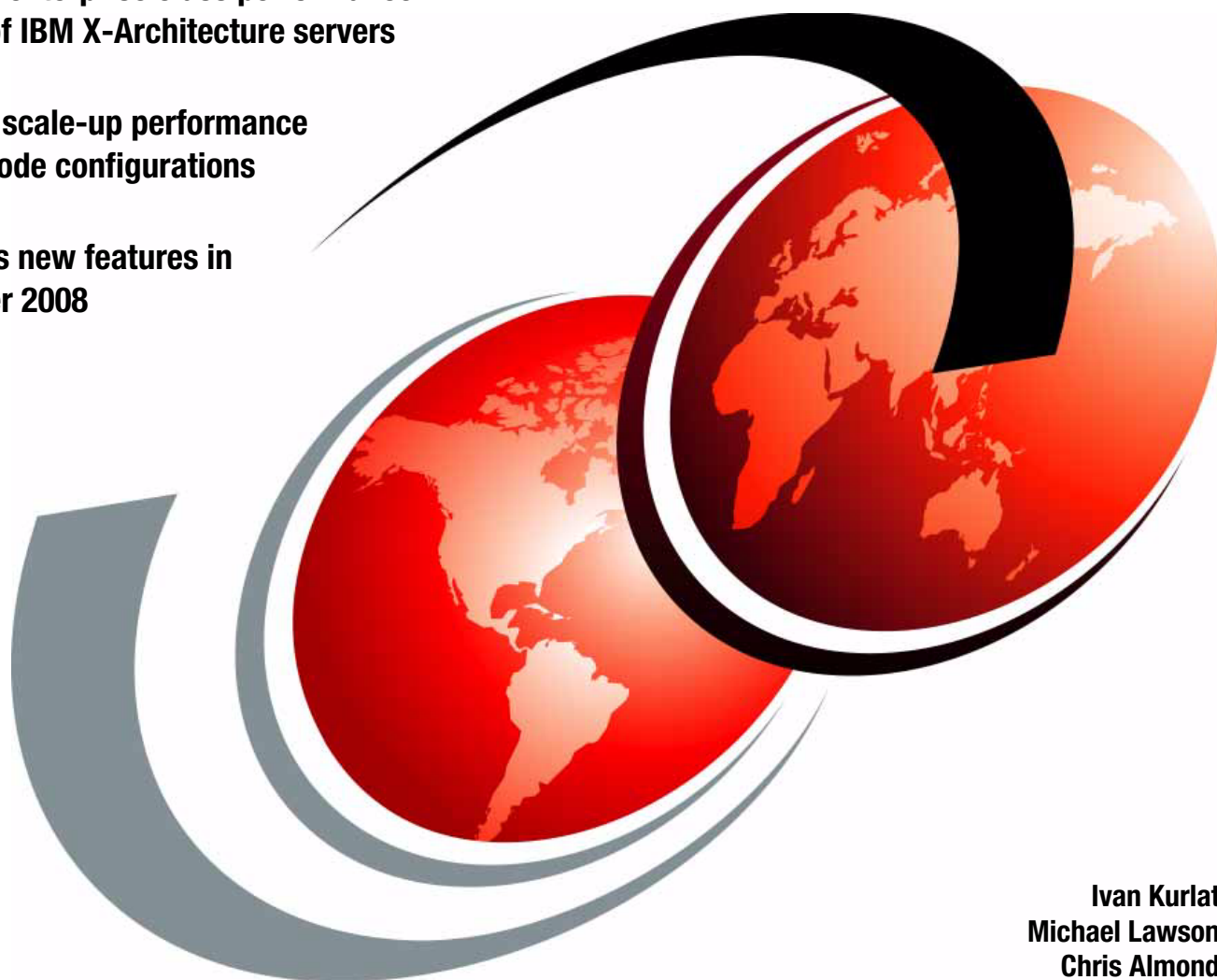


# Running Microsoft SQL Server 2008 on the IBM System x3950 M2

Utilize the enterprise class performance features of IBM X-Architecture servers

Maximize scale-up performance in multi-node configurations

Introduces new features in SQL Server 2008



Ivan Kurlat  
Michael Lawson  
Chris Almond





International Technical Support Organization

**Running Microsoft SQL Server 2008 on the  
IBM System x3950 M2**

March 2009

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

**First Edition (March 2009)**

This document applies to Microsoft SQL Server 2008 and Microsoft Windows Server 2008 running on the IBM System x3950 M2.

This document created or updated on March 25, 2009.

**© Copyright International Business Machines Corporation 2009. All rights reserved.**

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

# Contents

<b>Notices</b> .....	v
Trademarks .....	vi
<b>Preface</b> .....	vii
The team that wrote this paper .....	vii
Acknowledgements .....	viii
Become a published author .....	ix
Comments welcome .....	ix
<b>Chapter 1. The IBM System x3950 M2 enterprise database server</b> .....	1
1.1 Introduction to the IBM System x3950 M2 .....	2
1.2 IBM XA-64e fourth-generation chipset .....	3
1.3 Processors .....	4
1.4 System memory .....	7
1.5 Multinode configurations .....	9
1.6 Broadcom dual Gigabit Ethernet controller .....	12
1.7 SAS disk subsystem .....	12
1.8 PCI subsystem .....	13
1.9 Redundancy .....	13
<b>Chapter 2. SQL Server 2008</b> .....	15
2.1 SQL Server 2008 Editions .....	16
2.2 New and enhanced features of SQL Server 2008 .....	16
2.2.1 SQL Server installation .....	16
2.2.2 Database engine new and enhanced features .....	17
2.2.3 Analysis Services new and enhanced features .....	21
2.2.4 Reporting services new architecture design .....	21
2.2.5 Additional enhancements and features .....	23
2.3 64-bit computing and SQL Server 2008 .....	24
2.3.1 Windows, SQL Server, and 32-bit versus 64-bit .....	24
2.4 Windows Server 2008 editions .....	26
2.4.1 Comparing Windows Server 2008 editions .....	26
2.4.2 Windows Datacenter models .....	27
2.4.3 Processor and memory limits .....	27
<b>Chapter 3. Platform synergy</b> .....	29
3.1 Scalable hardware implementation .....	30
3.2 Affinity in Windows Server 2008 .....	32
3.2.1 NUMA optimization for Windows Server 2008 .....	32
3.2.2 Process and thread scheduling .....	33
3.3 Affinity in SQL Server 2008 .....	35
3.3.1 SQL Server Operating System .....	35
3.3.2 Processor and I/O affinity .....	36
3.3.3 Soft NUMA .....	38
3.3.4 Network affinity .....	40
3.3.5 Memory .....	43
3.3.6 Resource Governor .....	44
3.4 Server consolidation .....	49
3.4.1 Concept of consolidation .....	49

3.4.2	Forms of consolidation . . . . .	50
3.4.3	Consolidation strategies . . . . .	51
3.4.4	Considering service availability . . . . .	56
<b>Chapter 4.</b>	<b>Configuration . . . . .</b>	<b>67</b>
4.1	x3950 M2 configuration . . . . .	68
4.1.1	System setup . . . . .	68
4.1.2	Firmware update . . . . .	68
4.2	Windows Server 2008 x64 configuration . . . . .	70
4.2.1	Windows installation, service pack, updates, and drivers . . . . .	70
4.2.2	Windows settings . . . . .	70
4.2.3	Anti-virus software . . . . .	71
4.3	SQL Server 2008 x64 configuration . . . . .	71
4.3.1	SQL Server 2008 installation, service packs, and hot fixes . . . . .	71
4.3.2	SQL Server 2008 settings . . . . .	72
4.4	Storage configuration . . . . .	74
4.4.1	Storage setup . . . . .	75
4.4.2	Storage file placement . . . . .	76
4.5	Database workloads . . . . .	77
4.5.1	Maintenance operations . . . . .	77
4.5.2	Application workloads . . . . .	78
4.6	Performance analysis process . . . . .	79
4.6.1	Performance baseline . . . . .	79
4.6.2	Hardware resources . . . . .	80
4.6.3	Other diagnostic and performance tools . . . . .	82
<b>Appendix A.</b>	<b>Unattended install . . . . .</b>	<b>87</b>
A.1	Benefits of unattended installation . . . . .	88
A.2	Installation using the configuration file . . . . .	88
A.2.1	SQL Server 2008 installation GUI . . . . .	88
A.2.2	Manual configuration file generation . . . . .	89
A.2.3	Performing the install . . . . .	90
A.3	Installation from the command prompt . . . . .	90
A.3.1	Command prompt SQL Sever 2008 fix installation . . . . .	92
<b>Related publications</b>	. . . . .	<b>93</b>
IBM Redbooks publications	. . . . .	93
Online resources	. . . . .	93
How to get Redbooks publications.	. . . . .	93
Help from IBM	. . . . .	93

# 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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:

Active Memory™  
DS4000®  
DS8000®  
eServer™

IBM®  
Redbooks®  
Redbooks (logo) ®  
ServerProven®

System Storage™  
System x®  
X-Architecture®  
xSeries®

The following terms are trademarks of other companies:

AMD, ATI, ES1000, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

VMware, the VMware "boxes" logo and design are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions.

Excel, Microsoft, SQL Server, Windows Server, Windows Vista, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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



# Preface

The IBM System x3950 M2 server provides extraordinary levels of scalability and processing power needed to run the most demanding enterprise database workloads. The x3950 M2 system is based on the fourth generation of IBM X-Architecture® and is built on eX4 technology. As a scale-up platform, it offers unprecedented flexibility for supporting multi-node configurations using up to 96 processor cores.

The System x3950 M2 is the ultimate in scale-up design. Scale-up refers to the idea of increasing processing capacity by adding additional processors (and memory and I/O bandwidth) to a single server, making it more powerful. This is precisely what the x3950 M2 is designed to do: to scale up by adding chassis to a hardware partition to form two-node, four-node, and eight-node configurations. SQL Server 2008 is also designed for scale-up.

Thus, when your database application needs to scale-up to handle increased demands, the x3950 M2 and SQL Server 2008 together can grow from a one-node server to an enterprise-class four-node server with up to 96 processor cores and 1 TB of physical memory, which the 64-bit SQL Server 2008 x64 can take full advantage of.

This IBM Redbooks® publication presents technical considerations and configuration examples for scaling up Microsoft® SQL Server® 2008 to support very high workloads.

## The team that wrote this paper

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

**Ivan Kurlat** is an IT Specialist and currently the Team Leader for SQL Services within the IBM Argentina Global Server Systems Operations team.

**Michael Lawson** is a database and virtualization specialist at the IBM Center for Microsoft Technologies in Kirkland, WA. He has a B.A. in Information Sciences and Mathematics from the University of California at Santa Cruz. Michael has 30 years of experience as a database administrator, the last 14 years with Microsoft SQL Server. Since joining IBM in 1999, he has supported IBM customers running SQL Server on System x® servers. He has MCDBA (Microsoft Certified Database Administrator) certification. Michael is also an author of the Redpaper *SQL Server 2005 on the IBM eServer xSeries 460 Enterprise Server*.



*The authors (l-r): Mike and Ivan*

Production of this IBM® Redbooks publication was managed by Chris Almond, an ITSO Project Leader and IT Architect based in Austin, Texas.

## Acknowledgements

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

- ▶ Ron Arndt
- ▶ Chuck Bauman
- ▶ Peter Donovan
- ▶ Sue Goodwin
- ▶ Tim Graves
- ▶ Vinay Kulkarni
- ▶ Kevin Powell
- ▶ Heather Richardson

From the International Technical Support Organization:

- ▶ Tamikia Barrow
- ▶ Linda Robinson
- ▶ David Watts

We would also like to thank the authors of the following IBM Redbooks publications for their contributions to this paper:

- ▶ Authors of *SQL Server 2005 on the IBM eServer xSeries 460 Enterprise Server*, REDP-4093:
  - Michael Lawson
  - Yuichiro Tanimoto
  - David Watts

- ▶ Authors of *Consolidating Microsoft SQL Server on the IBM System x3950 M2*, REDP-4385:
  - Daniel Soares de Barros
  - Simon Champion
  - David Watts

## Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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

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

## Comments welcome

Your comments are important to us!

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

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

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


- ▶ Send your comments in an e-mail to:

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

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400





# The IBM System x3950 M2 enterprise database server

Delivering an industry-leading, 64-bit framework for high-performance scalable computing, the IBM System x3950 M2 is built on the power of IBM eX4 technology, the fourth generation of IBM X-Architecture. The eX4 technology drives the x3950 M2 to deliver the performance, availability, expandability, and manageability required for the next generation of Microsoft's enterprise database platform, SQL Server 2008.

Top 4-way performance, expandable to 16-way (with 96 cores), and 64-bit memory addressability provide an optimized high-end database platform. At the crossroads of a major industry transition to mainstream 64-bit applications and multicore processors, eX4 Architecture delivers a formidable combination of performance, availability, and investment protection that is not yet available in an industry-standard x86 server.

Extensive chipset development experience and industry-leading performance and availability breakthroughs have uniquely positioned IBM to provide a robust and powerful server, offering innovation that delivers real business and IT results.

This chapter covers the following topics:

- ▶ 1.1, "Introduction to the IBM System x3950 M2" on page 2
- ▶ 1.2, "IBM XA-64e fourth-generation chipset" on page 3
- ▶ 1.3, "Processors" on page 4
- ▶ 1.4, "System memory" on page 7
- ▶ 1.5, "Multinode configurations" on page 9
- ▶ 1.6, "Broadcom dual Gigabit Ethernet controller" on page 12
- ▶ 1.7, "SAS disk subsystem" on page 12
- ▶ 1.8, "PCI subsystem" on page 13
- ▶ 1.9, "Redundancy" on page 13

## 1.1 Introduction to the IBM System x3950 M2

The x3950 M2 (Figure 1-1) is a high-performance, 2-way to 16-way scalable server that is powered by fourth-generation IBM X-Architecture and Intel® Xeon processors each with up to 6 cores. With a 64-bit framework for high-performance scalable computing, the x3950 M2 is built on the new IBM eX4 technology and is optimized for ERP applications, high-end databases, and server consolidation.



Figure 1-1 The System x3950 M2

For additional details on the x3950 M2 see the IBM Redbooks publication *Planning, Installing, and Managing the IBM System x3950 M2*, SG24-7630, available from:

<http://www.redbooks.ibm.com/abstracts/sg247630.html>

The key features of the x3950 M2 are:

- ▶ Four-way<sup>1</sup> capable server
- ▶ Scalable to two, three, or four nodes to provide a multi-node server configuration with up to 16 six-core processors (totalling 96 cores), 1 TB of RAM, and 28 PCI-X slots
- ▶ XA-64e fourth-generation chipset (see 1.2, “IBM XA-64e fourth-generation chipset” on page 3 for more information)
- ▶ Up to four Intel Xeon® processors, each with up to 6 cores, that support 64-bit addressing with the Intel 64 Technology architecture (see 1.3, “Processors” on page 4)
- ▶ Support for Intel Virtualization Technology (VT)
- ▶ Support for an embedded hypervisor option
- ▶ Support for an internal removable flash drive installed in a dedicated USB connector on the system board
- ▶ 4 GB or 8 GB memory standard expandable to 256 GB (using 8GB DIMMs), using high-performance PC2-5300 ECC DDR2 DIMMs
- ▶ Active Memory™ with Memory ProteXion, memory mirroring, memory hot-swap and hot-add, and ChipKill
- ▶ Seven half-length 64-bit PCI Express x8 slots, two of which are hot-swap
- ▶ Four internal hot-swap drive bays for up to 587 GB of internal storage (with 146.8 GB disks)

<sup>1</sup> Four-way means 4 processor sockets. In this document, we use *way* to indicate a processor socket regardless of whether it is a dual-core processor, a quad-core processor, or a six-core processor.

- ▶ Integrated LSI 1078 Serial-Attached SCSI (SAS) controller that supports RAID-0 and RAID-1 standard and an optional ServeRAID-MR10k RAID controller for enabling additional RAID features and a 256 MB battery backed cache
- ▶ Integrated Dual-port Broadcom 5709C PCI Express Gigabit Ethernet controller
- ▶ Onboard Baseboard Management Controller and Remote Supervisor Adapter II adapter
- ▶ Three-year warranty on-site, 9 hours a day, 5 days a week, next business-day response

## 1.2 IBM XA-64e fourth-generation chipset

The x3950 M2 uses the fourth generation of the IBM XA-64e chipset. The architecture consists of the following components:

- ▶ One to four Xeon MP dual-core or quad-core processors
- ▶ Hurricane 4 Memory and I/O Controller (MIOC)
- ▶ Eight high-speed memory buffers
- ▶ Two PCI Express bridges
- ▶ One South bridge

Figure 1-2 shows a block diagram of the x3950 M2.

**Note:** The unit GB refers to gigabytes while the unit Gb refers to gigabits.

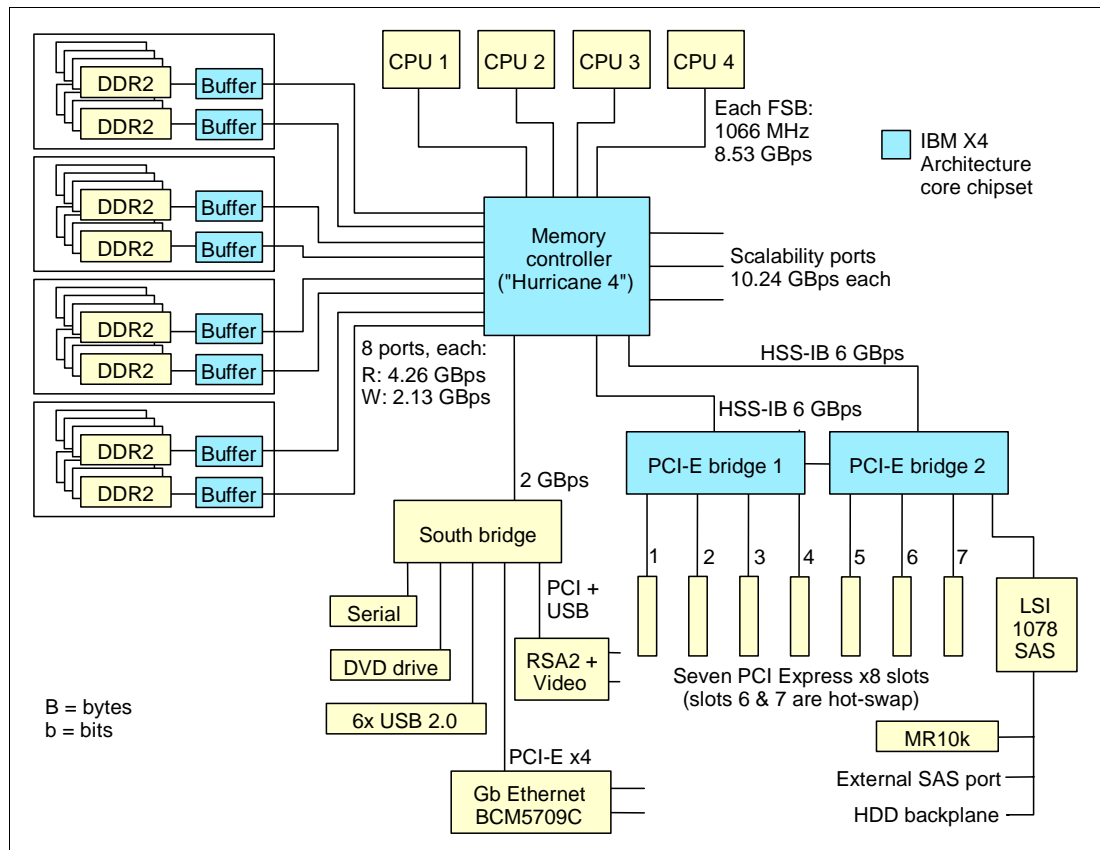


Figure 1-2 x3950 M2 system block diagram

Each memory port out of the memory controller has a peak read throughput of 4.26 GBps and a peak write throughput of 2.13 GBps. DIMMs are installed in matched pairs, two-way interleaving, to ensure that the memory port is fully utilized.

Because there are eight memory ports, spreading installed DIMMs over all four memory ports can improve performance. The eight independent memory ports provide simultaneous access to memory. With four memory cards installed, and eight DIMMs in each card, peak read memory bandwidth is 34.1 GBps and peak write bandwidth is 17.1 GBps.

The memory controller routes all traffic from the eight memory ports, four microprocessor ports, and the three PCI bridge ports. The memory controller also has embedded DRAM which, in the x3950 M2, holds a snoop filter lookup table. This filter ensures that snoop requests for cache lines go to the appropriate microprocessor bus and not all four of them, which improves performance.

Figure 1-2 on page 3 also shows that PCI bridge 1 supplies four of the seven PCI Express x8 slots on four independent PCI Express buses. PCI bridge 2 supplies the other three PCI Express x8 slots plus the onboard SAS devices, including the optional ServeRAID-MR10k. A separate South bridge supplies all the other onboard PCI devices like the USB ports, onboard Ethernet, and the standard RSA II.

## 1.3 Processors

The x3850 M2 and the x3950 M2 use either the Intel Xeon Processor E7210 dual-core, the Intel Xeon Processor 7300 series quad-core *Tigerton*, or the Intel MP Xeon Processor 7400 series six-core *Dunnington*. All models have two processors installed. One, two, three, or four processors are supported. Installed processors must be identical in model, speed, and cache size. You can connect multiple models of x3950 M2 to form larger configurations (see 1.5, “Multinode configurations” on page 9).

### Comparing the Tigerton and Dunnington processors

Figure 1-3 on page 5 compares the block diagrams of the quad-core and six-core processors.



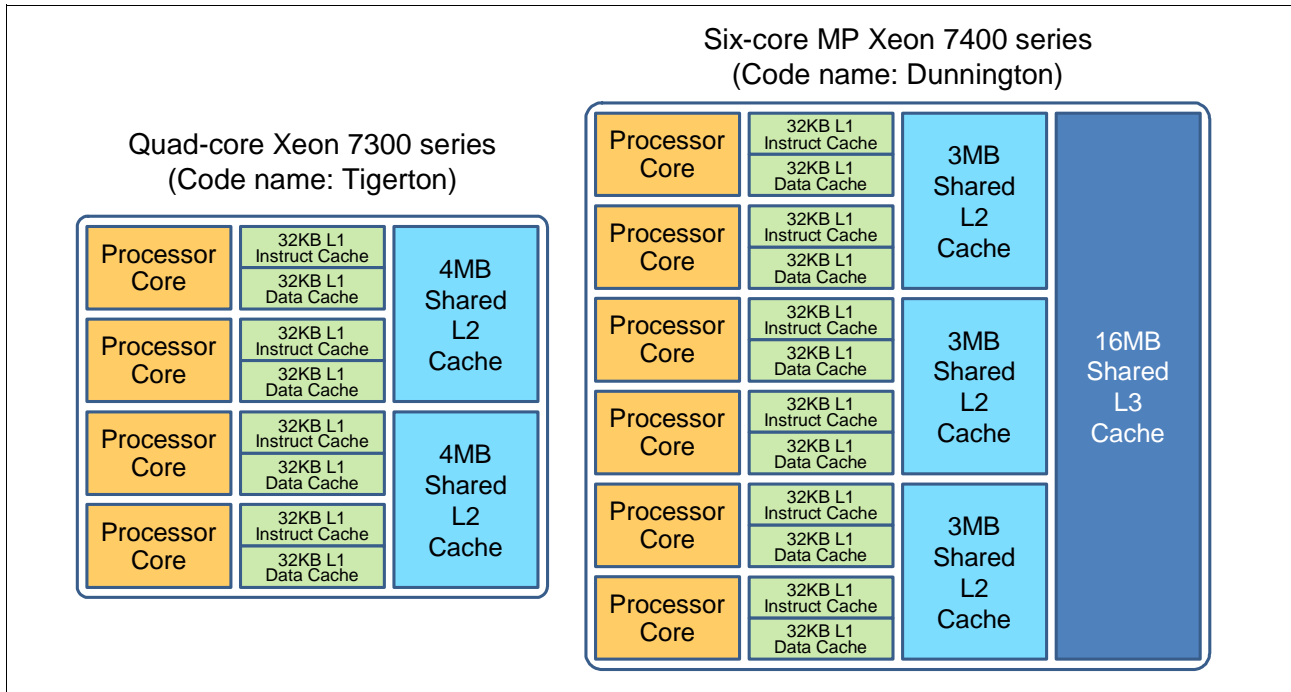


Figure 1-3 Comparing the quad-core and six-core processors

Each core has separate L1 instruction and data caches, execution units (integer, floating point, and so on), registers, issue ports, and pipelines for each core. A multi-core processor achieves fully parallel threads because these resources are not shared between the cores. The L2 cache is shared between pairs of cores. The Tigerton does not have an L3 cache.

The six-core Dunnington offers 43% greater performance, for a database workload running TPC-C, versus the similar quad-core Tigerton with the same power envelope. The 16MB L3 cache and the additional two cores per processor on the Dunnington make a significant contribution to this improved performance.

With double, quadruple, or six times the number of cores for the same number of sockets, it is critical that the memory subsystem meets the demand for data throughput. The 34.1 GBps peak throughput of the eX4 Architecture with four memory cards is well-suited to the demands of quad-core and six-core processors.

### 1066 MHz front-side bus

The Tigerton and Dunnington Xeon MP processors use two 266 MHz clocks, out of phase with each other by 90°, and use both edges of each clock to transmit data (Figure 1-4).

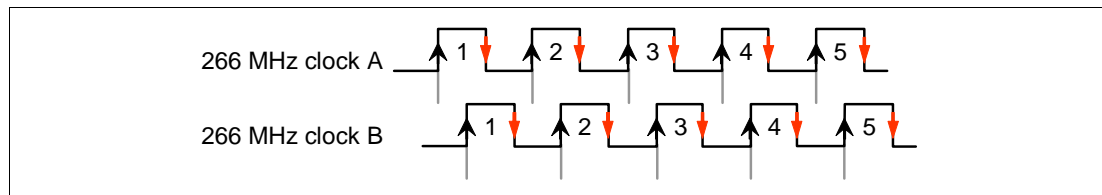


Figure 1-4 Quad-pumped front-side bus

A quad-pumped 266 MHz bus results in a 1066 MHz front-side bus. The bus is 8 bytes wide, which means it has an effective burst throughput of 8.53 GBps. This can have a substantial impact, especially on TCP/IP-based LAN traffic.

## Intel 64 Technology

First introduced in the Xeon DP *Nocona* processor, Intel 64 Technology is a 64-bit extension to the industry standard IA32 32-bit architecture. Intel 64 Technology adds:

- ▶ A set of new 64-bit general purpose registers (GPRs)
- ▶ 64-bit instruction pointers
- ▶ The ability to process data in 64-bit chunks

Even though the names of these extensions suggest that the improvements are simply in memory addressability, Intel 64 Technology is, in fact, a fully functional 64-bit processor.

There are three distinct operation modes available in Intel 64 Technology:

- ▶ 32-bit legacy mode

In this mode, processors with Intel 64 Technology act just like any other IA32-compatible processor. You can install your 32-bit operating system on such a system and run 32-bit applications, but you cannot make use of the new features, such as the flat memory addressing above 4 GB or the additional GPRs. Thirty-two-bit applications run just as fast as they would on any current 32-bit processor.

Most of the time, IA32 applications run even faster because there are numerous other improvements that boost performance regardless of the maximum address size.

- ▶ Compatibility mode

This is an intermediate mode of the full 64-bit mode. To run in compatibility mode, you must install a 64-bit operating system and 64-bit drivers to support both 32-bit applications and 64-bit applications.

With the compatibility mode, you can run a 64-bit operating system and still run unmodified 32-bit applications. Each 32-bit application is limited to a maximum of 4 GB of physical memory; however, the limit is imposed per-process and not system wide. This means that every 32-bit process has its own 4 GB of physical memory space, if sufficient physical memory is installed. This is an improvement over IA32, where the operating system kernel and the application had to share 4 GB of physical memory.

Additionally, the compatibility mode does not support the virtual 8086 mode, so real-mode applications are not supported. Sixteen-bit protected mode applications are, however.

- ▶ Full 64-bit mode

Intel refers to this mode as the *IA-32e mode*. For AMD™, it is the *long mode*. This mode is applied when a 64-bit OS and 64-bit application are used. In the full 64-bit operating mode, an application can have a virtual address space of up to 40 bits, equating to one terabyte (TB) of addressable memory. The amount of physical memory is determined by how many DIMM slots the server has and the maximum DIMM capacity supported and available at the time.

Applications that run in full 64-bit mode have access to the full physical memory range, depending on the OS, to the new GPRs, and to the expanded GPRs. However, it is important to understand that this mode of operation requires not only a 64-bit SO (and, of course, 64-bit drivers) but also a 64-bit application that has been recompiled to take full advantage of the various enhancements of the 64-bit addressing architecture.

For more information about the features of the latest Intel multicore processors, visit:

<http://www.intel.com/products/server/processors/index.htm?iid=process+server>

For more information about Intel 64, see:

<http://www.intel.com/technology/intel64/index.htm>

## 1.4 System memory

Memory is implemented in the x3950 M2 by memory cards. The server supports up to four memory cards. Each card has eight DIMM sockets, providing a total of up to 32. Using 8 GB DIMMs in every socket, the server can hold 256 GB of RAM. With four nodes connected together, this means a single system image has access to 1 TB of memory.

**Note:** For performance reasons, all nodes should have the same amount of memory.

Also, in a multinode scalable system, the XceL4v Dynamic Server Cache dynamically allocates 256 MB of main memory in each node for use as L4 cache. The result is a reduction in overall memory that is available to the operating system of 256 MB per node. Therefore, in a 16-way configuration there is a 1 GB reduction of main system memory.

In a multinode configuration, the memory in all nodes is combined to form a single coherent physical address space. Therefore, for any given region of physical memory, some processors are *closer* to it than other processors. Conversely, for any processor, some memory is considered *local* and other memory is *remote*. The partition descriptor table of the system is used to ensure optimal memory use.

The memory is two-way interleaved, meaning that memory DIMMs are installed in pairs. There are eight ports from the Hurricane 4 memory controller to memory, with each supporting up to 4.26 GBps read data transfers and 2.13 GBps write data transfers (see Figure 1-2 on page 3).

The DIMMs operate at 533 MHz, so that they are in sync with front-side bus. However, the DIMMs are 677 MHz PC2-5300 spec parts because they have better timing parameters than the 533 MHz equivalent. The memory throughput is 4.26 GBps, or 533 MHz x 8 bytes per memory port for a total of 34.1 GBps with four memory cards.

