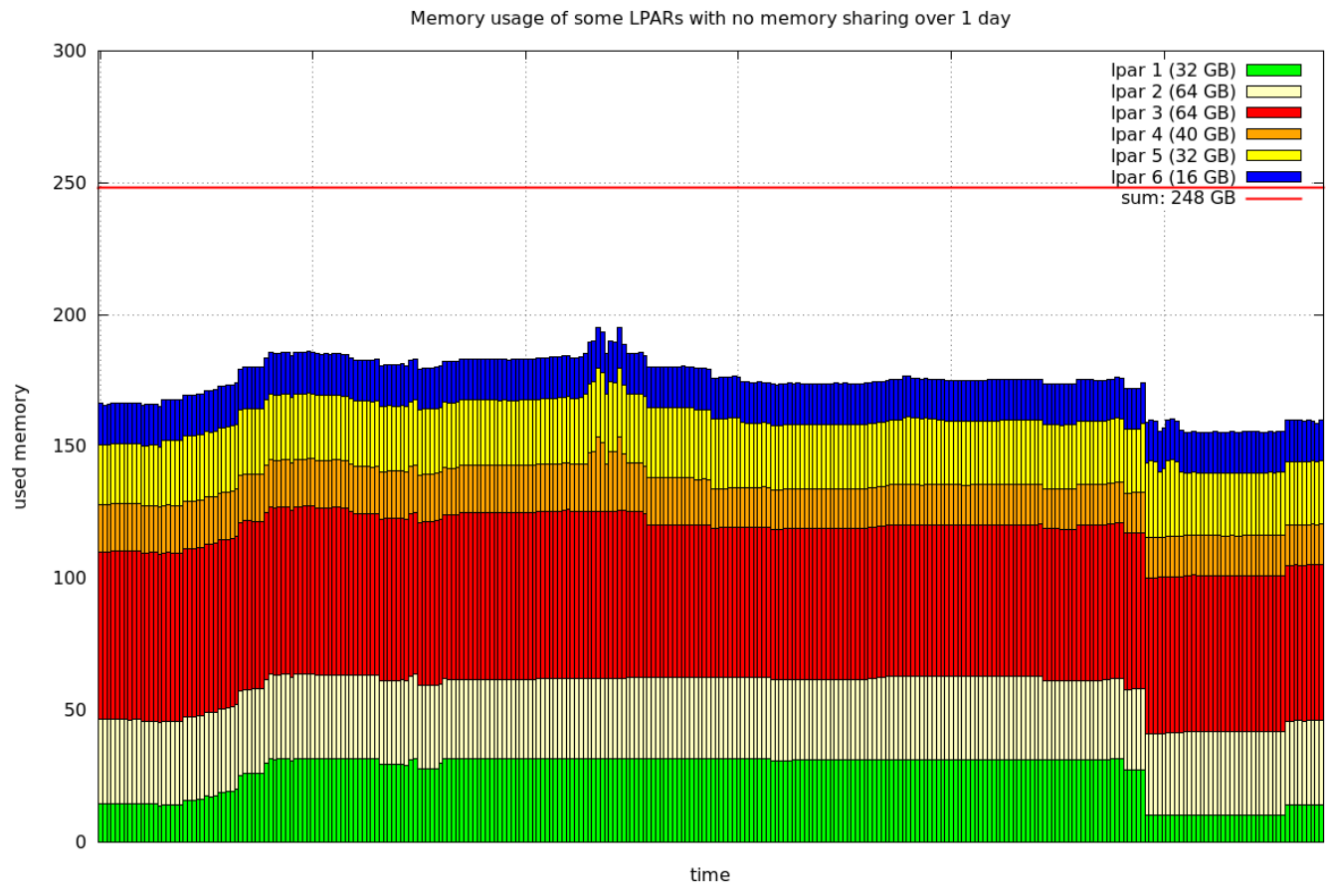


Bild 6.2: Tatsächlicher Speicherverbrauch einiger LPARs während eines Tages.

Analog zu den Shared Prozessoren kann für das Teilen von Speicher ein sogenannter Shared Memory Pool angelegt werden. Im Unterschied zu den Shared Prozessoren, gibt es aber maximal nur einen Shared Memory Pool. Im Prinzip können dann fast beliebig viele LPARs (maximal 1000) dem Shared Memory Pool zugewiesen werden. Die Summe der Hauptspeicher-Größen dieser LPARs darf die Hauptspeicher-Größe des Shared Memory Pools übersteigen. Beispielsweise könnte in der Situation von Bild 6.2 ein Shared Memory Pool der Größe 200 GB angelegt werden und alle dargestellten 6 LPARs mit zusammen 248 GB Speicher-Größe zugewiesen werden. Da nicht ausgeschlossen werden kann, das nicht doch einmal alle LPARs zusammen mehr als 200 GB tatsächlich benutzen, muß der fehlende Speicher von 248 GB - 200 GB = 48 GB natürlich in irgendeiner Form vorgehalten werden. Hier kommen die schon von Betriebssystemen bekannten Paging-Devices in Spiel. Für jede LPAR, die den Shared Memory Pool verwendet, muß ein eigenes Paging-Device (mit einer Kapazität von mindestens der maximalen Speichergröße der LPAR) bereit gestellt werden. Fehlender physikalischer Hauptspeicher wird dann durch Speicher auf den Paging-Devices ersetzt.

In Bild 6.3 ist ein Shared Memory Pool mit 2 Shared Memory LPARs gezeigt. Der Speicher der LPARs setzt sich dabei aus physikalischem Hauptspeicher des Shared Memory Pools und eventuell Speicher von Paging-Devices zusammen. Für die LPARs selber ist nicht feststellbar, ob ein Speicherbereich physikalischen Speicher oder Speicher auf einem Paging-Device verwendet. Die Zuordnung von physikalischem Speicher aus dem Shared Memory Pool auf die einzelnen LPARs erfolgt dynamisch durch den Hypervisor. Der Hypervisor kann nach Auslagern von physikalischen Speicherseiten auf ein Paging-Device den freigewordenen physikalischen Speicher einer anderen LPAR zuordnen. Greift die ursprüngliche LPAR wieder auf die ausgelagerten Speicherseiten zu, ordnet der Hypervisor freie physikalische Speicherseiten zu, startet I/O auf das Paging-Device um die ausgelagerten Daten zu lesen und speichert diese dann auf den neuen physikalischen Speicherseiten ab. Die zugehörige LPAR bekommt von diesen Hintergrund-Aktionen nichts mit. Der Speicherzugriff dauert lediglich etwas länger.

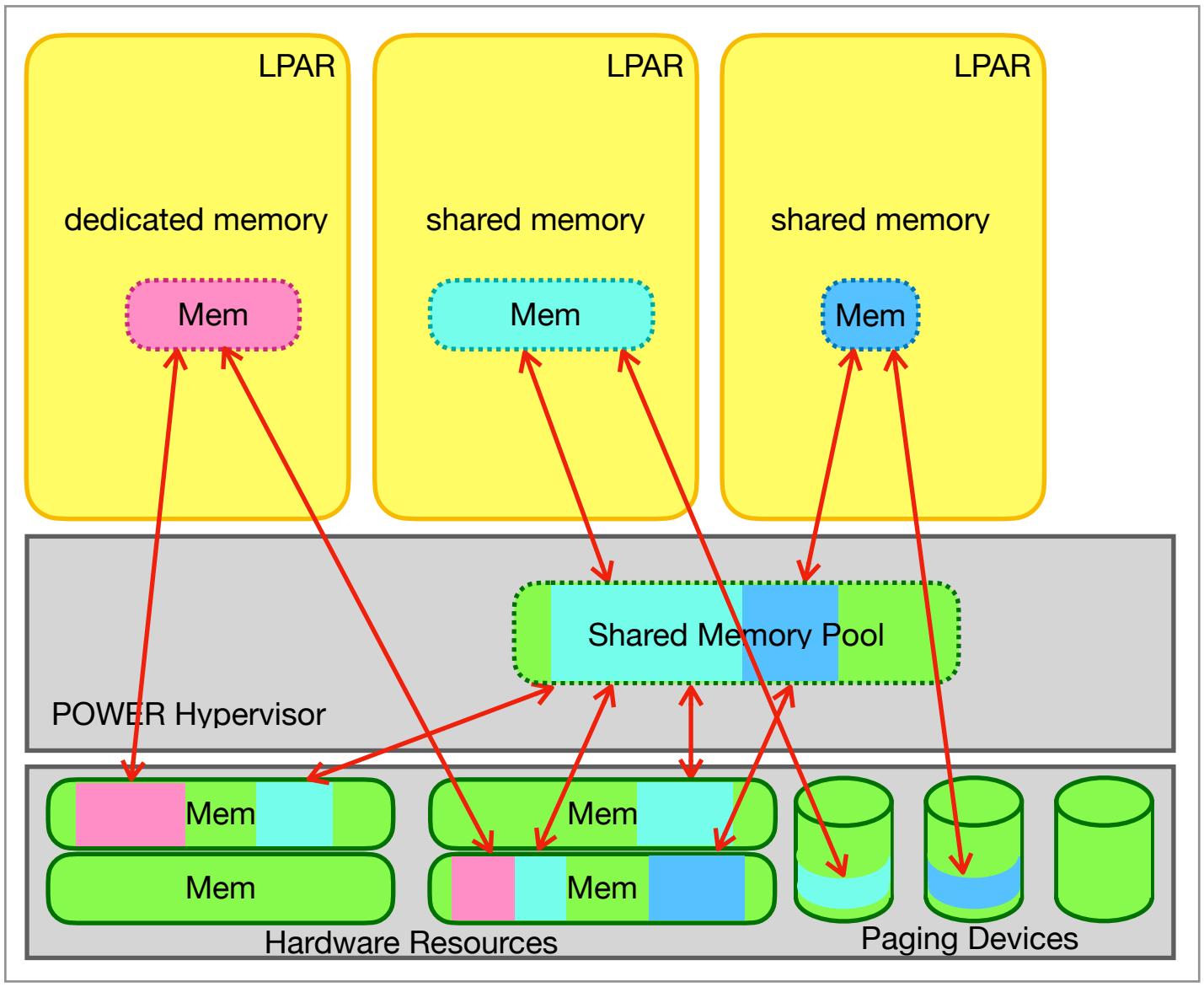


Bild 6.3: Shared Memory Pool mit 2 Shared Memory LPARs.

Da der Hypervisor selbst als Firmware nicht direkt selbst I/O auf die Paging-Devices starten kann, es braucht dazu den Kontext einer LPAR, muß einem Shared Memory Pool mindestens ein Virtual-I/O-Server zugeordnet werden, der das I/O von und zu den Paging-Devices übernimmt. Ein solcher Virtual-I/O-Server wird dabei als Paging Virtual-I/O-Server bezeichnet. Damit müssen die Paging-Devices auf dem Paging Virtual-I/O-Server sichtbar und zugreifbar sein. Der Hypervisor delegiert also das I/O zu den Paging-Devices an einen Paging Virtual-I/O-Server.

Das Teilen von physikalischem Hauptspeicher über einen Shared Memory Pool wird als Active Memory Sharing (AMS) bezeichnet.

6.2.1. Anlegen eines Shared Memory Pools

Bevor eine Shared Memory LPAR angelegt werden kann, muß zunächst erst einmal ein Shared Memory Pool angelegt werden. Dabei muß mindestens die Größe des Shared Memory Pools und ein oder zwei Paging Virtual-I/O-Server angegeben werden. Auf unserem Managed System *ms03* gibt es noch reichlich freien Hauptspeicher, wie die Ausgabe von „*ms lsmem*“ zeigt:

```
$ ms lsmem ms03
NAME  INSTALLED  FIRMWARE  CONFIGURABLE  AVAIL   MEM_REGION_SIZE
ms03  1048576    13568     1048576       790272  256
$
```

Es sind aktuell noch 790.272 MB (ca. 772 GB) freier physikalischer Hauptspeicher vorhanden. Der Shared Memory Pool auf *ms03* kann mit dem Kommando „*ms addmempool*“ (*add memory pool*) angelegt werden:

```
$ ms addmempool ms03 pool_mem=20G paging_vios_names=ms03-vio1
$
```

Mit dem Attribut *pool_mem* wird die Größe des Shared Memory Pools angegeben. Diese wird unmittelbar reserviert und steht damit für Dedicated Memory LPARs nicht mehr zur Verfügung. Das Auflisten des verfügbaren Speichers nach Anlegen des Shared Memory Pools zeigt, das jetzt weniger physikalischer Hauptspeicher verfügbar ist:

```
$ ms lsmem ms03
NAME  INSTALLED  FIRMWARE  CONFIGURABLE  AVAIL   MEM_REGION_SIZE
ms03  1048576    14336     1048576       769024  256
$
```

Hinweis: Nach Anlegen des Shared Memory Pools kann es einige Zeit dauern, bis der physikalische Speicher reserviert bzw. zugeordnet ist!

Mit dem Attribut *paging_vios_names* kann ein oder maximal zwei Paging Virtual-I/O-Server angegeben werden (durch Komma getrennt). Auch wenn wir hier mit nur einem Paging Virtual-I/O-Server *ms03-vio1* starten, sollten in der Praxis, aus Redundanz-Gründen, immer zwei Paging Virtual-I/O-Server angegeben werden. Fällt einer der Virtual-I/O-Server aus, übernimmt der zweite automatisch.

Der aktuelle Shared Memory Pool kann mit dem Kommando „*ms lsmempool*“ (*list memory pool*) angezeigt werden:

```
$ ms lsmempool ms03
                      POOL_MEM                PAGING
MS_NAME  CURR   AVAIL  FIRMWARE  MAX    VIOS_NAMES  MEM_DEDUP
ms03     20480  20224  256       20480  ms03-vio1   0
$
```

6.2.2. Hinzufügen eines Paging Devices

Für jede LPAR die Shared Memory verwenden soll, wird ein eigenes Paging-Device benötigt. Als Paging-Device kann einer der folgenden Gerätetypen verwendet werden:

- Logical Volume
- Lokale Platte
- SAN-Platte
- iSCSI-Platte

Werden zwei redundante Paging Virtual-I/O-Server verwendet, dann müssen die Paging-Devices SAN-Platten sein.

Auf unserem Virtual-I/O-Server *ms03-vio1* gibt es die folgenden Physical Volumes:

```
$ vios lspv ms03-vio1
PVNAME  PVID              VGNAME  PVSTATE
hdisk0  0000000000cafe00  None    -
hdisk1  0000000000cafe01  rootvg  active
hdisk2  0000000000cafe02  rootvg  active
hdisk3  0000000000cafe03  None    -
hdisk4  0000000000cafe04  None    -
$
```

Bei den Physical Volumes *hdisk3* und *hdisk4* handelt es sich jeweils um SAN-LUNs, wie das Kommando „*vios lsdev*“ zeigt:

```
$ vios lsdev ms03-vio1 hdisk3
NAME    STATUS     PHYSLOC                                          PARENT  DESCRIPTION
hdisk3  Available  U78AA.001.VYRGU0Q-P1-C5-T1-W500507680130A1C4-L0  fscsi4  MPIO IBM 2145
FC Disk
$
$ vios lsdev ms03-vio1 hdisk4
NAME    STATUS     PHYSLOC                                          PARENT  DESCRIPTION
hdisk3  Available  U78AA.001.VYRGU0Q-P1-C5-T1-W50050768013098BC-L0  fscsi4  MPIO IBM 2145
FC Disk
$
```

Diese SAN-LUNs sind bisher ungenutzt und sollen jetzt als erste Paging-Devices verwendet werden. Um ein Paging-Device einem Shared Memory Pool hinzuzufügen gibt es 2 verschiedene, aber äquivalente, Kommandos. Entweder kann das Kommando „*ms addpgdev*“ (*add paging device*) verwendet werden:

```
$ ms addpgdev ms03 ms03-vio1 hdisk3
$
```

Hierbei muß als Argument, neben dem Managed System, der Paging Virtual-I/O-Server und das Paging-Device angegeben werden.

Oder es kann das etwas kürzere Kommando „*vios addpgdev*“ verwendet werden:

```
$ vios addpgdev ms03-vio1 hdisk4
$
```

Dabei muß lediglich der Paging Virtual-I/O-Server und das Paging-Device angegeben werden.

Damit besitzt der Shared Memory Pool nun die ersten beiden Paging-Devices, Kommando „*ms lspgdev*“:

```
$ ms lspgdev ms03
                                                                 REDUNDANT
MS_NAME  DEVICE_NAME  PAGING_VIOS_NAME  STATE     DEVICE_NAME  PAGING_VIOS_NAME  STATE  SIZE   TYPE  LPAR_ID
ms03     hdisk3       ms03-vio1         Inactive  -            -                 -      51200  phys  none
ms03     hdisk4       ms03-vio1         Inactive  -            -                 -      51200  phys  none
$
```

6.2.3. Anlegen von LPARs mit Shared Memory

Ob eine LPAR dedizierten Speicher oder geteilten Speicher verwenden soll, kann beim Anlegen einer LPAR über das Attribut *mem_mode* entschieden werden. Dieses kann die folgenden beiden Werte annehmen:

```
mem_mode : memory mode
   ded - dedicated memory
   shared - shared memory
```

Wir legen die LPAR *shlpar1* mit Shared Memory an. Wie bei dediziertem Memory kann über die Attribute *min_mem*, *desired_mem* und *max_mem* vorgegeben werden wieviel Speicher die LPAR mindestens, normalerweise (gewünscht) und maximal haben soll.

```
$ lpar -m ms03 create shlpar1 mem_mode=shared desired_mem=4G max_mem=8G
    > shlpar1
$
```

Die Überprüfung des Profils *standard* zeigt das eine LPAR mit Shared Memory angelegt wurde. Der primäre Paging Virtual-I/O-Server ist *ms03-vio1*:

```
$ lpar -p standard lsmem shlpar1
             MEMORY            MEMORY              HUGE_PAGES           MEM          PAGING_VIOS
LPAR_NAME  MODE    AME  MIN   DESIRED  MAX   MIN   DESIRED  MAX   IO_ENTITLED  PRIMARY    SECONDARY
shlpar1    shared  0.0  1024  4096     8192  null  null     null  auto         ms03-vio1  -
$
```

Durch das Anlegen einer LPAR mit Shared Memory wird allerdings noch kein Paging-Device zugewiesen, wie die Ausgabe von „*ms lspgdev*“ zeigt:

```
$ ms lspgdev ms03
                                                                  REDUNDANT
MS_NAME  DEVICE_NAME  PAGING_VIOS_NAME  STATE     DEVICE_NAME  PAGING_VIOS_NAME  STATE  SIZE   TYPE  LPAR_ID
ms03     hdisk4       ms03-vio1         Inactive  -            -                 -      51200  phys  none
ms03     hdisk3       ms03-vio1         Inactive  -            -                 -      51200  phys  none
$
```

Bei beiden Paging-Devices ist noch keine LPAR zugeordnet (Spalte *LPAR_ID*). Daher aktivieren wir die LPAR als nächstes unter Verwendung des Profils *standard*:

```
$ lpar -p standard activate shlpar1
hmc01: chsysstate -m ms03 -r lpar -o on -n shlpar1 -f standard
ERROR: remote HMC command returned an error (1)
StdErr: HSCLA457 Partition shlpar1(6) cannot be activated because its profile is missing a
necessary virtual I/O adapter. Shared memory partitions are required to have at least one
virtual I/O adapter of any type other than virtual serial.
$
```

Das funktioniert leider so noch nicht. Wir hatten die LPAR *shlpar1* ohne virtuelle Adapter angelegt, allerdings gelten für die Verwendung von Shared Memory die folgenden beiden Bedingungen:

- Eine Shared Memory LPAR darf keine physikalischen I/O-Adapter besitzen.

- Eine Shared Memory LPAR muß mindestens einen virtuellen Adapter (außer den beiden standardmäßigen virtuellen seriellen Adaptern) besitzen.

Wir fügen einen virtuellen Ethernet Adapter in Slot *2* hinzu, um die LPAR aktivieren zu können:

```
$ lpar -p standard addeth shlpar1 2 900
$
```

(Das letzte Argument ist die *PVID* des virtuellen Ethernet Adapters, spielt aber hier keine Rolle, da dieser nicht verwendet wird.)

Ein Aktivieren der LPAR ist jetzt erfolgreich:

```
$ lpar -p standard activate shlpar1
$
```

Nun ist auch eines der beiden Paging-Devices zugewiesen, wie die Ausgabe von „*ms lspgdev*“ zeigt:

```
$ ms lspgdev ms03
                                                                  REDUNDANT
MS_NAME  DEVICE_NAME  PAGING_VIOS_NAME  STATE     DEVICE_NAME  PAGING_VIOS_NAME  STATE  SIZE   TYPE  LPAR_ID
ms03     hdisk4       ms03-vio1         Inactive  -            -                 -      51200  phys  none
ms03     hdisk3       ms03-vio1         Active    -            -                 -      51200  phys  6
$
```

Laut Ausgabe wurde die *hdisk3* als Paging-Device zugeordnet. Die Auswahl wird durch den Hypervisor automatisch durchgeführt. Es wird ein verfügbares Paging-Device ausgewählt, welches mindestens die Kapazität *max_mem* der LPAR besitzt. Damit ist sichergestellt das theoretisch sogar der komplette Speicher der LPAR auf Paging-Device ausgelagert werden könnte. Dies passiert aber nicht, es bleibt immer eine von der LPAR abhängige minimale Hauptspeichermenge zugeordnet.

6.2.4. Ändern einer Dedicated Memory LPAR auf Shared Memory

Schon bestehende LPARs mit dediziertem Memory können ohne weiteres auf Shared Memory umgestellt werden. Eine LPAR muß dazu allerdings heruntergefahren werden. Die Umstellung erfolgt dann im Profil der LPAR. Es muß lediglich das Attribut *mem_mode* von *ded* auf *shared* geändert werden. Anschließend kann die LPAR mit dem geänderten Profil wieder aktiviert werden und verwendet dann sofort Shared Memory.

Das Vorgehen ist anhand der LPAR *aix22* (befindet sich auf dem Managed System *ms03*) kurz gezeigt:

1. Herunterfahren der LPAR:

```
aix22 # shutdown

SHUTDOWN PROGRAM
Mon May  3 11:57:07 CEST 2021


Broadcast message from root@aix22 (tty) at 11:57:07 ...
```

```
PLEASE LOG OFF NOW ! ! !
All processes will be killed in 1 minute.



Broadcast message from root@aix22 (vty0) at 11:58:07 ...

! ! ! SYSTEM BEING BROUGHT DOWN NOW ! ! !

Stopping IBM.ConfigRM...
0513-044 The IBM.ConfigRM Subsystem was requested to stop.
…
Unmounting the file systems...
Unmounting the file systems...

Bringing down network interfaces: en0 lo0



open: No such file or directory

May  3 11:58:35 portmap: terminating on signal.
....Halt completed….
```

Alternativ kann das System auch mittels „*lpar osshutdown*“ heruntergefahren werden:

```
$ lpar osshutdown aix22
$
```

2. Ändern des Memory-Modes im Profil der LPAR (hier *standard*):

```
$ lpar -p standard chmem aix22 mem_mode=shared
$
```

3. Aktivieren der LPAR mit dem geänderten Profil:

```
$ lpar -p standard activate aix22
$
```

4. Überprüfung:

```
$ lpar lsmem aix22
             MEMORY          MEMORY         HUGE_PAGES    IO_ENTITLED_MEM         PAGING_VIOS
LPAR_NAME  MODE    AME  MIN   CURR  MAX   MIN  CURR  MAX  AUTO     CURR    PRIMARY          SECONDARY
aix22      shared  0.0  1024  2048  8192  0    0     0    1        351     ms03-vio1        -
$
```

```
$ ms lspgdev ms03
                                                                    REDUNDANT
MS_NAME  DEVICE_NAME  PAGING_VIOS_NAME  STATE     DEVICE_NAME  PAGING_VIOS_NAME  STATE  SIZE   TYPE  LPAR_ID
ms03     hdisk4       ms03-vio1         Active    -            -                 -      51200  phys  5
ms03     hdisk3       ms03-vio1         Active    -            -                 -      51200  phys  6
$
```

6.2.5. Logischer Speicher und Paging

Im Falle von LPARs mit dediziertem Speicher ist der einer LPAR zugewiesene Speicher physikalischer Speicher. Speicher kann in ganzzahligen Vielfachen der Logical Memory Block (*LMB*) Größe zugewiesen werden. In Bild 6.4 ist eine LPAR mit 1.536 MB dediziertem Speicher gezeigt. Bei einer *LMB* Größe von 256 MB sind das 6 Speicherblöcke. Jedem der 6 konfigurierten 256 MB Speicherblöcke ist ein Block physikalischer Speicher der Größe 256 MB eineindeutig zugeordnet.

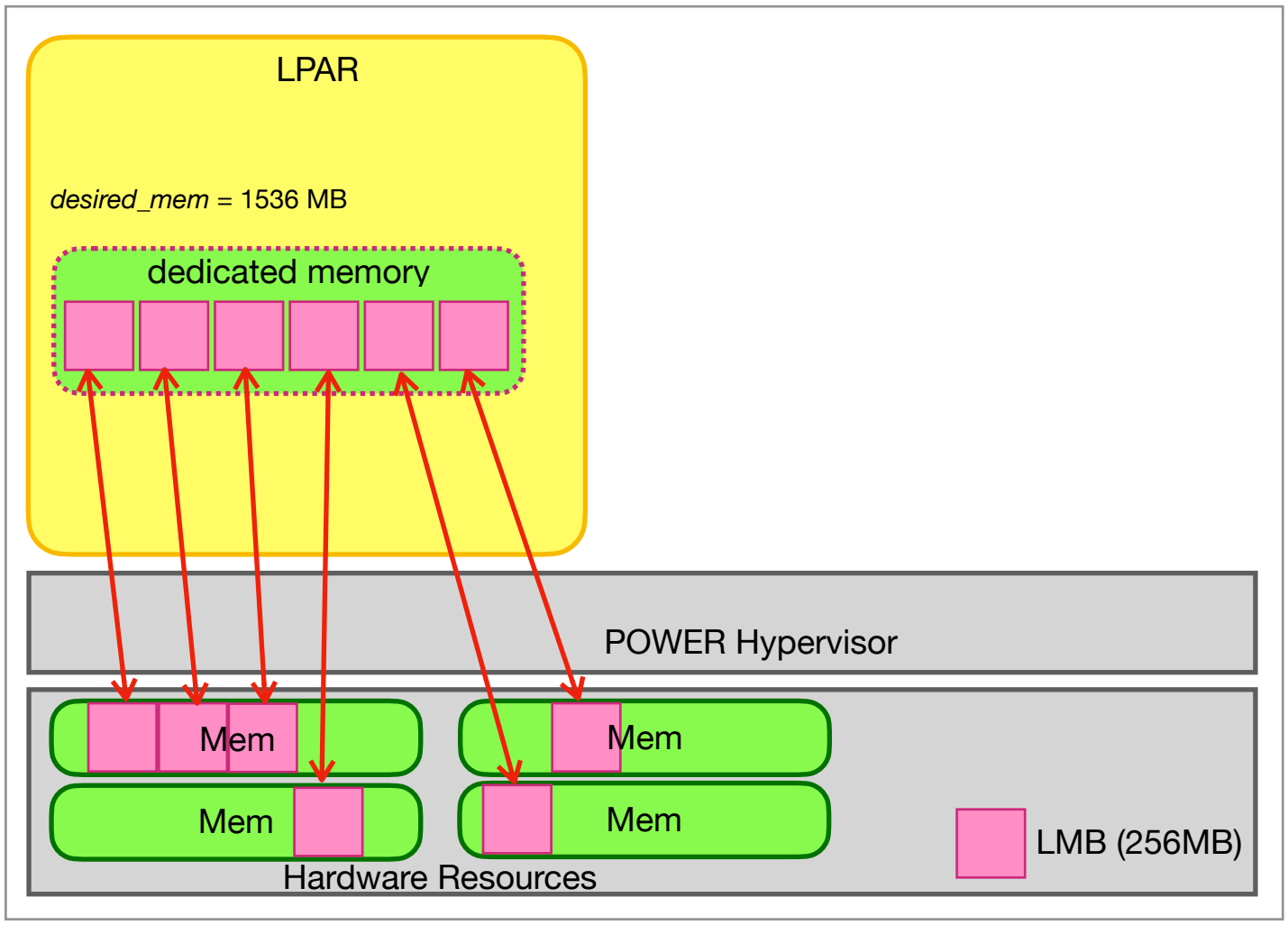


Bild 6.4: Zuweisung von physikalischem Speicher bei LPARs mit dediziertem Speicher in Einheiten der LMB-Größe (hier 256 MB).

Im Falle von LPARs mit geteiltem Speicher setzt sich der zugewiesene Speicher im allgemeinen aus physikalischem Speicher und Speicher auf Paging-Devices zusammen. Der über *desired_mem* konfigurierte und zugewiesene Speicher wird dann als logischer Speicher (*logical memory*) bezeichnet, da er ähnlich dem virtuellen Speicher in einem Betriebssystem, sich nicht nur aus physikalischem Speicher zusammensetzt. Auch hier wird der Speicher in Vielfachen der *LMB*-Größe zugewiesen. Auch die Größe des Shared Memory Pools muß in Vielfachen der *LMB*-Größe angegeben werden.

Wie in Bild 6.5 gezeigt, gibt es im Falle von Shared Memory aber zwei bedeutende Unterschiede zu der Zuweisung bei dediziertem Speicher. Zum Einen kann anstelle von physikalischem Speicher auch Speicher auf einem Paging-Device zugewiesen werden, zum Anderen erfolgt die Zuweisung von physikalischem Speicher (und Paging-Device Speicher) nicht in Vielfachen der *LMB*-Größe, sondern in Einheiten von Speicherseiten (bei POWER 4 KB).

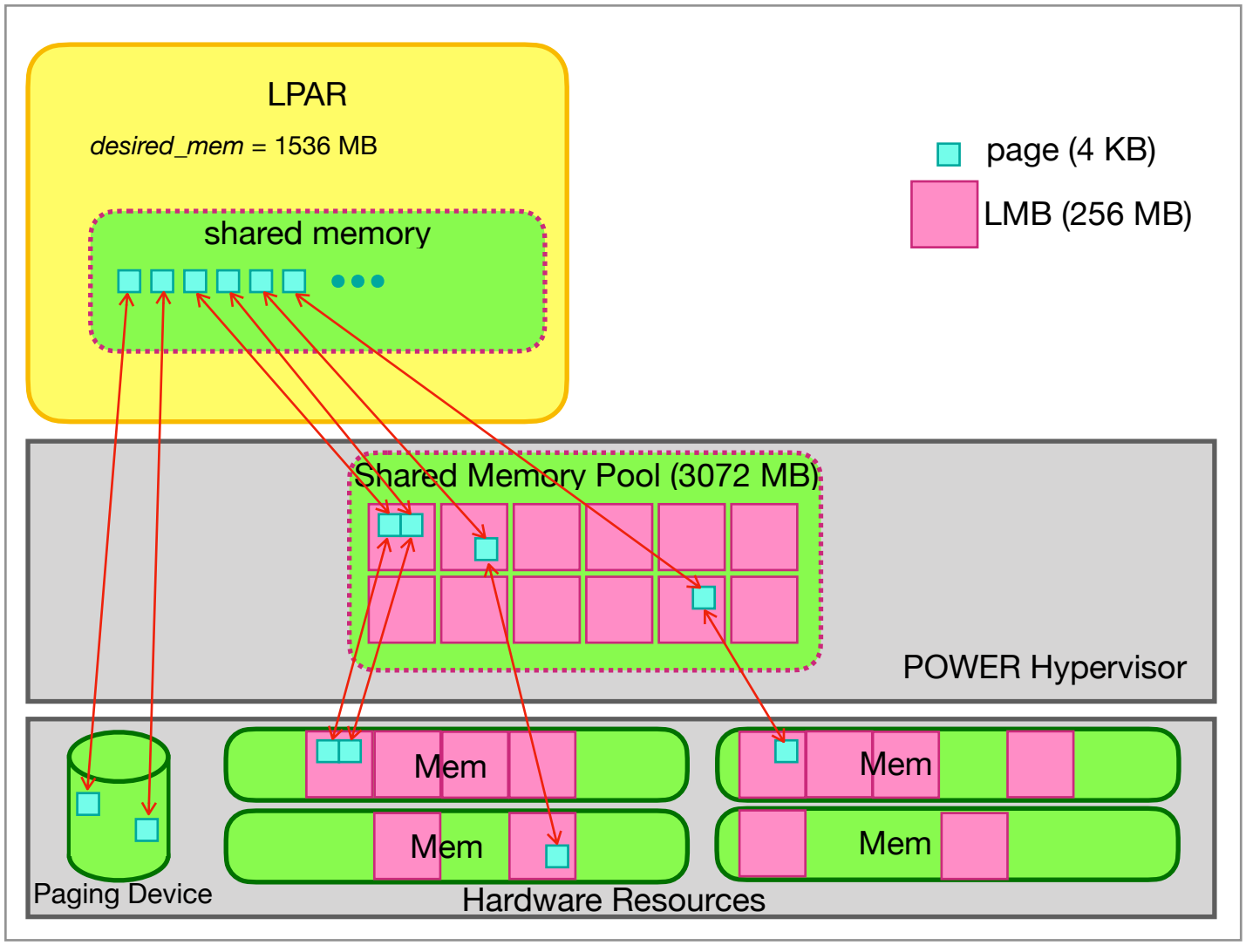


Bild 6.5: Zuweisung von logischem Speicher einer LPAR mit geteiltem Speicher in Einheiten der Seiten-Größe von 4 KB auf physikalischen Speicher und Paging-Device.

Physikalischer Speicher kann also in Einheiten von 4 KB zugewiesen (oder auch weggenommen) werden. Ein Paging in Einheiten von 256 MB würde für einen I/O zeitlich auch zu lange benötigen und würde in der Shared Memory LPAR zu spürbaren Verzögerungen führen.

6.2.6. Überwachung des logischen Speichers

Die Paging-Aktivität des Hypervisors lässt sich in einer Shared Memory LPAR überwachen. AIX stellt dazu einige Möglichkeiten zur Verfügung. So lässt sich überwachen wieviel physikalischer Speicher der LPAR aktuell zugeordnet ist, wieviele Seiten bisher wieder durch den Hypervisor eingelagert wurden und wieviel zeitliche Verzögerung dadurch entstanden ist.

Mit dem Kommando „*lparstat -m*“ können unter AIX Shared Memory Statistiken angezeigt werden:

```
aix22 # lparstat -m 1

System configuration: lcpu=16 mem=2048MB mpsz=1.00GB iome=351.00MB iomp=11 ent=0.20

physb   hpi  hpit  pmem  iomin   iomu   iomf  iohwm iomaf %entc  vcsw
----- ----- ----- ----- ------ ------ ------ ------ ----- ----- -----
16.39     1     1  0.68  302.1   10.9   48.9   12.5     0 205.3  2708
 0.27     0     0  0.68  302.1   10.9   48.9   12.5     0  12.5   609
 0.22     0     0  0.68  302.1   10.9   48.9   12.5     0   7.4   581
 9.80    96    30  0.68  302.1   10.9   48.9   12.5     0 100.4  1504
```

```
54.80     0     0  0.68  302.1   10.9   48.9   12.5     0 589.7  7024
81.46     1     1  0.68  302.1   10.9   48.9   12.5     0 907.1 11204
81.45     1     0  0.68  302.1   10.9   48.9   12.5     0 913.5 11236
79.88     0     0  0.68  302.1   10.9   48.9   12.5     0 911.6 10862
25.20     0     0  0.68  302.1   10.9   48.9   12.5     0 313.4  3839
 0.20     0     0  0.68  302.1   10.9   48.9   12.5     0   6.8   426
^C
aix22 #
```

Dabei werden unter Anderem die folgenden Attribute angezeigt:

- *mpsz* - Größe des Shared Memory Pools
- *hpi* - Anzahl der Hypervisor Page-Ins
- *hpit* - Verbrauchte Zeit für die Hypervisor Page-Ins in Millisekunden
- *pmem* - allozierter physikalischer Speicher in GB
- …

Ähnliche Informationen lassen sich auch mit dem Kommando *vmstat* und der Option ‚*-h*' ermitteln. Z.B.:

```
aix22 # vmstat -h -v
               524288 memory pages
               495111 lruable pages
                 2079 free pages
                    2 memory pools
               206873 pinned pages
                 90.0 maxpin percentage
                  3.0 minperm percentage
                 90.0 maxperm percentage
                  3.0 numperm percentage
                15311 file pages
                  0.0 compressed percentage
                    0 compressed pages
                  3.0 numclient percentage
                 90.0 maxclient percentage
                15311 client pages
                    0 remote pageouts scheduled
                    0 pending disk I/Os blocked with no pbuf
                    0 paging space I/Os blocked with no psbuf
                 2228 filesystem I/Os blocked with no fsbuf
                   47 client filesystem I/Os blocked with no fsbuf
                    0 external pager filesystem I/Os blocked with no fsbuf
                11129 Virtualized Partition Memory Page Faults
                 6437 Time resolving virtualized partition memory page faults
               278409 Number of 4k page frames loaned
                   53 Percentage of partition memory loaned
                 96.7 percentage of memory used for computational pages
aix22 #
```

Das Kommando *topas* wurde um die Option ‚*-L*' erweitert.

6.2.7. Ändern eines Shared Memory Pools

Ein Shared Memory Pool kann online geändert werden. LPARs die den Shared Memory Pool nutzen, können bei einer Änderung des Shared Memory Pools aktiv bleiben. Unter Anderem können die folgenden Attribute geändert werden:

- *pool_mem* - die Größe des Shared Memory Pools (muß ein Vielfaches der LMB-Größe sein)
- *max_pool_mem* - die maximale Größe des Shared Memory Pools
- *paging_vios_names* - ein oder zwei Paging Virtual-I/O-Server (durch Komma getrennte Liste)
- … - weitere Attribute sind in der Online Hilfe (*ms help chmempool*) aufgelistet

Der Shared Memory Pool auf dem Managed System wurde mit einer Größe von 20 GB angelegt. Die Größe soll auf 24 GB erweitert werden, wobei die maximale Größe 32 GB betragen soll. Das Kommando zum Ändern eines Shared Memory Pools ist „*ms chmempool*“ (*change memory pool*):

```
$ ms chmempool ms03 pool_mem=24G max_pool_mem=32G
$
```

Die neue Pool Größe kann mittels „*ms lsmempool*“ überprüft werden:

```
$ ms lsmempool ms03
                    POOL_MEM                PAGING
MS_NAME   CURR    AVAIL   FIRMWARE   MAX     VIOS_NAMES   MEM_DEDUP
ms03      24576   23896   269        32768   ms03-vio1    0
$
```

Aktuell besitzt der Shared Memory Pool nur einen Paging Virtual-I/O-Server und damit keine Redundanz bei Ausfall des Paging Virtual-I/O-Servers *ms03-vio1*. Das kann aber online geändert werden, falls ein weiterer Virtual-I/O-Server zur Verfügung steht:

```
$ ms chmempool ms03 paging_vios_names=ms03-vio1,ms03-vio2
$
```

Hat der zusätzliche Paging Virtual-I/O-Server Zugriff auf die gleichen Paging-Devices, dann wird dies automatisch erkannt:

```
$ ms lspgdev ms03
                                                                  REDUNDANT
MS_NAME   DEVICE_NAME   PAGING_VIOS_NAME   STATE    DEVICE_NAME   PAGING_VIOS_NAME   STATE      SIZE    TYPE   LPAR_ID
ms03      hdisk4        ms03-vio1          Active   hdisk4        ms03-vio2          Inactive   51200   phys   5
ms03      hdisk3        ms03-vio1          Active   hdisk3        ms03-vio2          Inactive   51200   phys   6
$
```

Beide Paging-Devices sind auch über den Paging Virtual-I/O-Server *ms03-vio2* zugreifbar. In diesem Fall sind sogar die Gerätenamen auf beiden Paging Virtual-I/O-Servern identisch, das ist aber im allgemeinen häufig nicht der Fall. Allerdings kennen die Shared Memory LPARs nur den beim Anlegen der LPAR konfigurierten ersten Paging Virtual-I/O-Server:

```
$ lpar lsmem aix22 shlpar1
             MEMORY          MEMORY         HUGE_PAGES    IO_ENTITLED_MEM      PAGING_VIOS
LPAR_NAME  MODE    AME  MIN   CURR  MAX   MIN  CURR  MAX  AUTO     CURR    PRIMARY    SECONDARY
aix22      shared  0.0  1024  2048  8192  0    0     0    1        351     ms03-vio1  -
shlpar1    shared  0.0  1024  4096  8192  0    0     0    1        60      ms03-vio1  -
$
```

Dies kann nur im Profil der LPAR geändert werden:

```
$ lpar -p standard chmem aix22 secondary_paging_vios_name=ms03-vio2
$
```

Damit sich die Änderung auswirkt, muß die LPAR heruntergefahren werden, und mit dem geänderten Profil wieder aktiviert werden:

```
$ lpar -i osshutdown aix22
$
$ lpar -p standard activate aix22
$
```

Erst dann kann die LPAR den zweiten Paging Virtual-I/O-Server verwenden und ist redundant konfiguriert:

```
$ lpar lsmem aix22 shlpar1
             MEMORY          MEMORY         HUGE_PAGES    IO_ENTITLED_MEM      PAGING_VIOS
LPAR_NAME  MODE    AME  MIN   CURR  MAX   MIN  CURR  MAX  AUTO     CURR    PRIMARY    SECONDARY
aix22      shared  0.0  1024  2048  8192  0    0     0    1        351     ms03-vio1  ms03-vio2
shlpar1    shared  0.0  1024  4096  8192  0    0     0    1        60      ms03-vio1  -
$
```

6.2.8. Switch-Over eines redundanten Paging-Devices

Sind für eine Shared Memory LPAR zwei redundante Paging Virtual-I/O-Server konfiguriert, kann auf das Paging-Device der LPAR auch dann zugegriffen werden, wenn der primäre Paging Virtual-I/O-Server ausfallen sollte.

Ein manuelles Wechseln des aktiven Paging Virtual-I/O-Server ist ebenfalls per Kommando „*lpar sopgdev*“ (*switch over paging device*) möglich:

```
$ lpar sopgdev aix22
$
```

Das Kommando wirkt sich sofort aus, der Hypervisor verwendet ab sofort den bisher inaktiven Paging Virtual-I/O-Server. Welcher Paging Virtual-I/O-Server gerade aktiv ist, ist nicht ganz leicht zu erkennen. Die beste Möglichkeit ist wahrscheinlich das I/O auf das Paging-Device zu überwachen. Nur auf dem gerade aktiven Paging Virtual-I/O-Server werden I/O auf das Paging-Device durchgeführt. Derjenige Paging Virtual-I/O-Server auf dem I/O auf das Paging-Device angezeigt wird, ist der gerade aktive Paging Virtual-I/O-Server. Die Eigenschaft primärer oder sekundärer Paging Virtual-I/O-Server zu sein, spielt dabei keine Rolle und ändert sich durch einen Switch-over auch nicht.

6.2.9. I/O Entitled Memory

Für I/O Operationen ist es essentiell das zum Zeitpunkt des I/O physikalischer Speicher verfügbar ist. Trifft beispielsweise ein Netzwerk-Paket ein und es gibt keinen physikalischen Speicher in dem das Paket abgelegt werden kann, da erst physikalischer Speicher über den Hypervisor aus dem Shared Memory Pool zugeordnet werden muß, dann kann dies die Latenz-Zeit der Netzwerk-Kommunikation deutlich erhöhen. Oder soll eine Platten Paging Operation durchgeführt werden und die Ziel-Speicherseite für die Operation ist selbst aktuellem keinem physikalischen Speicher zugeordnet, so erhöht dies die Dauer der Paging Operation deutlich. Im ungünstigsten Falle könnte eine I/O Operation auch fehlschlagen, weil ein vorgegebenes maximales Timeout erreicht wird.

Um dies zu verhindern wird bei Verwendung von Shared Memory eine gewisse maximale Menge an physikalischen Speicher für I/O-Operationen garantiert, unabhängig davon, wieviel andere LPARs gerade an physikalischem Speicher benötigen. Diese garantierte Menge an physikalischem Speicher wird als I/O Entitled Memory bezeichnet. Per Default wird die Größe automatisch ermittelt in Abhängigkeit der Art und Anzahl von virtuellen I/O Adaptern.

Wieviel I/O Entitled Memory einer LPAR gerade zugewiesen ist, kann mit dem Kommando „*lpar lsmem*“ angezeigt werden:

```
$ lpar lsmem aix22 shlpar1
             MEMORY          MEMORY         HUGE_PAGES    IO_ENTITLED_MEM      PAGING_VIOS
LPAR_NAME  MODE    AME  MIN   CURR  MAX   MIN  CURR  MAX  AUTO     CURR    PRIMARY    SECONDARY
aix22      shared  0.0  1024  2048  8192  0    0     0    1        351     ms03-vio1  ms03-vio2
shlpar1    shared  0.0  1024  4096  8192  0    0     0    1        60      ms03-vio1  -
$
```

Die LPAR *aix22* mit 4 virtuellen I/O Adaptern besitzt gerade *351* MB I/O Entitled Memory, wohingegen die LPAR *shlpar1* mit nur einem virtuellen Ethernet Adapter lediglich *60* MB I/O Entitled Memory besitzt.

6.2.10. Entfernen eines Paging-Devices

Die LPAR shlpar1 wurde nur zu Demonstrations-Zwecken angelegt, sie wird aber nicht wirklich benötigt. Wir deaktivieren die LPAR daher und entfernen dann das zugehörige Paging-Device *hdisk3*.

```
$ lpar -i shutdown shlpar1
$
```

Wie beim Hinzufügen eines Paging-Devices, gibt es auch beim Wegnehmen 2 Möglichkeiten. Entweder kann das Kommando „*ms rmpgdev*“ (*remove paging device*) verwendet werden, wobei neben dem Managed System auch der Paging Virtual-I/O-Server und das Paging-Device angegeben werden müssen:

```
$ ms rmpgdev ms03 ms03-vio1 hdisk3
$
```

Oder es kann das etwas kürzere, aber äquivalente, Kommando „*vios rmpgdev*“ verwendet werden:

```
$ vios rmpgdev ms03-vio1 hdisk3
$
```

Nachdem es nun nur noch ein Paging-Device gibt, das aktuell von der LPAR *aix22* in Verwendung ist, lässt sich die LPAR *shlpar1* natürlich auch nicht mehr starten:

```
$ lpar activate shlpar1
hmc01: chsysstate -m ms03 -r lpar -o on -n shlpar1
ERROR: remote HMC command returned an error (1)
StdErr: HSCL367F The activation with current configuration of one or more partitions failed
with the following error:
StdErr: Lpar 6: HSCLA435 There is not a device available in the reserved storage device
pool or shared memory pool that can be the paging space for partition shlpar1. This
partition requires a device with a size of at least 8192 MB. Add a device of at least that
size to the pool, and try the operation again. If you want to use a smaller device, then
modify the partition's profile to reduce the maximum memory setting for the partition so
that it is less than or equal to the size of the available device that you want to use,
then try the operation again.
StdErr:
StdErr:
$
```

Die Fehlermeldung weist eindeutig daraufhin, das ein Paging-Device von mindestens *8* GB Größe benötigt wird, um die LPAR starten zu können.

6.2.11. Entfernen eines Shared Memory Pools

Damit ein Shared Memory Pool entfernt werden kann, müssen zunächst alle LPARs die den Shared Memory Pool nutzen heruntergefahren werden:

```
$ lpar -i osshutdown aix22
$
```

Werden die LPARs weiterhin benötigt, müssen sie zu LPARs mit dediziertem Speicher umkonfiguriert werden. Diese Änderung kann nur im Profil einer LPAR durchgeführt werden:

```
$ lpar -p standard chmem aix22 mem_mode=ded
$
```

Nachdem alle Shared Memory LPARs gestoppt wurden, können als nächstes alle Paging-Devices weggenommen werden:

```
$ vios rmpgdev ms03-vio1 hdisk4
$
```

Als letztes kann dann der Shared Memory Pool gelöscht werden, dies geschieht mit dem Kommando „*ms rmmempool*“ (*remove memory pool*):

```
$ ms rmmempool ms03
$
```

6.3. Active Memory Expansion (AME)

Bei Active Memory Expansion (AME) werden gerade nicht benutzte Hauptspeicher-Seiten komprimiert, damit diese weniger Platz benötigen. Werden z.B. 10 GB Hauptspeicher mit einer Komprimierungsrate von 2 auf 5 GB reduziert, kann damit deutlich Hauptspeicher gespart werden. Eine LPAR kann dann entweder mit entsprechend weniger Hauptspeicher konfiguriert werden (kleinerer Wert von *desired_mem*), oder die LPAR kann mehr Arbeit bei gleicher Hauptspeicher-Größe verrichten. Die Komprimierung (und Dekomprimierung) von Speicher erfordert CPU-Ressourcen. Der zusätzlich gewonnene Speicher muß im Prinzip mit CPU-Ressourcen für die Komprimierung bezahlt werden. Die Komprimierung und Dekomprimierung muß durch das Betriebssystem durchgeführt werden. Bisher wird AME nur von AIX unterstützt.

In Bild 6.6 ist die Arbeitsweise von AME skizziert. Der Speicher ist in zwei Pools aufgeteilt, den Pool der nicht-komprimierten Speicherseiten (*Uncompressed Pool*) und den Pool der komprimierten Speicherseiten (*Compressed Pool*). Im Bild ist eine Komprimierung mit einer Rate von *1.75* angegeben. D.h. das *14* GB Daten, bei einer Komprimierungsrate von *1.75* auf *8* GB komprimiert werden können. Damit werden in *12* GB tatsächlich verfügbarem Speicher *4* GB + *14* GB = *18* GB Daten abgespeichert. Das entspricht einer Speichererweiterung um den Faktor *1.5*.

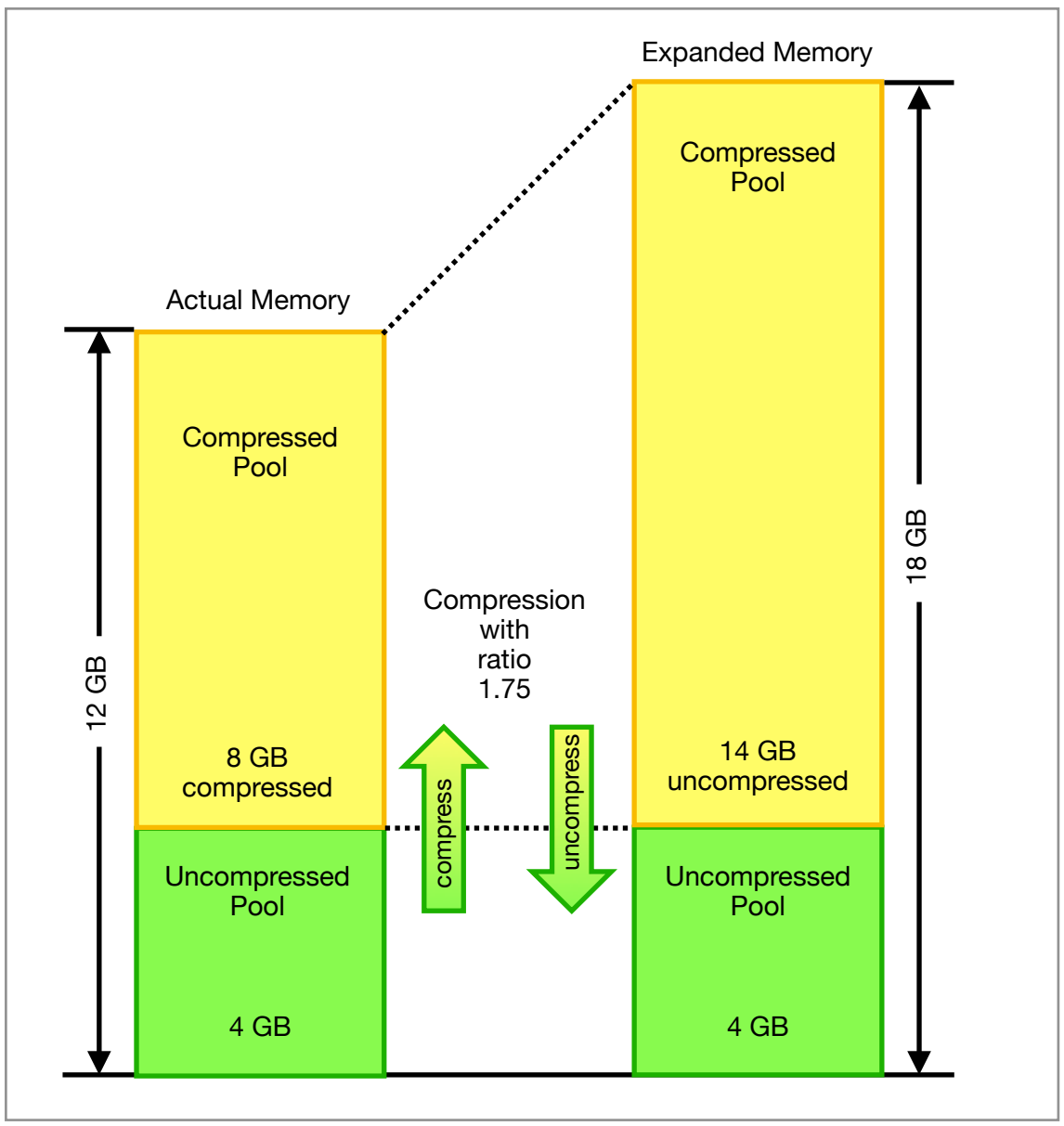


Bild 6.6: Durch AME werden 12 GB tatsächlicher Speicher zu 18 GB erweitertem Speicher mit einem AME-Faktor von 1.5.

Die Größen der beiden Pools sind nicht fix, sie hängen von der Speicherauslastung und dem Zugriffsmuster auf die Speicherseiten ab. Im Bild 6.6 ist also nur ein Moment festgehalten, im nächsten Moment können die Verhältnisse schon geringfügig anders aussehen. Der Zugriff von Prozessen auf komprimierte Speicherseiten ist nicht direkt möglich und auch nicht sinnvoll, da die Daten nicht mehr so vorliegen, wie der Prozeß sie abgespeichert hat. Greift ein Prozeß auf Daten zu, die durch Komprimierung im *Compressed Pool* liegen, muß das Betriebssystem diese zunächst entkomprimieren und im *Uncompressed Pool* ablegen. Komprimierte Speicherseiten können nicht auf einen Paging-Space ausgelagert werden.

AME kann für jede LPAR individuell aktiviert oder deaktiviert werden, allerdings nicht dynamisch. Soll für eine LPAR AME verwendet werden, muß der sogenannte AME-Faktor angegeben werden. Dieser gibt an um welchen Faktor der tatsächliche Speicher erweitert werden soll. Die dazu notwendige Komprimierungsrate ist größer, da ja der *Uncompressed Pool* unkomprimiert bleibt. Im Bild oben war die Komprimierungsrate z.B. *1.75* und der erzielte AME-Faktor *1.5*. Ein Teil des Speichers muß immer unkomprimiert bleiben, da sonst das Betriebssystem nicht mehr arbeiten kann.

Wie gut AME in der Praxis dann tatsächlich funktioniert, hängt sehr stark von mehreren Faktoren ab:

- Wie gut lassen sich die Daten komprimieren, d.h. welche Komprimierungsrate kann tatsächlich erreicht werden?
- Nach welcher Zeit erfolgt ein Zugriff auf Daten die komprimiert wurden.
- Passt das Working-Set an Speicherseiten in den *Uncompressed Pool*, der sich ergibt, nachdem weniger häufig benutzte Daten komprimiert wurden.

6.3.1. Konfigurieren von AME

Technisch ist die Konfiguration von AME sehr einfach. Dazu gibt es im Profil einer LPAR vom Typ *aixlinux* das Attribut *mem_expansion* (AME-Faktor). Dieses Attribut kann die folgenden Werte annehmen:

```
mem_expansion
   0 - disable memory expansion
   1.00 to 10.00 - memory expansion factor
```

Die Attribute *min_mem*, *desired_mem* und *max_mem* behalten für LPARs mit AME ihre Bedeutung und beziehen sich auf den physikalischen Speicher (dedizierter Speicher) oder den logischen Speicher (geteilter Speicher). Beim Anlegen einer LPAR kann der AME-Faktor direkt auf der Kommandozeile angegeben werden:

```
$ lpar create -m ms02 lpar7 desired_mem=4096 mem_expansion=1.5
    > lpar7
$
```

Soll eine bestehende LPAR AME verwenden, dann muß die LPAR deaktiviert werden und der AME-Faktor im Profil der LPAR konfiguriert werden:

```
$ lpar -p standard chmem aix22 mem_expansion=1.2
```

```
$
```

Nach dem nächsten Aktivieren der LPAR mit dem geänderten Profil wird dann AME verwendet.

6.3.2. Active Memory Expansion Planning Tool (*amepat*)

Bevor AME erstmalig aktiviert wird, empfiehlt es sich auf der betreffenden LPAR zunächst das Tool *amepat* (Active Memory Expansion Planning Tool) zum Einsatz zu bringen. Das Tool analysiert den existierenden Workload auf einem System und ermittelt wie gut sich die Daten komprimieren lassen. Basierend auf den gesammelten Daten während des Laufs werden einige AME-Faktoren mit dem geschätzten CPU-Aufwand aufgelistet und letztlich eine Empfehlung für eine Konfiguration gegeben:

```
aix09 # amepat 10

Command Invoked                : amepat 10

Date/Time of invocation        : Sun May 30 18:30:25 CEST 2021
Total Monitored time           : 28 mins 19 secs
Total Samples Collected        : 3

System Configuration:
---------------------
Partition Name                 : aix09
Processor Implementation Mode  : POWER8 Mode
Number Of Logical CPUs         : 32
Processor Entitled Capacity    : 0.80
Processor Max. Capacity        : 8.00
True Memory                    : 144.00 GB
SMT Threads                    : 4
Shared Processor Mode          : Enabled-Uncapped
Active Memory Sharing          : Disabled
Active Memory Expansion        : Disabled

System Resource Statistics:                Average             Min             Max
---------------------------               -----------         -----------     -----------
CPU Util (Phys. Processors)                0.81 [ 10%]         0.77 [ 10%]     0.87 [ 11%]
Virtual Memory Size (MB)                  73009 [ 50%]        73001 [ 50%]    73013 [ 50%]
True Memory In-Use (MB)                  116923 [ 79%]       116913 [ 79%]   116928 [ 79%]
Pinned Memory (MB)                        16273 [ 11%]        16273 [ 11%]    16273 [ 11%]
File Cache Size (MB)                      43540 [ 30%]        43538 [ 30%]    43542 [ 30%]
Available Memory (MB)                     69757 [ 47%]        69754 [ 47%]    69765 [ 47%]

Active Memory Expansion Modeled Statistics:
-------------------------------------------
Modeled Expanded Memory Size   : 144.00 GB
Average Compression Ratio      :   2.78

Expansion    Modeled True      Modeled              CPU Usage
Factor       Memory Size       Memory Gain          Estimate
---------    -------------     ------------------   -----------
     1.03        140.00 GB         4.00 GB [  3%]   0.00 [  0%]
     1.22        118.25 GB        25.75 GB [ 22%]   0.00 [  0%]
     1.41        102.25 GB        41.75 GB [ 41%]   0.00 [  0%]
     1.60         90.25 GB        53.75 GB [ 60%]   0.00 [  0%]
     1.79         80.50 GB        63.50 GB [ 79%]   0.00 [  0%]
     1.98         72.75 GB        71.25 GB [ 98%]   0.00 [  0%]
     2.17         66.50 GB        77.50 GB [117%]   0.14 [  2%]
```

```
Active Memory Expansion Recommendation:
---------------------------------------
The recommended AME configuration for this workload is to configure the LPAR
with a memory size of 66.50 GB and to configure a memory expansion factor
of 2.17.  This will result in a memory gain of 117%. With this
configuration, the estimated CPU usage due to AME is approximately 0.14
physical processors, and the estimated overall peak CPU resource required for
the LPAR is 1.01 physical processors.

NOTE: amepat's recommendations are based on the workload's utilization level
during the monitored period. If there is a change in the workload's utilization
level or a change in workload itself, amepat should be run again.

The modeled Active Memory Expansion CPU usage reported by amepat is just an
estimate.  The actual CPU usage used for Active Memory Expansion may be lower
or higher depending on the workload.
aix09 #
```

Die Beispiel-Ausgabe hier wurde nur über ein kurzes Zeitintervall gestartet und ist daher nicht aussagekräftig. Entscheidend ist die Auswahl des Zeitpunkts für den Start von *amepat* und die Dauer des Laufes. Eine repräsentative Aussage kann nur ermittelt werden, wenn die Zeitdauer ausreichend lange war und es während des Beobachtungszeitraums auch zu typischen Spitzenauslastungen kam.

Hinweis: Bei Cluster-Systemen ist besondere Vorsicht angebracht. In einem Fehlerfalle muß ein Cluster-Knoten unter Umständen Ressource Gruppen von anderen Cluster-Knoten mit übernehmen. Dies wird typischerweise bei der Konfiguration der Hauptspeicher-Größe berücksichtigt. Ein Lauf von *amepat* sollte daher am Besten durchgeführt werden, wenn zu übernehmende Ressource Gruppen aktuell auch auf dem System laufen.

6.3.3. Monitoring und Memory Deficit

Für die Überwachung von AME wurden einige AIX Performance Tools erweitert, wie z.B. *vmstat*, *lparstat* oder *svmon*. Auch *amepat* kann für die Überwachung von AME verwendet werden.

Eine Größe die überwacht werden sollte, ist das sogenannte Memory Defizit. In Bild 5.7 ist gezeigt, wie eine zu niedrige Komprimierungsrate zu fehlendem Speicher, dem Memory Defizit, führt. Die physikalische Hauptspeicher-Größe im Beispiel ist 12 GB, aufgeteilt in den Pool der unkomprimierten Speicherseiten mit 4 GB und dem Pool der komprimierten Speicherseiten mit 8 GB. Um die Expanded Memory Größe von 18 GB (AME-Faktor *1.5*) zu erreichen, müssen die Daten im *Compressed Pool* mindestens mit einer Komprimierungsrate von *1.75* komprimiert werden. Im Beispiel wurde allerdings nur eine Komprimierungsrate von *1.5* erreicht. Damit können im *Compressed Pool* nur 1.5 * 8 GB = 12 GB untergebracht werden. Zusammen mit den 4 GB des *Uncompressed Pools* ergibt sich dann eine Expanded Speichergröße von 16 GB anstelle von 18 GB. Es fehlen also 2 GB Speicher. Dieser fehlende Speicher wird als Memory Defizit bezeichnet und wird von den Überwachungs-Tools auch ausgegeben.

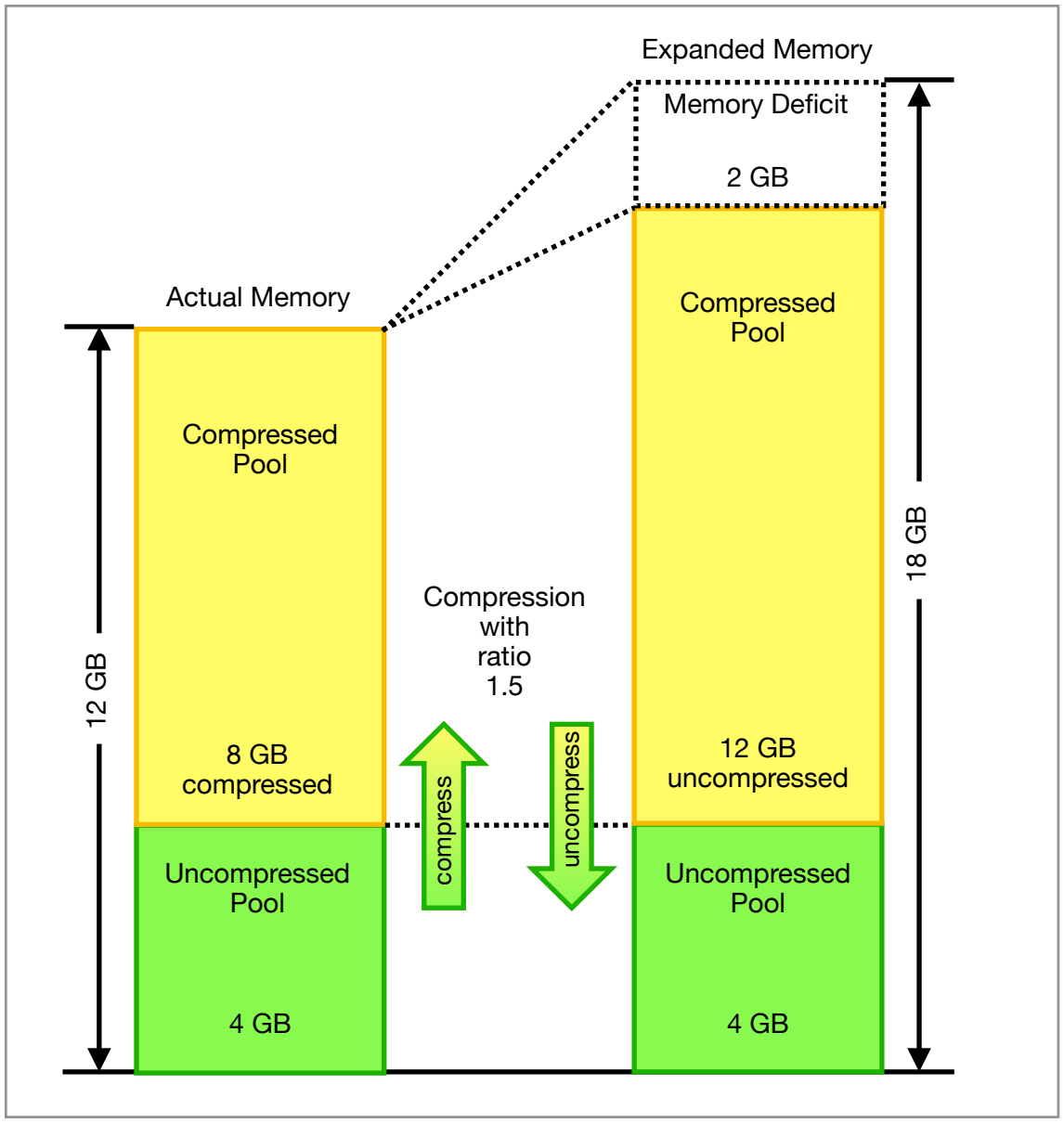


Bild 6.7: Entstandenes Memory Defizit von 2 GB aufgrund einer zu niedrigen Komprimierungsrate (1.5 anstelle von 1.75).

Ein Memory Defizit kann auf eine von zwei Arten aufgehoben werden:

1. Verkleinern des AME-Faktors und damit Verkleinerung des erweiterten Speichers.

2. Beibehalten der erweiterten Speichergröße durch Reduzierung des AME-Faktors und Hinzufügen von zusätzlichem Speicher zum Ausgleich.

Mit den oben genannten Monitoring Tools ist das Erkennen eines Memory Defizits sehr einfach, hier z.B. das Kommando *lparstat* mit der Option „*-c*“ (*compression*):

```
aix22 # lparstat -c 1

System configuration: type=Shared mode=Uncapped mmode=Ded-E smt=4 lcpu=16 mem=32512MB
tmem=29696MB psize=16 ent=2.00

%user  %sys  %wait  %idle physc %entc  lbusy  vcsw phint  %xcpu xphysc    dxm
----- ----- ------ ------ ----- ----- ------ ----- ----- ------ ------ ------
31.3   1.8    0.0   66.9  1.14  57.2   12.1  1654     0    0.0 0.0000      0
 30.8   1.6    0.0   67.6  1.13  56.3   11.7  1625     0    0.0 0.0000      0
 31.5   1.7    0.0   66.8  1.14  57.0   12.0  1715     0    0.0 0.0000      0
 57.1   1.5    0.0   41.4  1.95  97.7    8.4  1789     0    0.0 0.0000      0
 32.2   1.7    0.0   66.0  1.20  60.2    8.7  1889     0    0.0 0.0000      0
 ^C
aix22 #
```

Interessant sind hier die Spalten *%xcpu*, *xphysc* und *dxm*:

- *%xcpu*: Prozent der verbrauchten CPU-Ressourcen für AME
- *xphysc*: Verbrauch von physikalischen Prozessoren für AME
- *dxm*: Größe des Memory Defizits in MB

6.3.4. Ändern des AME-Faktors

Bei bestehender RMC-Verbindung zu den HMCs lässt sich der AME-Faktor einer LPAR dynamisch ändern. Hierzu kann mit dem Kommando „*lpar chmem*“ einfach der neue gewünschte AME-Faktor in Form des Attributs *mem_expansion* angegeben werden:

```
$ lpar chmem aix22 mem_expansion=1.2
$
```

Soll AME allerdings ausgeschaltet werden, ist dies wiederum nur über eine Änderung des Profils möglich:

```
$ lpar -p standard chmem aix22 mem_expansion=0
$
```

Die LPAR muß dann mit dem geänderten Profil neu aktiviert werden.

Kann die LPAR nicht deaktiviert werden, kann man den AME-Faktor auch auf den Wert *1.0* setzen. Damit ist zwar AME selbst nicht komplett deaktiviert, aber es werden keine Speicherseiten mehr komprimiert.

6.3.5. Hinzufügen und Wegnehmen von Speicher

Bei Verwendung von Active Memory Expansion kann Speicher dynamisch hinzugefügt und auch weggenommen werden. Die zugewiesene Speichermenge muß im Bereich *min_mem* und *max_mem* bleiben. Wird die Speichergröße geändert, wird mit Hilfe des AME-Faktors die neue Ziel-Speichergröße ermittelt. Fügt man bei einem AME-Faktor von 1.5 also beispielsweise 4 GB Speicher hinzu, dann ergibt das mit AME-Faktor einen Zuwachs von 6 GB Speicher für die AIX LPAR. Entsprechendes gilt auch für das Wegnehmen von Speicher.

6.3.6. Active Memory Expansion und Active Memory Sharing

Es ist möglich Active Memory Expansion und Active Memory Sharing zusammen zu verwenden.

7. I/O Virtualisierung

In diesem Kapitel werden die Möglichkeiten der I/O Virtualisierung von PowerVM vorgestellt. Die beiden zentralen Komponenten sind dabei der POWER Hypervisor und der sogenannte Virtual-I/O-Server. Der POWER Hypervisor besitzt Funktionen, die es ermöglichen virtuelle Prozessoren, virtuelle SCSI-Adapter und virtuelle FC-Adapter für LPARs zur Verfügung zu stellen. Er implementiert die Kommunikation und den Datentransport der virtuellen Adapter innerhalb eines Power Systems. Der Virtual-I/O-Server ist eine spezielle LPAR auf dem Managed System, mit einem für die Virtualisierung speziell erweiterten AIX-Betriebssystem. Dem Virtual-I/O-Server sind die physikalischen Adapter (Ethernet, SCSI, FC) zugeordnet und er übernimmt den Transport der Daten zwischen physikalischen Adaptern und virtuellen Adaptern.

Sobald ein Managed System eingeschaltet ist, ist der POWER Hypervisor aktiv. Speziell für die Virtualisierung von I/O stellt der Hypervisor die folgenden virtuellen Adapter Typen zur Verfügung:

- virtuelle serielle Adapter (für Client und Server), hierüber werden die Konsolen für die LPARs bereitgestellt
- virtuelle Ethernet Adapter (mit und ohne Trunking), mit der Unterstützung von IEEE 802.1q VLANs und QoS
- virtuelle FC Adapter (für Client und Server), damit können SAN-LUNs direkt einer Client-LPAR zur Verfügung gestellt werden (N_Port-ID Virtualization)
- virtuelle NIC Adapter (für Client und Server), was die Verwendung von SR-IOV mit automatischem Failover für Ethernet erlaubt
- virtuelle SCSI Adapter (für Client und Server), dabei können LUNs, Logical Volumes oder auch Dateien als Disks auf virtuelle SCSI Server-Adapter gemappt werden

Außerdem implementiert der POWER Hypervisor virtuelle Ethernet Switches. Diese verbinden LPARs auf der gleichen Hardware miteinander, erlauben aber in Zusammenarbeit mit den Virtual-I/O-Servern auch eine Anbindung an externe Netzwerke, ohne das eine LPAR hierfür einen physikalischen Ethernet-Adapter benötigen würde.

Die Virtual-I/O-Server komplementieren die Funktionalitäten des POWER Hypervisors, indem sie die Verbindung zwischen virtuellen Ressourcen des Hypervisors mit den physikalischen Adaptern der Hardware realisieren. Dies wird unter anderem erreicht, durch

- Shared Ethernet Adapter (SEA)
- NPIV-Mapping
- VSCSI-Mapping
- vNIC-Backing Device

In Bild 7.1 ist das Zusammenspiel zwischen POWER Hypervisor und Virtual-I/O-Server für Virtual Ethernet und Virtual FC skizziert. Die virtuellen Ethernet Adapter der Client-LPARs besitzen eine Anbindung an einen virtuellen Ethernet Switch (implementiert durch den Hypervisor). Der Virtual-I/O-Server ist durch einen virtuellen Ethernet Trunking Adapter ebenfalls an den internen virtuellen Ethernet Switch angebunden. Auf dem Virtual-I/O-Server schafft ein sogenannter Shared Ethernet Adapter eine Verbindung zwischen dem virtuellen Ethernet Trunking Adapter

und einem physikalischen Ethernet Port und damit die Verbindung zur Außenwelt. Der Shared Ethernet Adapter ist dabei eine durch Software auf dem Virtual-I/O-Server implementierte Layer 2 Bridge.

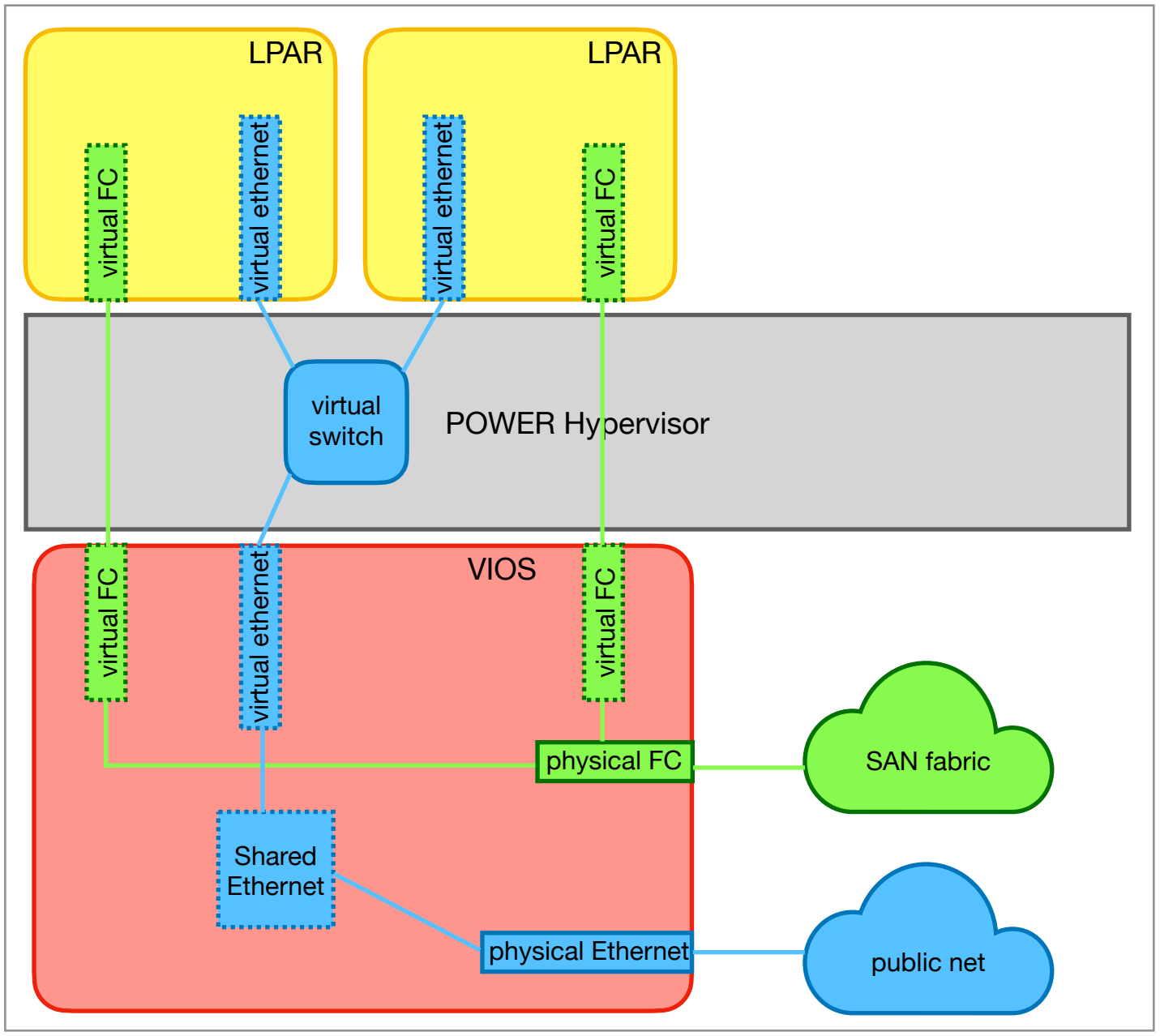


Bild 7.1: Virtual Ethernet und Virtual FC

Für den Zugriff auf Disks besitzen die beiden gezeigten LPARs jeweils einen virtuellen FC Client-Adapter, diese sind über den Hypervisor mit ihrem Gegenstück, dem virtuellen FC Server-Adapter, verbunden. Der Hypervisor transportiert dabei die Daten zwischen Client- und Server-Adapter. Die Anbindung an einen physikalischen FC Adapter erfolgt dann durch Mappen des virtuellen FC Server-Adapters auf einen der physikalischen FC Adapter auf dem Virtual-I/O-Server. Für das Mapping wird NPIV verwendet (N_Port-ID Virtualization). SAN-LUNs können dann direkt auf die WWN des virtuellen FC Client-Adapter gemappt werden.

7.1. Virtuelle Slots

Physikalische Adapter benötigen einen physikalischen Slot in einem Power System. Analog benötigen virtuelle Adapter einen virtuellen Slot. Wie die virtuellen Slots einer LPAR aktuell belegt sind, lässt sich mit Hilfe des Kommandos „*lpar lsvslot*“ (*list virtual slots*) anzeigen:

```
$ lpar lsvslot aix22
SLOT   REQ   ADAPTER_TYPE    STATE   DATA
0      Yes   serial/server   1       remote: (any)/any connect_status=unavailable hmc=1
1      Yes   serial/server   1       remote: (any)/any connect_status=unavailable hmc=1
5      No    eth             1       PVID=100 VLANS= ETHERNET0 1DC8DB485D1E
```

```
10    No   fc/client      1      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
20    No   fc/client      1      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
$
```

In den virtuellen Slots *0* und *1* sind zwei virtuelle serielle Adapter, diese werden bei jeder LPAR automatisch von PowerVM angelegt. Die Slots *2* bis *4* sind nicht benutzt und tauchen daher in der Ausgabe auch nicht auf. In Slot *5* befindet sich ein virtueller Ethernet Adapter mit der Port-VLAN-ID 100, angebunden an den virtuellen Ethernet Switch *ETHERNET0*. In den Slots *10* und *20* befindet sich je ein virtueller FC Client-Adapter. Die zugehörigen virtuellen FC Server-Adapter befinden sich auf den Virtual-I/O-Servern *ms03-vio1* in Slot *5* bzw. *ms03-vio2* in Slot *4*. Die primäre und sekundäre *WWPN* der virtuellen FC-Client Adapter sind ebenfalls angegeben.

Wieviele virtuelle Slots eine LPAR maximal haben kann, ist im Profil der LPAR hinterlegt und kann mit „*lpar lsattr*“ (*list attributes*) unter Angabe des Profil-Namens ausgegeben werden:

```
$ lpar -p standard lsattr aix22
                      AUTO   BOOT  POWER   MAX    AFFINITY  WORK
LPAR_NAME  NAME       START  MODE  CTRL   VSLOTS  GROUP     GROUP
aix22      standard   0      norm  none   30      none      none
$
```

Die LPAR *aix22* besitzt *30* virtuelle Slots. Die Anzahl der virtuellen Slots lässt sich nicht dynamisch ändern. Die Anzahl kann nur im Profil geändert werden und die Änderung wirkt sich erst aus, wenn die LPAR das nächste Mal, unter Angabe des Profil-Namens, aktiviert wird. Das Attribut ist *max_virtual_slots* und kann mit dem Kommando „*lpar chattr*“ (*change attributes*) im Profil geändert werden:

```
$ lpar -p standard chattr aix22 max_virtual_slots=40
$
```

7.2. Virtuelle serielle Adapter

Jede LPAR besitzt automatisch nach dem Anlegen 2 virtuelle serielle Adapter in den virtuellen Slots *0* und *1*. Die vorhandenen virtuellen seriellen Adapter lassen sich entweder mit dem Kommando „*lpar lsvslot*“, zusammen mit allen anderen virtuellen Adaptern, anzeigen oder auch mit dem Kommando „*lpar lsserial*“ (*list serial adapters*), welches nur virtuelle serielle Adapter auflistet:

```
$ lpar lsserial aix22
                                                                  REMOTE
LPAR_NAME  SLOT  REQ  STATE  TYPE    STATUS       HMC  LPAR_NAME  LPAR_ID  SLOT
aix22      0     Yes  1      server  unavailable  Yes  -          any      any
aix22      1     Yes  1      server  unavailable  Yes  -          any      any
$
```

Bei den beiden Adaptern handelt es sich um Server Adapter, nicht Client Adapter! Die LPAR bietet den Zugriff über den seriellen Adapter als Service an. Die Spalte HMC in der Ausgabe mit dem Wert „*Yes*“ gibt an, das ein Zugriff von der HMC aus auf die beiden seriellen Adapter unterstützt ist. Dies erlaubt eine Konsolen-Verbindung von der HMC aus auf die LPAR und dürfte die häufigste Nutzung der virtuellen seriellen Adapter sein.

