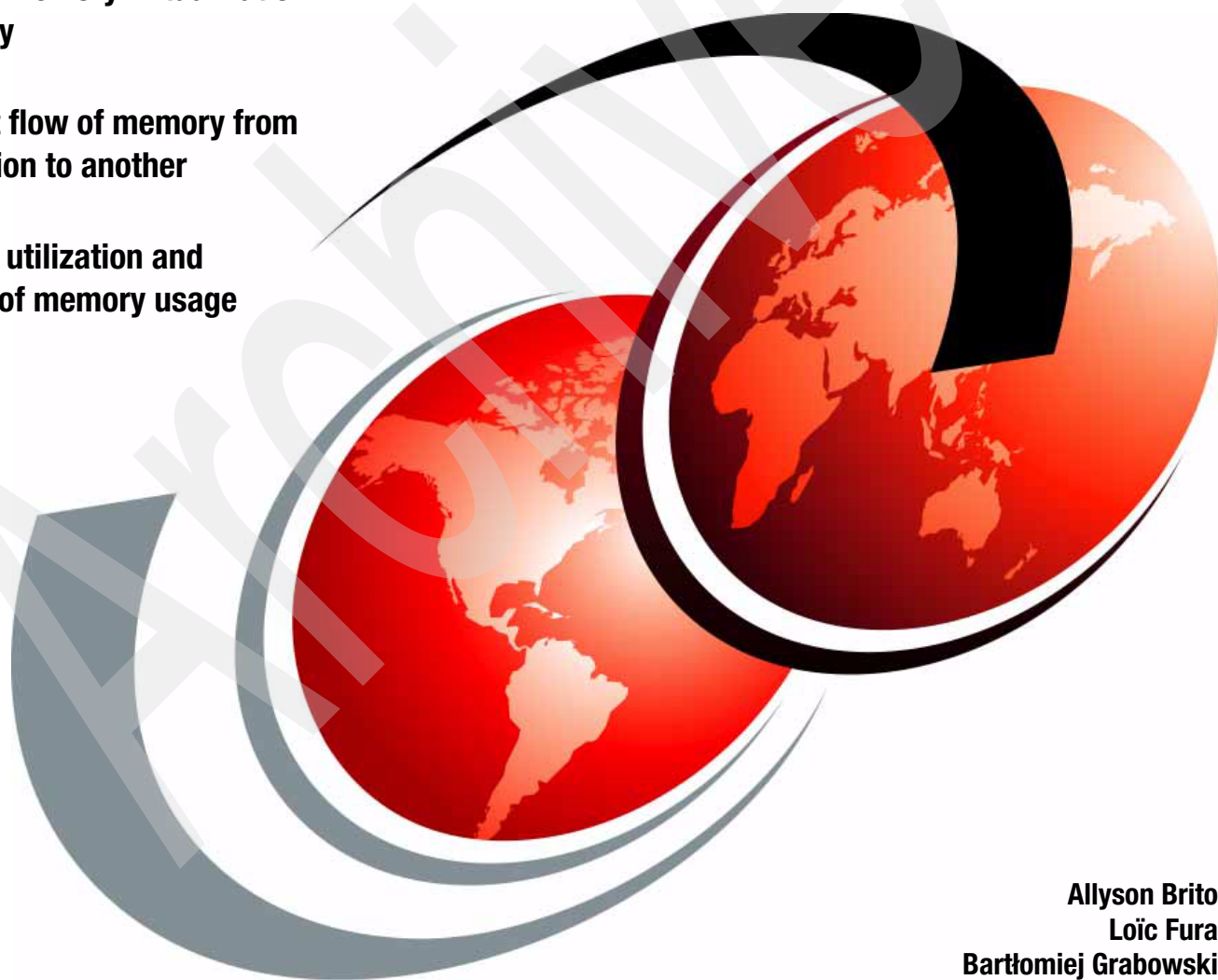


# IBM PowerVM Virtualization Active Memory Sharing

Advanced memory virtualization  
technology

Intelligent flow of memory from  
one partition to another

Increased utilization and  
flexibility of memory usage



Allyson Brito  
Loïc Fura  
Bartłomiej Grabowski





International Technical Support Organization

## **IBM PowerVM Virtualization Active Memory Sharing**

June 2011

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.

## **Second Edition (June 2011)**

This edition applies to AIX Version 7.1, IBM i Version 7.1, Novell Suse SLES11 kernel 2.6.32.12-0.7, HMC Version 7 Release 7.2 SP 1, Virtual I/O Server Version 2.2.010 FP 24 SP 1 running on IBM Power Systems with POWER7 processor-based technology.

© Copyright International Business Machines Corporation 2011. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
The team who wrote this paper .....	ix
Now you can become a published author, too! .....	x
Comments welcome .....	x
Stay connected to IBM Redbooks .....	xi
<b>Chapter 1. Overview</b> .....	1
1.1 Requirements .....	2
1.2 Dedicated and shared memory .....	2
1.3 Shared memory pool .....	4
1.4 Paging Virtual I/O Server .....	5
1.5 Shared memory partitions .....	6
1.5.1 Logical and physical memory .....	6
1.5.2 Memory classification .....	7
1.6 Usage examples .....	9
1.6.1 Logical memory overcommitment .....	9
1.6.2 Physical memory overcommit .....	12
<b>Chapter 2. Detailed architecture</b> .....	13
2.1 Dedicated versus shared memory model .....	14
2.1.1 Dedicated memory model .....	14
2.1.2 Shared memory model .....	15
2.2 Active Memory Sharing technical details .....	16
2.3 Active Memory Sharing components .....	19
2.3.1 Virtualization Control Point .....	20
2.3.2 Shared memory pool .....	20
2.3.3 Active Memory Sharing Manager .....	21
2.3.4 Paging Virtual I/O Server .....	21
2.3.5 Paging devices .....	22
2.3.6 Virtual Asynchronous Service Interface .....	23
2.3.7 I/O entitled memory .....	23
2.4 Active Memory Sharing supporting technologies .....	23
2.4.1 Page loaning .....	23
2.4.2 Collaborative Memory Manager .....	25
2.4.3 Memory affinity .....	25
2.4.4 Components specific to IBM i .....	26
2.5 Active Memory Sharing supported technologies .....	26
2.5.1 Active Memory Expansion .....	26
2.5.2 Partition suspend and resume .....	29
2.5.3 Live Partition Mobility .....	29
2.6 Processor and memory virtualization compared .....	31
<b>Chapter 3. Planning for Active Memory Sharing</b> .....	33
3.1 Active Memory Sharing prerequisites .....	34
3.2 Deployment considerations .....	34
3.2.1 Overcommitment .....	34

3.2.2	Workload selection . . . . .	37
3.2.3	Consolidation factors . . . . .	38
3.2.4	Paging device planning . . . . .	38
3.3	Sizing Active Memory Sharing . . . . .	38
3.3.1	Virtual I/O Server resource sizing . . . . .	39
3.3.2	Shared memory partition CPU sizing . . . . .	40
<b>Chapter 4.</b>	<b>Configuring and managing . . . . .</b>	<b>41</b>
4.1	Creating the paging devices . . . . .	42
4.2	Creating the shared memory pool . . . . .	43
4.3	Creating a shared memory partition . . . . .	49
4.4	Managing Active Memory Sharing . . . . .	51
4.4.1	Paging device assignment . . . . .	51
4.4.2	Adding paging devices . . . . .	53
4.4.3	Removing paging devices . . . . .	53
4.4.4	Changing the size of a paging device . . . . .	53
4.4.5	Managing the shared memory pool size . . . . .	54
4.4.6	Deleting the shared memory pool . . . . .	54
4.4.7	Dynamic LPAR for shared memory partitions . . . . .	55
4.4.8	Switching between dedicated and shared memory . . . . .	55
4.4.9	Starting and stopping the Virtual I/O Server . . . . .	56
4.4.10	Dual VIOS considerations . . . . .	56
<b>Chapter 5.</b>	<b>Monitoring . . . . .</b>	<b>59</b>
5.1	Management Console . . . . .	60
5.2	Virtual I/O Server monitoring . . . . .	64
5.3	Monitoring AIX . . . . .	66
5.3.1	The vmstat command . . . . .	66
5.3.2	The lparstat command . . . . .	69
5.3.3	The topas command . . . . .	70
5.4	Monitoring IBM i . . . . .	73
5.4.1	Checking the QAPMSHRMP table using SQL . . . . .	74
5.4.2	Checking QAPMSHRMP table using IBM System Director Navigator for i . . . . .	75
5.5	Monitoring Linux . . . . .	79
<b>Chapter 6.</b>	<b>Tuning . . . . .</b>	<b>83</b>
6.1	Shared memory pool tuning . . . . .	84
6.1.1	Shared memory pool configuration . . . . .	84
6.1.2	Virtual I/O Server configuration . . . . .	84
6.2	Shared memory partition tuning . . . . .	86
6.2.1	Memory sharing policy and memory load factor . . . . .	86
6.2.2	Logical partition . . . . .	87
6.2.3	AIX operating system . . . . .	88
6.2.4	IBM i operating system . . . . .	91
<b>Chapter 7.</b>	<b>Migrating to AMS . . . . .</b>	<b>93</b>
7.1	Measuring the memory utilization in dedicated mode . . . . .	94
7.2	Migrating from dedicated to shared memory mode . . . . .	95
7.2.1	Active Memory Sharing configuration . . . . .	95
7.2.2	Shared memory partition profile creation . . . . .	96
7.3	Monitoring the environment after the migration . . . . .	99
7.3.1	Monitoring memory usage after the migration . . . . .	99
7.3.2	Monitoring paging activity . . . . .	99
7.4	Migration checklist . . . . .	100

<b>Glossary</b> .....	103
<b>Related publications</b> .....	105
IBM Redbooks .....	105
Other publications .....	105
Help from IBM .....	106

Archived

Archived



# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.


# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Active Memory™  
AIX®  
IBM®  
iSeries®  
Micro-Partitioning™  
POWER Hypervisor™

Power Systems™  
POWER6®  
POWER7™  
PowerHA™  
PowerVM™  
POWER®

Redbooks®  
Redpaper™  
Redbooks (logo) ®  
System i®

The following terms are trademarks of other companies:

UNIX is a registered trademark of The Open Group in the United States and other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

This IBM® Redpaper™ publication introduces PowerVM™ Active Memory™ Sharing on IBM Power Systems™ based on POWER6® and later processor technology.

Active Memory Sharing is a virtualization technology that allows multiple partitions to share a pool of physical memory. This is designed to increase system memory utilization, thereby enabling you to realize a cost benefit by reducing the amount of physical memory required.

The paper provides an overview of Active Memory Sharing, and then demonstrates, in detail, how the technology works and in what scenarios it can be used. It also contains chapters that describe how to configure, manage, and migrate to Active Memory Sharing based on hands-on examples.

The paper is targeted to both architects and consultants who need to understand how the technology works to design solutions, and to technical specialists in charge of setting up and managing Active Memory Sharing environments.

For performance-related information, see:

<ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/pow03017usen/POW03017USEN.PDF>

## The team who wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Allyson Brito** is a senior IT Specialist working in Integrated Technology Delivery in IBM Brazil. He has 12 years of experience in the UNIX and Linux field 5 years at IBM. He holds a Bachelor's degree in Electronic Engineering from UFPA. His areas of expertise include Power Systems, Linux, clustering, virtualization, cloud computing, and database administration. He is a Certified Advanced Technical Expert for Power Systems AIX® (CATE) and a Redhat Certified Engineer (RHCE).

**Loïc Fura** is an IT Specialist working in the Product and Solution Support Center in IBM France. He has 3 years of experience in performance benchmarking on Power Systems. He has worked at IBM for 4 years. He holds a Master's degree in Computer Science, Mathematics and Statistics from Universite Montpellier II. His areas of expertise include Power Systems, PowerVM, AIX, Linux, VIOS, SAN, and database administration and tuning.

**Bartłomiej Grabowski** is an iSeries® Senior technical specialist in DHL IT Services Europe in the Czech Republic. He has 6 years of experience with IBM i. He holds a Bachelor's degree in Computer Science from Academy of Computer Science and Management in Bielsko-Biala. His areas of expertise include IBM i administration, PowerHA™ solutions based on hardware and software replication, Power Systems hardware, IBM i virtualization-based PowerVM and VIOS. He is an IBM Certified System Administrator.

Thanks to the following people for their contributions to this project:

### IBM US

Mala Anand, David J. Bennin, Edward B. Boden, Chang (Eddie) W. Chen, Ping Chen, Richard Conway, Jim Fall, Veena Ganti, Travis W. Haasch, David A. Hepkin,

Carol B. Hernandez, Wayne Holm, Robert C. Jennings, Ann S. Lund, Naresh Nayar, Ed Prosser, Morgan J. Rosas, Steven E. Royer, Carolyn Scherrer, Kate Tinklenberg, Mark VanderWiele, Ken Vossen, Patricia Y. Wang, Bill Brown, Ronald Forman

**IBM India**

Saravanan Devendra, Kiran Grover, Samvedna Jha, Gomathi Mohan, Murali Vn Sappa

**IBM Argentina**

Bruno Digiovani, Marcos Quezada

**IBM UK**

Nigel Griffiths, Chris Gibson

**IBM France**

Bruno Blanchard

**IBM Thailand**

Theeraphong Thitayanun

Thanks to the authors of the previous editions of this paper.

- ▶ Authors of the first edition, IBM PowerVM Virtualization Active Memory Sharing, published in April 2009, were: Bruno Digiovani, Oliver Stadler, and Federico Vagnini.

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks® publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099

2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:  
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:  
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:  
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:  
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:  
<http://www.redbooks.ibm.com/rss.html>

Archived

# Overview

IBM PowerVM design provides industrial strength virtualization technologies for AIX, IBM i and Linux operating systems on IBM Power Systems. PowerVM allows a broad set of configuration choices that enable increased asset utilization, enhanced infrastructure flexibility, and reduced cost.

Active Memory Sharing is an IBM PowerVM advanced memory virtualization technology that provides system memory virtualization capabilities to IBM Power Systems, allowing multiple logical partitions to share a common pool of physical memory.

The physical memory of an IBM Power System can be assigned to multiple logical partitions in either a dedicated or shared mode.

- ▶ The system administrator has the capability to assign some physical memory to a logical partition and some physical memory to a pool that is shared by other logical partitions.
- ▶ A single partition can have either dedicated or shared memory.

- Dedicated mode

The IBM Power System platform has been supporting the dedicated memory mode for years. It is the system administrator's task to optimize available memory distribution among logical partitions. If a logical partition's performance is impacted due to memory constraints and other logical partitions have unused memory, the administrator can manually react by issuing a dynamic memory reconfiguration. Reconfiguration is then subject to free memory availability and administrator reaction time.

- Shared mode

When using shared memory mode, it is the system that automatically decides the optimal distribution of the physical memory to logical partitions and adjusts the memory assignment based on demand for memory pages. The administrator just reserves physical memory for the shared memory pool and assigns logical partitions to the pool.

Active Memory Sharing can be exploited to increase memory utilization on the system either by decreasing the system memory requirement or by allowing the creation of additional logical partitions on an existing system.

Cloud Computing environments can benefit from the memory overcommitment provided by Active Memory Sharing to provision more virtual servers in the cloud, increasing the global system scalability and resource utilization.

This chapter provides an introduction to Active Memory Sharing technology with a high level description of components and functions.

## 1.1 Requirements

The minimum system requirements to use the Active Memory Sharing feature of IBM PowerVM are identified in Table 1-1.

*Table 1-1 System requirements*

	POWER6	POWER7™
Firmware	340_075	710_043
PowerVM Edition	Enterprise Edition	Enterprise Edition
Management console	HMC 7.3.4 SP3	HMC 7.7.2 SP1
Virtual I/O Server version	2.1.0.1-FP21	2.1.3.10-FP23
AIX version	AIX 6.1 TL 3 or later	AIX 6.1 TL 4 or later
IBM i version	IBM i 6.1.1 + latest cumulative PTF package	IBM i 6.1.1 + latest cumulative PTF package
Linux version	Novell SuSE SLES11 kernel 2.6.27.25-0.1-ppc64 or later	Novell SuSE SLES11 kernel 2.6.27.25-0.1-ppc64 or later
I/O devices	All I/O devices must be virtualized by VIOS	All I/O devices must be virtualized by VIOS

## 1.2 Dedicated and shared memory

An IBM Power System server can host multiple independent operating systems, each using a subset of system resources. The operating system runs inside a logical partition that provides access only to the resources configured by the system administrator, such as processors, memory, and I/O.

System resources can be either dedicated to a single logical partition or shared among a subset of logical partitions. The choice depends on several considerations that include performance expectations, global resource optimization, and cost. Typically, a single system is configured with both dedicated and shared resources.

A logical partition has exclusive access to all its dedicated resources. This setup may offer performance advantages on resource access time, but with a trade off to resource utilization that highly depends on the logical partition's load. On the server, there may be logical partitions with high stress on their dedicated resources and reduced performance, while other logical partitions have a very low usage of their resources.

Resource sharing allows multiple logical partitions to access the same resource under the control of a hypervisor that monitors load, applies allocation rules, and then time shares the access to the resource. The single logical partition treats the shared resource as though it



had complete access to it. It is the hypervisor that manages the real access, avoiding conflicts or interferences, and allowing access to those logical partitions that have the highest resource requirements.

For example, the Micro-Partitioning™ feature of PowerVM is widely used to share processors. An administrator can define a pool of physical processors and logical partitions can be created with a set of virtual processors and pool access rules. The system hypervisor assigns physical processors to virtual processors for a period of time that depends on access rules and the load of all logical partitions. The assignment is transparent to the operating system that assigns threads to virtual processors as though they were physical processors.

The Active Memory Sharing feature of PowerVM allows sharing of system memory. In addition to traditional dedicated memory assignments to single logical partitions, the administrator has the choice of creating a memory pool that can be shared among a set of logical partitions.

Each logical partition in the system can be configured to have either dedicated or shared memory, as shown in Figure 1-1.

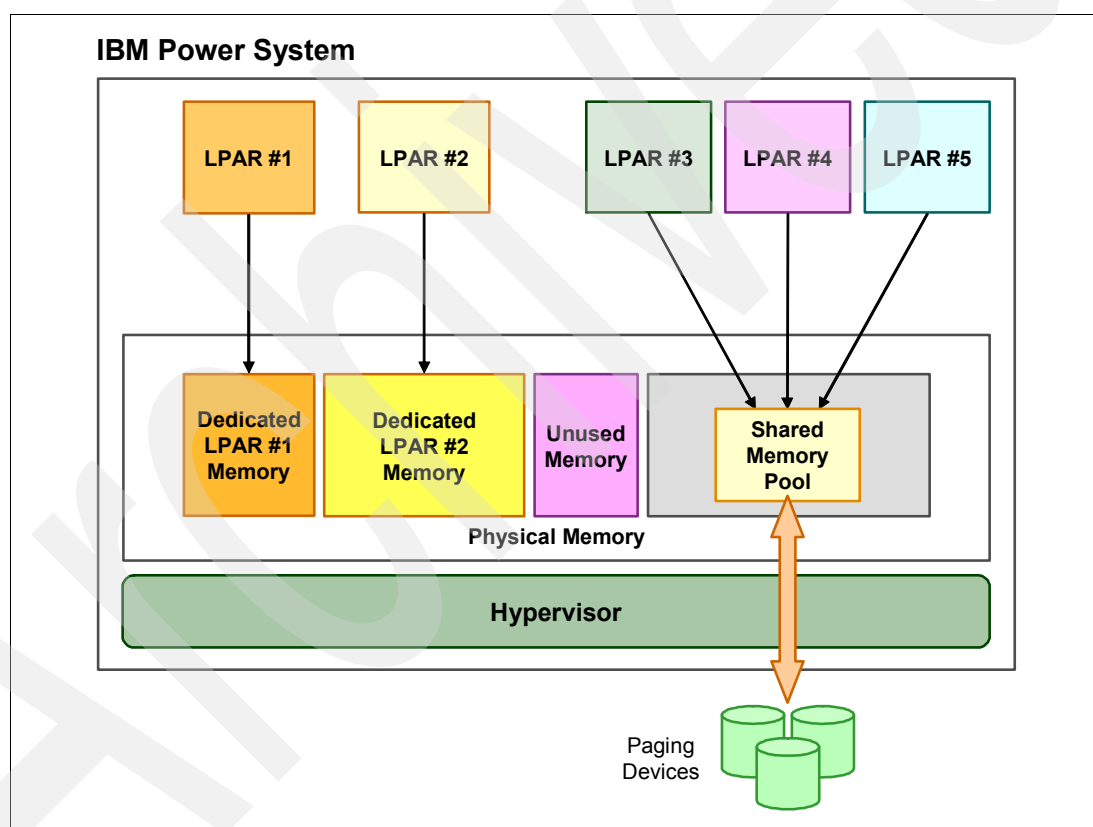


Figure 1-1 Shared and dedicated memory logical partitions

Dedicated memory partitions have system memory reserved based on their configuration. Memory size is a multiple of the system's *logical memory block size* (16, 32, 64, 128, or 256 MB) and it is all allocated for the logical partition as soon as it is activated.

Shared memory partitions are configured with a *logical memory space* that is a multiple of the system's logical memory block size, but physical memory is allocated by the Power Hypervisor from the shared memory pool based on the logical partition's runtime memory requirements. Memory allocation is made with a very fine granularity that depends on the hypervisor's page size, which is currently 4 KB.

Active Memory Sharing allows over-commitment of memory resources. Because Logical memory is mapped to physical memory based on logical partitions' memory demand, the sum of all logical partitions' logical memory can exceed the shared memory pool size; this is why paging devices are needed.

Each logical partition is allowed to use all assigned logical memory. When the cumulative usage of physical memory reaches the pool's size, the hypervisor can transparently steal memory from a shared memory partition and assign it to another shared memory partition. If the removed memory page contains data, it is stored on a paging device and the memory page content is cleared before it is assigned to another logical partition. If the newly assigned page contained data, it is restored from the disk device.

Since paging disk activity has a cost in terms of logical memory access time, the hypervisor keeps track of memory usage to steal memory that will likely not be used in the near future. The shared memory partition's operating system cooperates with the hypervisor by providing hints about page usage and by freeing memory pages to limit hypervisor paging.

## 1.3 Shared memory pool

A system enabled for the Active Memory Sharing feature can be configured with a single shared memory pool. The pool is created using the Management Console.

The shared memory pool is a collection of physical memory blocks that are managed as a whole by the hypervisor. The memory in the pool is reserved upon creation and it is no longer available for allocation to other dedicated memory partitions. The shared memory pool is directly managed by the hypervisor for exclusive use by shared memory partitions.

For example, on a system with 16 GB of real memory, you can create a 10 GB shared memory pool. The remaining 6 GB of memory will be available to create dedicated memory partitions. The 10 GB of memory for the shared memory pool is reserved immediately, even if no shared memory partitions are defined or activated.

The size of the shared memory pool can be dynamically changed at any time by using the Management Console. The pool can grow up to the maximum system memory available for logical partition use and it can be reduced to provide additional memory to dedicated memory partitions.

If no shared memory partitions are active, the pool can also be dynamically deleted, but this action will prevent the defined shared memory partitions from being activated again in shared memory mode.

The shared memory pool configuration requires the definition of a set of *paging devices* that are used to store excess memory pages on temporary storage devices. Access to the paging devices associated with a shared memory partition is provided by a paging Virtual I/O Server on the same system. At the time of pool creation, the paging Virtual I/O Servers that will provide paging service to the pool must be identified.

Each shared memory partition requires a dedicated paging device in order to be started. Paging device selection is made when a shared memory partition is activated, based on the availability and the size of the maximum logical memory configuration of the logical partition. If no suitable paging device is available, activation will fail with an error message providing the required size of the paging device.

Paging devices can be dynamically added or removed from the shared memory pool configuration. Device deletion is allowed only if it is not assigned to any running logical

partition. If the logical partition is activated after device removal, a new paging device is selected from the available set.

If no shared memory pool is available on the system, it is not possible to define any new shared memory partitions. Once the pool is available, it supports up to 1000.

**Note:** The number of partitions depends on the type and model of your machine. For the latest information, see the following resource:

<http://www.ibm.com/systems/power/hardware/reports/factsfeatures.html>

## 1.4 Paging Virtual I/O Server

When the hypervisor needs to free memory pages in the shared memory pool, the content of the memory must be stored on a paging device to be restored later when the data is accessed again. This activity is referred to as *paging activity* and is performed by the Virtual I/O Server defined for paging in the shared memory pool's configuration.

Multiple Virtual I/O Server logical partitions can be present on a system, but only two can be used by a single shared memory pool. The paging Virtual I/O Server must be configured with dedicated memory since it is providing services to the pool itself.

The paging Virtual I/O Server can handle up to 1000 shared memory partitions.

When the hypervisor decides that paging activity has to be performed, it sends a request to the paging Virtual I/O Server to copy a specific memory page belonging to a specific logical partition to or from the corresponding paging device. The paging Virtual I/O Server performs the action and notifies the hypervisor upon completion. Multiple paging requests can be issued at the same time.

A separate paging device is required for each active shared memory partition and it can be any one of the following:

- ▶ Logical volume
- ▶ Locally attached storage
- ▶ SAN attached storage
- ▶ iSCSI attached storage

On Management Console managed systems, the user can assign up to two paging VIOS partitions to a shared memory pool to provide multipath access to the paging devices. This redundant paging VIOS configuration improves the availability of the shared memory partitions in the event of a planned or unplanned VIOS outage.

When you configure redundant paging devices using dual paging VIOS, the devices must have the following characteristics:

- ▶ Physical volumes
- ▶ Located on SAN
- ▶ Must be accessible to both paging VIOS partitions

The selection of the paging device should take into account the response time that the device can provide. When paging activity is required, a shared memory partition's logical memory access time depends on the disk device response time, so it is important to use high performance and highly reliable devices.

**Note:** When using the Integrated Virtualization Manager (IVM), all paging devices are automatically created as logical volumes. Therefore, if you plan to use physical devices you have to manually perform modification using the command line interface.

## 1.5 Shared memory partitions

Shared memory partitions can be created on a system as soon as the shared memory pool has been defined. In order to be defined as a shared memory partition, the logical partition must meet the following requirements:

- ▶ Use shared processors.
- ▶ Use virtual I/O, including any of the following:
  - Virtual Ethernet adapters
  - Virtual SCSI adapters
  - Virtual Fibre Channel adapters
  - Virtual serial adapters

**Note:** All virtual adapters must be provided through a Virtual I/O Server only; the Virtual I/O Server must use dedicated memory.

- ▶ The operating system running in the logical partition can be either AIX, IBM i, or Linux, with the following minimum version levels:
  - AIX version 6.1 TL 03 on POWER6 and 6.1 TL 04 on POWER7 or later
  - IBM i version 6.1.1 or later
  - Novell SuSE SLES11kernel 2.6.27.25-0.1-ppc64 or later

**Note:** For the latest information on Linux support, see the following resources:

- ▶ IBM PowerVM Active Memory Sharing Performance  
<ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/pow03017usen/POW03017USEN.PDF>
- ▶ Using Active Memory Sharing on SLES11  
<http://www.ibm.com/developerworks/wikis/display/LinuxP/Using+Active+Memory+Sharing+on+SLES11>

An existing dedicated memory partition can be changed to shared memory mode by modifying its partition configuration and restarting the logical partition with this new configuration. Since shared memory partitions do not allow dedicated adapters, accurate planning is required.

### 1.5.1 Logical and physical memory

The memory section of a logical partition's profile has been enhanced to allow selection of either a dedicated or shared memory. The memory assigned to the logical partition is then defined as either *Dedicated Memory* or *Logical Memory*.

On a shared memory partition, two parameters define the memory configuration:

<b>Logical memory</b>	Quantity of memory that the operating system manages and can access. Logical memory pages that are in use may be backed up by either physical memory or a pool's paging device.
<b>Memory weight</b>	Relative number used by the hypervisor to prioritize the physical memory assignment from the shared memory pool to the logical partition. A higher value increases the probability that more physical memory is assigned to the logical partition.

As with dedicated memory partitions, shared memory partitions have minimum, desired, and maximum memory configuration values. In the case of a shared memory partition, these values control the logical memory size of a partition. When a partition is started, the hypervisor assigns an actual logical memory value that is equal to the desired value. The amount of physical memory a partition can use is limited by the amount of logical memory configured to a partition. For example, if a partition is configured with 10 GB of logical memory, the maximum amount of physical memory the partition can use is 10 GB, although the partition may use less than this amount.

If a logical partition's memory requirement changes with time, it is possible to dynamically modify the size of the logical memory assigned, provided that minimum and maximum logical memory limits are satisfied. For example, a logical partition with 4 GB of logical memory may host additional applications that require more working memory: the size of logical memory can then be increased to 8 GB only if the maximum logical memory size is greater or equal to 8 GB.

Logical memory content placement is under the control of the hypervisor that decides if it should be stored into physical memory or into a paging space device. The hypervisor manages the shared memory pool content by assigning physical memory to logical partitions and storing some logical memory on disk using the paging Virtual I/O Server.

When a virtual processor belonging to a shared memory partition references a logical memory page that is not backed up by a physical memory page, a *page fault* is issued at the hypervisor level and the virtual processor is suspended by the hypervisor, therefore freeing the physical processor it was using. The virtual processor is made runnable again only after the hypervisor has made the referenced page available.

Hypervisor page stealing is transparent from the operating system perspective but it may increase memory access time when the hypervisor needs to restore memory content from the paging space. The effect on performance may vary depending on which pages are chosen to be freed and how much memory each logical partition is actively accessing.

## 1.5.2 Memory classification

In order to globally optimize shared memory usage, there is cooperation among the hypervisor and the operating systems hosted by each shared memory partition.

The operating system knows the importance of logical memory pages and provides hints on page usage. It tags each logical memory page to indicate how vital it is, allowing the hypervisor to prioritize the pages that have a lower impact on the shared memory partition's performance.

### ***Loaned pages***

The loaning process allows improved cooperation between the hypervisor and each operating system. Instead of performing only page stealing, the hypervisor also requests to free some logical memory pages and the operating system can choose which pages are more

appropriate to free. By loaning pages, the operating system reduces the activity of the hypervisor, improving overall performance of the memory pool.

The AIX operating system makes it possible to tune the algorithm that selects the logical pages to be loaned. For a detailed description, see “Loaning” on page 88.

Figure 1-2 shows an example of logical to physical mapping made by the hypervisor at a given time. The shared memory partition owns the logical memory and provides the classification of page usage (page hints) to the hypervisor. The physical memory usage is managed exclusively by the hypervisor based on the current memory demand of the entire pool and on how the shared memory partition accesses the memory.

While I/O mapped pages are always assigned physical memory, all other pages may be placed either in physical memory or on the paging device. Free and loaned pages have no content from the shared memory partition’s point of view, and they do not need to be copied to the paging device.

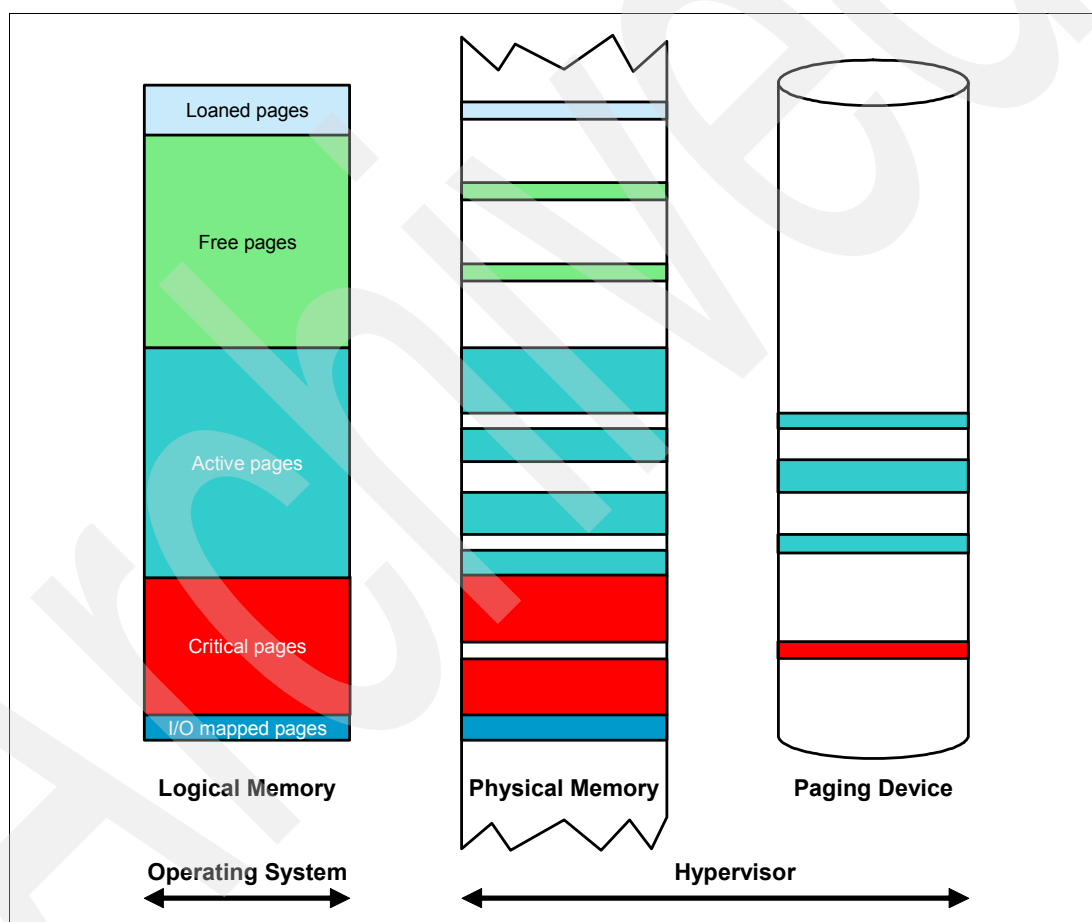


Figure 1-2 Logical to physical memory mapping example

The operating system keeps statistics of logical and physical memory usage that allow the administrator to monitor the status of the shared memory partition. In particular, it is possible to know at any moment how much physical memory is in use, how much free memory is in the loaned state, the number of hypervisor page faults related to the shared memory partition, and the average time spent dealing with such page faults.

**Note:** From a user-space application point of view, the introduction of Active Memory Sharing requires no change at all since the shared memory configuration has no effect on application behavior. Only memory access time might vary compared to that of dedicated memory, but the advantages of memory sharing can lead to better usage of system resources with significant benefits for the IT infrastructure.