There are a number of advanced features in the x3950 M2 memory subsystem, collectively known as *Active Memory*:

- ▶ Memory ProteXion

The Memory ProteXion feature (also known as *redundant bit steering*) provides the equivalent of a hot-spare drive in a RAID array. It is based in the memory controller, and it enables the server to sense when a chip on a DIMM has failed and to route the data around the failed chip.

Normally, 128 bits out of every 144 are used for data and the remaining 16 bits are used for ECC functions. However, the x3950 M2 needs only 12 bits to perform the same ECC functions, leaving 4 bits free. If a chip failure on the DIMM is detected by memory scrubbing, the memory controller can reroute data around that failed chip through these spare bits.

It can do this automatically without issuing a Predictive Failure Analysis (PFA) or light path diagnostics administrator alert, although an event is logged to the service processor log. After the second DIMM failure, PFA and light path diagnostics alerts occur on that DIMM as normal.

- ▶ Memory scrubbing

Memory scrubbing is an automatic daily test of all the system memory that detects and reports memory errors that might be developing before they cause a server outage.

Memory scrubbing and Memory ProteXion work together and do not require memory mirroring to be enabled to work properly.

When a bit error is detected, memory scrubbing determines if it is recoverable or not. If it is, Memory ProteXion is enabled and the data that was stored in the damaged locations is rewritten to a new location. The error is reported for the purpose of preventative maintenance. As long as there are enough good locations for the proper operation of the server, no further action is taken other than recording the error in the error logs.

If the error is not recoverable, then memory scrubbing sends an error message to the light path diagnostics, which then turns on the proper lights and LEDs to guide you to the damaged DIMM. If memory mirroring is enabled, then the mirrored copy of the data from the damaged DIMM is used until the system is stopped and the DIMM replaced.

Because the x3950 M2 is now capable of supporting a large amount of memory, IBM has added the *Initialization Scrub Control* setting to the BIOS, so that customers can choose when this scrubbing is done and therefore potentially speed up the start process.

- ▶ Memory mirroring

Memory mirroring is roughly equivalent to RAID-1 in disk arrays, in that usable memory is halved and a second copy of data is written to the other half. If 8 GB is installed, then the operating system sees 4 GB once memory mirroring is enabled. It is disabled in the BIOS by default. Because all mirroring activities are handled by the hardware, memory mirroring is operating-system independent.

When memory mirroring is enabled, there are certain restrictions on placement and size of memory DIMMs and the placement and removal of memory cards.

- ▶ Chipkill memory

Chipkill is integrated into the XA-64e chipset, so it does not require special Chipkill DIMMs and is not visible to the operating system.

When Chipkill is combined with Memory ProteXion and Active Memory, the x3950 M2 provides high reliability in the memory subsystem. When a memory chip failure occurs, Memory ProteXion handles the rerouting of data around the failed component as described previously. However, if a further failure occurs, the Chipkill component in the memory controller reroutes data.

The memory controller provides memory protection similar to disk array striping with parity, writing the memory bits over multiple memory chips on the DIMM. It can reconstruct the missing bit from the failed chip and continue working as usual. One of these additional failures can be handled for each memory port for a total of eight Chipkill recoveries.

- ▶ Hot-add and hot-swap memory

The x3950 M2 supports the replacement of failed DIMMs while the server is still running. This hot-swap support works with memory mirroring. The server also supports adding additional memory while the server is running. Adding memory requires OS support. These two features are mutually exclusive. Hot-add requires that memory mirroring be disabled and hot-swap requires that memory mirroring be enabled.

In addition, to maintain the highest levels of system availability, if a memory error is detected during POST or memory configuration, the server can automatically disable the failing memory bank and continue operating with reduced memory capacity. You can manually re-enable the memory bank after the problem is corrected with the Setup menu in the BIOS.

Memory mirroring, Chipkill, and Memory ProteXion provide multiple levels of redundancy to the memory subsystem.

Combining Chipkill with Memory ProteXion allows up to two memory chip failures for each memory port on the x3950 M2, for a total of eight failures sustained. For example, the first failure detected by the Chipkill algorithm on each port does not generate a light path

diagnostics error because Memory ProteXion recovers from the problem automatically. Each memory port can then sustain a second chip failure without shutting down. As long as memory mirroring is enabled, the third chip failure on that port sends the alert and takes the DIMM offline, but keeps the system running out of the redundant memory bank.

## 1.5 Multinode configurations

A single x3950 M2 chassis is the initial base building block, or *node*, for a scalable system. At their most basic, these nodes are comprised of a 4-way SMP-capable system with processors, memory, and I/O devices.

You can form a multinode configuration by adding one or more x3950 M2 nodes to another. The following configurations are possible:

- ▶ A two-node complex that consists of two x3950 M2 nodes, with up to eight processors and up to 512 GB RAM
- ▶ A three-node complex that consists of three x3950 M2 nodes, up to 12 processors and up to 768 GB RAM
- ▶ A four-node complex that consists of four x3950 M2 nodes, with up to 16 processors and up to 1 TB RAM

Unlike the x3950 and xSeries® 460, there is no special modular expansion enclosure for the x3950 M2. The multinode configuration is simply formed by another x3950 M2 or an x3850 M2 with the ScaleXpander Option Kit.

The ScaleXpander Option Kit includes the cables necessary to join two nodes together plus a small component (Figure 1-5) that enables the formation of a multinode complex.



Figure 1-5 ScaleXpander Option Kit component to enable multinode configurations

**Note:** In this paper, when we refer to an x3950 M2, we mean either an x3950 M2 or an x3850 M2 with the ScaleXpander Option Kit.

## Multinode scalable systems

A scalable system consists of one, two, three, or four x3950 M2 nodes (Figure 1-6). Each node can have one, two, three, or four processors, although at least two is recommended for redundancy reasons.

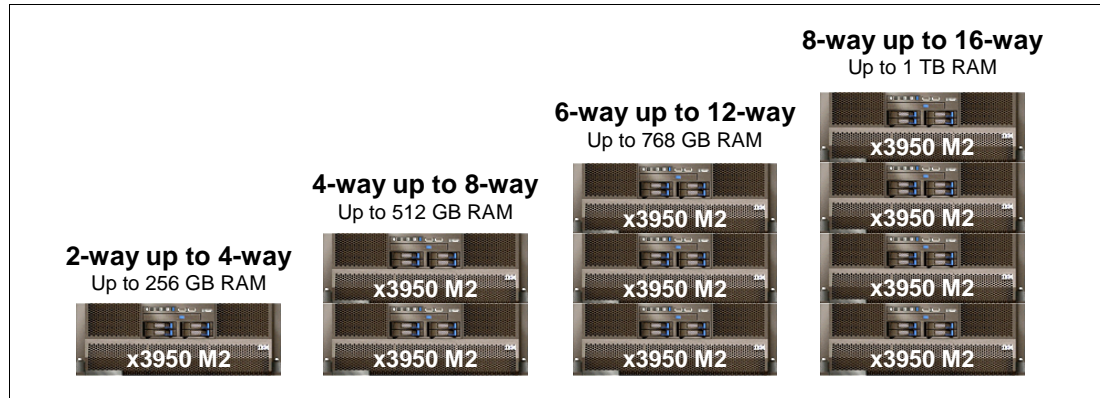


Figure 1-6 The multinode configurations supported

The following configuration rules apply:

- ▶ Combinations of x3950 M2s

You can connect one, two, three, or four x3950 M2 nodes. Specific cabling is required.

- ▶ Processors

Each node can have either one, two, three, or four processors. All processors must be the same speed and cache size. We recommend installing at least two processors in each node so that in the event of a processor failure the memory and I/O in that node will be available following automatic recovery.

- ▶ Memory

For performance reasons, you should have the same amount of memory in each node. A minimum of one memory card with two DIMMs totalling 4 GB is required in each node. This minimum configuration will severely limit performance. See 4.1.1, “System setup” on page 68.

- ▶ Firmware

All system firmware, including the system BIOS, BMC firmware, FPGA firmware, and RSA II firmware, must be at the same level for all systems. UpdateXpress System Pack is the preferred method for updating firmware (and drivers). See 4.1.2, “Firmware update” on page 68.

Disk drives in any of the x3950 M2s are seen by the OS as normal disk drives, and all PCI-Express slots and onboard Gigabit Ethernet ports in the x3950 M2 are visible.

A fully configured, four-node, scalable system with quad-core processors has 64 cores, 1TB of memory (using 8GB DIMMs), 28 PCI-Express adapters, 2.3 TB of raw disk space, and eight Gigabit Ethernet connections.

# Hardware partitioning

Figure 1-7 shows how a four-node x3950 M2 appears in the Scalable Partition Web interface tool.

The screenshot displays the Scalable Partition Web interface tool. On the left is a navigation menu with options like 'System Status', 'Power/Restart', 'Partition Control', 'Partition Configure', 'Reset Defaults', and 'Partition Reorder'. The main area is titled 'Scalable Complex Management' and shows four nodes, each with a 'Partition Control' panel (Start, Reset, Stop), a 'Standalone Boot' panel (Force, Undo), a 'Partition Configure' panel (Auto, Create, Custom, Delete), a 'Reset Defaults' panel (Reset), and a 'Partition Reorder' panel (Top System, dropdown menu, Redraw). Each node is represented by a blue box containing its ID, SN, and Primary status, with three ports (Port 1, Port 2, Port 3) shown below. A network diagram connects the ports of the four nodes. To the right is a table with columns 'System', 'Partition', and 'Mode'. A note at the bottom explains port status indicators: a red circle with an 'x' indicates a problem with the port or cabling, and a red system indicates it is unable to be scalable due to firmware or scalability key.

System	Partition	Mode
Stopped	Valid	Multinode
Stopped	Valid	Multinode
Stopped	Valid	Multinode
Stopped	Valid	Multinode

Note: When the port has a circle with a red x in it, it turns red, the cables turn red and the port name turns into a link, there is a problem with the port or the way the systems are cabled. If the system shows up red, the system is unable to be scalable due to firmware or scalability key.

Figure 1-7 The Scalable Partition Web interface tool

In Figure 1-7 on page 11 we show an expanded section of the RSA interface that is displayed by selecting **Server ♦ Scalable Partitioning ♦ Manage Partition(s)**.

Note the top node (Partition ID: 1 Chassis ID: 1) is marked as the primary node. The other three nodes are the secondary nodes. The black lines connecting the nodes indicate the connection paths for the scalability cables. Notice that every node is directly connected to every other node. This is optimal for performance. Assuming all processor sockets are populated, this complex will appear to the operating system as a single 16-way server.

**Note:** Instructions for using the Scalable Partition Web interface tool appear in the *IBM System x3850 M2 and System x3950 M2 Installation Guide*.

Without physically recabling this complex, the Scalable Partition Web interface tool can be used to create multiple partitions, which will each appear as separate servers to separate operating systems. For example:

- ▶ Two 2-node partitions could be created. In this case, two operating systems would be installed, one in each partition. Each operating system would see an 8-way server.
- ▶ A two-node partition and two single-node partitions. This would require three operating systems. The operating system running on the two-node system would see an 8-way server. Each operating system running on the single-node system would see a 4-way server.
- ▶ Four single-node partitions, which would require four operating systems, each running on a 4-way server.

## 1.6 Broadcom dual Gigabit Ethernet controller

The x3950 M2 offers a dual Gigabit Ethernet controller integrated standard. The x3950 M2 includes one dual-port Broadcom BCM5709C 10/100/1000 BASE-T MAC (Media Access Controller) on the PCI Express x4 bus. The BCM5709C has the following features:

- ▶ Full and half-duplex performance at all speeds (10/100/1000 Mbps, auto-negotiated)
- ▶ Two IEEE 802.3 Ethernet MAC addresses
- ▶ Integrated on-chip memory for buffering data transmissions
- ▶ Dual onboard DMA engines to maximize bus throughput and minimize CPU overhead
- ▶ IPMI support for system management

The Broadcom controller also includes software support for failover, layer-3 load balancing, and comprehensive diagnostics.

**Note:** Category 5 or better Ethernet cabling is required with RJ-45 connectors. If you plan to implement a Gigabit Ethernet connection, ensure that your network infrastructure is capable of the necessary throughput to match the I/O capacity of the server.

## 1.7 SAS disk subsystem

The x3950 M2 has a disk subsystem that is comprised of an LSI Logic 1078 Serial Attached SCSI (SAS) controller and four internal 2.5-in SAS hot-swap drive bays. The x3950 M2 supports internal RAID-0 and RAID-1. The optional ServeRAID-MR10k, part number 43W4280, provides additional RAID levels and a 256 MB battery backed cache.



SAS is the logical evolution of SCSI. SAS uses much smaller interconnections and offers longer cabling distances, smaller form factors, and greater addressability. The x3950 M2 has an external SAS x4 port that is used with the optional ServeRAID-MR10k. This port supports SAS non-RAID disk enclosures such as the EXP3000 and has an SFF-8088 connector.

## 1.8 PCI subsystem

There are five half-length full-height PCI Express x8 slots and two half-length full-height active PCI Express x8 slots internal to the x3950 M2 (see Figure 1-2 on page 3), and all are vacant in the standard models. All seven slots have the following characteristics:

- ▶ Separate bus from the other slots and devices
- ▶ PCI Express x8
- ▶ 40 Gbps full duplex
- ▶ 64-bit, each supporting 32-bit adapters as well

Slots 6 and 7 also support Active PCI hot-swap adapters. The optional ServeRAID-MR10k adapter does not use a PCI slot because it has a dedicated slot on the motherboard.

The PCI subsystem also supplies these I/O devices:

- ▶ LSI 1078 Serial-attached SCSI (SAS) controller
- ▶ Broadcom dual port 5709C 10/100/1000 Ethernet
- ▶ Six USB ports, two on the front panel, three on the rear, one onboard
- ▶ Remote Supervisor Adapter II adapter in a dedicated socket on the I/O board
- ▶ ATI™ ES1000™ 16MB video controller
- ▶ EIDE interface for the DVD-ROM drive
- ▶ Serial port
- ▶ Trusted Platform Module (TPM)

## 1.9 Redundancy

The x3950 M2 has the following redundancy features to maintain high availability:

- ▶ There are six hot-swap, multi-speed fans. These fans provide cooling redundancy and enable individual fan replacement without stopping the server. Each of the three groups of two fans is redundant. If a fan fails, the other fans speed up to continue to provide adequate cooling until the fan can be hot-swapped by the IT administrator. In general, failed fans should be replaced in 48 hours.
- ▶ The two Gigabit Ethernet ports can be configured as a team to form a redundant pair.
- ▶ The memory subsystem has a number of redundancy features, including memory mirroring and Memory ProteXion (see 1.4, “System memory” on page 7).
- ▶ Support is available for RAID disk arrays, both with the onboard LSI 1078 for RAID-0 and RAID-1. The optional ServeRAID-MR10k provides additional RAID features and a 256 MB battery backed cache. The x3950 M2 has four internal, hot-swap disk drive bays.
- ▶ The two, standard 1440 W hot-swap power supplies are redundant in all configurations at 220 V. At 110 V, the second power supply is not redundant.
- ▶ The scalability links between the nodes are redundant, with automatic failover between the dual cables, while running.





## SQL Server 2008

In this chapter, we discuss the enhancements and new features of the latest release of the Microsoft database management system, SQL Server 2008, as well as the reasons that the x3950 M2 is the perfect hardware platform on which to run it. Just as the x3950 M2 can run either 32-bit or 64-bit code, the SQL Server 2008 is available in both 32-bit and 64-bit editions. In the same way, as requirements for increased memory and processors grow, the x3950 M2 and SQL Server 2008 together can take full advantage of these additional resources, with excellent scalability.

It has been three years since Microsoft has introduced a major release of SQL Server. This new release, SQL Server 2008, has significant improvements in many of the features introduced by the previous version.

SQL Server 2008 also brings about new features on the following services: Database Engine, Analysis Services, Integration Services, Replication Services, Service Broker, and a new architecture design for Reporting Services service.

This chapter covers the following topics:

- ▶ 2.1, “SQL Server 2008 Editions” on page 16
- ▶ 2.2, “New and enhanced features of SQL Server 2008” on page 16
- ▶ 2.3, “64-bit computing and SQL Server 2008” on page 24
- ▶ 2.4, “Windows Server 2008 editions” on page 26

## 2.1 SQL Server 2008 Editions

There are seven editions of SQL Server 2008:

- ▶ SQL Server 2008 Enterprise Edition (32-bit and 64-bit)
- ▶ SQL Server 2008 Standard Edition (32-bit and 64-bit)
- ▶ SQL Server 2008 Workgroup Edition (32-bit and 64-bit)
- ▶ SQL Server 2008 Express Edition (32-bit and 64-bit)
- ▶ SQL Server 2008 Developer Edition (32-bit and 64-bit)
- ▶ SQL Server 2008 Web Edition (32-bit and 64-bit)
- ▶ SQL Server 2008 Compact Edition (32-bit only)

The Developer Edition contains all the features of the Enterprise Edition. The Developer Edition, however, is licensed for development and testing and not for production use. SQL Server 2008 Compact and Express Editions are available for free download. The latter is a replacement for Microsoft Desktop Engine (MSDE) and is optimal to learn and build small data-driven applications that are fast to deploy. Compact Edition is for stand-alone use and occasionally connected to applications for mobile devices, desktops, and Web clients.

The Enterprise Edition enables scale up to the levels of performance necessary to support the largest enterprise online transaction processing (OLTP), highly complex data analysis, data warehousing systems, and business intelligence applications. It contains comprehensive business intelligence as well as analytical capabilities and high availability features, such as failover clustering and database mirroring, all of which enable it to handle the most mission-critical enterprise workloads.

Enterprise Edition is the most comprehensive edition of SQL Server 2008 and is ideal for very large organizations and highly complex requirements. It is also available in a 120-day Evaluation Edition for the 32-bit or 64-bit platform.

For a detailed description of which features are supported in which edition, see *SQL Server 2008 Books Online* topic “Features Supported by the Editions of SQL Server 2008”, available from:

<http://msdn.microsoft.com/en-us/library/cc645993.aspx>

## 2.2 New and enhanced features of SQL Server 2008

In this section, we provide an overview of each of the major enhancements in SQL Server 2008.

### 2.2.1 SQL Server installation

SQL Server 2008 possess new setup structural design for installation, upgrades, maintenance, failover clustering, and command prompt installation. These new installation process enhancements are described in detail in Appendix A, “Unattended install” on page 87.

## 2.2.2 Database engine new and enhanced features

The database engine introduces enhancements in these areas:

- ▶ Availability
- ▶ Manageability
- ▶ Programmability
- ▶ Scalability and Performance
- ▶ Security

### Database mirroring

Database mirroring can be used to improve the availability of SQL Server databases. This is done by providing fast failover and automatic client redirection to a secondary server.

Compared to failover clustering, database mirroring has the following advantages:

- ▶ It keeps two copies of the database.
- ▶ It does not require specialized hardware.
- ▶ It simplifies setup and maintenance.

SQL Server 2008 improves log performance and also provides a new feature, automatic recovery from corrupted pages. We cover this in more detail in “Database mirroring” on page 60.

### Auditing

This feature comprises the capacity to audit different levels of activity and changes to the SQL Server system. With an easy to use GUI it is possible to define server or database audit specifications that can be stored either on a file or Windows® EventLog.

### Backup compression

SQL Server 2008 introduces a backup compression feature. Because a compressed backup is smaller than an uncompressed backup of the same database, compressing the backup requires less I/O and therefore it increases performance of backup significantly. On the other hand, backup compression increases CPU usage; however, with *Resource Governor* (see “Resource Governor” on page 19) this can be limited by creating a low-priority session. Only SQL Server 2008 Enterprise Edition can create a compressed database backup, but any SQL Server 2008 edition can restore a compressed database backup.

### Change Data Capture

Change Data Capture (CDC) records DML operations (such as INSERT, UPDATE, and DELETE) on one or more user tables and exposes the information using table-valued functions. Extract Transform Load (ETL) constitutes a good use of this feature since in an incremental manner it loads data in data warehouses or datamarts. From a technical viewpoint, CDC works by asynchronously reading the changes on source tables from the transaction log and inserting them into the tables defined for this feature.

### Change tracking

SQL Server 2008 provides a change tracking mechanism for applications. Once change tracking is configured for a table, any DML statement that affects a row in a table will cause that modification to be recorded. Values of the primary key column are the only information from the tracked table recorded with the change information. Those values identify the rows that have been changed. To obtain the latest data for those rows, an application can use the primary key column values to join the source table with the tracked table.

The difference between CDC and change tracking is that CDC captures historical change information by capturing both the fact that DML changes occur and the actual data that was changed, while in contrast, change tracking captures the fact that rows in a table were changed, but does not capture the data that was changed.

### **Data collector**

The data collector is a SQL Server component that runs continuously or on a user-defined schedule. It collects various sets of data and stores them in a relational database known as the management data warehouse. See “Management Data Warehouse” on page 83.

### **Detecting edition-related migration problems**

SQL Server 2008 introduces a new dynamic management view called *sys.dm\_db\_persisted\_sku\_features* to identify features that would prevent a database from being moved between different editions of SQL Server.

### **Events and performance counters**

New performance counters and trace events were introduced in order to track the usage of deprecated features. Additionally, DDL Triggers and Event Notifications were expanded to include numerous stored procedures that perform DDL-like operations.

### **Central Management Servers**

SQL Server 2008 introduces a new method for the administration of multiple instances by enabling designation of Central Management Servers. This feature enables an administrator to execute queries across different servers within a single Query window in SQL Server Management Studio. It is also useful to appraise policies across a group of servers (see “Policy-based management administration” on page 20).

### **Dynamic Management Views**

SQL Server 2008 introduces new Dynamic Management Views to present memory information.

- ▶ *sys.dm\_os\_memory\_brokers*

Memory brokers distributes memory allocations for SQL Server components in an equitable manner, based on projected usage. This dynamic management view provides information about memory brokers.

- ▶ *sys.dm\_os\_memory\_nodes*

SQL Server memory manager is used by allocations that are internal to SQL Server. Memory use from external components in the SQL server memory space can be indicated by tracking the difference between process memory counters from *sys.dm\_os\_process\_memory* and internal counters. Nodes are created per physical NUMA memory nodes. No allocations done directly through Windows memory allocation routines are tracked. This DMV provides information about memory allocations done only by using SQL Server memory manager interfaces.

- ▶ *sys.dm\_os\_nodes*

SQL OS creates node structures that imitate the hardware processor locality. Given that System x3950 is NUMA-aware, this DMV shows the configuration for the hardware NUMA nodes. These structures can be changed by using soft-NUMA. For more details on SQL OS see 3.3.1, “SQL Server Operating System” on page 35.

► `sys.dm_os_process_memory`

This DMV can be used to get memory allocations and during the performance hit by looking at `memory_utilization_percentage` & `page_fault_count` values are useful to see how memory is resourced.

► `sys.dm_os_sys_memory`

Returns memory information from the operating system. It returns the overall system state memory usage. Determining the overall system state is an important part of evaluating SQL Server memory usage.

## Optimize for ad hoc workload option

The *optimize for ad hoc workloads* option is a new server configuration option used to improve the efficiency of the plan cache for workloads that contain many single use ad hoc batches. Instead of storing the full compiled plan, whenever this option is set to 1, the Database Engine stores a small compiled plan stub in the plan cache when a batch is compiled for the first time. This technique reduces memory pressure because it prevents the plan cache from becoming filled with compiled plans that are not reused.

The compiled plan stub allows the Database Engine to recognize that this ad hoc batch has been compiled before but has only stored a compiled plan stub. When it is executed again, it removes the plan stub from the plan cache, and adds the full compiled plan to the cache.

The compiled stub plan is stored in the `sys.dm_exec_cached_plans` catalog view.

*Example 2-1 Returns the batch text of cached entries that are reused for ad hoc batches*

---

```
SELECT usecounts, cacheobjtype, objtype, text
FROM sys.dm_exec_cached_plans
CROSS APPLY sys.dm_exec_sql_text(plan_handle)
WHERE usecounts > 1
and objtype = 'Adhoc'
ORDER BY usecounts DESC;
GO
```

---

## Resource Governor

Resource Governor in SQL Server 2008 enables administrators to control the way in which CPU and memory resources are consumed within a single SQL Server instance.

Windows System Resource Manager (WSRM) only provided limited ability to control processes within Windows by permitting restrictions on what resources the process `sqlservr.exe` could consume. However, this affected every activity in the SQL Server instance.

Resource Governor is an important feature for SQL Server consolidation because administrators can ensure the best use of available SQL Server resources based on business requirements

## SQL Server Extended Events

SQL Server 2008 introduces SQL Server Extended Events, an event infrastructure for server systems. Extended Events enables opening windows into the run time of the host process by using events as trace points. Those events can then be aggregated in memory, sent to a file, or output to Event Tracing for Windows (ETW).

The possible scenarios where they can be used are:

- ▶ Troubleshooting the cause of working set trimming
- ▶ Troubleshooting excessive CPU usage
- ▶ Troubleshooting deadlocks
- ▶ Correlating request activity with ETW logs

### **Policy-based management administration**

Declarative Management Framework (DMF) is a policy-based management framework for the SQL Server Database Engine. Using policies, several tasks within the SQL Server Management Studio can be managed, specifically:

- ▶ System configuration
- ▶ Monitoring and preventive changes to the system
- ▶ Simplification of administration tasks
- ▶ Detection of compliance issues

### **Compressed storage of tables and indexes**

SQL Server 2008 supports row and page compression for both tables and indexes. The levels that can be attained include: a heap, a table that is stored as a clustered index, a nonclustered index, an indexed view. In the case of partitioned tables or indexes the compression option enables configuration for each partition.

SQL Server 2008 provides a stored procedure to estimate the compression savings. This stored procedure is called *sp\_estimate\_data\_compression\_savings*.

**Note:** You can find detailed information on the use of the stored procedure *sp\_estimate\_data\_compression\_savings* by visiting SQL Server 2008 Books Online at the following link:

<http://technet.microsoft.com/en-us/library/ms130214.aspx>

Once there, navigate through the following: **Database Engine** → **Technical Reference** → **Transact-SQL Reference** → **System Stored Procedures** → **Database Engine Stored Procedures** → **sp\_estimate\_data\_compression\_savings**

The previously mentioned features are only available in the SQL Server 2008 Enterprise and Developer editions.

**Note:** You can find detailed information on how to implement compression and the considerations of each case by visiting the SQL Server 2008 Books Online link and selecting: **Database Engine** → **Development** → **Designing and Implementing Structured Storage** → **Tables** → **Creating and Modifying Tables** → **Creating Compressed Tables and Indexes**

See also “Data compression” on page 78.

### **FILESTREAM storage**

FILESTREAM storage integrates SQL Server Database Engine with an NTFS system by storing BLOB as files on the file system. FILESTREAM uses the NT system cache for caching data. The SQL Server buffer pool is not used; for that reason, this memory is available for query processing.



## Security enhancements

Security enhancements in the Database Engine include:

- ▶ New encryption functions
- ▶ New feature for transparent data encryption
- ▶ Extensible key management
- ▶ DES algorithms:
  - TRIPLE\_DES\_3KEY
  - Clarification on DES Algorithms

## Scalability and performance enhancements

Scalability and performance enhancements in the Database Engine include filtered indexes and statistics, new table and query hints, and new query performance and processing features.

### 2.2.3 Analysis Services new and enhanced features

The following new and enhanced features are present in Analysis Services.

#### Analysis Services - Multidimensional Database

This latest release of Microsoft SQL Server Analysis Services introduces novel features and developments to the Multidimensional Database, including improvements to:

- ▶ Aggregation design
- ▶ Cube design
- ▶ Dimension design
- ▶ Backup and Restore

#### Analysis Services - Multidimensional Datamining

The latest release introduces new characteristics and enhancements to the Multidimensional Database, specifically those to enable:

- ▶ Creation of holdout test sets
- ▶ Filtering on model cases
- ▶ Cross-validation of multiple mining models
- ▶ Support for the data mining add-ins for Office 2007
- ▶ Enhancements to the Microsoft time series algorithm
- ▶ Drill through to structure cases and structure columns
- ▶ Aliasing mining model columns
- ▶ Querying the data mining schema rowsets
- ▶ Installation on the same server as SQL Server 2005 Analysis Services

### 2.2.4 Reporting services new architecture design

The most important change to Reporting Services (SSRS) is that it removes the dependency on Internet Information Services (IIS). To replace the functionality provided by IIS, SSRS now includes the following components:

- ▶ Native support for HTTP.sys and ASP.NET
- ▶ URL management for site and virtual directory names
- ▶ A new authentication layer
- ▶ Health monitoring through new memory management features

The new service for SSRS consolidates the following services into a single service: Report Server Web Service, Report Server Windows Service, and Report Manager.

## Service architecture

The Report Server Windows service is a consolidated set of applications that runs in a single process. SSRS stores the configuration for the entire service under files and databases, these being: ReportServer.config, ReportServerServices.exe.config, and the report server database.

Figure 2-1 shows the new architecture for SSRS service.

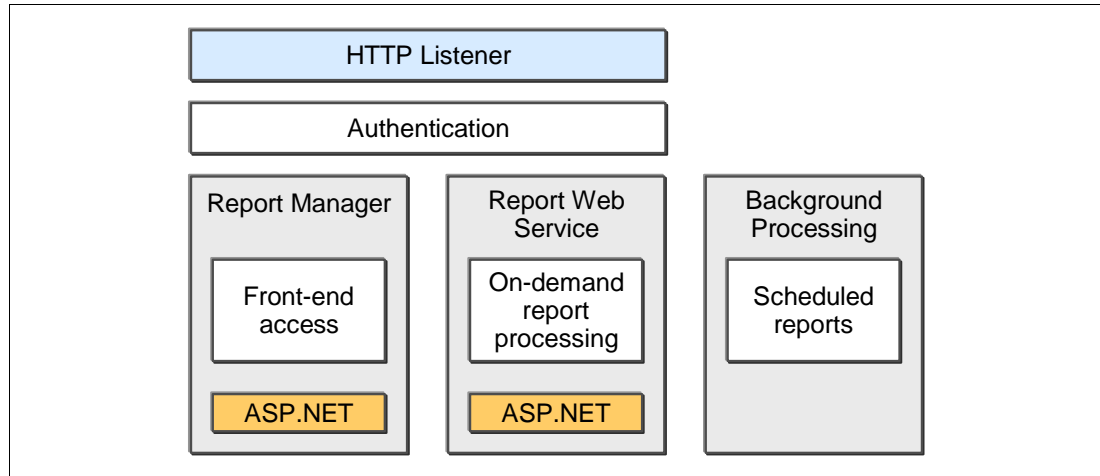


Figure 2-1 SSRS Service architecture

The service architecture layers are as follows:

- ▶ HTTP listener

SSRS includes an HTTP listener that monitors incoming requests directed to HTTP.sys. The hostname and the port are specified on a URL reservation that is configured on a case by case basis. Depending on the OS the port can be shared with other applications.

