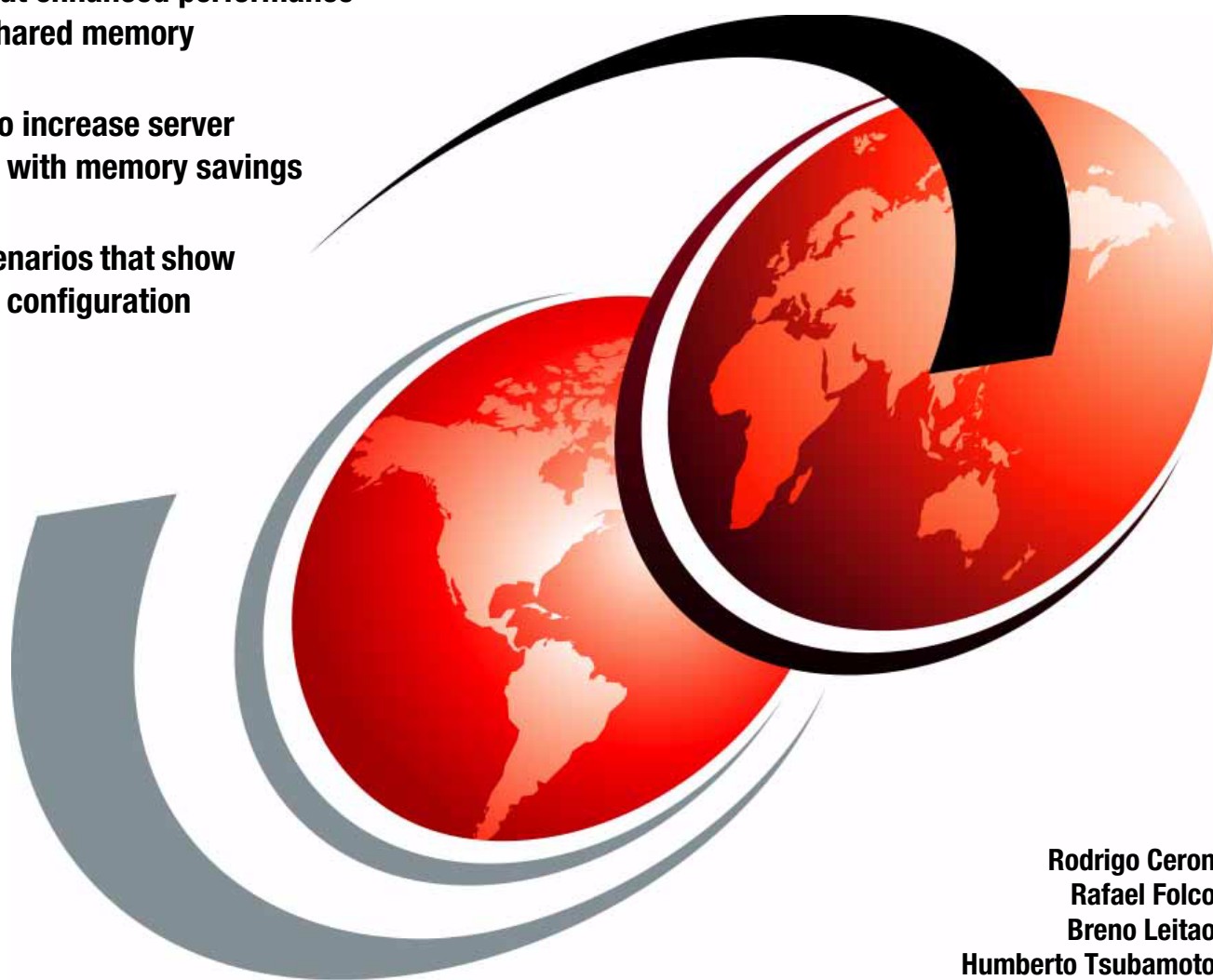


Power Systems Memory Deduplication

Learn about enhanced performance
through shared memory

See how to increase server
utilization with memory savings

Follow scenarios that show
setup and configuration



Rodrigo Ceron
Rafael Folco
Breno Leitao
Humberto Tsubamoto



International Technical Support Organization

Power Systems Memory Deduplication

February 2012

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

First Edition (February 2012)

This edition applies to AIX Version 7.1 TL1, IBM i Version 7.1, Red Hat Linux Enterprise 6.2, HMC Version 7 release 7.4.0 with PTF MH01275, SDMC 6.750.0, Virtual IO Server 2.2.1.1 running on IBM Power Systems with POWER7 processor-based technology and firmware level 740_042.

This document was created or updated on March 20, 2012.

© Copyright International Business Machines Corporation 2012. 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

Notices	v
Trademarks	vi
Preface	vii
The team who wrote this paper	vii
Now you can become a published author, too!	ix
Comments welcome	ix
Stay connected to IBM Redbooks	ix
Chapter 1. Overview	1
1.1 Hardware and software requirements	2
1.2 Active Memory Deduplication	3
Chapter 2. Concepts	5
2.1 Objectives of active memory management	6
2.2 Power systems memory management	6
2.2.1 Memory management with Active Memory Sharing	6
2.2.2 Active Memory Sharing context for Active Memory Deduplication	12
2.3 Active Memory Deduplication performance overview	14
Chapter 3. Planning for Active Memory Deduplication	15
3.1 Checking the requirements for Active Memory Deduplication	16
3.1.1 Managed system firmware	16
3.1.2 IBM PowerVM edition	24
3.1.3 Management console	29
3.1.4 Virtual I/O Server version	31
3.1.5 Operating system	31
3.2 Sizing your systems for Active Memory Deduplication	34
3.2.1 Memory sizing requirements	34
3.2.2 Processor sizing requirements	34
3.3 Enabling and disabling Active Memory Deduplication	35
3.3.1 Enabling Active Memory Deduplication using the HMC GUI	36
3.3.2 Disabling Active Memory Deduplication using the HMC GUI	37
3.3.3 Enabling or disabling Active Memory Deduplication using the HMC CLI	37
3.3.4 Enabling Active Memory Deduplication using the SDMC GUI	38
3.3.5 Disabling Active Memory Deduplication using the SDMC GUI	40
3.3.6 Enabling or disabling Active Memory Deduplication using the SDMC CLI	40
Chapter 4. Tunable parameters	41
4.1 Tuning the ratio of the deduplication table	42
4.2 Tuning the maximum size of the Active Memory Sharing pool	44
4.3 Spare Virtual I/O Server processor cycles	46
Chapter 5. Memory deduplication monitoring	49
5.1 Statistics	50
5.2 Monitoring tools	53
5.2.1 Monitoring in AIX	54
5.2.2 Monitoring in IBM i	54
5.2.3 Monitoring in Linux	55

5.2.4 Monitoring through the Systems Director Management Console	57
Chapter 6. Configuration scenarios	63
6.1 System configuration	64
6.2 Same workload with default Active Memory Deduplication settings	64
6.3 Different workloads with default Active Memory Deduplication settings	66
6.4 Multiple operating system types with memory overcommitment	67
6.5 Changes to the deduplication table ratio	68
6.6 Variations of the maximum Active Memory Sharing pool size	70
6.7 Virtual I/O Server processor sizing	73
Chapter 7. Best practices	75
7.1 Tuning best practices	76
7.1.1 Best practice for the deduplication table ratio	76
7.1.2 Best practice for the maximum Active Memory Sharing pool size	76
7.1.3 Best practice for Virtual I/O Server sizing	77
7.2 Tuning summary	77
Chapter 8. Troubleshooting	79
8.1 Common errors	80
8.1.1 Performance statistics are not displayed	80
8.1.2 Command not found message for amsstat on Linux	80
8.1.3 Few coalesced pages	81
8.1.4 ccol is always zero in lparstat	81
8.2 HSCLA errors	81
8.2.1 Error code HSCLA4F3 when changing the deduplication table	81
8.2.2 Error code HSCLA4F2 when disabling Active Memory Deduplication	81
8.2.3 Error code HSCLA4F1 when enabling Active Memory Deduplication	81
8.2.4 Error code HSCLA4F0 when enabling Active Memory Deduplication	82
Related publications	83
IBM Redbooks	83
Other publications	83
Help from IBM	83

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.

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](http://www.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:

Redbooks (logo) ®
Active Memory™
AIX®
developerWorks®
DS4000®

IBM®
Power Systems™
POWER6®
POWER7 Systems™
POWER7®

PowerHA®
PowerVM®
POWER®
Redbooks®
Redpaper™

The following terms are trademarks of other companies:

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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 IBM PowerVM® Active Memory™ Deduplication on IBM Power Systems™ based on IBM POWER7® processor technology.

Active Memory Deduplication is a virtualization technology through which memory pages with identical contents can be deduplicated in physical memory. This deduplication frees up physical memory positions so that more data can be held in memory at once.

Memory deduplication is intended to work in a shared memory environment. Deduplication works with Active Memory Sharing, a technology through which multiple partitions on a system can share a pool of physical memory. Sharing this pool sometimes creates an overcommitment of this physical memory. Active Memory Deduplication increases the performance of Active Memory Sharing because the savings can be used to lower memory overcommitment levels. Reduction of memory in use can also create room to increase the memory footprint of the logical partitions (LPARs).

This paper describes the advantages of Active Memory Deduplication and requirements for its use and implementation in your Power Systems. This paper also helps you set up Active Memory Deduplication on your Power System for use with IBM AIX®, IBM i, and Linux. It includes guidance for tuning parameters of Active Memory Deduplication and for troubleshooting issues.

This paper targets architects and consultants who need to understand how Active Memory Deduplication technology works to design performing solutions. The paper also targets technical specialists in charge of setting up and managing Active Memory Deduplication.

Before reading this paper, you must be familiar with the concepts of *PowerVM virtualization* and *Active Memory Sharing*. For more information, see the following publications:

- ▶ *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940
- ▶ *IBM PowerVM Virtualization Active Memory Sharing*, REDP-4470

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 (ITSO), Poughkeepsie Center.

Rodrigo Ceron is a Master Inventor and consultant in IBM Lab Services and Training, Brazil, Latin America. He has 12 years of experience in the UNIX/Linux arena and eight years working at IBM, where he has created seven patents in multiple areas. He graduated with honors in Computer Engineering from the University of Campinas (UNICAMP) and holds an IEEE CSDA credential. His areas of expertise include Power Systems high availability and performance. He has also published papers on Operations Research in IBM developerWorks®.

Rafael Folco is a Software Engineer at the STG Linux Technology Center, IBM Brazil. He has six years of experience with Linux and IBM PowerVM. He holds a postgraduate degree in Software Engineering. His areas of expertise include Linux performance analysis, development, and testing.

Breno Leitao is an Advisory Software Engineer at the Linux Technology Center in Brazil. He has 14 years of experience with Linux. He holds a degree in Computer Science from Universidade de Sao Paulo. His areas of expertise include operating systems, virtualization, and networking. He has written extensively on Linux, mainly about networking and debugging.

Humberto Tsubamoto is an IT Specialist in Integrated Technology Delivery in Brazil. He has 12 years of experience with UNIX and seven years of experience with Storage systems. He has worked at IBM for six years. He holds a Bachelor's degree in Electrical Engineering from Escola de Engenharia Maua (EEM). His areas of expertise include implementation, support, and performance analysis of IBM Power Systems, IBM PowerHA®, IBM PowerVM, IBM AIX, Solaris, SAN, and NAS storage.

The project that produced this publication was managed by Scott Vetter, PMP, IBM ITSO.

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

David J. Bennin and Richard M. Conway
ITSO, Poughkeepsie Center

Gary Cornell
Charles R. Farrell
Brian J. King
Adriana Kobylak
Wade B. Ouren
Rajendra D. Patel
Edward C. Prosser
Ronald D. Young
IBM US

Brad Ford
i400 Technology Inc.

Nigel Griffiths
IBM UK

Nicolas Guerin
IBM France

Josiah Samuel Sathiadass
IBM India

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

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

- ▶ Send your comments in an email to:

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>



Overview

This paper refers to an improvement in PowerVM Active Memory Sharing that reduces shared memory overcommitment by eliminating memory pages in the same LPAR or across LPARs on the same server. This process is call *Active Memory Deduplication*.

This chapter provides an overview of Active Memory Deduplication and includes the following sections:

- ▶ Hardware and software requirements
- ▶ Active Memory Deduplication

1.1 Hardware and software requirements

Power Systems Active Memory Deduplication is intended to be used with IBM PowerVM Active Memory Sharing. Most of the prerequisites of Active Memory Deduplication are the same as the ones for Active Memory Sharing. Although Active Memory Sharing is supported on IBM POWER6® Systems, Active Memory Deduplication is currently supported only on IBM POWER7 Systems™. Table 1-1 lists the prerequisites for Active Memory Deduplication.

Table 1-1 Prerequisites for Active Memory Deduplication

Component	Minimum level	Comments
Hardware	POWER7 Systems supporting 740_042 firmware or later	POWER7 is the minimum hardware level that supports Active Memory Deduplication. At the time of writing it is supported on the following servers: Power 710 8231-E1C Power 720 8202-E4C Power 730 8231-E2C Power 740 8205-E6C Power 750 8233-E8C Power 770 9117-MMC Power 780 9179-MHC
I/O devices	Virtual only	All I/O devices must be virtualized by the Virtual I/O Server (VIOS).
Managed system firmware	740_042 ^a	Validate through the hardware management interface.
PowerVM	Enterprise Edition	You must have a hardware feature code or CoD-enabled feature.
Management console	HMC 7.7.4 with PTF MH01275 ^b , or SDMC 6.750.0 ^c	Check this version on the HMC/SDMC home screen. At the time of writing this paper, the Integrated Virtualization Manager (IVM) does not support Active Memory Deduplication.
Virtual I/O Server	2.1.1.10-FP-21 ^d	Issue <code>ioslevel1</code> at the VIOS shell.
AIX	6.1 TL7 7.1 TL1 ^e	Issue <code>oslevel -s</code> at the AIX shell.
IBM i	IBM i 7.1.4	Issue <code>DSPPTF</code> at the IBM i command line.
Linux	Novell SLES11 SP2 RedHat RHEL 6.2 ^e	Read section “IBM i” on page 34 for further instructions.

- a. 740_042 is the minimum firmware level for Active Memory Deduplication support.
- b. This Hardware Management Console (HMC) code level is required by firmware level 740_042.
- c. Minimal required firmware level: 740_042.
- d. Although Active Memory Deduplication is supported on this VIOS level, upgrade to the most recent level.
- e. These levels support certain additional features, such as providing deduplication statistics from within the OS performance tools, and page hints (for Linux and IBM i).

Older versions of operating systems: Older versions of the supported operating systems down to the following levels can work with Active Memory Deduplication but without features such as statistics reporting and Active Memory Deduplication page hints:

- ▶ AIX: 6.1 TL4 or 7.1
- ▶ IBM i: IBM i 6.1.1 + latest cumulative PTF package
- ▶ Linux: Novell SLES11 and RedHat RHEL 6

1.2 Active Memory Deduplication

Active Memory Deduplication is a PowerVM technology that minimizes the existence of identical memory pages in main memory space. When running workloads on traditional virtual LPARs, multiple identical data are saved across different positions in main memory. This is quite common with memory pages containing code instructions.

To optimize memory use, Active Memory Deduplication avoids data duplication in multiple distinct memory spaces. Active Memory Deduplication coalesces the data in just one physical memory page and frees the other chunks with identical data. The result is multiple logical memory pages pointing to the same physical memory page, thus saving memory space.

Active Memory Deduplication only works with partitions configured with shared memory. That is, to enable Active Memory Deduplication in your system, you must configure your LPARs to use Active Memory Sharing. The goal of Active Memory Sharing is to share memory among multiple LPARs of a system and therefore increase server consolidation by creating memory overcommitment.

When the aggregate working memory set size of the partitions exceeds the shared pool size, disk I/O occurs. This condition is called *physical memory overcommitment*, but it does not frequently happen. Nevertheless, a way to minimize these disk operations is to reduce the amount of physical memory use, thus reducing the probability of physical memory overcommitment. Active Memory Deduplication achieves this goal, thus enhancing the performance of an Active Memory Sharing enabled environment.

To deduplicate memory, PowerVM changes its logical memory map to make all identical logical memory pages point to the same physical memory page, as shown in Figure 1-1. In this figure, the pages that contain the same data are shown in blue.

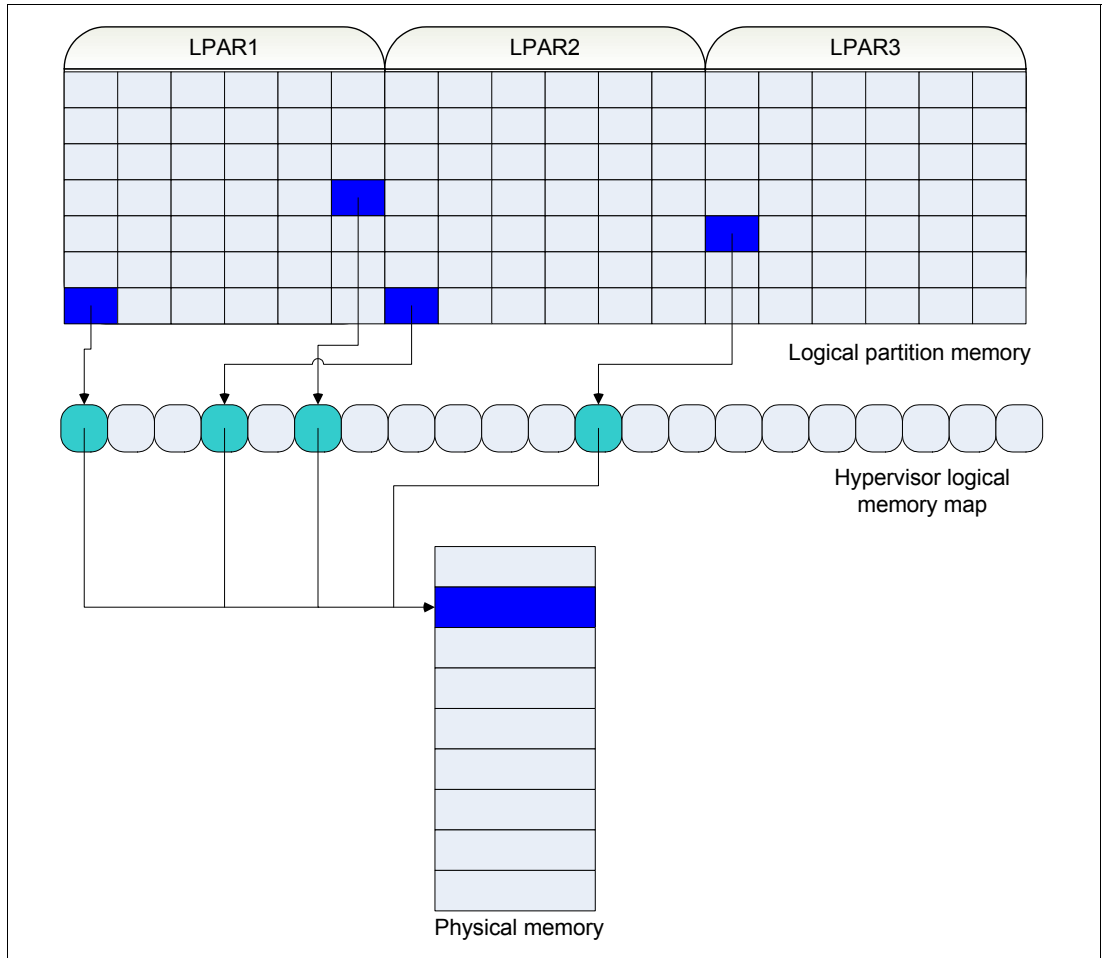


Figure 1-1 Memory structure with Active Memory Deduplication enabled

The memory savings obtained through the use of Active Memory Deduplication returns to the shared memory pool and is made available for reuse by the LPARs within the memory pool.



Concepts

This chapter introduces the main ideas behind the IBM PowerVM Advanced Memory Deduplication technology and how it helps you benefit from your Power systems.

This chapter explains several important concepts:

- ▶ Power Systems memory organization
- ▶ Hypervisor memory management
- ▶ Memory management when Active Memory Deduplication is active
- ▶ Advantages of using Active Memory Deduplication

This chapter includes the following sections:

- ▶ Objectives of active memory management
- ▶ Power systems memory management
- ▶ Active Memory Deduplication performance overview

2.1 Objectives of active memory management

Active Memory Deduplication makes system memory usage smarter. It minimizes the existence of identical memory chunks in main memory space.

Under usual workloads, a system stores data in and consumes data from main memory. Due to the nature of these workloads, the same data might be stored in multiple places all across the memory, especially data containing code instructions. Additionally, multi-user operating systems have a natural tendency to end up with duplicated memory pages.

Duplication of these memory pages is even more visible among virtualized systems on the same physical box where multiple operating system instances exist together in main memory. In this scenario, memory chunks for the same operating system or workload code are loaded into memory by each logical partition (LPAR). Storing identical data chunks in multiple different memory positions makes resources underperform.

To make a more efficient use of memory, Active Memory Deduplication avoids data duplication across separate memory positions. To accomplish this objective, Active Memory Deduplication coalesces the same data in just one memory position and frees up other duplicated memory blocks, thus optimizing memory use.

Memory terminology: The terms *memory page coalescing* and *memory deduplication* have the same meaning and are used interchangeably in this paper.

2.2 Power systems memory management

This section explains how memory management works in a shared memory environment without Active Memory Deduplication and then how it is managed when it is enabled.

2.2.1 Memory management with Active Memory Sharing

Modern system architectures have memory divided in logical blocks called memory *pages*. A server memory space is composed of millions of memory pages. A memory page is the base unit that is analyzed by PowerVM when it scans the memory looking for similar data.

Figure 2-1 shows a representation of a system containing three LPARs and a particular amount of memory organized in pages. Each square in the figure represents a memory page. Pages containing the exact same data are highlighted. In this example, four pages contain the same data across the memory. Two of these pages belong to the first LPAR, another belongs to the second LPAR, and one other to the third LPAR.

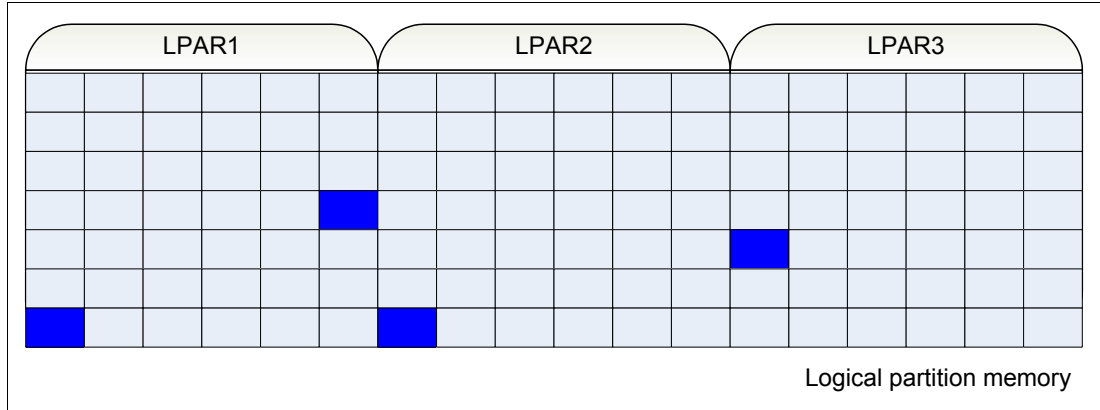


Figure 2-1 Duplicated memory pages in a LPAR or among multiple partitions

It is evident that the same data is repeatedly stored in memory in this scenario. A more advanced memory management system would store the data in memory only once and manage pointers to it instead of duplicating the data. Active Memory Deduplication deduplicates identical pages stored in separate memory chunks into only one page.

This memory translation is transparent to the operating system because it is a mechanism that is implemented in the hypervisor level, by PowerVM.

Hypervisor logical memory map

A table within PowerVM contains the mapping between the LPAR memory pages and the physical pages. A LPAR memory space is logical, that is, it does not directly reference physical memory. Every time a LPAR requests a memory page, PowerVM translates the logical memory page address to the corresponding physical memory page address. This table is called the *hypervisor logical memory map*.

The hypervisor logical memory map makes it possible to spread a memory chunk of a contiguous LPAR over the physical memory. This allows the hypervisor to more efficiently manage memory across different LPARs.

When Active Memory Deduplication is not in use, the hypervisor memory map contains only one-level memory address translation, as shown in Figure 2-2, which shows the same LPARs presented in Figure 2-1 on page 9. Figure 2-2 also introduces the hypervisor memory map showing the mapping of the duplicated pages, in blue, within the system memory.

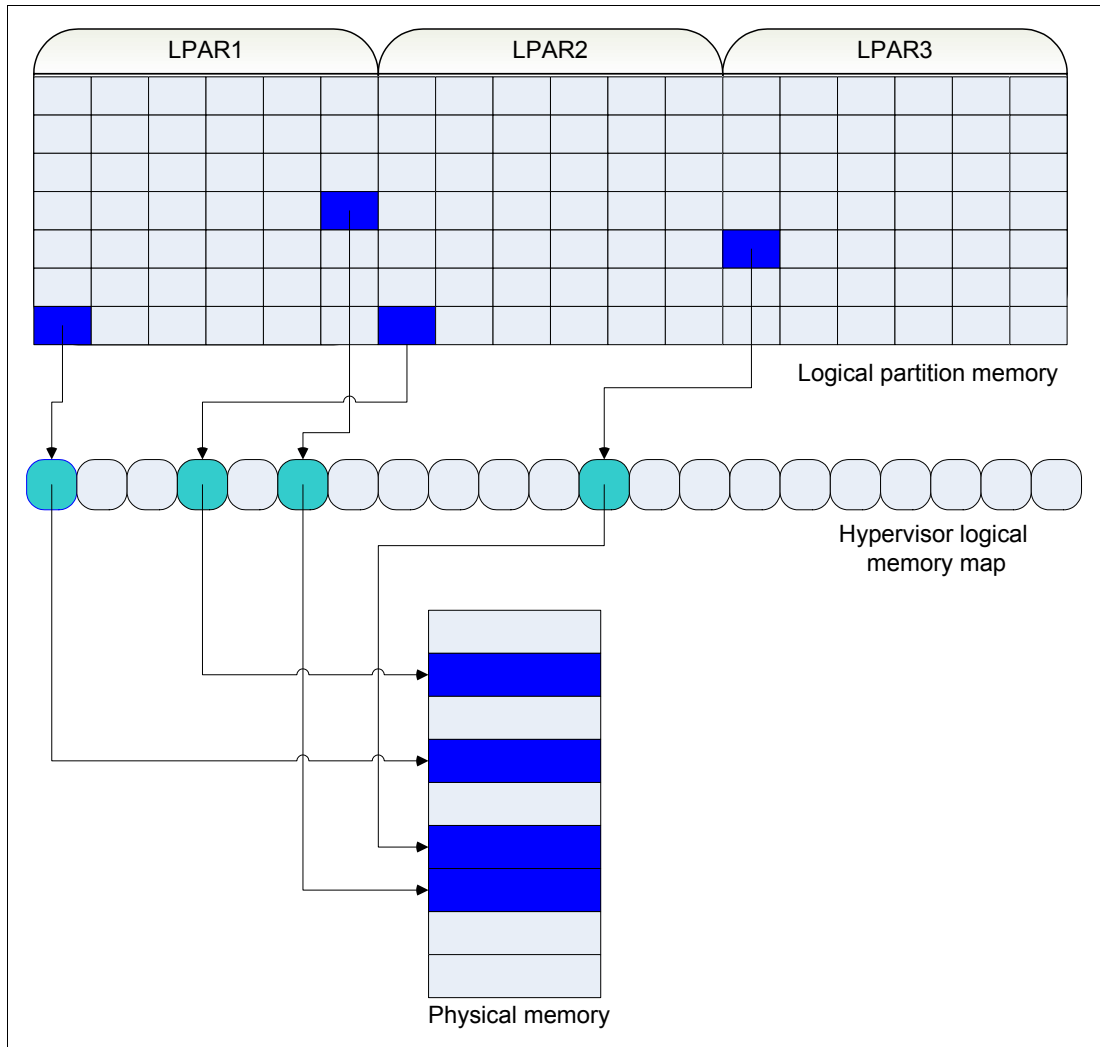


Figure 2-2 Hypervisor logical memory map translation without Active Memory Deduplication

Because the hypervisor logical memory map is managed at the hypervisor level, transparently coalescing multiple logical pages into a physical page is straightforward. For instance, when the hypervisor finds two pages containing the same data, the page deduplication algorithm modifies the hypervisor logical memory map entry to coalesce the two pages.

Suppose that two pages in the first LPAR and another page in the second LPAR are identical. When the pages are coalesced, the hypervisor simply translates them to the same single physical memory page, meaning that the three identical logical pages map to only one physical page. This improves the memory usage of the entire machine, as shown in Figure 2-3.

