
Part I: SAP Architecture and SQL Server Basic Configurations, Features Used, and Windows Configurations

SQL Server Technical Article

Writer: Juergen Thomas
Content Contributors: Martin Merdes
Reviewers: Hermann Daeubler, Martin Merdes, Elke Bregler, Clas Hortien, Mark Weber

Published: June 2009
Edition: Third Edition
Applies to: SQL Server 2008 and SQL Server 2005

Summary: This white paper in its third edition describes best practices that customers, system integrators, and partners can use to design and install more reliable, high availability SAP implementations that deliver improved performance, scalability, and security by using SQL Server 2005 and SQL Server 2008. The paper describes typical architectures, installation and configuration, and performance monitoring and tuning. After a lot of debates, the decision was made by the author not to restrict the third edition for SQL Server 2008 only, but to take the second edition of the paper and extend it with the new features of SQL Server 2008 and also add findings and experiences made during the last 3 years of customers running SAP applications on SQL Server 2005.
Copyright

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft, Excel, Hyper-V, SQL Server, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.
# Table of Contents

**Executive Summary** ................................................................. 1

**Introduction** .................................................................................. 2

- SAP-Microsoft Alliance ................................................................. 2
- SAP Solutions and SQL Server ................................................................ 3
**SAP Multilevel Client/Server Architecture** ..................................... 4

- SAP NetWeaver Application Server .................................................. 5
  - What Is an SAP System? ................................................................. 6
  - SAP NetWeaver Application Server Architecture ................................ 6
  - SAP NetWeaver Application Server with Windows ................................ 8
- User and User Request Assignment to SAP ABAP Processes .......... 10
- Memory Management of the SAP ABAP Engine ................................ 10
  - Windows Page File ........................................................................ 11
- SAP Java Application Server .......................................................... 12
  - User and User Request Assignment to SAP Java Processes ............ 12
  - Java Versions Used ........................................................................ 12

**SAP Implementations on SQL Server** ......................................... 13

- SAP Connections to SQL Server (ABAP Side) .................................. 13
- Security Context of SAP ABAP Connection ....................................... 14
  - SAP Connections to SQL Server (Java Side) ..................................... 15
- SAP Statement Execution ................................................................. 15
- SAP Database Schema ..................................................................... 17
  - Collation and Code Page .................................................................. 18
- Upgrading from SQL Server 2005 or SQL Server 2000 to SQL Server 2008 ... 20
  - High-Availability Upgrade to SQL Server 2008 ................................ 21
  - Support of Earlier SAP releases ....................................................... 23
- SAP Unicode Implementation on SQL Server .................................... 23
- Migration Considerations for SAP Products ....................................... 24

**Hardware Selections** .................................................................. 27

- Hardware Tendencies ....................................................................... 27
- Microsoft x64 Computing Platform ................................................. 28
- Microsoft IA64 Computing Platform ............................................... 28
- SAP Applications on a 64-Bit Platform ............................................. 29
- Memory Volumes ............................................................................. 29
- Storage Selection ............................................................................. 30
- General Remarks for Sharing Hardware Resources Between Different SAP Systems 31

**Installation and Configuration** ..................................................... 32

- SQL Server Installation with SAP .................................................... 32
Executive Summary

Companies face numerous challenges managing and integrating information across enterprise business processes. Customers need faster analysis and deeper business insight to improve their decision making and to respond to changing business needs. Companies frequently choose an Enterprise Resource Planning (ERP) solution to fulfill these business requirements. One of the leading ERP application providers is SAP AG. Their flagship applications are the NetWeaver based SAP ERP/Business Suites and SAP® R/3® industry solutions.

SAP Business Solutions are composed of a comprehensive range of products that empower the enterprise with a flexible, end-to-end solution. NetWeaver based SAP solutions can increase business productivity, enhance operational efficiency, and improve the Total Cost of Ownership (TCO). NetWeaver based SAP applications also offer the scalability needed to manage ever-increasing workloads.

A critical challenge in implementing an SAP solution is the selection of a data platform to deliver the advanced features and capabilities needed to support the most demanding workloads. The Microsoft® SQL Server® database software (either SQL Server 2008 or SQL Server 2005) is the RDBMS of choice for deploying secure, reliable, highly available, high-performing, and scalable SAP installations.

SQL Server 2008 and SQL Server 2005 provide an integrated data management and analysis solution. SQL Server enables SAP customers of varying sizes to share data across multiple platforms, applications, and devices, while making it easier to connect to internal and external systems. SQL Server high-availability features can minimize downtime in SAP implementations. SQL Server improves productivity by making it easier to create robust database extensions at a lower cost.

SAP solutions running on SQL Server realize native performance improvements. SQL Server contains built-in tools that simplify installation and make it easier to deploy and manage SAP implementations. In addition, the SQL Server engine dynamically tunes database parameters automatically in response to changing usage characteristics.

This series of white papers describes best practices that customers, system integrators, and partners can use to design, deploy, and operate high availability SAP implementations with SQL Server 2008 and SQL Server 2005. This series of papers is provided to highlight the common aspects of SAP with SQL Server 2008 and SQL Server 2005 implementations, as they relate with the specific characteristics of SAP business applications.

The papers describe typical architectures, installation, configuration, performance monitoring, performance tuning, and some elementary troubleshooting recommendations. They assume that the reader has at least a general understanding of SAP ERP solutions and SQL Server database concepts and features. The best practices described in this white paper were developed using the combined experiences of thousands of SAP customers worldwide.
Introduction

This is the third edition of a white paper that first started in 2005. Requirements and conditions of operating SAP landscapes on the Windows® operating system with SQL Server have become more feature rich and complex. Part of the increasing complexity comes from customers using more and more SAP products. But, beyond this, ever larger customers are also using SQL Server as the main database platform for their SAP landscape in very complex situations. And on the other side, SQL Server 2008 offers more features as well which can help SAP customers lower their TCO again. Hence some decisions writing this paper had to be taken. One was whether to concentrate entirely on new features of SQL Server 2008. We decided not to do so, but to take the existing second edition of this paper as base of the third. This, however, led to the result that the areas covered could easily generate a book of several hundred pages. But this would be opposed to the goal to provide this information for free to our customers or potential customers who are thinking about deploying SAP on Windows and SQL Server. Therefore, to cover a substantial number of areas and keep the white paper in a halfway digestible format, we decided to split it into three parts. So there will be three separate papers in this series of white papers:

Part I: Architecture, Microsoft (both SQL Server and Windows) Configurations for SAP including new SQL Server 2008 features
Part II: Configuration of SAP and High-Availability of SAP/SQL Server landscapes
Part III: SAP/SQL Server Performance Tuning and Monitoring

This paper represents Part I of this series.

SAP-Microsoft Alliance

Since 1993, SAP and Microsoft have been working together to provide a deeply integrated Microsoft platform and SAP solutions. SAP and Microsoft have a strong, long-term relationship that is driven by customer satisfaction.

As a result of this close cooperation; Microsoft is currently the most selected platform for R/3 and SAP application deployments:

- More than 56,000 SAP application installations run on Windows, which is more than all other platforms combined. In addition, over 50 percent of all new SAP deployments run on Windows.
- More than 23,000 SAP application installations worldwide are running with SQL Server as the RDBMS. The SQL Server installed base with SAP customer experienced a constant large growth due to a high percentage of new SAP systems being installed on SQL Server.
- The number SAP installations using SQL Server has grown in every quarter since 1993.

In addition, SAP and Microsoft are positioned to provide integrated business value. Examples include using SQL Server Business Intelligence (BI) features such as Reporting Services to directly access SAP Business Information Warehouse (BW) or
Integration of SQL Server Integration Services with the SAP Open Hub Services interface and Staging BAPI interface of SAP BW.

These activities only present a small part of the common efforts between Microsoft and SAP. Another great example is the common development of Duet.

**SAP Solutions and SQL Server**

SAP and Microsoft have been working together to develop tight integration between SAP solutions and SQL Server:

- **SAP R/3 for Microsoft Windows NT®** 3.51 was released in 1994. By mid-1995, SAP began using R/3 with SQL Server 6.0 and SQL Server 6.5 in customer implementations.

- In response to SAP customer demand, in 1998 Microsoft released SQL Server 7.0 with features that improved scalability and reliability on SAP systems on SAP systems. SQL Server 7.0 is the first SQL Server version on a complete new code base. SQL Server 7.0 was designed to accommodate feature requirements that made it a better platform for SAP.

- The release of SQL Server 2000 included features that dramatically improved the performance and administration on SAP systems. Using standard benchmarks, SQL Server 2000 performed best on a commodity hardware platform running the Windows Server® operating system. SQL Server 2000 also was the first SQL Server released as 64-bit version. This expanded the range of Windows/SQL Server into high-end database computing.

- At the end of 2005, Microsoft shipped SQL Server 2005. The first SQL Server release covering 2 different 64-bit platforms with x64 and IA64 plus a 32-bit platform with x86. SQL Server 2005 offered a major step in functionality beyond SQL Server 2000. This functionality enhanced the way SAP applications are run and administrated.

- Shipping SQL Server 2008 in summer 2008, Microsoft concentrated on the efficiency of storage, driving TCO low by reducing storage costs. Other features like transparent data encryption (TDE) or auditing addressed growing concerns around abilities to secure data. With the SQL Server 2008 release, x64 established itself as the new commodity 64-bit platform.

**SQL Server 2008 Enterprise and Windows Server 2008 for SAP**

SQL Server 2008 Enterprise is a comprehensive, integrated end-to-end data solution that delivers a more secure, reliable, and productive platform for enterprise data and BI applications. SQL Server 2008 delivers new and improved features that are tightly integrated with SAP products based on the following support considerations:

- For SAP products, SQL Server 2008 is supported using Windows Server 2003 with Service Pack 2 (SP2) and Windows Server 2008. SQL Server 2008 for SAP only is on 64-bit computing platforms like x64 and Itanium 64 (IA64). 32-bit is not supported for SQL Server 2008 in combination with SAP. This pays tribute to the steadily
increasing demand on memory for workloads generated by SAP as well as the development of x64 as an inexpensive 64-bit industry standard computing platform.

- SQL Server 2008 is qualified for use on SAP products that run on the NetWeaver 7.0 platform of SAP and later releases of SAP like SAP Business Suite 2008.
- The introduction of SQL Server 2008 does not change the support matrix of SQL Server 2005 with SAP products. SQL Server 2005 is supported from SAP R/3 4.6C up to the most recent releases of SAP.
- At release of this paper, SAP supports Windows Server 2008 and the Hyper-V™ technology for their NetWeaver 7.0 based applications and later releases of SAP software. It also is expected that SAP will support Windows Server 2008 R2, which is expected to be released in the second half of 2009 by Microsoft. Windows Server 2008 R2 is expected to introduce support for up to 256 CPU threads and live migration of Hyper-V virtual machines via a Shared Cluster File System.

SAP Multilevel Client/Server Architecture

This section describes how the SAP NetWeaver architecture relates to SQL Server, including an overview of the NetWeaver Application Server (formerly the SAP Web Application Server). This section also describes security features, statement execution, database schema, and information for migrating and upgrading to SQL Server 2008 or SQL Server 2005.

Note that some of the information described in this section is unique to SAP products and does not necessarily follow the practices used by SQL Server 2008 and SQL Server 2005 in other types of applications.

All SAP products employ a multitiered client/server architecture, as shown in the following diagram.

![Figure 1: Three Tiers of SAP systems](image)

The SAP multitiered client/server architecture is composed of three levels:

- **Presentation Tier**. This tier supports SAP Graphic User Interfaces (GUIs) such as SAP GUI, SAP WebGUI, and other products that connect to the SAP NetWeaver Application Server using one of the supported interfaces. The Presentation Tier also includes applications to access SAP using Web Services. For example, applications including smart clients and Microsoft Office applications can integrate SAP data,
such as when the Microsoft Excel® spreadsheet software is used with Web Services. Applications that use the SAP RFC interface are also part of the Presentation Tier. Especially in the Microsoft world, connecting to the Application Tier via RFC became common with the SAP .NET connector, which offers a bandwidth of .NET classes and methods that are mapped to SAP BAPIs that are accessible via RFC.

- **Application Tier.** This tier can contain multiple SAP NetWeaver Application Server instances. However, it needs to contain at least one application instance. If multiple instances are used in one system, each application instance is typically run on separate server hardware or virtual machine. The Application Tier and Database Tier can run on the same server hardware on small scale systems and in some very large hardware configurations. We talk about a two-tier configuration then. The complete processing of the business transactions and workflow is handled on the application side. There is no business logic that is pushed down for execution into the database layer. The Database Tier is used for data storage only. Technically, an SAP application instance is a collection of processes called ‘work processes’ in the SAP terminology. Based on a configuration profile for each individual instance, these ‘work processes’ fulfill different tasks and requests. In order to share user contexts and user data, the work processes of one instance share larger areas of memory. The business logic itself was originally programmed in a 4GL language called ABAP; it has been supplemented by the possibility to code business logic in Java as well.

- **Database Tier.** This Tier supports the SAP database, including the SAP Business Suite or R/3 and other SAP applications that are hosted on SQL Server. The Database tier typically runs one database schema for each SAP product using separate server hardware. The database servers can be connected to a storage area network (SAN), network attached storage (NAS), or locally-attached storage.

### SAP NetWeaver Application Server

The SAP NetWeaver Application Server is the base architecture of most of SAP’s products. The SAP NetWeaver Application Server has the basic task of running business logic developed in ABAP or in Java. For this purpose, the NetWeaver Application Server includes an ABAP and Java virtual machine (VM), many infrastructure services, and proprietary and standardized interfaces like WebServices/HTTP/SOAP.

### SAP Servers

On the left, the SAP Application Servers sample shows examples of instances for one SAP ERP system that uses multiple hardware servers. Some of the servers are running more than one SAP NetWeaver Application Server instance.

Figure 2: List of SAP instances of one SAP system using SAP GUI
An SAP system can contain dozens of NetWeaver Application Server instances running on multiple servers. All instances of one system work against one database.

What Is an SAP System?

In the following chapters, the term SAP system is used often. What does this mean? Usually this term implies that the production SAP ERP/CRM/EP/SCM/… system a company is running. This SAP system typically runs on multiple physical servers; one server is the dedicated database server, and a few other servers run what SAP calls application instances (ABAP and/or Java). On at least one server, a Central Instance or Central Services Instance (ABAP and/or Java) is running. This usually frames a SAP system. In small deployments, all the components mentioned can be installed on one physical server. Larger systems and the majority of customer installations usually try to isolate these components on separate physical servers.

SAP NetWeaver Application Server Architecture

SAP NetWeaver Application Server is the main building block for deploying highly scalable SAP Web applications and Web services.

NetWeaver 7.0 is the latest version of the SAP NetWeaver Application Server architecture, it contains both the well-known ABAP VM and a complete SAP Java stack. An overview of the SAP NetWeaver Application Server is shown in the following diagram. Some of the functions shown in boxes in the diagram are embedded functions executed by the ABAP or Java processes. These functions are in the diagram because they are important for understanding the behavior of the application server.
The SAP NetWeaver Application Server User tier (also called the Presentation Tier) connects to the SAP NetWeaver Application Server layer through HTTP/SOAP, Web Services, RFC, or the proprietary Diag interface used by the Standard SAP GUI. The SAP NetWeaver Application Server layer contains the following logical components:

- **Virtual machines (ABAP and Java).** The ABAP and Java virtual machines (VMs) are the heart of SAP NetWeaver Application Server, in two stacks. Nearly all core business report logic runs through the ABAP VM. The Java VM is mostly used to process logic, especially by newer generations of SAP products like SAP Enterprise Portals and SAP XI/PI and other components of products or smaller Industry specific products such as SAP xMII. There are SAP products that use both stacks (for example, SAP XI/PI) and products that use one of the stacks only (for example, SAP Enterprise Portal using SAP Java only). The stacks don’t share common database transactions. This means that if functionality provided by the Java stack is called from the SAP ABAP logic, the database transaction is committed before the Java logic starts to execute.