Ein Zugriff auf die Konsole einer LPAR ist sehr einfach mit dem Kommando „*lpar console*“ möglich:

```
$ lpar console aix22
```

```
 Open in progress

  Open Completed.





AIX Version 7
Copyright IBM Corporation, 1982, 2020.
Console login: root
root's Password:
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 7.1!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************

aix22:/root>
aix22:/root>
```

Soll die Konsolen-Verbindung beendet werden, sollte man sich in der LPAR zunächst ausloggen (exit) und kann die Verbindung dann, mittels „~.“, beenden (das ist die gleiche Kombination wie bei OpenSSH).

Zu jedem Zeitpunkt kann nur maximal eine Konsolen-Verbindung bestehen, der Versuch eine zweite Konsole für eine LPAR zu öffnen, führt zu einer Fehlermeldung:

```
$ lpar console aix22

A terminal session is already open for this partition.
 Only one open session is allowed for a partition.
 Exiting....  Received end of file, Exiting.
                                          Shared connection to hmc01 closed.
$
```

Eine bestehende Konsolen-Verbindung lässt sich mit dem Kommando „*lpar rmconsole*“ (*remove console*) beenden. Dabei spielt es keine Rolle wer die Konsole gerade besitzt, die Verbindung wird umgehend beendet:

```
$ lpar rmconsole aix22
$
```

In der bestehenden Konsolen-Sitzung wird dabei die folgende Meldung ausgegeben:

```
Connection has closed


 This session is no longer connected. Please close this window.
```

Eine weitere Möglichkeit besteht darin die Option „*-f*“ (*force*) beim Öffnen der Konsole zu verwenden. Eine eventuell bestehende Konsolen-Sitzung wird dabei automatisch beendet!

```
$ lpar console -f aix22

 Open in progress

  Open Completed.





AIX Version 7
Copyright IBM Corporation, 1982, 2020.
Console login:
```

Welche Funktionen im Zusammenhang mit virtuellen seriellen Adaptern von PowerVM unterstützt sind, lässt sich mit Hilfe der Online Hilfe des *LPAR-Tool*s auflisten:

```
$ lpar help serial
USAGE: lpar [<option> ...] <keyword> [<option> ...] [<argument> ...]

Recognized keywords for topic 'serial' are:
   [-h <hmc>] [-m <ms>] [-p <profile>] addserial [-c] [-d] [-f] [-l <detail_level>] [-w
<wait_time>] [-v] <lpar> <slot> [<remote_lpar_name> <remote_slot_num>]
   [-h <hmc>] [-m <ms>] [-p <profile>] chserial [-r] [-R] [-v] <lpar> <slot>
[<attributes> ...]
  [-h <hmc>] [-m <ms>] console [-f] [-v] <lpar>
   [-h <hmc>] [-m <ms>] [-p <profile>] lsserial [{-o <format>|-f|-j|-y}] [-F <fields>] [-s
<selections>] [-v] [<lpar> ...]
   [-h <hmc>] [-m <ms>] [-p <profile>] lsvslot [{-o <format>|-f|-j|-y}] [-t <type>] [-F
<fields>] [-s <selections>] [-v] <lpar>
   [-h <hmc>] [-m <ms>] rmconsole [-v] <lpar>
   [-h <hmc>] [-m <ms>] [-p <profile>] rmserial [-d] [-f] [-l <detail_level>] [-w
<wait_time>] [-v] <lpar> <slot>
$
```

Neben dem Auflisten von seriellen Adaptern und Verwendung der Konsole, können auch weitere virtuelle serielle Adapter angelegt, geändert und entfernt werden.

7.2.1. Konsole über Virtual-I/O-Server

Um das Hinzufügen eines virtuellen seriellen Adapters zu demonstrieren, soll in diesem Abschnitt ein relativ unbekanntes Feature von PowerVM angeschaut werden - die Möglichkeit eine Konsolen-Sitzung zu einer LPAR von einem Virtual-I/O-Server aus zu starten. Damit ist eine Konsole selbst dann verfügbar, wenn keine HMC verfügbar ist. Von dieser Möglichkeit wird aber in der Praxis in der Regel kein Gebrauch gemacht.

Auf dem Virtual-I/O-Server muss dazu lediglich ein virtueller serieller Client-Adapter angelegt werden, dabei wird als Remote-Endpunkt die Client-LPAR und die Slot-Nummer *0* des virtuellen seriellen Server-Adapters auf dem Client angegeben. Per Default legt das Kommando „*lpar addserial*“ virtuelle serielle Adapter vom Typ Server an, es muß daher die Option „*-c*“ (Client-Adapter) angegeben werden. Als Slot-Nummer auf dem Virtual-I/O-Server wird der Slot *19* angegeben, der bisher nicht in Benutzung ist:

```
$ lpar addserial -c ms03-vio1 19 aix22 0
$
```

Insgesamt hat der Virtual-I/O-Server nun neben den beiden standardmäßigen virtuellen seriellen Adaptern in Slot *0* und *1*, noch einen weiteren seriellen Adapter in Slot *19*:

```
$ lpar lsserial ms03-vio1
                                                                  REMOTE
LPAR_NAME  SLOT  REQ  STATE  TYPE    STATUS       HMC  LPAR_NAME  LPAR_ID  SLOT
ms03-vio1  1     Yes  1      server  unavailable  Yes  -          any      any
ms03-vio1  0     Yes  1      server  unavailable  Yes  -          any      any
ms03-vio1  19    No   1      client  unavailable  No   aix22      30       0
$
```

Damit kann jetzt vom Virtual-I/O-Server *ms03-vio1* jederzeit eine Konsolen-Sitzung zur LPAR *aix22* geöffnet werden. Dazu muß man sich als Benutzer *padmin* auf dem Virtual-I/O-Server einloggen und das Kommando „*mkvt -id*“ mit der LPAR-ID der LPAR *aix22* (LPAR-ID ist die *30* gemäß der Ausgabe oben) starten:

```
padmin@ms03-vio1> mkvt -id 30



AIX Version 7
Copyright IBM Corporation, 1982, 2020.
Console login: root
root's Password:
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 7.1!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************

aix22:/root>
aix22:/root> exit


AIX Version 7
Copyright IBM Corporation, 1982, 2020.
Console login: ~

Virtual terminal has been disconnected

padmin@ms03-vio1>
```

Die Konsolen-Sitzung kann mit „~.“ beendet werden. Ist der Login auf den Virtual-I/O-Server mit OpenSSH erfolgt, dann muß die Konsolen-Sitzung mit „~~.“ beendet werden, da in dem Fall die Kombination „~.“ die OpenSSH-Sitzung zum Virtual-I/O-Server beendet, aber nicht die Konsolen-Sitzung vom Virtual-I/O-Server zur LPAR!

Eine bestehende Konsolen-Sitzung zwischen Virtual-I/O-Server und LPAR kann als Benutzer *padmin* auf dem Virtual-I/O-Server mit dem Kommando „*rmvt -id*“ und der LPAR-ID der betreffenden LPAR beendet werden:

```
padmin@ms03-vio1> rmvt -id 30
padmin@ms03-vio1>
```

Der virtuelle serielle Client-Adapter kann bei Bedarf mit dem Kommando „*lpar rmserial*“ wieder entfernt werden:

```
$ lpar rmserial ms03-vio1 19
$
```

Bei der Verwendung von Konsolen-Sitzungen über Virtual-I/O-Server ist zu beachten, dass eine Konsolen-Sitzung zwischen Virtual-I/O-Server und LPAR nicht über die HMC (und damit nicht mit dem *LPAR-Tool*) beendet werden kann. Dies geht nur über ein Einloggen auf den Virtual-I/O-Server und Verwendung von *rmvt* als Benutzer *padmin*! Umgekehrt lässt sich eine Konsolen-Sitzung zwischen HMC und LPAR, nicht vom Virtual-I/O-Server aus beenden.

7.3. Virtual Ethernet

Mit Hilfe von virtuellen Ethernet Adaptern können LPARs im Netzwerk kommunizieren, ohne das dabei jede LPAR einen eigenen physikalischen Ethernet Adapter benötigt. Die virtuellen Ethernet Adapter sind dabei jeweils an interne virtuelle Ethernet Switches angebunden, welche durch den POWER Hypervisor implementiert sind. Jeder virtuelle Ethernet Switch unterstützt dabei den IEEE 802.1q Standard (VLANs und QoS). Zwei LPARs auf dem gleichen Managed System, die an den gleichen virtuellen Ethernet Switch angebunden sind, können direkt über den Hypervisor miteinander kommunizieren, hierbei kann ein Durchsatz von weit mehr als 20 GB/s erreicht werden. In Bild 7.2 ist diese Inter-Partitions Kommunikation innerhalb eines Managed Systems skizziert.

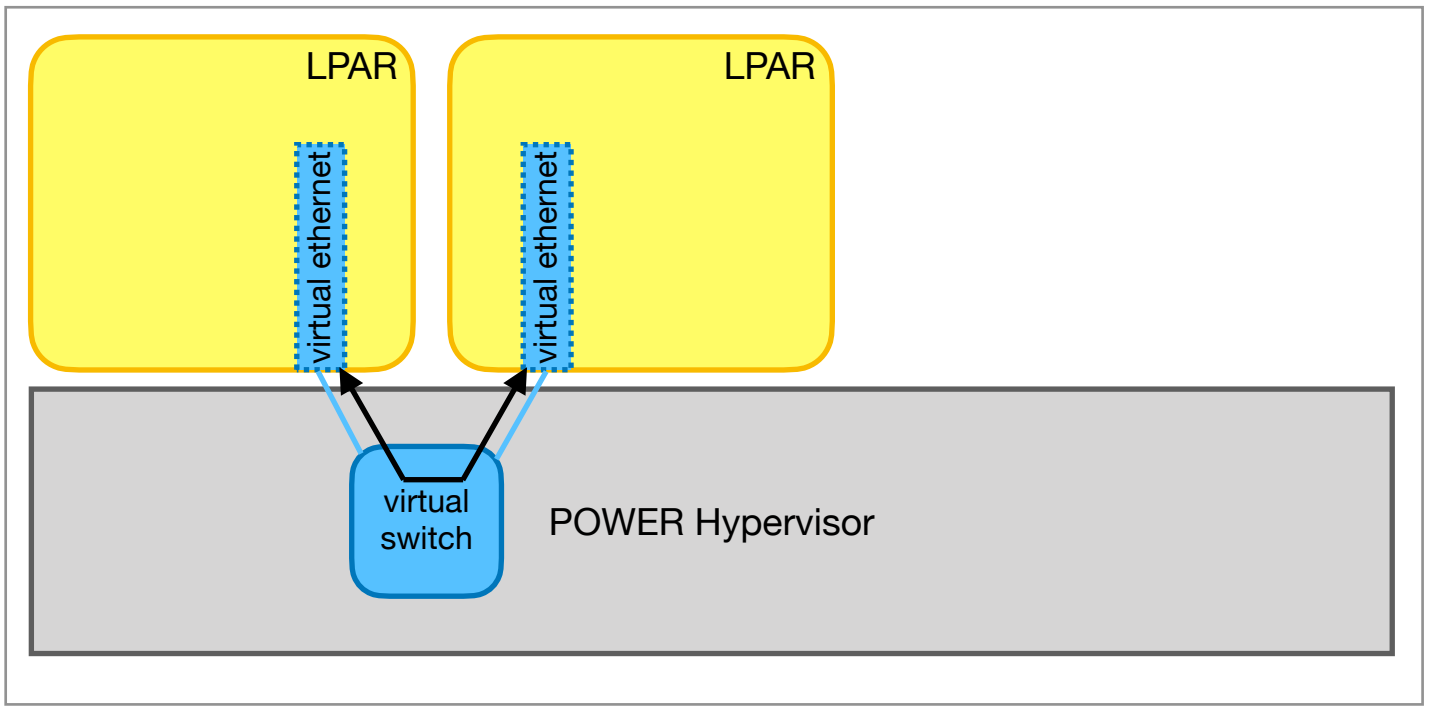


Bild 7.2: Inter-Partitions Kommunikation über einen virtuellen Ethernet Switch.

Damit LPARs auch mit Systemen außerhalb des Managed Systems kommunizieren können, stellt PowerVM sogenannte Trunking Adapter zur Verfügung. Ein Trunking Adapter ist dabei ein spezieller virtueller Ethernet Adapter, der über einen Shared Ethernet Adapter (SEA) auf einem Virtual-I/O-Server an einen physikalischen Ethernet Adapter angebunden ist. Der Shared Ethernet Adapter fungiert dabei als Layer-2 Bridge und wird in Software auf dem Virtual-I/O-Server implementiert. Netzwerk-Pakete einer LPAR, die an ein externes System gerichtet sind, werden vom virtuellen Ethernet Switch über einen Trunking Adapter an einen Virtual-I/O-Server

weitergeleitet, welcher dann die Pakete mit Hilfe des Shared Ethernet Adapters über einen physikalischen Ethernet Adapter an einen externen Netzwerk-Switch weiterleitet.

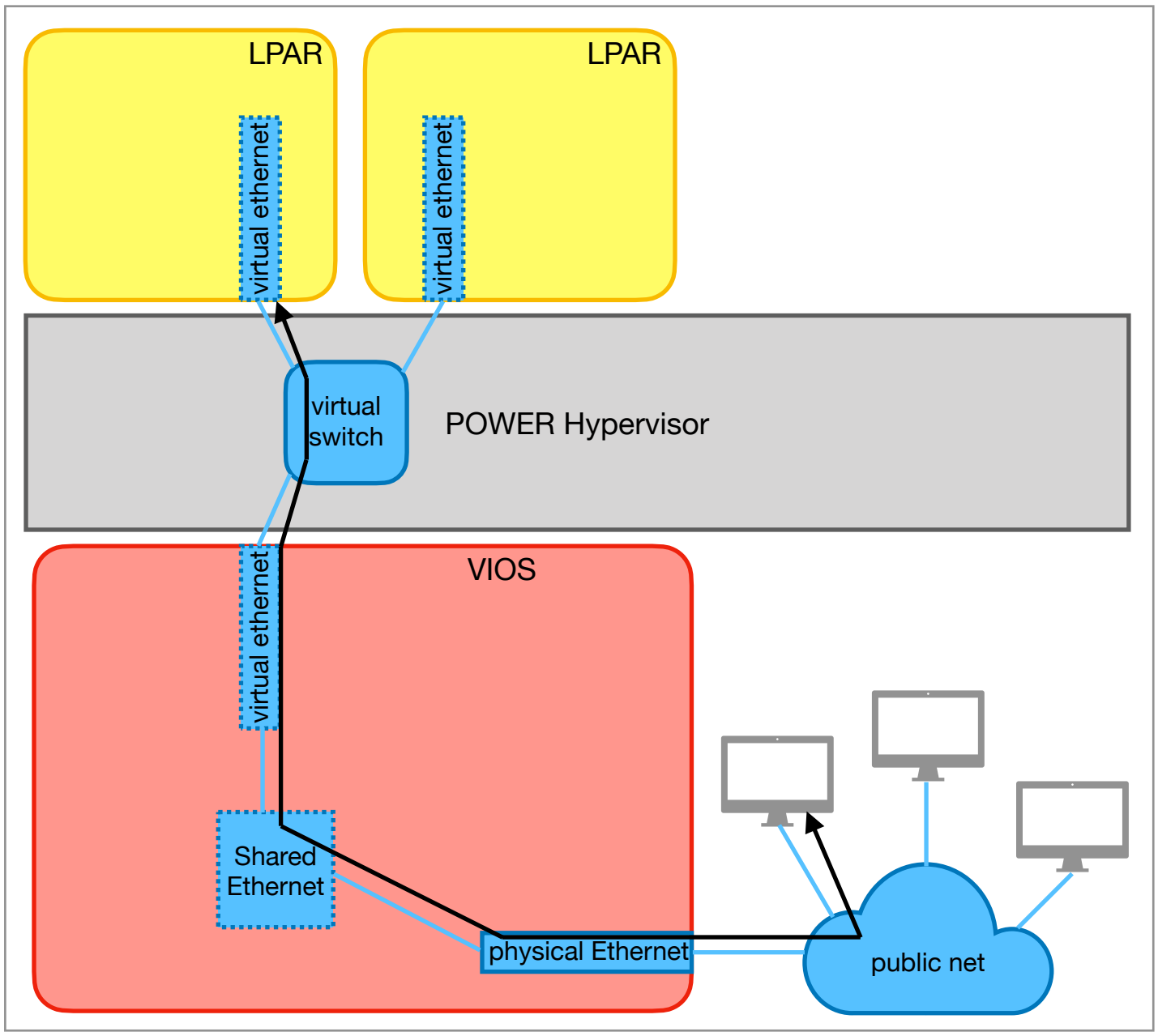


Bild 7.3: Ethernet Kommunikation ins externe Netzwerk

In Bild 7.3 ist der Kommunikationsweg einer LPAR zu einem externen System skizziert. Für die End-Systeme (LPAR und Host im externen Netzwerk) ist der Transport-Weg transparent.

In diesem Kapitel wird zunächst nur auf die Client-Seite von Virtual Ethernet eingegangen. Die Konfiguration von Shared Ethernet Adaptern (SEAs) auf einem Virtual-I/O-Server wird an späterer Stelle im Detail angesprochen.

7.3.1. VLANs und VLAN-Tagging

Jeder virtuelle Ethernet Switch unterstützt VLANs (Virtual Local Area Network) gemäß IEEE 802.1q. Dies bedeutet letztlich das innerhalb eines virtuellen Ethernet Switches alle Ethernet Pakete ein VLAN-Tag besitzen. Erlaubte VLAN-IDs sind *1-4095*. Jeder virtuelle Ethernet Adapter besitzt eine sogenannte Port-VLAN-ID (Attribut *port_vlan_id*), diese muß beim Anlegen eines virtuellen Ethernet Adapters angegeben werden. Wie in Bild 7.4 gezeigt, wird dabei jedem ausgehenden *untagged* Paket (ein Paket ohne VLAN-Header), durch den Hypervisor, ein VLAN-Header mit der Port-VLAN-ID hinzugefügt. Entsprechend wird bei eingehenden Paketen mit einer VLAN-ID , die gleich der Port-VLAN-ID des Ziel Adapters ist, der VLAN-Header vom Hypervisor wieder entfernt. Damit können LPARs miteinander kommunizieren, ohne selbst VLAN-Header zu verwenden. Pakete mit einem VLAN-Header werden als *tagged* (markiert) bezeichnet, Pakete ohne VLAN-Header werden entsprechend als *untagged*

(nicht markiert) bezeichnet. Das Hinzufügen eines VLAN-Headers zu einem Paket wird dementsprechend als *tagging* bezeichnet. Ethernet Adapter die keine VLAN-Header unterstützen, werden oft als VLAN *unaware* bezeichnet.

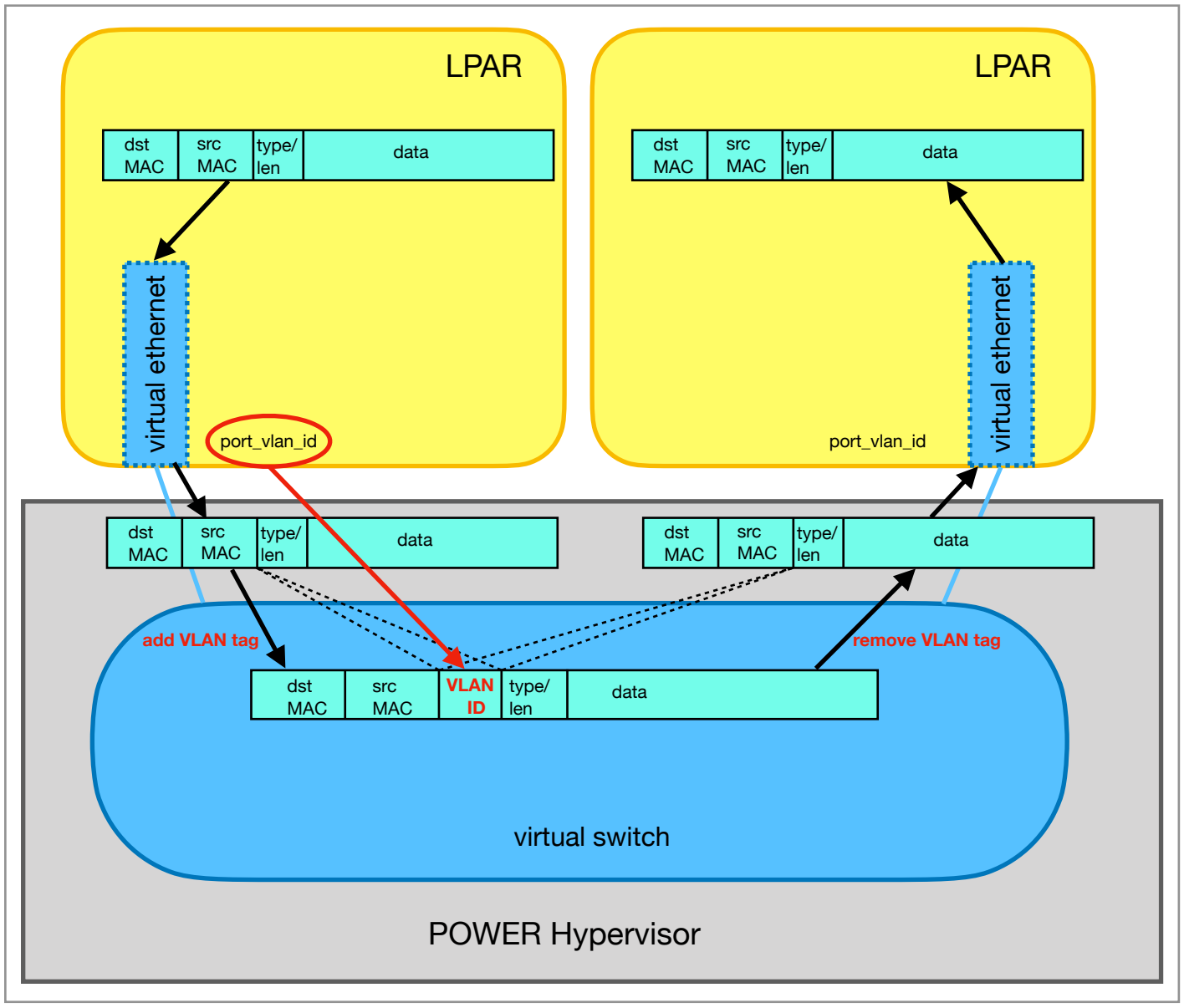


Bild 7.4: VLAN-Tagging mit der Port-VLAN-ID (*port_vlan_id*)

Wie das Bild 7.4 demonstriert können Ethernet Adapter die selbst kein VLAN-Tagging verwenden/unterstützen, trotzdem ohne Probleme in einer Umgebung mit VLANs funktionieren, indem vom Switch (hier POWER Hypervisor) das Tagging mit einer Port-VLAN-ID durchgeführt wird.

Ein virtueller Ethernet Adapter kann optional als IEEE 802.1q kompatibel angelegt werden (Attribut *ieee_virtual_eth*=yes). Er unterstützt dann zusätzlich zu *untagged* Paketen auch *tagged* Pakete für eine Liste von VLAN-IDs, die mit dem Attribut *addl_vlan_ids* angegeben werden können. Über einen virtuellen Ethernet Adapter können dann mehrere verschiedene VLANs von einer LPAR verwendet werden. Dabei muß der virtuelle Ethernet Adapter für die zusätzlichen VLAN-IDs den Paketen einen VLAN-Header hinzufügen. D.h. diese Pakete kommen dann schon als *tagged* Pakete beim virtuellen Switch an und der Hypervisor muß kein VLAN-Tag mehr hinzufügen.

Bild 7.5 zeigt die Zustellung eines Pakets mit VLAN-ID 200: Die VLAN-ID 200 gehört zu den zusätzlichen VLAN-IDs (*addl_vlan_ids*) des sendenden virtuellen Ethernet Adapters, daher muß der virtuelle Ethernet Adapter ein *tagged* Paket generieren. Dieses wird dann unverändert an den virtuellen Ethernet Switch weitergeleitet. Die VLAN-ID 200 gehört auch beim Ziel Ethernet Adapter zu den zusätzlichen VLAN-IDs. Der VLAN-Header wird daher vom Hypervisor nicht entfernt und das Paket wird inklusive VLAN-Header an die Ziel-LPAR über den virtuellen Ethernet Adapter weitergegeben. Der VLAN-Header bleibt während der ganzen Zustellung von der Quell-LPAR bis zur Ziel-LPAR erhalten!

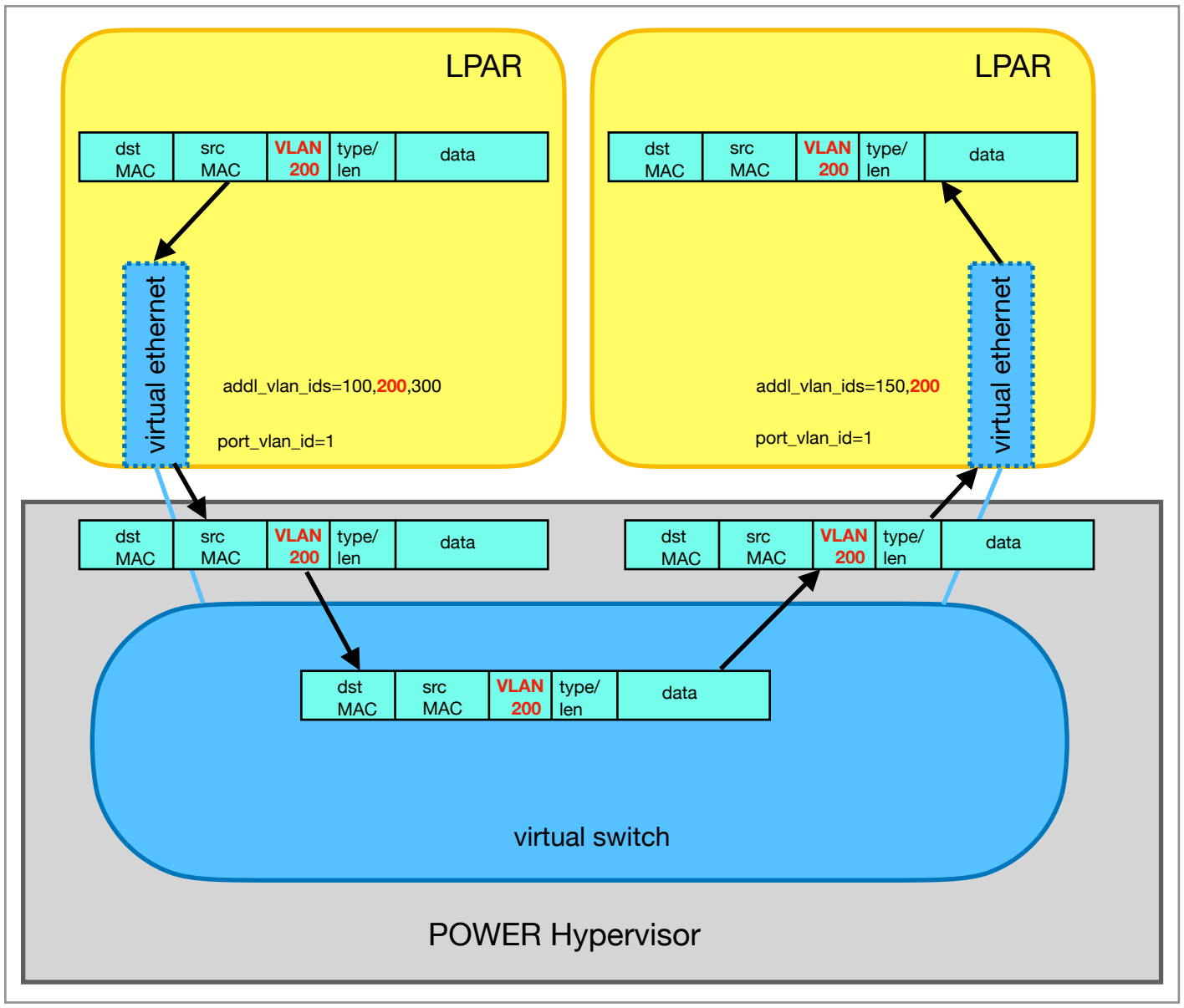


Bild 7.5: Zustellung von tagged Paketen, hier für das VLAN 200.

7.3.2. Hinzufügen eines virtuellen Ethernet Adapters

Soll einer aktiven LPAR ein virtueller Ethernet Adapter hinzugefügt werden, muß die LPAR eine aktive RMC-Verbindung zu einer HMC haben. Dies setzt einen aktiven Ethernet Adapter (physikalisch oder virtuell) voraus. Für den virtuellen Ethernet Adapter wird ein freier virtueller Slot benötigt.

```
$ lpar lsvslot aix22
SLOT  REQ  ADAPTER_TYPE   STATE  DATA
0     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
1     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
5     No   eth            1      PVID=100 VLANS= ETHERNET0 1DC8DB485D1E
10    No   fc/client      1      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
20    No   fc/client      1      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
$
```

Der virtuelle Slot *6* ist bei der LPAR *aix22* noch unbenutzt. Das Hinzufügen eines virtuellen Ethernet Adapters kann mit dem Kommando „*lpar addeth*“ durchgeführt werden. Es muß mindestens die gewünschte virtuelle Slot-Nummer für den Adapter und die gewünschte Port-VLAN-ID angegeben werden:

```
$ lpar addeth aix22 6 900
$
```

Im Beispiel wurde ein virtueller Ethernet Adapter für *aix22* mit der Port-VLAN-ID *900* im Slot *6* angelegt. Spielt die Slot-Nummer keine Rolle, dann kann anstelle einer Nummer auch das Schlüsselwort *auto* angegeben werden, das *LPAR-Tool* vergibt dann automatisch eine freie Slot-Nummer. Der virtuelle Adapter steht sofort zur Verfügung, muß aber erst noch dem Betriebssystem bekannt gemacht werden. Wie das genau geschieht, hängt vom verwendeten Betriebssystem ab. Im Falle von AIX gibt es hierzu das Kommando *cfgmgr*.

Nachdem der virtuelle Ethernet Adapter hinzugefügt wurde und bevor ein Lauf von *cfgmgr* gestartet wird, ist dem AIX Betriebssystem der LPAR *aix22* nur der virtuelle Ethernet Adapter *ent0* bekannt:

```
aix22 # lscfg -l ent*
  ent0             U9009.22A.8991971-V30-C5-T1  Virtual I/O Ethernet Adapter (l-lan)
aix22 #
```

Nach einem Lauf von *cfgmgr* erscheint dann der neu hinzugefügte virtuelle Ethernet Adapter als *ent1*:

```
aix22 # cfgmgr
aix22 # lscfg -l ent*
  ent0             U9009.22A.8991971-V30-C5-T1  Virtual I/O Ethernet Adapter (l-lan)
  ent1             U9009.22A.8991971-V30-C6-T1  Virtual I/O Ethernet Adapter (l-lan)
aix22 #
```

Hinweis: Unter AIX ist anhand des Gerätenamens für einen Ethernet Adapter nicht der Typ erkennbar. Unabhängig davon, ob ein Ethernet Adapter physikalisch oder virtuell oder eine Virtual Function eines SR-IOV Adapters ist, wird immer der Gerätename *ent* mit einer aufsteigenden Instanz-Nummer verwendet.

Soll ein IEEE 802.1q kompatibler virtueller Ethernet Adapter mit zusätzlichen VLAN-IDs angelegt werden, muß die Option „*-i*“ (*IEEE 802.1q compatible adapter*) verwendet werden. Alternativ kann aber auch das Attribut *ieee_virtual_eth=1* angegeben werden. Die zusätzlichen VLAN-IDs werden als kommaseparierte Liste angegeben:

```
$ lpar addeth -i aix22 7 900 100,200,300
$
```

Die Port-VLAN-ID ist die *900*, und die zusätzlichen VLAN-IDs sind *100*, *200* und *300*.

Hat eine LPAR keine aktive RMC-Verbindung oder ist nicht aktiv, dann kann ein virtueller Ethernet Adapter nur einem der Profile der LPAR hinzugefügt werden. Dies ist z.B. immer der Fall, wenn die LPAR gerade neu angelegt wurde und noch nicht installiert ist.

In diesem Fall muß bei den gezeigten Kommandos lediglich die Option „*-p*“ mit einem Profil-Namen verwendet werden. Welche Profile eine LPAR besitzt kann mittels „*lpar lsprof*“ (*list profiles*) einfach herausgefunden werden:

```
$ lpar lsprof aix22
NAME                      MEM_MODE  MEM   PROC_MODE  PROCS  PROC_COMPAT
standard                  ded       7168  ded        2      default
last*valid*configuration  ded       7168  ded        2      default
$
```

(Im Profil mit dem Namen *last*valid*configuration* ist die letzte aktive Konfiguration hinterlegt.)

Die im Profil *standard* definierten virtuellen Adapter lassen sich dann unter Angabe des Profil-Namens mit „*lpar lsvslot*“ anzeigen:

```
$ lpar -p standard lsvslot aix22
SLOT  REQ  ADAPTER_TYPE   DATA
0     Yes  serial/server  remote: (any)/any connect_status= hmc=1
1     Yes  serial/server  remote: (any)/any connect_status= hmc=1
5     No   eth            PVID=100 VLANS= ETHERNET0
6     No   eth            PVID=900 VLANS= ETHERNET0
7     No   eth            IEEE PVID=900 VLANS=100,200,300 ETHERNET0
10    No   fc/client      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
20    No   fc/client      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
$
```

Beim Hinzufügen des Adapters muß lediglich der entsprechende Profil-Name angegeben werden, ansonsten sieht das Kommando genauso aus, wie oben gezeigt:

```
$ lpar -p standard addeth -i aix22 8 950 150,250
$
```

Um den neuen Adapter in Slot *8* verfügbar zu machen, muß die LPAR unter Angabe des Profil-Namens *standard* neu aktiviert werden.

7.3.3. Virtuelle Ethernet Switches

Jedes Power System besitzt per Default den virtuellen Ethernet Switch *ETHERNET0*. Bei Bedarf können zusätzliche virtuelle Ethernet Switches angelegt werden. Besitzt ein Managed System nur eine Anbindung an genau ein externes Netzwerk, dann reicht die Verwendung des Default Switches *ETHERNET0* auch völlig aus.

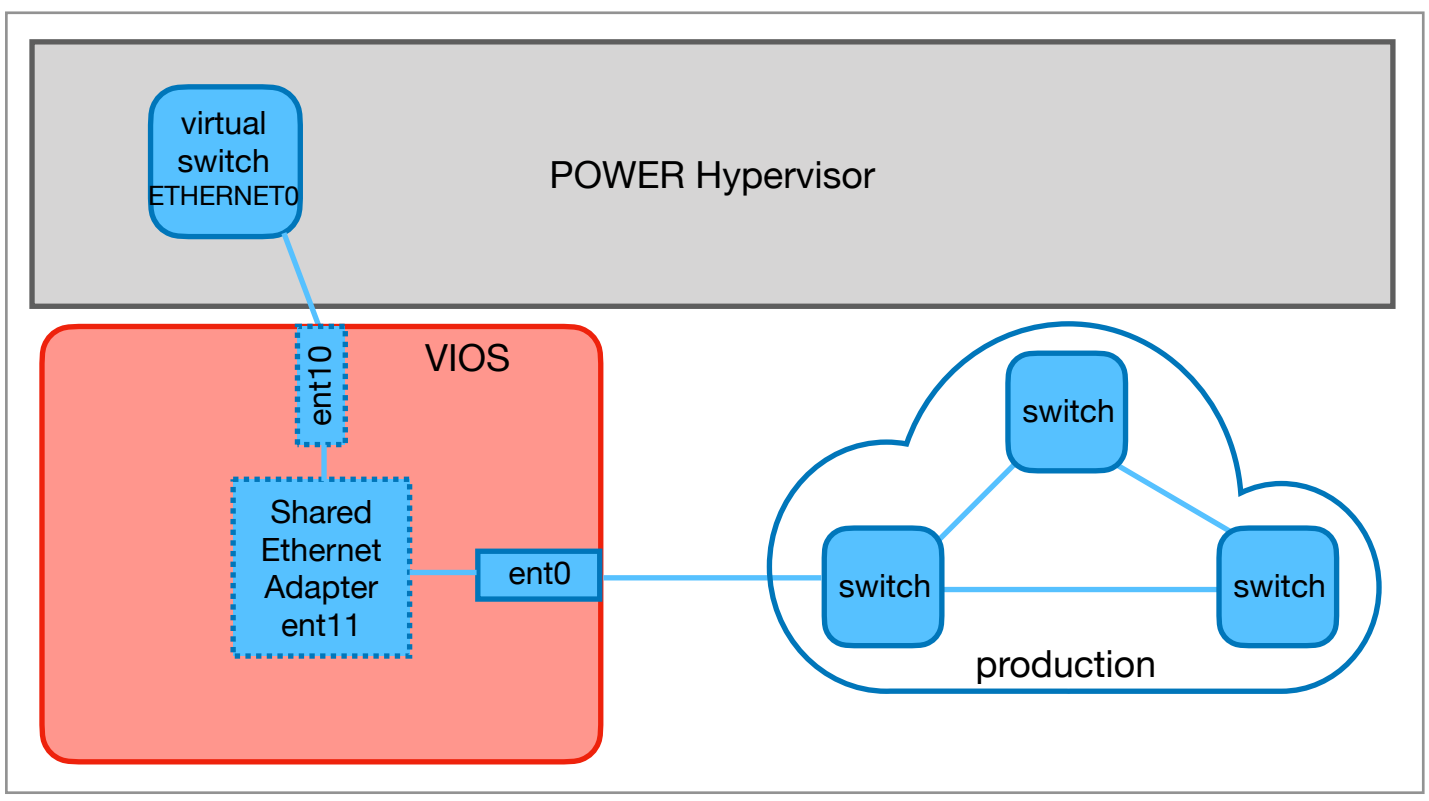


Bild 7.6: ETHERNET0 und ein externes Netzwerk

Das in Bild 7.6 gezeigte Netzwerk wird über den Shared Ethernet Adapter *ent11* und den virtuellen Switch *ETHERNET0* lediglich erweitert.

Gänzlich anders ist die Situation, wenn man ein weiteres unabhängiges Netzwerk an das Managed System anbindet. Bei Verwendung des gleichen virtuellen Ethernet Switches werden die beiden unabhängigen externen Netzwerke

innerhalb des Managed Systems über den gemeinsam genutzten virtuellen Ethernet Switch miteinander verbunden. In Bild 7.7 ist dies skizziert.

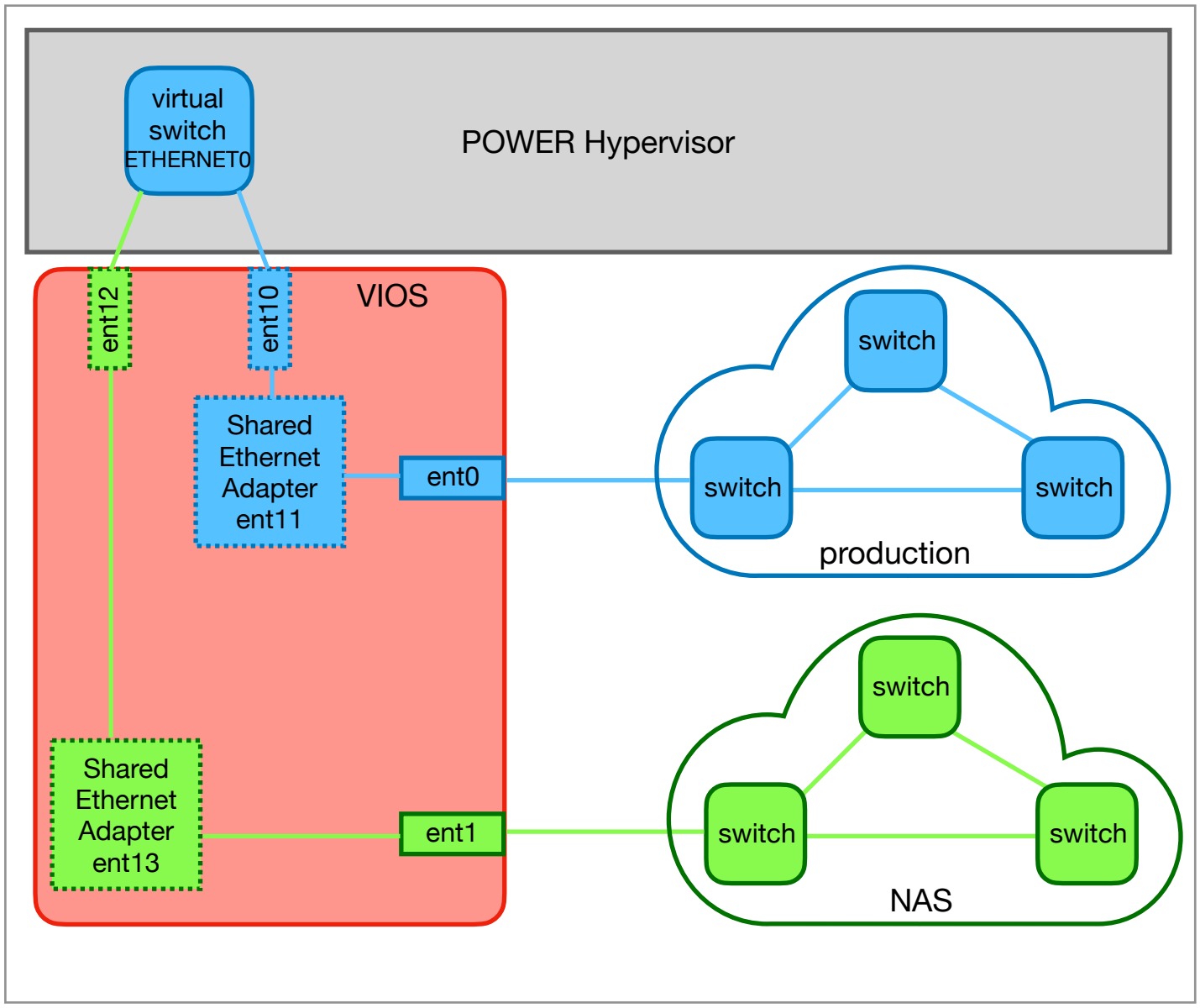


Bild 7.7: Verbinden zweier externer Netzwerke über ETHERNET0

In der Praxis stellt dies häufig kein ernsthaftes Problem dar, da meist unterschiedliche VLAN-IDs in den beiden externen Netzen verwendet werden. Es ist trotzdem unschön, wenn zwei physikalisch separierte Netzwerke den gleichen virtuellen Ethernet Switch in einem Managed System verwenden. Sollte in beiden Netzwerken ein VLAN mit der gleichen VLAN-ID verwendet werden, dann kann innerhalb des Managed Systems nur eines der beiden VLANs verwendet werden. Diese Konfiguration sollte auf jeden Fall vermieden werden und könnte beispielsweise bei einem Security Audit auch beanstandet werden.

Für jedes externe physikalische Netzwerk das an ein Managed System angebunden wird, sollte daher ein eigener virtueller Ethernet Switch verwendet werden. Die externen physikalisch getrennten Netzwerke bleiben dann auch innerhalb des Managed Systems voneinander getrennt. Bild 7.8 zeigt die Trennung mit Hilfe eines zusätzlichen virtuellen Ethernet Switches *ETHNAS*. Mit dieser Konfiguration können beide Netzwerke ohne weiteres gleiche VLAN-IDs verwenden, ohne das dies zu Problemen führt.

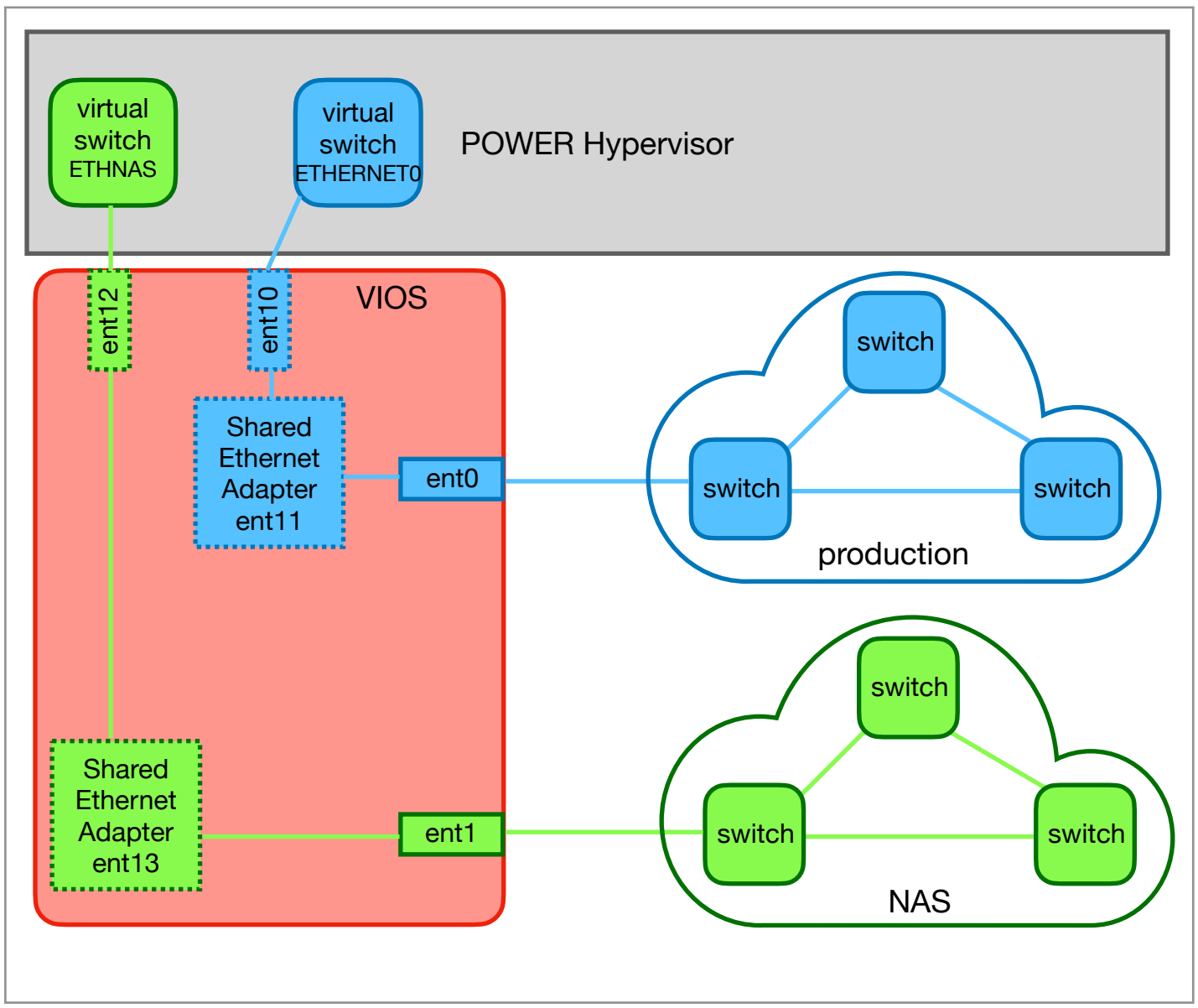


Bild 7.8: Trennung zweier Netzwerke durch Verwendung eines zusätzlichen virtuellen Ethernet Switch.

Die virtuellen Ethernet Switches eines Managed Systems lassen sich mit dem Kommando „*ms lsvswitch*“ (*list virtual switch*) anzeigen:

```
$ ms lsvswitch ms03
NAME  VSWITCH             SWITCH_MODE  VLAN_IDS
ms03  ETHERNET0(Default)  VEB          1,100,150,200,250,300,900,950
$
```

Ein weiterer virtueller Ethernet Switch kann mit dem Kommando „*ms addvswitch*“ (*add virtual switch*) hinzugefügt werden:

```
$ ms addvswitch ms03 ETHNAS
$
```

Ein neuer virtueller Ethernet Switch hat zunächst keine VLAN-IDs zugeordnet:

```
$ ms lsvswitch ms03
NAME  VSWITCH             SWITCH_MODE  VLAN_IDS
ms03  ETHERNET0(Default)  VEB          1,100,150,200,250,300,900,950
ms03  ETHNAS              VEB          none
$
```

Beim Anlegen eines virtuellen Ethernet Adapters kann über die Option „*-s*“ der virtuelle Ethernet Switch angegeben werden, an den der Adapter angebunden sein soll. Per Default ist das der Default Switch mit dem Namen *ETHERNET0*.

```
$ lpar addeth -s ETHNAS aix22 9 700
$
```

Die für den virtuellen Ethernet Adapter angegebene Port-VLAN-ID wird dann auch in der Liste der unterstützten VLAN-IDs des virtuellen Switches *ETHNAS* aufgelistet:

```
$ ms lsvswitch ms03
NAME  VSWITCH              SWITCH_MODE  VLAN_IDS
ms03  ETHERNET0(Default)   VEB          1,100,150,200,250,300,900,950
ms03  ETHNAS               VEB          700
$
```

7.3.4. Virtual Ethernet Bridge Mode (*VEB*)

Jeder virtuelle Ethernet Switch kann in einem von zwei Betriebsarten verwendet werden, dem Virtual Ethernet Bridge Mode (*VEB*) oder dem Virtual Ethernet Port Aggregator Mode (*VEPA*). Standardmäßig wird der *VEB*-Mode verwendet.

Beim Virtual Ethernet Bridge Mode können, wie in Bild 7.2 gezeigt ist, LPARs innerhalb eines Managed Systems über einen virtuellen Ethernet Switch direkt miteinander kommunizieren. Die Netzwerk-Pakete bleiben dabei innerhalb des Managed Systems. Hierdurch ergibt sich eine niedrige Latenz und ein hoher Durchsatz.

7.3.5. Virtual Ethernet Port Aggregator Mode (*VEPA*)

In vielen Umgebungen werden heutzutage Firewalls eingesetzt, welche nur Verbindungen zwischen erlaubten Hosts und Ports zulassen. Nicht-erwünschte Verbindungen werden von den Firewalls blockiert. Damit kann verhindert werden, das jedes System jeden beliebigen Port auf einem Server ansprechen kann. Befinden sich Quell- und Ziel-System als LPARs auf dem gleichen Managed System, so ist allerdings eine Kommunikation über den virtuellen Ethernet Switch unter Umgehung der Firewall möglich. Pakete zwischen LPARs im gleichen Managed System (gleicher virtueller Switch und gleiches VLAN) werden über den virtuellen Switch direkt zugestellt. Diese Pakete gehen nicht ins externe Netzwerk, wo eine Untersuchung der Pakete durch eine Firewall stattfinden könnte.

In einigen Umgebungen gibt es die Anforderung das jeglicher Netzwerk-Verkehr immer über eine Firewall gehen muß. Dies kann mit PowerVM durch den Virtual Port Aggregator Mode (*VEPA*) erreicht werden. Wie in Bild 7.9 zu sehen ist, werden Netz-Pakete immer an den Trunking Port und damit über einen Shared Ethernet Adapter letztlich ins externe Netzwerk weitergeleitet. Die Pakete verlassen damit das Managed System. Im externen Netzwerk können die Pakete dann z.B. durch eine Firewall untersucht werden und müssen dann, falls die Pakete nicht durch die Firewall geblockt werden, zurück an den physikalischen Ethernet Port des Managed Systems geschickt werden, wo sie dann über den Shared Ethernet Adapter und den Trunking Port wieder zum virtuellen Switch kommen und dann dort an die Ziel-LPAR weitergeleitet werden. Eine direkt Kommunikation zwischen zwei LPARs, so wie im *VEB*-Mode ist dann nicht mehr möglich.

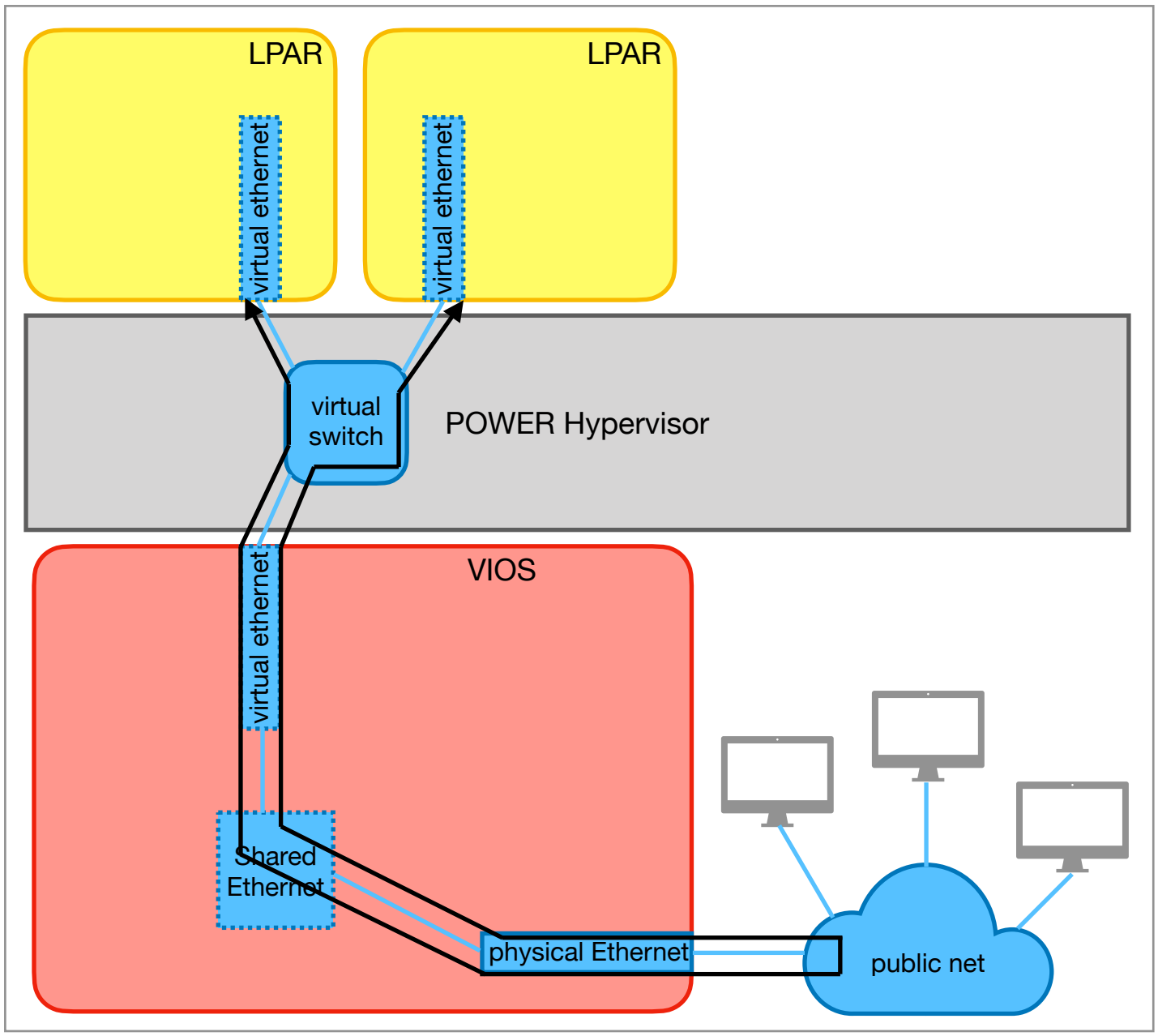


Bild 7.9: Virtueller Ethernet Switch mit VEPA Mode

Die Betriebsart eines virtuellen Ethernet Switches kann mit dem Kommando „*ms chvswitch*“ (*change virtual switch*) geändert werden. Das entsprechende Attribut heißt *switch_mode* und hat die beiden möglichen Werte *VEB* oder *VEPA*:

```
$ ms chvswitch ms03 ETHNAS switch_mode=VEPA
$
```

Allerdings sollte der *VEPA*-Mode nur dann aktiviert werden, wenn die Switch-Konfiguration im externen Netzwerk entsprechend angepasst wurde! Der *VEPA*-Mode funktioniert nur mit Unterstützung der externen Switches.

Der Netzwerk-Durchsatz zwischen den LPARs sinkt bei Verwendung des *VEPA*-Modes.

7.3.6. Virtuelle Netzwerke

Ein sehr nützliches, aber nicht sehr bekanntes, Feature von PowerVM ist die Möglichkeit virtuelle Netzwerke mit einem Namen zu versehen. Jedes VLAN an jedem virtuellen Switch ist dabei ein eigenes virtuelles Netzwerk. Die auf einem Managed System bekannten virtuellen Netzwerke (*vnetwork*) können mittels „*ms lsvnetwork*“ angezeigt werden:

```
$ ms lsvnetwork ms03
```

```
NAME  VNETWORK           VSWITCH    TAGGED  VLAN_ID
ms03  VLAN1-ETHERNET0    ETHERNET0  0       1
ms03  VLAN100-ETHERNET0  ETHERNET0  1       100
ms03  VLAN150-ETHERNET0  ETHERNET0  1       150
ms03  VLAN200-ETHERNET0  ETHERNET0  1       200
ms03  VLAN250-ETHERNET0  ETHERNET0  1       250
ms03  VLAN300-ETHERNET0  ETHERNET0  1       300
ms03  VLAN900-ETHERNET0  ETHERNET0  1       900
ms03  VLAN950-ETHERNET0  ETHERNET0  1       950
ms03  VLAN700-ETHNAS     ETHNAS     0       700
$
```

Beim Anlegen eins virtuellen Ethernet Adapters werden automatisch, in soweit noch nicht vorhanden, virtuelle Netzwerke angelegt. Per Default wird als Name die Zeichenkette „*VLAN*“, gefolgt von der VLAN-ID, einem Minuszeichen und dem Namen des virtuellen Ethernet Switches verwendet. Der Name des virtuellen Netzwerks lässt sich auf eine beliebige Zeichenkette ändern, hierzu dient das Kommando „*ms chvnetwork*“ (*change virtual network*) mit dem Attribut *new_name*:

```
$ ms chvnetwork ms03 VLAN900-ETHERNET0 new_name=ManagementNet
$
```

Gut gewählte Namen für die virtuellen Netzwerke (welche letztendlich VLANs sind), können die Administration deutlich vereinfachen:

```
$ ms lsvnetwork ms03
NAME  VNETWORK          VSWITCH    TAGGED  VLAN_ID
ms03  VLAN1-ETHERNET0   ETHERNET0  0       1
ms03  ProductionDB      ETHERNET0  1       100
ms03  TestDB            ETHERNET0  1       150
ms03  Websphere-Prod    ETHERNET0  1       200
ms03  Websphere-Test    ETHERNET0  1       250
ms03  MQ-Network        ETHERNET0  1       300
ms03  ManagementNet     ETHERNET0  1       900
ms03  NIM-Network       ETHERNET0  1       950
ms03  Netapp-Net        ETHNAS     0       700
$
```

Tip: Werden auf externen Switches Namen für die verschiedenen VLANs verwendet, sollte man auf den Managed Systems für die VLANs die gleichen Namen verwenden.

Ein weiterer Vorteil der virtuellen Netzwerke ist, das die Namen beim Anlegen von virtuellen Ethernet-Adaptern direkt angegeben werden können, anstelle von Switch-Name und VLAN-IDs:

```
$ lpar addeth aix22 11 Netapp-Net
$
```

Auch zusätzliche (*tagged*) VLANs können über den virtuellen Netzwerk-Namen angegeben werden:

```
$ lpar -p standard addeth -i aix22 12 NIM-Network TestDB,Websphere-Test
$
```

Wird ein virtuelles Netzwerk aktuell nicht mehr verwendet, dann wird es nicht automatisch entfernt. Sollte ein virtuelles Netzwerk nicht mehr gebraucht werden, kann es mittels „*ms rmvnetwork*“ (*remove virtual network*) entfernt werden:

```
$ ms rmvnetwork ms03 MQ-Network
$
```

Natürlich können virtuelle Netzwerke auch manuell angelegt werden, was in der Regel aber nicht notwendig ist. Zum Anlegen mit dem Kommando „*ms addvnetwork*“ (*add virtual network*), muß neben dem virtuellen Switch die VLAN-ID des Netzes angegeben werden:

```
$ ms addvnetwork ms03 testnet 999 ETHERNET0
$
```

Das virtuelle Netzwerk wird als *tagged* Netzwerk angelegt:

```
$ ms lsvnetwork -s vnetwork=testnet ms03
NAME  VNETWORK  VSWITCH    TAGGED  VLAN_ID
ms03  testnet   ETHERNET0  1       999
$
```

Soll das virtuelle Netzwerk als *untagged* angelegt werden, kann die Option „*-u*“ (*untagged*) verwendet werden.

7.3.7. Einem Adapter VLANs hinzufügen/wegnehmen

Bei IEEE 802.1q kompatiblen virtuellen Ethernet Adaptern können jederzeit VLAN-IDs hinzugefügt oder weggenommen werden. Die Port-VLAN-ID kann bei einer aktiven LPAR allerdings nicht geändert werden, dies geht nur über eine Änderung im Profil und Neu-Aktivierung der LPAR mit dem geänderten Profil.

Bei einer LPAR mit aktiver RMC-Verbindung können VLANs mit dem Kommando „*lpar addvlan*“ (*add VLAN*) jederzeit hinzugefügt werden. Dabei können die VLANs entweder über die VLAN-ID oder über den virtuellen Netzwerk-Namen angegeben werden:

```
$ lpar addvlan aix22 5 testnet
$
```

Das Wegnehmen eines VLANs geschieht analog durch das Kommando „*lpar rmvlan*“ (*remove VLAN*):

```
$ lpar rmvlan aix22 5 testnet
$
```

Ist ein virtueller Ethernet Adapter nicht IEEE 802.1q kompatibel, dann unterstützt er nur *untagged* Pakete. Man kann dann keine VLANs hinzufügen oder wegnehmen. Allerdings lässt sich die Eigenschaft der IEEE 802.1q Kompatibilität dynamisch zur Laufzeit ändern. Die Änderung kann mit dem Kommando „*lpar cheth*“ (*change ethernet*) mit der Option „*-i*“ (IEEE 802.1q kompatibel) durchgeführt werden:

```
$ lpar cheth -i aix22 6
$
```

Danach können mit „*lpar addvlan*“ weitere VLANs hinzugefügt werden.

Ist eine LPAR ausgeschaltet, oder hat keine aktive RMC Verbindung, können VLANs nur in einem Profil der LPAR hinzugefügt, oder weggenommen werden. Dazu wird bei „*lpar addvlan*“ bzw. „*lpar rmvlan*“ einfach zusätzlich die Option „*-p*“ mit einem Profil-Namen angegeben.

```
$ lpar -p standard rmvlan aix22 12 TestDB
$
```

7.3.8. Ändern von Attributen eines virtuellen Ethernet Adapters

Ein virtueller Ethernet Adapter besitzt die folgenden Attribute:

```
$ lpar help cheth
USAGE:
   lpar [-h <hmc>] [-m <ms>] [-p <profile>] cheth [-d] [-f] [-i] [-I] [-l <detail_level>]
[-q <qos_priority>] [-r] [-R] [-s <vswitch>] [-t <trunk_priority>] [-w <wait_time>] [-v]
<lpar> <slot> [<attributes> ...]

…

Valid attributes for DLPAR and profile:
   ieee_virtual_eth - IEEE 802.1Q compatible
      0 : no
      1 : yes
   addl_vlan_ids - comma separated list of additional VLANs
   qos_priority - QoS priority
      none : no QoS
      0-7 : priority to use

Valid attributes for profile only:
   port_vlan_id - VLAN-ID for untagged packets
   is_trunk - trunking port (VIOS only)
      0 : no
      1 : yes
   trunk_priority - priority of the trunk adapter
      integer betwen 1-15
   vswitch - virtual switch to connect adapter to
   mac_addr - MAC address to use
   allowed_os_mac_addrs - allowed MAC addresses
      none : OS defined MAC addresses are not allowed
      <mac>[,...] : comma separated list of 1-4 MAC addresses
      all : OS defined MAC addresses are allowed

…
$
```

Einige Attribute, wie *ieee_virtual_eth*, *addl_vlan_ids* und *qos_priority*, können online geändert werden. Alle anderen Attribute können jeweils nur im Profil einer LPAR geändert werden, z.B. *port_vlan_id* oder *vswitch*.

Das Kommando zum Ändern von virtuellen Ethernet Attributen ist „*lpar cheth*“.

7.3.9. Wegnehmen eines virtuellen Ethernet Adapters

Ein virtueller Ethernet Adapter kann natürlich auch wieder entfernt werden. Ist eine LPAR aktiv, ist dabei aber zu beachten, das nur ein virtueller Ethernet Adapter über den die RMC-Verbindung zu den HMCs nicht läuft, weggenommen werden kann. Außerdem darf der betreffende virtuelle Ethernet Adapter nicht mehr in Benutzung sein und das zugehörige Ethernet Device muß aus dem Kernel entfernt werden.

Sind diese Voraussetzungen erfüllt, kann der virtuelle Ethernet Adapter mit „*lpar rmeth*“ (*remove ethernet*) unter Angabe der Slot-Nummer entfernt werden:

```
$ lpar rmeth aix22 6
$
```

7.4. Virtual FC

Eine Möglichkeit für die Virtualisierung von Storage unter PowerVM ist die Verwendung von virtuellen FC Adaptern. Dabei ist ein virtueller FC-Client Adapter über den POWER Hypervisor mit einem virtuellen FC-Server Adapter auf einem Virtual-I/O-Server verbunden, wie in Bild 7.10 gezeigt. Auf dem Virtual-I/O-Server wird der virtuelle FC-Server Adapter dann mit einem der physikalischen FC-Ports verbunden (Mapping). Jeder der verbundenen virtuellen FC-Server Adapter kann dabei einen eigenen Login in die FC-Fabric durchführen. Jeder virtuelle FC-Server Adapter bekommt dabei eine eigene 24-bit FC-Adresse zugewiesen.

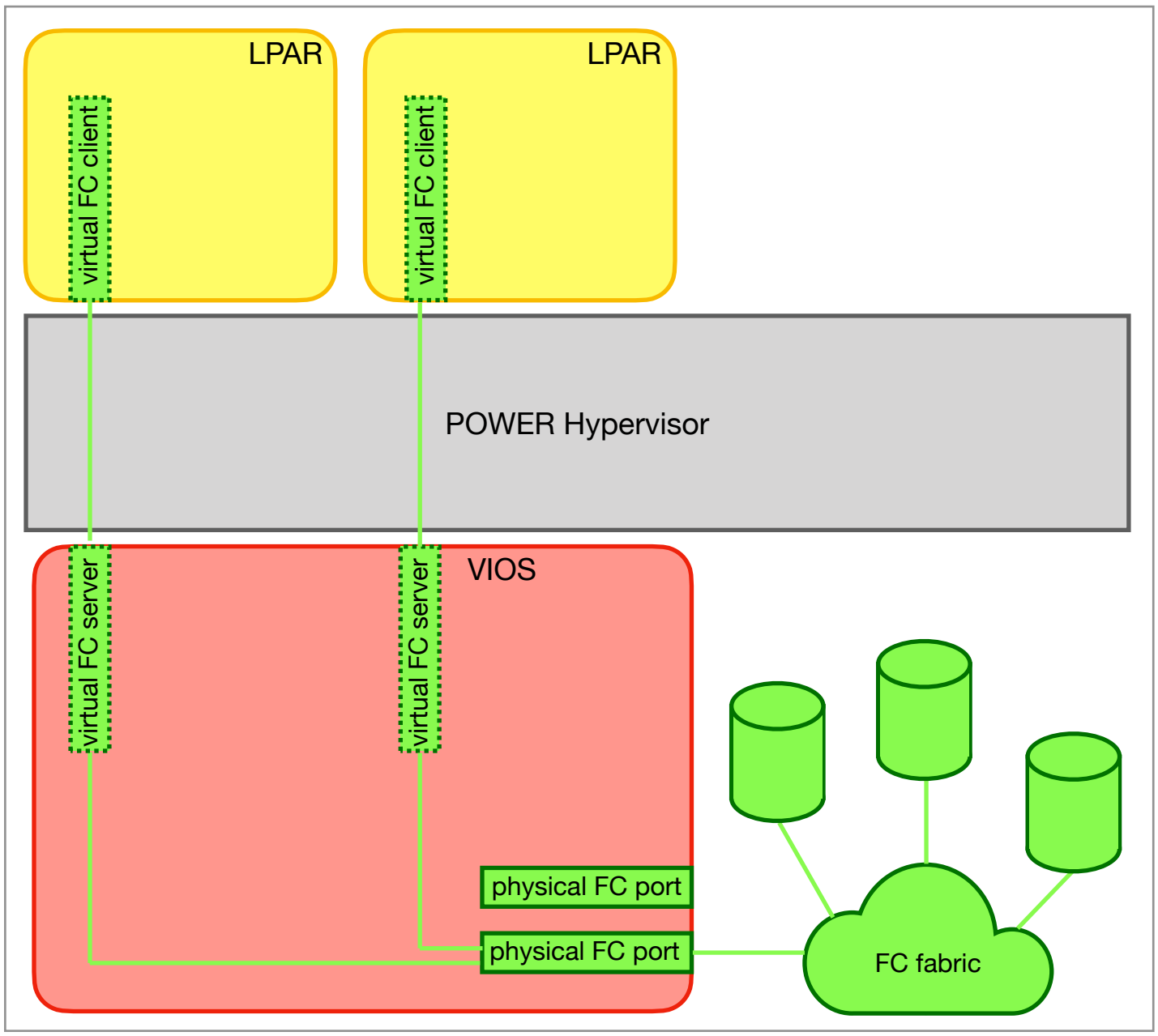


Bild 7.10: Kommunikationspfad virtueller FC Client Adapter zur SAN-LUN.

Der Vorteil von Virtual FC besteht darin, das jeder virtuelle FC-Client Adapter einen eigenen *N_Port* besitzt und damit direkt mit dem Storage in der FC-Fabric kommunizieren kann. Die Storage-LUNs können dem virtuellen FC -Client Adapter direkt zugewiesen. Der Virtual-I/O-Server selbst sieht normalerweise die Storage-LUNs der virtuellen FC Clients nicht. Das macht die Administration deutlicher einfacher als bei virtuellem SCSI, wo jede Storage-LUN auf dem Virtual-I/O-Server auf einen virtuellen SCSI Server Adapter gemappt werden muss (siehe nächstes Kapitel).

Bevor ein virtueller FC Adapter angelegt und gemappt wird, sieht die Situation auf einem Virtual-I/O-Server so wie in Bild 7.11 dargestellt aus. Der physikalische FC-Port ist an eine FC-Fabric angeschlossen und konfiguriert daher einen *N_Port*. Der physikalische FC-Port loggt sich in die Fabric ein (*FLOGI*) und bekommt die eindeutige *N_Port ID 8c8240* zugewiesen. Danach registriert der FC-Port seine *WWPN* (hier *10:00:00:10:9b:ab:01:02*) beim Simple Name Server (*SNS*) der Fabric (*PLOGI*). Danach kann der Virtual-I/O-Server über das Gerät *fcs0* mit anderen *N_Port*s in der Fabric kommunizieren.

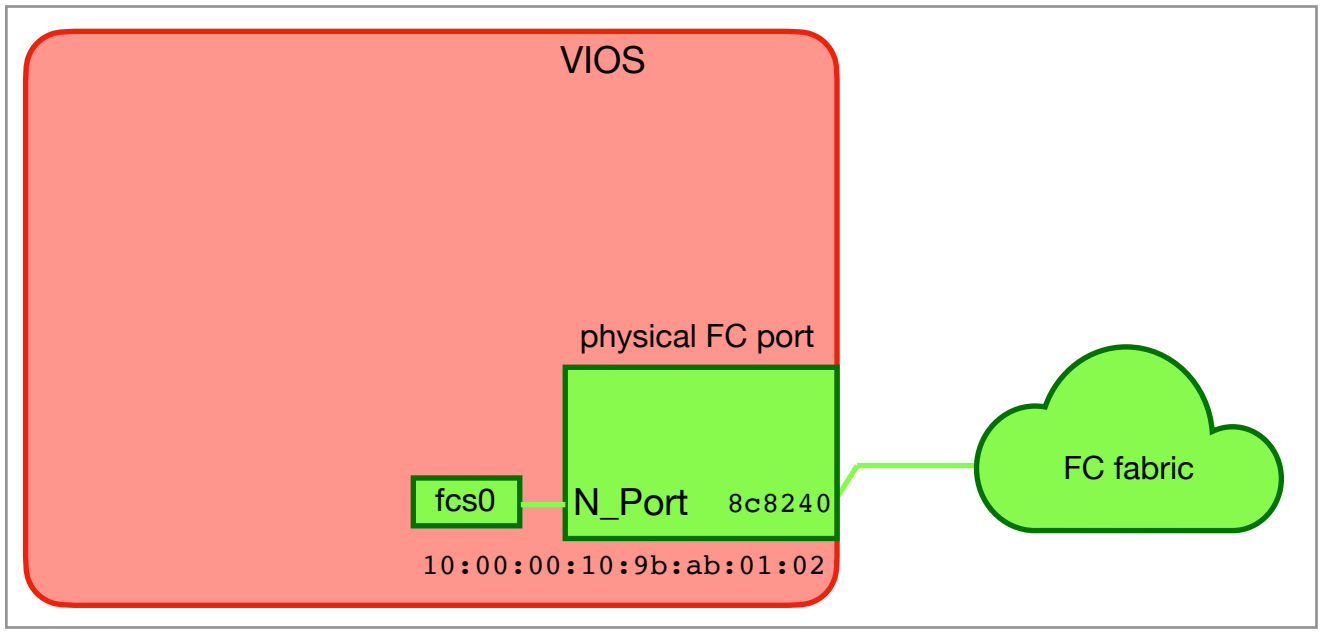


Bild 7.11: Physikalischer FC-Port ohne Virtual FC und NPIV

N_Port-ID Virtualisierung oder kurz *NPIV* ist eine Erweiterung des FC-Standards und erlaubt das sich über einen physikalischen FC-Port mehr als ein *N_Port* in die Fabric einloggen kann. Im Prinzip gab es diese Möglichkeit schon immer, allerdings nur im Zusammenhang mit FC Arbitrated Loop (*FC-AL*) und Fabrics. Mit *NPIV* können mehrere Client LPARs einen physikalischen FC-Port gemeinsam verwenden. Jeder Client hat dabei seinen eigenen eindeutigen *N_Port*.

In Bild 7.12 ist die Situation mit 2 virtuellen FC-Client Adaptern gezeigt. Jeder der Client Adapter hat eine eindeutige *WWPN*. Diese wird von PowerVM beim Erzeugen des virtuellen FC-Client Adapters zugewiesen (um Live-Partition Mobility unterstützen zu können, werden immer 2 *WWPN*s zugewiesen, wobei nur eine der beiden *WWPN*s aktiv ist). Jeder virtuelle FC-Client Adapter benötigt auf einem Virtual-I/O-Server einen Partner Adapter, den virtuellen FC-Server Adapter (oder auch *vfchost*). Dem virtuellen FC-Server Adapter muß auf dem Virtual-I/O-Server einer der physikalischen FC-Ports zugeordnet werden. Ist die Client-LPAR aktiv, dann loggt sich der virtuelle FC-Server Adapter in die Fabric ein (*FDISC*) und bekommt eine eindeutige *N_Port ID* zugewiesen. Im Bild ist das für den blauen Adapter die *8c8268* und für den roten Adapter die *8c8262*. Danach registriert der blaue Adapter seine Client-*WWPN* (hier *c0:50:76:07:12:cd:00:16*) beim Simple Name Server (*SNS*) der Fabric (*PLOGI*). Das gleiche macht der rote Adapter für seine Client-*WWPN* (hier *c0:50:76:07:12:cd:00:09*). Damit haben dann beide virtuellen FC-Client Adapter jeweils einen *N_Port* mit einer eindeutigen 24-bit ID und können damit mit anderen *N_Ports* in der Fabric kommunizieren.

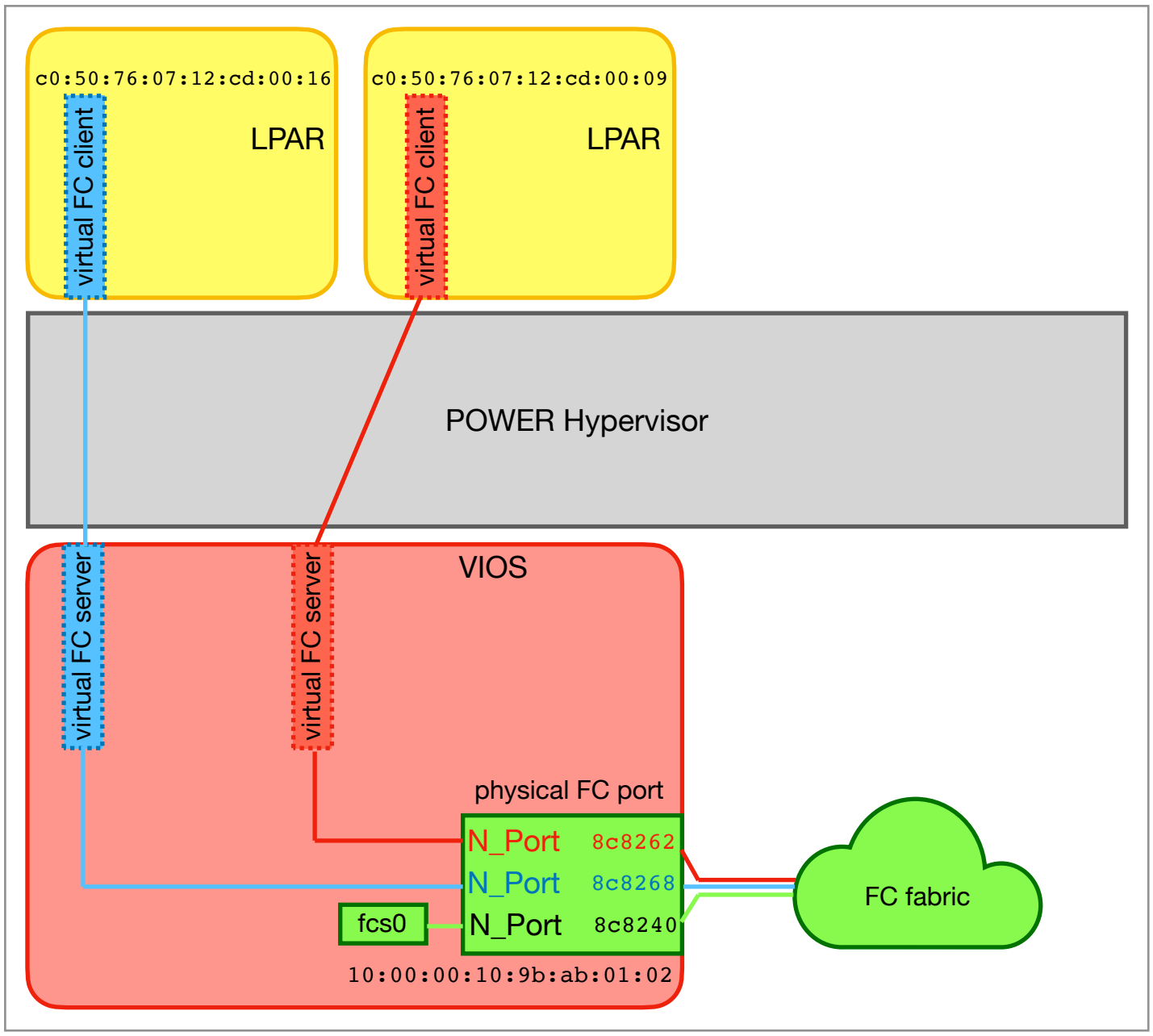


Bild 7.12: Physikalischer FC-Port mit Virtual FC und NPIV

Die zu Daten werden natürlich zwischen virtuellem FC-Client Adapter und virtuellem FC-Server Adapter nicht vom Hypervisor kopiert, das würde zuviel Performance kosten. Der Hypervisor gibt lediglich die physikalische Speicheradresse weiter, an der die Daten stehen und der physikalische FC-Port verwendet dann DMA (Direct Memory Access) um auf diese Daten dann zuzugreifen.