- ▶ Authentication layer

SSRS includes an authentication layer. By default it uses Windows integrated security and NTLM authentication. It also supports basic authentication, forms or custom authentication, and anonymous access.

- ▶ Report manager

Report manager is the client that provides Web front-end access for viewing and managing report server content and operations.

- ▶ Report Web service

The Report Web service is the engine for all on-demand report and model processing. To support interactive processing it begins by checking the authentication rules and then continues to handle the requirement.

- ▶ Background processing

Background processing manages all operations that run in the background and are initiated by the report server. Most of them involve scheduled report processing and subscription delivery, but also included are report server database maintenance tasks.

**Note:** Detailed information about the service architecture of Reporting Services is available from the SQL Server 2008 Books Online Web site at:

<http://technet.microsoft.com/en-us/library/ms130214.aspx>

Once there, select: **Reporting Services** → **Planning and Architecture** → **Service Architecture**

## 2.2.5 Additional enhancements and features

The enhancements mentioned in this section expand on existing functionality.

### Integration Services enhancements

Integration Services introduces improvements on the following areas:

- ▶ Component enhancements
  - New ADO.NET components
  - New data profiling task and data profile viewer
  - New Integration Services Connections project wizard to create a package that contains the connection information that is needed to connect to data sources and destinations.
  - New Script Environment
- ▶ Data management enhancements
  - Enhanced data type handling in the SQL Server import and export wizard
  - New date and time data types
  - Enhanced SQL statements to:
    - Perform multiple data manipulation language (DML) operations
    - Retrieve data about changes to a data source
    - Improve the performance of the bulk load operation when the data is sorted according to the clustered index on the table
- ▶ Performance and troubleshooting enhancements
  - Change data capture (see “Change Data Capture” on page 17)
  - New debug dump files

### Service Broker enhancements

SQL Server 2008 introduces new features for Service Broker in the following areas:

- ▶ Conversation priorities
- ▶ Diagnostic utility
- ▶ Service broker elements in object explorer
- ▶ System monitor object and counters

### Replication enhancements

SQL Server 2008 offers advances in peer-to-peer replication and Replication Monitor, and adds enhanced transaction replication for partitioned tables.

## 2.3 64-bit computing and SQL Server 2008

Intel and AMD both offer 64-bit extensions to the long-standing 32-bit x86 architecture. Collectively, these two technologies are referred to as “x64.” Internally, the architectures differ, but they can be considered comparable in what they offer. x64 offers full backwards compatibility for 32-bit applications. For scalability reasons, IBM has chosen to use the Intel 64-bit processor technology in servers such as the IBM System x3950 M2.

Intel also manufactures the Itanium® 2 processor, which is built on an entirely different 64-bit architecture. Itanium 2 is not backwards compatible with x86 applications or cross-compatible with x64 applications. This is because it implements a new instruction set called EPIC, which was designed to be the next-generation processor architecture for mainframe class scalability. However, its adoption has not been widespread and it is not a mass-market technology.

The following sections look at the limitations that are imposed by 32-bit hardware and software, and how 64-bit hardware and software addresses those limitations.

### 2.3.1 Windows, SQL Server, and 32-bit versus 64-bit

SQL Server 2008 is available in three versions: as a 32-bit version, EM64T 64-bit (x64), and Itanium 64-bit versions; however, the IBM System x3950 M2 only runs the 32-bit and x64 versions. Running a 64-bit version of SQL Server also requires a matching version of the operating system.

Table 2-1 shows the combinations of Windows Server® 2008 and SQL Server that run on the IBM System x3950.

*Table 2-1 Valid combinations of Windows Server and SQL Server running on System x3950 M2*

	Windows Server 2008, 32-bit	Windows Server 2008, x64
SQL Server 2005 32-bit (SP2)	Supported	Supported
SQL Server 2005 x64 (SP2)	Not valid	Supported
SQL Server 2008 32-bit	Supported	Supported
SQL Server 2008 x64	Not valid	Supported

SQL Server 2005 must have Service Pack 2 to run on Windows Server 2008.

Itanium versions of Windows and SQL Server are not supported on the x3950 M2

There are three combinations for running 32-bit and x64 versions of both Windows Server 2008 and 5SQL Server 2005:

- ▶ Both 32-bit
- ▶ x64 Windows and 32-bit SQL Server
- ▶ x64 Windows and x64 SQL Server

#### **Windows and SQL Server, both 32-bit**

When 32-bit Windows Server 2008 is run, either 32-bit SQL Server 2005 or 32-bit SQL Server 2008 can be run. All processes run in 32-bit, so there is no 64-bit option for SQL Server in this case.

With 4 GB of RAM, the /3GB parameter boot.ini can be used to switch the split to 1 GB for the kernel and 3 GB for the user portion of the Virtual Address Space (VAS).

If more than 4 GB of physical memory is installed, 32-bit SQL Server can use it as the database buffer pool. However, Physical Address Extension (PAE) must be enabled in Windows (add the /PAE switch to boot.ini) as well as Address Windowing Extensions (AWE) in SQL Server (using `sp_configure`).

**Note:** For systems with more than 16 GB of RAM, you cannot enable both /3GB and /PAE. See Section 9.11 of the Redbooks publication *Tuning IBM @server xSeries Servers for Performance*, SG24-5287 for details.

The following must all fit in the 2 GB user mode portion of the process address space: the entire SQL Server memory objects, binary code, data buffers, database page headers (512 MB for 64 GB of database buffers), sort area, connections, stored procedure caches, and open cursors. Basically, everything but the unmapped cached database pages.

To access the database buffer pool pages above the 4 GB line, SQL Server must map them into the address space below the 4 GB line (Figure 2-2). This mapping incurs some performance overhead.

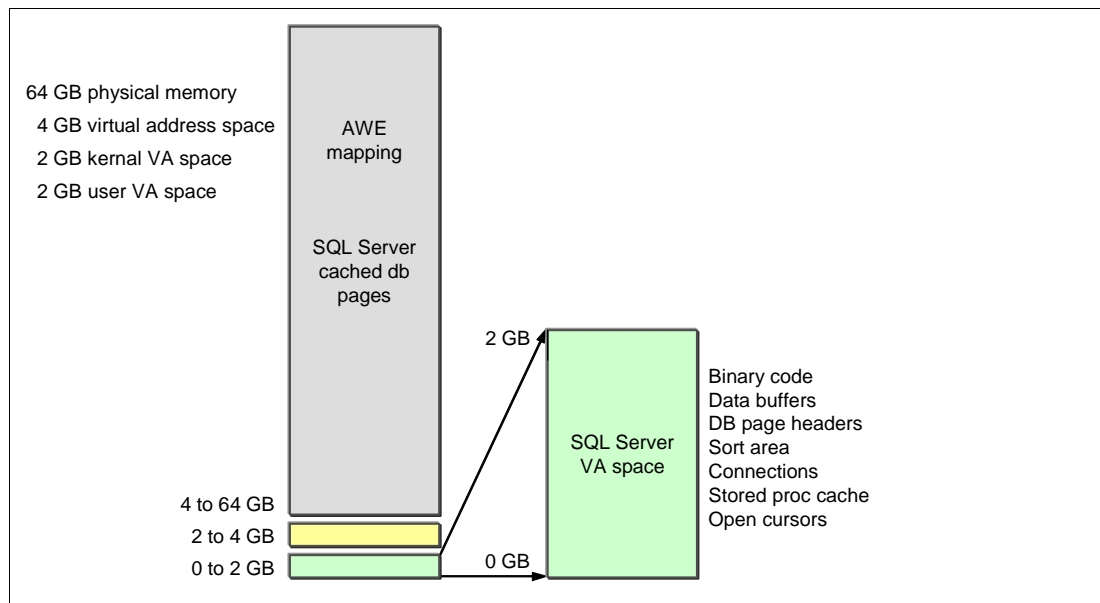


Figure 2-2 SQL Server 2008 uses AWE memory only for buffer pool pages

Whenever a SQL Server workload is constrained by the 2 GB user mode limit for reasons other than requiring more database buffer pages, it will not benefit from having more than 4 GB of physical memory. This is a hardware architectural bottleneck, which is relieved by using 64-bit hardware and software.

SQL Server 2008 32-bit experiences the same hardware architectural bottleneck. SQL Server 2008 manages all of its memory (including the AWE memory) dynamically, releasing and allocating memory in response to internal and external memory pressure.

## Windows Server x64 and SQL Server 32-bit

With a Windows Server 2008, x64 Edition installation, you can run either SQL Server 2005 with Service Pack 2 or 32-bit SQL Server 2008.

SQL Server 2008 runs in WOW64 (Windows on Windows) in a user mode address space of 4 GB. Unlike running the same application on 32-bit Windows, SQL Server 2008 has a full 4 GB of user mode address space given that the 64-bit kernel runs in its own address space. This possibility provides some relief for SQL Server workloads that are constrained by the 2 GB user mode address space of a 32-bit operating system.

SQL Server 2008 can also be configured to use AWE to access up to 64 GB of physical memory on x64 systems. The AWE memory above 4 GB must still be mapped into the lower 4 GB user mode address space to be used.

SQL Server 2008 32-bit benefits in the same way that SQL Server 2005 does with a 4 GB user mode address space that is not shared with the operating system. With AWE enabled, SQL Server 2008 can access up to 64 GB of physical memory on the x3950 M2. This results from the fact that maximum memory for a process running in the WOW64 on Intel x64 is 64 GB. SQL Server 2008 will manage all of its memory (including the AWE memory) dynamically, releasing and allocating memory in response to internal and external memory pressure.

### **Windows and SQL Server 2008, both 64-bit**

When Windows Server 2008 x64 is run, the x64 version of SQL Server 2008 can be installed.

SQL Server 2008 64-bit enjoys the same memory addressability as the 64-bit operating system. The 64-bit user mode address space is not limited to 4 GB, and it can use memory up to the operating system maximum for any purpose, not just for database buffers. AWE mapping is not required because the memory model is flat under 64-bit addressing. This provides the most efficient utilization of resources for SQL Server 2008.

See the Redpaper *Introducing Windows Server x64 on IBM @server xSeries Servers*, REDP-3982 for more information about 32-bit and 64-bit memory addressing.

<http://www.redbooks.ibm.com/abstracts/redp3982.html>

## **2.4 Windows Server 2008 editions**

In this section, we compare the editions of Windows Server 2008, discuss the Datacenter edition, and identify the processor and memory limits.

### **2.4.1 Comparing Windows Server 2008 editions**

Microsoft offers five editions of Windows Server 2008: Web, Standard, Enterprise, Datacenter and Itanium editions. For up to date technical specifications and system configuration limits among the editions, see the “Compare Technical Features and Specifications” page for Windows Server 2008 at the Microsoft web site:

<http://www.microsoft.com/windowsserver2008/en/us/compare-specs.aspx>

Windows Server 2008, Datacenter Edition can provide server clustering and scaling with respect to memory and processors that is beyond any other current or previous version of Windows Server. The Datacenter Edition is also part of a program designed to provide the highest level of availability and support.

Enterprise Edition and Datacenter Edition are similar in most aspects. But, you should pay close attention to the differences in high availability and image virtualization features between Enterprise and Datacenter editions if you need to support very large, “scale up” workloads.

## 2.4.2 Windows Datacenter models

IBM offers Microsoft Windows Server 2008, Datacenter Edition on the x3950 M2, either the 32-bit version or the 64-bit version.

There are two x3950 M2 Datacenter offerings:

### ***Datacenter Unlimited Virtualization***

Datacenter Unlimited Virtualization is optimized for customers requiring the highest level of scalability, reliability and virtualization. This offering supports stand-alone server solutions that scale from 2-sockets to 32-sockets. Datacenter Edition Unlimited Virtualization is an ideal solution for clients who require high performance and scalability in an SMP server environment. This end-to-end offering comes as a fully configured, certified, pre-installed system.

### ***Datacenter Unlimited Virtualization with High Availability***

The Datacenter Unlimited Virtualization with High Availability Program consists of specific IBM Datacenter UVHA offerings that have been both IBM ServerProven® and Microsoft Cluster Certified. Customer-tailored service offerings deliver worry-free installation and provide the support you need to deliver high uptime for your business-critical applications.

## 2.4.3 Processor and memory limits

Windows Server 2008 x64 Edition supports a significant amount of physical memory. The number of processors that are supported has also increased in Windows Server 2008, Datacenter Edition. The amount of physical memory that is supported for the Enterprise and Datacenter editions is greater than the amount of memory that can be installed in servers today.

The maximum configuration of the x3950 M2 is 16-way (four chassis with four processor sockets) and 1 TB of physical memory (using 8 GB DIMMs).

Microsoft licensing is based on physical processors (sockets) and not on the number of cores per processor.

Therefore, if an x3950 M2 is fitted with four quad-core or six-core Xeon processors, a license for Windows Server 2008, Standard Edition is sufficient. This is true even though the system contains 16 or 24 processor cores and the Windows operating system sees 16 or 24 processors.







## Platform synergy

Both the x3950 M2 and SQL Server 2008 scale very well to 16 processors (64 cores on quad-core sockets) and 1 TB of RAM. This type of scalability, where all system resources across the multi-node complex (processors, memory, and PCI resources) are seen as one unified hardware platform for the operating system and applications, is known as *scale-up*.

The key to effective scalability is *affinity*, that is, linking system resources that are in the same node to maximize performance. For example, for the best performance, any memory operations that a processor executes should involve RAM that is installed in the same node as the processor. Fetching memory from another node can impact performance.

This chapter describes scalability and affinity in more detail. It also explains how scalability and performance can become complicated in a server environment with hundreds, if not thousands, of supported users. In a high-end server, such as the x3950 M2, many workloads occur simultaneously on the server. Thus, in this environment, it is important to distribute server resources optimally. When affinity is not configured properly, contention of resources might occur, which often leads to serious performance degradation.

There are a number of parameters in SQL Server and Windows Server 2008 that are related to scalability and affinity. When these Microsoft products are combined with the x3950 M2, the solution demonstrates high scalability.

This chapter covers the following topics:

- ▶ 3.1, “Scalable hardware implementation” on page 30
- ▶ 3.2, “Affinity in Windows Server 2008” on page 32
- ▶ 3.3, “Affinity in SQL Server 2008” on page 35
- ▶ 3.4, “Server consolidation” on page 49

## 3.1 Scalable hardware implementation

In the standard single chassis four-way configuration, the x3950 M2 acts as an industry standard SMP system. Each processor has equal access to all system resources.

In an SMP environment, the concentration of memory access to the memory controller is an obstacle for scalability. The memory controller is a core component that manages I/O to and from memory. With SMP configurations, when the processors are added, the number of memory controllers does not change. As the number of transactions increases, so do requests to the memory controller, which causes a significant bottleneck. As a result, SMP is less efficient the more processors you have.

However, with multi-node x3950 M2 configurations, a NUMA-like architecture (non-uniform memory architecture) is implemented by connecting the scalability ports of each node together (see Figure 1-2 on page 3). These ports are directly connected to the memory controller in each node and allow high speed communication between processors located in different nodes. The ports act as though they were hardware extensions to the CPU local buses. They direct read and write requests to the appropriate memory or I/O resources, and they also maintain cache coherency between the processors.

The term *NUMA* is not completely correct because not only can memory be accessed in a non-uniform manner, but also I/O resources. PCI-e and USB devices might be associated with nodes. The exception to this are I/O devices such as diskette and CD-ROM drives that are disabled because the classic PC architecture precludes multiple copies of these items.

The main difference between PCI-X and PCI-e is that PCI-X, due to backward compatibility requirements, runs at the speed of the slowest device. All PCI-X devices must be able to run at lower speed to work in older systems. Because the data is transferred in parallel, either 32 or 64 bits at a time, this means that PCI-X 64-bit/133Mhz = 1GB/s.

On the other hand, PCI Express (PCI-e) is an entirely new bus architecture. It performs serial data transfers. Transfers are bi-directional, so data can flow to and from devices simultaneously. Since data is switched, more than 1 device can be transferring at the same time. So because PCI-e is a point-to-point serial connection, total bus bandwidth is no longer an issue. In System x3950 M2 PCI-e is specified as x8. This specifies the number of lanes that the slot offers. A *lane* is a bi-directional serial channel capable of around 250MB/s in each direction, so an x8 slot should be 2GB/s in each direction (total 4GB/s).

The key to effective scalability is to add memory controllers as you add nodes. With the four-way x3950 M2, there is one memory controller; the 16-way has four memory controllers.

In multi-node x3950 M2 configurations, the physical memory in each node is combined to form a single coherent physical address space. The result is a system where, for any given region of physical memory, some processors are closer to it than other processors. Conversely, for any processor, some memory is considered local and other memory is remote.

Memory can be described in one of two ways, depending on the relation to a given processor:

- ▶ *Local memory*: Memory is in the same node as the processor.
- ▶ *Remote memory*: Memory is installed in another node that is directly connected by scalability cables.

Figure 3-1 on page 31 shows an example of a local memory access (CPU 1) and a remote memory access (CPU 6).

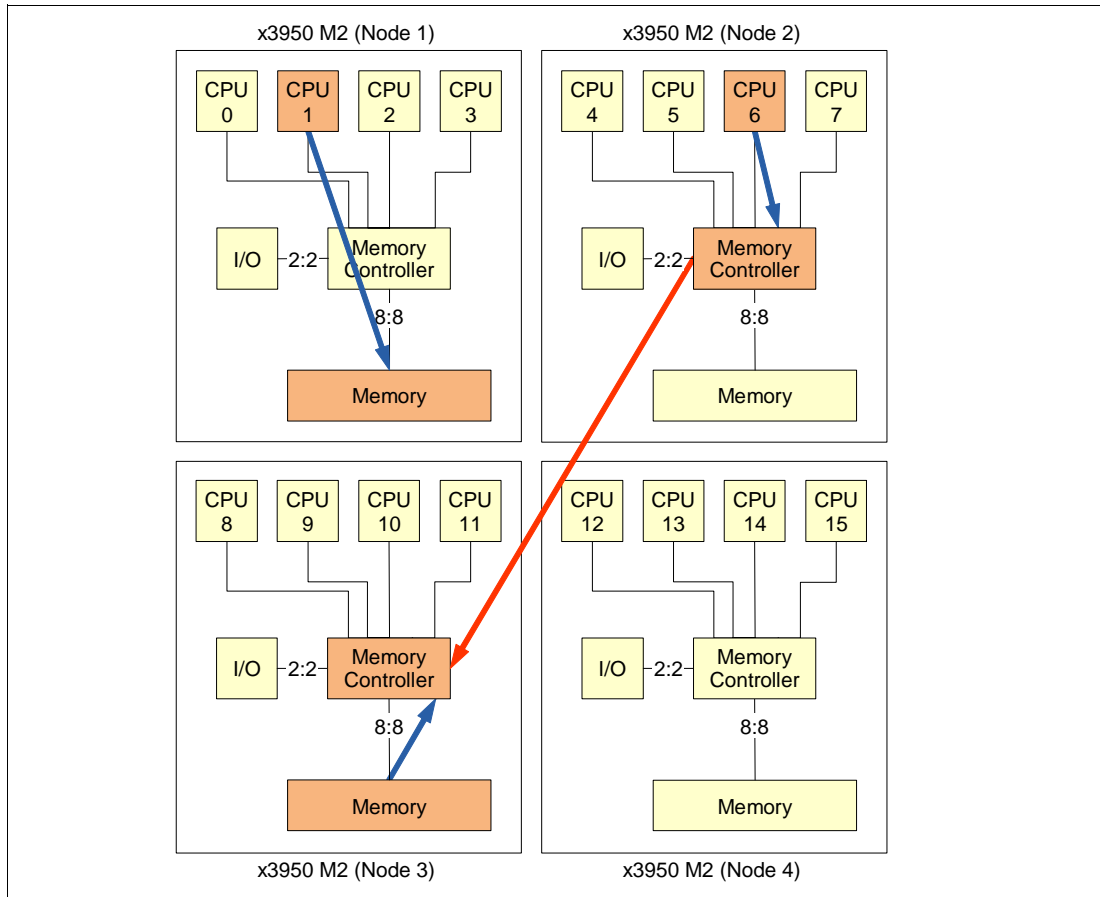


Figure 3-1 Local memory access and remote memory access

**Note:** Figure 3-1 shows a conceptual diagram of a four-node NUMA configuration on System x3950 M2. As shown in Figure 1-2 on page 3, the Memory Controller has 2 buses connecting to PCIe and 8 buses connecting to physical memory.

There are three important aspects that are associated with memory subsystem performance in this type of configuration: *memory latency*, *queuing time*, and *snooping*.

► Memory latency

Remote memory access takes more time than local memory access because of the travel time through the interconnects (one interconnect for a remote memory access). The delay incurred as a result of this long path is called *latency*. Remote access can decrease performance when compared with local memory access. So, to increase the performance of servers in a NUMA environment, the operating system and application must strive to limit the use of remote memory as much as possible. This concept is known as *affinity*.

Affinity in the x3950 M2 means that local resources (CPU, memory, PCI devices) are grouped together and that the operating system recognizes these groups and attempts to limit the resources used by a thread to just those in a group.

However, while affinity is important, this focus on latency misses the actual problem that NUMA is attempting to solve: shorten memory request queues.

- ▶ Memory queuing time

Memory latency is only part of the picture, but is often the aspect that receives the most focus. As with a large SMP-based design, the more processors that access the same memory controller and memory, the greater the potential for poor performance. The bottleneck then becomes the number of access operations in the queue to be actioned.

Another way to think about this is to consider the analogy of shopping at a local supermarket. Directly in front of you is a check-out lane with 10 customers standing in line but 20 meters to your left is another check-out lane with only two customers standing in line. Which would you go to? The check-out lane closest to your position has the lowest latency because you do not have far to travel. But the check-out lane 20 meters away has much greater latency because you have to walk that far.

Clearly most people would walk the 20 meters, incurring a delay, to arrive at a check-out lane with only two customers instead of 10. We think this way because our experience tells us that the time waiting to check-out with 10 people ahead of us (the request queue) is far longer than the time needed to walk to the “remote” check-out lane (latency) and wait for only two people ahead.

This analogy clearly communicates the performance effects of queuing time versus latency. In a computer server, with many concurrent outstanding memory requests, we would gladly incur some additional latency (walking) to spread memory transactions (check-out process) across multiple memory controllers (check-out lanes) because this greatly improves performance by reducing the queuing time.

- ▶ Snooping

In a traditional SMP design, before a piece of data is retrieved from memory, all caches that are local to every processor must be queried to determine whether that data is stored in a cache and whether it has been modified by another process running in that processor. For a small SMP, this *snooping* procedure (called the MESI protocol, for the four states that data can be in: modified, exclusive, shared, invalid) is sufficiently efficient. However, as you add more processors, the overhead becomes overwhelming.

The solution implemented in the x3950 M2 is a special fourth level of cache, the XceL4v cache, that maintains a table of all contents and eliminates the need to query every processor. In addition, for multi-node x3950 M2 configurations, the XceL4v acts as a cache to reduce latency across the scalability cables.

It is important to note that the traffic across the scalability cables is not only memory access. As you can see in Figure 3-1 on page 31, each node also has local PCIe slots that can have devices such as FC adapters, Ethernet controllers, and the like. The operating system and application must maximize overall performance for these devices also.

## 3.2 Affinity in Windows Server 2008

This section discusses how process, scheduling, and memory management are optimized for the x3950 M2 scalable architecture and Windows Server 2008. It also describes the relevant parameters in Windows Server 2008.

### 3.2.1 NUMA optimization for Windows Server 2008

Most editions of Windows Server 2008 are optimized for NUMA as listed in Table 3-1 on page 33. In general, when we discuss Windows Server 2008 in this section, we mean Windows Server 2008 editions that are optimized for NUMA.

Table 3-1 Versions of Windows Server optimized for NUMA

	x86 (32-bit)	x64 (64-bit)	IA64 (64-bit)
Windows 2008 Web Edition	No	Not applicable	Not applicable
Windows Server 2008 Standard Edition	No	No	Not applicable
Windows Server 2008 Enterprise Edition	Yes	Yes	Yes
Windows Server 2008 Datacenter Edition	Yes	Yes	Yes

Windows Server 2008 obtains the NUMA information from the SRAT table in the system BIOS while booting. That is, NUMA architecture servers must have the SRAT table and the x3950 M2 to use this function. Windows Server 2008 cannot recognize system topology without the SRAT table.

At startup, Windows Server 2008 builds a graph of the NUMA node access cost, that is, the relative distance (memory latency). The system uses the information on the SRAT table to determine the ideal node from which to obtain the resources. If adequate resources are not available on the ideal node, the operating system consults the table for the next best choice.

Windows Server 2008 incorporates further NUMA enhancements for paging. The operating system prefetches pages to the application's ideal node and migrates pages to the ideal node when a soft page fault occurs. A soft page fault occurs when the system finds the requested page elsewhere in memory, whereas a hard page fault requires reading the page in from disk.

You can find more information on the implementation of ACPI of Windows Server 2008 from:

<http://www.microsoft.com/whdc/default.mspx>

**Note:** See “Kernel Enhancements for Windows Vista® and Windows Server Longhorn” and “Advances in Memory Management for Windows” papers on the Windows Hardware Developer Central Site for information about ACPI implementation and NUMA enhancements on Windows Server 2008.

### 3.2.2 Process and thread scheduling

Process scheduling is optimized in the NUMA environment. When a new process is created, Windows Server 2008 uses a round-robin algorithm to assign it to the next NUMA node in the system. Each process has a local node and default processor that belong to it.

Windows Server 2008 ensures that whenever possible the application runs on the ideal processor and that any memory that the application allocates comes from the ideal processor's node. If the ideal node is unavailable or its resources are exhausted, the system uses the next best choice.

Earlier Windows releases allocated the memory from the current processor's node, and if its resources were exhausted, from any node. Thus, in earlier Windows releases, a process that ran briefly on a non-optimal node could have long-lasting memory allocations on that node, causing slower, less-efficient operation.

With Windows Server 2008 defaults, the system allocates the memory on the ideal node even if the process is currently running on some other node. If there is not available memory on the ideal node it would use the SRAT table for the closest to the ideal instead.

This type of resource allocation policy increases the possibility that a process and all its resources are on the same node or on the most optimal alternative.

The thread scheduler for Windows Server 2008 is also optimized for NUMA. The operating system schedules the ideal processor based on priority, or, if that processor is not available, it schedules the thread to the closest processor in the local node. If all processors in the local node are unavailable, the operating system schedules the thread to processors in the closest node to the ideal node.

This is sometimes referred to as *soft processor affinity* to contrast with the *hard processor affinity* you can configure with Windows System Resource Manager or within SQL Server 2008.

One of the most important differences between Windows Server 2003 and Windows Server 2008 is that the more recent version does differentiate between distances. This is probably the most important enhancement on scheduling in NUMA systems.

You can prevent Windows Server 2008 from scheduling certain processors (for example, non-local processors) by using *hard processor affinity*. Hard affinity is the function of binding specific CPUs to a process and is available with the following tools:

- ▶ Affinity option in applications such as SQL Server 2008

In SQL Server 2008, you can choose which processors can be used by the product by changing the affinity mask. By default, SQL Server 2008 can use all processors in the system.

See 3.3, “Affinity in SQL Server 2008” on page 35 for more detail.

- ▶ Windows System Resource Manager (WSRM)

WSRM is a tool that provides resource management of processors and memory resources. With WSRM, you can specify which CPUs you want each process to run on.

You can install WSRM from the Windows Server 2008 Manager. WSRM is included with Windows Server 2008 Enterprise Edition and Datacenter Edition and is covered by the license for these products.

In both cases, the editions of Windows Server 2008 that can install WSRM are listed in Table 3-2.

Table 3-2 Windows Server editions that include WSRM

	x86 (32-bit)	x64 (64-bit)	IA64 (64-bit)
Windows Server 2008 Standard Edition	No	No	No
Windows Server 2008 Enterprise Edition	Yes	Yes	Yes
Windows Server 2008 Datacenter Edition	Yes	Yes	Yes

Normally you should not use WSRM on a system that is running SQL Server 2008 in a production environment because the two schedulers (WSRM and the CPU Affinity Mask in SQL Server) might cause conflicts and system degradation. However, there are situations where you might want to use both. For example, if you wish to share the x3950 M2 between SQL Server and Internet Information Server (IIS), then you could use WSRM to restrict IIS to only processors 0 and 1, for example, and configure SQL Server to be restricted to just processors 2 through 7. It is important to ensure that no processors are available to both schedulers because this causes contention problems.

WSRM is recommended to allocate CPU resources between different database instances or other components as Analysis Services.

WSRM is best used when more than one service that consumes a significant amount of resources is installed on the same computer. These services could include any combination

of SQL Server, Analysis Services, Reporting Services, and Integration Services, your own services, and third-party services. It is best to use the built in controls for processor affinity for SQL Server. WSRM would also be used to control the memory consumption for Reporting Services and Integration Services since Analysis Services and SQL Server have their own controls.

### 3.3 Affinity in SQL Server 2008

This section describes the affinity options with SQL Server 2008 and NUMA optimization for SQL Server 2008.

#### 3.3.1 SQL Server Operating System

SQL Server 2008 maintains the architecture previously released on SQL Server 2005 of SQL Server Operating System (SQLOS) acting as a layer between SQL Server and the operating system. SQLOS is a user-level, highly configurable operating system with a powerful API that enables automatic locality and advanced parallelism.

SQLOS attempts to hide the complexity of the underlying hardware from high-level programmers. It also provides a comprehensive set of features to programmers who are willing to take advantage of the hardware underneath the system. SQLOS services include non-preemptive scheduling, memory management, deadlock detection, exception handling, hosting for external components such as CLR, and other services.

SQLOS has a hierarchical architecture, which means that SQLOS changes its structure based on the hardware platform that SQL Server 2008 is running on. Figure 3-2 illustrates SQLOS on an single node based server.