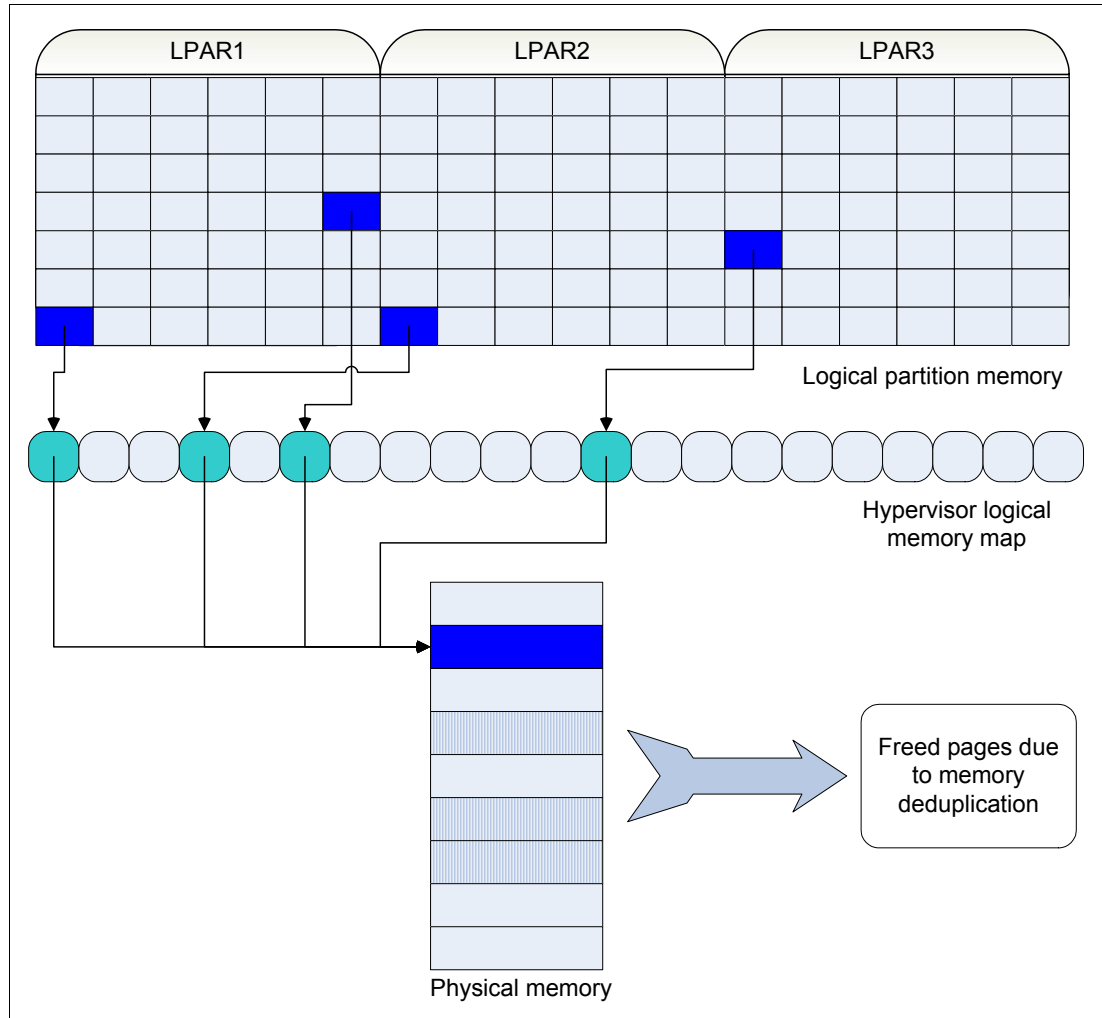


Figure 2-3 Hypervisor logical memory map translation with Active Memory Deduplication

Copy-on-write

A server might have many coalesced pages, that is, multiple LPAR pages transparently point to the same physical memory page. If so, the hypervisor must be able to duplicate this page back when the contents of this memory position is changed by any one of the LPARs.

To handle the case where a coalesced page is updated by an LPAR, a mechanism called Copy-on-Write (CoW) was created in PowerVM. CoW keeps track of all coalesced memory pages. Every time a LPAR changes in the contents of one of its coalesced pages, the hypervisor makes a duplicate copy of that coalesced physical page. The hypervisor then gives the new copy back to the write-requesting LPAR. This new duplicated page becomes a regular page and is only referenced by the LPAR that is modifying it. This change makes the write process safe and invisible to other LPARs that were sharing the physical page.

Deduplication table

In order to find duplicated pages, the hypervisor must scan the whole physical memory and compare pages. When the hypervisor performs this scan, it creates signatures for these physical pages and saves them in a *deduplication table*, as shown in Figure 2-4.

Whenever a page signature is calculated and it matches the signature of another page in the deduplication table, there is a good chance that both pages contain the same data. In this case, the two pages are compared to ensure that both contain the exact same data, as opposed to a simple hash collision. If both pages contain the exact same data, they are then deduplicated, meaning that one of the pages is freed, and the hypervisor logical memory map is properly updated. After that, the entries for both pages in the hypervisor logical memory map point to the same memory position, as shown in Figure 2-3 on page 11.

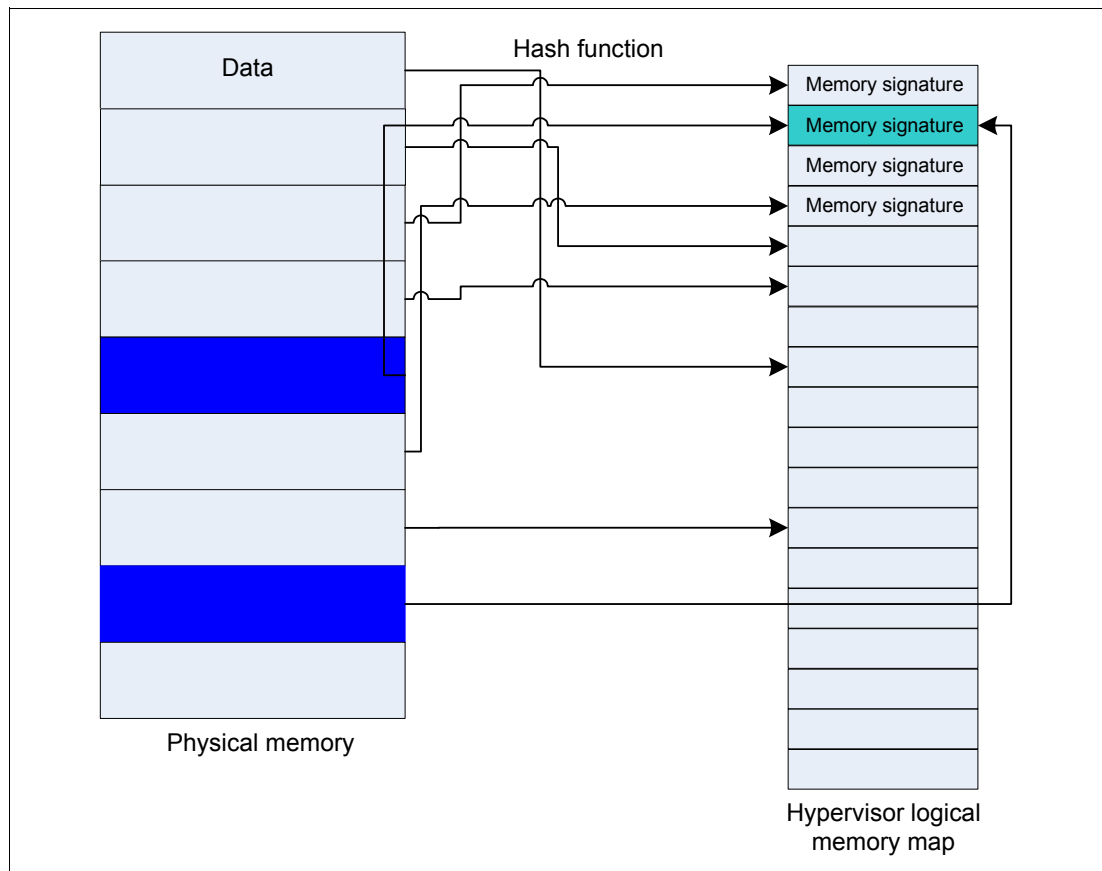


Figure 2-4 Deduplication table and the hash function

The size of the deduplication table is important for the effectiveness of memory deduplication. You may choose the default size or change it as described in Chapter 4, “Tunable parameters” on page 43.

Figure 2-5 shows a flowchart of the whole Active Memory Deduplication process performed by PowerVM.

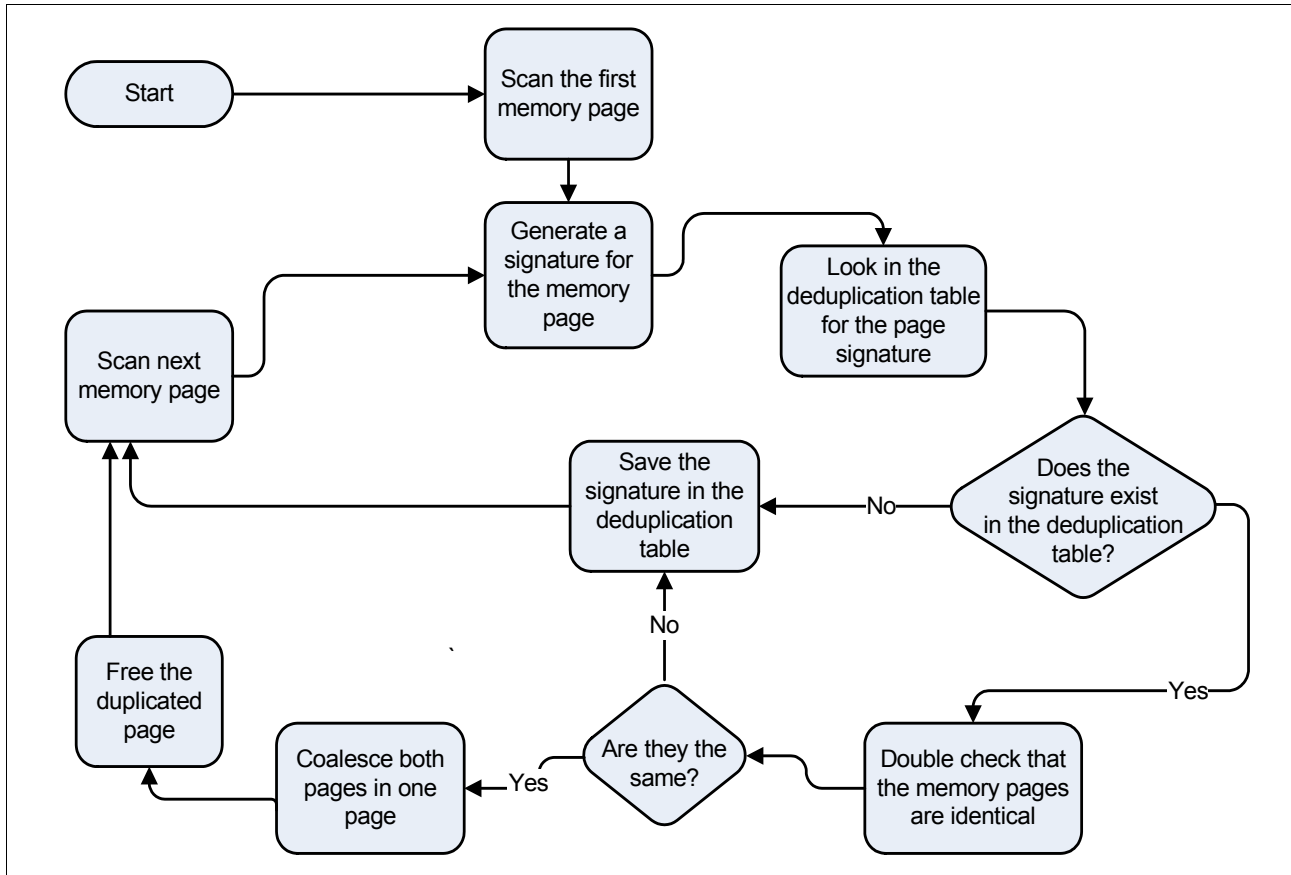


Figure 2-5 Active Memory Deduplication flowchart

Operating system collaboration

Active Memory Deduplication works independently of the operating system. Even older versions of the supported operating systems can benefit from Active Memory Deduplication, although they are not Active Memory Deduplication aware.

Newer versions of the supported operating systems can collaborate with PowerVM by indicating the pages that have good chances of being deduplicated. These pages are frequently pages containing code instructions, known as *text pages*. The versions of the operating systems that can collaborate with Active Memory Deduplication are listed in Chapter 3, “Planning for Active Memory Deduplication” on page 17.

Processor cycles

To ensure the processor cycles required by PowerVM to run Active Memory Deduplication are donated from the Virtual I/O Servers (VIOs) associated with the Active Memory Sharing pool, these VIOs need idle processing cycles that can be given to the hypervisor to coalesce pages. This is ensured by properly sizing the VIOS, as explained in section 3.2, “Sizing your systems for Active Memory Deduplication” on page 36.

2.2.2 Active Memory Sharing context for Active Memory Deduplication

Active Memory Sharing is a PowerVM technology that optimizes overall system memory use. It grants physical memory to LPARs on demand according to their workloads. To do so, the hypervisor manages a memory pool that is shared among LPARs. Depending on the workload, the amount of physical memory currently granted to LPAR varies.

Active Memory Deduplication is part of the Active Memory Sharing technology. Therefore, to enable Active Memory Deduplication, Active Memory Sharing must also be enabled and in use by the LPARs.

Active Memory Deduplication and LPARs: Memory deduplication occurs only among LPARs configured with shared memory, that is, partitions that use of Active Memory Sharing. After Active Memory Deduplication is turned on, all of the LPARs in the shared memory pool use it. In other words, you cannot turn on Active Memory Deduplication for only a selected subset of the shared memory LPARs.

Memory pool

The memory pool is a user-configurable amount of the system memory that is shared among multiple LPARs. Each LPAR is configured with a wanted amount of memory, but not all of this memory is mapped to physical memory. The amount of memory that is mapped to physical memory depends on the LPAR workload.

For example, one LPAR might not be using part of its wanted memory due to a low workload. The hypervisor might loan this unused memory to another LPAR that has a greater need for physical memory at that time. The slice of physical memory of one LPAR is reduced, whereas the other grows. This operation is controlled by the hypervisor and is transparent to the operating system.

Memory oversubscription

One of the goals of Active Memory Sharing is to increase memory use by minimizing the amount of physical memory that sits idle at any particular moment. Active Memory Sharing does so by allowing the shared pool memory to become overcommitted. In other words, the sum of the logical memory in the partitions is greater than the amount of physical memory defined in the pool. Figure 2-6 shows three LPARs that together add up more logical memory space than the pool.

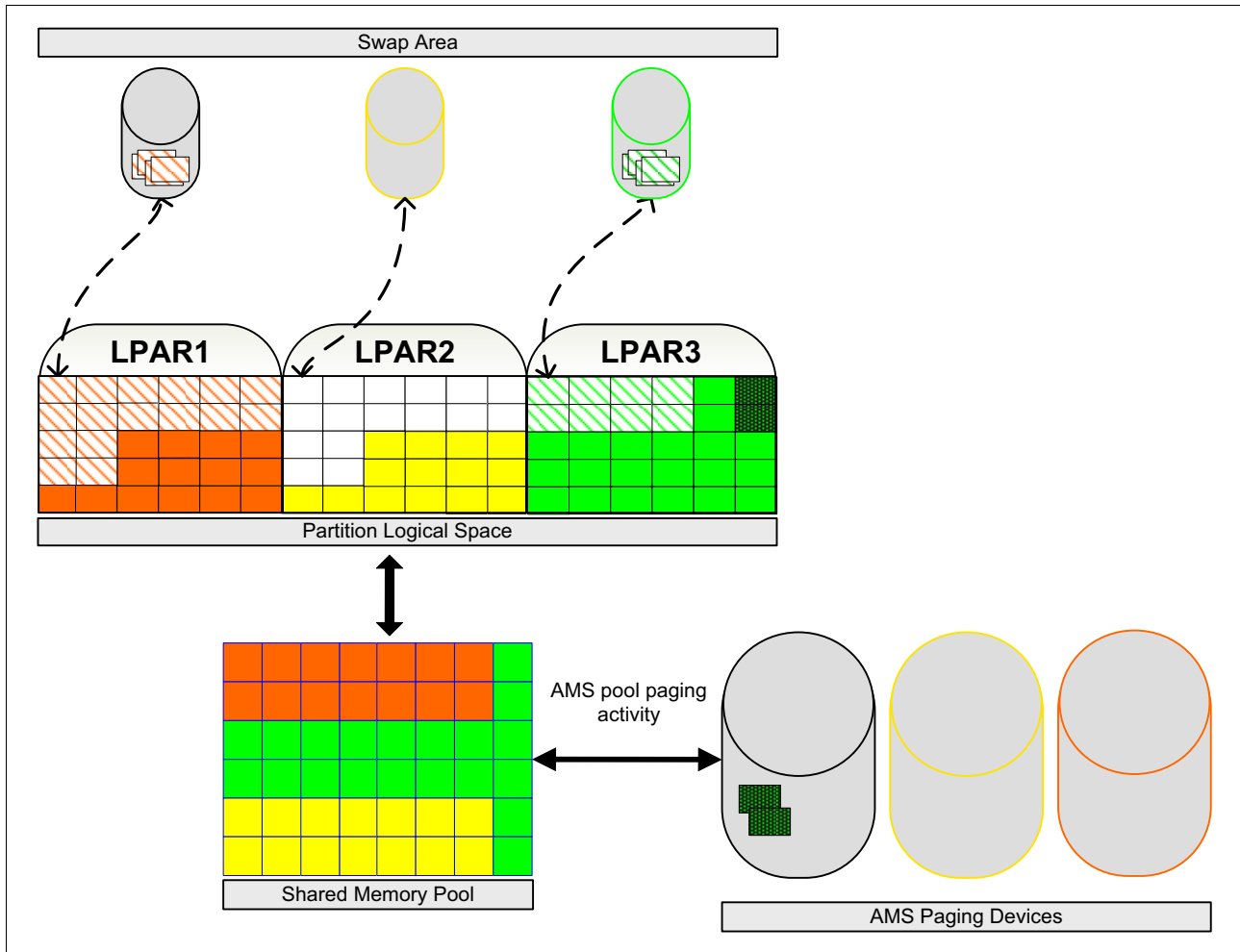


Figure 2-6 Memory pool architecture

If sized properly, the pool physical memory should be enough to hold the active memory of the partitions at a particular instant. If undersized, such as in Figure 2-6, some of the pages must be either swapped to disk by the operating systems (OS paging) or ultimately by the hypervisor through Active Memory Sharing paging.

When using Active Memory Deduplication, the coalesced pages reduce the overall amount of memory pages consumed within the shared memory pool. Consequently, more logical memory can be mapped to the physical memory.

Memory overcommitment: Active Memory Deduplication helps avoid physical memory overcommitment, which minimizes the amount of OS and Active Memory Sharing paging that occurs.

Cache

Active Memory Deduplication helps increase the main memory cache-hit ratio as more logical pages can be mapped to physical memory at any particular instant.

2.3 Active Memory Deduplication performance overview

This section gives a performance overview of a generic LPAR doing page coalescing while almost all of the other LPARs in the system are idle. This material is further explained in Chapter 6, “Configuration scenarios” on page 65, which covers all of the tuning options and shows detailed results for different scenarios.

This overview test scenario is simple: the LPAR is kept busy with a constant workload, and Active Memory Deduplication is either enabled or disabled, as shown in Figure 2-7.

Figure 2-7 shows how Active Memory Deduplication affects the use of LPAR memory several minutes after it is activated.

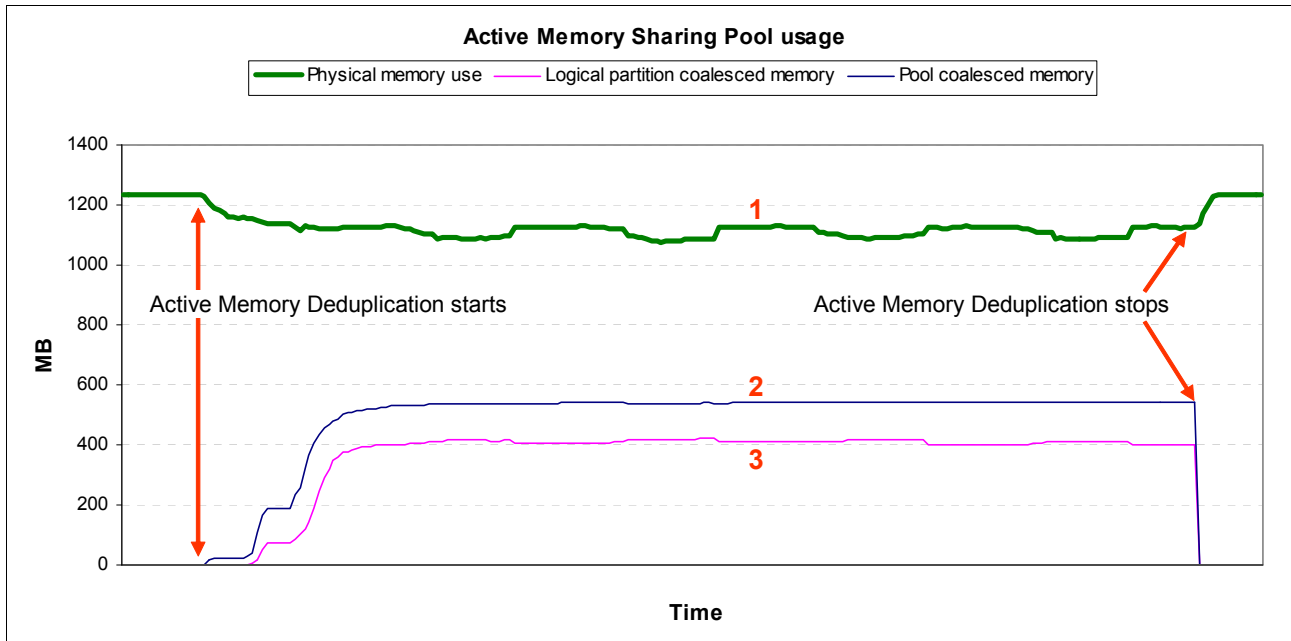


Figure 2-7 Memory deduplication on an LPAR

Line number 1 denotes the amount of real memory that the partition is using. Although the LPAR is configured to use 3 GB, it uses only 1.2 GB of physical memory. As soon as memory deduplication starts, you see a rise in the amount of memory that is deduplicated per partition (denoted by line number 3), and a rise in the amount of memory deduplicated in the entire shared memory pool among all LPARs (line number 2).

The physical memory usage decreases as soon as Active Memory Deduplication starts. In this scenario, the partition releases about 100 MB back into the pool. This all happens while the LPAR runs the same workload.

Enabling and disabling nondisruptive: Enabling and disabling Active Memory Deduplication is a nondisruptive operation.



Planning for Active Memory Deduplication

This chapter presents the points you should plan for and verify before implementing Active Memory Deduplication in your environment. After reading this section, you should be able to do these things:

- ▶ Verify whether your system meets the hardware and software requirements for Active Memory Deduplication.
- ▶ Size your environment for the implementation of Active Memory Deduplication.
- ▶ Modify your shared memory configuration to use Active Memory Deduplication.

This chapter includes the following sections:

- ▶ Checking the requirements for Active Memory Deduplication
- ▶ Sizing your systems for Active Memory Deduplication
- ▶ Enabling and disabling Active Memory Deduplication

3.1 Checking the requirements for Active Memory Deduplication

This section describes how to check the software and hardware levels required for Active Memory Deduplication, as described in Table 1-1 on page 2.

3.1.1 Managed system firmware

More than one method exists to check the firmware level of the managed system. This section shows the most common methods to check this information.

Checking the firmware level using HMC Updates

To check the firmware of the managed system from the Hardware Management Console (HMC), click **Updates** in the left pane of the HMC Welcome window. A list of managed servers and their firmware levels is displayed (Figure 3-1).

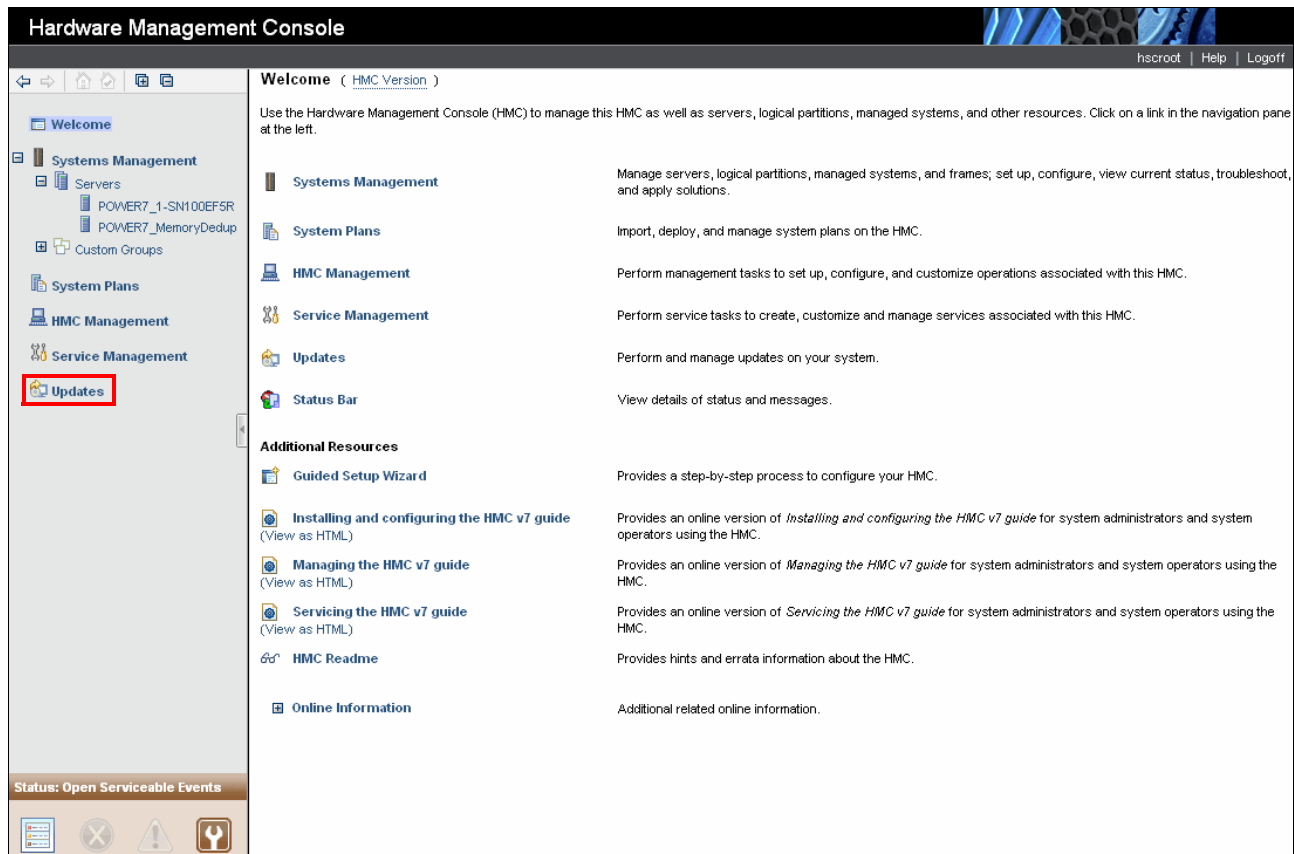


Figure 3-1 Updates link in the HMC Welcome window

Checking the firmware level using the HMC GUI Advanced System Management Interface

To check the firmware level of the managed system using the HMC GUI Advanced System Management Interface, perform the following steps:

1. In the left pane of the Welcome window of the HMC, expand **Systems Management** → **Servers**. A list of managed servers is displayed under **Servers**, as shown in Figure 3-2. In the left pane, click **Servers**.