7.4.1. NPIV-fähige FC-Adapter

Damit NPIV, und damit Virtual FC, verwendet werden kann, müssen die physikalischen FC-Adapter NPIV unterstützen. Das ist aber mittlerweile bei allen FC-Adaptern für Power Systeme der Fall. Welche FC-Adapter eines Virtual-I/O-Servers NPIV unterstützen kann sehr leicht mit dem Kommando "*vios lsnports*“ herausgefunden werden:

```
$ vios lsnports ms03-vio1
NAME  PHYSLOC                          FABRIC  TPORTS  APORTS  SWWPNS  AWWPNS  LABEL
fcs0  U78AA.001.VYRGU0Q-P1-C5-T1       1       64      62      2048    2023    Fabric1
fcs1  U78AA.001.VYRGU0Q-P1-C5-T2       1       64      53      2048    1977    Fabric1
fcs2  U78AA.001.VYRGU0Q-P1-C5-T3       1       64      62      2048    2023    Fabric2
fcs4  U78AA.001.VYRGU0Q-P1-C1-C1-T1    1       64      60      2032    2007    Fabric2
fcs5  U78AA.001.VYRGU0Q-P1-C1-C1-T2    1       64      58      2032    2018    Fabric1
fcs6  U78AA.001.VYRGU0Q-P1-C1-C1-T3    1       64      58      2032    2007    Fabric2
$
```

Es werden nur NPIV-fähige FC-Ports aufgelistet. Die Spalte *TPORTS* (*Target Ports*) gibt an, wieviele N_Port IDs maximal vom physikalischen FC-Port unterstützt werden, die Spalte *APORTS* (*Available Ports*) gib an, wieviele davon aktuell noch verfügbar sind. Alle aufgelisteten FC-Ports unterstützen bis zu 64 N_Port IDs. Der FC-Port *fcs0* verfügt noch über 62 nicht benutzte N_Port IDs, damit sind also aktuell 2 der 64 möglichen N_Port IDs in Verwendung. Jeder virtuelle FC Adapter benötigt 2 eindeutige WWPNs (eine der beiden WWPNs wird für Live Partition Mobility verwendet). Wieviele WWPNs ein physikalischer FC-Port unterstützt, steht in der Spalte *SWWPNS* (*Supported WWPNs*), wieviel davon noch verfügbar sind, in der Spalte *AWWPNS* (*Available WWPNs*). In der Spalte *LABEL* wird das Attribut *label* des physikalischen FC-Ports angezeigt. Dieses wird aber erst ab *IOS 3.1.1* unterstützt. Ist die *IOS* Version kleiner *3.1.1*, dann wird die Zeichenkette „*not_supported*“ angezeigt. Das *label* Attribut kann vom Administrator auf eine beliebige Zeichenkette gesetzt werden. Ein sinnvoller Wert ist der Name der FC-Fabric an welche der FC-Port angeschlossen ist. Das *label* Attribut kann mit dem Kommando „*vios chdev*“ gesetzt werden:

```
$ vios chdev -P ms03-vio1 fcs0 label=Fabric1
$
```

Der angegebene Wert wird in der ODM auf dem Virtual-I/O-Server abgespeichert.

Tipp: Für alle physikalischen FC-Ports sollte das *label* Attribut auf den Namen der angebundenen FC-Fabric gesetzt werden. Das muß nur einmal gemacht werden, danach kann jederzeit für jeden FC-Port mühelos festgestellt werden zu welcher Fabric der Port gehört. Das Label kann auch beim Mapping von virtuellen FC Adaptern verwendet werden.

Damit der POWER Hypervisor für jeden neuen virtuellen FC-Adapter eindeutige WWPNs vergeben kann, hat jedes Managed System einen Bereich von eindeutigen WWPNs reserviert (standardmäßig sind 65536 WWPNs reserviert). Alle diese reservierten WWPNs haben einen gemeinsamen Präfix (Teil der WWPN). Welcher Präfix dies für ein Managed System ist, und wieviele der reservierten WWPNs noch verfügbar sind, lässt sich mit dem Kommando „*ms lsfc*“ anzeigen:

```
$ ms lsfc ms03
NAME  WWPN_PREFIX   NUM_WWPNS_REMAINING
ms11  C05076030ABA  65226
$
```

Der Präfix (und damit die WWPNs) fängt immer mit *C05076* an. Dies ist eine Kombination aus „*C*“ und dem *OUI* „*005076*“ von IBM. Dabei deutet „*C*“ daraufhin das es sich um eine lokal vergebene WWPN handelt.

Anhand des WWPN-Präfix lässt sich auch nach zahlreichen LPM-Verschiebungen immer noch erkennen, auf welchem Managed System ein virtueller FC-Adapter einer LPAR ursprünglich erzeugt wurde.

7.4.2. Hinzufügen eines virtuellen FC Adapters

Soll einer aktiven LPAR ein virtueller FC Adapter hinzugefügt werden, muß die LPAR eine aktive RMC-Verbindung zu einer HMC haben. Für den virtuellen FC Adapter wird ein freier virtueller Slot benötigt. Da ein virtueller FC Client Adapter immer mit einem virtuellen FC Server Adapter auf einem Virtual-I/O-Server zusammen arbeitet, muß für jeden Client-Adapter immer auch ein Server-Adapter auf einem Virtual-I/O-Server angelegt werden. Auch auf dem Virtual-I/O-Server benötigt man dazu einen freien virtuellen Slot.

Welche virtuellen Slots auf der Client-LPAR noch verfügbar sind, lässt sich wiederum mit „*lpar lsvslot*“ (*list virtual slots*) ermitteln:

```
$ lpar lsvslot aix22
SLOT  REQ  ADAPTER_TYPE   STATE  DATA
0     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
1     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
5     No   eth            1      PVID=100 VLANS= ETHERNET0 1DC8DB485D1E
10    No   fc/client      1      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
20    No   fc/client      1      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
$
```

Hier ist beispielsweise der virtuelle Slot *11* noch verfügbar.

Die verfügbaren virtuellen Slots auf einem Virtual-I/O-Server lassen sich auf die gleiche Weise ermitteln, hier für *ms03-vio1*:

```
$ lpar lsvslot ms03-vio1
SLOT  REQ  ADAPTER_TYPE   STATE  DATA
0     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
1     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
5     No   fc/server      1      remote: aix22(5)/10
11    No   eth            1      PVID=900 VLANS= ETHERNET0 1DC8DB485D0B
20    No   eth            1      PVID=1 VLANS= ETHCTRL 1DC8DB485D14
21    No   eth            1      TRUNK(1) IEEE PVID=1 VLANS=100,200 ETHERNET0 1DC8DB485D15
22    No   eth            1      TRUNK(1) IEEE PVID=2 VLANS=150,250 ETHERNET0 1DC8DB485D16
125   No   fc/server      1      remote: aixsap01(9)/10
181   No   fc/server      1      remote: aixdbp02(11)/10
182   No   fc/server      1      remote: aixdbi02(13)/10
$
```

Hier wäre unter anderem der virtuelle Slot *111* verfügbar.

Um einen virtuellen FC Client Adapter für eine LPAR anzulegen, dient das Kommando „*lpar addfc*“ (*add virtual fc adapter*). Dabei muß neben der Client LPAR auch der Virtual-I/O-Server für den virtuellen FC Server Adapter angegeben werden:

```
$ lpar addfc aix22 11 ms03-vio1 111
aix22 slot 11 c05076030aba000e,c05076030aba000f -> ms03-vio1 slot 111 added by DLPAR
operation
aix22 slot 11 c05076030aba000e,c05076030aba000f -> ms03-vio1 slot 111 added to current
profile (standard)
ms03-vio1 slot 111 -> aix22 slot 11 added by DLPAR operation
$
```

Die Ausgabe des Kommandos zeigt, daß zunächst ein virtueller FC Client Adapter per DLPAR-Operation hinzugefügt wird. Als nächstes wird das aktuelle Profil der LPAR angepasst und anschließend wird der zugehörige FC Server Adapter auf dem Virtual-I/O-Server angelegt. Das Kommando legt als automatisch beide Adapter an, auf Wunsch kann dies durch Verwendung entsprechender Optionen („*-c*“ *client adapter only*) verhindert werden.

Das Heraussuchen einer freien virtuellen Slot-Nummer auf dem Virtual-I/O-Server kann in größeren Umgebungen mühsam sein, da ein Virtual-I/O-Server leicht mehrere hundert virtuelle Adapter besitzen kann. Die virtuelle Slot-Nummer auf dem Virtual-I/O-Server spielt im Prinzip aber gar keine Rolle. Außerdem gibt es keine Garantie dafür, das bei einer LPM-Operation die virtuelle Slot-Nummer auf dem Virtual-I/O-Server gleich bleibt. (Der virtuelle Slot

111 z.B. könnte ja auf dem Ziel Virtual-I/O-Server schon in Verwendung sein.) Lässt man die Angabe der virtuellen Slot-Nummer auf dem Virtual-I/O-Server im Kommando „*lpar addfc*“ einfach weg, dann wird automatisch eine freie Slot-Nummer auf dem Virtual-I/O-Server ermittelt:

```
$ lpar addfc aix22 11 ms03-vio1
aix22 slot 11 c05076030aba0010,c05076030aba0011 -> ms03-vio1 slot 38 added by DLPAR
operation
aix22 slot 11 c05076030aba0010,c05076030aba0011 -> ms03-vio1 slot 38 added to current
profile (standard)
ms03-vio1 slot 38 -> aix22 slot 11 added by DLPAR operation
$
```

Hier wurde beispielsweise die Slot-Nummer *38* auf dem Virtual-I/O-Server verwendet.

Möchte man auch auf der Client-LPAR die Slot-Nummer nicht vorgeben, kann man diese ebenfalls weglassen. Es wird dann auch für den Client eine freie virtuelle Slot-Nummer für den virtuellen FC Client Adapter ermittelt. Normalerweise ist das die erste freie Slot-Nummer. Dies lässt sich aber konfigurieren.

Hat eine LPAR keine aktive RMC-Verbindung oder ist nicht aktiv, dann kann ein virtueller FC Adapter nur einem der Profile der LPAR hinzugefügt werden. Dies ist z.B. immer der Fall, wenn die LPAR gerade neu angelegt wurde und noch nicht installiert ist.

In diesem Fall muß bei den gezeigten Kommandos lediglich die Option „*-p*“ mit einem Profil-Namen verwendet werden. Welche Profile eine LPAR besitzt kann mittels „*lpar lsprof*“ (*list profiles*) einfach herausgefunden werden:

```
$ lpar lsprof aix22
NAME                      MEM_MODE  MEM   PROC_MODE  PROCS  PROC_COMPAT
standard                  ded       7168  ded        2      default
last*valid*configuration  ded       7168  ded        2      default
$
```

(Im Profil mit dem Namen *last*valid*configuration* ist die letzte aktive Konfiguration hinterlegt.)

Die im Profil *standard* definierten virtuellen Adapter lassen sich dann unter Angabe des Profil-Namens mit „*lpar lsvslot*“ anzeigen:

```
$ lpar -p standard lsvslot aix22
SLOT  REQ  ADAPTER_TYPE   DATA
0     Yes  serial/server  remote: (any)/any connect_status= hmc=1
1     Yes  serial/server  remote: (any)/any connect_status= hmc=1
5     No   eth            PVID=100 VLANS= ETHERNET0
10    No   fc/client      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
20    No   fc/client      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
$
```

Beim Hinzufügen des Adapters muß lediglich der entsprechende Profil-Name angegeben werden, ansonsten sieht das Kommando genauso aus, wie oben gezeigt:

```
$ lpar -p standard addfc aix22 11 ms03-vio1
aix22 slot 11 c05076030aba0010,c05076030aba0011 -> ms03-vio1 slot 38 added to current
profile (standard)
ms03-vio1 slot 38 -> aix22 slot 11 added by DLPAR operation
$
```

Um den neuen Adapter in Slot *11* verfügbar zu machen, muß die LPAR unter Angabe des Profil-Namens *standard* neu aktiviert werden.

7.4.3. Zuordnen eines physikalischen FC-Ports (Mapping)

Unter AIX wird für den Gerätenamen eines virtuellen FC Client-Adapter der Standard-Name für FC-Adapter *fcs* verwendet. Nach dem Hinzufügen eines virtuellen FC Client-Adapters zu einer LPAR wird das zugehörige neue Gerät nicht automatisch vom Betriebssystem erkannt.

```
aix22 # lscfg -l fcs*
  fcs0             U9009.22A.679F95F-V5-C10-T1  Virtual Fibre Channel Client Adapter
  fcs1             U9009.22A.679F95F-V5-C20-T1  Virtual Fibre Channel Client Adapter
aix22 #
```

Die Ausgabe zeigt, daß nur die beiden virtuellen FC Client-Adapter in Slot *10* und *20* dem Betriebssystem bekannt sind. Der oben angelegte Adapter in Slot *11* ist dem Betriebssystem noch nicht bekannt.

Nach dem Hinzufügen eines virtuellen FC Client-Adapters in einer Client-LPAR, muß das Betriebssystem der Client-LPAR den neuen Client-Adapter erst erkennen. Im Falle von AIX als Betriebssystem muß dazu der Config-Manager *cfgmgr* gestartet werden:

```
aix22 # cfgmgr -v
cfgmgr is running in phase 2
----------------
…
----------------
Attempting to configure device 'fcs2'
Time: 0 LEDS: 0x2701
Invoking /usr/lib/methods/cfg_vfc -l fcs2
Number of running methods: 5
----------------
…
----------------
Attempting to configure device 'fscsi2'
Time: 1 LEDS: 0x2700
Invoking /usr/lib/methods/cfgefscsi -l fscsi2
Number of running methods: 1
----------------
Completed method for: fscsi2, Elapsed time = 19
Return code = 61
*** no stdout ****
*** no stderr ****
Method error (/usr/lib/methods/cfgefscsi -l fscsi2 ):
        0514-061 Cannot find a child device.
----------------
…
aix22 #
```

Die Ausgabe zeigt das der Adapter *fcs2* erkannt wurde. Allerdings meldet das zugehörige Protokoll-Gerät *fscsi2* den Fehler „*0514-061 Cannot find a child device*.“. Dies liegt daran, daß zwar der virtuelle FC Client- und virtuelle FC Server-Adapter angelegt wurden, aber der virtuelle FC Server-Adapter noch keinem physikalischen FC-Port zugeordnet wurde. Es fehlt damit die Anbindung an eine FC-Fabric, wie Bild 7.13 zeigt.

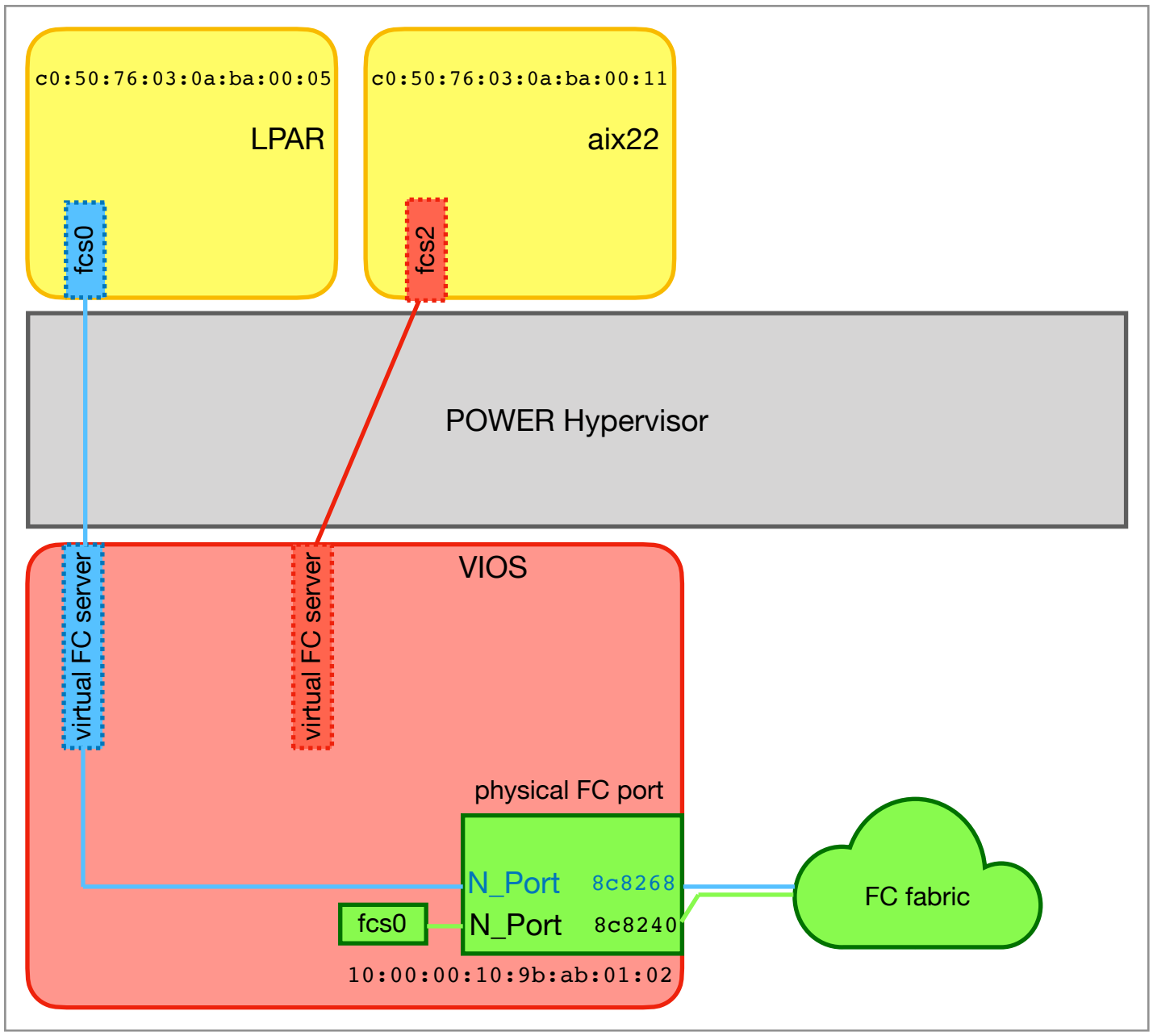


Bild 7.13: Für die LPAR *aix22* fehlt das Mapping auf einen physikalischen FC-Port.

Bei der linken LPAR besitzt der virtuelle FC Client-Adapter *fcs0* (blau) über den zugehörigen virtuellen FC Server-Adapter eine Anbindung an den physikalischen FC-Port *fcs0* auf dem Virtual-I/O-Server und damit eine Anbindung an eine FC-Fabric. Diese Zuordnung muß für den neuen virtuellen FC Server-Adapter (rot) erst noch gemacht werden.

Die bestehenden Zuordnungen (Mappings) auf einem Virtual-I/O-Server können mit dem Kommando „*vios lsnpiv*“ (*list NPIV mappings*) angezeigt werden:

```
$ vios lsnpiv ms03-vio1
NAME      SLOT  FC    FCLABEL  CLIENT        CLNTOS  VFCCLIENT  VFCSLOT  STATUS       PORTS
vfchost2  C5    fcs0  Fabric1  aix22(5)      AIX     fcs0       C10      LOGGED_IN      7
vfchost3  C125  fcs0  Fabric1  aixsap01(9)   AIX     fcs0       C10      LOGGED_IN      7
vfchost4  C181  fcs0  Fabric1  aixdbp02(11)  AIX     fcs0       C10      LOGGED_IN      5
vfchost5  C182  fcs0  Fabric1  aixdbi02(13)  AIX     fcs0       C10      LOGGED_IN      5
vfchost6  C38   -     -        -(5)          -       -          -        NOT_LOGGED_IN  0
$
```

Die Spalte *NAME* ganz links gibt den Gerätenamen des virtuellen FC Server-Adapters auf dem Virtual-I/O-Server an. Die Bezeichnung *vfchost* (*virtual FC host*) deutet eindeutig darauf hin, daß es sich um einen virtuellen FC Server (Host) Adapter handelt. Als nächstes wird die virtuelle Slot Nummer des virtuellen FC Server Adapters auf dem Virtual-I/O-Server in der Spalte *SLOT* gezeigt. Falls der Server (Host) Adapter einem physikalischen Port zugewiesen wurde, wird der physikalische FC-Port in der Spalte *FC* genannt, andernfalls wird ein „-„ ausgegeben.

Als zusätzliche Information wird das *label*-Attribut des physikalischen FC-Ports in der Spalte *FCLABEL* ausgegeben. Danach folgen Informationen über die Client-LPAR (Name und LPAR-ID), sowie das Betriebssystem des Clients und der Gerätename des zugehörigen virtuellen FC Client-Adapter und die zugehörige virtuelle Slot-Nummer auf der Client-LPAR. Als letztes wird der Status des virtuellen FC Server Adapter und falls in die Fabric eingelogged (*STATUS= LOGGED_IN*), wird die Anzahl der sichtbaren FC-Ports ausgegeben. Einer der sichtbaren FC-Ports ist immer der virtuelle FC Server Adapter selbst, die weiteren sichtbaren FC-Ports sind in der Regel FC-Ports des verwendeten Storage. Ein Storage-Port kann mehr als eine Disk (LUN) zur Verfügung stellen, daher liefert die Anzahl der sichtbaren Storage-Ports keinen Hinweis darauf wieviele LUNs dem Client zugewiesen wurden.

Der neu angelegte virtuelle FC Server-Adapter ist *vfchost6* (Slot *C38*). Der Ausgabe kann man leicht entnehmen das diesem noch kein physikalischer FC-Port zugeordnet wurde.

Die Zuordnung eines physikalischen FC-Ports für einen virtuellen FC Server-Adapter erfolgt mit Hilfe des Kommandos „*vios vfcmap*“ (*map virtual FC adapter*):

```
$ vios vfcmap ms03-vio1 vfchost6 fcs0
vfchost6 mapped to fcs0
$
```

Dabei wird nach dem Virtual-I/O-Server (auf dem das Mapping durchgeführt werden soll) das *vfchost*-Gerät und der physikalische FC-Port angegeben. Beim Mapping loggt sich der virtuelle FC Server-Adapter sofort in die Fabric ein:

```
$ vios lsnpiv ms03-vio1
NAME      SLOT  FC    FCLABEL  CLIENT        CLNTOS  VFCCLIENT  VFCSLOT  STATUS       PORTS
vfchost2  C5    fcs0  Fabric1  aix22(5)      AIX     fcs0       C10      LOGGED_IN      7
vfchost3  C125  fcs0  Fabric1  aixsap01(9)   AIX     fcs0       C10      LOGGED_IN      7
vfchost4  C181  fcs0  Fabric1  aixdbp02(11)  AIX     fcs0       C10      LOGGED_IN      5
vfchost5  C182  fcs0  Fabric1  aixdbi02(13)  AIX     fcs0       C10      LOGGED_IN      5
vfchost6  C38   fcs0  Fabric1  aix22(5)      AIX     fcs2       C11      LOGGED_IN      1
$
```

Anstelle des *vfchost*-Geräts erlaubt das Kommando „*vios vfcmap*“ auch die Slot-Nummer des virtuellen FC Server-Adapters anzugeben:

```
$ vios vfcmap ms03-vio1 38 fcs0
vfchost6 mapped to fcs0
$
```

Anhand der Slot-Nummer 38 (es kann auch *C38* angegeben werden) wird das zugehörige *vfchost*-Gerät gesucht und dann beim Mapping verwendet.

Anstelle des physikalischen FC-Ports kann auch das *label*-Attribut verwendet werden. Damit lässt sich beim Mapping auch direkt die gewünschte Fabric angeben, ohne das man den Gerätenamen des physikalischen FC-Ports kennen muß:

```
$ vios vfcmap ms03-vio1 vfchost6 Fabric1
vfchost6 mapped to fcs0
$
```

Haben mehrere FC-Ports das gleiche Label, dann wird der erste FC-Port verwendet!

Sowohl beim *vfchost*-Gerät, als auch beim physikalischen FC-Port, kann auch der Physical Location Code des Gerätes angegeben werden. Der physikalische FC-Port *fcs0* hat beispielsweise den Location Code *U78AA.001.VYRGU0Q-P1-C5-T1*. Anstelle des Gerätenamens *fcs0* kann daher auch *U78AA.001.VYRGU0Q-P1-C5-T1* angegeben werden. Dies ist natürlich keine übermäßige Vereinfachung. Aber es ist nicht erforderlich den kompletten Location Code anzugeben, es reicht einen eindeutigen Suffix des Location Codes anzugeben, wobei der Suffix immer nach einem Punkt „.“ oder Bindestrich „-„ anfangen muß. Die folgenden Suffixe sind auf dem Beispiel-System eindeutig:

```
   U78AA.001.VYRGU0Q-P1-C5-T1
   001.VYRGU0Q-P1-C5-T1
   VYRGU0Q-P1-C5-T1
   P1-C5-T1
   C5-T1
```

Damit könnte das Mapping auch wie folgt durchgeführt werden:

```
$ vios vfcmap ms03-vio1 C38 C5-T1
vfchost6 mapped to fcs0
$
```

Dabei wurde das *vfchost*-Gerät über die Slot-Nummer *C38* angegeben und der physikalische FC-Port über den eindeutigen Suffix *C5-T1*. Das Resultat ist das gleiche wie in allen anderen Beispielen oben: *vfchost6* wird auf *fcs0* gemappt.

Der Suffix *T1* ist hingegen nicht eindeutig, man bekommt beim Mapping eine entsprechende Fehlermeldung:

```
$ vios vfcmap ms03-vio1 vfchost6 T1
ERROR: more than one matching fcs adapter
did you mean: fcs0 , U78AA.001.VYRGU0Q-P1-C5-T1 (label: Fabric1)
did you mean: fcs4 , U78AA.001.VYRGU0Q-P1-C1-C1-T1 (label: Fabric1)

USAGE:
   vios [-h <hmc>] [-m <ms>] vfcmap [-v] <vios> <vfchost>|<physloc>|<client>[/{<fcs>|
<slot>}] <fcs>|<physloc>|<label>
$
```

Zusammen mit der Fehlermeldung werden alle Physical Location Codes angezeigt, für die *T1* ein Suffix ist.

7.4.4. Hinzufügen von LUNs

Nachdem ein virtueller FC Adapter angelegt und einem physikalischen FC-Port zugewiesen ist, können dem Adapter LUNs zugewiesen werden. Die LUNs werden dabei direkt den beiden WWPNs des virtuellen FC Client Adapter zugewiesen. Wird die Angabe der zweiten WWPN vergessen, dann wird dies in der Regel erst einmal nicht bemerkt, da zu einem Zeitpunkt immer nur eine der beiden WWPNs aktiv ist (Ausnahme bei LPM). Wird dann einige Zeit später eine LPM-Operation gestartet (oder eine Validierung durchgeführt), dann scheitert diese, da auf dem Ziel-Managed System die bisher inaktive WWPN aktiviert wird und dann eine Überprüfung durchgeführt wird, ob die gleichen Storage-Ports wie über die aktive WWPN erreicht werden können. Das ist dann bei Vergessen der zweiten

WWPN in der Zoning und LUN-Masking Konfiguration nicht der Fall. Damit kann auf dem Ziel-Managed System nicht auf alle LUNs zugegriffen werden.

Werden neue LUNs hinzugefügt, so sind diese dann zwar bei korrekter Konfiguration im SAN sofort sichtbar, werden aber vom Betriebssystem der Client-LPAR nicht automatisch erkannt und verwendet. Im Falle von AIX muß auf der Client-LPAR ein Lauf des Config-Managers gestartet werden.

```
aix22# cfgmgr
aix22#
```

7.4.5. Hinzufügen eines virtuellen FC Adapters mit Mapping

Ist beim Hinzufügen eines virtuellen FC Client-Adapter schon klar welchem physikalischen FC-Port der Adapter auf dem Virtual-I/O-Server zugeordnet werden soll. Dann kann das Mapping schon beim Anlegen des virtuellen Adapter durchgeführt werden. Dazu bietet das Kommando „*lpar addfc*“ die Möglichkeit den physikalischen FC-Port als zusätzliches Argument beim Anlegen mit anzugeben:

```
$ lpar addfc aix22 11 ms03-vio1 fcs0
aix22 slot 11 c050760b72b00012,c050760b72b00013 -> ms03-vio1 slot 38 added by DLPAR
operation
aix22 slot 11 c050760b72b00012,c050760b72b00013 -> ms03-vio1 slot 38 added to current
profile (standard)
ms03-vio1 slot 38 -> aix22 slot 11 added by DLPAR operation
vfchost6 mapped to fcs0
$
```

Natürlich kann genauso wie bei „*vios vfcmap*“ anstelle des physikalischen FC-Ports auch das Label oder ein eindeutiger Suffix des Physical Location Codes angegeben werden:

```
$ lpar addfc aix22 11 ms03-vio1 Fabric1
aix22 slot 11 c050760b72b00012,c050760b72b00013 -> ms03-vio1 slot 38 added by DLPAR
operation
aix22 slot 11 c050760b72b00012,c050760b72b00013 -> ms03-vio1 slot 38 added to current
profile (standard)
ms03-vio1 slot 38 -> aix22 slot 11 added by DLPAR operation
vfchost6 mapped to fcs0
$
```

7.4.6. Ändern von Attributen eines virtuellen FC Adapters

Ein virtueller FC Adapter besitzt die folgenden Attribute:

```
$ lpar help chfc
USAGE:
   lpar [-h <hmc>] [-m <ms>] [-p <profile>] chfc [-r] [-R] [-v] <lpar> <slot> [<attributes>
...]
```

```
…
 Valid attributes for profile only:
   wwpns - 2 comma-separated WWPNs
   is_required - adapter is required for LPAR activation
…
$
```

Aktuell gibt es nur die beiden Attribute *wwpns* und *is_required*. Beide Attribute können nicht dynamisch geändert werden. Eine Änderung ist nur im Profil möglich. Eine Änderung kann mit dem Kommando „*lpar chfc*“ (*change virtual FC adapter*) durchgeführt werden, dabei muß die Option „*-p*“ und ein Profilname verwendet werden.

7.4.7. Entfernen eines NPIV-Mappings

Die Zuordnung eines virtuellen FC Server-Adapter zu einem physikalischen FC-Port kann jederzeit auch wieder aufgehoben werden. Dabei sollte aber beachtet werden, das auf der Client-LPAR die über den entsprechenden virtuellen FC Client Adapter angebundenen Geräte (in der Regel LUNs und/oder Tapes) dann sofort nicht mehr erreichbar sind. Der Zugriff auf Tapes führt dann unmittelbar zu einem I/O Error. Beim Zugriff auf angebundene LUNs hängt das Verhalten davon ab, ob Multipathing verwendet wird (bei AIX wird standardmäßig das *AIXPCM* Path Control Module verwendet) und ob Pfade über andere virtuelle FC Client-Adapter zu den betroffenen LUNs noch verfügbar sind.

Als konkretes Beispiel für das Entfernen des Mappings für einen virtuellen FC Server-Adapter, ist im Folgenden das Vorgehen gezeigt um das Mapping für den virtuellen FC Adapter *fcs0* der Client-LPAR *aix22* zu entfernen.

Es sollte überprüft werden, ob alle verwendeten LUNs alternative Pfade über einen anderen virtuellen FC Adapter haben:

```
aix22 # lspath
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi1
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
Enabled hdisk0   fscsi1
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
aix22 #
```

In der Beispiel-Ausgabe haben alle LUNs, bis auf *hdisk2*, alternative Pfade über den zweiten virtuellen FC-Adapter *fcs1*. Die *hdisk2* besitzt nur Pfade über *fcs0*:

```
aix22 # lspath -l hdisk2
Enabled hdisk2 fscsi0
Enabled hdisk2 fscsi0
aix22 #
```

D.h. wenn das Mapping für *fcs0* entfernt wird, ist kein I/O mehr zu *hdisk2* möglich! Allerdings ist die *hdisk2* in unserem Fall nicht in Verwendung:

```
aix22 # lspv|grep hdisk2
hdisk2          cafe000000000002                    None
aix22 #
```

Daher entfernen wir die *hdisk2* permanent aus dem Betriebssystem:

```
aix22 # rmdev -dl hdisk2
hdisk2 deleted
aix22 #
```

Alle anderen LUNs haben zusätzliche Pfade über *fcs1* und bleiben daher verfügbar. Damit die Pfade über *fcs0* nicht mehr verwendet werden, entfernen wir diese Pfade:

```
aix22 # rmpath -dp fscsi0
paths Deleted
aix22 #
```

Über das Kommando „*vios lsnpiv*“ lässt sich der zugehörige Virtual-I/O-Server und das *vfchost*-Gerät für den virtuellen FC Client-Adapter *fcs0* leicht herausfinden:

```
$ vios lsnpiv ms03-vio1
NAME      SLOT  FC    FCLABEL  CLIENT        CLNTOS  VFCCLIENT  VFCSLOT  STATUS       PORTS
vfchost2  C5    fcs0  Fabric1  aix22(5)      AIX     fcs0       C10      LOGGED_IN      7
vfchost3  C125  fcs0  Fabric1  aixsap01(9)   AIX     fcs0       C10      LOGGED_IN      7
vfchost4  C181  fcs0  Fabric1  aixdbp02(11)  AIX     fcs0       C10      LOGGED_IN      5
vfchost5  C182  fcs0  Fabric1  aixdbi02(13)  AIX     fcs0       C10      LOGGED_IN      5
vfchost6  C38   fcs0  Fabric1  aix22(5)      AIX     fcs2       C11      LOGGED_IN      1
$
```

Nun kann die Zuordnung über das Kommando „*vios vfcunmap*“ (*unmap virtual FC adapter*) aufgehoben werden:

```
$ vios vfcunmap ms03-vio1 vfchost2
vfchost2 unmapped
$
```

Anstelle des Gerätenamens für den *vfchost* Adapter kann, wie beim Mappen, die Slot-Nummer oder ein eindeutiger Suffix des Physical Location Codes angegeben werden.

7.4.8. Ändern eines NPIV-Mappings

Die Zuordnung eines virtuellen FC Server-Adapters zu einem physikalischen FC-Port kann dynamisch geändert werden. Allerdings kann die Änderung des Mappings Auswirkungen auf das I/O der Client-LPAR haben. Wird das Mapping auf einen anderen physikalischen FC-Port geändert, ist die Änderung im Besten Fall (fast) unterbrechungsfrei. Die Pfade über den virtuellen FC Client-Adapter sind dann kurze Zeit nicht verfügbar, funktionieren allerdings nach dem Ummappen sofort wieder. Ob dies I/O-Probleme in der Client-LPAR verursachen kann, hängt vom Betriebssystem, der Konfiguration der virtuellen FC Client-Adapter (insbesondere das Attribut *dyntrk*) und der I/O-Auslastung ab. Um mögliche Probleme zu vermeiden, sollte man die betroffenen Pfade in der Client-LPAR vor dem Ummappen wegnehmen, ähnlich dem Vorgehen beim Wegnehmen des Mappings. Im Folgenden ist dies für den Fall der AIX LPAR *aix22* kurz beschrieben. Der virtuelle FC Client-Adapter *fcs0* von *aix22* soll vom physikalischen FC-Port *fcs0* auf dem Virtual-I/O-Server *ms03-vio1* auf den physikalischen FC-Port *fcs4* umgemappt werden. Der physikalische FC-Port *fcs4* befindet sich in der gleichen Fabric, damit müsste man nach dem Ummappen exakt die gleichen LUNs wieder erreichen können.

Als erstes wird wieder überprüft, ob alle verwendeten LUNs alternative Pfade über einen anderen virtuellen FC Adapter haben:

```
aix22 # lspath
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi1
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
Enabled hdisk0   fscsi1
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
aix22 #
```

Für die *hdisk2* ist das nicht der Fall, sie besitzt nur Pfade über *fcs0*:

```
aix22 # lspath -l hdisk2
Enabled hdisk2 fscsi0
Enabled hdisk2 fscsi0
aix22 #
```

Allerdings ist die *hdisk2* in unserem Fall nicht in Verwendung:

```
aix22 # lspv|grep hdisk2
hdisk2          cafe000000000002                    None
aix22 #
```

Daher entfernen wir die *hdisk2* temporär aus dem Betriebssystem:

```
aix22 # rmdev -l hdisk2
hdisk2 Defined
aix22 #
```

Nach dem Ummappen sollte die *hdisk2* wieder erreichbar sein, daher behalten wir die Definition von *hdisk2* in der ODM.

Alle anderen LUNs haben zusätzliche Pfade über *fcs1* und bleiben daher verfügbar. Damit die Pfade über *fcs0* nicht mehr verwendet werden, entfernen wir diese Pfade:

```
aix22 # rmpath -p fscsi0
paths Defined
aix22 #
```

Nachdem nun der virtuelle FC Client-Adapter *fcs0* nicht mehr in Verwendung ist, kann das Mapping auf dem Virtual-I/O-Server geändert werden:

```
$ vios vfcmap ms03-vio1 vfchost2 fcs4
vfchost2 mapped to fcs4
$
```

Eine kurze Überprüfung der NPIV-Mappings zeigt, das sich der virtuelle FC Server-Adapter sofort wieder in die Fabric eingeloggt hat, und die Storage-Ports auch wieder sichtbar sind:

```
$ vios lsnpiv ms03-vio1
NAME      SLOT  FC    FCLABEL  CLIENT        CLNTOS  VFCCLIENT  VFCSLOT  STATUS       PORTS
vfchost2  C5    fcs4  Fabric1  aix22(5)      AIX     fcs0       C10      LOGGED_IN      7
vfchost3  C125  fcs0  Fabric1  aixsap01(9)   AIX     fcs0       C10      LOGGED_IN      7
vfchost4  C181  fcs0  Fabric1  aixdbp02(11)  AIX     fcs0       C10      LOGGED_IN      5
vfchost5  C182  fcs0  Fabric1  aixdbi02(13)  AIX     fcs0       C10      LOGGED_IN      5
vfchost6  C38   fcs0  Fabric1  aix22(5)      AIX     fcs2       C11      LOGGED_IN      1
$
```

Als letztes können die entfernte *hdisk2* und die entfernten Pfade durch einen Lauf des Config-Managers wieder ins Betriebssystem konfiguriert werden:

```
aix22 # cfgmgr
aix22 #
```

Danach sind alle Pfade wieder verfügbar, und auch die temporär entfernte *hdisk2* taucht wieder auf:

```
aix22 # lspath
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi1
```

```
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
Enabled hdisk0   fscsi1
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
aix22 #
```

7.4.9. Wegnehmen eines virtuellen FC Adapters

Wird ein virtueller FC Client-Adapter nicht mehr benötigt, dann kann er natürlich auch wieder entfernt werden. Geräte, die ausschließlich über den zu entfernenden Adapter erreichbar sind, können dann nicht mehr verwendet werden und sollten daher gelöscht werden. Ein virtueller FC Client-Adapter kann mit dem Kommando „*lpar rmfc*“ (*remove virtual FC adapter*) entfernt werden.

Hinweis: Ein virtueller FC Client-Adapter kann nicht entfernt werden, wenn er noch in Benutzung ist bzw. das zugehörige *fcs*-Gerät noch im Betriebssystem konfiguriert ist.

```
$ lpar rmfc aix22 10
hmc01: chhwres -m ms03 -r virtualio --rsubtype fc  -o r -p aix22 -s 10
ERROR: remote HMC command returned an error (1)
StdErr: HSCL294D Dynamic remove of virtual I/O resources failed:
StdErr:
StdErr:
StdErr: 0931-029 The specified slot contains a device or devices that are currently
StdErr:  configured. Unconfigure the following device or devices with the rmdev
StdErr:  command and try again.
StdErr:
StdErr: fcs0
StdErr:
StdErr:
StdErr:
StdErr:
StdErr: The OS return code is 1.
StdErr:
$
```

Der Versuch resultiert in einer Fehlermeldung, welche darauf hinweist das noch ein Gerät (hier *fcs0*) in Verwendung ist. Es ist also nicht möglich einen aktiven virtuellen FC-Adapter „aus Versehen“ zu löschen.

Als Beispiel ist gezeigt wie der virtuelle FC Client-Adapter *fcs0* der LPAR *aix22* korrekt entfernt werden kann:

Als erstes sollte man sich die Informationen, wie WWPNs, Slot-Nummern und Virtual-I/O-Server, für den zu entfernenden virtuellen FC Client-Adapter notieren. Falls es ein Problem geben sollte, kann man ihn dann relativ einfach wieder anlegen:

```
$ lpar lsvslot aix22
SLOT  REQ  ADAPTER_TYPE   DATA
0     Yes  serial/server  remote: (any)/any connect_status= hmc=1
```

```
1     Yes  serial/server  remote: (any)/any connect_status= hmc=1
5     No   eth            PVID=100 VLANS= ETHERNET0
10    No   fc/client      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
20    No   fc/client      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
$
```

Der virtuelle FC Client-Adapter *fcs0* war der Adapter in Slot *10* (das kann man vorsichtshalber mit „*vios lsnpiv*“ noch einmal prüfen), der zugehörige virtuelle FC Server-Adapter ist im Slot *5* des Virtual-I/O-Servers *ms03-vio1*. Die beiden WWPNs des Adapters sind *c05076030aba0002* und *c05076030aba0003*.

Es sollte überprüft werden, ob alle verwendeten LUNs alternative Pfade über einen anderen virtuellen FC Adapter haben:

```
aix22 # lspath
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi1
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
Enabled hdisk0   fscsi1
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
aix22 #
```

In der Beispiel-Ausgabe haben alle LUNs, bis auf *hdisk2*, alternative Pfade über den zweiten virtuellen FC-Adapter *fcs1*. Die *hdisk2* besitzt nur Pfade über *fcs0*:

```
aix22 # lspath -l hdisk2
Enabled hdisk2 fscsi0
Enabled hdisk2 fscsi0
aix22 #
```

D.h. wenn das Mapping für *fcs0* entfernt wird, ist kein I/O mehr zu *hdisk2* möglich! Allerdings ist die *hdisk2* in unserem Fall nicht in Verwendung:

```
aix22 # lspv|grep hdisk2
hdisk2          cafe000000000002                    None
aix22 #
```

Daher entfernen wir die *hdisk2* permanent aus dem Betriebssystem:

```
aix22 # rmdev -dl hdisk2
```

```
hdisk2 deleted
aix22 #
```

Alle anderen LUNs haben zusätzliche Pfade über *fcs1* und bleiben daher verfügbar. Damit die Pfade über *fcs0* nicht mehr verwendet werden, entfernen wir diese Pfade:

```
aix22 # rmpath -dp fscsi0
paths Deleted
aix22 #
```

Als letztes muß das Gerät *fcs0* für den virtuellen FC Client-Adapter aus dem Betriebssystem entfernt werden. Da der virtuelle FC Client-Adapter entfernt wird, wird auch die Definition in der *ODM* nicht mehr benötigt. Daher sollte beim Entfernen die Option „*-d*“ bei *rmdev* verwendet werden, um die *ODM* Definitionen für das Gerät ebenfalls zu entfernen. Das Gerät *fcs0* besitzt mindestens ein Kind-Gerät (*fscsi0*), welches ebenfalls entfernt werden muß. Es wird die Option „*-R*“ (recursive) von *rmdev* verwendet:

```
aix22 # rmdev -Rdl fcs0
fscsi0 deleted
fcs0 deleted
aix22 #
```

Auf dem Virtual-I/O-Server *ms03-vio1* besteht immer noch die Zuordnung des virtuellen FC Server-Adapters zum physikalischen FC-Port. Dieses Mapping muß entfernt werden, um den virtuellen FC Server-Adapter löschen zu können. Mittels „*vios lsnpiv*“ können die aktuellen NPIV-Mappings des Virtual-I/O-Server *ms03-vio1* noch einmal angezeigt werden, um das *vfchost* Gerät des Mappings herauszufinden:

```
$ vios lsnpiv ms03-vio1
NAME      SLOT  FC    FCLABEL  CLIENT        CLNTOS  VFCCLIENT  VFCSLOT  STATUS       PORTS
vfchost2  C5    fcs4  Fabric1  aix22(5)      AIX     fcs0       C10      LOGGED_IN      7
vfchost3  C125  fcs0  Fabric1  aixsap01(9)   AIX     fcs0       C10      LOGGED_IN      7
vfchost4  C181  fcs0  Fabric1  aixdbp02(11)  AIX     fcs0       C10      LOGGED_IN      5
vfchost5  C182  fcs0  Fabric1  aixdbi02(13)  AIX     fcs0       C10      LOGGED_IN      5
vfchost6  C38   fcs0  Fabric1  aix22(5)      AIX     fcs2       C11      LOGGED_IN      1
$
```

Der zum virtuellen FC Client-Adapter gehörige Server-Adapter ist *vfchost2*! Das Mapping kann wieder mit „*vios vfcunmap*“ entfernt werden:

```
$ vios vfcunmap ms03-vio1 vfchost2
vfchost2 unmapped
$
```

Schließlich können jetzt virtueller FC Client- und Server-Adapter durch den POWER Hypervisor entfernt werden, dazu dient das Kommando „*lpar rmfc*“:

```
$ lpar rmfc aix22 10
aix22 slot 10 -> ms03-vio1 slot 5 removed by DLPAR operation
aix22 slot 10 -> ms03-vio1 slot 5 removed from current profile (standard)
ms03-vio1 slot 5 -> aix22 slot 10 removed by DLPAR operation
$
```

Die Ausgabe bestätigt, das auch der virtuelle FC Server-Adapter auf dem Virtual-I/O-Server entfernt wurde.

Soll ein virtueller FC Client-Adapter aus einem Profil einer LPAR entfernt werden, dann muß lediglich die Option „-*p*“ mit dem Profilnamen angegeben werden.

7.4.10. Verwendung von vorgegebenen WWPNs

Gelegentlich kommt es vor, das für einen virtuellen FC Client-Adapter, ein Paar von vorgegebenen WWPNs verwendet werden soll. Z.B. wenn man nach dem Löschen eines virtuellen FC Client-Adapters feststellt, das dieser doch benötigt wird.

Im letzten Kapitel wurde ein virtueller FC Client-Adapter der LPAR *aix22* entfernt. Der Adapter hatte die beiden WWPNs *c05076030aba0002* und *c05076030aba0003*. Der Adapter soll im Folgenden wieder mit den ursprünglichen WWPNs angelegt werden, alle LUNs müssten dann wieder über den Adapter erreichbar sein.

Beim Anlegen eines virtuellen FC Client-Adapters können vorgegebene WWPNs als zusätzliches Argument angegeben werden:

```
$ lpar addfc aix22 10 ms03-vio1 5 c05076030aba0002,c05076030aba0003 fcs4
aix22 slot 10 c05076030aba0002,c05076030aba0003 -> ms03-vio1 slot 5 added by DLPAR
operation
aix22 slot 10 c05076030aba0002,c05076030aba0003 -> ms03-vio1 slot 5 added to current
profile (standard)
ms03-vio1 slot 5 -> aix22 slot 10 added by DLPAR operation
vfchost2 mapped to fcs4
$
```

Man hätte die virtuelle Slot-Nummer *5* für den Virtual-I/O-Server auch weglassen können, es wäre dann automatisch eine freie Slot-Nummer ausgewählt worden (nicht notwendigerweise die *5*). Das Mapping auf den physikalischen FC-Port *fcs4* wurde auch gleich mitangegeben.

Ein Blick auf die virtuellen Slots zeigt, daß der virtuelle FC Client-Adapter mit den gewünschten WWPNs wieder im Slot *10* angelegt wurde:

```
$ lpar lsvslot aix22
SLOT  REQ  ADAPTER_TYPE   DATA
0     Yes  serial/server  remote: (any)/any connect_status= hmc=1
1     Yes  serial/server  remote: (any)/any connect_status= hmc=1
5     No   eth            PVID=100 VLANS= ETHERNET0
10    No   fc/client      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
20    No   fc/client      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
$
```

Auch das Mapping auf dem Virtual-I/O-Server *ms03-vio1* wurde wieder eingerichtet:

```
$ vios lsnpiv ms03-vio1
NAME      SLOT  FC    FCLABEL  CLIENT        CLNTOS  VFCCLIENT  VFCSLOT  STATUS       PORTS
vfchost2  C5    fcs4  Fabric1  aix22(5)      AIX     fcs0       C10      LOGGED_IN      7
vfchost3  C125  fcs0  Fabric1  aixsap01(9)   AIX     fcs0       C10      LOGGED_IN      7
vfchost4  C181  fcs0  Fabric1  aixdbp02(11)  AIX     fcs0       C10      LOGGED_IN      5
vfchost5  C182  fcs0  Fabric1  aixdbi02(13)  AIX     fcs0       C10      LOGGED_IN      5
vfchost6  C38   fcs0  Fabric1  aix22(5)      AIX     fcs2       C11      LOGGED_IN      1
$
```

Als letztes können die Pfade und LUNs auf der Client-LPAR wieder mit einem Lauf des Config-Managers ins Betriebssystem konfiguriert werden:

```
aix22 # cfgmgr
aix22 #
```

Es stehen wieder alle Pfade und auch die gelöschte *hdisk2* zur Verfügung:

```
aix22 # lspath
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi0
Enabled hdisk1   fscsi0
Enabled hdisk2   fscsi0
Enabled hdisk3   fscsi0
Enabled hdisk4   fscsi0
Enabled hdisk5   fscsi0
Enabled hdisk0   fscsi1
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
Enabled hdisk0   fscsi1
Enabled hdisk1   fscsi1
Enabled hdisk3   fscsi1
Enabled hdisk4   fscsi1
Enabled hdisk5   fscsi1
aix22 #
```

Wie gehabt, kann der virtuelle FC Client-Adapter mit vorgegebenen WWPNs auch nur im Profil konfiguriert werden, dazu ist nur die Option „*-p*“ mit dem Profilnamen zu verwenden.

7.5. Virtual SCSI

Eine weitere Möglichkeit für die Virtualisierung von Storage unter PowerVM ist die Verwendung von virtuellen SCSI Adaptern. Wie bei virtuellem FC ist dabei ein virtueller SCSI-Client Adapter über den Hypervisor mit einem virtuellen SCSI-Server Adapter auf einem Virtual-I/O-Server verbunden. Der virtuelle SCSI-Client Adapter hat dabei die Rolle eines SCSI-Initiators, der virtuelle SCSI-Server Adapter auf dem Virtual-I/O-Server ist das SCSI-Target. Im Unterschied zu physikalischem SCSI, bei dem es mehrere SCSI-Targets geben kann, die an einen physikalischen SCSI Host Bus Adapter (HBA) angebunden sind, ist bei virtuellem SCSI maximal ein SCSI-Target angebunden. Die Verbindung des virtuellen SCSI-Client Adapters zum virtuellen SCSI-Server Adapter mittels Hypervisor nutzt einen sogenannten logischen Kanal. Dies ist eine Punkt-zu-Punkt Verbindung zwischen den beiden Adaptern.

Im Unterschied zum virtuellen FC müssen im Falle von virtuellem SCSI alle Geräte auf die der virtuelle SCSI-Client Adapter zugreifen soll, einzeln dem virtuellen SCSI-Server Adapter zugeordnet werden. Dies kann bei ein paar hundert Geräten (in größeren Umgebungen können das auch mehr als tausend Geräte sein) schnell sehr

unübersichtlich werden. In Bild 7.14 ist der Kommunikationspfad vom virtuellen SCSI-Client Adapter über den Hypervisor zum virtuellen SCSI-Server Adapter und dann weiter zu den Festplatten gezeigt.

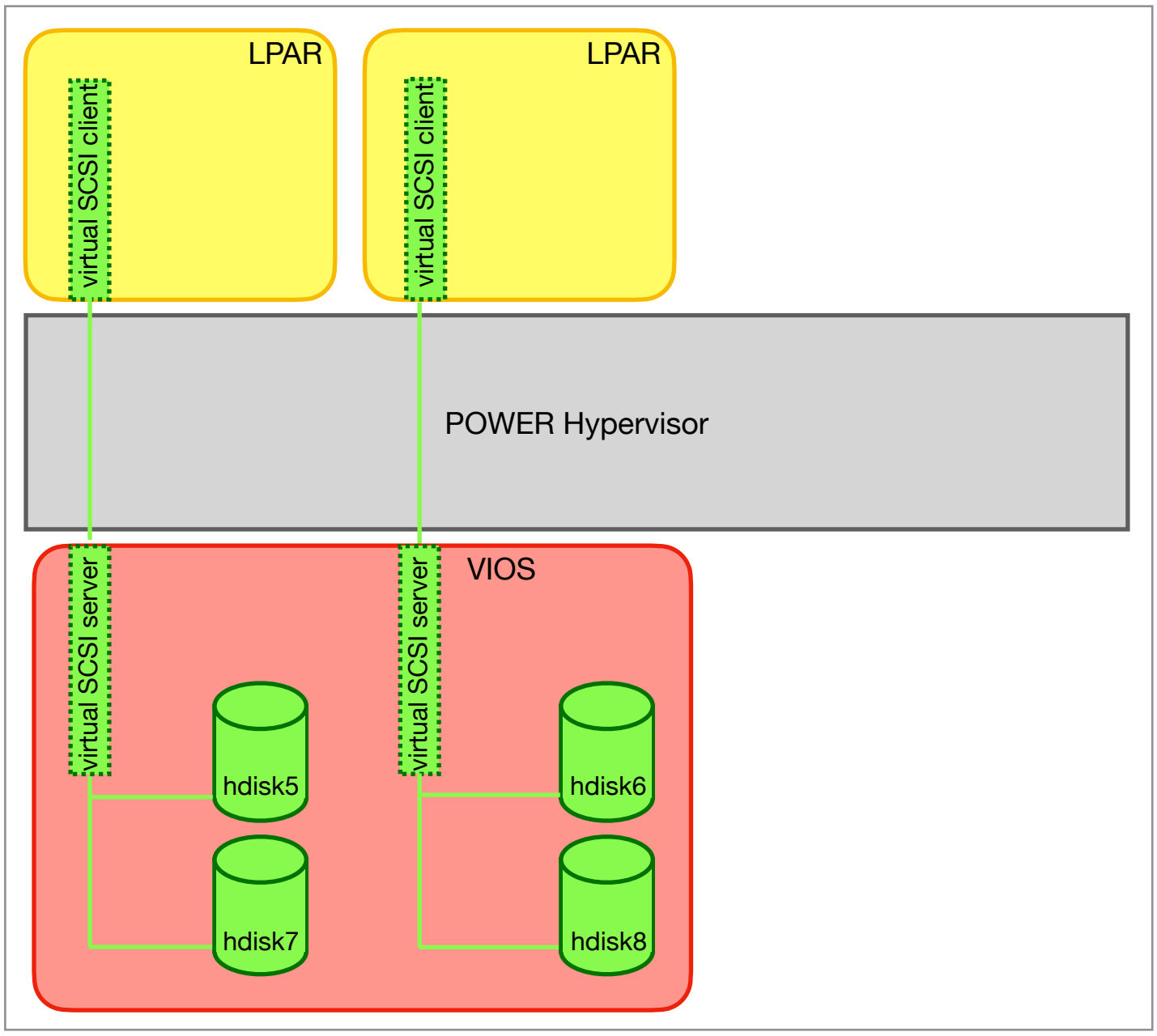


Bild 7.14: Kommunikationspfad virtueller SCSI Client Adapter zu den LUNs.

Die Festplatten werden auf den Virtual-I/O-Servern dem virtuellen SCSI-Server Adapter zugeordnet, indem ein sogenanntes virtuelles Target Device erzeugt wird, welches auf das Ziel-Gerät verweist. Das Ziel-Gerät wird als Backing-Device bezeichnet. Als Backing-Devices können Festplatten (*hdisk*), Tapes (*rmt*) oder Logical-Volumes verwendet werden. Ein Backing-Device muß aber nicht zwangsläufig ein Gerät sein, sondern kann auch eine Datei im Dateisystem sein, siehe Virtual Optical Media Library und Storage-Pools später.

7.5.1. Hinzufügen eines virtuellen SCSI Adapters

Soll einer aktiven LPAR ein virtueller SCSI Adapter hinzugefügt werden, muß die LPAR eine aktive RMC-Verbindung zu einer HMC haben. Für den virtuellen SCSI Adapter wird ein freier virtueller Slot benötigt. Da ein virtueller SCSI Client Adapter immer mit einem virtuellen SCSI Server Adapter auf einem Virtual-I/O-Server zusammen arbeitet, muß für jeden Client-Adapter immer auch ein Server-Adapter auf einem Virtual-I/O-Server angelegt werden. Auch auf dem Virtual-I/O-Server benötigt man dazu einen freien virtuellen Slot.

Welche virtuellen Slots auf der Client-LPAR noch verfügbar sind, lässt sich mit „*lpar lsvslot*“ (*list virtual slots*) ermitteln:

```
$ lpar lsvslot aix22
SLOT  REQ  ADAPTER_TYPE   STATE  DATA
0     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
1     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
5     No   eth            1      PVID=100 VLANS= ETHERNET0 1DC8DB485D1E
10    No   fc/client      1      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
20    No   fc/client      1      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
$
```

Hier ist beispielsweise der virtuelle Slot *11* noch verfügbar.

Die verfügbaren virtuellen Slots auf einem Virtual-I/O-Server lassen sich auf die gleiche Weise ermitteln, hier für *ms03-vio1*:

```
$ lpar lsvslot ms03-vio1
SLOT  REQ  ADAPTER_TYPE   STATE  DATA
0     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
1     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
5     No   fc/server      1      remote: aix22(5)/10
11    No   eth            1      PVID=900 VLANS= ETHERNET0 1DC8DB485D0B
20    No   eth            1      PVID=1 VLANS= ETHCTRL 1DC8DB485D14
21    No   eth            1      TRUNK(1) IEEE PVID=1 VLANS=100,200 ETHERNET0 1DC8DB485D15
22    No   eth            1      TRUNK(1) IEEE PVID=2 VLANS=150,250 ETHERNET0 1DC8DB485D16
125   No   fc/server      1      remote: aixsap01(9)/10
181   No   fc/server      1      remote: aixdbp02(11)/10
182   No   fc/server      1      remote: aixdbi02(13)/10
$
```

Hier wäre unter anderem der virtuelle Slot *111* verfügbar.

Um einen virtuellen SCSI-Client Adapter für eine LPAR anzulegen, dient das Kommando „*lpar addscsi*“ (*add virtual SCSI adapter*). Dabei muß neben der Client LPAR auch der Virtual-I/O-Server für den virtuellen SCSI Server Adapter angegeben werden:

```
$ lpar addscsi aix22 11 ms03-vio1 111
aix22: slot 11 -> ms03-vio1/111 added by DLPAR operation
aix22: slot 11 -> ms03-vio1/111 added to current profile (standard)
ms03-vio1: slot 111 -> aix22/11 added by DLPAR operation
$
```

Die Ausgabe des Kommandos zeigt, daß zunächst ein virtueller SCSI-Client Adapter per DLPAR-Operation hinzugefügt wird. Als nächstes wird das aktuelle Profil der LPAR angepasst und anschließend wird der zugehörige SCSI-Server Adapter auf dem Virtual-I/O-Server angelegt. Das Kommando legt als automatisch beide Adapter an, auf Wunsch kann dies durch Verwendung entsprechender Optionen („*-c*“ *client adapter only*) verhindert werden.

Das Heraussuchen einer freien virtuellen Slot-Nummer auf dem Virtual-I/O-Server kann in größeren Umgebungen mühsam sein, da ein Virtual-I/O-Server leicht mehrere hundert virtuelle Adapter besitzen kann. Die virtuelle Slot-Nummer auf dem Virtual-I/O-Server spielt im Prinzip aber gar keine Rolle. Außerdem gibt es keine Garantie dafür, das bei einer LPM-Operation die virtuelle Slot-Nummer auf dem Virtual-I/O-Server gleich bleibt. (Der virtuelle Slot *111* z.B. könnte ja auf dem Ziel Virtual-I/O-Server schon in Verwendung sein.) Lässt man die Angabe der virtuellen

Slot-Nummer auf dem Virtual-I/O-Server im Kommando „*lpar addscsi*“ einfach weg, dann wird automatisch eine freie Slot-Nummer auf dem Virtual-I/O-Server ermittelt:

```
$ lpar addscsi aix22 11 ms03-vio1
aix22: slot 11 -> ms03-vio1/33 added by DLPAR operation
aix22: slot 11 -> ms03-vio1/33 added to current profile (standard)
ms03-vio1: slot 33 -> aix22/11 added by DLPAR operation
$
```

Hier wurde beispielsweise die Slot-Nummer *33* auf dem Virtual-I/O-Server verwendet.

Möchte man auch auf der Client-LPAR die Slot-Nummer nicht vorgeben, kann man diese ebenfalls weglassen. Es wird dann auch für den Client eine freie virtuelle Slot-Nummer für den virtuellen SCSI-Client Adapter ermittelt. Normalerweise ist das die erste freie Slot-Nummer. Dies lässt sich aber konfigurieren.

Unter AIX wird für den Gerätenamen eines virtuellen SCSI-Client Adapter der Name *vscsi* verwendet. Nach dem Hinzufügen eines virtuellen SCSI-Client Adapters zu einer LPAR wird das zugehörige neue Gerät nicht automatisch vom Betriebssystem erkannt.

```
aix22 # lscfg -l vscsi*
lscfg: device vscsi* not found.
aix22 #
```

Die Fehlermeldung zeigt eindeutig, das der oben angelegte Adapter in Slot *11*, dem Betriebssystem noch nicht bekannt ist.

Nach dem Hinzufügen eines virtuellen SCSI-Client Adapters in einer Client-LPAR, muß das Betriebssystem der Client-LPAR den neuen Client-Adapter erst erkennen. Im Falle von AIX als Betriebssystem muß dazu der Config-Manager *cfgmgr* gestartet werden:

```
aix22 # cfgmgr -v
cfgmgr is running in phase 2
----------------
…
----------------
Attempting to configure device 'vscsi0'
Time: 0 LEDS: 0x25b3
Invoking /usr/lib/methods/cfg_vclient -l vscsi0
Number of running methods: 1
----------------
Completed method for: vscsi0, Elapsed time = 1
Return code = 0
*** no stdout ****
*** no stderr ****
…
aix22 #
```

Die Ausgabe zeigt das der Adapter *vscsi0* erkannt wurde. Allerdings werden keine Kind-Geräte aufgelistet. Dies liegt daran, daß zwar der virtuelle SCSI-Client und virtuelle SCSI-Server Adapter angelegt wurden, aber es wurden dem virtuellen SCSI-Server noch keine Ziel-Geräte zugeordnet.

Hat eine LPAR keine aktive RMC-Verbindung oder ist nicht aktiv, dann kann ein virtueller SCSI Adapter nur einem der Profile der LPAR hinzugefügt werden. Dies ist z.B. immer der Fall, wenn die LPAR gerade neu angelegt wurde und noch nicht installiert ist.

In diesem Fall muß bei den gezeigten Kommandos lediglich die Option „*-p*“ mit einem Profil-Namen verwendet werden. Welche Profile eine LPAR besitzt kann mittels „*lpar lsprof*“ (*list profiles*) einfach herausgefunden werden:

```
$ lpar lsprof aix22
NAME                      MEM_MODE  MEM   PROC_MODE  PROCS  PROC_COMPAT
standard                  ded       7168  ded        2      default
last*valid*configuration  ded       7168  ded        2      default
$
```

(Im Profil mit dem Namen *last*valid*configuration* ist die letzte aktive Konfiguration hinterlegt.)

Die im Profil *standard* definierten virtuellen Adapter lassen sich dann unter Angabe des Profil-Namens mit „*lpar lsvslot*“ anzeigen:

```
$ lpar -p standard lsvslot aix22
SLOT  REQ  ADAPTER_TYPE   DATA
0     Yes  serial/server  remote: (any)/any connect_status= hmc=1
1     Yes  serial/server  remote: (any)/any connect_status= hmc=1
5     No   eth            PVID=100 VLANS= ETHERNET0
10    No   fc/client      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
20    No   fc/client      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
$
```

Beim Hinzufügen des Adapters muß lediglich der entsprechende Profil-Name angegeben werden, ansonsten sieht das Kommando genauso aus, wie oben gezeigt:

```
$ lpar -p standard addscsi aix22 11 ms03-vio1
aix22: slot 11 -> ms03-vio1 slot 33 added to current profile (standard)
ms03-vio1: slot 33 -> aix22 slot 11 added by DLPAR operation
$
```

Um den neuen Adapter in Slot *11* verfügbar zu machen, muß die LPAR unter Angabe des Profil-Namens *standard* neu aktiviert werden.

7.5.2. Zuordnen von virtuellen Target-Devices (Mapping)

Damit in der Client-LPAR auf Storage über einen virtuellen SCSI-Client Adapter zugegriffen werden kann, muß auf dem Virtual-I/O-Server erst einmal dem zugehörigen virtuellen SCSI-Server Adapter Storage zugeordnet werden (VSCSI-Mapping). In diesem Kapitel geht es um das Mapping von Platten und Logical Volumes, die Möglichkeit auch Dateien zuzuordnen werden später separat im Kapitel 8 (*Virtual-I/O-Server*) betrachtet.

Welche virtuellen SCSI-Server Adapter es auf einem Virtual-I/O-Server gibt, lässt sich mit dem Kommando „*vios lsvscsi*“ (*list VSCSI*) anzeigen:

```
$ vios lsvscsi ms03-vio1
SVSA    SLOT  CLIENT    LUNS
vhost0  C33   aix22(5)  0
$
```

Das Kommando zeigt alle virtuellen SCSI-Server Adapter zusammen mit der virtuellen Slot-Nummer des Adapters und dem Namen der zugehörigen Client-LPAR an. In der Spalte *LUNS* wird die Anzahl der zugewiesenen virtuellen Target-Devices angegeben. Der Geräte-Name der virtuellen SCSI-Server Adapter auf den Virtual-I/O-Servern ist *vhost* mit einer fortlaufenden Nummer.

Neben dem *vhost*-Adapter auf welchen ein Gerät gemappt werden soll, benötigt man noch ein verfügbares Gerät das zugeordnet werden soll (Platte oder Logical Volume). Welche Platten auf einem Virtual-I/O-Server verfügbar sind, kann mit dem Kommando „*vios lspv*“ (*list physical volumes*) angezeigt werden:

```
$ vios lspv ms03-vio1
PVNAME  PVID              VGNAME  PVSTATE
hdisk0  00dead00beef0003  None    -
hdisk1  00dead00beef0005  rootvg  active
hdisk2  00dead00beef0008  rootvg  active
hdisk3  00dead00beef000e  None    -
hdisk4  00dead00beef0001  None    -
$
```

Neben den beiden Platten *hdisk1* und *hdisk2* der *rootvg* gibt es noch 3 weitere Platten, welche aktuell nicht in Benutzung sind. Wir verwenden als Beispiel für ein VSCSI-Mapping die freie *hdisk0*.

Hinweis: Bei Platten, welche auf *vhost*-Adapter gemappt werden sollen, sollte das Platten-Attribut *reserve_policy* auf *no_reserve* gesetzt werden. Dies lässt sich mit dem Kommando „*vios chdev*“ durchführen:

```
$ vios chdev ms03-vio1 hdisk0 reserve_policy=no_reserve
$ vios chdev ms03-vio1 hdisk3 reserve_policy=no_reserve
$ vios chdev ms03-vio1 hdisk4 reserve_policy=no_reserve
$
```

Wird eine externe Platte (LUN) über mehr als einen Virtual-I/O-Server gemappt, ist dies zwingend erforderlich, da ansonsten der erste Virtual-I/O-Server die Platte reserviert und die weiteren Virtual-I/O-Server dann nicht auf die Platte zugreifen können.

Das Mapping kann mit dem Kommando „*vios map*“ (*map SCSI target device*) durchgeführt werden. Dabei muß mindestens das zu mappende Gerät und der *vhost*-Adapter angegeben werden:

```
$ vios map ms03-vio1 hdisk0 vhost0
$
```

Das VSCSI-Mapping erzeugt ein Kind-Gerät des virtuellen SCSI-Server Adapter *vhost0*. Das Kind-Gerät hat per Default den Namen *vtscsi* mit einer fortlaufenden eindeutigen Nummer. Bild 7.15 zeigt die Platten als Kind-Geräte des SAS-Adapters *sas0* bzw. des FC Protokoll-Geräts *fscsi4* und das virtuelle Target-Gerät *vtscsi0* als Kind-Gerät des virtuellen SCSI-Server Adapters *vhost0*. Die Zuordnung zur Platte *hdisk0* als sogenanntes Backing-Device erfolgt dabei über das Attribut *aix_tdev* von *vtscsi0*, welches auf die Platte *hdisk0* verweist.

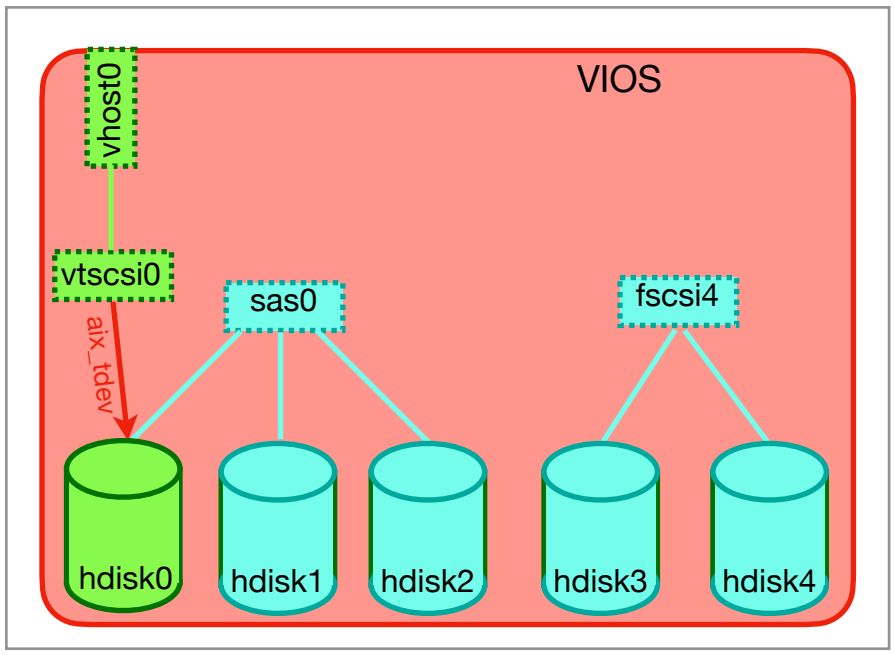


Bild 7.15: Mapping von *hdisk0* auf den Adapter *vhost0*.

Auf die gleiche Weise können weitere Geräte dem Adapter *vhost0* und damit dem zugehörigen virtuellen SCSI-Client Adapter hinzugefügt werden. Eine kurze Überprüfung der virtuellen SCSI-Server Adapter zeigt das nun eine LUN dem Adapter *vhost0* zugeordnet ist:

```
$ vios lsvscsi ms03-vio1
SVSA     SLOT   CLIENT     LUNS
vhost0   C33    aix22(5)   1
$
```

Natürlich lassen sich die zugeordneten LUNs auch anzeigen. Dazu muß man lediglich als weiteres Argument einen der *vhost*-Adapter angeben:

```
$ vios lsvscsi ms03-vio1 vhost0
VTD       STATUS      BACKING  BDPHYSLOC  MIRRORED  LUN
vtscsi0   Available   hdisk0   -          N/A       0x8100000000000000
$
```

Beim Mapping wird also ein neues virtuelles Gerät (*vtscsi0*) als Kind des virtuellen SCSI-Server Adapter *vhost0* erzeugt:

```
$ vios lsdev ms03-vio1 vtscsi0
NAME      STATUS      PHYSLOC                        PARENT  DESCRIPTION
vtscsi0   Available   U9009.22A.8991971-V1-C33-L1    vhost0  Virtual Target Device - Disk
$
```

Das Gerät besitzt die folgenden Attribute:

```
$ vios lsattr ms03-vio1 vtscsi0
ATTRIBUTE        VALUE               DESCRIPTION                          USER_SETTABLE
LogicalUnitAddr  0x8100000000000000  Logical Unit Address                 False
aix_tdev         hdisk0              Target Device Name                   False
allow520blocks   yes                 Allow 520-byte blocks, if supported  False
client_reserve   no                  Client Reserve                       True
mig_name                             N/A                                  True
mirrored         false               Metro/Global Mirror                  True
$
```

Beim Erzeugen des Geräts mit dem Kommando „*vios map*“ wird das Attribut *aix_tdev* auf das angegebene Ziel des Mappings gesetzt (hier *hdisk0*). Das Attribut kann nach dem Erzeugen nicht mehr geändert werden.

Damit die „neue“ Platte auf dem Client verfügbar ist, muß unter AIX der Config-Manager *cfgmgr* gestartet werden:

```
aix22 # cfgmgr -l vscsi0 -v
----------------
Attempting to configure device 'vscsi0'
Time: 0 LEDS: 0x25b3
Invoking /usr/lib/methods/cfg_vclient -l vscsi0
Number of running methods: 1
----------------
Completed method for: vscsi0, Elapsed time = 0
Return code = 0
***** stdout *****
hdisk9
*** no stderr ****
----------------
Time: 0 LEDS: 0x539
Number of running methods: 0
----------------
Attempting to configure device 'hdisk9'
Time: 0 LEDS: 0x25b4
Invoking /etc/methods/cfgscsidisk -l hdisk9
Number of running methods: 1
----------------
Completed method for: hdisk9, Elapsed time = 1
Return code = 0
…
aix22 #
```

(Über die Option „*-l vscsi0*“ wurde der Config-Manager angewiesen den Geräte-Teilbaum ab dem Gerät *vscsi0* neu zu scannen.)

In der Ausgabe von *cfgmgr* ist zu sehen, das die Platte *hdisk9* erkannt worden ist. Die Platte ist vom Typ „*Virtual SCSI Disk Drive*“:

```
aix22 # lscfg -l hdisk9
  hdisk9           U9009.22A.8991971-V5-C11-T1-L8100000000000000  Virtual SCSI Disk Drive
aix22 #
```

Über den Physical Location Code lässt sich auch die Zuordnung zum entsprechenden virtuellen Target Device auf dem Virtual-I/O-Server herstellen. Der Teil „*V5-C11-T1*“ ist der virtuelle Slot *11* der LPAR, dort wurde der virtuelle SCSI-Client Adapter angelegt. Der hintere Teil „*L8100000000000000*“ gibt die LUN-ID an, das ist der Wert des Attributs *LogicalUnitAddr* des virtuellen Target Devices *vtscsi0* auf dem Virtual-I/O-Server.

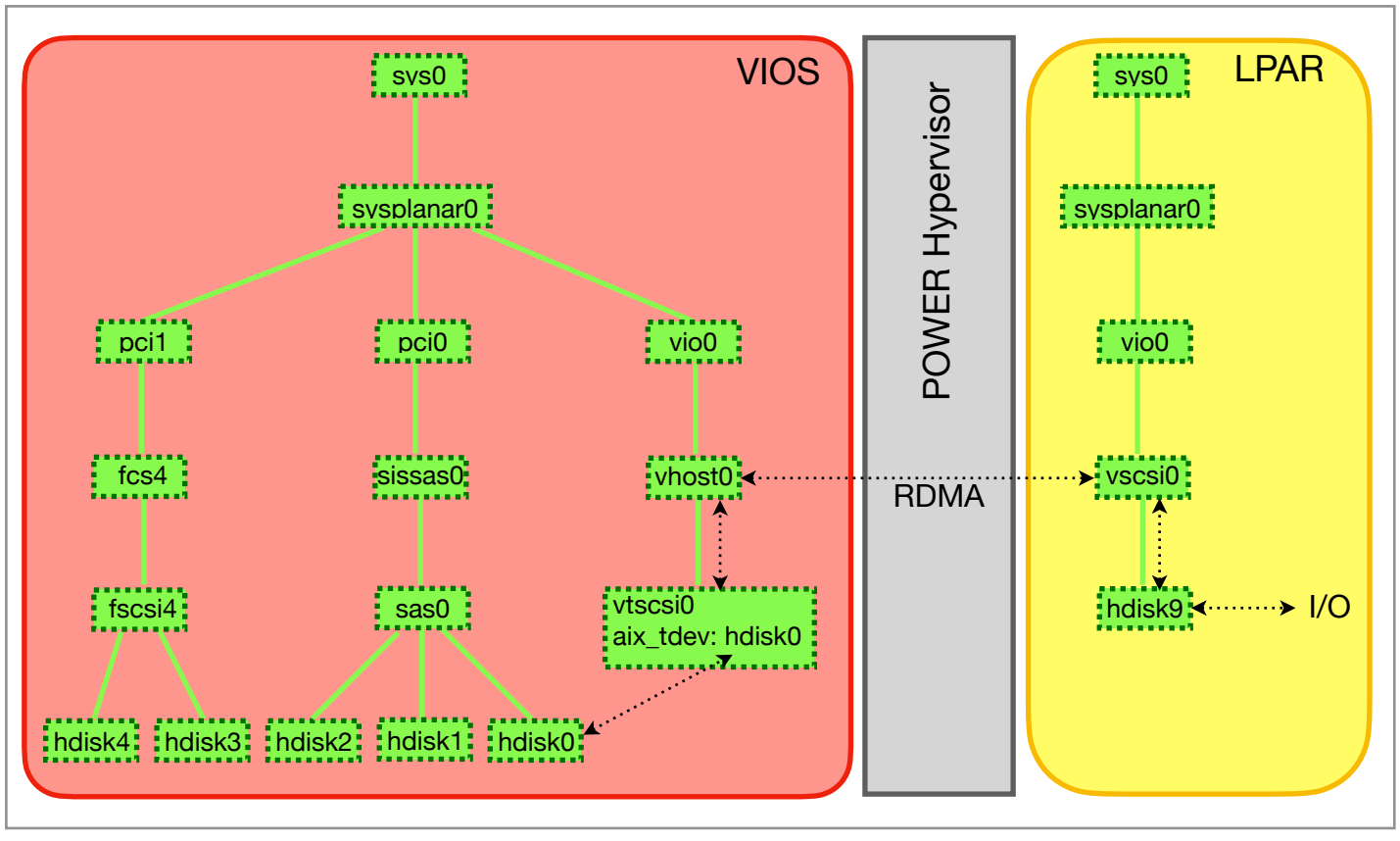


Bild 7.16: Gerätebäume auf Virtual-I/O-Server und Client-LPAR und I/O-Pfad.

Das Bild 7.16 zeigt den für VSCSI relevanten Teil der Gerätebäume auf Virtual-I/O-Server und Client-LPAR. Zusätzlich ist auch der I/O-Pfad beim Zugriff auf eine VSCSI-LUN (*hdisk9*) auf der Client-LPAR eingezeichnet.

Der Default Name *vtscsi0* für das virtuelle Target Device gibt keinen Hinweis welchem Client dieses zugeordnet ist. Bei ein paar hundert VSCSI-Mappings kann es von Vorteil sein einem virtuellen Target Device einen beschreibenden Namen zu geben, der darauf hinweist zu welchem Client das Gerät gehört und eventuell auch welchem Gerät (*hdisk*) dieses auf dem Client entspricht. Beim Mappen mit „*vios map*“ kann optional ein Gerätename für das virtuelle Target Device angegeben werden. Im folgenden Beispiel wird das Logical Volume *lv00* dem virtuellen SCSI-Server Adapter *vhost0* zugeordnet, wobei dem erzeugten virtuellen Target Device der Name *aix22_hd01* gegeben wird:

```
$ vios map ms03-vio1 lv00 vhost0 aix22_hd01
$
```

Das Logical Volume muß vor dem Mapping schon existieren, es wird durch das Mapping nicht angelegt! Der Client-LPAR *aix22* sind jetzt 2 Geräte zugeordnet:

```
$ vios lsvscsi ms03-vio1 vhost0
VTD         STATUS     BACKING  BDPHYSLOC  MIRRORED  LUN
aix22_hd01  Available  lv00     -          N/A       0x8200000000000000
vtscsi0     Available  hdisk0   -          N/A       0x8100000000000000
$
```

Auf der AIX LPAR ist ein weiterer Lauf des Config-Managers notwendig, damit AIX das neue Gerät erkennt.

Hinweis: Sollen Logical Volumes oder Dateien als Backing-Devices (Ziel des virtuellen Target Devices) zugeordnet werden, dann geht dies um einiges komfortabler über die Verwendung von Storage-Pools. Diese werden in einem eigenen Unterkapitel in Kapitel 8 ausführlich besprochen.

7.5.3. Entfernen eines VSCSI-Mappings

Das Mapping zwischen Backing-Device und *vhost*-Adapter kann jederzeit wieder aufgehoben werden. Gibt es auf der Client-LPAR keinen Pfad über einen anderen Adapter, ist ein Zugriff auf das Gerät dann nicht mehr möglich.

Die Client-LPARs hat die folgenden VSCSI-Mappings auf dem Virtual-I/O-Server:

```
$ vios lsvscsi ms03-vio2  vhost0
VTD      STATUS     BACKING  BDPHYSLOC  MIRRORED  LUN
vtscsi0  Available  hdisk3   -          N/A       0x8100000000000000
vtscsi1  Available  hdisk4   -          N/A       0x8200000000000000
$
```

Das Mapping für das Backing-Device *hdisk3* soll auf dem Virtual-I/O-Server *ms03-vio2* entfernt werden. Die zugehörige Client-LPAR zum virtuellen SCSI-Server Adapter lässt sich mit „*vios lsvscsi*“ leicht ermitteln:

```
$ vios lsvscsi ms03-vio2
SVSA    SLOT  CLIENT        LUNS
vhost0  C33   aix22(5)      2
vhost1  C34   aixsap01(9)   7
vhost2  C35   aixdbp02(11)  21
vhost3  C36   aixdbi02(13)  13
$
```

Die zugehörige Client-LPAR ist *aix22* (LPAR-ID *5*) und der Adapter *vhost0* ist im virtuellen Slot *33* des Virtual-I/O-Servers. Der zugehörige virtuelle SCSI-Client Adapter lässt sich dann der Ausgabe von „*lpar lsvslot*“ entnehmen:

```
$ lpar lsvslot aix22
SLOT  REQ  ADAPTER_TYPE   STATE  DATA
0     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
1     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
5     No   eth            1      PVID=100 VLANS= ETHERNET0 1DC8DB485D1E
10    No   fc/client      1      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
11    No   scsi/client    1      remote: ms03-vio1(1)/33
20    No   fc/client      1      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
21    No   scsi/client    1      remote: ms03-vio2(2)/33
$
```

Der virtuelle SCSI-Client Adapter in Slot *21* der Client-LPAR *aix22* ist mit dem virtuellen SCSI-Server Adapter in Slot *33* des Virtual-I/O-Servers *ms03-vio2* verbunden. Anhand der Slot-Nummer lässt sich das zugehörige *vscsi*-Gerät in der Client-LPAR identifizieren:

```
aix22 # lscfg -l vscsi\*
  vscsi0           U9009.22A.8991971-V5-C11-T1  Virtual SCSI Client Adapter
  vscsi1           U9009.22A.8991971-V5-C21-T1  Virtual SCSI Client Adapter
aix22 #
```

Das zu entfernende Mapping geht also über den virtuellen SCSI-Client Adapter *vscsi1* der LPAR, der Physical Location Code enthält die Slot-Nummer *21*: *V5-C21-T1*.