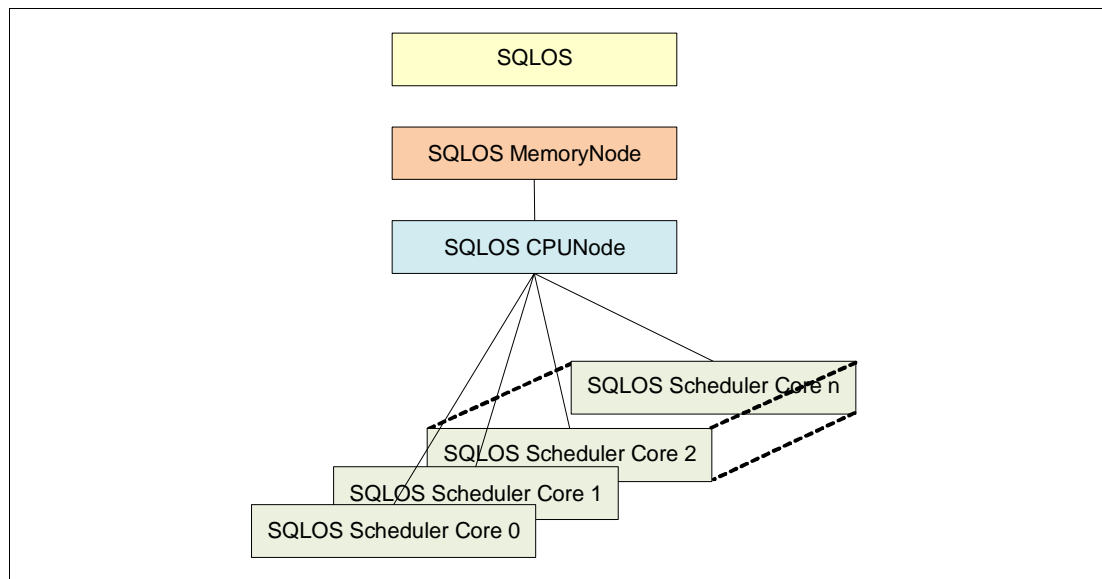


Figure 3-2 SQLOS on a one node server

Figure 3-3 on page 36 illustrates SQLOS on a two-node NUMA server such as a two-node (8-way) x3950 M2.

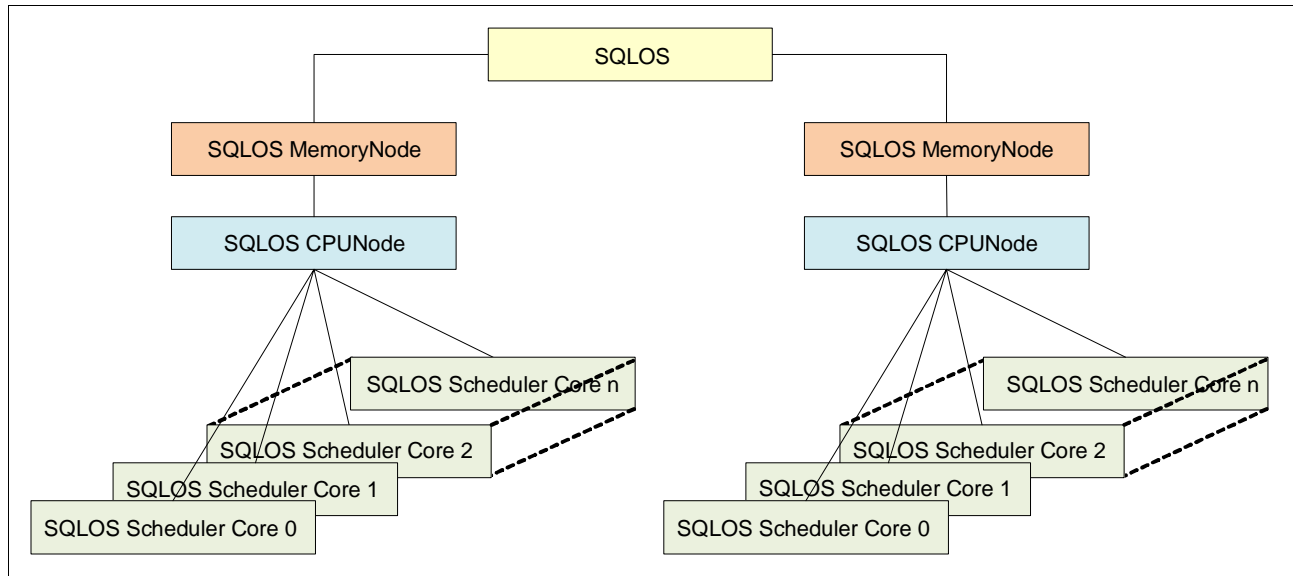


Figure 3-3 SQLOS on a NUMA server (eight-way x3950 M2 - two nodes)

Notice that with SQLOS on a single node based server, SQLOS only has one memory node, while SQLOS on the two-node x3950 M2 has two memory nodes. In a four-node there will be four memory nodes. SQLOS recognizes the two nodes and the information of locality regarding processors and memory in the server.

### 3.3.2 Processor and I/O affinity

By default, no specific processor affinity is set in SQL Server 2008, and all processors can be scheduled to perform all tasks. SQL Server 2008 still attempts to localize the use of resources to take advantage of the NUMA design of the x3950 M2. If you wish to precisely define how system resources are used, hard processor affinity can be enabled in SQL Server 2008. It is possible to configure the affinity setting per instance.

There are two ways to define an affinity mask in SQL Server 2008:

- ▶ Use SQL Server Management Studio graphical interface.
- ▶ Use the `sp_configure` stored procedure.

To change processor affinity, do the following:

1. Select **Start ♦ Programs ♦ Microsoft SQL Server 2008 ♦ SQL Server Management Studio**. The Server Properties window opens.
2. Connect to the SQL Server instance.
3. Right-click the instance icon in **Object Explore**, and click **Properties**.
4. Click **Processors** in Select a page in the left pane. See Figure 3-4 on page 37.
5. Select or clear the check marks in the Processor Affinity column to specify which processors you want this particular database instance to use.



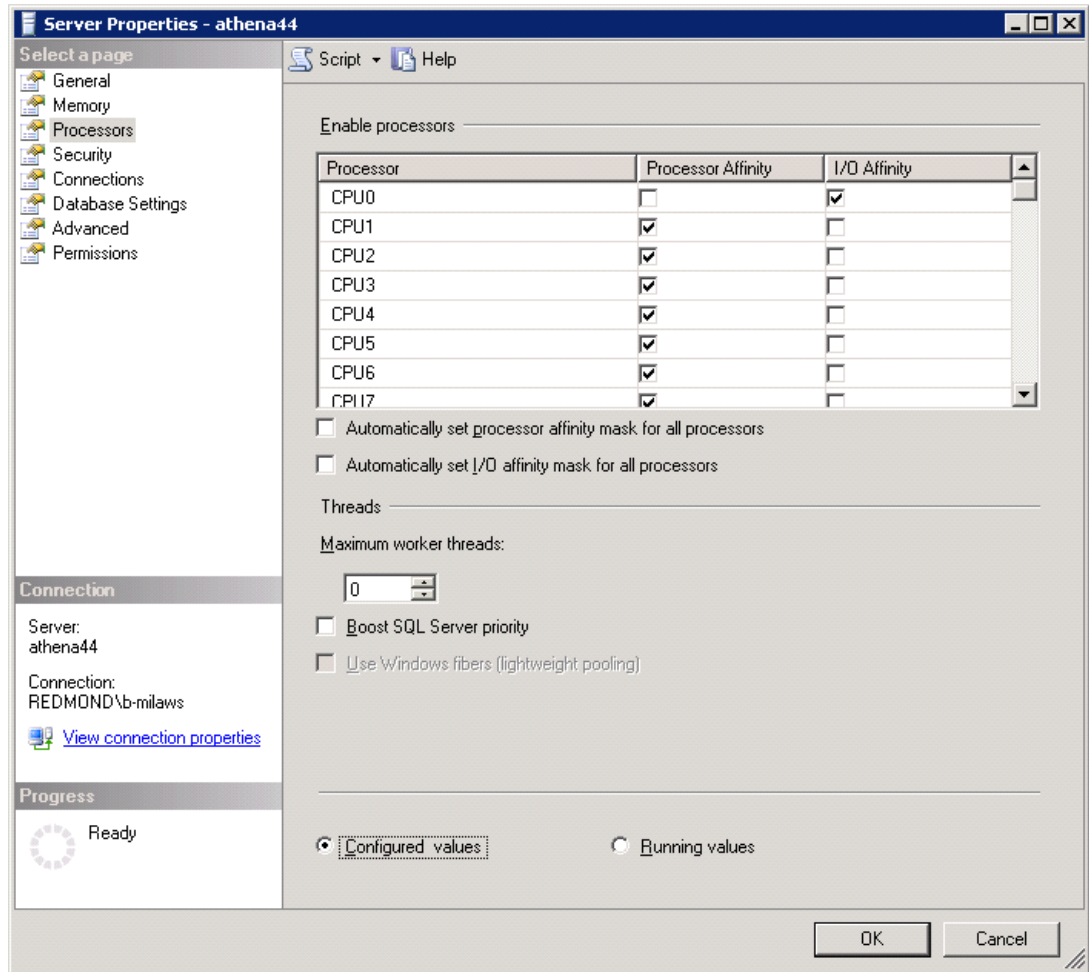


Figure 3-4 Processor affinity and I/O affinity in SQL Server 2008

I/O affinity can associate SQL Server disk I/O to a specified subset of processors so that specified processors handle all disk I/O.

This option can work effectively in the OLTP environment where high load is generated, especially in cases of a server with 16 or more processors. I/O affinity can enhance the performance of SQL Server threads that are issuing I/O. For example, as shown in Figure 3-4, we assign CPU 0 for I/O affinity and the rest for processor affinity. However, I/O affinity does not always improve performance, so you must be careful before using it.

Note that this function does not support hardware affinity for individual disks or disk controllers.

I/O affinity can be configured in either SQL Server Management Studio (as shown in Figure 3-4) or the `sp_configure` stored procedure (as described in “Affinity I/O mask” on page 72)

In order to specify any of the affinity mask options by executing `sp_configure` it is necessary to calculate the value for the `@configvalue` parameter.

SQL Server uses a bitmap to number each core (as show in Table 3-3 on page 38).

Table 3-3 Number of bytes and number of cores supported

# Bytes affinity mask	# Cores supported
1	8
2	16
3	24
4	32

To cover more than 32 CPUs, configure a four-byte affinity mask for the first 32 CPUs and up to a four-byte Affinity64 mask for the remaining CPUs.

Although the `sp_configure` stored procedure can accept integer values for the `@configvalue`, the recommendation is to work with the hexadecimal number of the actual bitmap. For example, if we need to configure an affinity mask for cores 30 and 31 we will use the 0xC0000000 hex number instead of its decimal -1073741824.

For an example of how to use `sp_configure` to specify the affinity settings shown in Figure 3-4 on page 37, see “Affinity I/O mask” on page 72.

**Note:** For more information regarding the Affinity Mask or Affinity64 Mask option see SQL Server 2008 Books Online.

### 3.3.3 Soft NUMA

After configuring affinity on the cores, the next step is configuring SQL Server’s Soft NUMA nodes. The work that comes into a certain Soft NUMA node is handled by that Soft NUMA node exclusively. This keeps the processors, caches, memory, and data related to the work all local to the Soft NUMA node.

For example, you can divide a 2-way x3950 M2 (4 x quad-core = 32 cores) that has two physical nodes into three logical nodes as follows:

- ▶ Logical NUMA node 0 comprised of processors 0 to 15 in physical node 1
- ▶ Logical NUMA node 1 comprised of processors 16 to 23 in physical node 2
- ▶ Logical NUMA node 2 comprised of processors 24 to 31 in physical node 2

To specify the logical node arrangement in SQL Server 2008, use a binary mask that represents cores. The mask has a bit for each core as 31 . . . 876543210 with the first core as zero from right to left. Set each core bit to 1 to select a core or 0 to not select a core.

Table 3-4 shows the masks for each of these logical nodes.

Table 3-4 Node identifier masks

Node #	Binary	Hexadecimal
Logical NUMA node 0	00000000000000001111111111111111	0x0000FFFF
Logical NUMA node 1	00000000111111110000000000000000	0x00FF0000
Logical NUMA node 2	11111111000000000000000000000000	0xFF000000

Configuring the logical NUMA nodes requires that you directly edit the registry as follows:

1. First configure affinity mask and affinity I/O mask options in the instance as shown in Example 3-1 on page 39 for affinities on all cores of the server.

*Example 3-1 Configure script for setting affinity mask and affinity I/O mask with sp\_configure*

```
EXEC sys.sp_configure N'affinity mask', 0xFFFFFFFF
GO
EXEC sys.sp_configure N'affinity I/O mask', 0
GO
RECONFIGURE WITH OVERRIDE
GO
```

2. Start **regedit**.

3. Create the registry keys and DWORD values as follows:

- [HKLM\SOFTWARE\Microsoft\Microsoft SQL Server\100\NodeConfiguration\Node0]  
"CPUMask"=DWORD: 0000FFFF
- [HKLM\SOFTWARE\Microsoft\Microsoft SQL Server\100\NodeConfiguration\Node1]  
"CPUMask"=DWORD: 000FF0000
- [HKLM\Microsoft\Microsoft SQL Server\100\NodeConfiguration\Node2]  
"CPUMask"=DWORD: FF000000

These updates are shown in Figure 3-5.

4. Map the logical nodes to the port numbers as described in 3.3.4, "Network affinity" on page 40.

5. Restart the server.

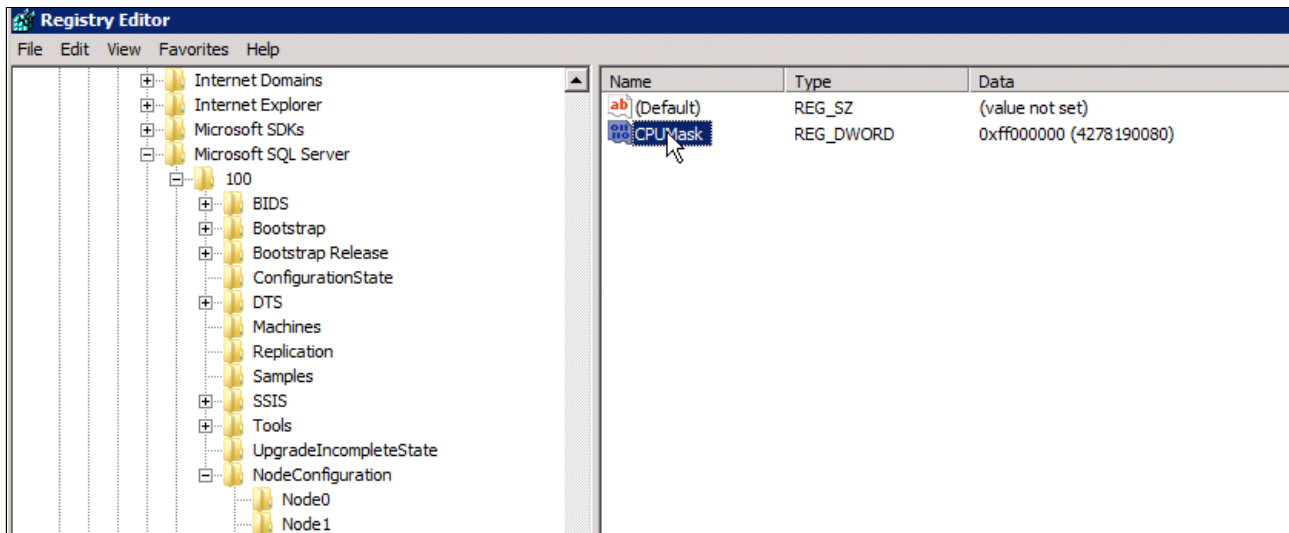


Figure 3-5 Registry settings to configure Soft NUMA nodes

A logical node that spans multiple physical nodes cannot be created. For example, it is not possible to configure a logical node with processor 18 in physical node 1 and processor 4 in physical node 2. However, it is possible to use processors in multiple physical nodes for one workload.

**Note:** To view the configuration on the nodes you can query sys.dm\_os\_nodes catalog view.

### 3.3.4 Network affinity

Network affinity was first introduced in SQL Server 2005 and it is also supported in SQL Server 2008. This feature provides clients with the ability to connect to specific nodes. Network Affinity can be set by IP address (the IP address you connect to determines your Soft NUMA node) or by IP port (the IP port you connect to determines your Soft NUMA node).

**Tip:** In the SQL Server 2008 Books Online, this network affinity is called *NUMA affinity*.

To maximize the effectiveness of the network affinity, it is best to have a network adapter physically residing on each hardware node, each with its own IP address.

For example, consider an eight-way x3950 M2 and two network cards in the server, one in each node. In this case, we can set a port-based connection affinity.

We can configure these settings in the SQL Server Configuration Manager GUI or the Windows registry.

When this setting is used, the workload of Port 1 can only be processed by Soft NUMA node 0, the workload of Port 2 is only processed by Soft NUMA node 1; and the workload of Port 3 is only processed by Soft NUMA node 3 (see Table 3-4 on page 38 for information on how the Soft NUMA nodes are formed)

Thus, each workload takes full advantage of local memory access because the requested data is located in local memory whenever possible. If the memory in the local node is insufficient, then memory in another node would be allocated.

To configure network affinity in SQL Server 2008, use a binary mask that represents nodes, similar to that described in 3.3.3, “Soft NUMA” on page 38.

The mask has a bit for each node as 210 with the first node (node 0) as the rightmost bit in the mask. Set each node bit to 1 to select a node or 0 to not select a node.

For example:

- ▶ To specify just node 0, use mask 001 or hex 0x1.
- ▶ To specify just node 1, use mask 010 or hex 0x2.
- ▶ To specify just nodes 2, use mask 100 or hex 0x4.

To specify all nodes, use a mask of -1 or leave the field blank.

In our example, we configured the affinity of a two-node x3950 M2 (2-way) by mapping node 1 to port 49200, node 2 to port 49205 and node 3 to port 49215. You can do this as follows:

1. Select **Start ♦ Programs ♦ Microsoft SQL Server 2008 ♦ Configuration Tools ♦ SQL Server Configuration Manager**.
2. In SQL Server Configuration Manager, expand **SQL Server 2008 Network Configuration** and expand **Protocols for <instance name>**, where <instance name> is the database instance you wish to configure.
3. In the details pane, double-click **TCP/IP**. If “Listen All” is enabled, then SQL Server will be listening for connections on all of the IP addresses the server has, as shown in Figure 3-6 on page 41.

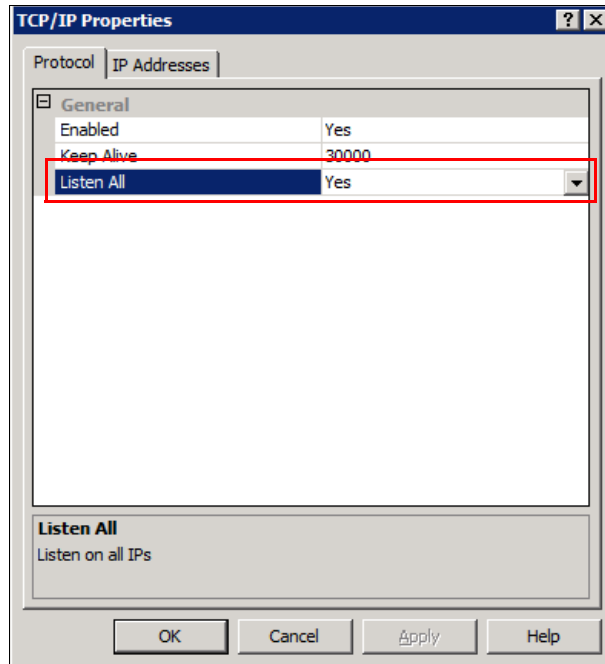


Figure 3-6 Protocol setting for affinity in SQL Server Configuration Manager

4. Assuming the use of “Listen All”, go to the “IP All” section on the “IP Addresses” tab and go to the “TCP Port” line. This is where you will list the IP ports that SQL Server will be listening on all IP addresses. To affinity each port for certain Soft NUMA nodes you should list them as:

port1[mask1], port2[mask2], . . . .

In this case we are going to configure ports affinity as:

1433,49200[0x01],49205[0x02],49215[0x04]

It will look as shown in Figure 3-7 on page 42.

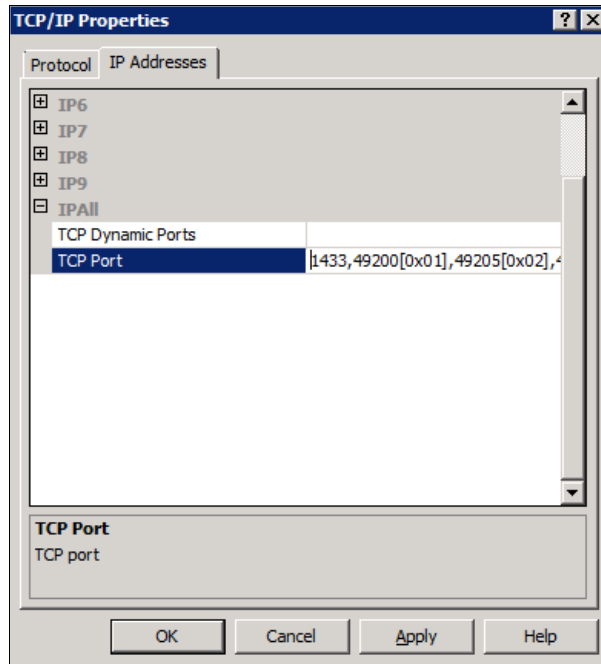


Figure 3-7 Port setting for NUMA affinity in SQL Server Configuration Manager

This means that users that connect to SQL Server via any IP address on port 1433 can have their work assigned to any Soft NUMA node. Connections via port 49200 will be affinity to Soft NUMA node 1, port 49205 to Soft NUMA node 2 and port 49215 to Soft NUMA node 3.

**Note:** If you leave Listen All as **No** (disabled) in Figure 3-6 on page 41, you will need to set the connection affinities for the specific addresses that are enabled, as shown in Figure 3-7.

5. Restart SQL Server 2008 Service so that the changes take affect.
6. Finally, the users need to be configured so they connect to the desired port.

Consider a two-way x3950 M2 with two physical nodes. We want to use 24 processors for OLAP workload and the remaining processors for data loading.

First, make three logical nodes as previously described. Then specify masks as described in the following paragraph.

To assign the OLAP workload to port 49200, specify node mask of logical nodes 0 and 1 (mask 011 or 0x3) in the TCP Port field, and to assign the data loading workload to port 49205, specify logical node 3 (mask 100 or 0x4) in the TCP Port field.

49200[0x03],49205[0x04]

Figure 3-8 on page 43 shows the port settings for Soft NUMA nodes with affinity 0 and 1 to OLAP workload and Soft NUMA node 2 to OLTP workload.

When you perform this configuration, the OLAP workload to Port 49200 uses twenty four cores (0 to 23) and the data loading workload to port 49205 uses eight cores (24 to 31).

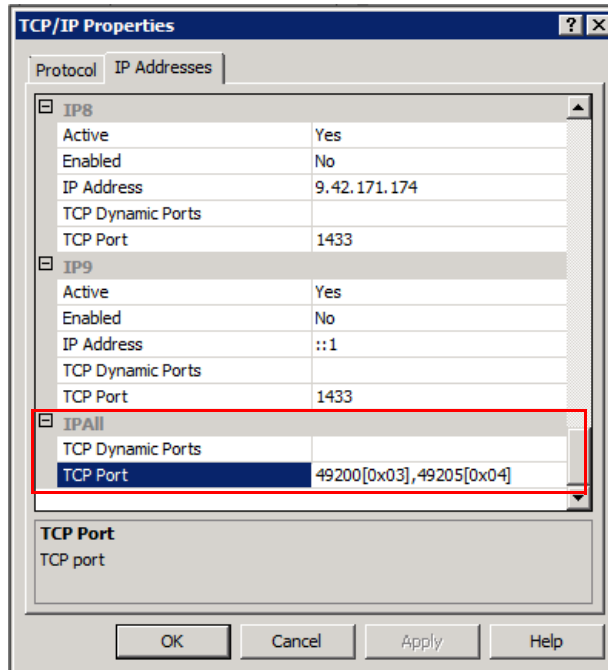


Figure 3-8 Port settings for Soft NUMA nodes

### 3.3.5 Memory

SQL Server can allocate memory effectively in multi-node configurations such as the x3950 M2. As previously described, SQL Server 2008 has SQLOS, and it attempts to localize the use of resources to take advantage of the NUMA design of the x3950 M2. That is, SQL Server 2008 tries to allocate data in the same memory as that of the requesting processor to leverage local memory access.

With SQL Server 2008, how much memory is allocated on each node can be viewed by using the DBCC MEMORYSTATUS command.

In addition, memory usage can be monitored by using the Windows performance counter: SQLServer:BufferNode.

**Tip:** To see the effect of the NUMA awareness in SQL Server 2008, disable the NUMA optimizations with trace flag 8015. Then confirm memory usage with the DBCC MEMORYSTATUS command. See SQL Server 2008 Books Online to learn how to set trace flags.

To configure memory per instance, use the SQL Server Management Studio or the `sp_configure` stored procedure. To change the memory setting with SQL Server Management Studio, follow these steps:

1. Select **Start** ♦ **Programs** ♦ **Microsoft SQL Server 2008** ♦ **SQL Server Management Studio**.
2. Connect to SQL Server instance.
3. Right-click the instance icon in Object Explore, and select **Properties**.
4. Select **Memory** in Select a page in the left pane.
5. Change Minimum server memory and Maximum server memory, indicated in red in Figure 3-9 on page 44.

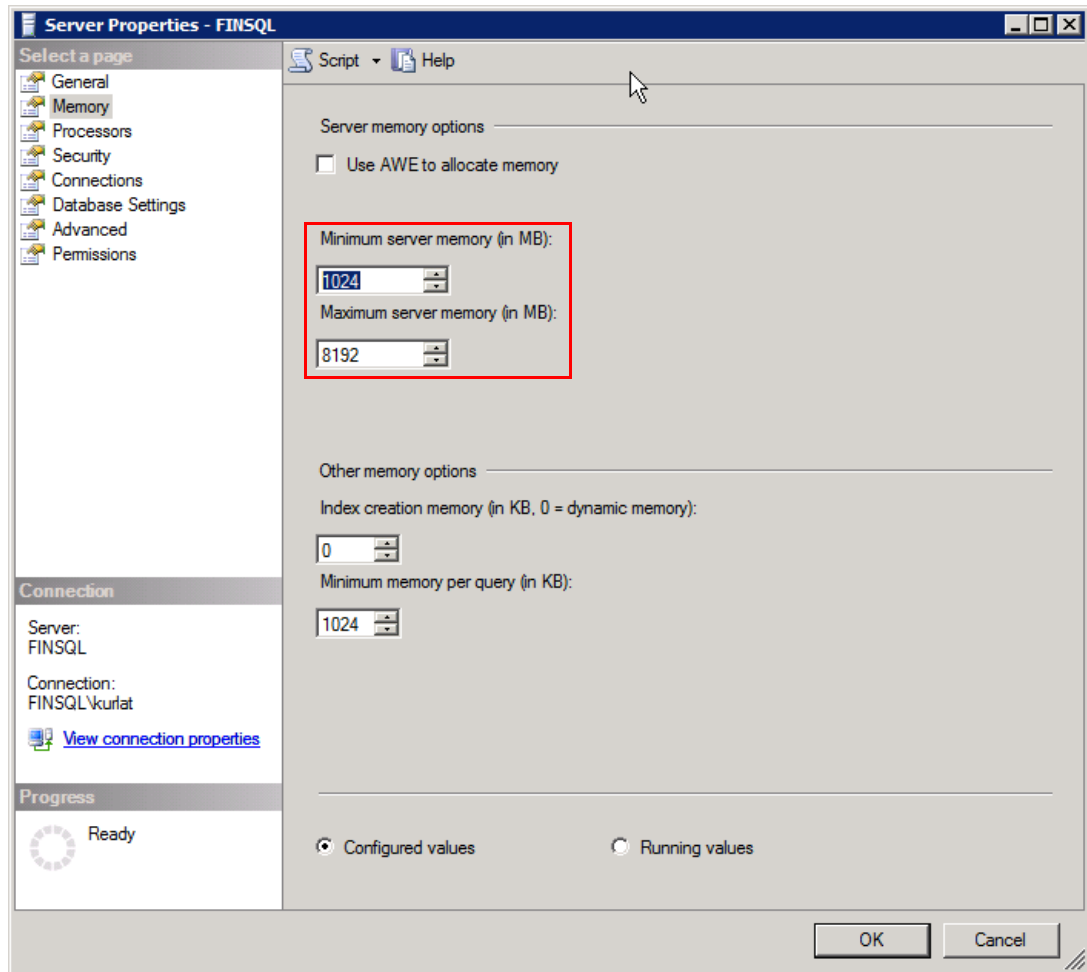


Figure 3-9 Memory setting in SQL Server Management Studio

To change the parameters using **sp\_configure**:

*Example 3-2 Change memory configuration using sp\_configure*

---

```
EXEC sys.sp_configure N'min server memory (MB)', N'1024'
GO
EXEC sys.sp_configure N'max server memory (MB)', N'8192'
GO
RECONFIGURE WITH OVERRIDE
GO
```

---

### 3.3.6 Resource Governor

Resource Governor is a new feature included in SQL Server 2008 for management of resource contention between sessions running on the same SQL Server instance. It enables you to manage SQL Server workloads and resources by specifying limits on resource consumption by incoming requests to the database engine. Resource Governor does this by incorporating the concepts of:

- ▶ Classification functions
- ▶ Workload groups
- ▶ Resource pools



## Classification functions

Classification functions are rules that are used to group SQL Server sessions. Sessions are groups based on a set of user-specified criteria and are stored in a SQL function. This allows Resource Governor to manage the group of sessions collectively.

## Workload groups

A workload group is a logical container for grouping sessions requests with similar classification criteria. Resource Governor predefines two fixed workload groups: Internal and Default. The Internal workload group cannot be changed, but the group can be monitored. All sessions that do not apply to any classification criteria, or if there is a general classification failure, will be classified into the default workload group.

## Resource pools

The resource pools represents the physical separation of the resources in the server. A resource pool is like a virtual SQL Server instance inside a SQL Server instance. The resource pool has two parts: The first one enables minimum resource reservation and the second part specifies the maximum possible resource consumption. The first does not overlap with the other pools and the second is shared between all the other pools.

The resources for the pool are set by specifying one of the following for each resource:

- ▶ MIN or MAX for CPU
- ▶ MIN or MAX for Memory

**Note:** The sum of MIN values across all the resource pools cannot exceed 100% of the server resources. MAX value can be set between MIN value and 100%.

Resource Governor predefines two fixed resource pools:

- ▶ **Internal Pool:** Represents the resource consumed by SQL Server itself. This pool *only* contains the internal group and it cannot be altered.
- ▶ **Default Pool:** The first predefined user pool. It can be changed but not dropped.