- **ABAP and Java SCS.** The SCS instances for the two parts of the SAP NetWeaver Application stack combine two major functions.
  One function is to handle the association of a user request to a certain instance. This is done by the message server process. Also, all communications between different instances within one SAP system passes through the message server. The other vital function is to provide the logical locking SAP uses on the application side. The SAP Enqueue process performs this function. The Enqueue processes keep a memory area where objects locked by business logic is stored. These processes handle SAP logical locking management. There must be at least one Enqueue process for each SAP system. SAP ABAP and Java stacks use separate Enqueue processes. In some cases, when there is a gap between a user confirmation and the end of an asynchronous update, an SAP object such as the Material or Customer needs to be locked before the database is accessed. In these cases, the Enqueue processes handle the locking and unlocking of requests on the SAP NetWeaver Application Server layer system-wide. This avoids extensive locking on the database level, which could harm concurrency severely. The Enqueue processes keep the database from being flooded with locks and blocks objects not yet released. It is worthwhile to note that the level of locking of the Enqueue is on an object basis. Such an object locked could be materialized in many database tables and can be reflected by various rows in these tables. Hence locking the object logically on the application layer offloads the database and contributes to enhanced scalability. The two different SCS instances can be installed alone on a server. Because the CPU consumption is expected to be pretty low, you can install multiple SCS instances on one server or have other ‘normal’ SAP instances running on the same server.

- **Dispatcher (Queue Manager).** In both Stacks (ABAP and Java) there are processes that fulfill a dispatching role. These processes receive requests from the outside via the various interfaces. The Dispatchers queue and distribute requests to other SAP processes fulfilling specific tasks. The Dispatcher process on the ABAP and Java side also maintain communication with the Presentation tier layer.
  The first time a request from the Presentation tier is made to SAP NetWeaver Application Server, it is assigned to the Dispatcher process of a particular instance by the Message Server Process that is part of the ASCS or SCS part of the two stacks. The Dispatcher processes locate a free ABAP or Java process in the ABAP or
Java instance with the requested functionality and assign the request. All of the different processes in one instance communicate back to the Dispatcher processes. One dispatcher process exists for each instance of an ABAP or Java stack.

- **Central Management Services.** Central Management Services include Batch Scheduling, Memory Services, and Development Workbenches.
- **Data persistence layer.** Each stack (VM) has a data persistence layer that operates on different schemas within the SAP database. Very few tables are shared between the ABAP and Java stacks.

Be aware that the SAP ABAP stack and the SAP Java stack of the SAP NetWeaver Application Server can be installed separately. During installation either both or only one of the stacks can be installed.

For more information, see “SAP NetWeaver: Providing the Foundation to Enable and Manage Change” at: http://www.sap.com/solutions/netweaver/index.epx

### SAP NetWeaver Application Server with Windows

In the following section, we will discuss the SAP ABAP and Java stack of the SAP NetWeaver Application Server. In contrast to products developed on a Windows platform such as SQL Server, SAP NetWeaver Application Server is a multiprocess application, not a multithreaded application.

The SAP processes that are configurable within one SAP NetWeaver Application Server instance on the ABAP stack include:

- **Dialog process.** This process handles user interaction initiated from the Presentation tier. Normally, there are multiple processes offering dialog services in each instance.
- **Update process.** SAP processes execute asynchronous changes on the database. There has to be at least one update process for each system. There can be instances without an update process. In most cases, multiple Update processes are configured.
- **Batch process.** This process handles long running, non-interactive jobs in the background, for example, Payroll Calculation. SAP tasks can be scheduled to run at a certain point in time or when a specified event occurs.
- **Spool process.** This process enables print services for an SAP system. The process sends a print request to the Windows spool manager.
- **Message server process.** This process enables communication between the different NetWeaver Application Server instances within one SAP system. The message server runs on the SAP CI. It is a single point of failure.
- **Gateway process.** This process is responsible for external communication between NetWeaver Application Servers.
**Enqueue process**: This process is (which can be multiple processes) responsible for the logical locking on the application layer. In older systems, there could be only one SAP instance per SAP system running this type of process (usually Central Instance). Today this process type is separated out of the traditional SAP instance configuration.

Figure 4: Process list of SAP processes of one SAP instance using SAP GUI

Above, the sample lists different processes in a normal SAP instance of an SAP system running multiple instances. The sample does not show the dispatcher, message server, or gateway process.

One specific SAP NetWeaver Application Server instance represents a collection of processes. The NetWeaver Application Server Layer can be distributed over several servers to perform business logic. In the commodity server space, it was common to run one SAP application instance per server. This changed lately with the increasing number of CPU core on industry standard hardware. With 16 CPU cores and linear addressable memory of 64 GB or more on 64-bit platforms, it is becoming more and more common to consolidate more than one SAP application instance on one server hardware as it always has been the case on larger hardware. In most cases, the reason for having multiple different instances running on a single hardware server is to gain flexibility in the configuration of different instances or to dedicate some instances for special purposes. Different SAP application instances of SAP NetWeaver Application Server can be configured differently based on the user or job assignments to those instances.

SAP NetWeaver Application Server instances can be configured to enable anything from only one or two types of processes to nearly all types of processes. The specific configuration depends on the size of the SAP system and on the available hardware.

The type of processes within an application instance can be changed while the application instance is up and running. This flexibility is helpful in systems that experience dramatically different workload profiles during a 24-hour period. Customers can define different operation modes to run at different times of the day with a different mix of different SAP processes within one instance.
User and User Request Assignment to SAP ABAP Processes

SAP has various methods of assigning users in the login phase for a specific application server instance. One method is based on login requests. The login request first contacts the SAP Message Server process of the specific application stack. This process checks to see whether there is a special assignment for the user who wants to log in. If this is the case, the user is assigned to a specific SAP application instance based on this assignment and eventually on workload on different application instances, if the assignment defines a group of application instances. If no special assignment is defined, the message server assigns the user to an application instance based on workload consideration. The login request is then transferred to the dispatcher process of that application instance. The dispatcher then assigns the login request to a free dialog process within its application instance. After a successful login, the user remains within the boundary of this SAP application instance. A transfer to another instance is not possible.

After the successful login, the user starts to work by calling an SAP transaction like ‘Creating a customer order’. This request goes directly from the user interface of the user to the dispatcher of the SAP application instance. The dispatcher checks for a free dialog process in the ABAP case, and if one is found, the dispatcher assigns the request. As soon as the request is completed, the user context is rolled out of the dialog process again, so that the process can be used to serve requests from other users. This means that there is no fixed assignment of an SAP user to a process within an SAP instance, but only a fixed assignment to an SAP instance. The dispatcher within the SAP instance balances incoming requests of logged-in users and active requests on the process resources available. If no processes of the type required are available, a queue in the dispatcher builds up.

Logon groups provide a more transparent method for assigning users to SAP application instances. In this case, a user is assigned to a logon group instead of having an assignment directly to a SAP application instance. The logon group itself is then assigned to one or more SAP application instances. The assignment of a logon group to an SAP application instance can be changed on the fly. This enables additional flexibility balancing unexpected workloads and provides an abstraction layer which can be helpful for any type of system maintenance. However, one user context cannot be moved to another instance when the assignment of a logon group is changed. Logon groups, together with other abstraction features, can be used to perform online patching and maintenance on SAP application servers or instances without interrupting business.

Memory Management of the SAP ABAP Engine

As described earlier, a user is kept within an instance, and the context of the user can be ‘rolled in’ and ‘rolled out’ of a work process within an instance. For this purpose, all SAP processes share an area of memory called Roll Memory. Roll Memory is memory that is used to keep user contexts stored. A user context describes authorizations and permissions of the user, and it eventually points to data the user still has allocated.

Another important memory area of the SAP application server (ABAP as well as Java) is the table buffers. Table buffers are buffers that store infrequently modified data for
access. All the customizing information for the particular SAP Business modules is stored in database tables. In order not to retrieve this kind of information tens of thousands of times from the database, this kind of data, which usually is not modified at all, gets stored in the SAP table buffers on the application instances. This greatly reduces the number of database statements.

The third structure of SAP memory management that is important to this discussion is Extended Memory (EM), which applies to the SAP ABAP application instance only. Extended memory is memory that can be accessed by each of the processes within one application instance. Like all VMs, the ABAP VM needs to have memory that is dispatchable to user contexts that load data from the database. EM is used to store data that a user loads from database during performing a user’s request.

Because many user contexts can be executed in parallel by the different SAP processes of one instance and one user can read substantial amounts of data from the database, the amount of EM needed can add up to many GB. For 32-bit, which was the first Windows platform SAP got ported 15 years ago, the 2-GB address space of 32-bit would have been too restrictive. Therefore, targeting a solution to realize the SAP EM via shared memory between the processes was not an option. Instead SAP went the way of realizing their EM over a Memory Mapped File. This Memory Mapped File can be accessed by every one of the SAP processes of an instance. An SAP process can map segments of the Memory Mapped File into the particular SAP process. Segments mapped into one process cannot be accessed by another process of SAP. The advantage of the Memory Mapped File is that its size is limited by the real memory plus the size of the Windows page file and not by the virtual address space of the particular platform.

A fourth very big chunk of memory allocated by an SAP ABAP instance is the Program Buffer, which caches precompiled ABAP reports instead of steadily rereading those from the database. Usually this part of buffer has a size of 500 MB or higher. This buffer is realized as shared memory that can be accessed by each of the SAP processes within an instance.

Besides the four different memory areas described, an SAP ABAP application instance creates over a dozen smaller shared memory areas. These areas are shared among the processes of the instance. None of these memory areas are shared between among SAP instances.

In the 32-bit world, these large buffers could cause issues with address space fragmentation. For example, in some cases SAP application instances did not restart after Windows Server Service Packs. Usually the reason was that the large Extended Memory Window or the SAP ABAP program buffer couldn’t find large enough window in the limited 32Bit virtual address space. A Windows component basing DLLs differently in a service pack or a hotfix could cause problems in the SAP application instances allocating their buffers. The problem has been completely eliminated, however, because most customers have moved or are in the process of moving to 64-bit.

**Windows Page File**

In former days, SAP OSS Note and installation documentation recommended configuring the Windows page file in the size of 2-3 times the real memory. If the physical memory of a server was 2 GB or 4 GB overall, this could make sense. However, now that application servers have 32 or more GB, it no longer makes sense.
In our experience, in most deployments, a Windows page file that is the size of the real memory is sufficient. More would simply be wasted space.

**SAP Java Application Server**

A lot of the SAP ABAP mechanisms are similarly implemented on the SAP Java side. Usual Java Virtual Machine (JVM) are already multithreaded processes executing requests. A principle that did not get carried over is the separation of different types of processes for different types of work. A default installation contains only a Java dispatcher, one Java process that is a Java VM only, the Software Development Manager (SDM), and the Java SCS. The Java SCS runs the Java message server and the Enqueue server. An additional JVM can be configured to start if a single JVM cannot keep up with the load. In contrast to the ABAP side, where a user context exists in shared memory and gets used by an SAP ABAP process during the processing, the user context in the SAP Java stack exists in the JVM and is processed by one of the threads of the JVM. In order to coordinate different JVMs in one instance or in multiple instances, the Java Enqueue process is required. The disadvantage of the fact that the user contexts exist in the JVM is that a crash of a JVM will delete the user contexts of multiple users, whereas on the ABAP side, only one user context is destroyed. In order to run larger systems like SAP Enterprise Portal using multiple SAP Java instances on different servers, an SAP Web Dispatcher is required. The SAP Web Dispatcher communicates with the Java message server of a system in order to get a list of the active Java instances and their configuration. The SAP Web Dispatcher then optimizes the user assignment to the different SAP Java instances.

**User and User Request Assignment to SAP Java Processes**

As with the ABAP side, the message server is the first point of contact for a new user logging in against the SAP Java stack. If there are multiple SAP Java instances within a system, the SAP Web Dispatcher needs to be added in front of the message server process. As in the ABAP part the user’s login request finally gets assigned to a Java dispatcher process within on SAP Java instance. This dispatcher process now assigns the user’s request to the SAP JVM (or one of the Java VMs). If further requests come from users, the requests are routed to the specific JVM in the SAP Java instance that keeps the context of that user already.

**Java Versions Used**

Including SAP NetWeaver 7.0, SAP requires the use of the J2SE 1.4.2 version of SUN for SAP on Windows. Because this SUN J2EE release is rather old and never got officially ported to JVM for the masses, SAP OSS Note #941595 explains how to get to the x64 J2SE 1.4.2 SDK for x64 on Windows.

With SAP Basis 7.10, SAP will rely on SAP’s own JVM, which is on Java5 standard. However as of June 2009, SAP’s own Java JVM cannot be used for NetWeaver 7.0 because there are incompatibilities which can cause error situations.
For many years, SAP used the DataDirect JDBC driver to communicate with the JDBC persistence layer. SAP now uses Microsoft's JDBC driver. SQL Server 2008 and SQL Server 2005 require the Microsoft JDBC driver 1.2 or higher. For information about how to obtain and install or upgrade to the Microsoft JDBC, see SAP OSS Note #639702 and OSS Note #1109274. The Microsoft JDBC driver also can be used for SAP Java installations of SAP Basis release 6.40.

SAP Implementations on SQL Server

SAP Connections to SQL Server (ABAP Side)

Each SAP ABAP process establishes a connection to SQL Server. There is no sharing of connections between the different SAP processes of an instance. SAP does not use connection pooling. Every one of the ABAP processes establishes connections at startup and keeps them until the process is shut down or restarted. You can observe hundreds or even thousands of connections to SQL Server when a complete SAP system is up and running. In former releases of SQL Server, the number of connections SAP processes needed to open against SQL Server could become exceedingly inflated, because in the way SAP was interacting with SQL Server (using client side cursors). Using Client side cursors, SQL Server did not allow a connection to execute a second statement while a previous statement was still in flight. This shortcoming was resolved in SQL Server 2005. With the introduction of Multiple Active Result Sets (MARS), multiple open client-side cursors can use the same connection. In contrast to earlier SQL Server releases, the number of connections of an SAP process to SQL Server 2008 and SQL Server 2005 now is limited to two connections for each SAP process as shown in the following diagram.

Figure 5: SAP process connecting to SQL Server

The first connection SAP establishes is used for modification of data and in rare cases for reading data if a read-committed isolation level is needed. On the second connection, SAP reads most of the data on an uncommitted read isolation level (also known as a 'dirty read'). SAP introduced this principle in 1993/94 for all databases but Oracle. There is no disadvantage doing so because the SAP NetWeaver Application Server is designed to deal with eventual Phantom Records or other anomalies that can be introduced by reading uncommitted data. So imagine an SAP system with 10 application server instances on 10 different small servers. Every application server
instance is configured with an average of 30 SAP processes. In The SAP application layer of such a system will open 10 X 30 X 2 = 600 connections to SQL Server. Both the ABAP and the Java sides of the SAP NetWeaver Application Server have reconnect mechanisms that reestablish lost connections to SQL Server. Depending on the circumstances, all processes of the SAP application instance might be switched into reconnect status and forced to reconnect when the next request is assigned to the SAP process. For more information about the Reconnect Architecture and Design, see SAP OSS Note #98051.