Als nächstes muß die zu *hdisk3* auf dem Virtual-I/O-Server gehörende Platte auf der Client-LPAR ermittelt werden. Das virtuelle Target-Device *vtscsi0* des Virtual-I/O-Servers besitzt die LUN-ID *0x8100000000000000*. Die dazu gehörende *hdisk* lässt sich mittels *lscfg* herausfinden:

```
aix22 # lscfg -l hdisk\* |grep 81
  hdisk9      U9009.22A.8991971-V5-C11-T1-L8100000000000000        Virtual SCSI Disk Drive
aix22 #
```

Es ist die *hdisk9* der Client-LPAR. Wir überprüfen ob diese Pfade über andere Adapter besitzt:

```
aix22 # lspath -l hdisk9
Enabled hdisk9 vscsi0
Enabled hdisk9 vscsi1
aix22 #
```

Neben dem Pfad über den Adapter *vscsi1* hat die Platte noch einen weiteren aktiven (*Enabled*) Pfad über den Adapter *vscsi0*. Damit die Client-LPAR den Pfad über den Adapter *vscsi1* nicht mehr verwendet (dieser wird ja durch das Entfernen des Mappings nicht mehr funktionieren), löschen wir den Pfad über den Adapter *vscsi1*:

```
aix22 # rmpath -dl hdisk9 -p vscsi1
path Deleted
aix22 #
```

Sämtliches I/O zu *hdisk9* geht jetzt ausschließlich über den Adapter *vscsi0*!

Jetzt kann als Letztes das Mapping der *hdisk3* zum Adapter *vhost0* auf dem Virtual-I/O-Server entfernt werden:

```
$ vios unmap ms03-vio2 vtscsi0
$
```

(Anstelle des virtuellen Target-Devices *vtscsi0* kann auch das Backing-Device *hdisk3* angegeben werden.)

Besitzt die Platte in der Client-LPAR für welche auf einem Virtual-I/O-Server das Mapping entfernt werden soll keine Pfade über andere Adapter, muß ein anderes Vorgehen gewählt werden. Um dieses zu zeigen, entfernen wir auch das zweite Mapping für die *hdisk9* der Client-LPAR *aix22*. Nachdem oben einer der beiden Pfade entfernt wurde, hat die LPAR nur noch einen aktiven Pfad:

```
aix22 # lspath -l hdisk9
Enabled hdisk9 vscsi0
aix22 #
```

Der letzte Pfad einer *hdisk* lässt sich nicht mit *rmpath* entfernen:

```
aix22 # rmpath -dl hdisk9 -p vscsi0
Method error (/etc/methods/ucfgdevice):
        0514-076 Cannot perform the requested function because the specified
                 paths include all paths for the device.

aix22 #
```

Die Platte muß in diesem komplett aus dem Betriebssystem entfernt werden. Das geht natürlich nur, wenn sie nicht in Benutzung ist. In unserem Beispiel ist die Platte nicht in Benutzung:

```
aix22 # lspv | grep hdisk9
hdisk9          00dead00beef000e                    None
aix22 #
```

Die *hdisk9* gehört zu keiner Volume Group, sie kann also ohne Probleme aus dem Betriebssystem entfernt werden:

```
aix22 # rmdev -dl hdisk9
hdisk9 deleted
aix22 #
```

Anschließend kann das Mapping (jetzt auf *ms03-vio1*) entfernt werden:

```
$ vios unmap ms03-vio1 vtscsi0
$
```

7.5.4. Wegnehmen eines virtuellen SCSI Adapters

Soll ein virtueller SCSI Adapter entfernt werden, müssen vorher auf der Client-LPAR alle Pfade über diesen Adapter entfernt werden, oder sogar die entsprechenden Platten komplett aus dem Betriebssystem entfernt werden.

Wir demonstrieren das Vorgehen wieder am Beispiel der LPAR *aix22* und entfernen dort zunächst den virtuellen SCSI Adapter im Slot *11* der Client-LPAR:

```
$ lpar lsvslot aix22
SLOT  REQ  ADAPTER TYPE   STATE  DATA
0     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
1     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
5     No   eth            1      PVID=100 VLANS= ETHERNET0 1DC8DB485D1E
10    No   fc/client      1      remote: ms03-vio1(1)/5 c05076030aba0002,c05076030aba0003
11    No   scsi/client    1      remote: ms03-vio1(1)/33
20    No   fc/client      1      remote: ms03-vio2(2)/4 c05076030aba0000,c05076030aba0001
21    No   scsi/client    1      remote: ms03-vio2(2)/33
$
```

Der zugehörige virtuelle SCSI-Server Adapter befindet sich auf dem Virtual-I/O-Server *ms03-vio1* im Slot *33*. Zunächst schauen wir uns aber an, welche Platten auf der Client-LPAR den virtuellen SCSI-Client Adapter in Slot *11* verwenden. Dazu benötigen wir den Gerätenamen des entsprechenden *vscsi*-Adapters unter AIX:

```
aix22 # lscfg -l vscsi\*
  vscsi0           U9009.22A.8991971-V5-C11-T1  Virtual SCSI Client Adapter
  vscsi1           U9009.22A.8991971-V5-C21-T1  Virtual SCSI Client Adapter
aix22 #
```

Der Adapter *vscsi0* befindet sich im virtuellen Slot *11* (*V5-C11-T1*). Welche Platten Pfade über diesen Adapter besitzen lässt sich mit dem Kommando *lspath* herausfinden:

```
aix22 # lspath -p vscsi0
Enabled hdisk9  vscsi0
Enabled hdisk10 vscsi0
aix22 #
```

Beide Platten haben aber noch Pfade über den Adapter *vscsi1*:

```
aix22 # lspath -l hdisk9
Enabled hdisk9 vscsi0
Enabled hdisk9 vscsi1
aix22 # lspath -l hdisk10
Enabled hdisk10 vscsi0
Enabled hdisk10 vscsi1
aix22 #
```

Es müssen also jeweils nur die Pfade der beiden Platten über den Adapter *vscsi0* entfernt werden. Dies kann man entweder für jede der Platten einzeln machen, so wie im letzten Unterkapitel oder für alle betroffenen Platten zusammen. Wir haben uns hier für die letzte Variante entschieden:

```
aix22 # rmpath -dp vscsi0
paths Deleted
aix22 #
```

Auf die beiden Platten kann jetzt nur noch über die verbleibenden Pfade über den Adapter *vscsi1* zugegriffen werden.

Damit der virtuelle SCSI-Client Adapter entfernt werden kann, muß das zugehörige Gerät *vscsi0* aus dem Betriebssystem entfernt werden:

```
aix22 # rmdev -dl vscsi0
vscsi0 deleted
aix22 #
```

Als nächstes müssen die Mappings auf dem zugehörigen virtuellen SCSI-Server Adapter entfernt werden. Dazu schauen wir uns zunächst an welcher der *vhost*-Adapter auf dem Virtual-I/O-Server den Slot *33* verwendet:

```
$ vios lsvscsi ms03-vio1
SVSA    SLOT  CLIENT        LUNS
vhost0  C33   aix22(5)      2
vhost1  C34   aixsap01(9)   7
vhost2  C35   aixdbp02(11)  21
vhost3  C36   aixdbi02(13)  13
$
```

Es ist gleich der erste Adapter in der Ausgabe - *vhost0*. Über diesen sind 2 LUNs gemappt. Welche das sind, kann durch Angabe des Adapters angezeigt werden:

```
$ vios lsvscsi ms03-vio1 vhost0
VTD      STATUS     BACKING  BDPHYSLOC  MIRRORED  LUN
vtscsi0  Available  hdisk3   -          N/A       0x8100000000000000
vtscsi1  Available  hdisk4   -          N/A       0x8200000000000000
$
```

Beide Mappings müssen entfernt werden:

```
$ vios unmap ms03-vio1 vtscsi0
$ vios unmap ms03-vio1 vtscsi1
$
```

Als letztes können nun der virtuelle SCSI-Client Adapter und der zugehörige virtuelle SCSI-Server Adapter entfernt werden:

```
$ lpar rmscsi aix22 11
aix22: slot 11 -> ms03-vio1/33 removed by DLPAR operation
aix22: slot 11 -> ms03-vio1/33 removed from current profile (standard)
ms03-vio1: slot 33 -> aix22/11 removed by DLPAR operation
$
```

Das Kommando entfernt den virtuellen SCSI-Client Adapter auf der Client-LPAR, als auch den zugehörigen virtuellen SCSI-Server Adapter auf dem Virtual-I/O-Server.

Im Falle das eine oder mehrere Platten auf der LPAR nur Pfade über den zu entfernenden virtuellen SCSI-Client Adapter haben, müssen die Platten aus dem Betriebssystem entfernt werden. Ohne Adapter kann auf die Platten dann ja nicht mehr zugegriffen werden. Wir zeigen auch diesen Fall, indem wir den verbleibenden zweiten virtuellen SCSI-Client Adapter ebenfalls entfernen. Betroffen sind auch hier nur die beiden Platten *hdisk9* und *hdisk10*:

```
aix22 # lspath -p vscsi1
Enabled hdisk9  vscsi1
Enabled hdisk10 vscsi1
aix22 #
```

Pfade über einen weiteren Adapter haben beide Platten nicht:

```
aix22 # lspath -l hdisk9
Enabled hdisk9 vscsi1
aix22 # lspath -l hdisk10
Enabled hdisk10 vscsi1
aix22 #
```

Die Platten können nur aus dem Betriebssystem entfernt werden, wenn diese nicht in Benutzung sind. Dies ist in unserem Beispiel aber der Fall:

```
aix22 # lspv | grep hdisk9
hdisk9          00dead00beef000e                    None
aix22 # lspv | grep hdisk10
hdisk10         00dead00beef0001                    None
aix22 #
```

Beide Platten gehören zu keiner Volume Gruppe und können daher entfernt werden:

```
aix22 # rmdev -dl hdisk9
hdisk9 deleted
aix22 # rmdev -dl hdisk10
hdisk10 deleted
aix22 #
```

Damit der virtuelle SCSI-Client Adapter entfernt werden kann, muß das zugehörige Gerät *vscsi1* aus dem Betriebssystem entfernt werden:

```
aix22 # rmdev -dl vscsi1
vscsi1 deleted
aix22 #
```

Das Löschen der Platten und des *vscsi*-Gerätes kann auch in einer Operation durchgeführt werden, indem man die Option „*-R*“ (*recursive*) von *rmdev* verwendet:

```
aix22 # rmdev -Rdl vscsi1
hdisk9 deleted
hdisk10 deleted
vscsi1 deleted
aix22 #
```

Nachdem nun die gemappten Platten in der Client-LPAR nicht mehr in Verwendung sind, kann das VSCSI-Mapping auf dem Virtual-I/O-Server entfernt werden. Der zugehörige virtuelle SCSI-Server Adapter ist in Slot *33* auf dem Virtual-I/O-Server *ms03-vio2*:

```
$ vios lsvscsi ms03-vio2
SVSA    SLOT  CLIENT        LUNS
vhost0  C33   aix22(5)      2
vhost1  C34   aixsap01(9)   7
vhost2  C35   aixdbp02(11)  21
vhost3  C36   aixdbi02(13)  13
$
```

Auch hier ist der Adapter wieder der *vhost0*, mit 2 zugeordneten Backing-Devices:

```
$ vios lsvscsi ms03-vio2 vhost0
VTD      STATUS     BACKING  BDPHYSLOC  MIRRORED  LUN
vtscsi0  Available  hdisk3   -          N/A       0x8100000000000000
vtscsi1  Available  hdisk4   -          N/A       0x8200000000000000
$
```

Wir entfernen die beiden Mappings wieder mit „*vios unmap*“:

```
$ vios unmap ms03-vio2 vtscsi0
$ vios unmap ms03-vio2 vtscsi1
$
```

Abschließend können jetzt die virtuellen SCSI Adapter entfernt werden (Client- und Server-Adapter):

```
$ lpar rmscsi aix22 21
aix22: slot 21 -> ms03-vio2/33 removed by DLPAR operation
aix22: slot 21 -> ms03-vio2/33 removed from current profile (standard)
ms03-vio2: slot 33 -> aix22/21 removed by DLPAR operation
$
```

7.6. SR-IOV

Single Root I/O Virtualization oder kurz SR-IOV ist eine PCI Express (PCIe) Spezifikation, welche das Teilen (Sharing) von PCI Express Adaptern unter mehreren Betriebssystemen erlaubt. Ähnlich wie bei Micro-Partitioning, bekommen mehrere Gast-Betriebssysteme (LPARs) Anteile an einem physikalischen PCIe Adapter. Der große Vorteil im Hinblick auf Performance besteht darin, das die Virtualisierung in Hardware von dem physikalischen PCIe Adapter komplett übernommen wird. Ein Virtual-I/O-Server, wie für andere Virtualisierungs-Techniken benötigt, ist

für SR-IOV nicht notwendig. Die LPARs können den PCIe Adapter direkt ansprechen und verwenden. Dies spart insbesondere Prozessor-Ressourcen und reduziert zusätzlich die Latenz beim I/O.

Für die Unterstützung von Virtualizierung werden physikalische Funktionen (PF) und virtuelle Funktionen (VF) eingeführt. Physikalische Funktionen unterstützen alle Merkmale von PCIe, insbesondere die Konfiguration des physikalischen Geräts (Chips) und I/O. Virtuelle Funktionen sind leichtgewichtige Versionen der physikalischen Funktionen, welche keine Konfiguration des physikalischen Gerätes (Chips) erlauben, sondern nur I/O unterstützen. In Bild 7.17 ist dies schematisch dargestellt.

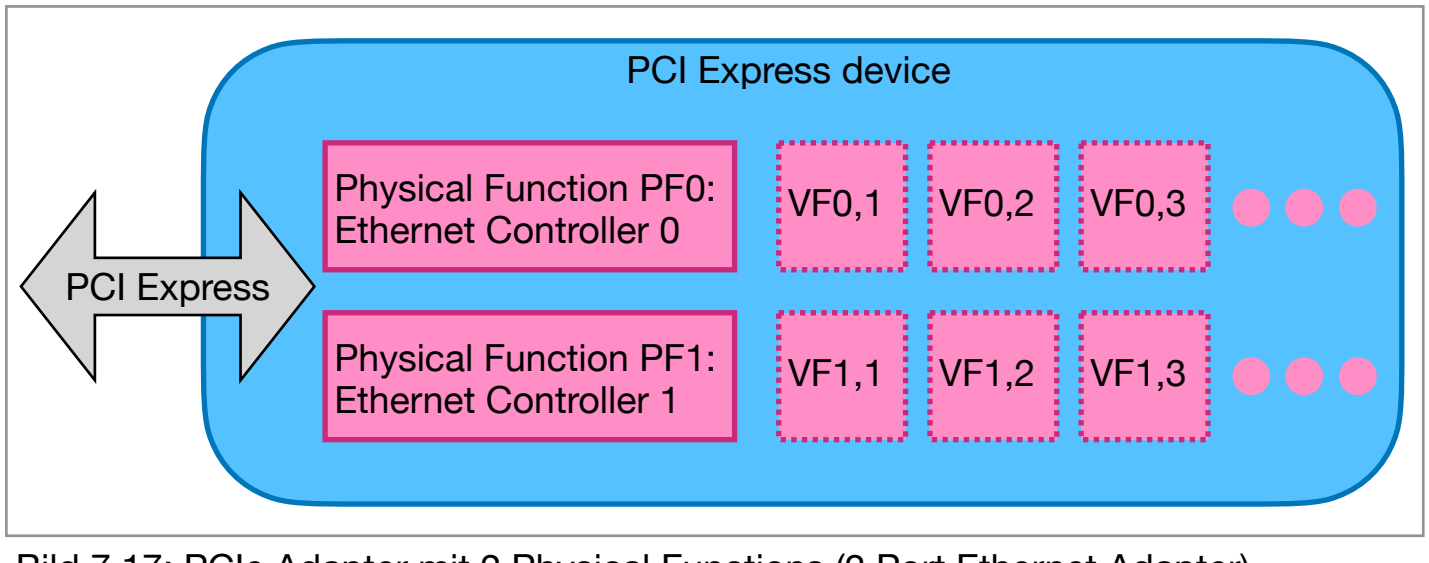


Bild 7.17: PCIe Adapter mit 2 Physical Functions (2 Port Ethernet Adapter).

Die physikalische Funktion *PF0* erlaubt die Konfiguration des Ethernet Controllers für den ersten Port der PCIe Karte, außerdem unterstützt *PF0* auch I/O (Transfer von Ethernet Paketen) über den ersten Port der Karte. Die virtuellen Funktionen *VF0,x* erlauben nur I/O aber keine Umkonfiguration des physikalischen Ethernet Ports.

Einer LPAR kann eine der virtuellen Funktionen zugeordnet werden, womit die LPAR dann die Möglichkeit besitzt, I/O über das zugeordnete physikalische Gerät zu machen.

Aktuell unterstützt PowerVM nur SR-IOV fähige Ethernet Adapter.

7.6.1. Aktivieren des Shared Modes

SR-IOV fähige PCIe Karten befinden sich zunächst in der Betriebsart „*dedicated*“. Das entspricht klassischen Karten ohne SR-IOV Unterstützung. Die Karte muß dann einer LPAR (in der Regel einem Virtual-I/O-Server) zugewiesen werden und wird dann exklusiv von dieser benutzt. Ein direkter Zugriff von weiteren LPARs ist dann nicht möglich.

Eine List der SR-IOV fähigen PCIe Adapter lässt sich mit Hilfe des Kommandos „*ms lssriov*“ (*list SR-IOV*) anzeigen:

```
$ ms lssriov ms03
PHYS_LOC                  SLOT_ID   CONFIG_STATE  SRIOV_STATUS  ADAPTER_ID  PHYS_PORTS
LOGICAL_PORTS
U78AA.001.VYRGU0Q-P1-C4   21010020  dedicated     null          null        null
null
U78AA.001.VYRGU0Q-P1-C6   2102001b  dedicated     null          null        null
null
U78AA.001.VYRGU0Q-P1-C11  21020013  dedicated     null          null        null
null
```

```
U78AA.001.VYRGU0Q-P1-C12  21030014  dedicated     null          null        null
null
$
```

Es gibt 4 PCIe Adapter-Karten in den Slots *P1-C4*, *P1-C6*, *P1-C11* und *P1-C12*. Alle 4 Karten sind noch im *dedicated* Modus (Spalte *CONFIG_STATE*). Um die Virtualizierung mit SR-IOV nutzen zu können, müssen die Karten erst einmal in den sogenannten „*Shared Mode*“ gebracht werden. Dies geschieht mit Hilfe des Kommandos „*ms chsriov*“ (*change SR-IOV*), wobei als Argument die Slot-ID (Spalte *SLOT_ID*) oder ein eindeutiger Suffix des Physical Location Codes angegeben werden muß:

```
$ ms chsriov ms03 21010020 shared
$ ms chsriov ms03 P1-C6 shared
$ ms chsriov ms03 21020013 shared
$ ms chsriov ms03 P1-C12 shared
$
```

Jeder SR-IOV Adapter bekommt bei der Umstellung in den *Shared Mode* eine eindeutige Adapter-ID zugewiesen. Die Adapter werden einfach fortlaufend nummeriert. Möchte man für einen Adapter eine bestimmte Adapter-ID verwenden, kann das Attribut *adapter_id* verwendet werden:

```
$ ms chsriov ms03 21020013 shared adapter_id=1
$ ms chsriov ms03 P1-C6 shared adapter_id=2
$ ms chsriov ms03 21010020 shared adapter_id=3
$ ms chsriov ms03 P1-C12 shared adapter_id=4
$
```

Bei der Umstellung auf den Shared Mode werden die Adapter neu initialisiert. Dies kann einige Zeit in Anspruch nehmen. Während ein Adapter initialisiert wird, sind nicht alle Informationen verfügbar, wie die folgende Ausgabe zeigt:

```
$ ms lssriov ms03
PHYS_LOC                  SLOT_ID   CONFIG_STATE  SRIOV_STATUS  ADAPTER_ID   PHYS_PORTS
LOGICAL_PORTS
U78AA.001.VYRGU0Q-P1-C4   21010020  sriov         initializing  unavailable  unavailable
unavailable
U78AA.001.VYRGU0Q-P1-C6   2102001b  sriov         running       2            4
48
U78AA.001.VYRGU0Q-P1-C11  21020013  sriov         initializing  unavailable  unavailable
unavailable
U78AA.001.VYRGU0Q-P1-C12  21030014  sriov         missing       unavailable  unavailable
unavailable
$
```

Der Adapter in Slot *P1-C6* ist initialisiert und betriebsbereit (running), die beiden Adapter in Slot *P1-C4* und *P1-C11* sind noch beim Initialisieren und der Adapter in Slot *P1-C12* hat noch nicht mit der Initialisierung begonnen.

Nach einigen Sekunden (in der Regel weniger als eine Minute) sind alle Adapter initialisiert und die Ausgabe sieht dann wie folgt aus:

```
$ ms lssriov ms03
PHYS_LOC                  SLOT_ID   CONFIG_STATE  SRIOV_STATUS  ADAPTER_ID  PHYS_PORTS
LOGICAL_PORTS
U78AA.001.VYRGU0Q-P1-C11  21020013  sriov         running       1           4           48
U78AA.001.VYRGU0Q-P1-C6   2102001b  sriov         running       2           4           48
```

```
U78AA.001.VYRGU0Q-P1-C4   21010020  sriov         running       3           4           48
U78AA.001.VYRGU0Q-P1-C12  21030014  sriov         running       4           4           48
$
```

Alle 4 Adapter besitzen jeweils 4 physikalische Ports (Spalte *PHYS_PORTS*) und unterstützen bis zu 48 logische Ports (*LOGICAL_PORTS*). Logische Ports können vom Administrator erzeugt werden, wobei diese dann gleichzeitig einer LPAR zugewiesen werden. Jedem logischen Port ist dabei eine der verfügbaren virtuellen Funktionen zugeordnet. Die physikalischen Funktionen können nur vom Hypervisor verwendet werden. Damit ist sichergestellt das LPARs keine Änderungen an der Konfiguration eines SR-IOV Adapters vornehmen können.

7.6.2. Konfiguration der physikalischen SR-IOV Ports

Die physikalischen Ports der SR-IOV Adapter lassen sich ebenfalls mit dem Kommando „*ms lssriov*“ anzeigen, dazu muß die Option „*-p*“ (*physical ports*) angegeben werden:

```
$ ms lssriov -p ms03
PHYS_PORT_LOC                 STATE  LABEL  TYPE  ADAPTER PPORT  USED  MAX  CONN_SPEED  MTU
U78AA.001.VYRGU0Q-P1-C11-T1   1      -      ethc  1       0      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C11-T2   1      -      ethc  1       1      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C11-T3   0      -      eth   1       2      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C11-T4   1      -      eth   1       3      0     4    1000        1500
U78AA.001.VYRGU0Q-P1-C6-T1    1      -      ethc  2       0      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C6-T2    1      -      ethc  2       1      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C6-T3    0      -      eth   2       2      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C6-T4    1      -      eth   2       3      0     4    1000        1500
U78AA.001.VYRGU0Q-P1-C4-T1    1      -      ethc  3       0      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C4-T2    1      -      ethc  3       1      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C4-T3    0      -      eth   3       2      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C4-T4    0      -      eth   3       3      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C12-T1   1      -      ethc  4       0      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C12-T2   1      -      ethc  4       1      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C12-T3   0      -      eth   4       2      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C12-T4   0      -      eth   4       3      0     4    0           1500
$
```

Hat ein physikalischer Port einen Link, dann ist das zum Einen an der Spalte *STATE* (Wert *1*) und zum Anderen an der Spalte *CONN_SPEED* (Mb/s) zu erkennen. Z.B. hat der Port *P1-C11-T1* (Adapter-ID *1*, Port-ID *0*) einen Link mit einer Geschwindigkeit von 10 Gb/s (10.000 Mb/s).

Für jeden physikalischen Port können eine Reihe von Attributen konfiguriert werden. Unter anderem kann die Geschwindigkeit (*conn_speed*), Fluß-Kontrolle und die MTU-Größe (*max_recv_packet_size*) konfiguriert werden. Die konfigurierbaren Attribute sind in der Online Hilfe aufgeführt und kurz beschrieben:

```
$ ms help chsriov
USAGE:
   ms [-h <hmc>] chsriov [-v] <ms> {<slot_id>|<physloc> {dedicated|shared} | {<adapter_id>
<phys_port_id>|<physloc>} <attributes>} [<attributes> ...]

DESCRIPTION:

Switches an SR-IOV adapter in a managed system either to dedicated or shared mode,
or sets attributes for an SR-IOV physical port.
An adapter can be specified either by the slot-ID or a unique suffix of the physical
```

```
location code.
A physical port can be specified either by the adapter-ID and physical port-ID or
a unque suffix of the physical location code of the port.

Attributes when switching an adapter to shared mode:
   adapter_id - 1-32, default: assign next available adapter ID

Attributes for an SR-IOV physical port:
   conn_speed - ethernet speed
      auto : autonegotation
      10 : 10 Mbps
      100 : 100 Mbps
      1000 : 1 Gbps
      10000 : 10 Gbps
      40000 : 40 Gbps
      100000 : 100 Gbps
   max_recv_packet_size - MTU
      1500 - 1500 byts
      9000 - 9000 bytes (jumbo frames)
   phys_port_label - label for the physical port
      1-16 characters
      none - to clear the label
   phys_port_sub_label - sublabel for the physical port
      1-8 characters
      none - to clear the sublabel
   recv_flow_control
      0 - disable
      1 - enable
   trans_flow_control
      0 - disable
      1 - enable
   veb_mode
      0 - disable virtual ethernet bridge mode
      1 - enable virtual ethernet bridge mode
   vepa_mode
      0 - disable virtual ethernet port aggregator mode
      1 - enable virtual ethernet port aggregator mode
   (see the IBM documentation for additional attributes)

…
$
```

Sehr nützlich ist die Möglichkeit die physikalischen Ports mit einem Label (*phys_port_label*) zu versehen. Das Label ist eine beliebige Zeichenkette (maximal 16 Zeichen), die der Administrator zuweisen kann, es kann z.B. verwendet werden um den Zweck des angeschlossenen Netzwerks kenntlich zu machen.

Die physikalischen Ports *P1-C11-T1* und *P1-C6-T1* gehen in unserem Beispiel beide ins Management-LAN. Wir weisen daher als Label die Zeichenkette *Mgmt* zu und setzen die MTU-Größe auf *9000* (Jumbo-Frames). Zum Setzen der Attribute wird wieder das Kommando „*ms chsriov*“ verwendet, wobei für physikalische Ports die Adapter-ID und die Port-ID des physikalischen Ports angegeben werden müssen. Alternativ kann aber auch ein eindeutiger Suffix des Physical Location Codes des Ports angegeben werden:

```
$ ms chsriov ms03 1 0 phys_port_label=Mgmt max_recv_packet_size=9000
$ ms chsriov ms03 P1-C6-T1 phys_port_label=Mgmt max_recv_packet_size=9000
$
```

Die Änderung der Attribute wirkt sich unmittelbar aus. Nicht alle Attribute können geändert werden, wenn ein physikalischer Port in Verwendung ist und schon Logische Ports erzeugt wurden.

Nach der Änderung der Attribute sehen die physikalischen Ports wie folgt aus:

```
$ ms lssriov -p ms03
PHYS_PORT_LOC                 STATE  LABEL  TYPE  ADAPTER PPORT  USED  MAX  CONN_SPEED  MTU
U78AA.001.VYRGU0Q-P1-C11-T1   1      Mgmt   ethc  1       0      0     20   10000       9000
U78AA.001.VYRGU0Q-P1-C11-T2   1      -      ethc  1       1      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C11-T3   0      -      eth   1       2      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C11-T4   1      -      eth   1       3      0     4    1000        1500
U78AA.001.VYRGU0Q-P1-C6-T1    1      Mgmt   ethc  2       0      0     20   10000       9000
U78AA.001.VYRGU0Q-P1-C6-T2    1      -      ethc  2       1      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C6-T3    0      -      eth   2       2      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C6-T4    1      -      eth   2       3      0     4    1000        1500
U78AA.001.VYRGU0Q-P1-C4-T1    1      -      ethc  3       0      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C4-T2    1      -      ethc  3       1      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C4-T3    0      -      eth   3       2      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C4-T4    0      -      eth   3       3      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C12-T1   1      -      ethc  4       0      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C12-T2   1      -      ethc  4       1      0     20   10000       1500
U78AA.001.VYRGU0Q-P1-C12-T3   0      -      eth   4       2      0     4    0           1500
U78AA.001.VYRGU0Q-P1-C12-T4   0      -      eth   4       3      0     4    0           1500
$
```

Es empfiehlt sich alle verwendeten physikalischen Ports mit einem entsprechenden Label zu versehen. Aus Gründen der Redundanz sollte jedes Netzwerk zweifach angebunden sein, so wie für das *Mgmt*-Netzwerk oben.

7.6.3. Hinzufügen von logischen SR-IOV Ports

Damit eine LPAR eine virtuelle Funktion eines SR-IOV Adapters benutzen kann, muß für die LPAR ein sogenannter logischer Port erzeugt werden. Welche logischen Ports es schon gibt, lässt sich mit dem Kommando „*ms lssriov*“ mit der Option „*-l*“ (*logical port*) anzeigen:

```
$ ms lssriov -l ms03
LOCATION_CODE  ADAPTER  PPORT  LPORT  LPAR  CAPACITY  CURR_MAC_ADDR  CLIENTS
$
```

Da die SR-IOV Adapter gerade erst auf Shared umgestellt wurden, gibt es natürlich bisher noch keine logischen Ports. Um einer LPAR einen logischen SR-IOV Port hinzuzufügen, wird das Kommando „*lpar addsriov*“ (*add SR-IOV logical port*) verwendet. Es muß neben der LPAR die Adapter-ID und die Port-ID des physikalischen Ports angegeben werden. Alternativ kann aber auch ein eindeutiger Suffix des Physical Location Codes des physikalischen Port angegeben werden:

```
$ lpar addsriov aix22 P1-C11-T1
$
```

Das Erzeugen kann einige wenige Sekunden dauern. Eine kurze Überprüfung zeigt, das tatsächlich ein logischer Port angelegt wurde:

```
$ ms lssriov -l ms03
LOCATION_CODE                  ADAPTER  PPORT  LPORT     LPAR   CAPACITY  CURR_MAC_ADDR
CLIENTS
U78AA.001.VYRGU0Q-P1-C11-T1-S1 1        0      27004001  aix22  2.0       a1b586737e00   -
$
```

Ähnlich wie bei einem Managed System für virtuelles Ethernet ist auch auf den SR-IOV Adaptern für jeden physikalischen Ethernet Port ein interner Switch implementiert, siehe Bild 7.18. Jedem logischen Port ist dabei eine der virtuellen Funktionen zugeordnet. Die zugehörigen LPARs greifen auf die logischen Ports über den PCI Express Bus (PCIe-Switch) direkt zu.

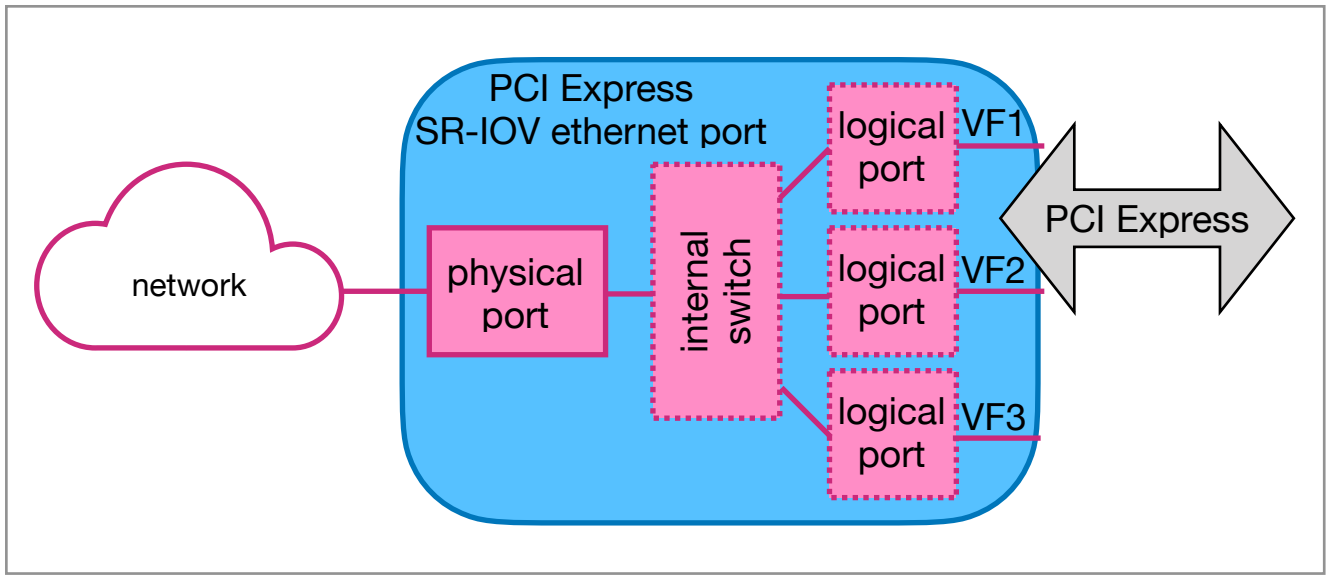


Bild 7.18: SR-IOV Ethernet Port mit internem Switch und 3 logischen Ports.

Eine LPAR kann ohne weiteres mehrere logischen SR-IOV Ports besitzen. Mit dem Kommando „*lpar lssriov*“ (*list SR-IOV logical ports*) lassen sich alle logischen Ports einer LPAR anzeigen:

```
$ lpar lssriov aix22
LPORT      REQ  ADAPTER  PPORT  CONFIG ID  CAPACITY  MAX CAPACITY  PVID  VLANS
                                  _                    _
CURR_MAC_ADDR  CLIENTS
27004001  Yes  1        0      0          2.0       100.0         0     all
a1b586737e00   -
$
```

Es gibt eine ganze Reihe von Attributen die für einen logischen Port gleich beim Anlegen angegeben werden können. Unter Anderem können die folgenden Eigenschaften konfiguriert werden:

- *capacity* - die garantierte Kapazität für den logischen Port.
- *port_vlan_id* - die VLAN-ID für nicht getaggte Pakete oder 0 um VLAN-Tagging auszuschalten.
- *promisc_mode* - promiscous Mode ein- oder ausschalten.

Die vollständige List der Attribute und ihre möglichen Werte kann man der Online Hilfe („*lpar help addsriov*“) entnehmen.

Als Beispiel fügen wir der LPAR *aix22* einen weiteren logischen Port mit Port VLAN-ID *55* und einer Kapazität von *20%* hinzu:

```
$ lpar addsriov aix22 P1-C4-T2 port_vlan_id=55 capacity=20
$
```

Der erzeugte logische Port bekommt damit einen garantierten Anteil von *20%* an der Bandbreite des physikalischen Ports *P1-C4-T2*! Die LPAR hat damit jetzt 2 logische SR-IOV Ports:

```
$ lpar lssriov aix22
LPORT     REQ  ADAPTER  PPORT  CONFIG_ID  CAPACITY  MAX_CAPACITY  PVID  VLANS
CURR_MAC_ADDR  CLIENTS
27004001  Yes  1        0      0          2.0       100.0         0     all
a1b586737e00   -
2700c003  Yes  3        2      1          20.0      100.0         55    all
a1b586737e01   -
$
```

Nachdem die logischen Ports mittels PowerVM Hypervisor der LPAR hinzugefügt wurden, erscheinen diese im Zustand *Defined*. Die logischen Ports tauchen unter AIX als *ent*-Devices auf, wie alle anderen Ethernet Adapter auch!

```
aix22 # lsdev -l ent\*
ent0 Available       Virtual I/O Ethernet Adapter (l-lan)
ent1 Defined   00-00 PCIe2 10GbE SFP+ SR 4-port Converged Network Adapter VF
(df1028e214100f04)
ent2 Defined   01-00 PCIe2 100/1000 Base-TX 4-port Converged Network Adapter VF
(df1028e214103c04)
aix22 #
```

Nach einem Lauf des Config-Managers sind die neuen *ent*-Devices im Zustand *Available* und können genau so benutzt werden, wie alle anderen Ethernet Adapter.

7.6.4. Ändern eines logischen SR-IOV Ports

Einige (wenige) Einstellungen eines logischen SR-IOV Ports lassen sich online ändern, unter Anderem

- *port_vlan_id* - die VLAN-ID für untagged Pakete
- *pvid_priority* - die Priorität von untagged Paketen

Das Kommando zum Ändern der Attribute lautet „*lpar chsriov*“ (*change SR-IOV logical port*). Neben der LPAR muß die ID des betreffenden logischen Port angegeben werden:

```
$ lpar chsriov aix22 2700c003 port_vlan_id=26
$
```

Welche Attribute online geändert werden können, kann in der Online Hilfe („*lpar help chsriov*“) nachgeschaut werden.

7.6.5. Wegnehmen von logischen SR-IOV Ports

Wird ein logischer SR-IOV Port nicht mehr benötigt, kann er natürlich auch wieder entfernt werden. Das zugehörige *ent*-Gerät darf aber im Betriebssystem nicht mehr in Verwendung sein!

Als Beispiel entfernen wir den logischen Port für *ent2* wieder. Falls noch eine IP-Adresse konfiguriert sein sollte, kann das Standard Interface *en2* mit dem Kommando *ifconfig* heruntergenommen werden:

```
aix22 # ifconfig en2 detach
aix22 #
```

Danach kann der Ethernet Adapter *ent2* aus dem Betriebssystem entfernt werden:

```
aix22 # rmdev -Rdl ent2
ent2 deleted
aix22 #
```

Nun kann der zugehörige logische Port mit dem Kommando „*lpar rmsriov*“ entfernt werden:

```
$ lpar rmsriov aix22 2700c003
$
```

Als Argument muß wieder neben der LPAR die *ID* des logischen Ports angegeben werden.

7.6.6. SR-IOV Adapter von Shared zurück auf Dedicated setzen

Die Betriebsart eines SR-IOV Adapters kann nur geändert werden, wenn keine logischen Ports vorhanden sind! Um also einen solchen Adapter zurück auf *dedicated* zu setzen, müssen alle logischen Ports entfernt werden. Die Liste der logischen Ports erhält man bei Verwendung der Option „*-l*“ (*logical ports*) beim Kommando „*ms lssriov*“:

```
$ ms lssriov -l ms03
LOCATION_CODE                    ADAPTER  PPORT  LPORT     LPAR   CAPACITY  CURR_MAC_ADDR
CLIENTS
U78AA.001.VYRGU0Q-P1-C11-T1-S1   1        0      27004001  aix22  2.0       a1b586737e00   -
U78AA.001.VYRGU0Q-P1-C11-T1-S3   3        2      2700c003  aix22  20.0      a1b586737e01   -
$
```

Wurden alle logischen Ports gelöscht, siehe voriges Unterkapitel, dann kann der SR-IOV Adapter mit Hilfe von „*ms chsriov*“ zurück auf *dedicated* gesetzt werden:

```
$ ms chsriov ms03 P1-C11-T1 dedicated
$
```

7.7. Virtual Network Interface Controller (vNIC)

Der große Nachteil von SR-IOV, sowie oben beschrieben, besteht darin, das das Verschieben (LPM) von LPARs mit logischen SR-IOV Ports nicht möglich ist. Nach der Einführung von SR-IOV auf Power Systemen gab es eine Reihe von Vorschlägen für Workarounds. Allerdings bedingen alle diese Workarounds zum Einen eine besondere Konfiguration und zum Anderen eine Reihe von Umkonfigurationen die vor und nach einer LPM-Operation durchzuführen sind. Im praktischen Alltag werden dadurch LPM-Operationen aber unnötig verkompliziert.

Mit der Einführung von vNICs können Client-LPARs SR-IOV Adapter benutzen und trotzdem LPM unterstützen. Wie schon bei VSCSI und VFC wird dazu ein Paar von Adaptern benutzt: auf der Client-LPAR wird der sogenannte vNIC Adapter in einem virtuellen Slot verwendet und auf einem Virtual-I/O-Server wird ein zugehöriger vNIC-Server Adapter verwendet. Der logische SR-IOV Port wird dem Virtual-I/O-Server zugeordnet. Der vNIC-Server Adapter, auch als vNIC-Backing-Device bezeichnet, dient dabei als Proxy für den logischen SR-IOV Port. Das Zusammenspiel der verschiedenen Adapter ist in Bild 7.19 dargestellt.

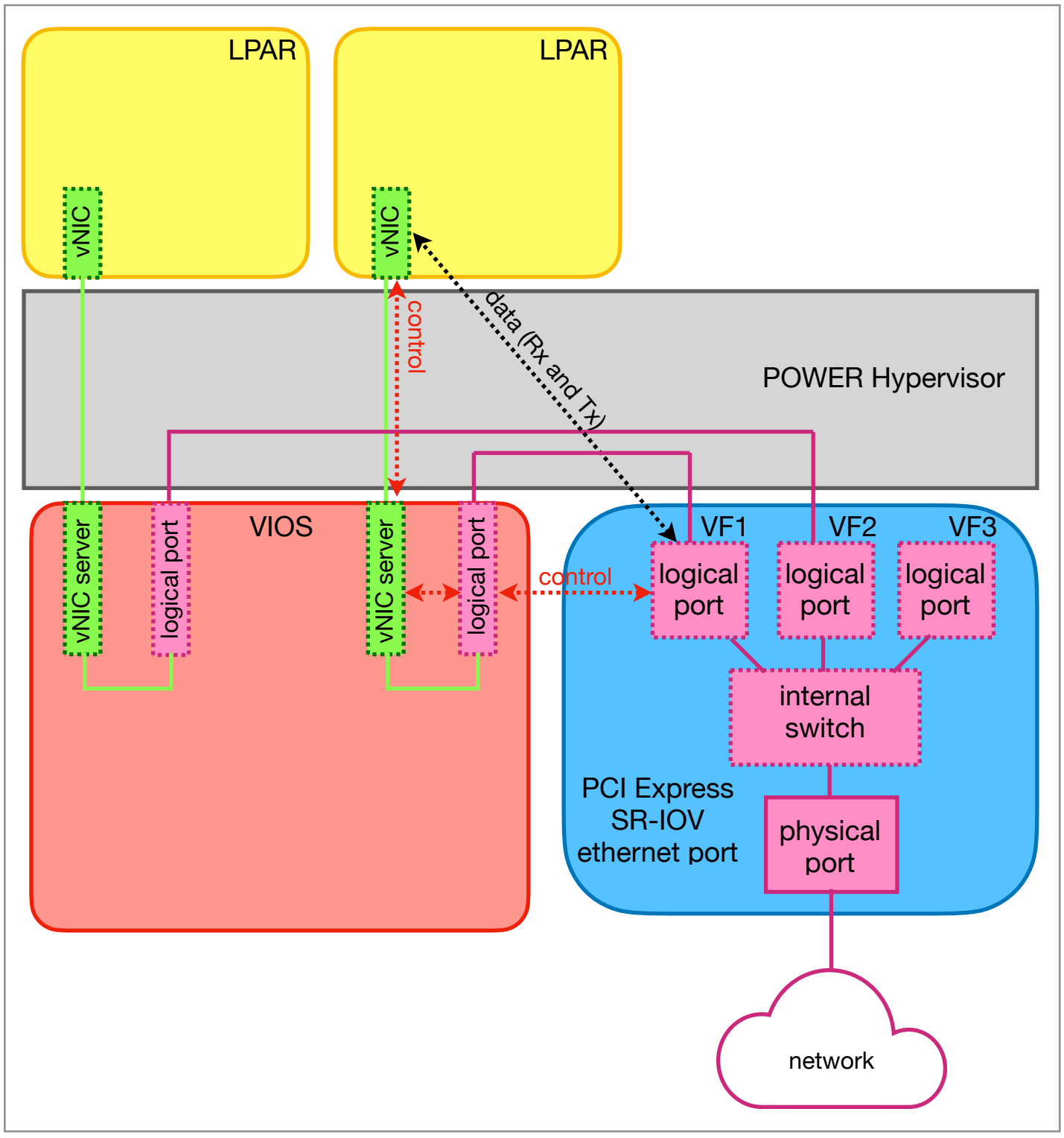


Bild 7.19: Kommunikationspfad von vNIC für Kontroll-Informationen und Daten.

Um eine gute Performance zu erreichen, werden nur Kontroll-Informationen vom vNIC Adapter der Client-Adapter zum vNIC Server Adapter übertragen, welche von diesem über den zugehörigen logischen SR-IOV Port zum entsprechenden logischen Port (Virtual Function) des SR-IOV Adapters übertragen werden. Die Daten selbst werden zwischen vNIC Client-Adapter und dem logischen Port des SR-IOV Adapters per DMA (Direct Memory Access) mit Hilfe des Hypervisors übertragen. Es findet also insbesondere kein Kopieren der Daten über den Virtual-I/O-Server statt. Der vNIC Adapter auf dem Client ist ein rein virtueller Adapter, daher funktioniert LPM auch problemlos. Der Client besitzt den logischen SR-IOV Port nicht und greift auch nicht selbst über den PCIe Bus (Switch) auf diesen zu.

7.7.1. Erzeugen eines vNIC Adapters

Obwohl das Konstrukt eines vNIC Adapters nicht ganz einfach ist, schließlich besteht es aus mindestens 3 Adaptern (vNIC-Client Adapter, vNIC-Server Adapter und logischer SR-IOV Port), lässt sich ein vNIC Adapter relativ leicht mit nur einem Kommando anlegen. Im einfachsten Fall muß lediglich der Virtual-I/O-Server für das vNIC-Backing-Device und der physikalische SR-IOV Port angegeben werden. Das Kommando zum Erzeugen eines vNIC Adapters ist „*lpar addvnic*“ (*add vNIC adapter*):

```
$ lpar addvnic aix22 6 ms03-vio1 P1-C11-T1
$
```

Hier wurde ein vNIC Adapter im virtuellen Slot Nummer *6* der Client-LPAR *aix22* angelegt. Das vNIC-Backing-Device wird auf dem Virtual-I/O-Server ms03-vio1 angelegt, zusammen mit dem logischen SR-IOV Port für den physikalischen SR-IOV Port mit Physical Location Code *P1-C11-T1*. Anstelle des Physical Location Codes kann auch die Adapter-ID und physikalische Port-ID des physikalischen SR-IOV Ports angegeben werden.

Auf einer AIX-LPAR muß ein Lauf des Config-Managers gestartet werden, damit der vNIC-Client Adapter verwendet werden kann. Der vNIC-Client Adapter bekommt, wie alle anderen Ethernet Adapter, einen Gerätenamen mit Präfix *ent*:

```
aix22 # cfgmgr
aix22 #
aix22 # lsdev -l ent\*
ent0 Available  Virtual I/O Ethernet Adapter (l-lan)
ent1 Available  Virtual NIC Client Adapter (vnic)
aix22 #
```

Die Konfiguration des neuen Ethernet Adapters funktioniert genauso wie für jeden anderen Ethernet Adapter!

Die existierenden vNIC Adapter einer LPAR können mit dem Kommando „*lpar lsvnic*“ (*list vNIC*) angezeigt werden:

```
$ lpar lsvnic aix22
LPAR_NAME  SLOT  MODE  FAILOVER  PVID  PRIORITY  MAC_ADDR      ALLOWED_VLAN_IDS  BKDEVS
aix22      6     ded   Yes       0     0         81253aa07306  all               1
$
```

Die Ausgabe zeigt das es nur einen vNIC Adapter im Slot *6* der LPAR gibt. Es gibt ein vNIC-Backing-Device (Spalte *BKDEVS*). Alle Attribute des vNIC Adapters haben Default Werte. Beim Erzeugen eines vNIC Adapters können die Attribute auf der Kommandozeile angegeben werden. Eine Übersicht der möglichen Attribute findet man z.B. in der Online Hilfe („*lpar help addvnic*“).

Die zugehörigen vNIC-Backing-Devices der vNIC Adapter können ebenfalls aufgelistet werden, dazu muß lediglich die Option „*-a*“ (*all vNIC backing devices*) bei „*lpar lsvnic*“ verwendet werden:

```
$ lpar lsvnic -a aix22
                          FAILOVER                                          PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PORT  PORT      CAPACITY  CAPACITY
aix22      6     Yes       50        1     Operational  ms03-vio1  1        0     27004005  2.0       100.0
$
```

Der Wert *Operational* in der Spalte *STATUS* zeigt das der zugehörige physikalische SR-IOV Port einen Link hat, es können also Netzwerk-Pakete gesendet und empfangen werden. Der zum vNIC-Backing-Device gehörende logische SR-IOV Port *27004005* besitzt eine garantierte Kapazität von *2%* des physikalischen Ports.

Der logische SR-IOV Port *27004005* wurde auf dem Virtual-I/O-Server *ms03-vio1* angelegt:

```
$ lpar lssriov ms03-vio1
LPORT     REQ  ADAPTER  PPORT  CONFIG_ID  CAPACITY  MAX_CAPACITY  PVID  VLANS  MAC_ADDR  CLIENTS
27004005  Yes  1        0      16384      2.0       100.0         0     all    a1b586737e00   aix22
$
```

Das zugehörige vNIC-Backing-Device auf dem Virtual-I/O-Server kann mit „*lpar lsvnicbkdev*“ angezeigt werden:

```
$ lpar lsvnicbkdev ms03-vio1
                                FAILOVER            PHYS  LOGICAL                MAX
LPAR_NAME  ACTV  STATUS         PRIORITY  ADAPTER   PORT  PORT      CAPACITY  CAPACITY
ms03-vio1  1     Operational    50        1         0     27004005  2.0       100.0
$
```

Die Ausgabe ist ähnlich zur Ausgabe von „*lpar lsvnic -a*“.

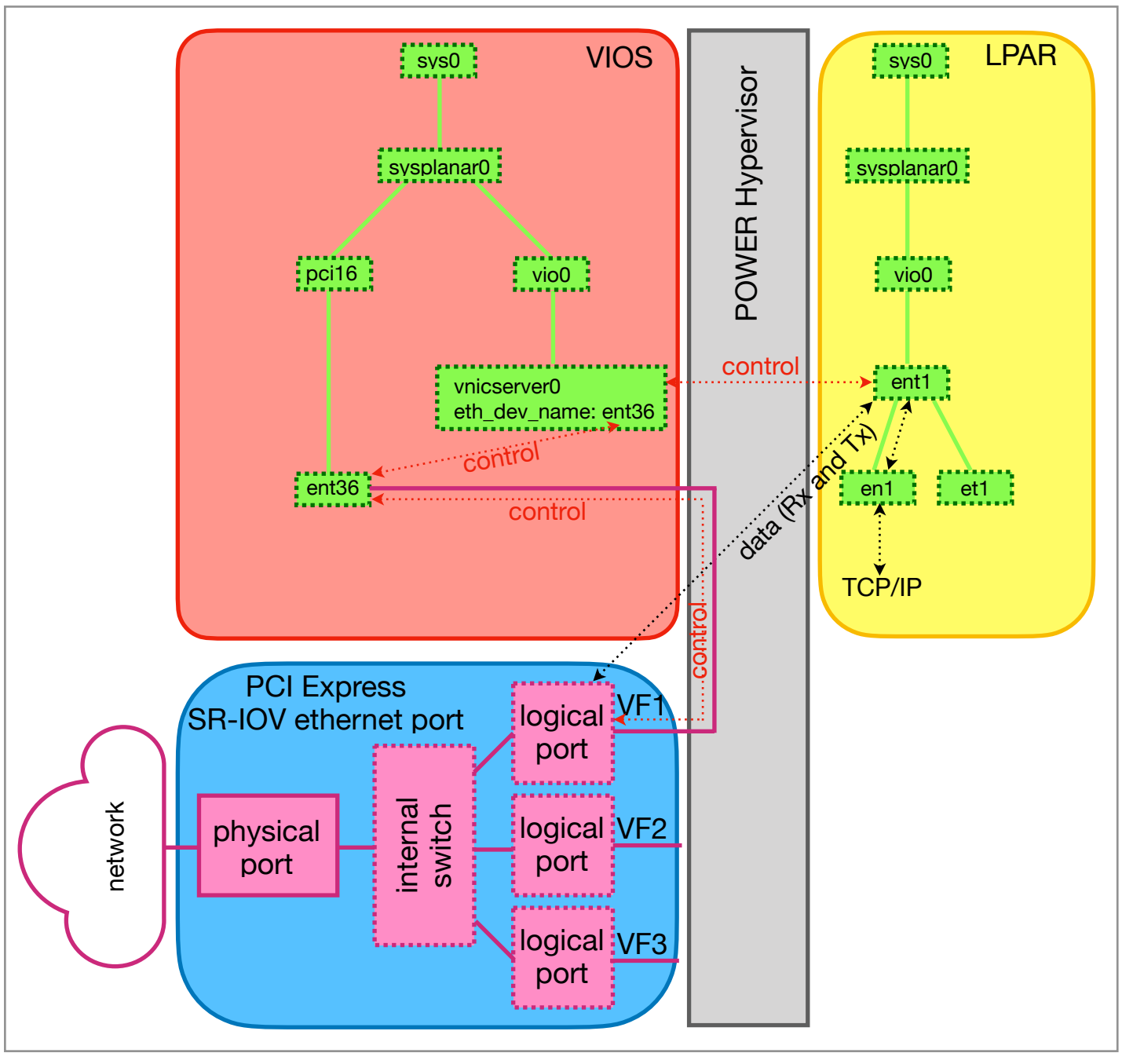


Bild 7.20: Gerätebäume auf Virtual-I/O-Server und Client-LPAR, sowie I/O-Pfad.

In Bild 7.20 sind Teile der Gerätebäume auf Virtual-I/O-Server und Client-LPAR gezeigt, sowie das Zusammenspiel der verschiedenen Geräte.

7.7.2. Ändern eines vNIC Adapters

Jeder vNIC Adapter besitzt eine Reihe von Attributen, die beim Erzeugen des vNIC Adapters gesetzt werden können, siehe „*lpar help addvnic*“. Einige der Attribute lassen sich aber auch nach dem Anlegen des vNIC Adapters online ändern, wie z.B.:

- *port_vlan_id* - Die VLAN-ID von untagged Paketen.
- *pvid_priority* - Priorität von untagged Paketen (QoS), mögliche Werte *0* bis *7* mit dem Default *0*.

Andere Attribute können nur im Profil geändert werden, wie z.B.:

- *allowed_vlan_ids* - Liste der erlaubten VLAN-IDs
- *mac_addr* - Die zu verwendende MAC-Adresse. Generell sollte die automatisch vergebene MAC-Adresse verwendet werden.

Die Attribute eines vNIC Adapters können mit dem Kommando „*lpar chvnic*“ (*change vNIC adapter*) geändert werden:

```
$ lpar chvnic aix22 6 pvid_priority=1 port_vlan_id=1200
$
```

Im Beispiel wurde die QoS-Priorität auf *1* und die PVID auf *1200* gesetzt. Sollen Änderungen nur im Profil der LPAR gemacht werden, muß die Option „*-p*“ mit dem Profil-Namen verwendet werden.

Eine vollständige Liste aller änderbaren Attribute ist in der Online Hilfe („*lpar help chvnic*“) aufgelistet.

7.7.3. Hinzufügen eines vNIC Backing-Devices

Für eine höhere Verfügbarkeit kann ein vNIC Adapter mehr als ein vNIC-Backing-Device haben. Fällt ein logischer SR-IOV Port aus (Link Down oder Ausfall des Virtual-I/O-Servers) übernimmt das nächste vNIC-Backing-Device bzw. dessen zugehöriger logischer SR-IOV Port. Zu einem Zeitpunkt ist immer nur eines der vNIC-Backing-Devices (logischer SR-IOV Port) aktiv. In Bild 7.21 ist eine Client-LPAR mit einem vNIC Adapter und 2 vNIC-Backing-Devices gezeigt. Zwei physikalische SR-IOV Ports sind an das gleiche externe Netzwerk angebunden. Für maximale Redundanz werden die beiden vNIC-Backing-Devices auf zwei verschiedenen Virtual-I/O-Servern angelegt. Egal ob ein physikalischer Ethernet Link ausfällt, oder ein Virtual-I/O-Server ausfällt, kann immer ein vNIC Failover auf eine funktionierende Kombination von Virtual-I/O-Server und logischem SR-IOV Port gemacht werden. Die vNIC-Backing-Devices überwachen den Status der logischen SR-IOV Ports und melden diesen in Form von Heartbeats regelmäßig an den Hypervisor. Bei einem Ausfall entscheidet der POWER Hypervisor welches noch funktionierende vNIC-Backing-Device aktiviert wird.

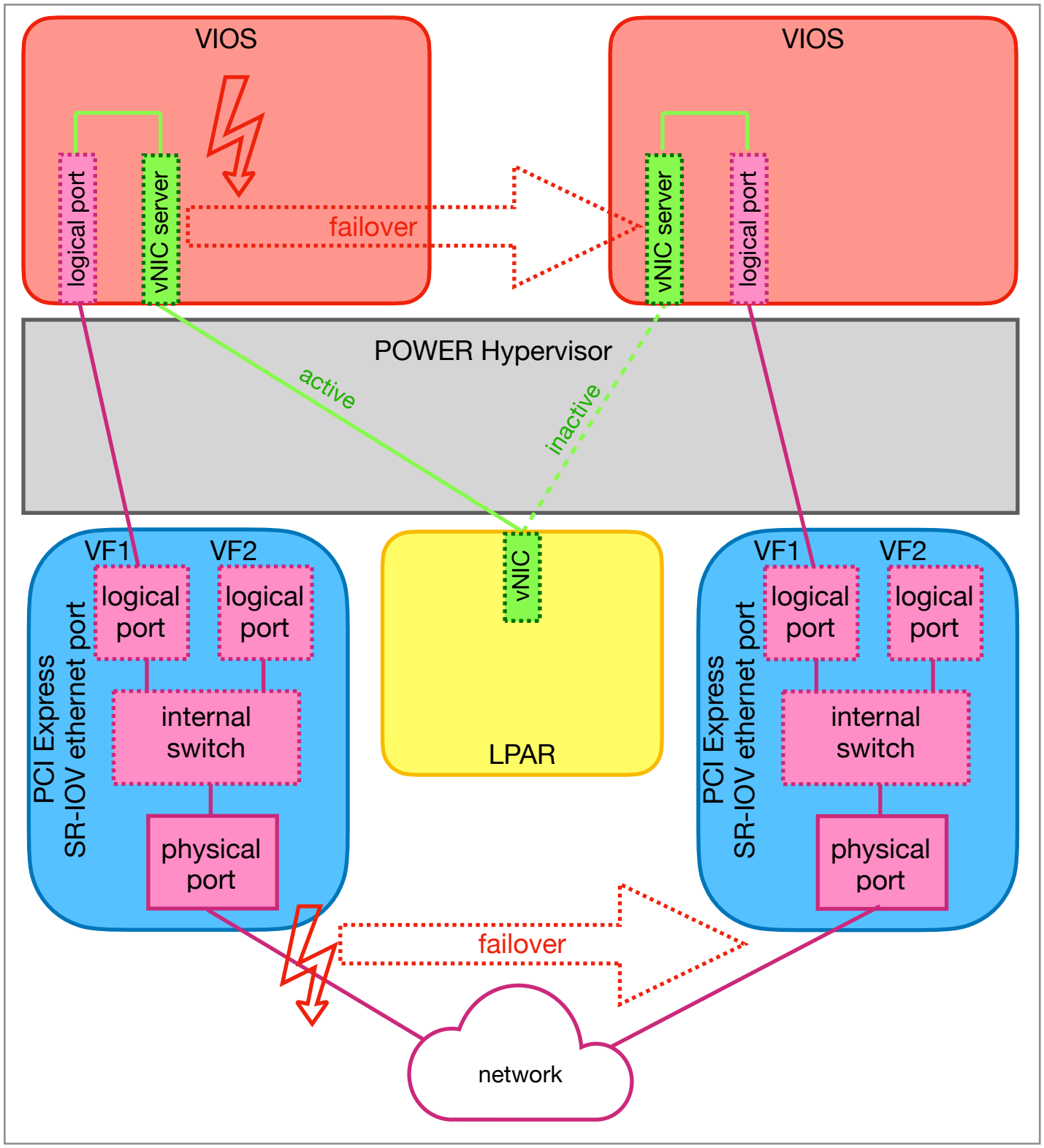


Bild 7.21: vNIC Adapter mit 2 vNIC-Backing-Devices und vNIC Failover.

Jedes vNIC-Backing-Device besitzt eine konfigurierbare Failover-Priorität (Attribut *failover_priority*). Diese kann Werte zwischen *1* und *100* annehmen, wobei ein niedrigerer Wert eine höhere Priorität bedeutet. Die Default Failover-Priorität ist *50*. Der Hypervisor wählt in einem Fehlerfall das vNIC-Backing-Device mit der höchsten Failover-Priorität (niedrigster Wert) als neues aktives vNIC-Backing-Device aus.

Ein vNIC-Backing-Device kann einem vNIC Adapter mit dem Kommando „*lpar addvnicbkdev*“ (*add vNIC backing device*) hinzugefügt werden:

```
$ lpar addvnicbkdev aix22 6 ms03-vio2 C6-T1
$
```

Der vNIC Adapter in Slot *6* von *aix22* besitzt nun 2 vNIC-Backing-Devices. Damit ist der vNIC Adapter gegen den Ausfall eines vNIC-Backing-Devices abgesichert, sollte das aktive vNIC-Backing-Device ausfallen, wird ein vNIC Failover auf das zweite vNIC-Backing-Device ausgeführt.

```
$ lpar lsvnic -a aix22
                              FAILOVER                                          PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PORT  PORT      CAPACITY  CAPACITY
aix22      6     Yes       50        1     Operational  ms03-vio1  1        0     27004005  2.0       100.0
aix22      6     Yes       50        0     Operational  ms03-vio2  2        0     27008004  2.0       100.0
$
```

Die Ausgabe zeigt, das das vNIC-Backing-Device für den logischen SR-IOV Port *27004005* aktuell das aktive Gerät ist (Wert 1 in der Spalte *ACTV*). Beide vNIC-Backing-Devices haben den Status *Operational* und beide haben den gleichen Wert für die Failover-Priorität.

Hinweis: Zwei vNIC-Backing-Devices des gleichen vNIC Adapters können keine logischen Ports verwenden, die zum gleichen physikalischen Port gehören! Würde der physikalische Link eines solchen physikalischen Ports heruntergehen (Link Down), würden beide vNIC-Backing-Devices gleichzeitig ausfallen.

7.7.4. Ändern eines vNIC-Backing-Devices

Das dynamische Ändern der Attribute eines vNIC-Backing-Devices ist nur eingeschränkt möglich. Die änderbaren Attribute sind die folgenden:

- *capacity*: die garantierte Kapazität des zugehörigen logischen SR-IOV Ports.
- *max_capacity*: die maximale Kapazität des zugehörigen logischen SR-IOV Ports.
- *failover_priority*: die Failover-Priorität (Werte zwischen *1* und *100*).

Von diesen Attributen kann nur das Attribut *failover_priority* dynamisch geändert werden. Die beiden anderen Attributen können nur in einem Profil der LPAR geändert werden.

Das Kommando zum Ändern der Attribute lautet „*lpar chvnicbkdev*“ (*change vNIC backing device*). Das vNIC-Backing-Device kann auf eine von 4 Arten angegeben werden:

- die logische Port ID des zugehörigen logischen Ports, z.B. *27008004*.
- einen eindeutigen Suffix des Physical Location Codes des zugehörigen logischen Ports, z.B. *P1-C6-T1-S4*.
- die Adapter ID und physikalische Port ID des zugehörigen physikalischen Ports, z.B. „*2 0*“.
- einen eindeutigen Suffix des Physical Location Codes des zugehörigen physikalischen Ports, z.B. *P1-C6-T1*.

Bevor wir eine Änderung durchführen, listen wir erst noch einmal die vNIC-Backing-Devices der LPAR aix22 auf:

```
$ lpar lsvnic -a aix22
                              FAILOVER                                          PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PORT  PORT      CAPACITY  CAPACITY
aix22      6     Yes       50        1     Operational  ms03-vio1  1        0     27004005  2.0       100.0
aix22      6     Yes       50        0     Operational  ms03-vio2  2        0     27008004  2.0       100.0
$
```

Als Beispiel ändern wir das zweite vNIC-Backing-Device (27008004). Die Failover-Priorität setzen wir auf den Wert 60 und die Kapazität (nicht dynamisch änderbar) erhöhen wir auf 4%:

```
$ lpar chvnicbkdev aix22 6 27008004 failover_priority=60 capacity=4
INFO: capacity skipped from DLPAR operation
$
```

Das zu ändernde vNIC-Backing-Device wird über die Slot-Nummer des vNIC Adapters und der logischen Port ID identifiziert. Das Kommando liefert den Hinweis zurück das die Kapazität nicht aktualisiert wurde (diese kann nur offline im Profil geändert werden). Die Auflistung der vNIC-Backing-Devices zeigt das die Failover-Priorität geändert wurde, die Kapazität ist aber gleich geblieben:

```
$ lpar lsvnic -a aix22
                            FAILOVER                                          PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PORT  PORT      CAPACITY  CAPACITY
aix22      6     Yes       50        1     Operational  ms03-vio1  1        0     27004005  2.0       100.0
aix22      6     Yes       60        0     Operational  ms03-vio2  2        0     27008004  2.0       100.0
$
```

Im aktuellen Profil der LPAR wurde die Kapazität aber geändert, so daß beim nächsten Aktivieren der LPAR mit dem geänderten Profil die Kapazität auf 4% gesetzt werden wird:

```
$ lpar -p standard lsvnic -a aix22
                            FAILOVER                        PHYS              MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  VIOS_NAME  ADAPTER  PORT  CAPACITY  CAPACITY
aix22      6     Yes       50        ms03-vio1  1        0     2.0       100.0
aix22      6     Yes       60        ms03-vio2  2        0     4.0       100.0
$
```

Hinweis: Indirekt lässt sich auch die Kapazität ändern, wenn man das betreffende vNIC-Backing-Device entfernt und dann mit geänderter Kapazität wieder hinzufügt.

7.7.5. vNIC Failover

In diesem Unterkapitel soll die Funktionalität des vNIC Failover genauer angeschaut werden. Besitzt ein vNIC Adapter genau zwei vNIC-Backing-Devices, dann wird bei Ausfallen des gerade aktiven vNIC-Backing-Devices zwangsläufig das zweite vNIC-Backing-Device aktiviert, eine Auswahl kann dann nicht stattfinden. Anders sieht die Situation aus, wenn es mehr als zwei vNIC-Backing-Devices gibt. Fällt das aktive vNIC-Backing-Device aus, gibt es dann mindestens zwei weitere vNIC-Backing-Devices. In diesem Fall kommt die Failover-Priorität (Attribut failover_priority) der vNIC-Backing-Devices zum tragen. Der Hypervisor wählt dasjenige vNIC-Backing-Device aus, welches die höchste Priorität (niedrigster Wert) besitzt.

Um den vNIC Failover im allgemeinen Fall genauer anzuschauen, fügen wir ein drittes vNIC-Backing-Device zu dem vNIC Adapter in Slot *6* der LPAR *aix22* hinzu. Verwendet wird für maximale Redundanz ein dritter Virtual-I/O-Server *ms03-vio3*:

```
$ lpar addvnicbkdev aix22 6 ms03-vio3 C4-T3 failover_priority=55
$
```

Damit hat der vNIC Adapter von *aix22* die folgenden vNIC-Backing-Devices:

```
$ lpar lsvnic -a aix22
                                FAILOVER                                                PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PHYS  PORT  LOGICAL   CURRENT     MAX
aix22      6     Yes       50        1     Operational  ms03-vio1  1        0     27004005  2.0         100.0
aix22      6     Yes       60        0     Operational  ms03-vio2  2        0     27008004  2.0         100.0
aix22      6     Yes       55        0     Operational  ms03-vio3  3        2     2700c00a  2.0         100.0
$
```

In Bild 7.22 ist das Zusammenspiel zwischen den vNIC-Backing-Devices (vNIC-Server) und dem POWER Hypervisor gezeigt. Jeder vNIC-Server überwacht den Status seines zugehörigen logischen SR-IOV Port. Der Status wird in Form einer Heartbeat Mitteilung in kurzen Abständen an den Hypervisor gemeldet.

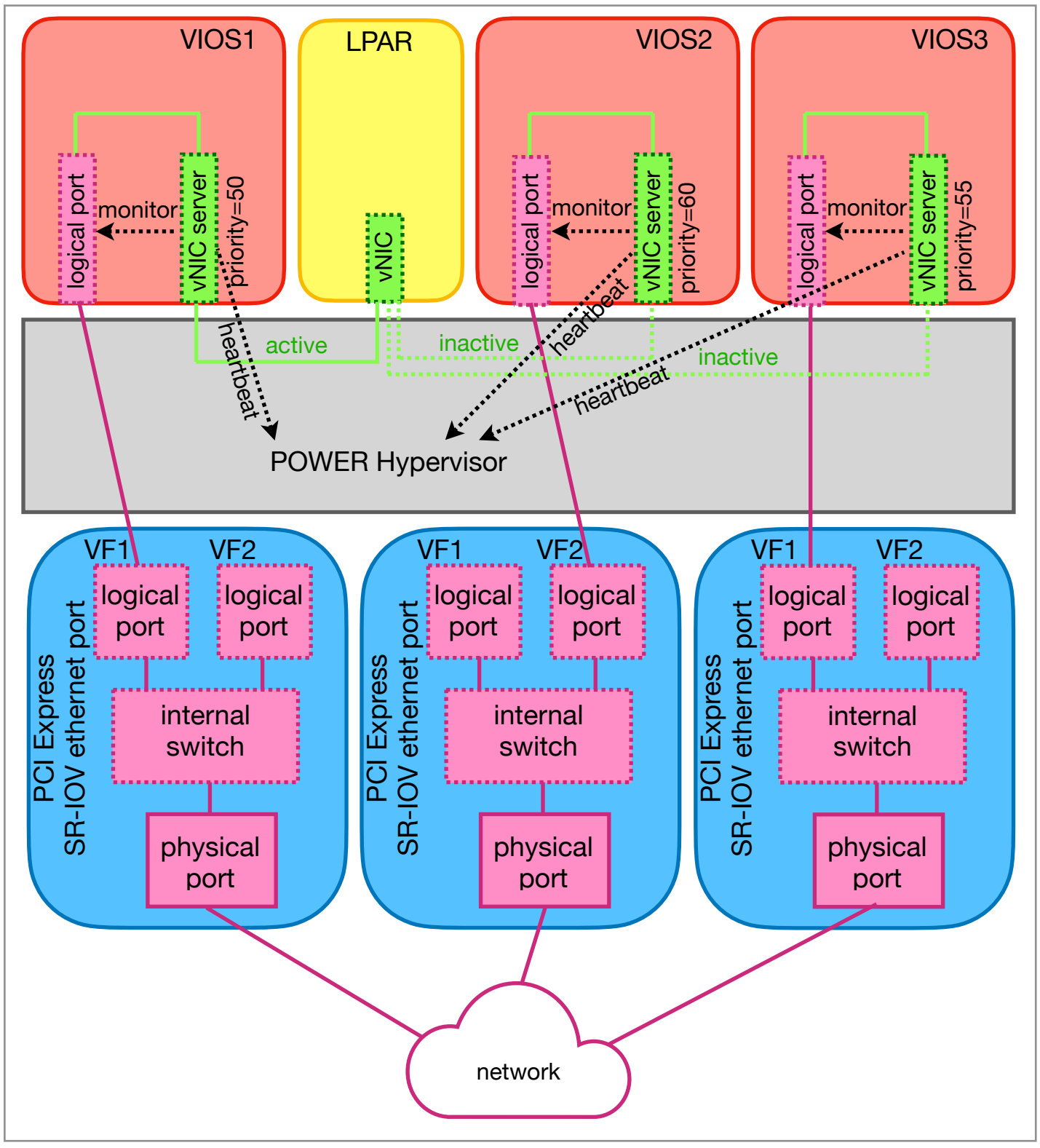


Bild 7.22: Zusammenspiel von vNIC-Backing-Devices und Hypervisor für vNIC Failover.

Fällt in dieser Konfiguration der Link des linken physikalischen Ports aus, bemerkt der zugehörige vNIC-Server auf dem ersten Virtual-I/O-Server dies bei seiner Überwachung des logischen SR-IOV Ports und meldet dies in einer Heartbeat-Mitteilung an den Hypervisor. Der Hypervisor entscheidet dann anhand der Failover-Prioritäten der verbleibenden funktionierenden vNIC-Backing-Devices, welches vNIC-Backing-Device mit zugehörigem logischen

Port aktiviert werden soll. Im gezeigten Fall haben die verbleibenden vNIC-Backing-Devices die Failover-Prioritäten *60* (*ms03-vio2*) bzw. *55* (*ms03-vio3*). Das vNIC-Backing-Device mit der höchsten Failover-Priorität (niedrigster Wert), in diesem Fall das vNIC-Backing-Device auf *ms03-vio3* mit der Priorität *55*, wird dann das neue aktive vNIC-Backing-Device. Der Hypervisor deaktiviert das bisher verwendete vNIC-Backing-Device, aktiviert das neue vNIC-Backing-Device und meldet das neu zu verwendende vNIC-Backing-Device an den vNIC-Client Adapter der LPAR.

Der Zustand der vNIC-Backing-Devices ist dann der Folgende:

```
$ lpar lsvnic -a aix22
                                FAILOVER                                                PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PORT  PORT      CAPACITY  CAPACITY
aix22      6     Yes       50        0     Link Down    ms03-vio1  1        0     27004005  2.0       100.0
aix22      6     Yes       60        0     Operational  ms03-vio2  2        0     27008004  2.0       100.0
aix22      6     Yes       55        1     Operational  ms03-vio3  3        2     2700c00a  2.0       100.0
$
```

Das dritte vNIC-Backing-Device mit Failover-Priorität *55* ist jetzt aktiv (Spalte *ACTV*), das ursprünglich aktive vNIC-Backing-Device hat jetzt den Status „*Link Down*“. Natürlich überwacht der vNIC-Server den ausgefallenen logischen Port auch weiterhin. Damit ist sichergestellt das der vNIC-Server erkennt wenn der logische Port wieder verfügbar ist und dies entsprechend auch an den Hypervisor meldet.

Wenn der Link des physikalischen Ports wieder hoch kommt, wechselt der Status des zugehörigen vNIC-Backing-Devices wieder auf *Operational*. Es stellt sich dann die Frage ob auf das jetzt wieder verfügbare vNIC-Backing-Device zurück gewechselt werden soll, oder nicht. Dies kann über das Attribut *auto_priority_failover* konfiguriert werden. Das Attribut besitzt 2 mögliche Werte:

- 0: automatisches Failover ist deaktiviert.

- 1: automatisches Failover ist aktiviert.

Ist das Attribut *auto_priority_failover* auf *1* gesetzt, wird immer auf das vNIC-Backing-Device mit der höchsten Priorität gewechselt. Da in unserem Falle das ursprüngliche vNIC-Backing-Device mit einem Wert von *50* die höchste Failover-Priorität hat, wird dieses sofort wieder aktiviert wenn es verfügbar wird:

```
$ lpar lsvnic -a aix22
                                FAILOVER                                                PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PORT  PORT      CAPACITY  CAPACITY
aix22      6     Yes       50        1     Operational  ms03-vio1  1        0     27004005  2.0       100.0
aix22      6     Yes       60        0     Operational  ms03-vio2  2        0     27008004  2.0       100.0
aix22      6     Yes       55        0     Operational  ms03-vio3  3        2     2700c00a  2.0       100.0
$
```

Das automatische Failover gilt aber nicht nur in dieser „Failback“ Situation, sondern generell. D.h. wann immer ein vNIC-Backing-Device eine höhere Failover-Priorität hat, wird sofort ein Failover auf dieses vNIC-Backing-Device durchgeführt. Dies kann in den folgenden Situationen auftreten:

- Ein weitere vNIC-Backing-Device wird einem vNIC Adapter hinzugefügt mit einer höheren Failover-Priorität als das aktive vNIC-Backing-Device.

- Die Failover-Priorität eines inaktiven vNIC-Backing-Devices wird geändert, wodurch dieses eine höhere Priorität als das aktuell aktive vNIC-Backing-Device bekommt.

- Die Failover-Priorität des aktiven vNIC-Backing-Devices wird geändert, wodurch dieses nicht mehr die höchste Priorität besitzt.
- Das aktive vNIC-Backing-Device (mit der höchsten Failover-Priorität) fällt aus.
- Ein ausgefallenes vNIC-Backing-Device mit einer höheren Failover-Priorität als das aktive vNIC-Backing-Device wird wieder verfügbar (Link ist wieder oben).

Automatic Priority Failover bedeutet also, das immer das vNIC-Backing-Device mit der höchsten Priorität aktiv ist.

7.7.6. Manuelles Aktivieren eines vNIC Backing-Devices

Mit dem Kommando „*lpar actvnicbkdev*“ (*activate vNIC backing device*) kann der Administrator jederzeit ein beliebiges (verfügbar, im Status *Operational*) vNIC-Backing-Device aktivieren. Dabei ist zu beachten, das das automatische Priority Failover dabei immer deaktiviert wird.

```
$ lpar actvnicbkdev aix22 6 P1-C6-T1
$
```

Neben der Slot-Nummer des vNIC Adapters muß der entsprechende vNIC Adapter spezifiziert werden. Hier kann wieder entweder der zugehörige logische SR-IOV Port oder der physikalische SR-IOV Port angegeben, entweder mit ID oder mit einem eindeutigen Suffix des Physical Location Codes.

Das automatische Priority Failover wurde, wie oben beschrieben, automatisch deaktiviert:

```
$ lpar lsvnic aix22
LPAR_NAME  SLOT  MODE  FAILOVER  PVID  PRIORITY  MAC_ADDR      ALLOWED_VLAN_IDS  BKDEVS
aix22      6     ded   No        0     0         81253aa07306  all               3
$
```

7.7.7. Wegnehmen eines vNIC Backing-Devices

Nicht mehr benötigte vNIC-Backing-Devices können jederzeit wieder entfernt werden. Das Kommando hierfür lautet „*lpar rmvnicbkdev*“ (*remove vNIC backing device*). Bevor wir eines der 3 vNIC-Backing-Devices wegnehmen, hier noch einmal die Übersicht über die aktuellen vNIC-Backing-Devices der LPAR *aix22*:

```
$ lpar lsvnic -a aix22
                                FAILOVER                                          PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PORT  PORT      CAPACITY  CAPACITY
aix22      6     Yes       50        1     Operational  ms03-vio1  1        0     27004005  2.0       100.0
aix22      6     Yes       60        0     Operational  ms03-vio2  2        0     27008004  2.0       100.0
aix22      6     Yes       55        0     Operational  ms03-vio3  3        2     2700c00a  2.0       100.0
$
```

Wir entfernen das dritte vNIC-Backing-Device mit der logischen Port ID *2700c00a*. Dieses ist zur Zeit nicht das aktive vNIC-Backing-Device.

```
$ lpar rmvnicbkdev aix22 6 2700c00a
$
```

Eine erneute Übersicht über die vNIC-Backing-Devices der LPAR *aix22* zeigt das das genannte vNIC-Backing-Device erfolgreich entfernt wurde:

```
$ lpar lsvnic -a aix22
                            FAILOVER                                          PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PORT  PORT      CAPACITY  CAPACITY
aix22      6     Yes       50        1     Operational  ms03-vio1  1        0     27004005  2.0       100.0
aix22      6     Yes       60        0     Operational  ms03-vio2  2        0     27008004  2.0       100.0
$
```

Der Versuch das gerade aktive vNIC-Backing-Device mit der logischen Port ID *27004005* wegzunehmen, schlägt allerdings fehl:

```
$ lpar rmvnicbkdev aix22 6 27004005
hmc01: chhwres -m ms03 -r virtualio --rsubtype vnic -o s -p aix22   -s 6 -a
'backing_devices-=sriov///1/0///'
ERROR: remote HMC command returned an error (1)
StdErr: HSCLAB44 The active backing device for a virtual NIC cannot be deleted.
$
```

Das aktive vNIC-Backing-Device kann nicht entfernt werden! Möchte man das vNIC-Backing-Device dennoch entfernen, muß man zuerst ein anderes vNIC-Backing-Device aktivieren. In unserem Falle ist nur noch ein weiteres vNIC-Backing-Device verfügbar:

```
$ lpar actvnicbkdev aix22 6 27008004
$
```

Anschließend ist ein entfernen des ersten vNIC-Backing-Devices problemlos möglich:

```
$ lpar rmvnicbkdev aix22 6 27004005
$
```

Damit bleibt nur noch ein vNIC-Backing-Device:

```
$ lpar lsvnic -a aix22
                            FAILOVER                                          PHYS  LOGICAL   CURRENT     MAX
LPAR_NAME  SLOT  FAILOVER  PRIORITY  ACTV  STATUS       VIOS_NAME  ADAPTER  PORT  PORT      CAPACITY  CAPACITY
aix22      6     Yes       60        1     Operational  ms03-vio2  2        0     27008004  2.0       100.0
$
```

Da ein vNIC Adapter immer mindestens ein vNIC-Backing-Device besitzen muß, lässt sich das letzte vNIC-Backing-Device nicht wegnehmen:

```
$ lpar rmvnicbkdev aix22 6 27008004
hmc01: chhwres -m ms03 -r virtualio --rsubtype vnic -o s -p aix22   -s 6 -a
'backing_devices-=sriov///2/0///'
ERROR: remote HMC command returned an error (1)
StdErr: HSCLAB43 This operation is not allowed because it will cause all of the backing
devices for the virtual NIC to be deleted.
$
```

Hier bleibt nur die Möglichkeit den kompletten vNIC Adapter zu löschen.