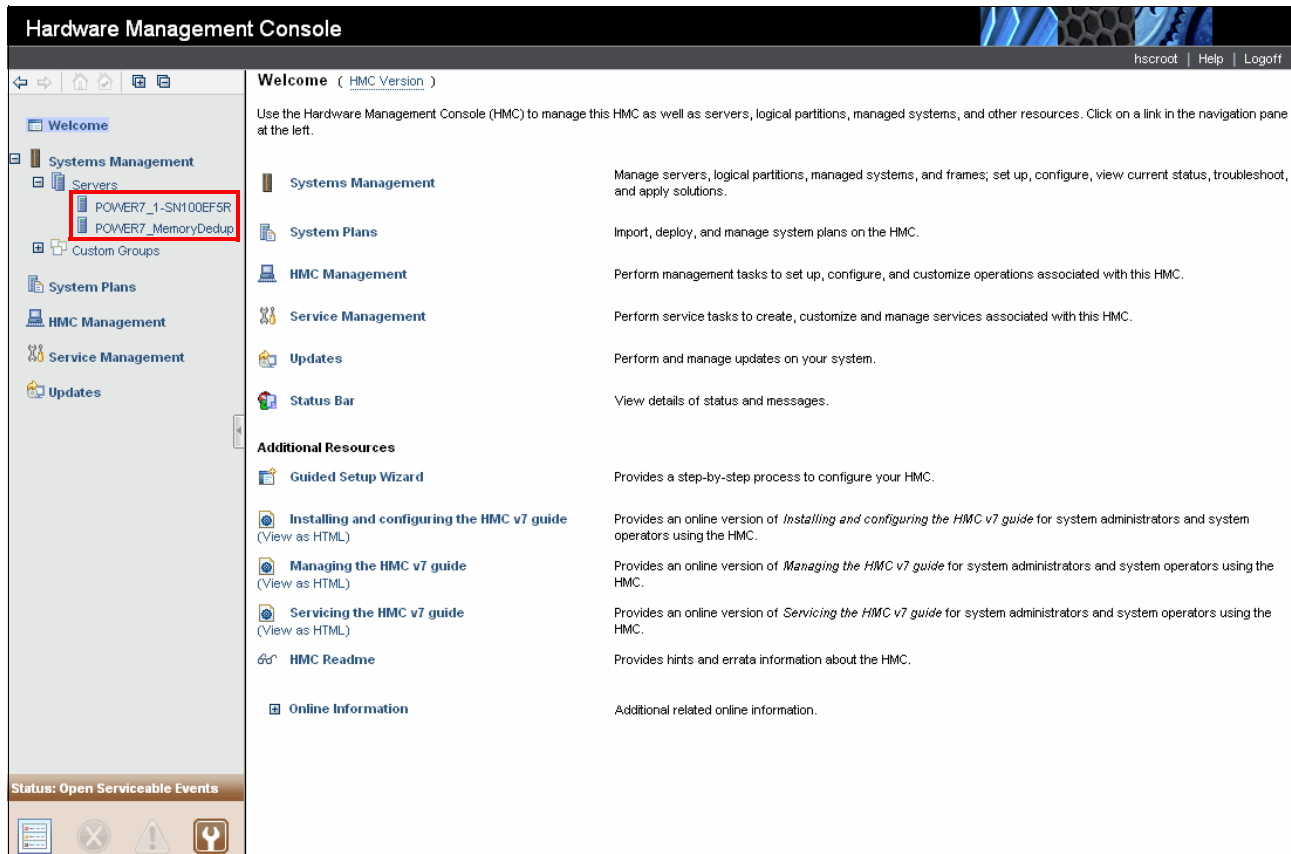


Figure 3-2 Servers list in the HMC Welcome window

2. In the right pane (Figure 3-3), select the check box of the managed system to verify.

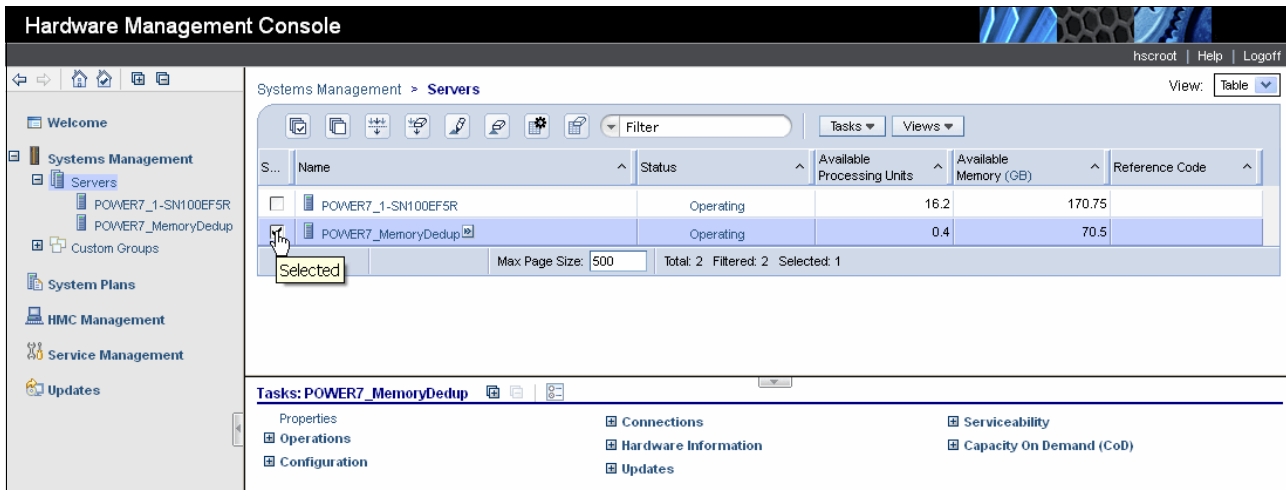


Figure 3-3 Selecting a managed system in the HMC

3. In the **Servers** field, click the arrow and then select **Operations** → **Launch Advanced System Management (ASM)**, as shown in Figure 3-4.

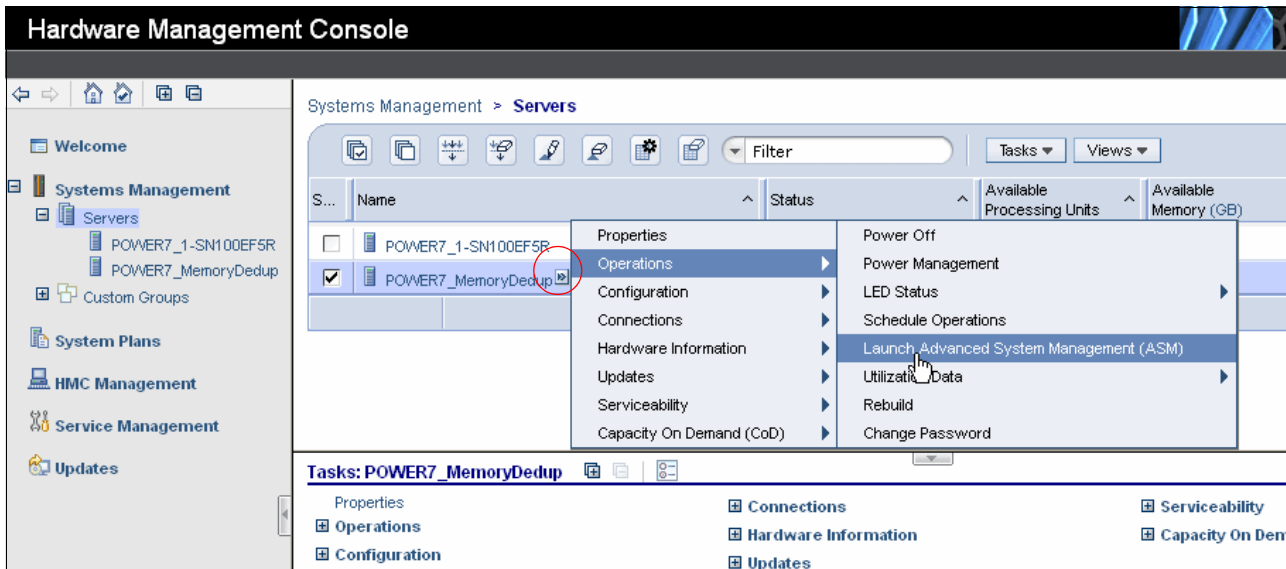


Figure 3-4 Launch Advanced System Management Interface window in the HMC

4. In the Launch ASM Interface window (Figure 3-5), click **OK** to open the Advanced System Management Interface.

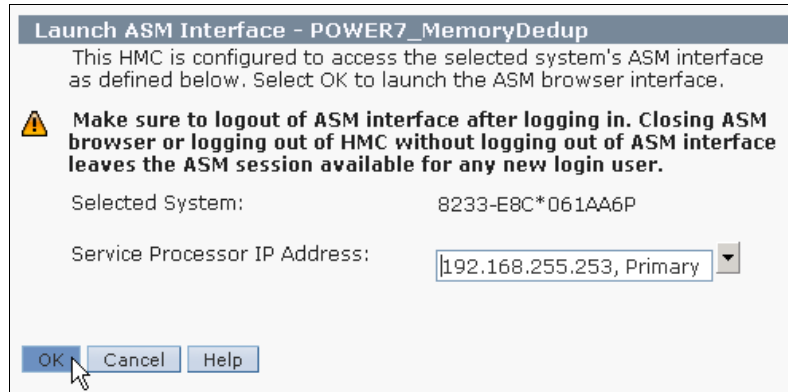


Figure 3-5 Launch ASM Interface confirmation window in the HMC

5. In the upper right of the Advanced System Management Interface Welcome window, check the firmware version, as depicted in Figure 3-6.

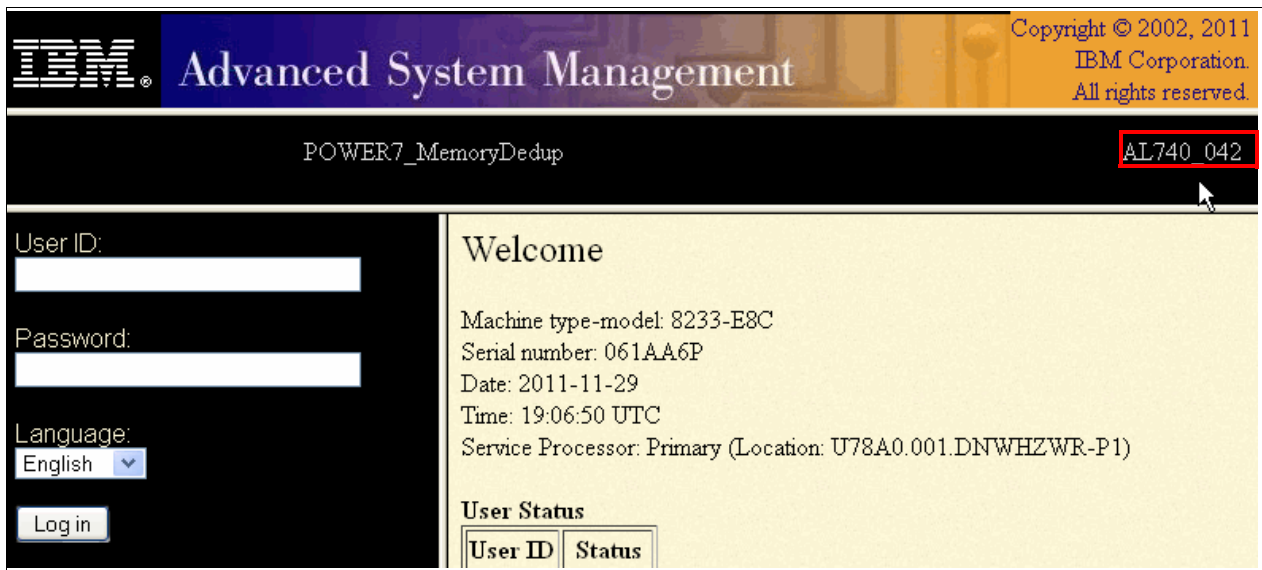


Figure 3-6 Advanced System Management Interface Welcome window

Checking the firmware level using the SDMC GUI Advanced System Management Interface

To check the firmware level of a managed system using the SDMC Advanced System Management Interface, perform the following steps:

1. In the SDMC Home window (Figure 3-7), on the **Resources** tab, expand **Hosts**, and then click the managed system for which you want to check the system firmware level.

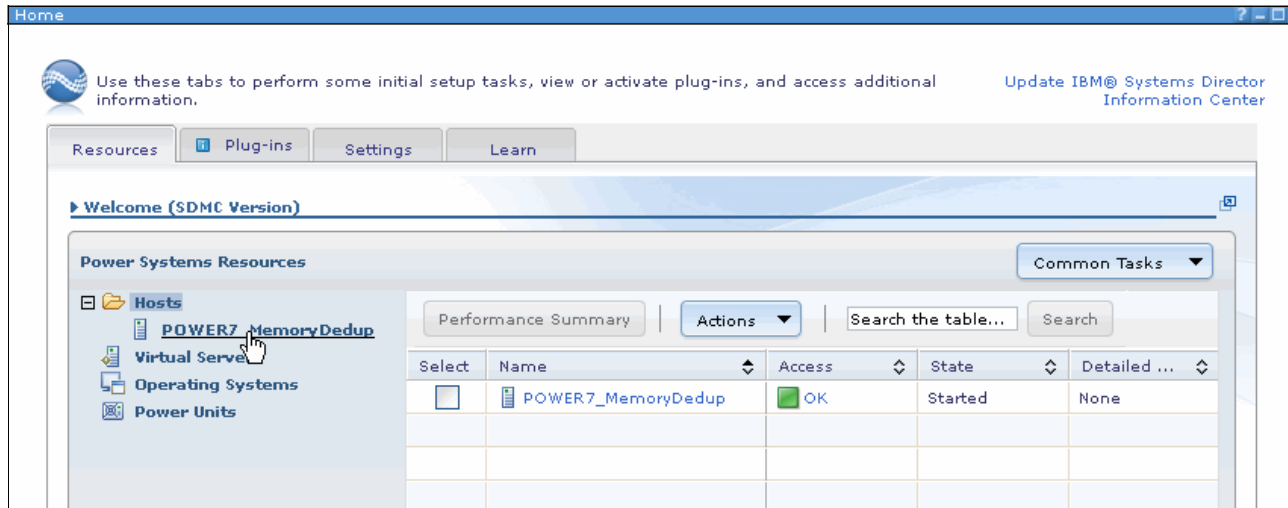


Figure 3-7 Welcome window managed systems list in the SDMC

2. In the SDMC Home window, under the **Resources** tab, right-click the managed system link, as shown in Figure 3-8.

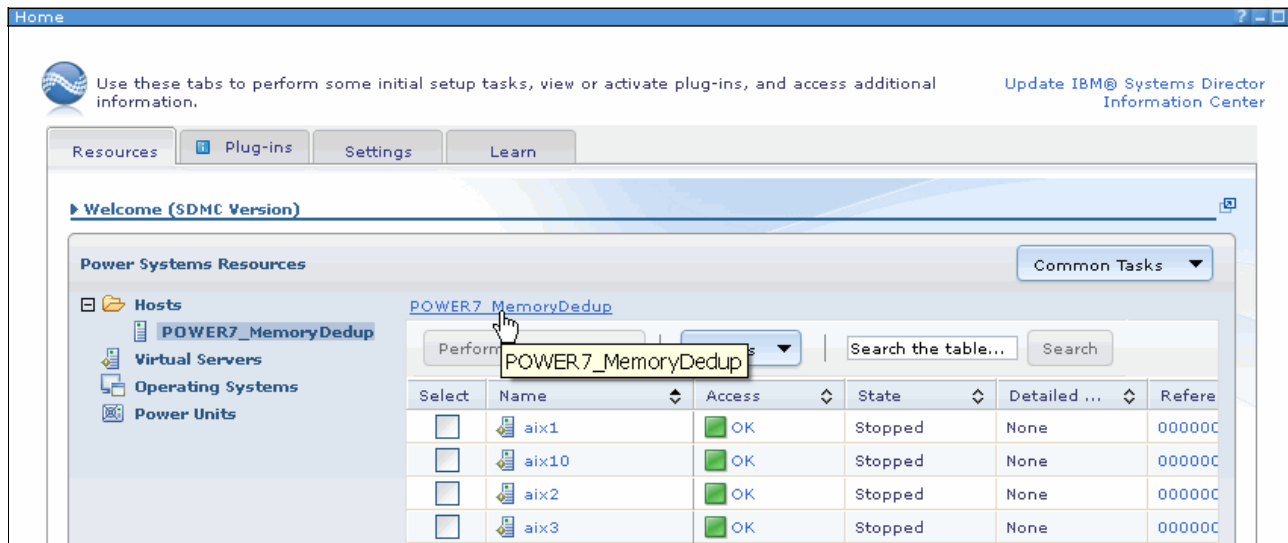


Figure 3-8 Selecting a managed system in the SDMC

- In the menu, select **Operations** → **Launch Advanced Systems Management (ASM)**, as shown in Figure 3-9.

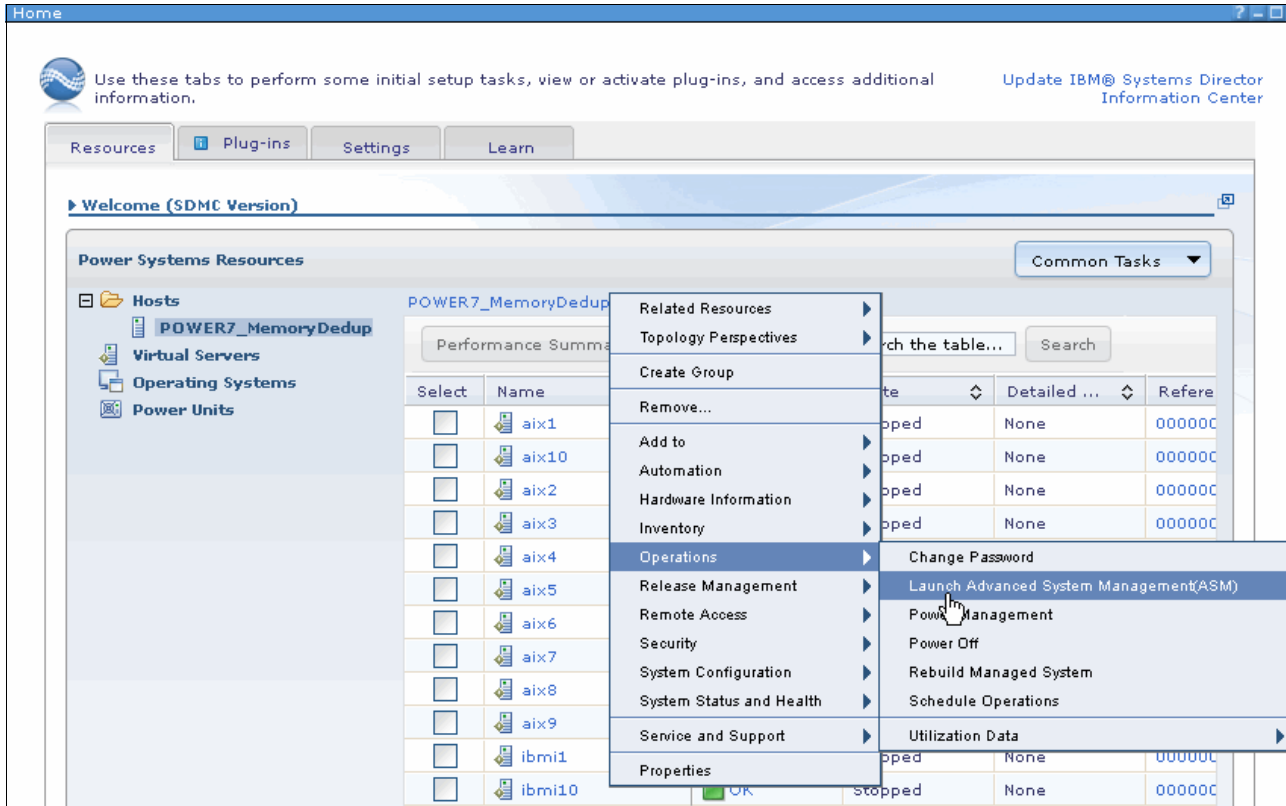


Figure 3-9 Opening the Advanced System Management Interface in the SDMC

- In the Launch Advanced System Management Interface confirmation window, click **OK**, as shown in Figure 3-10.

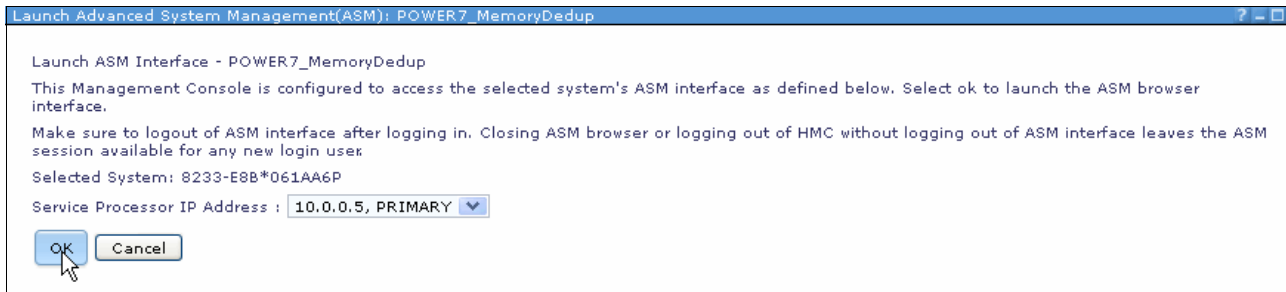


Figure 3-10 Launch Advanced System Management Interface window in the SDMC

5. In the upper right of the Advanced System Management Interface Welcome window, check the firmware version, as depicted in Figure 3-11.

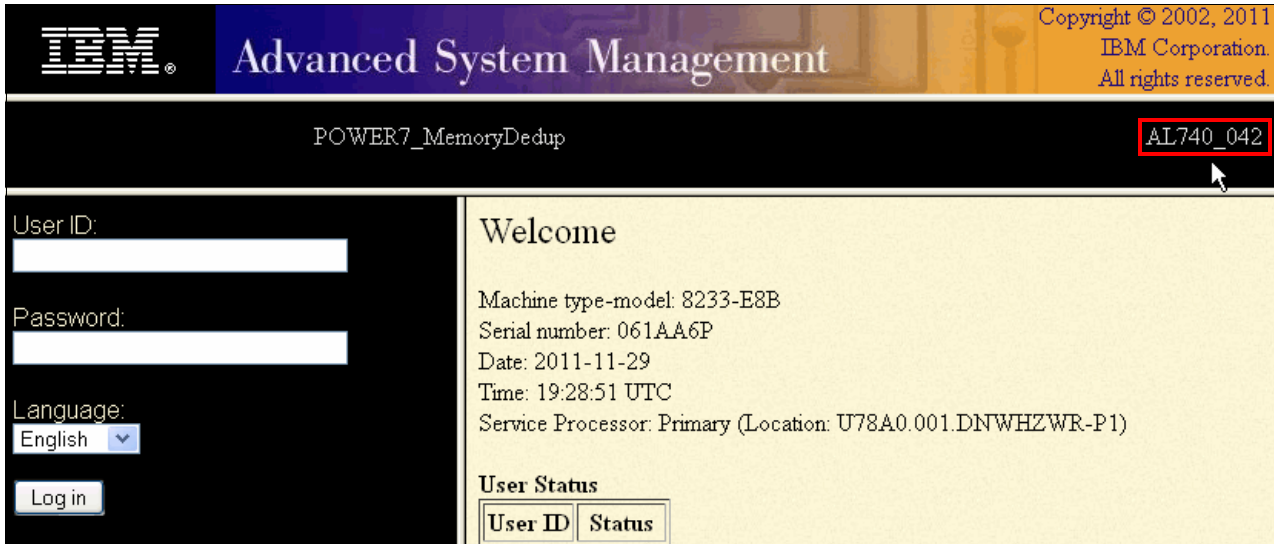


Figure 3-11 Advanced System Management Interface Welcome window

Checking the firmware level using CLI tools

The Virtual I/O Server (VIOS), AIX, IBM i, and Linux offer command-line interface (CLI) tools to show the firmware level of the managed system. If you have a running VIOS, AIX, or Linux logical partition (LPAR) on the managed system, you can use any of these partitions to gather the firmware version information.

Example 3-1 shows how to gather the firmware level information in the VIOS, Example 3-2 in AIX, Example in Linux, and Example 3-3 in IBM i.

Example 3-1 Displaying the managed system firmware level in the VIOS

```
$ lsfware
system:AL740_042 (t) AL740_042 (p) AL740_042 (t)$
```

Example 3-2 Displaying the managed system firmware level in AIX

```
root@aix1dedup / # lsmcode -c
The current permanent system firmware image is AL740_042
The current temporary system firmware image is AL740_042
The system is currently booted from the temporary firmware image.
```

Example 3-3 Displaying the managed system firmware level in IBM i

```
Display Firmware Status

Service partition . . . . . : No
Firmware update policy . . . . . : *HMC
Server IPL source . . . . . : Temporary
Firmware product ID/release . . . . . : 5733907 V1R3M0

---Server firmware---
Fix          PTF
Copy         pack   ID
*ACTIVE     AL740_042  MH01271
*TEMP       AL740_042  MH01271
*PERM       AL740_042  MH01271

Bottom

Press Enter to continue

F3=Exit  F5=Refresh  F12=Cancel
```

Example 3-4 Displaying the managed system firmware level in Linux

```
[root@linux1 ~]# lsmcode
Version of System Firmware is AL740_042 (t) AL740_042 (p) AL740_042 (t)
```

Active Memory Sharing and Active Memory Deduplication support on IBM POWER® systems: Although Active Memory Sharing is supported on POWER6 systems, Active Memory Deduplication is only supported on POWER7 systems.

3.1.2 IBM PowerVM edition

Active Memory Sharing and Active Memory Deduplication are supported only on the PowerVM Enterprise Edition. To check the capability of a managed system to use Active Memory Sharing and Active Memory Deduplication, you can use the GUI or CLI of the HMC or SDMC. This section shows the procedure to check capability using all of these methods.

Checking system capabilities using the HMC GUI

To check the capability of a managed system using the HMC GUI, perform the following steps:

1. Open the HMC managed server list by following steps 1 and 2 outlined in “Checking the firmware level using the HMC GUI Advanced System Management Interface” on page 19.
2. In the Servers panel of the HMC, check the check box for the desired managed system. Select **Properties**, as shown in Figure 3-12.

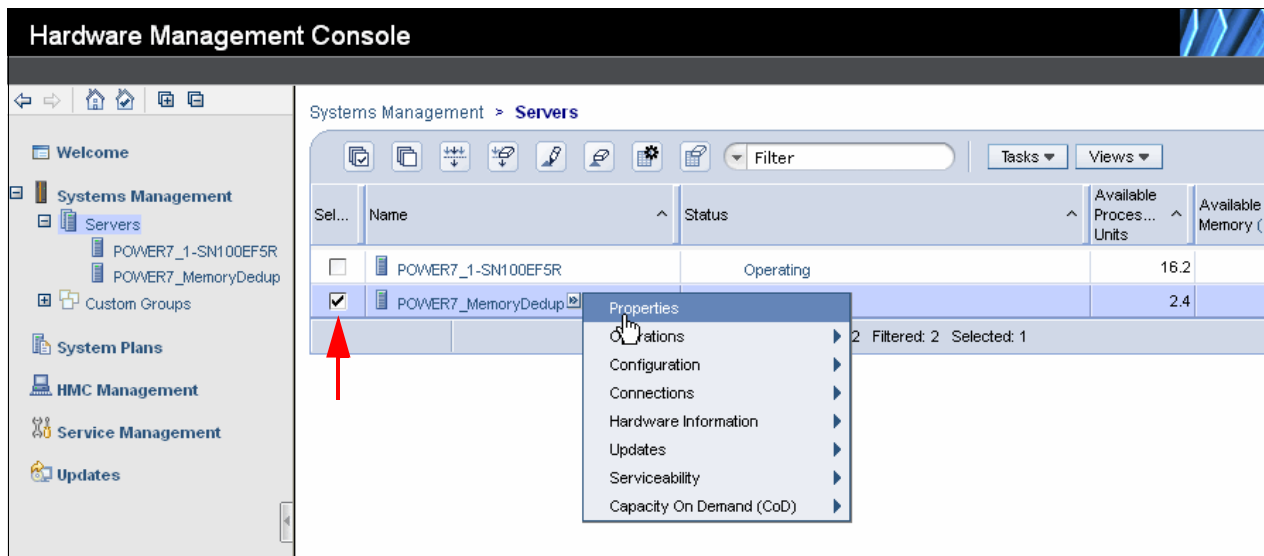


Figure 3-12 Selecting Properties for a managed system in the HMC

3. In the Properties window, select the **Capabilities** tab, as shown in Figure 3-13.

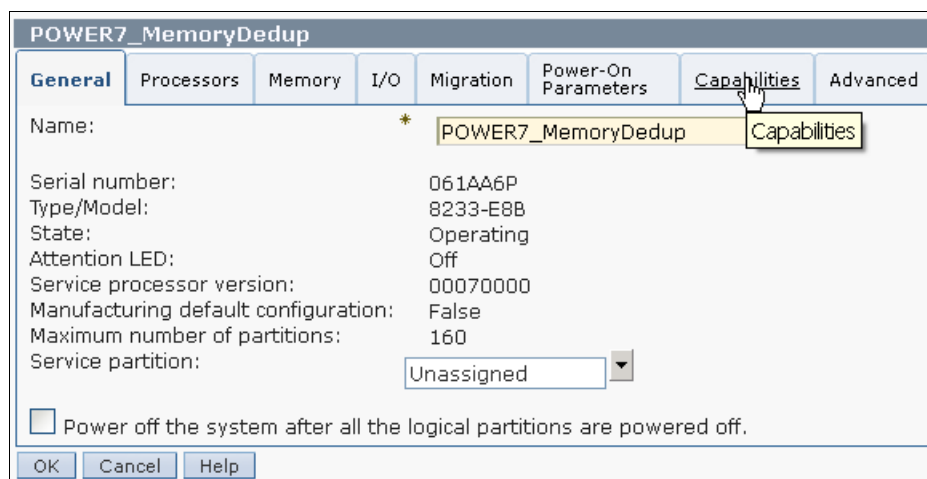
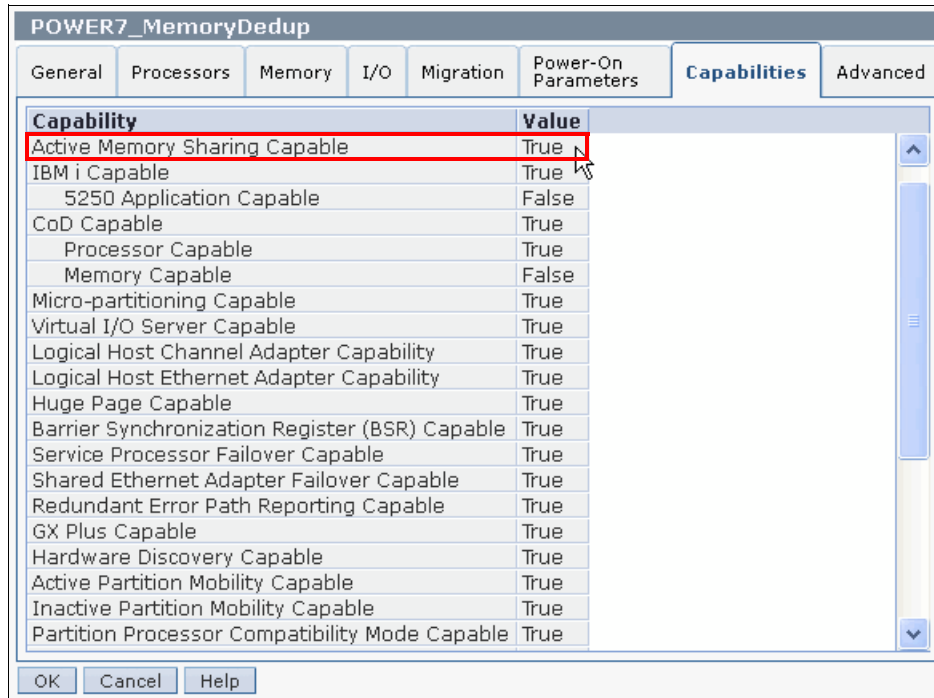


Figure 3-13 Selecting Capabilities tab in the HMC managed system Properties window

4. In the **Capability** list for the managed system, find the **Active Memory Sharing Capable** field, as shown in Figure 3-14. If its value is True, the managed system is Active Memory Sharing capable, and IBM PowerVM Enterprise edition is enabled on your system.