Security Context of SAP ABAP Connection

The SAP NetWeaver ABAP side benefits from SQL Server Integrated Security. The security context of SAP transactions is established during the installation of SAP NetWeaver Application Server:

- SAP creates two Windows users named SAPService<SID> (for example, SAPServicePRD) and <sid>adm, (for example, prdadm). <SID> and <sid> represent the three-character SAP System ID (<SID>), for example, PRD. The SAP SID is a unique system identifier across a customer's particular product landscape namespace.

- Windows users are created as logins in SQL Server. The two Windows users log into SQL Server using Integrated Security. For ABAP only installations, SAP recommends installing SQL Server Integrated Security to accept Windows authenticated connections only. For SAP databases, these two logins are assigned to the SQL Server sysadmin role.

- A SQL Server login with lowercase name of the <SID> (for example, prd) is created for each user that owns a schema in the SAP database. Each user is assigned to the SQL Server serveradmin role. Each user owning a schema in the SAP database is assigned to the database db_owner role.

- After the SAP process, SAPService<SID> establishes a connection using Integrated Security. The process acts as a user owning a schema in the SAP database by executing: setuser '<sid>'

- In the more recent SAP releases, an integrated security login named SAP_<SID>_LocalAdmin is created as well. This Windows group usually contains the Windows users <sid>adm and SAPService<SID>.

For troubleshooting, note the following:

- When accessing an SAP database with SQL Server query tools, the setuser '<sid>' command must be executed before data can be read. This command will basically impersonate the user context to the context of the schema owner of the SAP schema.

- Because the SAP customer can install more than one schema in one SAP database, ensure that the schema is correct. Most schemas have a set of objects that are named in the same manner in both schemas.
The preceding SAP security considerations involve the last three releases of the SAP NetWeaver Application Server, which are supported by SQL Server 2008 and/or SQL Server 2005. In older SAP releases like products running on the SAP 4.6C Basis, an SAP database could contain only one schema owned by the database role db_owner.

SAP Connections to SQL Server (Java Side)

Using the SAP Java stack against SQL Server requires SQL Server to be installed in mixed security mode. Because only the latest SQL Server JDBC driver (1.2) to supports Integrated Security, this functionality is not yet used by SAP in the Java stack. Hence SAP uses SQL Server Authentication to log in to SQL Server. The SAP Java stack creates a SQL Server login. During the installation of the SAP Java instance, you define the password for this user. The password itself then is stored in the SAP Java Secure Store as shown in the following figure.

![Figure 6: SAP Secure Store – changing password to connect to SQL Server](image)

The user created has the SQL Server login named SAP<SID>DB. This user is assigned to the SQL Server’s public server role and is assigned to the SAP database in the role of db_owner with the user SAP<SID>DB.

This user owns a schema of the same name in the SAP database. This schema contains all the SAP tables required and used by the SAP Java stack.

SAP Statement Execution

The NetWeaver Application Server executes a combination of parameterized statements and static stored procedures against SQL Server. When the application server uses stored procedures, a single Transact-SQL Statement is wrapped in a stored procedure. Reasons for this way of statement execution are historically influenced. The intention of using parameterized statements or stored procedures is to cache and reuse the execution plans. This keeps the number of statement compilations and recompilations extremely low (usually in the low single digits per second).

In using stored procedures, SAP dynamically creates them from changes to the ABAP code. After a change of ABAP source code, new stored procedures will be created for all
SQL statements within the changed ABAP function. The usage of stored procedures was eliminated entirely with all products relying on NetWeaver 7.0. In these newer products, SAP relies entirely on parameterized SQL statements.

In addition:

- SAP uses a minimum number of ad hoc queries, fewer than one per second.
- There are relatively few database locks, because 99.9 percent of all reads are uncommitted. The number of database locks typically ranges from a few hundred to a few thousand. Especially few shared locks are seen.
- In SQL Server 2005 and SQL Server 2008, all Transact-SQL statements and stored procedure executions are prepared on the client side.
- In SQL Server 2005, SAP only supports the OLE DB programming interface integrated in SQL Native Access Client (SNAC). SNAC needs to be installed on each application server to allow the SAP NetWeaver Application Server to connect to SQL Server 2005. SNAC contains the Client APIs of SQL Server 2005. SNAC is deployed by SQL Server only and is independent of Microsoft Data Access Components (MDAC) that ship with Windows and Windows service packs. For SAP products running on SAP basis release 4.6C against SQL Server 2005, MDAC still is used independent of the availability of SNAC. This has implications in the usage of SQL Server database mirroring. Because SNAC includes vital functionality for the automatic failover to the mirror node, the functionality of automatic failover in database mirroring is not available for SAP products running on 4.6C basis against SQL Server 2005.
- In SQL Server 2008, a complete new SAP porting layer is used to connect to SQL Server 2008 exclusively. This newly designed and programmed porting layer uses the ODBC of SNAC to connect to SQL Server 2008. Both implementations of SAP porting layers are included in the same DLL (dbmsslib.dll) and hence there is no work for customers in exchanging DLLs. Please note that with every new main release of SQL Server we implement a new SNAC version. SAP uses the SNAC version of SQL Server 2005 to connect via OLE DB to SQL Server 2005. SAP then uses the SQL Server 2008 version of SNAC to connect via ODBC to SQL Server 2008. See the following figure for the driver used with SQL Server 2008.
SAP Database Schema

An SAP business application’s database schema can include up to 75,000 tables, depending on the SAP product. The following database structures are used to facilitate the processing of data in these tables.

- **Clustered and nonclustered indexes.** Nearly all tables use the primary key constraint of SQL Server. This results in a clustered index over the fields of the primary key. No other constraints are used. Nearly 20 to 25 percent of the tables have one or more nonclustered indexes. The column order of deployed, nonclustered indexes can be changed. In addition, a clustered index can be made nonclustered and conversely, a nonclustered index can become a clustered index. The major exception of this rule is indexing on some larger tables in SAP BW where no clustered index is used and also no primary key constraint is used. Although a primary key index can be changed between clustered to nonclustered, the order and number of columns included in the primary key MUST remain the same; internal SAP processing is likely dependent upon it.

- **Additional indexes.** In standard deployment, many tables have multiple nonclustered indexes defined on them. It is very common for customers to create additional indexes to suit their customization of SAP business logic or customer-coded ABAP business logic. Indexed views or indexes on computed columns cannot be created within SAP because these objects can’t be represented in the SAP Data Dictionaries. Most of the nonclustered indexes at least cover two columns. The exception again is SAP BW where quite a lot of indexes on larger tables just cover one integer column.

- **Index column order.** The order of index columns is nonstandard, with the most unselective columns coming first. The order of index columns follows the order of columns in the table that is defined in the SAP Data Dictionary. Except for the primary key index, the order of the columns is free and can be changed (although this is usually unnecessary). The order of the columns in the primary key index cannot be changed, because an ABAP construct might rely on the defined order of the primary key.

- **Index creation.** Creating indexes or changing existing indexes should be performed through the SAP Table Maintenance Transaction so that SAP has the ability to export the index. This also ensures that the index modifications will be maintained across SAP version upgrades.

- **Table Partitioning.** SAP only uses SQL Server table partitioning in their SAP BW product. The business logic of SAP BW manages creating new partitions or dropping partitions. There is no need to interfere in any way from a DBA side in this logic; SAP takes care of it for you.

- **Triggers.** Database triggers are occasionally used by some SAP applications and for some specific activities during release upgrades.

- **Default values/constraints.** Most columns of SAP tables have default values assigned to them, because most columns are defined as not nullable columns. In former releases of SQL Server, this had some impact on the space consumption.
SQL Server 2008 mitigates the impact on space consumption by using a new row format with SQL Server 2008 Row Compression.

**Collation and Code Page**

Older SAP products used code page (cp) 850_BIN of SQL Server. This changed with SAP’s introduction of Unicode. Today all SAP products running against SQL Server 2008 and SQL Server 2005 leverage SQL Server code page 850_BIN2 as the server code page. The only exceptions to this are with the SAP Mobile Sales client, SAP Business One, and other SAP products that don’t leverage NetWeaver as their platform. This relates not only to the SAP database, but also to the complete SQL Server instance the SAP database is running on. There is a significant difference between SQL Server cp850_BIN and cp850_BIN2 that affects Unicode sort orders. Therefore codepage cp850_BIN is not suitable to run either SAP Java schemas or SAP Unicode schemas. Installing SQL Server 2008 or SQL Server 2005 cp850_BIN2 must be chosen as code page for the complete SQL Server 2008 or SQL Server 2005 instance as shown in the screenshots later.

While upgrading a SAP database that was installed under older releases of SAP applications from SQL Server 2000 to SQL Server 2008 or SQL Server 2005, such a database needs to be converted to cp850_BIN2. This needs to happen while still running on SQL Server 2000 and prior to the upgrade to SQL Server 2005/2008. For more information, including the procedure and an executable that performs, this conversion, SAP OSS Note 600027. After you ensure that the SAP database is on cp850_BIN2, you can start the upgrade to SQL Server 2008 or SQL Server 2005.

**SQL Server 2005 Setup Screen**

The following figure shows the code page selection screen during a SQL Server 2005 installation. For SAP with SQL Server 2005, select:

**Binary Order based on code point comparison, for use with 850 (Multilingual)**

**DO NOT** choose the former selection: **Binary sort order for cp850**
Figure 8: Collation selection for SQL Server 2005
The selection of the database collation in SQL Server 2008 is a bit less obvious than in the setup of former SQL Server releases. If you do not use the VBScript program SAP delivers to install SQL Server because you prefer to install manually, it is easy to miss the tab to change SQL Server collation. It is basically in the same screen as the definition of the user accounts SQL Server’s services should run in. But it is the second tab, as shown here.

Figure 9: Collation tab in SQL Server 2008
The next screenshot shows the selection of the collation definition in SQL Server 2008. Be aware that you needed to scroll up in the installation of SQL Server 2005, whereas you need to scroll down in the SQL Server 2008 installation.

Figure 10: Select correct collation for SAP NetWeaver based products in SQL Server 2008
Be aware that the SAP installation programs prevent installation against a SQL Server Instance setup with a different collation.

Data Types Used by SAP

SAP uses simple data types, including:

- Variable length character types only.
- Normal integers and a few small integers.
- Float and decimal, especially for the SAP BW fact tables.

In addition, in SQL Server 2008 and SQL Server 2005, the varbinary(max) data type is used instead of the image data type for BLOB and CLOB fields. The data type is changed from image to varbinary(max) as a metadata operation only while performing the post-upgrade steps after a successful upgrade from SQL Server 2000 to SQL Server 2008 or SQL Server 2005 upgrade. No data movement takes place when the data type is changed. For more information, see SAP OSS Note #799058.

Note that SAP does not use user-defined data types and SQL variants.

Upgrading from SQL Server 2005 or SQL Server 2000 to SQL Server 2008

With products based on SAP NetWeaver 7.0 or later, SAP supports only the 64-bit versions of SQL Server 2008. This restriction was imposed due to the work of getting a completely new database porting layer done and the significant movement of customers adapting NetWeaver 7.0 based products. Also, with most SAP customers, 32-bit had already outlived itself and was creating significantly more problems than 64-bit.

SAP supports upgrades to SQL Server 2008 from both SQL Server 2005 and SQL Server 2000.

In both cases, you must apply the SAP Basis Support Package #14 and BW Support Package #16 or later to the system before you move to SQL Server 2008. You must also use at least Patch Level 182 of the SAP NetWeaver kernel 7.00 or later. SQL Server 2008 RTM CU2 (which shipped in the middle of November 2008) or later is recommended. For more information, see SAP OSS Note #1152240.

When upgrading from SQL Server 2000, a minimum base of SP4 is required. In this case, you must already be running on code page cp850_BIN2. For more information on using this code page on SQL Server 2000, see SAP OSS Note #600027. When upgrading from SQL Server 2005 to SQL Server 2008, a minimum base of SQL Server 2005 with SP2 is required.

After an in-place upgrade of SQL Server 2008 has completed, a special tool called SAPInst needs to be run in order to adapt to SQL Server 2008. For more information, see SAP OSS Note #683447.

There are the two alternative methods of upgrading to SQL Server 2008:

- Uninstall the older version of SQL Server and then install SQL Server 2008. Then re-attach or restore the SQL Server 2005 or SQL Server 2000 SAP databases to SQL Server 2008.
Install SQL Server 2008 in the same location without uninstalling SQL Server 2005 or SQL Server 2005 (in-place upgrade).

In both cases the SAPInst tool needs to be run in order to adapt the database to SQL Server 2008. You should expect the run time for upgrades from SQL Server 2000 to take longer when the SQL Server upgrade or attachment of the SQL Server 2000 database is also performed. But also the run time of the SAPInst tool will be longer when you are upgrading from SQL Server 2000, because more changes need to be applied.

High-Availability Upgrade to SQL Server 2008

SQL Server 2008 provides new functionality that can help retain high availability during upgrades. This section demonstrates how to use in-place upgrade to move to SQL Server 2008 with minimal downtime. The procedure uses log shipping and database mirroring. Information about applying the procedure to Microsoft Cluster Services is also included.

During an upgrade from SQL Server 2000 to SQL Server 2005, an in-place upgrade couldn’t be done with a read-only database attached to the SQL Server 2000 instance. The only solution was to wipe out the log shipping destination and rebuild it from scratch after the SQL Server instance was upgraded to SQL Server 2005. This caused major pain, shipping terabytes of backups to disaster recovery destinations and rebuilding those log shipping destination systems. It also exposed customers for periods of up to several days. In SQL Server 2008, this works differently. To paint the scenario of an upgrade, let’s assume the following configuration:

- In the main datacenter, database mirroring with automatic failover is used to provide high availability. This means we are looking at three SQL Server instances (the principal server, the mirror server, and the failover witness) that will need to be upgraded from SQL Server 2005 to SQL Server 2008.
- An additional disaster recovery site exists, which runs a log shipping destination on a SQL Server 2005 instance.
- Log shipping and database mirroring are active.

In order to minimize downtime, the following steps can be taken while the SAP application is still productive:

1. Install the SNAC10 database client on all SAP application servers. Because the installation does not overwrite SNAC of SQL Server 2005, it doesn’t interfere with SQL Server 2005 client connectivity. The installation doesn’t require a reboot.
2. Suspend log shipping and start an in-place upgrade of the SQL Server 2005 instance that contains the log shipping destination.
3. After step 2 is complete, reactivate SQL Server log shipping. The SQL Server 2005 log records will still be applied to the database in the log shipping destination, because the database still is in SQL Server 2005 metadata format and SQL Server 2008 binaries can handle such cases.
4. Now take the database mirroring witness offline and upgrade the witness instance to SQL Server 2008 (in place).
5. After step 4 is complete, bring the witness back online. A SQL Server 2008 witness can be used in SQL Server 2005 database mirroring.

6. This step should be performed close to the downtime window, because after this step is complete, a failover from the principal to the mirror will result in an upgrade of the system to SQL Server 2008. Upgrade the mirror server of the database mirroring scenario:
   
   a. Take the witness offline and suspend mirroring.
   c. Resume mirroring and bring the witness online again.

As in the case with log shipping, the SQL Server 2008 instance on the mirror will accept log records from SQL Server 2005 and is able to apply those to the mirror database. Also, this mirror database still is in the metadata format of SQL Server 2005.