## 1.6 Usage examples

The goal of memory sharing is to optimize the usage of the memory pool by assigning the physical memory to the logical partitions that need it most at a specific point in time. This optimization can be either used to reduce global memory requirements of logical partitions or to allow logical partitions to increase their memory footprint during peak memory demand periods.

Multiple memory sharing scenarios are possible, depending on the overcommitment type of physical memory. The following sections discuss the different types of memory overcommitment and their advantages with respect to workload types.

### 1.6.1 Logical memory overcommitment

In a logical overcommitment scenario, the memory sizing of the shared memory partitions is made taking into account memory demands throughout an interval, such as a day, and making sure that the global requirement of physical memory never exceeds the physical memory in the pool.

In this configuration, it is possible to optimize existing physical memory on the system or to reduce the global memory needs.

#### ***Existing memory optimization***

Consider a set of logical partitions that have been correctly sized and are using dedicated memory. The logical partition may be changed into shared logical partitions in a memory pool that contains the same amount of memory that the previously dedicated partitions occupied. The logical memory assigned to each shared memory partition is configured to be larger than the size in dedicated mode.

For example, four logical partitions with 10 GB of dedicated memory each can be configured to share a memory pool of 40 GB, each with 15 GB of logical memory assigned.

This new configuration does not change the global memory requirements, and every logical partition can have the same amount of physical memory it had before. However, memory allocation is highly improved since an unexpected memory demand due to unplanned peak of one logical partition can be satisfied by the shared pool. In deed unused memory pages from shared-memory partitions can be automatically assigned to the more demanding one automatically.

As long as the shared memory pool does not need to provide extra pages at the same moment to all logical partitions, hypervisor paging will be minimal.

If hypervisor paging activity increases too much, it is possible to add additional memory to the pool, and all shared memory partitions will take advantage of the increase in memory availability.

This differs from a dedicated memory configuration where the new memory would have to be statically assigned to only a few selected logical partitions.

### ***Reduced memory needs***

Good knowledge of physical memory requirements over time of multiple partitions allows you to configure systems with a reduced memory configuration.

For example, two logical partitions are known to require 8 GB each at peak time, but their concurrent requirement never exceeds 10 GB. The shared memory pool can be defined with 10 GB available and each logical partition is configured with 10 GB of logical memory. On a dedicated memory configuration, 16 GB of memory are required instead of the 10 GB required with the shared memory setup. This scenario is shown in Figure 1-3 with two AIX logical partitions.

The db\_server logical partition starts a job that allocates 7 GB of logical memory while the web\_server logical partition is kept idle. Partition db\_server progressively increases the real memory usage based on how the job accesses the logical memory pages. The partition web\_server decreases its real memory usage accordingly. The hypervisor asks AIX to loan memory and steals some memory pages.

When the db\_server partition ends its job, another job on the web\_server is started using 7 GB of logical memory. Following the new job memory accesses, the hypervisor starts to move memory pages from the db\_server partition that is idle and assigns them to the web\_server partition with the same loaning and stealing technique.



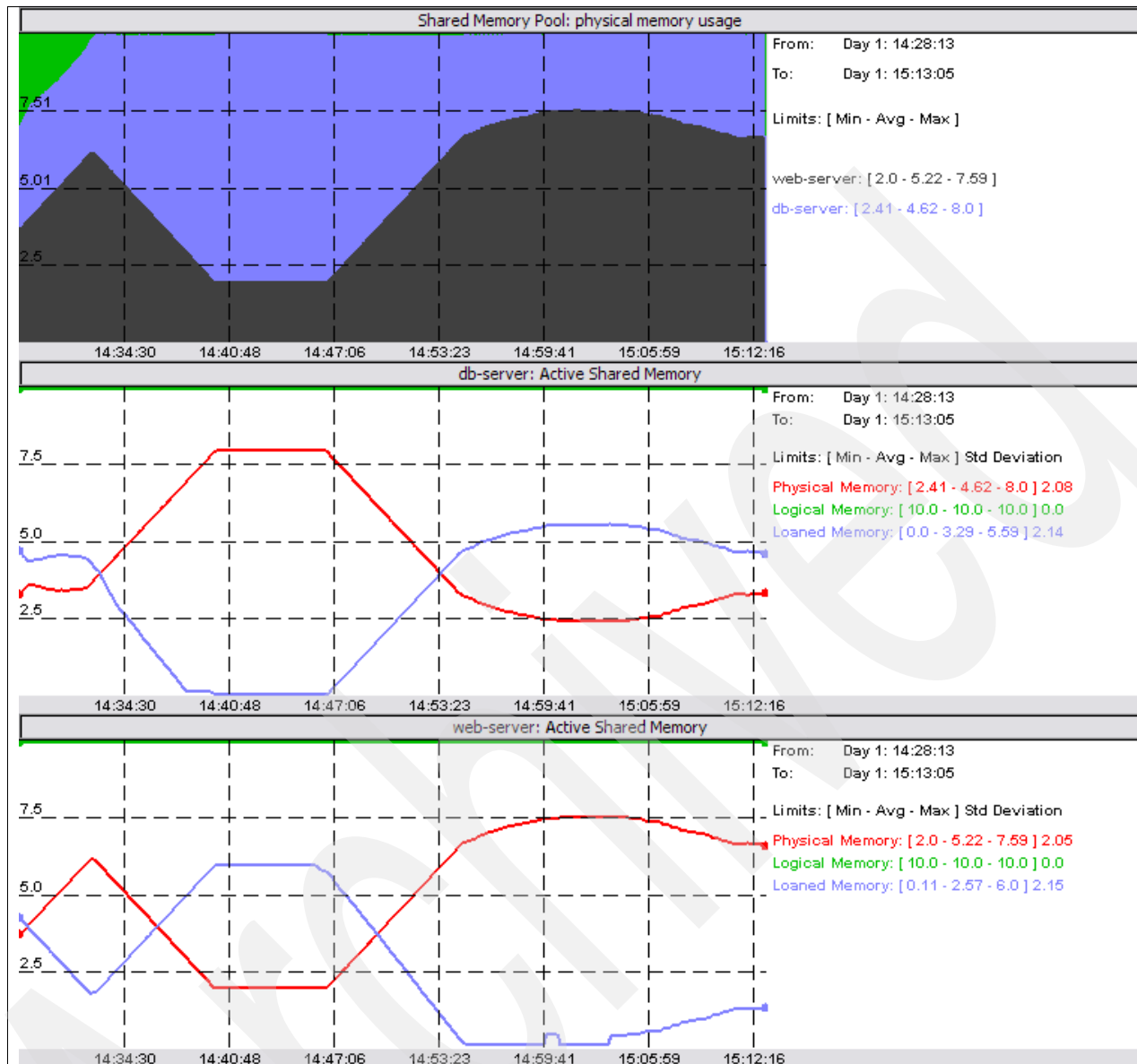


Figure 1-3 Logical overcommitment of memory example

The memory footprint required to host a set of shared memory partitions may be greatly reduced. Since shared memory partitions do not actively use all their memory at the same time, the hypervisor work is limited, and most of the stolen pages may be free pages and little I/O is required.

The hypervisor monitors memory demand on each shared memory partition and is able to meet the memory demand. When needed, page stealing is always started from the logical partitions that have lower memory activity in order to limit disk I/O operations.

The period in time when the hypervisor may have to perform high activity is during the transition of workloads from one logical partition to another. The first logical partition probably holds most of the memory even if it does not actively access it, while the second needs to allocate additional physical memory for its increased activity. During this short period, applications may monitor some increased memory access latency that is relieved as soon as the hypervisor finishes memory reallocation.

Logical overcommitment may be a good opportunity for workloads that have the following characteristics:

- ▶ They overlap peaks and valleys in memory usage. For example night and day activities or applications accessed by users in different time zones.
- ▶ They have low average memory residency requirements.
- ▶ They do not have sustained loads such as retail headquarters and university environments.
- ▶ Fail-over and backup logical partitions that are used for redundancy which require resources only when the primary server goes down. Resources do not have to be dedicated to a redundant server.

## 1.6.2 Physical memory overcommit

Physical overcommitment occurs when the sum of all logical memory that is actually being referenced at a point in time exceeds the physical memory in the shared memory pool. The hypervisor must then make frequent use of the paging devices to back up the active memory pages.

In this scenario, memory access times vary depending on whether logical pages are available in physical memory or on a paging device. The rate of hypervisor page faults determines application performance and throughput, but all logical partitions are allowed to work.

Not all workloads will be affected by memory latency, and overcommitment allows the creation of a larger number of logical partitions than with a dedicated memory configuration. There are scenarios where hypervisor paging is well suited for the configuration's needs.

Example configurations where physical overcommit may be appropriate are:

- ▶ Workloads that use a lot of file cache. The operating system can control cache size according to hypervisor requirements in order to limit hypervisor paging.
- ▶ Workloads that are less sensitive to memory latency, such as file servers, print servers, and some network applications.
- ▶ Logical partitions that are inactive most of the time.

## Detailed architecture

This chapter provides detailed information about the Active Memory Sharing architecture.

The main characteristics of the two memory models currently supported are described. The technical details about Active Memory Sharing are presented and the components and technologies involved in supporting Active Memory Sharing are discussed.

Finally, this chapter offers a comparison between PowerVM processor and memory virtualization to reinforce these concepts.

## 2.1 Dedicated versus shared memory model

This section describes the main characteristics of the two memory models currently supported.

### 2.1.1 Dedicated memory model

In the dedicated memory model, the physical memory is distributed among the partitions, with some memory dedicated to the hypervisor to maintain internal data structures. Any remaining memory goes to the free pool, as depicted in Figure 2-1. Dynamic logical partition (DLPAR) operations on dedicated memory partitions move memory among partitions or between partitions and the free memory pool.

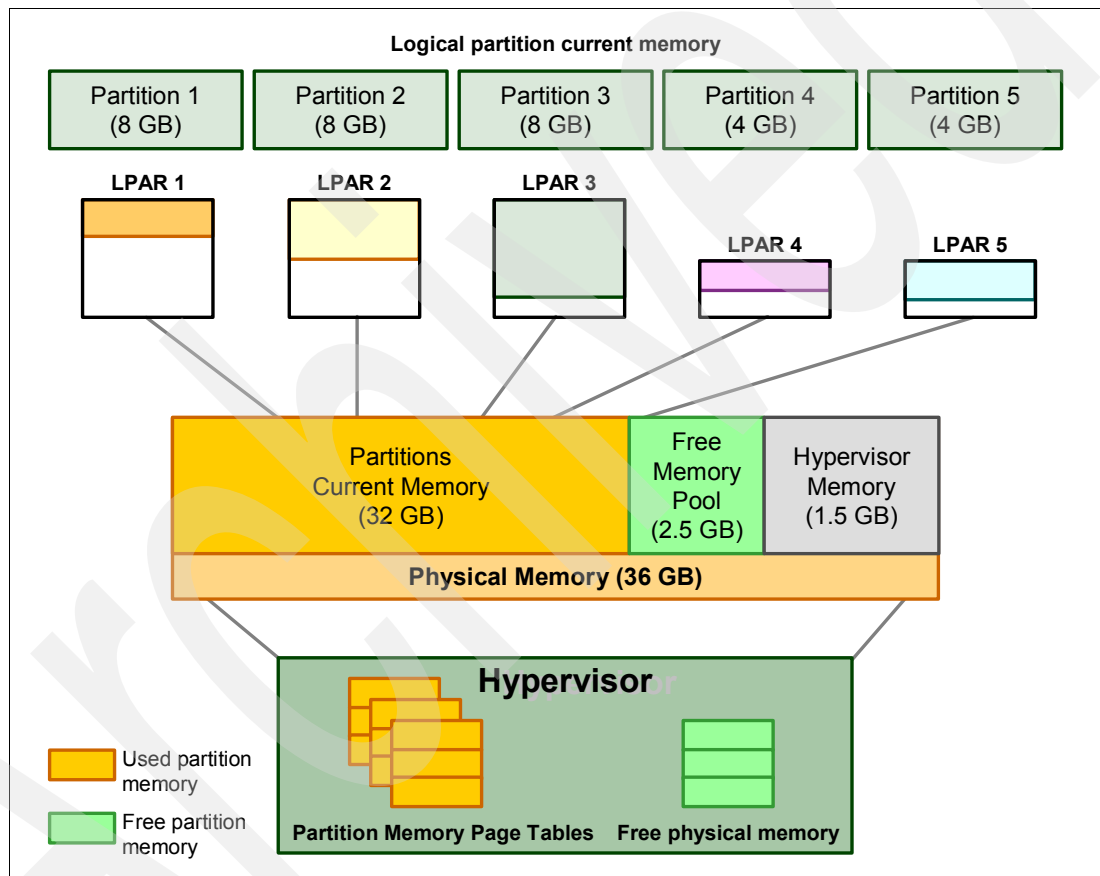


Figure 2-1 Dedicated memory model

When a partition is defined, the minimum, desired, and maximum memory for the partition are specified. The minimum memory value represents the minimum amount of memory required to activate the partition. The maximum value represents the maximum amount of logical memory that can be dynamically allocated to the partition. The desired memory is the amount of memory that will be allocated for the partition if available. It ranges between the minimum and the maximum memory values.

The configured memory is the same as the physical memory in a dedicated memory partition. The partition is allocated an amount of memory greater than or equal to the minimum value specified in the partition definition and less than or equal to the desired value specified in the partition definition, depending on the amount of physical memory available in the system.

The memory allocated to a dedicated memory partition can be changed dynamically through a dynamic LPAR operation.

## 2.1.2 Shared memory model

In the shared memory model, LPARs share memory from a single pool and LPARs' configured memory becomes logical memory.

The shared memory pool is part of the real physical memory that is virtualized by the hypervisor to allocate physical memory to the shared memory partitions and the hypervisor, as shown in Figure 2-2.

The size of the shared memory pool can be changed by:

- ▶ Adding memory from the free pool
- ▶ Shrinking and moving (reallocation) of memory from dedicated memory partitions

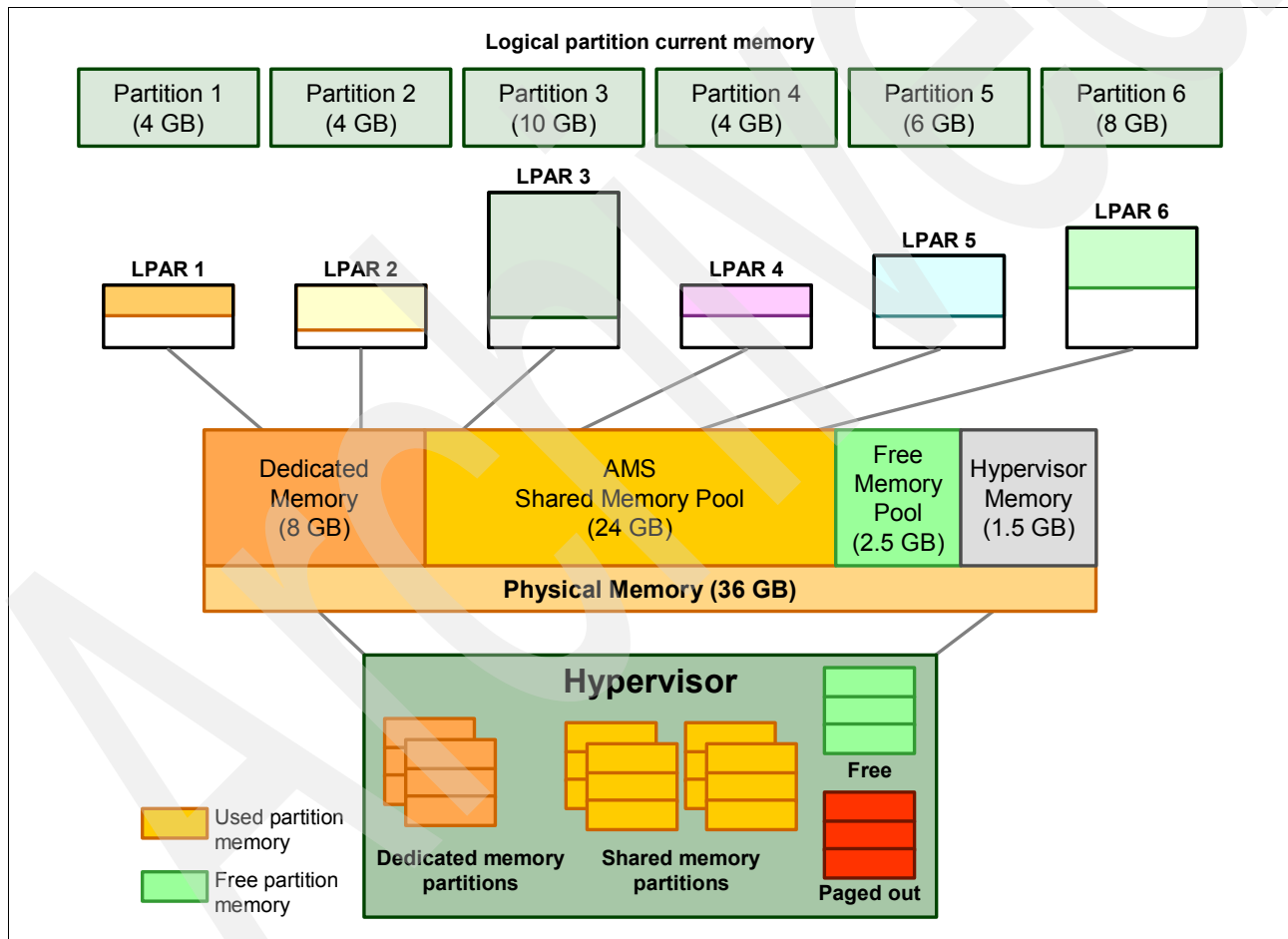


Figure 2-2 Shared memory model

When a shared memory partition is defined, the minimum, desired, and maximum logical memory for the partition are also specified. In addition to these values, you can change the shared memory weight by editing the partition profile. The memory weight is one of the factors taken into consideration when the hypervisor decides which partition receives the physical memory.

The Management Console calculates the I/O entitled memory based on the I/O configuration. The I/O entitled memory is the maximum amount of physical memory guaranteed to be available for I/O mapping.

**Important:**

- ▶ You can change the I/O entitled memory value using a dynamic LPAR operation. Normally, you do not need to change this value unless monitoring tools are reporting excessive I/O mapping failure operations.
- ▶ The activation of a shared memory partition fails if there is not enough physical memory in the shared pool to satisfy the I/O entitled memory capacity needed for the partition's I/O configuration. To see the required amount of I/O memory assignment see Figure 5-3 on page 64.

## 2.2 Active Memory Sharing technical details

Active Memory Sharing allows selected LPARs to share memory from a single pool of physical memory. This function is supported by a new level of abstraction managed by the hypervisor.

**Important:** An LPAR's configured memory (minimum, desired, and maximum) becomes logical memory in an Active Memory Sharing environment.

Figure 2-3 on page 17 shows the components involved in Active Memory Sharing, which are described in the next sections of this chapter.

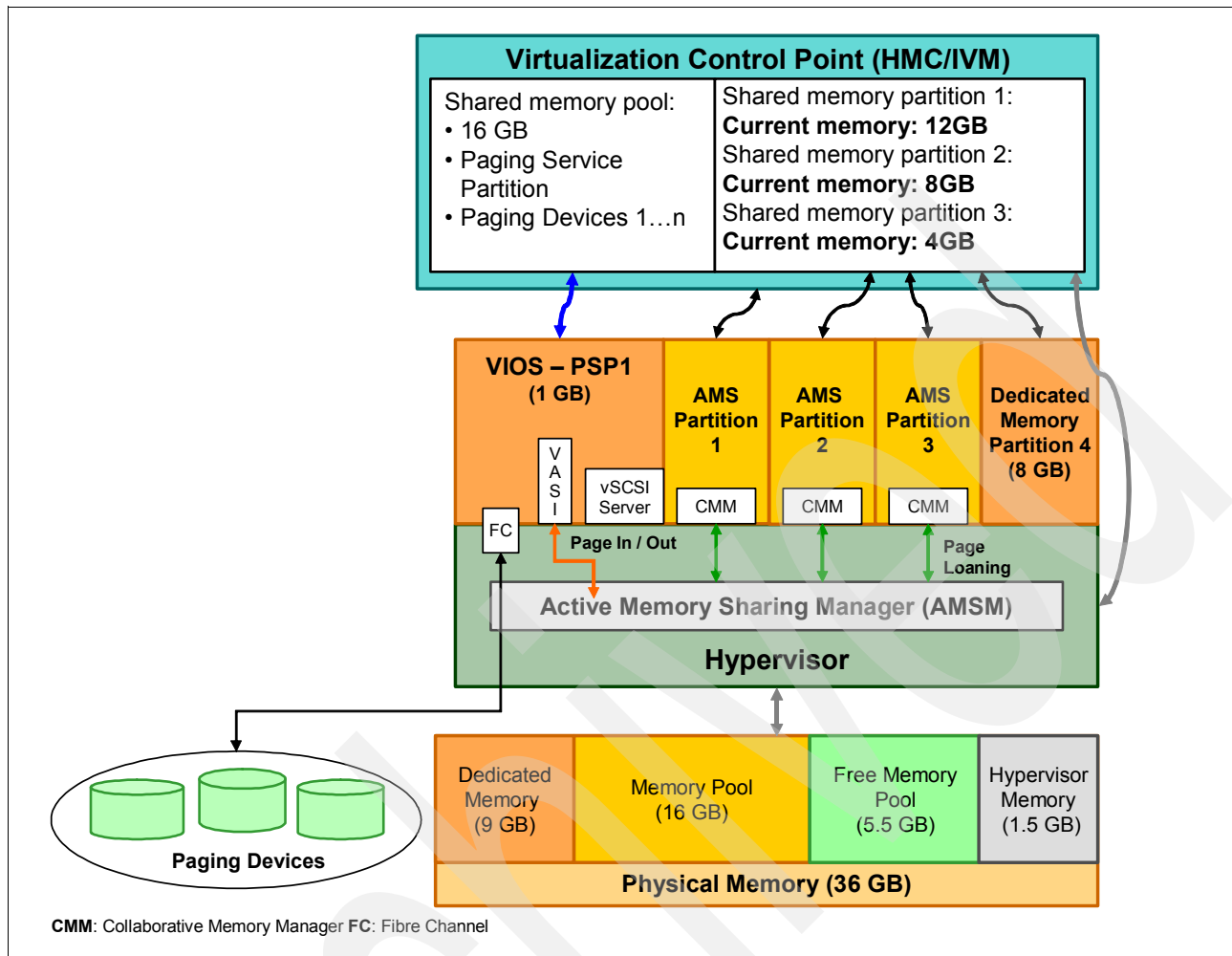


Figure 2-3 Active Memory Sharing architecture

Active Memory Sharing support is analogous to traditional operating system page-based virtual memory support where processes are presented with virtual address spaces that are divided into pages. These virtual memory pages are either mapped into physical memory (page frames) or mapped to blocks on a disk.

The Active Memory Sharing provides these same capabilities, but in the case of Active Memory Sharing, the virtual address space is the LPAR's logical memory. With Active Memory Sharing, the LPAR's logical memory is divided into virtual memory pages, and the hypervisor is responsible for mapping these virtual memory pages into physical memory or to disk as shown in Figure 2-4 on page 18.

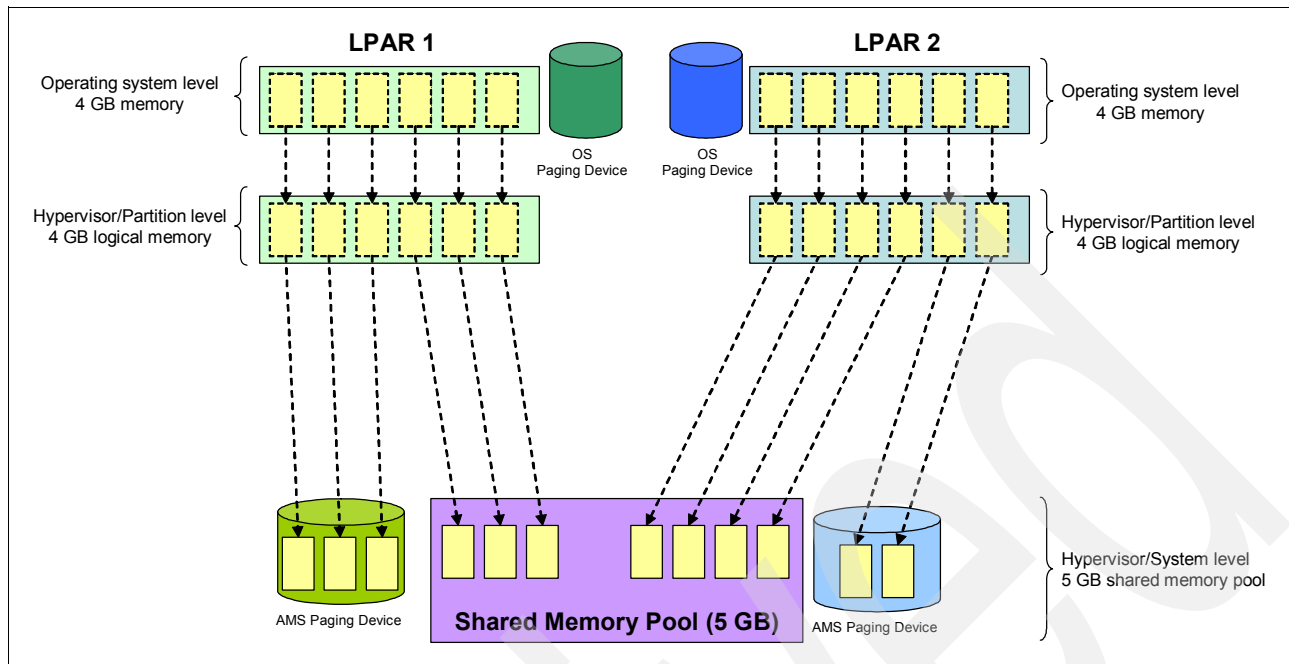


Figure 2-4 Logical memory mapping

**Note:** The real physical memory is part of the shared memory pool that is virtualized by the hypervisor to allocate resources to the shared memory partitions. So, there is no explicit binding of memory to processors (memory affinity). Also, there is no binding of physical memory to an LPAR's logical memory in an Active Memory Sharing environment.

For Active Memory Sharing, the operating system assists the hypervisor for the following requests:

- ▶ Loan memory pages
- ▶ Steal aged pages
- ▶ Save pages content to a local paging device, frees and loans it to the hypervisor

This assistance is enabled through the memory loaning function in the operating system. As described in 2.4.1, "Page loaning" on page 23, this allows an operating system to page out the aged page contents and loan them to the hypervisor to use it to expand another shared memory partition's physical memory. The reason for this is that the operating system is in a better position to determine its working set relative to the available memory.

This on-demand memory allocation enables workloads that have variable memory requirements to time-share the same memory in a consolidated environment leading to improved memory utilization, as shown in Figure 2-5 and Figure 2-6 on page 19.



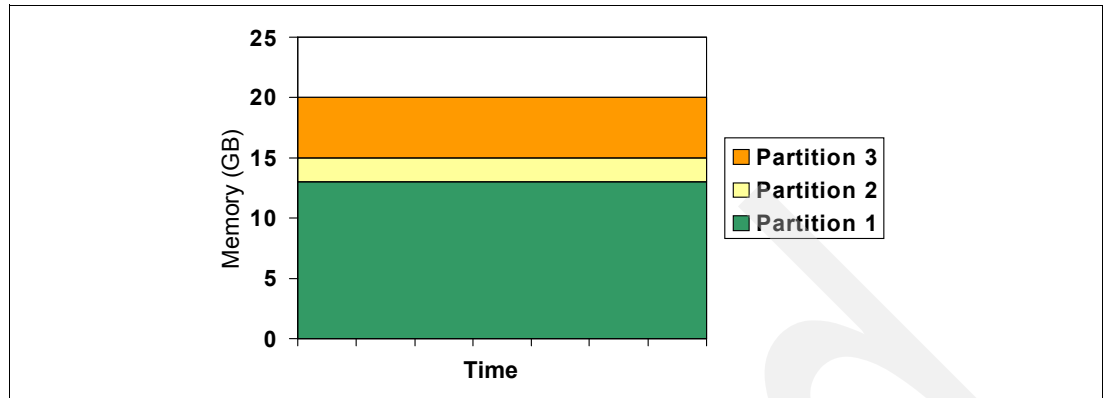


Figure 2-5 Partitions with dedicated memory

Figure 2-5 shows three partitions, each configured with dedicated memory, amounting to 20 GB in total. Even if some of these partitions are using less memory than what is assigned, the excess memory cannot be used to create another partition.

Figure 2-6 has four partitions with similarly configured memory as the partitions in Figure 2-5, but they are sharing memory and therefore improving memory utilization. In this case an additional partition can be accommodated with the same amount of memory.

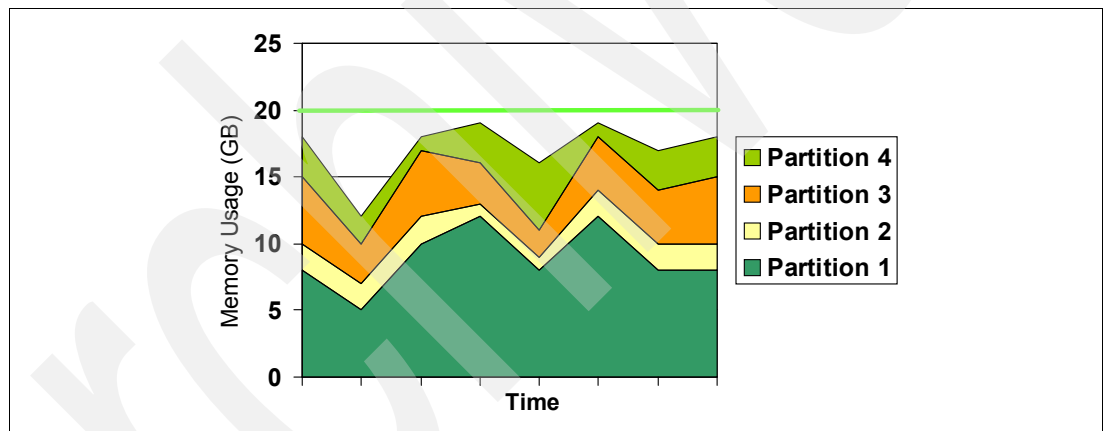


Figure 2-6 Partition with shared memory

The sum of the logical memory of all shared memory partitions in a system is allowed to exceed the real physical memory allocated to a shared memory pool in the system. The intent is to allow memory intensive shared memory partitions on a system to use portions of the physical memory not currently being used by other shared memory partitions. To do so, each shared memory partition is allowed to perceive itself as owning more of the memory, which results in total logical memory being oversubscribed.

These scenarios will be described in more detail in Chapter 3, “Planning for Active Memory Sharing” on page 33.

## 2.3 Active Memory Sharing components

This section describes the components that support Active Memory Sharing.

## 2.3.1 Virtualization Control Point

The Virtualization Control Point provides the administrator interface to the Active Memory Sharing functions and communicates with the management interfaces of the Active Memory Sharing environment.

The best way to manage virtual resources is via the Management Console or IVM.

Through a user interface, you can configure and manage a shared memory pool and configure partitions to use the memory resources from the pool.

Figure 2-7 shows part of the user interface to memory pool management in the Management Console environment.

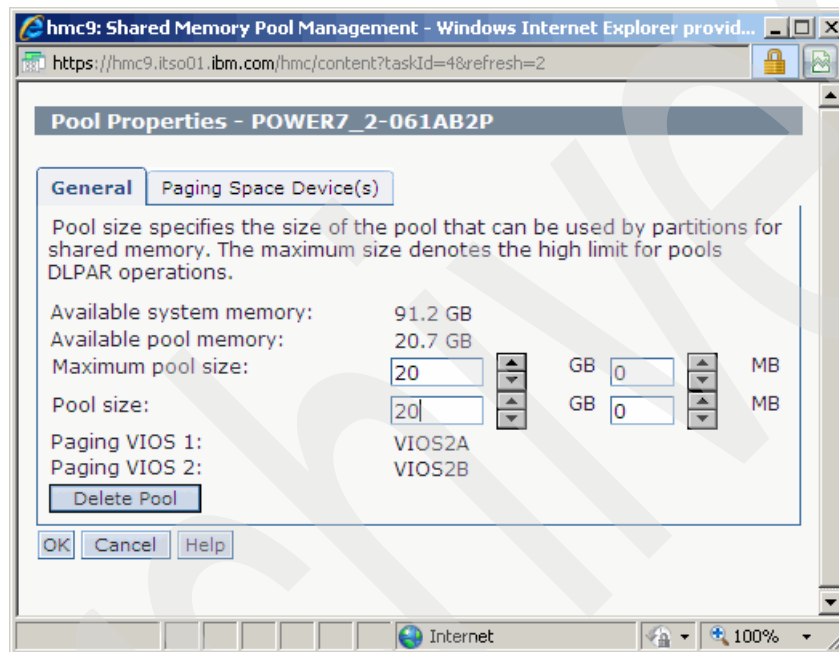


Figure 2-7 Management Console Memory Pool Management panel

## 2.3.2 Shared memory pool

The shared memory pool is a configured collection of physical memory managed by the Active Memory Sharing Manager. The system administrator determines the amount of physical memory allocated to the shared pool, in multiples of Logical Memory Blocks (LMBs). The shared memory pool must be created before a shared memory partition is created.

A Management Console user interface guides the user through the steps to select the desired and maximum shared memory pool size, the paging Virtual I/O Servers, and the paging devices for the shared memory pool.

The system administrator must select the size of the shared memory pool based on the number of shared memory partitions that will run concurrently in the pool and its workloads characteristics. The system administrator also selects the number of paging devices assigned to a shared memory pool.

**Note:** Using a Management Console, the administrator has the ability to change the shared memory pool size and change the paging devices assigned to the shared memory pool dynamically.

### 2.3.3 Active Memory Sharing Manager

The Active Memory Sharing Manager (AMSM) is a hypervisor component that manages the shared memory pool and the memory of the partitions associated with the shared memory pool. The AMSM allocates the physical memory blocks that comprise the shared memory pool.

The AMSM uses hypervisor paging to try to keep all partition working sets (pages needed by current workloads) resident in the shared memory pool in an overcommitted system. If a Collaborative Memory Manager (an OS feature that gives hints about memory page usage) is available in an operating system, it cooperates with the AMSM to free up partition pages upon request.