Capability	Value
Active Memory Sharing Capable	True
IBM i Capable	True
5250 Application Capable	False
CoD Capable	True
Processor Capable	True
Memory Capable	False
Micro-partitioning Capable	True
Virtual I/O Server Capable	True
Logical Host Channel Adapter Capability	True
Logical Host Ethernet Adapter Capability	True
Huge Page Capable	True
Barrier Synchronization Register (BSR) Capable	True
Service Processor Failover Capable	True
Shared Ethernet Adapter Failover Capable	True
Redundant Error Path Reporting Capable	True
GX Plus Capable	True
Hardware Discovery Capable	True
Active Partition Mobility Capable	True
Inactive Partition Mobility Capable	True
Partition Processor Compatibility Mode Capable	True

Figure 3-14 Managed system capabilities in the HMC

Checking system capabilities using the SDMC GUI

To check the capability of a managed system using the SDMC GUI, perform the following steps:

1. In the SDMC Home window, expand **Hosts** and select the desired managed system. At the center of the window, right-click the managed system link, and in the menu, select **Properties**, as shown in Figure 3-15.

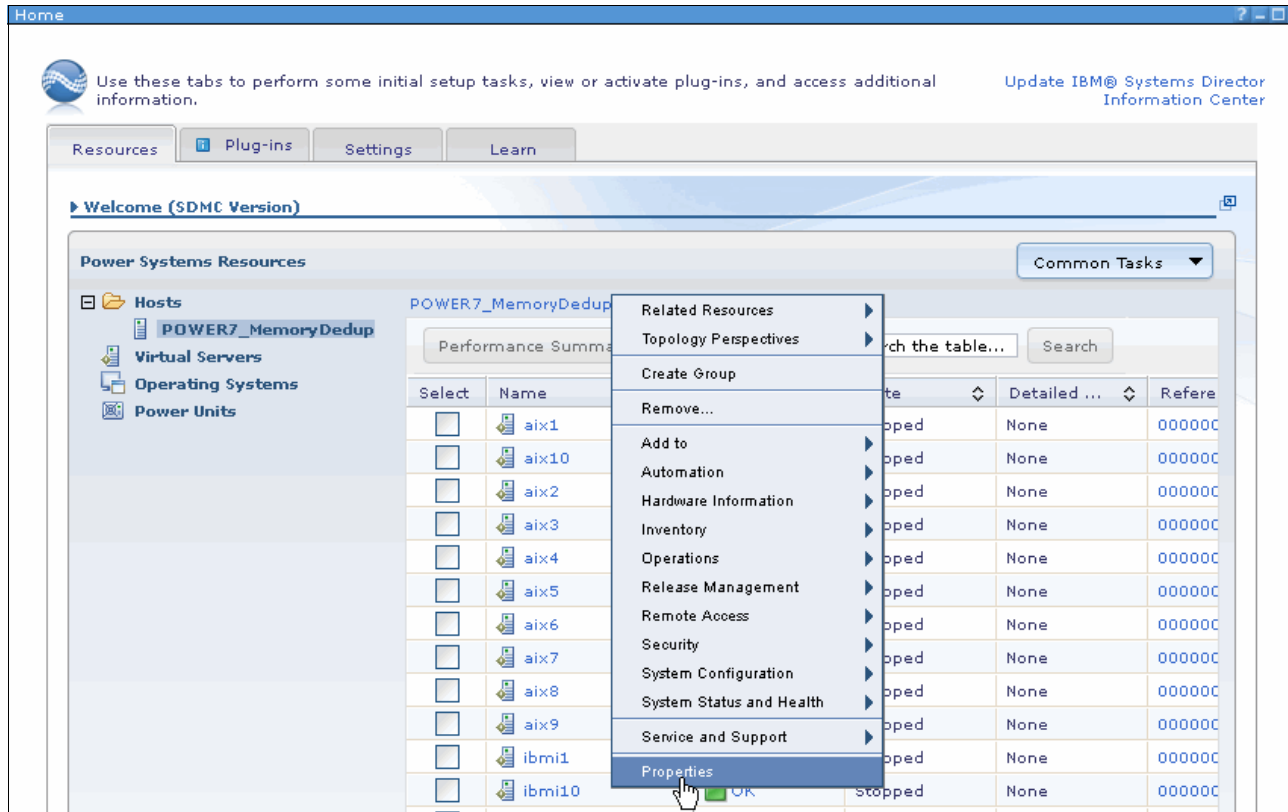


Figure 3-15 Selecting Properties for a managed system in the SDMC

- In the Resource Explorer window, under the **Additional Properties** list, click the **Edit Host** link, as shown in Figure 3-16.



Figure 3-16 Opening the Host Edit window in the SDMC

- In the Edit Host window, select the **Capabilities** tab, as shown in Figure 3-17.

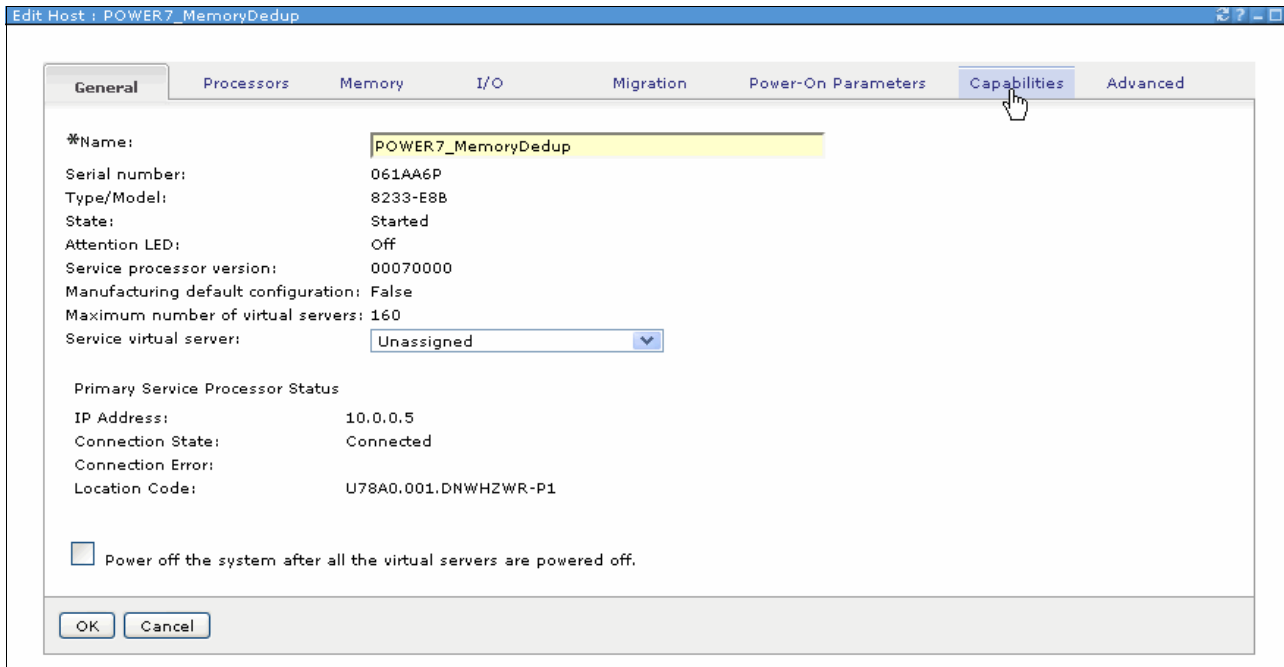


Figure 3-17 Selecting the Capability tab from the SDMC Edit Host window

- Under the **Capabilities** tab, as shown in Figure 3-18, find the **Active Memory Sharing Capable** field. If its value is True, the managed system is Active Memory Sharing capable, and IBM PowerVM Enterprise edition is enabled on your system.

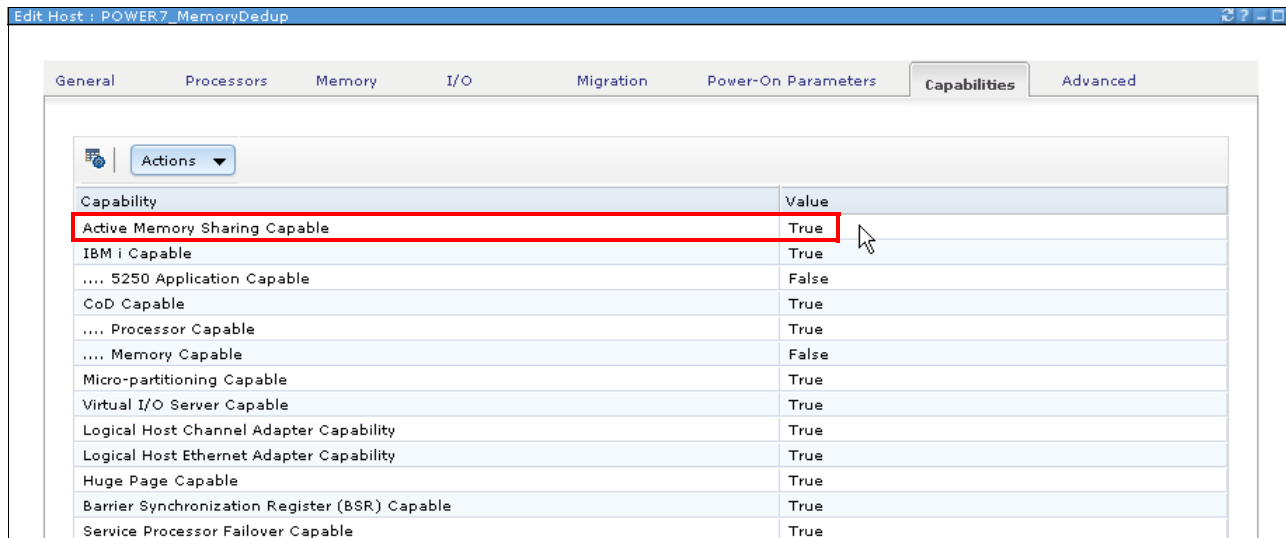


Figure 3-18 Managed system capabilities in the SDMC

Checking system capabilities using HMC CLI tools

You can check the managed system capabilities using the HMC `lssyscfg` command. The syntax of the command is as follows:

`lssyscfg -r sys -m <managed system> -F name,capabilities`

In Example 3-5, the items in bold show that the system is Active Memory Sharing and Active Memory Deduplication capable.

Example 3-5 Displaying managed system capabilities using the HMC CLI

```
hscroot@hmc6:~> lssyscfg -r sys -m POWER7_MemoryDedup -F name,capabilities
POWER7_MemoryDedup,"active_lpar_mobility_capable,inactive_lpar_mobility_capable,ac
tive_lpar_share_idle_procs_capable,active_mem_dedup_capable,active_mem_expansion_c
apable,active_mem_sharing_capable,autorecovery_power_on_capable,bsr_capable,cod_pr
oc_capable,custom_mac_addr_capable,electronic_err_reporting_capable,firmware_power
_saver_capable,hardware_power_saver_capable,hardware_discovery_capable,hca_capable
,huge_page_mem_capable,lhea_capable,lpar_affinity_group_capable,lpar_avail_priorit
y_capable,lpar_proc_compat_mode_capable,lpar_remote_restart_capable,lpar_suspend_c
apable,os400_lpar_suspend_capable,micro_lpar_capable,os400_capable,os400_net_insta
ll_capable,redundant_err_path_reporting_capable,shared_eth_failover_capable,sp_fai
lover_capable,vet_activation_capable,virtual_eth_dlpar_capable,virtual_eth_qos_cap
able,virtual_fc_capable,virtual_io_server_capable,virtual_switch_capable,vlan_stat
_capable,vtpm_capable"
```

Checking system capabilities using the SDMC CLI

The command and syntax to check the capabilities of the managed system in the SDMC CLI are the same as for the HMC CLI, as shown in Example 3-6. As command aliases are available, you might not need to use the `smcli` command prefix.

Example 3-6 Displaying the managed system capabilities using the SDMC CLI

```
sysadmin@sdmc2:~> smcli lssyscfg -r sys -m POWER7_MemoryDedup -F name,capabilities
POWER7_MemoryDedup,"active_lpar_mobility_capable,inactive_lpar_mobility_capable,ac
tive_lpar_share_idle_procs_capable,active_mem_dedup_capable,active_mem_expansion_c
apable,active_mem_sharing_capable,autorecovery_power_on_capable,bsr_capable,cod_pr
oc_capable,custom_mac_addr_capable,electronic_err_reporting_capable,firmware_power
_saver_capable,hardware_power_saver_capable,hardware_discovery_capable,hca_capable
,huge_page_mem_capable,lhea_capable,lpar_affinity_group_capable,lpar_avail_priorit
y_capable,lpar_proc_compat_mode_capable,lpar_remote_restart_capable,lpar_suspend_c
apable,os400_lpar_suspend_capable,micro_lpar_capable,os400_capable,os400_net_insta
ll_capable,redundant_err_path_reporting_capable,shared_eth_failover_capable,sp_fai
lover_capable,vet_activation_capable,virtual_eth_dlpar_capable,virtual_eth_qos_cap
able,virtual_fc_capable,virtual_io_server_capable,virtual_switch_capable,vlan_stat
_capable,vtpm_capable,vsi_on_veth_capable,vsn_phase2_capable"
```

3.1.3 Management console

To check the management console version for either the HMC or SDMC, you can use the GUI or the CLI. This section describes each method.

Checking the HMC version using the GUI

In the Welcome window of the HMC, position the mouse over the HMC Version link. A box with version information is displayed, as shown in Figure 3-19.

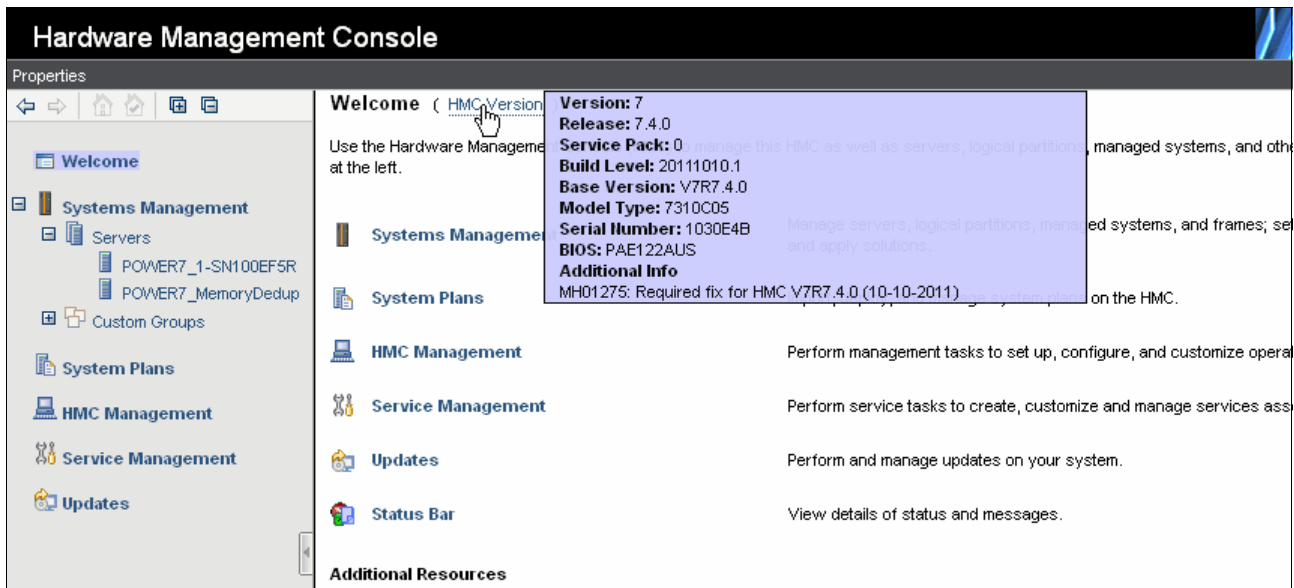


Figure 3-19 Checking the HMC version using the GUI

Checking the SDMC version using the GUI

On the SDMC Home window of the SDMC, position the mouse over the SDMC Version link. A box with version information is displayed, as shown in Figure 3-20.

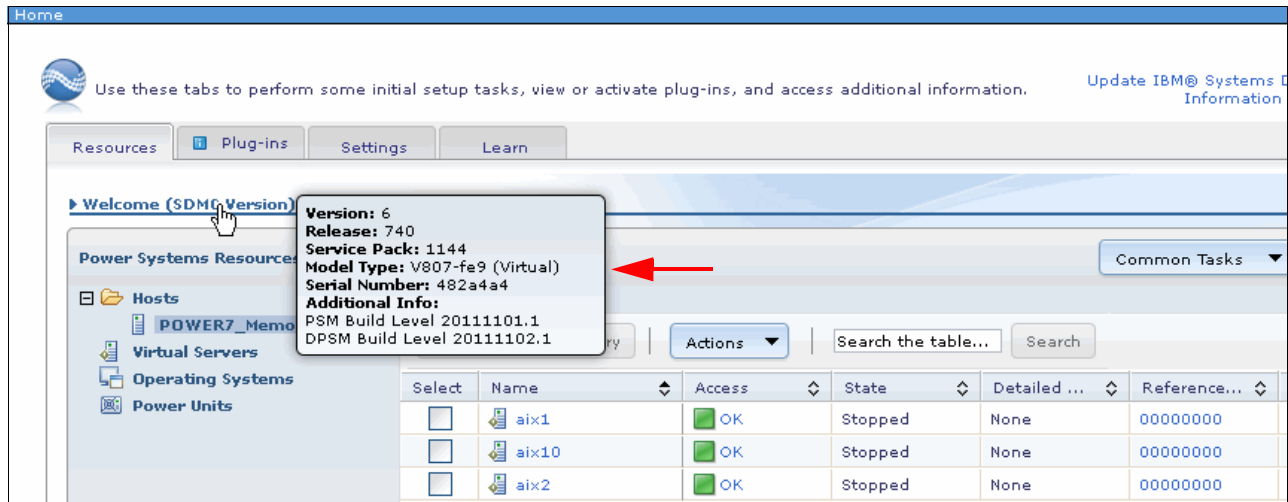


Figure 3-20 SDMC the version level using the GUI

Checking the HMC version using the CLI

To retrieve the HMC version level from the CLI, use the `lshmc -V` command, as shown in Example 3-7.

Example 3-7 Displaying the HMC version using the CLI

```
hscroot@hmc6:~> lshmc -V
"version= Version: 7
Release: 7.4.0
Service Pack: 0
HMC Build level 20111010.1
MH01275: Required fix for HMC V7R7.4.0 (10-10-2011)
", "base_version=V7R7.4.0
"
```

Checking the SDMC version using the CLI

You can check the SDMC version using the `lsconfig -V` command, as shown in Example 3-8.

Example 3-8 Displaying the SDMC version using the CLI

```
sysadmin@sdmc2:~> lsconfig -V
"version= Version: 6
Release: 730
Service Pack: 0
SDMC Build level 1.Thu May 12 10:04:41 CST 2011
PSM Build Level 20110607.1
DPSM Build Level 20110607.1
", "base_version=V6R730 "
MF53082: Required fix for SDMC V6R730 (06-15-2011)
"
```

3.1.4 Virtual I/O Server version

From the Virtual I/O Server (VIOS), shell as the *padmin* user, issue the `ioslevel` command to check your VIOS version. See Example 3-9.

Example 3-9 Checking the VIOS version

```
$ ioslevel  
2.2.1.1
```

3.1.5 Operating system

This section describes how to check the prerequisites for each of the three operating systems you are able to run on Power: AIX, IBM i, and Linux.

AIX

To check the version of your AIX operating system, use the `oslevel -s` command, as shown in Example 3-10.

Example 3-10 Checking the operating system version

```
root@aix1dedup / # oslevel -s  
7100-01-01-1141
```

IBM i

In IBM i, use the **DSPPTF** command to check the operating system's version, as shown in Figure 3-21.

```
MAIN                                IBM i Main Menu                                System:  X001AA6P

Select one of the following:

    1. User tasks
    2. Office tasks
    3. General system tasks
    4. Files, libraries, and folders
    5. Programming
    6. Communications
    7. Define or change the system
    8. Problem handling
    9. Display a menu

    11. IBM i Access tasks

    90. Sign off

Selection or command
====> DSPPTF

F3=Exit  F4=Prompt  F9=Retrieve  F12=Cancel  F23=Set initial menu
```

Figure 3-21 IBM i DSPPTF command

Figure 3-22 illustrates the result of the **DSPPTF** command.

```

                                Display PTF Status
                                System:  X001AA6P
Product ID . . . . . : 5770999
IPL source . . . . . : ##MACH#A
Release of base option . . . . . : V7R1M0 L00

Type options, press Enter.
  5=Display PTF details  6=Print cover letter  8=Display cover letter

    PTF
Opt  ID      Status
RE11067  Permanently applied
RE10187  Permanently applied
RE10084  Permanently applied
RE10026  Permanently applied
QLL2924  Permanently applied
MF99002  Permanently applied
MF99001  Permanently applied
MF52514  Permanently applied
                                IPL
                                Action
                                None
                                None
                                None
                                None
                                None
                                None
                                None
                                None
                                None
                                None
                                None
                                More...

F3=Exit  F11=Display alternate view  F17=Position to  F12=Cancel

```

Figure 3-22 IBM i DSPPTF command result

Linux

To check the version of your Linux operating system, use the **cat /etc/redhat-release** command in RedHat Linux (shown in Example 3-11). You can use the **cat /etc/SuSE-release** command on SuSE SLES Linux (shown in Example 3-12).

Example 3-11 Displaying the RedHat Linux version

```
[root@linux1 ~]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 6.2 GA (Santiago)
```

Example 3-12 Displaying the SLES version

```
linux11:~ # cat /etc/SuSE-release
SUSE Linux Enterprise Server 11 (ppc64)
VERSION = 11
PATCHLEVEL = 2
```

3.2 Sizing your systems for Active Memory Deduplication

To get good performance results with Active Memory Deduplication, your systems must be sized appropriately. Deduplication requires the resources listed in “Processor cycles” on page 13:

- ▶ Memory space required for PowerVM to manage deduplicated pages
- ▶ Amount of processing resources required to compute memory page signatures

3.2.1 Memory sizing requirements

The amount of memory used for page deduplication control is managed directly by the hypervisor and is not taken from the shared memory pool. This fact implies that the memory configuration of your LPARs, whether they are VIOS, AIX, Linux, or IBM clients, does not need to be modified when you turn on Active Memory Deduplication for your shared memory LPARs.

The amount of memory used by the hypervisor for page deduplication control can be computed with the following formula:

$$\text{deduplication table size} = \text{AMS max pool size} \times \text{deduplication table ratio}$$

You can tune both the Active Memory Sharing maximum pool size and the deduplication table ratio to manipulate this amount of memory. This principle is explained in further detail in Chapter 4, “Tunable parameters” on page 43. However, the memory savings from deduplicating memory, even in a small environment, more than compensate for this memory consumption.

LPAR memory: The memory configuration of your existing shared memory LPARs does not need to be revisited to implement Active Memory Deduplication in your environment.

3.2.2 Processor sizing requirements

The processing power required to deduplicate memory pages is donated by the VIOSs when they are idle. Therefore, size them with an appropriate slack (additional capacity) to ensure that they have idle cycles to donate. The amount of slack needed is described in the configuration scenarios in section 6.7, “Virtual I/O Server processor sizing” on page 75. Best practice values are suggested in Chapter 7, “Best practices” on page 77.

As a rule of thumb, start by allowing your VIOSs to obtain extra processing units from the processor shared pool. Follow VIOS processor configuration best practices described in *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940:

- ▶ Uncap your VIOSs.
- ▶ Set them to the highest uncapped weight of 255.

Then, to create an initial slack for memory deduplication, make the number of virtual processors greater than the amount of desired processing units by at least 0.1 units. Here are several configuration examples for a VIOS:

- ▶ If the number of configured desired processing units is between 0.1 and 0.9, make the number of virtual processors at least 1.
- ▶ If the number of configured desired processing units is between 1.0 and 1.9, make the number of virtual processors at least 2.
- ▶ If the number of configured desired processing units is between 2.0 and 2.9, make the number of virtual processors at least 3.

Keep in mind that this process does not consider any analysis of the workload type, average amount of processor units available in the shared processor pool, or any other factor. This way, you create some initial slack in the VIOSs so that you can turn on Active Memory Deduplication without compromising your current VIOS load. In most cases, having a slack of 0.1 processing units is enough. A more in-depth analysis and other alternatives for creating slacks is presented in section 4.3, “Spare Virtual I/O Server processor cycles” on page 48.

As you can see, sizing your existing shared memory environment to enable Active Memory Deduplication requires very small changes in resource requirements. If you already have a 0.1 processing unit slack in your VIOSs, then you really do not need to do anything but turn on Active Memory Deduplication.

3.3 Enabling and disabling Active Memory Deduplication

Active Memory Deduplication acts within an Active Memory Sharing pool, and to use Active Memory Deduplication, you first need to define an Active Memory Sharing pool. See *IBM PowerVM Virtualization Active Memory Sharing*, REDP-4470, for the steps in creating and configuring an Active Memory Sharing pool.

After creating and configuring the Active Memory Sharing pool, you must enable Active Memory Deduplication. You can enable it either from the GUI or CLI of the HMC or SDMC, as explained in the following sections.

No additional configuration modifications are necessary to allow the LPARs to use Active Memory Deduplication. The LPARs only need to be configured with shared memory to indicate which partitions are needed. See *IBM PowerVM Virtualization Active Memory Sharing*, REDP-4470, to verify the steps in allowing LPARs to use Active Memory Sharing.

3.3.1 Enabling Active Memory Deduplication using the HMC GUI

To enable Active Memory Deduplication by using the HMC GUI:

1. In the Servers pane of the HMC, check the desired managed system check box, as shown in Figure 3-23.

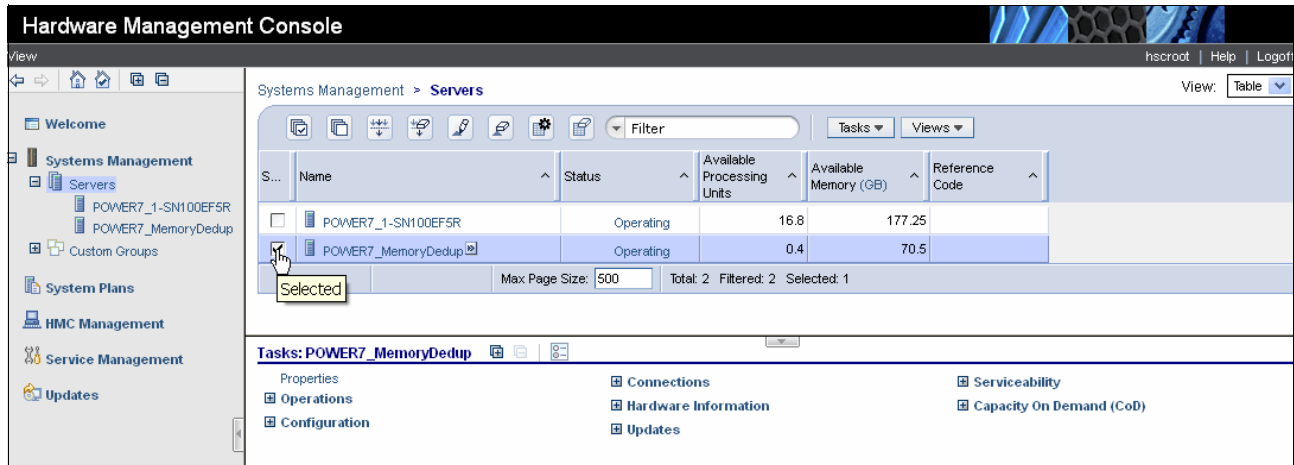


Figure 3-23 Selecting a managed system in the HMC

2. In the Tasks pane of the HMC (Figure 3-24), expand **Configuration** → **Virtual Resources**. Click **Shared Memory Pool Management**.