After these steps are completed, the downtime can begin at any time. In the downtime, complete the following steps:

1. Stop the SAP application. This is the beginning of the application downtime.
2. Stop log shipping.
3. Initiate a manual failover to the mirror server and stop the SQL Server instance of the principal server.
4. Now the mirror database will open up. Because the SQL Server instance already runs on SQL Server 2008, the metadata format of the mirror database is changed to SQL Server 2008. Therefore opening the database can take a few minutes.
5. After the mirror is up and running, start the SAP application instances again. Because the SQL Server 2008 client is installed on those servers already, it should be possible to connect without a problem.

The preceding step marks the end of the application downtime. The following steps complete the procedure:

1. Now start an in-place upgrade of the former principal. After this SQL Server instance is up and running on SQL Server 2008. It will move into the role of the mirror after resuming database mirroring.
2. Now, without further intervention, the changes performed to get the new principal to the metadata format of SQL Server 2008 are applied to the new mirror (former principal) via database mirroring.
3. Start log shipping again from the new principal to the log shipping destination. Again in this case, the changes applied to the principal to get it to the SQL Server 2008 metadata format will be applied via transaction log backups to the log shipping destination.

Following such a process, we had a customer case where three database servers with a 6.5TB SAP ERP database were upgraded with a downtime of less than 20 minutes.

Using Microsoft Cluster Services, you can use a similar process, upgrading the inactive node first and then following on the active node. However such a process is recommended only if your SQL Server 2005 cluster installation is healthy. If there are doubts about the state of the SQL Server 2005 cluster installation, a completely new
installation might be a better idea. This part also is described in the official SAP upgrade documentation. The documentation can be found on: http://service.sap.com/instguides -> Other documentation -> Database Upgrades -> MS SQL Server

Support of Earlier SAP releases

As described in the SAP documentation, there only will be possibilities to install against SQL Server 2008 using SAP NetWeaver SR3 based installation DVDs. It is not, however, a problem to upgrade a NetWeaver SR1 based system to SQL Server 2008. This also will be supported by SAP as long as the requirements for SAP Basis and BW Support Package levels are fulfilled. And when installing a new system which is NetWeaver SR1 or SR2 based, the installation needs to go over the path of installing against SQL Server 2005, applying the necessary Support Packages and SAP kernels and then upgrading to SQL Server 2008. This the way that pre-SR3 releases of NetWeaver 7.00 will be supported by SAP to move onto SQL Server 2008.

SAP Unicode Implementation on SQL Server

SAP introduced the Unicode version of their products more than 5 years ago. In the case of SQL Server, SAP took the only available Unicode implementation by leveraging the `nvarchar` data type. The SQL Server `nvarchar` data type stores the data in UCS2. Every character is stored as 2 bytes in the database files. This implementation is different from the SAP Unicode implementation of some of SQL Server’s competitors. Those products mostly offer UTF16 (maxDB) or UTF8 (Oracle and DB2). The way UTF16 and UCS2 store data on disk is not too different, except for UTF16 handling surrogates. The nature of the UTF8 Unicode implementation on the database side is that 2 bytes are not always used to store data; anything between 1 and 4 bytes might be used. Therefore different storage requirements might be necessary for the different Unicode storage algorithms that are implemented inside the different databases.

Initially SAP released measures of some primitive installations stating that UTF16/UCS2 implementations like SQL Server would require 60% more space compared to competitors using UTF8 (SAP OSS Note #790099). However, these were blank small installation tests on pure single-byte collation, and they lacked real customer data. Revised numbers by SAP based on customer experience are stating an additional growth after a migration to Unicode of 30-40% compared to using Unicode on SQL Server (SAP OSS Note #1139642).

Our experiences of quite a few Unicode migrations, including platform migrations that went from non-Unicode to Unicode, look a bit different:

- First and foremost, any negative impact of the UCS2 Unicode implementation compared to UTF8 is greatly overestimated.
- The pure database sizes after Unicode migrations usually ended up between less than 10% smaller volumes than before and up to more than 15% more than the origin non-Unicode database. It all depends on how much double-byte data is in the system already and how fragmented the data is. Take the Microsoft ERP
There was a small set of Kanji in that database. There were 2.5 years of data in that database. Some of the tables were defragmented pretty well, some others not at all. Microsoft ended up having 4 percent less data volume to start with after the Unicode migration.

- After the migration, the growth rate will be higher, because the pages that are tightly filled with data get data inserted and hence need to be split. Normalization of growth usually occurs over the course of the first 4-6 weeks.

- In a lot of cases, the content of the original non-Unicode database is often fragmented, whereas after the migration to Unicode, larger portions of the data and indexes remain in the highly packed space, because older data is no longer modified. Hence the question is not so much how large the database ends up after the migration but what the growth rate is when the non-Unicode state is compared to the Unicode state of the database.

- Growth is to be expected between non-Unicode and Unicode. For SQL Server, our experiences in SAP ERP are somewhere in the 20-35 percent space. It certainly depends on several different workload parameters again and the double-byte usage. SAP BW shows as good as no differences, given that the majority of the data is made up by different numeric data types. Character data is a minority in SAP BW.

- SQL Server 2008 row level compression, which becomes the default for SAP with SQL Server 2008, saves between 15 percent and 20 percent. This is a net gain (not including gains provided by data reorganization) of space in SAP ERP Unicode systems.

- If SQL Server 2008 Page Dictionary compression is used on selected tables, you might also be able to achieve tremendous savings, which can compensate or even over-compensate the higher growth SAP Unicode on SQL Server may cause.

- SQL Server Development has already taken steps to change the way how UCS2 data is stored. The current plan is that in the new way of storing UCS2 data, it will consume roughly the same amount of space as if it were UTF8 encoded. The new storage format is expected to be completely transparent, so that no changes are needed from the SAP side in order to support it. It is expected that the new functionality will be released in first half of 2010 in a SQL Server release Microsoft announced under the name SQL Server 2008 R2. This new functionality will especially benefit customers with predominant single-byte collation usage on SAP Unicode implementations.

**Migration Considerations for SAP Products**

Migrating SAP products from a non-SQL Server platform to a SQL Server 2008 or SQL Server 2005 platform is generally supported by SAP under these conditions:

- SAP only supports the SAP Migration toolkit as the tool for migrating the SAP database schema and data from a non-SQL Server database to SQL Server. No other tools are supported.

- The migration is a complete data unload on the source system and a complete data load on the destination system.
- Only objects including tables, views, and indexes created by using SAP object maintenance transactions are migrated. No objects created purely on the database are migrated, such as an index that is created on the database outside of SAP table maintenance. SAP migration tools will identify objects to be migrated in the SAP Data Dictionary, not by scanning through the metadata tables on the particular database system. Hence only objects and data in those objects that are represented in the SAP Data Dictionary will be exported.

- With some tuning in the migration processing, you may be able to achieve a throughput of 200GB/h or more during the data export and import phase.

- Typically, up to 10-terabyte data volumes can be migrated over a weekend, according to companies that specialize in such migrations.

- Most of the aspects of a heterogeneous database migration also apply for the migration from a non-Unicode to a Unicode SAP system.

The migration itself needs to be executed under the lead of a person who is certified for SAP platform migrations by SAP. SAP does offer trainings to achieve this grade of certification. Microsoft SAP CoE members are certified for such migrations as well. A variety of third-party partners offer these services. Over the last eight years, several hundred migrations from all platforms have been executed to the SQL Server platform.

The method and the tuning done for the export and import part of a heterogeneous SAP migration to the SQL Server platform is applicable for a Unicode migration and vice versa. The critical time window that needs to be minimized is the time of the export and import. The following steps can increase throughput of such an export/import phase:

- Don’t restrict the SAP Export/Import processes to the database servers. Leverage the existing application server layer to run Export and Import processes. If you offload the Export/Import processes to the application server layer, a few dozen Export/Import processes can run simultaneously.

- Reorganize larger tables on the destination system before exporting. If you are facing a Unicode migration with SQL Server as destination as well, it can make sense to run DBCC INDEXDEFRAG as online data reorganization tool on several of the big tables. Other databases might have some online reorganization tools as well.

- Export unsorted from the destination system. For many years the rule was to export the data sorted from the destination system, especially if you were exporting from some of SQL Server’s competitors. However, this is an extremely time-consuming and resource-consuming effort. By exporting the table content from those competitors in an unsorted manner, the export time can be reduced dramatically. This also applies for SAP BW migrations where unsorted exports are recommended.

- Check SAP OSS Note #1054852 – Recommendations for migrations to SQL Server.

- Split large tables into multiple jobs. However, be careful with the WHERE clauses SAP tools may define for each of the export parts. These WHERE clauses may not line up with existing indexes, meaning that each export job would perform a full table scan. Therefore, adjust the WHERE clause for each export statement manually so that they align with existing indexes on that table.

- Use trace flag 610 and 620 for SQL Server 2005 CU8 or later and trace flag 610 for SQL Server 2008 on the destination SQL Server instance during the data
import. These trace flags reduce the space required in the transaction log of SQL Server dramatically. When these trace flags were not set, we saw transaction logs that were leveraged up to a volume of several hundred GB. For more information, see SAP OSS Note #1241751. If you use these trace flags, expect a higher volume of writes against the data files. Essentially you are trading less write activity on the transaction log for more write activity against the data files. For the rest of the life cycle of an SAP system, these trace flags might not be beneficial. Hence those trace flags can be disabled again after the migration.

- Even with these trace flags enabled, make sure that the SQL Server transaction log is large enough and doesn’t need to be extended during the migration.

- Ultimately the throughput of the export and import processes is constrained by the limitations of the I/O system. In many migrations, however, throughput of 300 megabytes (MB) per second on the source system has been observed. Similar write throughput has been seen on the destination system, at least during peaks. Also note that the characteristics of write workload change dependent on the usage of trace flags 610/620. Without these trace flags, high spikes of writes, which are caused by the checkpoint writer, can occur. Using the trace flags, there is more synchronous write activity into the database files. Hence one can predict a steady stream of 100 MB per second write activity or even more.

- Importing as Unicode, it might be important to have SQL Server’s `tempdb` on fast storage devices. Dependent on how many concurrent loads are going on, SQL Server might use `tempdb` to sort the import batches (usually 10K rows) in `tempdb` instead of SQL Server Buffer Pool. Another possibility could be changing the batch size used by the SAP import processes (R3Load) as described in the next point. Or optionally to put tempdb onto a RAM disk or SSD disks just for the phase of the data import.

- Change the batch size of R3Load. The batch size determines the number of rows that R3Load will try to send to SQL Server. SQL Server will sort these rows as one package and then inserts them into the database. As default, R3Load will commit such a package. In older releases of R3Load, the batch size could be set with Windows environment parameter `BCP_BATCH_SIZE`. The number of rows after commit was separate specified with the R3load parameter `-c`. The value of `-c` had to be at least equal to the value of `BCP_BATCH_SIZE` or a multiple of this value. That is, if the batch size was changed, a modification of the "R3load -c" parameter was often necessary.

In newer R3Loads (see OSS Note #1156361), the batch size can be also defined as a command-line argument of R3Load. The batch size is specified with the parameter `-b`. As mentioned earlier, the number of rows to be committed must be equal to or a multiple of the batch size. A R3load start with a changed batch size can look like:

```
R3load -i XYZ.cmd -l log -c 75000 -loadprocedure fast BATCHSIZE=5000
```

SAP Microsoft have joined forces to improve heterogeneous platform migrations. This joint effort has improved the R3Load process and led to the development of a new tool called EasyMig which currently is in Beta Testing
Hardware Selections

Hardware Tendencies

Over the last three years, the hardware and hardware platform offerings have changed radically. In 2005, IA64 was the only established 64-bit computing platform available. At that point in time many customers even invested in new 32-bit deployments of SAP and SQL Server. In the last three years, however, x64 has established itself as the standard computing platform from the smallest work station up to 32-socket servers.

As x64 became the standard platform, throughput has increased with 2-socket or 4-socket servers, which are industry-standard hardware. During the last three years, the number of cores per socket/processor has increased from 2 to 6 cores, nearly tripling database throughput under SAP workloads. These tendencies are expected to continue into the future; Intel and AMD have published information on their future designs, which go up to 8 or 12 cores and 16 or more CPU threads per socket/processor. For more information, see the following links:


The change is apparent in the difference between standard database benchmarks executed in 2001 and those executed in the year 2005. The performance achieved by 32-socket/processor servers in 2001 was achieved by 4-socket/processor industry-standard servers in 2005. It is expected that by the end of 2009, 4-socket/processor industry-standard servers will be available that deliver database throughputs similar to what fully equipped 64-socket/processor servers were delivering at the beginning of 2006. Combined with 64-bit address spaces, the current processor technologies and platforms offer tremendous throughput already, and they will continue to provide tremendous growth in the future.

How will this growth in throughput be achieved? In contrast to 10 years ago, when frequency increases provided most of the throughput growth, most current increases are achieved by increasing the number of cores per socket/processor. This is a very attractive model for SQL Server customers who license on a per socket/processor basis. Frequency could increase slightly over time.

We expect another focus of the processor development and hardware development in coming years to be energy-saving measures. Switching off cores and other processor components in order to save energy (primarily on running the server but also to cool the data centers) will be a huge focus over the next few years. For example, Windows Server 2008 R2 is expected to show the first effects of the push to many more cores per socket and energy-saving features. This release is scheduled for the October 2009. Besides its ability to address up to 256 CPU threads (instead of the current limit of 64), Windows Server 2008 R2 is expected to use some features of these new processors in order to reduce energy consumption of servers that are not loaded all the time.

The growth in throughput of industry-standard hardware architectures also opens the question of consolidation of SAP systems on fewer servers. This consequently raises
additional questions around virtualization. We hope to be able to provide some answers, discussion, and methods in Part II of this white paper.

What about the 32-bit platform? From SAP’s perspective, here’s the short answer: It is dead as a platform for SAP. As already described, SAP is not supporting the 32-bit platform for either their newer releases or SQL Server 2008. We recommend that future investments be made in one of the 64-bit platforms rather than 32-bit.

**Microsoft x64 Computing Platform**

The SQL Server 2008 and SQL Server 2005 (64-bit) x64 computing platform is a Microsoft architecture that extends the x86 instruction set to 64 bits. SQL Server 2008 and SQL Server 2005 use the 64-bit editions of Windows Server 2003 Windows Server 2008. Drivers and software specifically compiled for the x64 instruction set are required to take full advantage of 64bit. The 64-bit versions of Windows Server 2003 and the 64-bit editions of Windows Server 2008 support both the AMD Opteron and Intel processors with Extended Memory Technology (EM64T).

Note that the SQL Server 2008 and SQL Server 2005 (64-bit) x64 platform does not run 64-bit IA64 versions of the Windows Server operating system or 64-bit IA64 drivers that are compiled for Itanium. Basically the 64-bit IA64 platform is not compatible with the SQL Server 2008 and SQL Server 2005 (64-bit) x64 platform. Different sets of executables are required.

This platform has meanwhile established itself as commodity 64-bit platform. It already is used successfully for SAP Application tier and for SAP database servers in small, mid-sized, and even higher-end SAP ERP and SAP BW systems. The advantage of this platform is an outstanding price/performance ratio.

From a performance point of view, commodity platform servers are basically doubling pure computing performance nearly every 18 months. Comparing an 11-year-old four-processor Pentium Pro 200 database server performance to the most recent and fastest 4-socket/processor six-core x64 server on Windows, you can see throughput that is nearly 40-50 times higher with Standard SAP benchmarks using the most recent server running on x64 Windows. Despite steady increases of SAP workload in customer scenarios, more and more productive SAP landscapes can be realized on inexpensive commodity-type hardware.