The Active Memory Sharing Manager keeps a list of free physical pages. This list is per memory pool and contains free pages from the partitions associated with that pool. When a page fault occurs, the AMSM assigns a free page to handle that page fault. This is done until the AMSM's free list reaches a low water mark. At that point, the Active Memory Sharing Manager takes memory from other partitions, through page stealing, based on the following:

- ▶ The hypervisor's perception of the relative memory loads (See 6.2.1, "Memory sharing policy and memory load factor" on page 86)
- ▶ The partition's shared memory weight

When a partition requires a memory page for I/O operations (I/O mapped page), the Active Memory Sharing Manager always assigns physical memory pages from the partition's entitled memory capacity.

**Note:** The flow of memory between partitions is not instantaneous. The Hypervisor must see sustained demand from a partition before giving it significantly more memory at the expense of other partitions.

**Note:** The Active Memory Sharing Manager guarantees that the contents of stolen pages are not readable to any other partitions by zeroing the contents of the page, if necessary, before the pages are allocated to another partition.

### 2.3.4 Paging Virtual I/O Server

A Paging Virtual I/O Server is a partition that provides paging services for a shared memory pool and manages the paging spaces for shared memory partitions associated with a shared memory pool.

A Virtual I/O Server enabled as a Paging Virtual I/O Server is designed to serve one shared memory pool. Using the Management Console it is possible to assign a second paging Virtual I/O Server to the shared memory pool to service the shared memory partitions in a redundant Virtual I/O Server configuration.

In Management Console managed systems, the Management Console provides a user interface to enable a Virtual I/O Server for Active Memory Sharing paging services. The Virtual I/O Server partition can be enabled for Active Memory Sharing services dynamically.

In IVM-managed systems, the Virtual I/O Server hosting the Active Memory Sharing is always enabled for paging services.

**Note:** The number of shared memory partitions depends on your server. For more information see 1.1, “Requirements” on page 2.

**Important:** Paging space devices can only be assigned to one shared memory pool at a time. You cannot assign the same paging space device to a shared memory pool on one system and to another shared memory pool on another system at the same time.

The Management Console also provides a user interface to disable Active Memory Sharing paging services on a Virtual I/O Server partition. Active Memory Sharing paging services can be disabled on a Virtual I/O Server as long as it is not currently the paging Virtual I/O Server assigned to a shared memory pool.

### 2.3.5 Paging devices

Active Memory Sharing enables dynamic memory management among multiple LPARs by allocating memory on demand. As a result, the hypervisor has to use a paging device to back up the excess memory that it cannot back up using the physical memory.

#### Operating system paging

In dedicated memory environments, AIX paging device tuning and optimizations are rarely necessary when the memory is set based on peak demand, therefore providing enough memory for the working set. This avoids paging.

With IBM i, operating system paging is different from AIX due to a single-level storage architecture. You also size the memory for peaks with IBM i, but if operating system paging occurs, the paging is not performed to a specific paging device. Single-level storage allows disk operations, such as paging, to be spread across all the disks in storage pools.

Typically, the default paging device of the primary boot disk for AIX is enough for normal paging activity, and the space needed for the configuration is significantly smaller than the memory allocated to the partition. IBM i is similar in a sense that normal paging activity is typically not an issue as long as memory is sized properly.

Operating system paging is primarily used in Active Memory Sharing for loaning pages from current working sets to the hypervisor. Therefore, if loaning is not enabled, you do not need to optimize the OS paging device.

**Important:** In an Active Memory Sharing environment, the performance of the operating system paging device takes on a different important role. There are also two levels of paging: the partition's operating system paging and hypervisor paging.

#### Hypervisor paging devices

The Active Memory Sharing hypervisor uses paging devices to manage memory across all shared memory partitions. If loaning is not enabled (which can only be accomplished in AIX), hypervisor uses the paging device to page out the contents of a physical page before allocating it to another shared memory partition. When the first shared memory partition touches the logical page whose contents are paged out, the hypervisor finds a free page and pages in the contents to service the first shared memory partition's page fault.

In the case where loaning is enabled in all the shared memory partitions on the system, there will be significantly lower levels of paging done by the hypervisor; however, there will be spikes of hypervisor paging activity.

**Important:** It is suggested that the partition devices remain separate from the hypervisor paging devices, if possible. The I/O operations from shared memory partitions might compete with I/O operations resulting from hypervisor paging.

### 2.3.6 Virtual Asynchronous Service Interface

A Virtual Asynchronous Service Interface (VASI) is a virtual device that allows communications between the Virtual I/O Server and the hypervisor.

In AMS environment, this device is used for handling hypervisor paging activity.

### 2.3.7 I/O entitled memory

The purpose of the I/O entitled memory size is to define the minimum amount of physical memory that must remain available when the partition needs it for I/O operations, no matter how much demand for more physical memory is placed by other partitions.

If the memory entitlement is too small, the effect is a failed or delayed I/O operation. For example, networking often requires a pre-reserved amount of physical memory for receiving inbound communications (such as Ethernet) packets. If the physical memory does not exist at the time these packets are being written into the partition's memory, the effect of this latency might contribute to networking time-outs. Similarly, if a disk paging operation is needed and the physical page to act as the target of a disk read cannot be pinned, the effect is added latency in the disk read until such physical memory can be pinned.

When a partition is being set up to do I/O operations (such as disk read or writes, or TCP/IP communications), a portion of the physical memory must remain unmoved during the period of those operations. The hypervisor temporarily reserves a certain number of physical pages, which it will not use for page sharing.

**Note:** The entitled memory size is not a specification of absolute physical pages but just an observation that a number of physical pages must be available for certain purposes for some potentially extended period of time.

## 2.4 Active Memory Sharing supporting technologies

Several technologies are involved in supporting active memory sharing. They are described in this section.

### 2.4.1 Page loaning

Active Memory Sharing uses memory loaning instead of memory ballooning. Using ballooning, the pages that a hypervisor wants to use are allocated by a ballooning device driver and then supplied to the hypervisor. Those pages cannot be used until they are explicitly returned by the hypervisor. This method was considered because it avoids internal operating system modifications. However, it is slow to respond to increases in operating

system memory requirements and is limited to the amount of memory that the operating system can pin.

With page loaning, the operating system responds with pages that are the least expensive to load. As the load increases, the operating system takes back these hints and the pages associated with them and provides feedback to the hypervisor about increased operating system load.

Memory loaning introduces a new class of page frames, named loaned page frames. Loaned pages are free pages from an operating system perspective, with the exception that these pages cannot be used to cache file data. They can be used for working storage data, at which point they will naturally transition out of the loaned state as part of being allocated to a system user.

If loaning is enabled, the operating system will page out contents of its pages and loan pages to the hypervisor as shown in Figure 2-8. If memory loaning is disabled, only the hypervisor will page out using the hints, we call that *page stealing* (see Figure 2-9 on page 25). Critical pages are paged out in case of a high oversubscription ratio and high pressure for memory in one or more of the shared memory partitions.

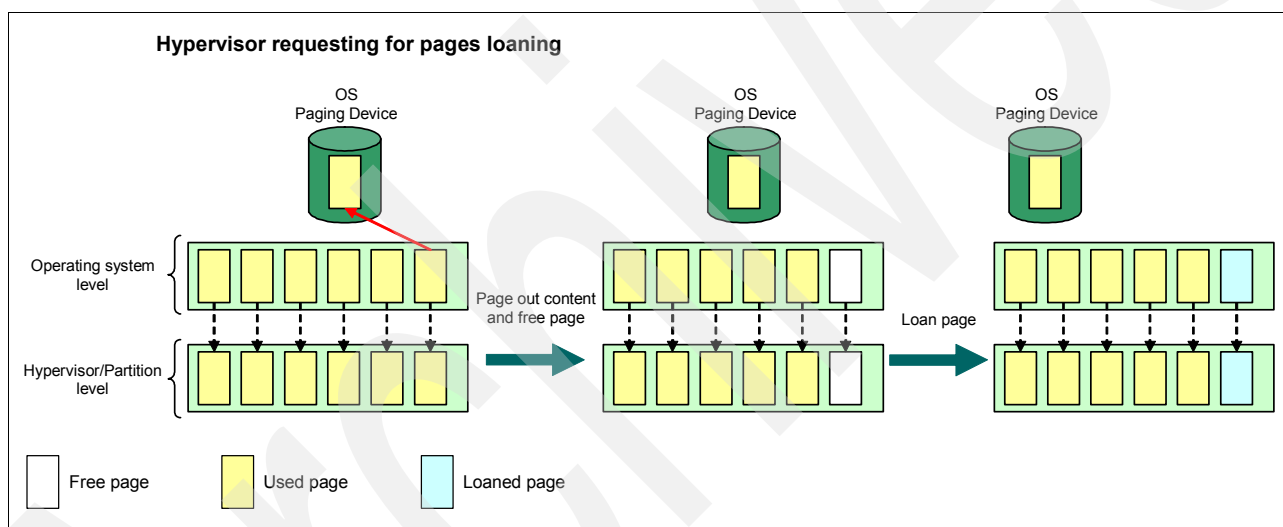


Figure 2-8 Memory loaning

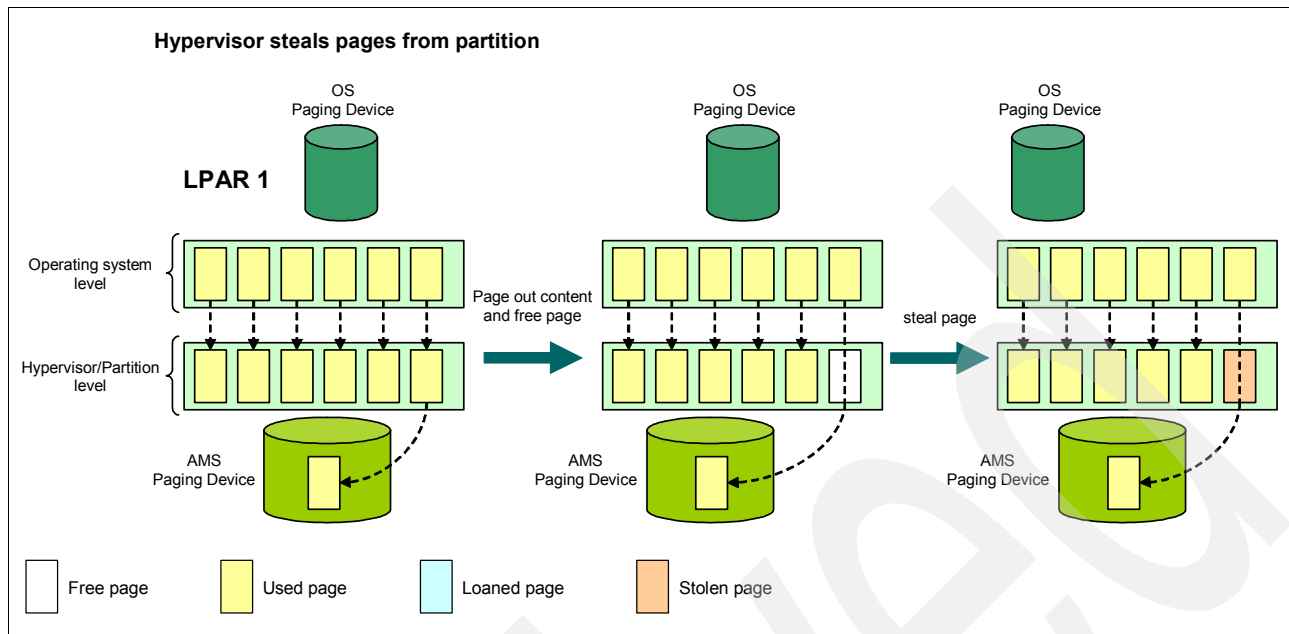


Figure 2-9 Hypervisor paging - Page stealing

## 2.4.2 Collaborative Memory Manager

The Collaborative Memory Manager (CMM) is an operating system feature that provides hints about memory page usage that the PowerVM hypervisor uses to target pages to manage the physical memory of a shared memory partition.

The Active Memory Sharing Manager, designed to manage the shared memory across all partitions, does not have a view of the hot (active) and cold (aged) pages of each partition. The hypervisor does not select logical pages from a partition if that page is part of the partition's working set. However, in the oversubscription case the hypervisor cannot avoid this, so it collaborates with the operating system to receive hints on what pages to improve the probability of selecting a logical page that will not be required by the partition in the near future.

Operating systems running in each shared memory partition, such as AIX and IBM i, use the CMM to specify hints on each logical page usage to the hypervisor.

In general, the hypervisor targets logical memory pages in increasing order of importance based on the hints.

## 2.4.3 Memory affinity

Memory affinity information is not reported by the partition firmware for a partition running in Active Memory Sharing mode.

- ▶ For AIX, all memory belongs to the same affinity domain, and a single virtual memory pool to represent all of the memory is used.
- ▶ For IBM i, the Main Storage Management (MSM) component of SLIC makes all memory appear as node 0 if the partition is running in an Active Memory Sharing pool.

## 2.4.4 Components specific to IBM i

The information in this section applies only to the IBM i operating system.

### Materialize Resource Management Data (MATRMD) Instruction

Information about the Active Memory Sharing configuration and paging activity that is provided by POWER® Hypervisor™ is available above the Machine Interface layer to higher level IBM i operating system software and applications. This information might be useful for evaluating partition performance behaviors due to memory paging activity by the hypervisor.

You can access this information using the collection services. The data about Active Memory Sharing is logged in a new table named QAPMSHRMP. For further details see 5.4, “Monitoring IBM i” on page 73.

Uses for this information gathered by the materialization options might be service and performance APIs written specifically to retrieve this information.

**Important:** You can only obtain shared memory pool data if the Management Console has authorized the partition to materialize shared pool utilization data. This is the same mechanism currently used to control partition access to shared processor pool data.

### Hypervisor CPU Dispatch PDC events

PDC events for Hypervisor CPU Dispatch activity are collected from the Dispatch Trace Log Buffers in each partition logical processor when Performance Explore (PEX) is started with a definition specifying BASEEVT(\*CPUSWT).

Dispatch Trace Log Buffer entries are created by the POWER Hypervisor when a partition logical processor is dispatched to run on a hardware processor, and among other things contains the reason for the processor preempt/dispatch.

If the partition is running in Active Memory Sharing mode, a new reason for preempting a partition logical processor will be due to a page fault when accessing logical real memory that is not resident in physical real memory. A value of 0x08 in the dispatch reason code field indicates that a logical real memory page fault caused the processor to be preempted.

**Note:** This new value in the dispatch reason field appears in the current QTWHRC field of trace records in the QAYPETSWSW PEX file.

## 2.5 Active Memory Sharing supported technologies

The Active Memory Sharing feature is compatible with most other PowerVM technologies and can also interact with some of them. This section describes the supported technologies.

### 2.5.1 Active Memory Expansion

Active Memory Expansion (AME) is a feature available on POWER7 and later POWER processors, and only for AIX. AME allows a partition to expand its memory up to a given factor. This memory expansion is obtained using in-memory data compression.

AME and AMS are distinct but compatible features related to memory virtualization. They can be used independently or together.



When configuring a partition with AMS, you define logical memory. The AME factor you set then applies on this logical memory amount.

For example, a partition defined with a desired logical memory of 4 GB and an AME factor of 2 will provide the operating system a total of 8 GB of expanded memory. The AME factor has no influence on how much physical memory from the shared memory pool the hypervisor will provide to the partition. Therefore, the total amount of memory provided by AMS (physical memory from the shared pool plus memory on the paging devices) will still be 4 GB. The remaining 4 GB are provided by in-memory compression done at the operating system level. This is illustrated in Figure 2-10.

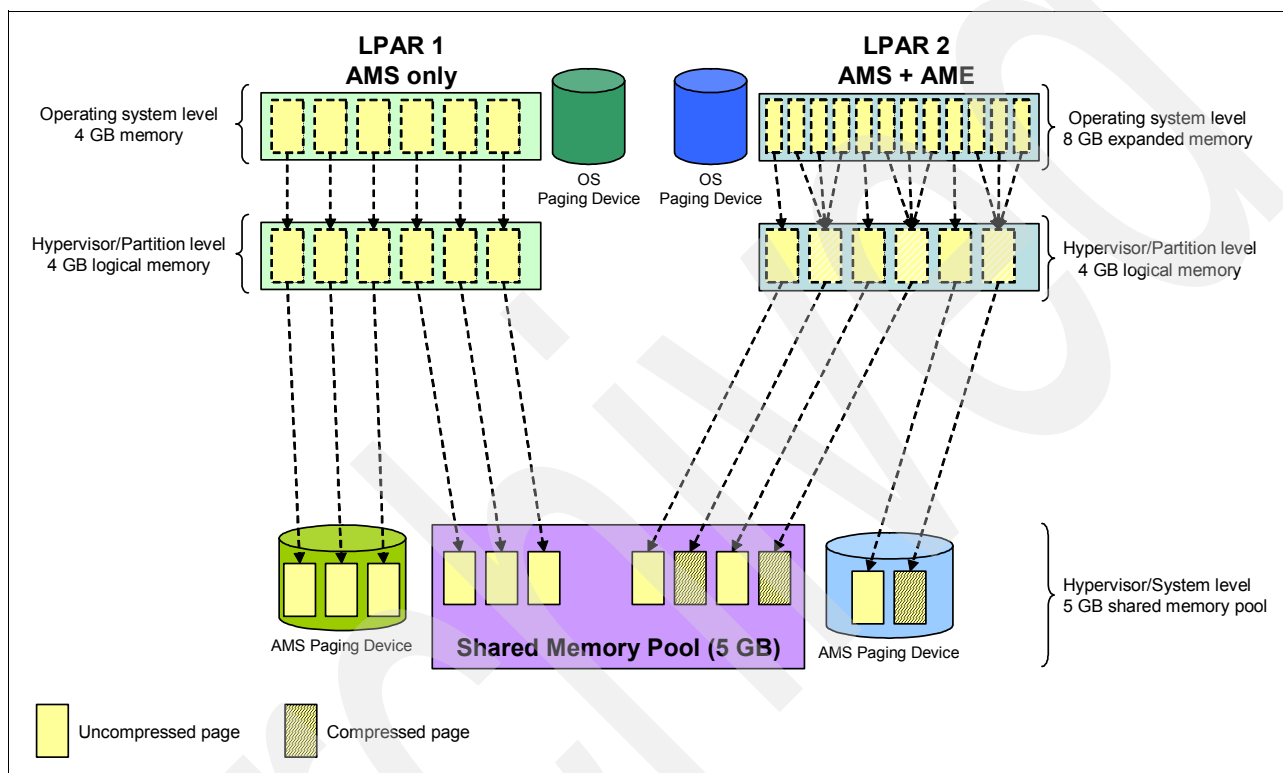


Figure 2-10 Active Memory Sharing and Active Memory Expansion

When both features are used at the same time, there are interactions between them, especially regarding the AMS page loaning technology.

The difference is summarized as follows:

- ▶ Without AME, the page loaning mechanism pages out content of its pages before loaning them to the hypervisor (Figure 2-8 on page 24).
- ▶ With AME enabled, the page loaning mechanism will first try to compress page content to free pages and loan them as illustrated in Figure 2-11 on page 28. When compression is not possible anymore, it will start paging out content of pages to loan them. If the content of the page targeted for paging is compressed, it will be page out in its compressed form on the paging device.

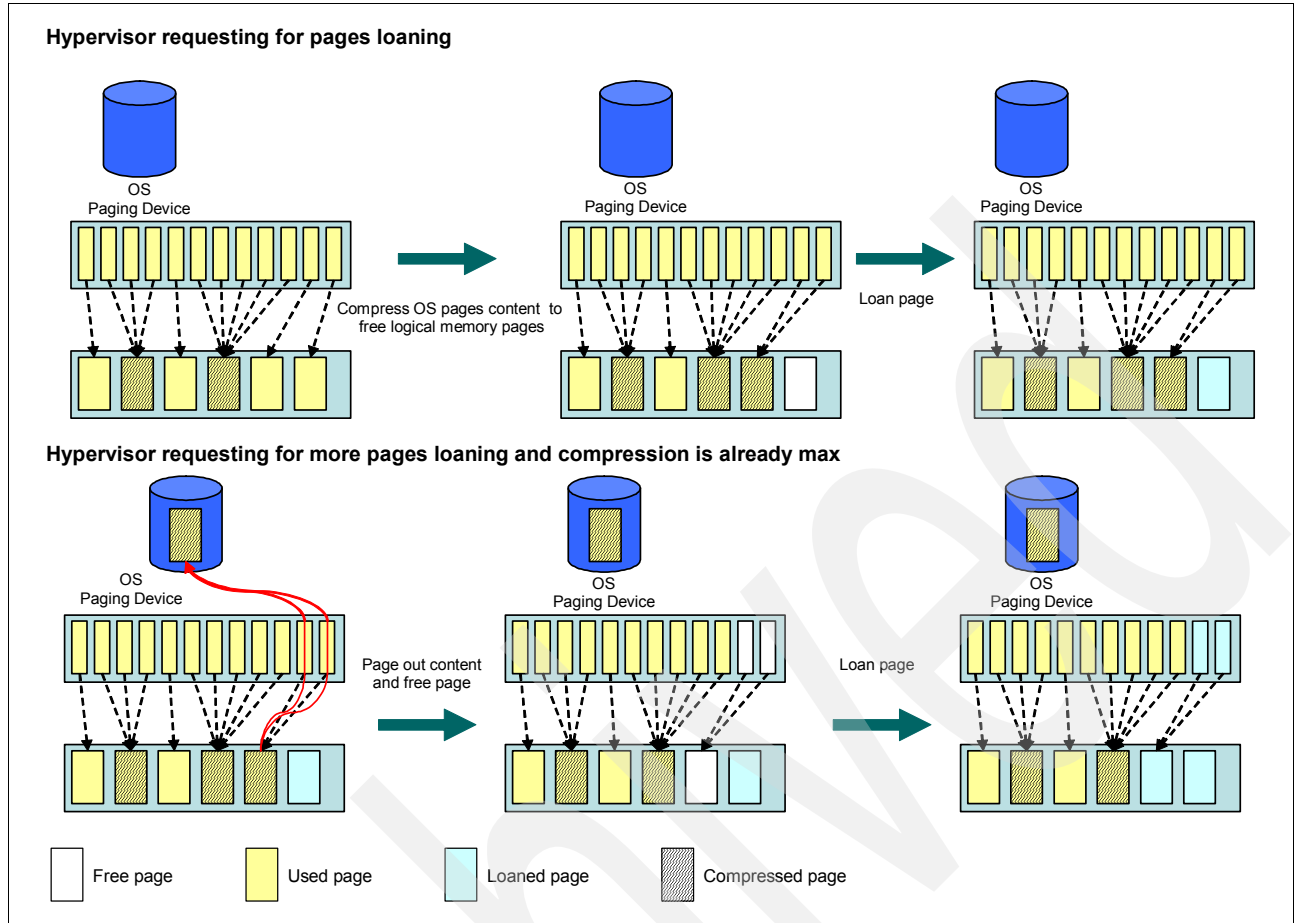


Figure 2-11 Page Loaning with Active Memory Sharing

In case of hypervisor pagination, stolen pages are stored in their compressed form on an AMS paging device as shown in Figure 2-12.

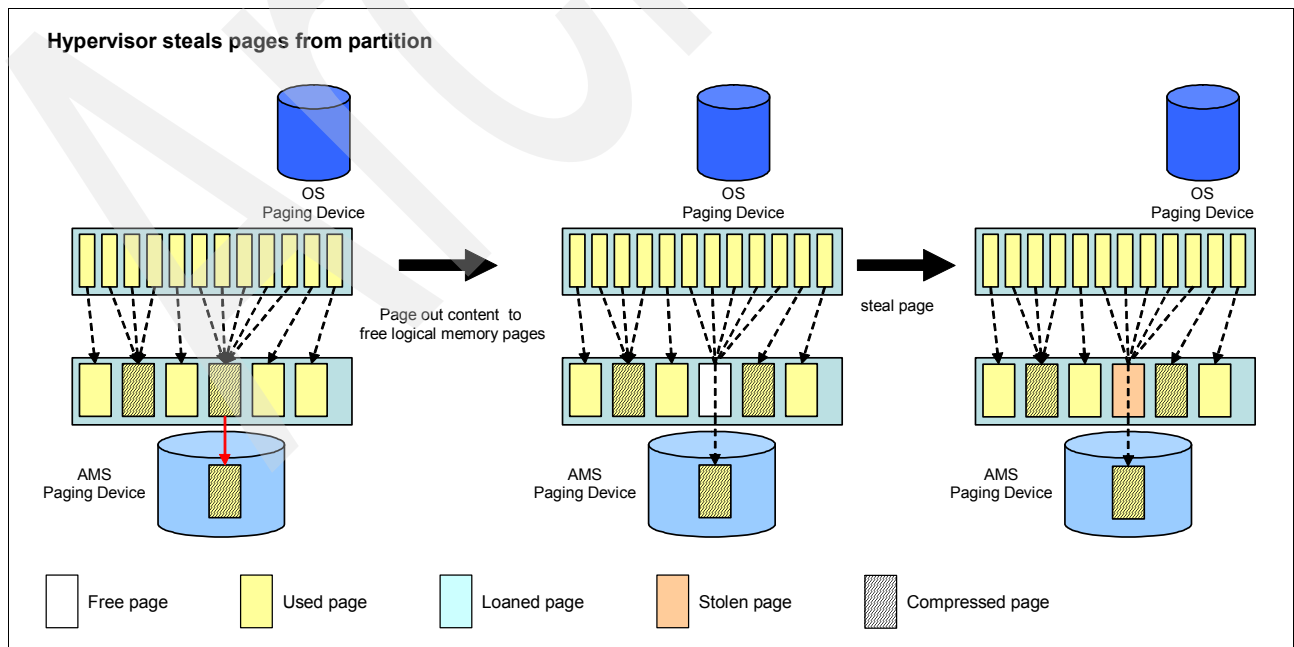


Figure 2-12 Hypervisor paging with Active Memory Sharing - Page stealing

When combined with AMS, AME will provide more memory to partitions, while AMS will improve the global system memory usage. Thus these two features help you to get the most from your memory.

## 2.5.2 Partition suspend and resume

AIX and IBM i partitions are capable of suspension. When a logical partition is suspended, the state of the logical partition is saved in persistent storage, and the server resources that were used by that logical partition are made available for use by other logical partitions. At a later time, a suspended logical partition and its applications can be resumed.

Most of the requirements for suspension are already met by shared memory partitions. For details about the requirements for suspend mode see:

<http://publib.boulder.ibm.com/infocenter/powersys/v3r1m5/index.jsp?topic=/p7hat/iphatphibreqs.htm>

In the case of AMS, the partition will use the AMS paging device to be suspended. In order to do so, the size of the paging device must be equal to or greater than 110% of the partition Maximum Logical Memory.

## 2.5.3 Live Partition Mobility

Shared-memory partitions are eligible for Live Partition Mobility as long as the target system has a Shared Memory Pool defined and an available paging device with a size at least equal to the partition Maximum Logical Memory.

In the case of dual VIOS configuration, the Management Console will prompt you to specify whether you want redundancy for the paging VIOS on the target system, as shown in Figure 2-13 on page 30.

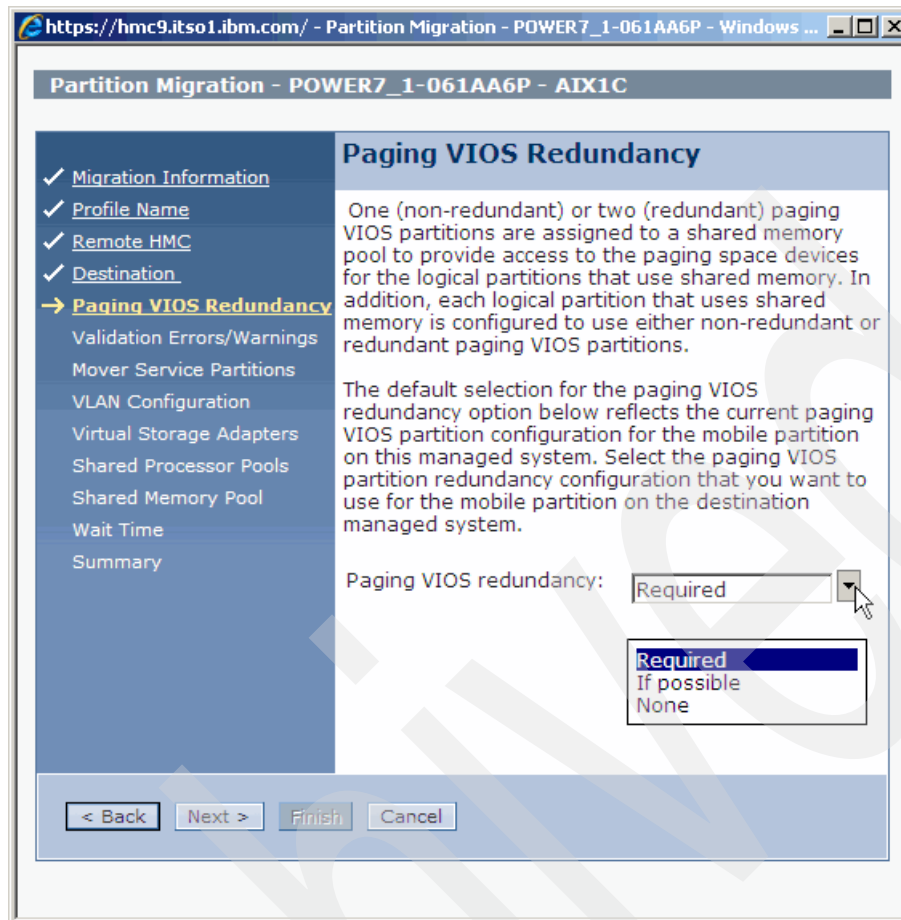


Figure 2-13 Live Partition Mobility - Paging VIOS redundancy

If “Paging VIOS redundancy” is set to **Required** the Management Console will make sure there is a redundant paging device available on the target system; otherwise, the migration will fail if none is found. If it is set to **If possible**, the Management Console will look for an available redundant device but will allow the partition to migrate even if no redundant paging device is available.

**Note:** You might lose paging VIOS redundancy by choosing “**If possible**” if no redundant paging device is available on the target system.

In a dual VIOS configuration, an additional step is displayed during the migration process where you must select the “Primary paging VIOS,” as shown in Figure 2-14 on page 31.

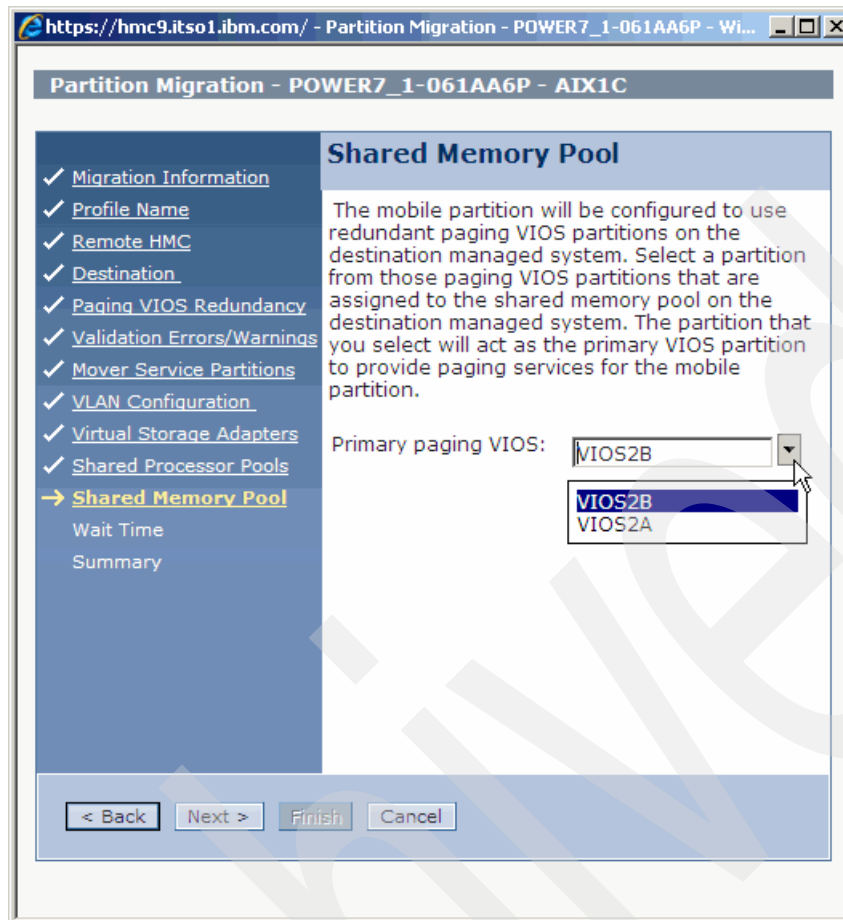


Figure 2-14 Primary paging VIOS choice during partition migration

During the migration, the partition memory content is copied to the target system. All memory pages that are on the AMS paging device are paged back in the physical memory before the copy starts.

## 2.6 Processor and memory virtualization compared

Active Memory Sharing maximizing memory utilization is analogous to shared processor LPAR maximizing processor utilization. There are notable similarities and differences between these two technologies, as identified in Table 2-1 and Table 2-2 on page 32.

Table 2-1 Similarities between shared processors and shared memory technologies

Similar features	Shared processors	Shared memory
Resource sharing. Multiple LPARs share resources from a single pool	Processor Pool	Memory Pool
Over-Commit	Number of virtual processors can exceed the number of physical processors in the shared memory pool	Logical memory configured for all partitions can exceed the real physical memory in the shared pool
Maximize utilization	Unused cycles from one LPAR can be dispatched to any uncapped-mode LPAR	Unused pages from one LPAR can be reassigned to another LPAR
Priority	LPARs are given priority weight for utilizing excess cycles	LPARs are given weight to enforce priority in allocating memory
Number of pools	Multiple (when using an HMC)	One

Table 2-2 Differences between shared processor and shared memory technologies

Differing features	Shared processor	Shared memory
Dependency	Does not require Active Memory Sharing	Does require shared processor LPAR
Resource entitlement	Processor entitlement guaranteed	No physical memory guaranteed, except for I/O entitled memory
Shared weight or priority	Configured shared weight is used to dispatch only excess cycles beyond CPU entitlement of an LPAR	Shared weight is used along with additional factors to allocate memory beyond I/O entitlement
Shared pool	No user action is required to configure shared processor pool	Memory shared pool has to be created by the user
Physical versus virtual devices	Supports both physical and virtual devices.	Active Memory Sharing is only allowed with virtual devices (network, disk, and so on)

**Important:** In Active Memory Sharing technology, entitlement does not mean the actual capacity, as it does in shared processor technology.