7.7.8. Wegnehmen eines vNIC Adapters

Damit ein vNIC Adapter wieder entfernt werden kann, darf er nicht mehr in Verwendung sein. Zusätzlich muß unter AIX das zugehörige *ent*-Gerät vorher aus dem Betriebssystem entfernt werden.

Auf der LPAR *aix22* gibt es unter AIX aktuell die folgenden beiden *ent*-Geräte:

```
aix22 # lsdev -l ent\*
ent0 Available  Virtual I/O Ethernet Adapter (l-lan)
ent1 Available  Virtual NIC Client Adapter (vnic)
aix22 #
```

Der Ethernet Adapter *ent1* ist vom Typ vNIC-Client Adapter. Dieser soll im folgenden entfernt werden. Zunächst lösen wir das Interface *en1* vom Ethernet Adapter *ent1* ab:

```
aix22 # ifconfig en1 detach
aix22 #
```

Damit wird sofort jeglicher IP-Traffic über *en1*/*ent1* gestoppt! Als nächstes kann der vNIC Ethernet Adapter aus dem Betriebssystem entfernt werden:

```
aix22 # rmdev -dl ent1
ent1 deleted
aix22 #
```

Um den vNIC Adapter nun mittels PowerVM zu entfernen, benötigen wir die virtuelle Slot-Nummer des vNIC Adapters:

```
$ lpar lsvnic aix22
LPAR_NAME  SLOT  MODE  FAILOVER  PVID  PRIORITY  MAC_ADDR      ALLOWED_VLAN_IDS  BKDEVS
aix22      6     ded   No        0     0         81253aa07306  all               1
$
```

Die LPAR *aix22* besitzt nur einen vNIC Adapter in Slot *6*.

Das Entfernen des Adapters kann mit dem Kommando „*lpar rmvnic*“ (*remove vNIC adapter*) ausgeführt werden, dabei wird neben der LPAR die virtuelle Slot-Nummer des vNIC Adapters angegeben:

```
$ lpar rmvnic aix22 6
$
```

Die zugehörigen vNIC-Backing-Devices (vNIC-Server) und logischen SR-IOV Ports auf den Virtual-I/O-Servern werden dabei durch den Hypervisor automatisch entfernt.

8. Virtual-I/O-Server

Bei den bisherigen Themen ging es hauptsächlich um LPARs bzw. die virtuellen Gastsysteme oder VMs. Viele der vorgestellten Virtualisierungsmöglichkeiten erfordern allerdings einen (oder auch mehrere) Virtual-I/O-Server. Virtuelles I/O muß letztlich dann doch physikalische I/O Adapter verwenden, welche dann typischerweise Virtual-I/O-Servern zugeordnet sind. In diesem Kapitel geht es daher um Themen rund um Virtual-I/O-Server.

8.1. Planung und Anlegen eines Virtual-I/O-Servers

Egal ob Virtual Ethernet, Virtual SCSI, Virtual FC oder vNIC, alle diese Virtualisierungsmöglickeiten erfordern mindestens einen Virtual-I/O-Server.

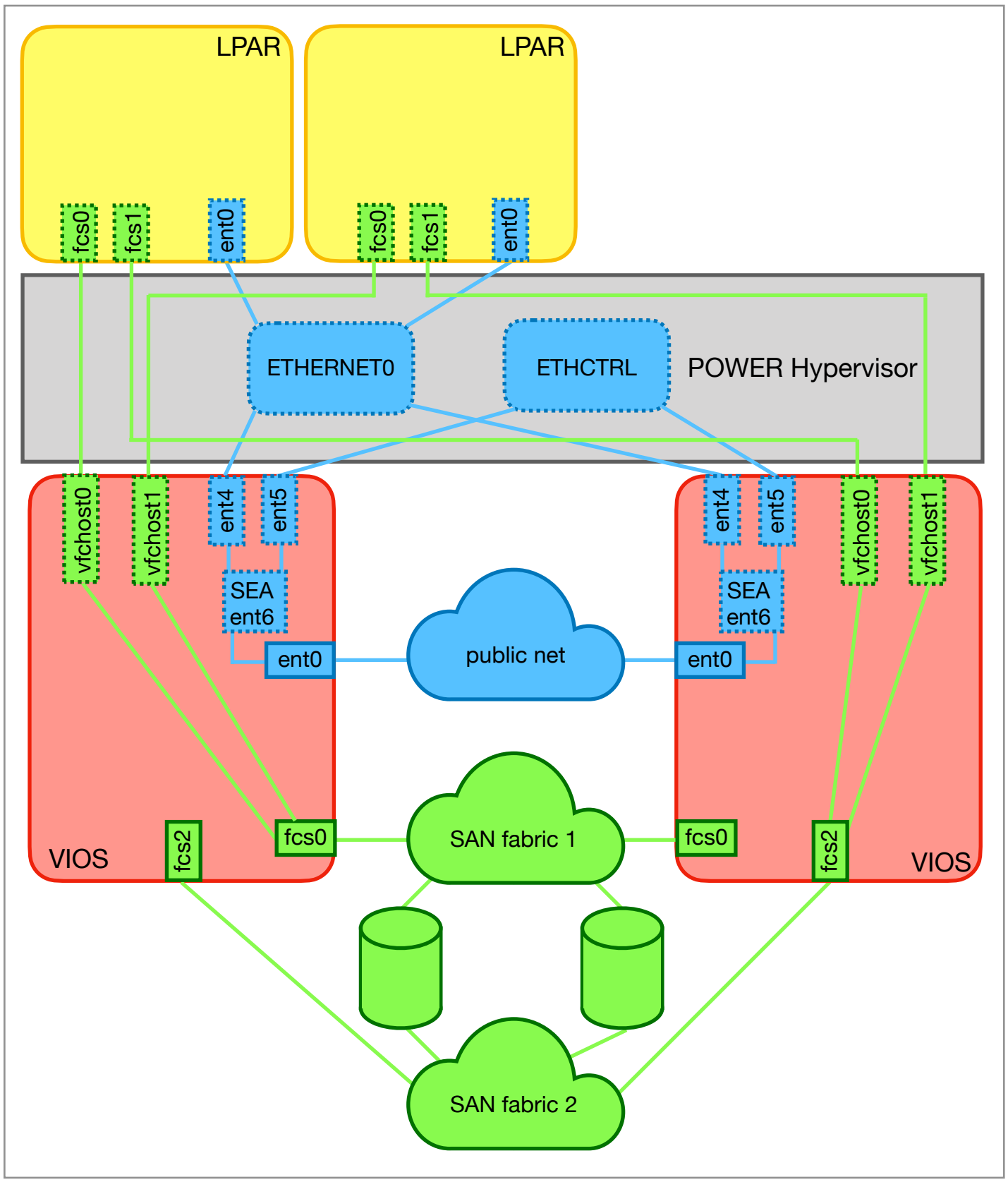


Bild 8.1: Redundante Virtual-I/O-Server Konfiguration

In der Praxis wird fast immer eine Konfiguration mit zwei redundanten Virtual-I/O-Servern verwendet. Wie in Bild 8.1 skizziert, besitzen dabei beide Virtual-I/O-Server Anbindungen an die gleichen externen Netzwerke und SAN Fabrics. Auf High-End Managed Systems findet man häufig weitere Virtual-I/O-Server, um die Last gleichmäßiger zu verteilen, oder um z.B. Produktions-LPARs und Test-LPARs über verschiedene Virtual-I/O-Server voneinander zu trennen.

Damit ein Virtual-I/O-Server seine Funktion wahrnehmen kann, benötigt er Zugriff auf die physikalische Hardware des Managed Systems. Vorhandene I/O Adapter müssen daher den Virtual-I/O-Servern zugeordnet werden, mit der Ausnahme von SR-IOV Netzwerk-Adaptern, wenn SR-IOV verwendet werden soll. Sinnvollerweise sollte zunächst einmal eine Bestandsaufnahme gemacht werden, welche Adapter-Karten überhaupt verfügbar sind und in welchen I/O-Slots die Karten eingebaut wurden. Eine Auflistung aller I/O-Slots eines Managed Systems erhält man mit dem Kommando „*ms lsslot*“ (*list I/O slots*):

```
$ ms lsslot ms12
DRC_NAME                  DRC_INDEX  IOPOOL  LPAR_NAME  DESCRIPTION
U78D3.001.VYR0ETG-P1-C9   21010010   none    -          PCIe3 4-Port 16Gb FC Adapter
U78D3.001.VYR0ETG-P1-C10  21010012   none    -          Empty slot
U78D3.001.VYR0ETG-P1-C11  21020013   none    -          PCIe3 2 PORT 25/10 Gb NIC&ROCE
SFP28 ADAPTER
U78D3.001.VYR0ETG-P1-C12  21030014   none    -          PCIe3 4-Port 16Gb FC Adapter
U78D3.001.VYR0ETG-P1-C49  21040015   none    -          PCIe3 x8 SAS RAID Internal Adapter
6Gb
U78D3.001.VYR0ETG-P1-T4   21010017   none    -          Universal Serial Bus UHC Spec
U78D3.001.WZS0FUH-P1-C8   21010018   none    -          PCIe3 2 PORT 25/10 Gb NIC&ROCE
SFP28 ADAPTER
U78D3.001.VYR0ETG-P1-C5   2101001A   none    -          Empty slot
U78D3.001.VYR0ETG-P1-C6   2102001B   none    -          PCIe3 4-Port 16Gb FC Adapter
U78D3.001.VYR0ETG-P1-C7   2103001C   none    -          Empty slot
U78D3.001.VYR0ETG-P1-C50  2104001D   none    -          PCIe3 x8 SAS RAID Internal Adapter
6Gb
U78D3.001.VYR0ETG-P1-C4   21010020   none    -          PCIe3 2 PORT 25/10 Gb NIC&ROCE
SFP28 ADAPTER
U78D3.001.VYR0ETG-P1-C2   21010021   none    -          PCIe3 2 PORT 25/10 Gb NIC&ROCE
SFP28 ADAPTER
U78D3.001.VYR0ETG-P1-C3   21010022   none    -          PCIe3 4-Port 16Gb FC Adapter
$
```

Bei dem Beispiel System *ms12* handelt es sich um eine S922 (Modell G), mit 4 mal „*PCIe3 4-Port 16Gb FC Adapter*“ und 4 mal „*PCIe3 2 PORT 25/10 Gb NIC&ROCE SFP28 ADAPTER*“. Das System wurde mit einer Split-Backplane (2 mal „*PCIe3 x8 SAS RAID Internal Adapter*“ geordert und kann daher ohne externe Platten mit 2 Virtual-I/O-Servern konfiguriert werden. Jeder der beiden Virtual-I/O-Server bekommt einen der beiden SAS-Adapter mit internen Platten zugewiesen. Da die 4 SR-IOV Netzwerk-Adapter mittels SR-IOV verwendet werden sollen, müssen nur die 4 FC Adapter auf die beiden Virtual-I/O-Server aufgeteilt werden.

Die beiden Virtual-I/O-Server sollen *ms12-vio1* und *ms12-vio2* heißen. Wir legen zunächst den ersten Virtual-I/O-Server *ms12-vio1* als Shared-Prozessor LPAR mit *4* Prozessoren und *8 GB* Speicher an (*uncap_weight=255*):

```
$ lpar -m ms12 create ms12-vio1 lpar_env=vioserver desired_procs=4 max_procs=8
desired_mem=8G max_mem=16G max_virtual_slots=300 uncap_weight=255
.
    > ms12-vio1
$
```

Als maximale Werte wurde jeweils das Doppelte des normalen Wertes angegeben. Wichtig ist die Angabe des Attributs *lpar_env* mit dem Wert *vioserver*, da ansonsten eine normale LPAR vom Typ *aixlinux* angelegt wird! Der zweite Virtual-I/O-Server sollte mit den gleichen Werten angelegt werden. Dies kann erreicht werden, indem man das letzte Kommando mit geändertem LPAR-Namen wiederholt, alternativ kann man aber auch beim Erzeugen eine Source-LPAR angeben, die als Konfigurationsvorlage (Blueprint) dient, in diesem Falle der gerade erzeugte Virtual-I/O-Server:

```
$ lpar -m ms12 create -s ms12-vio1 ms12-vio2
    > ms12-vio2
$
```

Der erzeugte zweite Virtual-I/O-Server besitzt die gleiche Konfiguration wie der erste Virtual-I/O-Server! Beim Erzeugen einer neuen LPAR wird standardmäßig der Profil-Name *standard* verwendet:

```
$ lpar -p standard lsmem -m ms12
                    MEMORY            MEMORY            HUGE_PAGES
LPAR_NAME  MODE  AME  MIN   DESIRED  MAX    MIN  DESIRED  MAX
ms12-vio1  ded   0.0  1024  8192     16384  -    -        -
ms12-vio2  ded   0.0  1024  8192     16384  -    -        -
$ lpar -p standard -m ms12 lsproc                    PROC         PROCS
PROC_UNITS                    UNCAP      PROC
LPAR_NAME  MODE    MIN  DESIRED  MAX  MIN  DESIRED  MAX  SHARING_MODE  WEIGHT  POOL
ms12-vio1  shared  1    4        8    0.4  0.4      0.8  uncap         255     DefaultPool
ms12-vio2  shared  1    4        8    0.4  0.4      0.8  uncap         255     DefaultPool
$
```

Als nächstes weisen wir den beiden Virtual-I/O-Servern jeweils einen der beiden SAS-Adapter zu. Dem ersten Virtual-I/O-Server (*ms12-vio1*) weisen wir den SAS-Adapter in Slot *P1-C49* zu, DRC-Index *21040015* und dem zweiten Virtual-I/O-Server (*ms12-vio2*) den SAS-Adapter in Slot *P1-C50*, DRC-Index *2104001D*. Die Adapter können mit dem Kommando „*ms addslot*“ (*add I/O slot*) hinzugefügt werden, wobei das Profil angegeben werden muß, da die Virtual-I/O-Server bisher noch deaktiviert sind:

```
$ lpar -p standard addslot ms12-vio1 21040015
$ lpar -p standard addslot ms12-vio2 2104001D
$
```

Jeder der beiden POWER9 Prozessoren eines S922 Servers besitzt 4 PCIe4.0 Controller. Der SAS-Adapter im Slot *P1-C49* teilt sich dabei einen PCIe4.0 Controller mit den Adapter-Karten in den Slots *P1-C10*, *P1-C11* und *P1-C12*. Der SAS-Adapter im Slot *P1-C50* teilt sich einen PCIe4.0 Controller mit den Adapter-Karten in den Slots *P1-C5*, *P1-C6* und *P1-C7*. Wir weisen daher den FC-Adapter in Slot *P1-C12* (DRC-Index *21030014*) dem Virtual-I/O-Server *ms12-vio1* zu, der schon den Adapter in Slot *P1-C49* besitzt und den FC-Adapter in Slot *P1-C6* (DRC-Index *2102001B*) dem Virtual-I/O-Server *ms12-vio2*, der schon den Adapter in Slot *P1-C50* besitzt:

```
$ lpar -p standard addslot ms12-vio1 21030014
$ lpar -p standard addslot ms12-vio2 2102001B
$
```

Damit bleiben noch die letzten beiden FC-Adapter in den Slot *P1-C3* (DRC-Index *21010022*)und *P1-C9* (DRC-Index *21010010*). Beide Slots besitzen einen eigenen PCIe4.0 Controller. Wir weisen den FC-Adapter in Slot *P1-C9* dem Virtual-I/O-Server *ms12-vio1* und den FC-Adapter in Slot *P1-C3* dem Virtual-I/O-Server *ms12-vio2* zu:

```
$ lpar -p standard addslot ms12-vio1 21010010
$ lpar -p standard addslot ms12-vio2 21010022
$
```

Damit sind außer den SR-IOV Adaptern alle anderen Adapter-Karten den beiden Virtual-I/O-Servern zugewiesen. Da die Virtual-I/O-Server bisher nicht aktiviert wurden, steht die Zuordnung bisher nur in den Profilen der beiden Virtual-I/O-Server. Die Zuordnung der Slots im Profil kann mit dem Kommando „*lpar lsslot*“ (*list I/O slots*) unter Angabe des Profil-Namens aufgelistet werden, hier für *ms12-vio1* und das Profil *standard*:

```
$ lpar -p standard lsslot ms12-vio1
DRC_NAME                  DRC_INDEX  REQ  IOPOOL  DESCRIPTION
U78D3.001.VYR0ETG-P1-C9   21010010   No   none    PCIe3 4-Port 16Gb FC Adapter
U78D3.001.VYR0ETG-P1-C12  21030014   No   none    PCIe3 4-Port 16Gb FC Adapter
U78D3.001.VYR0ETG-P1-C49  21040015   No   none    PCIe3 x8 SAS RAID Internal Adapter 6Gb
$
```

Und entsprechend für *ms12-vio2*:

```
$ lpar -p standard lsslot ms12-vio2
DRC_NAME                  DRC_INDEX  REQ  IOPOOL  DESCRIPTION
U78D3.001.VYR0ETG-P1-C6   2102001B   No   none    PCIe3 4-Port 16Gb FC Adapter
U78D3.001.VYR0ETG-P1-C50  2104001D   No   none    PCIe3 x8 SAS RAID Internal Adapter 6Gb
U78D3.001.VYR0ETG-P1-C3   21010022   No   none    PCIe3 4-Port 16Gb FC Adapter
$
```

Damit können die beiden Virtual-I/O-Server als nächstes installiert werden.

8.2. Installation Virtual-I/O-Server

Die Software zur Installation der Virtual-I/O-Server kann über die Entitled Software Support (ESS) Webseite von IBM (https://www.ibm.com/eserver/ess) bezogen werden. Zum aktuellen Zeitpunkt (31.05.2021) konnten die folgenden ISO-Images für die Version *3.1.2.20* (DVD) bzw. *3.1.2.21* (Flash) heruntergeladen werden:

- *Virtual_IO_Server_Base_Install_3.1.2.20_DVD_1_of_2_042021_LCD8250106.iso*
- *Virtual_IO_Server_Base_Install_3.1.2.20_DVD_2_of_2_042021_LCD8250206.iso.ZIP*
- *Virtual_IO_Server_Base_Install_3.1.2.21_Flash_042021_LCD8250307.iso*

Die DVD-Images können unter AIX mit dem Kommando *burn_cd* auf eine DVD gebrannt werden, das Flash-Image kann mit dem Kommando *dd* auf einen USB-Stick kopiert werden.

Im folgenden werden die verschiedenen Möglichkeiten der Installation gezeigt.

8.2.1. Installation über CD oder DVD

Verfügt das Managed System über ein DVD-Laufwerk, dann können die Virtual-I/O-Server über DVD installiert werden. In vielen Fällen besitzen aber gerade neuere Managed Systems kein eigenes DVD-Laufwerk mehr. Es bleibt dann aber immer noch die Möglichkeit über USB zu installieren.

Wir zeigen daher die Installation über DVD für ein älteres System *ms04* (*Power740*). Das DVD-Laufwerk hat hier den Physical Location Code *P2-D9* und ist dem SAS-Adapter im Slot *P1-T9* zugeordnet. Das DVD-Laufwerk kann also immer nur von einem Virtual-I/O-Server benutzt werden, diesem muß der SAS-Adapter im Slot *P1-T9* zugeordnet sein!

Für die Installation über DVD müssen die beiden DVD-Images zunächst auf DVD gebrannt werden. Ein DVD-Writer steht als Device *cd0* auf unserem AIX-System *aixnim* zur Verfügung. Das erste DVD-Image kann unmittelbar mit dem Kommando *burn_cd* auf eine leere DVD gebrannt werden:

```
aixnim # burn_cd -d /dev/cd0
Virtual_IO_Server_Base_Install_3.1.2.20_DVD_1_of_2_042021_LCD8250106.iso
Running readcd ...

…
aixnim #
```

Das zweite DVD-Image liegt im *ZIP*-Format vor und muß daher zunächst ausgepackt werden, bevor es dann auf eine zweite leere DVD gebrannt werden kann:

```
aixnim # unzip Virtual_IO_Server_Base_Install_3.1.2.20_DVD_2_of_2_042021_LCD8250206.iso.ZIP
Archive:  Virtual_IO_Server_Base_Install_3.1.2.20_DVD_2_of_2_042021_LCD8250206.iso.ZIP
ZIP/390 Build:10/01/13 by Data 21
  inflating: Virtual_IO_Server_Base_Install_3.1.2.20_DVD_2_of_2_042021_LCD8250206.iso
aixnim #
aixnim # burn_cd -d /dev/cd0
Virtual_IO_Server_Base_Install_3.1.2.20_DVD_2_of_2_042021_LCD8250206.iso
Running readcd ...

…
aixnim #
```

Die beiden gebrannten DVDs sollten auf jeden Fall beschriftet werden, so daß jederzeit erkennbar ist, was auf der DVD drauf ist. Z.B.: „*VIOS 3.1.2.20 1/2*“ und „*VIOS 3.1.2.20 2/2*“.

Der SAS-Adapter mit dem DVD-Laufwerk wurde dem Virtual-I/O-Server *ms04-vio1* zugeordnet. Wir legen daher die erste Installations-DVD („*VIOS 3.1.2.20 1/2*“) in das DVD-Laufwerk ein und aktivieren den Virtual-I/O-Server. Wir wählen den Boot-Mode *SMS* und geben das Profil *standard* an, gleichzeitig starten wir eine Konsolen-Sitzung (Option ‚*-c*‘):

```
$ lpar -p standard activate -b sms -c ms04-vio1

Open in progress

 Open Completed.
  Network     SCSI     Speaker

PowerPC Firmware
Version AL740_167
```

```
SMS 1.7 (c) Copyright IBM Corp. 2000,2008 All rights reserved.
-------------------------------------------------------------------------------
Main Menu
1.   Select Language
2.   Setup Remote IPL (Initial Program Load)
3.   Change SCSI Settings
4.   Select Console
5.   Select Boot Options









-------------------------------------------------------------------------------
Navigation Keys:

                                                X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:5
```

Als erstes muß der Menüpunkt 5 „*Select Boot Options*“ ausgewählt werden:

```
-------------------------------------------------------------------------------
Navigation Keys:

                                                X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:5

PowerPC Firmware
Version AL740_167
SMS 1.7 (c) Copyright IBM Corp. 2000,2008 All rights reserved.
-------------------------------------------------------------------------------
Multiboot
1.   Select Install/Boot Device
2.   Configure Boot Device Order
3.   Multiboot Startup <OFF>
4.   SAN Zoning Support









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:1
```

Hier muß dann der Menüpunkt 1 „*Select Install/Boot Device*“ ausgewählt werden:

```
PowerPC Firmware
```

```
Version AL740_167
SMS 1.7 (c) Copyright IBM Corp. 2000,2008 All rights reserved.
-------------------------------------------------------------------------------
Select Device Type
1.   Diskette
2.   Tape
3.   CD/DVD
4.   IDE
5.   Hard Drive
6.   Network
7.   List all Devices









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:3
```

Bei der Auswahl der Geräte-Typen muß der Punkt 3 „*CD/DVD*“ ausgewählt werden:

```
-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:3

PowerPC Firmware
Version AL740_167
SMS 1.7 (c) Copyright IBM Corp. 2000,2008 All rights reserved.
-------------------------------------------------------------------------------
Select Media Type
 1.   SCSI
 2.   SSA
 3.   SAN
 4.   SAS
 5.   SATA
 6.   USB
 7.   IDE
 8.   ISA
 9.   List All Devices









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:5
```

Der Typ des DVD-Laufwerks ist SATA, also muß der Punkt 5 ausgewählt werden:

```
-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:5

PowerPC Firmware
Version AL740_167
SMS 1.7 (c) Copyright IBM Corp. 2000,2008 All rights reserved.
-------------------------------------------------------------------------------
Select Media Adapter
1.             /pci@80000002000000a/pci1014,0339@0/sata
2.   List all devices









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:1
```

Der einzige auswählbare Adapter ist die Nummer 1:

```
check /pci@80000002000000a/pci1014,0339@0/sata/disk@60000

PowerPC Firmware
Version AL740_167
SMS 1.7 (c) Copyright IBM Corp. 2000,2008 All rights reserved.
-------------------------------------------------------------------------------
Select Device
Device  Current  Device
Number  Position  Name
1.        -      SATA CD-ROM
        ( loc=U78AA.001.WZSHV1R-P2-D9 )









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:1
```

Der Adapter besitzt nur ein DVD-Laufwerk, daher ist hier erneut der Punkt 1 auszuwählen:

```
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:1

PowerPC Firmware
Version AL740_167
SMS 1.7 (c) Copyright IBM Corp. 2000,2008 All rights reserved.
-------------------------------------------------------------------------------
Select Task

SATA CD-ROM
    ( loc=U78AA.001.WZSHV1R-P2-D9 )

1.   Information
2.   Normal Mode Boot
3.   Service Mode Boot









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:2
```

Für die Installation des Virtual-I/O-Servers ist ein normaler Boot erforderlich, als Menüpunkt 2 „*Normal Mode Boot*“:

```
-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:2

PowerPC Firmware
Version AL740_167
SMS 1.7 (c) Copyright IBM Corp. 2000,2008 All rights reserved.
-------------------------------------------------------------------------------
Are you sure you want to exit System Management Services?
1.   Yes
2.   No
```

```
-------------------------------------------------------------------------------
Navigation Keys:

                                             X = eXit System Management Services
-------------------------------------------------------------------------------
Type menu item number and press Enter or select Navigation key:1
```

Als letztes muß noch bestätigt werden das der SMS Mode verlassen werden soll, Menüpunkt 1 „*Yes*“. Das System beginnt von der eingelegten DVD zu booten:

```
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM                         IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM     STARTING SOFTWARE   IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM        PLEASE WAIT...   IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM                         IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM IBM
\
Elapsed time since release of system processors: 140918 mins 14 secs
```

Nach kurzer Zeit (einige Sekunden) erscheint dann auch die Boot-Meldung des Virtual-I/O-Servers und die Abfrage nach der verwendeten System Konsole:

```
-------------------------------------------------------------------------------
                       Welcome to the Virtual I/O Server.
                   boot image timestamp: 01:20:51 03/18/2021
                 The current time and date: 07:30:59 06/04/2021
        processor count: 1;  memory size: 4096MB;  kernel size: 51876481
boot device: /pci@800000020000000a/pci1014,0339@0/sata/disk@60000:\ppc\chrp\bootfile.exe
-------------------------------------------------------------------------------
```

```
******* Please define the System Console. *******

Type a 1 and press Enter to use this terminal as the
  system console.
Pour definir ce terminal comme console systeme, appuyez
  sur 1 puis sur Entree.
Taste 1 und anschliessend die Eingabetaste druecken, um
  diese Datenstation als Systemkonsole zu verwenden.
Premere il tasto 1 ed Invio per usare questo terminal
  come console.
Escriba 1 y pulse Intro para utilizar esta terminal como
  consola del sistema.
Escriviu 1 1 i premeu Intro per utilitzar aquest
  terminal com a consola del sistema.
Digite um 1 e pressione Enter para utilizar este terminal
  como console do sistema.

1
```

Hier muß lediglich die angezeigte Nummer eingetippt und mit der *RETURN*-Taste bestätigt werden.

Als nächstes erfolgt die Abfrage nach der Sprache während der Installation:

```
>>>  1 Type 1 and press Enter to have English during install.
     2 Entreu 2 i premeu Intro per veure la installaci en catal.
     3 Entrez 3 pour effectuer l'installation en franais.
     4 Für Installation in deutscher Sprache 4 eingeben
        und die Eingabetaste drücken.
     5 Immettere 5 e premere Invio per l'installazione in Italiano.
     6 Digite 6 e pressione Enter para usar Portugus na instalao.
     7 Escriba 7 y pulse Intro para la instalacion en espanol.



    88  Help ?


>>> Choice [1]:1
```

Hier ist die gewünschte Sprache während der Installation auszuwählen, wir bleiben hier bei Englisch und wählen daher die Nummer 1 aus.

```
                      Welcome to Base Operating System
                      Installation and Maintenance

Type the number of your choice and press Enter.  Choice is indicated by >>>.

>>> 1 Start Install Now with Default Settings

    2 Change/Show Installation Settings and Install
```

```
    3 Start Maintenance Mode for System Recovery

    4 Make Additional Disks Available

    5 Select Storage Adapters





    88  Help ?
    99  Previous Menu

>>> Choice [1]:
```

Wir bleiben bei den Defaults und starten die Installation durch Auswahl des Punktes 1:

```
                            System Backup Installation Summary

Disks:  hdisk0
Use Physical Location Maps:  No
Shrink File Systems:  No
Import User Volume Groups:  No
Recover Devices:  No
Selected Edition:  standard




>>> 1   Continue with Install
                        +-----------------------------------------------------
    88  Help ?          |    WARNING: Base Operating System Installation will
    99  Previous Menu   |    destroy or impair recovery of ALL data on the
                        |    destination disk hdisk0.
>>> Choice [1]:
```

An dieser Stelle ist die letzte Möglichkeit noch einmal zurück zu gehen und die Einstellungen zu überprüfen, was wir aber nicht tun. Wir bestätigen daher die vorgeschlagene Auswahl 1 „*Continue with Install*“:

```
                         Installing Base Operating System




       Please wait...
```

```
      Approximate     Elapsed time
   % tasks complete   (in minutes)

        8               0      1% of mksysb data restored.
```

Die Installation startet unmittelbar. Im unteren Bereich wird ein Fortschrittsangabe in Prozent angegeben mit zusätzlichen kurzen Hinweisen, was im Moment bei der Installation genau gemacht wird.

Die Installation bleibt nach einiger Zeit stehen, und fordert auf die zweite DVD einzulegen:

```
      Approximate     Elapsed time
   % tasks complete   (in minutes)

        70               3      84% of mksysb data restored.

…

      Please remove volume 1, insert volume 2, and press the ENTER key.
```

Nachdem die zweite DVD eingelegt und die ENTER-Taste gedrückt wird, geht die Installation sofort weiter:

```
                      Installing Base Operating System



      Please wait...









      Approximate     Elapsed time
   % tasks complete   (in minutes)

        71               10      86% of mksysb data restored.
```

Wenn die Installation abgeschlossen ist, wird der Virtual-I/O-Server sofort automatisch rebootet:

```
Licensed Materials - Property of IBM

5765CD200
   Copyright International Business Machines Corp. 1985, 2021.
   Copyright AT&T 1984, 1985, 1986, 1987, 1988, 1989.
   Copyright Regents of the University of California 1980, 1982, 1983, 1985, 1986, 1987,
1988, 1989.
   Copyright BULL 1993, 2021.
   Copyright Digi International Inc. 1988-1993.
   Copyright Interactive Systems Corporation 1985, 1991.
   Copyright ISQUARE, Inc. 1990.
   Copyright Innovative Security Systems, Inc. 2001-2006.
   Copyright Mentat Inc. 1990, 1991.
   Copyright Open Software Foundation, Inc. 1989, 1994.
   Copyright Sun Microsystems, Inc. 1984, 1985, 1986, 1987, 1988, 1991.

All rights reserved.
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.
Rebooting . . .
```

Der Reboot nach der Installation dauert etwas länger als ein normaler Reboot, da beim ersten Booten einige Initialisierungen vorgenommen werden. Nach einiger Zeit erscheint die Aufforderung nach dem Login:

```
…
System reconfiguration in progress.  Please wait.
/ filesystem not converted.
        Small inode extents are already enabled.

Rebuild of Scriptinfo file is complete

Setting vpm_fold_policy to 4 in nextboot file
Setting vpm_fold_policy to 4
0513-059 The lldpd Subsystem has been started. Subsystem PID is 4129060.
0513-059 The ecpvdpd Subsystem has been started. Subsystem PID is 4718936.
IBM Virtual I/O Server

                      login:
```

Hier muß eine Anmeldung als Benutzer *padmin* erfolgen. Es gibt kein Standard-Passwort! Beim ersten Anmelden muß ein neues Passwort vergeben werden:

```
                      login: padmin
[compat]: 3004-610 You are required to change your password.
        Please choose a new one.

padmin's New password: XXXXXXXX
Enter the new password again: XXXXXXXX

Indicate by selecting the appropriate response below whether you
accept or decline the software maintenance terms and conditions.
Accept (a) |  Decline (d) |  View Terms (v) > a
$
```

Außerdem müssen die Software-Lizenzbestimmungen von IBM durch Eingabe von ‚*a*' (*Accept*) angenommen werden. Danach ist man als Benutzer *padmin* eingeloggt und kann die weitere Konfiguration vornehmen. Bevor man ein IOS Kommando starten kann, muss noch die I/O-Server Lizenz akzeptiert werden, ansonsten kann kein Kommando ausgeführt werden und es erscheint immer ein Warnhinweis:

```
$ ioslevel
The I/O Server license must be accepted before running this command.
Only the chlang command may be run before accepting the license.
To view the license, run the license command with the -view option.
If the -lang flag is not specified the license will be displayed in
the current locale (set by the chlang command).  If the license is not
available in the language selected, English will be the default.
To accept the license, run the license command with the -accept option.

Locale currently set to: "en_US".

Usage: license {[-view] [-accept]} [-lang Name]
       license [-ls]

View and accept the I/O Server license agreement.

        -accept Accept the license agreement.
        -lang   Specifies the language-territory (locale name) the license
                is displayed in.
        -ls     Lists available locales.
        -view   Displays the license agreement.


$
```

Wir akzeptieren die Lizenz durch Ausführen des Kommandos „*license -accept*":

```
$ license -accept

  Current system settings are different from the best practice recommendations for a VIOS.

  To view the differences between system and the recommended settings, run the following:
  $rules -o diff -s -d

  To deploy the VIOS recommended default settings, run the following:
  $rules -o deploy -d
  $shutdown -restart

$
```

Danach lassen sich auch alle Kommandos starten:

```
$ ioslevel
3.1.2.20
$
```

8.3. Geräte Verwaltung

Virtualisierung über Virtual-I/O-Server wird in vielen Fällen durch das Zusammenspiel von physikalischen Geräten und virtuellen Geräten ermöglicht. Daher ist die Administration von Geräten insbesondere auf einem Virtual-I/O-Server von zentraler Bedeutung. Die verfügbaren Kommandos für die Administration von Geräten können mittels „*vios help dev*“ aufgelistet werden:

```
$ vios help dev
USAGE: vios [<option> ...] <keyword> [<option> ...] [<argument> ...]

Recognized keywords for topic 'dev' are:
   [-h <hmc>] [-m <ms>] cfgdev <vios> [<device>|<physloc>]
   [-h <hmc>] [-m <ms>] chdev [-f] [-P] <vios> <device>|<physloc> <attribute> ...
   [-h <hmc>] [-m <ms>] chkdev [{-o <format>|-f|-j|-y}] [-F <fields>] [-s <selections>]
<vios>
   [-h <hmc>] [-m <ms>] lsattr [{-o <format>|-f|-j|-y}] [-F <fields>] [-s <selections>] [-
R] <vios> <device>|<physloc> [<attribute>]
   [-h <hmc>] [-m <ms>] lsdev [-c] [-d] [{-o <format>|-f|-j|-y}] [-F <fields>] [-s
<selections>] [-t <device_type>] <vios> [<device>|<physloc>]
   [-h <hmc>] [-m <ms>] lspv [{-o <format>|-f|-j|-y}] [-F <fields>] [-s <selections>]
<vios>
   [-h <hmc>] [-m <ms>] rmdev [-d] [-f] [-R] [-v] <vios> <device>|<physloc>
$
```

Bei allen Kommandos können Geräte entweder in Form des Gerätenamens, wie *ent0* oder *fcs2*, oder als Physical Location Code (bzw. eindeutiger Suffix eines Physical Location Codes) angegeben werden, wie *P1-C3-T1* oder *P1-C49*.

8.3.1. Anzeigen von verfügbaren Geräten

Für das Anzeigen von Geräten auf einem Virtual-I/O-Server kann das Kommando „*vios lsdev*“ (*list devices*) verwendet werden. Wird nur der Virtual-I/O-Server angegeben, werden sämtliche Geräte des Virtual-I/O-Servers ausgegeben:

```
$ vios lsdev ms13-vio1
NAME        STATUS     PHYSLOC                          PARENT      DESCRIPTION
L2cache0    Available  -                                sysplanar0  L2 Cache
cache0      Defined    -                                -           SSD Cache virtual device
cengine0    Defined    -                                -           SSD Cache engine
cluster0    Available  -                                -           Cluster Node
…
$
```

Über die Option ‚*-t*‘ kann die Ausgabe auf einen angegebenen Typ von Gerät eingeschränkt werden, z.B. nur Geräte vom Typ *disk*:

```
$ vios lsdev -t disk ms13-vio1
NAME    STATUS     PHYSLOC                                          PARENT  DESCRIPTION
hdisk0  Available  U78D3.001.VYR0AL4-P1-C49-L207762C200-L0         sas0    SAS 4K RAID 0
Disk Array
```

```
hdisk1  Available  U78D3.001.VYR0AL4-P1-C49-L80775F5000-L0          sas0    SAS RAID 0 Disk
Array
hdisk2  Available  U78D3.001.VYR0AL4-P1-C49-L6077605A00-L0          sas0    SAS RAID 0 Disk
Array
hdisk3  Available  U78D3.001.VYR0AL4-P1-C7-T1-W500507680130A1C4-L0  fscsi4  MPIO IBM 2145
FC Disk
hdisk4  Available  U78D3.001.VYR0AL4-P1-C7-T1-W50050768013098BC-L0  fscsi4  MPIO IBM 2145
FC Disk
$
```

Mögliche Typen für die Option ‚*-t*' sind dabei:

```
     adapter - list adapters only
     disk - list disks only
     ent4ip - list adapters over which an interface can be configured
     ent4sea - list adapters available as target-device for SEA
     lv - list logical volumes and volume groups
     optical - list optical devices
     sea - list all SEAs over which an interface can be configured
     tape - list tape devices only
     tape4vtd - list tape devices available to create virtual target devices
     tty - list tty devices only
     usb_disk - list USB disks only
     vent4sea - list virtual ethernet adpaters available for SEA creation
```

Natürlich können Geräte auch direkt angegeben werden:

```
$ vios lsdev ms13-vio1 ent15
NAME   STATUS     PHYSLOC                          PARENT  DESCRIPTION
ent15  Available  U78D3.001.VYR0AL4-P1-C11-T2-S3  pci6    PCIe3 10GbE SFP+ SR 4-port
Converged Network Adapter VF (df1028e214100f04)
$
```

Oder auch per eindeutigem Suffix des Physical Location Codes:

```
$ vios lsdev ms13-vio1 P1-C11-T2-S3
NAME   STATUS     PHYSLOC                          PARENT  DESCRIPTION
ent15  Available  U78D3.001.VYR0AL4-P1-C11-T2-S3  pci6    PCIe3 10GbE SFP+ SR 4-port
Converged Network Adapter VF (df1028e214100f04)
$
```

Ist der angegebene Suffix nicht eindeutig, wird eine Fehlermeldung zusammen mit einer Auflistung der möglichen Geräte angezeigt:

```
$ vios lsdev ms13-vio1 P1-C7-T1
ERROR: 'P1-C7-T1' matches more than one device:
   fcs4
   fscsi4
USAGE:
   vios [-h <hmc>] [-m <ms>] lsdev [-c] [-d] [{-o <format>|-f|-j|-y}] [-F <fields>] [-s
<selections>] [-t <device_type>] <vios> [<device>|<physloc>]
$
```

In diesem Falle haben sowohl *fcs4* als auch *fscsi4* den gleichen Physical Location Code *P1-C7-T1*. Anstelle von *P1-C7-T1* kann *fcs4* oder *fscsi4* angegeben werden, oder alternativ kann der Location Code mit einem ‚/‚ und dem Geräte-Typ *fcs* oder *fscsi* angegeben werden:

```
$ vios lsdev ms13-vio1 P1-C7-T1/fcs
NAME  STATUS     PHYSLOC                      PARENT  DESCRIPTION
fcs4  Available  U78D3.001.VYR0AL4-P1-C7-T1  pci1    PCIe3 4-Port 16Gb FC Adapter
(df1000e314101406)
$
```

Sollen für ein Gerät die Kind-Geräte angezeigt werden, kann hierfür die Option ‚*-c*‘ (*child devices*) verwendet werden:

```
$ vios lsdev -c ms13-vio1 sas0
NAME      STATUS     PHYSLOC                                  PARENT  DESCRIPTION
hdisk0    Available  U78D3.001.VYR0AL4-P1-C49-L207762C200-L0  sas0    SAS 4K RAID 0 Disk
Array
hdisk1    Available  U78D3.001.VYR0AL4-P1-C49-L80775F5000-L0  sas0    SAS RAID 0 Disk Array
hdisk2    Available  U78D3.001.VYR0AL4-P1-C49-L6077605A00-L0  sas0    SAS RAID 0 Disk Array
pdisk0    Available  U78D3.001.VYR0AL4-P2-D3                  sas0    Physical SAS 4K Disk
Drive
pdisk1    Available  U78D3.001.VYR0AL4-P2-D2                  sas0    Physical SAS Disk
Drive
pdisk2    Available  U78D3.001.VYR0AL4-P2-D1                  sas0    Physical SAS Disk
Drive
ses0      Available  U78D3.001.VYR0AL4-P2-Y1                  sas0    SAS Enclosure
Services Device
sfwcomm8  Available  -                                        sas0    SAS Storage Framework
Comm
$
```

Über die Option ‚*-d*‘ können sogenannte Vital Product Data (*VPD*) angezeigt werden:

```
$ vios lsdev -d ms13-vio1 fcs0
  fcs0             U78D3.001.VYR0AL4-P1-C2-T1  PCIe3 4-Port 16Gb FC Adapter
(df1000e314101406)

        Part Number.................01FT695
        Serial Number...............Y050GX06U00D
        EC Level....................P14609
        Customer Card ID Number.....578E
        Manufacturer................001D
        FRU Number..................01FT699
        Device Specific.(ZM)........3
        Network Address.............100000108AA821D0
        Device Specific.(Z0)........0000000C
        Device Specific.(Z1)........00000001
        Device Specific.(Z2)........00000000
        Device Specific.(Z3)........08090000
        Device Specific.(Z4)........01000001
        Device Specific.(Z5)........2E323537
        Device Specific.(Z6)........2E323537
        Device Specific.(Z7)........C0022C40
        Device Specific.(Z8)........200000108AA821D0
        Device Specific.(Z9)........12.4.257.27
        Device Specific.(ZA)........12.4.257.27
        Device Specific.(ZB)........00000000
        Device Specific.(ZC)........00040000
        Device Specific.(ZD)........000000FF
        Hardware Location Code......U78D3.001.VYR0AL4-P1-C2-T1
```

```
  PLATFORM SPECIFIC

  Name:  fibre-channel
    Model:  01FT695
    Node:  fibre-channel@0
    Device Type:  fcp
    Physical Location: U78D3.001.VYR0AL4-P1-C2-T1

$
```

Die meisten Geräte besitzen eine Reihe von Attributen, die zum Teil vom Administrator geändert werden können. Die Attribute eines Gerätes lassen sich mit dem Kommando „*vios lsattr*“ (*list device attributes*) anzeigen:

```
$ vios lsattr ms13-vio1 fcs0
ATTRIBUTE      VALUE          DESCRIPTION
USER_SETTABLE
DIF_enabled    no             DIF (T10 protection) enabled                        True
bus_mem_addr   0x80218000     Bus memory address                                  False
init_link      auto           INIT Link flags                                     False
intr_msi_1     42             Bus interrupt level                                 False
intr_priority  3              Interrupt priority                                  False
io_dma         256            IO_DMA                                              True
label          Fabric1_Prod   User defined Label                                  True+
lg_term_dma    0x800000       Long term DMA                                       True
max_xfer_size  0x400000       Maximum Transfer Size                               True
msi_type       msix           MSI Interrupt type                                  False
num_cmd_elems  3072           Maximum number of COMMANDS to queue to the adapter  True
num_io_queues  8              Desired number of IO queues                         True
srad_id        0              SRAD index                                          False
$
```

Auch hier kann wieder ein eindeutiger Suffix des Physical Location Codes verwendet werden:

```
$ vios lsattr ms13-vio1 P1-C7-T1/fcs
ATTRIBUTE      VALUE              DESCRIPTION
USER_SETTABLE
DIF_enabled    no                 DIF (T10 protection) enabled                        True
bus_mem_addr   0xc0218000         Bus memory address                                  False
init_link      auto               INIT Link flags                                     False
intr_msi_1     47                 Bus interrupt level                                 False
intr_priority  3                  Interrupt priority                                  False
io_dma         256                IO_DMA                                              True
label          Fabric1_Prod_Nbg   User defined Label                                  True+
lg_term_dma    0x800000           Long term DMA                                       True
max_xfer_size  0x400000           Maximum Transfer Size                               True
msi_type       msix               MSI Interrupt type                                  False
num_cmd_elems  3072               Maximum number of COMMANDS to queue to the adapter  True
num_io_queues  8                  Desired number of IO queues                         True
srad_id        1                  SRAD index                                          False
$
```

Soll nur ein bestimmtes Attribut angezeigt werden, kann das gewünschte Attribut einfach als weiteres Argument der Kommandozeile hinzugefügt werden:

```
$ vios lsattr ms13-vio1 fcs4 num_cmd_elems
value
3072
$
```

8.3.2. Hinzufügen von Geräten

Auf einem Virtual-I/O-Server werden neue Geräte normalerweise automatisch hinzugefügt. Wird beispielsweise ein virtueller I/O-Adapter angelegt, wird das zugehörige Gerät automatisch in das Betriebssystem des Virtual-I/O-Servers konfiguriert. Ein manueller Lauf des Config-Managers, wie bei AIX LPARs ist normalerweise nicht erforderlich.

Sollte trotzdem ein Mal ein neues Gerät manuell hinzugefügt werden müssen, kann dies mit dem Kommando „*vios cfgdev*“ (*configure device*) getan werden. Es wird letztlich der Config-Manager *cfgmgr* des unterliegenden AIX gestartet.

Das Kommando kann nur mit dem Virtual-I/O-Server als Argument gestartet werden. Dabei werden alle existierenden Adapter auf neue Geräte überprüft und diese gegebenenfalls ins Betriebssystem konfiguriert:

```
$ vios cfgdev ms13-vio1
$
```

Es kann aber auch ein (existierendes) Gerät als zusätzliches Argument angegeben werden, die Überprüfung auf neue Geräte findet dann nur in dem Teilbaum ab dem angegebenen Gerät statt. Dies kann auf größeren Systemen mit extrem vielen Geräten unter Umständen deutlich schneller sein:

```
$ vios cfgdev ms13-vio1 fcs4
$
```

Auch hier kann das (existierende) Gerät wiederum als eindeutiger Suffix eines Physical Location Codes angegeben werden.

8.3.3. Ändern von Geräten

Geräte-Attribute können mit dem Kommando „*vios chdev*“ (*change device*) geändert werden. Neben dem Virtual-I/O-Server muß der Gerätename (oder Physical Location Code), sowie das zu ändernde Attribut und dem neuen Wert angegeben werden:

```
$ vios chdev ms13-vio1 fcs1 label=Fabric2
$
```

Dabei wird die Änderung per Default dynamisch und in der *ODM* ausgeführt. Soll die Änderung nur in der *ODM*, aber nicht dynamisch, durchgeführt werden, kann die Option ‚*-P*‘ (*persistent only*) verwendet werden:

```
$ vios chdev -P ms13-vio1 fcs1 label=Fabric2
$
```

Gibt es für ein Attribut ein Bereich von gültigen Werten, oder eine Liste von gültigen Werten, kann diese mit der Option ‚*-R*' (*range*) von „*vios lsattr*" abgefragt werden:

```
$ vios lsattr -R ms13-vio1 fcs1 num_cmd_elems
200...4096 (+1)
$
```

Es können beliebig viele Attribut in einem Kommando abgeändert werden:

```
$ vios chdev ms13-vio1 fcs1 label=Fabric2 num_cmd_elems=4096
$
```

8.3.4. Wegnehmen von Geräten

Auch das Wegnehmen von nicht mehr benötigten Geräten geschieht auf einem Virtual-I/O-Server typischerweise automatisch. Sollen Geräte manuell entfernt werden, kann hierzu das Kommando „*vios rmdev*" (*remove devices*) verwendet werden. Dabei muß neben dem Virtual-I/O-Server das zu entfernende Gerät angegeben werden:

```
$ vios rmdev ms13-vio1 hdisk3
$
```

Die Definition für das Gerät bleibt dabei in der ODM des Virtual-I/O-Servers:

```
$ vios lsdev ms13-vio1 hdisk3
NAME    STATUS   PHYSLOC                                        PARENT  DESCRIPTION
hdisk3  Defined  U78D3.001.VYR0AL4-P1-C7-T1-W500507680130A1C4-L0  fscsi4  MPIO IBM 2145 FC
Disk
$
```

Soll das Gerät auch aus der ODM entfernt werden, muß die Option ‚*-d*' verwendet werden!

Ist ein Gerät in Benutzung, kann es natürlich nicht entfernt werden. Beim Versuch das Gerät zu entfernen, wird eine Fehlermeldung ausgegeben, typischerweise mit der Information in welcher Weise das Gerät aktuell in Benutzung ist:

```
$ vios rmdev ms13-vio1 hdisk4
hmc01: viosvrcmd -m ms13 -p ms13-vio1 -c \"rmdev -dev hdisk4 -ucfg\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: Device "hdisk4" is in use as an AMS backing device.
StdErr:
StdErr: rc=71
$
```

In diesem Fall ist das Gerät (*hdisk4*) als Paging-Device für einen Shared Memory Pool konfiguriert.

Gelegentlich soll ein Adapter samt Kind-Geräten entfernt werden. Im Prinzip müssen zunächst alle Kind-Geräte entfernt werden, bevor dann das Gerät für den Adapter selbst entfernt werden kann. Mit der Option ‚*-R*' (*recursive*) können Geräte rekursiv entfernt werden. Um z.B. den FC-Adapter *fcs4* samt aller Kind-Geräte (*fscsi4*, *sfwcomm4* und *hdisks*) zu entfernen, kann die Option ‚*-R*' auf das Gerät *fcs4* angewendet werden:

```
$ vios rmdev -R ms13-vio1 fcs4
$
```

Sollen gleichzeitig die Definitionen auch aus der *ODM* gelöscht werden, muß zusätzlich die Option ‚*-d*' verwendet werden.

8.4. Error Report

Der Error Report ist bei AIX und damit auch bei Virtual-I/O-Servern die zentrale Sammelstelle für Fehlermeldungen. Alle Fehler, die das Betriebssystem erkennt, werden über den *errdemon* geloggt und können vom Administrator jederzeit abgerufen werden. Zur Anzeige von Meldungen des Error Reports dient das Kommando „vios errlog". Wird nur der Virtual-I/O-Server angegeben, wird eine Zusammenfassung aller Meldungen auf dem betreffenden Virtual-I/O-Server ausgegeben:

```
$ vios errlog ms13-vio1
IDENTIFIER  TIMESTAMP   TYPE  CLASS  RESOURCE_NAME  DESCRIPTION
4B436A3D    0531052421  T     H      fscsi0         LINK ERROR
DC73C03A    0531051421  T     S      fscsi0         SOFTWARE PROGRAM ERROR
8C577CB6    0521111321  I     S      vnicserver0    VNIC Transport Event
60D73419    0521101121  I     S      vnicserver0    VNIC Client Login
E48A73A4    0521092321  I     H      ent45          BECOME PRIMARY
E15C5EAD    0520131421  T     H      ent37          Physical link up
F596EFAC    0520083421  T     H      ent37          Physical link down
E87EF1BE    0517150021  P     O      dumpcheck      The largest dump device is too small.
8D424E06    0509095621  I     H      ent31          ADAPTER FAILURE
AA8AB241    0507075921  T     O      OPERATOR       OPERATOR NOTIFICATION
F31FFAC3    0321142821  I     H      hdisk3         PATH HAS RECOVERED
DE3B8540    0321142321  P     H      hdisk3         PATH HAS FAILED
D5676F6F    0321142221  T     H      fscsi4         ATTACHED SCSI TARGET DEVICE ERROR
B8C78C08    0319122621  I     H      ent7           SEA HA PARTNER LOST
A6D1BD62    0319122221  I     H      unspecified    Firmware Event
C62E1EB7    0314103021  P     H      hdisk4         DISK OPERATION ERROR
37F3CC40    0219145721  P     U      RMCdaemon      RSCT has detected that system time has
m
06DE59EC    1117194020  I     U      vhost0         Logging an informational error for VIO
s
…
$
```

(**Hinweis:** In der Ausgabe wurden viele Meldungen weggelassen, um in wenigen Zeilen möglichst viele Arten von Fehlermeldungen exemplarisch zu zeigen.)

Pro Zeile wird eine Fehlermeldung angezeigt, dabei werden die wichtigsten Informationen, wie Zeitstempel (*TIMESTAMP*), Typ und Klasse ausgegeben. Es wird die betroffene Ressource, sowie ein kurze Beschreibung angezeigt. Die neueste Fehlermeldung ist immer die oberste Fehlermeldung. Die Anzahl der auszugebenden Meldungen lässt sich über die Option ‚*-n*' (*number*) einschränken:

```
$ vios errlog -n 5 ms13-vio1
IDENTIFIER  TIMESTAMP   TYPE  CLASS  RESOURCE_NAME  DESCRIPTION
4B436A3D    0531052421  T     H      fscsi0         LINK ERROR
4B436A3D    0531052421  T     H      fscsi0         LINK ERROR
4B436A3D    0531052421  T     H      fscsi0         LINK ERROR
4B436A3D    0531052421  T     H      fscsi0         LINK ERROR
4B436A3D    0531052421  T     H      fscsi0         LINK ERROR
$
```

Details lassen sich mit der Option ‚*-a*' (*all informations*) anzeigen, dabei schränkt man am Besten die Anzahl der gezeigten Meldungen gleichzeitig mit der Option ‚*-n*' ein. Ansonsten kann die Ausgabe sehr lang werden:

```
$ vios errlog -n 1 -a ms13-vio1
---------------------------------------------------------------------------
LABEL:           FCP_ERR4
IDENTIFIER:      4B436A3D

Date/Time:       Mon May 31 05:24:00 2021
Sequence Number: 7342
Machine Id:      00CA09503A00
Node Id:         ms13-vio1
Class:           H
Type:            TEMP
WPAR:            Global
Resource Name:   fscsi0
Resource Class:  driver
Resource Type:   emfscsi
Location:        U78D3.001.VYR0AL4-P1-C2-T1


Description
LINK ERROR

            Recommended Actions
            PERFORM PROBLEM DETERMINATION PROCEDURES

Detail Data
SENSE DATA
0000 0020 0000 0327 0000 0000 0203 0101 1000 0010 9BB9 32E1 0000 0000 008C 8240
…
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
$
```

Jede Meldung hat eine eindeutige Sequence-Nummer, die bei den detaillierten Informationen ausgegeben wird, im Beispiel die *7342*. Diese Sequence-Nummer kann als zusätzliches Argument bei „*vios errlog*" angegeben werden, um genau eine Meldung herauszufiltern. Das ist leider nicht sehr praktisch, da in der Zusammenfassung die Sequence-Nummer nicht mit ausgegeben wird. (Das liegt am unterliegenden Kommando auf dem Virtual-I/O-Server.)

Möchte man gezielt Meldungen auswählen, empfiehlt sich der Selection-Mechanismus des LPAR-Tools mit der Option ‚*-s*'. Hier kann nach beliebigen Kriterien ausgewählt werden, was letztlich angezeigt werden soll. Z.B. lassen sich damit relativ leicht alle Meldungen zu einer bestimmten Ressource auflisten, hier Meldungen zur *hdisk3*:

```
$ vios errlog -s resource_name=hdisk3 ms13-vio1
IDENTIFIER  TIMESTAMP   TYPE  CLASS  RESOURCE_NAME  DESCRIPTION
F31FFAC3    0321142821  I     H      hdisk3         PATH HAS RECOVERED
DE3B8540    0321142321  P     H      hdisk3         PATH HAS FAILED
F31FFAC3    0321142221  I     H      hdisk3         PATH HAS RECOVERED
DE3B8540    0321141621  P     H      hdisk3         PATH HAS FAILED
F31FFAC3    0321123121  I     H      hdisk3         PATH HAS RECOVERED
DE3B8540    0321122521  P     H      hdisk3         PATH HAS FAILED
F31FFAC3    0321122421  I     H      hdisk3         PATH HAS RECOVERED
DE3B8540    0321121521  P     H      hdisk3         PATH HAS FAILED
F31FFAC3    0321110221  I     H      hdisk3         PATH HAS RECOVERED
DE3B8540    0321104921  P     H      hdisk3         PATH HAS FAILED
F31FFAC3    0321092721  I     H      hdisk3         PATH HAS RECOVERED
DE3B8540    0321091321  P     H      hdisk3         PATH HAS FAILED
$
```

Als zirkulärer Log kann der Error Report nicht beliebig groß werden. Nach einiger Zeit werden alte Einträge automatisch überschrieben.

Jeder Administrator einer PowerVM-Umgebung sollte die Error Reports aller Virtual-I/O-Server immer im Blick haben.

8.5. Shared Ethernet Adapter

Trotz SR-IOV und vNIC ist Shared Ethernet immer noch die am häufigsten verwendete Virtualisierungslösung, wenn es um die Virtualisierung von Ethernet geht. Dabei implementiert der POWER Hypervisor virtuelle interne IEEE802.1q kompatible Netzwerk-Switches, die im Zusammenspiel mit sogenannten Shared Ethernet Adaptern oder kurz SEA die Anbindung an externe Netzwerke übernehmen. Die Shared Ethernet Adapter werden durch den Virtual-I/O-Server in Software als Layer-2 Bridge implementiert.

Wie in Bild 8.2 gezeigt, kann ein Shared Ethernet Adapter mehrere sogenannte Trunking Adapter besitzen. Der gezeigte SEA hat die 3 Trunking Adapter *ent8*, *ent9* und *ent10*, welche alle 3 an den virtuellen Switch mit dem Namen *ETHMGMT* angebunden sind. Alle Trunking Adapter unterstützen im gezeigten Fall VLAN-Tagging. Neben den Port-VLAN-IDs (*PVID*s) besitzen die 3 Trunking Adapter noch weitere VLANs über VLAN-Tagging. Neben der Anbindung an den virtuellen Switch über die Trunking Adapter besitzt der SEA noch eine Anbindung an ein externes Netzwerk über den physikalischen Netzwerk-Adapter (*ent0*). Netzwerk-Pakete von Client-LPARs an externe Systeme werden über einen der Trunking-Adapter zum SEA und dann über den zugehörigen physikalischen Netzwerk-Adapter ins externe Netzwerk weitergeleitet. Netzwerk-Pakete von externen Systemen an Client-LPARs werden vom SEA über den Trunking Adapter mit dem richtigen VLAN an den virtuellen Switch weitergeleitet, welcher die Pakete dann an die Client-LPAR weiterleitet.

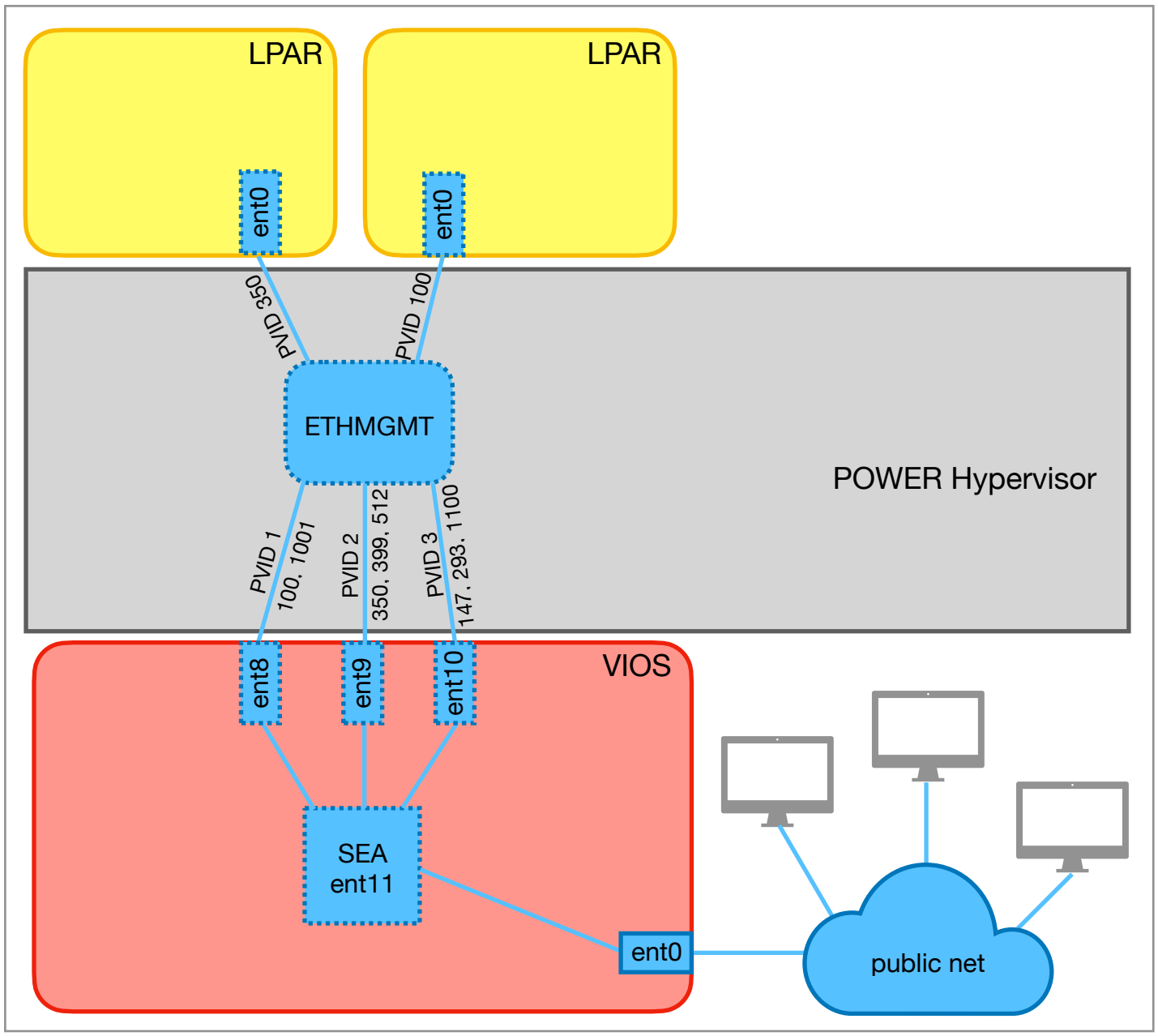


Bild 8.2: SEA mit mehreren Trunking Adaptern und VLANs

Im einfachsten Fall kann ein SEA aus nur einem Trunking Adapter bestehen. Ein SEA kann bis zu 16 Trunking Adaptern besitzen, wobei jeder der Trunking-Adapter neben der Port-VLAN-ID bis zu 20 weitere VLANs besitzen kann.

Welche SEAs es auf einem Virtual-I/O-Server schon gibt, lässt sich mit Hilfe des Kommandos „*vios lssea*“ (*list SEAs*) herausfinden:

```
$ vios lssea ms05-vio1
                                          TIMES   TIMES    TIMES    BRIDGE
NAME   HA_MODE  PRIORITY  STATE        PRIMARY  BACKUP  FLIPFLOP  MODE
ent33  Sharing  1         PRIMARY_SH   1        1       0         Partial
ent34  Sharing  1         PRIMARY_SH   1        1       0         Partial
$
```

Zu jedem SEA werden einige grundlegende Informationen angezeigt, wie z.B. der HA-Mode (siehe später), die Priorität des SEA, sowie Informationen wie häufig der SEA schon Primary bzw. Backup war.

8.5.1. SEA ohne VLAN-Tagging

Wird kein VLAN-Tagging verwendet (äußerst unwahrscheinlich für die meisten Umgebungen), werden für einen SEA nur 2 Adapter benötigt, ein Trunking-Adapter für die Anbindung an den virtuellen Hypervisor Switch und ein physikalischer Adapter für die Anbindung an ein externes Netzwerk. Der Trunking-Adapter kann IEEE802.1q kompatibel sein, muß er aber nicht, da ja kein VLAN-Tagging verwendet wird. Für die anzugebende PVID kann ein beliebiges VLAN verwendet werden, die gewählte VLAN-ID wird nur innerhalb des virtuellen Switches verwendet.

Wie schon im Kapitel über Virtual Ethernet besprochen, sollte jedes angebundene externe Netzwerk einen eigenen virtuellen Switch verwenden, um die extern getrennten Netze nicht innerhalb des Managed Systems zusammen zuführen. Daher legen wir zunächst einen neuen virtuellen Switch mit dem Kommando „*ms addvswitch*“ an:

```
$ ms addvswitch ms05 ETHTEST1
$
```

Dem neuen virtuellen Switch *ETHTEST1* sind noch keine VLANs bekannt:

```
$ ms lsvswitch ms05
NAME  VSWITCH             SWITCH_MODE  VLAN_IDS
ms05  ETHNAS              VEB          1,2,13,19
ms05  ETHERNET0(Default)  VEB          1,2,100,110,200
ms05  ETHCTRL             VEB          1,2
ms05  ETHTEST1            VEB          none
$
```

Für den benötigten Trunking-Adapter wird eine freie Slot-Nummer benötigt, wir listen daher alle verwendeten virtuellen Slots auf, um eine geeignete freie Slot-Nummer auszuwählen:

```
$ lpar lsvslot ms05-vio1
SLOT  REQ  ADAPTER_TYPE   STATE  DATA
0     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
1     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
10    No   eth            1      PVID=100 VLANS= ETHERNET0 A556570DFD0A
20    No   eth            1      PVID=1 VLANS= ETHCTRL A556570DFD14
21    No   eth            1      TRUNK(1) IEEE PVID=1 VLANS=100,110 ETHERNET0 A556570DFD15
22    No   eth            1      TRUNK(1) IEEE PVID=2 VLANS=200 ETHERNET0 A556570DFD16
40    No   eth            1      PVID=2 VLANS= ETHCTRL A556570DFD28
41    No   eth            1      TRUNK(1) IEEE PVID=1 VLANS=13 ETHNAS A556570DFD29
42    No   eth            1      TRUNK(1) IEEE PVID=2 VLANS=19 ETHNAS A556570DFD2A
$
```

Als Slot-Nummer für den Trunking-Adapter haben wir uns für den freien Slot *51* entschieden. Der Trunking-Adapter kann mit dem Kommando „*lpar addeth*“ (*add virtual ethernet*) angelegt werden. Dabei muß eine Trunking-Priorität (*1-15*) angegeben werden. Zusätzlich muß der zu verwendende virtuelle Switch und die Slot-Nummer für den Adapter angegeben werden. Als PVID verwenden wir hier die VLAN-ID 1:

```
$ lpar addeth -t 1 -s ETHTEST1 ms05-vio1 51 1
$
```

(**Hinweis:** Die Angabe einer Trunking-Priorität macht den virtuellen Ethernet-Adapter zu einem Trunking-Adapter!)

Das Kommando „*vios lssea*“ bietet die Option ‚*-c*‘ (*candidates*), um mögliche Kandidaten für einen Shared Ethernet Adapter aufzulisten. Dabei werden sowohl Kandidaten für den physikalischen Adapter des SEA, als auch Kandidaten für die Trunking-Adapter des SEA aufgelistet:

```
$ vios lssea -c ms05-vio1
NAME   STATUS     PHYSLOC                        PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C7-T4   pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent2   Available  U78AA.001.VYRGU0Q-P1-C7-T3   pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent1   Available  U78AA.001.VYRGU0Q-P1-C7-T2   pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent38  Available  U8205.E6C.05E4E5Q-V1-C51-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Die Option ‚*-c*' ist sehr nützlich auf größeren Systemen mit einer Vielzahl von Adaptern. Es werden nur unbenutzte Adapter angezeigt. Adapter die in anderen SEAs schon in Verwendung sind, werden nicht aufgelistet.

Der gerade angelegte Trunking-Adapter in Slot *51* ist der Ethernet Adapter *ent38*. Der physikalische Port *ent1* soll für den neu anzulegenden SEA verwendet werden. Das Kommando um einen SEA anzulegen ist „*vios mksea*" (*make SEA*). Es muß mindestens der physikalische Adapter für den SEA, erstes Argument nach dem Namen des Virtual-I/O-Servers, und ein Trunking-Adapter angegeben werden:

```
$ vios mksea ms05-vio1 ent1 ent38
SEA ent39 created
$
```

Möchte man alle zu einem SEA zugehörigen Adapter (virtuelle und physikalische) sehen, kann die Option ‚*-a*' (*all adapters*) von „*vios lssea*" verwendet werden:

```
$ vios lssea -a ms05-vio1 ent39
SEA    LNAGG  NAME   TYPE     STATUS  SPEED                  VSWITCH   MODE  PHYSLOC
ent39  -      ent1   real     Up      1000 Mbps Full Duplex  -         -     U78AA.001.VYRGU0Q-P1-C7-T2
ent39  -      ent38  virtual  -       -                      ETHTEST1  VEB   U8205.E6C.05E4E5Q-V1-C51-T1
$
```

Der physikalische Adapter *ent1* (Spalte *TYPE*, Wert *real*) hat den Status *Up* und hat aktuelle eine Geschwindigkeit von *1000 Mbps*. Der einzige Trunking-Adapter (Spalte *TYPE*, Wert *virtual*) ist an den virtuellen Switch *ETHTEST1* angebunden.

In Bild 8.3 ist der neu angelegte SEA dargestellt. In der linken oberen Client-LPAR wird ein ungetaggtes Ethernet Frame über den virtuellen Ethernet Adapter *ent0* der LPAR (mit PVID *1*) an den virtuellen Hypervisor Switch *ETHTEST1* weitergeleitet. Der Frame wird mit einem VLAN-Tag für die PVID *1* versehen und zum Trunking-Port für das VLAN *1* weitergeleitet. Dort wird das VLAN-Tag wieder entfernt (da die VLAN-ID mit der PVID des Trunking-Ports übereinstimmt) und dann an den SEA *ent39* des Virtual-I/O-Servers als untagged Ethernet Frame weitergeleitet. Der SEA gibt das Ethernet Frame über seinen physikalischen Adapter *ent1* an das angebundene externe Netzwerk als untagged Frame weiter.

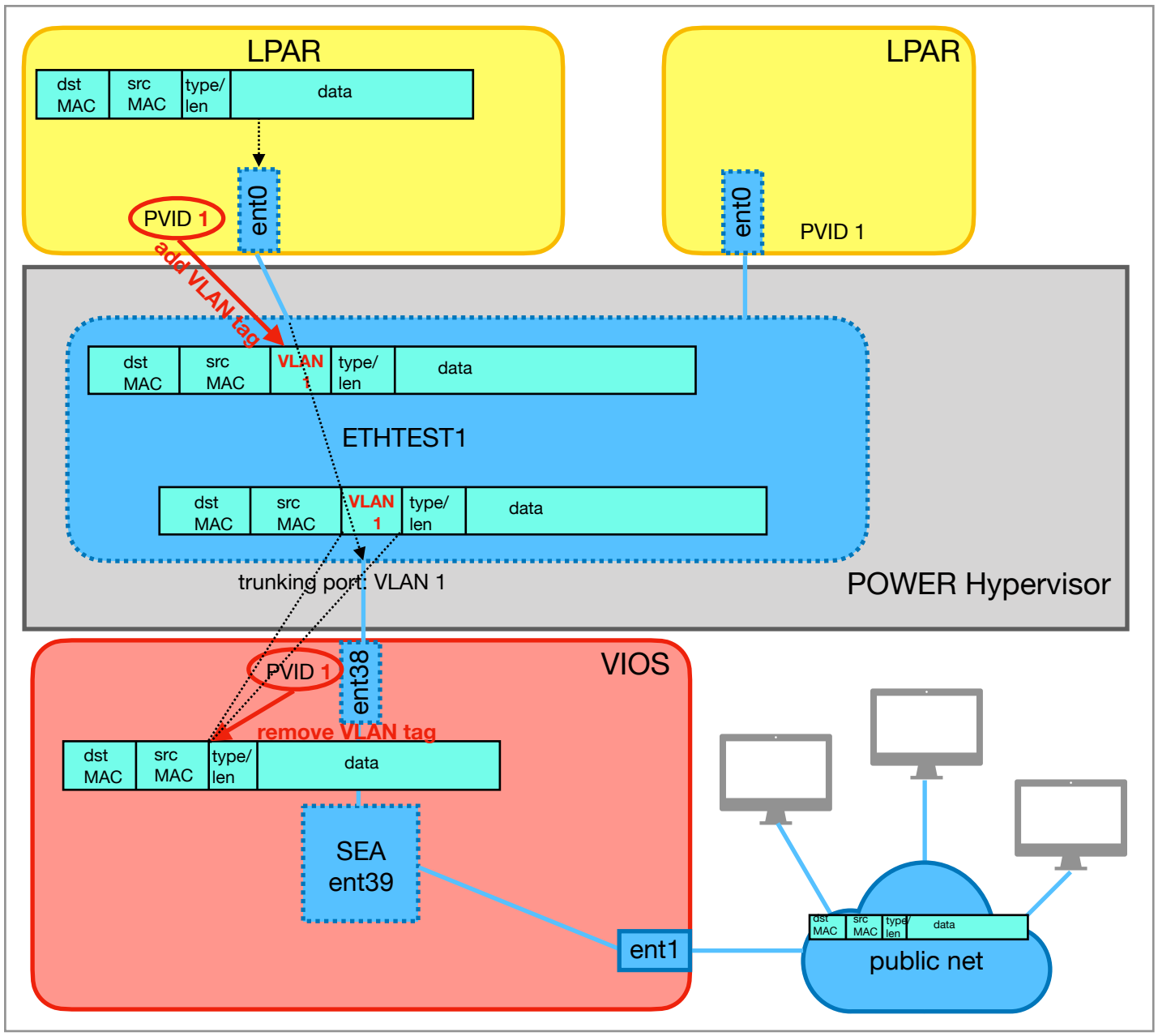


Bild 8.3: Transport eines Ethernet Frames mit Tagging und Untagging auf dem virtuellen Switch *ETHTEST1*

Untagged Frames aus dem externen Netz an eine LPAR innerhalb des Managed Systems nehmen den umgekehrten Weg.

Client-LPARs müssen die PVID *1* verwenden, wenn sie mit der Außenwelt Daten austauschen wollen, da es nur für das VLAN *1* einen Trunking-Port gibt!

8.5.2. SEA mit VLAN-Tagging

Werden VLANs verwendet, das dürfte der Standard sein, gibt es mehrere Möglichkeiten wie ein Shared Ethernet Adapter genau aussehen kann. Ein Trunking-Adapter unterstützt bis zu 20 VLANs neben der Port-VLAN-ID. Erst wenn mehr als 20 VLANs verwendet werden sollen, müssen zusätzliche Trunking-Adapter verwendet werden. Ein SEA kann bis zu 16 Trunking-Adapter besitzen. Wieviele Trunking-Adapter letztlich verwendet werden, hängt vom Administrator und der Anzahl der zu unterstützenden VLANs ab. Sollen z.B. 10 VLANs verwendet werden, können zwischen einem und zehn Trunking-Adapter verwendet werden. In der Praxis wird typischerweise eine kleinere Anzahl von VLANs auf einzelne Trunking-Adapter konfiguriert (typischerweise 3 bis 6 VLANs), das hängt aber stark von der Umgebung ab und soll hier nicht weiter betrachtet werden.

Jeder Shared Ethernet Adapter unterstützt maximal ein VLAN *untagged*. Typischerweise werden aber alle benötigten VLANs als *tagged* VLANs verwendet und für die notwendigen Port-VLAN-IDs der Trunking-Adapter werden unbenutzte VLANs verwendet. In vielen Umgebungen werden dafür einstellige und niedrige zweistellige VLAN-IDs reserviert, die dann nicht für VLANs verwendet werden.

Im folgenden wird ein weiterer Shared Ethernet Adapter angelegt, dieses Mal mit VLAN-Tagging und den VLANs *100*, *110*, *200*, *205* und *210*. Es sollen 2 Trunking-Adapter angelegt werden, einer mit den VLANs *100* und *110* und der zweite mit den VLANs *200*, *205* und *210*. Zunächst wird aber ein weiterer virtueller Switch für den neuen Shared Ethernet Adapter angelegt:

```
$ ms addvswitch ms05 ETHTEST2
$
```

Für die beiden Trunking-Adapter verwenden wir die beiden freien virtuellen Slots 6*1* und *62*:

```
$ lpar addeth -t 1 -i -s ETHTEST2 ms05-vio1 61 1 100,110
$ lpar addeth -t 1 -i -s ETHTEST2 ms05-vio1 62 2 200,205,210
$
```

Zur Kontrolle sind hier noch einmal die Kandidaten für einen weiteren SEA aufgelistet:

```
$ vios lssea -c ms05-vio1
NAME   STATUS     PHYSLOC                       PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C7-T4    pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent2   Available  U78AA.001.VYRGU0Q-P1-C7-T3    pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent39  Available  U8205.E6C.05E4E5Q-V1-C61-T1   vio0    Virtual I/O Ethernet Adapter (l-lan)
ent41  Available  U8205.E6C.05E4E5Q-V1-C62-T1   vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Als physikalischen Adapter verwenden wir *ent2* und als Trunking-Adapter die beiden gerade angelegten Adapter *ent39* (Slot *61*) und *ent41* (Slot *62*):

```
$ vios mksea ms05-vio1 ent2 ent39 ent41
SEA ent42 created
$
```

Anstelle der Gerätenamen kann auch der Physical Location Code (bzw. ein eindeutiger Suffix) angegeben werden:

```
$ vios mksea ms05-vio1 P1-C7-T3 C61-T1 C62-T1
SEA ent42 created
$
```

Alle Adapter eines SEAs lassen sich wie bisher mittels „*vios lssea*“ anzeigen:

```
$ vios lssea -a ms05-vio1 ent42
SEA    LNAGG  NAME   TYPE     STATUS  SPEED                  VSWITCH   MODE  PHYSLOC
ent42  -      ent2   real     Up      1000 Mbps Full Duplex  -         -     U78AA.001.VYRGU0Q-P1-C7-T3
ent42  -      ent41  virtual  -       -                      ETHTEST2  VEB   U8205.E6C.05E4E5Q-V1-C62-T1
ent42  -      ent39  virtual  -       -                      ETHTEST2  VEB   U8205.E6C.05E4E5Q-V1-C61-T1
$
```

Die Verteilung der unterstützten VLANs des SEAs lässt sich mit der Option ‚-*V*' (*VLANs*) auflisten:

```
$ vios lssea -V ms05-vio1 ent42
SEA     LNAGG   NAME    TYPE      VSWITCH    MODE   ACTIVE   PRIORITY   PVID   VLAN_TAG_IDS
ent42   -       ent2    real      -          -      -        -          -      -
ent42   -       ent41   virtual   ETHTEST2   VEB    True     1          2      200,205,210
ent42   -       ent39   virtual   ETHTEST2   VEB    True     1          1      100,110
$
```

In Bild 8.4 ist der erzeugte Shared Ethernet Adapter dargestellt.

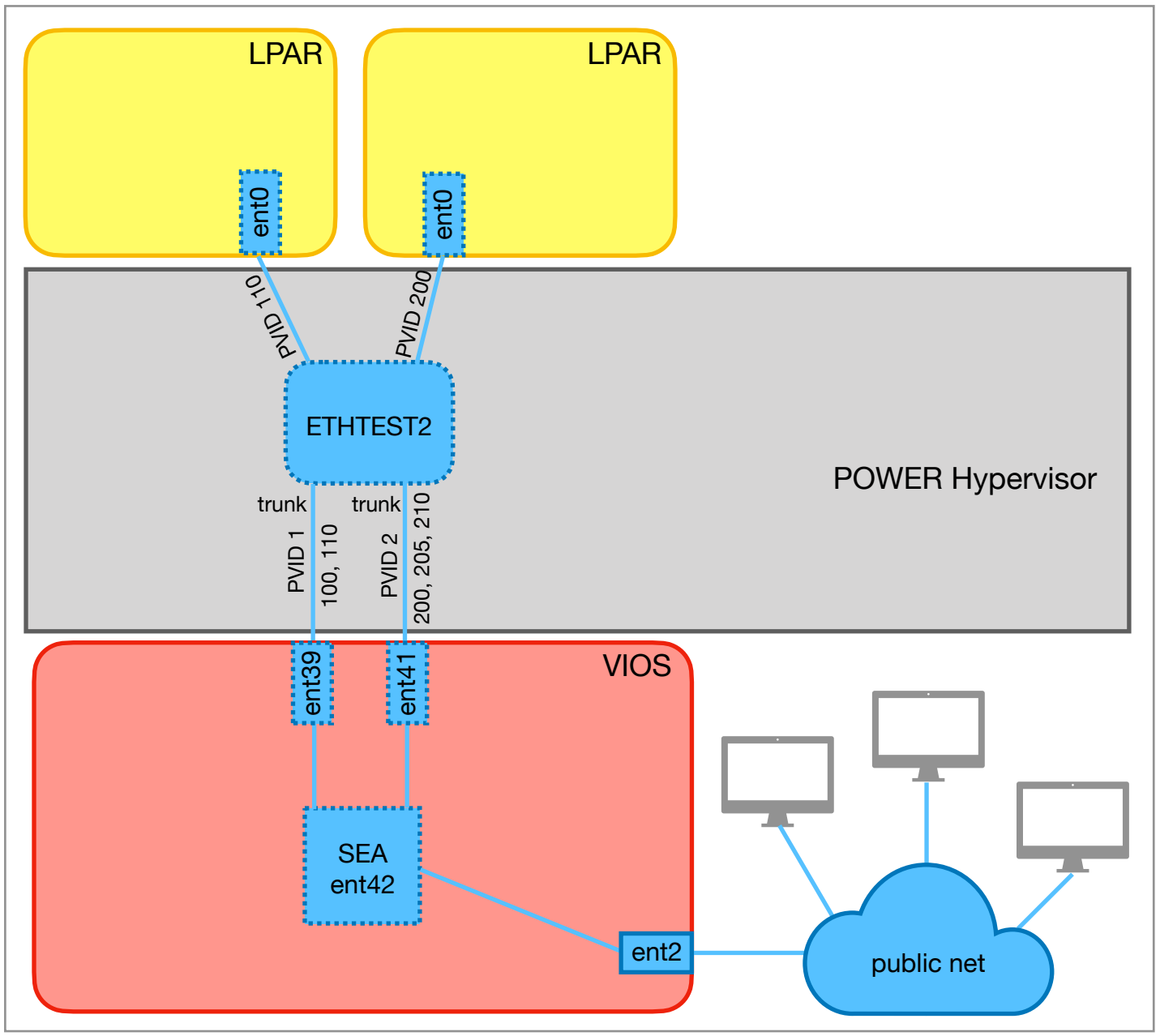


Bild 8.4: SEA mit 2 Trunking-Adaptern und 5 VLANs

Im Folgenden wird der Weg eines Ethernet Frames von einer LPAR mit PVID *110* zu einem externen Host in einzelnen Schritten dargestellt:

1. Die LPAR versendet ein *untagged* Ethernet Frame über den virtuellen Ethernet Adapter *ent0* (Bild 8.5a).

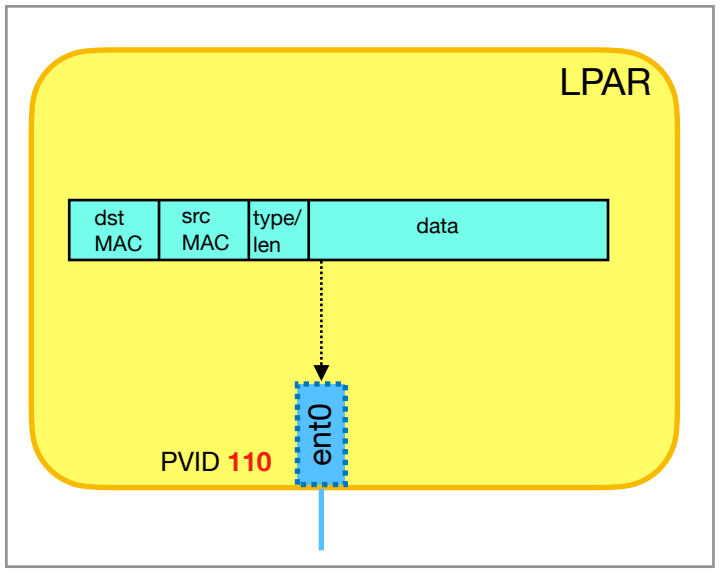


Bild 8.5a: LPAR mit PVID *110* versendet Ethernet Frame über den virtuellen Ethernet Adapter *ent0*.