## Treating of sessions by the Resource Governor

The following steps describe how Resource Governor treats incoming sessions and describes the relationship between the components discussed previously (see Figure 3-10 on page 46):

1. There is an incoming connection for a session (Session 1..N)
2. The session is classified by a user-defined function during the login process:
  - a. Login authentication
  - b. LOGON trigger execution
  - c. Classification
3. The session workload is routed to a workload group.
4. The workload group uses the resource pool with which it is associated. A workload group can be associated with only one resource pool.
5. The resource pool provides and limits the resources required by the session or application. A resource pool can be used by several workload groups.

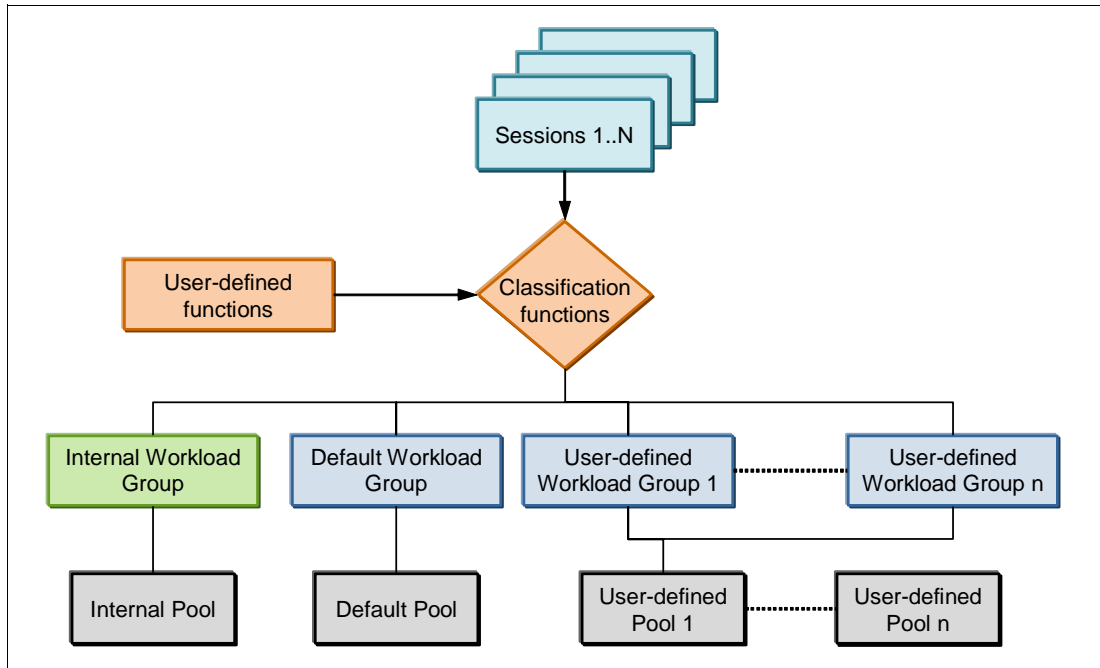


Figure 3-10 Relationships between Resource Governor components

## Setting up Resource Governor

Continuing with the example in the previous sections we are going to set up resource pools, workload groups, and classification functions for OLAP and data load applications.

1. Start Resource Governor.

```
ALTER RESOURCE GOVERNOR RECONFIGURE
GO;
```

**Tip:** To check whether the Resource governor is enabled execute:

```
select is_enabled from sys.resource_governor_configuration
```

If it returns 1 the Resource Governor is enabled in the instance.

2. Write classification for the different type of applications.

```
CREATE FUNCTION dbo.classifier() RETURNS sysname
WITH SCHEMABINDING
AS
BEGIN
-- Declare the variable to hold value returned in sysname
DECLARE @group_name sysname
if (APP_NAME() like '%OLAP%')
    set @group_name = 'OlapWorkLoadGroup'
else if (APP_NAME() like '%LOAD%')
    set @group_name = 'DataLoadWorkLoadGroup'
else
    set @group_name = 'default'
RETURN @group_name
END
GO
```

**Note:** The following system functions can be used for classification: HOST\_NAME(), APP\_NAME(), SUSER\_NAME(), SUSER\_SNAME(), IS\_SRVROLEMEMBER() and IS\_MEMBER()

3. Bind the classifier function to the Resource Governor.

```
ALTER RESOURCE GOVERNOR WITH (CLASSIFIER_FUNCTION=dbo.classifier);
GO
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
```

**Tip:** Column classifier\_function\_id in catalog view sys.resource\_governor\_configuration will show the object\_id of the classifier function. To check it, execute:

```
select OBJECT_NAME(classifier_function_id) from
sys.resource_governor_configuration
```

4. Create Resource Pools for OLAP and Data Load applications.

```
CREATE RESOURCE POOL OLAP_Pool
WITH
(
    MIN_CPU_PERCENT = 10, -- Setup 10% MIN CPU.
    MAX_CPU_PERCENT = 100, -- Setup 100% MAX CPU.
    MIN_MEMORY_PERCENT = 30, -- Setup 30% MIN MEMORY.
    MAX_MEMORY_PERCENT = 50 -- Setup 50% MAX MEMORY.
)
```

GO

```
CREATE RESOURCE POOL DataLoad_Pool
WITH
(
    MIN_CPU_PERCENT = 10, -- Setup 10% MIN CPU.
    MAX_CPU_PERCENT = 100, -- Setup 100% MAX CPU.
    MIN_MEMORY_PERCENT = 10, -- Setup 10% MIN MEMORY.
    MAX_MEMORY_PERCENT = 20 -- Setup 20% MAX MEMORY.
)
```

GO

**Note:** For information on parameters and their defaults see SQL Server 2008 Books Online under “CREATE RESOURCE POOL (Transact-SQL).”

5. Create Workload Groups binding the Resource Pools that were created before.

```
CREATE WORKLOAD GROUP OlapWorkLoadGroup
USING OLAP_Pool;
CREATE WORKLOAD GROUP DataLoadWorkLoadGroup
USING DataLoad_Pool;
```

**Note:** For information on non default parameters see SQL Server 2008 Books Online under “CREATE WORKLOAD GROUP (Transact-SQL).”

## Monitoring Resource Governor components

SQL Server 2008 introduces new catalog views, dynamic management views, event classes, and performance counters to monitor and check configurations on this new feature.

### ► Performance counters

- Instance per Pool: SQLServer: Resource Pool Stats
- Instance per Group: SQLServer: Workload Group Stats

### ► Dynamic management views

- Three new views for Resource Governor stats
  - sys.dm\_resource\_governor\_workload\_groups
  - sys.dm\_resource\_governor\_resource\_pools
  - sys.dm\_resource\_governor\_configuration
- Columns added on SQL Server DMVs
  - sys.dm\_exec\_sessions.group\_id
  - sys.dm\_exec\_requests.group\_id

### ► Catalog views

Three catalog views for Resource Governor metadata

- sys.resource\_governor\_configuration
- sys.resource\_governor\_resource\_pools
- sys.resource\_governor\_workload\_groups

### ► SQL Server event class

Updated and new SQL Server event classes

- CPU Threshold Exceeded Event Class
- PreConnect:Starting Event Class
- PreConnect:Completed Event Class

**Note:** For complete details about the DMV changes, catalog views, and so forth, see SQL Server 2008 Books Online under “Resource Governor DDL and System Views.”

In Example 3-3 we are querying DMV sys.resource\_governor\_workload\_groups JOIN with sys.dm\_exec\_sessions to obtain the number of connections on each Workload Group.

*Example 3-3 Query count connections per workload group.*

---

```
select rgg.name, COUNT(*) from sys.dm_exec_sessions s
inner join sys.resource_governor_workload_groups rgg
on s.group_id = rgg.group_id
group by rgg.name
```

---

## Limitations on Resource Governor

The limitations on the Resource Governor are:

- Resource Governor can only manage Database Engine resources. Analysis Services, Reporting Services, and so forth, are not included in this solution, although it is possible to limit this application's resource with Windows System Resource Manager.
- The Resource Governor feature is installed and managed per instance. Each instance is controlled individually.
- The Resource Governor controls CPU bandwidth and Memory; it does not control I/O or connection affinity to TCP/IP ports.

- ▶ It is only included in Enterprise, Developer, and Evaluation editions of SQL Server.

### Running example test

After we set up the Resource Governor as described in the previous section, we tried a test with RML utility OSTRESS.

**Note:** For further information about RML tools, see <http://www.microsoft.com/downloads> under the topic “RML Tools.” There are x86 and x64 editions available for download.

We simply executed a select count(\*) from a table on database AdventureWorks2008 (a sample database on SQL Server 2008). This execution did 100000 iterations with 500 concurrent users. We first executed the OSTRESS without running Resource Governor and we saw that all CPUs affinitized with the instance raised up to 100%.

But running the same test with Resource Governor on, setting the connection to use the OLAP Workload Group resulted the performance shown in Figure 3-11.

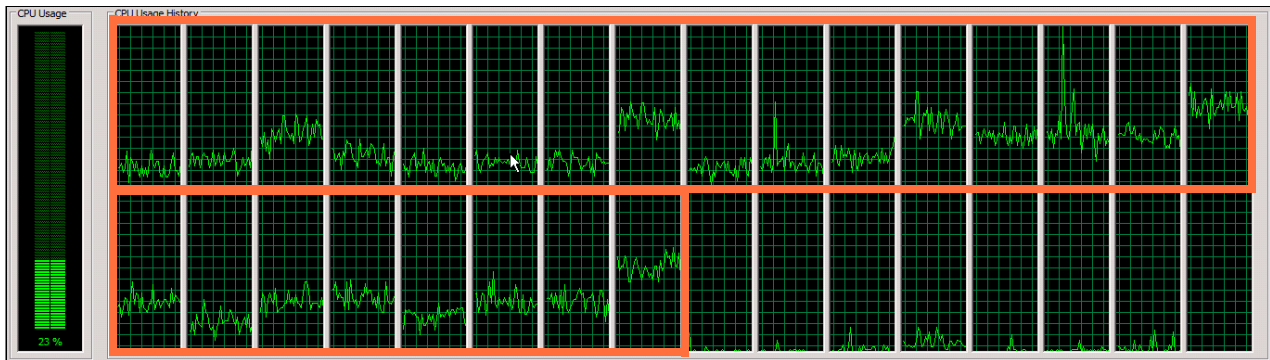


Figure 3-11 Windows Task Manager - CPU Usage History

Figure 3-11 shows how the Resource Governor uses NUMA nodes 0 and 1 for this workload as highlighted, but not node 2..

## 3.4 Server consolidation

This section describes server consolidation. It introduces the model of general server consolidation and specifically discusses database server consolidation. This section also explains vertical consolidation and horizontal consolidation, both of which are used in database server consolidation.

### 3.4.1 Concept of consolidation

*Consolidation* is more than just an effort aimed at cost reduction. It is an opportunity to reexamine the needs of your business, to improve the performance and scalability of the environment to meet current and near future needs, and to improve the quality of the whole environment by ensuring that standardization is achieved in all related processes and procedures.

When “consolidation” is applied to a SQL Server environment, the desired outcomes are:

- ▶ Reduce the physical number of SQL Servers.
- ▶ Reduce the number of SQL Server instances.

- ▶ Reduce the number of SQL Server databases.
- ▶ Centralize all SQL Servers to one physical location.
- ▶ Centralize SQL Server administration.
- ▶ Improve SQL Server utilization rates.
- ▶ Improve system monitoring and alerting.
- ▶ Implement consistent standards for the whole SQL Server environment.
- ▶ Improve data security and limit user access as the business needs require.

### 3.4.2 Forms of consolidation

A SQL Server server consolidation exercise involves a combination of five basic forms:

- ▶ **Physical server consolidation**

Physical server consolidation is the act of reducing the actual number of installed physical servers. It is generally believed that servers cannot be shared safely. For that reason many servers are rarely used to their full potential. However, after careful analysis of the workload of each server (a requirement), databases can be migrated to refreshed server hardware that is designed to handle the workload being placed on it.

- ▶ **Storage consolidation**

Storage consolidation can involve simply moving the data from a number of servers to a large locally attached disk storage sub-system in a new server. Alternatively, it can also involve the implementation of a centralized disk sub-system that is then shared over a Storage Area Network (SAN) using high-speed network connections. The right solution depends entirely on business requirements and available budget. When physical server consolidation occurs, it follows that there will be some form of storage consolidation.

- ▶ **Geographic consolidation**

Geographic consolidation takes server systems located in different rooms, buildings, cities, or countries and brings them together in a single, or at least reduced, number of sites, which can reduce costs because real estate is expensive. Close attention must be paid to both the server workload and the network requirements. Without sufficient network connectivity and bandwidth, performance could be adversely affected, which might have serious business implications that outweigh any potential real estate cost benefits.

- ▶ **Logical server consolidation**

Logical server consolidation relates to the analysis of the SQL Server instances that have been installed on existing SQL servers. Since the release of SQL Server 2000, it has been possible to install multiple instances of SQL Server on the same physical server. Each instance is effectively a logical server because it has its own network interface and server resources. There are only a few situations when a database needs its own SQL Server instance and, in most cases, databases can be moved from their own dedicated SQL Server instances to a SQL Server instance that is shared with many other databases, assuming the workload requirements have been taken into consideration.

- ▶ **Database consolidation**

Database consolidation requires in-depth analysis of database designs to determine where databases can be combined, which reduces the number of databases. In practice, this requires a high degree of effort and can require changes to the related application. However, there might be a database that is located in several different countries that shares a common design but contains country-specific data. If these identically structured databases are geographically consolidated into one site, conceivably the databases could

be combined and access segregated with views and user permissions controls in SQL Server 2008.

### 3.4.3 Consolidation strategies

The two general approaches to server consolidation are described in this section, together with guidance on determining what mix of strategies to deploy for your consolidation project and how and when to apply them to a SQL Server consolidation. There are two dimensions to consider when consolidating SQL Servers:

- ▶ The number of physical servers after consolidation
- ▶ The number of SQL Server instances after consolidation

#### Vertical consolidation - Scale up

Vertical consolidation describes the process of building a single, very powerful logical server that has the capacity to assume the databases and processing from a number of smaller, disparate systems.

Traditionally, once a 4-way server had been utilized to its maximum, it was very expensive to add more processing capacity to the same logical server, so more Windows server instances were created, increasing the Windows server management overhead. With the release of Windows Server 2003 supporting NUMA technologies, and its successor Windows Server 2008 with all the enhancements on NUMA technologies (see 3.2.1, “NUMA optimization for Windows Server 2008” on page 32), it became cost-effective and low-risk to transform a 4-way server into an 8-, 12-, or 16-way server. The IBM System x3950 M2 has been specifically designed for this purpose.

Starting with a single x3950 M2 system (a node), this server platform can grow as your business requirements grow, with the addition of up to four interconnected nodes, to provide mainframe-class levels of processing power and availability.

Building a single logical server limits the administration and maintenance to one Windows server instance, while providing the processing power of many.

#### Horizontal consolidation - Scale out

Horizontal consolidation refers to the act of commoditizing a server function and then building multiple replicas of that server template to meet overall processing demand. Some load balancing technology is required to ensure that the work is evenly spread among all servers and there are limitations on how work can be distributed, making this approach unsuitable for many types of applications. Probably the most common implementation of horizontal consolidation occurs with Web farms, where many identical servers have the same role and the user connection is stateless.

Horizontal consolidation can deliver standardization and uniformity, which simplifies system maintenance and administration. However, there is not necessarily a reduction in the number of physical servers.

With regards to SQL Server, horizontal consolidation can be equated to using multiple instances of SQL Server in place of a single instance.

#### SQL Server instances - single or multiple?

Using single or multiple instances equates to employing vertical or horizontal consolidation for SQL Server instances. Hosting all databases in a single instance can be considered to be vertical consolidation, while creating multiple SQL instances on a server, or cluster of servers, is a practical application of horizontal consolidation. This section discusses the criteria to

assess before a decision can be reached on how many instances of SQL Server are required. “Less is more” is the guiding principle here.

In 2.3.1, “Windows, SQL Server, and 32-bit versus 64-bit” on page 24, we describe the memory limitations that are inherent with 32-bit versions of Windows Server and SQL Server. One way of addressing this limitation is to install multiple instances of SQL Server so that more of the available RAM can be utilized, albeit with an increased management overhead for each additional instance.

With 64-bit Windows Server and SQL Server, the addressable memory space is flat and the memory limitations are removed, so more databases can be hosted in a single instance. Now the rationale for deciding how many SQL Server instances are required comes down to the following considerations:

- ▶ How many different sort orders and collations are required by your databases? If they all use the same sort order and collation, they can all share the same instance. Each SQL Server instance has a single default sort order and collation.
- ▶ Do you need to limit DBA access to certain databases? A user with DBA rights for a SQL Server instance has DBA access to all databases in that instance. If you must restrict DBA access to certain databases, they must be hosted in a separate SQL instance with user access configured accordingly.
- ▶ What are your maintenance and availability requirements? Each SQL instance must have SQL Server patches applied to it individually. You might choose to group all databases with a similar service window in the same SQL Server instance so that they can all be taken down at the same time without impacting the user availability.
- ▶ Do you have two (or more) SQL user IDs or database names that are identical and cannot be changed? In this case, you must accommodate the databases in separate SQL Server instances. Normally, you resolve the naming conflicts and then host in the same instance, but if this is not possible, you must use separate instances.
- ▶ Do you need to continue running SQL Server 2005 alongside SQL Server 2008? If you have some databases that cannot be migrated for whatever reason (application compatibility, vendor support, and so forth) you might have to continue running SQL Server 2005. You can install a SQL Server 2005 instance on the same consolidated hardware as your SQL Server 2008 instances.

**Note:** Although it is possible to install SQL Server 2000 versions in the same box with SQL Server 2005 and SQL Server 2008, there are several issues to consider:

- ▶ SQL Server 2000 is already in an Extended Support Phase. Our recommendation is to upgrade all remaining instances under this version.
  - ▶ SQL Server 2000 is not supported on Windows Server 2008, so you will need to install a Windows Server 2003 Edition.
- ▶ Do you need to run SQL Server with different hotfix or service pack levels installed? You might have some databases that do not work with the latest service packs or you might have a vendor that has only certified their application with a certain level of hotfix or service pack. Using a separate SQL Server instance, you can maintain it at the required level of hotfix or service pack, while other installed instances are kept up to date.
  - ▶ Do you have different types of processing executed on each database? Are there databases with OLAP and OLTP processing? You might have databases with a high percentage of batch processing during working and non-working hours and heavy report querying, or databases with high transactional processing like retail, IVR related, and so forth. With different type of processing applications you may prefer a multiple instance



architecture, although SQL Server 2008 provides new features like Resource Governor for limiting consumption of resources.

Depending on the licensing model you have chosen, the number of SQL Server instances you arrive at might require you to purchase additional SQL Server licenses. However, there are several options available from Microsoft that are discussed at length at:

<http://www.microsoft.com/sqlserver/2008/en/us/how-to-buy.aspx>

### ***Resource contention***

When you use multiple instances, be careful how you configure the allocation of processors and memory. It is important that a processor not be shared between instances because doing so can cause performance degradation. For example, if you have two instances of SQL Server 2008 running on an 4-way server, you can incorrectly configure hard affinity to processors 0 to 1 on both. The result is that both instances must share processors 0 to 1 while processors 2 to 4 are idle.

**Note:** By default, each instance has affinity to all installed processors. If you are running multiple instances, you must reconfigure affinity so that instances do not share processors.

Memory configuration is also important because sharing memory among instances can, in the worst case, cause system failure. For example, if you configure a minimum allocated memory for each instance but the total exceeds the amount of installed RAM, then this might cause an instance to fail.

### ***Clustering issues***

Mistakes in configuring multiple instances often cause serious problems in the environments in which SQL Server 2008 is running in an active/active cluster with Microsoft Clustering Service (MSCS). MSCS is a high availability solution implemented in Windows Server 2008 Enterprise Edition and Datacenter Edition. Two or more servers can be configured in a cluster and can switch the server in case of serious problems. For example, if an application stops because of a critical problem, the other server takes over the application after a short switching time. Thus, the application can continue to run normally.

If you have all nodes in a cluster configured to run applications and, during a failure, the surviving nodes take on the additional load from the failed node, this is called *active-active* clustering. If instead one of the nodes is idle and its purpose is only to assume the load of a server in the event that server fails, this is called an *active-passive* cluster.

When you run SQL Server in an active-active cluster with each instance using all processors and memory in each server, if you have not configured SQL Server properly, you might encounter serious problems when the takeover happens. If one server fails, both SQL Server instances try to run on the other server with the same configurations as before the failure, and both instances attempt to use all processors and all memory. With both instances using all processors, performance is degraded. Regarding memory, however, if the total of the minimum allocated memory for each instance exceeds the amount of installed RAM, then the failed-over instance might not restart.

You can avoid resource contention and performance degradation by taking the following precautions:

- ▶ Use active-passive clustering.

In active-passive clustering, one server is reserved for standby. Thus, there are no resource contentions if takeover occurs.

- Configure processors and memory of each instance appropriately on Active/Active MSCS. For example, consider having two 4-way 32 GB x3950 M2 (4 quad-core CPUs = 16 cores each) servers. You can avoid resource contention after takeover by configuring 8 cores and 16 GB of RAM to both instances (that is, the sum total of both instances should be set to sixteen cores and 32 GB of RAM).

Like the active-passive option, half of both servers will be idle. In addition, you should carefully select the four processors so that cluster node A, for example, assigns affinity to processors 0 to 7 and cluster node B assigns affinity to processors 8 to 15.

- Use active-active clustering and use `sp_configure` to reallocate memory usage. In this scenario, you have SQL Server issue the `sp_configure` procedure when it restarts to dynamically reallocate memory to all instances so that the total amount of memory allocated does not exceed the amount of memory installed.

If you plan to implement this method of failure recovery, carefully test the scenario to ensure that all instances do start properly with the correct amount of memory allocated.

We cover more cluster configuration issues later in this section. (See “Server clustering” on page 58).

### Performance Multiple Instances

To gain good performance when you have configured multiple instances, you should map instances to a particular processor (or processors) in each node. For example, consider a 2-way x3950 M2 (Figure 3-12):

- Instance 1 is mapped to processors 0-3 in node 1.
- Instance 2 is mapped to processors 4 and 5 in node 2.
- Instance 3 is mapped to processors 6 and 7 in node 2.

Because the SQLOS on the x3950 M2 is NUMA-aware, this configuration maximizes the use of local memory access.

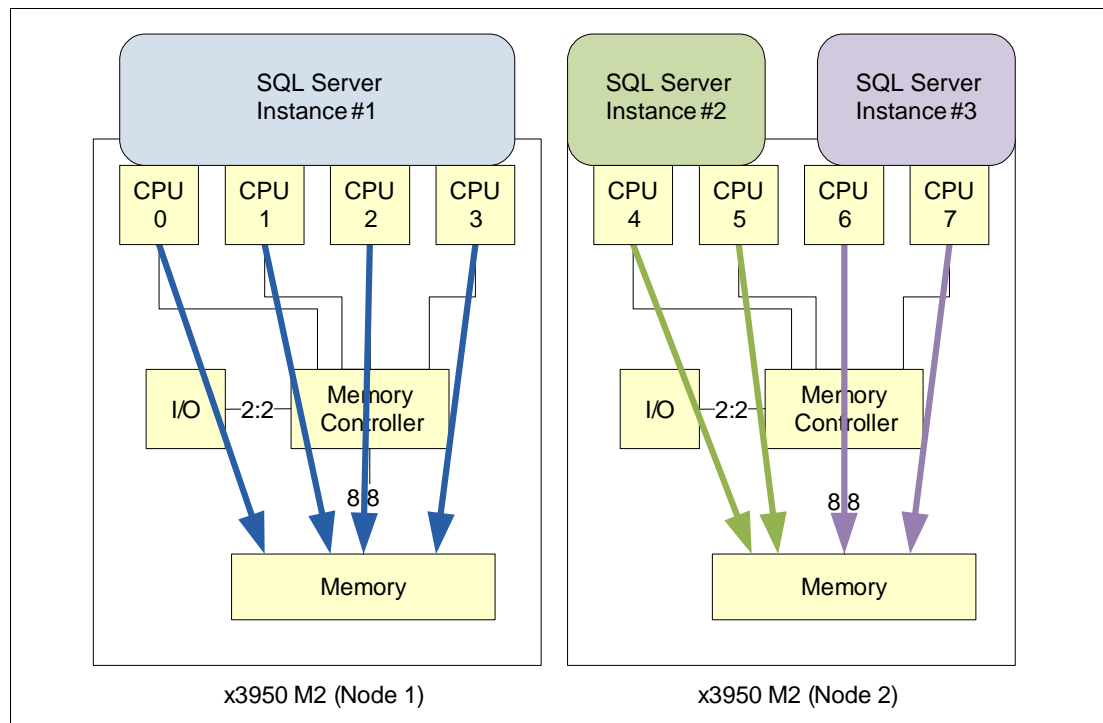


Figure 3-12 Memory access from multiple instances configured NUMA-conscious

You should avoid configuring an instance to use processors from multiple nodes because this requires remote memory access (across the scalability connections), and performance can suffer. For example, do not configure processors 3 and 4 in the same instance. Of course, if you wish to create an instance with more than four processors, then this is unavoidable.

**Operation and maintenance of the database**

You can start and stop SQL Server 2008 per instance. So, if one database needs continuous availability and the other database does not require this level of availability, you might consider multiple instances to reduce the operation and maintenance workload. For example, you can apply the Service Pack to SQL Server 2008 per instance. However, you need to stop the corresponding SQL Server service in Windows Server 2008. If you have multiple instances, you can apply the Service Pack separately. That is, you can apply Service Pack to the database that does not need continuous availability, and you can leave the continuously available database as it is.

**Version of SQL Server 2008 (32-bit or 64-bit)**

32-bit applications can use only 2 GB of virtual address space (VAS) because of the user space on 32-bit Windows. That is, 32-bit SQL Server 2008 can use a maximum of 2 GB of VAS per instance, or 3 GB if you are using 4 GB tuning in Windows Server 2008 (see 2.3, “64-bit computing and SQL Server 2008” on page 24).

When you run 32-bit SQL Server 2008 on Windows Server 2008 x64 using WOW 64 (Windows on Windows), SQL Server 2008 can use a maximum of 4 GB as user space, as shown in Table 3-5. Then, if you use a single instance, you can use 2 GB (or 4 GB on x64 Windows) of VAS for multiple databases.

Table 3-5 Address space in 32-bit and 64-bit

Windows Server 2008	SQL Server 2008	Virtual memory limits	Physical memory limits
32-bit	32-bit	2 GB (3 GB with boot.ini flag)	64 GB
64-bit (x64)	32-bit	4 GB	64 GB
64-bit (x64)	64-bit (x64)	8 TB	2 TB

Alternatively, if you use multiple instances, you can use 2 GB (or 4 GB) x *n* (where *n* is the number of instances) as a VAS for multiple databases. In 64-bit Windows Server 2008 and SQL Server 2008, VAS is available to a maximum of 16 TB (8 TB for the kernel and 8 TB for the user).

**Performance Single Instance**

If you use a single instance on a multiple node x3950 M2, you have some remote memory access in a workload, as shown in Figure 3-12 on page 54. Although memory allocation of SQL Server 2008 is NUMA optimized, some remote memory access can still happen. For example, in Figure 3-13 on page 56, CPU 4 has to obtain the data from remote memory if CPU 0 has previously located the data in its local memory.

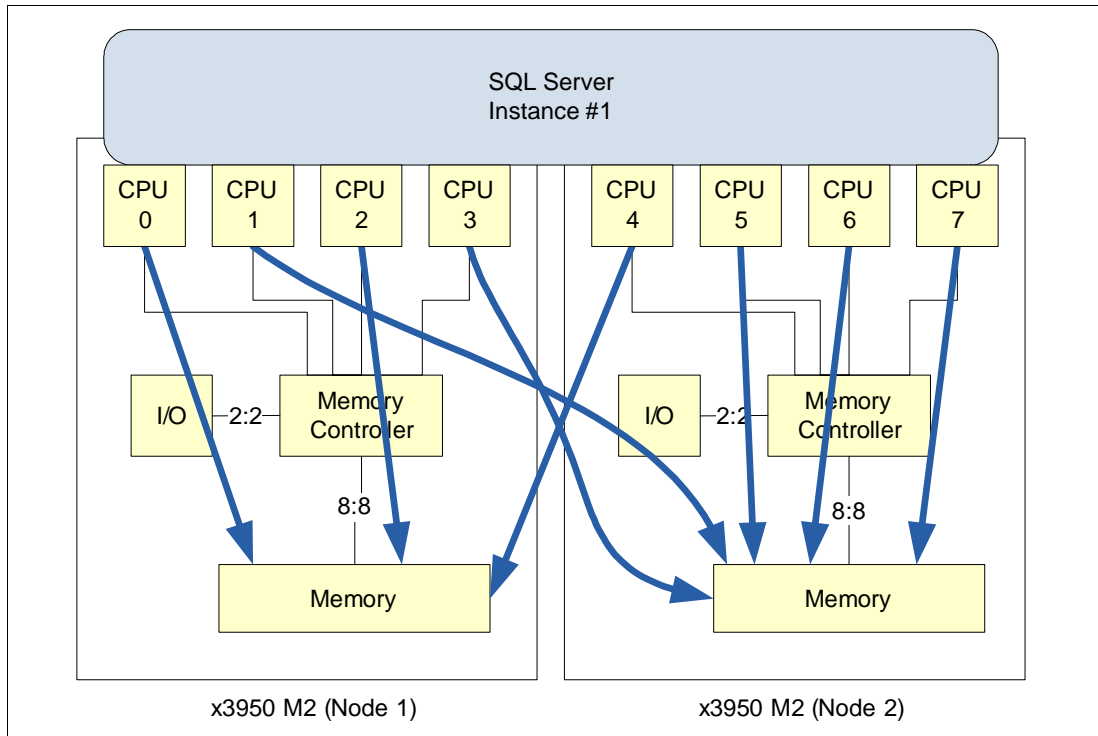


Figure 3-13 Memory access from single instance

In multiple instances, if the instance has affinity to the node set correctly, there is little remote access. With 64-bit SQL Server 2008, the limitations of VAS are eliminated. However, when you are reducing remote memory access, you should examine multiple instances for multiple databases with SQL Server 2008 64-bit.

A drawback of having multiple databases in a single instance is that the buffer pool is shared throughout the databases, so there is the potential that one database can “starve” the others of buffer pool resources. The drawbacks of having multiple instances are that you must do a two-phase commit and that you need Distributed Transaction Coordinator (DTC) to implement transactions across multiple databases.