## Planning for Active Memory Sharing

This chapter describes the following topics related to planning for your Active Memory Sharing environment:

- ▶ “Active Memory Sharing prerequisites” on page 34 provides a quick overview of all the prerequisites you need to deploy your Active Memory Sharing environment.
- ▶ “Deployment considerations” on page 34 discusses some basic guidelines to identify and choose your candidate workloads for Active Memory Sharing.
- ▶ “Sizing Active Memory Sharing” on page 38 describes tools and procedures to size your Active Memory Sharing environment.

## 3.1 Active Memory Sharing prerequisites

Table 3-1 provides the minimum levels required for Active Memory Sharing.

Table 3-1 Active Memory Sharing requirements

Component	Minimum level POWER6	Minimum level POWER7	Comments
Hardware	POWER6 processor-based server	POWER7 processor-based server	This hardware level is required because it provides the mechanism to enable AMS.
I/O devices	Virtual only	Virtual only	All I/O devices must be virtualized by VIOS.
Managed system firmware	340_075	710_043	Verify this on the ASMI home screen.
PowerVM	Enterprise Edition	Enterprise Edition	A hardware feature code, or CoD enabled feature.
Management console	HMC 7.3.4 SP3	HMC 7.7.2 SP1	► Check this on HMC/IVM home screen. ► Use <b>Updates</b> link to check version.
Virtual I/O Server	2.1.0.1-FP21	2.1.3.10-FP-23	Issue <b>ioslevel</b> at the VIOS shell.
AIX	6.1.3.0 TL 3	6.1.3.0 TL 4	Issue <b>oslevel -s</b> at the AIX shell.
IBM i	IBM i 6.1.1 + latest cumulative PTF package	IBM i 6.1.1 + latest cumulative PTF package	Issue <b>DSPPTF</b> at the IBM i command line.
Linux	Novell SuSE SLES11 kernel 2.6.27.25-0.1-ppc6 4 or newer	Novell SuSE SLES11 kernel 2.6.27.25-0.1-ppc 64 or newer	Issue <b>uname -r</b> at Novell Suse Linux prompt.

**Note:** For information about how to update system levels, consult the product documentation.

## 3.2 Deployment considerations

With Active Memory Sharing, you can optimize memory utilization through consolidation of workloads and achieve increased overall memory utilization. In the following sections, we describe the concepts and basic rules to identify those workloads and how to deploy an Active Memory Sharing environment.

### 3.2.1 Overcommitment

In an Active Memory Sharing environment you can configure more logical memory than the available physical memory configured in the shared memory pool, resulting in an overcommitment scenario.



Each shared memory partition perceives itself as owning more of the memory, which results in total logical memory being oversubscribed.

There are three possible shared memory scenarios that can be obtained depending on the environment and workloads. They are:

**Non overcommit**

The amount of real memory available in the shared pool is enough to cover the total amount of logical memory configured. The example in Figure 3-1 shows partitions in which the logical memory total does not exceed the shared memory pool size.

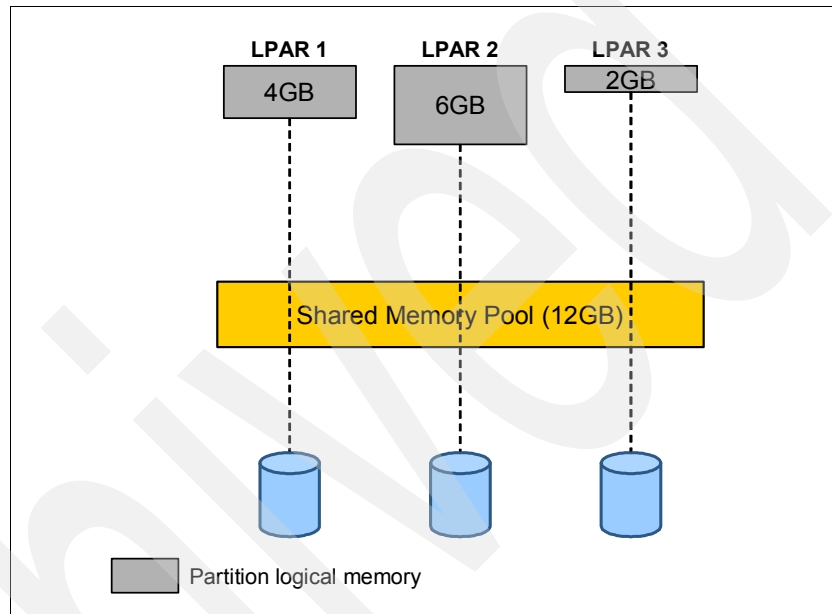


Figure 3-1 Non overcommit

**Logical overcommit**

The logical memory in use at a given time is equal to the physical memory available in the shared memory pool. That is, the total logical configured memory can be higher than the physical memory; however, the working set never exceeds the physical memory. Figure 3-2 on page 36 shows three logical partitions defined with 10 GB of logical memory each, and the sum of their memory usage never exceeds the shared memory pool size across time.

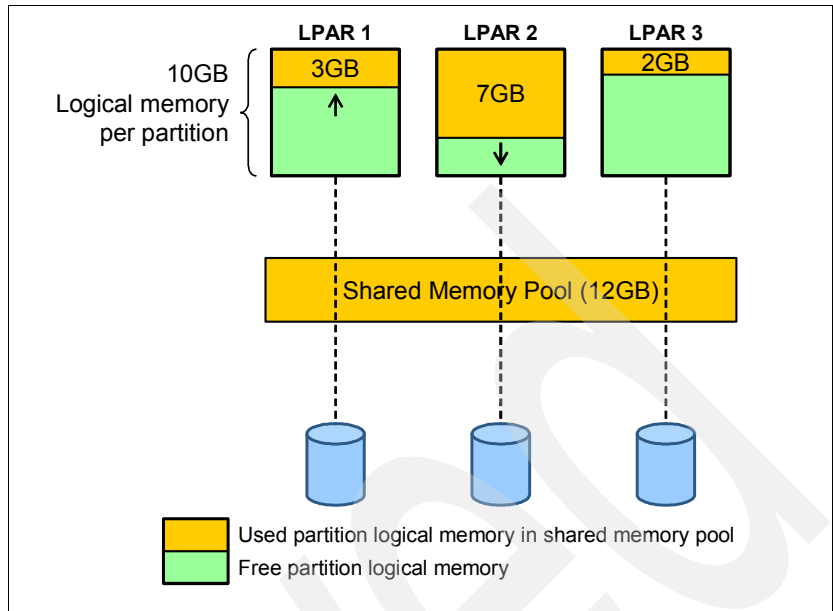


Figure 3-2 Logical overcommit

**Physical overcommit** The working set memory requirements can exceed the physical memory in the shared pool. Therefore, logical memory has to be backed by both the physical memory in the pool and by the paging devices. Figure 3-3 shows LPAR 3 as having 1 GB of its used memory on a paging device because the sum of the used memory is more than the size of the shared memory pool.

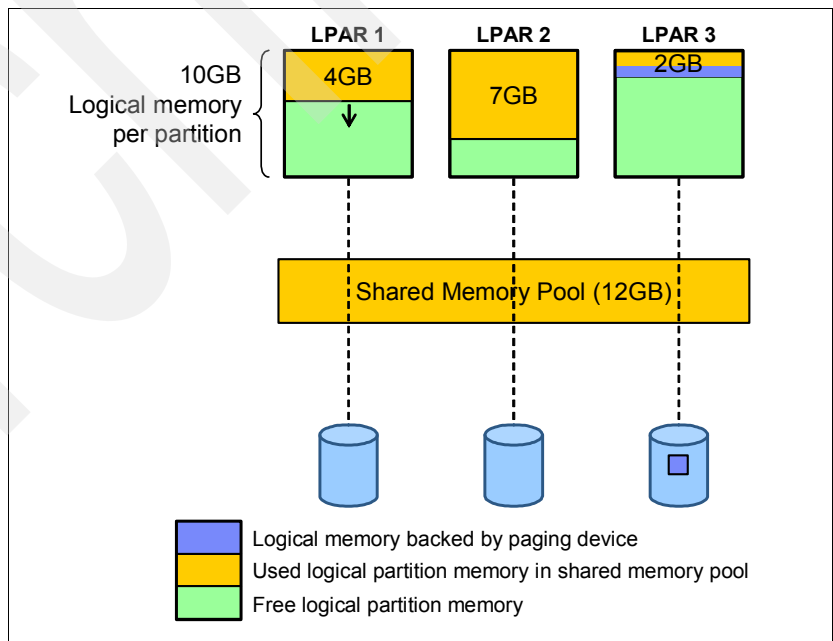


Figure 3-3 Physical overcommit

In the case of physical overcommitment, the hypervisor backs the excess logical memory using paging devices that are accessed through its paging Virtual I/O Server.

## 3.2.2 Workload selection

When selecting workloads to deploy using Active Memory Sharing, you first have to understand the workloads and the memory requirements of those workloads. It is important to monitor the workload peaks and valleys for a period of time (day/week/month) in dedicated memory mode to appropriately size the shared memory pools.

If you find deployed workloads that are not maximizing physical memory consumption, those workloads would be the prime candidates for Active Memory Sharing. In addition, there are some general principles to consider for workload selection.

As described in 3.2.1, “Overcommitment” on page 34, there are multiple scenarios where memory can be overcommitted. Along with the workload selection, memory configuration decisions must be made. The following sections offer general rules to select what workloads fit which scenarios.

### Logical overcommit

Active Memory Sharing logical overcommit is favored for workloads that have the following characteristics:

- ▶ Workloads that time multiplex. For example, in AM/PM scenarios, peaks and valleys of multiple workloads overlap leading to logical overcommit levels without consuming more than the physical memory available in the pool.
- ▶ Workloads that have low average memory residency requirements, where consolidating them would lead to logical overcommit.
- ▶ Workloads that do not have sustained loads, such as retail headquarters and university environments.
- ▶ Failover and backup partitions that are used for redundancy that require resources only when the primary server goes down with resources that do not have to be dedicated to redundant servers.
- ▶ Test and development environments.
- ▶ Private Cloud Computing environments.

### Physical overcommit

Active Memory Sharing physical overcommit is favored for workloads that have the following characteristics:

- ▶ Workloads that currently run on the AIX operating system and use a lot of AIX file cache.
- ▶ Workloads that are less sensitive to I/O latency such as file servers, print servers, and network applications.
- ▶ Workloads that are inactive most of the time.
- ▶ Public Cloud Computing environments.

### Dedicated memory partitions

Dedicated memory partitions are appropriate for the following workloads:

- ▶ Workloads that have high quality of service requirements.
- ▶ Workloads that have high sustained memory consumption due to sustained peak load.
- ▶ Workloads that mandate predictable performance.
- ▶ High Performance Computing workloads such as scientific computational intensive workloads that have sustained high CPU utilization and have high memory bandwidth requirements.

### 3.2.3 Consolidation factors

After workloads are selected for consolidation, the following factors must be considered:

- ▶ Logical to physical memory ratio suitable for the selected workloads.
- ▶ Shared memory weight determination for the workloads, assigning priority to your workloads.
- ▶ Based on the memory ratio, determine paging device configuration. If you have a high ratio you need to optimize your paging device using the recommendations given in the next section and in Chapter 6, “Tuning” on page 83.
- ▶ In a physical overcommit environment, determine whether aggressive loaning combined with the application load levels provide acceptable performance. The loaning level is set up on an operating system basis. Therefore, a mix of loaning levels can coexist on the same system.
- ▶ Optimize utilization through rebalancing resources. Physical memory might limit the number of shared processor partitions even though the system CPU and memory bandwidth utilization levels are low.

### 3.2.4 Paging device planning

Planning an Active Memory Sharing paging device is no different from planning for an operating system paging device.

When paging occurs, you want to perform it as fast as possible when high performance is a priority. Follow these guidelines to configure your paging devices for maximum performance and availability:

- ▶ Configure some level of disk redundancy, such as mirroring or RAID5.
- ▶ Use a small stripe size.
- ▶ Spread the I/O load across as much of the disk subsystem hardware as possible.
- ▶ Use a write cache on either the adapter or storage subsystem.
- ▶ Where possible, use physical volumes in preference over logical volumes.
- ▶ Size your storage hardware according to your performance needs.
- ▶ Ensure that PVIDs for paging devices for physical volumes set up by the HMC are cleared before use.

**Note:** For further details, see Chapter 6, “Tuning” on page 83.

## 3.3 Sizing Active Memory Sharing

This section describes some models that can be used to size the requirements for Active Memory Sharing.

### 3.3.1 Virtual I/O Server resource sizing

The Virtual I/O Server should be configured as before for its standard disk and network I/O hosting for LPARs. The IBM Systems Workload Estimator tool can be used to help size the VIOS server. For more information about this tool, see:

<http://www.ibm.com/systems/support/tools/estimator/index.html>

With Active Memory Sharing, the Virtual I/O Server takes a new role. If designated as a paging Virtual I/O Server, then the storage subsystem and paging rate must also be taken into consideration.

Table 3-2 provides a guide for estimating CPU entitlement per shared memory partition depending on estimated page rate and storage subsystem type.

*Table 3-2 Estimated CPU entitlement requirements based on activity and storage type*

Paging rate	Storage type			
	Internal storage	Entry level storage	Mid range storage	High end storage
Light	0.005	0.010	0.020	0.020
Moderate	0.010	0.020	0.040	0.080
Heavy	0.020	0.040	0.080	0.160

**Note:** The heavier the paging rate, the more I/O operations are occurring within the Virtual I/O Server partition, thus resulting in an increased need for processor entitlement.

Figure 3-4 plots Table 3-2 on page 39 information to show the relationship between previous values.

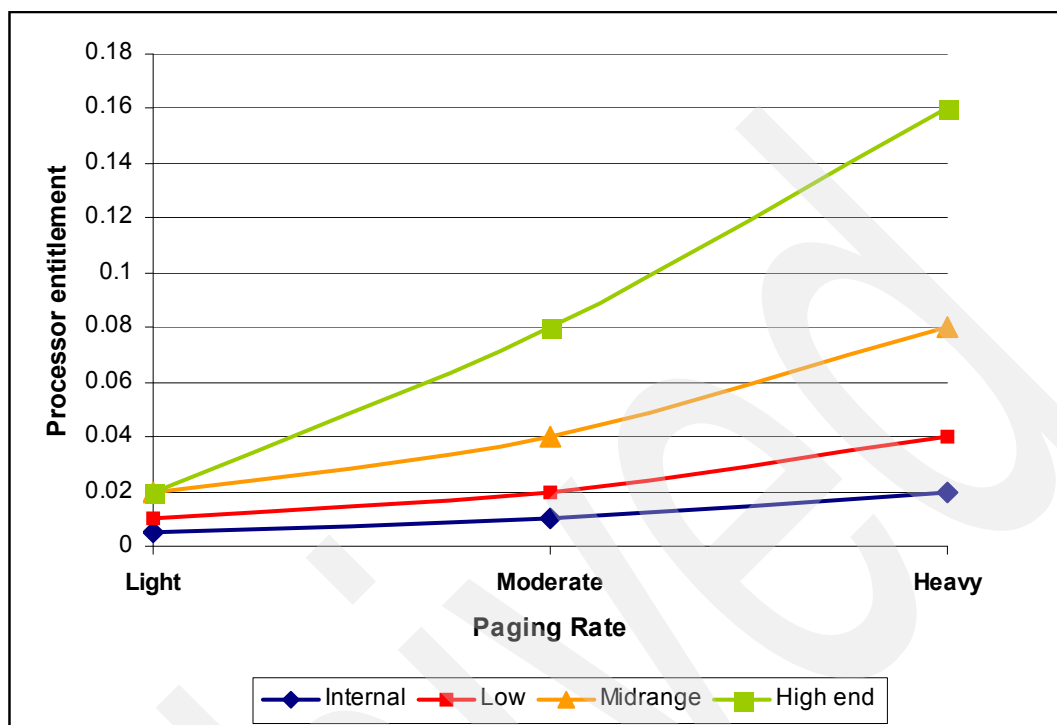


Figure 3-4 Estimated additional CPU entitlement needed to support AMS paging device

As an example, if five shared memory partitions are being deployed, using a midrange storage solution for the shared memory pool paging devices, it would be recommended to use an additional  $(5 * 0.04) = 0.2$  CPU entitlement for the Virtual I/O Server to ensure that enough computing resources are available to service five shared memory partitions simultaneously experiencing moderate hypervisor paging.

**Important:** You can deploy a VIOS just to perform AMS paging and therefore separate paging I/O from other VIOS functions, such as virtual disk and virtual network I/O in case of performance concerns. It is recommended to test a configuration before going into production.

### 3.3.2 Shared memory partition CPU sizing

Most of the tasks associated with Active Memory Sharing require the hypervisor to consume additional CPU cycles. In an oversubscription memory configuration, this increase in CPU utilization happens to be a function of:

- ▶ The access rate of the pages of physical memory
- ▶ Physical to logical memory ratio
- ▶ Rate of disk access

If the memory is not overcommitted, the CPU utilization increase will be minimal due to the additional software layer of virtualization.

## Configuring and managing

This chapter describes how to configure Active Memory Sharing. The configuration is done using the following three steps:

- ▶ Create the paging devices on the Virtual I/O Server.
- ▶ Create the shared memory pool on the Management Console or IVM.
- ▶ Configure the shared memory partitions.

**Note:** The examples in the following sections were created using Management Console GUI panels and the Management Console command line interface. The same actions can also be performed using IVM.

## 4.1 Creating the paging devices

A paging device is required for each shared memory partition. The size of the paging device must be equal to or larger than the maximum logical memory defined in the partition profile. The paging devices are owned by a Virtual I/O Server. A paging device can be a logical volume or a whole physical disk. Disks can be local or provided by an external storage subsystem through a SAN.

If you are using whole physical disks, there are no actions required other than making sure that the disks are configured and available on the Virtual I/O Server and any PVID is cleared.

**Note:** If you plan to use a dual Virtual I/O Server configuration and take advantage of redundancy, your paging devices have to be provided through a SAN and accessible from both Virtual I/O Servers.

Example 4-1 shows the creation of two logical volumes that will be used as paging devices. If you are using logical volumes, you must create a volume group and the logical volumes using the `mkvg` and `mklv` commands as shown in this example.

*Example 4-1 Creating logical volumes as paging devices on the Virtual I/O Server*

```
$ mkvg -vg amspaging hdisk2
amspaging
$ mklv -lv amspaging01 amspaging 5G
amspaging01
$ mklv -lv amspaging02 amspaging 5G
amspaging02
$ lsvg -lv amspaging
amspaging:
LV NAME          TYPE      LPs    PPs    PVs  LV STATE    MOUNT POINT
amspaging01      jfs       40     40     1    closed/syncd N/A
amspaging02      jfs       40     40     1    closed/syncd N/A
```

If you are using IVM to manage your system, this concept, though valid, is more automated. IVM will, upon creating a memory pool, prompt you for a volume group to use for paging. IVM will then automatically create paging devices of the correct size with respect to requirements of your partitions. IVM will also automatically resize the paging device when the partition maximum memory increases, and will delete the device if, through IVM, you delete the partition or switch it to dedicated memory.

If you are configuring SAN storage for paging devices, you should use best practices when assigning SAN LUNs to AMS pools. When possible, use WWPN zoning to ensure that physical paging devices intended for a given AMS pool are only accessible from the one or two Virtual I/O Server partitions that are supporting the AMS pool. If the paging device is composed of logical volumes on the SAN, each logical volume should be zoned to allow access from only the Virtual I/O Server partition where the logical volume was created. By using zoning to enforce isolation of the paging devices, you will avoid problems where a device is unintentionally assigned for multiple uses simultaneously. When the device is configured for multiple uses, this exposes the contents of the paging devices to possible data integrity problems including corruption.

For information about SAN zoning, see:

<http://www.redbooks.ibm.com/abstracts/sg246116.html?Open>



## 4.2 Creating the shared memory pool

This section describes how to create the shared memory pool using the Management Console. In this example we create a shared memory pool with a size of 20 GB and nine paging devices. Paging devices are disks provided through a SAN and mapped on both Virtual I/O Servers. A paging device is required for each shared memory partition. Therefore, the pool will be able to accommodate nine shared memory partitions.

Perform the following steps to create the shared memory pool:

1. On the Management Console, select the managed system on which the shared memory pool will be created. Then select **Configuration** → **Shared Memory Pool Management**, as shown in Figure 4-1.

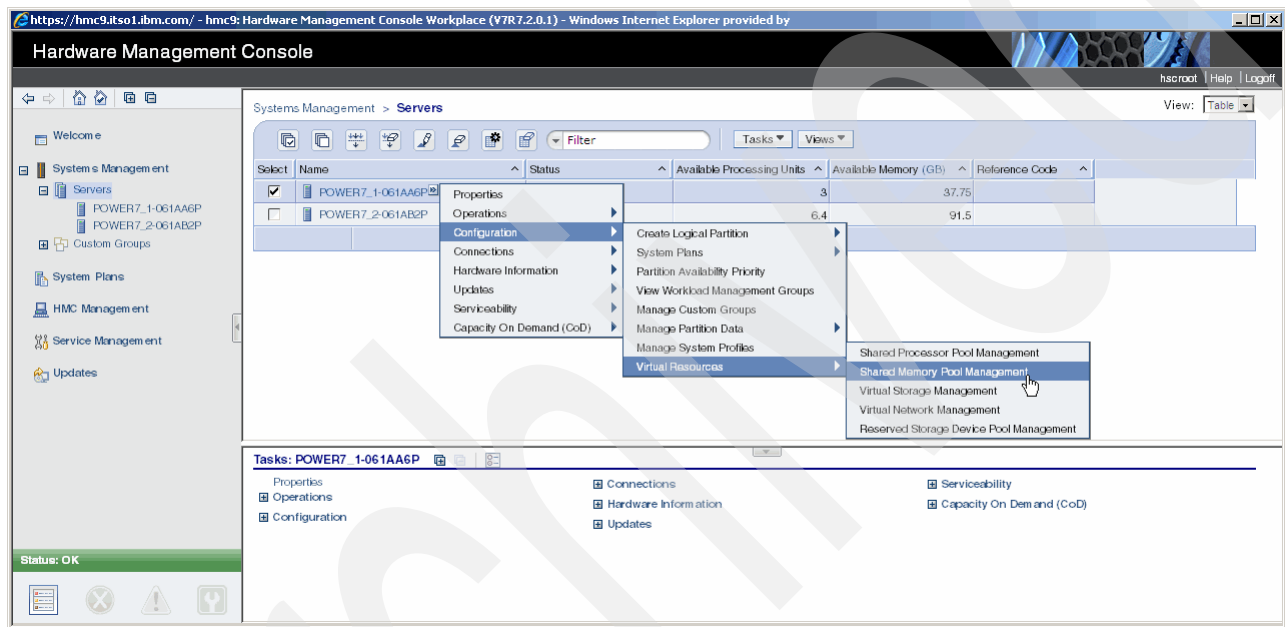


Figure 4-1 Creating a shared memory pool

2. The Welcome window appears, requiring no input. Select **Next** to continue.
3. Enter the Pool size and the Maximum pool size as shown in Figure 4-2 on page 44 and click **Next**.

The pool size cannot be larger than the amount of available system memory. The memory used by dedicated memory partitions cannot be used for the shared memory pool. In this case, the pool is created on a managed system where we cannot create a pool larger than 62.8 GB.

The Maximum pool size is a soft limit that can be changed dynamically. Keep the Maximum pool size to the required minimum because for each 16 GB of pool memory the hypervisor reserves 256 MB of memory for page tracking. Therefore, if you set the maximum too high you will be misallocating memory.

**Note:** Paging space devices can only be assigned to one shared memory pool at a time. You cannot assign the same paging space device to a shared memory pool on one system and to another shared memory pool on another system at the same time.

https://hmc9.itso1.ibm.com/ - hmc9: Shared Memory Pool Management - Windows ...

### Create Shared Memory Pool - POWER7\_1-061AA6P

✓ Welcome

→ **General**

Paging VIOS

Paging Space Device(s)

Summary

#### General

A shared memory pool defines the amount of shared memory available on the system. Any memory assigned to the pool is not available for use by dedicated memory partitions.

Available system memory: 62.8 GB

Maximum pool size:  GB  MB

Pool size:  GB  MB

< Back   Next >   Finish   Cancel

Figure 4-2 Defining the Pool size and Maximum pool size

4. As shown in Figure 4-3, select the Virtual I/O Server partitions that will be used from the pull-down menu and click **Next**.

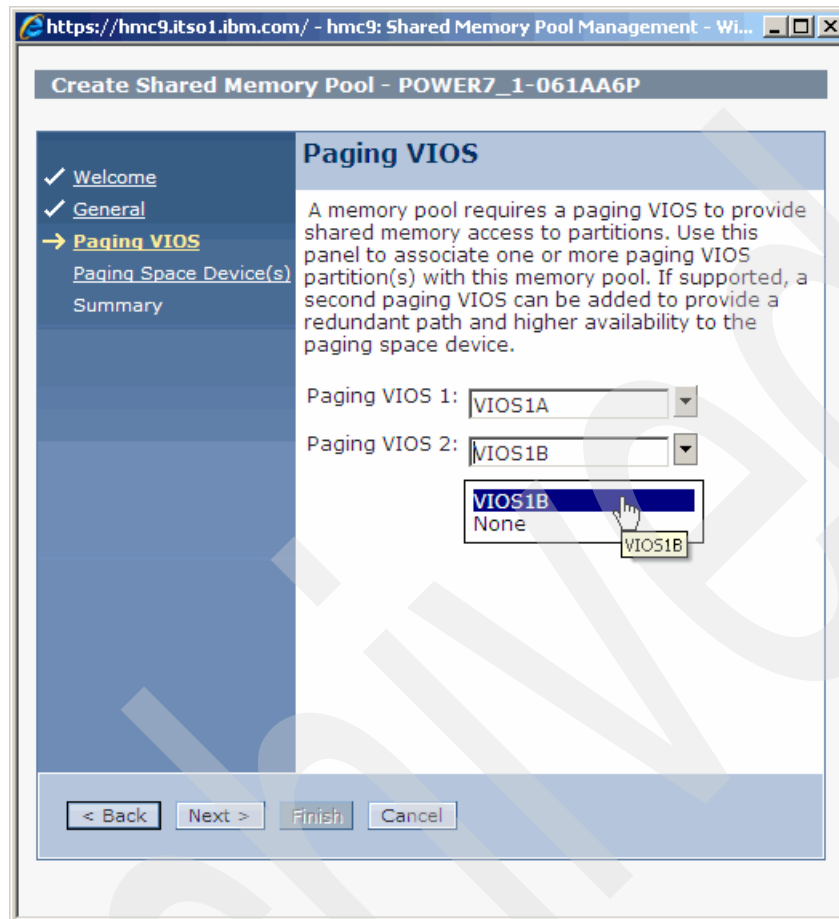


Figure 4-3 Selecting paging space partitions

You can select a maximum of two **Paging Virtual I/O Servers** for redundancy.

- Click **Select Devices**, as shown in Figure 4-4.

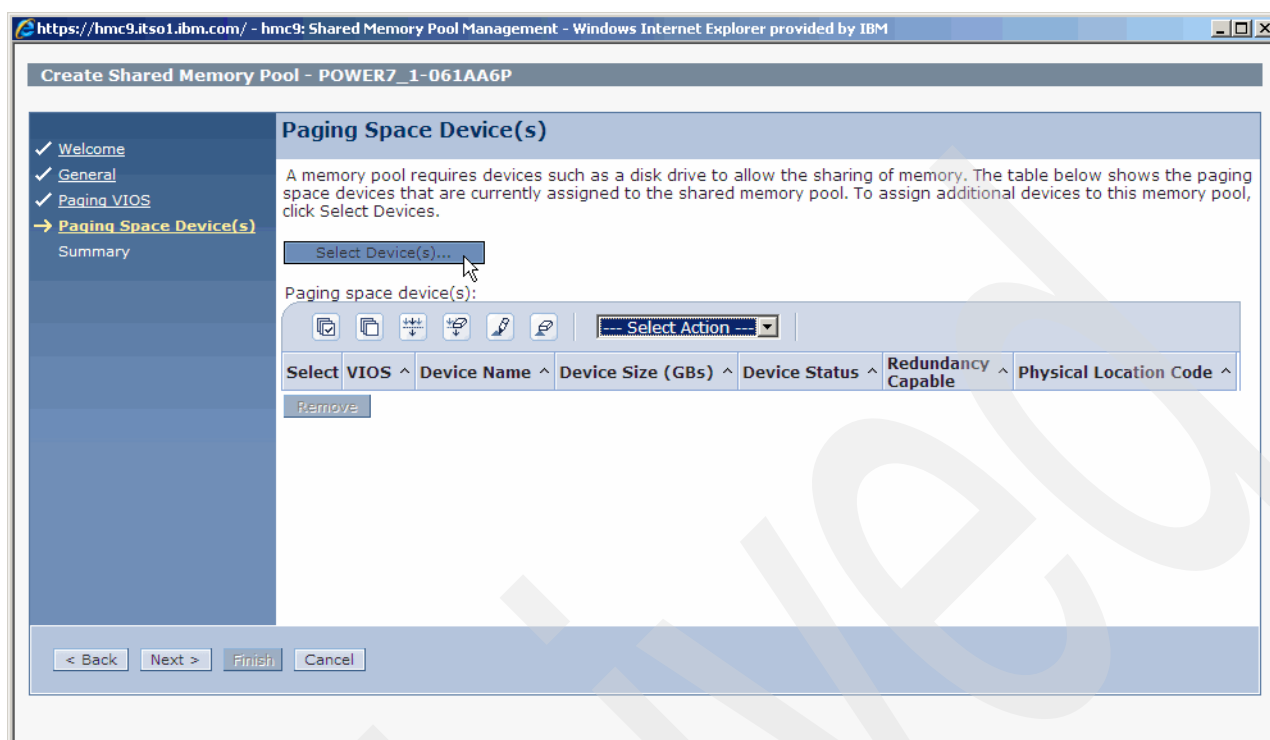


Figure 4-4 Selecting paging devices

- Click **Refresh**, as shown in Figure 4-5 on page 47, to display the list of available paging devices. Select the paging devices that should be assigned to the shared memory pool and click **OK**.

In this example, devices with a maximum size of 10 GB are filtered. These devices are displayed in the **Common devices list** because both Virtual I/O Servers can access them. All devices are 10 GB in size. This means that the maximum logical memory setting defined in the partition profile of each of the nine partitions cannot be greater than 10 GB.

**Important:** The **Redundancy Capable** property only indicates the *capability* for the device to be redundant. It does not mean the device *is* redundant. A redundant device must be seen by both Virtual I/O Servers and therefore is displayed in the **Common devices list**.

#### Notes:

- ▶ Logical volumes or physical disks that have been mapped to a virtual SCSI adapter will not appear in the selection list.
- ▶ For an IBM i shared memory partition, the paging device must be larger than the maximum memory defined in the partition profile. An IBM i partition requires 1 bit extra for every 16 byte page it allocates. This is a factor of 129/128. The paging size must be greater than or equal to  $\text{max mem} * 129/128$ . That means an IBM i partition with 10 GB of memory needs a paging device with a minimum size of 10.08 GB of paging space.

https://hmc9.itso1.ibm.com/#tableTop\_2f8a2f8a - hmc9: Shared Memory P...

### Paging Space Device Selection - POWER7\_1-061AA6P

To display the available paging space devices in the device lists, you must first select filter parameters and then click Refresh. You can list all available paging space devices by selecting All as the device type, or you can narrow your search by selecting a device type, maximum size, or minimum size.

Device Type:

☒ Maximum Size (in GBs):

☐ Minimum Size (in GBs):

Choose from the following list of devices. You can choose more than one paging space device to be added to the pool. Paging space devices should be assigned to only one shared memory pool at a time. You should not assign a paging space device to this shared memory pool if it is already assigned to another shared memory pool on another system.

After you have made your selections, select the OK button to assign the selected devices to the memory pool.

Common device list:

Select	VIOS ^	Device Name ^	Device Size (GBs) ^	Redundancy Capable ^
<input checked="" type="checkbox"/>	VIOS1A VIOS1B	hdisk3	10.0	True
<input checked="" type="checkbox"/>	VIOS1A VIOS1B	hdisk4	10.0	True
<input checked="" type="checkbox"/>	VIOS1A VIOS1B	hdisk5	10.0	True
<input checked="" type="checkbox"/>	VIOS1A VIOS1B	hdisk6	10.0	True
<input checked="" type="checkbox"/>	VIOS1A VIOS1B	hdisk7	10.0	True
<input checked="" type="checkbox"/>	VIOS1A VIOS1B	hdisk8	10.0	True
<input checked="" type="checkbox"/>	VIOS1A VIOS1B	hdisk9	10.0	True
<input checked="" type="checkbox"/>	VIOS1A VIOS1B	hdisk10	10.0	True
<input checked="" type="checkbox"/>	VIOS1A VIOS1B	hdisk11	10.0	True

Figure 4-5 Selecting paging devices

7. In the Summary window, shown in Figure 4-6, click **Finish** to start the creation of the shared memory pool.

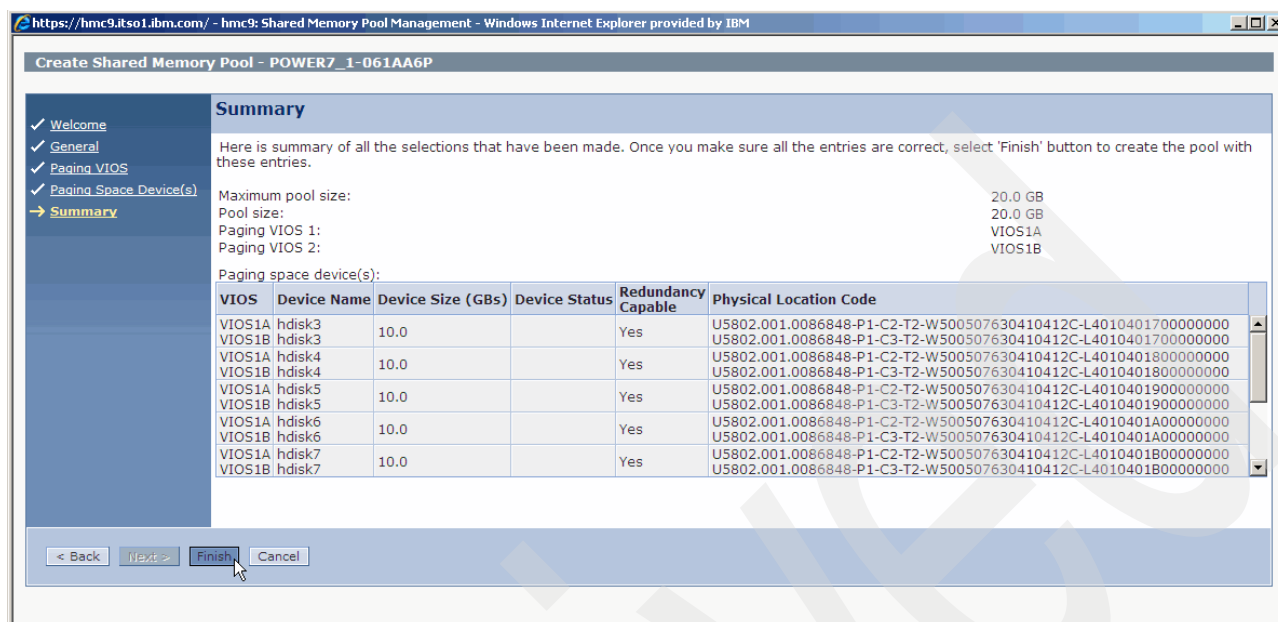


Figure 4-6 Finishing shared memory pool creation

Using the `lshwres` command, the shared memory pool can be displayed on the Management Console command line interface. Using the `-r mempool` flag, as shown in Example 4-2, the attributes of the shared memory pool are displayed.