**Microsoft IA64 Computing Platform**

IA64 on Intel Itanium processors was the first 64-bit platform officially supported by Microsoft from the year 2003 on. Windows Server 2003 SP2 and SQL Server 2000 (IA64) were available on this platform from 2003 on. Meanwhile, Windows Server 2008, SQL Server 2008, and SQL Server 2005 are available on this platform as well. Today the largest SQL Server implementations run on this platform.

In the industry-standard hardware space, however, the IA64 platform has not been able to break through, and as a result it plays only a minor role. Hence most implementations of SAP/SQL Server that use IA64 use it only to support the database on hardware offering between 16 and 32 sockets/processors.
SAP Applications on a 64-Bit Platform

The main problem with running SAP application instances using 32-bit computing is that the address space for a single process of the SAP instance is limited to 3 GB. Due to SAP’s extensive buffering for the NetWeaver Application Server, those buffers use a large amount of memory in the address space of a process. Consequently SAP was on the forefront supporting the Microsoft IA64 and x64 platforms.

The recommendation for all SAP customers is to use a 64-bit platform to run the SAP Application Server Layer as soon as possible. As for the database layer, investments into the 32-bit platform are to be avoided.

All recent SAP products are available on IA64 and x64. Starting with 4.6C, SAP supports IA64. Starting with SAP 6.20 Basis, native x64 support is available. Transition to 64-bit with SAP is not an all or nothing move. The transition can be done server-by-server or system-by-system. SAP does support platform-heterogeneous systems between the three different platforms. In the past, it might have been common to run a bigger IA64 database server in a high-end system while leaving the SAP application server tier on x86 commodity application server. This still can be done using x64 servers for the SAP application tier. Even having the SAP Central Instance running in a cluster with the IA64 database server and having the rest of application server instances on x64 works, because SAP stores the SAP executables on the Central Instance in platform-dependent directories.

Memory Volumes

Sizing of SAP systems usually is done by hardware vendors. However, when you look at successful implementation on all sizes, certain rules-of-thumb emerge.

Assuming you have a dedicated database server that should be able to be pushed to its limits, you should have a minimum of 4 GB physical memory per processor core (not per socket). Running on an industry standard 4-socket Quad-Core server, this would require at least 64 GB of memory. With the upcoming new generation of new Intel and AMD processors in 2009 and 2010, it is expected that the 4-GB-per-core recommendation will move to at least 6 GB per core. This 50 percent increase is a result of the fact that per-core performance significantly increases and hence creates more memory demand. At least on the Intel side, simultaneous Hyperthreading is introduced again. Measurements with first generations of these processors in the labs of Intel, Microsoft and hardware vendors revealed that the new Hyperthreading can increase throughput 20 to 40 percent.

Configuring a 64-bit server for running an SAP ABAP/Java instance, you should calculate 4 to 6 GB real memory per processor core to fully leverage the available processor resources.
Storage Selection

Based on current pricing of disk storage in the desktop space, it may seem that disks are cheap. This thinking can lead to some misperceptions when planning an SAP deployment: In fact, the exact opposite can be true, depending on your business needs and the importance and sensitivity of SAP applications in your environment. For example, your business may require your disk storage to be:

- Extremely failure tolerant.
- Flexible in deployment.
- Sharable among systems.
- Capable of eventually delivering some type of replication.

Taking these considerations into account and prioritizing those abilities, your first choice is usually between direct attached storage and SAN/NAS devices. Direct attached storage provides better price and performance than any of the SAN/NAS storage solutions. You can configure the storage in great failure-tolerant ways, and it shows relative flexibility in deployments. The point where direct attached storage usually can’t compete with SAN/NAS devices is sharing and storage replication, which are nearly impossible with direct attached storage. In an average company using SAP, the number of SAP systems and databases would require you to maintain a great deal of direct attached storage. For example, if your company uses five SAP products in production, test, and development systems, you are looking at 15 databases, and eventually, depending on the size of the systems, 15 different servers running these databases. Actual SAP deployments can contain many more products and servers; this relatively small number of databases and database servers reveals the maintenance efforts involved throughout the complete SAP landscape if you choose direct attached storage.

Many deployments use SAN/NAS-based storage instead. We have seen this not only in large customer environments, but also in midrange customer environments, where storage consolidation efforts led to investment into SAN and NAS devices. In many of smaller SAP customer environments, the theory of flexible deployment and sharing of storage works very well. Storage consolidation for SAP development systems using SAN/NAS devices is also effective in most cases.

However, in large customer environments, you may need to dedicate one SAN/NAS frame to the production database of a bigger SAP system, especially for SAP ERP or SAP BW systems with high workloads. Throughput requirements in I/O operations per second can even prevent you from leveraging available disk space for test systems or other software. Especially in high-end customer deployments, you may need to avoid sharing the disks of a SAN device with other databases so that you can achieve deterministic and solid performance for the main production SAP database.

Many customers share disks within a SAN/NAS device for smaller SAP systems with less workload, but even in this case, you need to think ahead and plan carefully, because the SAN/NAS device is a single point of failure. You should group SAP databases and third-party products (like tax management software) on SAN devices so that certain workflows (like working through customer orders or creating deliveries) can still be executed. Avoid creating a situation in which maintenance on one SAN device affects the complete SAP landscape because databases of applications are stored on it, making it basically impossible to work with the remaining SAP applications that are up and
running. You should also bear in mind that the user community might develop very
different usage times for different pieces of SAP products. For example, SAP SCM usage
might peak over the weekend, so the best time for maintenance might be during the
week. For SAP BW, weekend availability may not be as important as 24-hour availability
during the work week.

Besides consolidation aspects, SAN and NAS devices often are chosen because they
offer features that enable you to use snapshot backups using the Windows Volume
Shadow Copy Service (VSS) or the SQL Server Virtual Backup Device Interface (VDI).
Most of the bigger SAN/NAS vendors offer such features, which in our experience have
worked well for specific customer situations.

Another advantage of SAN/NAS devices usually is the huge cache of multiple GBs these
devices are equipped with. Because of these caches, not every I/O operation coming
from SQL Server hits the underlying disks; they are served out of those caches. In our
experience, the cache hit ratio of these caches is nowhere close to the cache hit ratios
of SQL Server, but more I/O throughput per disk can be achieved this way than it can
by using direct attached storage with small caches on the controllers.

**Important:** Even if you use SAN/NAS devices, the question of number of disks utilized
is determined by the volume of the database only in rare cases. **But the number of
disks is determined by the throughput in I/O operations per seconds (IOPS).**
Therefore discussions around storage investments or configurations should be driven by
IOPS, failure tolerance, and according to SLAs behind those requests and less by how
many disk volume is required. We plan to provide detailed discussions in Part III of this
white paper.

**General Remarks for Sharing Hardware Resources Between Different SAP Systems**

When you look at sharing hardware resources, whether servers or storage, the driving
factor is usually cost savings. This is a good way think when you are considering the
dimensions of the server, databases, and even systems. However, there are people who
look at an SAP application in a very different way. They look at it from a business
process perspective. For example, they think in terms of the processes of fulfilling
orders, sending out invoices, executing payroll, and so on. These people have a
scorecard that indicates whether the business process as a whole is running or not.
The SAP application is a part of these business processes or workflows. It might even
be the actual heart of the process; but it is certainly not the only software/hardware
component that makes the whole business process. When considering sharing of
hardware between different software components, you should to analyze the effect of
hardware sharing on different business processes. Here, some smaller software
components that supplement SAP applications for very specific functionality come into
focus: fax server software, tax calculation software, software to move EDI documents
to IDOCs, and so on. The effects of hardware sharing on business processes are
important even for the main applications, when you think about the tight coupling
between SAP CRM and SAP ERP for some business processes. These aspects of looking
into the SAP landscape from the view of a business process are very important for
disaster recovery planning.
Installation and Configuration

Careful planning is the cornerstone of a successful SAP and SQL Server installation. For example, no installation should begin before all prerequisite tasks are finished, such as defining the machine names, developing the physical layout of the SAP product database, and sizing the disk space requirements. Eleven years ago, when decisions about how SAP should be implemented on SQL Server 7.0 were made, the base criterion was to create a SAP implementation that required the least administration and maintenance on any database. When decisions about whether to leverage features like file groups (also known as table spaces) were made, the primary considerations were administration, maintenance, and support. This is the reason that file groups are not included.

For customer implementations, the overarching theme should be to avoid complicating the landscape and configuration: keep it simple. SAP landscapes are complex by nature. Having anywhere between four and ten production SAP applications (plus test and development systems for those applications) working on ABAP and/or Java stack is inherently complex. Any additional complexity of database configurations or high availability (HA) configurations (that is not truly necessary) only induces higher maintenance, increases the risk of human error during states of high workload, and demands increased documentation. Therefore try to achieve a lean and minimally complex deployment of your database instances for SAP applications.

SQL Server Installation with SAP

The installation of SAP products does not require SQL Server 2008 or SQL Server 2005 to be configured in a special way, with the exception of the code page. For more information, see “Collation and Code Page” earlier in this document.

During the initial data load, the SAP database is set to the simple recovery model. This setting is reversed when the data load and the installation of the SAP product is completed.

SAP provides a setup script on their installation DVDs that installs SQL Server in the correct configuration for SAP. This configuration also includes installation of recommended SQL Server Service Packs and patches. The script is called SQL4SAP.

Which SQL Server Build to Use

For SQL Server 2005 and SQL Server 2008, you should always use the latest build. Microsoft releases single hotfixes only in exceptional cases. Instead, Microsoft releases cumulative updates (CUs) for SQL Server 2008 and SQL Server 2005 every two months. CUs use an RTM or service pack as a base and cumulatively build upon it. This means a fix in CU8 is also included in CU9.

In the same way the most recent service pack always is supported by SAP. For more information, see SAP OSS Note #62988.

Volume and Growth Projections

Projections for the volume and growth of SAP product databases over time are often underestimated. The growth rate is typically higher than expected. When the growth...
projections are in error, SAP products can require reconfiguration and adaptation on the hardware or on the database in order to satisfy hardware growth requirements.

**Note:** Any sizing of an SAP system must be performed by an SAP-certified hardware partner.

For the initial configuration, ensure that the size of the database is sufficient to sustain the first 6 to 12 months of production. In addition, assume that each subsequent release of an SAP product will tend to increase the weekly or monthly growth rate, as compared to its predecessor. For example:

- For small companies, SAP application databases can grow 2 to 5 GB each week.
- For midrange companies, SAP application databases can grow 5 to 15 GB each week.
- For larger companies, the databases can grow 15 to 25 GB each week.
- For very large implementations, the database can grow 50 GB or more each week.

Even if SAP archiving is used, most customer systems display a steady and consistent growth rate in database size over time. Therefore it does not matter if a database of 2 terabytes suddenly reveals 800 GB or more of free space after severe data archiving. Unlike some competitors’ RDBMSs, where the sizes of backups are defined by the sizes of the data files, SQL Server does not infuse any disadvantage having larger free space in the data files. That is, SQL Server backs up only the real used extents within the database files. Hence free space within the database doesn’t become a liability for backup/restore processing. You do not need to shrink the SAP database because of unused space. Shrinking the database can take a long time, because data may need to be moved. It can also result in extents where the logical and physical order are opposite, which has in some cases disturbed sophisticated storage back-end logic. We do not recommend using the *autoshrink* option for an SAP database.

**Number and Size of SQL Server Data Files**

The considerations for defining the number and size of data files include:

- **Disk configuration.** Evaluate the available storage hardware, including the number of disk arrays and partitions and the amount of available storage space on disk arrays. It is obvious that a database file cannot become larger than the volume where it is stored. You should keep enough free space on the volumes in order to allow autogrow of the data files. The more volumes you use, the more data files you need.

- **Nonproduction systems.** Keep in mind that each SAP installation consists of many systems: In addition to the production system, there are also development, consolidation, and test systems. Customers typically synchronize the test systems by means of a system copy of the production system. The number of data files does matter here when using a database backup/restore for the system copy. You cannot change the number and size of the database files during the restore. Therefore you should also keep the disk configuration of all nonproduction systems in mind when defining the number of data files of the production system.

- **Estimated database growth.** Consider the estimated database growth during production. After the database size reaches the limits of the disk array, you may
simply want to add another array. Having more database files makes it easier to
distribute the I/O load of the database on the existing and future disk arrays.

- **Update performance.** When performing highly concurrent INSERTs, UPDATEs, and
DELETEs on the same table you may see a performance decrease if the number of
data files is insufficient. You typically see this on an SAP system for the update
tables (VBHEADER, VBDATA, VBMOD) or queuing tables. When loading the database
during installation or heterogeneous system copy, you might see the same issue.
The reasons for this are hotspots on some administration pages (PFS pages). The
performance of SELECTs is not affected by the number of data files.

- **Manageability.** For best performance, the data files should be equally filled. This
feature, called *proportional fill*, is described later in this paper. It does not work
optimally after a single data file is full. A DBA should regularly check the free space
of each data file. Having too many data files results in huge administration
overhead. Though SQL Server can support 32,767 files per database, it’s not a good
idea to approach this number of files. At this point in time, we suggest that a
reasonable limit would be 64 data files.

**Performance Impact**

Before discussing the impact of the number of data files, we will define the
nomenclature a bit. Over the last six years, many changes have occurred in processor
technology. From Microsoft side, today we talk about *sockets* or *processors*, which are
the physical pieces of silicon formerly called *processor* and *base* for SQL Server
processor licensing. We look at cores where one socket today usually has up to 6 cores
and CPU threads, which usually is the number of cores of a socket when no
Hyperthreading is used, or double the amount of cores when Hyperthreading is
available and used.

The number of data files is chosen during SAP product installation. Three or four data
files (the number suggested by SAP installers) usually are appropriate for smaller SAP
installations. In SQL Server, data access performance within a data file is related not to
the size of the data file but to its concurrency on internal administration pages within
the data file.

For the best performance, the ideal number of data files for the SAP installation
depends in most cases on the number of CPU threads or cores dedicated to SQL Server.
Imagine that SQL Server can have one request active per CPU thread on a server.
Assuming that there would be one data file only for the database and all 16 requests
actively processed by SQL Server would operate against the same table (like against
the SAP ERP update request queuing table VBDATA) performing inserts and deletes
requiring allocations and deallocations, there would be one or two administration pages
only, which would need to deal with the concurrency of the 16 threads performing their
tasks. Assuming that there are more data files, chances are extremely high that the
requests would be distributed over the multiple data files and hence involve a larger
number of administration pages. This means that the concurrency on individual
administration pages is lower during such modification operations.

In an earlier version of this paper, we gave a hard-and-fast recommendation to shoot
for a 1:1 ratio of CPU threads and data files. All in all, this was a very conservative
recommendation, taking into account that the database server could grow dramatically
in size before causing contention on SQL Server administration pages. Looking into deployed customer systems and the hardware that shows up on the horizon, we decided to soften this recommendation a bit. In deployed customer systems, we observed that there is not a problem with having 3 to 5 processor cores working against one data file. However, we saw a small number of cases where 6 or 8 CPU threads working against one data file caused concurrency issues under heavy allocation/deallocation workload on some internal administration pages of SAP queuing tables.