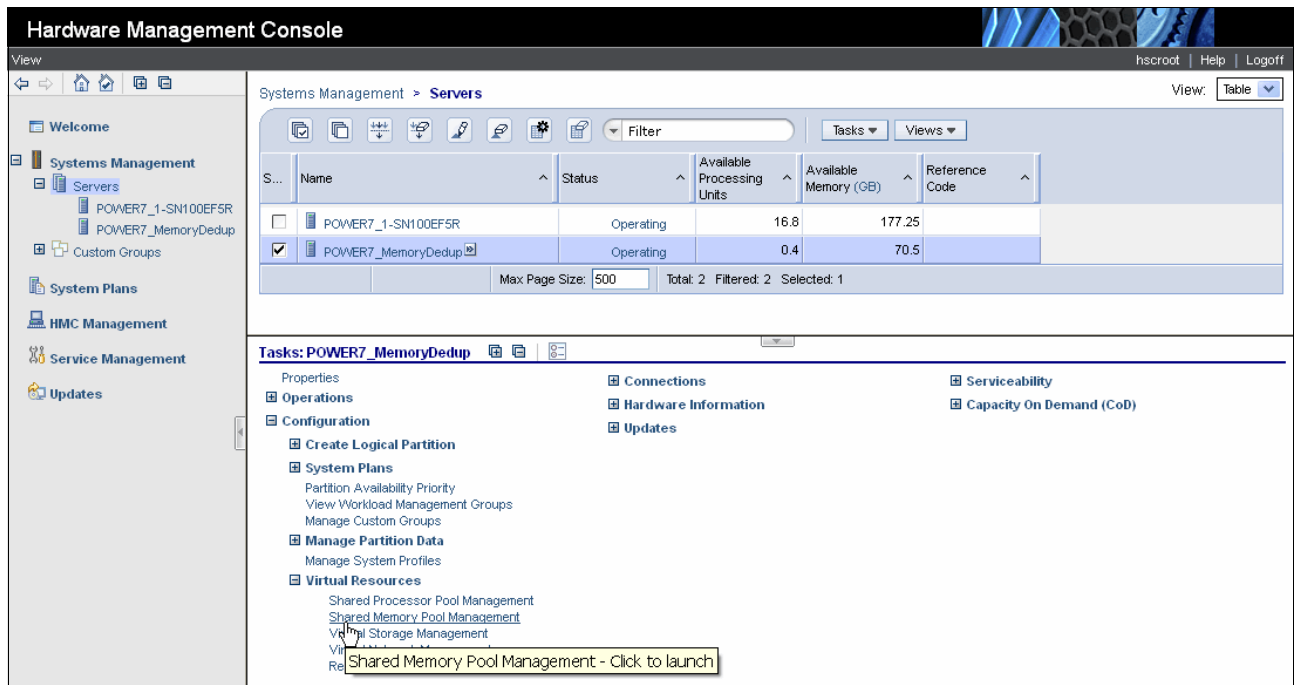


Figure 3-24 Opening the Shared Memory Pool Management window in the HMC

3. In the Pool Properties window (Figure 3-25), check the **Enable Active Memory Deduplication** check box. Click **OK**.

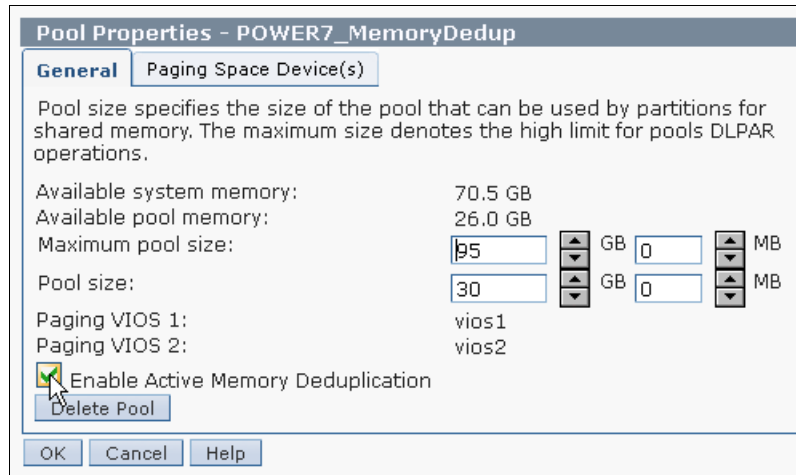


Figure 3-25 Enabling Active Memory Deduplication from the HMC Pool Properties window

3.3.2 Disabling Active Memory Deduplication using the HMC GUI

To disable Active Memory Deduplication, follow the same steps described in 3.3.1, “Enabling Active Memory Deduplication using the HMC GUI” on page 38. In the Pool Properties window (Figure 3-26), clear the option **Enable Active Memory Deduplication**, and then click **OK**,

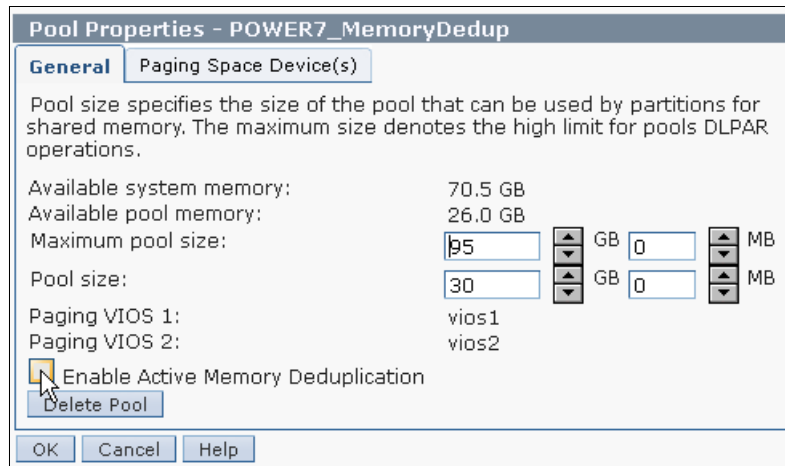


Figure 3-26 Disabling Active Memory Deduplication from the HMC Pool Properties window

3.3.3 Enabling or disabling Active Memory Deduplication using the HMC CLI

To enable Active Memory Deduplication, use the `chhwres` command, as follows:

```
chhwres -r mempool -m <managed system> -o s -a "mem_dedup=1"
```

To disable Active Memory Deduplication, set the `mem_dedup` parameter to 0, as follows:

```
chhwres -r mempool -m <managed system> -o s -a "mem_dedup=0"
```

To check if Active Memory Deduplication is enabled or disabled, use the `lshwres` command, as follows:

```
lshwres -r mempool -m <managed system> -F mem_dedup>
```

If the command returns 1, Active Memory Deduplication is enabled. If it returns a value of 0, Active Memory Deduplication is disabled. Remember to replace the **managed system** field with the name of your managed system, as in the previous examples.

3.3.4 Enabling Active Memory Deduplication using the SDMC GUI

To enable Active Memory Deduplication using the SDMC GUI:

1. In the Welcome window of the SDMC, expand the **Hosts** menu in the left panel. A list of managed systems is displayed. Click the desired managed system.
2. At the top center of the Welcome window, right-click the link of the managed system and select **System Configuration** → **Virtual Resources** → **Shared Memory Pool Management**, as shown in Figure 3-27.

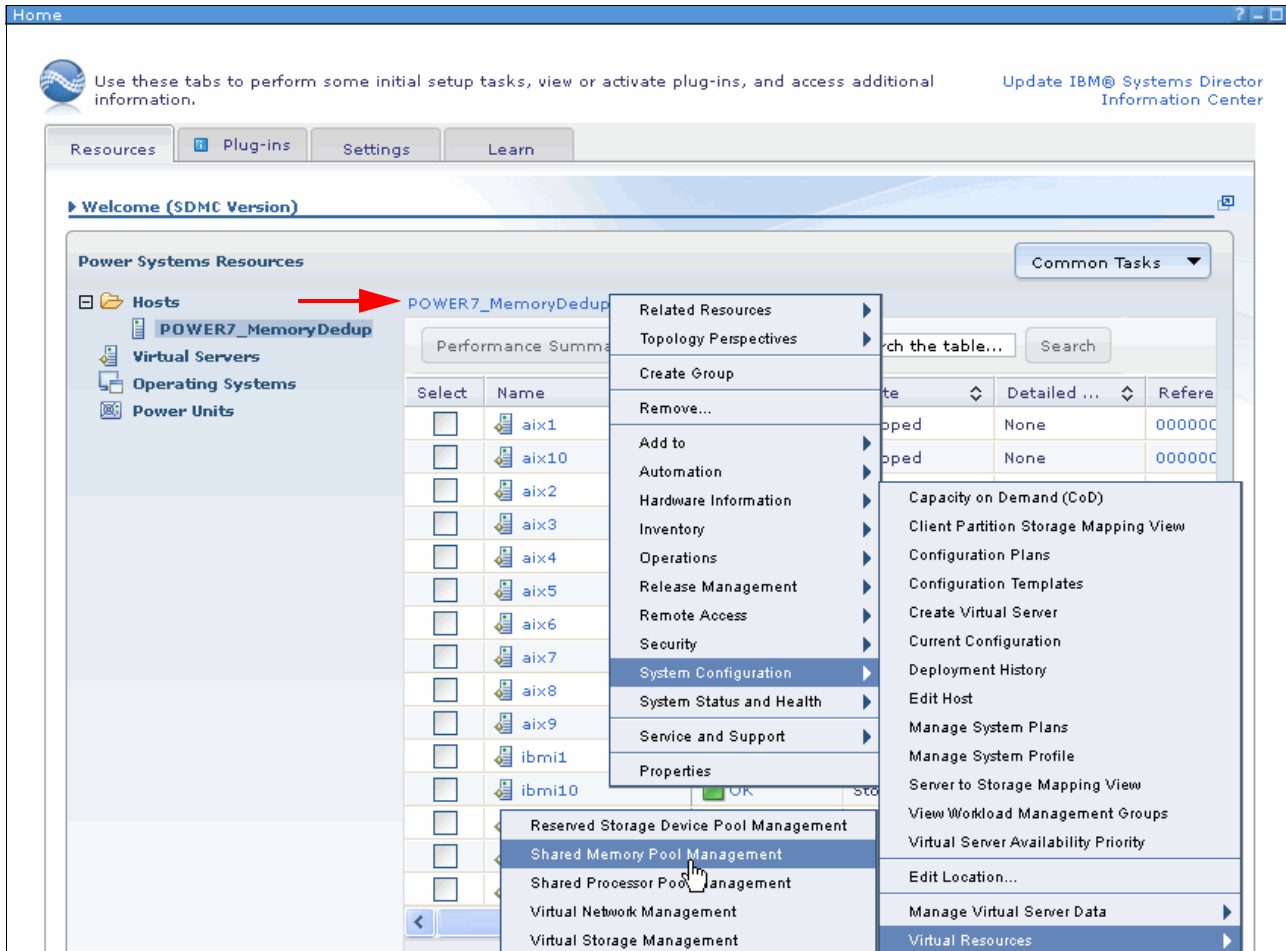


Figure 3-27 Opening the Shared Memory Pool Management window in SDMC

3. In the Shared Memory Pool Management window (Figure 3-28), check the **Enable Active Memory Deduplication** check box. Scroll down the window and click **OK** to confirm the configuration.

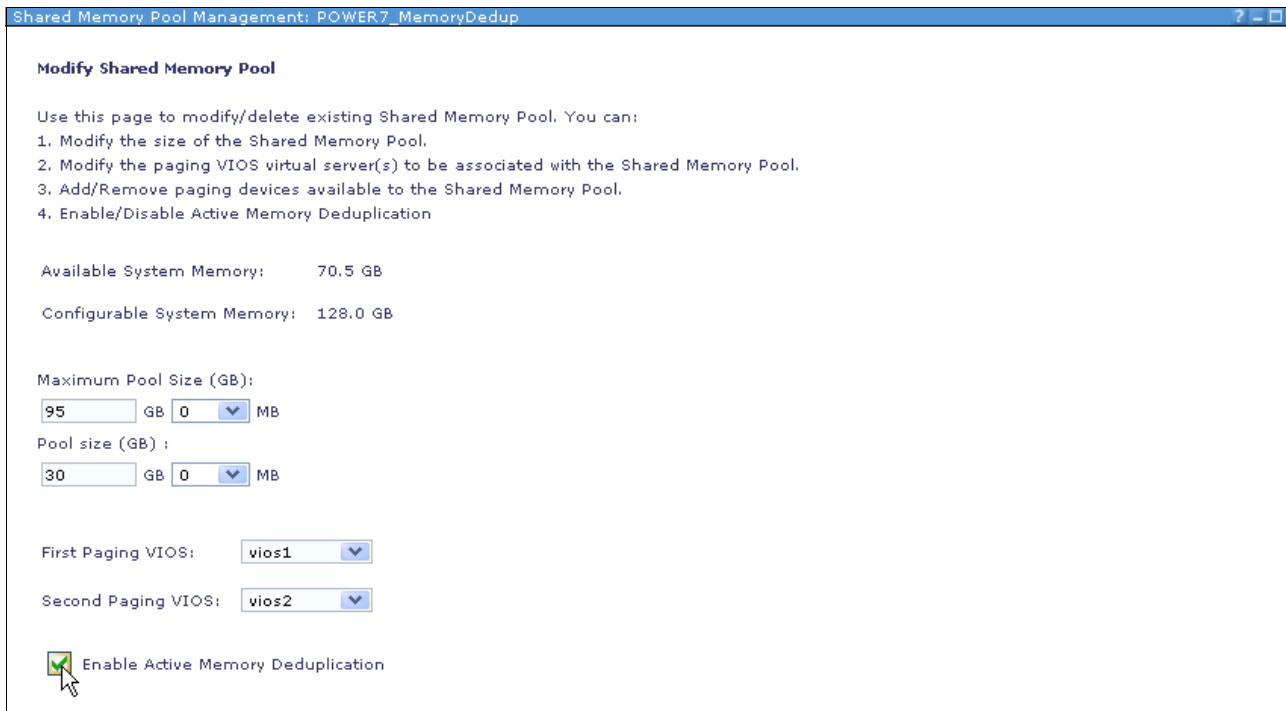


Figure 3-28 Enabling Active Memory Deduplication from the SDMC Shared Pool Management window

3.3.5 Disabling Active Memory Deduplication using the SDMC GUI

To disable Active Memory Deduplication using SDMC, follow the same steps described in section 3.3.4, “Enabling Active Memory Deduplication using the SDMC GUI” on page 40. Open the shared memory pool window (Figure 3-29) and then uncheck the **Enable Active Memory Deduplication** check box. After that, scroll down the window and click the **OK** button to continue, as shown in Figure 3-29.

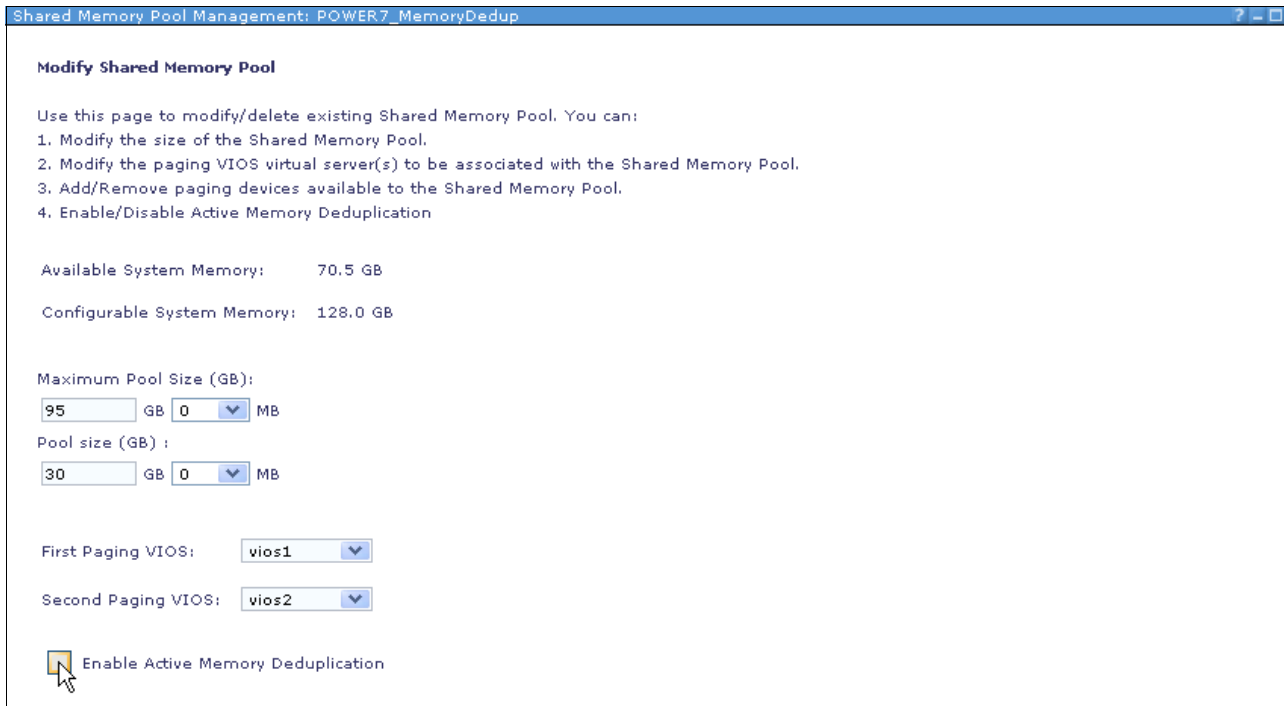


Figure 3-29 Disabling Active Memory Deduplication using the SDMC

3.3.6 Enabling or disabling Active Memory Deduplication using the SDMC CLI

The commands to enable, disable, and view Active Memory Deduplication configuration are the same as those of the CLI. The only difference is that you have to put the `smcli` command before the CLI of the commands.

To enable Active Memory Deduplication, use the `chhwres` command:

```
smcli chhwres -r mempool -m <managed system> -o s -a "mem_dedup=1"
```

To disable Active Memory Deduplication, simply set `mem_dedup` option to 0:

```
smcli chhwres -r mempool -m <managed system> -o s -a "mem_dedup=0"
```

To check if Active Memory Deduplication is enabled or disabled, use the `lshwres` command:

```
smcli lshwres -r mempool -m <managed system> -F mem_dedup
```

If the command returns 1, Active Memory Deduplication is enabled. If the command returns 0, Active Memory Deduplication is disabled. Remember to replace the **managed system** field with the name of your managed system.



Tunable parameters

You can customize the following Active Memory Deduplication values to obtain a better performance in terms of the amount of memory saved with deduplication:

- ▶ The size of the deduplication table
- ▶ The maximum size of the Active Memory Sharing pool

However, another configuration plays an indirect role in Active Memory Deduplication performance. Creating the signatures of physical pages to store in the deduplication table has processor requirements. This processing power comes from processor cycles donated to the hypervisor by the Virtual I/O Servers (VIOs). If the VIOs are not sized appropriately, then these donated processor cycles might not exist or might be insufficient for an efficient memory deduplication process. Therefore, you have one additional user-customized parameter that indirectly relates to Active Memory Deduplication: spare (idle) VIOS processor cycles.

This chapter presents the parameters that can be tuned within Active Memory Deduplication and what effects changes to your environment might cause. After completing this chapter, you should be able to perform the following functions:

- ▶ Identify all of the tunables
- ▶ List the values of each tunable
- ▶ Change each of the tunables

This chapter includes the following sections:

- ▶ Tuning the ratio of the deduplication table
- ▶ Tuning the maximum size of the Active Memory Sharing pool
- ▶ Spare Virtual I/O Server processor cycles

4.1 Tuning the ratio of the deduplication table

The deduplication table ratio defines the size of the deduplication table. The deduplication ratio is applied to the maximum Active Memory Sharing pool size and yields the table size. The size of the deduplication table is determined by the following equation:

$$\text{deduplication table size} = \text{AMS max pool size} \times \text{deduplication table ratio}$$

Varying the ratio, assuming that the maximum Active Memory Sharing pool size is kept constant, gives a smaller or bigger deduplication table. Active Memory Deduplication uses this memory space to store information about which pages are deduplicated and potential deduplication candidate pages.

When the deduplication table is enlarged, Active Memory Deduplication has more memory space to store the information it needs to maintain deduplicated pages. Also, the potential to deduplicate memory pages within the shared memory pool increases. When the deduplication table is made smaller, less memory space is available for the table to store this information, so the potential to deduplicate memory pages decreases.

A first analysis of these statements might suggest that it is always a good idea to have the biggest possible deduplication table size. However, having a large deduplication table requires more memory to store more signatures and other information.

Permissible deduplication table ratios are outlined in this section. Its optimal suggested value is derived from our tests in Chapter 6, “Configuration scenarios” on page 65 and suggested as a best practice in Chapter 7, “Best practices” on page 77.

The current system configuration for the deduplication table ratio can only be obtained from command line in the HMC or the SDMC. Example 4-1 outlines how to get it using the HMC.

Example 4-1 Listing the value of the deduplication table ratio using the HMC

```
hscroot@hmc6:~> lshwres -r mempool -m POWER7_MemoryDedup
curr_pool_mem=81920,curr_avail_pool_mem=79992,curr_max_pool_mem=97280,pend_pool_me
m=81920,pend_avail_pool_mem=79992,pend_max_pool_mem=97280,sys_firmware_pool_mem=33
9,"paging_vios_names=vios1,vios2","paging_vios_ids=1,2",mem_dedup=1,dedup_table_ra
tio=1:1024
```

Using commands in the SDMC: All of the HMC commands outlined in this chapter can be run on the SDMC. Using the `smcli` prefix is optional. For checking the deduplication table ratio on the SDMC, you can issue the following command:

```
smcli lshwres -r mempool -m POWER7_MemoryDedup
```

In Example 4-1, the command `lshwres` is used to query the server `POWER7_MemoryDedup` for its memory pool configuration. The parameter you are interested in is the `dedup_table_ratio`. In this example, the current ratio is 1/1024. Applying this value to the formula introduced earlier, and also applying the value in the example for `curr_max_pool_mem` (measured in MB), you calculate the size of our deduplication table as follows:

$$\text{Deduplication table size} = 97280 * 1/1024 = 95$$

Thus, the size of the deduplication table in this example is 95 MB. This memory is reserved by the hypervisor for Active Memory Deduplication use only and is taken from the shared memory pool. Users planning to use Active Memory Deduplication must take this value into consideration when implementing this technology onto their systems.

To verify the supported deduplication table ratios for your server, you need to execute a similar command on the HMC command line, as shown in Example 4-2.

Example 4-2 Listing the supported deduplication table ratios

```
hscroot@hmc6:~> lshwres -r mem -m POWER7_MemoryDedup --level sys
configurable_sys_mem=131072,curr_avail_sys_mem=25600,pend_avail_sys_mem=25600,inst
alled_sys_mem=131072,max_capacity_sys_mem=deprecated,deconfig_sys_mem=0,sys_firmwa
re_mem=7168,mem_region_size=256,configurable_num_sys_huge_pages=0,curr_avail_num_s
ys_huge_pages=0,pend_avail_num_sys_huge_pages=0,max_num_sys_huge_pages=8,requested
_num_sys_huge_pages=0,huge_page_size=16384,total_sys_bsr_arrays=0,bsr_array_size=4
096,curr_avail_sys_bsr_arrays=0,max_mem_pools=1,max_paging_vios_per_mem_pool=2,def
ault_hpt_ratios=1:64,"possible_hpt_ratios=1:32,1:64,1:128,1:256,1:512",default_ded
up_table_ratio=1:1024,"possible_dedup_table_ratios=1:256,1:512,1:1024,1:2048,1:409
6,1:8192"
```

In this example, the command `lshwres` is used to query server `POWER7_MemoryDedup` for its memory configuration. The parameter of importance is the `possible_dedup_table_ratios`. This parameter outlines the possible ratios that you can use to tune your system to achieve a better Active Memory Deduplication performance.

The deduplication table ratios you can use, from the smallest to the biggest, are as follows:

- ▶ 1 / 8192
- ▶ 1 / 4096
- ▶ 1 / 2048
- ▶ 1 / 1024 (default factory value)
- ▶ 1 / 512
- ▶ 1 / 256

As of today (firmware family 740) these ratios are the ones available, and they are the same no matter what POWER7 model with which you use Active Memory Deduplication.

In order to change the deduplication table, you use the command `chhwres`, as shown in Example 4-3. This command is run on the HMC command line.

Ratio change dynamic: Changing the value of the deduplication table ratio is a dynamic operation.

Example 4-3 Changing the value of the deduplication table ratio

```
hscroot@hmc6:~> chhwres -r mempool -m POWER7_MemoryDedup -o s -a
"dedup_table_ratio=1:256"
```

In this example, you change the deduplication table ratio by setting the `dedup_table_ratio` parameter to 1/256. Running the command produces no output. It is a good idea to check and see whether the changes have been committed using the `lshwres` command, as shown in Example 4-1 on page 44.

Important: Whenever a deduplication table ratio value changes, all of the deduplicated memory pages are broken out into separate unique physical pages, and another deduplication table is created. The process of deduplicating memory pages then is restarted.

4.2 Tuning the maximum size of the Active Memory Sharing pool

The Active Memory Sharing pool size is defined by two parameters:

- ▶ Pool size
- ▶ Maximum pool size

The maximum pool size is a soft limit for the pool size and can be changed dynamically. Maximum pool size affects Active Memory Deduplication as it is used in the computation of the size of the deduplication table:

$$\text{deduplication table size} = \text{AMS max pool size} \times \text{deduplication table ratio}$$

A change in the value of the maximum Active Memory Sharing pool size changes the size of the deduplication table as well. An increase in the maximum pool size increases the deduplication table size. A decrease in the maximum pool size decreases the deduplication table size. The implications these actions have for Active Memory Deduplication are the same as discussed in 4.1, “Tuning the ratio of the deduplication table” on page 44.

To verify the current maximum size for the Active Memory Sharing pool, you can use the HMC command line, as shown in Example 4-1 on page 44, and look for `curr_max_pool_mem`. You can also verify the current maximum size for the Active Memory Sharing pool from the HMC GUI, as follows:

1. In the left pane, expand **Servers**, and select one of the listed servers.
2. In the lower right pane, click **Shared Memory Pool Management**, as shown in Figure 4-1, to open the Active Memory Sharing pool configuration panel. Make sure that you open the panel for the correct server.

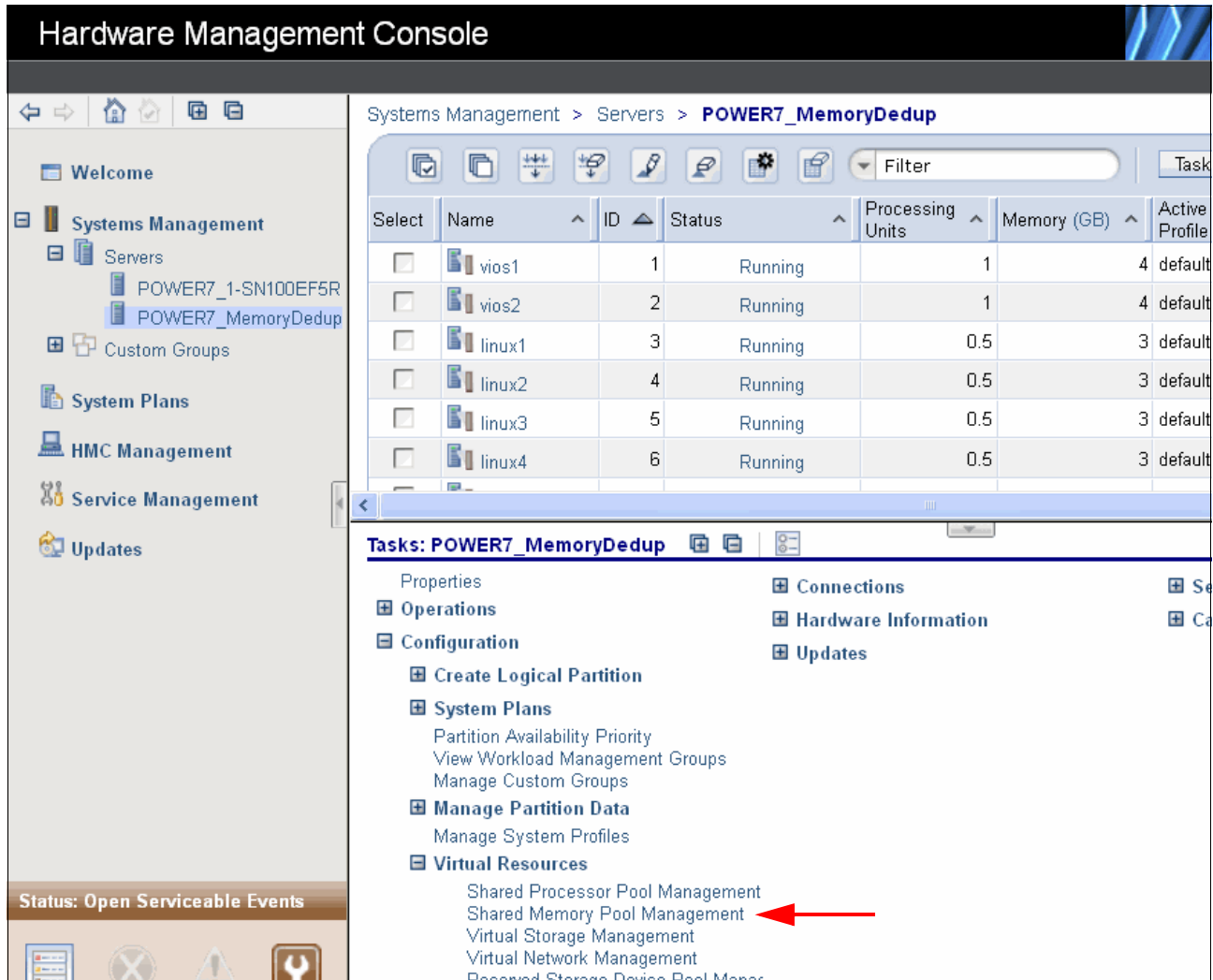


Figure 4-1 Selecting the Shared Memory Pool Management link in the HMC

- In the Pool Properties configuration panel for the Active Memory Sharing pool (Figure 4-2), you can change the Active Memory Sharing maximum pool size. Input a new value and click **OK**. This operation takes effect immediately and affects the size of the deduplication table.