*Example 4-2 Displaying the shared memory pool using lshwres*

```
hscroot@hmc9:~> lshwres -m POWER7_1-061AA6P -r mempool
curr_pool_mem=20480,curr_avail_pool_mem=20224,curr_max_pool_mem=20480,pend_pool_me
m=20480,pend_avail_pool_mem=20224,pend_max_pool_mem=20480,sys_firmware_pool_mem=25
6,"paging_vios_names=VIOS1A,VIOS1B","paging_vios_ids=1,2"
```

When using the `lshwres` command with the `--rsubtype pgdev` flag, as shown in Example 4-3, the attributes and status of each paging device are displayed.

*Example 4-3 Displaying paging devices using lshwres*

```
hscroot@hmc9:~> lshwres -m POWER7_1-061AA6P -r mempool --rsubtype pgdev
device_name=hdisk3,paging_vios_name=VIOS1A,paging_vios_id=1,size=10240,type=phys,state=Inac
tive,phys_loc=U5802.001.0086848-P1-C2-T1-W500507630410412C-L4010401700000000,is_redundant=1
,redundant_device_name=hdisk3,redundant_paging_vios_name=VIOS1B,redundant_paging_vios_id=2,
redundant_state=Inactive,redundant_phys_loc=U5802.001.0086848-P1-C3-T2-W500507630410412C-L4
0104017000000000,lpar_id=none
[...]
device_name=hdisk11,paging_vios_name=VIOS1A,paging_vios_id=1,size=10240,type=phys,state=Ina
ctive,phys_loc=U5802.001.0086848-P1-C2-T1-W500507630410412C-L4011401A00000000,is_redundant=
1,redundant_device_name=hdisk11,redundant_paging_vios_name=VIOS1B,redundant_paging_vios_id=
2,redundant_state=Inactive,redundant_phys_loc=U5802.001.0086848-P1-C3-T2-W500507630410412C-
L4011401A00000000,lpar_id=none
```

## 4.3 Creating a shared memory partition

The process to create a shared memory partition is similar to creating a partition with dedicated memory. The main difference is in the memory section, where you have to specify that the partition will use shared memory, as shown in Figure 4-7.

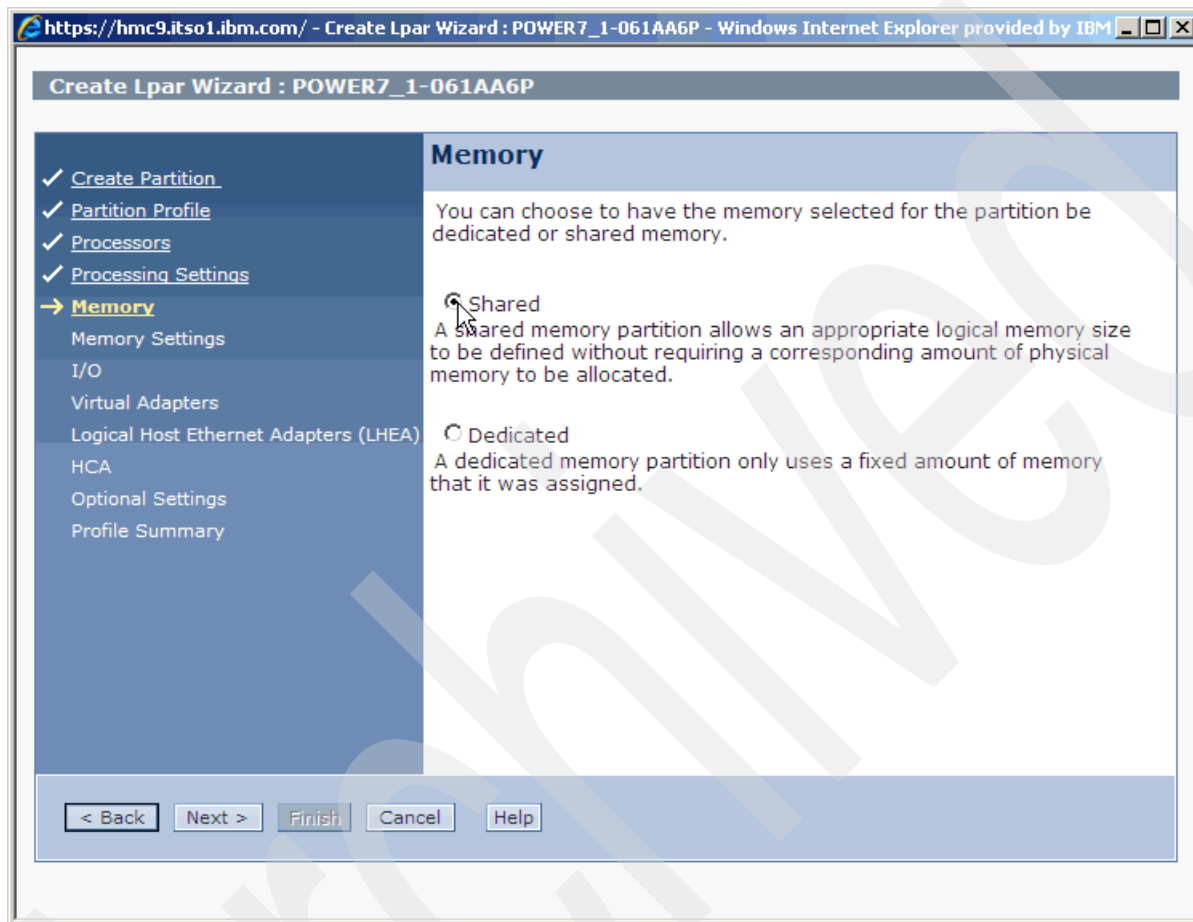


Figure 4-7 Defining a shared memory partition

As shown in Figure 4-8 on page 51, the memory settings are similar to those of a dedicated memory partition. The **Minimum Memory** setting defines how much logical memory is required to activate the partition. The **Desired Memory** setting is the amount of logical memory that the partition should get. **Maximum Memory** defines the maximum amount of logical memory that can be assigned to the partition using dynamic LPAR.

**Notes:**

- ▶ In a shared memory partition the Minimum, Desired, and Maximum settings do not represent physical memory values. These settings define the amount of logical memory that is allocated to the partition.
- ▶ The paging device is selected based on the Maximum Memory setting. If there is no paging device available that is larger than or equal to the Maximum Memory, the partition will fail to activate.

**Custom Entitled Memory** defines how much memory is allowed to be permanently kept in the memory pool to handle I/O activities. It should not be modified unless you have specific requirements regarding I/Os. For more information see 6.2.2, “Logical partition” on page 87.

You can choose one or two paging VIOS for redundancy. When using two VIOS, VIOS1 is set as the primary paging VIOS for this partition and will therefore be used to handle the paging activity. In case of failure, the hypervisor will switch to the secondary VIOS. Because primary and secondary VIOS are defined at the partition level, you can load balance the paging workload on the two VIOS by defining different paging VIOS on each partition. For partitions using dual VIOS, you must make sure there are redundant paging devices available. See 4.4.1, “Paging device assignment” on page 51 for details about paging device redundancy.

The **Memory Capacity Weight** setting is one of the factors used by the hypervisor to determine which shared memory partition should receive more memory from the shared memory pool. The higher this setting is, the more the partition is likely to get physical memory. However, the main factor for physical memory allocation remains partition activity. This means that a partition with a higher workload will get more physical memory even if it has a lighter weight. This weight factor is mainly useful for situations where two or more partitions have similar activity at the same time and you want to favor one or some of them.

Active Memory Expansion is fully compatible with AMS and therefore allows you to choose an **Active memory expansion factor** if desired. See 2.5.1, “Active Memory Expansion” on page 26 for more details about AMS behavior with AME.



Figure 4-8 Defining memory settings

**Note:** By design, the Management Console prevents you from assigning a physical I/O slot or Logical Host Ethernet adapter to a shared memory partition.

## 4.4 Managing Active Memory Sharing

This section describes how the components that are part of Active Memory Sharing are managed. The following topics are covered:

- ▶ Management of paging devices
- ▶ Management of the shared memory pool
- ▶ Management of shared memory partitions
- ▶ Dual VIOS considerations

### 4.4.1 Paging device assignment

There is no fixed relationship between a paging device and a shared memory partition when a system is managed using the Management Console. The smallest suitable paging device will automatically be selected when the shared memory partition is activated for the first time. Once a paging device has been selected for a partition, this device will be used again as long as it is available at time the partition is activated. However, if the paging device is unavailable,

for example because it has been deconfigured or is in use by another partition, a new suitable paging device will be selected.

Regarding IVM, the link is persistent and no implicit stealing of paging devices is done. When using IVM, if you want to reuse a paging device for another partition, you must explicitly remove it from the old partition.

The IVM CLI allows you to specify which paging device to use for each partition. IVM will pick one by default automatically, but you can override this manually.

Keep in mind that when using IVM you must have paging devices for every shared memory partition, whether activated or not (unless manually configuring using the IVM CLI).

To see which partition is using which paging device, select **Shared Memory Pool Management** from the Management Console, then select the **Paging Space Device(s)** tab, as shown in Figure 4-9.

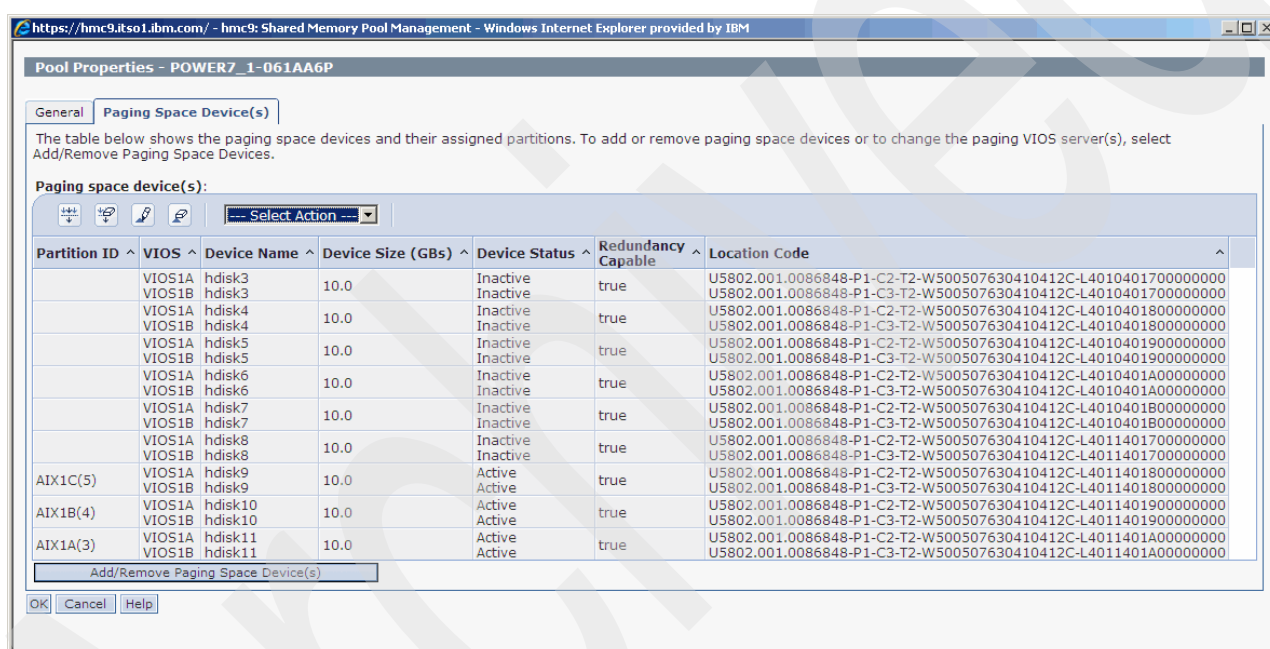


Figure 4-9 Pool properties settings

You can also use the `lshwres` command to display the paging device assignment, as shown in Example 4-4.

#### Example 4-4 Displaying paging devices using lshwres

```
hscroot@hmc9:~> lshwres -m POWER7_1-061AA6P -r mempool --rsubtype pgdev
device_name=hdisk9,paging_vios_name=VIOS1A,paging_vios_id=1,size=10240,type=phys,state=Active,phys_loc=U5802.001.0086848
-P1-C2-T2-W500507630410412C-L4011401800000000,is_redundant=1,redundant_device_name=hdisk9,redundant_paging_vios_name=VIO
S1B,redundant_paging_vios_id=2,redundant_state=Active,redundant_phys_loc=U5802.001.0086848-P1-C3-T2-W500507630410412C-L4
0114018000000000,lpar_name=AIX1C,lpar_id=5
device_name=hdisk10,paging_vios_name=VIOS1A,paging_vios_id=1,size=10240,type=phys,state=Active,phys_loc=U5802.001.008684
8-P1-C2-T2-W500507630410412C-L4011401900000000,is_redundant=1,redundant_device_name=hdisk10,redundant_paging_vios_name=V
IOS1B,redundant_paging_vios_id=2,redundant_state=Active,redundant_phys_loc=U5802.001.0086848-P1-C3-T2-W500507630410412C-
L4011401900000000,lpar_name=AIX1B,lpar_id=4
device_name=hdisk11,paging_vios_name=VIOS1A,paging_vios_id=1,size=10240,type=phys,state=Active,phys_loc=U5802.001.008684
8-P1-C2-T2-W500507630410412C-L4011401A00000000,is_redundant=1,redundant_device_name=hdisk11,redundant_paging_vios_name=V
IOS1B,redundant_paging_vios_id=2,redundant_state=Active,redundant_phys_loc=U5802.001.0086848-P1-C3-T2-W500507630410412C-
L4011401A00000000,lpar_name=AIX1A,lpar_id=3
```

## 4.4.2 Adding paging devices

To add a paging device, you first have to provision the physical disk or logical volume on the Virtual I/O Server as shown in 4.1, “Creating the paging devices” on page 42. Then you have to add the paging devices to the pool as shown in step 6 on page 46. Instead of the Management Console GUI you can also use the Management Console command line and enter the **chhwres** command, as shown in Example 4-5.

**Note:** Logical volumes or physical disks that have been mapped to a virtual SCSI adapter will not appear in the selection list. However, it is possible to map a storage device to a virtual SCSI adapter even if it is defined as a paging device to the shared memory pool. No warning message will be issued when the **mkvdev** command is executed on the Virtual I/O Server.

## 4.4.3 Removing paging devices

To remove a paging device, first deactivate the shared memory partition that is using the paging device. As long as a paging device is in use by a shared memory partition it cannot be removed. Remove the paging device from the pool using the Management Console. After the paging device is removed you can deconfigure the corresponding storage devices from the Virtual I/O Server. This can be done using the Management Console GUI or by issuing the **chhwres** command on the Management Console command line, as shown in Example 4-5.

When using IVM, if the paging device was already used by a partition, you must use the CLI to remove it.

**Important:** Do not manage a paging device that is assigned to the shared memory pool using the Virtual I/O Server command line.

## 4.4.4 Changing the size of a paging device

It is not possible to increase or decrease the size of an existing paging device. You have to add a new paging device with the desired size and then remove the old device. Before removing the old paging device, the shared memory partition using it must be deactivated.

Adding and removing a paging device can be done using the Management Console GUI or the Management Console command line. Example 4-5 shows how an existing paging device of 5 GB size is removed and a new 10 GB paging device is added using the **chhwres** command.

### Example 4-5 Removing and adding paging devices using *chhwres*

```
hscroot@hmc9:~> lshwres -m POWER7_1-061AA6P -r mempool --rsubtype pgdev
device_name=amspaging01,paging_vios_name=VIOS1A,paging_vios_id=1,size=5120,type=logical,state=Inactive,is_redundant=0,lp
ar_id=none
device_name=amspaging02,paging_vios_name=VIOS1A,paging_vios_id=1,size=5120,type=logical,state=Inactive,is_redundant=0,lp
ar_id=none

hscroot@hmc9:~> chhwres -m POWER7_1-061AA6P -r mempool --rsubtype pgdev -o r --device amspaging02 -p VIOS1A
hscroot@hmc9:~> chhwres -m POWER7_1-061AA6P -r mempool --rsubtype pgdev -o a --device amspaging03 -p VIOS1A

hscroot@hmc9:~> lshwres -m POWER7_1-061AA6P -r mempool --rsubtype pgdev
device_name=amspaging01,paging_vios_name=VIOS1A,paging_vios_id=1,size=5120,type=logical,state=Inactive,is_redundant=0,lp
ar_id=none
device_name=amspaging03,paging_vios_name=VIOS1A,paging_vios_id=1,size=10240,type=logical,state=Inactive,is_redundant=0,lp
ar_id=none
```

## 4.4.5 Managing the shared memory pool size

The size of the shared memory pool can be increased and decreased dynamically. To increase the shared memory pool, the unused memory has to be available. If you want to increase the shared memory pool beyond the maximum pool size, you first have to increase the maximum pool size to a value that is greater than or equal to the desired new pool size. The maximum pool size can be increased dynamically.

### Notes:

- ▶ Keep the maximum pool size to the required minimum because for each 16 GB of pool memory the hypervisor reserves 256 MB of memory. Though not a requirement, reduce the maximum pool size when you are reducing the pool size.
- ▶ Reducing the size of the shared memory pool might take additional time if there is not enough free memory available. In such a case, the hypervisor first has to page out memory. If you display the pool size during the time when the pool is in the process of being decreased the figure displayed will be changing.

## 4.4.6 Deleting the shared memory pool

A shared memory pool cannot be deleted while any shared memory partitions are configured to use the pool. The partitions must be removed or changed to dedicated memory partitions before the shared memory pool is deleted.

Using the Management Console, when a partition is changed from shared memory to dedicated memory in the partition profile, you must activate it so that the partition picks up the new memory mode. As long as the partition has not been activated in dedicated memory mode the shared memory pool cannot be deleted. This is also true if the LPARs using the pool are inactive, but the VIOS managing that pool is active.

When using the IVM, the change from shared memory to dedicated memory is immediate.

To delete the shared memory pool select **Shared Memory Pool Management** from the Management Console and then click **Delete Pool**, as show on Figure 4-10.

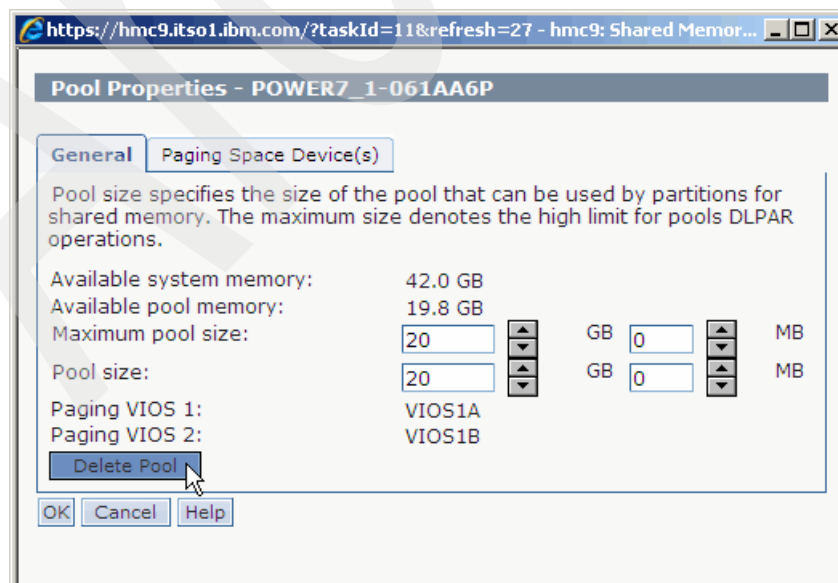


Figure 4-10 Shared memory pool deletion

After the shared memory pool is deleted, you have to delete the storage pool that was automatically created for the paging devices if you want to reuse them. Select **Reserved Storage Device Pool Management** on the Management Console, then click **Delete Pool** as shown on Figure 4-11.



Figure 4-11 Paging devices reserved storage pool deletion

#### 4.4.7 Dynamic LPAR for shared memory partitions

The amount of logical memory of a shared memory partition can be changed within the minimum and maximum boundaries defined in the partition profile. Increasing the logical memory in a shared memory partition does not mean that the amount of physical memory that is assigned through the hypervisor is changed. The amount of physical memory that a shared memory partition actually gets depends on the availability of free memory in the shared memory pool.

The memory weight of a shared memory partition can also be changed dynamically.

When you add or remove virtual SCSI, virtual Ethernet, or virtual FC adapters, the I/O entitled memory will automatically be adjusted. The values used are based on default values and normally do not need to be changed. After they are changed, the values will remain at their new setting.

#### 4.4.8 Switching between dedicated and shared memory

To change a partition from using shared memory to dedicated memory or back again, the memory mode setting in the partition profile must be changed. Note that when you switch from dedicated to shared you must not have any physical I/O devices configured. A pop-up window with a warning message will appear in this case. When you click **Yes**, all physical I/O assignments will be removed.

**Note:** To activate the changed memory mode, the partition must be deactivated and reactivated. When using IVM, the partition status must be deactivated for the change to be allowed.

#### 4.4.9 Starting and stopping the Virtual I/O Server

Before activating a shared memory partition, the Virtual I/O Server used for paging has to be active.

**Note:** The shared memory partitions cannot be activated until the Management Console has established an RMC connection with the Virtual I/O Server.

Before shutting down a Virtual I/O Server assigned as a paging space partition that has active shared memory partitions as clients, you must shut down the clients first. Otherwise, the shared memory partitions will lose their paging resources, and system paging errors will occur (as would happen with native paging devices). When you try to use the Management Console to deactivate a Virtual I/O Server with active shared memory partitions as clients, a warning message will be displayed to assist you in identifying this condition.

#### 4.4.10 Dual VIOS considerations

When using a Dual VIOS configuration, you can configure your shared memory pool with two paging VIOS (see Figure 4-3 on page 45). After you define a shared memory partition using this shared memory pool, you have the possibility to assign two VIOS as paging VIOS for the shared memory partitions: a primary and a secondary paging VIOS (see Figure 4-8 on page 51). The primary paging VIOS will be selected for AMS paging activity at partition activation.

##### Failover and load balancing

In case of a VIOS outage, affected partitions will switch to their secondary paging VIOS. Unless rebooted, the partitions will remain on the switched VIOS because there is no fallback to the primary paging VIOS, unless the current paging VIOS has an outage of course. It is possible to force a failover to the other VIOS using the Management Console command shown in Example 4-6.

##### Example 4-6

---

```
hscroot@hmc9:~> lshwres -r mem -m POWER7_1-061AA6P --level lpar -F lpar_name
curr_paging_vios_name
AIX1C VIOS1A
AIX1B VIOS1A
AIX1A VIOS1B
hscroot@hmc9:~> chhwres -r mem -m POWER7_1-061AA6P -p AIX1C -o so
hscroot@hmc9:~> lshwres -r mem -m POWER7_1-061AA6P --level lpar -F lpar_name
curr_paging_vios_name
AIX1C VIOS1B
AIX1B VIOS1A
AIX1A VIOS1B
```

---

Because the primary and secondary paging VIOS are defined at the partition level, it is possible to evenly assign paging VIOS as primary and secondary to your shared memory

partitions to load balance AMS paging activity. Keep in mind that if VIOS maintenance or failure occurs, all your partitions will be on the same remaining VIOS. You will have to force a failover for some of your partitions when the updated/failing VIOS is back online. To do so use the Management Console command line as shown in Example 4-6 to evenly spread your partitions over the two paging VIOS.

## Paging devices

With dual VIOS configuration, if your partition has a primary and a secondary paging VIOS defined, both VIOS will need to access a common paging device with a size greater than or equal to the partition maximum memory, otherwise the partition will fail to activate. It is still possible to force the activation of a partition using a non redundant paging device by using the **chsysstate** command with the **--force** flag on the Management Console as shown on Example 4-7; however, the partition will not support failover.

### Example 4-7

---

```
hscroot@hmc9:~> chsysstate -r lpar -m POWER7_1-061AA6P -o on -n AIX1C -f AMS
HSCLA47C Partition AIX1C cannot be activated with the paging Virtual I/O Server (VIOS)
partition configuration specified in the profile because one of the paging VIOS partitions
is not available, or a paging space device that can be used with that paging VIOS
configuration is not available. However, this partition can be activated with a different
paging VIOS partition configuration now. If this partition is configured to use redundant
paging VIOS partitions, then this partition can be activated to use a non-redundant paging
VIOS partition. If this partition is configured to use non-redundant paging VIOS
partitions, then this partition can be activated to use a different paging VIOS partition
than the one specified in the profile. If you want to activate this partition with the
paging VIOS configuration that is available now, then run the chsysstate command with the
--force option to activate this partition.
```

```
hscroot@hmc9:~> chsysstate -r lpar -m POWER7_1-061AA6P -o on -n AIX1C -f AMS --force
```

---

Archived





# Monitoring

Metrics for monitoring Active Memory Sharing are available on:

- ▶ Management Console
- ▶ AIX or IBM i shared memory partitions
- ▶ Virtual I/O Server
- ▶ Linux

This chapter describes the commands and tools that are available.

## 5.1 Management Console

The Management Console provides Active Memory Sharing statistics through a utilization data collection feature. A mempool resource type has been added to the **lslparutil** command. Some fields have also been added to the lpar resource type.

The following fields are available for the mempool resource type:

<b>curr_pool_mem</b>	Current amount of physical memory allocated to the shared memory pool.
<b>lpar_curr_io_entitled_mem</b>	Current amount of physical memory defined as I/O entitled memory by all shared memory partitions.
<b>lpar_mapped_io_entitled_mem</b>	Current amount of physical memory actually used as I/O entitled memory by all shared memory partitions.
<b>page_faults</b>	Number of page faults by all shared memory partitions since system was booted.
<b>lpar_run_mem</b>	Current amount of physical memory used by all shared memory partitions.
<b>page_in_delay</b>	Page-in delay for all shared memory partitions since system was booted.
<b>sys_firmware_pool_mem</b>	The amount of memory in the shared pool that is reserved for use by the firmware.

The following fields are available for the lpar resource type:

<b>mem_mode</b>	Memory mode of the partition (ded for dedicated memory partitions; shared for shared memory partitions).
<b>curr_mem</b>	Current amount of logical memory for the partition.
<b>curr_io_entitled_mem</b>	Current amount of logical memory defined as I/O entitled memory for the partition.
<b>mapped_io_entitled_mem</b>	Current amount of logical memory actually mapped as I/O entitled memory for the partition.
<b>phys_run_mem</b>	Current amount of physical memory allocated to the partition by the hypervisor.
<b>run_mem_weight</b>	Current memory weight for the partition.
<b>mem_overage_cooperation</b>	Current amount of logical memory that is not backed by physical memory. This will be a negative value because it is calculated as <b>phys_run_mem</b> minus <b>curr_mem</b> . Note, at the time of writing, this resource type is not part of the IVM.

The performance data is taken in a timely manner by the management console. The default sampling is 5 minutes, but you can change the value to better meet your requirements, using the **ch1parutil** command. In Example 5-1 we used the **lslparutil** command to list the current **sample\_rate** for both systems then **ch1parutil** to change to 1 minute only in the POWER7\_1-061AA6P system.

*Example 5-1 Listing and changing the sample rate*

---

```
hscroot@hmc9:~> lslparutil -r config
type_model_serial_num=8233-E8B*061AA6P,name=POWER7_1-061AA6P,sample_rate=300
type_model_serial_num=8233-E8B*061AB2P,name=POWER7_2-061AB2P,sample_rate=300
```

```
hscroot@hmc9:~> chlparutil -r config -m POWER7_2-061AB2P -s 60
```

```
hscroot@hmc9:~> lslparutil -r config  
type_model_serial_num=8233-E8B*061AA6P,name=POWER7_1-061AA6P,sample_rate=60  
type_model_serial_num=8233-E8B*061AB2P,name=POWER7_2-061AB2P,sample_rate=300
```

---

The **lslparutil** in Example 5-2 shows how to get information based on the performance data from the last 24 hours. It is also possible to check the last days (**-d**), minutes (**--minutes**), and others. For a complete reference refer to the **lslparutil** man pages. The output of the following example as been truncated for readability purposes.

*Example 5-2 Using lslparutil to monitor memory utilization from LPARs running on AMS mode*

---

```
hscroot@hmc9:~> lslparutil -m POWER7_2-061AB2P -d 7 -r lpar -F  
time,lpar_name,phys_run_mem
```

```
04/28/2011 11:46:31,IBMi2A,2048  
04/28/2011 11:46:31,IBMi2B,488  
04/28/2011 11:46:31,LINUX2A,699  
04/28/2011 11:46:31,LINUX2B,473  
04/28/2011 11:46:31,AIX2A,1120  
04/28/2011 11:46:31,AIX2B,1156  
[...]  
04/27/2011 12:10:05,IBMi2A,2048  
04/27/2011 12:10:05,IBMi2B,2076  
04/27/2011 12:10:05,LINUX2A,2641  
04/27/2011 12:10:05,LINUX2B,5282  
04/27/2011 15:46:05,AIX2A,1431  
04/27/2011 15:46:05,AIX2B,1373
```

---

When displaying the utilization with the Management Console GUI, additional values are calculated from the collected data, such as memory overcommitment. Figure 5-1 shows an example. To display the utilization data with the Management Console GUI, select a managed system and then **Operations** → **Utilization Data** → **View**. Utilization data collection has to be enabled for the data to be available.

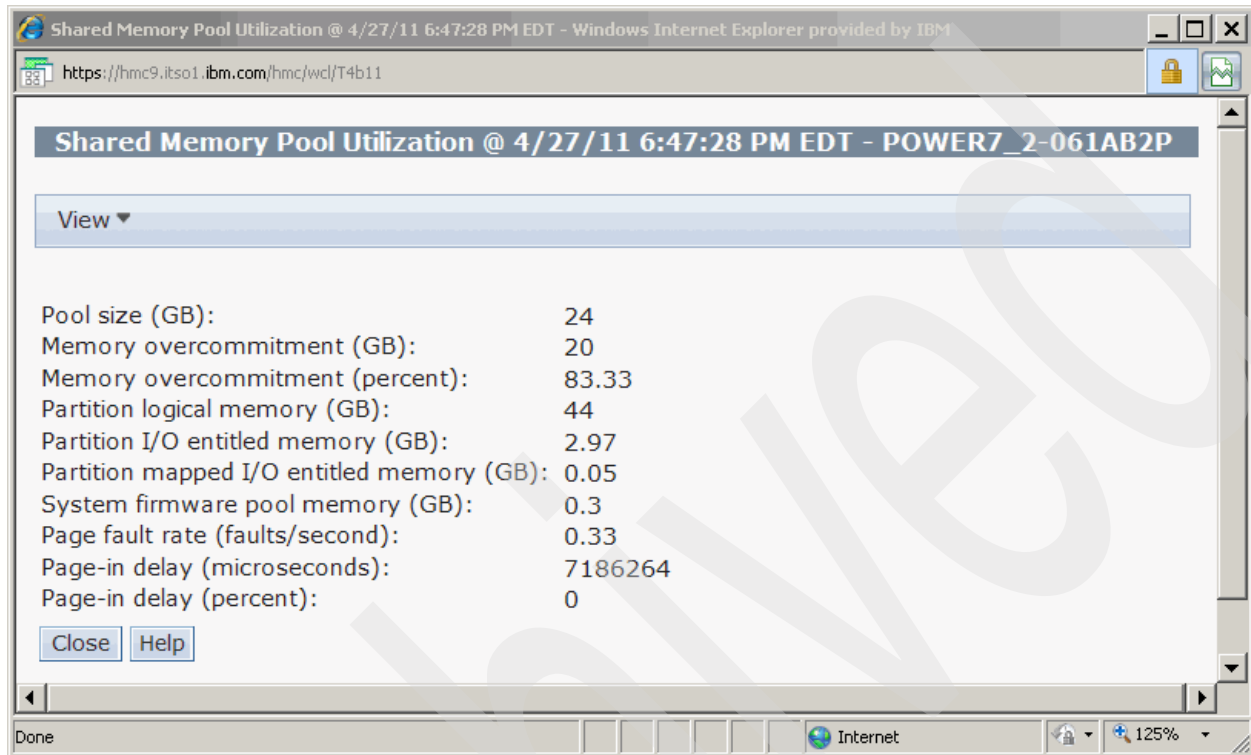


Figure 5-1 Displaying shared memory pool utilization using the Management Console

As shown in Figure 5-2 on page 63, you can also check the memory distribution per partition from the Management Console by selecting a managed system, then **Operations** → **Utilization Data** → **View**, selecting a snapshot, **View** → **Partition** → **Memory**.