SQL Server 2008 introduces Resource Governor as a countermeasure for this particular situation. As discussed previously, by using Resource Governor, SQL Server 2008 can limit the amount of CPU and Memory that an application can use within an instance of SQL Server.

Although Resource Governor can limit the percentage of CPU and memory that a session consumes within an instance of SQL Server, it cannot limit the amount of I/Os and TCP bandwidth. Because of this issue it is still important to analyze how the different databases use shared resources as tempdb for determining the correct strategy.

### 3.4.4 Considering service availability

The effect of reducing the number of physical servers with a server consolidation exercise is the concentration of your business systems on the remaining server infrastructure. The consolidation inevitably leads to a consolidation of your risk; for example, a server outage that might have affected one system before consolidation might now affect ten systems.

A crucial part of any consolidation project is to identify and quantify the server infrastructure risks that exist for your business. After they have been identified, the server consolidation architecture must be designed to mitigate those risks. In this section, we cover the risk assessment process and what strategies are available for providing high availability for those systems that need it.

### **Assessing critical systems**

Before considering what can be done to protect your service availability, you need to know which systems are worth protecting and what value they add to the business. A comprehensive review of all systems in-scope for consolidation should be performed with representatives from your organization who can identify the applications that serve its core business functions. They must provide a financial value against each critical application, so that the cost of an outage to the organization can be quantified. It is this cost that should drive the decisions about what mitigation is taken in response to the identified risk.

Each organization and application has its own unique risk factors in this equation; however, some of the common elements are:

- ▶ Value of lost orders
- ▶ Value of lost employee productivity and overtime costs for catch-up
- ▶ Cost of recreating lost data
- ▶ Penalty payments for missed service level agreements (SLAs)
- ▶ Loss of customer good will
- ▶ Damage to the brand image and reputation
- ▶ Impact of legal action against organization or executives

You should assess what currently exists in the way of server infrastructure to ensure service availability for the identified critical systems. This might well highlight existing deficiencies in the deployed infrastructure that can be addressed with the consolidation project.

The architectural infrastructure design for the consolidated server environment should reflect the significance of system downtime and surrounding processes should address the data integrity requirements. These are introduced in the next section.

### **Ensuring system availability**

There are a number of elements involved in ensuring that a newly consolidated server environment experiences high levels of service availability and this subject could probably be a paper all on its own. We have attempted to highlight the major elements that together contribute to the availability levels of a server infrastructure and we introduce some SQL Server 2008 features and enhancements that can protect SQL Server service availability.

#### ***Server room environment***

When you are consolidating physical servers into a central server room facility, it is crucial that these basics be in place before consolidation begins:

- ▶ Good electrical provision so that you can provide separate power feeds to each server
- ▶ Sufficient cooling for the current servers and those that are in place after the consolidation
- ▶ Secure physical access control to ensure that unauthorized people cannot interfere with the servers
- ▶ Adequate space around each server rack for routine access to the servers during installation or server maintenance

### ***Hardware fault tolerance***

Consider deploying server hardware that contains redundant components that can keep the server running if there is a failure. The following areas offer good protection against individual component failures in a server chassis:

- ▶ Two or more power supplies from two separate power sources
- ▶ Two or more network cards with network card teaming software
- ▶ Hardware RAID protection with a hot spare drive for local disk sub-system
- ▶ Redundant fans in the server chassis for reliable cooling
- ▶ If using a SAN, two or more SAN interface cards (host bus adapters)
- ▶ IBM Chipkill memory, ECC memory, redundant bit steering, memory scrubbing, and memory mirroring for protection against memory failure
- ▶ Hot-swap or hot-add capabilities for memory, hard drives, or PCI-X interface slots so that they can be added or replaced without stopping the server

Fault tolerant servers are a key component of a consolidated server platform.

### ***System monitoring and alerting***

Having redundant hardware components is great, but the value is diminished if hardware failures are not spotted and acted upon quickly. If a server has two power supplies and one fails, the server stays on and service is not interrupted. However, if the faulty power supply is not quickly replaced, the server would be disabled if the second supply also fails.

There are many tools available for server hardware monitoring. All IBM servers include a license for the IBM Director systems management suite. IBM Director, by supporting system management industry standards, can also manage a wide variety of non-IBM systems and operating systems.

### ***Server clustering***

Server clustering with Windows Server 2008 and SQL Server 2008 provides a good solution for protecting against the total failure of a physical server.

A cluster built with Windows Server 2008 consists of between two and sixteen servers, called nodes; one of the main enhancements in cluster configuration on Windows Server 2008 is that servers do not need to be identical to each other. Windows Server 2008 introduces a new validation wizard that checks whether the system, storage, and network is suitable for the cluster. This validation check includes the following categories:

- ▶ **System configuration test:** Analyzes whether the server meets the requirements of the cluster, for example, operating system version and software updates.
- ▶ **Network test:** Checks whether the network meets specific requirements, such as network redundancy.
- ▶ **Storage test:** Analyzes whether the storage meets the requirements for the cluster, for example, if it can handle SCSI commands or cluster operations correctly.

These tests, along with a new improved setup process, make Windows Server 2008 clustering solutions a very good option for High Availability configurations.

**Note:** For more information about Windows Server 2008 Clustering new features and enhancements see:

<http://www.microsoft.com/windowsserver2008/en/us/clustering-home.aspx>

A high-speed shared disk sub-system is also mandatory and, in most situations, requires a SAN.

In a Windows cluster, SQL Server 2008 can be installed on each physical node. It is possible to install SQL Server multiple times on the same physical server or node. These installations are referred to as SQL Server instances and can be configured to run on all or some of the available cluster nodes. However, a SQL Server instance can only run on one node at a time.

Figure 3-14 shows a SQL Server cluster that has six instances of SQL Server installed, four being hosted on Node A and two on Node B. Each SQL Server instance can be considered to be a virtual server because each instance has its own unique server names and IP addresses and presents itself to users just like a standalone physical SQL Server.

This configuration is known as an *active-active* cluster because both nodes have active SQL Server instances. As we explained previously, in this kind of architecture it is very important to affinity instances so that the sum of all the instances on the cluster does not use more than 100% of the resources on the whole cluster. The reason for this is that in the case of a node failover one of the nodes has to support the workload of all the instances.

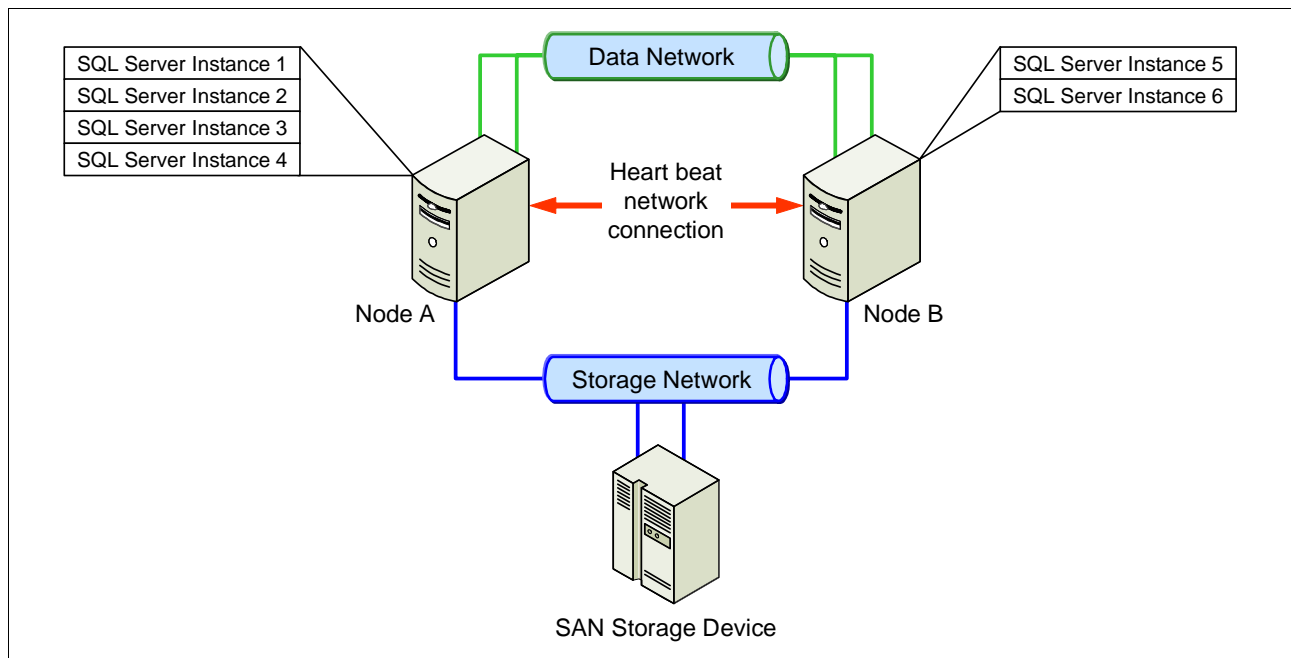


Figure 3-14 Two node basic active-active cluster configuration

When a cluster node suffers a failure, the cluster detects that there is a problem with the failed node. All SQL Server instances that were running on the failed node are restarted on the other available nodes in the cluster and service is restored. SQL Server 2008 has automatic database recovery features that do not require manual intervention in most cases.

Figure 3-15 on page 60 shows how a basic two-node cluster would look after Node A has failed. The four SQL Server instances that were running on Node A prior to the failure have been restarted on Node B. This occurred with no manual intervention. The disruption to service availability equals the length of time that it takes to restart the failed SQL Server instances plus the automatic database recovery time.

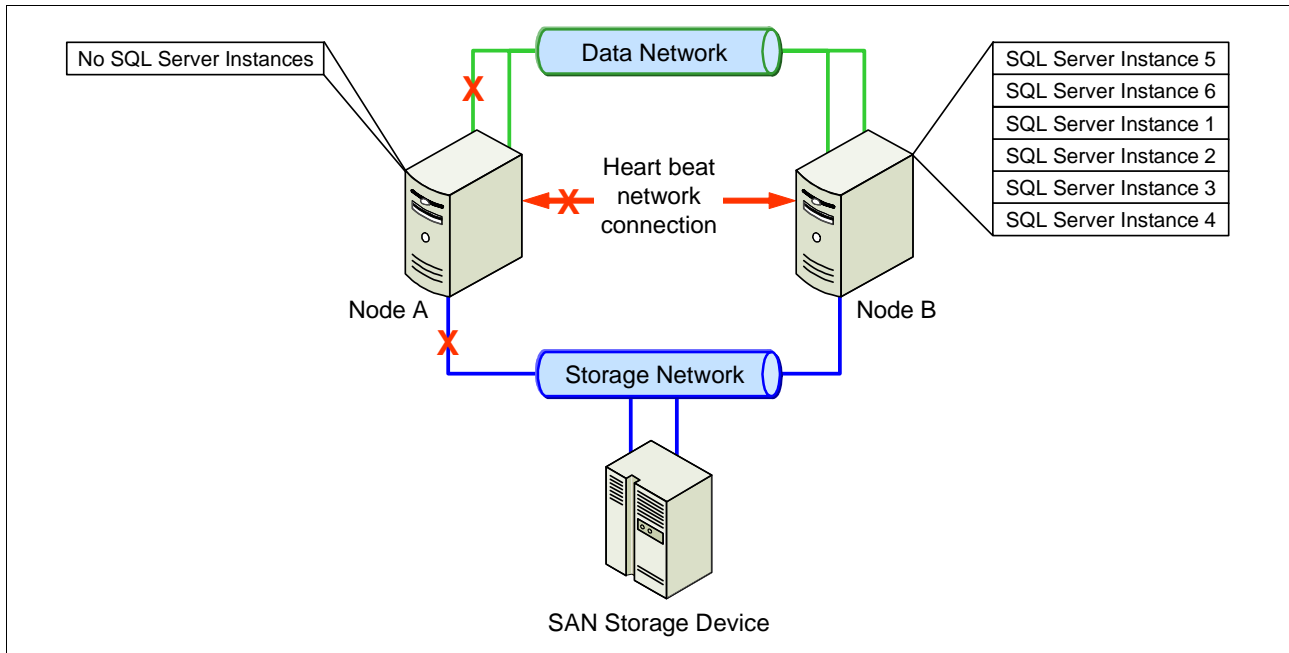


Figure 3-15 Two node basic active-active cluster configuration - Failure of Node A

### Database mirroring

In this section we cover the basic concepts of Database Mirroring architecture, operation modes, and the enhancements introduced with SQL Server 2008.

Database mirroring was first introduced in SQL Server 2005 as an option for High Availability for databases between heterogeneous servers. SQL Server 2008 introduces enhancements to performance over the previous version, and a new feature for automatic recovery from corrupted pages. We cover these features later on the section, but first we discuss the basic concepts of Database Mirroring architecture and operation modes.

With database mirroring you can take a database (the principal) and create a duplicate copy of it (the mirror) on another SQL Server server. SQL Server ensures that the principal and mirror are kept synchronized at all times. There are different levels of synchronization available, but essentially the choice is between more performance and less protection or more protection and less performance.

In its most basic configuration, database mirroring requires two servers, one for the principal and one for the mirror. The two SQL Servers ensure that all updates to the principal are reflected in the mirror. If the principal stops, user access is disrupted until the mirror is made active, which is a manual process. When the mirror is made active it assumes the principal role, and when the principal server is powered on again it assumes the mirror role.

Figure 3-16 on page 61 illustrates a basic database mirroring setup.



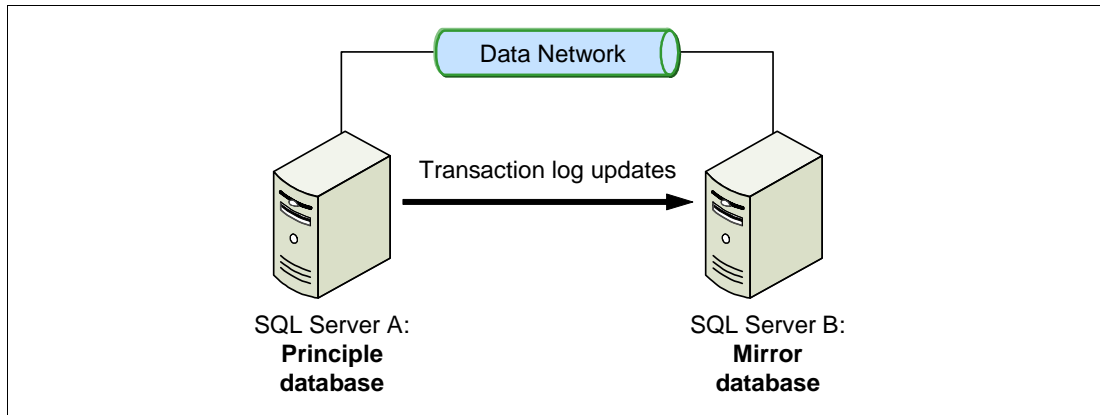


Figure 3-16 Basic database mirroring setup

Database mirroring can be set up between different server hardware. It can be performed between different physical sites (depending on network bandwidth requirements and availability), and does not demand a shared disk sub-system or dedicated heart-beat network connection. As a result, it can be a much cheaper alternative to creating a clustered environment while offering similar levels of protection in certain configurations.

This basic configuration has two possible synchronization modes.

- ▶ High-Performance mode
- ▶ High-Safety mode without automatic failover

High-Performance mode is the asynchronous database mirroring mode. In this mode SQL Server enhances performance on behalf of high availability. In High-Performance mode the principal server sends the log for a transaction to the mirror and the confirmation to the client without waiting for the mirror to acknowledge. Asynchronous operation permits the principal server to run with minimum transaction latency. This mode can be useful in disaster recovery scenarios where the servers participating in mirroring are separated by a significant distance or when there is low bandwidth between the servers. One issue in this mode is that the only failover procedure is by forcing service with possible data loss. In Figure 3-17 we describe the acknowledge process of the asynchronous mode.

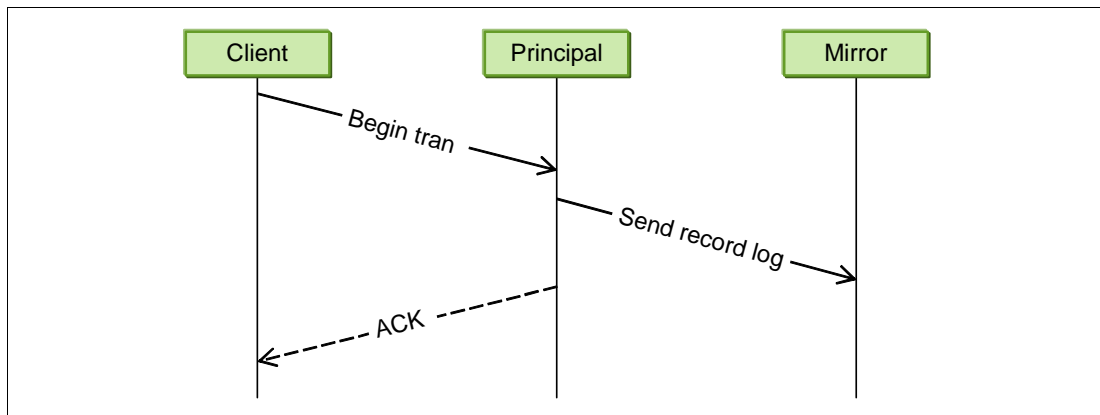


Figure 3-17 ACK process of Asynchronous mode

High-Safety mode without automatic failover is a synchronous database mirroring mode known simply as *High Safety*. In synchronous mode the operation is as follows:

1. The principal server receives the transaction from the client and writes the log for the transaction on the transaction log.
2. The principal server writes the transaction to the database and concurrently sends the log for the transaction to the mirror server. The principal waits for the acknowledge of the mirror server before confirming a transaction commit or rollback to the client.
3. The mirror server writes the log to disk and sends acknowledgement to the principal server.
4. When the principal server receives the acknowledgement of the mirror server, sends a confirmation to the client.

Figure 3-18 illustrates the acknowledge process of the synchronous mode.

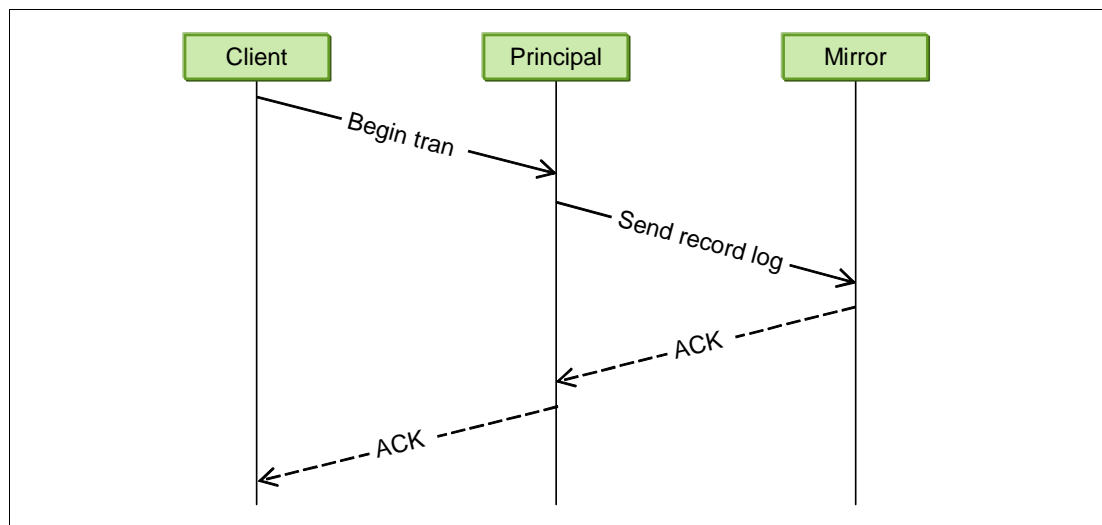


Figure 3-18 ACK process of Synchronous mode

The third mode for database mirroring is also a synchronous mode, but with automatic failover. This mode is named *High-Safety mode with automatic failover*. The process of synchronic acknowledge is the same as the one shown in Figure 3-18. Database mirroring needs a third server, known as *Witness*, to monitor the availability of both the principal and the mirror server to manage the automatic failover. The server must have SQL Server 2008 installed on it to be part of the mirroring solution, but this server can be used for multiple database mirroring setups. Figure 3-19 on page 63 shows how this might look.

**Note:** Because Database Mirroring is not a new feature on SQL Server 2008, it is possible that you have to upgrade an existing installation from SQL Server 2005 to SQL Server 2008. There are several scenarios for upgrading SQL Server 2005 to SQL Server 2008 when database mirroring is set up. For information on this topic search SQL Server 2008 Books Online for “How to: Minimize Downtime for Mirrored Databases When Upgrading Server Instances.”

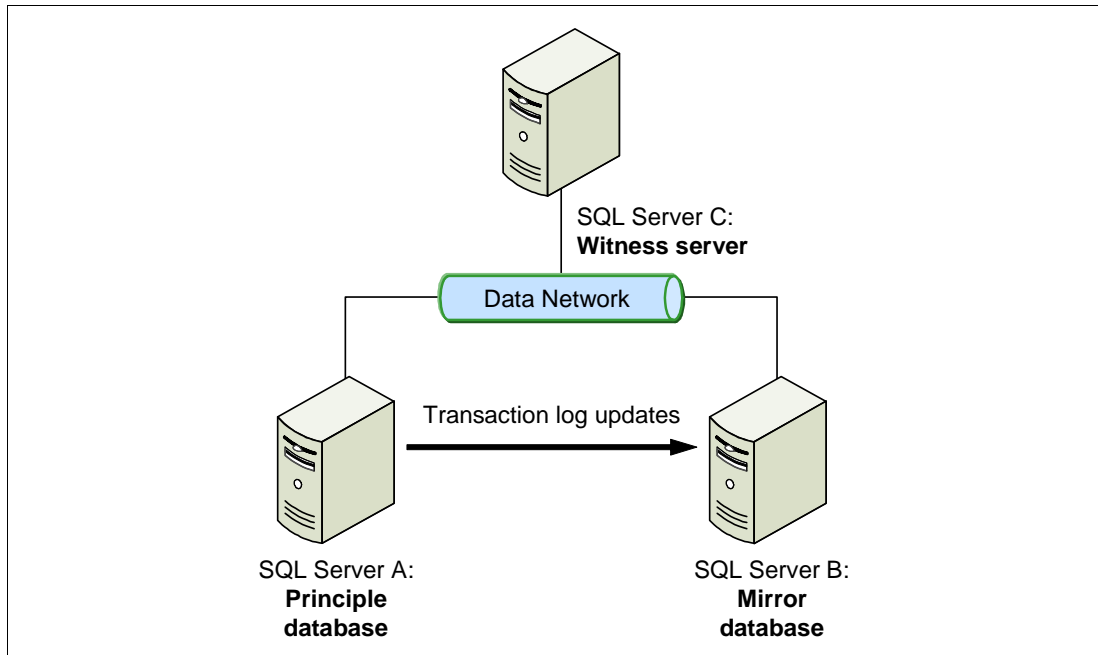


Figure 3-19 Architecture diagram of High-Safety Mode with automatic failover mirroring setup

The witness communicates regularly with the principal and mirror SQL Servers. If the witness detects that the principal is not available, it automatically makes the mirror active and accessible, much like a clustered environment would do.

Even with a witness server, if the principal stops, service availability is disrupted because any outstanding transactions must be applied to the mirror before it can be brought online. However, the SQL Server instance that hosts the database will already be running and Microsoft claims that database mirroring can provide quicker failover times than clustering. What determines failover time for mirroring is how long database recovery takes. This is a function of the amount of database update activity (actually, the number of transactions that have not yet been applied to the mirror copy of the database). Failover time does not depend on the database size.

SQL Server 2008 introduces enhancements to database mirroring. These performance enhancements are related to how the transactions are managed between the principal and the mirror server. The enhancements are the following:

- ▶ Write-ahead on the incoming log stream on the mirror server
- ▶ Improved use of log send buffers
- ▶ Compression of the stream of transaction log records
- ▶ Compression of stream data for which at least a 12.5 percent compression ratio can be achieved
- ▶ Page read-ahead during the undo phase

The new feature that SQL Server 2008 introduces in database mirroring is the possibility of automatic page repair during a database mirroring session. When one of the partners detects a bad page it requests a good copy of the page from the other partner. If the request works it replaces the unreadable page with the copy. A page recovery in the principal server is only possible when the state of mirroring is "SYNCHRONIZED."

When a page corruption occurs on the principal server, SQL Server 2008 inserts a row in the suspect\_pages table with the appropriate error status, marks the damaged page as “restore pending” (not accessible), and asks the mirror server for the page. If the mirror server can access the page it sends a copy to the principal server. The principal server recovers the page and if there are any deferred transactions pending it resolves them.

When the damaged page is on the mirror server, during the repair process the mirror session status is set to SUSPENDED until the page is recovered.

### ***Data backups***

In a consolidated server environment, your data storage is concentrated into a smaller number of repositories, and the data backup infrastructure must be able to process all backup tasks in the available backup window. Failure to complete backup jobs in the prescribed time frame might infringe on application usage windows and affect performance. Therefore, the backup infrastructure must be sized and configured to reflect the greater volume of data coming through a consolidated infrastructure.

SQL Server 2008 introduces enhancements to backups by enabling compression for database backups. This new feature provides benefits to the performance of backups by reducing the amount of I/Os. Therefore, if the backups are executed by network, it reduces the Network I/Os also.

One issue that appears with backup compression and other compression features on SQL Server 2008 is the increase of CPU consumption. As we discussed in 3.3.6, “Resource Governor” on page 44, we can set up workload groups for limiting the amount of CPU of a specific application in process. On the other hand, as we also explained previously, we can set up affinity on a specific IP Address or SQL Server network port for the backup application and limit the number of CPUs that this process will use.

### ***Network design***

We have covered providing two or more network cards in the consolidated server platform for fault tolerance; however, there might be a network throughput justification for using multiple network cards. Network interface cards (NICs) that are configured with teaming software can have greater effective bandwidth that prevents network I/O from being a system bottleneck. Consideration should also be given to building fault tolerance into the network infrastructure.

Implementation of a tiered network infrastructure can provide protection from the loss of network components that maintain service availability in this circumstance. Figure 3-20 on page 65 illustrates a simplified tiered network design. A server is connected to the red and blue sides of the network. If either server switch fails, the remaining switches maintain server connectivity in the network. The same is true for the core switches; the result is good fault tolerance in the network.

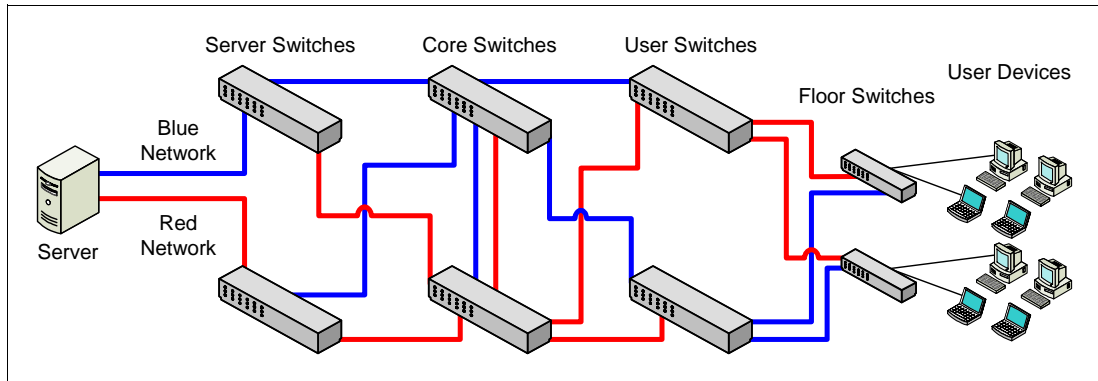


Figure 3-20 Tiered network design with fault tolerance

## Virtualization technologies

Server virtualization plays an extremely important role in the area of server consolidation. However, it is important to recognize that virtualization is concerned entirely with physical server consolidation and the removal of actual server boxes. It does not necessarily reduce the logical server count. There is little server management reduction after a successful virtualization project because the number of logical servers largely remains the same.

We recommend considering virtualization technologies for SQL Server in these situations:

- ▶ Test and development SQL Server environments. Virtual servers are ideal for delivering flexible and safe test and development environments for SQL Server and many other applications. You can install all SQL Server hotfixes and service packs on a virtual test SQL Server to test databases before they are installed in the production environment.
- ▶ SQL Server systems that have custom configuration requirements. Sometimes a small number of databases requires settings that are not compatible with other databases on the same physical server. The database might require a particular combination of sort order and collation, or maybe specific regional settings are required that the majority of other databases do not need. If the server resources required are low, it could be a viable solution to use a virtual server to host this type of environment.
- ▶ Replacing old server hardware when new server hardware does not support earlier operating system versions. As server hardware ages, the risk of hardware failure increases. Usually, when server hardware is refreshed, the operating system is upgraded to the current version. In some cases, that is not possible because of application compatibility issues or constraints. However, new server hardware often has limited support for previous versions of Windows Server. With a virtual server hosted on new server hardware, the current operating system can be maintained and the old server hardware could be removed after successful server virtualization.

We suggest that you also apply the following guidelines to your virtualization efforts:

- ▶ Avoid virtualizing applications that need high levels of CPU or I/O resources.
- ▶ Keep consolidated SQL Server environments and virtualized SQL Server environments on separate hardware. In other words, with hosted virtualization products like VMware® Server and Microsoft Virtual Server, do not run SQL Server in the host operating system.

A server virtualization project must be approached with the same rigor as a SQL Server consolidation project and the quality of the information gathering has a big influence on the outcome of the virtualization effort.

For further information on consolidation technics, see *Consolidating Microsoft SQL Server on the IBM System x3950 M2*, REDP-4385.





# Configuration

In this chapter, we discuss setting up and configuring the x3950 M2 to run SQL Server 2008 x64 on Windows Server 2008, with an emphasis on performance tuning. We cover these topics in the approximate order that you encounter them when you are setting up the complete environment: first the hardware, then the operating system, then SQL Server, then the storage, and, finally, the database server in operation.

The topics covered are:

- ▶ 4.1, “x3950 M2 configuration” on page 68
- ▶ 4.2, “Windows Server 2008 x64 configuration” on page 70
- ▶ 4.3, “SQL Server 2008 x64 configuration” on page 71
- ▶ 4.4, “Storage configuration” on page 74
- ▶ 4.5, “Database workloads” on page 77
- ▶ 4.6, “Performance analysis process” on page 79

## 4.1 x3950 M2 configuration

In this section, we first consider the physical hardware setup and then the BIOS settings.