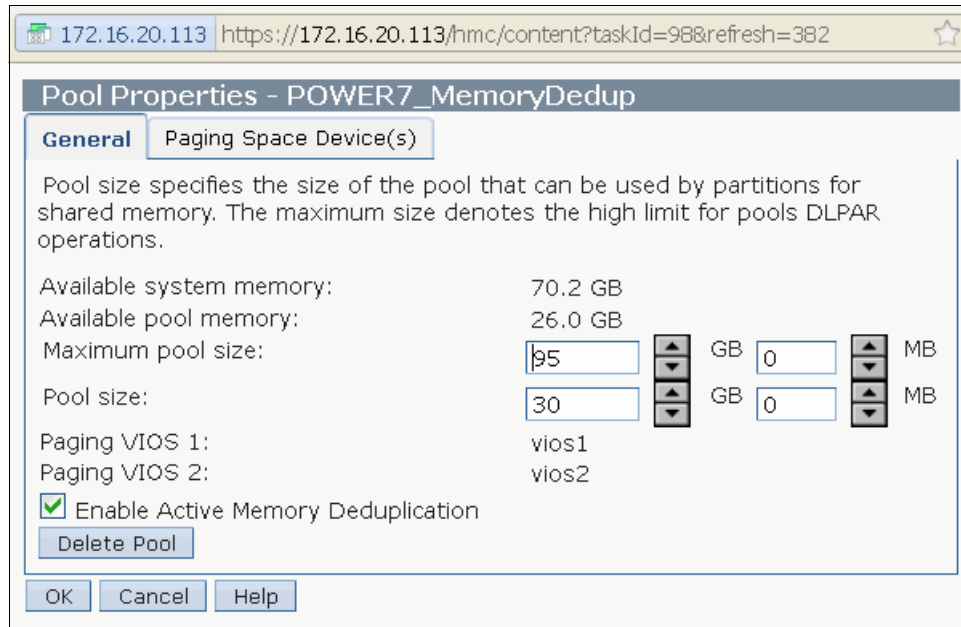


Figure 4-2 Active Memory Sharing pool configuration panel

Important: Whenever the value of the Active Memory Sharing maximum pool size changes, all of the deduplicated memory pages are broken out into separate unique physical pages, and another deduplication table is created. The process of deduplicating memory pages then restarts.

Changing pool size to change deduplication table size: The authors believe that changing the size of the maximum Active Memory Sharing pool size to alter the size of the deduplication table is not the best path to follow. Changing the maximum Active Memory Sharing pool size also changes the amount of memory required by the hypervisor for Active Memory Sharing page tracking. For every 16 GB chunk of pool maximum memory, the hypervisor reserves 256 MB for that purpose. If you want to change the size of the deduplication table, consider changing the deduplication table ratio first.

4.3 Spare Virtual I/O Server processor cycles

Although the number of spare VIOS cycles is not an Active Memory Deduplication tunable and cannot be directly configured, it plays an important role in deduplicating memory pages. As explained in the section “Processor cycles” on page 13, the memory deduplication work carried out by the hypervisor requires processor cycles. These cycles are donated to the hypervisor by the VIOSs associated with the Active Memory Sharing pool.

If your VIOSs are sized tightly and you activate Active Memory Deduplication, memory pages deduplication occurs more slowly. This slowdown happens due to the lack of processor cycles needed to compute the work for deduplicating memory pages. Therefore, you must ensure that the VIOSs are able to donate some of their processor cycles by sizing their processing requirements with an appropriate slack.

The greater the slack in processing power your VIOSs have, the more processor cycles can be used for memory deduplication. However, as demonstrated in section 6.7, “Virtual I/O Server processor sizing” on page 75, this extra processing power is close to zero. You can adjust the VIOSs processing power slack by either of two methods:

- ▶ Method 1: Increasing the amount of its desired processing units to ensure that there is slack compared to the usual processor workload of the VIOS
- ▶ Method 2: Making the Virtual I/O Server capable of creating this slack by borrowing extra resources from the processor shared pool

If you want to ensure that this processor slack always exists, then choose the first method, whether you make your VIOS capped or uncapped. The second method assumes an uncapped processor configuration with a slack between the desired processing units and the desired amount of virtual processors. However, these extra cycles to be borrowed from the processor pool might not always be available.

You can use the SDMC or the HMC to edit the VIOS processor configuration. In this case, you use the HMC, as shown in Figure 4-3.

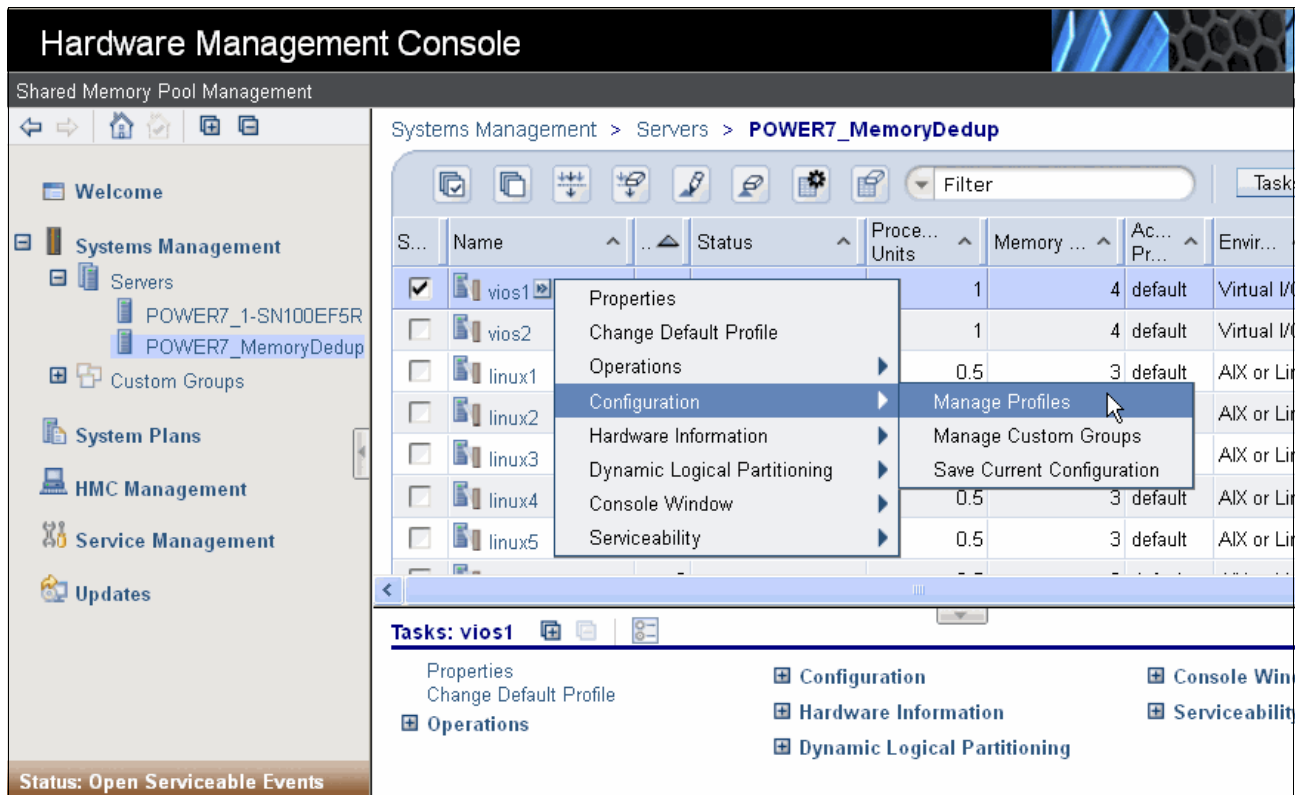


Figure 4-3 Opening the VIOS profile in the HMC

Figure 4-4 shows where to adjust the settings for both methods 1 and 2. Remember that if you proceed with method 2, the slack in VIOS processing comes from the difference in the settings of **Desired processing units** and **Desired virtual processors** fields. Also, for method 2, select the field **Uncapped** check box and set the **Weight** box to a high value.¹

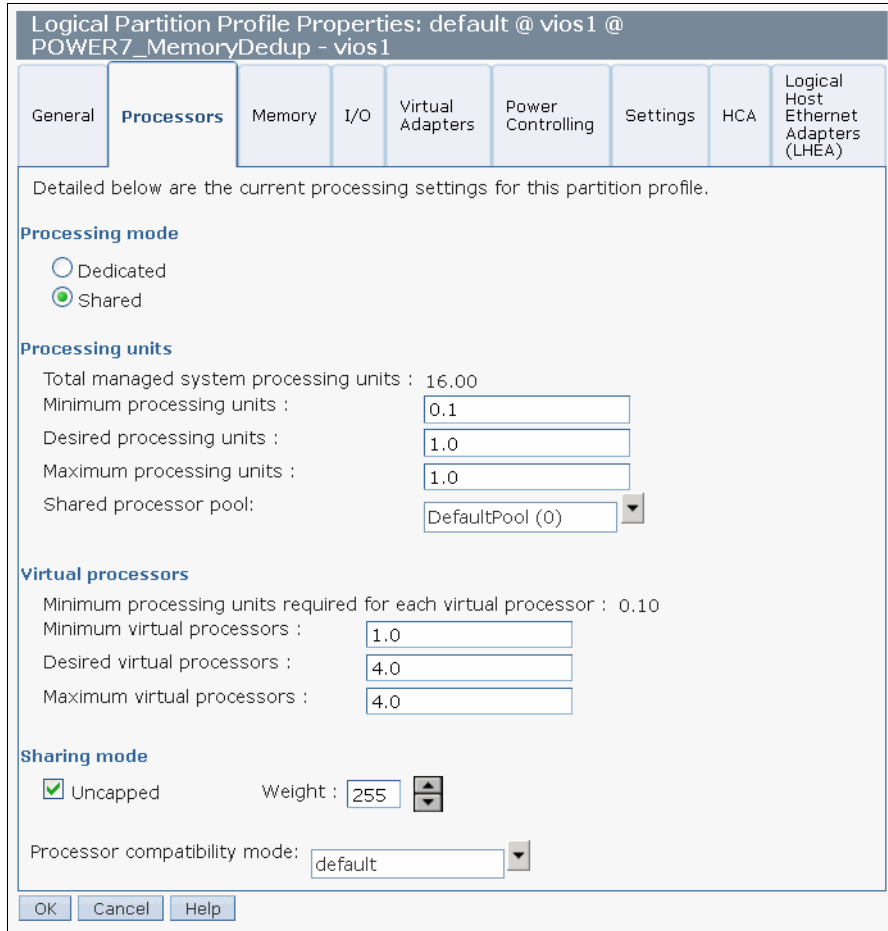


Figure 4-4 VIOS processor settings

Dynamic logical partitioning: The changes to the VIOS processor configuration can also be made through a dynamic logical partitioning operation to prevent bringing the VIOS down for the changes to take effect. See *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590 for details.

¹ Suggested weight value for VIOS is 255.



Memory deduplication monitoring

This chapter explains how to monitor memory resources in a virtualized IBM Power server with Active Memory Deduplication enabled. After reading the chapter, you should be able to perform the following tasks:

- ▶ Identify the statistics available for measuring Active Memory Deduplication activity and performance
- ▶ Understand which tools and commands are available to list Active Memory Deduplication statistics

This chapter thus includes the following sections:

- ▶ Statistics
- ▶ Monitoring tools

The sections are organized by operating system type: AIX, IBM i, and Linux.

5.1 Statistics

Memory page coalescing is a transparent operation in which the hypervisor detects duplicate pages and directs all the user read pages to a single copy. The hypervisor then reclaims the other duplicate physical memory pages.

Memory page coalescing efficiency is measured in physical page savings as opposed to coalescing processor requirements. The overall page savings depend on the workloads throughout the logical partitions (LPARs). The more similar they are, the greater the savings.

The metrics related to memory deduplication can be seen in the Systems Director Management Console (SDMC) through the Hardware Management Console (HMC). The metrics are also visible using performance tools provided by the operating systems. The following metrics can be monitored for Active Memory Deduplication:

- ▶ Pool coalesced memory
- ▶ LPAR coalesced memory
- ▶ Virtual I/O Server (VIOS) processor cycles uses for coalescing

You can use these metrics to properly size your VIOS processor requirements and your shared memory pool. The deduplication table size can also be adjusted. See Chapter 4, “Tunable parameters” on page 43 for details about how to tune Active Memory Deduplication settings.

Pool coalesced memory

Pool coalesced memory refers to the total amount of physical memory coalesced among all of the LPARs during the Active Memory Deduplication activity. This global metric is shared with all of the LPARs that are authorized to access pool-wide statistics. This value represents the current snapshot for pool-wide coalesced pages.

The pool coalesced memory counter increases once for each deduplicated page, no matter how many virtual pages among all LPARs reference it. An example of how the pool coalesced memory counter works is shown in Figure 5-1 on page 53.

Logical partition coalesced memory

Logical partition coalesced memory indicates the amount of memory assigned to the LPAR coalesced with other identical memory pages either from within the LPAR or from another LPAR. This value is referred to in this paper as *coalesced memory*.

Memory coalescing takes place even when a single LPAR is running. No fixed relationship exists between this metric and the overall pool-wide coalesced memory. The sum of coalesced bytes within all individual LPARs does not match the pool coalesced bytes. Furthermore, the LPAR coalesced memory might be higher than the pool coalesced memory.

The coalesced bytes counter increases for every page that has been coalesced with any other identical pages in memory. Given that, zeroed pages are taken into account for memory coalescing. Zeroed pages are pages of available memory pages kept by the operating system for quick allocation. They are zeroed to ensure that data is not readable by the next process they are assigned to.

You might think that memory-idle LPARs (a high memory read-to-write ratio) cause massive memory deduplication, but this is not always the case. Memory-idle LPARs might not result in a high amount of coalesced pages because the unused pages might not necessarily have been zeroed by the operating system.

Figure 5-1 shows an example of pool coalesced count and LPAR coalesced count.

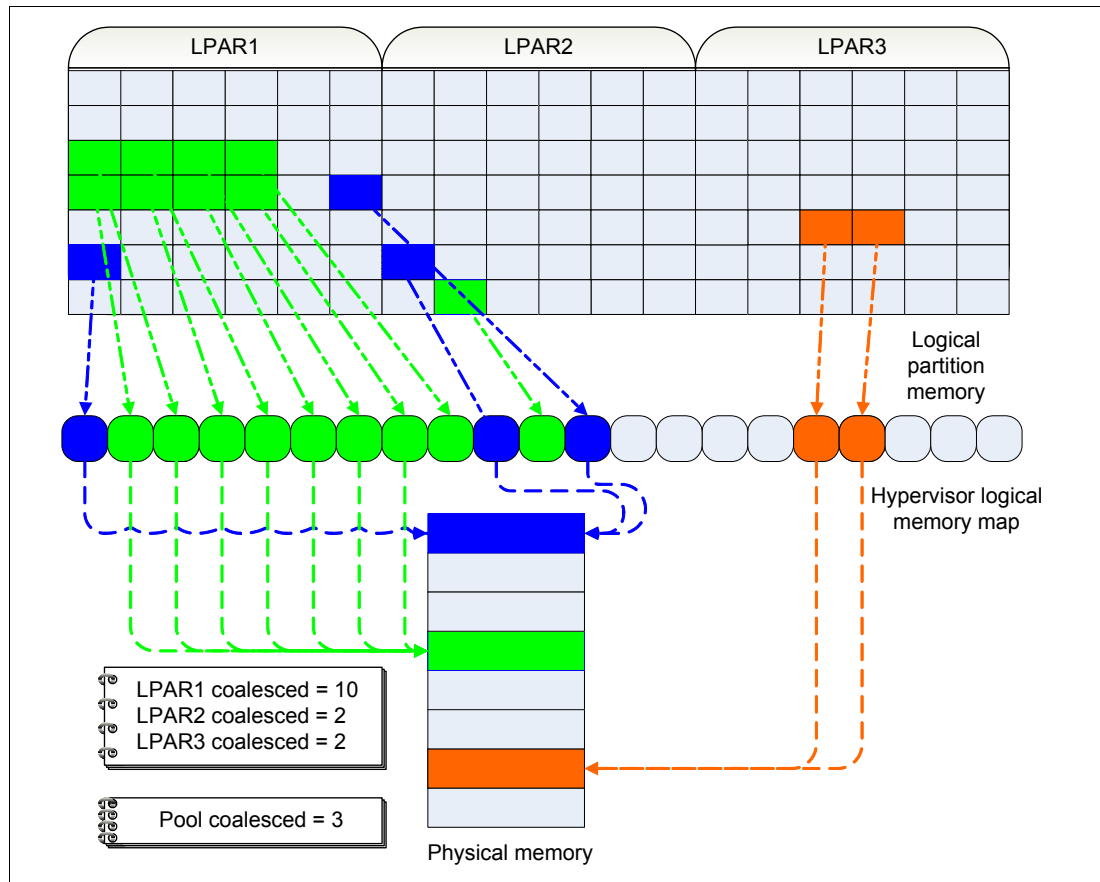


Figure 5-1 Memory coalescing counters

Note how the counters work according to this figure. Three LPARs and three memory pages are duplicated in the system: the red, green, and blue pages. Thus, these pages get deduplicated and stored only once in main memory, as depicted in the physical memory representation of Figure 5-1.

The pool coalesced memory counter increases by three pages, whereas the LPAR 1 coalesced memory counter increases by 10 pages (eight green pages plus two blue pages). LPAR 2 coalesced memory counter increases by two pages (one blue page plus one green page). Finally, the coalesced memory counter for LPAR 3 increases by two pages (two orange pages).

To estimate the amount of memory savings due to deduplication, you can use the following formula:

$$\text{Total Memory Savings} = (\Sigma \text{Coalesced}) - \text{Pool Coalesced}$$

The terms of the equation are defined as follows:

- ▶ Total Memory Savings is the estimated memory savings for the system.
- ▶ Coalesced is the amount of coalesced memory within a LPAR.
- ▶ Pool Coalesced is the amount of pool-wide coalesced memory.

The estimated memory savings for the example in Figure 5-1 on page 53 is calculated as follows:

$$\text{Total Memory Savings} = (10 + 2 + 2) - 3 = 11$$

In this case, you save an amount of physical memory equivalent to 11 memory pages, which correspond to 44 KB of memory freed back to the shared memory pool.

VIOS processor cycles spent on coalescing

On a regular basis, the number of VIOS processor cycles used for page coalescing is calculated and compared to the time base. Comparisons that yield percentage deltas close to zero mean that the processor cycles used for page coalescing are insignificant. This is a global metric that is shared with all of the LPARs that are authorized to access pool-wide statistics.

5.2 Monitoring tools

This section shows the tools used to monitor memory statistics relevant to an Active Memory Deduplication environment. Statistics can be retrieved from the SDMC or HMC and also directly from the operating systems running in the LPARs. The HMC retrieval method is not shown here as the output statistics are similar to the SDMC.

To monitor pool-wide statistics from within the operating system, you must enable the LPARs to collect performance information, as shown in Figure 5-2.

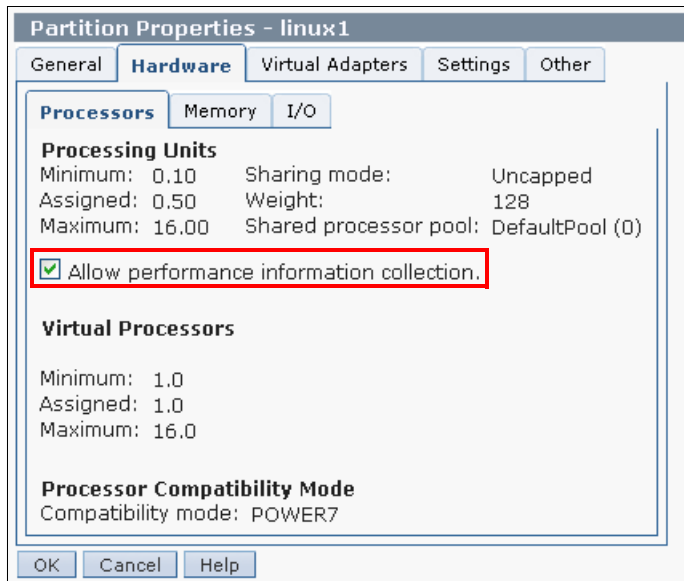


Figure 5-2 Enabling the LPARs to collect performance statistics

Standard memory monitoring tools, such as **svmon** and **vmstat**, can also be used to collect additional information about memory usage. However, these tools do not show the metrics described in section 5.1, "Statistics" on page 52. To monitor memory deduplication metrics on each operating system type, see the following sections in this chapter.

5.2.1 Monitoring in AIX

Monitoring coalesced memory in an AIX partition is simple. The `lparstat` command has been enhanced to display the amount of coalesced pages and also the VIOS processor cycles used for page coalescing. These metrics are shown in Example 5-1.

Example 5-1 Monitoring memory coalescing in AIX with lparstat

```
# lparstat -mpw 1
System configuration: lcpu=4 mem=3072MB mpsz=40.00GB iome=111.00MB iomp=10 ent=0.50
```

physb	hpi	hpit	pmem	iomin	iomu	iomf	iohwm	iomaf	pgcol	mpgcol	ccol	%entc	vcswh
99.42	0	0	1.10	48.2	12.2	50.8	14.5	0	395.2	517.1	0.0	199.8	574
99.45	0	0	1.10	48.2	12.2	50.8	14.5	0	395.2	517.2	0.0	199.8	592
99.25	0	0	1.10	48.2	12.2	50.8	14.5	0	395.2	517.3	0.0	199.5	538
99.36	0	0	1.10	48.2	12.2	50.8	14.5	0	395.1	517.4	0.0	199.7	510
99.05	0	0	1.10	48.2	12.2	50.8	14.5	0	395.2	517.5	0.0	199.7	625
99.07	0	0	1.10	48.2	12.2	50.8	14.5	0	395.2	517.6	0.0	199.7	540
99.33	0	0	1.10	48.2	12.2	50.8	14.5	0	395.2	517.6	0.0	199.6	537
99.05	0	0	1.10	48.2	12.2	50.8	14.5	0	395.2	517.8	0.0	199.7	640
99.16	0	0	1.10	48.2	12.2	50.8	14.5	0	395.3	517.8	0.0	199.2	547

In this report, you see the Active Memory Deduplication statistics using the `lparstat` command.

pmem	Indicates the physical memory, in GB, that is currently allocated to the LPAR by the hypervisor.
pgcol	Indicates the amount of LPAR memory, in MB, that is currently coalesced due to Active Memory Deduplication activity.
mpgcol	Indicates the amount of coalesced memory, in MB, in the entire shared memory pool due to Active Memory Deduplication activity. If the partition is not authorized to access pool-wide statistics, this metric displays zero.
ccol	Indicates the amount of processor power, measured in number of processing units, used for coalescing pages during Active Memory Deduplication activity. If the partition is not authorized to access pool-wide statistics, this metric displays zero.

For more details about memory monitoring in AIX, see *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590.

5.2.2 Monitoring in IBM i

At the time of writing this paper, monitoring the amount of memory coalesced on IBM i LPARs is not possible.

To compensate, you can use the SDMC to monitor the pool coalesced memory and also the total of memory coalesced memory by all the LPARs, as shown in 5.2.4, “Monitoring through the Systems Director Management Console” on page 58.

5.2.3 Monitoring in Linux

In Linux, memory deduplication statistics are shown by `amsstat` command, which is provided by the `powerpc-utils` package. RedHat Enterprise Linux and SuSE Linux Enterprise Server include the `powerpc-utils` package within their Power server releases.

The `amsstat` command captures memory statistics relevant to an Active Memory Sharing environment. You can run this command once or set it up to run repeatedly for a specified number of times with a particular interval, in seconds, between consecutive runs.

The standard output for `amsstat` is shown in Example 5-2.

Example 5-2 Monitoring memory coalescing in Linux with amsstat

```
# amsstat
Tue Nov 22 12:00:17 EST 2011
System Memory Statistics:
    MemTotal:                3087488 kB
    MemFree:                  2623680 kB
    Buffers:                   106816 kB
    Cached:                    156416 kB
    Inactive:                   79680 kB
    Inactive(anon):            35392 kB
    Inactive(file):            44288 kB
    SwapTotal:                 3276672 kB
    SwapFree:                  3206848 kB
    DesMem:                    3072 MB
Entitlement Information:
    entitled_memory:           116391936
    mapped_entitled_memory:    2924544
    entitled_memory_group_number: 32772
    entitled_memory_pool_number: 0
    entitled_memory_weight:    0
    entitled_memory_pool_size: 42949672960
    entitled_memory_loan_request: -210063360
backing_memory:             2312380416
coalesced_bytes:           183795712
pool_coalesced_bytes:     588681216
    cmo_enabled:                1
    cmo_faults:                 27041
    cmo_fault_time_usec:       175967701
    cmo_primary_psp:            1
    cmo_secondary_psp:         2
CMM Statistics:
    disable:                    0
    debug:                      0
    min_mem_mb:                 256
    oom_kb:                     1024
    hotplug_delay:              5
    delay:                      1
    loaned_kb:                  0
    loaned_target_kb:          0
    oom_freed_kb:              0
VIO Bus Statistics:
    cmo_entitled:               116391936
    cmo_reserve_size:           12183552
    cmo_excess_size:            104208384
    cmo_excess_free:            104208384
    cmo_spare:                  1562624
    cmo_min:                    7813120
    cmo_desired:                12183552
    cmo_curr:                   3346432
    cmo_high:                   11968512
VIO Device Statistics:
1-lan@3000000b:
    cmo_desired:                4243456
    cmo_entitled:              4243456
```

```

cmm_allocated:                2928640
cmm_allocs_failed:            0
v-scsi@30000191:
cmm_desired:                  2125824
cmm_entitled:                 2125824
cmm_allocated:                139264
cmm_allocs_failed:            0
v-scsi@30000192:
cmm_desired:                  2125824
cmm_entitled:                 2125824
cmm_allocated:                139264
cmm_allocs_failed:            0
v-scsi@3000019a:
cmm_desired:                  2125824
cmm_entitled:                 2125824
cmm_allocated:                139264
cmm_allocs_failed:            0

```

In this report, you see the following Active Memory Deduplication statistics:

- backing_memory** Amount of physical memory, in bytes, that is currently assigned to the partition. This value changes over time based on the load of all of the partitions in the shared memory pool. Using more memory than this amount incurs performance penalties, such as page loaning, and paging starts to occur.
- coalesced_bytes** Number of bytes assigned to the LPAR which have been coalesced with other identical pages either from within the LPAR, or from another LPAR.
- pool_coalesced_bytes** Number of bytes in the system shared memory pool which have been coalesced with identical pages.

At the time of writing this paper, no option exists in the **amsstat** command to monitor VIOS processor cycles used for memory page coalescing.

In an Active Memory Sharing environment, the amount of physical memory assigned to a LPAR changes as memory is loaned to other partitions by the Cooperative Memory Manager (CMM) module.

The section called System Memory Statistics in the **amsstat** report shows how the total amount of memory provided to the partition is being used. The reported values are taken from the `/proc/meminfo` pseudo-file.

You can also monitor other statistics relevant to Active Memory Sharing in the **amsstat** report. The fields in the section called Entitlement Information are specific to an Active Memory Sharing environment and are taken from the `/proc/ppc64/lparcfg` file.

Refer to the **amsstat** man page for a more detailed explanation of the metrics available.

5.2.4 Monitoring through the Systems Director Management Console

The SDMC can be used to measure system statistics through its GUI. Although the SDMC GUI shows information about Active Memory Deduplication memory-related statistics, the number of processor cycles used by Active Memory Deduplication cannot be shown using the GUI as of the time of writing this paper.

Enabling statistics collection in the SDMC

To start collecting certain basic statistics on the shared memory pool utilization, you need to turn on utilization data collection:

1. Log on to your SDMC.
2. In the Home panel of the SDMC, select your POWER server and click the **Actions** button, as shown in Figure 5-3.