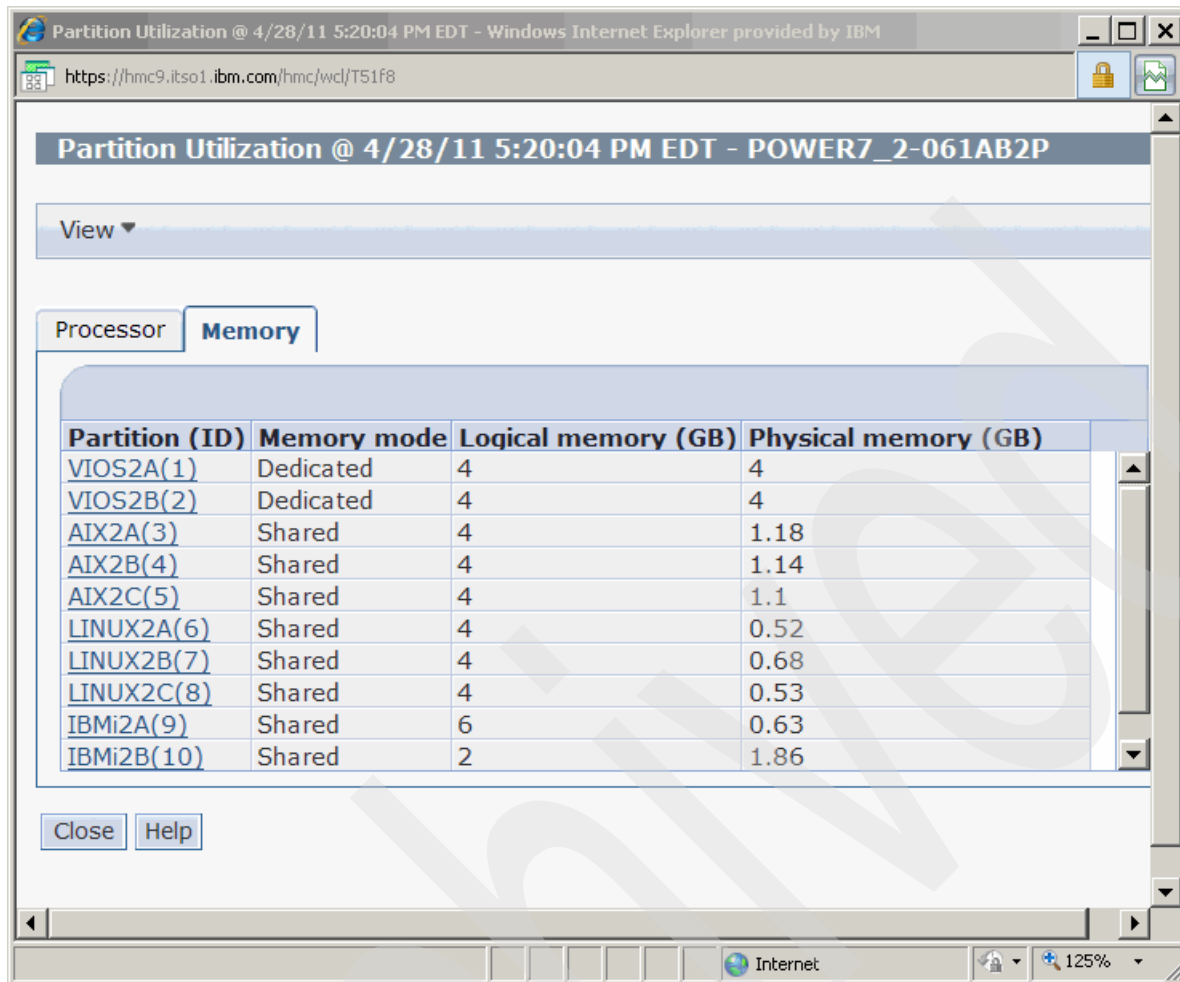


Figure 5-2 Memory utilization per partition

The I/O entitled memory for a specific shared memory partition can be displayed using the partition properties. Select a partition, then click **Properties** → **Hardware** → **Memory**. Click the Memory Statistics button to display the window, as shown in Figure 5-3 on page 64. The following values are displayed:

**Assigned I/O Entitled Memory**

I/O entitled memory assigned to the partition by the Management Console or IVM.

**Minimum I/O Entitled Memory Usage**

Minimum I/O entitled memory reported by the OS, or through the Management Console or IVM using the **lshwres** command.

**Optimal I/O Entitled Memory Usage**

Optimal amount of I/O entitled memory reported by the OS, Management Console, or IVM.

**Maximum I/O Entitled Memory Usage**

High water mark of used I/O entitled memory reported by the OS. The high water mark can be reset using the Reset Statistics button, or using the **chhwres** command on the Management Console or IVM interface.

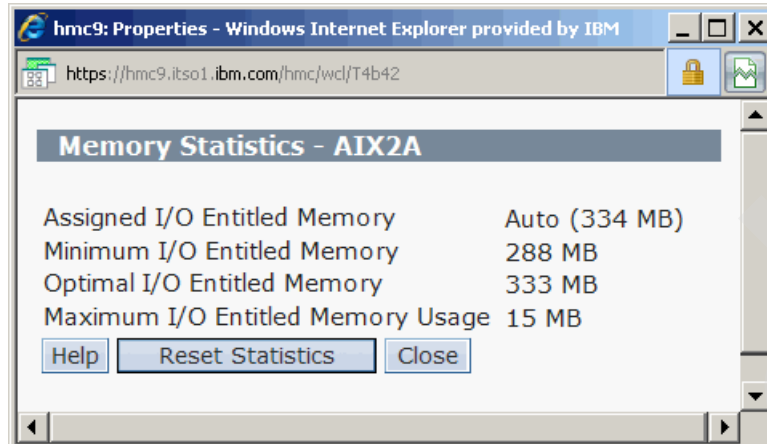


Figure 5-3 I/O entitled memory statistics

## 5.2 Virtual I/O Server monitoring

The I/O performance of the physical disks or logical volumes that serve as paging devices on the Virtual I/O Server should be monitored. The Virtual I/O Server provides the **viostat** and **topas** commands for displaying I/O performance data.

In AMS deployments, Virtual I/O Server has the critical role to provide paging devices to the shared pool; therefore, the most important aspects to monitor in a VIOS are processors and disk I/O utilization for the paging devices.

In terms of processing you can run the **topas** command to see how the VIOS is performing, as shown in the Example 5-3.

Example 5-3 Monitoring VIOS using topas

Topas Monitor for host: VIOS2A							EVENTS/QUEUES		FILE/TTY	
Thu Apr 28 09:36:54 2011 Interval: 2							Cswitch	369	Readch	2781
							Syscall	263	Writech	245
CPU	User%	Kern%	Wait%	Idle%	Physc	Entc	Reads	31	Rawin	0
ALL	0.1	4.0	0.0	95.9	0.04	7.0	Writes	3	Ttyout	85
							Forks	0	Igets	0
Network	KBPS	I-Pack	O-Pack	KB-In	KB-Out		Execs	0	Namei	53
Total	36.4	27.4	2.0	36.1	0.3		Runqueue	1.0	Dirblk	0
							Waitqueue	0.0		
Disk	Busy%	KBPS	TPS	KB-Read	KB-Writ		MEMORY			
Total	0.0	0.0	0.0	0.0	0.0		PAGING	Real,MB	4096	
							Faults	0	% Comp	24
FileSystem		KBPS	TPS	KB-Read	KB-Writ		Steals	0	% Noncomp	5
Total		2.7	30.4	2.7	0.0		PgspIn	0	% Client	5
							PgspOut	0		
Name	PID	CPU%	PgSp	Owner			PageIn	0	PAGING SPACE	
pager0	2490458	2.8	1.0	root			PageOut	0	Size,MB	1536
vmmd	458766	0.6	1.2	root			Sios	0	% Used	1
topas	5308632	0.2	1.6	padmin					% Free	99
xmgc	851994	0.2	0.4	root			NFS (calls/sec)			
seaproc	7274572	0.1	1.0	padmin			SerV2	0	WPAR Activ	0
getty	7667830	0.0	0.6	root			Cliv2	0	WPAR Total	0

sched	196614	0.0	0.4	root	SerV3	0	Press: "h"-help
gil	1441836	0.0	0.9	root	Cliv3	0	"q"-quit

To identify what disks are being used as paging devices at the moment, use **lshwres** on the Management Console, as shown in Example 5-4. Paging activity can be monitored by checking I/O activity on disks used as paging devices, using the **viostat** command as shown in Example 5-5. In cases where logical volumes are being used as paging devices you can check how these devices are performing using the **lvmstat** command, as shown in Example 5-6.

*Example 5-4 Disks being used as paging devices at the moment*

```
hscroot@hmc9:~> lshwres -r mempool --rsubtype pgdev -m POWER7_2-061AB2P -F
paging_vios_name,device_name,lpar_name
VIO2A,hdisk3,AIX2C
VIO2A,hdisk4,AIX2B
VIO2A,hdisk5,AIX2A
VIO2A,hdisk6,LINUX2C
VIO2A,hdisk7,LINUX2A
VIO2A,hdisk8,LINUX2B
VIO2A,hdisk9,IBMi3C
VIO2A,hdisk10,IBMi2B
VIO2A,hdisk11,IBMi2A
```

*Example 5-5 Using viostat to check disk performance*

```
$ viostat

System configuration: lcpu=8 drives=12 ent=0.50 paths=39 vdisks=26
tty:      tin      tout      avg-cpu: % user % sys % idle % iowait physc % entc
          0.0      0.3              0.0  0.1  99.9    0.0  0.0   0.2

Disks:    % tm_act   Kbps      tps      Kb_read   Kb_wrtn
hdisk0    0.0       11.4      0.5      3951847   2740372
hdisk1    0.0       75.7      1.6      17345601  27235122
hdisk2    0.0        0.0      0.0        185        0
hdisk3    0.0        0.8      0.2       37764     447392
hdisk4    0.0        0.7      0.1       17824     386204
hdisk5    0.0        1.3      0.3      185004     584404
hdisk6    0.0        0.4      0.1       33948     186540
hdisk7    0.0        0.0      0.0        37         0
hdisk8    0.0        0.0      0.0        37        36
hdisk9    0.0        0.3      0.1       36309     126086
hdisk10   0.0        0.8      0.2       98251     346357
hdisk11   0.0        0.9      0.2      120018     385034
```

*Example 5-6 Checking paging device performance*

```
$ oem_setup_env
# lvmstat -v amspaging
Logical Volume   iocnt    Kb_read   Kb_wrtn    Kbps
  amspaging03    64372    257488        0    0.98
```

**Note:** LVM statistics collection must be enabled for you to use **lvmstat**. Enable this feature with either of these commands: **lvmstat -e -v <vg\_name>** or **lvmstat -e -l <lv\_name>**.

## 5.3 Monitoring AIX

The following AIX commands have been enhanced to provide information for Active Memory Sharing:

- ▶ **vmstat**
- ▶ **lparstat**
- ▶ **topas**

### 5.3.1 The vmstat command

Hypervisor paging information is displayed when the **vmstat** command is issued with the **-h** option, as shown in Example 5-7.

*Example 5-7 Displaying hypervisor paging information using vmstat -h*

# vmstat -h 10

System configuration: lcpu=8 mem=10240MB ent=1.00 **mmode=shared mpsz=20.00GB**

kthr		memory		page				faults				cpu				hypv-page						
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa	pc	ec	hpi	hpit	pmem	loan
3	0	1245552	115568	0	0	0	0	0	0	7	742	219	11	10	79	0	0.25	25.5	326	6	3.97	4.64
4	0	1247125	111240	0	0	0	0	0	0	2	543	199	11	4	85	0	0.18	17.7	274	9	3.97	4.65
4	0	1248872	108790	0	0	0	0	0	0	4	593	199	13	7	81	0	0.21	20.8	284	9	3.98	4.65
4	0	1251021	101579	0	0	0	0	0	0	1	608	252	15	7	79	0	0.24	24.1	246	10	3.98	4.67
2	0	1252582	93616	0	0	0	0	0	0	2	691	193	12	7	82	0	0.21	20.9	241	9	3.98	4.69
2	0	1255414	86513	0	0	0	0	0	0	1	754	223	27	8	65	0	0.39	39.1	237	7	3.99	4.71

The following fields highlighted in Example 5-7 have been added for Active Memory Sharing:

**mmode** Shows shared if the partition is running in shared memory mode. This field is not present on dedicated memory partitions.

**mps** Size of the shared memory pool.

**hpi** Number of hypervisor page-ins for the partition. A hypervisor page-in occurs if a page is being referenced that is not available in real memory because it has been paged out by the hypervisor previously.

**hpit** Time spent in hypervisor paging in milliseconds for the partition.

**pmem** Amount of physical memory backing the logical memory, in gigabytes.

**loan** Amount of the logical memory, in gigabytes, which is loaned to the hypervisor. The amount of loaned memory can be influenced through the `vmo_ams_loan_policy` tunable. See 6.2, “Shared memory partition tuning” on page 86 for more details.

As highlighted in Example 5-8 on page 67, the **vmstat -v -h** command shows the number of AMS memory faults and the time spent in milliseconds for hypervisor paging since boot time. It also shows the number of 4 KB pages that AIX has loaned to the hypervisor and the percentage of partition logical memory that has been loaned.



Example 5-8 Displaying hypervisor paging information using `vmstat -v -h`

---

```
# vmstat -v -h
2621440 memory pages
2407164 lruable pages
1699043 free pages
    1 memory pools
365286 pinned pages
    90.0 maxpin percentage
    3.0 minperm percentage
    90.0 maxperm percentage
    2.3 numperm percentage
56726 file pages
    0.0 compressed percentage
    0 compressed pages
    2.3 numclient percentage
    90.0 maxclient percentage
56726 client pages
    0 remote pageouts scheduled
    0 pending disk I/Os blocked with no pbuf
14159 paging space I/Os blocked with no psbuf
2228 filesystem I/Os blocked with no fsbuf
    0 client filesystem I/Os blocked with no fsbuf
    0 external pager filesystem I/Os blocked with no fsbuf
421058 Virtualized Partition Memory Page Faults
3331187 Time resolving virtualized partition memory page faults
488154 Number of 4k page frames loaned
    18 Percentage of partition memory loaned
    33.0 percentage of memory used for computational pages
```

---

If the Virtualized Partition Memory Faults are increasing, it means that hypervisor paging has occurred at some time. It can be an indicator that the physical memory is overcommitted.

### Impact of Active Memory Sharing on `vmstat` metrics

When using Active Memory Sharing some of the metrics displayed by the `vmstat` command do not represent the same information displayed for a dedicated memory partition. A similar effect occurs in the percentage used CPU metrics when shared processor partitions are used.

The **mem** field shows the amount of available logical memory. Unlike in a dedicated memory partition, where the logical memory is always backed by physical memory, this is not the case in a shared memory partition. The partition in Example 5-7 on page 66 has 10 GB of logical memory. This does not mean that there is actually this amount of logical memory available to the partition. To see how much physical memory the partition currently has assigned you have to look at the **pmem** column. In this case it shows that the partition only has 3.99 GB of physical memory assigned at this moment.

The **fre** column shows the number of free logical pages. As Example 5-9 on page 68 shows, this is not necessarily true in a shared memory partition, where the **fre** column shows the amount of free logical memory. Although this column shows 8.4 GB (2203204 4-KB pages) as free, the physical memory actually assigned is only 5.39 GB. This means that the hypervisor has stolen memory from the partition and provided it to another partition. The partition shown in Example 5-9 has the `ams_loan_policy` set to 0. Therefore, no memory has been loaned.

#### Example 5-9 Shared memory partition with some free memory not backed by physical memory

```
# vmstat -h 2
```

System configuration: lcpu=8 mem=10240MB ent=1.00 mmode=shared mpsz=20.00GB

kthr		memory				page				faults				cpu				hypv-page					
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa	pc	ec	hpi	hpit	pmem	loan	
0	0	444694	<b>2203204</b>	0	0	0	0	0	0	0	7	3711	552	1	1	98	0	0.03	2.9	0	0	<b>5.39</b>	0.00
0	0	444699	2203199	0	0	0	0	0	0	0	4	3187	539	0	1	98	0	0.03	2.5	0	0	5.39	0.00
0	0	444701	2203197	0	0	0	0	0	0	0	3	3290	541	1	1	98	0	0.03	2.7	0	0	5.39	0.00
0	0	444699	2203199	0	0	0	0	0	0	0	0	725	113	1	1	98	0	0.03	3.2	0	0	5.39	0.00

When loaning is enabled (ams\_loan\_policy is set to 1 or 2), AIX will loan pages when the hypervisor initiates a request. AIX will remove free pages that are loaned to the hypervisor from the free list. Example 5-10 shows a partition that has a logical memory size of 10 GB. It has also assigned 9.94 GB of physical memory. Of this assigned 9.94 GB of physical memory, 8.3 GB (2183045 4-KB pages) is free because there is no activity in the partition.

#### Example 5-10 Shared memory partition not loaning memory

```
# vmstat -h 2
```

System configuration: lcpu=8 mem=10240MB ent=1.00 mmode=shared mpsz=20.00GB

kthr		memory			page				faults				cpu				hypv-page						
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa	pc	ec	hpi	hpit	pmem	loan	
1	0	442321	2183045		0	0	0	0	0	0	4	652	119	1	1	98	0	0.03	3.4	0	0	9.94	0.00
1	0	442320	2183046		0	0	0	0	0	0	12	3365	527	1	1	98	0	0.03	3.1	0	0	9.94	0.00
1	1	442320	2182925		0	42	0	0	0	0	83	1810	392	1	2	97	0	0.04	4.1	0	0	9.94	0.00

Example 5-11 shows the same partition a few minutes later. In the intervening time, the hypervisor requested memory and the partition loaned 3.30 GB to the hypervisor. AIX has removed the free pages that it has loaned from the free list. The free list has therefore been reduced by 875634 4-KB pages.

#### Example 5-11 Shared memory partition loaning memory

```
# vmstat -h 2
```

System configuration: lcpu=8 mem=10240MB ent=1.00 mmode=shared mpsz=20.00GB

kthr		memory				page				faults				cpu				hypv-page					
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa	pc	ec	hpi	hpit	pmem	loan	
0	0	442355	1307291		0	0	0	0	0	0	20	3416	574	1	2	98	0	0.04	3.7	0	0	6.70	3.30
1	0	442351	1307295		0	0	0	0	0	0	17	3418	588	1	1	98	0	0.03	2.7	0	0	6.70	3.30
2	0	442353	1307293		0	0	0	0	0	0	15	3405	558	1	1	98	0	0.03	2.7	0	0	6.70	3.30

From a performance perspective it is important to monitor the number of hypervisor page-ins. If there are non-zero values it means that the partition has to wait for pages to be brought into real memory by the hypervisor. Because these pages have to be read from a paging device, the page-ins have an impact on application performance.

Example 5-7 on page 66 shows a situation where pages for a shared memory partition are being paged in by the hypervisor. The **hpi** column shows activity.

## AIX paging and hypervisor paging

When using Active Memory Sharing, paging can occur on the AIX level or on the hypervisor level. When you see non-zero values in the **pi** or **po** column of `vmstat`, it means that AIX is performing paging activities.

In a shared memory partition, AIX paging occurs not only when the working set exceeds the size of the logical memory, as in a dedicated partition. This can happen even if the LPAR has less physical memory than logical memory. AIX is dependent on the amount of logical memory available. So, if an LPAR is configured with 4 GB of logical memory but is running a workload that uses 4.5 GB of memory, the workload will page to the AIX paging spaces regardless of how much physical memory the hypervisor has allocated to the LPAR.

Another reason is that AIX is freeing memory pages to loan them to the hypervisor. If the loaned pages are used pages, AIX has to save the content to its paging space before loaning them to the hypervisor. This behavior will especially occur if you have selected an aggressive loaning policy (`ams_loan_policy=2`).

### 5.3.2 The `lparstat` command

The `lparstat` command has been enhanced to display statistics about shared memory. Most of the metrics show the I/O entitled memory statistics.

**Note:** The I/O memory entitlement does not have to be changed unless you encounter I/O performance problems. In such a case, check the **iomaf** column for failed I/O allocation requests for I/O memory entitlement.

When using the `lparstat -m` command the following attributes are displayed (Example 5-12 on page 70).

<b>mpsz</b>	Size of the memory pool in GB
<b>iome</b>	I/O memory entitlement in MB
<b>iomp</b>	Number of I/O memory entitlement pools. Details about each pool can be displayed using the <code>lparstat -me</code> command
<b>hpi</b>	Number of hypervisor page-ins
<b>hpit</b>	Time spent in hypervisor paging in milliseconds
<b>pmem</b>	Allocated physical memory in GB
<b>iomin</b>	Minimum I/O memory entitlement for the pool
<b>iomu</b>	Used I/O memory entitlement in MB
<b>iomf</b>	Free I/O memory entitlement in MB
<b>iohwm</b>	High water mark of I/O memory entitlement usage in MB
<b>iomaf</b>	Number of failed allocation requests for I/O memory entitlement

#### Example 5-12 The lparstat -m command

```
# lparstat -m 1
```

System configuration: lcpu=8 mem=10240MB mpsz=20.00GB iome=334.00MB iomp=10 ent=1.00

physb	hpi	hpit	pmem	iomin	iomu	iomf	iohwm	iomaf	%entc	vcs
1.82	0	0	8.95	287.7	12.4	34.3	18.0	0	3.1	496
1.62	0	0	8.95	287.7	12.4	34.3	18.0	0	2.6	596
1.64	0	0	8.95	287.7	12.4	34.3	18.0	0	2.6	592
1.60	0	0	8.95	287.7	12.4	34.3	18.0	0	2.5	601
1.64	0	0	8.95	287.7	12.4	34.3	18.0	0	2.5	598

When using the **lparstat -me** command the I/O memory entitlement details for the partition are displayed as shown in Example 5-13.

#### Example 5-13 The lparstat -me command

```
# lparstat -me
```

System configuration: lcpu=8 mem=10240MB mpsz=20.00GB iome=334.00MB iomp=10 ent=1.00

physb	hpi	hpit	pmem	iomin	iomu	iomf	iohwm	iomaf	%entc	vcs
0.11	1055380	10218330	8.95	287.7	12.4	34.3	18.0	0	0.2	67360383
iompn: iomin iodes iomu iores iohwm iomaf										
ent0.txpool			2.12	16.00	2.00	2.12	2.00	0		
ent0.rxpool_4			4.00	16.00	3.50	4.00	3.50	0		
ent0.rxpool_3			4.00	16.00	2.00	16.00	2.70	0		
ent0.rxpool_2			2.50	5.00	2.00	2.50	2.00	0		
ent0.rxpool_1			0.84	2.25	0.75	0.84	0.75	0		
ent0.rxpool_0			1.59	4.25	1.50	1.59	1.50	0		
ent0.phypmem			0.10	0.10	0.09	0.10	0.09	0		
fcs1			136.25	136.25	0.30	136.25	2.80	0		
fcs0			136.25	136.25	0.30	136.25	2.69	0		
sys0			0.00	0.00	0.00	0.00	0.00	0		

**iodes** Desired entitlement of the I/O memory pool in MB

**iores** Reserved entitlement of the I/O memory pool in MB

### 5.3.3 The topas command

When using the **topas -L** command, the logical partition view with the Active Memory Sharing statistics highlighted in bold in Example 5-14 is displayed. The **IOME** field shows the I/O memory entitlement configured for the partition, whereas the **iomu** column shows the I/O memory entitlement in use. For a description of the other fields, refer to 5.3.1, “The vmstat command” on page 66.

#### Example 5-14 The topas -L command

Interval:2	Logical Partition: AIX1C	Fri Apr 29 11:08:02 2011
Psize: 16.0	Shared SMT 4	Online Memory: 10.00G
	Power Saving: Disabled	
Ent: 1.00	Mode: Un-Capped	Online Logical CPUs: 8
<b>Mmode: Shared</b>	<b>IOME: 334.00</b>	Online Virtual CPUs: 2

#### Partition CPU Utilization

%usr	%sys	%wait	%idle	physc	%entc	app	vcs	phint	hpi	hpit	pmem	iomu
0.5	1.1	0.0	98.4	0.03	2.55	13.92	394	0	0	0	8.95	12.45

---

LCPU	MINPF	MAJPF	INTR	CSW	ICSW	RUNQ	LPA	SCALLS	USER	KERN	WAIT	IDLE	PHYSC	LCSW
1	411	0	118	121	56.0	0	101	1.52K	27.6	60.7	0.0	11.7	0.01	137
4	0	0	23.0	0	0	0	0	0	0.0	53.5	0.0	46.5	0.00	19.0
0	317	0	256	150	84.0	0	100	115.00	25.5	46.3	0.0	28.2	0.01	215
3	0	0	9.00	0	0	0	0	0	0.0	0.6	0.0	99.4	0.00	9.00
2	0	0	9.00	0	0	0	0	0	0.0	0.6	0.0	99.4	0.00	9.00

Press the **e** key to view details about the I/O memory entitlement, as shown in Example 5-15.

#### Example 5-15 Displaying I/O memory entitlement using topas

Interval:2	Logical Partition: AIX1C	Fri Apr 29 11:10:28 2011
Psize: 16.0	Shared SMT 4	Online Memory: 10.00G
	Power Saving: Disabled	
Ent: 1.00	Mode: Un-Capped	Online Logical CPUs: 8
Mmode: Shared	IOME: 334.00	Online Virtual CPUs: 2

Partition CPU Utilization									
physb	%entc	vcs	hpi	hpit	pmem	iomu	iomf	iohwm	iomaf
0.0	3.0	364	15	65	8.95	12.45	34.34	18.03	0.00

---

iomprn	iomin	iodes	iomu	iores	iohwm	iomaf
fcs1	136.2	136.2	0.3	136.2	2.8	0.0
fcs0	136.2	136.2	0.3	136.2	2.7	0.0
ent0.rxp001_3	4.0	16.0	2.0	16.0	2.7	0.0
ent0.rxp001_4	4.0	16.0	3.5	4.0	3.5	0.0
ent0.rxp001_2	2.5	5.0	2.0	2.5	2.0	0.0
ent0.txp001	2.1	16.0	2.0	2.1	2.0	0.0
ent0.rxp001_0	1.6	4.2	1.5	1.6	1.5	0.0
ent0.rxp001_1	0.8	2.2	0.8	0.8	0.8	0.0
ent0.phypmem	0.1	0.1	0.1	0.1	0.1	0.0
sys0	0.0	0.0	0.0	0.0	0.0	0.0

### Monitoring multiple partitions with topas

The **topas -C** command displays the CEC view, which is especially useful for monitoring multiple partitions on a managed system. The Active Memory Sharing statistics highlighted in bold in Example 5-16 on page 72 are displayed. The following values are shown:

- pmem** Amount of physical memory assigned to each individual partition.
- Mode** Mode in which the partition is running. The mode is displayed in a set of 3 characters. See Table 5-1 for an explanation.

Table 5-1 Partition mode

The first character indicates the CPU in the partition	The second character indicates the memory mode of the partition.	The third character indicates the energy state of the partition
<ul style="list-style-type: none"> <li>► For shared partitions: <ul style="list-style-type: none"> <li>– <b>C</b> SMT enabled and capped</li> <li>– <b>c</b> SMT disabled and capped</li> <li>– <b>U</b> SMT enabled and uncapped</li> <li>– <b>u</b> SMT disabled and uncapped</li> </ul> </li> <li>► For dedicated partitions: <ul style="list-style-type: none"> <li>– <b>S</b> SMT enabled and not donating</li> <li>– <b>d</b> SMT disabled and donating</li> <li>– <b>D</b> SMT enabled and donating</li> <li>– - SMT disabled and not donating</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>► <b>M</b> In shared memory mode (For shared partitions) and AME disabled</li> <li>► - Not in shared memory mode and AME disabled</li> <li>► <b>E</b> In shared memory mode and AME enabled</li> <li>► <b>e</b> Not in shared memory mode and AME enabled</li> </ul>	<ul style="list-style-type: none"> <li>► <b>S</b> Static power save mode enabled</li> <li>► <b>d</b> Power save mode is disabled</li> <li>► <b>D</b> Dynamic power save mode enabled</li> <li>► - Unknown / Undefined</li> </ul>

**Note:** Monitoring multiple partitions with topas is not supported for Linux partitions.

#### Example 5-16 The topas -C command

```

Topas CEC Monitor          Interval: 10          Fri Apr 29 14:59:04 2011
Partitions Memory (GB)      Processors
Shr: 5   Mon:38.0 InUse:13.0 Shr: 4   PSz: 16   Don: 0.0 Shr_PhysB 0.11
Ded: 0   Avl: -            Ded: 0   APP: 31.7 Stl: 0.0 Ded_PhysB 0.00

Host      OS  Mod Mem InU Lp  Us Sy Wa Id  PhysB  Vcsw Ent  %EntC PhI  pmem
-----shared-----
VIOS1A    A61 Ued 4.0 1.1 8    1 1 0 96  0.03  0  0.50  5.8  0    -
AIX1A     A71 UEd 10 4.7 8    0 1 0 98  0.03  966  1.00  2.6  0  1.89
AIX1C     A71 UEd 10 1.8 8    0 1 0 98  0.03  0  1.00  2.6  0  4.69
VIOS1B    A61 Ued 4.0 1.1 8    0 1 0 98  0.02  0  0.50  3.6  0    -
AIX1B     A71 UEd 10 4.8 8    0 0 0 99  0.02  0  1.00  1.6  0  3.55

Host      OS  Mod Mem InU Lp  Us Sy Wa Id  PhysB  Vcsw %istl %bstl
-----dedicated-----

```

**Note:** In the test environment used at the time of writing this paper, this version of topas had an incorrect output regarding the **Mod** column: none of these partitions was using Active Memory Expansion by the time this screenshot was taken. The column should have displayed: **UMd** for the shared-memory partitions or **U-d** for the VIOS.

Press the **m** key in the CEC view to display the attributes of the shared memory pool (Example 5-17).

#### Example 5-17 Displaying shared memory pool attributes using topas

```

Topas CEC Monitor          Interval: 10          Fri Apr 29 15:00:24 2011
Partitions Memory (GB)      Memory Pool(GB)      I/O Memory(GB)
Mshr: 3   Mon: 38.0 InUse: 13.5 MPSz: 20.0 MPUse: 10.1 Entl: 1002.0se: 37.3
Mded: 2   Avl: 24.5          Pools: 1

mpid  mpsz  mpus  mem  memu  iome  iomu  hpi  hpit
-----
0      20.00 10.13 30.00 11.41 1002.0 37.3  0    0

```

## 5.4 Monitoring IBM i

Information about the Active Memory Sharing configuration and paging activity that is provided by the POWER Hypervisor is available above the Machine Interface layer to IBM i. The IBM i Collection Services have been enhanced and new data structures have been added to obtain that information.

The fields identified in Table 5-2 are useful for evaluating partition performance behaviors due to real memory paging activity by the hypervisor.

Table 5-2 QAPMSHRMP field details

Field name	Description
INTNUM	Interval number. The nth sample database interval based on the start time specified in the Create Performance Data (CRTPFRTA) command.
DATETIME	Interval date and time. The date and time of the sample interval.
INTSEC	Elapsed interval seconds. The number of seconds since the last sample interval.
SMPOOLID	Shared memory pool identifier. The identifier of the shared memory pool which this partition is using.
SMWEIGHT	Memory weight. Indicates the variable memory capacity weight assigned to the partition. Valid values are hex 0 - 255. The larger the value, the less likely this partition is to lose memory.
SMREALUSE	Physical real memory used. The amount of shared physical real memory, in bytes, that was being used by partition memory at sample time.
SMACCDLY	Real memory access delays. The number of partition processor waits that have occurred because of page faults on logical real memory.
SMACCCWAIT	Real memory access wait time. The amount of time, in milliseconds, that partition processors have waited for real memory page faults to be satisfied.
SMOVRCAP	Partition memory overhead capacity. The maximum amount of memory, in bytes, that the partition is allowed to assign to data areas shared between the partition operating system and the firmware.
SMENTIOC	Entitled memory capacity for I/O. The amount of memory, in bytes, currently assigned to the partition for use by I/O requests.
SMMINIOC	Minimum entitled memory capacity for I/O. The minimum amount of entitled memory, in bytes, needed to function with the current I/O configuration.
SMOPTIOC	Optimal entitled memory capacity for I/O. The amount of entitled memory, in bytes, that would allow the current I/O configuration to function without any I/O memory mapping delays.
SMIOCUSE	Current I/O memory capacity in use. The amount of I/O memory, in bytes, currently mapped by I/O requests.
SMIOCMAX	Maximum I/O memory capacity used. The maximum amount of I/O memory, in bytes, that has been mapped by I/O requests since the partition was last IPLed or the value was reset by an explicit request.
SMIOMDLY	I/O memory mapping delays. The cumulative number of delays that have occurred because insufficient entitled memory was available to map an I/O request since the partition was last IPLed.

Field name	Description
MPACCDLY	Pool real memory access delays. The number of virtual partition memory page faults within the shared memory pool for all partitions.
MPACCWAIT	Pool real memory access wait time. The amount of time, in milliseconds, that all partitions' processors have spent waiting for page faults to be satisfied within the shared memory pool.
MPPHYMEM	Pool physical memory. The total amount of physical memory, in bytes, assigned to the shared memory pool.
MPLOGMEM	Pool logical memory. The summation, in bytes, of the logical real memory of all active partitions served by the shared memory pool.
MPENTIOC	Pool entitled I/O memory. The summation, in bytes, of the I/O entitlement of all active partitions served by the shared memory pool.
MPIOCUSE	Pool entitled I/O memory in use. The summation, in bytes, of I/O memory mapped by I/O requests from all active partitions served by the shared memory pool.

**Note:** Active Memory Sharing data is only populated in partitions that are part of the shared memory pool.

The expected consumer of this information for the initial release of the Active Memory Sharing support is the collection services, with the intent to log as much data as possible in existing DB fields. Other potential consumers of the information gathered are programs written specifically to retrieve this information using service/performance APIs.

### 5.4.1 Checking the QAPMSHRMP table using SQL

Example 5-18 shows how to obtain Active Memory Sharing data using the Start Structured Query Language (STRSQL) command.

*Example 5-18 Sample query to gather QAPMSHRMP data*

---

```
SELECT * FROM QPFRDATA/QAPMSHRMP
```

---



Display Data

Data width . . . . . : 691

Position to line . . . . . Shift to column . . . . .

.....+.....1.....+.....2.....+.....3.....+.....4.....+.....5.....+.....6.....+.....7.....+.....