The recommendation regarding the number of data files depends on the system load. Therefore we introduced the following categorization of four systems sizes (which we call T-shirt sizing) based on the number of CPU cores on the database server. The number of CPU cores is a good indicator for the expected system load. We purposely did not use the number of CPU threads. Therefore a system falls into the same category, whether you turn on Hyperthreading or not. The number of CPU cores is related to the cores used by SQL Server. When running an SAP instance on the same server as SQL Server, you should only count the number of CPU cores which are typically busy by SQL Server.

- **Small sized systems**, where 4 data files should be fine. These systems usually run on dedicated database servers that have around 4 cores.
- **Medium sized systems**, where at least 8 data files are required. These systems usually run on dedicated database servers that have between 8 and 16 CPU cores.
- **Large sized systems** where a minimum of 16 data files are required. These are usually systems that run on hardware that has between 16 and 32 CPU cores.
- **Xtra Large sized systems**. Upcoming hardware over the next years certainly will support up to 256 CPU cores. However, we don’t necessarily expect a lot of customers deploying this kind of hardware for one dedicated database server, servicing one database of an SAP application. For XL systems we recommend 32 to 64 data files.

Changing the number of data files should only be considered if a system is far beyond the range of our T-shirt sizing. You should not add data files just because a file is full. Instead, increase the size of the existing files so that each file contains approximately the same amount of free space. The number of recommended data files is not dependent on the size of the database.

If you need to increase the number of data files, the best way is to completely export and then reimport the database using R3load. This method ensures an equally distribution of the data for each table over the data files. However, this is a very time consuming procedure. You typically only want to reimport the whole database if it is really necessary. Because an SAP Unicode migration includes the reimport of the whole database, you could use this event for adapting the number of data files.

The easiest way to increase the number of data files is simply to add additional files. For the best distribution and leverage of SQL Server’s proportional fill, create new data files of the same size and distribute the files evenly over the disk arrays. At the same time, increase the size of the existing files so that each file contains approximately the same amount of free space (both the new and the old data files). This enables SQL Server to distribute new data evenly between the old and new data files.
Decreasing the number of data files typically requires a database reload. Shrinking and removing data files is not recommended, because this will not result in an equally data distribution. If you have too many data files and do not have the time to reload the database, it is better to keep the huge number of files.

Number of LUNs or Partitions

Now that you have defined the number of data files for an SAP database, you need to determine how many LUNs or partitions you will need to store the data files. Because you are using highly complex SANs that are trimmed for sharing files among many systems, you might think of using only one LUN. If you come from UNIX environments, you might even think of using dozens of LUNs, which is necessary with some UNIX systems. None of these extreme approaches is a good idea. If you have only one LUN, you can experience performance issues. Because there are I/O queues in the operating system I/O path, and because the HBA drivers often provide queuing on a per-LUN basis, having a single LUN can throttle the number of I/Os that are issued against SQL Server data files. Besides this, there are some limitations of LUN sizes by the specific SAN vendors: for example, there are SAN devices that allow large multiterabyte LUNs, but when SAN replication is used, the LUN size may be much smaller than the theoretical limit. On the other side, there is no need to have 48 or 64 LUNs just because there are that many data files. 16 or 32 LUNs are sufficient for larger systems. Lesser if there are lesser data files.

Based on our experience with database files and LUNs, we strongly recommend that you isolate transaction log file(s) onto a separate LUN. Ideally, this LUN is completely separated, even on the disk level, from the data files. (Most SAN vendors provide a way to set aside a few disks and isolate them to provide a write optimized, mirrored disk resource like a RAID 1 LUN). With SQL Server’s synchronous, write-log first transaction log behavior, the transaction log write latency is a critical potential bottleneck that has occasionally created substantial pain in customer environments. You can avoid this pain by using proper SAN and database data file architecture.

Usage of Windows Mountpoints

Since SQL Server 2005, Windows mountpoints are supported even in Microsoft Cluster Services (MSCS) configurations. From a performance point of view there is no reason not to go with mountpoints. None of the customers who used mountpoints instead of drive letters has observed any negative impact on performance so far. Customers who chose with mountpoints have been larger customers who had larger numbers of data files. Because we have not observed any problems with them, we feel confident in recommending mountpoints to customers for whom they would be a good solution. The larger advantage is that you can operate with more LUNs/partitions without running into the limitations of drive letters under Windows.

SAP and SQL Server Filegroups

SQL Server allows the creation of filegroups for placing individual tables into dedicated locations. SQL Server filegroups are similar to Oracle tablespaces.
However, because filegroups require more complex administration, they are counterproductive for providing easy file management. Therefore, SAP does not support SQL Server filegroups. Using filegroups in SAP databases can lead to errors in SAP release upgrades or failures while applying support packages because SAP products cannot create objects in filegroups. Even in highly intense SAP benchmarks that were conducted with hardware vendors, no filegroups were used. Tests performed with filegroups confirmed that there was no performance advantage in their use.

Not leveraging filegroups prevents use of the single filegroup restore and piecemeal backup/restore features of SQL Server. However, considering the SAP schema underneath NetWeaver based applications, there also is no real reasonable approach to splitting the SAP schemas so that an SAP application like SAP ERP can run with one filegroup not accessible.

Proportional Fill

In figure 11, the sample demonstrates the proportional fill feature, which is used to spread the data between the data files according to each file’s available free space. The considerations for using the proportional fill include:

- **Extending the data file size.** As volume increases, manually extend the size of the data files by the same amount. Assume that each file was created to the same size.

- **Proportionally fill the data files.** When data is inserted into the database, the data files are filled proportionally using the free space in each data file.

Figure 11: SAP ERP database with 12 data files of the same size

If the data files were initially created to the same size, SQL Server distributes the data of each table evenly over the existing files with the size of a SQL Server extent (64 KB). This makes the read/write workload even, simplifies placing the data files over a storage backend, and avoids hot spots on specific disks.

- **Recalculating the proportional fill factor.** Periodically, SQL Server recalculates the proportional fill factor for each file based on the available free space.

When one of the data files becomes full, the proportional fill goes out of balance. In this case, it fills the other files, grows only one of the files, and then fills this file. Then it grows the next file, and so on.
• **Setting autogrowth.** On SQL Server 2005 and SQL Server 2008, avoid situations where automatic growth is executed by manually increasing the size of the data files proportionally in advance. Although the data files can be manually manipulated proactively, leave autogrowth on as a safety measure in case of an emergency such as when the database runs full. The reason for this recommendation is that SQL Server only grows one of the data files and then fills this one up before growing the next data file. This undermines the ideal proportionate fill that you want to achieve in SAP configurations. The autogrowth rate set by SAP when creating the database files is 10%, which is a good extension size. However, in cases where database files become larger and eventually go into a few hundred gigabytes, consider setting the extension to a fixed size.

Another reason to avoid autogrowth and instead extend database files manually is the fact that SQL Server doesn’t persist a record of which files have been autogrown. The information about which files have been autogrown is stored in memory, not in a table. When SQL Server is restarted, the information is no longer available. Consider a situation where the customer is not monitoring the free space in the data files. Over the past two weeks, two of the eight data files have been autogrown. During the planned weekend downtime, SQL Server restarts. SQL Server has no record of the fact that it autogrew two of the eight files. If the database runs full again, SQL Server doesn’t have a record of files grown before; it might as well grow one of the files that it did grow before, whereas the other six files that have not changed might not be grown immediately.

• **SQL Server 2008 autogrowth changes:** In SQL Server 2008, a change was made to the way autogrowth works. The change affects a single trace flag, trace flag 1117. If trace flag 1117 is enabled as startup trace flag for SQL Server 2008, all data files are grown by the portion that is set for each of the files, if the database is full and the files need to be extended. In previous versions, only one file was extended. If you set this trace flag and if you set the same growth portion for each file, autogrowth should work in a way that it doesn’t unbalance the proportionate fill between the data files. Even with this feature in place, we still recommend that you monitor free space in SAP transaction DB02 or DBA Cockpit or with the SQL Server command DBCC SHOWFILESTATS. If the database files run full, we still recommend manual extension of the file by the same amount. For more information about trace flag 1117, see OSS Note #1238993.

### Number of Log Files for SAP Databases

The number of database transaction log files is specified during SAP product installation. Because the SQL Server transaction log is written sequentially, typically only one transaction log file is required. The considerations for defining the number and size of log files include:

• **Using multiple log files.** Multiple physical log files on different partitions do not improve performance. Multiple log files are generally used on a case-by-case basis, if there is insufficient space on one volume. SQL Server does not write to multiple log files in parallel. In order to ensure that the transaction log file is not lost in a hardware failure, duplicate the log in storage by using at least RAID 1.

• **Setting the physical log file size.** It is important to create a physical log file that is of a sufficient size, even if the log file is set to autogrow. The SAP installation
program typically uses a default log file size of 1 GB. For larger SAP installations, consider setting the initial size of the transaction log file to 5 or 10 GB. In most high-end SAP systems, the size of the transaction log file ranges up over 100 GB. Operations such as creating or re-creating indexes can create large volumes of data in the transaction log files. For example, if database mirroring is suspended, the data that is not sent to the mirror remains in the transaction log. Remaining suspended for many hours or even multiple days can lead to very high volumes of transaction log records needing to be stored in the transaction log.

- **Setting the size of virtual log files.** Internally, the physical transaction log file is administered by virtual log files. SQL Server sees these virtual log files as a ring buffer. It fills up the first one, moves and fills the next one, and so on, and after it reaches the end, it returns to the first virtual log file, which the system expects to have been emptied in the meantime by a transaction log backup. The size and number of virtual log files depends on the size of the physical log file, plus the growth portion. For performance reasons, it is better to have fewer larger virtual log files, rather than a large number of small virtual log files. During installation, SAP initially configures the growth factor of the log file to 10 percent, which is a good size when you start with a log file size of 5 GB or 10 GB. But if the log file grows to 50 GB or more, 10 percent growth could have a significant effect.

The basic problem is that log file extension works differently than data file extension. Extending data files usually takes only a few seconds; the time it takes does not depend on the amount of the extension. This is because in SQL Server 2008 and SQL Server 2005, the pages in the expanded file space are no longer immediately zeroed out, as they were in previous version. This is not true, however, for log files. The portion the by which log files is extended must be zeroed out. If the log file is automatically grown because the transaction log ran full, all operations usually need to wait until the log file extension has finished. The larger the extension, the longer it takes to zero out the extension, and while the extension is being zeroed out, the system cannot respond to modification requests. Therefore it might make sense to switch to a fixed extension size of 2 or 3 GB when operating with transaction log files of larger sizes.

**tempdb Considerations**

tempdb is a system database that is used by SQL Server to store temporary objects. For example, tempdb is used by queries to execute large join, sort, and group operations when the SQL Server buffer pool cannot provide enough memory. In SQL Server 2008 and SQL Server 2005, tempdb is also heavily used by base features including the snapshot isolation level read committed snapshot isolation, which is used for online index maintenance or to assemble data that is pushed into text/image or varchar(max)/varbinary(max) data type columns.

SAP products such as SAP ERP, SAP Customer Relations Management (CRM), and SAP BW stress tempdb in different ways, and each has different performance requirements. Basically, the usage of tempdb differs according to the type of workload. The two types of workloads are an online transaction workload as in the case of SAP ERP, SAP CRM, and SAP Enterprise Portal (SAP EP) and an online analytical processing (OLAP) workload such as those created by SAP BW, SEM, and to a degree SAP Strategic Enterprise Management (SCM):
**Online transaction type workloads (OLTP).** SAP products with online transaction workloads, including SAP CRM and SAP ERP, use tempdb infrequently for larger join operations, aggregation, and smaller sorts. tempdb typically does not load or sort gigabytes of data. Overall system performance is less dependent on the throughput capabilities of tempdb. For installation, set the tempdb space from 2 to 4 GB with four to eight spindles. For SAN storage, tempdb can share space with the tempdb log files.

**OLAP-oriented workloads.** For SAP BW, SAP SEM, and to a degree SAP SCM, tempdb can expand to larger sizes. For example, join operations, especially those that fully scan the fact table of a cube, can use gigabytes of space to store temporary result sets, large sorts, or hash probes or to build aggregates. In these cases, tempdb is used to perform most of the work because the memory space usually is not sufficient to process gigabytes of data. In extreme cases, if the whole fact table is read in order to build an aggregate, tempdb can grow to the size of the fact table, up to several hundred gigabytes.

**SAP BW tempdb performance.** For SAP BW, SAP SEM, and to a degree SAP SCM, tempdb I/O performance can become a major bottleneck when executing reporting queries that use the fact table or perform aggregation. In order to prevent bottlenecks, set the tempdb size to 1.5 times the space of the largest fact table. Manage tempdb strictly like a normal SAP BW database. Eventually use one data file of tempdb on the same partition with each data file of the SAP BW (SAP SEM and SAP SCM) database. In this case, the tempdb space should provide fast read/write performance. In addition, do not place tempdb data files together on disks that contain the transaction logs of any databases. You also can put the tempdb log file with the log files of the SAP BW database. In these cases, we recommend that you use SQL Server trace flag 1118 as startup trace flag (for more information, see this Knowledge Base article). This trace flag disables mixed extent allocations on all databases. This trace flag, in combination with multiple data files for SQL Server tempdb, could increase the throughput of tempdb. If you are using SQL Server 2005, trace flag 1118 requires an additional patch. For more information, see this Knowledge Base article.

**SSD disks for tempdb.** New SSD technologies look promising for speeding up tempdb interaction with SAP BW, SAP SEM, and SAP SCM. An attractive alternative is to move tempdb to 2-4 SSD disks. In this case, the SSD disks should be configured in a RAID1+0 configuration. However, one you would need to calculate to have one controller exclusively for those few SSD disks, due to the large throughput those can achieve. Other alternatives could be products like the ones described at this link [http://www.fusionio.com/Products.aspx](http://www.fusionio.com/Products.aspx). Such SSD-based devices could accelerate SAP BW performance.

---

**Multi-Instance Support for SAP Components**

SAP products support running multiple instances of SQL Server 2008 or SQL Server 2005 on the same server. Multiple instance SAP configurations are typically used to run test, development, and demo systems on one server or these days even for production.

An alternative is to run two or three SAP databases on one SQL Server 2008 or SQL Server 2005 instance. The advantage of having multiple SAP databases handled by one SQL Server 2008 or SQL Server 2005 instance is easy administration. The trade-off is
that different SAP systems must share the resources of one SQL Server instance, including data cache, processors, and tempdb. The resources used by each instance can be restricted, although adding this restriction entails more administration overhead.

In order to restrict resources consumed by different SQL Server instances on one server, you can use the following SQL Server options, which are both discussed in detail later in this paper:

- **affinity mask**
- **max server memory**

Both these parameters allow restricting processor and memory resource usable for one specific SQL Server instances. The affinity mask options could be useful if multiple instances should be put on a larger server based on NUMA architecture. The parameters could be used to locate or restrict different SQL Server instances to different NUMA nodes of the hardware. The problem with managing the CPU resource consumption of SQL Server instances with an affinity mask can be:

- The granularity of one CPU (as displayed by Windows Task Manager – Dual-Core processor = 2 CPU, Hyperthreading = 2 CPUs) might not be fine enough.
- Setting the affinity mask options might not provide enough flexibility for the specific usage. (For example, consider a system used for an afternoon in a training class).

In these cases, we recommend that you use Windows System Resource Manager (WSRM) to restrict CPU resources with SQL Server. WSRM provides more flexibility in granularity as well as in throttling. For example, WSRM will start throttling down the CPU consumption of an instance to its defined limits, if the overall CPU consumption on the server goes beyond 75 percent. However, it is not recommended that you use WSRM for restricting the memory consumption of a SQL Server instance. The **max server memory** option of SQL Server is the best way to control memory consumption.