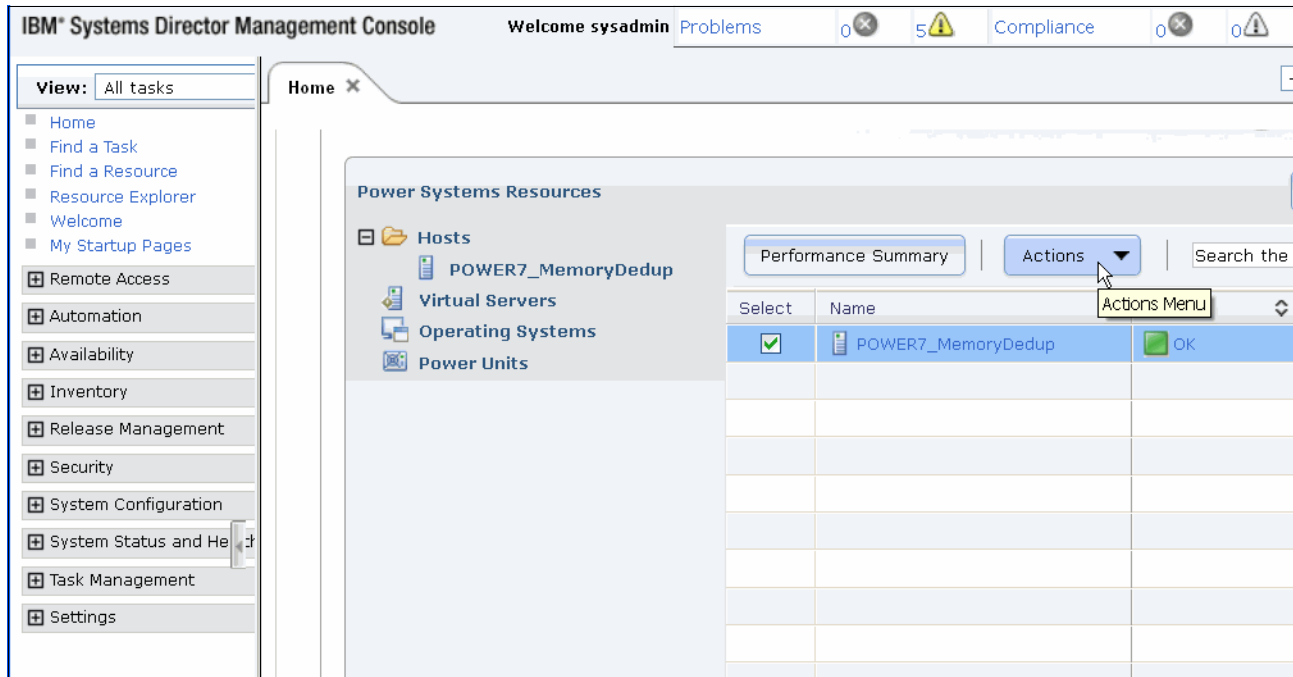


Figure 5-3 Selecting Actions for the POWER server in SDMC

- From **Actions**, select **Operations** → **Utilization data** → **Change Sampling Rate**, as shown in Figure 5-4.

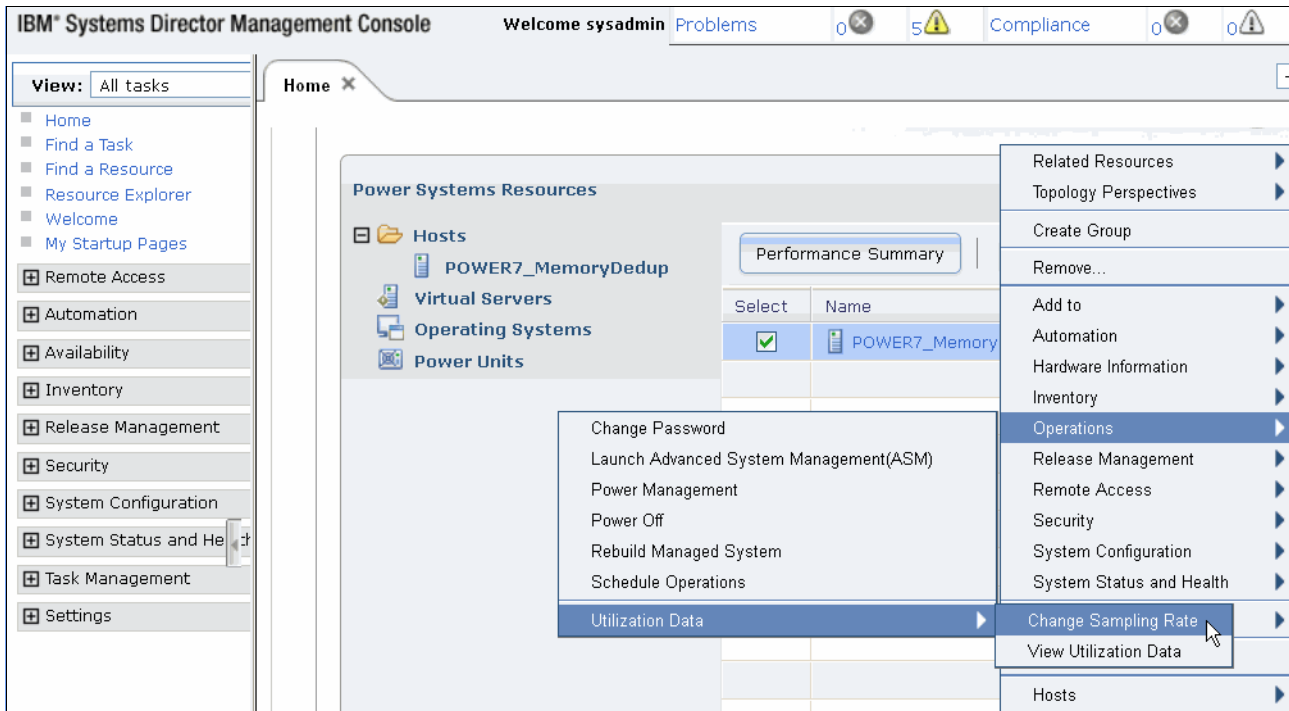


Figure 5-4 Selecting Sampling Rate in SDMC

- On the **Change Sampling Rate** tab (Figure 4-5), change the sampling rate to the desired value. Click **OK**.

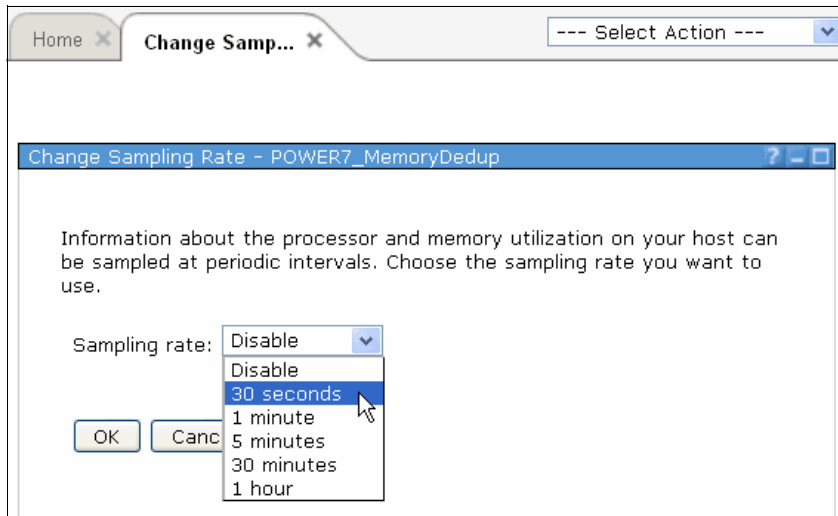


Figure 5-5 Changing the sampling rate of Power server in SDMC

Viewing SDMC collected statistics on shared memory metrics

After you setting up statistics collection, you can start viewing the results:

1. Open the menu the same way you did in Figure 5-4 on page 60, and choose **View Utilization Data**.
2. On the **View Utilization** tab (Figure 5-6), select a time interval for which you want the statistics to be retrieved. Click **Show**, and the SDMC gathers the utilization data within the selected intervals.

The screenshot shows a web browser window with the title 'View Utilization Data'. The main content area is titled 'View Events - POWER7_MemoryDedup'. It features four radio buttons for selecting a time interval: 'Snapshot' (selected), 'Hourly', 'Daily', and 'Monthly'. Below these are two rows of input fields for 'Beginning:' and 'Ending:'. Each row includes a 'Time' field (set to '3:00:56 PM'), an 'Example' field (set to '(12:30:00 PM)'), a 'Date' field (set to 'Nov 28, 2011' for beginning and 'Nov 29, 2011' for ending), and an 'Example' field (set to 'Jan 31, 2000'). An 'Event type' dropdown menu is set to 'All'. At the bottom, there are 'Show' and 'Close' buttons, with a mouse cursor pointing to the 'Show' button.

Figure 5-6 Selecting an interval for the collected statistics

3. Select a measurement point in time for which you want to view the data, and click the **Show Detail** button, as shown in Figure 5-7. You can select an older point in time by clicking the **Next** button until you find the point in which you are interested, and then show its details.

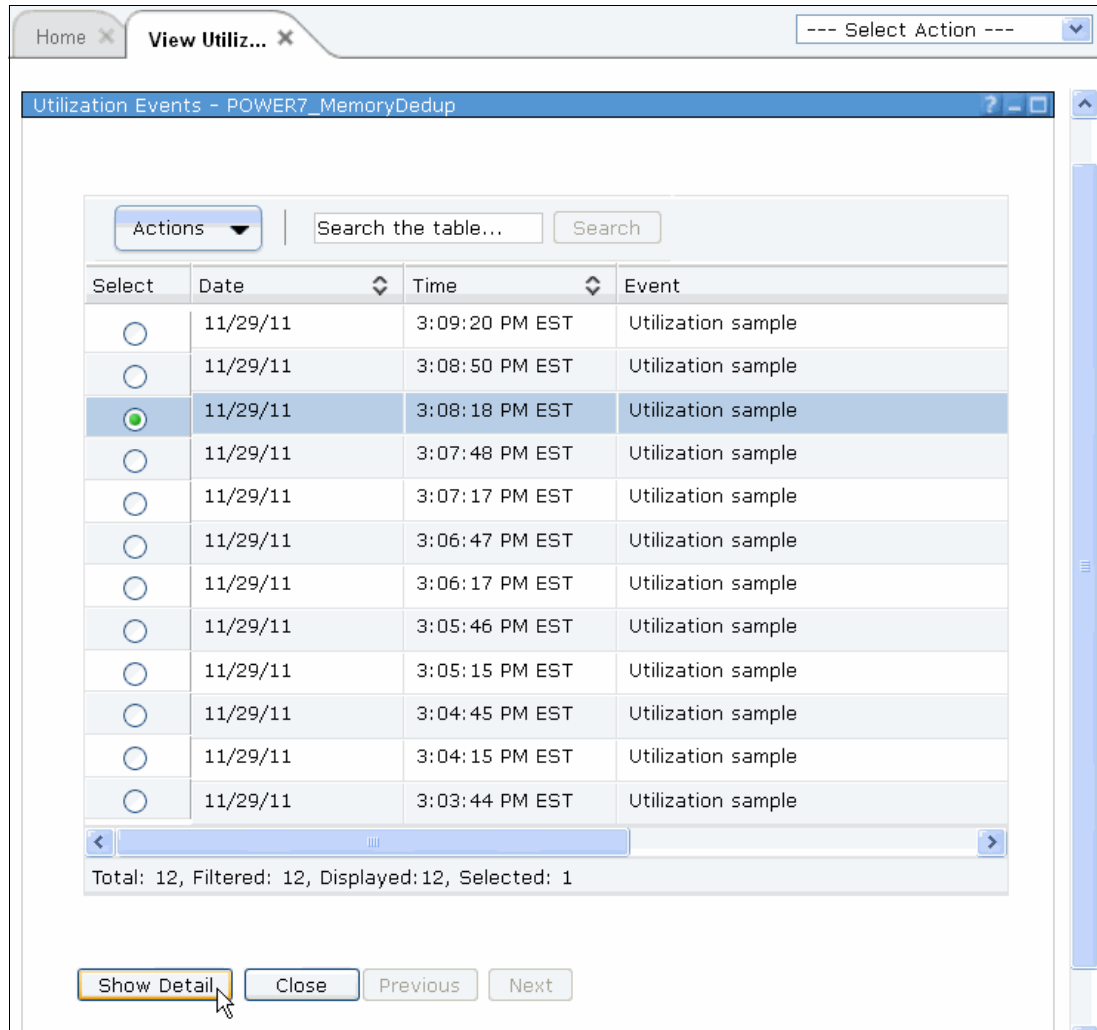


Figure 5-7 Selecting a measurement point in time

4. Change the view to display the Shared Memory Pool statistics, as shown in Figure 5-8.

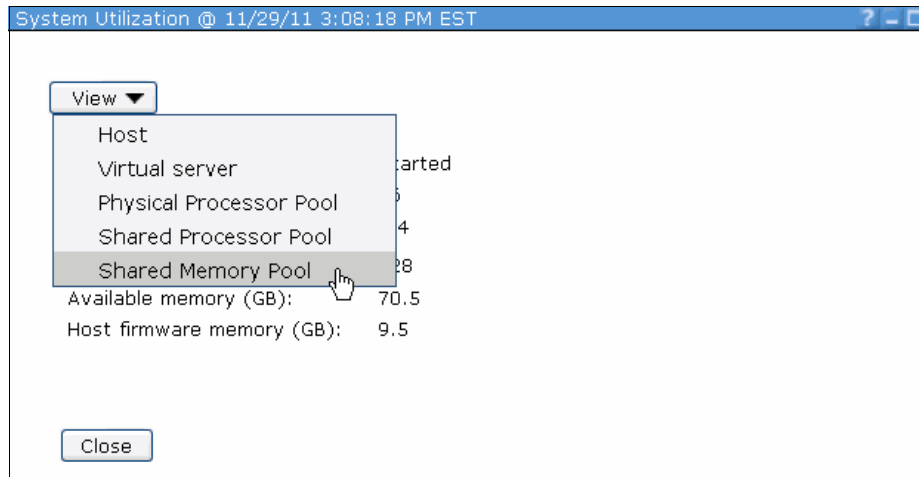


Figure 5-8 Selecting statistics for the shared memory pool

You can now see the values related to Active Memory Deduplication for that point in time, as shown in Figure 5-9.

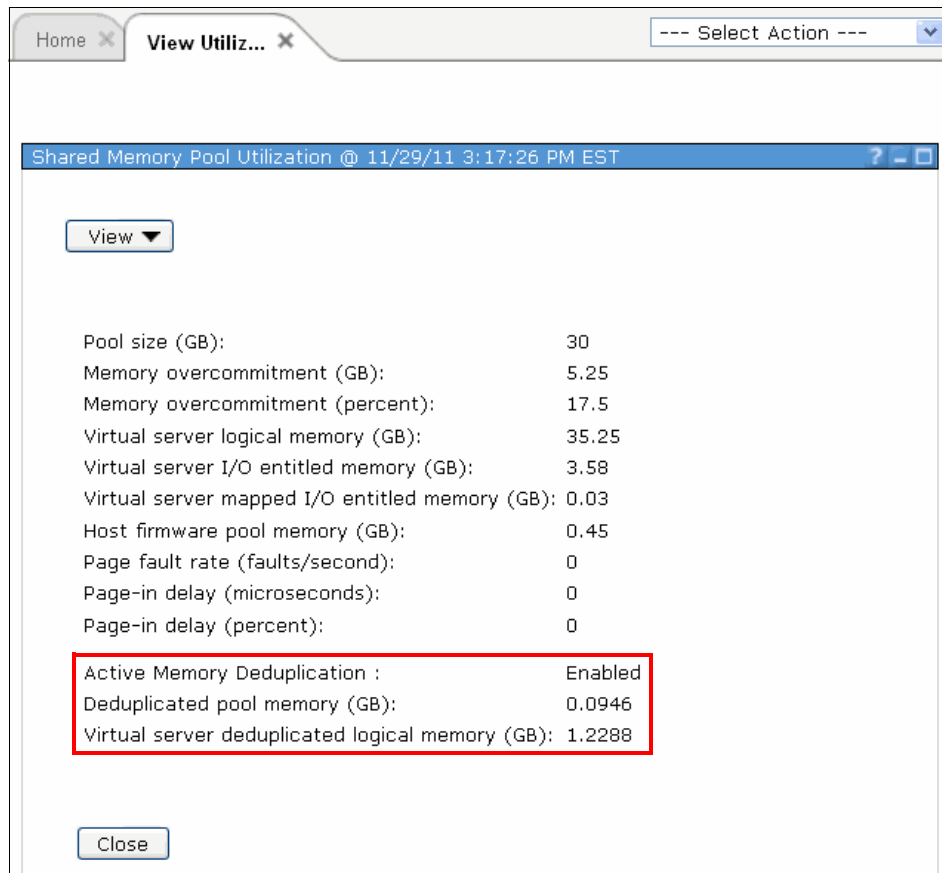


Figure 5-9 Statistics for Active Memory Deduplication

The following metrics are reported by this process:

Active Memory Deduplication status	Enabled or disabled.
Deduplicated pool memory	Amount of deduplicated memory in the shared pool.
Virtual server deduplicated logical memory	Number of LPARs of virtual memory that have been deduplicated.

You can also retrieve these metrics by using the **lparstat** command on a command line as shown in Example 5-3. They are represented by the parameters `mem_dedup`, `dedup_pool_mem`, and `lpar_dedup_mem`.

It is possible to measure the overall number of processor cycles used for memory deduplication since the time it was activated on the system. This metric is measured in processor cycles. To convert it manually to a useful value requires that you sample the cycle count periodically and look at the relative deltas. It is suggested that you use your own operating system method for measuring this value, as explained in the previous sections.

However, if you want to gather this data, use the **lslparutil** command, as shown in Example 5-3. The number of cycles used for deduplication is represented by the parameter `dedup_cycles`, which is the number of cycles used since Active Memory Deduplication was activated on the system.

Example 5-3 Retrieving Active Memory Sharing statistics on processor consumption

```
sysadmin@sdmc2:~> smcli lslparutil -r mempool -m POWER7_MemoryDedup
time=11/23/2011
12:07:03,event_type=sample,resource_type=mempool,sys_time=11/23/2011
12:11:16,curr_pool_mem=30720,lpar_curr_io_entitled_mem=2665,lpar_mapped_io_entitled_mem=37,lpar_run_mem=33792,sys_firmware_pool_mem=406,page_faults=178948,page_in_delay=26769514468,mem_dedup=1,dedup_pool_mem=0.0091,lpar_dedup_mem=0.0544,dedup_cycles=31838208792
```

HMC command option: The **lslparutil** command can also be run directly from the HMC command-line interface.



Configuration scenarios

This chapter presents different configuration scenarios for the Active Memory Deduplication tuning and how this tuning affects saved memory and processor requirements.

This chapter includes the following sections:

- ▶ System configuration
- ▶ Same workload with default Active Memory Deduplication settings
- ▶ Different workloads with default Active Memory Deduplication settings
- ▶ Multiple operating system types with memory overcommitment
- ▶ Changes to the deduplication table ratio
- ▶ Variations of the maximum Active Memory Sharing pool size
- ▶ Virtual I/O Server processor sizing

Comparing workloads and results: The results presented in this section do not represent absolute performance numbers. They do provide a way to compare workloads independent of factors that influence the raw performance of a system, such as processor throughput, processor count, I/O access speeds, and so on. Always test a production workload on a test system to ensure that the performance meets your needs.

The tools used to collect the performance statistics are described in 5.2, “Monitoring tools” on page 55.

6.1 System configuration

All of the configuration scenarios presented in this chapter are executed on an IBM POWER7 processor-based server. The server is configured with a dual VIOS and 10 LPARs of each operating system type, totaling 32 LPARs. Three operating systems have been used: AIX, Linux, and IBM i.

Table 6-1 shows the general system configuration used for all of the tests described in this chapter. Additional test-specific setup is shown within the particular test sections.

Table 6-1 System configuration used in scenarios

Component	Value
System model	POWER7 processor-based server
System memory	128 GB
System processor	16 core 3.0 GHz
LPARs	10 Linux 10 AIX 10 IBM i Two VIOS
LPAR memory entitlement (AIX, Linux, IBM i)	3 GB
LPAR processor entitlement (AIX, Linux, IBM i)	0.5 entitled processing units One virtual processor
LPAR storage (AIX, Linux, IBM i)	20 GB in SAN IBM DS4000® drives

6.2 Same workload with default Active Memory Deduplication settings

In this scenario, we enable Active Memory Deduplication with the default settings and demonstrate how much memory is freed by the hypervisor by running the same workload on the LPARs. The default deduplication table ratio is 1/1024, and the 10 LPARs run Linux as the operating system. Different operating systems do not mix in this test.

Table 6-2 summarizes the configuration used in this scenario.

Table 6-2 Configuration: Same workload with default Active Memory Deduplication settings

Setting	Value
Active Memory Deduplication	Enabled
Active Memory Sharing pool size	30 GB
Active Memory Sharing maximum pool size	95 GB
Operating system running on the LPARs	Linux (10 LPARs)
Workload	Same workload on all LPARs
Deduplication table ratio	1/1024 (default)

The Active Memory Sharing pool size is 30 GB and maximum pool size is 95 GB for the base scenario. By using the formula in section 4.1, “Tuning the ratio of the deduplication table” on page 44, you can calculate the deduplication table size to be 95 MB:

$$\text{Deduplication table size} = 95 \text{ GB} \times 1/1024 = 95 \text{ MB}$$

First, we start the workload on all partitions with Active Memory Deduplication disabled. Then, during the run, we enable Active Memory Deduplication with the default settings. Memory statistics are collected during all the duration of the test run.

When we run the same workload in all LPARs, a huge potential exists for memory deduplication because many executable and data pages in memory are expected to be identical. As a result, we obtain a good amount of memory pages being coalesced and freed back to the Active Memory Sharing pool. This is a typical situation customers running database systems encounter during their day-to-day workload: the same database code being executed on multiple partitions.

Figure 6-1 shows the amount of memory coalesced within the LPAR and for the Active Memory Sharing pool.

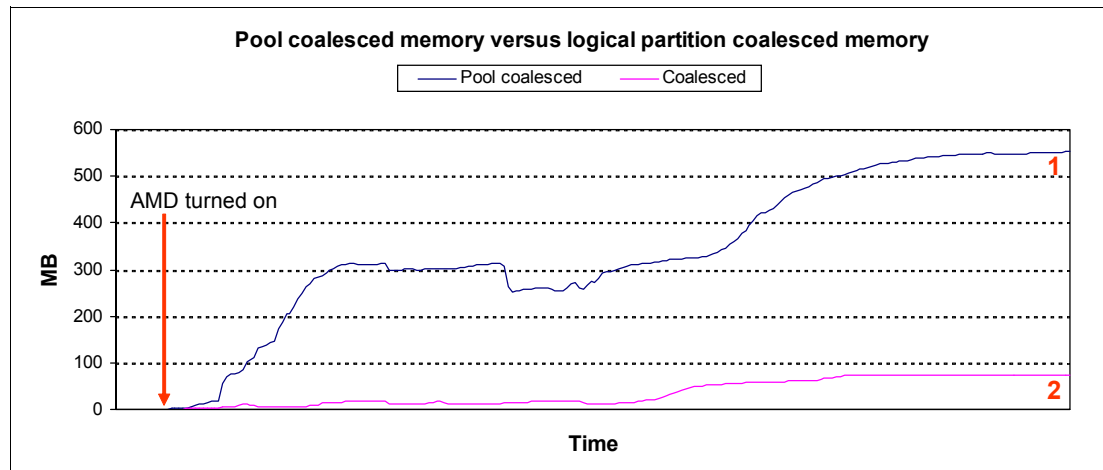


Figure 6-1 Memory coalesced with default settings and the same workload

The chart shows the performance statistics collected from one of the logical partitions used in this scenario. The amount of coalesced pages increases with time after we turn on Active Memory Deduplication and then stabilizes when the workload execution ends.

The number of coalesced bytes for the pool is represented by line number 1. This line indicates the total amount of physical memory freed to the Active Memory Sharing pool by all of the Active Memory Sharing partitions.

Line number 2 indicates the number of coalesced bytes within the partition where we collected the data. This value is the total logical memory deduplicated in that single LPAR.

LPAR and pool coalesced bytes: The coalesced bytes in a LPAR might be higher than the pool coalesced bytes depending of the workload. For example, if a partition has five identical pages in logical memory, then it reports as coalesced the amount of memory equivalent to five pages, whereas the shared pool reports as coalesced the amount of memory for only one physical page.

6.3 Different workloads with default Active Memory Deduplication settings

This scenario is similar to the one presented in section 6.2, “Same workload with default Active Memory Deduplication settings” on page 66. The difference is that we now use 10 Linux LPARs to run three different workloads.

Table 6-3 summarizes the configuration for this test scenario:

Table 6-3 Configuration: Different workloads with default Active Memory Deduplication settings

Setting	Value
Active Memory Deduplication	Enabled
Active Memory Sharing pool size	30 GB
Active Memory Sharing maximum pool size	95 GB
Operating system running on the LPARs	Linux (10 LPARs)
Workload	Five LPARs running a Java workload Three LPARs running a stream workload Two LPARs running a file system-intensive workload
Deduplication table ratio	1/1024 (default)

When the LPARs run a similar workload, it is more likely that more identical pages exist in main memory. However, when workloads are different, finding identical memory pages happens less frequently, and thus the expected gains with memory deduplication are expected to be smaller.

Figure 6-2 shows the amount of deduplicated memory in the Active Memory Sharing pool for two scenarios: one representing three different workloads (line 1), and another representing the same workload on all partitions (line 2), as listed in Table 6-3. In the latter case, five LPARs run a standard Java workload, three others run a streaming workload, and another two run a file system-intensive workload.

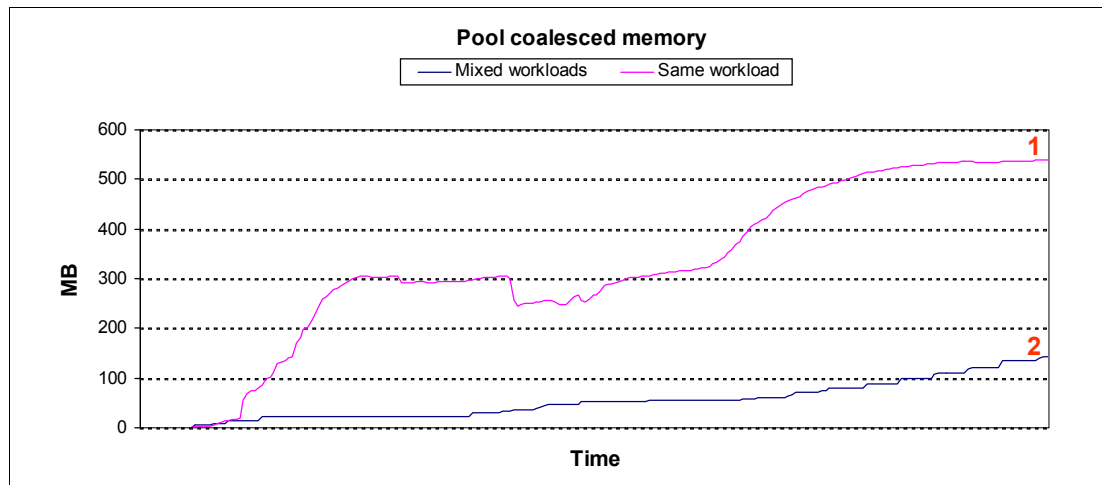


Figure 6-2 Memory deduplication according to workload types

You can see that the amount of coalesced pages depends directly on the workload. Thus, the more similar the workloads among the partitions are, the more memory is deduplicated and made available for further use.

The pool coalesced memory when running the same workload on all partitions is almost four times greater than when running different workloads.

6.4 Multiple operating system types with memory overcommitment

Active Memory Deduplication results in better performance of your Active Memory Sharing pool when it is configured to have logical memory overcommitment, that is, the sum of the desired memory of the LPARs in the shared memory pool exceeds the pool size.

If the Active Memory Sharing pool memory is not logically overcommitted, memory deduplication savings do not really bring much benefit to your environment. Already enough physical memory exists for all of the partitions within the pool at any instant in time. In fact, the freed memory becomes available to the pool, but the other LPARs do not benefit from using it at all. However, cache benefits result from this freed memory.

You can demonstrate this principle in a scenario composed of multiple operating systems configured with shared memory, where the Active Memory Sharing settings force a logical memory overcommitment. It is typical for multiple operating systems to force a logical memory overcommitment, a situation in which the systems can truly increase performance by using Active Memory Deduplication to reduce memory overcommitment by coalescing pages.

Table 6-4 shows the configuration details considered for this test scenario.

Table 6-4 Configuration: Multiple OS types with memory overcommitment

Setting	Value
Active Memory Deduplication	Enabled
Active Memory Sharing pool size	75 GB
Active Memory Sharing maximum pool size	100 GB
Operating system running on the LPARs	AIX (10 LPARs), Linux (10 LPARs), IBM i (10 LPARs)
Workload	Same workload for each operating system (set of 10 LPARS)
Deduplication table ratio	1/1024 (default)

In this scenario, we combine 30 LPARs with three different operating systems running the same workload within the same type of operating system. Each LPAR has 3 GB of memory, totaling a requirement of 90 GB. However, we force logical memory overcommitment by setting the size of the Active Memory Sharing pool to be only 75 GB.

We start the workloads with Active Memory Deduplication disabled and monitor the amount of memory pages that are loaned to the hypervisor. Then, we enable Active Memory Deduplication and compare the number of loaned pages. Figure 6-3 shows the results.