Interval number	Interval date time	Elapsed interval seconds	Memory pool ID	Partition capacity weight
1	2011-04-22-17.25.00.000000	242	0	128
2	2011-04-22-17.30.00.000000	300	0	128
3	2011-04-22-17.35.00.000000	300	0	128
4	2011-04-22-17.40.00.000000	300	0	128
5	2011-04-22-17.45.00.000000	300	0	128
6	2011-04-22-17.50.00.000000	300	0	128
7	2011-04-22-17.55.00.000000	300	0	128
8	2011-04-22-18.00.00.000000	300	0	128
9	2011-04-22-18.05.00.000000	300	0	128
10	2011-04-22-18.10.00.000000	300	0	128
11	2011-04-22-18.15.00.000000	300	0	128
12	2011-04-22-18.20.00.000000	300	0	128
13	2011-04-22-18.25.00.000000	300	0	128
14	2011-04-22-18.30.00.000000	300	0	128

More...

F3=Exit      F12=Cancel      F19=Left      F20=Right      F24=More keys

MA G 03/032

I902 - Session successfully started

Figure 5-4 Sample query execution output

## 5.4.2 Checking QAPMSHRMP table using IBM System Director Navigator for i

This section describes the method to obtain Active Memory Sharing data in graphic format using the IBM System Director Navigator for i. For additional information regarding setup and requirements see *IBM Systems Director Navigator for i*, SG24-7789.

Use the following steps to display performance data using the IBM System Director Navigator for i:

1. Start your browser and go to <http://<systemname>:2001>.
2. Select **Performance** → **Investigate Data** → **Collection Services** → **Collection Services Database Files** → **select QAPMSHRMP**. Choose Collection Library and Collection Name and click **Display**, as shown in Figure 5-5 on page 76.

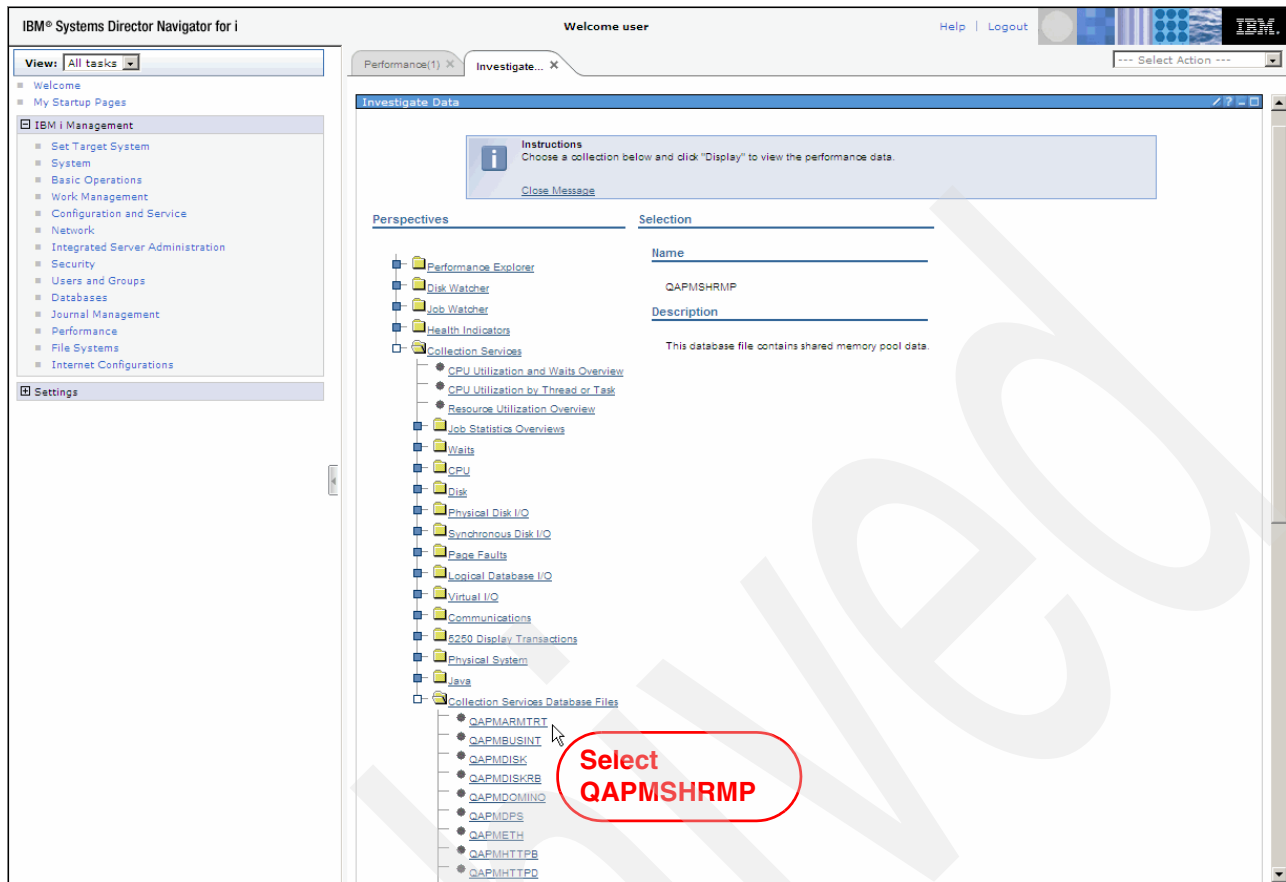


Figure 5-5 IBM Systems Director Navigator for i - selecting performance collection data

- As shown in Figure 5-6, you can select specific collection intervals or select all and then click **Done**.

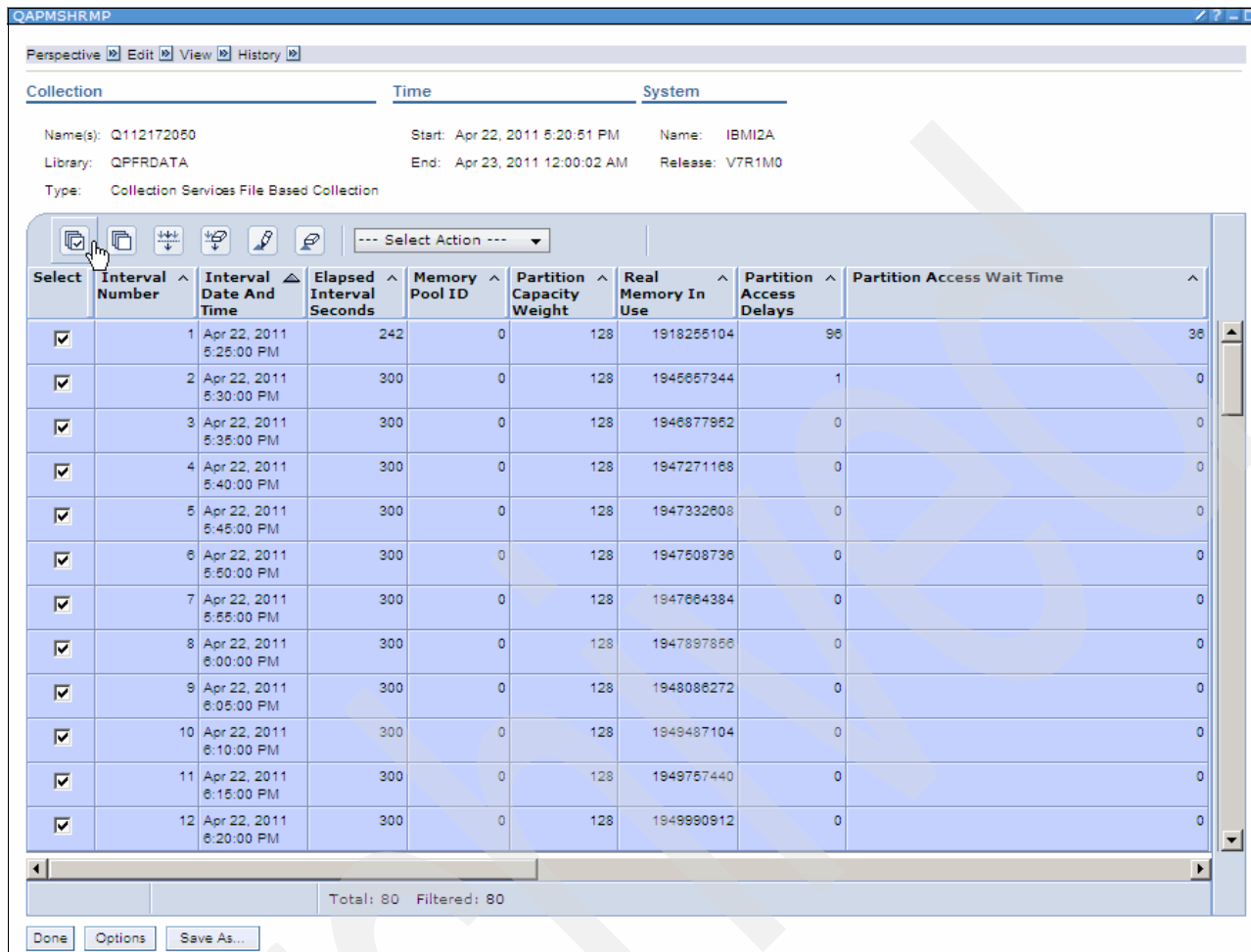


Figure 5-6 Collection intervals

- After you have selected intervals, on the same panel click **Select Action** and choose **Show as a chart**, as shown Figure 5-7.

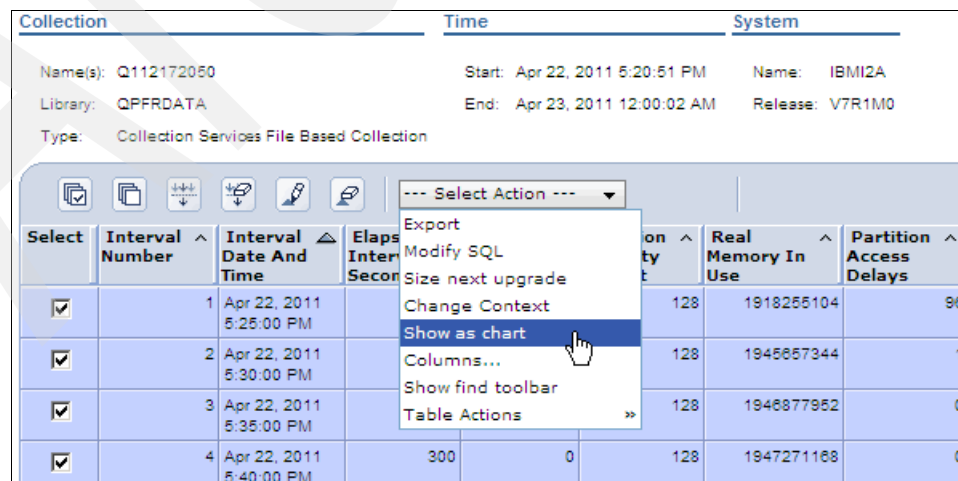


Figure 5-7 Select Show as chart to produce the graph

- As shown on Figure 5-8, you can select to specify the category that you want to present on the graph. Select **Real Memory in Use** to display memory usage, then click **Add** and **OK**. The result is a graph that shows real memory usage in the specified intervals, as shown in Figure 5-9.

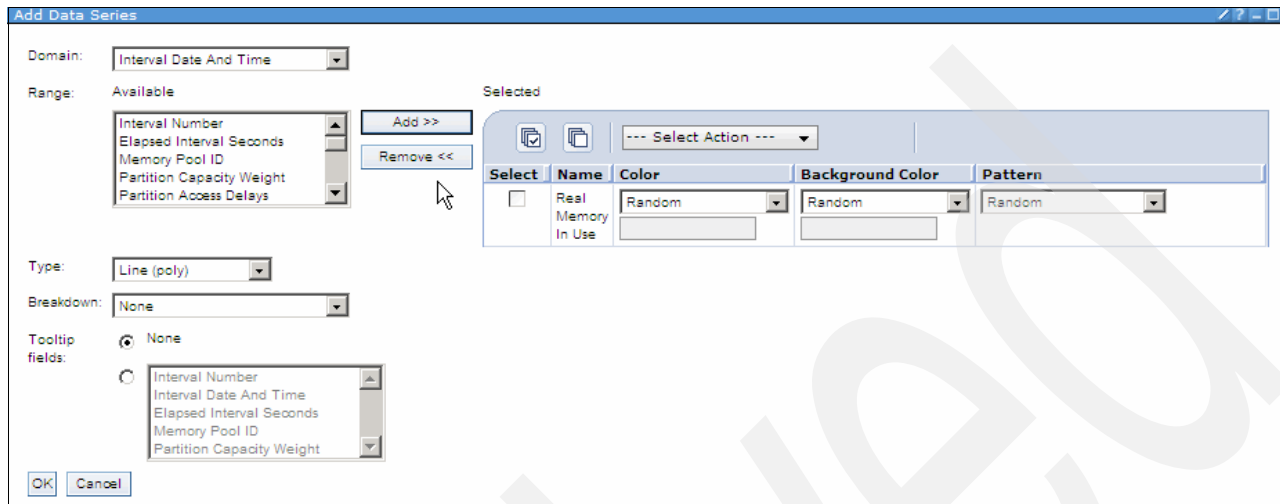


Figure 5-8 Select the category

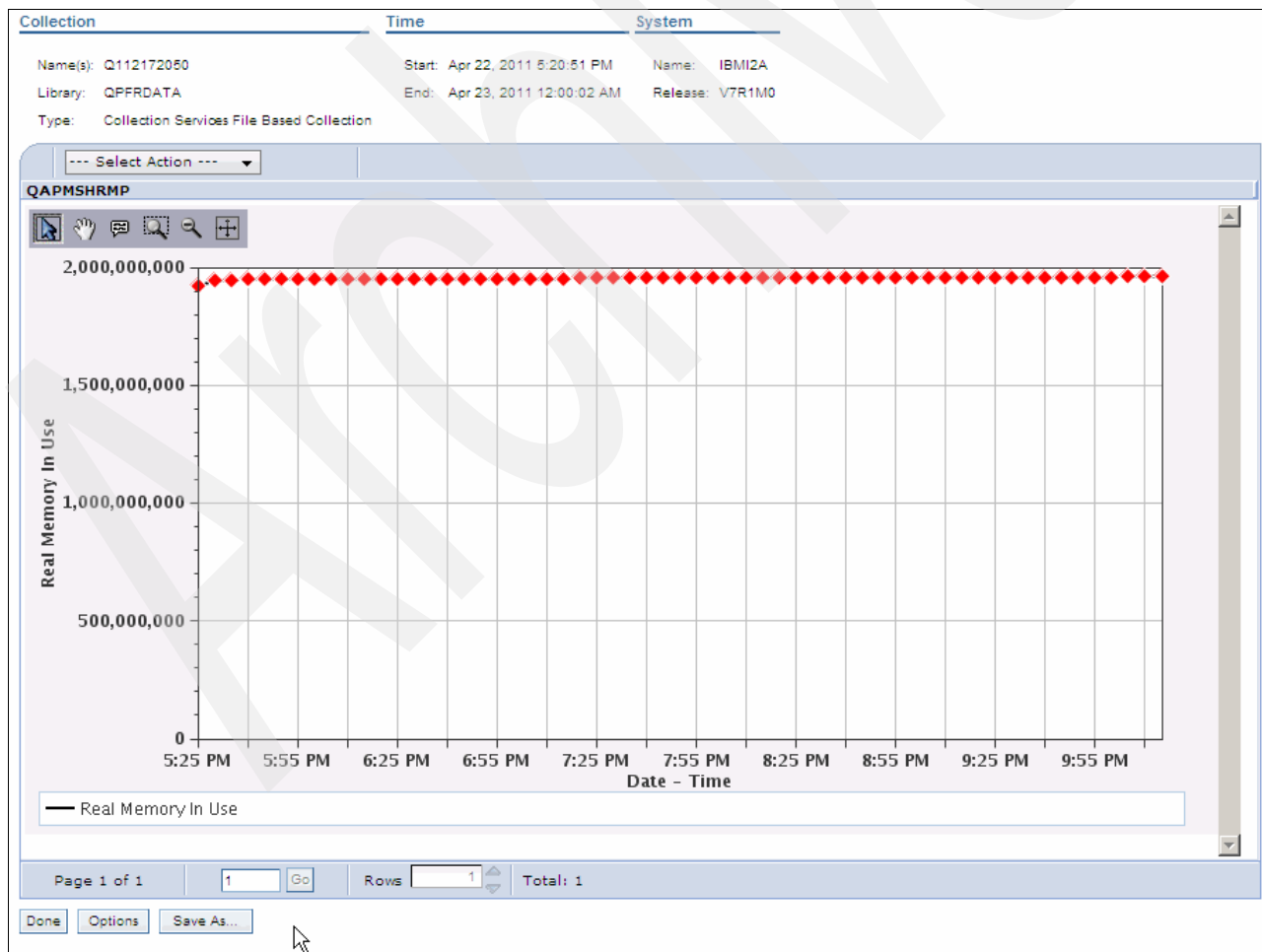


Figure 5-9 Real memory in use

You can navigate through values as you usually do in the interactive query environment. Another available option is to execute this query using IBM System i® Navigator or any query tool that uses a database protocol that is allowed to connect to the IBM i database.

## 5.5 Monitoring Linux

Performance information about Linux is available on /proc and /sys filesystems; however, the most effective way to monitor AMS performance is by using the **amsstat** tool available from the powerpc-utils project at sourceforge (<http://sourceforge.net/projects/powerpc-utils>). This tool consolidates all AMS performance data, as shown in Example 5-19.

*Example 5-19 Using amsstat to monitor AMS performance*

---

```

LINUX2A:~ # amsstat
Tue Apr 26 16:32:25 EDT 2011
System Memory Statistics:
    MemTotal:                2901568 kB
    MemFree:                  2676096 kB
    Buffers:                   22464 kB
    Cached:                    61952 kB
    Inactive:                   91392 kB
    Inactive(anon):            33920 kB
    Inactive(file):            57472 kB
    SwapTotal:                 4194176 kB
    SwapFree:                   4145728 kB
    DesMem:                     8192 MB
Entitlement Information:
    entitled_memory:           368050176
    mapped_entitled_memory:     4739072
    entitled_memory_group_number: 32774
    entitled_memory_pool_number: 0
    entitled_memory_weight:     0
    entitled_memory_pool_size:  21474836480
    entitled_memory_loan_request: 997240832
    backing_memory:             2294312960
    cmo_enabled:                 1
    cmo_faults:                  370096
    cmo_fault_time_usec:        489855309
    cmo_primary_psp:             1
    cmo_secondary_psp:           2
CMM Statistics:
    disable:                    0
    debug:                      0
    min_mem_mb:                  256
    oom_kb:                      1024
    delay:                       1
    loaned_kb:                   5174272
    loaned_target_kb:            5174272
    oom_freed_kb:                973824
VIO Bus Statistics:
    cmo_entitled:                368050176
    cmo_reserve_size:            24832000
    cmo_excess_size:             343218176
    cmo_excess_free:             343218176

```

```

        cmo_spare:                1562624
        cmo_min:                  7813120
        cmo_desired:              24832000
        cmo_curr:                 5361664
        cmo_high:                 20631552
VIO Device Statistics:
  1-lan@30000002:
    cmo_desired:                 4243456
    cmo_entitled:                4243456
    cmo_allocated:              4239360
    cmo_allocs_failed:          0
  vfc-client@30000003:
    cmo_desired:                 8450048
    cmo_entitled:                8450048
    cmo_allocated:              393216
    cmo_allocs_failed:          0
  vfc-client@30000004:
    cmo_desired:                 8450048
    cmo_entitled:                8450048
    cmo_allocated:              589824
    cmo_allocs_failed:          0
  v-scsi@30000005:
    cmo_desired:                 2125824
    cmo_entitled:                2125824
    cmo_allocated:              139264
    cmo_allocs_failed:          0

```

Table 5-3 describes the most relevant fields from the **amsstat** output. Consult the man pages for a complete list.

*Table 5-3 Definitions of amsstat command output fields*

Field name	Description
<b>System memory statistics</b>	
Memtotal	Maximum memory available to the system.
Memfree	Unused memory.
Inactive	Free memory from buffer or cache and available to be used because it was not recently used and can be reclaimed for other purposes.
Swaptotal	Available swap memory.
Swapfree	Free swap memory.
Desmem	Desired memory from LPAR profile.
<b>Entitlement information</b>	
Entitled_memory	Memory given to the operating system and available to be mapped for IO operations.
Mapped_entitled_memory	Memory currently mapped for IO operations.
Entitled_memory_weight	The weighting used by firmware to help prioritize partitions for memory loaning.

Field name	Description
Entitled_memory_pool_size	Total size of memory of the memory pool that the LPAR belongs to.
Entitled_memory_loan_request	Amount of memory that firmware would like to give via the CMM. A positive values means the amount of memory expected from partition to loan. A negative value means the operating system can take back that number for its own use.
Backing_memory	Physical memory currently reserved for access by the partition.
Cmo_enabled	If set to 1, indicates partition is running in shared mode. A zero value indicates the partition is running in dedicated mode.
Cmo_faults	The number of page faults since the operating system was booted. Increases in this value indicate memory contention between partitions running in the shared pool.
Cmo_fault_time_usec	The amount of time since the boot time, in microseconds, that the operating system has been suspended by platform firmware to process cmo_faults.
<b>CMM statistics</b>	
Disable	Indicates whether CMM mode is disabled. If it is set to 1, it is disabled, a value of 0 means it is enabled. If CMM is disabled no loaning will occur.
Oom_kb	The number of kilobytes of memory taken back from firmware by the CMM module for the operating system when an out of memory signal from the kernel is caught by CMM.
Delay	The number of seconds that CMM waits between requests to firmware for the number of pages that firmware would like the operating system to loan.
Loaned_kb	The amount of memory, in kilobytes, that the operating system has given back to the platform firmware to be used by other partitions. This value fluctuates to meet the demands of all of the partitions in the shared memory pool of an AMS environment.
Loaned_target_kb	The amount of memory, in kilobytes, that the firmware would like the operating system to loan for use by other partitions. This value can be greater than loaned_kb if firmware would like additional pages to be loaned or it can be less than loaned_kb if firmware is providing additional pages to the operating system.
Oom_freed_kb	The amount of memory, in kilobytes, that is no longer being loaned by CMM as a result of out-of-memory kernel signals.
<b>Vio bus statistics</b>	
Cmo_desired	The amount of memory, in kilobytes, that the device has requested from the bus to provide optimal performance.
Cmo_entitled	The amount of memory, in kilobytes, that the device is guaranteed that it can map for IO operations.
Cmo_allocated	The amount of memory, in kilobytes, that the device has currently mapped for IO operations.

Field name	Description
Cmo_allocs_failed	When the amount of memory allocated (cmo_allocated) has exhausted both the entitled memory (cmo_entitled) and the bus excess pool, memory mapping failures will occur. This field is a cumulative counter. Large changes in this value would indicate resource contention that might require system tuning.

**Note:** On Linux systems running in shared memory mode, commands like **top**, **vmstat** and **free** can show total memory values changing over time. This is an expected behavior caused by hypervisor assigning memory to the partition based on the current system workload.





## Tuning

This chapter describes how to tune Active Memory Sharing. Proper configuration can be obtained by monitoring and then optimizing two main areas:

- ▶ The shared memory pool
- ▶ Each single shared memory partition

## 6.1 Shared memory pool tuning

The performance of a shared memory configuration depends on the number of page faults managed by the hypervisor plus the time required to perform paging activity on the paging devices.

Memory sharing is designed to reduce the memory size required to run system workload and to automatically reallocate memory where it is most needed. Memory reduction is accomplished by allowing over-commitment of real memory. Page stealing and page faults are required to keep the shared memory partitions running in this environment, and they cannot be completely avoided. A balance between single logical partition performance and the advantage of physical memory optimization becomes the goal of tuning.

The number of acceptable page faults depends on the difference between the forecasted and effective over-commitment of memory, and this value varies from case to case. Use the tools described in Chapter 5, “Monitoring” on page 59, to track changes. Then modify the configuration accordingly while tuning a system for production workloads.

### 6.1.1 Shared memory pool configuration

A shared memory pool has one global configuration value: the size of the pool. The size can be dynamically changed based on pool usage and the number of page faults.

If the system has unallocated memory, it can be added to the memory pool. By increasing the size of the pool, you affect the over-commitment of the pool by reducing the need to page out the memory to the Virtual I/O Server-managed paging devices. The hypervisor assigns the additional memory pages to the logical partitions with higher memory demand. If page faults and paging activity are present during pool enlargement, they might not decrease immediately because some memory pages still have to be read from the disk.

Pool size reduction can be done if memory is required for dedicated memory partitions. The removal of memory from the pool must be carefully planned. The hypervisor has to free physical memory and might need to page out memory to disks. In this case, shared memory partition memory access time might be affected. Memory is removed from the shared memory pool in logical memory blocks, and the system free memory will increase by small increments. It might take a considerable amount of time for the hypervisor to reduce the pool by the full amount of memory requested.

**Note:** It is recommended that you perform pool size reduction when the load on the shared memory partitions is low.

### 6.1.2 Virtual I/O Server configuration

The Virtual I/O Server is involved in shared memory pool tuning because it performs the disk paging activities on behalf of the hypervisor.

Disk access time contributes to memory access time only when the hypervisor requires paging activity. This can be monitored by the page-in delay statistics on the operating system on the shared memory partition, or by the Management Console or IVM performance data.

#### Paging device selection

The memory pool's paging devices can be backed either by a disk device or a logical volume provided by the logical volume manager (LVM) of the Virtual I/O Server.

The LVM can split a single device into logical volumes, allowing the device to support multiple paging devices. It is *not* recommended, though possible, to use the physical storage that is occupied by the Virtual I/O Server operating system. However, logical volumes that belong to other storage pools can be used.

The Virtual I/O Server's physical volumes are preferred for paging devices because their usage provides a slightly shorter instruction path length to the physical storage and also simplifies the performance monitoring of the device because it is a unique paging device. A logical volume should be chosen when you cannot use the storage system to split the disk space in LUNs, or if disk resources are scarce.

To improve paging device performance, also consider keeping the client virtual disk devices provided by the Virtual I/O Server separate from the hypervisor paging devices. If both workloads share the same devices, they might compete for disk access. A possible solution is to have two Virtual I/O Servers, one handling paging devices and the other virtual disks, but this setup is not a requirement.

Paging device assignment to shared memory partitions is made at logical partition startup by the Management Console or IVM, based on the size of the logical partition's maximum logical memory and the size of each defined paging device. The smallest possible paging device is chosen. If available, the same device is used at each logical partition activation; otherwise, a new one is selected.

Because the selection of a paging device is automatically assigned by the Management Console or IVM, it is recommended that you create all paging devices on storage with similar performance characteristics. Although it is possible to have a mixture of device types (for example, logical volumes on internal SAS disks and LUNs on an IBM DS8300 subsystem), you have no control over which device is assigned to which logical partition. Using the IVM CLI, you can manually assign paging devices to partitions.

## Storage configuration

Disk paging activity has the following characteristics:

- ▶ Very random I/O operations
- ▶ Typically performed in blocks of 4 KB in size

Paging activity can be improved by using performance-oriented disk storage devices that are configured to match as closely as possible the I/O patterns requested by the Virtual I/O Server. Disk availability is also important to avoid disk failures that might lead to unavailability of memory pages and then shared memory partition failure.

Consider the following suggestions for disk paging setup:

- ▶ Spread the I/O load across as many physical disks as possible.
- ▶ Use disk caches to improve performance. Due to the random access nature of paging, write caches provide benefits, whereas read caches might not have an overall benefit.
- ▶ Use a striped configuration, if possible with a 4 KB stripe size, which is ideal for a paging environment. Because the Virtual I/O Server cannot provide striped disk access, striping must be provided by a storage subsystem.
- ▶ Disk redundancy is recommended. When using a storage subsystem, a configuration using mirroring or RAID5 is appropriate. The Virtual I/O Server cannot provide redundancy.
- ▶ Perform the usual I/O tuning on the Virtual I/O Server to improve disk access time, such as queue depth values and specific parameters provided by the storage provider.

## CPU configuration

Paging activity by the Virtual I/O Server is performed using CPU resources. Although the amount of CPU consumption might not be a concern in many environments, it should be monitored to avoid performance effects, especially if the Virtual I/O Server is involved in providing other services such as shared Ethernet adapters. The usual monitoring and tuning apply, as with any new environment.

As a general guideline, configure the Virtual I/O Server as an uncapped shared processor partition with enough virtual processors to manage peak loads, and a high processor weight.

## 6.2 Shared memory partition tuning

In all shared resource environments, it is important to evaluate either global resource consumption or the behavior of each resource user. When multiple logical partitions compete for memory access, it is important to compare performance values from all logical partitions and be aware that changes to one logical partition's configuration might affect others.

### 6.2.1 Memory sharing policy and memory load factor

AMS will allow memory pages to flow between partitions based on hypervisor's perception of the relative memory loads.

Periodically, the hypervisor computes a memory load factor for each AMS partition. This is based on a combination of the following metrics:

- ▶ Accumulated running average of hypervisor page faults incurred by the partition
- ▶ Accumulated running average of page table faults incurred by the partition
- ▶ Accumulated running average of OS memory load reported by the operating system of the partition
- ▶ Memory efficiency of the partition (proportion of used memory pages to unused pages that have been assigned to the partition)
- ▶ User-assigned memory weight

Each of these contributing metrics is normalized in a fashion that shows their relationship to the other partitions. The calculated load factors of each partition are then compared and analyzed so the hypervisor can make decisions about how to share the memory.

Apart from the memory weight factor, there is no possible tuning here. However, keeping in mind how AMS works will help you better fine-tune your infrastructure.

The flow of memory between partitions is not instantaneous. The hypervisor must see sustained demand from a partition before giving it significantly more memory at the expense of other partitions.

By default, the hypervisor will not share the full amount of physical memory from the shared pool between the shared-memory partitions if this is not needed. Each partition will only receive the physical memory it needs according to the memory allocation it does. Therefore, as long as the sum of the working set of each partition does not exceed the shared memory pool size, you might see free memory in the pool.

## 6.2.2 Logical partition

The parameters that define a shared memory partition are:

- ▶ The logical memory size
- ▶ The memory weight
- ▶ The I/O entitled memory

All three parameters can be dynamically modified, as explained here.

### Logical memory size

The logical memory size is the quantity of memory that the operating system can address. Depending on how much logical memory is used and the memory demand on the entire memory pool, the logical memory can use either physical memory or a paging device.

If the shared memory partition requires more memory than the provided logical memory, the operating system is capable of providing virtual memory by using its own paging space. If an operating system performs local paging, it does not affect the shared memory pool performance.

Logical memory should be sized based on the maximum quantity of memory that the logical partition is expected to use during peak hours, or the maximum physical memory that the administrator wants to let the logical partition allocate. The usage of physical memory should then be monitored to check whether over-commitment occurs, and can be either increased or reduced.

Logical memory can be resized. When it is reduced, the operating system must first free the logical memory by using its local paging space to keep providing the same amount of virtual memory in use by the applications.

Logical memory reduction of selected shared memory partitions can be performed to reduce the load on the memory pool. The paging activity is then moved from the pool to each operating system.

If logical memory is increased, the logical partition's addressable memory is increased, but the effective quantity of physical memory assigned depends on the global load on the pool and on the logical partition's access to logical memory. Note that increasing the logical memory does *not* imply that the logical partition will have more physical memory pages assigned to it.

### Memory weight

When there is physical memory over-commitment, the hypervisor has to decide how much memory to assign to each logical partition and which pages have to be copied on the paging devices. The choice is made by evaluating global memory usage, global memory load, and the memory weight assigned by all active shared memory partitions.

If the page fault rate for one logical partition becomes too high, it is possible to increase the memory weight assigned to the logical partition to improve the possibility of its receiving physical memory. The weight change is immediately applied, but you must monitor the effects over a short interval because hypervisor paging might be required to rebalance memory assignment.

**Note:** Memory weight is just one of the parameters used by the hypervisor to decide how many physical pages are assigned to the shared memory partitions.

## I/O entitled memory

The I/O entitled memory value represents the memory that is allowed to be permanently kept in the memory pool to handle I/O activity.

When a shared memory partition is started, the I/O entitled memory is automatically configured based on the number and the type of adapters defined for the logical partition. This value has been defined to satisfy I/O requests for most configurations. If the operating system shows that some I/O has been delayed, it is possible to increase the I/O entitlement.

I/O entitlements can be added by performing a dynamic LPAR reconfiguration on the logical partition's memory configuration.

**Note:** After you begin to manually tune entitled memory, the Management Console and IVM will no longer automatically manage it for the partition. For example, if you add or remove a virtual adapter, the entitled memory assigned to the partition is not adjusted by the Management Console or through the IVM.

## CPU configuration

The effect of Active Shared Memory on computing time is minimal, and is primarily tied to additional address translation. Some additional CPU cycles are required to manage the paging activity managed by the Virtual I/O Server, but they are similar to the cycles that are required in case the over-commitment of memory has to be managed on a dedicated memory partition by the operating system that is issuing paging activity on its own paging device.

When a logical partition accesses a logical memory page that is not in the physical memory pool, the virtual processor that is performing the access cannot proceed and it is no longer dispatched on a physical processor until the logical memory page is available. The temporary unavailability of that virtual processor shifts the load of the logical partition on the remaining virtual processors.

The number of configured virtual processors on a shared memory partition should be calculated so that when a high page fault rate occurs, the number of running virtual processors is able to sustain the workload. For example, if your sizing requires 3.0 processing units and 6 virtual processors to handle all the load peaks, configure an additional 3 virtual processors.

**Note:** AIX memory tuning described in “Loaning” on page 88 can be used to shift paging activity from the hypervisor to AIX. Reduced hypervisor paging lowers the need for additional virtual processors.

## 6.2.3 AIX operating system

There are specific tuning capabilities for AIX that can be used for Active Memory Sharing. They are related to loaning and I/O memory.

### Loaning

AIX interacts with the hypervisor for shared memory handling by classifying the importance of logical memory pages and by receiving the request to loan them from the hypervisor. Loaned logical memory pages are kept free by AIX for a longer time and are not shown in the free statistics of commands such as `vmstat` or `lparstat`. AIX can reclaim loaned memory if needed.

When the hypervisor needs to reduce the number of physical memory pages assigned to a logical partition, it first selects loaned memory pages, with no affect on logical partition's memory access time. If additional memory pages are required, the hypervisor first selects logical free memory and then logical used memory, but both might cause hypervisor page faults if referenced by AIX.