### SAP Database Configuration

This section describes the settings that are used by a SQL Server 2008 instance running an SAP database. It also discusses the special settings for SAP in Lock Manager in SQL Server 2008 or SQL Server 2005.

Most of the SQL Server 2008 and SQL Server 2005 parameters described in this section are set correctly when the SAP software is installed. These installation settings cover most needs, even on high-end systems. These parameter settings are typically changed only occasionally to adapt to specific configurations of the hardware (like memory).

Note that most of the tuning and balancing on memory allocations, I/O, and so on in a SQL Server instance is automated within the boundaries configured with the parameters which are discussed next

### SQL Server Configuration for SAP

SQL Server 2008 and SQL Server 2005 contain approximately 50 instance global configuration settings that are usually accessed using sp_configure. Typically, only a small number of these configuration settings require a change to be made in order to run SAP products proficiently. There are no different configuration settings necessary to
tune specifically for different SAP products and their SQL Server usage profile. The same configurations fit the needs of all the different SAP products.

This section describes the options that are commonly used for tuning the SAP workload.

max server memory and min server memory

In all SQL Server configurations with SAP, we recommend a fixed memory allocation. However, you must ensure that adequate memory is left for the operating system. So what is enough for the operating system? In our current deployments these rules of thumb worked out pretty well:

- 16 GB real memory give SQL Server up to 13.0-13.5 GB memory
- 32 GB real memory give SQL Server up to 26-28 GB memory
- 64 GB real memory give SQL Server up to 52-56 GB memory
- 128 GB real memory give SQL Server up to 104-112 GB memory
- 256 GB real memory give SQL Server up to 225-240 GB memory

Be aware that SQL Server will allocate memory outside its buffer pool. In the information above, we simply assume that an SAP workload that requires more memory for SQL Server buffer pool also will require more memory from outside the SQL Server buffer pool as well. Memory allocated outside of SQL Server also includes the stack of SQL Server worker thread (64-bit). For instance, if 200 worker threads on IA64 are instantiated, up to 800 MB of memory will be consumed by SQL Server outside of the buffer pool. In addition, many agents and products run on such servers in modern data centers. They all require server memory as well.

Assume that even a lightly loaded SAP ABAP or Java instance requires at least 6 GB of memory to be allocated in order to run without problems. Eventually, more memory will
be required with increasing workload on such an instance. If the SAP instance is heavily loaded, the memory requirements would range up to 24 or 32 GB. In pure two-tier installations where the database instance and the only SAP application server run on the same server, the main amount of memory needs to be left for the SAP application instance in most of the cases. To get an idea of where to start tuning, divide the memory in a 5 to 1 ratio. Do this for all SAP applications except SAP BW, SAPSEM, and SAP SCM. Five parts of the physical server memory should be left for the SAP instance, and one part should be allocated for SQL Server. For the exceptions of SAP BW, SAP SEM, and SAP SCM start with a ratio of 2.5 to 1, where 2.5 is the amount of memory allocated to SAP and 1 is the amount allocated to SQL Server. Based on those initial settings, monitor memory consumption for both SAP and SQL Server, and adjust accordingly. We plan to discuss this in more detail in Part III of this white paper.

When additional application servers with SAP application instances are added, SQL Server will require more and more memory to deliver the same performance. Therefore over time an SAP instance running on the same sever with SQL Server will need to be reduced in the memory footprint and load up to the point where you should move the instance off the database server.

Note that the SQL Server 'server memory' parameters can be adjusted dynamically without restarting SQL Server.

awe enabled

The awe enabled option is used to extend the SQL Server buffer pool beyond the virtual address space on 32-bit for high-end database servers. Address Windowing Extensions (AWE) should not be used with any of the 64-bit platforms. AWE administrated memory can be used only for data pages. Caches and structures such as statement or procedure cache, lock structures, and buffer headers remain in the virtual address space and do not use memory accessed over AWE functionality.

- In order to use the awe enabled parameter to exploit larger amounts of memory on the 32-bit platform, the /PAE option must be set in the boot.ini file of Windows Server 2003, or it needs to be added to the Windows Server 2008 BCD store with this command ‘BCDEDIT /SET PAE ForceEnable’
  For more information about the Windows Server 2008 BCD store, see the following: http://technet.microsoft.com/nl-nl/library/cc721886.aspx
- The user context used to start SQL Server requires local LOCK PAGES IN MEMORY permissions in order to take advantage of AWE.
- For SAP, the 3 GB virtual address space is typically used. The 3 GB memory is enabled by adding the /3gb option in the boot line in the boot.ini file.

In some cases, using AWE on 32-bit servers with up to 16 GB of real memory can achieve a positive customer experience for some SAP workloads. However, using more than 16 GB of memory on a 32-bit SQL Server 2005 platform under an SAP workload is not recommended. We only describe this parameter because we still have customers running 32-bit versions of SQL Server 2005. However, if you are running serious workloads on your SAP applications, you should move to 64-bit as soon as possible.

Although it still exists in SQL Server 2008, this option should not apply to any SAP installations. As stated previously, SAP exclusively supports 64-bit releases of SQL
max degree of parallelism

The **max degree of parallelism** option affects the maximum number of threads (CPUs) that can be used for parallel query execution. The **max degree of parallelism** option has a default setting of 0 after installation. This means all processors can be used in parallel to execute a query. Before a potential parallel query is executed, SQL Server 2008 and SQL Server 2005 check the available resources such as processors, available worker threads, and memory to determine the number of processors available to service the query. Based on those checks, a query might not be executed using all CPU threads available, instead using only a few CPU threads (or even just one CPU thread).

In case of executing a query in parallel, multiple streams of data get sorted in parallel and are merged afterwards, making the consumption of buffers in the cache higher. In addition, the momentary CPU consumption by a single query usually increases with the degree of parallelism.

A typical example where this run time decision on query parallelism is disturbing would be the load SAP BW applies to the database. A few complex queries might benefit from being executed on multiple CPU threads, but the majority might be executed single threaded to avoid overcommitting processor and memory resources, because the first few queries executed in parallel would consume the majority of those resources. Although a query executed in parallel could be much faster, there is a point at which the parallel query execution becomes inefficient and can even extend the execution time. For example, parallel queries performing small joins and aggregations on small data sets can become inefficient when they execute on all 16 processors of a server. Due to different degrees of parallelism chosen at execution time, response times for one query can be different depending on resource availability such as CPU and memory. The most severe effect reported, though, is the varying response times experienced by SAP system’s end users. Like administrators, end users want predictability in a system; they want predictable performance in the most important areas of demand. Allowing SQL Server to execute queries in parallel compromises this predictability by varying run times of one and the same query due to different run time decisions on parallel
execution. Therefore the general recommendation for SQL Server Instances running SAP application workloads is to set the max degree of parallelism option to 1.

For offline database administration tasks, set max degree of parallelism to 0. This will speed up tasks like creating indexes, rebuilding indexes (if you want to perform those offline), checkdb, and other maintenance work. Such tasks run faster in this configuration because they can leverage more CPUs. This option can also be set to 0 for tasks like creating aggregates in an SAP BW system.

Toggling the setting of max degree of parallelism can be done online. You do not need to restart the SQL Server process for this change to take effect.

max worker threads

The max worker threads option defines the maximum number of worker threads that are available to handle user requests. In contrast to competitor databases, an incoming connection to SQL Server 2005 does not result in a shadow process. Rather, the incoming connection is assigned to a scheduler thread. One scheduler thread is assigned to each of the processors, cores, or threads (CPUs) on which the SQL Server instance is allowed to run. For more information, see the “affinity mask” section.

Because the worker threads are created dynamically on demand, the setting represents not the worker threads created at startup, but the ceiling that can be achieved.

Incoming connections are assigned to the scheduler threads in a round-robin manner. This provides an even distribution of the hundreds of connections, or in many cases, more than thousand of connections that can result in large three-tier SAP systems.

Every scheduler thread has a pool of worker threads. When a query request arrives over a connection, the scheduler thread assigns a worker thread, which executes the query request. After the request has been served, the worker thread is available to serve another request coming from any of the connections assigned to the scheduler thread. However, if the execution of a request is blocked by waiting on I/O or by a lock on a row which should be accessed, the request will remain assigned to the worker thread and will go into an inactive status as long as the request is blocked.
There is no 1:1 relationship between the number of connections and worker threads. SQL Server 2008 and SQL Server 2005 can reassign connections to other schedulers if the original scheduler is overloaded, thereby avoiding an uneven CPU load.

For SQL Server 2008 and SQL Server 2005, the **max worker threads** option is dynamic and should remain at the default of 0. Note that this setting is different from the one used with SQL Server 2000, where a value had to be set because the logic used was not dynamic. Starting with SQL Server 2005, this option is dynamic; the maximum number of worker threads changes based on the platform (32-bit or 64-bit) and on the number of CPU threads available (cores or threads in case of Hyperthreading). Generally, it will start with a maximum of 256 on 4 CPU threads and will add another 4 threads per CPU thread on 32-bit or another additional 8 threads per CPU thread on 64-bit.

Be aware that every 64-bit worker thread will be able to allocate a maximum of 4 MB real memory (IA64) or 2 MB (x64) compared to 512 KB on 32-bit as stack out of real memory. This partially explains the recommendation to leave some space unallocated by the SQL Server buffer pool as described earlier in this paper.

In most cases, setting max worker threads to 0 has not caused problems. There were cases of running SAP Enterprise Portal where the number of maximum worker threads had to be increased to overcome issues.

The actual number of max worker threads can be evaluated with this query:

```sql
SELECT max_workers_count FROM sys.dm_os_sys_info
```

The stacksize of the worker threads can be evaluated with this query:

```sql
SELECT stack_size_in_bytes/(1024.0 * 1024.0) as stack_size_in_MB
FROM sys.dm_os_sys_info
```

### xp_cmdshell

This option disables or enables the usage of an extended procedure called **xp_cmdshell**, which enables operating system commands to be issued via SQL Server. By default this parameter is disabled after SQL Server is installed. However SAP needs the extended procedure **xp_cmdshell** for the database monitoring framework. In order to have all capabilities of the SAP database monitoring framework, we recommend that you set this option to 1.

### affinity mask, affinity64 mask, affinity I/O mask, affinity64 I/O mask

These options define the specific processors on which SQL Server worker and I/O threads can execute. When SQL Server 2008 or SQL Server 2005 runs as the dedicated database server or with an SAP application, the affinity mask option uses the default setting of 0.

The 0 setting permits SQL Server to execute on all processors. In most situations, the 0 setting provides the best performance because it avoids trapping SQL Server connections on a specific processor, while leaving capacity on other processors. The 0 setting is used with dedicated database servers and SAP two-tier configurations.
You can also use the affinity mask option in cases when multiple SQL Server instances and a number of SAP instances run on consolidated hardware. For example:

- When multiple instances of SQL Server run on a single server
- To assign each SQL Server instance to particular processes on the server

The affinity mask option is represented as a binary pattern that is expressed by a decimal number. The affinity mask settings are shown in the following table.

<table>
<thead>
<tr>
<th>Processor</th>
<th>affinity mask setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>First processor only</td>
<td>1</td>
</tr>
<tr>
<td>Second processor only</td>
<td>2</td>
</tr>
<tr>
<td>First and second processor only</td>
<td>3</td>
</tr>
<tr>
<td>First four processors</td>
<td>15</td>
</tr>
<tr>
<td>First eight processors</td>
<td>255</td>
</tr>
</tbody>
</table>

Unless you need to locate or restrict specific SQL Server instances to specific NUMA node, there might be a better possibility than affinity mask if you only need to cap the CPU consumption of a specific SQL Server instance. If you need to cap the CPU resource consumption, WSRM would be a better choice. For more information, see here.

Usage of the I/O affinity mask is not recommended with SAP workload. If affinity mask is used in a consolidation scenario, the affinity I/O mask should be set to exactly the same values as the affinity mask for the worker threads.

**lightweight pooling**

The **lightweight pooling** option is not recommended in general because it results in only minor performance improvements. This option is typically used for benchmark workloads that are extremely uniform and specific. Over the years Windows was able to minimize the differences between dealing with threads and lightweight pooling structures (sometimes known as fibers). So keep your hands off this option.

**network packet size**

The **network packet size** option determines the size of packets transmitted over the network. SQL Server 2005 allocates three buffers that use the parameter size for every connection. This parameter does not have an effect if the application requesting client contains a different value. That is, the client side specification of network packet size takes precedence over the server side setting. At connection time, SAP NetWeaver Application Server overwrites the network packet size parameter in SQL Server 2005 when it is set to a value other than 8192.

Because the value 8192 is set by the SAP client application, the network buffers are allocated outside of the SQL Server buffer pool, in the SQL Server virtual address space. SQL Server usually leaves a virtual address space of approximately 300 to 350 megabytes (MB) outside the buffer pool. In large SAP systems, up to 150 MB of the 300 to 350 MB can be allocated by the network buffers. In turn, this can lead to problems with the virtual address space in a 32-bit computing environment.
As a workaround in 32Bit, SQL Server can be forced to leave more virtual address space by using the `–g xxx` startup option to define a value greater than 256 MB, which is the default in SQL Server 2005. In this option, `xxx` is a number specifying the amount of virtual address space to leave, in MB. This workaround is not required with 64-bit computing platforms.

For SQL Server 2008, SAP applications changed their behavior with the move away from the OLED based porting layer to the new ODBC based SAP porting layer. SAP doesn’t set the network packet size anymore; the value that is set in the SQL Server configuration is taken. We recommend leaving it at a default of 4096. A lot of tests conducted by different hardware vendors, SAP and Microsoft confirmed that there is no decrease in performance impact by this change.

**priority boost**

The *priority boost* option defines the priority of SQL Server processes. A value of 1 gives SQL Server processes a slightly higher Windows scheduler priority. The current recommendation is to set the *priority boost* option to 0, which leaves SQL Server processes with a normal priority.

In the past, SAP and Microsoft have recommended having the *priority boost* option set to 1. However, this recommendation changed due to situations where operating system network threads were starved in favor of the SQL Server process, thereby causing failure situations in cluster failover situations. Independent of the SQL Server release, this recommendation still holds true.

**recovery interval (min)**

The *recovery interval (min)* option is used to control the checkpoint interval. SAP recommends using the default setting of 0. In customer scenarios, using 0 causes a checkpoint interval to occur every 30 to 60 seconds in situations where no other event triggered a checkpoint.

SQL Server checkpoint intervals are extremely sensitive to disk resources. SQL Server balances the number of I/Os during the checkpoints based on the response time from previously issued checkpoint I/O requests. You can see the high peak of page writes per second at the beginning of the checkpoint if you look at the *Checkpoint pages/sec* SQL Server Buffer Manager performance counter.
You will observe a peak at the start of the checkpoint and after that a reduction of checkpoint pages written per second after SQL Server realizes that the I/O subsystem is getting overloaded. In the past, there have been no customer issues with I/O flooding caused by SQL Server checkpoints under SAP workloads. Therefore the default value of 0 should be left unchanged. If checkpoints do flood the system, it is better to limit the volume a checkpoint can write instead of delaying checkpoints.

backup compression default (SQL Server 2008)

This option should be set to 1 if you want to use SQL Server 2008 backup compression. This setting does not exist in SQL Server 2005. The setting affects all interactively issued backup commands as well as native backup commands issued by SQL Server Agent, SQL Server Maintenance Wizards, or log shipping plans. However, the setting does not affect any third party backup tools that issue a SQL Server backup using the Windows VSS or SQL Server VDI interface. Backups issued using these interfaces are noncompressed, regardless of this configuration setting. Third-party vendors explicitly need to change their coding to request a compressed SQL Server backup using the VDI or VSS interface. We plan to provide more detailed descriptions of backup compression and results seen in Part II of this white paper.

optimize for ad hoc workload (SQL Server 2008)

This option is to cache a skeleton plan in the statement cache of an ad hoc statement that was executed only one time. At the moment the statement is executed a second time, a full execution plan will be stored in the statement cache. However the option addresses real ad-hoc (non-prepared) statements only.