2. Das Ethernet Frame wird an den angebundenen virtuellen Switch *ETHTEST2* weitergeleitet und dort mit einem VLAN-Header versehen (Bild 8.5b). Die VLAN-ID ist die PVID des virtuellen Ethernet Adapters.

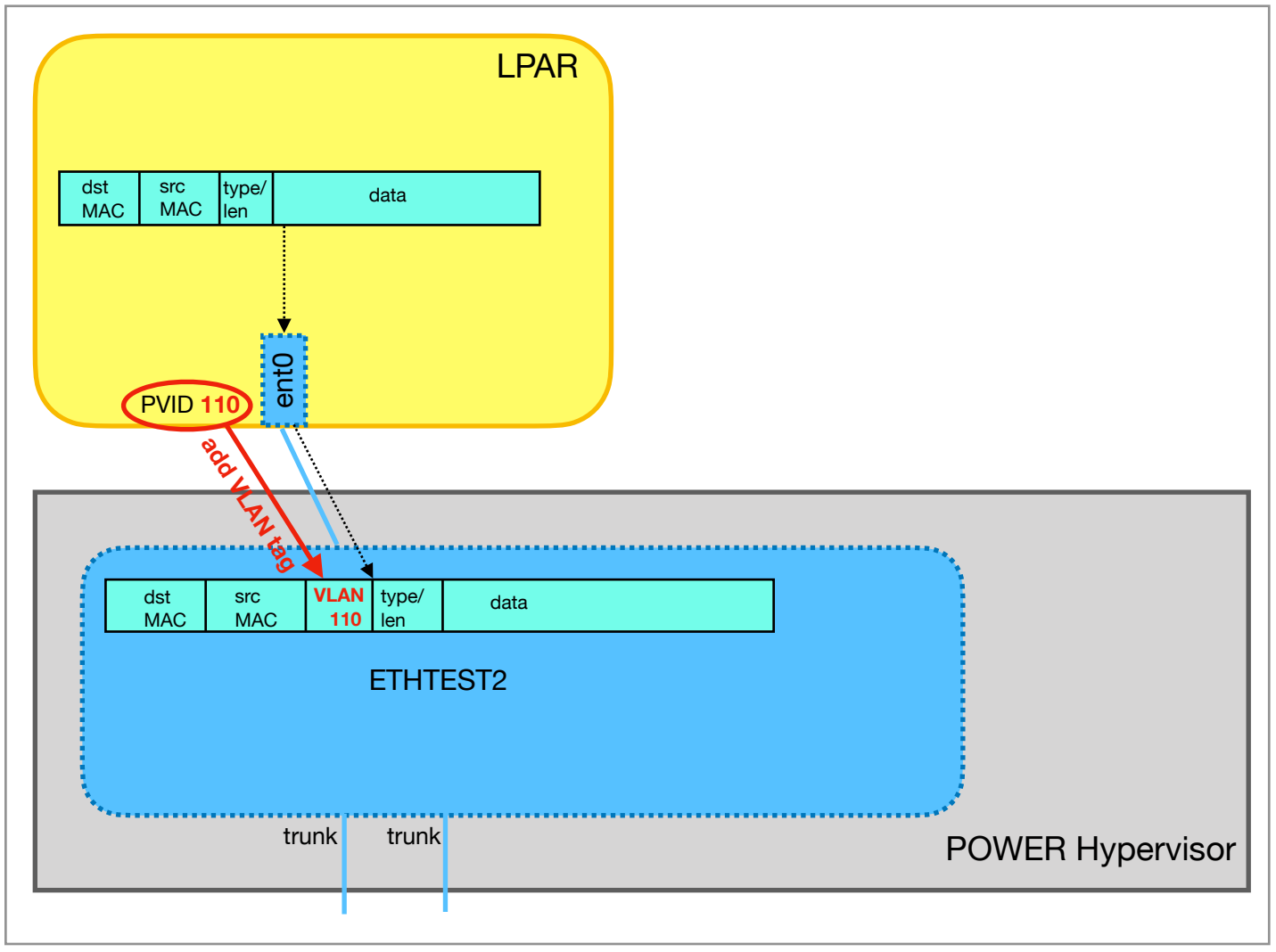


Bild 8.5b: Das Ethernet Frame wird an den virtuellen Switch *ETHTEST2* weitergegeben und dort mit der VLAN-ID *110* (PVID) getagged.

3. Da das Ziel nicht direkt am virtuellen Ethernet Switch *ETHTEST2* angebunden ist, verwendet der virtuelle Ethernet Switch *ETHTEST2* den Trunking-Adapter für das VLAN *110* um den Frame weiterzuleiten. Der Trunking-Adapter für das VLAN *110* ist der Adapter *ent39* (Bild 8.5c), der zum Shared Ethernet Adapter *ent42* des dargestellten Virtual-I/O-Server gehört.

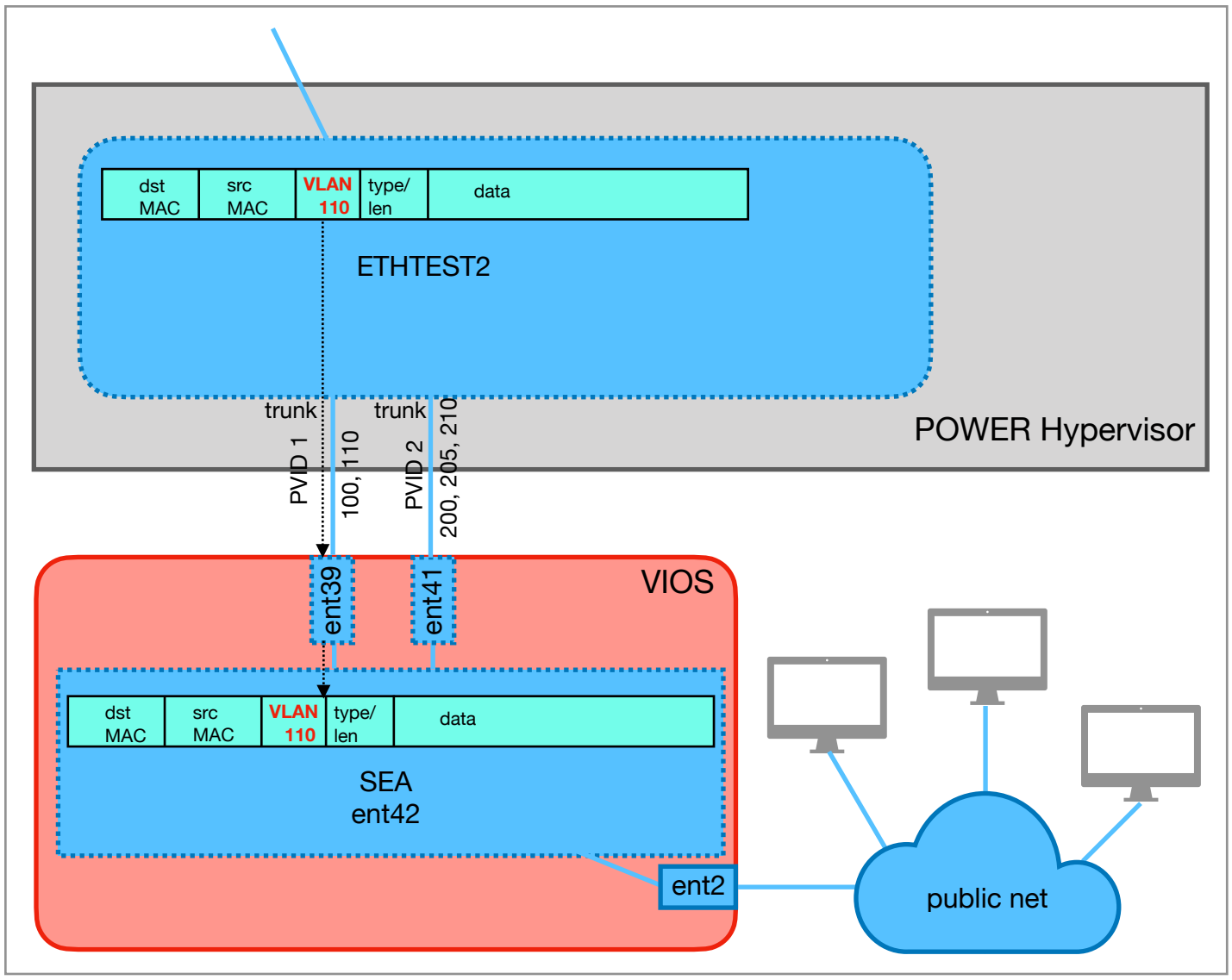


Bild 8.5c: Das Ethernet Frame wird vom virtuellen Switch *ETHTEST2* über den Trunking Adapter *ent39* für das VLAN *110* an den SEA *ent42* weitergeleitet.

4. Bild 8.5d zeigt schließlich wie der Shared Ethernet Adapter *ent42* den Ethernet Frame über seinen physikalischen Adapter *ent2* ins externe Netzwerk weiterleitet. Das Ethernet Frame ist immer noch mit der VLAN-ID *110* versehen. Die Switches im externen Netzwerk leiten dann das Ethernet Frame zum Zielsystem weiter. Im Bild ist angenommen das das Ziel-System selbst VLAN-Tagging unterstützt und das Frame mit VLAN-Header zugesendet bekommt, es ist aber auch möglich das das Ziel-System einen ungetaggten Port mit der PVID *110* verwendet und damit das Frame ohne VLAN-Header bekommt.

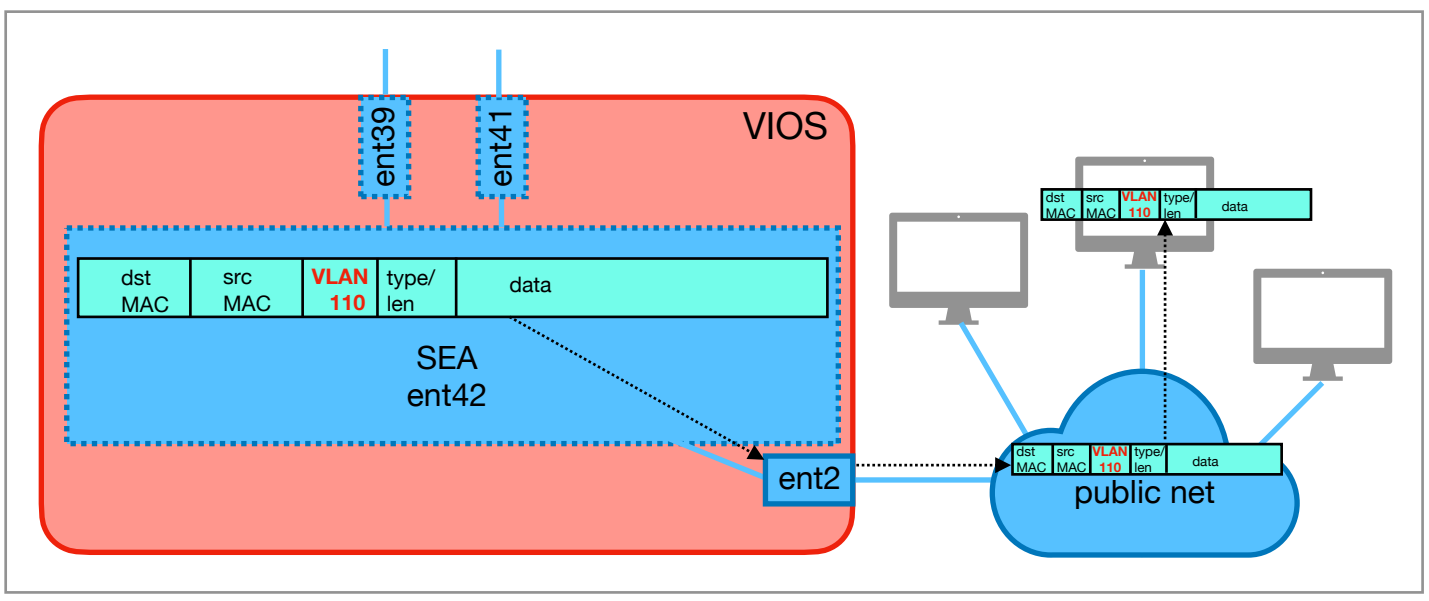


Bild 8.5d: Der SEA *ent42* sendet den Ethernet Frame über den physikalischen Adapter *ent2* ins externe Netzwerk. Dort wird er dann an das Ziel-System weitergeleitet.

Nachfolgend sind die Schritte beim Transport eines Ethernet Frames von einem externen Host zu einer LPAR dargestellt:

1. Ein externer Host versendet ein Ethernet Frame in das VLAN 110 an eine LPAR (Bild 8.6a). Das Ethernet Frame wurde entweder vom externen Host selbst mit einem VLAN-Header versehen, oder der VLAN-Header wurde vom angebundenen Netzwerk-Switch in das Ethernet Frame eingefügt.

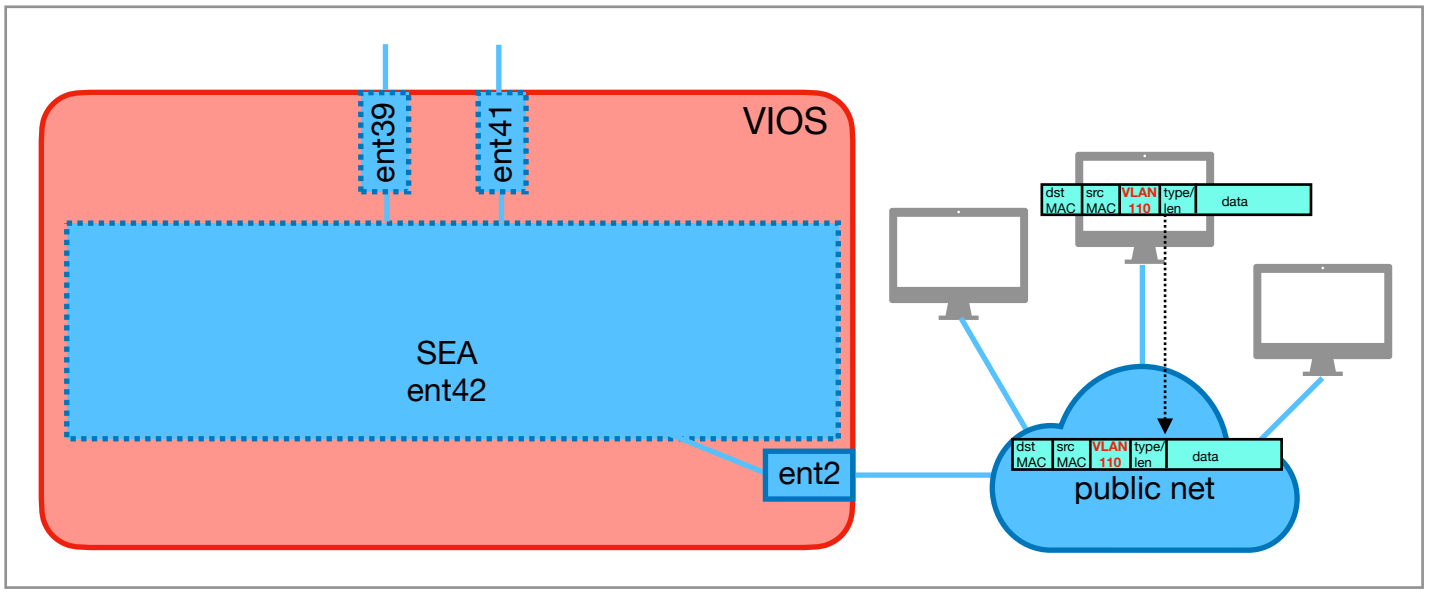


Bild 8.6a: Ein externer Host sendet ein Ethernet Frame ins angebundene Netzwerk.

2. Das Ethernet Frame wird in Richtung Ziel-LPAR an den physikalischen Adapter *ent2* des Virtual-I/O-Servers weitergeleitet, der Bestandteil des Shared Ethernet Adapters *ent42* auf dem Virtual-I/O-Server ist (Bild 8.6b).

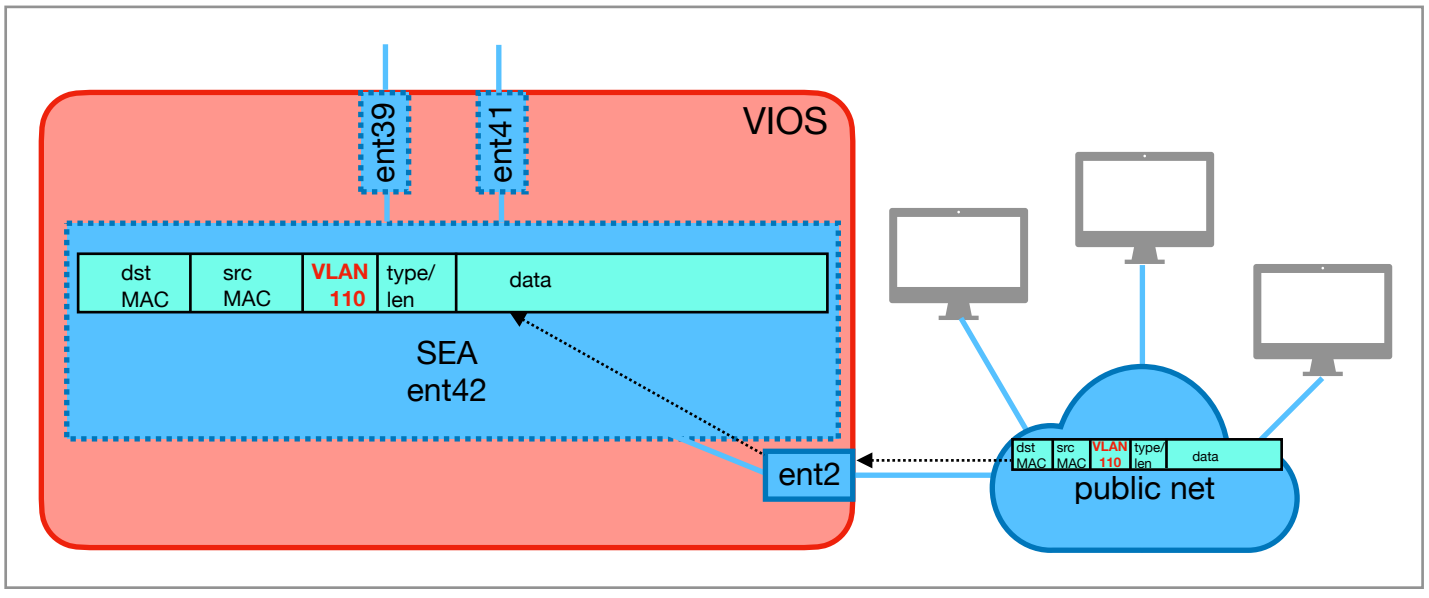


Bild 8.6b: Das Ethernet Frame wird von den externen Switches an den physikalischen Adapter *ent2* des Managed Systems weitergeleitet. Dieser ist Teil des SEAs *ent42* auf einem Virtual-I/O-Server.

3. Der Shared Ethernet Adapter *ent42* wählt den Trunking-Adapter *ent39* für das Weiterleiten des Ethernet Frames an den virtuellen Switch *ETHTEST2* aus, da der Adapter *ent39* das VLAN *110* unterstützt (Bild 8.6c).

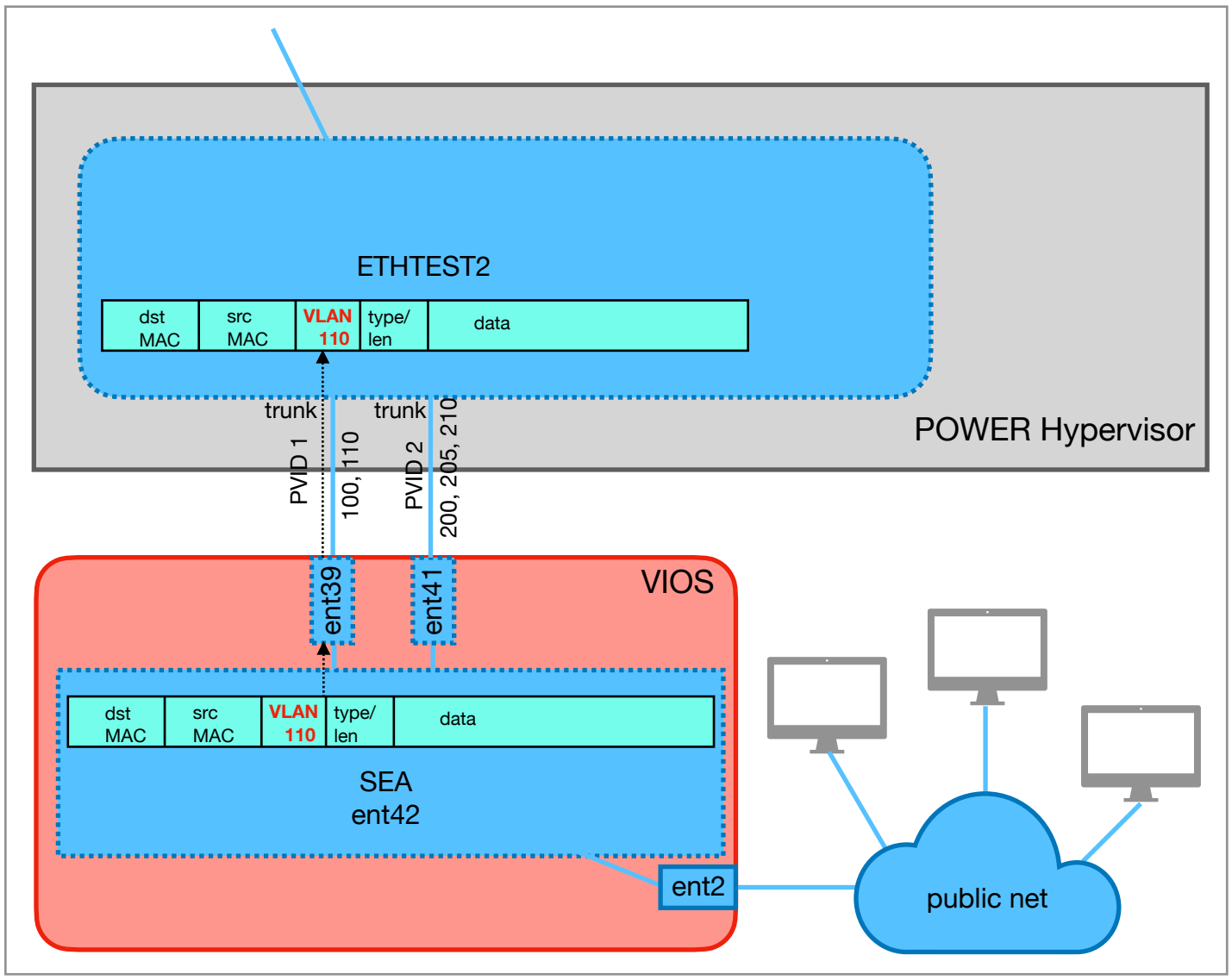


Bild 8.6c: Da der Ethernet Frame die VLAN-ID *110* hat, gibt der SEA *ent42* den Frame über den Trunking-Adapter *ent39* an den virtuellen Switch *ETHTEST2* weiter.

4. Der virtuelle Ethernet Switch *ETHTEST2* leitet den Ethernet Frame über den Ziel-Port an den Adapter *ent0* der Ziel-LPAR weiter, Bild 8.6d. Da der virtuelle Adapter das VLAN *110* als Port-VLAN-ID besitzt, wird beim Transport des Frames der VLAN-Header entfernt. Der Adapter *ent0* bekommt den Ethernet Frame als ungetaggtes Frame.

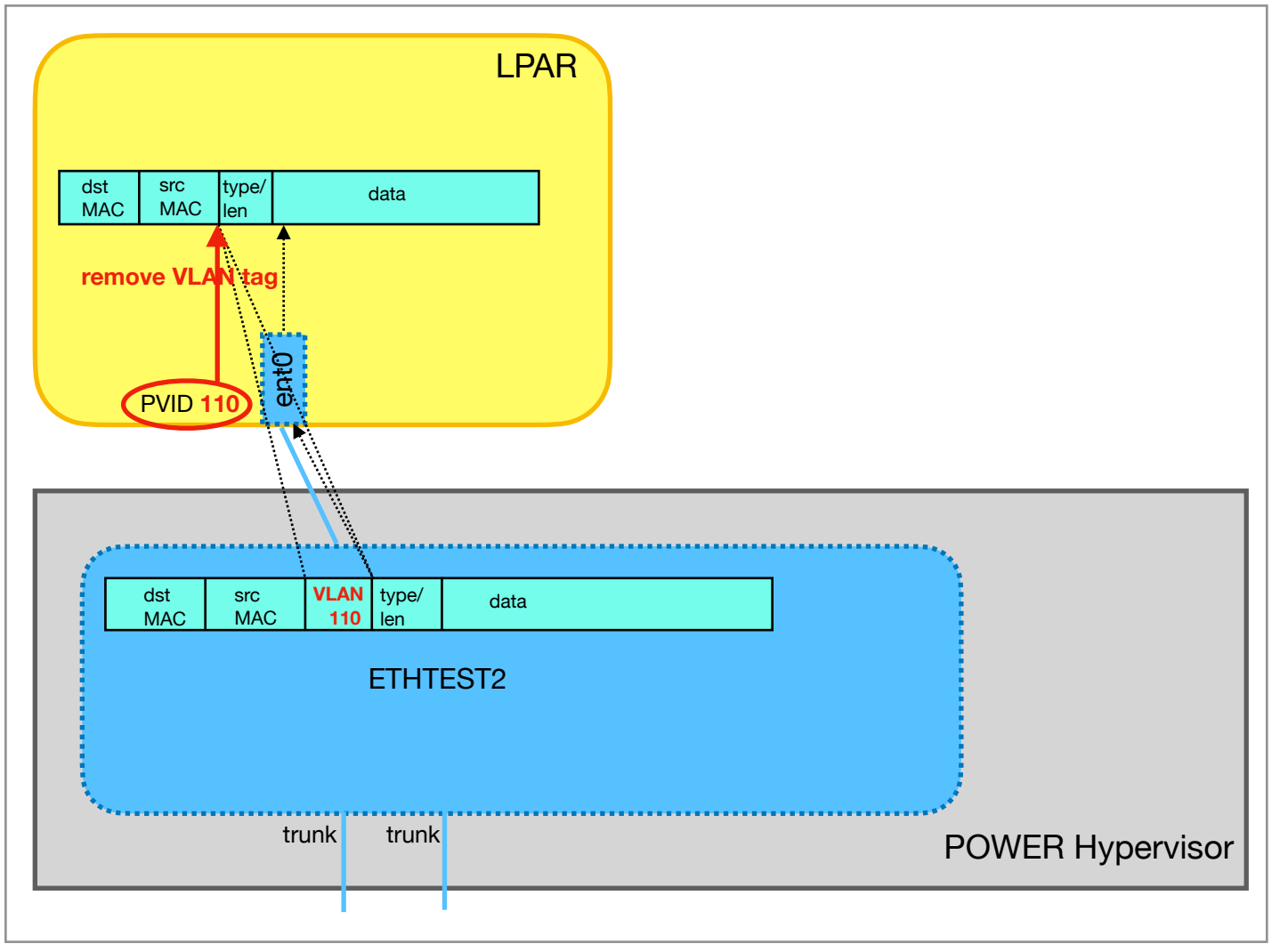


Bild 8.6d: Der virtuelle Ethernet Switch ETHTEST2 leitet den Ethernet Frame an den Adapter der Ziel-LPAR weiter, dabei wird der VLAN-Header entfernt.

Eine besondere Situation ergibt sich wenn eine LPAR eine der PVIDs der Trunking-Adapter als VLAN-ID verwendet. Bei dem oben angelegten SEA haben die beiden Trunking-Adapter die PVIDs *1* und *2*. Wir schauen im Folgenden 2 LPARs an, welche die VLAN-IDs *1* bzw *2* verwenden. Zunächst schauen wir wieder den Weg von der LPAR zum externen Host an:

1. Die LPARs versenden je ein *untagged* Ethernet Frame über den virtuellen Ethernet Adapter *ent0* (Bild 8.7a).

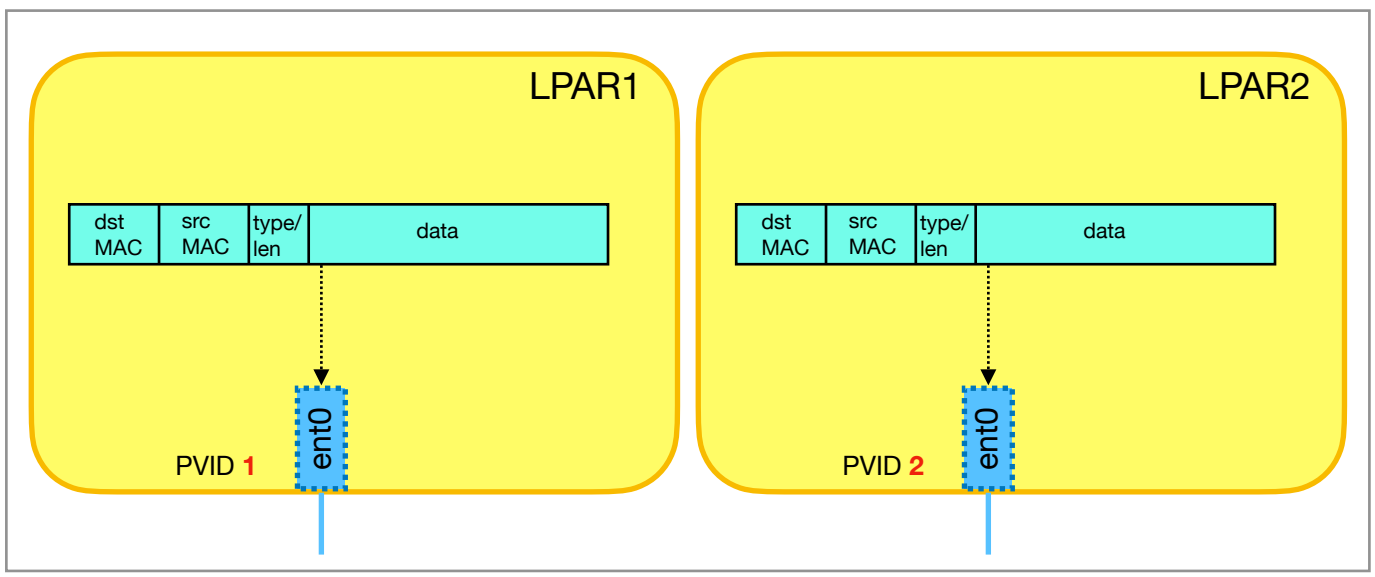


Bild 8.7a: *LPAR1* mit PVID *1* und *LPAR2* mit PVID *2* versenden jeweils ein Ethernet Frame über den virtuellen Ethernet Adapter *ent0*.

2. Die beiden Ethernet Frames werden an den angebundenen virtuellen Switch *ETHTEST2* weitergeleitet und dort mit einem VLAN-Header versehen (Bild 8.7b). Der Frame von LPAR1 bekommt die VLAN-ID 1 und der Frame von LPAR2 bekommt die VLAN-ID 2 als VLAN-Header.

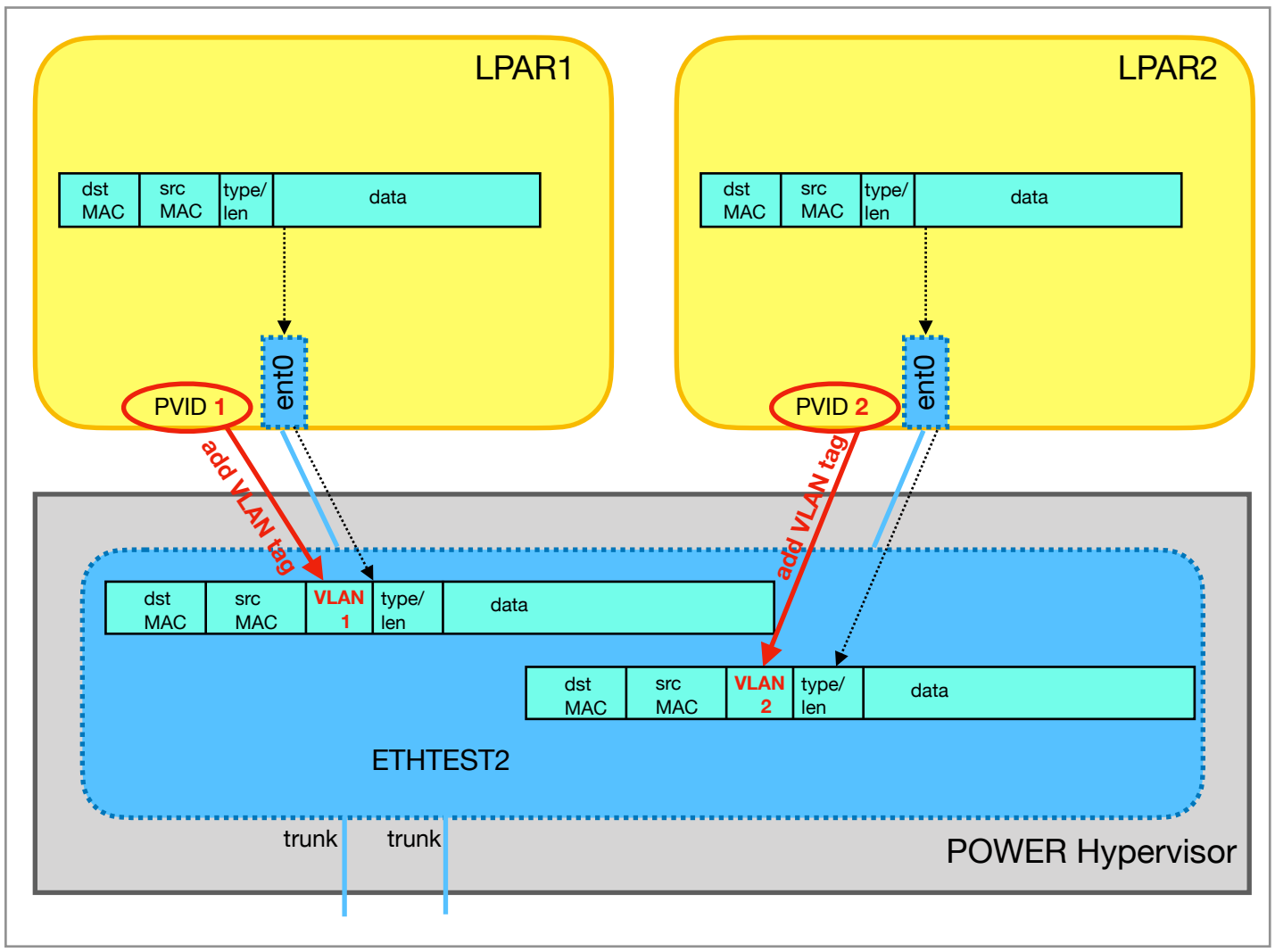


Bild 8.7b: Die Ethernet Frames werden an den virtuellen Switch *ETHTEST2* weitergegeben und dort mit der VLAN-ID *1* bzw. *2* getagged.

3. Das Frame von *LPAR1* mit der VLAN-ID *1* wird vom virtuellen Switch über den zugehörigen Trunking-Adapter *ent39* an den SEA *ent42* weitergeleitet. Da das VLAN *1* die PVID des Trunking-Adapters ist, wird der VLAN-Header dabei entfernt (Bild 8.7c). Das Frame von *LPAR2* mit der VLAN-ID *2* wird ebenfalls an den SEA *ent42* weitergeleitet. Allerdings ist hier der zugehörige Trunking-Adapter der Adapter *ent41*. Auch hier wird der VLAN-Header entfernt, da das VLAN *2* die PVID des Trunking-Adapters *ent41* ist. Beide Frames sind jetzt untagged! Zu welchem VLAN die beiden Ethernet Frames ursprünglich gehört haben, ist nicht mehr erkennbar.

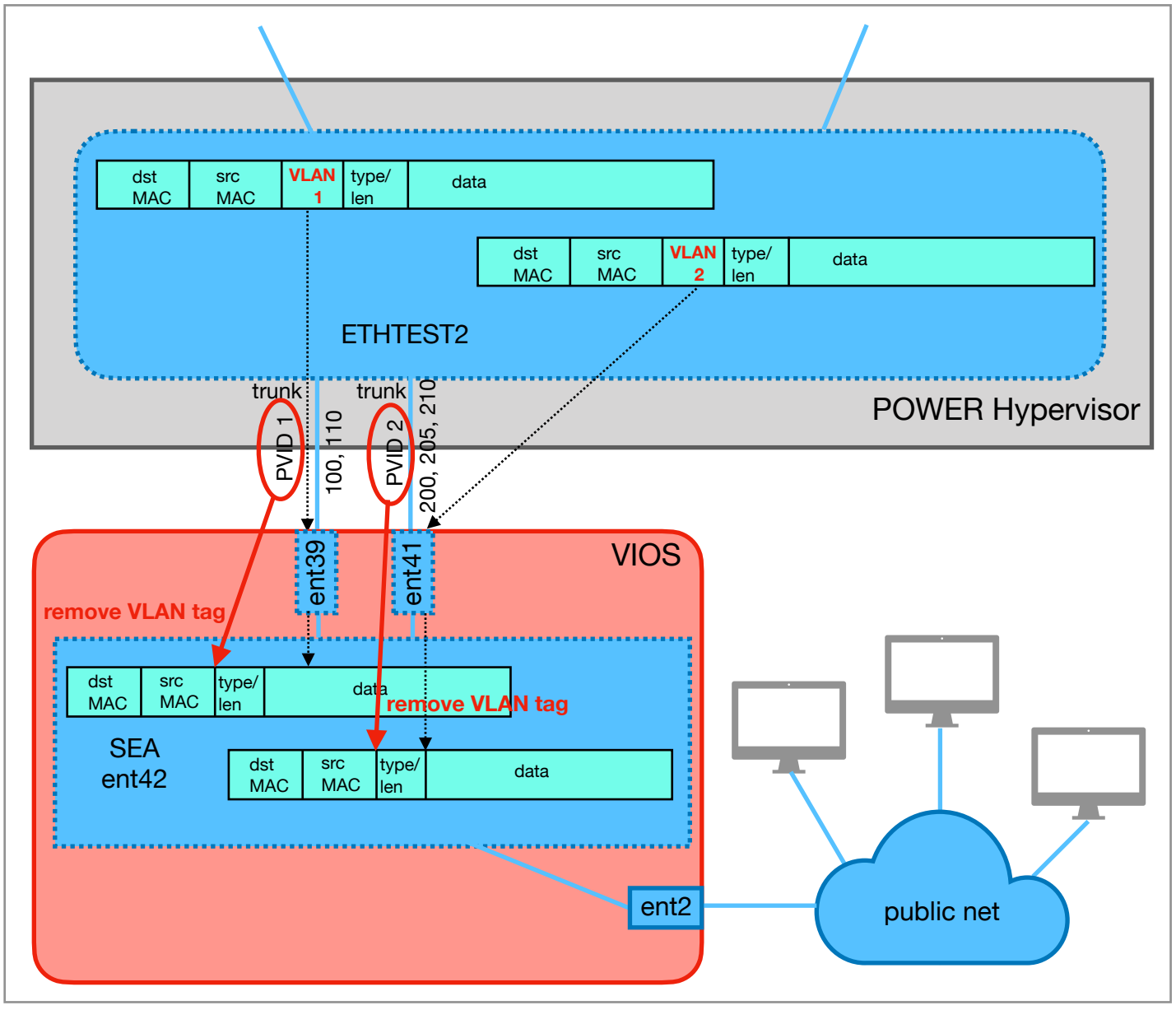


Bild 8.7c: Die beiden Ethernet Frames werden vom virtuellen Switch *ETHTEST2* über die Trunking Adapter *ent39* bzw. ent41 an den SEA *ent42* weitergeleitet. Dabei werden die VLAN-Header entfernt.

4. Beide *untagged* Ethernet Frames werden vom Shared Ethernet Adapter *ent42* über den physikalischen Adapter *ent2* an das externe Netzwerk weitergeleitet (Bild 8.7d).

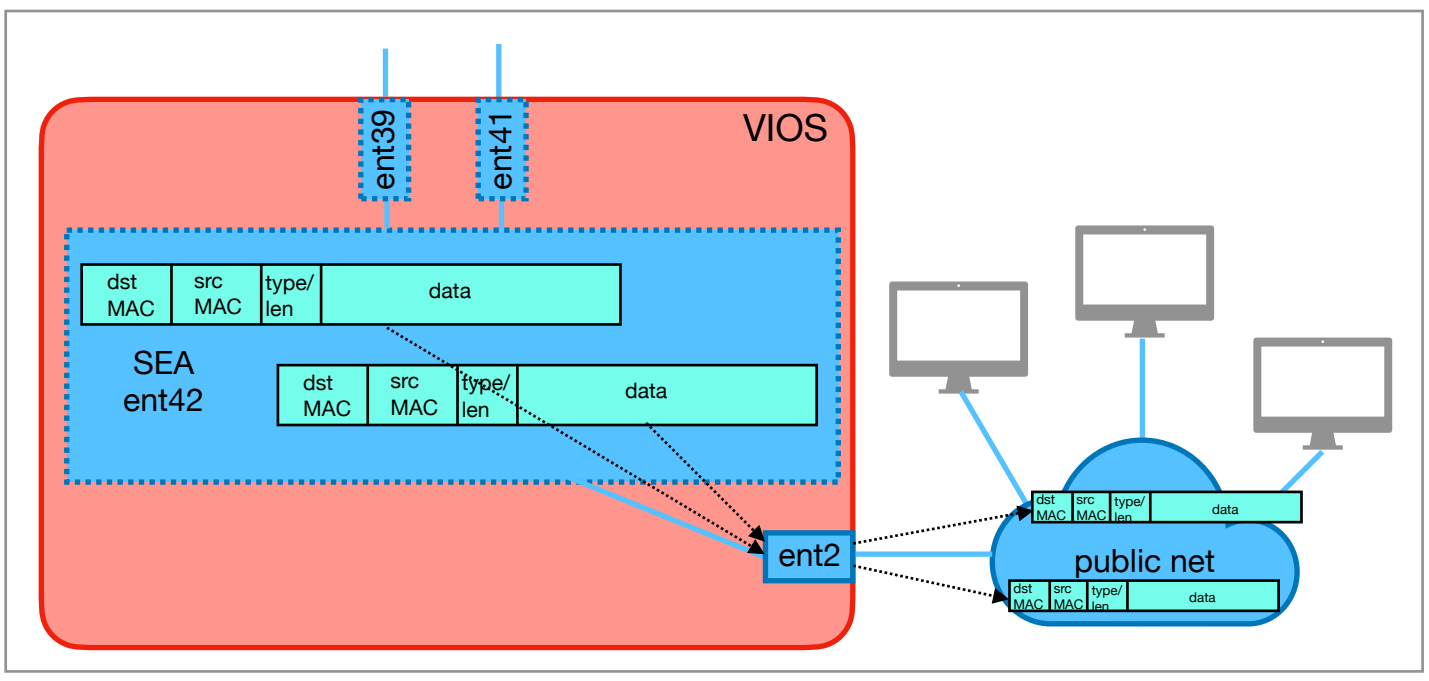


Bild 8.7d: Der SEA *ent42* leitet beide *untagged* Ethernet Frames über den physikalischen Adapter *ent2* an das externe Netzwerk weiter.

5. Ob die Zielsysteme tatsächlich erreichbar sind, hängt davon ab ob diese mit *untagged* Frames von *ent2* aus erreichbar sind. Für den physikalischen Adapter *ent2* dürfte auf dem zugehörigen Switch-Port eine Port-VLAN-ID konfiguriert sein. D.h. *untagged* Frames die von *ent2* ins externe Netzwerk gesendet werden, werden dieser Port-VLAN-ID zugeordnet.

Interessant ist der Weg eines Frames vom externen Netzwerk zur Client-LPAR für den Fall das das Frame keinen VLAN-Header besitzt:

1. Ein Ethernet Frame ohne VLAN-Header wird vom externen Netzwerk an den physikalischen Adapter *ent2* des Managed Systems weitergeleitet (Bild 8.8a). Der physikalische Adapter *ent2* gehört zum Shared Ethernet Adapter *ent42* auf dem gezeigten Virtual-I/O-Server.

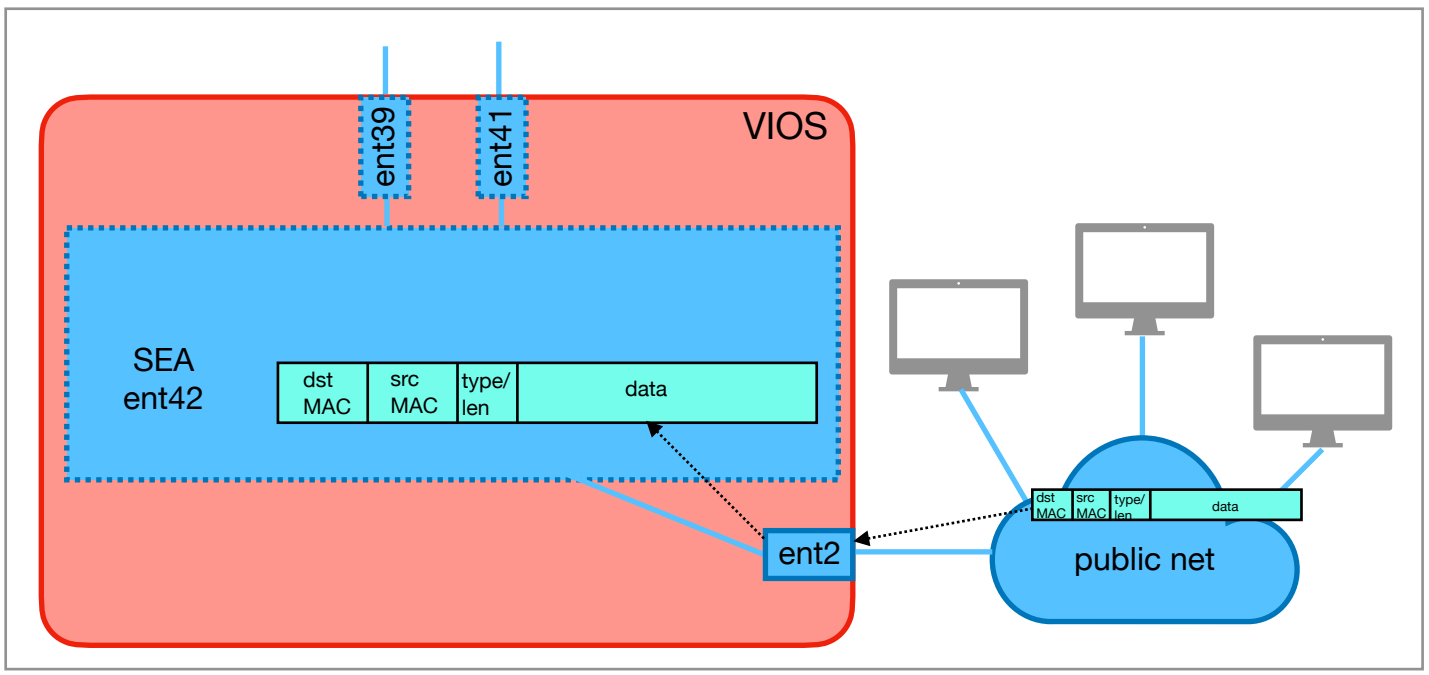


Bild 8.8a: Das externe Netzwerk sendet ein Ethernet Frame ohne VLAN-Header an den physikalischen Adapter *ent2*, welcher zum SEA *ent42* gehört.

2. Der Shared Ethernet Adapter *ent42* muß das Ethernet Frame weiterleiten. Allerdings ergibt sich hier das Problem das nicht klar ist welcher der beiden Trunking-Adapter zu verwenden ist. Das Ethernet Frame gehört zu keinem VLAN, da es *untagged* ist. Beide Trunking-Adapter *ent39* und *ent41* können prinzipiell *untagged* Frames weiterleiten. Wird der Frame über *ent39* weitergeleitet, wird das Frame mit der PVID *1* von *ent39* getaggt. Wird der Frame über *ent41* weitergeleitet, wird der Frame mit der PVID *2* von *ent41* getaggt. Je nachdem welcher Trunking-Adapter verwendet wird, würde der Frame also einem anderen VLAN zugeordnet! Bei einem SEA mit den maximal möglichen 16 Trunking-Adapters gäbe es dann schon 16 verschiedene Möglichkeiten.

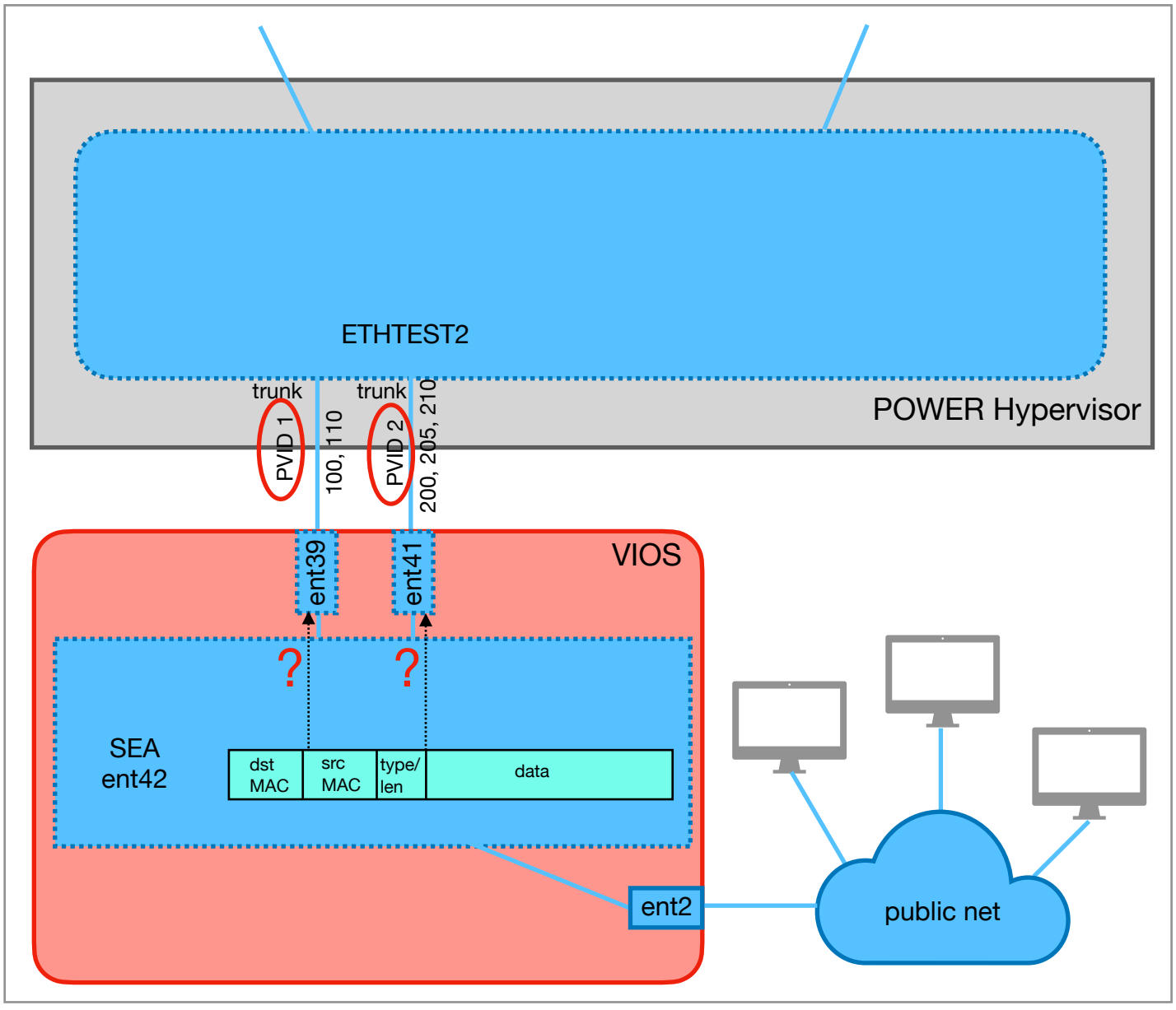


Bild 8.8b: Aus Sicht des SEAs *ent42* gibt es 2 verschiedene Möglichkeiten den Ethernet Frame weiterzuleiten, über *ent39* (PVID *1*) oder *ent41* (PVID *2*).

3. Über welchen Trunking-Adapter *untagged* Ethernet Frames von einem Shared Ethernet Adapter weitergeleitet werden, ist in der Konfiguration des SEAs festgelegt. Das entsprechende Attribut heisst *pvid_adapter*. Es kann beim Anlegen eines SEAs mit „*vios mksea*“ optional angegeben werden. Per Default wird der erste angegebene Trunking-Adapter verwendet. Wie Bild 8.8c zeigt, ist bei dem Shared Ethernet Adapter *ent42* der Trunking-Adapter *ent39* im Attribut *pvid_adapter* hinterlegt. Daher wird das *untagged* Frame über *ent39* an den virtuellen Switch *ETHTEST2* weitergeleitet, wobei ein VLAN-Header mit der PVID *1* von *ent39* hinzugefügt wird. Der Wert von *pvid_adapter* lässt sich über das Kommando „*vios lsattr*“ einfach anzeigen:

```
$ vios lsattr ms05-vio1 ent42 pvid_adapter
value
ent39
$
```

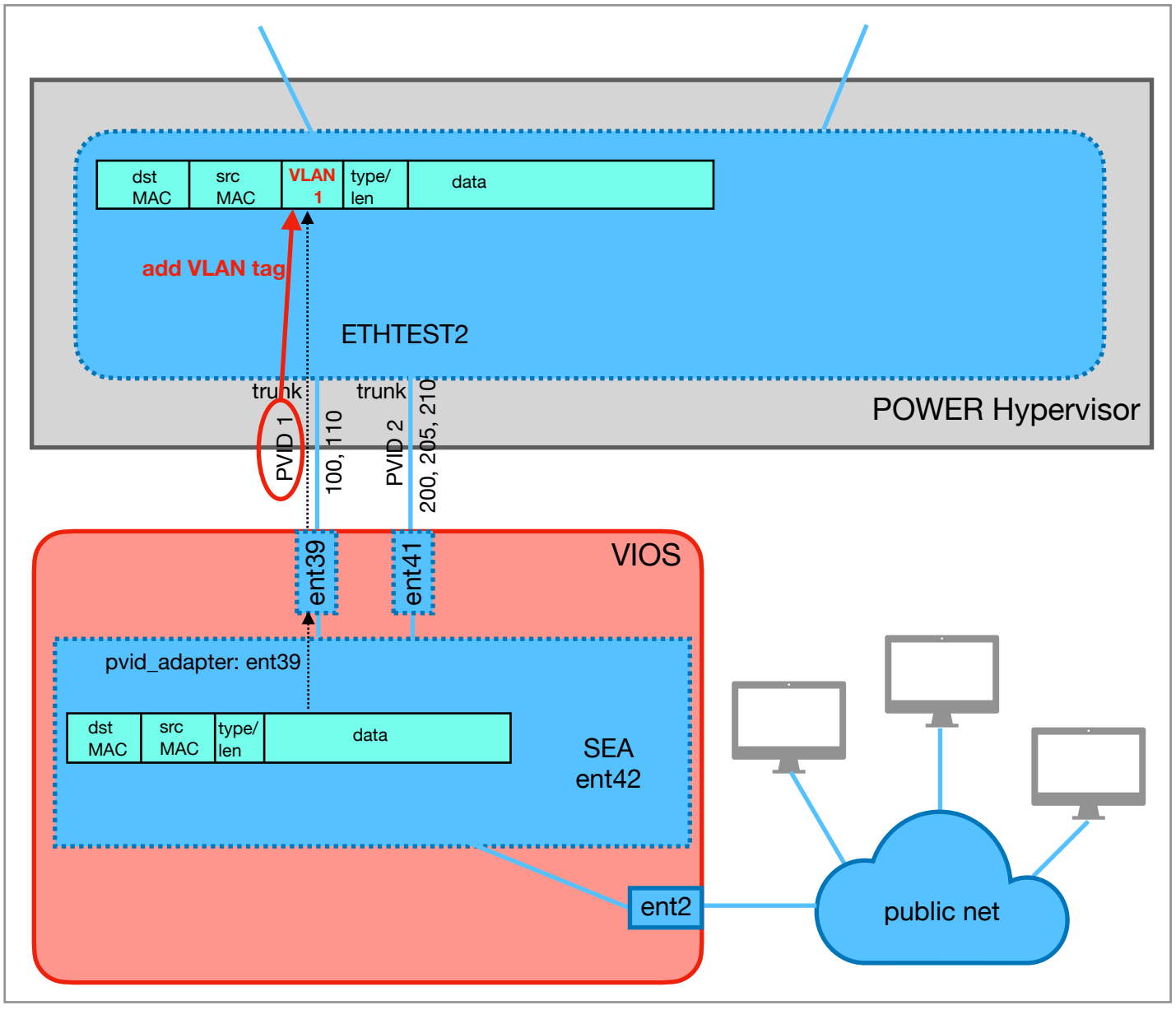


Bild 8.8c: Das Attribut *pvid_adapter* des SEA *ent42* bestimmt an welchen Trunking-Adapter *untagged* Frames gehen, hier an *ent39* mit der PVID *1*. Das Frame bekommt die PVID *1* von *ent39* als VLAN-ID.

4. Der virtuelle Switch *ETHTEST2* leitet den Ethernet Frame an *LPAR1* weiter, da der Frame zum VLAN *1* gehört und der virtuelle Adapter *ent0* von *LPAR1* diese VLAN-ID als PVID konfiguriert hat. (Natürlich muß auch die Ziel MAC-Adresse mit der MAC-Adresse von *ent0 LPAR1* übereinstimmen, aber davon gehen wir aus.) Der VLAN-Header wird bei der Weiterleitung entfernt.

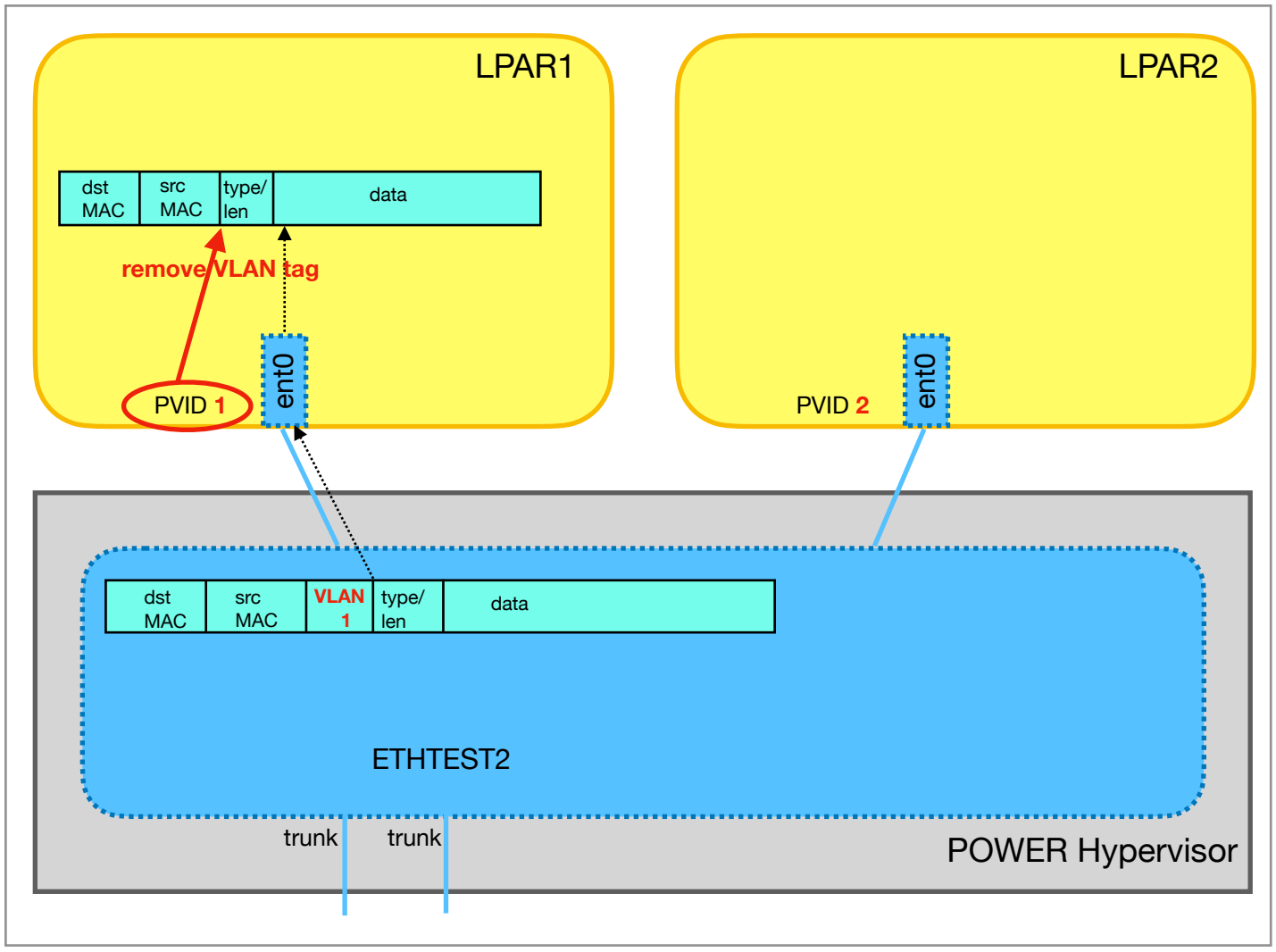


Bild 8.8d: Der Ethernet Frame wird an das Ziel, *LPAR1*, zugestellt. Da die VLAN-ID *1* als Port-VLAN-ID des virtuellen Ethernet Adapters *ent0* konfiguriert ist, wird der VLAN-Header entfernt.

Hinweis: Es ist nicht möglich *LPAR2* (VLAN *2*) von einem externen System aus zu erreichen. Jedes von extern kommende Ethernet Frame ohne VLAN-Header wird vom Shared Ethernet Adapter *ent42* immer an den Trunking-Adapter *ent39* weitergeleitet, da dieser über das Attribut *pvid_adapter* konfiguriert ist. Damit werden externe, nicht getaggte, Frames immer dem VLAN *1* zugeordnet, da das die PVID des Trunking-Adapters *ent39* ist.

Best Practice ist für die PVIDs der Trunking-Adapter nicht benutzte VLAN-IDs zu verwenden. Alle VLANs, die von Client-LPARs verwendet werden, sollten als zusätzliche VLAN-IDs auf den Trunking-Adaptern konfiguriert sein.

8.5.3. Einige SEA Attribute

Jeder SEA besitzt eine Reihe von Attributen, die beim Anlegen gesetzt werden können, oder auch teilweise nachträglich noch geändert werden. Die Attribute können mittels „*vios help mksea*“ aufgelistet werden:

```
$ vios help mksea
USAGE:
   vios [-h <hmc>] [-m <ms>] mksea [-M [-a]] [-v] <vios> <target_device> <ent> ...
[<attribute> ...]

…

ATTRIBUTES

   accounting : enable per-client accounting
   adapter_reset : reset real adapter on HA takeover
```

```
   ctl_chan : control channel adapter for SEA failover
   fb_delay : delay before failback occurs (seconds)
   ha_mode : high availability mode
   jumbo_frames : enable Gigabit Ethernet Jumbo frames
   large_receive : enable receive TCP segment aggregation
   largesend : enable hardware transmit TCP resegmentation
   link_time : time in seconds required for the link to be declared
               healthy after a status change
   lldpsvc : enable IEEE 802.1qbg services
   noauto_failback : disable auto failback
   pvid : PVID to use for the SEA device
   pvid_adapter : default virtual adapter to use for untagged packets
   real_adapter : physical adapter associated with the SEA
   virt_adapters : list of virtual adapters associated with the SEA

…
```

Weitere Attribute können der IBM Dokumentation entnommen werden, oder aber auch ganz einfach mittels „*vios lsattr*“ für einen SEA angezeigt werden.

Im Zeitalter von 100 Gb/s Ethernet ist das Attribut *jumbo_frames* sicher eines der wichtigsten Attribut. In der Regel sollten Jumbo-Frames verwendet werden, indem beim Anlegen eines SEAs mit „*vios mksea*“ das Argument *jumbo_frames=yes* angegeben wird.

Viele physikalische Netzwerk-Karten bieten Hardware-Unterstützung für einige TCP-Operationen, die mit *large_receive=yes* und *largesend=1* aktiviert werden können.

Eine ausführliche Beschreibung dieser Attribute findet man in diversen Dokumenten von IBM.

8.5.4. Konfigurieren einer IP-Adresse auf dem SEA

Auf einem Shared Ethernet Adapter kann, wie auf jedem anderen Ethernet Adapter auch, eine IP-Adresse konfiguriert werden. Die IP-Adresse kann auf dem zugehörigen *en*-Interface (*Standard Ethernet*) oder *et*-Interface (IEEE802.3) konfiguriert werden. Es werden *untagged* Ethernet Frames verwendet, womit die IP-Adresse intern dem VLAN mit der Port-VLAN-ID des Default-Trunking Adapters (*pvid_adapter*) gehört.

Eine IP-Adresse kann mit dem Kommando „*vios mktcpip*“ (*make TCP/IP*) konfiguriert werden. Es muß neben dem Virtual-I/O-Server mindestens das zu konfigurierende Interface und die IP-Adresse angegeben werden. Optional können auch Netzmaske und Gateway für das Interface angegeben werden:

```
$ vios mktcpip ms05-vio1 en42 2.222.1.15 255.255.255.0
$
```

Die konfigurierten IP-Adressen lassen sich mit dem Kommando „*vios lstcpip*“ (*list TCP/IP*) und der Option „*-i*“ (*interfaces*) anzeigen:

```
$ vios lstcpip -i ms05-vio1
NAME  ADDRESS        NETMASK        STATE   MAC
en7   -              -              detach  "a1:b5:8a:07:03:01"
en3   173.15.201.48  255.255.255.0  up      "a1:b5:8a:07:5a:63"
et3   -              -              detach  "a1:b5:8a:07:5a:63"
et7   -              -              detach  "a1:b5:8a:07:03:01"
en15  -              -              detach  "a1:b5:8a:07:03:02"
```

```
et15  -              -              detach  "a1:b5:8a:07:03:02"
en42  2.222.1.15     255.255.255.0  up      "a1:b5:8a:07:b3:05"
et42  -              -              detach  "a1:b5:8a:07:b3:05"
$
```

Wird die IP-Adresse nicht mehr benötigt, kann das Interface mit Hilfe des Kommandos „*vios rmtcpip*“ (*remove TCP/IP*) dekonfiguriert werden:

```
$ vios rmtcpip ms05-vio1 en42
$
```

Jeder SEA besitzt das Attribut *pvid*, welches angibt zu welchem VLAN *untagged* Frames auf dem Shared Ethernet Adapter selbst gehören. Das Attribut muß immer auf die PVID des Default-Trunking Adapters gesetzt sein (*pvid_adapter*)!

Soll auf einem Shared Ethernet Adapter eine IP-Adresse für eines der getaggten VLANs konfiguriert werden, ist das auch möglich. Hierzu muß zunächst auf dem Shared Ethernet Adapter ein VLAN Ethernet Adapter mit dem gewünschten VLAN erzeugt werden. Das kann mit dem Kommando „*vios mkvlan*“ (*make VLAN*) gemacht werden:

```
$ vios mkvlan ms05-vio1 ent42 100
adapter ent36 created
$
```

Neben dem Virtual-I/O-Server muß der SEA und das gewünschte VLAN angegeben werden. Es wird ein neuer Ethernet Adapter als Kind-Gerät des SEAs erzeugt. Alle Ethernet Frames dieses Adapters werden mit der VLAN-ID *100* getaggt. Die IP-Adresse kann dann wie oben auf dem zugehörigen en-Interface *en36* mit dem Kommando „*vios mktcpip*“ konfiguriert werden:

```
$ vios mktcpip ms05-vio1 en36 2.100.9.12 255.255.255.0
$
```

Dekonfigurieren der IP-Adresse kann bei Bedarf wie oben mit „*vios rmtcpip*“ durchgeführt werden:

```
$ vios rmtcpip ms05-vio1 en36
$
```

Zum Entfernen des VLAN Ethernet Adapters muß „*vios rmdev*“ verwendet werden, ein „*vios rmvlan*“ gibt es nicht:

```
$ vios rmdev -d ms05-vio1 ent36
$
```

Die Option „*-d*“ sorgt dafür das auch die ODM-Einträge entfernt werden.

8.5.5. Hinzufügen und Wegnehmen von VLANs (nicht HA)

Der oben angelegte Shared Ethernet Adapter auf *ms05-vio1* unterstützt aktuell die folgenden VLANs:

```
$ vios lssea -V ms05-vio1 ent42
```

```
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent42  -      ent2   real     -         -     -       -         -     -
ent42  -      ent41  virtual  ETHTEST2  VEB   True    1         2     200,205,210
ent42  -      ent39  virtual  ETHTEST2  VEB   True    1         1     100,110
$
```

Wird ein weiteres VLAN benötigt, z.B. VLAN *120*, dann kann das neue VLAN einem der bestehenden Trunking-Adapter als zusätzliches VLAN hinzugefügt werden, z.B. dem Trunking Adapter *ent39*. Das Kommando „*lpar addvlan*“ (*add VLAN*), mit dem einem virtuellen Ethernet Adapter weitere VLANs hinzugefügt werden können, erwartet eine Slot-Nummer für den zu ändernden virtuellen Ethernet Adapter. Die Slot-Nummer von *ent39* lässt sich mit dem Kommando „*vios lsdev*“ leicht anzeigen:

```
$ vios lsdev ms05-vio1 ent39
NAME   STATUS     PHYSLOC                       PARENT  DESCRIPTION
ent39  Available  U8205.E6C.05E4E5Q-V1-C61-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Der Trunking-Adapter hat die Slot-Nummer *61*! Damit kann das VLAN *120* nun dem virtuellen Ethernet Adapter in Slot *61* wie folgt hinzugefügt werden:

```
$ lpar addvlan ms05-vio1 61 120
$
```

Das zusätzliche VLAN ist sofort verfügbar (falls auf den externen Netzwerk-Switches der Zugriff auf das VLAN *120* für den physikalischen Ethernet Port des SEAs erlaubt ist):

```
$ vios lssea -V ms05-vio1 ent42
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent42  -      ent2   real     -         -     -       -         -     -
ent42  -      ent41  virtual  ETHTEST2  VEB   True    1         2     200,205,210
ent42  -      ent39  virtual  ETHTEST2  VEB   True    1         1     100,110,120
$
```

Sollen gleich mehrere zusätzliche VLANs hinzugefügt werden, werden diese einfach mit Komma getrennt beim Kommando „*lpar addvlan*“ aufgelistet:

```
$ lpar addvlan ms05-vio1 61 125,126,127
$
```

Damit können 3 weitere VLANs über den SEA genutzt werden:

```
$ vios lssea -V ms05-vio1 ent42
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent42  -      ent2   real     -         -     -       -         -     -
ent42  -      ent41  virtual  ETHTEST2  VEB   True    1         2     200,205,210
ent42  -      ent39  virtual  ETHTEST2  VEB   True    1         1     100,110,120,125,126,127
$
```

Das Wegnehmen von nicht mehr benötigten VLANs geschieht analog mit dem Kommando „*lpar rmvlan*“ (*remove VLAN*). Auch hier können mehrere VLANs angegeben werden:

```
$ lpar rmvlan ms05-vio1 61 120,125,127
$
```

Zu beachten ist dabei das Client-LPARs die diese VLANs gerade nutzen sofort die Verbindung zu externen Systemen verlieren. Es erfolgt keine Überprüfung bei dem Kommando ob die VLANs eventuell noch in Benutzung sind!

Hinweis: Werden SEAs mit *ha_mode=auto* oder *ha_mode=sharing* verwendet, ist das Hinzufügen und Wegnehmen von VLANs etwas komplizierter und wird später diskutiert.

8.5.6. Hinzufügen und Wegnehmen von Trunking Adaptern (nicht HA)

Ein Shared Ethernet Adapter kann auch um weitere Trunking-Adapter erweitert werden. Es können maximal 16 Trunking-Adapter von einem Shared Ethernet Adapter verwendet werden.

Wir erweitern den Shared Ethernet Adapter *ent42*, der schon in den Beispielen oben verwendet wurde:

```
$ vios lssea -V ms05-vio1 ent42
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent42  -      ent2   real     -         -     -       -         -     -
ent42  -      ent41  virtual  ETHTEST2  VEB   True    1         2     200,205,210
ent42  -      ent39  virtual  ETHTEST2  VEB   True    1         1     100,110
$
```

Zunächst muß ein weiterer virtueller Trunking-Adapter angelegt werden. Der Slot *63* ist bisher noch ungenutzt („*lpar lsvslot*“). Die schon von *ent42* verwendeten Trunking-Adapter haben die Trunking-Priorität *1*, sind an den virtuellen Switch *ETHTEST2* angebunden und sind IEEE802.1Q kompatibel. Als PVID verwenden wir die VLAN-ID *3*, die vom SEA bisher nicht genutzt wird. Als zusätzliche VLANs konfigurieren wir die VLAN-IDs *1020* und *1090*:

```
$ lpar addeth -i -t 1 -s ETHTEST2 ms05-vio1 63 3 1020,1090
$
```

Die von einem SEA verwendeten Trunking-Adapter sind im Attribut *virt_adapters* hinterlegt:

```
$ vios lsattr ms05-vio1 ent42 virt_adapters
value
ent39,ent41
$
```

Der SEA *ent42* verwendet die Adapter *ent39* und *ent41*. Um einen Trunking-Adapter hinzuzufügen, muß der zusätzliche Adapter einfach dem Attribut *virt_adapters* hinzugefügt werden. Der notwendige Gerätename des neuen Trunking-Adapters lässt sich einfach mittels „*vios lssea*“ und der Option „*-c*“ (*candidates*) herausfinden:

```
$ vios lssea -c ms05-vio1
NAME   STATUS     PHYSLOC                       PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C7-T4    pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent43  Available  U8205.E6C.05E4E5Q-V1-C63-T1   vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Der neu angelegte Trunking-Adapter in Slot *63* ist offensichtlich *ent43*. Das Attribut *virt_adapters* lässt sich mit dem Kommando „*vios chdev*“ ändern:

```
$ vios chdev ms05-vio1 ent42 virt_adapters=ent39,ent41,ent43
$
```

Die VLANs des neuen Trunking-Adapters sind sofort über den SEA verfügbar (falls die externe Switch-Konfiguration angepasst wurde):

```
$ vios lssea -V ms05-vio1 ent42
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent42  -      ent2   real     -         -     -       -         -     -
ent42  -      ent41  virtual  ETHTEST2  VEB   True    1         2     200,205,210
ent42  -      ent39  virtual  ETHTEST2  VEB   True    1         1     100,110
ent42  -      ent43  virtual  ETHTEST2  VEB   True    1         3     1020,1090
$
```

Der SEA *ent42* besitzt nun 3 Trunking Adapter, mit den VLANs *100*, *110*, *200*, *205*, *210*, *1020* und *1090*.

Soll ein Trunking-Adapter nicht mehr vom SEA verwendet werden, kann er in der gleichen Weise weggenommen werden. Der entsprechende Adapter muß lediglich im Attribut *virt_adapters* des SEAs entfernt werden:

```
$ vios chdev ms05-vio1 ent42 virt_adapters=ent39,ent41
$
```

Die VLANs (*1020* und *1090*) des weggenommenen Trunking-Adapters werden unmittelbar nicht mehr vom SEA unterstützt!

Als letztes kann der Trunking-Adapter dann gelöscht werden, wenn er nicht mehr benötigt werden sollte. Das Löschen geschieht mittels „*lpar rmeth*“:

```
$ lpar rmeth ms05-vio1 63
$
```

8.5.7. Löschen eines Shared Ethernet Adapters (nicht HA)

Bevor ein Shared Ethernet Adapter gelöscht wird, sollte sichergestellt werden das keine Client-LPAR den SEA noch verwendet!

Der zu löschende SEA ist der bisher in allen Beispielen verwendete *ent42*:

```
$ vios lssea -V ms05-vio1 ent42
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent42  -      ent2   real     -         -     -       -         -     -
ent42  -      ent41  virtual  ETHTEST2  VEB   True    1         2     200,205,210
ent42  -      ent39  virtual  ETHTEST2  VEB   True    1         1     100,110
$
```

Das Löschen des SEAs kann mit dem Kommando „*vios rmsea*“ (*remove SEA*) durchgeführt werden:

```
$ vios rmsea ms05-vio1 ent42
$
```

Das Kommando wirkt sich sofort aus. Sollten Client-LPARs doch noch den SEA *ent42* verwendet haben, dann sind diese jetzt von externen Systemen abgeschnitten, da Ethernet Frames nicht mehr in ein externes Netzwerk weitergeleitet werden. Andere LPARs des gleichen Managed Systems sind weiterhin erreichbar (außer der virtuelle Switch verwendet den VEPA-Mode).

```
$ vios lssea ms05-vio1
                                          TIMES   TIMES    TIMES    BRIDGE
NAME   HA_MODE  PRIORITY  STATE        PRIMARY  BACKUP  FLIPFLOP  MODE
ent33  Sharing  1         PRIMARY_SH   1        1       0         Partial
ent34  Sharing  1         PRIMARY_SH   1        1       0         Partial
ent39  -        -         -            -        -       -         -
$
```

Der SEA *ent42* taucht nicht mehr in der Liste der SEAs auf.