Used logical memory pages can be used to cache file data or to store working storage (process code and data). AIX tries to maximize its usage of logical memory by caching as much file data as possible. In a shared memory partition, excessive file caching might cause unwanted memory pool consumption.

When AIX starts to loan logical memory pages, it first selects pages used to cache file data.

It is possible to tune the loaning process by modifying the `ams_loan_policy` attribute using the `vmo` command, as follows.

```
vmo -a ams_loan_policy=0    Disable page loaning.
vmo -a ams_loan_policy=1    Default value. Only select file cache pages for loaning.
vmo -a ams_loan_policy=2    Aggressive loaning: when file cache is depleted, continue
                             loaning by paging out working storage pages.
```

When page loaning is disabled, AIX stops adding pages in the loaning state, even if requested by the hypervisor. When the hypervisor needs to reduce the logical partition's memory footprint and free pages have been already selected, either file cache pages and working storage can be moved on the hypervisor's paging space.

With the default loaning configuration, AIX performs loaning and targets only pages used for file caching. The working storage pages are never selected for loaning. When the hypervisor needs to reduce the number of physical memory pages assigned to the logical partition, it first selects loaned pages and then free and used memory pages. The effect of loaning is to reduce the number of hypervisor page faults because AIX reduces the number of active logical pages and classifies them as loaned.

**Note:** The default page loaning configuration is suitable for most production workloads, but some environments can take advantage of a different loaning setup. For example, some applications perform direct I/O and cache data themselves and therefore AIX file cache is minimal and default loaning has minimal effect.

Loaning changes should be monitored, to check the results.

The aggressive loaning policy allows AIX to loan a higher number of logical memory pages by using pages owned by applications when all file cache pages have been loaned. When AIX selects a working storage page, it is first copied to the local paging space. This setup reduces the effort of the hypervisor by moving paging activity to AIX.

**Note:** Aggressive loaning can cause additional AIX paging activity. It is recommended that you increase the size of the operating system's paging space and place it on fast disk devices. Start with a paging space size equal to the size of logical memory assigned to the logical partition.

Figure 6-1 on page 90 illustrates the effect of the three page loaning configurations. The upper half of the figure shows the status of the logical memory as provided by AIX. At a specific point in time, all logical memory has physical memory assigned to it, except for

loaned pages, as shown by the vertical bar near logical memory. AIX is using most of the active logical pages to cache file data.

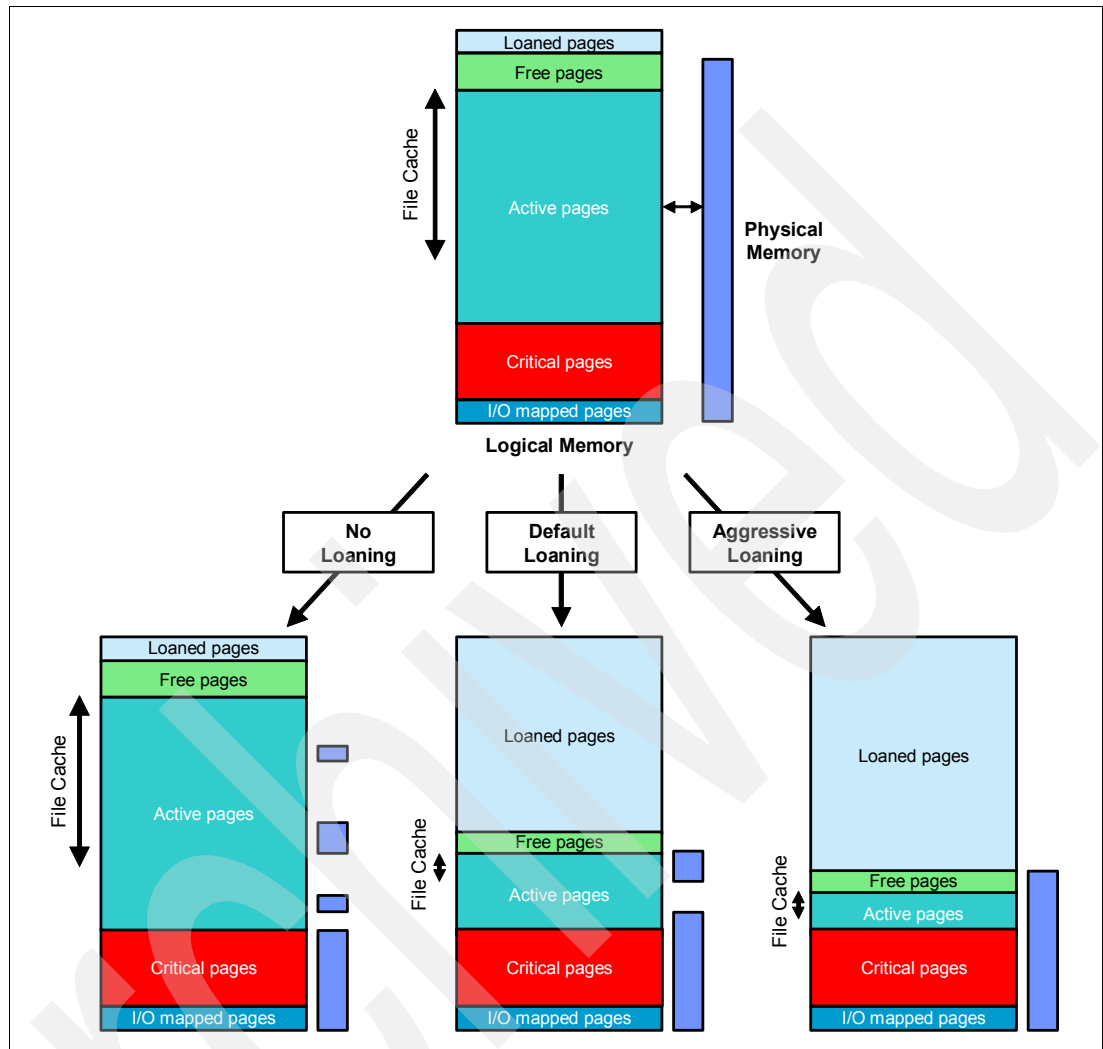


Figure 6-1 AIX loaning tuning example

When the hypervisor needs to reduce the physical memory assigned to the logical partition, it starts asking for loaning and then begins stealing logical memory pages. The bottom half of the figure shows three possible scenarios, depending on the loaning configuration of AIX.

### **Loaning disabled**

When loaning is disabled, AIX performs no action and the hypervisor steals first free and then active memory pages. Both file cache and working set pages are sent to the hypervisor's paging space because the hypervisor cannot differentiate for what purpose the pages are used by the operating system.

### **Default loaning**

With the default loaning configuration, AIX first reduces the number of logical pages assigned to the file cache and loans them. The hypervisor steals first the loaned pages, then free pages, and finally copies a few working storage pages into its paging space.



### Aggressive loaning

If page loaning is set to aggressive, AIX either reduces the file cache or frees additional working storage pages by copying them into the AIX paging space. The number of loaned pages is greater than is the case with default loaning. The hypervisor uses the loaned pages and might not need to perform any activity on its paging space.

### I/O memory pages

I/O memory pages are used by AIX to perform I/O operations. They must be kept in the shared memory pool to avoid delays in I/O.

I/O entitled memory is defined when the logical partition is started. It represents the maximum number of I/O memory pages that AIX can use. AIX normally uses only a subset of such entitlement, and the default values are suited for most configurations.

You can monitor actual memory entitlement usage, using the **lparstat** command as shown in Example 6-1, and then decide if the I/O entitlement of the logical partition should be dynamically increased.

Example 6-1 I/O memory entitlement monitoring

```
# lparstat -m 2
```

System configuration: lcpu=8 mem=10240MB mpsz=20.00GB iome=334.00MB iomp=10 ent=1.00

physb	hpi	hpit	pmem	iomin	iomu	iomf	iohwm	iomaf	%entc	vcs
1.64	0	0	2.88	287.7	12.4	34.3	14.6	0	2.6	752
1.80	0	0	2.88	287.7	12.4	34.3	14.6	0	3.0	484

The **iomaf** value displays the number of times that the operating system has attempted to get a page for I/O and failed. If the value is not zero (0), the I/O entitlement should be increased. The **iomu** and **iomf** values represent, respectively, the I/O entitlement used and the free I/O entitlement. If the **iomu** value is near the logical partition's I/O entitlement, or if the **iomf** value is constantly low, then the I/O entitlement should be increased.

I/O entitlements can be added by performing a dynamic LPAR reconfiguration on the logical partition's memory configuration.

## 6.2.4 IBM i operating system

There are several special considerations when tuning ASM on the IBM i operating system.

### Loaning

On IBM i it is not possible to change the behavior of the loaning process. Everything is managed by hypervisor.

### I/O memory pages

You can monitor actual memory entitlement usage by checking the IBM i Collection Services QAPMSHRMP table. Fields SMENTIOC, SMMINIOC, SMOPTIOC, SMIOCUSE, SMIOCMAX, SMIOMDLY display I/O dependencies for the actual configuration. If Partition I/O mapping delays is not zero, the I/O entitlement should be increased.

I/O entitlements can be added by performing a dynamic LPAR reconfiguration on the logical partition's memory configuration. This parameter is set to auto adjustment by default.

## Migrating to AMS

The previous chapters presented the technical background about the AMS architecture, and how to plan for, configure, monitor, and tune it. You are now ready to apply this knowledge to implement a migration from dedicated mode to AMS.

This chapter provides an overview about the migration process. The following topics are covered:

- ▶ Measuring memory utilization in dedicated mode
- ▶ Migrating from dedicated to shared memory mode
- ▶ Monitoring the environment after the migration
- ▶ Migration Checklist

## 7.1 Measuring the memory utilization in dedicated mode

Migrating to shared memory mode can be seen as a consolidation project, in which there are idle resources that can be used more efficiently by sharing between partitions.

The first step in a migration is to select which partitions can take advantage of being part of a shared memory pool. This selection is based on workload characteristics that are discussed in 3.2.2, “Workload selection” on page 37. On production systems it is expected that some partitions will have to be in dedicated memory mode due to specific requirements, such as performance demands or the need for a consistent amount of memory without substantial variation in the workload, leaving little capacity for other partitions to take advantage of.

Table 7-1 shows a scenario in which there are six LPARs, on six different operating system, that will be analyzed as possible candidates to be migrated to a shared memory pool.

Table 7-1 Partitions currently running in dedicated mode

LPAR name	OS	Dedicated memory
AIX2A	AIX 7.1	4 GB
AIX2B	AIX 7.1	4 GB
IBMi2A	IBM i 7.1	4 GB
IBMi2B	IBMi 7.1	4 GB
LINUX2A	Suse SLES11	4 GB
LINUX2B	Suse SLES11	4 GB
Total		24 GB

Figure 7-1 shows the memory graph obtained from the partitions running in dedicated mode.

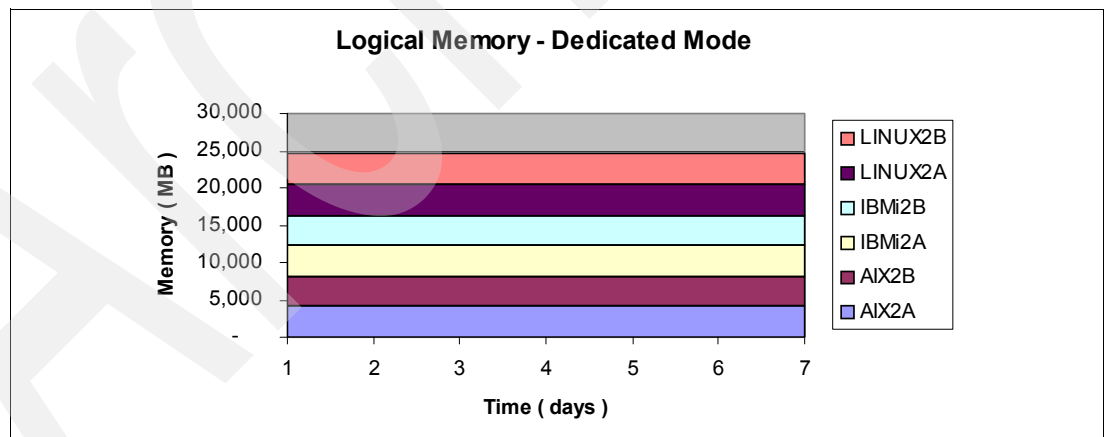


Figure 7-1 Memory assigned to partitions in dedicated mode

Figure 7-2 shows the memory usage behavior of each partition as provided by the partition's operating system.

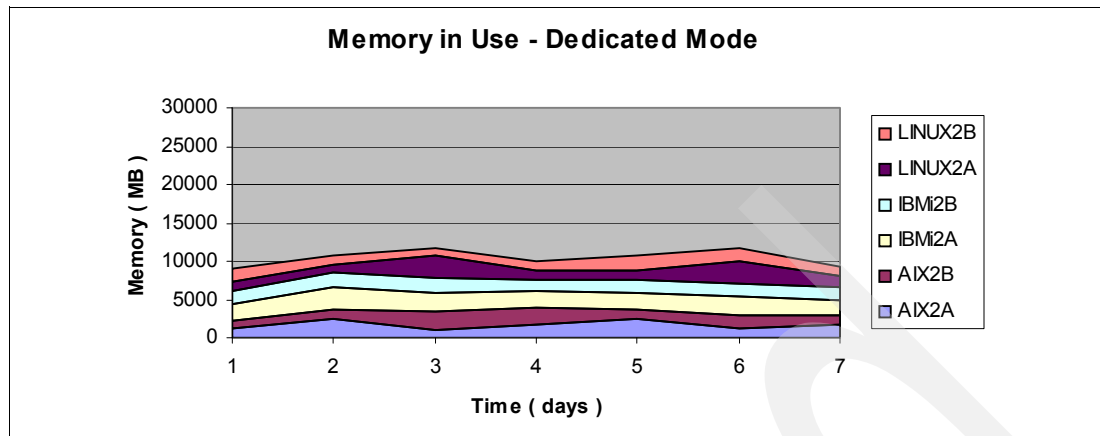


Figure 7-2 Memory used by partitions running in dedicated mode

Analyzing the memory utilization graph you can see that around 15.5 GB of memory is enough to run the current workload. Therefore, these partitions are good candidates to be migrated to a shared memory pool.

**Note:** While IBM i is running in dedicated mode we cannot measure efficiently how much memory is being used because IBM i just takes all the memory dedicated to its partition. This behavior changes when it is running in shared memory mode, as shown in Figure 7-5 on page 99.

## 7.2 Migrating from dedicated to shared memory mode

The process used to migrate partitions from dedicated to shared memory mode is detailed in this section.

### 7.2.1 Active Memory Sharing configuration

This section describes the steps needed to prepare the shared memory environment for migration.

#### Logical memory sizing

After selecting the partitions you plan to migrate to a shared memory environment (based on the discussion in the previous section) you now have to decide about the logical memory configuration.

The physical memory that will be allocated to the shared memory partitions will vary from 0 MB to the desired logical memory amount defined in the partitions' profiles, depending on their memory needs and the global workload. Therefore, we suggest that the desired logical memory amount should be at least equal to the desired dedicated memory the partition has. You could even consider setting the desired logical memory amount to a higher value if you noticed your partition sometimes uses all its memory in dedicated mode. This would allow the partition to benefit from more physical memory if the global workload on the system allows it.

For example, a dedicated memory partition with 4 GB of desired memory, which sometimes shows some paging activity (100 MB used on the OS paging device), could have a logical desired memory amount of 4.5 GB.

Once you have decided on the desired logical memory amount of all your partitions, you can think about the maximum logical memory amount. This value will allow you to dynamically change the logical memory amount of the partition using dynamic LPAR operations. This is useful if you have partitions that might have their memory needs increase in the future due to workload growth.

### Paging devices creation

The paging device size for a given shared memory partition has to be at least equal to the maximum logical memory of the partition. (There is an exception for IBM i partitions; see 4.2, “Creating the shared memory pool” on page 43.) A suggested size for the paging device is 110% of the maximum logical memory. This would ensure that you have a paging device big enough and allows you to benefit from the partition suspension capability if it is needed (see 2.5.2, “Partition suspend and resume” on page 29).

Once you have determined the size for each of your paging devices you should be ready to create them on your VIOS. Keep the following considerations in mind when doing so:

- ▶ Paging devices on logical volumes cannot be redundant.
- ▶ Redundant paging devices must be accessible by both VIOS.

For more details, see 4.2, “Creating the shared memory pool” on page 43.

### Shared memory pool creation

When all paging devices are ready, proceed to the shared memory pool creation by following the procedure in 4.2, “Creating the shared memory pool” on page 43.

Decide on a shared memory pool size. The goal when sizing the shared pool is to try to guarantee enough physical memory for the partitions to operate with minimal paging activity and optimized memory usage (see “Logical overcommit” on page 37). Typically, the sum of the dedicated memory of the selected partition should be higher than the shared memory pool size.

Because you have analyzed your partitions’ memory needs, you should have an idea of the combined amount of memory used by all your partitions during peak times (see Figure 7-2 on page 95). Based on this amount of used memory, you can easily size your shared memory pool. If you have no special requirements, we suggest using 125% of this amount for the shared memory pool size. In our example it would be  $15.5 \times 1.25$ , or approximately 19 GB.

Nevertheless, the shared memory pool size can be dynamically adjusted at any time. See 4.4.5, “Managing the shared memory pool size” on page 54.

When you reach the step for the paging device selection make sure you select all the paging devices your prepared and check their redundancy. Your redundant paging devices (if any) should be displayed in the **Common Device List** section.

Follow the guidance provided in the Management Console wizard to complete the creation of the shared memory pool.

## 7.2.2 Shared memory partition profile creation

At this point, the shared memory pool has been created, and the desired and maximum logical memory amount for each partition has been calculated. The selected partitions should not have any dedicated adapters, or should be able to switch all the dedicated adapters to virtual adapters (such as NPIV or Virtual Ethernet).

For environments using Management Console, we suggest creating a new partition profile for AMS. This new profile can either be created from a copy of the last activated profile, or by using the **Save current configuration** option on the Management Console as shown in Figure 7-3.

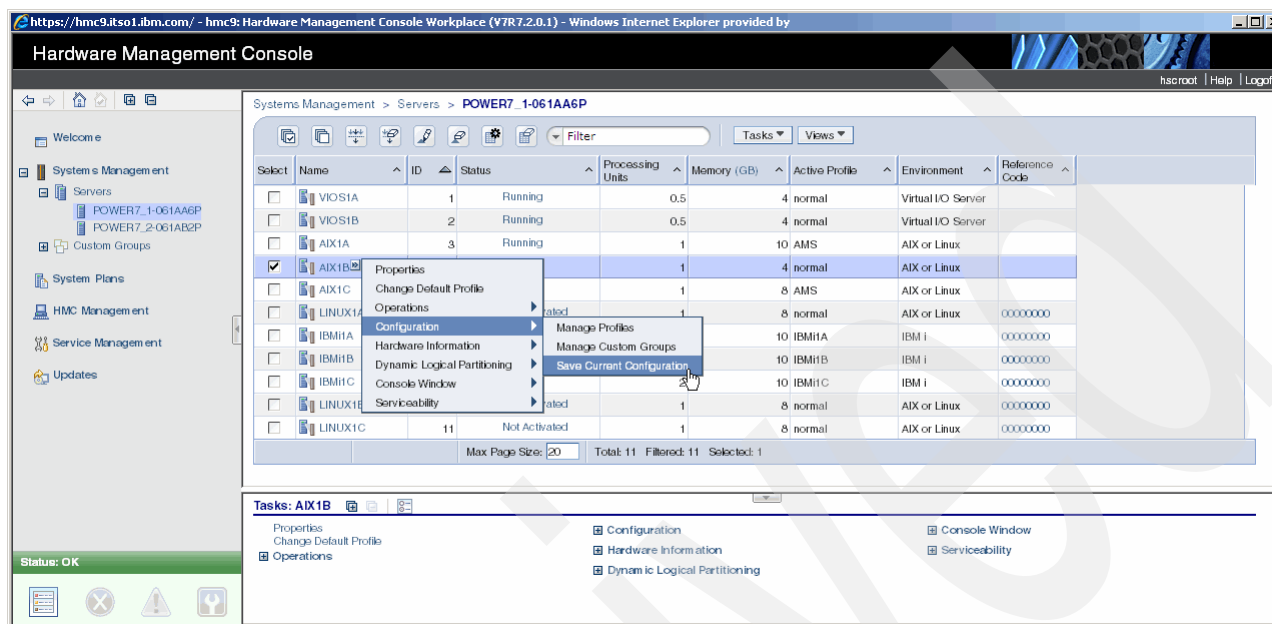


Figure 7-3 Create profile from current configuration

To edit this newly created profile, go to the Memory tab and select **Shared** as show in Figure 7-4 on page 98. If you have any dedicated adapter defined on the profile at this time, the Management Console will display a warning; remove the adapter from the profile by accepting the warning.

Make the appropriate entries in the fields displayed on this panel. Complete the fields for **Desired** and **Maximum memory** by entering the amounts you calculated previously. Set the **Minimum memory** value to the lowest acceptable value for the OS to boot. The **memory weight** value is relative to the other partitions using the shared memory pool; if you leave it at 0 (zero) for all partitions, they will all have the same priority. If you want to favor some partitions, you can set their memory weight higher than the others (see 4.3, “Creating a shared memory partition” on page 49). The **Primary** and **Secondary Paging VIOS** fields let you choose the VIOS to be used for paging activity. If you planned a redundant paging device for the partition you are currently migrating, you can set both VIOS; otherwise, choose the one which holds the paging device.

**Note:** If the paging VIOS is configured incorrectly, you will not be warned until you try to start the partition with that profile.

You can manually define the entitled memory for I/O operations in the **Custom I/O entitled memory** field. You should leave this parameter unset by default because it will be automatically managed. You can still modify it later, and dynamically if needed. See “I/O entitled memory” on page 88.

https://hmc9.itso1.ibm.com/ - hmc9: Manage Profiles - Windows Internet Explorer provided by 1...

**Logical Partition Profile Properties: AMSprofile @ AIX1B @ POWER7\_1-061AA6P - AIX1B**

General Processors **Memory** I/O Virtual Adapters Power Controlling Settings HCA Logical Host Ethernet Adapters (LHEA)

Detailed below are the current memory settings for this partition profile.

**Memory mode**

☐ Dedicated

☒ Shared

**Logical Memory**

Minimum memory : 2 GB 0 MB

Desired memory : 4 GB 0 MB

Maximum memory : 8 GB 0 MB

**Shared Memory Options**

Available pool memory 19545 MB

Memory weight (0-255) 0

Primary Paging VIOS: VIOS1A

Secondary Paging VIOS: VIOS1B

☐ Custom I/O entitled memory 0

**Active Memory Expansion**

☐ Active memory expansion factor (1.00 - 10.00) 0.0

OK Cancel Help

Figure 7-4 Enable shared memory mode

You can activate and size **Active Memory Expansion** on this panel. See 2.5.1, “Active Memory Expansion” on page 26 for details about this selection.

Unless you have virtual devices to define, shared memory partition profile creation is finished. You can define this new profile as the default profile.

### Partition shutdown - reactivate

You must activate your partitions with the newly defined profile for the partitions to use Active Memory Sharing.

You have to plan a maintenance window for this short outage to complete a full shutdown of the partition before activating it using the new AMS profile. If your profile configuration is good (especially with respect to the dual VIOS and the redundant paging device availability) your partition will start in shared memory mode.

You can check the paging device that has been allocated to your partition after it is activated. See 4.4.1, “Paging device assignment” on page 51.



## 7.3 Monitoring the environment after the migration

After migration, continue monitoring the environment to fine-tune the shared memory pool size and paging activity.

### 7.3.1 Monitoring memory usage after the migration

Figure 7-5 shows the physical memory allocated to logical partitions after migration to shared memory mode.

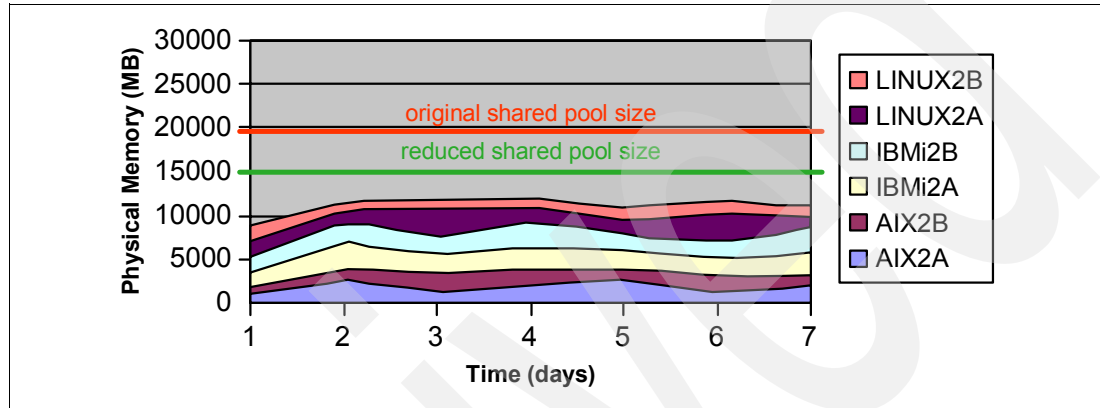


Figure 7-5 Physical memory allocated from the shared memory pool to shared memory partitions

As you can see, physical memory allocated to IBM i partitions varies because the AMS loaning mechanism encourages the operating system to free unneeded memory to the shared memory pool. Physical memory allocated to the partitions can be retrieved using the `ls1parut11` command, as discussed in Chapter 5, “Monitoring” on page 59.

Using this graph we can observe that the cumulative physical memory used by shared memory partitions is now around 12 GB due to IBM i working cooperatively with other partitions and loaning pages to the shared pool. We could consider decreasing the shared memory pool size to  $12 \text{ GB} \times 1.25 = 15 \text{ GB}$ , which represents an overall 37.5% memory savings. Of course, this is not a rule and the amount of memory saved is directly related to the workload, as discussed in 5.1, “Management Console” on page 60.

**Note:** For IBM i partitions, after migration a new table QAPMSHRMP is created by the Collection Services. See 5.4, “Monitoring IBM i” on page 73 for further details.

### 7.3.2 Monitoring paging activity

One critical factor for performance is the paging activity, which should be constantly monitored as discussed in Chapter 5, “Monitoring” on page 59.

If your monitoring reveals high and regular paging activity, it means you are in the physical overcommit case most of the time. If this paging activity has an impact on the global performance of your environment you should consider increasing the shared memory pool size.

If monitoring shows high paging activity only during rare peak periods for some partitions and you prefer not to increase the shared memory pool just for that, you can consider providing a better paging device for these partitions to shorten the peak duration.

## 7.4 Migration checklist

Table 7-2 is a checklist that can be used to organize projects that migrate partitions from dedicated to shared memory mode.

Table 7-2 Migration checklist

Check	Step	Comments
<input type="checkbox"/>	Identify which partitions should be part of migration scope.	
<input type="checkbox"/>	Check 1.1, "Requirements" on page 2 to verify system requirements prior to start. If any upgrade is needed, perform it prior to starting the migration to AMS.	
<input type="checkbox"/>	Check the current memory allocated to partitions running in dedicated mode.	Dedicated Memory: _____
<input type="checkbox"/>	Monitor the memory usage of each partition during a time period long enough to be representative of the total workload (suggestion: 30 days). If you do not have a capacity tool you can create a (.csv) to create a data set.	Suggested format of .csv file: host name, time stamp, memory in use
<input type="checkbox"/>	Plot a stacked graph with data set collected in the previous step for all the partitions.	
<input type="checkbox"/>	Size the shared pool and the VIOS paging devices. <sup>a</sup>	Shared Memory Pool Size: _____  VIOS Paging Device List: Quantity Size ____ x _____ (GB) - Redundant? ____ ____ x _____ (GB) - Redundant? ____ ____ x _____ (GB) - Redundant? ____ ____ x _____ (GB) - Redundant? ____
<input type="checkbox"/>	Create the paging devices.	
<input type="checkbox"/>	Create the shared memory pool. <sup>b</sup>	
<input type="checkbox"/>	Create a copy of the current profile to a new one and enable AMS.	
<input type="checkbox"/>	Make the AMS profile the default partition profile.	
<input type="checkbox"/>	Activate the partitions using the new AMS profile.	

Check	Step	Comments
<input type="checkbox"/>	Run OS commands to ensure the OS is running in AMS mode, as described in Chapter 5, “Monitoring” on page 59.	
<input type="checkbox"/>	Monitor the memory used by the partition by using <code>lslparutil</code> command, as shown in 5.1, “Management Console” on page 60.	
<input type="checkbox"/>	Monitor the VIOS activity to check substantial usage of Paging Devices.	
<input type="checkbox"/>	Based on the results of your monitoring, resize the shared memory pool to avoid physical paging, when possible.	

- a. Follow the guidelines presented on “Paging devices creation” on page 96.
- b. If the system has memory constraints and there is not enough memory to create a minimum shared memory pool, consider using Live Partition Mobility to move partitions to another system (no outage) or deactivate partitions (if outage is allowed).



# Glossary

**AMS** Active Memory Sharing.

**Collaborative Memory Manager** Operating System feature that gives hints on memory page usage to the hypervisor.

**HMC** Hardware Management Console. A tool for planning, deploying, and managing partitions on IBM Power Systems.

**I/O entitled memory** Maximum amount of physical memory guaranteed to be available for I/O mapping.

**IVM** Integrated Virtualization Manager. A tool for planning, deploying, and managing partitions on IBM Power and Blade Centers.

**Logical memory** Desired configured memory not necessarily backed up by physical memory.

**Logical overcommit** State when, even though you configure more logical than physical memory available, the current working set can be backed up by the physical shared memory pool.

**LPAR** Logical Partition.

**Memory loaning** Method used to respond to hypervisor loan requests with pages that are least expensive to donate. Those pages can be unloaned to be used immediately as the load increases in the OS.

**Oversubscription ratio** Value obtained by dividing the sum of all logical memory of the running partitions and the physical memory available in the pool. Helpful to determine the percentage of over commitment.

**Paging device** Physical or logical device used for paging VIOS to page out memory pages.

**Paging space** An area of non-volatile storage used to hold portions of a shared memory partition's logical memory that does not reside in the shared memory pool.

**Paging Virtual I/O Server** Virtual I/O Server partition that provides access to paging devices.

**Partition paging space size** Amount of paging space that the partition requires to back up its logical memory pages.

**Physical overcommit** State when total logical memory in use is larger than physical memory in the shared pool.

**Pinning** Locking a page of logical memory into physical memory.

**PTF** Program Temporary Fix. The packaging and distribution mechanism for fixes to IBM i licensed programs.

**SDMC** System Director Management Console. Is the successor to the Hardware Management Console (HMC) and the Integrated Virtualization Manager (IVM).

**Shared memory pool** Configured collection of physical memory units managed by Active Memory Sharing manager.

**Shared memory weight** Parameter used as one of the factors for distributing the physical memory between shared memory partitions.

**SLIC** System Licensed Internal Code. Provides the hardware and software abstraction that defines the Machine Infrastructure architecture to the IBM i operating system layer.

**VIOS** Virtual I/O Server.

**Virtualization Control Point** Administrative interface for Active Memory Sharing functions, such as HMC or IVM.



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

## IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *PowerVM Virtualization on IBM System p: Introduction and Configuration Fourth Edition*, SG24-7940
- ▶ *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590
- ▶ *Hardware Management Console V7 Handbook*, SG24-7491
- ▶ *Integrated Virtualization Manager on IBM System p5*, REDP-4061
- ▶ *IBM Systems Director Navigator for i*, SG24-7789

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, drafts, and additional materials, at the following website:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Other publications

These publications are also relevant as further information sources:

- ▶ *IBM PowerVM Active Memory Sharing Performance Paper*  
<ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/pow03017usen/POW03017USEN.PDF>
- ▶ *Active Memory Expansion Performance*, available at:  
[http://www-03.ibm.com/support/techdocs/atsmastr.nsf/5cb5ed706d254a8186256c71006d2e0a/dd66565f18701097862576c7000cef66/\\$FILE/POW03038.pdf](http://www-03.ibm.com/support/techdocs/atsmastr.nsf/5cb5ed706d254a8186256c71006d2e0a/dd66565f18701097862576c7000cef66/$FILE/POW03038.pdf)
- ▶ *IBM i Virtualization and DS4000 Read-me First*, available at:  
[http://www-03.ibm.com/systems/resources/systems\\_i\\_os\\_i\\_virtualization\\_and\\_ds4000\\_readme.pdf](http://www-03.ibm.com/systems/resources/systems_i_os_i_virtualization_and_ds4000_readme.pdf)
- ▶ *Using Active Memory Sharing on SLES11*  
<http://www.ibm.com/developerworks/wikis/display/LinuxP/Using+Active+Memory+Sharing+on+SLES11>
- ▶ *System Hardware information* (online product documentation)  
<http://publib.boulder.ibm.com/infocenter/systems/scope/hw/index.jsp?topic=/iphat/iphatmpvp.htm&searchQuery=%61%63%74%69%76%65%20%6d%65%6d%6f%72%79%20%73%68%61%72%69%6e%67&searchRank=%30&pageDepth=%32>

## Help from IBM

IBM Support and downloads

[ibm.com/support](https://ibm.com/support)

IBM Global Services

[ibm.com/services](https://ibm.com/services)







# IBM PowerVM Virtualization Active Memory Sharing



**Redpaper™**

**Advanced memory  
virtualization  
technology**

**Intelligent flow of  
memory from one  
partition to another**

**Increased utilization  
and flexibility of  
memory usage**

This IBM Redpaper publication introduces PowerVM Active Memory Sharing on IBM Power Systems based on POWER6 and later processor technology.

Active Memory Sharing is a virtualization technology that allows multiple partitions to share a pool of physical memory. This is designed to increase system memory utilization, thereby enabling you to realize a cost benefit by reducing the amount of physical memory required.

The paper provides an overview of Active Memory Sharing, and then demonstrates, in detail, how the technology works and in what scenarios it can be used. It also contains chapters that describe how to configure, manage, and migrate to Active Memory Sharing based on hands-on examples.

The paper is targeted to both architects and consultants who need to understand how the technology works to design solutions, and to technical specialists in charge of setting up and managing Active Memory Sharing environments.

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
**[ibm.com/redbooks](http://ibm.com/redbooks)**

REDP-4470-01