### 4.1.1 System setup

When you set up the physical hardware, the best performance is achieved by provisioning each of the x3950 M2 nodes uniformly. This complements the NUMA architecture by providing local resources on each node.

In each x3950 M2 node, memory is installed in four memory cards, each of which has eight DIMM slots, for a total of 32 DIMMs. In order of importance, there are three aspects to the memory configuration that affect performance:

1. Install and use all four memory cards in every node
2. Balance the amount of RAM installed in every node
3. Populate every DIMM socket

As illustrated in Figure 1-2 on page 3, there are eight independent memory ports. Therefore, to optimize performance, you need to install four memory cards and then spread the DIMMs, installed in matched pairs, across all four memory cards before filling each card with two more DIMMs.

The optimal memory configuration uses every slot on every card on every node, with the same size DIMMs, so that the amount of memory is balanced across all the nodes. Other memory configurations that do not use all the memory cards or all the DIMM slots on the cards, or that are unbalanced across the nodes, penalize performance to some degree. In general, more memory is better for performance.

In x3950 M2 multi-node configurations, performance can be affected by the installation of PCI cards such as network adapters, Fibre Channel (FC) HBAs, and so on. To distribute the load equally, the optimal solution is to spread the adapters across all the nodes. For example, in a four node configuration with four FC cards, put one card in each node. Also, you should connect at least one Ethernet connection on each node. Each x3950 M2 has two integrated Gigabit Ethernet controllers.

### 4.1.2 Firmware update

After the hardware is physically set up, flash the firmware on all the nodes to the same supported level. Stability and performance issues can arise if the firmware is not the same on all nodes, or if it is not at a supported level.

#### UpdateXpress System Pack

We recommend using the UpdateXpress System Pack (UXSP) product to update the firmware. Follow this link:

<http://www.ibm.com/systems/support/>

Select **System x** ♦ **Product family** ♦ **System x3950 M2** ♦ **Operating system** ♦ **Windows Server 2008 x64** ♦ **Download**. Select the current System Update Package for your machine type.

This provides access to the downloadable System Pack, which is a bundle of firmware and Windows drivers for the x3950 M2 that have been tested together. Using the UpdateXpress System Pack Installer (UXSPI), you can create bootable media, for example, a CD or USB key, which can be used to apply the System Pack firmware on a server which has no



operating system installed yet. In addition, you can apply this package of updates to your server from within Windows. If your server is connected to the Web, UXSPI can compare the server's firmware and driver levels with the current levels on the IBM Support site and download and apply the firmware and drivers to replace those that are not current.

## Critical Updates

Some firmware may require individual, critical updates not included in the System Pack. The following list identifies the firmware that must be flashed and a recommended order in which to do this. Your configuration might include other devices that must also be updated.

1. BIOS
2. BMC firmware
3. Field Programmable Gate Array (FPGA) firmware
4. Remote Supervisor Adapter II (RSA II) firmware
5. SAS BIOS
6. SAS hard drives
7. ServeRAID firmware
8. Network adapter firmware
9. Diagnostics

If you need to apply firmware updates without using the UXSP product, you can use an RSA II remote console session using the remote diskette/ISO function. You can update BIOS, BMC, RSA II, Network adapter, SAS BIOS, ServeRAID, and SAS hard drives by using this method. Some firmware updates also are provided as standalone Windows packages.

**Note:** If you are running Windows Server 2008, Datacenter Edition, be sure that the firmware is certified for Datacenter.

## BIOS settings

When you make BIOS settings, keep them identical on all nodes. Begin by loading the default BIOS settings. The following parameters should be examined and tested with your workload to determine what maximizes performance.

### ► Processor hardware prefetcher

By default, hardware prefetching, which is what processors use to prefetch extra cache lines for every memory request, is disabled. Tests in the performance lab have shown that disabling this feature provides the best performance for many commercial application types with a random workload such as online transaction processing (OLTP).

A Business Intelligence (BI) workload may benefit from enabling this option, due to its sequential memory access pattern.

To enable hardware prefetching, go to BIOS Setup (press F1 when prompted at boot) and select **Advanced Setup** ♦ **CPU Options** and set **Processor Hardware Prefetcher** to **Enabled**. This setting affects all processors in the chassis. Perform the same change on each node.

### ► Processor adjacent sector prefetch

Also in **Advanced Setup** ♦ **CPU Options**, when this setting is enabled, the processor retrieves both sectors of a cache line when it requires data that is not currently in its cache. When it is disabled, the processor only fetches the sector of the cache line that contains the data requested. The default is disabled.

Enabling this setting might positively affect performance, especially a BI workload with a sequential memory access pattern. Typically, it affects certain benchmarks by a few percent, although in most real applications, the effect is negligible. This control is provided for benchmark users that wish to fine tune configurations and settings.

### ► **Memory array settings**

This setting is in **Advanced Setup ♦ Memory Settings ♦ Memory Array Setting**. The choices are:

- Use **High Performance Memory Array** (default), the best performance.
- Use **Hot Add Memory**, if you intend to add memory while the server is running, at the expense of reduced performance.
- Use **Full Array Memory Mirroring**, if you want the highest level of memory redundancy, at the expense of reducing available memory by half and reducing performance.

Each node of a partition in a multinode complex must have the same memory array setting.

For additional information about x3950 M2 settings, see *Planning, Installing, and Managing the IBM System x3950 M2*, SG24-7630.

## 4.2 Windows Server 2008 x64 configuration

In this section, we cover those aspects of the Windows operating system that are relevant to a server that is running SQL Server 2008. In general, no special settings are required. You are likely to find that leaving the defaults offers the best results.

### 4.2.1 Windows installation, service pack, updates, and drivers

After installing Windows Server 2008 x64, apply the most recent service pack, security patches, and currently supported drivers. Unsupported or back-level software can cause system instability, security vulnerabilities, and performance problems. Use UXSP to detect and apply driver updates. See “UpdateXpress System Pack” on page 68.

### 4.2.2 Windows settings

The Windows page file size is not important for performance. On a server dedicated to running SQL Server, there should be no paging. A modest-sized page file for taking mini-dumps, located on local storage, is sufficient. Complete system dumps of physical memory configurations of 64 GB and larger are impractical for debugging.

To optimize memory management and eliminate paging: Run gpedit.msc (Global Policy Manager) and click **Computer Configuration ♦ Windows Setting ♦ Security Settings ♦ Local Policies ♦ User rights assignment ♦ Lock pages in memory**, then add the SQL service group or user account.

For instant file initialization: Run gpedit.msc (Global Policy Manager) and click **Computer Configuration ♦ Windows Setting ♦ Security Settings ♦ Local Policies ♦ User rights assignment ♦ Perform Volume Maintenance**, then add the SQL service group or user account.

**Note:** Enabling instant file initialization accepts a small security risk. It might be possible to view data previously deleted on disk. See SQL Server 2008 Books Online topic “*Database File Initialization*.”

To maximize performance at the expense of energy savings, open **Control Panel ♦ Power Options**, then select the **High Performance** power plan.

To lower the priority of foreground programs: **Control Panel ♦ System ♦ Advanced system settings ♦ Advanced ♦ Performance ♦ Settings ♦ Advanced ♦ Background services**. The SQL Server installation configures this setting.

To reduce unnecessary processor usage: **Control Panel ♦ System ♦ Advanced system settings ♦ Advanced ♦ Performance ♦ Settings ♦ Visual Effects ♦ Adjust for best performance**.

Optionally, to avoid User Account Control prompts (consider security issues): **Start ♦ All Programs ♦ Administrative Tools ♦ System Configuration ♦ Tools tab**, Launch Disable UAC, and then reboot the system.

Consider configuring Soft NUMA to partition the cores and memory for specific workloads. See 3.3.3, “Soft NUMA” on page 38.

### 4.2.3 Anti-virus software

There are steps you can take to prevent anti-virus software from impacting database operations and performance.

If you decide to install anti-virus software on the server running SQL Server, configure it so that updating the virus signatures and scanning the system occur during periods of low levels of activity. Exclude the database file extensions MDF, LDF, and NDF from the virus scan, or exclude the folders that they are stored in. This prevents the SQL Server from trying to open SQL Server files that the anti-virus software has already opened, which can result in the database being marked as suspect.

See the Microsoft Knowledge Base article *Guidelines for choosing antivirus software to run on the computers that are running SQL Server* for more information:

<http://support.microsoft.com/kb/309422>

## 4.3 SQL Server 2008 x64 configuration

This section covers installing and configuring SQL Server 2008 x64.

### 4.3.1 SQL Server 2008 installation, service packs, and hot fixes

Before installation, create one or more domain user accounts to be used as SQL Server service logins, if you want SQL Server 2008 to communicate with other clients and servers.

Install SQL Server 2008, and install any applicable service packs, cumulative updates, and hot fixes.

See Appendix A, “Unattended install” on page 87 for an example showing how to complete an unattended installation from the command line. An unattended installation provides a quick way to reinstall the original configuration during a disaster recovery process and by its nature forces you to document the options used during installation. It provides a way to reliably install a large number of SQL Server instances which have identical configurations. It is even possible to install clustered instances and add and remove nodes using an unattended install.

## 4.3.2 SQL Server 2008 settings

After SQL Server is installed, consider adjusting the following configuration settings. SQL Server does excellent self-tuning. In general, the default settings work best.

**Note:** These settings can be made using the `sp_configure` command. Some of these options can be set only after configuring **show advanced options** as 1.

### Affinity mask

This configuration option restricts a particular SQL Server instance so that it runs on a subset of the cores. If you are running only one instance of SQL Server 2008 and nothing else, then allowing SQL Server to use all cores creates the best performance. When the server has more than 32 cores, you need to use both the affinity mask and the affinity64 mask options.

Multiple instances with heavy resource demands should be configured to partition the hardware resources. Use **affinity mask** and **max server memory** to place each instance within a node, on its own node, or on a group of nodes, for best performance.

**Note:** There is a difference between running with the default, `sp_configure 'affinity mask', 0`, and affinitying all the processors. For example, on a 16 core server, `sp_configure 'affinity mask', 0xFFFF` prevents the SQL threads from moving between the cores. The threads are pinned to the core they start on. All the cores can be used by SQL Server; however, the threads cannot move between the cores as they can when the affinity mask is set to zero. In some cases, affinitying all the processors can provide a performance benefit because it reduces context switching, which occurs when threads migrate between cores.

### Affinity I/O mask

This configuration option provides a way to isolate a portion of the I/O processing within a SQL Server instance and place it on one or more dedicated cores. I/O affinity is the ability to dedicate a scheduler to the processing of I/O requests. Normally, in the default case, when this mask is set to zero, I/O requests are processed on the core they are initiated on. Generally this gives the best performance. However, in some workloads with high I/O rates, overall performance can be improved by dedicating one or more cores to exclusively processing I/O requests and nothing else. When the server has more than 32 cores, you need to use both the affinity I/O mask and the affinity64 I/O mask options.

**Note:** Do not enable the same core in both the affinity mask and the affinity I/O mask. Another way to say this is that the bit for a core must either be zero in both masks, or it must be one in one mask and zero in the other mask. It cannot be one in both masks.

An example of a valid method of using affinity masks for a 16 core server is as follows:

```
exec sp_configure 'affinity mask', 0xFFFFE
exec sp_configure 'affinity I/O mask', 0x0001
```

This correctly sets core 0 for I/O request processing and cores 1 through 15 for other (non-I/O) SQL Server processing.

An example of an *invalid* method of using affinity masks for a 16 core server is as follows:

```
exec sp_configure 'affinity mask', 0xFFFFE
exec sp_configure 'affinity I/O mask', 0x0003
```

This attempts to set cores 0 and 1 for I/O request processing and cores 1 through 15 for other (non-I/O) SQL Server processing. This setting is incorrect because it assigns core 1 to both I/O request processing and other (non-I/O) SQL Server processing.

## **AWE**

It is not necessary to configure AWE for SQL Server 2008 x64. 64-bit applications do not require AWE because access to memory is not limited to 4 GB. For 32-bit versions of SQL Server this setting allows SQL Server to use up to 64 GB of memory. Using this feature is advisable only when a value is specified for Max Server memory.

## **Max degree of parallelism**

Setting this option to **1** prevents the query optimizer from choosing parallel query plans. Parallel query plans reduce the time that it takes to complete a query at the expense of increased resource utilization and reduced total throughput. Using multiple processors to run a query is usually not desirable in an OLTP workload, although it is desirable in a BI workload. Note that it is possible to assign query hints to individual queries to control the degree of parallelism. This is useful for a mixed workload with both OLTP and BI.

## **Max server memory**

When max server memory is kept at the default setting, SQL Server acquires and frees memory in response to internal pressure (to expand because of work requests) and external pressure (to shrink from the operating system). On a dedicated server that is running a heavy workload, SQL Server continues to acquire memory until the maximum amount of physical memory is nearly reached. In this case, you might find that guaranteeing memory for the operating system, by setting **max server memory** to 2 GB to 4 GB less than the maximum physical memory, gives better performance.

The typical recommendation is 2 GB less than total physical memory on systems with up to 64 GB, 4 GB less than total physical memory on systems with 128 GB or more, and an amount between 2 GB and 4 GB less than total physical memory on systems between 64 GB and 128 GB.

Multiple instances should always be configured with **max server memory** set so that the instances do not compete for memory.

This parameter only effects the size of the buffer pool cache. SQL Server allocates memory for other purposes, so the actual amount of memory consumed by SQL Server will be larger than the amount specified. Max server memory is divided equally between the nodes. For example, 252 GB on a 2-node server would result in a per node maximum buffer pool cache size of 126 GB.

## **Network packet size**

Setting this option to 8192 may improve performance because of better page alignment with SQL Server's 8KB pages. The default setting, 4096, is best for most applications. Test this to find the optimal setting.

## **Priority boost**

Setting this option to **1** changes the base priority of SQL Server from 7 to 13. On a dedicated server, this might improve performance, although it can also cause priority imbalances between SQL Server functions and operating system functions, leading to networking or other errors. **Priority boost** should not be used when you are implementing failover clustering.

## Recovery interval

The default is 0, indicating automatic configuration. In practice, this generally means a recovery time of less than one minute and a checkpoint approximately every minute for active databases. Keep **recovery interval** set at 0 (self-configuring) unless the checkpoints impair performance because they occur too frequently. If so, try increasing the value in small amounts. See SQL Server Books Online topic “*Understanding Recovery Performance*.”

## Large pages

Large-page support might benefit a dedicated x64 SQL Server with a large memory configuration. Large pages improve performance by increasing the efficiency of the translation look-aside buffer (TLB) in the CPU. Large pages can be used for the buffer pool and for the code pages for SQL Server.

To enable large pages for the buffer pool use trace flag -T834. This causes SQL Server to allocate the maximum memory specified at startup.

To enable large pages for the code pages create this registry key:  
HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Image File Execution  
Options\sqlservr.exe. Insert this new registry DWORD:UseLargePages=1.

## 4.4 Storage configuration

In this section, we describe how you can configure the disk storage for SQL Server if you are connecting the x3950 M2 to an FC-attached SAN. A SAN offers great configuration flexibility. Similarly, SQL Server 2008 offers great flexibility because you can put database objects in multiple files and group files into multiple file groups. Each of the various database workloads (OLTP, BI, and the maintenance operations), imposes a characteristic I/O profile on the storage. Over time, new workloads can be introduced that might alter the I/O profile.

One of the most important and difficult design decisions that affects performance in a database implementation is how to make the best use of the flexibility offered by the SAN and SQL Server to maximize performance for the unique I/O profile exhibited initially and in the future. The problem has a chicken and egg quality because the configuration of the storage is based on the I/O profile, but the I/O profile can best be determined by measuring it on a running system.

For starters, try to model your proposed storage configuration on paper, using a spreadsheet or a software tool designed to do this type of modeling. You can also attempt to extrapolate from the performance characteristics of an earlier implementation of the proposed environment. If this is too complicated or you do not know the characteristics, then you can use the rules that are described in the sections that follow.

The Microsoft white paper *Predeployment I/O Best Practices* discusses many of the storage issues discussed in this section. It also includes links to other resources - tools and papers - on this topic. It is available from:

<http://sqlcat.com/whitepapers/archive/2007/11/21/predeployment-i-o-best-practices.aspx>

The IBM white paper *SQL Server 2005 with IBM System Storage DS8000 & DS4000 Best Practices Guide* provides design, configuration, and planning information for implementing Microsoft SQL Server with IBM System Storage™ arrays. This paper is planned to be updated for SQL Server 2008 and the DS5000 storage product. It is available from:

<http://ibm.com/support/techdocs/atmsastr.nsf/WebIndex/WP101126>

## 4.4.1 Storage setup

The steps that are required to configure and present the Logical Unit Numbers (LUNs) to the operating system as physical disks are:

- ▶ Host to storage connection

This addresses the paths between the x3950 M2 and the storage server. The components of these paths are the FC HBAs in the x3950 M2, the cables, the FC switches, and the storage server controllers. The guiding principle from a performance perspective is to spread the peak I/O traffic evenly through these components. From a redundancy perspective, no single component should cause failure, which implies a minimum of two HBAs, two switches, two controllers, and at least two paths to each LUN.

- ▶ Create arrays

An array is composed of a set of disk *spindles*, a RAID level and a *stripe unit size*. In general, a greater number of spindles provides greater throughput (I/O per second and MB per second). You will get better performance, if you plan to use significantly less than the total disk capacity. RAID-10 usually performs better than RAID-5 (especially for random write workloads) while providing approximately half of the capacity and more resiliency to disk failures.

A large stripe unit size benefits a sequential workload, and a small stripe unit size benefits a random workload. If you have no other information, a stripe unit size of 64 KB is advisable as a starting point from which you can tune. For a more in-depth discussion of choosing these parameters, see the Microsoft white paper, *Disk Subsystem Performance Analysis for Windows*, available from:

[http://www.microsoft.com/whdc/device/storage/subsys\\_perf.msp](http://www.microsoft.com/whdc/device/storage/subsys_perf.msp)

- ▶ LUNs

LUNs are created from arrays. Using an entire array to create a single LUN is the simplest and most advisable design because multiple LUNs created from a single array can interfere with the performance of the other LUNs, which share spindles. The likely result is erratic performance. One LUN per array also simplifies performance monitoring.

- ▶ Configure controller cache

In an enterprise-level SAN, significant cache is available, which can enhance performance greatly. In general, take the database option, when one is offered. This includes settings such as:

- Read caching ON
- Cache read-ahead multiplier OFF (if the access is random)
- Write caching ON (be sure the cache is battery backed to avoid data corruption or power failures)

- ▶ Creating Windows logical disks

When you create partitions in Windows Server 2008 they are automatically aligned. Use an NTFS allocation size of 64 KB.

**Note:** SQL Server 2008 supports mount points. Mount points reduce the number of drive letters required, which can be a limitation when building a multi-instance cluster.

- ▶ Stress and performance test

To validate that the storage you have configured is reliable for use with SQL Server, you can run SQLIOSim for several days, to detect rare, intermittent failures. You can obtain SQLIOSim from:

<http://support.microsoft.com/default.aspx?scid=kb;en-us;231619>

To verify the performance characteristics of the volumes that you have created, you can use the SQLIO utility, available from:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=9A8B005B-84E4-4F24-8D65-CB53442D9E19&displaylang=en>

Using SQLIO, you can put a load on the storage system based on the I/O profile that you anticipate. You should keep a baseline set of statistics for each LUN (for example, 8 KB, 64 KB, 1 MB, random/sequential, read/write, and various queue depths) that you can use to compare against actual SAN performance when you run the database application.

**Note:** The utilities mentioned here do not use SQL Server. They are standalone programs that generate a synthetic load on the storage system. Also, it is not advisable to use them to stress the OS boot drives (the default for SQLIO), only the FC-attached storage.

Another tool, similar to SQLIO, is IOMeter, which you can download here:

<http://www.iometer.org/>

## 4.4.2 Storage file placement

This section addresses placing the various database files on the logical disks that appear in the operating system. There are several conflicting goals to meet: availability, performance, and cost. Designing the LUNs, the database files, and their placement for a large SAN and a large complex database is a difficult task. In fact, there are so many details, complex interactions, and possibilities, that automation of the design or implementation can be very effective.

Automation is even more important when the workload is dynamic because the storage configuration can be modified frequently while the database application is running. The following general guidelines supplement whatever level of automation is available:

- ▶ Availability

To maximize availability, the log and backup files should be placed on arrays that are separate from the data files. This is because you can recover from an outage caused by media failure of the data files by using the log and backup files to restore the database. To reduce the chance of media failure, use RAID-10 over RAID-5. To reduce backup time and recovery time, which contributes to availability, use multiple data files and multiple backup devices to allow parallel data transfer to all devices simultaneously. To reduce recovery time, use RAID-10 and put the source and target files for these operations in separate arrays.

- ▶ Performance

To maximize performance, all files should be put in RAID-10, especially data files that are subject to random workloads. To reduce disk contention, data and log files for the same database should be put in separate arrays. A large component of disk access time is moving the disk arm, so isolating a log file in its own arrays allows sequential writes to proceed with minimal disk arm movement, improving performance.



Dividing a database into multiple file groups and putting them in separate arrays can improve performance by distributing the I/O load over more spindles. One approach is to separate base tables and their non-clustered indexes into separate file groups and place them on separate arrays. The query in Example 4-5 on page 82 provides I/O statistics for balancing I/O in this way.

Tempdb, which is a global resource for each SQL Server instance, can cause a scalability bottleneck. Concurrent access to tempdb can be improved by splitting the tempdb data into multiple files. Begin by creating one file per CPU. Use the query shown in Example 4-1 to check for contention in Tempdb. Tempdb should be on its own RAID-10 array, separate from the user databases, for best performance.

*Example 4-1 Tempdb query*

---

```
select session_id, wait_duration_ms, resource_description
  from sys.dm_os_waiting_tasks
 where wait_type like 'PAGE%LATCH_%'
    and resource_description like '2:%'
```

---

You can improve performance by sizing data, log, and tempdb files so that they do not have to grow automatically during normal operations. A starting point for sizing the log of a user database is to make it 20% of the database size if the database is less than 10 GB, and to make it 10% of the database size if the database is more than 10 GB. A starting point for Tempdb is to make it 20% of the largest database in the instance and make the Tempdb log 1/2 the size of Tempdb itself. Arrays that are well below their maximum space capacity perform better than ones that are nearly full.

► Cost

When it is not economical to put each file in its own array, consolidating files with similar workloads is another option. For example, combine read-only data files in the same array, randomly accessed data files in the same array, and log files in the same array. Avoid putting files with disparate workloads in the same set of spindles. Combine files that are accessed at different times of the day in the same array. When it is not economical to use RAID-10, RAID-5 for the data files and backups might be acceptable.

Try to match the performance requirements of the various files with the performance characteristics of the arrays, based on the measurements you made with SQLIO. The goal is to avoid wasting performance capability on files that do not need it.

## 4.5 Database workloads

This section identifies typical database workloads and their I/O, CPU, and memory characteristics.

### 4.5.1 Maintenance operations

Maintenance operations include create, load, back up database, restore, integrity check, and index rebuild. These operations can be characterized as heavy sequential read and write that use large block sizes. Usually they are scheduled during off-peak periods and the primary issue is how quickly they can complete.

With the exception of integrity check and index rebuild, which have a CPU-intensive and memory-intensive middle phase, these operations are limited by the performance of the disk arrays involved and require little CPU or memory resources. It is possible to shorten the elapsed time of these operations by implementing multiple file groups on separate devices

and performing the operation in parallel for each file group. See, for example, SQL Server Books Online topic “Optimizing Backup and Restore Performance.”

Backup compression is a new feature in the SQL Server 2008 Enterprise Edition. Backup compression compresses database backups going to disk and tape. It trades off increased CPU for reduced disk or tape space required for backups. Because less data has to be read and written, backup compression may significantly reduce the time required to back up and restore a database.

## 4.5.2 Application workloads

There are two main types of database application workloads: a BI workload and an OLTP workload. Some environments operate a mixed workload with both types.

### OLTP

An OLTP workload is characterized by random read/writes against the data files and sequential writes against the log file. Usually there is no tempdb activity. Because of the checkpoint process, the write activity to the data files occurs in spikes, usually once a minute for an active database.

To support the database recovery function, which depends on rolling forward the log files, the log must be periodically copied to another location. This is a sequential read activity that must occur simultaneously with the continuous sequential write activity to the log.

### Decision Support System (DSS)

A DSS workload is characterized by heavy sequential reads that use large block sizes from the data files and little or no activity against the application database log file. There is usually heavy sequential read/write use of tempdb when large intermediate data sets cannot be stored in memory and must be sorted. There might be phases of high CPU usage, when the working data is able to fit in memory.

### Data compression

Data compression is a new feature in the SQL Server 2008 Enterprise Edition which is similar to backup compression, discussed previously. Data compression compresses databases (actually tables and indexes) on disk. It trades off increased CPU for reduced disk space. Because less data has to be read, data compression may significantly reduce the time required to query a database.

Figure 4-1 on page 79 shows a comparison of SQL Server 2008 DSS query performance with and without compression. The horizontal line at 100% is the baseline performance with no compression. The blue, burgundy, and yellow bars show normalized query execution time, read Mbytes per query, and query CPU consumption with compression relative to the results without compression, respectively. In this study, the original database size was 1665 GB and the on disk compressed size was 955 GB, a 43% reduction of space used.

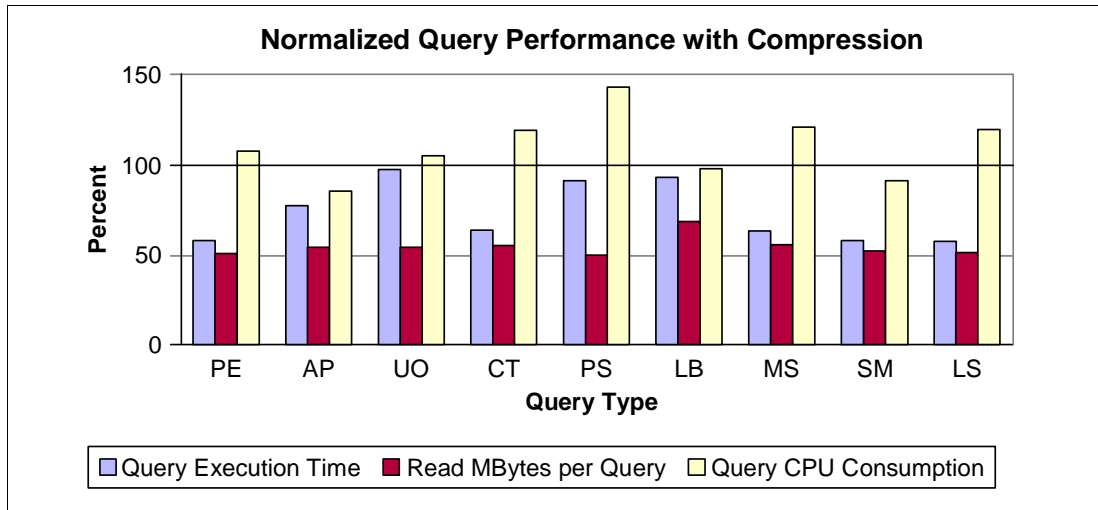


Figure 4-1 SQL Server 2008 DSS query performance with and without compression

### Mixed OLTP and DSS

When both OLTP and DSS workloads are run concurrently, there can be physical and logical contention. For example, a DSS query might be reading the same data that an OLTP transaction is updating. This is precisely the scenario that the database snapshot isolation feature was created for. By keeping row versions of the data in tempdb, SQL Server 2008 can avoid reader-writer blocking. The impact on the hardware is then a heavier use of tempdb.

Contention for CPU resources can arise within a single instance. In this case, it can be resolved by using the Soft NUMA feature to partition CPUs by workload in a single SQL Server instance as described in 3.3.3, “Soft NUMA” on page 38.

## 4.6 Performance analysis process

In this section, we describe the process of analyzing performance. This analysis process includes the practice of keeping a performance baseline, how to monitor the hardware resources on a server that is running SQL Server, and the other SQL Server specific tools that are available.

**Important:** Although this paper focuses on hardware, we have observed that, while it is possible to make improvements in the 10% to 50% range by tuning hardware, it is also possible to make orders of magnitude of improvement by performing tuning activities at the database and application levels.

### 4.6.1 Performance baseline

Keeping a performance baseline is a basic best practice. SQL Server database administrators (DBAs) often create a separate database that contains baseline information about how an application runs hour to hour, day to day, week to week, and month to month. Then, they can query this database to find averages and trends to create a profile of their applications when they are running well.

You can learn the profile of your application by determining:

- ▶ Typical transaction rate

- ▶ Typical I/Os per second; read and write to database, log, and tempdb files; and the random/sequential pattern
- ▶ Typical CPU utilization, user and kernel
- ▶ Typical memory utilization

The concept of keeping a *baseline* is very general. It includes keeping performance logs and Excel® spreadsheets (in reality, any repository of historical measurements) on any component, including the operating system, the database management system, the application, and the hardware. The value of keeping this historical information is that when something changes unexpectedly in performance, you can refer to this baseline data and find clues in what has changed from then to now. This can lead more quickly to a resolution of the problem, than if you did not have this baseline data.

One procedure for collecting a performance baseline is as follows:

1. Create a Counter Log, called *Baseline\_A*, using the Performance Monitor GUI with a few key counters that monitor CPU, memory, disk, and network.
2. Run the command shown in Example 4-2 from the command line, all on one line.

*Example 4-2 Settings for capturing baseline performance data*

---

```
logman update counter Baseline_A -v mmddhhmm -b 1/1/2015 0:00:00 -e 1/1/2015
23:59:59 -r -cnf 4:00:00 -si 00:15 -f tsv -o "C:\Perflogs\Baseline_A" -u
Administrator *
```

---

After you have run the command and when you start the Counter Log, your statistics are collected every 15 seconds and they are saved in a log file named with the current date/time stamp in the C:\Perflogs directory. It closes the current log file every four hours and creates a new log file. It runs until you stop it. You can create another Counter Log, called *SQL\_A*, and include some of the SQL Server counters such as **Transactions/sec** for your databases. *SQL\_A* might include more counters than *Baseline\_A*. This way, you have a relatively small counter log for quick diagnosis and a larger one for more complete analysis.

## 4.6.2 Hardware resources

In this section we discuss monitoring the four hardware resources (CPU, memory, disk, and network) from the perspective of SQL Server 2008 running its typical workloads.

### CPU

If the total processing percentage (**% Processor Time**) is higher than 80% for sustained periods, then you might have a CPU bottleneck. Normally the kernel mode time (**% Privileged Time**) is low for SQL Server. If it is high, there might be a problem with a disk driver or the disk subsystem.