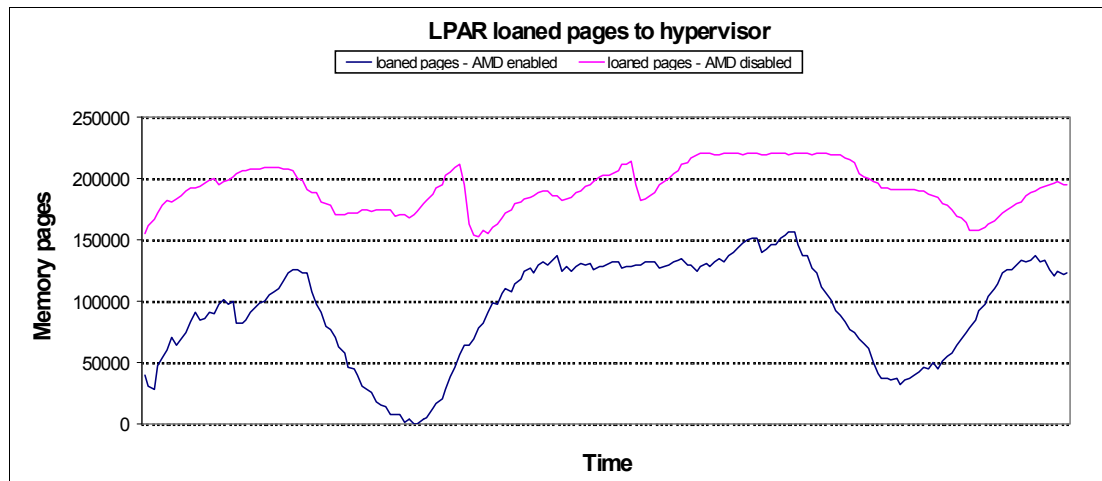


Figure 6-3 Loaned pages with Active Memory Deduplication enabled versus Active Memory Deduplication disabled

You can see the benefits of using Active Memory Deduplication by observing that the number of memory pages the LPAR loans to the hypervisor decreases. This means lesser overcommitment of the shared pool logical memory. In a scenario where the sum of the working memory set sizes of the partitions exceeds the pool memory (physical memory overcommitment), The larger memory set helps reduce I/O onto the Active Memory Sharing paging devices, thus improving overall performance.

6.5 Changes to the deduplication table ratio

In the next scenario, we present three different situations encountered when adjusting the deduplication table size. The variation in the amount of coalesced memory pages is shown for the three different cases, according to the ratio used:

- ▶ Minimum ratio: 1/8192
- ▶ Default ratio: 1/1024
- ▶ Maximum ratio: 1/256

Table 6-5 shows the configuration details considered for this test scenario.

The Active Memory Sharing maximum pool size for this scenario is 95 GB, and the deduplication table size varies according to the deduplication table ratio used. See section 4.1, “Tuning the ratio of the deduplication table” on page 44 for details about how to determine the deduplication table size.

Table 6-5 Configuration: Changes to the deduplication table

Setting	Value
Active Memory Deduplication	Enabled
Active Memory Sharing pool size	30 GB
Active Memory Sharing maximum pool size	95 GB

Setting	Value
Operating system running on the LPAR	Linux (10 LPARs)
Workload	Same workload on all LPARs
Deduplication table ratio	Minimum: 1/8192 Default: 1/1024 Maximum: 1/256
Deduplication table size	Minimum: 11.8 MB Default: 95 MB Maximum: 380 MB

We run the test in three steps, each running the same workload:

1. We set the deduplication table ratio to the minimum value of 1/8192.
2. We set the deduplication table ratio to the default value of 1/1024.
3. We set the maximum ratio of 1/256.

In order to start each run from a clean memory state, we clear the memory cache and set the deduplication table ratio before each run. We can clear cache memory in Linux using this command:

```
echo 1 > /proc/sys/vm/drop_caches
```

Figure 6-4 compares different deduplication table ratios and their memory savings for each case.

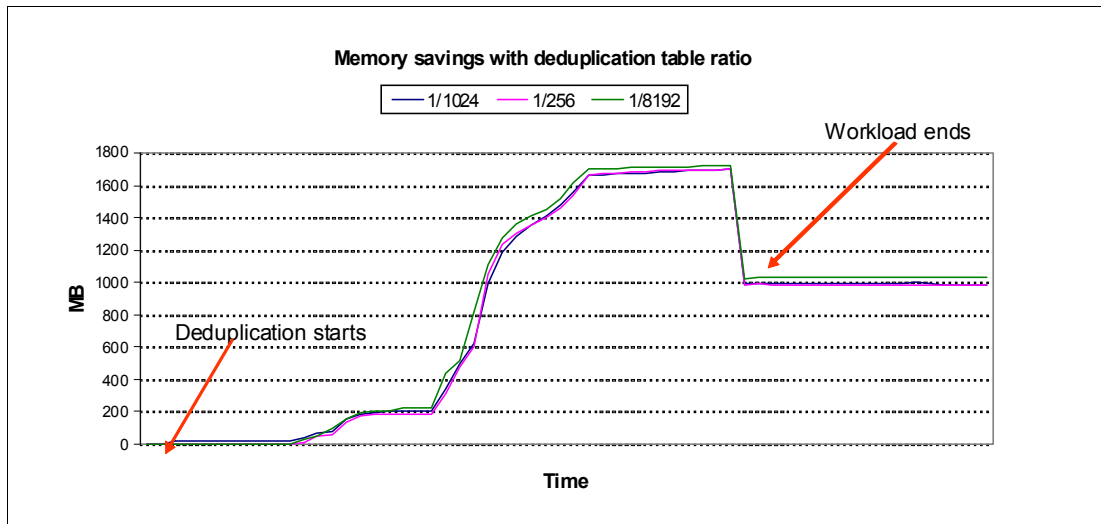


Figure 6-4 Memory savings when varying the deduplication table ratio

By looking at the memory savings comparison chart, we can reach the following conclusions:

- ▶ As soon as we turn Active Memory Deduplication on, we observe that memory pages are gradually coalesced as time goes by.
- ▶ The peak indicates that no more pages exist to be coalesced for the running workload.
- ▶ The amount of memory savings is almost equal for all of the three deduplication table ratios.
- ▶ When the workload ends, all of the three lines follow the same trend and stabilize at around 1000 MB. This congruence happens because, from that point on, only the operating system code and data pages tend to be identical among the LPARs and get deduplicated.

An analysis of memory consumption versus coalesced memory is presented in Table 6-6.

Table 6-6 Memory consumption versus coalesced memory based on deduplication table ratio

Deduplication table ratio	Deduplication table size	Pool coalesced memory
1/256	380 MB	135 MB
1/1024	95 MB	135 MB
1/8192	11 MB	135 MB

Using 1/1024 suits most workloads and is a good starting point for tuning. As you can see, for our particular test, the minimum ratio of 1/8192 yields the best tuning in terms of memory savings versus memory used by the deduplication table. However, the value we choose might not always be best. If you have more partitions than the amount used for these tests and different workloads, you might need to use greater ratios.

6.6 Variations of the maximum Active Memory Sharing pool size

In this scenario, we show the amount of memory that is coalesced in an environment of 10 AIX LPARs running the same workload. Between the test runs, we change the Active Memory Sharing maximum pool size (`max_pool_size`). Changing the maximum pool size affects the size of the deduplication table.

The tests are run with the following maximum pool sizes:

- ▶ Small: 60 GB
- ▶ Baseline: 95 GB
- ▶ Large: 110 GB

Table 6-7 shows the configuration details considered for this test scenario.

Table 6-7 Configuration: variations of the maximum Active Memory Sharing pool size

Setting	Value
Active Memory Deduplication	Enabled
Active Memory Sharing pool size	30 GB
Active Memory Sharing maximum pool size	Smaller: 60 GB Baseline: 95 GB Larger: 110 GB
Operating system running on the LPARs	AIX (10 LPARs)
Workload	Same workload on all LPARs
Deduplication table ratio	1/1024 (default)
Deduplication table size	Smaller: 60 MB Baseline: 95 MB Larger: 110 MB

The deduplication table ratio is kept as 1/1024, the default Active Memory Deduplication setting for this parameter. During all of the three runs, each with different maximum pool sizes, the workload was kept the same on all of the partitions.

The test results for the amount of pool coalesced memory is shown in Figure 6-5.

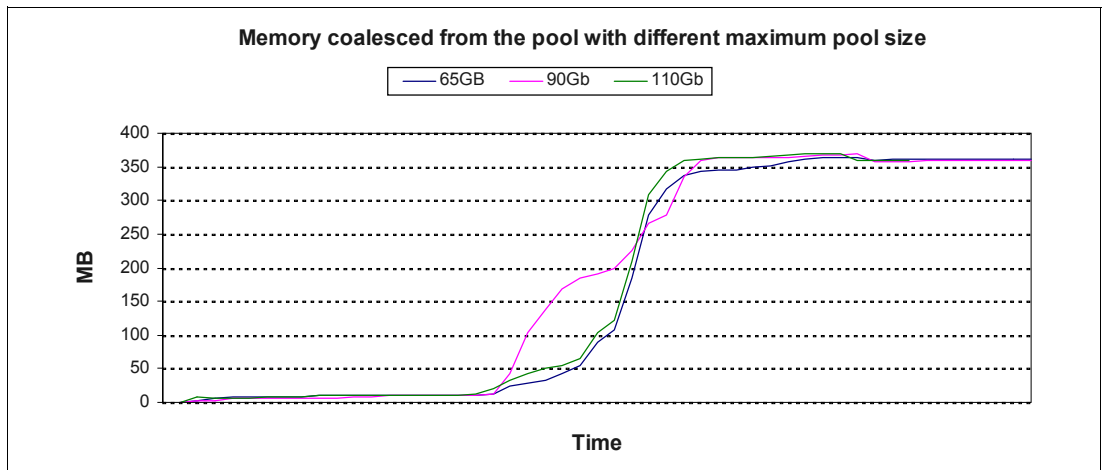


Figure 6-5 Coalesced pool memory versus maximum pool size

Test results for LPAR coalesced memory is shown in Figure 6-6.

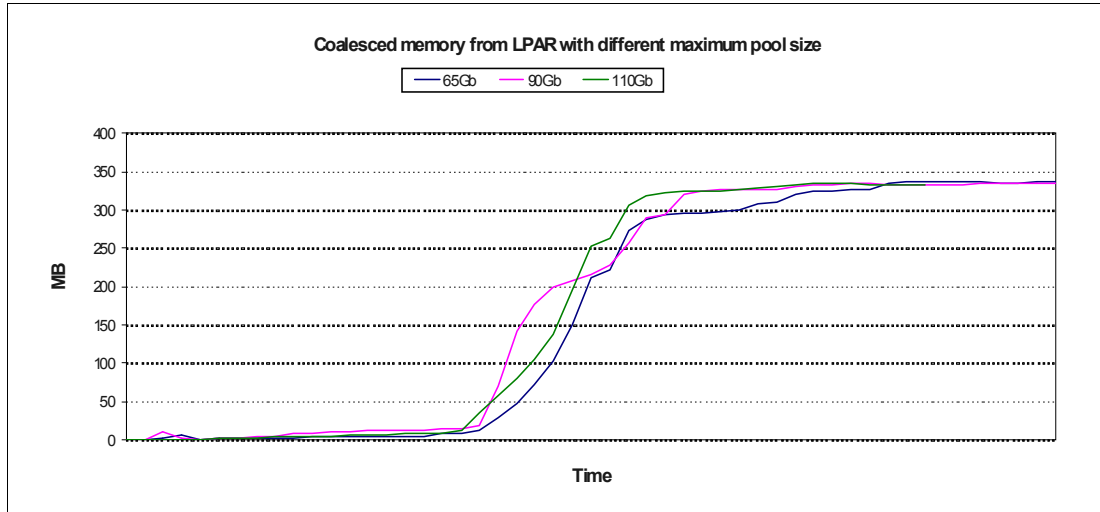


Figure 6-6 LPAR coalesced memory versus maximum pool size

Notice that changing the maximum pool size does not affect the size of the deduplication table beyond a point that changes the trend of the deduplication performance. Table 6-8 summarizes this principle.

Table 6-8 Memory consumption versus coalesced memory according to Active Memory Sharing maximum pool size

Active Memory Sharing maximum pool size	Deduplication table size	Pool coalesced memory
65 GB	65 MB	370 MB
90 GB	95 MB	370 MB
110 GB	110 MB	370 MB

As you can see, the variations we use on the Active Memory Sharing maximum pool size do not affect the results for this particular case.

You might conclude that the smaller the deduplication table the better, as long as it does not become too small and start affecting Active Memory Deduplication performance. However, changing the maximum pool size simply to vary the deduplication table size is not suggested because the maximum pool size is also a setup parameter for Active Memory Sharing.

Set up your maximum Active Memory Sharing pool size according to your Active Memory Sharing requirements. Later, if you want to vary the size of the deduplication table, start by tuning the deduplication table ratio as demonstrated in 6.5, “Changes to the deduplication table ratio” on page 70.

6.7 Virtual I/O Server processor sizing

In this scenario, we examine two different situations in which the processing units and virtual processors of the VIOS are adjusted. The test environment is based on a dual VIOS environment with each VIOS sized as follows:

- ▶ Undersized: 0.1 processing units, one virtual processor, capped LPAR
- ▶ Properly sized: 1.0 processing units, four virtual processors, uncapped LPAR

In this test, we have 10 AIX LPARs running the same workload. The memory pool usage over time is shown for the two cases in Figure 6-7. For each test run, both VIOSs are set up with equal configuration, except for the parameters mentioned in Table 6-9.

Table 6-9 Configuration: VIOS processor sizing

Setting	Value
Active Memory Deduplication	Enabled
Active Memory Sharing pool size	30 GB
Active Memory Sharing maximum pool size	95 GB
Operating system running on the LPARs	AIX (10 LPARs)
Workload	Same workload on all LPARs
Deduplication table ratio	1/1024 (default)
Undersized VIOS	Processing units: 0.1 Virtual processors: 1 LPAR mode: capped
Properly sized VIOS	Processing units: 1.0 Virtual processors: 4 LPAR mode: uncapped (weight=255)

Figure 6-7 shows the levels of coalesced pages in the memory pool during both tests to compare the two scenarios.

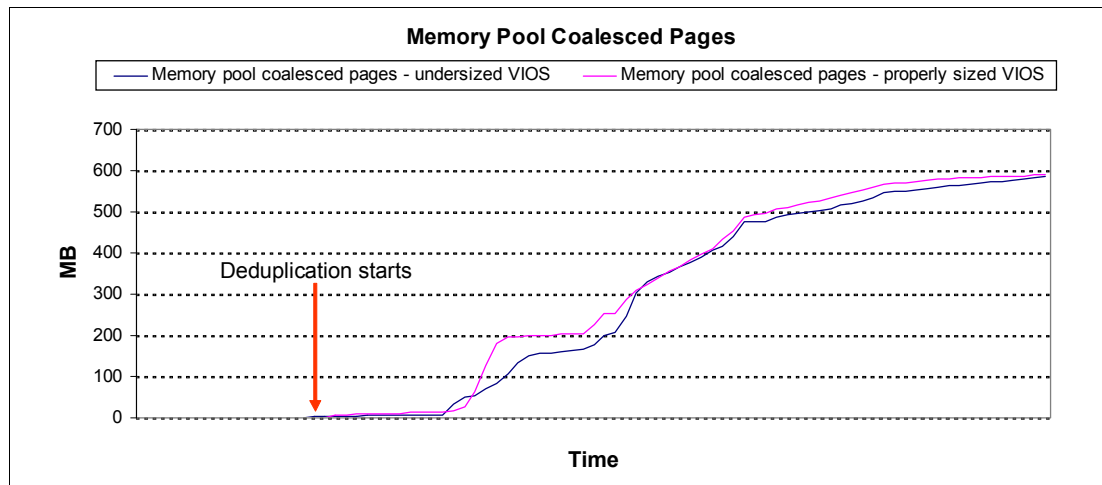


Figure 6-7 Memory pool coalesced pages comparison

You can see only a slight difference in the amount of coalesced pool memory between the two configurations. The performance of properly sized VIOSs is only somewhat better.

Notice in Figure 6-8 how close to the entitled processor capacity the undersized VIOS processor consumption is. This proximity suggests that, in an undersized configuration, processor utilization is constantly close to a bottleneck situation.

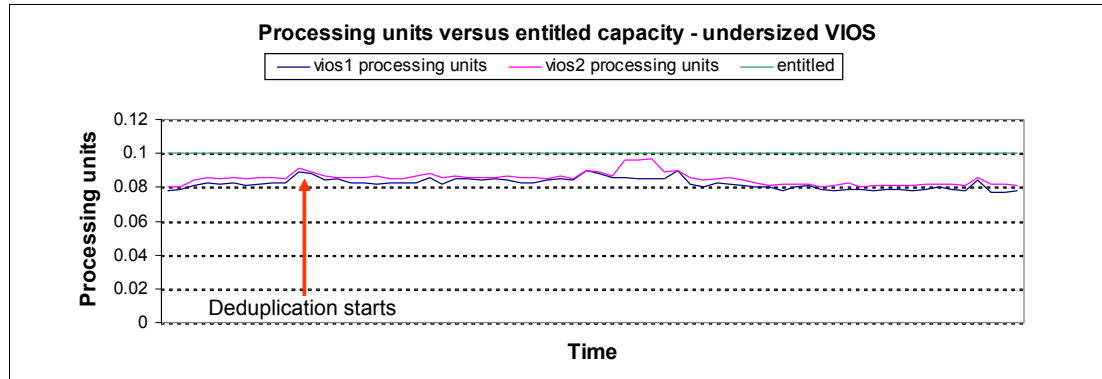


Figure 6-8 Processing units versus entitled processor capacity for an undersized VIOS configuration

In Figure 6-9, it is possible to see that both VIOSs do not have any processor shortage when properly sized. Notice in this figure that the overall consumption is around 0.10 and 0.15 processing units. This value increases in comparison to the undersized scenario, but the increase is almost unnoticeable.

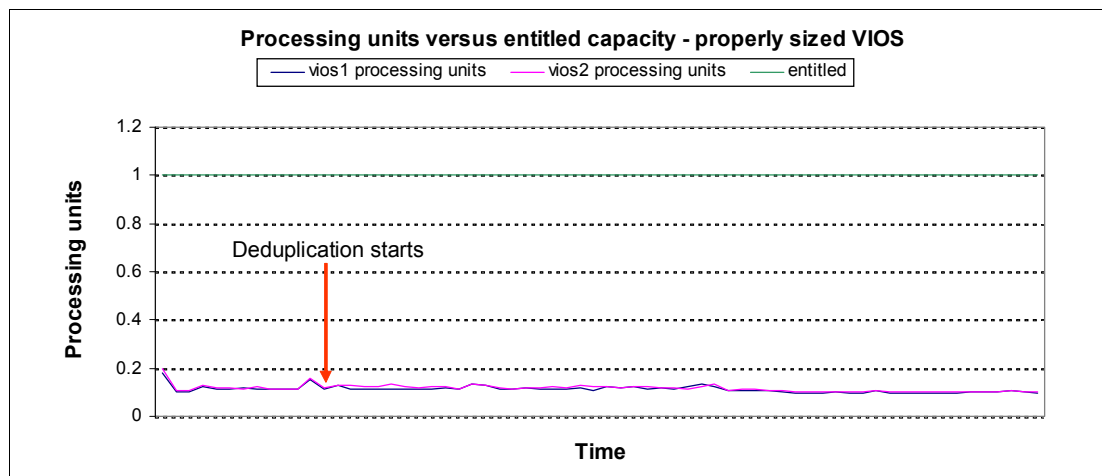


Figure 6-9 Processing units versus entitled capacity for a properly sized VIOS configuration

The test results show that Active Memory Deduplication processing requirements are in practice very low. Even with a lack of processing units, run queues of both VIOSs do not change much. Also, the amount of coalesced memory pool pages is about the same in both cases.

Although processor consumption is not an issue for enabling Active Memory Deduplication, it does increase VIOS processing load. The VIOSs can surely take advantage of additional available processing units, it is suggested that you always properly size them. After all, the VIOSs also handle other PowerVM-related features, and processor shortages cause several performance problems.



Best practices

This chapter describes best practice guidelines for using Active Memory Deduplication. Upon reading this chapter, you should be able to do these things:

- ▶ Optimally tune Active Memory Deduplication parameters
- ▶ Optimally size your environment in terms of processor requirements

This chapter includes the following sections:

- ▶ Tuning best practices
- ▶ Tuning summary

See Chapter 4, “Tunable parameters” on page 43, for the steps on how to change parameter values.

7.1 Tuning best practices

If you have no prior experience with Active Memory Deduplication, this section describes best practices based on the test results in Chapter 6, “Configuration scenarios” on page 65. Here we present the best tuning start points for general workloads:

- ▶ Deduplication table ratio
- ▶ Maximum Active Memory Sharing pool size
- ▶ Virtual I/O Server (VIOS) processor sizing

7.1.1 Best practice for the deduplication table ratio

The deduplication table ratio is one of the factors that define the size of the deduplication table. This ratio is your entry point for most effectively tuning Active Memory Deduplication. Whenever you want to change the table size, the deduplication table ratio is the parameter that you should tune.

According to our tests in section 6.5, “Changes to the deduplication table ratio” on page 70, using a small ratio creates room for potentially missing coalescing opportunities. The deduplication table gets too small to be able to compare a considerable amount of page signatures at an instant in time.

A large deduplication table ratio is beneficial when you have many systems running the same workload. Test results show that running the same workload on 10 logical partitions (LPARs) is not enough to increase deduplicated memory gains substantially when increasing the ratio beyond the default configuration (1/1024). We only make the deduplication table larger and gain nothing in return.

Most mid-range Power servers are not configured to have more than a couple of dozen partitions adding up to a few hundred GB of memory. When servers have a limited number of partitions and limited memory, the default ratio of 1/1024 is perfectly suitable and yields good savings compared to the memory footprint consumed by the deduplication table.

Overall, it is suggested that you begin tuning the deduplication table ratio with the default setting of 1/1024. If your Power server has more than about 30 LPARs, then consider increasing the ratio by one factor at a time until you observe no additional gains in memory savings. Go from 1/1024 to 1/512, and only then to 1/256 if needed.

7.1.2 Best practice for the maximum Active Memory Sharing pool size

The maximum shared memory pool size is a parameter that does not uniquely relate to Active Memory Deduplication. Its value is used to calculate the deduplication table size. It is suggested that you do not change this value to force a change in the deduplication table size. Changing the maximum pool size also forces the hypervisor to change the amount of memory reserved for Active Memory Sharing activity.

In essence, let your Active Memory Sharing configuration requirements dictate the maximum memory pool size. If you later believe that your settings for this parameter cause the deduplication pool to be too large, then tune the deduplication table ratio as explained in 7.1.1, “Best practice for the deduplication table ratio” on page 78.

7.1.3 Best practice for Virtual I/O Server sizing

As you can see from the test results in section 6.7, “Virtual I/O Server processor sizing” on page 75, the processor requirements for deduplicating memory is virtually zero. Do not be surprised when you try to run similar tests yourself and the processor units used by Active Memory Deduplication is reported as 0.0. The truth is that, in practice, activating Active Memory Deduplication produces noticeable impact on processor use.

Processor impact: Processor consumption for deduplicating memory is virtually zero.

Processor resources are hardly impacted by deduplication. However, in a highly available environment that requires the most performance out of the server, it is suggested that you add an extra slack (spare capacity) of 0.1 processing units to your VIOS configuration. Adding this slack ensures that Active Memory Deduplication can still run even when your VIOS processor use peaks. If you already have such a slack, then no further change is required.

If you conclude that you need to add slack, see section 4.3, “Spare Virtual I/O Server processor cycles” on page 48. If you have a dual VIOS environment, then ensure both VIOSs have a slack of 0.1 processing units so that you have a slack of 0.1 in case one VIOS fails.

7.2 Tuning summary

Table 7-1 summarizes the best practices on the values of Active Memory Deduplication tunables. The values shown here suit most environments.

Table 7-1 Suggested best practices for Active Memory Deduplication parameter tuning

Parameter	Suggested value
Deduplication table ratio	1/1024
Maximum Active Memory Sharing pool size	Determined by Active Memory Sharing requirements, not Active Memory Deduplication
VIOS processing units slack	+0.1 per VIOS

Note: The default configuration of Active Memory Deduplication parameters fits well in almost all shared memory scenarios. Change them only if you need to fine-tune your environment and have statistical data to back up your decisions.



Troubleshooting

This section targets usual problems that you might face when enabling Active Memory Deduplication in your environment.

This chapter includes the following sections:

- ▶ Common errors
- ▶ HSCLA errors

8.1 Common errors

The following errors are described according to the frequency that they appear. Also, the error messages related to Active Memory Deduplication starting with message code HSCLA4 are described.

If you have any other problems, consult the next level of support from your service provider.

8.1.1 Performance statistics are not displayed

The shared memory pool coalesced bytes value is not reported by the corresponding operating system command.

In Linux, **amsstat** does not display the **pool_coalesced_bytes** field, as shown in Example 8-1:

Example 8-1 Linux amsstat command results without the pool_coalesced_bytes field

```
[root@linux1 ~]# amsstat | grep pool_coal -A 2 -B 2
backing_memory:          1402494976
coalesced_bytes:         182829056
pool_coalesced_bytes:    313352192
cmo_enabled:             1
cmo_faults:              94
```

In AIX, the **mpgcol** column displays zero, as shown in Example 8-2.

Example 8-2 AIX mpgcol column with a zero value

```
root@aix1dedup /root # lparstat -mp
```

```
System configuration: lcpu=4 mem=3072MB mpsz=40.00GB iome=111.00MB iomp=10
ent=0.50
```

```
physb  hpi  hpit  pmem  pgcol  mpgcol  ccol  %entc  vcsw
-----
0.00   214   55   0.82  654.4  0.0    0.0   0.0  301550
```

To be able to collect this data on both operating systems, configure the logical partitions (LPARs) to allow performance information collection as shown in Section 5.2, “Monitoring tools” on page 55

8.1.2 Command not found message for amsstat on Linux

In Linux, if the user is trying to monitor the coalesced bytes and the **amsstat** command is not found, it means that the **powerpc-utils** package is not installed. This package should be installed. See section 5.2.3, “Monitoring in Linux” on page 56 for more information about the topic.

8.1.3 Few coalesced pages

The amount of coalesced memory pages depends on multiple factors. First, it depends on the existence of duplicated data within the system memory. Thus, if the machine is running different operating systems, then it is less probable that a page match occurs. This also occurs at the application code level. The more dissimilar the applications running on the LPARs are, the less probable it is that deduplication occurs.

If you have multiple Power servers, your objective should be grouping the LPARs onto the same server by operating system and application type. This grouping increases the chance that identical memory pages coexist within the system memory.

You might already have a system in which the LPARs are grouped by OS and application type but you still feel that too few pages are being coalesced. In this case, use Active Memory Deduplication tunables to enhance system performance. Setting these tunables is described in Chapter 4, “Tunable parameters” on page 43.

8.1.4 ccol is always zero in lparstat

If the **ccol** field in **lparstat** shows 0.0 all the time, this is not considered an error. The amount of processing power used by Active Memory Deduplication is not noticeable. Thus, this value should be near zero almost all of the time.

8.2 HSCLA errors

Error messages related to Active Memory Deduplication starting with HSCLA4 are described in this section.

8.2.1 Error code HSCLA4F3 when changing the deduplication table

If you are changing the deduplication table ratio and Active Memory Deduplication is not enabled on the system, the following error is returned:

```
HSCLA4F3 The deduplication table ratio cannot be specified when Active Memory Deduplication is disabled for the shared memory pool.
```

You should turn on Active Memory Deduplication before changing the ratio. See section 3.3, “Enabling and disabling Active Memory Deduplication” on page 37 for information about how to turn on Active Memory Deduplication in your system.

8.2.2 Error code HSCLA4F2 when disabling Active Memory Deduplication

If you get the error code HSCLA4F2 when disabling Active Memory Deduplication, it means that Active Memory Deduplication is already disabled for the shared memory pool.

8.2.3 Error code HSCLA4F1 when enabling Active Memory Deduplication

If you get the error code HSCLA4F1 when enabling Active Memory Deduplication, it means that Active Memory Deduplication is already enabled for the shared memory pool.

8.2.4 Error code HSCLA4F0 when enabling Active Memory Deduplication

If the system returns the error code HSCLA4F0 when you try to enable Active Memory Deduplication, it means that the managed system does not support Active Memory Deduplication. Check to make sure that your system has all of the required requirements mentioned in Chapter 3, “Planning for Active Memory Deduplication” on page 17.

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.

- ▶ *IBM PowerVM Virtualization Active Memory Sharing*, REDP-4470
- ▶ *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590
- ▶ *Hardware Management Console V7 Handbook*, SG24-7491
- ▶ *IBM Systems Director Management Console: Introduction and Overview*, SG24-7860

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

ibm.com/redbooks

Other publications

This publication is also relevant as further information source:

- ▶ *Active Memory Deduplication*

https://www.ibm.com/developerworks/mydeveloperworks/blogs/aixpert/entry/active_memory_deduplication131?lang=en

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Power Systems Memory Deduplication



Learn about enhanced performance through shared memory

See how to increase server utilization with memory savings

Follow scenarios that show setup and configuration

This IBM Redpaper publication introduces IBM PowerVM Active Memory Deduplication on IBM Power Systems based on IBM POWER7 processor technology.

Active Memory Deduplication is a virtualization technology through which memory pages with identical contents can be deduplicated in physical memory. This deduplication frees up physical memory positions so that more data can be held in memory at once.

Memory deduplication is intended to work in a shared memory environment. Deduplication works with Active Memory Sharing, a technology through which multiple partitions on a system can share a pool of physical memory. Sharing this pool sometimes creates an overcommitment of this physical memory. Active Memory Deduplication increases the performance of Active Memory Sharing because the savings can be used to lower memory overcommitment levels. Reduction of memory in use can also create room to increase the memory footprint of the logical partitions (LPARs).

This paper describes the advantages of Active Memory Deduplication and requirements for its use and implementation in your Power Systems. This paper also helps you set up Active Memory Deduplication on your Power System for use with IBM AIX, IBM i, and Linux. It includes guidance for tuning parameters of Active Memory Deduplication and for troubleshooting issues.

This paper targets architects and consultants who need to understand how Active Memory Deduplication technology works to design performing solutions. The paper also targets technical specialists in charge of setting up and managing Active Memory Deduplication.

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**