Allerdings sind die virtuellen Trunking-Adapter noch vorhanden:

```
$ vios lssea -c ms05-vio1
NAME   STATUS     PHYSLOC                       PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C7-T4    pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent2   Available  U78AA.001.VYRGU0Q-P1-C7-T3    pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent39  Available  U8205.E6C.05E4E5Q-V1-C61-T1   vio0    Virtual I/O Ethernet Adapter (l-lan)
ent41  Available  U8205.E6C.05E4E5Q-V1-C62-T1   vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Der vom SEA *ent42* verwendete physikalische Adapter *ent2* steht wieder zur Verfügung, genauso wie die beiden Trunking-Adapter *ent39* und *ent41*. Werden die Trunking-Adapter nicht mehr benötigt, z.B. um einen neuen SEA anzulegen, können diese mit dem Kommando „*lpar rmeth*“ entfernt werden:

```
$ lpar rmeth ms05-vio1 61
$
$ lpar rmeth ms05-vio1 62
$
```

Ein Blick auf potentielle Kandidaten für einen SEA zeigt, das die beiden ehemaligen Trunking-Adapter nicht mehr vorhanden sind:

```
$ vios lssea -c ms05-vio1
NAME   STATUS     PHYSLOC                       PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C7-T4    pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent2   Available  U78AA.001.VYRGU0Q-P1-C7-T3    pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
$
```

8.5.8. HA-SEA mit Failover

Der Vorteil der bisher gezeigten Netzwerk-Virtualisierung mit Shared Ethernet ist die extrem einfache Konfiguration. Nach Anlegen von Trunking-Adaptern auf einem Virtual-I/O-Server und Erzeugen des Shared Ethernet Adapters mit „*vios mksea*“ können Client-LPARs den Shared Ethernet Adapter über einfach anzulegende Virtual Ethernet Adapter nutzen. Ein großer Nachteil ist allerdings das die bisher gezeigten Shared Ethernet keine Redundanz besitzen. Fällt der Virtual-I/O-Server mit dem Shared Ethernet Adapter aus, oder hat der physikalische Ethernet Adapter des SEAs ein Problem, verlieren die Client-LPARs sofort die Anbindung an das externe Netzwerk.

Um dieses Problem zu beheben wurde der HA-Mode von Shared Ethernet Adaptern eingeführt. Dabei wird auf einem zweiten Virtual-I/O-Server ein zweiter Shared Ethernet Adapter mit der gleichen Konfiguration angelegt. Zu einem Zeitpunkt übernimmt immer nur einer der beiden SEAs die Anbindung an das externe Netzwerk, der andere SEA bleibt passiv und wartet auf einen Ausfall des gerade aktiven SEAs. Der aktive SEA ist der sogenannte Primary SEA, der passive SEA ist der sogenannte Backup SEA.

Bild 8.9 zeigt eine hochverfügbare Shared Ethernet Adapter Konfiguration mit einem Primary SEA auf dem linken Virtual-I/O-Server und einem Backup SEA auf dem rechten Virtual-I/O-Server. Die Trunking-Adapter auf beiden SEAs stimmen in ihrer VLAN-Konfiguration überein. Beide besitzen 2 Trunking-Adapter mit den PVIDs *1* und *2*, sowie den zusätzlichen VLAN-IDs *100* und *1001*, sowie *350*, *399* und *512*. Die beiden Trunking-Adapter des linken Virtual-I/O-Servers haben die *trunk_priority 1*, die beiden Trunking-Adapter des rechten Virtual-I/O-Servers die *trunk_priority 2*. Alle Trunking-Adapter eines Shared Ethernet Adapters müssen die gleiche Priorität haben! Je niedriger der Wert der *trunk_priority*, desto höher ist die Priorität (Werte zwischen *1* und *15* sind erlaubt). Das wichtigste SEA Attribut für eine HA Konfiguration ist das Attribut *ha_mode* mit den beiden möglichen Werten *auto* und *standby* für eine HA-Failover Konfiguration wir im Bild 8.9 gezeigt.

Hinweis: Die Gerätenamen der einzelnen Adapter müssen auf den beiden Virtual-I/O-Servern nicht übereinstimmen. Es macht die Administration etwas einfacher wenn die Namen auf beiden Seiten gleich sind, das ist aber nicht notwendig und lässt sich auch nicht immer erreichen.

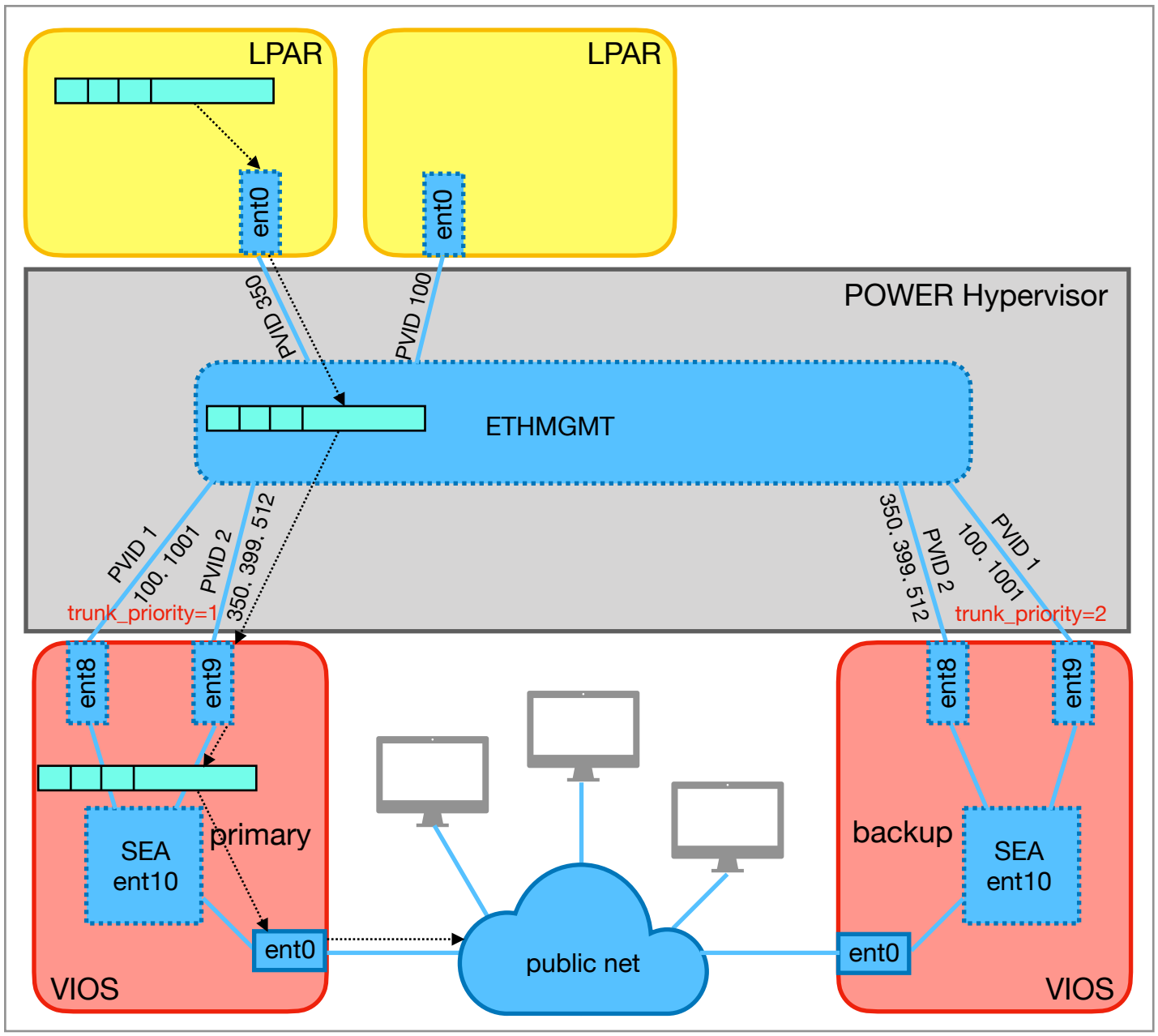


Bild 8.9: High-Availability Konfiguration mit Primary und Backup SEA.

Da der linke SEA die höhere Trunking Priorität hat (niedriger Wert von *trunk_priority*), übernimmt der linke SEA das Weiterleiten von Ethernet Frames an das externe Netzwerk. Sollte der linke Virtual-I/O-Server ausfallen, der Link des physikalischen Adapters auf Down gehen, oder die Netzwerk-Karte ausfallen, dann über nimmt der rechte Virtual-I/O-Server das Weiterleiten von Ethernet Frames und wird zum neuen Primary SEA, wie in Bild 8.10 gezeigt. Für die Client-LPARs ist dies transparent und es gehen in der Regel nur einige wenige Ethernet Frames verloren. Da die meisten Applikationen TCP als unterliegendes Transport-Protokoll verwenden, werden einige Datenpakete einfach erneut übertragen.

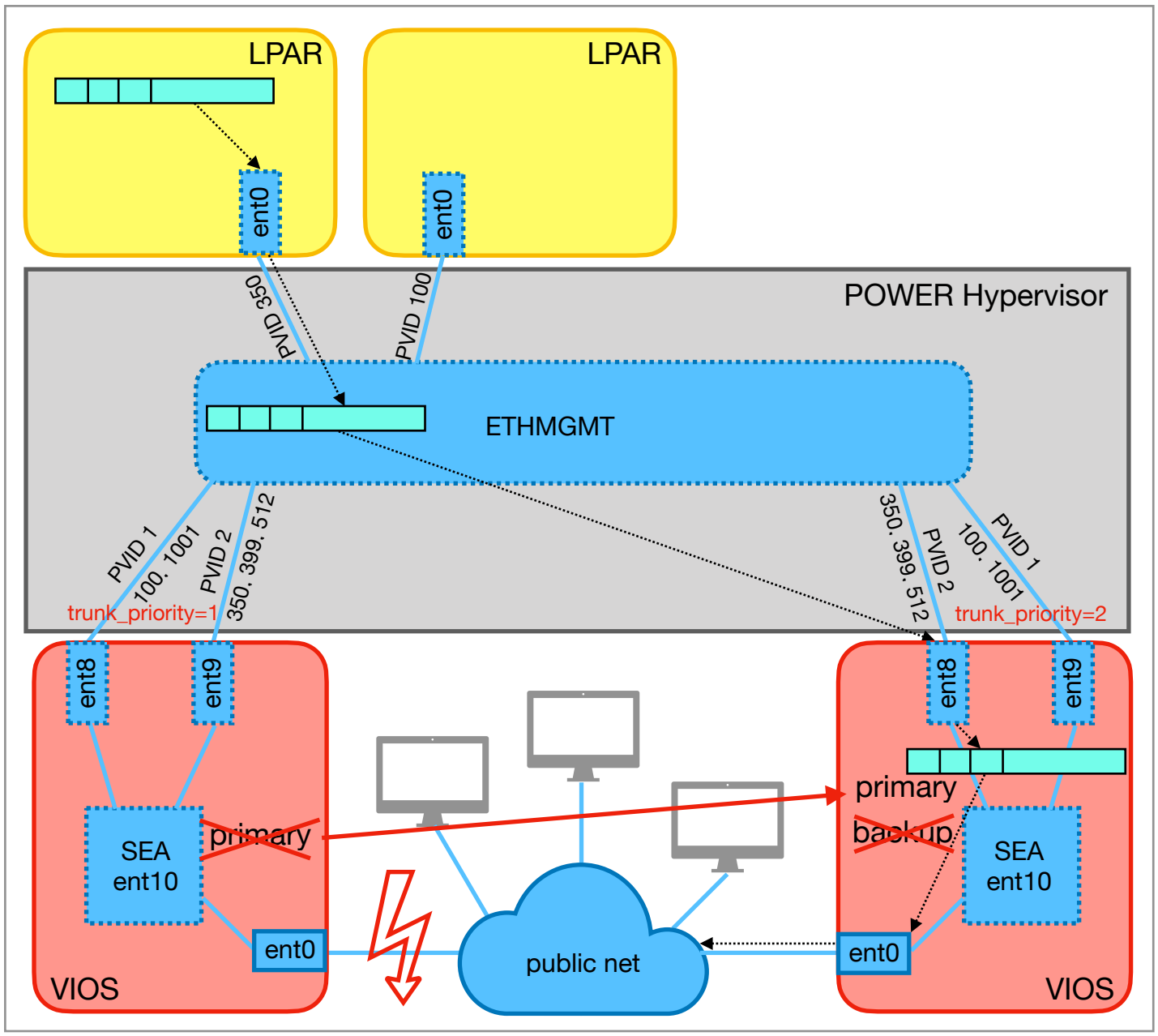


Bild 8.10: Ausfallszenario bei einer HA SEA-Konfiguration.

Damit jeder der beiden SEA immer den Status des anderen SEA kennt, müssen die beiden SEAs miteinander kommunizieren können. Dabei werden sogenannte Heartbeats untereinander ausgetauscht, die dem jeweils anderen SEA den aktuellen Zustand signalisieren. Im einfachsten Fall wird dazu einfach das VLAN *4095* über einen der Trunking-Adapter verwendet. Das VLAN *4095* ist reserviert und kann nicht durch den Administrator konfiguriert werden. Nachteil dieser einfachen Lösung:

- Ist nicht auf allen Managed Systems unterstützt.

- Kann nicht verwendet werden wenn der virtuelle Switch den VEPA-Modus benutzt.

Als Alternative kann ein separater Control-Channel konfiguriert werden. Das ist ein normaler virtueller Ethernet Adapter (kein Trunking-Adapter), der zusätzlich beim Erzeugen eines Shared Ethernet Adapter über das Attribut *ctl_chan* angegeben werden kann. Ein separater Control-Channel wird auf allen Managed Systems unterstützt.

Trunking-Adapter und Control-Channel müssen nicht an den gleichen internen virtuellen Switch angebunden werden. Soll der VEPA-Mode verwendet werden, dann müssen Trunking-Adapter und Control-Channel sogar an verschiedene Switches angebunden werden. Der virtuelle Switch an den ein Control-Channel angebunden ist, muß den VEB-Mode verwenden!

Wir empfehlen für Control-Channel einen eigenen virtuellen Switch zu verwenden, z.B. *ETHCTRL*:

```
$ ms addvswitch ms05 ETHCTRL
$
```

8.5.9. Erzeugen von SEAs mit HA Mode (HA-SEA)

Im nachfolgenden Beispiel sollen 2 SEAs mit Failover angelegt werden. Es sollen jeweils 2 Trunking Adapter verwendet werden mit den PVIDs *1* und *2*, sowie den zusätzlichen VLANs *11*, *12*, *13* und *14*, *15* mit einem eigenen neuen virtuellen Switch *ETHTEST3*. Für den Austausch der Heartbeats zwischen den beiden SEAs sollen Control-Channel über einen eigenen virtuellen Switch *ETHCTRL* verwendet werden.

Wir fügen zunächst die beiden virtuellen Switches *ETHCTRL* für die Control-Channel und *ETHTEST3* für die Trunking-Adapter hinzu:

```
$ ms addvswitch ms05 ETHCTRL
$ ms addvswitch ms05 ETHTEST3
$
```

Beide Virtual-I/O-Server haben freie virtuelle Slots ab Slot-Nummer *70*, welche wir für den Control-Channel und die Trunking-Adapter verwenden. Für den Control-Channel verwenden wir auf beiden Virtual-I/O-Servern den Slot *70* und die PVID *1*:

```
$ lpar addeth -s ETHCTRL ms05-vio1 70 1
$ lpar addeth -s ETHCTRL ms05-vio2 70 1
$
```

Für die beiden Trunking-Adapter verwenden wir jeweils die Slots 71 und 72 mit den PVIDs 1 bzw 2 und den gerade angelegten virtuellen Switch ETHTEST3. Die Trunking-Adapter von *ms05-vio1* bekommen die Trunking-Priorität *1* (höchste) und für die Trunking-Adapter von *ms05-vio2* wird die Trunking-Priorität *2* vergeben:

```
$ lpar addeth -i -t 1 -s ETHTEST3 ms05-vio1 71 1 11,12,13
$ lpar addeth -i -t 1 -s ETHTEST3 ms05-vio1 72 2 14,15
$ lpar addeth -i -t 2 -s ETHTEST3 ms05-vio2 71 1 11,12,13
$ lpar addeth -i -t 2 -s ETHTEST3 ms05-vio2 72 2 14,15
$
```

Ob man zuerst den SEA auf *ms05-vio1* oder *ms05-vio2* anlegt spielt im Prinzip keine Rolle. Wir legen zunächst den SEA mit der höheren Priorität auf *ms05-vio1* an. Dazu lassen wir uns wieder die Kandidaten für einen SEA mit dem Kommando „*vios lssea -c*“ anzeigen:

```
$ vios lssea -c ms05-vio1
NAME   STATUS     PHYSLOC                        PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C7-T4   pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent2   Available  U78AA.001.VYRGU0Q-P1-C7-T3   pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent39  Available  U8205.E6C.05E4E5Q-V1-C70-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent41  Available  U8205.E6C.05E4E5Q-V1-C71-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent42  Available  U8205.E6C.05E4E5Q-V1-C72-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Beim Anlegen eines HA-SEAs muß zwingend das Attribut *ha_mode* mit einem der Werte *auto* oder *standby* (oder aber *sharing* für Load-Sharing, siehe später) angegeben werden. Wird das Attribut nicht angegeben, ist der Default-Wert *disabled*, und es kann nur ein SEA angelegt werden. Der virtuelle Ethernet Adapter für den Control-Channel wird über das Attribut *ctl_chan* angegeben. Über das zusätzliche Attribut *jumbo_frames* und den Wert *yes* erlauben wir die Benutzung von Jumbo-Frames über den Shared Ethernet Adapter:

```
$ vios mksea ms05-vio1 ent3 ent41 ent42 ha_mode=auto ctl_chan=ent39 jumbo_frames=yes
SEA ent43 created
$
```

Der erste SEA ist angelegt:

```
$ vios lssea ms05-vio1
                                          TIMES   TIMES    TIMES    BRIDGE
NAME   HA_MODE  PRIORITY  STATE        PRIMARY  BACKUP  FLIPFLOP  MODE
ent33  Sharing  1         PRIMARY_SH   1        1       0         Partial
ent34  Sharing  1         PRIMARY_SH   1        1       0         Partial
ent39  -        -         -            -        -       -         -
ent43  Auto     1         PRIMARY      1        0       0         All
$
```

Der aktuelle HA-Mode wird in der Spalte *HA_MODE* angezeigt, für den neu angelegten SEA ist das *Auto*. Die Priorität des SEAs wird ebenfalls angezeigt, genauso wie der aktuelle Zustand des SEAs, hier *PRIMARY* für *ent43*. Der neu angelegte SEA nimmt sofort seine Arbeit auf. Allerdings gibt es noch keine Hochverfügbarkeit, solange nicht der zweite SEA auf dem zweiten Virtual-I/O-Server angelegt worden ist. Daher legen wir als nächstes den zweiten SEA an. Dazu listen wir zunächst wieder die möglichen Kandidaten auf:

```
$ vios lssea -c ms05-vio2
NAME   STATUS     PHYSLOC                        PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C6-T2   pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent4   Available  U78AA.001.VYRGU0Q-P1-C6-T3   pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent5   Available  U78AA.001.VYRGU0Q-P1-C6-T4   pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent38  Available  U8205.E6C.05E4E5Q-V2-C70-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent39  Available  U8205.E6C.05E4E5Q-V2-C71-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent40  Available  U8205.E6C.05E4E5Q-V2-C72-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Als physikalischen Ethernet Adapter verwenden wir den Port *4* der 4-Port Gigabit Ethernet Karte, *ent5*. Control-Channel ist *ent38* und die beiden Trunking-Adapter sind *ent39* und *ent40*:

```
$ vios mksea ms05-vio2 ent5 ent39 ent40 ha_mode=auto ctl_chan=ent38 jumbo_frames=yes
SEA ent41 created
$
```

Mit dem Anlegen des zweiten SEAs haben wir jetzt Hochverfügbarkeit. Der gerade angelegte SEA *ent41* fungiert zunächst als Backup SEA:

```
$ vios lssea ms05-vio2
                                          TIMES   TIMES    TIMES    BRIDGE
NAME   HA_MODE  PRIORITY  STATE        PRIMARY  BACKUP  FLIPFLOP  MODE
ent33  Sharing  1         PRIMARY_SH   1        1       0         Partial
```

```
ent34   Sharing   1          PRIMARY_SH   1          1          0          Partial
ent41   Auto      1          BACKUP       0          1          0          All
$
```

Die Verteilung der VLANs lässt sich wieder mit Hilfe des Kommandos „*vios lssea -V*“ anzeigen, hier für den Primary SEA:

```
$ vios lssea -V ms05-vio1 ent43
SEA     LNAGG   NAME    TYPE      VSWITCH    MODE   ACTIVE   PRIORITY   PVID   VLAN_TAG_IDS
ent43   -       ent3    real      -          -      -        -          -      -
ent43   -       ent42   virtual   ETHTEST3   VEB    True     1          2      14,15
ent43   -       ent41   virtual   ETHTEST3   VEB    True     1          1      11,12,13
ent43   -       ent39   control   ETHCTRL    -      -        -          6      None
$
```

Die resultierende Konfiguration auf den beiden Virtual-I/O-Servern ist in Bild 8.11 dargestellt. Der virtuelle Switch für die Control-Channel der beiden SEAs besitzt keine Trunking-Adapter und damit auch keine Anbindung an ein externes Netzwerk.

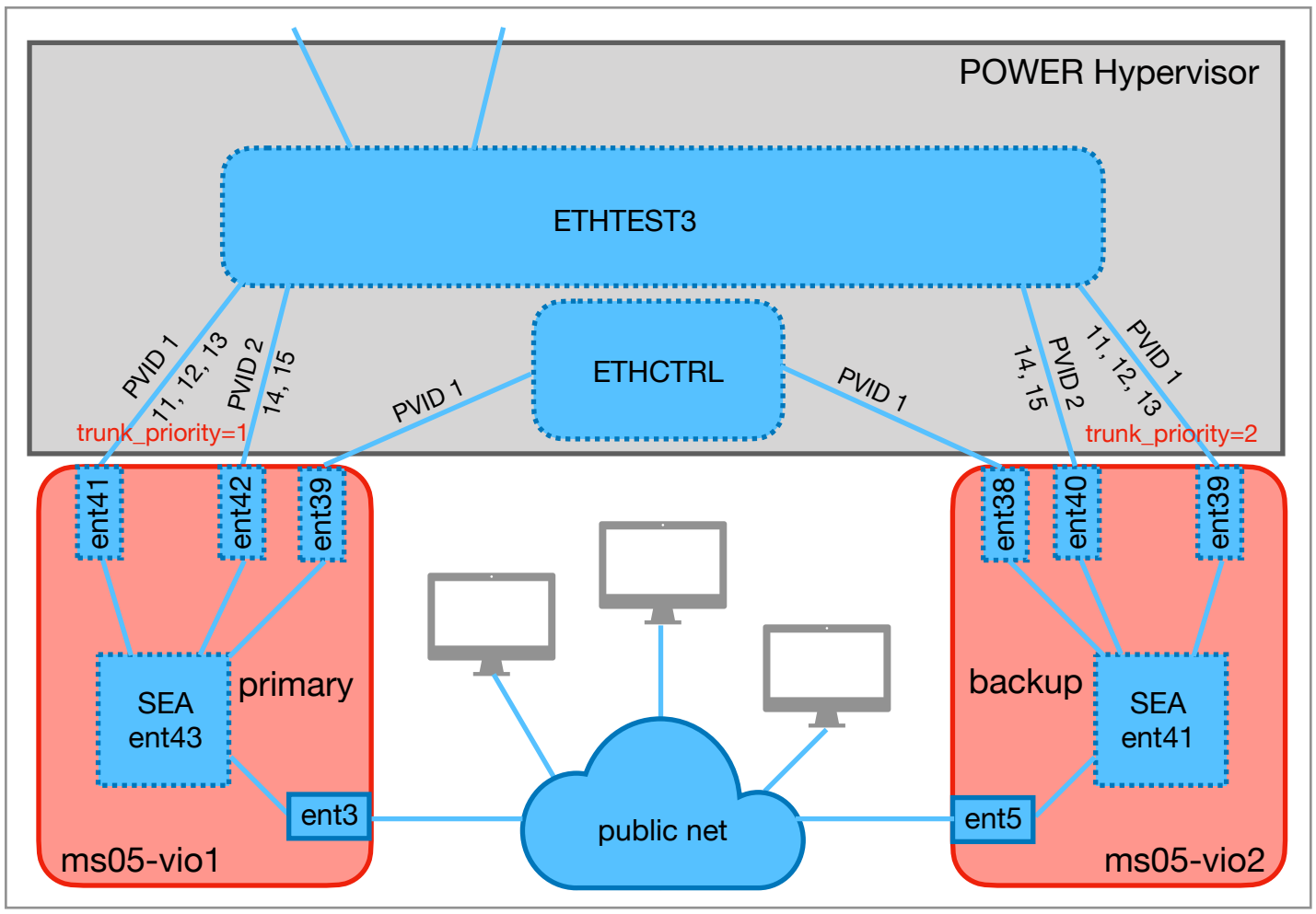


Bild 8.11: High-Availability SEAs mit Control-Channel.

8.5.10. HA-Modes *auto* und *standby*

Über das Attribut *ha_mode* kann konfiguriert werden, welcher der beiden Shared Ethernet Adapter der Primary (aktive) sein soll. Für HA-Failover kann für jeden der beiden SEAs einer der beiden Werte *auto* oder *standby* konfiguriert werden. Der Wert von *ha_mode* entscheidet zusammen mit der Trunking-Priorität (*trunk_priority*) darüber welcher SEA Primary und welcher SEA Backup ist. Alle möglichen Kombinationen sind in der folgenden Tabelle dargestellt:

ha_mode (SEA1)	*ha_mode* (SEA2)	SEA1 *trunk_priority=1*	SEA2 *trunk_priority=2*
standby	*standby*	Primary	Backup
standby	*auto*	Backup	Primary
auto	*standby*	Primary	Backup
auto	*auto*	Primary	Backup

Haben beide SEAs den gleichen Wert für *ha_mode*, entscheidet die Trunking-Priorität (*trunk_priority*), welcher SEA Primary ist. Der SEA mit der höheren Priorität (niedrigerer Wert) ist der Primary SEA. Hat ein SEA den Wert *standby* und der andere SEA den Wert *auto*, dann ist der SEA mit dem Wert *auto* für *ha_mode* der Primary SEA.

Ein manueller Failover lässt sich durch Ändern des Attributs *ha_mode* durchführen. Ausgangszustand ist: beide SEAs sind mit dem Wert *auto* konfiguriert, daher ist *ms05-vio1* mit der höheren Trunking-Priorität der Primary SEA (*ent43*):

```
$ vios lssea ms05-vio1 ent43
                                        TIMES    TIMES    TIMES    BRIDGE
NAME    HA_MODE   PRIORITY  STATE       PRIMARY  BACKUP  FLIPFLOP  MODE
ent43   Auto      1         PRIMARY     1        0       0         All
$
```

Wird der *ha_mode* des Primary SEAs auf *standby* geändert, wird der bisherige Backup SEA der neue Primary. Das Attribut lässt sich über das Kommando „*vios chdev*“ ändern:

```
$ vios chdev ms05-vio1 ent43 ha_mode=standby
$
```

Der bisherige Primary SEA auf *ms05-vio1* wird sofort zum Backup SEA, der bisherige Backup SEA auf *ms05-vio2* zum Primary SEA:

```
$ vios lssea ms05-vio1 ent43
                                        TIMES    TIMES   TIMES  BRIDGE
NAME    HA_MODE   PRIORITY  STATE       PRIMARY  BACKUP  FLIPFLOP  MODE
ent43   Standby   1         BACKUP      1        1       0         None
$
$ vios lssea ms05-vio2 ent41
                                       TIMES    TIMES   TIMES  BRIDGE
NAME    HA_MODE   PRIORITY  STATE      PRIMARY  BACKUP  FLIPFLOP  MODE
ent41   Auto      2         PRIMARY    1        1       0         All
$
```

Soll der SEA auf *ms05-vio1* wieder Primary werden, muß das Attribut *ha_mode* wieder auf *auto* gesetzt werden:

```
$ vios chdev ms05-vio1 ent43 ha_mode=auto
$
```

8.5.11. Hinzufügen und Wegnehmen von VLANs (HA-SEA)

Das Hinzufügen und Wegnehmen von zusätzlichen VLAN-IDs ist ähnlich einfach, wie im Falle eines einfachen Shared Ethernet Adapters. Zu beachten ist lediglich das die VLAN-IDs immer beiden Shared Ethernet Adaptern hinzugefügt oder weggenommen werden müssen.

Zur Demonstration soll den beiden Trunking-Adaptern mit den VLANs *14* und *15* noch das VLAN *16* hinzugefügt werden. Wir starten mit dem Primary SEA auf *ms05-vio1*. Der Ausgabe von „*vios lssea -V*“ lässt sich der Gerätename des Trunking-Adapters entnehmen:

```
$ vios lssea -V ms05-vio1 ent43
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent43  -      ent3   real     -         -     -       -         -     -
ent43  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15
ent43  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent43  -      ent39  control  ETHCTRL   -     -       -         6     None
$
```

Der Trunking-Adapter mit den VLANs *14* und *15* ist *ent42*. Als nächstes benötigt man die Slot-Nummer des Trunking-Adapters:

```
$ vios lsdev ms05-vio1 ent42
NAME   STATUS     PHYSLOC                       PARENT  DESCRIPTION
ent42  Available  U8205.E6C.05E4E5Q-V1-C72-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Der Trunking-Adapter hat die virtuelle Slot-Nummer *72* (*C72*). Mit Hilfe der gefundenen Slot-Nummer und dem Kommando „*lpar addvlan*“ kann das VLAN *16* dem Trunking-Adapter hinzugefügt werden:

```
$ lpar addvlan ms05-vio1 72 16
$
```

Das neue VLAN *16* steht sofort zur Verfügung. Sollten Client-LPARs das VLAN *16* verwenden, können sie sofort System außerhalb des Managed Systems erreichen:

```
$ vios lssea -V ms05-vio1 ent43
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent43  -      ent3   real     -         -     -       -         -     -
ent43  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15,16
ent43  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent43  -      ent39  control  ETHCTRL   -     -       -         6     None
$
```

Die Ausgabe von „*vios lssea -V*“ oben zeigt das das VLAN *16* aktiv ist (Spalte *ACTIVE* ist *True*). Sollte allerdings ein Failover auftreten (automatisch, aufgrund eines Fehlers, oder manuell ausgelöst), geht die Verfügbarkeit des VLAN *16* wieder verloren, da das VLAN bisher nur auf dem Primary SEA auf Virtual-I/O-Server *ms05-vio1* konfiguriert ist:

```
$ vios lssea -V ms05-vio2 ent41
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent41  -      ent5   real     -         -     -       -         -     -
ent41  -      ent40  virtual  ETHTEST3  VEB   False   2         2     14,15
ent41  -      ent39  virtual  ETHTEST3  VEB   False   2         1     11,12,13
```

```
ent41  -      ent38  control  ETHCTRL   -     -       -         6     None
$
```

Die Ausgabe zeigt das der Trunking-Adapter *ent40* des Backup SEA *ent41* auf *ms05-vio2* das VLAN *16* nicht unterstützt! Auch in diesem Fall hatte der Trunking-Adapter die Slot-Nummer *72* und das VLAN *16* kann mit dem Kommando „*lpar addvlan*“ hinzugefügt werden:

```
$ lpar addvlan ms05-vio2 72 16
$
```

Eine erneute Auflistung der VLANs des Backup SEAs zeigt, dass der SEA jetzt auch das VLAN *16* im Failover-Fall unterstützen würde:

```
$ vios lssea -V ms05-vio2 ent41
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent41  -      ent5   real     -         -     -       -         -     -
ent41  -      ent40  virtual  ETHTEST3  VEB   False   2         2     14,15,16
ent41  -      ent39  virtual  ETHTEST3  VEB   False   2         1     11,12,13
ent41  -      ent38  control  ETHCTRL   -     -       -         6     None
$
```

Das Wegnehmen von einem oder mehreren VLANs funktioniert analog mit dem Kommando „*lpar rmvlan*“ und ist im folgenden nicht gezeigt. Es ist wieder darauf zu achten das die VLANs auf den Trunking-Adaptern beider SEAs weggenommen werden. Ansonsten ist das Vorgehen wie bei einem einfachen SEA, was weiter oben schon vorgestellt wurde.

8.5.12. Hinzufügen und Wegnehmen von Trunking Adaptern (HA-SEA)

Auch beim Hinzufügen von Trunking-Adaptern ist darauf zu achten, dass bei beiden SEAs ein Trunking-Adapter hinzugefügt werden muß. Die Trunking-Adapter müssen mit den gleichen VLANs angelegt werden. Bei der Trunking-Priorität ist darauf zu achten die gleiche Priorität wie bei den übrigen Trunking-Adaptern eines SEAs zu verwenden. Ein Trunking-Adapter mit einer verschiedenen Trunking-Priorität lässt sich einem bestehenden SEA nicht hinzufügen.

Wir starten indem wir in den Slots *73* der beiden Virtual-I/O-Server jeweils einen Trunking-Adapter mit der PVID *3* und den zusätzlichen VLANs *17*, *18* und *19* anlegen:

```
$ lpar addeth -i -t 1 -s ETHTEST3 ms05-vio1 73 3 17,18,19
$ lpar addeth -i -t 2 -s ETHTEST3 ms05-vio2 73 3 17,18,19
$
```

Wir starten beim Hinzufügen des neuen Trunking-Adapters wieder beim Primary SEA *ent43* auf *ms05-vio1*. Der gerade angelegt Trunking-Adapter kann wieder mittels „*vios lssea -c*“ leicht angezeigt werden:

```
$ vios lssea -c ms05-vio1
NAME   STATUS     PHYSLOC                        PARENT  DESCRIPTION
ent2   Available  U78AA.001.VYRGU0Q-P1-C7-T3   pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
```

```
ent44  Available  U8205.E6C.05E4E5Q-V1-C73-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Der neue Trunking-Adapter ist *ent44* und muß in die Liste der virtuellen Adapter (Attribut *virt_adapters*) des SEAs *ent43* aufgenommen werden, diese umfasst aktuell die Adapter:

```
$ vios lsattr ms05-vio1 ent43 virt_adapters
value
ent41,ent42
$
```

Der neue Trunking-Adapter kann mit Hilfe von „*vios chdev*“ in die Liste aufgenommen werden:

```
$ vios chdev ms05-vio1 ent43 virt_adapters=ent41,ent42,ent44
$
```

Der neue Trunking-Adapter ist sofort aktiv und die zusätzlichen VLANs können sofort verwendet werden:

```
$ vios lssea -V ms05-vio1 ent43
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent43  -      ent3   real     -         -     -       -         -     -
ent43  -      ent44  virtual  ETHTEST3  VEB   True    1         3     17,18,19
ent43  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15,16
ent43  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent43  -      ent39  control  ETHCTRL   -     -       -         6     None
$
```

Auf dem aktuellen Backup SEA ist der neu angelegte Trunking-Adapter noch nicht bekannt. Bei einem Failover würde man daher die zusätzlichen VLANs *17*, *18* und *19* wieder verlieren. Daher listen wir auch auf dem zweiten Virtual-I/O-Server wieder die Kandidaten für einen SEA auf, um den Trunking-Adapter zu identifizieren:

```
$ vios lssea -c ms05-vio2
NAME   STATUS     PHYSLOC                       PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C6-T2    pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent4   Available  U78AA.001.VYRGU0Q-P1-C6-T3    pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent42  Available  U8205.E6C.05E4E5Q-V2-C73-T1   vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Der Trunking-Adapter in Slot *73* ist *ent42*. Die Trunking-Adapter von SEA *ent41* auf *ms05-vio2* sind aktuell:

```
$ vios lsattr ms05-vio2 ent41 virt_adapters
value
ent39,ent40
$
```

Auch hier kann der neue Trunking-Adapter einfach durch Erweitern des Attributs *virt_adapters* hinzugefügt werden:

```
$ vios chdev ms05-vio2 ent41 virt_adapters=ent39,ent40,ent42
$
```

Das Wegnehmen eines Trunking-Adapters geht analog und wurde schon für den Fall eines einfachen SEAs gezeigt. Zu beachten ist wieder, das der Trunking-Adapter auf beiden SEAs weggenommen werden muß, da die beiden SEAs sonst unterschiedliche Konfigurationen besitzen.

Falls die weggenommenen Trunking-Adapter nicht mehr benötigt werden, sollten diese mittels „*lpar rmeth*“ entfernt werden.

8.5.13. Löschen von HA-SEAs

Prinzipiell erfolgt das Löschen von HA-SEAs, genauso wie das Löschen eines einfachen SEAs, mit dem Kommando „*vios rmdev*“. Es muß lediglich darauf geachtet werden, das auch beide SEAs gelöscht werden.

Um einen Failover zu vermeiden, sollte man beim Löschen mit dem Backup SEA beginnen.

In unserem Falle ist der Backup SEA *ent41* auf *ms05-vio2*:

```
$ vios rmdev -d ms05-vio2 ent41
$
```

Anschließend kann dann auch der Primary SEA *ent43* auf *ms05-vio1* gelöscht werden:

```
$ vios rmdev -d ms05-vio1 ent43
$
```

Die Client-LPARs verlieren damit sofort den Zugriff auf das externe Netzwerk für die von den gelöschten SEAs unterstützten VLAN-IDs!

8.5.14. HA-SEA mit Load-Sharing

Der große Vorteil des SEA-Failover besteht in der höheren Verfügbarkeit durch die Redundanz (2 SEAs). Allerdings hat die SEA-Failover Konfiguration auch einen Nachteil: es ist immer nur einer der beiden SEAs aktiv (Primary) und leitet den Netzwerk-Verkehr weiter. Damit ist auch immer nur einer der beiden physikalischen Adapter der beiden SEAs aktiv. D.h. nur 50% der theoretisch verfügbaren Bandbreite kann genutzt werden.

Mit Einführung des SEA Load-Sharing hat IBM dies geändert. Die beiden SEAs teilen die VLANs untereinander auf und jeder der beiden SEAs leitet den Netzwerk-Verkehr für seine VLANs weiter. Damit können beide physikalischen Netzwerk-Adapter der SEAs genutzt werden und somit ein höherer Netzwerk-Durchsatz erzielt werden (theoretisch die Summe der Bandbreiten der physikalischen Netzwerk-Adapter).

Bild 8.12 zeigt eine Load-Sharing Konfiguration mit separatem Control-Channel. Beide SEAs besitzen 2 Trunking-Adapter, wobei der linke SEA (*primary_sh*) Netzwerk-Verkehr für den ersten Trunking-Adapter mit den VLANs *11*, *12* und *13* weiterleitet und der rechte SEA (*backup_sh*) Netzwerk-Verkehr für den zweiten Trunking-Adapter mit den VLANs *14* und *15* weiterleitet. Im Bild zu sehen sind auch Ethernet Frames der linken oberen LPAR für das VLAN

12, welche über den aktiven Trunking-Adapter für das VLAN *12* an den SEA links unten weitergeleitet werden und dann schließlich über den physikalischen Netzwerk-Adapter *ent3* an das externe Netzwerk weitergegeben werden. Die zweite LPAR, im VLAN *14*, versendet ihrerseits ein Ethernet Frame für das VLAN *14*, welches über den aktiven Trunking-Adapter für das VLAN *14* an den rechten SEA weitergeleitet wird, welcher dann das Ethernet Frame über den physikalischen Netzwerk-Adapter *ent5* in das gleiche externe Netzwerk weiterleitet.

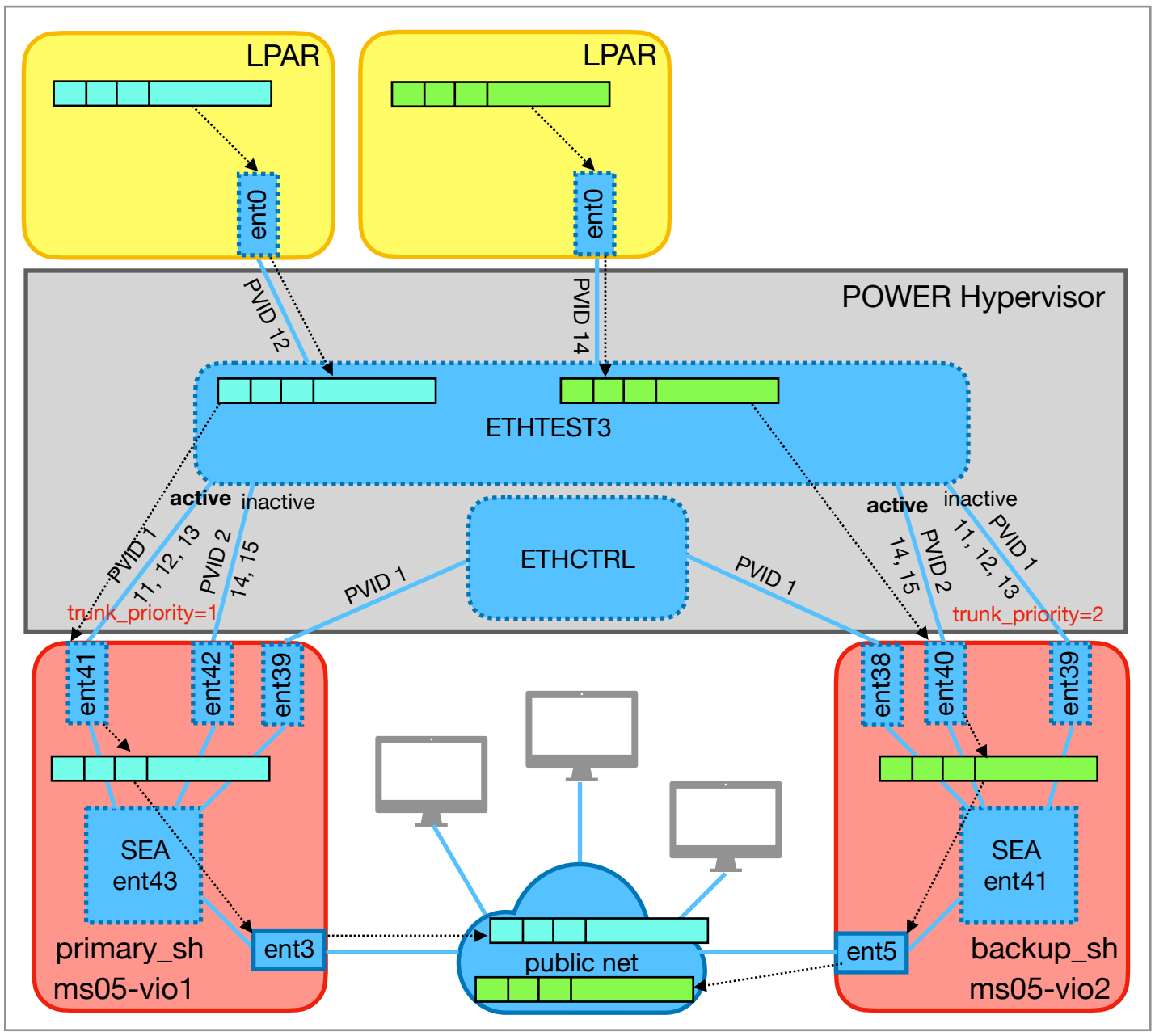


Bild 8.12: SEAs mit Load-Sharing.

Bei beiden SEAs ist eine Hälfte der Trunking-Adapter aktiv und die andere Hälfte inaktiv. Im Fehlerfalle übernimmt dann der verfügbare SEA alle VLANs, indem er alle Trunking-Adapter aktiviert.

Für die Aufteilung der VLANs auf die beiden SEAs wird wie folgt vorgegangen:

- Die erste Hälfte der Trunking-Adapter wird dem SEA mit der höheren Trunking-Priorität (niedrigerer Wert bei *trunk_priority*) zugewiesen, die zweite Hälfte der Trunking-Adapter wird dem SEA mit der niedrigeren Trunking-Priorität zugewiesen. Es zählt die Reihenfolge der Trunking-Adapter des SEAs mit der höheren Trunking-Priorität (Primary SEA).

- Ist die Anzahl der Trunking-Adapter ungerade, dann wird dem SEA mit der höheren Trunking-Priorität (Primary SEA) ein Trunking-Adapter mehr zugewiesen.

Hinweis: Genau wie im Falle von Failover ist die Verwendung eines Control-Channels auf bei Load-Sharing optional.

8.5.15. Erzeugen von SEAs mit Load-Sharing

Um das Erzeugen SEAs mit Load-Sharing zu demonstrieren, wurden die folgenden Trunking-Adapter auf den beiden Virtual-I/O-Servern *ms05-vio1* und *ms05-vio2* angelegt:

```
$ lpar lseth ms05-vio*
LPAR_NAME  SLOT  REQ  STATE  TRUNK  IEEE  QOS   MAC_ADDR      PVID  VSWITCH
ADDL_VLAN_IDS
…
ms05-vio1  70    No   1      -      0     none  A1B58A075A46  1     ETHCTRL    -
ms05-vio1  71    No   1      1      1     none  A1B58A075A47  1     ETHTEST3   11,12,13
ms05-vio1  72    No   1      1      1     none  A1B58A075A48  2     ETHTEST3   14,15
ms05-vio1  73    No   1      1      1     none  A1B58A075A49  3     ETHTEST3   17,18,19
ms05-vio1  74    No   1      1      1     none  A1B58A075A4A  4     ETHTEST3   20,21,22
ms05-vio1  75    No   1      1      1     none  A1B58A075A4B  5     ETHTEST3   23
…
ms05-vio2  70    No   1      -      0     none  A1B58A070346  1     ETHCTRL    -
ms05-vio2  71    No   1      2      1     none  A1B58A070347  1     ETHTEST3   11,12,13
ms05-vio2  72    No   1      2      1     none  A1B58A070348  2     ETHTEST3   14,15
ms05-vio2  73    No   1      2      1     none  A1B58A070349  3     ETHTEST3   17,18,19
ms05-vio2  74    No   1      2      1     none  A1B58A07034A  4     ETHTEST3   20,21,22
ms05-vio2  75    No   1      2      1     none  A1B58A07034B  5     ETHTEST3   23
…
$
```

Wir haben jeweils 5 Trunking-Adapter angelegt (Slot *71* bis *75*) und einen Control-Channel (Slot *70*). Die Trunking-Adapter von Virtual-I/O-Server *ms05-vio1* haben wieder die höhere Trunking-Priorität *1*. Als virtuellen Switch haben wir den schon existierenden virtuellen Switch *ETHTEST3* verwendet.

Wir starten wieder mit dem Primary SEA auf *ms05-vio1* und lassen uns zunächst wieder die Kandidaten für einen SEA anzeigen:

```
$ vios lssea -c ms05-vio1
NAME   STATUS     PHYSLOC                        PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C7-T4     pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent2   Available  U78AA.001.VYRGU0Q-P1-C7-T3     pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent39  Available  U8205.E6C.05E4E5Q-V1-C70-T1    vio0    Virtual I/O Ethernet Adapter (l-lan)
ent41  Available  U8205.E6C.05E4E5Q-V1-C71-T1    vio0    Virtual I/O Ethernet Adapter (l-lan)
ent42  Available  U8205.E6C.05E4E5Q-V1-C72-T1    vio0    Virtual I/O Ethernet Adapter (l-lan)
ent43  Available  U8205.E6C.05E4E5Q-V1-C73-T1    vio0    Virtual I/O Ethernet Adapter (l-lan)
ent44  Available  U8205.E6C.05E4E5Q-V1-C74-T1    vio0    Virtual I/O Ethernet Adapter (l-lan)
ent45  Available  U8205.E6C.05E4E5Q-V1-C75-T1    vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Das Anlegen des SEAs geht wie gehabt mit dem Kommando „*vios mksea*“, wobei für das Attribut *ha_mode* der Wert *sharing* für Load-Sharing angegeben werden muss:

```
$ vios mksea ms05-vio1 ent3 ent41 ent42 ent43 ent44 ent45 ha_mode=auto ctl_chan=ent39
jumbo_frames=yes
```

```
SEA ent46 created
$
```

Der neu angelegte SEA ist dabei noch im Zustand Primary, da der zweite SEA noch nicht angelegt wurde:

```
$ vios lssea ms05-vio1 ent46
                                    TIMES    TIMES   TIMES  BRIDGE
NAME   HA_MODE  PRIORITY  STATE    PRIMARY  BACKUP  FLIPFLOP  MODE
ent46  Sharing  1         PRIMARY  1        0       0         All
$
```

Der Spalte *HA_MODE* kann man entnehmen das der SEA aber für Load-Sharing konfiguriert ist. Interessant ist auch die letzte Spalte *BRIDGE_MODE*: der Wert *All* gibt an das der SEA aktuell für alle Trunking-Adapter, und damit für alle unterstützten VLANs, die Weiterleitung übernimmt. Das bestätigt auch die Ausgabe von „*vios lssea -V*“:

```
$ vios lssea -V ms05-vio1 ent46
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent46  -      ent3   real     -         -     -       -         -     -
ent46  -      ent45  virtual  ETHTEST3  VEB   True    1         5     23
ent46  -      ent44  virtual  ETHTEST3  VEB   True    1         4     20,21,22
ent46  -      ent43  virtual  ETHTEST3  VEB   True    1         3     17,18,19
ent46  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15
ent46  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent46  -      ent39  control  ETHCTRL   -     -       -         1     None
$
```

Als nächstes listen wir auf dem zweiten Virtual-I/O-Server *ms05-vio2* die potentiellen Kandidaten für einen SEA auf:

```
$ vios lssea -c ms05-vio2
NAME   STATUS     PHYSLOC                        PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C6-T2   pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent4   Available  U78AA.001.VYRGU0Q-P1-C6-T3   pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent5   Available  U78AA.001.VYRGU0Q-P1-C6-T4   pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent38  Available  U8205.E6C.05E4E5Q-V2-C70-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent39  Available  U8205.E6C.05E4E5Q-V2-C71-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent40  Available  U8205.E6C.05E4E5Q-V2-C72-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent41  Available  U8205.E6C.05E4E5Q-V2-C73-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent42  Available  U8205.E6C.05E4E5Q-V2-C74-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent43  Available  U8205.E6C.05E4E5Q-V2-C75-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Damit kann der zweite SEA angelegt werden:

```
$ vios mksea ms05-vio2 ent5 ent39 ent40 ent41 ent42 ent43 ha_mode=sharing ctl_chan=ent38
jumbo_frames=yes
SEA ent44 created
$
```

Der Status des neu angelegten SEAs ist *BACKUP_SH*:

```
$ vios lssea ms05-vio2 ent44
                                    TIMES    TIMES   TIMES  BRIDGE
NAME   HA_MODE  PRIORITY  STATE     PRIMARY  BACKUP  FLIPFLOP  MODE
```

```
ent44  Sharing  2         BACKUP_SH  0        1       0         Partial
$
```

Die Endung „*_SH*“ soll dabei andeuten das der Adapter sich aktuell im Load-Sharing-Mode befindet!

Der Status des zuerst angelegten SEAs hat sich geändert, wie die Ausgabe von „*vios lssea*“ zeigt:

```
$ vios lssea ms05-vio1 ent46
                                        TIMES    TIMES   TIMES  BRIDGE
NAME   HA_MODE  PRIORITY  STATE        PRIMARY  BACKUP  FLIPFLOP  MODE
ent46  Sharing  1         PRIMARY_SH   1        0       0         Partial
$
```

Der Zustand (Spalte *STATE*) hat sich von *PRIMARY* (kein Load-Sharing) auf *PRIMARY_SH* (Load-Sharing) geändert. Auch der Wert in der Spalte *BRIDGE_MODE* hat sich geändert und ist jetzt *Partial*. Dies bedeutet das der SEA für einige Trunking-Adapter der aktive SEA ist, für andere Trunking-Adapter nicht. Für welche VLANs ein SEA der aktive SEA ist, kann der Ausgabe von „*vios lssea -V*“ entnommen werden:

```
$ vios lssea -V ms05-vio1 ent46
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent46  -      ent3   real     -         -     -       -         -     -
ent46  -      ent45  virtual  ETHTEST3  VEB   False   1         5     23
ent46  -      ent44  virtual  ETHTEST3  VEB   False   1         4     20,21,22
ent46  -      ent43  virtual  ETHTEST3  VEB   True    1         3     17,18,19
ent46  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15
ent46  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent46  -      ent39  control  ETHCTRL   -     -       -         1     None
$
```

Die Ausgabe zeigt das die Trunking-Adapter *ent41*, *ent42* und *ent43* aktiv sind (Spalte *ACTIVE* ist *True*) und Trunking-Adapter *ent44* und *ent45* inaktiv sind (Spalte *ACTIVE* ist *False*). Da die Anzahl der Trunking-Adapter ungerade ist, bekommt der Primary SEA einen Trunking-Adapter mehr zugewiesen als der Backup SEA. Für den zweiten SEA sind die Rollen dann genau vertauscht:

```
$ vios lssea -V ms05-vio2 ent44
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent44  -      ent5   real     -         -     -       -         -     -
ent44  -      ent43  virtual  ETHTEST3  VEB   True    2         5     23
ent44  -      ent42  virtual  ETHTEST3  VEB   True    2         4     20,21,22
ent44  -      ent41  virtual  ETHTEST3  VEB   False   2         3     17,18,19
ent44  -      ent40  virtual  ETHTEST3  VEB   False   2         2     14,15
ent44  -      ent39  virtual  ETHTEST3  VEB   False   2         1     11,12,13
ent44  -      ent38  control  ETHCTRL   -     -       -         1     None
$
```

8.5.16. Hinzufügen und Wegnehmen von VLANs (Load-Sharing)

Das dynamische Hinzufügen und Wegnehmen von VLANs bei SEAs mit Load-Sharing ist durch IBM unterstützt. D.h. es können prinzipiell jederzeit VLANs hinzugefügt oder weggenommen werden. Allerdings ist es dabei möglich, das es zu einer kurzzeitigen Verzögerung bei der Weiterleitung von Ethernet Frames kommen kann.

IBM empfiehlt vor dem Hinzufügen oder Wegnehmen von VLANs beide SEAs auf Failover zurückzusetzen (*ha_mode=auto*):

```
$ vios chdev ms05-vio1 ent46 ha_mode=auto
$ vios chdev ms05-vio2 ent44 ha_mode=auto
$
```

Dann können VLANs hinzugefügt oder weggenommen werden, beginnend mit dem Primary SEA (hier wird das VLAN *16* dem Trunking-Adapter *ent42* von *ms05-vio1* hinzugefügt):

```
$ lpar addvlan ms05-vio1 72 16
$
```

Das VLAN steht sofort zur Verfügung.

Anschließend werden für den Backup SEA die gleichen VLANs hinzugefügt oder weggenommen (auf *ms05-vio2* ist der Trunking-Adapter *ent40*):

```
$ lpar addvlan ms05-vio2 72 16
$
```

Nachdem alle Änderungen auf beiden SEAs durchgeführt wurden, sollten die VLANs der beiden SEAs noch einmal kontrolliert werden („*vios lssea -V*“):

```
$ vios lssea -V ms05-vio1 ent46
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent46  -      ent3   real     -         -     -       -         -     -
ent46  -      ent45  virtual  ETHTEST3  VEB   True    1         5     23
ent46  -      ent44  virtual  ETHTEST3  VEB   True    1         4     20,21,22
ent46  -      ent43  virtual  ETHTEST3  VEB   True    1         3     17,18,19
ent46  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15,16
ent46  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent46  -      ent39  control  ETHCTRL   -     -       -         1     None
$
$ vios lssea -V ms05-vio2 ent44
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent44  -      ent5   real     -         -     -       -         -     -
ent44  -      ent43  virtual  ETHTEST3  VEB   False   2         5     23
ent44  -      ent42  virtual  ETHTEST3  VEB   False   2         4     20,21,22
ent44  -      ent41  virtual  ETHTEST3  VEB   False   2         3     17,18,19
ent44  -      ent40  virtual  ETHTEST3  VEB   False   2         2     14,15,16
ent44  -      ent39  virtual  ETHTEST3  VEB   False   2         1     11,12,13
ent44  -      ent38  control  ETHCTRL   -     -       -         1     None
$
```

Das hinzugefügte VLAN *16* wird bei beiden SEAs korrekt angezeigt.

Als letztes werden die SEAs wieder für Load-Sharing konfiguriert, indem das Attribut *ha_mode* wieder auf *sharing* gesetzt wird:

```
$ vios chdev ms05-vio1 ent46 ha_mode=sharing
$ vios chdev ms05-vio2 ent44 ha_mode=sharing
$
```

Eine Überprüfung der Verteilung der VLANs zeigt das beide SEAs wieder (für ihre VLANs) aktiv sind:

```
$ vios lssea -V ms05-vio1 ent46
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent46  -      ent3   real     -         -     -       -         -     -
ent46  -      ent45  virtual  ETHTEST3  VEB   False   1         5     23
ent46  -      ent44  virtual  ETHTEST3  VEB   False   1         4     20,21,22
ent46  -      ent43  virtual  ETHTEST3  VEB   True    1         3     17,18,19
ent46  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15,16
ent46  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent46  -      ent39  control  ETHCTRL   -     -       -         1     None
$
```

8.5.17. Hinzufügen und Wegnehmen von Trunking Adaptern (Load-Sharing)

Auch beim Hinzufügen und Wegnehmen von Trunking-Adaptern im Falle von Load-Sharing werden die beiden SEAs zunächst in den Failover HA-Mode (*ha_mode=auto*) gebracht:

```
$ vios chdev ms05-vio1 ent46 ha_mode=auto
$ vios chdev ms05-vio2 ent44 ha_mode=auto
$
```

Um Trunking-Adapter hinzuzufügen, müssen diese zunächst angelegt werden. Das wurde schon mehrfach gezeigt. Wir beschränken uns daher hier das Wegnehmen eines Trunking-Adapters zu zeigen. Wir listen zunächst alle Trunking-Adapter des Primary SEAs auf *ms05-vio1* auf:

```
$ vios lssea -V ms05-vio1 ent46
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent46  -      ent3   real     -         -     -       -         -     -
ent46  -      ent45  virtual  ETHTEST3  VEB   True    1         5     23
ent46  -      ent44  virtual  ETHTEST3  VEB   True    1         4     20,21,22
ent46  -      ent43  virtual  ETHTEST3  VEB   True    1         3     17,18,19
ent46  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15,16
ent46  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent46  -      ent39  control  ETHCTRL   -     -       -         1     None
$
```

Entfernt werden soll hier der Trunking-Adapter für das VLAN *23* (*ent45*). Die Trunking-Adapter sind bekanntermaßen im Attribut *virt_adapters* des SEAs hinterlegt:

```
$ vios lsattr ms05-vio1 ent46 virt_adapters
value
ent41,ent42,ent43,ent44,ent45
$
```

Um den Trunking-Adapter *ent45* wegzunehmen, muß lediglich aus der Liste der Trunking-Adapter der Adapter *ent45* entfernt werden:

```
$ vios chdev ms05-vio1 ent46 virt_adapters=ent41,ent42,ent43,ent44
$
```

Eine Kontrolle der Trunking-Adapter zeigt, dass der Adapter *ent45* erfolgreich entfernt wurde:

```
$ vios lssea -V ms05-vio1 ent46
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent46  -      ent3   real     -         -     -       -         -     -
ent46  -      ent44  virtual  ETHTEST3  VEB   True    1         4     20,21,22
ent46  -      ent43  virtual  ETHTEST3  VEB   True    1         3     17,18,19
ent46  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15,16
ent46  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent46  -      ent39  control  ETHCTRL   -     -       -         1     None
$
```

Auf dem Backup SEA ist der entsprechende Trunking-Adapter für das VLAN *23* noch bekannt:

```
$ vios lssea -V ms05-vio2 ent44
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent44  -      ent5   real     -         -     -       -         -     -
ent44  -      ent43  virtual  ETHTEST3  VEB   False   2         5     23
ent44  -      ent42  virtual  ETHTEST3  VEB   False   2         4     20,21,22
ent44  -      ent41  virtual  ETHTEST3  VEB   False   2         3     17,18,19
ent44  -      ent40  virtual  ETHTEST3  VEB   False   2         2     14,15,16
ent44  -      ent39  virtual  ETHTEST3  VEB   False   2         1     11,12,13
ent44  -      ent38  control  ETHCTRL   -     -       -         1     None
$
```

Der Trunking-Adapter für das VLAN *23* hat hier den Gerätenamen *ent43*. Die Liste der Trunking-Adapter des Backup SEAs enthält noch den Adapter *ent43*:

```
$ vios lsattr ms05-vio2 ent44 virt_adapters
value
ent39,ent40,ent41,ent42,ent43
$
```

Die im Attribut *virt_adapters* hinterlegte Liste lässt sich mit Hilfe von „*vios chdev*“ überschreiben:

```
$ vios chdev ms05-vio2 ent44 virt_adapters=ent39,ent40,ent41,ent42
$
```

Eine kurze Überprüfung zeigt das der Trunking-Adapter nicht mehr verwendet wird:

```
$ vios lssea -V ms05-vio2 ent44
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent44  -      ent5   real     -         -     -       -         -     -
ent44  -      ent42  virtual  ETHTEST3  VEB   False   2         4     20,21,22
ent44  -      ent41  virtual  ETHTEST3  VEB   False   2         3     17,18,19
ent44  -      ent40  virtual  ETHTEST3  VEB   False   2         2     14,15,16
ent44  -      ent39  virtual  ETHTEST3  VEB   False   2         1     11,12,13
ent44  -      ent38  control  ETHCTRL   -     -       -         1     None
$
```

Sollten die beiden weggenommenen Trunking Adapter nicht mehr benötigt werden, sollten diese mit Hilfe von „*lpar rmeth*“ gelöscht werden! Darauf verzichten wir an dieser Stelle.

Als letztes werden die SEAs wieder für Load-Sharing konfiguriert, indem das Attribut *ha_mode* wieder auf *sharing* gesetzt wird:

```
$ vios chdev ms05-vio1 ent46 ha_mode=sharing
$ vios chdev ms05-vio2 ent44 ha_mode=sharing
$
```

Eine Überprüfung der Verteilung der VLANs zeigt das beide SEAs wieder (für ihre VLANs) aktiv sind:

```
$ vios lssea -V ms05-vio1 ent46
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent46  -      ent3   real     -         -     -       -         -     -
ent46  -      ent44  virtual  ETHTEST3  VEB   False   1         4     20,21,22
ent46  -      ent43  virtual  ETHTEST3  VEB   False   1         3     17,18,19
ent46  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15,16
ent46  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent46  -      ent39  control  ETHCTRL   -     -       -         1     None
$
```

Die Verteilung der VLANs auf die beiden SEAs hat sich geändert gegenüber der Ausgangssituation. Zu Beginn gab es 5 Trunking-Adapter, wobei die ersten 3 Trunking-Adapter auf dem Primary SEA aktiv waren. Durch das Wegnehmen eines Trunking-Adapters haben beide SEAs nur noch 2 aktive Trunking-Adapter. Der Trunking-Adapter *ent43* mit den VLANs *17*, *18* und *19* war vor dem Wegnehmen des Trunking-Adapters auf dem Primary SEA aktiv. Nach dem Wegnehmen des Trunking-Adapters ist dieser nun auf dem Primary SEA inaktiv.

Hinweis: Durch das Hinzufügen oder Wegnehmen von Trunking-Adaptern kann sich die Verteilung der VLANs ändern!

8.5.18. Ändern des HA-Modes zwischen Failover und Load-Sharing

Es ist möglich zur Laufzeit zwischen den 3 HA-Modes standby, auto und sharing zu wechseln. Dies wurde schon oben ausgenutzt beim Hinzufügen und Wegnehmen von VLANs oder Trunking-Adaptern.

Die nachfolgende Tabelle zeigt die verschiedenen Möglichkeiten für *ha_mode* auf den beiden SEAs und den resultierenden Zustand der beiden SEAs:

ha_mode (SEA1)	*ha_mode* (SEA2)	SEA1 *trunk_priority=1*	SEA2 *trunk_priority=2*
standby	*standby*	Primary	Backup
standby	*auto*	Backup	Primary
standby	*sharing*	Backup	Primary
auto	*standby*	Primary	Backup
auto	*auto*	Primary	Backup
auto	*sharing*	Primary	Backup
sharing	*standby*	Primary	Backup
sharing	*auto*	Primary	Backup
sharing	*sharing*	Primary_SH	Backup_SH

Der Tabelle kann man entnehmen das Load-Sharing nur für den Fall aktiv ist, wenn beide SEAs das Attribut *ha_mode* auf *sharing* gesetzt haben. In allen anderen Fällen ist Failover aktiv!

Gewechselt werden kann der HA-Mode durch Ändern des Attributs *ha_mode* mit dem Kommando „*vios chdev*“, z.B.:

```
$ vios chdev ms05-vio1 ent44 ha_mode=sharing
$
```

8.5.19. Löschen von SEAs (Load-Sharing)

Beim Löschen von SEAs mit Load-Sharing spielt die Reihenfolge in der die beiden SEAs gelöscht werden keine Rolle. In der Regel werden SEAs nur gelöscht, wenn sie aktuell nicht mehr in Verwendung sind. Nachdem der erste der beiden SEAs gelöscht wurde, wird der verbleibende SEA automatisch zum Primary SEA und wird für alle unterstützten VLANs der aktive SEA. D.h. für einen Teil der VLANs gibt es einen Failover.

Um die Kommandos noch einmal zu demonstrieren, starten wir mit dem Löschen des Backup SEAs *ent44* auf dem Virtual-I/O-Server *ms05-vio2*:

```
$ vios rmsea ms05-vio2 ent44
$
```

Der zunächst noch verbleibende SEA *ent46* auf dem Virtual-I/O-Server *ms05-vio1* wird zum Primary ohne Load-Sharing und übernimmt die Weiterleitung sämtlicher VLANs:

```
$ vios lssea -V ms05-vio1 ent46
SEA    LNAGG  NAME   TYPE     VSWITCH   MODE  ACTIVE  PRIORITY  PVID  VLAN_TAG_IDS
ent46  -      ent3   real     -         -     -       -         -     -
ent46  -      ent44  virtual  ETHTEST3  VEB   True    1         4     20,21,22
ent46  -      ent43  virtual  ETHTEST3  VEB   True    1         3     17,18,19
ent46  -      ent42  virtual  ETHTEST3  VEB   True    1         2     14,15,16
ent46  -      ent41  virtual  ETHTEST3  VEB   True    1         1     11,12,13
ent46  -      ent39  control  ETHCTRL   -     -       -         1     None
$
```

Man sieht an der Ausgabe das alle Trunking-Adapter des SEA *ent46* aktiv sind (Spalte *ACTIVE* - Wert *True*). Auch diesen SEA löschen wir mit Hilfe des Kommandos „*vios rmsea*“:

```
$ vios rmsea ms05-vio1 ent46
$
```

Da auch die verwendeten Trunking-Adapter nicht mehr benötigt werden, listen wir diese zunächst auf um die zugehörigen virtuellen Slot-Nummern zu sehen:

```
$ vios lssea -c ms05-vio1
NAME   STATUS     PHYSLOC                        PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C7-T4   pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
```

```
ent2   Available  U78AA.001.VYRGU0Q-P1-C7-T3   pci1    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent39  Available  U8205.E6C.05E4E5Q-V1-C70-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent41  Available  U8205.E6C.05E4E5Q-V1-C71-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent42  Available  U8205.E6C.05E4E5Q-V1-C72-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent43  Available  U8205.E6C.05E4E5Q-V1-C73-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent44  Available  U8205.E6C.05E4E5Q-V1-C74-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent45  Available  U8205.E6C.05E4E5Q-V1-C75-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Mit Kenntnis der Slot-Nummern löschen wir den ehemaligen Control-Channel (Slot *70*) und die Trunking-Adapter (Slot *71-75*):

```
$ lpar rmeth ms05-vio1 70
$ lpar rmeth ms05-vio1 71
$ lpar rmeth ms05-vio1 72
$ lpar rmeth ms05-vio1 73
$ lpar rmeth ms05-vio1 74
$ lpar rmeth ms05-vio1 75
$
```

Das gleiche wird auch auf dem zweiten Virtual-I/O-Server *ms05-vio2* durchgeführt. Wir listen zunächst die Adapter auf:

```
$ vios lssea -c ms05-vio2
NAME   STATUS     PHYSLOC                       PARENT  DESCRIPTION
ent3   Available  U78AA.001.VYRGU0Q-P1-C6-T2   pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent4   Available  U78AA.001.VYRGU0Q-P1-C6-T3   pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent5   Available  U78AA.001.VYRGU0Q-P1-C6-T4   pci3    4-Port Gigabit Ethernet PCI-Express
Adapter (e414571614102004)
ent38  Available  U8205.E6C.05E4E5Q-V2-C70-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent39  Available  U8205.E6C.05E4E5Q-V2-C71-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent40  Available  U8205.E6C.05E4E5Q-V2-C72-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent41  Available  U8205.E6C.05E4E5Q-V2-C73-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent42  Available  U8205.E6C.05E4E5Q-V2-C74-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
ent43  Available  U8205.E6C.05E4E5Q-V2-C75-T1  vio0    Virtual I/O Ethernet Adapter (l-lan)
$
```

Um dann anschließend auch hier den ehemaligen Control-Channel (Slot *70*) und die Trunking-Adapter (Slot *71-75*) zu löschen:

```
$ lpar rmeth ms05-vio2 70
$ lpar rmeth ms05-vio2 71
$ lpar rmeth ms05-vio2 72
$ lpar rmeth ms05-vio2 73
$ lpar rmeth ms05-vio2 74
$ lpar rmeth ms05-vio2 75
$
```

8.6. Storage Pools

Für die schnelle Bereitstellung von Client-LPARs ist die Verwendung von SAN-LUNs mittels NPIV in vielen Fällen nicht geeignet. Die SAN-LUNs müssen auf den externen Storage Systemen zunächst angelegt werden und anschließend muss das Zoning im SAN angepasst werden, damit die neuen SAN-LUNs auch für die WWPNs der Client-LPAR sichtbar sind. Auch die Verwendung von VSCSI für das Mapping der SAN-LUNs auf die Client-LPARs erfordert einigen Aufwand. Jede SAN-LUN wird dabei per VSCSI einem oder mehreren Client-LPARs zugeordnet, was zu einer großen Anzahl von SAN-LUNs auf den Virtual-I/O-Servern führen kann.

Eine Möglichkeit Storage für Client-LPARs schneller bereit zustellen besteht in der Verwendung von Storage Pools auf den Virtual-I/O-Servern. Nachdem ein Storage Pool einmal angelegt ist, kann Storage für Client-LPARs mit nur einem Kommando zur Verfügung gestellt werden. Auf dem Storage Pool werden dabei sogenannte Backing-Devices erzeugt, die per Virtual SCSI den Client-LPARs zugeordnet werden können. Storage für Client-LPAR kann damit per PowerVM von den Virtual-I/O-Servern zur Verfügung gestellt werden. Damit kann z.B. eine Boot-Platte für eine neue Client-LPAR innerhalb von wenigen Sekunden angelegt und sofort benutzt werden.

PowerVM bietet zwei verschiedene Arten von Storage Pools an: lokale Storage Pools und Shared Storage Pools. Ein lokaler Storage Pool, oder auch einfach Storage Pool, wird immer nur von einem Virtual-I/O-Server zur Verfügung gestellt. Jeder Virtual-I/O-Server kann seine eigenen unabhängigen Storage Pools besitzen. Ein Shared Storage Pool hingegen wird von mehreren Virtual-I/O-Servern, die in einem Cluster zusammengefasst sind, zur Verfügung gestellt werden. Der Zugriff auf den Shared Storage Pool ist von jedem der Virtual-I/O-Server der zum Cluster gehört möglich. Shared Storage Pools werden in diesem Kapitel nicht behandelt.

Es gibt zwei Arten von lokalen Storage Pools: Logical Volume Storage Pools und File Storage Pools. Bei einem Logical Volume Storage Pool wird für die Client-LPARs Storage in Form von Logical Volumes zur Verfügung gestellt, beim File Storage Pool in Form von Dateien.

In Bild 8.13 ist ein Logical Volume Storage Pool dargestellt. Der Storage Pool ist in Form einer Volume Group realisiert und bezieht daher seine Storage Kapazität über die zugehörigen Physical Volumes. Um Storage für Client-LPARs bereit zustellen, werden Logical Volumes in dem Storage Pool erzeugt, im Bild die Logical Volumes *bd01*, *bd02* und *bd03*. Die Logical Volumes werden dabei als Backing-Devices bezeichnet, da sie letztlich als Speicherort für die Daten der Client-LPARs dienen. Die Zuordnung eines Backing-Devices zu einer Client-LPAR, genauer einem *vhost*-Adapter welcher eins-zu-eins einem virtuellen SCSI-Adapter einer Client-LPAR zugeordnet ist, erfolgt über ein sogenanntes virtuelles Target Device (*vtscsi0*, *vtscsi1* und *vtscsi2* im Bild). Das virtuelle Target Device ist ein Kind-Gerät eines der *vhost*-Adapter und zeigt über das Attribut *aix_tdev* auf das entsprechende Backing-Device. Beim Mapping wird das virtuelle Target Device unterhalb des *vhost*-Adapters erzeugt.

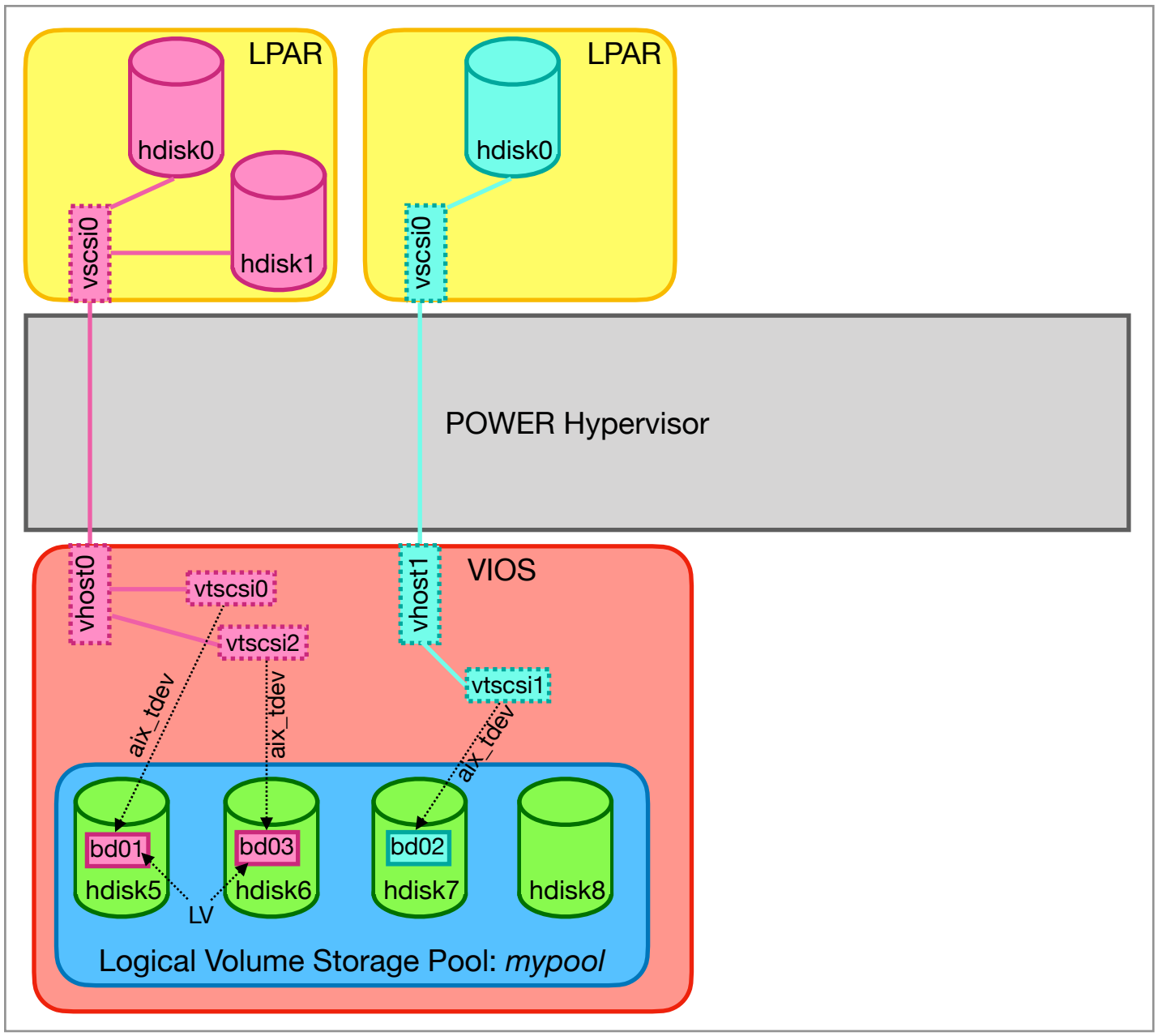


Bild 8.13: Logical Volume Storage Pool

Solange der Storage Pool noch freie Kapazität besitzt, können jederzeit weitere Backing-Devices angelegt und Client-LPARs zugeordnet werden. Die Bereitstellung von Storage für Client-LPAR ist damit sehr flexibel und vor allen Dingen sehr schnell und unterliegt komplett der Kontrolle des PowerVM Administrators.

Neben dem Logical Volume Storage Pool sind auch File Storage Pools unterstützt. In Bild 8.14 ist ein solcher File Storage Pool gezeigt, er ist als Dateisystem implementiert. Das unterliegende Logical Volume liegt in dem Logical Volume Storage Pool *mypool*. Als Name für das Logical Volume wird der Storage Pool Name verwendet, im Bild *filepool*. Das Dateisystem wird unterhalb von */var/vio/storagepools/filepool* gemountet, wobei die letzte Pfad-Komponente gleich dem Storage Pool Namen ist. Als Backing-Devices werden Dateien verwendet, wobei der Dateiname gleich dem Backing-Device Namen ist. Das Mapping wird weiterhin über virtuelle Target Devices realisiert, im Bild *vtscsi3* und *vtscsi4*. Das Attribut *aix_tdev* der virtuellen Target Devices zeigt dabei auf die jeweilige Datei im File Storage Pool.

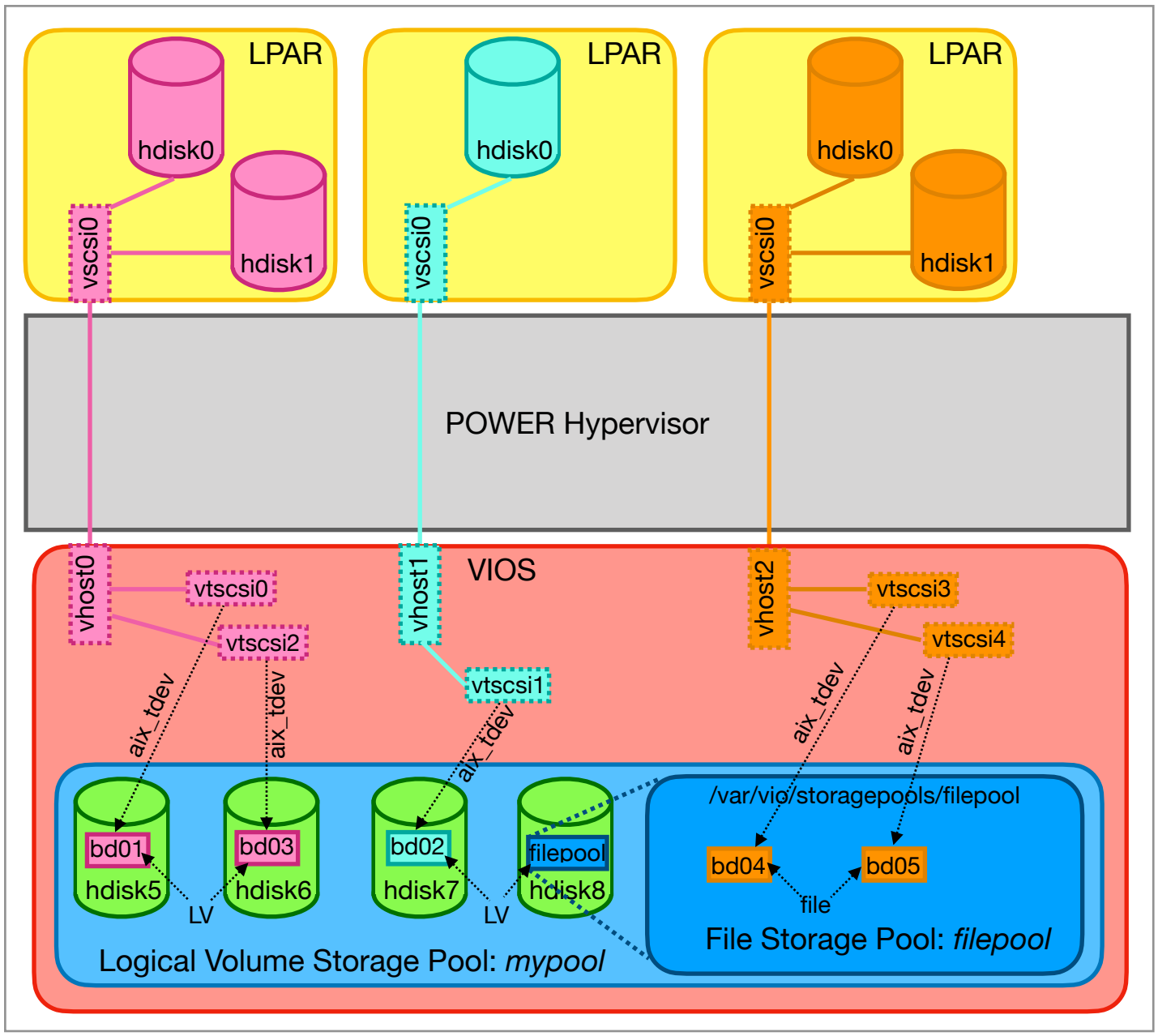


Bild 8.14: File Storage Pool

8.6.1. Erzeugen eines Logical Volume Storage Pools

Welche Storage Pools es aktuell auf einem Virtual-I/O-Server gibt und welchen Typ diese Pools haben, kann mit dem Kommando „*vios lssp*“ (*list storage pools*) angezeigt werden:

```
$ vios lssp ms05-vio1
POOL    TYPE    SIZE        FREE        ALLOC       BDS
rootvg  LVPOOL  558.00 GB   68.00 GB    512.00 MB   0
$
```

Da ein Logical Volume Storage Pool letztlich eine Volume Group auf dem Virtual-I/O-Server ist, gibt es immer mindestens den Storage Pool *rootvg*. Neben dem Typ wird noch die Größe, freier Platz und die aktuelle Anzahl an Backing-Devices des Storage Pools angezeigt (Spalte *BDS*).

Für einen neuen Logical Volume Storage Pool werden freie Physical Volumes benötigt. Welche Physical Volumes ein Virtual-I/O-Server besitzt, kann mittels „*vios lspv*“ (*list physical volumes*) angezeigt werden:

```
$ vios lspv ms05-vio1
PVNAME  PVID              VGNAME  PVSTATE
hdisk0  00ecafe53ce0a4be  rootvg  active
```