To see if SQL Server has work queued that could run, but is waiting for a busy CPU, you must check inside SQL Server. This is because SQL Server does not queue its work to the Windows operating system. SQL Server has a user-mode scheduler in which it queues tasks that are waiting.

This query results in an instantaneous view of the number of tasks queued in SQL Server that can be run, as shown in Example 4-3.

*Example 4-3 Tasks that can be run in SQL Server*

```
select AVG (runnable_tasks_count)
  from sys.dm_os_schedulers
 where status = 'VISIBLE ONLINE'
```

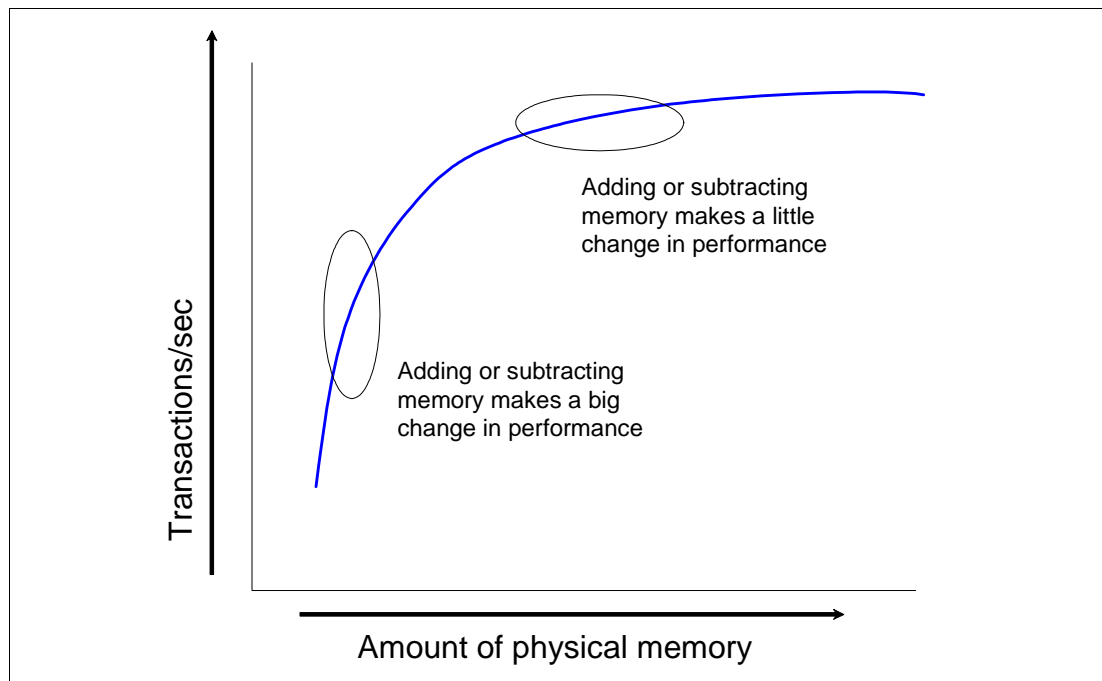
If SQL Server frequently has a non-zero number of runnable tasks, then adding additional processors will likely result in more throughput.

## Memory

For a server that is running only SQL Server, there should be no paging. That is, **Memory\Pages/sec** should be zero. Most of the memory that SQL Server uses is allocated for the Buffer Pool, which consists of 8 KB pages. **Buffer Manager\Database pages** gives the number of buffer pages that contain database content.

Take the number of buffer pages, multiply it by 8 KB, and compare it to the total amount of physical memory in the server. This gives you an indication of how much of your physical memory is actually being used. Use DBCC MemoryStatus to obtain a snapshot of how memory is allocated on each of the NUMA nodes and how it is consumed by the pools and caches in SQL Server.

Memory usage in SQL Server 2008 is very complex. However, there is a simple procedure for determining whether a workload will benefit from additional memory. It is based on the observation that most applications respond to increased memory according to the general graph shown in Figure 4-2.



*Figure 4-2 How more memory affects performance*

The procedure is to measure the application performance with two amounts of memory, your current amount of memory, and a small amount less. If the difference in performance is not great, then that means you are at the top part of the curve and adding more memory does not improve performance much. On the other hand, if the difference in performance is great, then you are at the bottom part of the curve and adding memory is likely to improve performance, up to the point where the curve begins to flatten out.

The Perfmon counters that can indicate memory pressure, in the **SQL Server:Buffer Manager** object, are:

- ▶ Buffer cache hit ratio (low)
- ▶ Page life expectancy (low)
- ▶ Checkpoint pages/sec (high)
- ▶ Lazy writes/sec (high)

The best way to monitor NUMA memory usage is to use the Perfmon counters in the **SQLServer:BufferNode** object because it shows memory usage by NUMA node.

## Disk

If CPU utilization is low, but response time or run time is not okay, you might have a disk bottleneck. The **Physical Disk** Perfmon counters might provide the insight needed for tuning most of the sequential I/O workloads. However, in the OLTP workload, the disk load of the log and checkpoint processes are adjusted by SQL Server and so are invisible to Perfmon. When writing to the log, the disk queue length is kept at 1. Likewise, the checkpoint is controlled so that it does not swamp the disk and severely reduce throughput. In this case, use the query shown in Example 4-4 to capture reads, writes, and I/O stalls by database file.

*Example 4-4 SQL Server I/O statistics by file*

---

```
select m.name, v.*, m.physical_name
  from sys.dm_io_virtual_file_stats (null, null) v
      ,sys.master_files m
 where v.database_id = m.database_id
      and v.file_id   = m.file_id
```

---

Using this information, you can compute reads/sec, writes/sec, read and write block sizes, and average milliseconds of wait time (I/O stalls) for read and write. If these wait times are large, putting the files in faster disk arrays can improve database performance.

Using one of the new dynamic management views (DMV), it is now possible to obtain I/O statistics by individual table and index, which can be used to make decisions about dividing up the tables into file groups and for application level tuning. See the query shown in Example 4-5. See also “Dynamic management views (DMV)” on page 84.

*Example 4-5 Query I/O statistics by table and index*

---

```
select *
  from sys.dm_db_index_operational_stats(null, null, null, null)
```

---

## Network

Network performance is rarely an issue with SQL Server. There is a new Perfmon SQL Server counter object, **Wait Statistics**, that includes the counter **Network IO waits**, which can prove useful. However, be aware that network waits can occur because the client is not consuming the packets fast enough, which would not indicate a bottleneck in the server hardware.

### 4.6.3 Other diagnostic and performance tools

The tools discussed in this section are useful for troubleshooting and tuning performance.

## SQL Trace and SQL Profiler

SQL Trace and SQL Profiler are related tools. SQL Trace is the name for the tracing facility that is configured by using a set of system stored procedures. This facility does not drop trace events when the server is under stress (unlike SQL Profiler). SQL Profiler is a GUI application front end for SQL Trace.

SQL Trace and SQL Profiler are good tools to use to find poorly performing queries, to determine the cause of deadlocks, or to collect a sample workload to replay for stress testing. A trace template can be used to collect workloads in the format needed by the Database Engine Tuning Advisor.

## Database Engine Tuning Advisor

Database Engine Tuning Advisor (DTA) is a tool for tuning the physical design of a database. DTA recommends changes in indexes and partitioning for an existing database based upon a typical workload (a SQL trace) that is collected from the production system. You can use DTA to identify indexes that are not being used by your application. These indexes can be dropped because their maintenance creates unnecessary overhead. You can use DTA to identify new indexes that currently do not exist. These indexes can be added, potentially reaping large performance benefits.

## Management Data Warehouse

SQL Server 2008 provides a complete framework for monitoring, centrally collecting, and reporting on server activity, query statistics, and disk usage data for all SQL Server instances in the enterprise. The database administrator uses wizards to configure data collection on each SQL Server instance, uploading on a periodic schedule to a central SQL Server instance which hosts the Management Data Warehouse (MDW) relational database. Default schedules provide for purging old data. The entire framework is highly customizable. User defined collectors and user defined reports can be added and collection and purging schedules can be tailored.

Data is collected from T-SQL queries such as DMVs and custom queries, SQL Tracing, and Perfmon counters. The initial reports provide interactive drill down from summary level charts to detailed breakouts with numeric quantities, query text, and query plans. The purpose of this framework is to aid the database administrator in managing a large farm of SQL Server instances efficiently and effectively. The data and reports can be used for capacity planning, performance tuning and troubleshooting.

## SQL Server Management Studio (SSMS) Standard Reports

Right-clicking an object in the left pane of a SQL Server instance in SSMS, then **Reports**, then **Standard Reports**, reveals a set of troubleshooting and monitoring reports based on the default trace and dynamic management views. The default trace is a minimal log of significant changes occurring in the SQL Server instance which is always running by default.

For a particular SQL Server instance the reports include:

- ▶ **Server Dashboard:** Configuration details, including non default settings, and summary CPU and IO usage.
- ▶ **Configuration Changes History and Schema Changes History:** To help answer the question: What has changed recently?
- ▶ **Memory Consumption and Scheduler Health:** How memory and CPU resources are utilized.
- ▶ **Activity reports:** Selected session activity such as blocking, cursors, and top resource usage.

- ▶ Performance reports: Selected query statistics such as average and total CPU and average and total IO.

For a Database instance the standard reports give per database information on disk usage, backup and restore events, transactions, index usage, and user activity.

There are standard reports for Logins, Management, Notification Services and SQL Server Agent. For complete documentation on all the standard reports, see:

<http://blogs.msdn.com/buckwoody/archive/2008/04/17/sql-server-management-studio-standard-reports-the-full-list.aspx>

## Dynamic management views (DMV)

One of the design goals of SQL Server 2008 is for users to be able to effectively troubleshoot and tune their databases. DMVs provide this visibility to the internal workings of SQL Server 2008.

The server-wide DMVs are:

- ▶ **sys.dm\_db\_**

These DMVs are for databases that provide space and usage information by file, index, partition, task, and session. Note that queries that return index fragmentation statistics can cause intensive I/O on that index.

- ▶ **sys.dm\_exec\_**

These DMVs are for execution related information, including query plans, cursors, and sessions.

- ▶ **sys.dm\_io\_**

These DMVs provide I/O information for data and log files, pending I/O information, and tape status.

- ▶ **sys.dm\_os\_**

These DMVs are for SQLOS related information, including memory, scheduling, tasks, and wait statistics.

- ▶ **sys.dm\_tran\_**

These DMVs are for transaction-related information including active transactions, locking, snapshots, and the version store. Note that selecting row version information could return many rows and be resource intensive.

- ▶ **sys.dm\_xe\_**

These DMVs are for viewing Extended Event packages and sessions. Extended Events provide a low level, low overhead framework for collecting performance and troubleshooting data which can be integrated with data collected from the operating system. The monitoring can be very precisely targeted. For example, it is possible to collect the wait events for a single query execution.

These are the component-specific DMVs:

- ▶ **sys.dm\_broker\_** for the Service Broker feature
- ▶ **sys.dm\_cdc\_** for the Change Data Capture feature
- ▶ **sys.dm\_clr\_** for the Common Language Runtime feature
- ▶ **sys.dm\_db\_mirroring\_** for the Database Mirroring feature
- ▶ **sys.dm\_fts\_** for the Full Text Search feature
- ▶ **sys.dm\_qn\_** for the Query Notifications feature
- ▶ **sys.dm\_repl\_** for the Replication feature
- ▶ **sys.dm\_resource\_governor\_** for the Resource Governor feature



Catalog views expose static metadata that describes all the user viewable objects in a SQL Server instance. For example, `sys.master_files` gives all the database file names that are known to the SQL Server instance. By combining catalog views with dynamic management views, you can create queries that interpret the internal data for troubleshooting and performance analysis.

## SQLdiag

The SQLdiag utility, which has its roots in the Microsoft Product Support Services group (actually first written by Ken Henderson), is a general purpose diagnostic tool. It collects performance logs, event logs, SQL traces, SQL blocking data, and SQL configuration data.

An especially useful mode for using this tool is to start it, reproduce an issue, and shut it down. The captured information that results from this mode can be analyzed for troubleshooting. You can extensively customize the data SQLdiag collects.

## Query hints and plan guides

Query hints are not new in SQL Server 2008. They can be used to force the query optimizer to choose a specific query plan. They are useful when the optimizer occasionally does not choose the most efficient plan. Plan guides are an extension of this.

Using a plan guide, it is possible to modify a query, providing query hints, without changing the text of the query. This is useful when you have a third party application and do not want to or are unable to modify the code.

New in SQL Server 2008 is the ability to specify a specific query plan as a query hint either directly or via a plan guide.

## Replay Markup Language (RML) utilities for SQL Server

The RML utilities are free, unsupported tools from the Microsoft Product Support Services group. The RML utilities provide a framework for replaying and stress testing a SQL Server workload based on a captured SQL Server trace. The ReadTrace utility preprocesses the captured trace into .rml files, so that it can be efficiently replayed. The OSTRESS utility accepts these .rml files and replays the workload in the original sequence or in stress mode. ReadTrace can also populate a database from the trace data, which can be analyzed using the Reporter utility.

OSTRESS can also be used stand alone to run the same query on multiple connections at once. See Example 4-6, which performs a query on the same table from 500 TCP/IP connections.

*Example 4-6 Using OSTRESS to run a query on 500 connections*

---

```
ostress.exe -Stcp:"DB3" -q -E -dNetStress -Q"select count(*) from wait_stats"  
-n500 -o"C:\Temp\ostress_tcp"
```

---

The 9.x version of the utilities can be used with SQL Server 2008; however, they do not support the new features of SQL Server 2008. The RML utilities can be obtained here:

<http://support.microsoft.com/kb/944837>

## Windows Performance Toolkit (xPerf/ETW)

Windows Performance Toolkit provides two main programs; xperf and xperfview. Xperf captures traces using the Event Tracing for Windows (ETW) infrastructure. Xperfview reads the ETW trace and presents interactive graphs and summary tables. These tools can be used to measure and analyze system and application performance.

The toolkit can be obtained from:

<http://msdn.microsoft.com/en-us/performance/cc825801.aspx>

### **Performance Analysis of Logs (PAL) tool**

PAL is an unsupported tool from CodePlex, Microsoft's open source project hosting Web site. The idea behind PAL is to use the power of the computer to read through a Perfmon log to locate when a particular counter has exceeded a threshold, so that this does not have to be done manually. Perfmon templates and predefined threshold values for the major Microsoft products are provided, including SQL Server. The exceptions located are presented via an HTML report with charts and graphs.

PAL can be obtained here:

<http://www.codeplex.com/PAL>



# A

## Unattended install

SQL Server 2008 introduces enhancements to the entire installation process. In this appendix we describe the improvements to the command prompt installation or unattended installation.

There are two ways to perform an unattended installation of SQL Server 2008, both of which are covered in this appendix. You can install SQL Server 2008 by:

1. Using a configuration file
1. Passing all options as parameters on the command line

Topics in this appendix are:

1. A.1, "Benefits of unattended installation" on page 88
1. A.2, "Installation using the configuration file" on page 88
1. A.3, "Installation from the command prompt" on page 90

## A.1 Benefits of unattended installation

SQL Server 2008 command prompt installation, or unattended installation, is a powerful tool for administrators. The use of unattended installation has the following advantages:

- ▶ It provides an easy and fast way to reinstall a SQL Server during a disaster recovery situation.
- ▶ It provides a reliable way to install a large number of SQL Servers with identical configurations. With Windows scripting it is possible to install SQL Server, Service Packs, configure security, maintenance jobs, and so forth, with one click or by deploying an SMS installation package.
- ▶ It eliminates the need for repetitive DBA tasks and thus the chance for human mistakes during the configuration of a new SQL Server instance or client.
- ▶ It helps you keep up to date with new installations by adding new SQL Server fixes to the unattended installation scripts.
- ▶ The unattended installation configuration file is a self-documenting file of how SQL Servers are installed.

## A.2 Installation using the configuration file

Before installing this way, you must create the configuration file. You can create the configuration file in one of two ways: either automatically as a result of running the SQL Server 2008 install GUI (setup), or you can create the file manually.

### A.2.1 SQL Server 2008 installation GUI

Despite the numerous changes on SQL Server 2008 GUI (setup), this section only covers the features of setup that help with the unattended installation process.

SQL Server 2005 GUI Installation originally included enhancements on error logging under the bootstrap folder. However, the GUI also introduces, as part of a normal installation, the generation of the configuration.ini file with all the features selected during the normal procedure as shown in Figure A-1 on page 89.

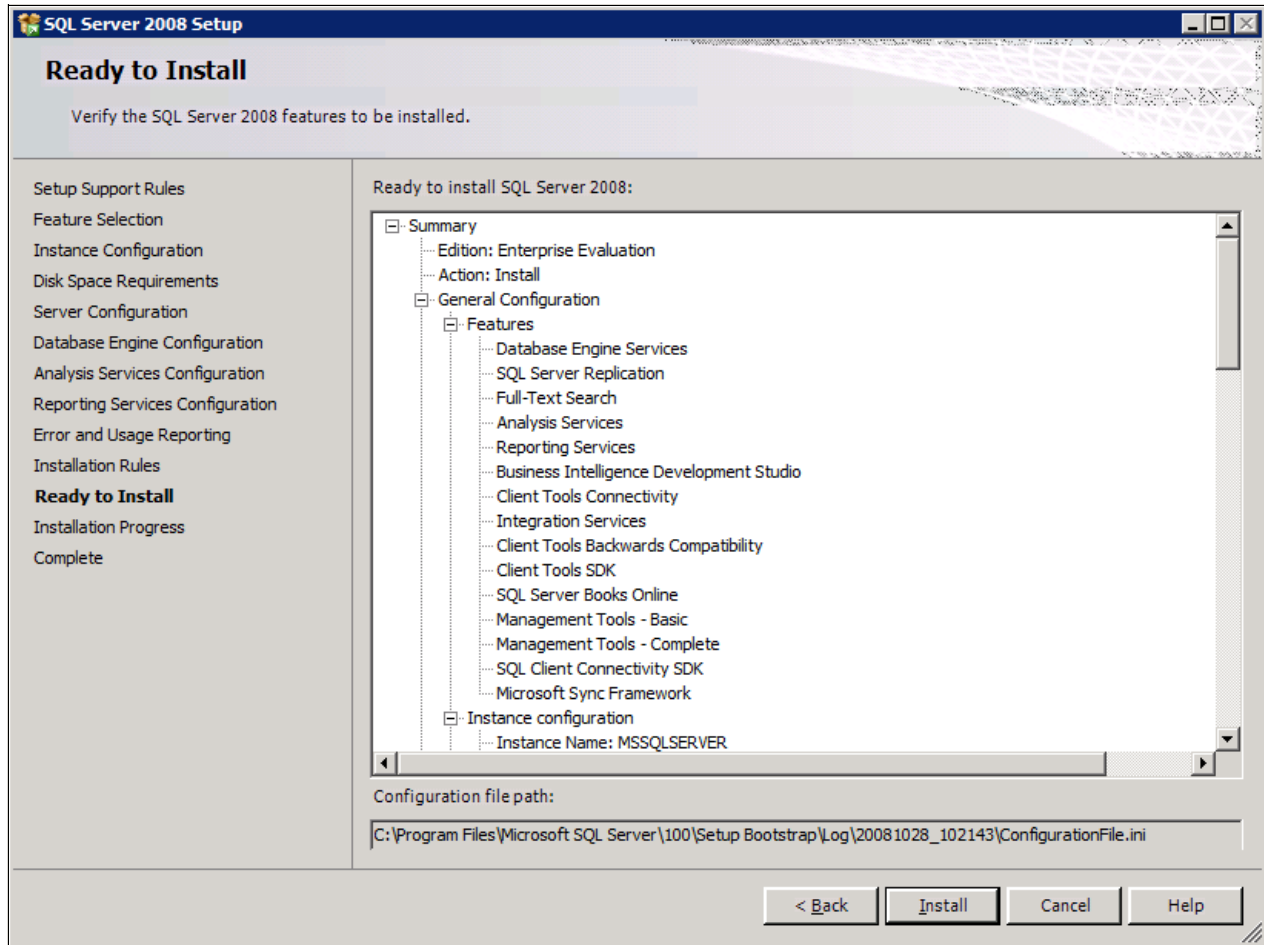


Figure A-1 Configuration.ini file generation

The location of the configurationfile.ini file is also displayed in this dialog.

**Tip:** The name configurationfile.ini for the file is just for the example. The name does not need to be configurationfile.ini or configuration.ini, but it is recommended that the extension be *ini*.

## A.2.2 Manual configuration file generation

The unattended installation procedure allows you to install any SQL Server 2008 component. The components for the installation are specified as parameters in a Configuration.ini file, or as parameters on the command prompt (that is, without Configuration.ini file). The parameters are divided in groups according to the features of SQL Server 2008:

- ▶ SQL Server Setup
- ▶ Analysis Services
- ▶ Database Engine
- ▶ Integration Services
- ▶ Reporting Services
- ▶ SQL Server Browser
- ▶ SQL Server Agent

For more information on SQL Server 2008 group of parameters see SQL Server 2008 Books On-line under “How to: Install SQL Server 2008 from the Command Prompt”

The ini file for the Command Prompt installation process has no different structure than a normal ini file.

The file needs a header between brackets and a row per option selected for the installation. In Example A-1 we are creating Configuration.ini file for a simple installation of Database Engine with Error Reporting option enabled, informing installation directories, named instance name, and so forth.

*Example A-1: Example of Configuration.ini file format*

---

```
[SQLSERVER2008]
ACTION="Install"
ERRORREPORTING="1"
FEATURES="SQL"
INSTALLSHAREDDIR="C:\Program Files\Microsoft SQL Server"
INSTALLSHAREDWOWDIR="C:\Program Files (x86)\Microsoft SQL Server"
INSTANCEDIR="E:\Program Files\Microsoft SQL Server"
INSTANCENAME="CMDINSTALL"
...
```

---

### A.2.3 Performing the install

Now that the configuration file has been created, it is simply a matter of issuing the setup command from the directory where the setup files are located (either the supplied installation media or the network share where the files are stored). The command is:

```
Setup.exe /q /ACTION=Install /Configurationfile=<path to configuration file>
```

## A.3 Installation from the command prompt

In this section, we describe how to install SQL Server 2008 from the command prompt, without the use of a configuration file.

The first step is to define the parameters and their values. In this example we install a Database Engine named instance with the name CMDINSTALL. In Example A-2 we are specifying the parameters for the SQL Server Setup, Database Engine, and SQL Server Agent groups.

*Example A-2: Command prompt installation parameters*

---

```
SQL Server Setup:
/ACTION=Install
/ERRORREPORTING=1
/FEATURES=SQL
/INDICATEPROGRESS
/INSTALLSHAREDDIR="C:\Program Files\Microsoft SQL Server"
/INSTALLSHAREDWOWDIR="C:\Program Files (x86)\Microsoft SQL Server"
/INSTANCEDIR="E:\Program Files\Microsoft SQL Server"
/INSTANCENAME="CMDINSTALL"
/Q
```

```
Database Engine:
/INSTALLSQLDATADIR="E:\Program Files\Microsoft SQL Server\CMDINSTALL"
/SECURITYMODE
/SQLBACKUPDIR
/SQLCOLLATION="SQL_Latin1_General_CP1_CS_AS"
/SQLSVCAccount="DOMAIN\SQLServiceAccount"
/SQLSVCPASSWORD="PASSWORD"
/SQLSVCStartuptype=2
/SQLSYSADMINACCOUNTS="LIST SYSADMIN ACCOUNTS"
```

```
SQL Server Agent
/AGTSVCAccount="DOMAIN\SQLServiceAccount"
/AGTSVCPASSWORD="PASSWORD"
/AGTSVCSTARTUPTYPE=2
```

Now, to run the setup installer, we supply all these parameters on the command line. In our example, the command line is shown in Example A-3 (the entire command is entered on one line).

*Example A-3: Command to install SQL Server 2008*

```
setup.exe /ACTION=Install /ERRORREPORTING=1 /FEATURES=SQL /INDICATEPROGRESS
/INSTALLSHAREDWOWDIR="C:\Program Files\Microsoft SQL Server"
/INSTALLSHAREDWOWDIR="C:\Program Files (x86)\Microsoft SQL Server"
/INSTANCEDIR="E:\Program Files\Microsoft SQL Server" /INSTANCENAME="CMDINSTALL" /Q
/INSTALLSQLDATADIR="E:\Program Files\Microsoft SQL Server\CMDINSTALL"
/SQLCOLLATION="SQL_Latin1_General_CP1_CS_AS"
/SQLSVCAccount="FINSQL\SQLServiceAccount" /SQLSVCPASSWORD="sqlSQL123"
/SQLSVCStartuptype=2 /SQLSYSADMINACCOUNTS="FINSQL\kurlat"
/AGTSVCAccount="FINSQL\SQLServiceAccount" /AGTSVCPASSWORD="sqlSQL123"
/AGTSVCSTARTUPTYPE=2
```

Once you press Enter to begin the install, you will see output similar to Figure A-2.

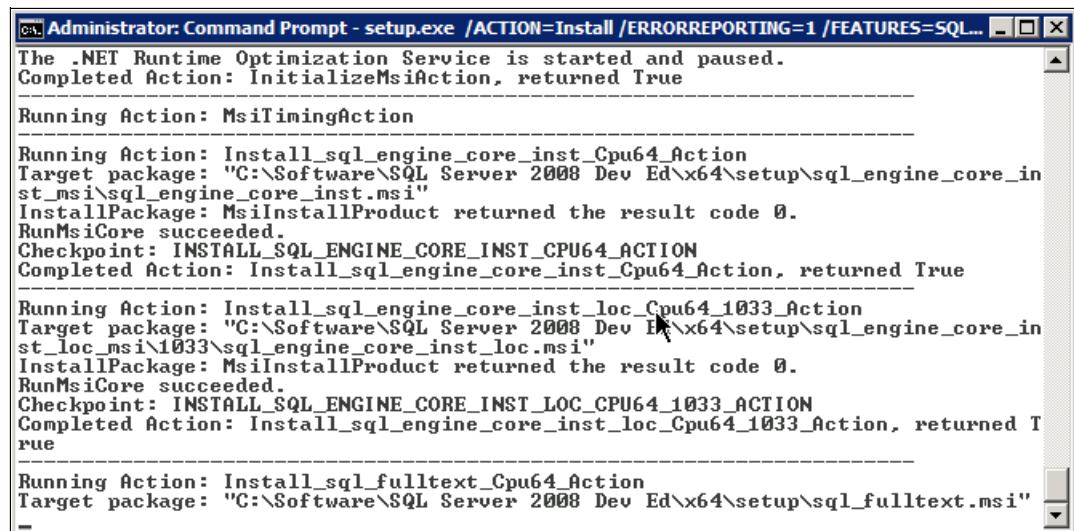


Figure A-2 Setup progress

Once the process completes, the output is similar to Figure A-3 on page 92.

```
Administrator: Command Prompt
INSTALL\ConfigurationState, Name = SQL_FullText_Adv
Completed Action: FinalizeTimingConfigAction, returned True
-----
Running Action: NotifyProgressComplete
Completed Action: NotifyProgressComplete, returned True
-----
Running Action: ProduceStatusLogsBeforeFinishPage
Completed Action: ProduceStatusLogsBeforeFinishPage, returned True
-----
Running Action: FinalizeProgressStatus
Completed Action: FinalizeProgressStatus, returned True
-----
Running Action: RebootMessageAction
Completed Action: RebootMessageAction, returned True
-----
Running Action: CloseUI
Stop action skipped in UI Mode Quiet
Completed Action: CloseUI, returned True
-----

Setup result: 0
C:\Software\SQL Server 2008 Dev Ed>
```

Figure A-3 Ended process

**Tips:**

- ▶ By default, the installer logs everything to the folder c:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log\<YYYYMMDD\_HHMMSS\.
- ▶ The command line installation creates a configuration file for later use. This generated file is stored in the same folder as the log.

### A.3.1 Command prompt SQL Sever 2008 fix installation

As with any other component on SQL Server 2008, a fix also might be installed within command prompt. The syntax for a fix installation is:

```
FIXINSTALLATIONFILE.exe /Q /INDICATEPROGRESS="True" /INSTANCENAME="NAMEINSTANCE"
```



# 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 publications

For information about ordering these publications, see “How to get Redbooks publications” on page 93. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *IBM System x3950 M2 and x3850 M2 Technical Introduction*, REDP-4362
- ▶ *Planning, Installing, and Managing the IBM System x3950 M2*, SG24-7630
- ▶ *Consolidating Microsoft SQL Server on the IBM System x3950 M2*, REDP-4385
- ▶ *SQL Server 2005 on the IBM eServer xSeries 460 Enterprise Server*, REDP-4093
- ▶ *Introducing Windows Server x64 on IBM @server xSeries Servers*, REDP-3982

## Online resources

These Web sites are also relevant as further information sources:

- ▶ IBM System x3950 M2 Product Overview  
<http://ibm.com/systems/x/hardware/enterprise/x3950m2/>
- ▶ Microsoft SQL Server 2008 Books Online  
<http://technet.microsoft.com/en-us/library/ms130214.aspx>

## How to get Redbooks publications

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

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

## Help from IBM

IBM Support and downloads

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

IBM Global Services

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







# Running Microsoft SQL Server 2008 on the IBM System x3950 M2



**Utilize the enterprise class performance features of IBM X-Architecture servers**

The IBM System x3950 M2 server provides extraordinary levels of scalability and processing power needed to run the most demanding enterprise database workloads. The x3950 M2 system is based on the fourth generation of IBM X-Architecture and is built on eX4 technology. As a scale-up platform, it offers unprecedented flexibility for supporting multi-node configurations using up to 96 processor cores.

**Maximize scale-up performance in multi-node configurations**

The System x3950 M2 is the ultimate in scale-up design. Scale-up refers to the idea of increasing processing capacity by adding additional processors (and memory and I/O bandwidth) to a single server, making it more powerful. This is precisely what the x3950 M2 is designed to do: to scale up by adding chassis to a hardware partition to form two-node, four-node, and eight-node configurations. SQL Server 2008 is also designed for scale-up.

**Introduces new features in SQL Server 2008**

Thus, when your database application needs to scale-up to handle increased demands, the x3950 M2 and SQL Server 2008 together can grow from a one-node server to an enterprise-class four-node server with up to 96 processor cores and 1 TB of physical memory, which the 64-bit SQL Server 2008 x64 can take full advantage of.

This IBM Redbooks publication presents technical considerations and configuration examples for scaling up Microsoft SQL Server 2008 to support very high workloads.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

### **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

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

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