```
hdisk1  00ecafe57511a6a3  rootvg  active
hdisk2  none              None    -
hdisk3  none              None    -
hdisk4  none              None    -
hdisk5  none              None    -
hdisk6  none              None    -
$
```

Allerdings sind einige der Physical Volumes eindeutig schon in Verwendung (*hdisk0* und *hdisk1*), bei anderen Physical Volumes kann man dies an der Ausgabe nicht eindeutig erkennen. Mit Hilfe der Option „*-a*“ (*available*) lassen sich nur diejenigen Physical Volumes anzeigen, die für VSCSI verfügbar sind. Physical Volumes die durch Mapping schon in Benutzung sind, lassen sich über die Spalte *VTDS* erkennen:

```
$ vios lspv -a ms05-vio1
PVNAME  PVID  SIZE       VTDS
hdisk2  none   10.00 GB  -
hdisk3  none   10.00 GB  -
hdisk4  none   10.00 GB  vtscsi4
hdisk6  none   10.00 GB  -
$
```

Von den Physical Volumes *hdisk2* bis *hdisk6* ist die *hdisk4* schon durch ein VSCSI-Mapping benutzt (*vtscsi0*) und die *hdisk5* taucht in der Ausgabe hier gar nicht mehr auf (sie ist als Paging-Device für AMS in Verwendung).

Das Physical Volume *hdisk2* ist allerdings noch verfügbar und soll für den ersten eigenen Logical Volume Storage Pool verwendet werden. Ein Storage Pool lässt sich mit dem Kommando „*vios mksp*“ (*make storage pool*) erzeugen. Für einen Logical Volume Storage Pool muss mindestens ein Physical Volume angegeben werden:

```
$ vios mksp ms05-vio1 pool1 hdisk2
hmc01: viosvrcmd -m ms05 -p ms05-vio1 -c \"mksp pool1 hdisk2\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr:
StdErr:
StdErr: Some error messages may contain invalid information
StdErr: for the Virtual I/O Server environment.
StdErr:
StdErr: 0516-358 mkvg: Illegal name; the prefix of the name is reserved.
StdErr: Choose a different name.
StdErr: rc=1
$
```

Der Versuch den Storage Pool mit Namen *pool1* zu erzeugen ist allerdings fehlgeschlagen. Der Präfix „*pool*“ ist reserviert, womit *pool1* als Name nicht verwendet werden kann. Ein neuer Versuch mit dem Namen *testpool* ist dann aber erfolgreich:

```
$ vios mksp ms05-vio1 testpool hdisk2
$
```

Der neue Storage Pool wird dann beim Kommando „*vios lssp*“ auch angezeigt:

```
$ vios lssp ms05-vio1
POOL      TYPE    SIZE       FREE       ALLOC      BDS
rootvg    LVPOOL  558.00 GB   68.00 GB  512.00 MB  0
testpool  LVPOOL    9.93 GB    9.93 GB    8.00 MB  0
```

```
$
```

In der Spalte *ALLOC* wird die *PP Size* der unterliegenden Volume Group angezeigt.

8.6.2. Erzeugen von Backing-Devices

Ein Storage Pool erlaubt die einfache und schnelle Erzeugung von Backing-Devices für Client-LPARs. Solange ein Storage Pool noch freie Kapazitäten besitzt, können jederzeit Backing-Devices erzeugt und Client-LPARs zugeordnet werden (VSCSI-Mapping).

Um ein Backing-Device in einem Storage Pool zu erzeugen, gibt es das Kommando „*vios mkbdsp*“ (*make backing-device in storage pool*). Neben dem Storage Pool muß ein Name für das zu erzeugende Backing-Device angegeben werden, sowie die gewünschte Größe:

```
$ vios mkbdsp ms05-vio1 testpool bd001 1g
$
```

In der Übersicht der Storage Pools des Virtual-I/O-Servers ist die Anzahl der Backing-Devices für den Storage Pool *testpool* von 0 auf 1 gestiegen:

```
$ vios lssp ms05-vio1
POOL      TYPE    SIZE       FREE       ALLOC      BDS
rootvg    LVPOOL  558.00 GB   68.00 GB  512.00 MB  0
testpool  LVPOOL    9.93 GB    8.93 GB    8.00 MB  1
$
```

Natürlich können die Backing-Devices eines Storage-Pools ebenfalls aufgelistet werden, hierzu gibt es das Kommando „*vios lsbdsp*“ (*list backing-devices of storage pool*):

```
$ vios lsbdsp ms05-vio1 testpool
BDNAME  SIZE       VTD   SVSA
bd001     1.00 GB  None  None
$
```

Das gerade erzeugte Backing-Device wird aufgelistet, ist allerdings noch keinem *vhost*-Adapter (VSCSI-Mapping) zugeordnet. Um das Backing-Device einer Client-LPAR zuordnen zu können, benötigt man den zur Client-LPAR zugehörigen *vhost*-Adapter. Mit dem Kommando „*vios lsvscsi*“ lassen sich alle *vhost*-Adapter eines Virtual-I/O-Servers auflisten:

```
$ vios lsvscsi ms05-vio1
SVSA    SLOT  CLIENT              LUNS
vhost0  C25   lpar1(3)            2
vhost1  C28   lpar2(4)            2
vhost2  C31   lpar3(5)            2
$
```

Die Ausgabe zeigt für jeden *vhost*-Adapter (Spalte *SVSA*) die virtuelle Slot-Nummer auf dem Virtual-I/O-Server, sowie die Client-LPAR und die Anzahl der zugeordneten Backing-Devices (Spalte *LUNS*).

Um das gerade erzeugte Backing-Device z.B. der Client-LPAR *lpar1* (*vhost0*) zuzuordnen, kann erneut das Kommando „*vios mkbdsp*“ verwendet werden. Es muß das schon erzeugte Backing-Device, sowie der *vhost*-Adapter angegeben werden, dem das Backing-Device zugeordnet werden soll:

```
$ vios mkbdsp ms05-vio1 testpool bd001 vhost0
$
```

Anstelle des Kommandos „*vios mkbdsp*“ kann auch das schon früher benutzte Kommando „*vios map*“ verwendet werden:

```
$ vios map ms05-vio1 bd001 vhost0
$
```

Eine Überprüfung der *vhost*-Adapter zeigt, das die Anzahl der LUNs (Backing-Devices) für den Adapter *vhost0* auf 3 Backing-Devices angestiegen ist:

```
$ vios lsvscsi ms05-vio1
SVSA    SLOT  CLIENT              LUNS
vhost0  C25   lpar1(3)            3
vhost1  C28   lpar2(4)            2
vhost2  C31   lpar3(5)            2
$
```

Möchte man die LUNs (Backing-Devices) im Detail sehen, kann zusätzlich der *vhost*-Adapter angegeben werden:

```
$ vios lsvscsi ms05-vio1 vhost0
VTD      STATUS     BACKING                              BDPHYSLOC  MIRRORED  LUN
vtopt0   Available  /var/vio/VMLibrary/AIX710506_2.iso  -          N/A       0x8100000000000000
vtscsi0  Available  lpar1_hd00                           -          N/A       0x8200000000000000
vtscsi5  Available  bd001                                -          N/A       0x8300000000000000
$
```

Von der Installation ist noch eine virtuelle AIX Installations-CD eingelegt (*AIX710506_2.iso*) und beim virtuellen Target Gerät *vtscsi0* (Backing-Device *lpar1_hd00*) handelt es sich vermutlich um die Boot-Platte der LPAR *lpar1*. Das neu zugeordnete Backing-Device *bd001* ist als Gerät *vtscsi5* gemappt.

Das das Backing-Device *bd001* nun einer LPAR zugeordnet ist, lässt sich natürlich auch anhand der Ausgabe von „*vios lsbdsp*“ sehen:

```
$ vios lsbdsp ms05-vio1 testpool
BDNAME  SIZE       VTD      SVSA
bd001     1.00 GB  vtscsi5  vhost0
$
```

Die Zuordnung eines Backing-Devices zu einer Client-LPAR ist dieser Ausgabe nicht direkt zu entnehmen. Um die Zuordnung für den Administrator optisch einfacher zu machen, verwenden viele Administratoren die Möglichkeit den Namen des virtuellen Target Devices (default *vtscsiN*) mit anzugeben. Dies ist im Folgenden kurz gezeigt.

Zunächst benötigt man ein weiteres Backing-Device:

```
$ vios mkbdsp ms05-vio1 testpool bd002 1g
$
```

Beim Mapping kann der gewünschte Name für das virtuelle Target Device als letztes Argument angegeben werden:

```
$ vios mkbdsp ms05-vio1 testpool bd002 vhost0 lpar1_hd02
$
```

Oder unter Verwendung von „*vios map*“:

```
$ vios map ms05-vio1 bd002 vhost0 lpar1_hd02
$
```

Der benutzerdefinierte Name für das virtuelle Target Device taucht dann entsprechend in den Ausgaben der Kommandos auf:

```
$ vios lsbdsp ms05-vio1 testpool
BDNAME  SIZE       VTD         SVSA
bd001     1.00 GB  vtscsi5     vhost0
bd002     1.00 GB  lpar1_hd02  vhost0
$
```

Der Namensbestandteil *lpar1* des virtuellen Target Devices deutet nun direkt auf die zugehörige Client-LPAR hin. Dies kann eine große Hilfe sein, wenn sehr viele Backing-Devices in einer Umgebung zum Einsatz kommen.

Damit die gemappten Backing-Device in der Client-LPAR auch sichtbar sind, muß unter AIX ein Lauf des Config-Managers *cfgmgr* erfolgen:

```
lpar1 # lspv
hdisk0          00fbabe641e13592                    rootvg          active
lpar1 #
lpar1 # cfgmgr -l vio0
lpar1 #
lpar1 # lspv
hdisk0          00fbabe641e13592                    rootvg          active
hdisk1          none                                None
hdisk2          none                                None
lpar1 #
```

Die gemappten Backing-Devices tauchen erst nach dem Config-Manager Lauf in der Auflistung als Physical Volumes auf. Der Physical Location Code der Physical Volumes lässt sich mit Hilfe des AIX Kommandos lscfg anzeigen:

```
lpar1 # lscfg -l hdisk*
  hdisk0           U8205.E6C.05E4E5Q-V3-C5-T1-L8200000000000000  Virtual SCSI Disk Drive
  hdisk1           U8205.E6C.05E4E5Q-V3-C5-T1-L8300000000000000  Virtual SCSI Disk Drive
  hdisk2           U8205.E6C.05E4E5Q-V3-C5-T1-L8400000000000000  Virtual SCSI Disk Drive
lpar1 #
```

Über die LUN-ID im Physical Location Code (z.B. *L8400000000000000*) und der Ausgabe der Mappings für die Client-LPAR auf dem Virtual-I/O-Server („*vios lsvscsi*“) lässt sich leicht eine Zuordnung zwischen Backing-Device auf dem Virtual-I/O-Server und dem Physical Volume in der AIX LPAR herstellen:

```
$ vios lsvscsi ms05-vio1 vhost0
VTD         STATUS     BACKING                              BDPHYSLOC  MIRRORED  LUN
vtopt0      Available  /var/vio/VMLibrary/AIX710506_2.iso  -          N/A       0x8100000000000000
```

```
vtscsi0     Available  lpar1_hd00                          -          N/A       0x8200000000000000
vtscsi5     Available  bd001                               -          N/A       0x8300000000000000
lpar1_hd02  Available  bd002                               -          N/A       0x8400000000000000
$
```

Die *hdisk2* der LPAR *lpar1* beispielsweise hat gemäß *lscfg* die LUN-ID *8400000000000000*. Diese LUN-ID findet man in der Ausgabe von „*vios lsvscsi*“ in der letzten Zeile beim Backing-Device *bd002* wieder (die Spalte *LUN* ist hier relevant).

Die neuen Physical Volumes können auf der LPAR beliebig genutzt werden, z.B. zum Erstellen einer Volume Group:

```
lpar1 # mkvg -S hdisk1
0516-1254 mkvg: Changing the PVID in the ODM.
vg00
lpar1 #
```

In den bisherigen Beispielen wurde immer zunächst das Backing-Device erzeugt (Kommando „*vios mkbdsp*“) und dann im zweiten Schritt einem *vhost*-Adapter zugeordnet (Kommando „*vios mkbdsp*“ oder „*vios map*“). Das Kommando „*vios mkbdsp*“ erlaubt es beide Schritte mit einer Operation durchzuführen. Dabei wird neben der Größe des Backing-Devices einfach der gewünschte *vhost*-Adapter angegeben. Optional kann auch noch ein Name für das zu erzeugende virtuelle Target Device angegeben werden:

```
$ vios mkbdsp ms05-vio1 testpool bd003 1g vhost0 lpar1_hd03
$
```

Das geht natürlich noch etwas schneller als mit 2 separaten Kommandos.

8.6.3. Vergrößern von Backing-Devices

Bei der Verwendung von Physical Volumes kommt es immer wieder vor das ein Physical Volume zu voll ist und vergrößert werden soll. Der Vorteil von Storage Pools besteht unter anderem in der einfachen und schnellen Administration der Backing-Devices. Im Folgenden soll gezeigt werden, wie ein Backing-Device vergrößert werden kann.

Ausgangspunkt ist in der Regel ein volles Filesystem oder eine volle Volume Group auf der Client-LPAR. In unserem Falle ist die oben auf LPAR *lpar1* angelegte Volume Group *vg00* voll:

```
lpar1 # lsvg -p vg00
vg00:
PV_NAME           PV STATE          TOTAL PPs   FREE PPs    FREE DISTRIBUTION
hdisk1            active            238         0           00..00..00..00..00
lpar1 #
```

Die LUN-ID des Physical Volumes *hdisk1* kann mittels *lscfg* ermittelt werden:

```
lpar1 # lscfg -l hdisk1
  hdisk1           U8205.E6C.05E4E5Q-V3-C5-T1-L8300000000000000  Virtual SCSI Disk Drive
lpar1 #
```

Die LUN-ID ist *8300000000000000*, der zugehörige *vscsi*-Adapter hat die Slot-Nummer *5* (*V3-C5-T1*). Als nächstes benötigt man den zugehörigen Virtual-I/O-Server, dieser lässt sich über die Ausgabe von „*lpar lsvslot*“ herausfinden:

```
$ lpar lsvslot lpar1
SLOT  REQ  ADAPTER_TYPE   STATE  DATA
0     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
1     Yes  serial/server  1      remote: (any)/any connect_status=unavailable hmc=1
5     No   scsi/client    1      remote: ms05-vio1(1)/25
$
```

Im Slot *5* findet sich wie erwartet ein virtueller SCSI-Client Adapter (es gibt hier nur einen) und der zugehörige Virtual-I/O-Server ist *ms05-vio1* mit *vhost*-Adapter im Slot *25*.

Eine Auflistung der *vhost*-Adapter auf dem Virtual-I/O-Server *ms05-vio1* mit dem Kommando „*vios lsvscsi*“ und der Suche nach der Slot-Nummer *25* ergibt dann den Adapter *vhost0*:

```
$ vios lsvscsi ms05-vio1
SVSA    SLOT  CLIENT              LUNS
vhost0  C25   lpar1(3)            5
vhost1  C28   lpar2(4)            2
vhost2  C31   lpar3(5)            2
$
```

Als nächstes benötigen wir den Namen des Backing-Devices für die gesuchte LUN-ID *8300000000000000*.

```
$ vios lsvscsi ms05-vio1 vhost0
VTD          STATUS     BACKING                              BDPHYSLOC  MIRRORED  LUN
vtopt0       Available  /var/vio/VMLibrary/AIX710506_2.iso   -          N/A       0x8100000000000000
vtscsi0      Available  lpar1_hd00                           -          N/A       0x8200000000000000
vtscsi5      Available  bd001                                -          N/A       0x8300000000000000
lpar1_hd02   Available  bd002                                -          N/A       0x8400000000000000
lpar1_hd03   Available  bd003                                -          N/A       0x8400000000000000
$
```

Das gesuchte Backing-Device ist *bd001*.

Um das Backing-Device *bd001* zu vergrößern, benötigen wir etwas Platz im Storage-Pool:

```
$ vios lssp ms05-vio1
POOL      TYPE    SIZE       FREE       ALLOC      BDS
rootvg    LVPOOL  558.00 GB   68.00 GB  512.00 MB  0
testpool  LVPOOL    9.93 GB    6.93 GB    8.00 MB  3
$
```

Wir vergrößern das Backing-Device *bd001* um 1 GB. Das Kommando zum Vergrößern lautet „*vios chbdsp*“ (*change backing-device in storage pool*):

```
$ vios chbdsp ms05-vio1 testpool bd001 +1g
$
```

Nachdem das Backing-Device vergrößert wurde, muß in der Client-LPAR dem Logical Volume Manager mitgeteilt werden das sich die Größe von Physical Volumes in einer der Volume Groups geändert hat. Das Kommando hierzu lautet „*chvg -g*“:

```
lpar1 # chvg -g vg00
lpar1 #
```

Nach der Vergrößerung gibt es wieder verfügbaren Platz in der Volume Group:

```
lpar1 # lsvg -p vg00
vg00:
PV_NAME           PV STATE          TOTAL PPs   FREE PPs    FREE DISTRIBUTION
hdisk1            active            494         256         00..00..58..99..99
lpar1 #
```

Hinweis: Das Verkleinern eines Backing-Devices ist nicht möglich!

8.6.4. Löschen von Backing-Devices

Wird ein Backing-Device nicht mehr benötigt, kann es mit Hilfe des Kommandos „*vios rmbdsp*“ (*remove backing-device from storage pool*) gelöscht werden:

```
$ vios rmbdsp ms05-vio1 testpool bd001
$
```

Das Backing-Device ist unwiderruflich gelöscht und kann nicht wieder hergestellt werden, außer es gibt ein Backup der Daten. Daher sollte man sich absolut sicher sein, das das Backing-Device zum Einen nicht mehr durch die Client-LPAR in Benutzung ist und zum Anderen die Daten auf dem Backing-Device nicht mehr benötigt werden.

Das Löschen des Backing-Device auf dem Virtual-I/O-Server resultiert nicht in einer Fehlermeldung falls die Client-LPAR das Backing-Device noch benutzt!

Ist das zugehörige Physical Volume in der Client-LPAR noch in Benutzung, schlägt jeder I/O auf das Physical Volume fehl:

```
lpar1 # lsvg -p vg00
0516-062 : Unable to read or write logical volume manager
        record. PV may be permanently corrupted. Run diagnostics
lpar1 #
```

Möchte man zunächst erst einmal nur das Mapping wegnehmen, aber das Backing-Device selbst nicht löschen, sollte das Kommando „*vios unmap*“ verwendet werden:

```
$ vios unmap ms05-vio1 bd001
$
```

Das Backing-Device und die Daten bleiben dann erhalten und nur der Zugriff der Client-LPAR ist nicht mehr möglich. Im Falle eines Falles lässt sich das Backing-Device aber dann wieder zuordnen und der Zugriff auf die Daten ist dann auch wieder möglich.

8.6.5. Vergrößern und Verkleinern eines Logical Volume Storage Pools

Werden einem Storage Pools immer weitere Backing-Devices hinzugefügt und/oder bestehende Backing-Devices vergrößert, geht dem Storage Pool früher oder später die Kapazität aus. Versucht man dann z.B. ein Backing-Device zu vergrößern, kommt es zu Fehlermeldungen:

```
$ vios chbdsp ms05-vio1 testpool bd001 +3g
hmc01: viosvrcmd -m ms05 -p ms05-vio1 -c \"chbdsp -sp testpool -bd bd001 -size 3072M\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr:
StdErr:
StdErr: Some error messages may contain invalid information
StdErr: for the Virtual I/O Server environment.
StdErr:
StdErr:
StdErr:
StdErr: Some error messages may contain invalid information
StdErr: for the Virtual I/O Server environment.
StdErr:
StdErr: 0516-404 allocp: This system cannot fulfill the allocation request.
StdErr: There are not enough free partitions or not enough physical volumes
StdErr: to keep strictness and satisfy allocation requests.  The command
StdErr: should be retried with different allocation characteristics.
StdErr: rc=1
$
```

Schaut man sich die freien Kapazitäten der Storage Pools an, erkennt man das der Storage Pool *testpool* offensichtlich zu wenig freie Kapazität besitzt, um ein Backing-Device um 3 GB zu erweitern:

```
$ vios lssp ms05-vio1
POOL      TYPE    SIZE       FREE       ALLOC      BDS
rootvg    LVPOOL  558.00 GB   68.00 GB  512.00 MB  0
testpool  LVPOOL    9.93 GB    1.93 GB    8.00 MB  3
$
```

Sind noch ungenutzte Physical Volumes auf dem Virtual-I/O-Server verfügbar, dann kann der Logical Volume Storage Pool erweitert werden. In unserem Falle gibt es noch 2 ungenutzte Physical Volumes:

```
$ vios lspv -a ms05-vio1
PVNAME  PVID  SIZE       VTDS
hdisk3  none   10.00 GB  -
hdisk4  none   10.00 GB  vtscsi4
hdisk6  none   10.00 GB  -
$
```

Ein Storage Pool vom Typ Logical Volume kann erweitert werden, indem dem Pool weitere Physical Volumes hinzugefügt werden. Das Kommando hierfür ist „*vios addsppv*“:

```
$ vios addsppv ms05-vio1 testpool hdisk3
$
```

(Sollte das Physical Volume schon einmal in Verwendung gewesen sein und hat noch eine VGDA, dann kann die Option „*-f*“ verwendet werden, um ein Hinzufügen zu erzwingen.)

Die freie Kapazität ist von *1.93* GB auf *11.86* GB gewachsen:

```
$ vios lssp ms05-vio1
POOL      TYPE    SIZE       FREE       ALLOC      BDS
rootvg    LVPOOL  558.00 GB   68.00 GB  512.00 MB  0
testpool  LVPOOL   19.86 GB   11.86 GB    8.00 MB  3
$
```

Der Versuch das Banking-Device *bd001* um 3 GB zu vergrößern ist anschließend auch erfolgreich:

```
$ vios chbdsp ms05-vio1 testpool bd001 +3g
$
```

Sollte der freie Platz in dem Storage Pool wieder zur Neige gehen, können weitere verfügbare Physical Volumes hinzugefügt werden:

```
$ vios addsppv ms05-vio1 testpool hdisk6
$
```

Welche Physical Volumes ein Logical Storage Volume Pool verwendet kann mit „*vios lssp*“ und der Angabe des Storage Pools angezeigt werden:

```
$ vios lssp ms05-vio1 testpool
PVNAME  PVID              SIZE
hdisk2  00ecafe53d2c1269    9.93 GB
hdisk3  00ecafe541b1be91    9.93 GB
hdisk6  00ecafe541b81905    9.93 GB
$
```

Ein Logical Volume Storage Pool kann auch wieder verkleinert werden, falls nicht alle Kapazitäten genutzt werden. Analog dem Hinzufügen von Physical Volumes können Physical Volumes auch wieder weggenommen werden. Wir nehmen das zuletzt hinzugefügte Physical Volume *hdisk6* wieder weg. Das Kommando hierfür ist „*vios rmsppv*“:

```
$ vios rmsppv ms05-vio1 testpool hdisk6
$
```

In vielen Fällen wird das Wegnehmen aber nicht ganz so leicht funktionieren, da sich üblicherweise Teile von einem oder mehreren Backing-Devices auf dem wegzunehmenden Physical Volume befinden, wie das folgende Beispiel zeigt:

```
$ vios rmsppv ms05-vio1 testpool hdisk3
hmc01: viosvrcmd -m ms05 -p ms05-vio1 -c \"chsp -rm -sp testpool hdisk3\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: 0516-914 rmlv: Warning, all data belonging to logical volume
StdErr: bd001 on physical volume hdisk3 will be destroyed.
StdErr: rmlv: Do you wish to continue? y(es) n(o)?
StdErr: Cannot delete a physical volume with allocated
StdErr:    partitions. Use either migratepv to move the partitions or
StdErr:    reducevg with the -rmlv option to delete the partitions.
```

```
StdErr:
StdErr: rc=2
$
```

Das Backing-Device muß entweder auf andere Physical Volumes verschoben werden, falls genügend freie Kapazität vorhanden ist, oder es muß gelöscht werden (dazu darf keine Client-LPAR das Backing-Device in Verwendung haben). Wir stellen zunächst erst einmal fest welche Backing-Devices das Physical Volume *hdisk3* verwenden:

```
$ vios lspv -l ms05-vio1 hdisk3
LVNAME  LPS  PPS  DISTRIBUTION         MOUNT
bd001   137  137  00..137..00..00..00  N/A
$
```

In diesem Falle liegt lediglich das Backing-Device *bd001* teilweise auf dem Physical Volume. Das Backing-Device ist nicht in Verwendung, da wir es nur zu Test-Zwecken angelegt hatten, und kann daher gelöscht werden:

```
$ vios rmbdsp ms05-vio1 testpool bd001
$
```

Anschließend ist auch das Entfernen des Physical Volumes *hdisk3* aus dem Logical Volume Storage Pool testpool erfolgreich:

```
$ vios rmsppv -f ms05-vio1 testpool hdisk3
$
```

8.6.6. Löschen eines Logical Volume Storage Pools

Wird ein Logical Volume Storage Pool nicht mehr benötigt kann er natürlich auch wieder gelöscht werden. Es sollte aber unbedingt vorher sichergestellt werden, dass keine Client-LPAR Backing-Devices des Storage Pools diesen noch benutzen. Außerdem sollte sichergestellt werden, das die Daten auf eventuell vorhandenen Backing-Devices gesichert werden, falls diese noch benötigt werden.

Wir demonstrieren das Löschen eines Logical Volume Storage Pools am Beispiel des bisher verwendeten Storage Pools *testpool*. Zunächst listen wir die noch vorhandenen Backing-Devices im Storage Pool auf:

```
$ vios lsbdsp ms05-vio1 testpool
BDNAME  SIZE       VTD         SVSA
bd002     3.00 GB  lpar1_hd02  vhost0
bd003     3.00 GB  lpar1_hd03  vhost0
$
```

Es gibt nur 2 Backing-Devices, welche beide aktuell *vhost0* zugeordnet sind. Über das Kommando „*vios lsvscsi*“ finden wir die zugehörige Client-LPAR heraus:

```
$ vios lsvscsi ms05-vio1
SVSA    SLOT  CLIENT             LUNS
vhost0  C25   lpar1(3)           4
vhost1  C28   lpar2(4)           2
vhost2  C31   lpar3(5)           2
$
```

Die Client-LPAR ist *lpar1*, welche aktuell 4 Backing-Devices zugeordnet hat. Wir listen die Backing-Devices der LPAR *lpar1* (*vhost0*) auf:

```
$ vios lsvscsi ms05-vio1 vhost0
VTD         STATUS     BACKING                              BDPHYSLOC  MIRRORED  LUN
lpar1_hd02  Available  bd002                                -          N/A       0x8500000000000000
lpar1_hd03  Available  bd003                                -          N/A       0x8300000000000000
vtopt0      Available  /var/vio/VMLibrary/AIX710506_2.iso  -          N/A       0x8100000000000000
vtscsi0     Available  lpar1_hd00                           -          N/A       0x8200000000000000
$
```

Die im Storage Pool liegenden Backing-Devices *bd002* und *bd003* haben die LUN-IDs (Spalte *LUN*) *0x8500000000000000* bzw. *0x8300000000000000*. Wir überprüfen auf der *lpar1* mit dem Kommando *lscfg* um welche zugehörigen Physical Volumes es sich handelt:

```
lpar1 # lscfg -l hdisk\*
  hdisk0           U8205.E6C.05E4E5Q-V3-C5-T1-L8200000000000000  Virtual SCSI Disk Drive
  hdisk1           U8205.E6C.05E4E5Q-V3-C5-T1-L8300000000000000  Virtual SCSI Disk Drive
  hdisk2           U8205.E6C.05E4E5Q-V3-C5-T1-L8500000000000000  Virtual SCSI Disk Drive
lpar1 #
```

Die beiden Physical Volumes sind *hdisk2* und *hdisk1*. Eine kurze Überprüfung mit dem Kommando lspv zeigt das diese nicht in Benutzung sind:

```
lpar1 # lspv
hdisk0          00ecafe541e13592                    rootvg          active
hdisk1          00ecafe53e3abb01                    None
hdisk2          none                                None
lpar1 #
```

Wir löschen die beiden Physical Volumes auf der Client-LPAR:

```
lpar1 # rmdev -dl hdisk1
hdisk1 deleted
lpar1 # rmdev -dl hdisk2
hdisk2 deleted
lpar1 #
```

Da die Backing-Devices *bd002* und *bd003* nur zu Test-Zwecken angelegt wurden, können diese ohne Erstellen einer Sicherung gelöscht werden:

```
$ vios rmbdsp ms05-vio1 testpool bd002
$ vios rmbdsp ms05-vio1 testpool bd003
$
```

Eine kurze Kontrolle zeigt das der Storage Pool *testpool* nun keine Backing-Devices mehr enthält:

```
$ vios lsbdsp ms05-vio1 testpool
$
```

Das Löschen eines Logical Volume Storage Pools erfolgt nicht mit dem Kommando „*vios rmsp*“! Dieses Kommando kann nur für File Storage Pools verwendet werden. Ein Logical Volume Storage Pool kann gelöscht werden, indem

man alle Physical Volumes entfernt. Als Folge wird der Storage Pool dann selbst gelöscht. Das Kommando dafür lautet „*vios rmsppv*“:

```
$ vios rmsppv ms05-vio1 testpool hdisk2
$
```

Eine Auflistung aller Storage Pools auf dem Virtual-I/O-Server zeigt das der Storage Pool *testpool* gelöscht wurde:

```
$ vios lssp ms05-vio1
POOL    TYPE    SIZE    FREE   ALLOC  BDS
rootvg  LVPOOL  571392  69632  512    0
$
```

8.6.7. Erzeugen eines File Storage Pools

Ein File Storage Pool bezieht seine Storage Kapazität aus einem Logical Volume Storage Pool. Daher muß beim Erzeugen eines File Storage Pool immer ein Logical Volume Storage Pool angegeben werden. Das Kommando zum Erzeugen eines File Storage Pool ist das gleiche wie beim Logical Volume Storage Pool, „*vios mksp*“, lediglich die Argumente unterscheiden sich:

```
$ vios mksp ms05-vio1 filepool rootvg 10g
$
```

Wir haben hier den immer vorhandenen Logical Volume Storage Pool *rootvg* angegeben. Für die Praxis ist dies aber in der Regel keine gute Idee und es sollte unbedingt ein eigens für Storage Provisioning angelegter Logical Volume Storage Pool angegeben werden. Es muß die gewünschte Kapazität für den zu erzeugenden File Storage Pool angegeben werden.

Der neu angelegte File Storage Pool taucht in der Auflistung der Storage Pools sofort auf:

```
$ vios lssp ms05-vio1
POOL      TYPE    SIZE       FREE       ALLOC      BDS
rootvg    LVPOOL  558.00 GB   58.00 GB  512.00 MB  0
filepool  FBPOOL    9.96 GB    9.96 GB  512.00 MB  0
$
```

Der Typ eines Storage Pools ist über die Spalte *TYPE* klar erkennbar. Technisch wird ein Logical Volume in dem angegebenen Logical Volume Storage Pool (hier *rootvg*) angelegt. Das Logical Volume bekommt den im Kommando angegebenen Namen *filepool*. Der Virtual-I/O-Server erzeugt dann ein Filesystem auf dem Logical Volume und mountet das neue Dateisystem unter dem Pfad */var/vio/storagepools/filepool*.

Die Administration von Backing-Devices (Erzeugen, Mappen, Unmappen, Vergrößern, Löschen) funktioniert genauso wie weiter oben beschrieben. Der einzige Unterschied besteht darin das die Backing-Devices jetzt keine Logical Volumes mehr sind, sondern als Dateien im Dateisystem angelegt werden. Hier ein kurzes Beispiel:

```
$ vios mkbdsp ms05-vio1 filepool bd001 1g vhost0
$
```

Das Kommando ist absolut identisch zu den weiter oben verwendeten Kommandos!

Auch beim Auflisten der Backing-Devices in einem Storage-Pool ist kein Unterschied zum Falle Logical Volume Storage Pool erkennbar:

```
$ vios lsbdsp ms05-vio1 filepool
BDNAME  SIZE       VTD      SVSA
bd001     1.00 GB  vtscsi5  vhost0
$
```

Das Backing-Device *bd001* hat die Größe *1* GB und ist über das virtuelle Target Device *vtscsi5* dem Adapter *vhost0* zugeordnet.

8.6.8. Vergrößern eines File Storage Pools

Das Vergrößern eines File Storage Pools ist für den Administrator einfacher als das Vergrößern eines Logical Volume Storage Pools. Es wird lediglich ausreichend freie Kapazität im zugehörigen Logical Volume Storage Pool benötigt. Vergrößert werden kann ein File Storage Pool dann ganz einfach mit dem Kommando „vios chsp“, wobei die hinzuzufügende Storage-Kapazität angegeben werden muß:

```
$ vios chsp ms05-vio1 filepool +5g
$
```

Im Beispiel wurde der Storage Pool *filepool* um *5* GB vergrößert:

```
$ vios lssp ms05-vio1
POOL      TYPE    SIZE       FREE       ALLOC      BDS
rootvg    LVPOOL  558.00 GB   53.00 GB  512.00 MB  0
filepool  FBPOOL   14.94 GB   13.94 GB  512.00 MB  1
$
```

Die Kapazität ist von *9.96* GB auf *14.94* GB angewachsen.

8.6.9. Löschen eines File Storage Pools

Für das Löschen eines File Storage Pools gibt es das Kommando „*vios rmsp*“. Es sollte aber, wie im Falle des Löschens eines Logical Volume Storage Pools, sichergestellt werden, das die Backing-Devices nicht in Verwendung sind oder nicht mehr benötigt werden. Wir demonstrieren dies kurz an dem File Storage Pool *filepool*.

Als erstes generieren wir die Liste der Backing-Devices des Storage Pools:

```
$ vios lsbdsp ms05-vio1 filepool
BDNAME  SIZE       VTD      SVSA
bd001     1.00 GB  vtscsi5  vhost0
$
```

Es gibt nur ein Backing-Device (*bd001*), welches dem Adapter *vhost0* zugeordnet ist. Die zugehörige Client-LPAR kann mit dem Kommando „*vios lsvscsi*“ ermittelt werden:

```
$ vios lsvscsi ms05-vio1
SVSA    SLOT  CLIENT              LUNS
vhost0  C25   lpar1(3)            3
vhost1  C28   lpar2(4)            2
vhost2  C31   lpar3(5)            2
$
```

Die von der Client-LPAR *lpar1* verwendeten Backing-Devices sind:

```
$ vios lsvscsi ms05-vio1 vhost0
VTD      STATUS     BACKING                                BDPHYSLOC  MIRRORED  LUN
vtopt0   Available  /var/vio/VMLibrary/AIX710506_2.iso     -          N/A       0x8100000000000000
vtscsi0  Available  lpar1_hd00                             -          N/A       0x8200000000000000
vtscsi5  Available  /var/vio/storagepools/filepool/.bd001  -          N/A       0x8300000000000000
$
```

Das Backing-Device *bd001* hat die LUN-ID *0x8300000000000000*. Auf der AIX LPAR ermitteln wir das zugehörige Physical Volume über das Kommando *lscfg*:

```
lpar1 # lscfg -l hdisk\*
  hdisk0           U8205.E6C.05E4E5Q-V3-C5-T1-L8200000000000000  Virtual SCSI Disk Drive
  hdisk1           U8205.E6C.05E4E5Q-V3-C5-T1-L8300000000000000  Virtual SCSI Disk Drive
lpar1 #
```

Das zugehörige Physical Volume ist *hdisk1*! Eine kurze Überprüfung mittels *lspv* zeigt, das die *hdisk1* nicht in Benutzung ist:

```
lpar1 # lspv
hdisk0          00ecafe541e13592                    rootvg          active
hdisk1          none                                None
lpar1 #
```

Das Physical Volume kann daher problemlos entfernt werden:

```
lpar1 # rmdev -dl hdisk1
hdisk1 deleted
lpar1 #
```

Nun kann das Backing-Device auf dem Virtual-I/O-Server gelöscht werden. Dabei wird gleichzeitig das Mapping entfernt:

```
$ vios rmbdsp ms05-vio1 filepool bd001
$
```

Der jetzt leere File Storage Pool kann dann letztlich entfernt werden, das Kommando ist „*vios rmsp*“:

```
$ vios rmsp ms05-vio1 filepool
$
```

Eine kurze Überprüfung zeigt das der File Storage Pool nicht mehr vorhanden ist:

```
$ vios lssp ms05-vio1
POOL    TYPE    SIZE    FREE   ALLOC  BDS
rootvg  LVPOOL  571392  69632  512    0
$
```

8.7. Virtual Media Repository

PowerVM unterstützt auf Virtual-I/O-Servern sogenannte Virtual Media Repositorien, auch Virtual Media Library (oder Virtual Optical Media Library) genannt. In einer Virtual Media Repository können virtuelle Medien (Kopien von CDs oder DVDs) abgelegt werden. LPARs können auf diese virtuellen Medien mit Hilfe eines virtuellen optischen Laufwerk zugreifen. Dies ermöglicht beispielsweise die einfache Installation von LPARs. Mit Hilfe von virtuellen Medien können aber auch Updates (TL Updates oder SP Updates) oder sogar Upgrades (Migration von einer AIX-Version auf eine andere AIX-Version) durchgeführt werden. Eine weitere Anwendung ist das Erzeugen von *mksysb*s einer LPAR auf einem virtuellen Medium. Natürlich kann eine LPAR dann auch mit Hilfe des erzeugten *mksysb* auf dem virtuellen Medium wieder hergestellt werden (Recovery).

8.7.1. Anlegen einer Virtual Media Repository

Die über eine Virtual Media Repository (oder Virtual Media Library) zur Verfügung gestellten virtuellen Medien liegen auf dem Virtual-I/O-Server letztlich einfach in Form von Dateien vor. D.h. es wird Festplattenplatz auf dem Virtual-I/O-Server benötigt. Festplattenplatz wird auf einem Virtual-I/O-Server in Form von sogenannten Storage-Pools verwaltet (siehe vorheriges Kapitel über Storage Pools). Daher lassen wir uns zunächst einmal die verfügbaren Storage-Pools auf unserem Virtual-I/O-Server *ms12-vio1* mit dem Kommando „*vios lssp*“ anzeigen:

```
$ vios lssp ms12-vio1
POOL        TYPE    SIZE        FREE        ALLOC       BDS
rootvg      LVPOOL  558.00 GB   501.00 GB   512.00 MB   0
diskpool1   LVPOOL   99.88 GB    99.88 GB    64.00 MB   0
diskpool2   LVPOOL   99.88 GB    99.88 GB    64.00 MB   0
$
```

Den Storage-Pool *rootvg* gibt es immer, dieser ist einfach die Volume Group *rootvg*, die es auf jedem AIX- und Virtual-I/O-Server gibt. In unserem Fall gibt es noch zwei weitere Storage-Pools mit Namen *diskpool1* und *diskpool2*.

Bevor wir eine Virtual Media Repository anlegen, überprüfen wir kurz mit Hilfe von „*vios lsrep*“ ob es schon eine Virtual Media Repository auf unserem Virtual-I/O-Server gibt:

```
$ vios lsrep ms12-vio1
                 PARENT
SIZE  FREE  POOL  SIZE  FREE
-     -     -     -     -
$
```

Das ist offensichtlich nicht der Fall!

Eine Virtual Media Repository muss in einem Storage-Pool liegen (dieser muss vom Typ *LVPOOL* sein). Beim Anlegen einer Virtual Media Repository muss die gewünschte Größe angegeben werden. Das Kommando zum Erzeugen einer Virtual Media Repository ist „*vios mkrep*“:

```
$ vios mkrep ms12-vio1 diskpool1 10G
$
```

Erneutes Ausführen von „*vios lsrep*“ sollte jetzt die neu angelegte Virtual Media Repository anzeigen:

```
$ vios lsrep ms12-vio1
                                    PARENT
SIZE       FREE       POOL       SIZE       FREE
  9.96 GB    9.96 GB  diskpool1   99.88 GB   89.88 GB
$
```

Es kann nur eine Virtual Media Repository pro Virtual-I/O-Server geben. Der Versuch eine zweite Virtual Media Repository anzulegen, schlägt dementsprechend fehl:

```
$ vios mkrep ms12-vio1 diskpool2 10G
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"mkrep -sp diskpool2 -size 10G\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: DVD repository already exists
StdErr:
StdErr: rc=4
$
```

Soll die Virtual Media Repository im Storage-Pool *rootvg* angelegt werden, dann kann die Angabe des Storage-Pools beim Kommando „*vios mkrep*“ entfallen, *rootvg* ist der Default Storage-Pool. Die Größe muss allerdings weiterhin angegeben werden:

```
$ vios mkrep ms12-vio1 10G
$ vios lsrep ms12-vio1
                                   PARENT
SIZE       FREE       POOL    SIZE       FREE
  9.96 GB    9.96 GB  rootvg  558.00 GB  491.00 GB
$
```

Beim Anlegen einer Virtual Media Repository wird im angegebenen Storage-Pool ein Logical Volume mit Namen *VMLibrary* erzeugt. Die Größe des Logical Volumes entspricht der im Kommando „*vios mkrep*“ angegebenen Größe. Das Logical Volume wird anschließend mit einem *jfs2*-Dateisystem versehen und unter dem Mountpunkt */var/vio/VMLibrary* gemountet. Alle virtuellen Medien werden in diesem Dateisystem als gewöhnliche Dateien abgelegt:

```
ms12-vio1 $ df -g /var/vio/VMLibrary
Filesystem    GB blocks      Free %Used    Iused %Iused Mounted on
/dev/VMLibrary     10.00      9.96    1%        4     1% /var/vio/VMLibrary
ms12-vio1 $
```

8.7.2. Erzeugen von virtuellen Medien

Es gibt drei verschiedene Möglichkeiten virtuelle Medien in der Virtual Media Repository zu erzeugen:

1. Kopieren einer physikalischen CD oder DVD.

2. Kopieren eines existierenden Images, z.B. ISO-Image.

3. Erzeugen eines leeren virtuellen Mediums.

In allen drei Fällen wird das erzeugte virtuelle Medium als Datei in der Virtuellen Media Repository (*/var/vio/VMLibrary*) angelegt. Wie bei physikalischen optischen Datenträgern können die erzeugten virtuellen Medien entweder nur lesbar (*ro*, *read-only*) oder auch beschreibbar (*rw*, *read-write*) sein.

Die Namen für die erzeugten virtuellen Medien unterliegen den folgenden Einschränkungen:

- Der Name darf maximal eine Länge von 37 Zeichen besitzen.
- Erlaubte Zeichen sind alphanumerische Zeichen und die beiden Zeichen „.“ (Punkt) und „_“ (Unterstrich).
- Ein „.“ (Punkt) am Anfang des Namens ist nicht erlaubt.

Welche virtuellen Medien über eine Virtual Media Repository verfügbar sind, kann mit dem Kommando „*vios lsmedia*“ angezeigt werden:

```
$ vios lsmedia ms12-vio1
NAME  FILE SIZE  OPTICAL  ACCESS
-     -          -        -
$
```

Da die Virtual Media Repository gerade erst erzeugt wurde, enthält diese natürlich noch keine virtuellen Medien!

Nachfolgend ist anhand einer Reihe von Beispielen das Erzeugen von virtuellen Medien gezeigt.

8.7.3. Kopieren eines physikalischen optischen Datenträgers

Um eine physikalische CD oder DVD kopieren zu können, benötigt der Virtual-I/O-Server ein physikalisches CD- oder DVD-Laufwerk. (Die meisten neueren Systeme verfügen über keine optischen Laufwerke mehr.)

Mit Hilfe des Kommandos „*vios lsdev*“ kann ganz einfach überprüft werden, ob ein Virtual-I/O-Server ein CD- oder DVD-Laufwerk besitzt:

```
$ vios lsdev -t optical ms12-vio1
NAME  STATUS     PHYSLOC                  PARENT  DESCRIPTION
cd0   Available  U78AA.001.VYRGU3O-P2-D9  sata0   SATA DVD-RAM Drive
$
```

Der in den Beispielen verwendete Virtual-I/O-Server verfügt also über ein SATA DVD-RAM Laufwerk.

Als erstes muss die physikalische CD oder DVD in das physikalische Laufwerk eingelegt werden. Wir haben eine AIX 7.2 Installations DVD für den TL5 eingelegt (die erste von zwei DVDs).

Dann kann das Kommando „*vios mkmedia*“ zum Erzeugen einer Kopie in der Virtual Media Repository gestartet werden. Als Argumente müssen ein Name für das zu erstellende virtuelle Medium angegeben werden und der Gerätename des physikalischen DVD-Laufwerks (hier *cd0*) mit dem eingelegten optischen Medium:

```
$ vios mkmedia ms12-vio1 AIX_720500_DVD_1.iso cd0
$
```

Sollte kein gültiges optisches Medium in dem CD- oder DVD-Laufwerk eingelegt sein, erhält man die folgende Fehlermeldung:

```
$ vios mkmedia ms12-vio1 AIX_720500_DVD_1.iso cd0
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"mkvopt -name AIX_720500_DVD_1.iso -dev cd0\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: Unable to create virtual optical disk.
StdErr:
StdErr: rc=4
$
```

Eine kurze Überprüfung mit dem Kommando „*vios lsmedia*“ zeigt, dass ein virtuelles Medium mit dem Namen *AIX_720500_DVD_1.iso* angelegt wurde:

```
$ vios lsmedia ms12-vio1
NAME                  FILE SIZE  OPTICAL  ACCESS
AIX_720500_DVD_1.iso    3.56 GB  None     rw
$
```

Allerdings wurde das virtuelle Medium mit Lese- und Schreibberechtigungen angelegt. D.h. das virtuelle Medium kann damit bei der Benutzung unter Umständen überschrieben werden. Die gewünschte Berechtigung, „*ro*“ oder „*rw*“ kann beim Erzeugen eines virtuellen Mediums als zusätzliches Argument angegeben werden.

Wir legen die zweite Installations-DVD in das physikalische Laufwerk ein und erzeugen auch für diese eine Kopie in der Virtual Media Repository. Dieses Mal geben wir beim Erzeugen des virtuellen Mediums zusätzlich das Argument „*ro*“ für *read-only* an:

```
$ vios mkmedia ms12-vio1 AIX_720500_DVD_2.iso cd0 ro
$
```

Die Auflistung aller virtuellen Medien zeigt das das gerade erstellte virtuelle Medium nur Leseberechtigungen („*ro*“) besitzt:

```
$ vios lsmedia ms12-vio1
NAME                  FILE SIZE  OPTICAL  ACCESS
AIX_720500_DVD_1.iso    3.56 GB  None     rw
AIX_720500_DVD_2.iso    4.00 GB  None     ro
$
```

Die Berechtigungen können mittels „*vios chmedia*“ leicht geändert werden:

```
$ vios chmedia ms12-vio1 AIX_720500_DVD_1.iso ro
$
```

Nun besitzen beide erzeugten virtuellen Medien lediglich Leseberechtigungen:

```
$ vios lsmedia ms12-vio1
NAME                   FILE SIZE  OPTICAL  ACCESS
AIX_720500_DVD_1.iso     3.56 GB  None     ro
AIX_720500_DVD_2.iso     4.00 GB  None     ro
$
```

8.7.4. Kopieren eines ISO-Images

Mittlerweile werden CDs und DVDs in den meisten Fällen nicht mehr in Form von physikalischen Datenträgern verwendet, sondern in Form von ISO-Images. Das sind Dateien (meist mit der Endung *.iso*), welche ein komplettes Dateisystem im ISO-Format enthalten. Diese können über Webserver leicht zur Verfügung gestellt werden. IBM stellt beispielsweise ISO-Images für beliebige Software (inklusive AIX und IOS) über den URL https://www.ibm.com/servers/eserver/ess zur Verfügung. (Ein gültiger Softwarewartungsvertrag und eine IBMid sind für den Download von Software nötig.)

Wir haben für die nachfolgenden Beispiele das folgende ISO-Image von der genannten IBM-Webseite heruntergeladen und in das Home-Verzeichnis von *padmin* auf dem Virtual-I/O-Server *ms12-vio1* kopiert:

```
-rw-r--r--    1 padmin  system   8114339840 Jan 31 15:11
AIX_v7.2_Install_7200-05-01-2038_flash_012021_LCD8236409.iso
```

Mit dem Kommando “vios mkmedia“ kann eine Kopie dieses ISO-Images in der Virtual Media Repository erzeugt werden:

```
$ vios mkmedia ms12-vio1 AIX_720501_flash.iso \
/home/padmin/AIX_v7.2_Install_7200-05-01-2038_flash_012021_LCD8236409.iso ro
$
```

Ein Blick auf die verfügbaren virtuellen Medien („*vios lsmedia*“) zeigt das das ISO-Image jetzt über die Virtual Media Repository verfügbar ist:

```
$ vios lsmedia ms12-vio1
NAME                   FILE SIZE  OPTICAL  ACCESS
AIX_720500_DVD_1.iso     3.56 GB  None     ro
AIX_720500_DVD_2.iso     4.00 GB  None     ro
AIX_720501_flash.iso     7.56 GB  None     ro
$
```

Hinweis: Die Ausgangsdatei unter */home/padmin* kann anschließend gelöscht werden.

Sollte das Erstellen des virtuellen Mediums fehl schlagen, könnte zu wenig freier Platz in der Virtual Media Repository die Ursache sein. Die Virtual Media Repository lässt sich aber sehr leicht vergrößern, siehe „Vergrößern der Virtual Media Repository“

8.7.5. Erzeugen eines leeren virtuellen Mediums

Bisher haben wir virtuelle Medien als Kopien von physikalischen Datenträgern oder ISO-Images erstellt. In manchen Fällen soll aber ein virtuelles Medium von einer Client-LPAR beschrieben werden, z.B. um ein *mksysb* auf einer virtuellen CD abzulegen.

Für solche Fälle kann mit dem Kommando „*vios mkmedia*“ auch ein leeres virtuelles Medium mit einer gewünschten Kapazität erzeugt werden:

```
$ vios mkmedia ms12-vio1 mksysb_aix01.iso 4G
$
```

Hinweis: Die gewünschte Größe kann einfach als Argument nach dem Namen des virtuellen Mediums angegeben werden. Alternativ kann auch die Form „*size=4G*“ verwendet werden.

Da das virtuelle Medium zum Schreiben eines *mksysb* verwendet werden soll, haben wir kein Argument „*ro*“ für *read-only* angegeben. Auflisten der virtuellen Medien zeigt das das neue virtuelle Medium Schreibberechtigungen besitzt:

```
$ vios lsmedia ms12-vio1
NAME                  FILE SIZE  OPTICAL  ACCESS
AIX_720500_DVD_1.iso    3.56 GB  None     ro
AIX_720500_DVD_2.iso    4.00 GB  None     ro
AIX_720501_flash.iso    7.56 GB  None     ro
mksysb_aix01.iso        4.00 GB  None     rw
$
```

8.7.6. Ändern von virtuellen Medien

Es sind nur zwei Änderungen bei virtuellen Medien möglich. Es kann der Name geändert werden und die Berechtigungen können geändert werden. Dies ist im folgenden an zwei Beispielen demonstriert. Ausgangssituation sind die folgenden virtuellen Medien in unserer Virtual Media Repository:

```
$ vios lsmedia ms12-vio1
NAME                  FILE SIZE  OPTICAL  ACCESS
AIX_720500_DVD_1.iso    3.56 GB  None     ro
AIX_720500_DVD_2.iso    4.00 GB  None     ro
AIX_720501_flash.iso    7.56 GB  None     ro
mksysb_aix01.iso        4.00 GB  None     rw
$
```

Eine Änderung kann mit dem Kommando „vios chmedia“ durchgeführt werden.

Wir ändern den Namen des virtuellen Mediums *mksysb_aix01.iso* ab auf *mksysb_72_aix01.iso*:

```
$ vios chmedia ms12-vio1 mksysb_aix01.iso mksysb_72_aix01.iso
$ vios lsmedia -s access=rw ms12-vio1
NAME                 FILE SIZE  OPTICAL  ACCESS
mksysb_72_aix01.iso    4.00 GB  None     rw
$
```

Hinweis: Mit der Option „*-s access=rw*“ wird die Ausgabe auf Medien beschränkt, welche die Berechtigung *rw* (*read-write*) besitzen.

Als weiteres Beispiel ändern wir für das Medium *AIX_720501_flash.iso* die Berechtigungen auf *rw* (*read-write*) ab:

```
$ vios chmedia ms12-vio1 AIX_720501_flash.iso rw
$ vios lsmedia -s access=rw ms12-vio1
NAME                  FILE SIZE  OPTICAL  ACCESS
AIX_720501_flash.iso    7.56 GB  None     rw
mksysb_72_aix01.iso     4.00 GB  None     rw
$
```

Die Größe eines Mediums kann nicht nachträglich abgeändert werden (zumindest nicht mir IOS-Kommandos).

8.7.7. Vergrößern der Virtual Media Repository

Die einzige Änderung die man an einer Virtual Media Repository selbst durchführen kann, ist eine Vergrößerung. Dazu muss natürlich noch entsprechend viel verfügbarer Platz im unterliegenden Storage Pool (häufig *rootvg*) vorhanden sein.

Beim Kopieren eines ISO-Images ist der folgende Fehler aufgetreten:

```
$ vios mkmedia ms12-vio1 AIX_720501_flash.iso \
/home/padmin/AIX_v7.2_Install_7200-05-01-2038_flash_012021_LCD8236409.iso ro
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"mkvopt -name AIX_720501_flash.iso -file /home/
padmin/AIX_v7.2_Install_7200-05-01-2038_flash_012021_LCD8236409.iso -ro\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: Unable to create virtual optical disk.
StdErr:
StdErr: Error occurred. For more information check the trace file /home/ios/logs/
ioscli_global.trace
StdErr: rc=4
$
```

Eine mögliche Ursache ist das der freie Platz in der Virtual Media Repository nicht ausreicht um eine Kopie des ISO-Images zu erzeugen. Eine Überprüfung der aktuellen Größe und des freien Platzes mit „*vios lsrep*“ zeigt, das dies hier der Fall ist:

```
$ vios lsrep ms12-vio1
                                    PARENT
SIZE       FREE       POOL    SIZE       FREE
  9.96 GB    2.39 GB  rootvg  558.00 GB  491.00 GB
$
```

Es sind lediglich etwas mehr als 2 GB freier Platz verfügbar. Das zu kopierende ISO-Image hat aber eine Größe von knapp 8 GB.

Im Parent Storage Pool (*rootvg*) sind noch 491 GB freier Platz verfügbar. Wir vergrößern die Virtual Media Repository um weitere 20 GB mit dem Kommando „*vios chrep*“:

```
$ vios chrep ms12-vio1 20G
```

```
$ vios lsrep ms12-vio1
                                    PARENT
SIZE       FREE       POOL    SIZE       FREE
29.88 GB   22.31 GB  rootvg  558.00 GB  471.00 GB
$
```

Ein erneuter Versuch das ISO-Image oben zu kopieren ist jetzt erfolgreich:

```
$ vios mkmedia ms12-vio1 AIX_720501_flash.iso \
/home/padmin/AIX_v7.2_Install_7200-05-01-2038_flash_012021_LCD8236409.iso ro
$
```

8.7.8. Erzeugen von virtuellen optischen Laufwerken

Um virtuelle optische Medien der Virtual Media Repository nutzen zu können, benötigen LPARs ein virtuelles optisches Laufwerk. Ein virtuelles optisches Laufwerk ist ein virtuelles SCSI-Gerät, dementsprechend benötigt eine LPAR einen virtuellen SCSI-Client-Adapter. Der zugehörige virtuelle SCSI-Server-Adapter muss auf dem virtuellen I/O-Server dessen Virtual Media Repository genutzt werden soll angelegt werden. Das Anlegen von virtuellen SCSI-Adaptern ist im Kapitel Virtual SCSI beschrieben. Wir gehen im Folgenden davon aus das die benötigten virtuellen SCSI-Adapter existieren.

Bevor wir das erste virtuelle optische Laufwerk anlegen, überprüfen wir kurz für welche LPARs es virtuelle SCSI-Server-Adapter (*vhost*-Adapter) auf unserem virtuellen I/O-Server gibt:

```
$ vios lsvscsi ms12-vio1
SVSA    SLOT  CLIENT    LUNS
vhost0  C25   aix01(3)  0
vhost1  C28   aix03(4)  0
vhost2  C30   aix04(5)  0
vhost3  C31   aix05(6)  0
$
```

Das Kommando zum Erzeugen eines virtuellen optischen Laufwerks ist „*vios mkvopt*“. Als Argument muss neben dem Virtual-I/O-Server der *vhost*-Adapter der LPAR angegeben werden, für die ein Laufwerk angelegt werden soll:

```
$ vios mkvopt ms12-vio1 vhost0
$
```

Hinweis: Das LPAR-Tool verwendet „*mkvopt*“ um ein virtuelles optisches Laufwerk zu erzeugen, das *padmin*-Kommando „*mkvopt*“ erzeugt hingegen ein virtuelles optisches Medium!

Der Adapter *vhost0* gehört zur LPAR *aix01*. Eine Übersicht über die existierenden virtuellen optischen Laufwerke erhält man mit „*vios lsvopt*“:

```
$ vios lsvopt ms12-vio1
VTD     MEDIA     SIZE
vtopt0  No Media  n/a
$
```

Per Default wird als Gerätename der Präfix „*vtopt*“ gefolgt von einer eindeutigen Instanznummer verwendet.

Damit eine LPAR das neue optische Laufwerk auch verwenden kann, muss dieses in das laufende Betriebssystem herein konfiguriert werden. Im Falle von AIX erfolgt das mit Hilfe von *cfgmgr*:

```
aix01 # cfgmgr -l vio0
aix01 # lsdev -l cd0
cd0 Available  Virtual SCSI Optical Served by VIO Server
aix01 #
```

Hinweis: Man kann den *cfgmgr* auch ohne Option starten. Das Angeben von „*-l vio0*“ schränkt die Suche nach neuen Geräten auf virtuelle Geräte ein.

Das virtuelle optische Laufwerk taucht in AIX als Gerät *cd0* (*Virtual SCSI Optical Served by VIO Server*) auf.

Wir legen ein weiteres virtuelles Laufwerk an, dieses Mal für die LPAR *aix03* (*vhost1*):

```
$ vios mkvopt ms12-vio1 vhost1
$
```

Das zweite virtuelle optische Laufwerk erhält den Namen *vtopt1*:

```
$ vios lsvopt ms12-vio1
VTD     MEDIA     SIZE
vtopt0  No Media  n/a
vtopt1  No Media  n/a
$
```

Leider lässt sich an diesen generischen Gerätenamen nicht erkennen zu welcher LPAR die virtuellen optischen Laufwerke gehören. Verwendet man das Kommando „*vios lsvscsi*“ und gibt den *vhost*-Adapter an, lässt sich erkennen welches virtuelle optische Laufwerk zu dem angegebenen *vhost*-Adapter (und damit zu einer bestimmten LPAR) gehört:

```
$ vios lsvscsi ms12-vio1 vhost1
VTD     STATUS     BACKING  BDPHYSLOC  MIRRORED  LUN
vtopt0  Available  -        -          N/A       0x8100000000000000
$
```

Das ist aber etwas mühsam.

Beim Anlegen eines virtuellen optischen Laufwerks kann zusätzlich der gewünschte Name für das Gerät angegeben werden. Diese Möglichkeit sollte man nutzen und Namen vergeben, an denen man die Zuordnung zu den LPARs sofort erkennen kann. Als Beispiel erzeugen wir ein virtuelles optisches Laufwerk für die LPAR *aix04* (*vhost2*) und verwenden als Namen für das Laufwerk „*aix04_cd*“:

```
$ vios mkvopt ms12-vio1 vhost2 aix04_cd
$
```

Der Ausgabe von „*vios lsvopt*“ kann jetzt sofort entnommen werden, das das neue virtuelle Laufwerk zur LPAR *aix04* gehört:

```
$ vios lsvopt ms12-vio1
VTD       MEDIA     SIZE
aix04_cd  No Media  n/a
vtopt0    No Media  n/a
vtopt1    No Media  n/a
$
```

8.7.9. Einlegen und Auswerfen von virtuellen Medien

Damit ein virtuelles Medium von einer LPAR verwendet werden kann, muss das Medium zunächst in das zugehörige virtuelle optische Laufwerk eingelegt werden. Dies erfolgt durch Aufruf des Kommandos „*vios loadopt*“.

Bevor wir das durchführen, listen wir zunächst erst noch einmal die verfügbaren virtuellen Medien und die existierenden virtuellen optischen Laufwerke auf:

```
$ vios lsmedia ms12-vio1
NAME                  FILE SIZE  OPTICAL  ACCESS
AIX_720500_DVD_1.iso    3.56 GB  None     ro
AIX_720500_DVD_2.iso    4.00 GB  None     ro
AIX_720501_flash.iso    7.56 GB  None     rw
$ vios lsvopt ms12-vio1
VTD       MEDIA     SIZE
aix04_cd  No Media  n/a
vtopt0    No Media  n/a
vtopt1    No Media  n/a
$
```

Wir legen das virtuelle Medium *AIX_720500_DVD_1.iso* in das Laufwerk der LPAR *aix04* (*aix04_cd*) ein:

```
$ vios loadopt ms12-vio1 AIX_720500_DVD_1.iso aix04_cd
$
```

Die Ausgabe von „*vios lsvopt*“ zeigt das in das Laufwerk *aix04_cd* jetzt das genannte virtuelle Medium eingelegt ist:

```
$ vios lsvopt ms12-vio1
VTD       MEDIA                 SIZE
aix04_cd  AIX_720500_DVD_1.iso    3.56 GB
vtopt0    No Media              n/a
vtopt1    No Media              n/a
$
```

Auf der Client-LPAR kann dann das virtuelle Medium sofort genutzt werden. Z.B. in dem man das im Medium enthaltene Dateisystem mountet:

```
aix04 # mount -o ro -V cdrfs /dev/cd0 /cdrom
aix04 # ls -l /cdrom
total 80
drwxr-xr-x    2 4000     4000           2048 Dec  8 2020  7200-05/
-rw-r--r--    1 4000     4000             16 Dec  8 2020  OSLEVEL
```

```
-rw-r--r--    1 4000     4000            901 Dec  8 2020  README.aix
drwxrwxr-x    4 4000     4000           2048 Dec  8 2020  RPMS/
-rw-r--r--    1 4000     4000           6529 Dec  8 2020  bosinst.data
-rw-r--r--    1 4000     4000          15430 Dec  8 2020  image.data
drwxr-xr-x    4 4000     4000           2048 Dec  8 2020  installp/
drwxr-xr-x    3 4000     4000           2048 Dec  8 2020  ppc/
drwxr-xr-x    3 4000     4000           2048 Dec  8 2020  root/
drwxr-xr-x   10 4000     4000           2048 Dec  8 2020  usr/
aix04 #
```

Um ein virtuelles Medium wieder auszuwerfen, kann das Kommando „*vios unloadopt*“ verwendet werden. Allerdings darf das Medium dann nicht mehr in der Client-LPAR in Benutzung sein, wie das folgende Beispiel zeigt:

```
$ vios unloadopt ms12-vio1 aix04_cd
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"unloadopt -vtd aix04_cd\“
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: Unable to perform the requested operation.
StdErr: aix04_cd’ is currently reserved by the client.
StdErr:
StdErr: Use the -release flag to remove the reserve
StdErr:
StdErr: rc=41
$
```

Das Dateisystem des virtuellen Mediums ist noch in der LPAR gemountet. Man kann zwar mit der Option „*-r*“ erzwingen das das Medium ausgeworfen wird, aber dann hängt das gemountete Dateisystem in der Client-LPAR in der Luft und ist nicht mehr benutzbar. Wir hängen daher zunächst das Dateisystem in der Client-LPAR aus

```
aix04 # umount /cdrom
aix04 #
```

und versuchen dann erneut das Medium auszuwerfen:

```
$ vios unloadopt ms12-vio1 aix04_cd
$
```

Es gibt keine Fehlermeldung und ein Auflisten der virtuellen optischen Laufwerke zeigt auch, das das Medium nicht mehr eingelegt ist:

```
$ vios lsvopt ms12-vio1
VTD       MEDIA     SIZE
aix04_cd  No Media  n/a
vtopt0    No Media  n/a
vtopt1    No Media  n/a
$
```

Ein *read-only* Medium kann ohne weiteres von mehreren LPARs (virtuellen optischen Laufwerken) gleichzeitig genutzt werden:

```
$ vios lsmedia ms12-vio1
NAME                  FILE SIZE  OPTICAL   ACCESS
```

```
AIX_720500_DVD_1.iso    3.56 GB  aix04_cd  ro
AIX_720500_DVD_1.iso    3.56 GB  vtopt0    ro
AIX_720500_DVD_2.iso    4.00 GB  None      ro
AIX_720501_flash.iso    7.56 GB  None      rw
$
```

Bei Medien mit Schreibberechtigungen ist das aber nicht der Fall. Ein solches Medium kann nur einmal zeitgleich verwendet werden:

```
$ vios loadopt ms12-vio1 mksysb_aix01.iso aix04_cd
$ vios loadopt ms12-vio1 mksysb_aix01.iso vtopt0
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"loadopt -disk mksysb_aix01.iso -vtd vtopt0\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: The specified virtual disk is not read-only, and cannot be
StdErr: loaded into multiple devices simultaneously
StdErr:
StdErr: rc=4
$
```

Was die Fehlermeldung auch ganz eindeutig sagt!

8.7.10. Löschen von virtuellen optischen Laufwerken

Im Folgenden zeigen wir das Löschen von virtuellen optischen Laufwerken anhand von einigen konkreten Beispielen. Das Kommando zum Entfernen von virtuellen optischen Laufwerken ist „*vios rmvopt*“. Die zu löschenden Laufwerke sind:

```
$ vios lsvopt ms12-vio1
VTD       MEDIA                 SIZE
aix04_cd  AIX_720500_DVD_1.iso    3.56 GB
vtopt0    AIX_720500_DVD_1.iso    3.56 GB
vtopt1    No Media              n/a
$
```

Wie die Ausgabe zeigt befinden sich in zwei der virtuellen optischen Laufwerke (*aix04_cd* und *vtopt0*) noch virtuelle Medien. Es sollen alle drei angezeigten Laufwerke entfernt werden.

Wir starten mit dem letzten Laufwerk (*vtopt1*). Dieses enthält aktuell kein virtuelles Medium und wir entfernen dieses virtuelle optische Laufwerk mit „*vios rmvopt*“:

```
$ vios rmvopt ms12-vio1 vtopt1
$
```

Das virtuelle optische Laufwerk wurde auf dem Virtual-I/O-Server entfernt:

```
$ vios lsvopt ms12-vio1
VTD       MEDIA                 SIZE
aix04_cd  AIX_720500_DVD_1.iso    3.56 GB
vtopt0    AIX_720500_DVD_1.iso    3.56 GB
```

```
$
```

Eine kurze Überprüfung auf der LPAR *aix03*, zu welcher das Laufwerk gehört hat, zeigt das für AIX das virtuelle optische Laufwerk aber noch verfügbar ist:

```
aix03 # lsdev -l cd0
cd0 Available  Virtual SCSI Optical Served by VIO Server
aix03 #
```

Da das Laufwerk nicht in Verwendung war, ist dies kein Problem. Aber es ist natürlich unschön Geräte im Kernel zu haben, die gar nicht mehr existieren. Nach dem nächsten Reboot der LPAR würde das Gerät den Zustand *defined* haben. Wir löschen das Gerät auch in der LPAR:

```
aix03 # rmdev -dl cd0
cd0 deleted
aix03 #
```

Die korrekte Reihenfolge wäre zunächst das virtuelle CD Laufwerk in der Client-LPAR aus dem Betriebssystem zu entfernen (bei AIX: *rmdev*) und erst dann das virtuelle optische Laufwerk auf dem Virtual-I/O-Server entfernen!

Als nächstes entfernen wir das Laufwerk *vtopt0*. Das dort eingelegte Medium ist auf der Client-LPAR (*aix01*) gemountet. Auch hier werden wir erstmal die falsche Reihenfolge der notwendigen Schritte ausführen, um zu sehen was dabei passiert. Das Kommando zum Entfernen des Laufwerks ist erfolgreich:

```
$ vios rmvopt ms12-vio1 vtopt0
$
```

Was ist mit dem gemounteten Dateisystem auf der Client-LPAR (*aix01*) passiert? Wir starten ein *ls*-Kommando auf der LPAR:

```
aix01 # ls -l /cdrom/usr
/cdrom/usr/bin not found
/cdrom/usr/ccs not found
/cdrom/usr/lib not found
/cdrom/usr/lpp not found
/cdrom/usr/sbin not found
/cdrom/usr/share not found
/cdrom/usr/swlag not found
/cdrom/usr/sys not found
total 0
aix01 #
```

Der Zugriff auf die Daten des virtuellen Mediums funktioniert jetzt natürlich nicht mehr!

Wir hätten also vor dem Entfernen des Laufwerks auf dem Virtual-I/O-Server zunächst das Dateisystem auf der Client-LPAR aushängen müssen, dann das virtuelle CD Gerät in der LPAR entfernen und erst dann das virtuelle Laufwerk auf dem Virtual-I/O-Server löschen dürfen!

Wir korrigieren das, indem wir den *umount*-Befehl und das Löschen des Geräts mit *rmdev* noch nachholen:

```
aix01 # umount /cdrom
aix01 # rmdev -dl cd0
cd0 deleted
aix01 #
```

Für das Löschen des letzten virtuellen optischen Laufwerks verwenden wir jetzt die korrekte Reihenfolge der Schritte! Das virtuelle optische Laufwerk *aix04_cd* gehört zur LPAR *aix04*. Wir überprüfen auf der Client-LPAR *aix04* ob die virtuelle CD in Verwendung ist:

```
aix04 # mount | grep cdrfs
         /dev/cd0         /cdrom           cdrfs  Feb 07 16:08 ro
aix04 #
```

Das virtuelle CD Laufwerk ist in Verwendung. Wir beenden die Verwendung, indem wir das Dateisystem aushängen:

```
aix04 # umount /cdrom
aix04 #
```

Als nächstes löschen wir das Gerät *cd0* aus dem Betriebssytem:

```
aix04 # rmdev -dl cd0
cd0 deleted
aix04 #
```

Erst jetzt entfernen wir das virtuelle optische Laufwerk auf dem Virtual-I/O-Server:

```
$ vios rmvopt ms12-vio1 aix04_cd
$
```

8.7.11. Löschen von virtuellen Medien

Werden virtuelle Medien nicht mehr benötigt, z.B. ISO-Images von alten AIX-Versionen, können diese gelöscht werden, um wieder Platz in der Virtual Media Repository zu schaffen. Wir zeigen das Löschen mit Hilfe von „*vios rmmedia*“ an einigen Beispielen.

Aktuell haben wir die folgenden virtuellen Medien in unserer Virtual Media Repository auf dem Virtual-I/O-Server *ms12-vio1*:

```
$ vios lsmedia ms12-vio1
NAME                   FILE SIZE  OPTICAL   ACCESS
AIX_720500_DVD_1.iso     3.56 GB  vtopt0    ro
AIX_720500_DVD_2.iso     4.00 GB  None      ro
AIX_720501_flash.iso     7.56 GB  None      rw
mksysb_aix01.iso         4.00 GB  aix04_cd  rw
$
```

Zwei der Medien (*AIX_720500_DVD_1.iso* und *mksysb_aix01.iso*) sind aktuell in virtuellen optischen Laufwerken eingelegt und damit potentiell in Verwendung.

Wir löschen zunächst das virtuelle Medium *AIX_720501_flash.iso*, das aktuell in keinem Laufwerk eingelegt ist:

```
$ vios rmmedia ms12-vio1 AIX_720501_flash.iso
$
```

Eine Überprüfung zeigt das das Medium erfolgreich entfernt wurde:

```
$ vios lsmedia ms12-vio1
NAME                  FILE SIZE  OPTICAL   ACCESS
AIX_720500_DVD_1.iso    3.56 GB  vtopt0    ro
AIX_720500_DVD_2.iso    4.00 GB  None      ro
mksysb_aix01.iso        4.00 GB  aix04_cd  rw
$
```

Als nächstes löschen wir das virtuelle Medium *mksysb_aix01.iso*. Das Medium ist zur Zeit im virtuellen optischen Laufwerk *aix04_cd* eingelegt und könnte potentiell vom Client (*aix04*) in Verwendung sein. Dies lässt sich überprüfen, indem wir versuchen das Medium auszuwerfen. Dies ist nur erfolgreich wenn der Client das Medium nicht in Verwendung hat:

```
$ vios unloadopt ms12-vio1 aix04_cd
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"unloadopt -vtd aix04_cd\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: Unable to perform the requested operation.
StdErr: 'aix04_cd' is currently reserved by the client.
StdErr:
StdErr: Use the -release flag to remove the reserve
StdErr:
StdErr: rc=41
$
```

Das Medium ist also aktuell auf dem Client in Verwendung! Wir gehen auf den Client und hängen das gemountete Dateisystem aus:

```
aix04 # umount /cdrom
aix04 #
```

Anschließend versuchen wir erneut das Medium aus dem Laufwerk auszuwerfen:

```
$ vios unloadopt ms12-vio1 aix04_cd
$
```

Das war nun erfolgreich. Damit ist das zu löschende virtuelle Medium nicht mehr in Verwendung und auch in keinem Laufwerk mehr eingelegt. Daher können wir das Medium nun mit dem Kommando „*vios rmmedia*“ löschen:

```
$ vios rmmedia ms12-vio1 mksysb_aix01.iso
$
```

Das Löschen war jetzt erfolgreich:

```
$ vios lsmedia ms12-vio1
NAME                  FILE SIZE  OPTICAL  ACCESS
AIX_720500_DVD_1.iso    3.56 GB  vtopt0   ro
AIX_720500_DVD_2.iso    4.00 GB  None     ro
$
```

Hinweis: Bei virtuellen Medien die nur Leseberechtigung besitzen, kann das virtuelle Medium ohne weiteres in mehreren virtuellen optischen Laufwerken eingelegt und von mehreren Client-LPARs in Verwendung sein.

Versucht man ein virtuelles Medium zu löschen, welches in ein Laufwerk eingelegt und vom Client in Benutzung ist, erhält man die folgende Fehlermeldung:

```
$ vios rmmedia ms12-vio1 AIX_720500_DVD_1.iso
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"rmvopt -name AIX_720500_DVD_1.iso\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: Unable to perform operation while the virtual DVD is loaded.
StdErr:
StdErr: rc=4
$
```

Man kann zwar versuchen ein Löschen mit der Option „*-f*“ (*force*) zu erzwingen, das ist aber nicht erfolgreich wenn das Medium vom Client in Benutzung ist:

```
$ vios rmmedia -f ms12-vio1 AIX_720500_DVD_1.iso
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"rmvopt -f -name AIX_720500_DVD_1.iso\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: Unable to perform the requested operation.
StdErr: 'vtopt0' is currently reserved by the client.
StdErr:
StdErr: rc=41
$
```

Bevor ein virtuelles Medium gelöscht werden kann, muss also auf jeden Fall auf dem Client ein gemountetes CD-Filesystem zuerst ausgehangen werden, bevor das Löschen des virtuellen Mediums erfolgreich sein kann.

8.7.12. Löschen einer Virtual Media Repository

Wird eine Virtual Media Repository nicht mehr benötigt, kann sie mit dem Kommando „*vios rmrep*“ gelöscht werden. Allerdings dürfen dann keine virtuelle Medien auf Client-LPARs in Verwendung sein!

Bevor wir versuchen die Virtual Media Repository zu löschen, schauen wir uns zunächst die Größe an:

```
$ vios lsrep ms12-vio1
                                    PARENT
SIZE       FREE       POOL    SIZE       FREE
39.84 GB   24.72 GB  rootvg  528.00 GB  430.00 GB
$
```

Die Virtual Media Repository ist offensichtlich nicht leer, es sind nur *24.72 GB* der insgesamt *39.84 GB* frei.

Die enthaltenen virtuellen Medien sind:

```
$ vios lsmedia ms12-vio1
NAME                  FILE SIZE  OPTICAL   ACCESS
AIX_720500_DVD_1.iso    3.56 GB  aix04_cd  ro
AIX_720500_DVD_2.iso    4.00 GB  vtopt0    ro
AIX_720501_flash.iso    7.56 GB  None      ro
$
```

Zwei der virtuellen Medien sind in virtuellen optischen Laufwerken geladen. Dabei ist das erste Medium (*AIX_720500_DVD_1.iso*) auf der LPAR *aix04* gemountet. Das zweite Medium (*AIX_720500_DVD_2.iso*) ist nur eingelegt, aber nicht benutzt. Das ist natürlich nicht an der Ausgabe oben zu erkennen, sondern kann auf den LPARs überprüft werden.

Im ersten Schritt sollte sichergestellt werden, das keines der virtuellen Medien auf einer Client-LPAR in Verwendung ist. Die Überprüfung kann durch den Versuch das virtuelle Medium auszuwerfen erfolgen.

Wir versuchen das erste Medium aus dem virtuellen Laufwerk *aix04_cd* auszuwerfen:

```
$ vios unloadopt ms12-vio1 aix04_cd
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"unloadopt -vtd aix04_cd\"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: Unable to perform the requested operation.
StdErr: 'aix04_cd' is currently reserved by the client.
StdErr:
StdErr: Use the -release flag to remove the reserve
StdErr:
StdErr: rc=41
$
```

Das war nicht erfolgreich, das Medium ist also auf dem Client in Verwendung und wir müssen auf dem Client einen *umount* für das CD-Dateisystem ausführen. Der Client ist die LPAR *aix04*:

```
aix04 # umount /cdrom
aix04 #
```

Ein erneuter Versuch das Medium auszuwerfen ist jetzt erfolgreich:

```
$ vios unloadopt ms12-vio1 aix04_cd
$
```

Als nächstes versuchen wir das zweite Medium aus dem Laufwerk *vtopt0* auszuwerfen:

```
$ vios unloadopt ms12-vio1 vtopt0
$
```

Das war sofort erfolgreich, was bedeutet das das Medium auf der zugehörigen LPAR (*aix01*) nicht in Verwendung war.

Ein erneutes Auflisten aller virtuellen Medien zeigt das keines der virtuellen Medien jetzt noch in einem virtuellen optischen Laufwerk eingelegt ist:

```
$ vios lsmedia ms12-vio1
NAME                   FILE SIZE  OPTICAL  ACCESS
AIX_720500_DVD_1.iso     3.56 GB  None     ro
AIX_720500_DVD_2.iso     4.00 GB  None     ro
AIX_720501_flash.iso     7.56 GB  None     ro
$
```

Wir versuchen daher jetzt die Virtual Media Repository zu löschen. Das Kommand ist „*vios rmrep*“:

```
$ vios rmrep ms12-vio1
hmc01: viosvrcmd -m ms12 -p ms12-vio1 -c \"rmrep \"
ERROR: remote HMC command returned an error (1)
StdErr: HSCL2970 The IOServer command has failed because of the following reason:
StdErr: DVD repository contains file backed DVD media. Use -f to remove
StdErr:
StdErr: rc=4
$
```

Standardmäßig löscht das Kommando nur eine leere Virtual Media Repository. Wir müssen aber nun nicht alle virtuellen Medien einzeln mit „*vios rmmedia*“ löschen, denn das kann auch „*vios rmrep*“ erledigen, wenn die Option „*-f*“ (*force*) angegeben wird:

```
$ vios rmrep -f ms12-vio1
$
```

Das Kommando ist erfolgreich, die Virtual Media Repository wurde gelöscht. Das Kommando „*vios lsrep*“ zeigt das es keine Virtual Media Repository mehr gibt:

```
$ vios lsrep ms12-vio1
                 PARENT
SIZE  FREE  POOL  SIZE  FREE
-     -     -     -     -
$
```

8.8. Link Aggregations

to be done

```
$ vios help lnagg
USAGE: vios [<option> ...] <keyword> [<option> ...] [<argument> ...]

Recognized keywords for topic 'lnagg' are:
   [-h <hmc>] [-m <ms>] addlnaggadapter [-b] [-v] <vios> <lnagg> <ent>
```

```
   [-h <hmc>] [-m <ms>] chlnagg [-f] [-v] <vios> <lnagg> <attribute> ...
   [-h <hmc>] [-m <ms>] failoverlnagg [-v] <vios> <lnagg>
   [-h <hmc>] [-m <ms>] lslnagg [-a|-c] [{-o <format>|-f|-j|-y}] [-F <fields>] [-s
<selections>] <vios>
   [-h <hmc>] [-m <ms>] mklnagg [-M] [-a] [-v] <vios> <ent> ... [<attribute> ...]
   [-h <hmc>] [-m <ms>] rmlnagg [-M] [-a] [-v] <vios> <lnagg>
   [-h <hmc>] [-m <ms>] rmlnaggadapter [-v] <vios> <lnagg> <ent>
$
```

9. Live Partition Mobility (LPM)

to be done

```
$ lpar help lpm
USAGE: lpar [<option> ...] <keyword> [<option> ...] [<argument> ...]

Recognized keywords for topic 'lpm' are:
   [-h <hmc>] [-m <ms>] lslparmigr [{-o <format>|-f|-j|-y}] [-F <fields>] [-s <selections>]
[-v] [<lpar> ...]
   [-h <hmc>] [-m <ms>] migrate [-l <detail_level>] [-w <wait_time>] [-v] <lpar>
<target_ms> [<attributes> ...]
   [-h <hmc>] [-m <ms>] stopmigr [-l <detail_level>] [-w <wait_time>] [-v] <lpar>
   [-h <hmc>] [-m <ms>] validate [-l <detail_level>] [-w <wait_time>] [-v] <lpar>
<target_ms> [<attributes> ...]
$
```