If you are using SQL Server 2008 in combination with SAP applications, this feature shouldn’t be necessary because both the ABAP engine and the SAP Java engine use prepared statements. Hence the option doesn’t have any impact.

**SAP Settings for SQL Server Lock Manager**

SQL Server introduced new locking mechanisms with the development of SQL Server 7.0. The two goals of the design, called dynamic locking, were to:

- Provide row level locks for most of the operations
- Save locking costs for nonconcurrent operations that would acquire a lot of locks

If a lock is acquired at higher granularity than a row, it can cover more rows while consuming fewer lock resources (each lock structure takes approximately 100 bytes) and the locking overhead. However, this comes at a price of lower concurrency. So for example, if you select all the rows of a table, you will prevent locking all rows individually by acquiring a lock at the table level. However, this blocks concurrent
modifying transactions. Similarly, if every individual row is locked, you will get higher concurrency but add the overhead of acquiring/releasing locks on each row and demand the additional locking resources. Most of the SELECT statements of SAP do not cause a lock on a row because the reads are executed as so called ‘dirty reads’ (as explained here). Most of the other data manipulating queries (DML) submitted by SAP products end up with normal row locks because most modifications are done on a per row basis.

As a first step in the dynamic locking algorithms, SQL Server determines a locking granularity based on data evaluated and estimated through query compilation like cardinality estimations or estimations on rows being processed. The locking granularities can be row, page, or table locks. If a lock granularity of page or table lock is desired for starting the execution of the query, you might not get that lock granularity, because other conflicting locks would block the execution. In such cases the initially sought lock granularity will actually get lowered. While locking granularity is chosen at the start of query execution, there is also a possibility that lock granularity might change during execution. SQL Server may choose to escalate the lock at a coarser level of granularity depending on the number of locks required and the availability of memory at run time. SQL Server only supports escalating the locks during execution to the table level. In SQL Server 2008 a change was implemented for partitioned tables. In the case of a partitioned table, SQL Server 2008 never escalates to a table lock, but always escalates to a partition lock, even if it means locking all partitions. This is the only difference in dynamic locking between SQL Server 2005 and SQL Server 2008. If, when attempting to escalate the lock granularity during execution, conflicting locks are encountered, the attempt fails and the query continues to execute in row lock granularity. SQL Server 2008 never dynamically escalates row locks to page locks during execution of a query.

Triggering Lock Escalation:
A lock escalation is triggered when any of the following conditions is true:

- The number of locks held (as opposed to the number of locks acquired) by a statement on an index or a heap within a statement exceeds the threshold (approximately 5000 by default). These locks include intent locks as well. Note that lock escalation will not trigger if:
  - The transaction acquires 2,500 locks each on two index/heap(s) in a single statement.
  - The transaction acquires 2,500 locks on the nonclustered index and 2,500 locks on the corresponding base table in a single statement.
  - The same heap/index is referenced more than one time in a statement; the locks on each instance of those are counted separately. So for example, in the case of a self-join on a table t1, if each instance has 3,000 locks within the statement, lock escalation is not triggered.
  - The memory taken by lock resources is greater than 40 percent of the non-AWE (32-bit) or regular (64-bit) enabled memory when the locks configuration option is set to 0, the default value. In this case, the lock memory is allocated dynamically as needed.

For example, SQL Server might initially request table locks in the following query:
UPDATE TABLE1 set PRICE=PRICE*1.1;

If table1 contains approximately one million rows and certain other schema conditions are met, SQL Server might start executing the query with a table lock in place. If any other existing lock would conflict with the table lock, the next cheapest solution might be to acquire page locks. If that again could be hindered by conflicting locks, the query would start with row locks. However, after 5,000 locks are acquired on one heap or index, SQL Server would check again for escalating to a table lock.

If you use the SAP deployed stored procedure sap_lock in this scenario, the first few columns of the result set contain the following entries for a table lock in the first few column of the result set:

<table>
<thead>
<tr>
<th>Tablename</th>
<th>spid</th>
<th>dbid</th>
<th>IndId</th>
<th>Type</th>
<th>Resource</th>
<th>Mode</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE1</td>
<td>100</td>
<td>5</td>
<td>0</td>
<td>TAB</td>
<td></td>
<td>X</td>
<td>GRANT</td>
</tr>
</tbody>
</table>

If the same UPDATE statement is executed under page locks, the output of the first few columns of the sap_lock output would look something like this:

<table>
<thead>
<tr>
<th>Tablename</th>
<th>spid</th>
<th>dbid</th>
<th>IndId</th>
<th>Type</th>
<th>Resource</th>
<th>Mode</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE1</td>
<td>51</td>
<td>7</td>
<td>0</td>
<td>TAB</td>
<td></td>
<td>IX</td>
<td>GRANT</td>
</tr>
<tr>
<td>TABLE1</td>
<td>51</td>
<td>7</td>
<td>1</td>
<td>PAG</td>
<td>1:379004</td>
<td>X</td>
<td>GRANT</td>
</tr>
<tr>
<td>TABLE1</td>
<td>51</td>
<td>7</td>
<td>1</td>
<td>PAG</td>
<td>4:24363</td>
<td>X</td>
<td>GRANT</td>
</tr>
<tr>
<td>TABLE1</td>
<td>51</td>
<td>7</td>
<td>1</td>
<td>PAG</td>
<td>4:24430</td>
<td>X</td>
<td>GRANT</td>
</tr>
<tr>
<td>TABLE1</td>
<td>51</td>
<td>7</td>
<td>1</td>
<td>PAG</td>
<td>1:24685</td>
<td>X</td>
<td>GRANT</td>
</tr>
</tbody>
</table>

In this case, the table lock is an intent exclusive (IX) lock, which indicates that lower granular exclusive (X) locks are held within the table by the same session.

The output of sap_lock where the UPDATE statement is executed using row locks would look like this:

<table>
<thead>
<tr>
<th>Tablename</th>
<th>spid</th>
<th>dbid</th>
<th>IndId</th>
<th>Type</th>
<th>Resource</th>
<th>Mode</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE1</td>
<td>51</td>
<td>7</td>
<td>0</td>
<td>TAB</td>
<td></td>
<td>IX</td>
<td>GRANT</td>
</tr>
<tr>
<td>TABLE1</td>
<td>51</td>
<td>7</td>
<td>1</td>
<td>PAG</td>
<td>1:379004</td>
<td>IX</td>
<td>GRANT</td>
</tr>
<tr>
<td>TADIR</td>
<td>51</td>
<td>7</td>
<td>1</td>
<td>KEY</td>
<td>(190147febc17)</td>
<td>X</td>
<td>GRANT</td>
</tr>
<tr>
<td>TADIR</td>
<td>51</td>
<td>7</td>
<td>1</td>
<td>KEY</td>
<td>(450197cb2748)</td>
<td>X</td>
<td>GRANT</td>
</tr>
<tr>
<td>TADIR</td>
<td>51</td>
<td>7</td>
<td>1</td>
<td>KEY</td>
<td>(9902ae588d11)</td>
<td>X</td>
<td>GRANT</td>
</tr>
</tbody>
</table>

Again the intent exclusive (IX) locks on the table and page indicate the presence of lower granular exclusive (X) locks on the table and on the specific page, which can only be row locks.

Another typical example where SQL Server might try to use a higher granularity level could be:

```
UPDATE TABLE1 with (UPDLOCK) set PRICE=PRICE*1.1
WHERE STATUS = 'NEW';
```
In this query, the lock granularity SQL Server would like to start with is dependent on whether an index exists on the column STATUS and how big the expected result set would be. Like before, if table TABLE1 has one million rows and if there is no index on the column STATUS, the whole table would need to be scanned in order to locate the rows fitting the predicate of the WHERE clause. Therefore SQL Server would likely try to start with a different granularity than row locking. Dependent on competing conflicting locks, the granularity is decided then.

In both cases an index on the column STATUS would help to increase the chances of row locks, under the assumption that the majority of values in column STATUS do have different values than 'NEW'.

How to Monitor Table Lock Escalation

You can use SQL Server startup trace flag 611 to detect table lock escalations. Each lock escalation is recorded in the SQL Server error log. In addition to the occurrence and the time stamp of the occurrence, the SQL statement handle is reported. If you take that handle, and executing the following statement returns the text of the query, revealing the affected table(s):

```
SELECT text FROM sys.dm_exec_sql_text(<handle>)
```

For example, this SELECT statement contains the query handle:

```
SELECT text FROM sys.dm_exec_sql_text(0x03000500275CDB1E2E0BB000959700001000000000000000)
```

How to Restrict Dynamic Locking in SQL Server 2005

The locking granularity chosen to start the query can be controlled and overwritten explicitly by using the `sp_indexoption` stored procedure on affected tables and indexes. This is done during the installation of SAP products on several tables in the SAP database schema. For example, typical queuing tables might include VBHDR, VBMOD, and VBDATA for storing update requests, and ARFCRDATA, ARFCSDATA, TRFCQUEUE for handling RFC processing; all have their dynamic locking behavior restricted. For more information, see OSS Note 327494. Although this note is for SQL Server 2000, the section describing dynamic locking and how SAP deals with it applies to SQL Server 2005 as well. The command to check the correct lock escalation settings on those tables would be:

```
SELECT INDEXPROPERTY(OBJECT_ID('<table name>'), '<index_name>', 'IsPageLockDisallowed')
```

There are two ways to suppress lock escalations to table locks:

- **Trace flag 1211**: This trace flag disables lock escalation at the current threshold (5000) on a per index/heap per statement basis. When this trace flag is in effect, the locks are never escalated. It also instructs SQL Server to ignore the memory acquired by the lock manager up to a maximum statically allocated lock memory or
60 percent of non-AWE (32-bit) or regular (64-bit) of the dynamically allocated memory. When this threshold is reached, an out-of-lock-memory error is being generated. This can potentially be damaging because an application can exhaust SQL Server memory by acquiring large numbers of locks. This can even stall the server or degrade its performance to an unacceptable level. For these reasons, strong caution must be exercised when using this trace flag.

- **Trace flag 1224**: This trace flag is similar to trace flag 1211 with one key difference. It enables lock escalation when the lock manager acquires 40 percent of the statically allocated memory or 40 percent of non-AWE (32-bit) or regular (64-bit) dynamically allocated memory. Additionally, if this memory cannot be allocated due to other components taking up more memory, the lock escalation can be triggered earlier. SQL Server generates an out-of-memory error if memory allocated to lock manager exceeds the statically allocated memory or 60 percent of non-AWE (32-bit) or regular (64-bit) memory for dynamic allocation.

If both trace flags (1211 and 1224) are set at the same time, the trace flag 1211 takes precedence. You can use DBCC TRACESTATUS (-1) to find the status of all trace flags enabled in SQL Server.

### How to Restrict Dynamic Locking in SQL Server 2008

There is a change in SQL Server 2008 that provides the ability to disable lock escalation on a specific table. In that situation any lock escalation during running a statement is suppressed and the invoking statement uses row locks from the beginning. No checks are executed to see whether any statement would execute better using any other lock granularity. The command to do so would look like this:

```sql
ALTER TABLE '<schema name>'.'<table name>'
SET (LOCK_ESCALATION = DISABLE)
```

Are there chances of hitting situations where a page or table lock will block other concurrent accesses during daily production with SAP products? With SAP standard delivered coding, the only situation observed so far was with the table DDLOG deleting massive amounts of rows in one DELETE statement. Customer written ABAP programs changing massive amounts of rows and using relatively unrestricted predicates has been an occasional source of such situations. Other customer written programs where the predicates were insufficiently supported by the existing indexing architecture have also led to such situations. Due to the fact that SAP reads ‘dirty’ most of the time, the number of locks held by SAP applications usually is very low, typically in the hundreds and low thousands. Also, given the fact that the buffer sizes for SQL Server running under SAP workloads are going into the gigabytes, we have not observed an escalation that is caused by lock structures occupying 40 percent of the buffer pool.

There might be situations where it could be interesting in SAP BW to investigate blocking locks caused be lock escalation. You can use the following query to return these locks:
SELECT * FROM sys.dm_db_index_operational_stats(db_id(), NULL, NULL, NULL)
WHERE index_lock_promotion_attempt_count >0 AND
index_lock_promotion_count >0

Special Configurations for SQL Server 64-Bit Enterprise Editions

In order to avoid SQL Server buffer pool pages being paged out to the Windows page file(s), enable the **Lock pages in memory** policy for the user account that starts SQL Server services. This needs to be done in the local security settings, as demonstrated in the following graphic. For more information about how to do this, see SQL Server Books Online.
Figure 16: Setting **Lock pages in memory**

### Using Virus Scanners on the Database Server

In most cases, customers run virus scanners on the database servers. However, best practice is to exclude all the partitions with data and log files of the SAP database and `tempdb` from real time monitoring or from scanning jobs. Because the other system databases, like `master` and `msdb`, are not used as frequently when SAP applications are run, we have customers who include those in the real-time virus monitors or scanning jobs. Other customers exclude those files and restrict scanning to the drive the operating system is installed on. The database server should be secured by other means so that it is not vulnerable to malware. This includes both strict access controls and operational discipline such as never using an Internet browser to download data or executables off the Internet. This practice has always been practiced in the UNIX world, but it also applies to Windows.

### SQL Server on NUMA Architectures

#### What Is NUMA Architecture?

NUMA stands for non-uniform memory access. The term *NUMA* describes server architectures that are built in a modular manner, where units of processor and memory are combined over buses or switch architectures. The nature of such a modular architecture (and its name) arises from whether memory gets accessed within the same module or on a different one. Usually those modules are called nodes. A simplified demonstration of such a configuration could look like this.
Figure 17: Simple demonstration of a NUMA node

Figure 17 shows two nodes of a single server, where each node has four CPUs with local memory on the same node. It becomes obvious that there would be higher latency if P1 on Node A needs to access memory of node B than if it needs to access memory on the same node. The usual latency impact is in the range of 3 to 5 times slower when accessing memory on the remote node.

What hardware is built up in this manner? Basically, all proprietary hardware architectures that use more than four sockets/processors follow this architecture principal. To most people’s surprise, all AMD Opteron based servers are based on this type of architecture, even two-socket servers. This is because AMD Opterons have the memory controller in the processor silicon, and they also have some memory banks connected to each of the sockets/processors. So effectively, one socket/processor actually implies the existence of single NUMA node on its own.

In contrast, the standard Intel architecture up to four sockets/processors has a central memory controller that is connected to the processors via the Front Side Bus. In this case, the hardware looks like a uniform architecture where every processor or CPU thread has the same latency to memory. In the current case with a four-socket six-core Intel Xeon, 24 CPU threads might form one node with a uniform latency to memory. For more information, see the following Wikipedia article: [http://en.wikipedia.org/wiki/Non-Uniform_Memory_Access](http://en.wikipedia.org/wiki/Non-Uniform_Memory_Access). The new Intel Nehalem architecture will move away from the Front Side Bus architecture, and it will also have the memory controller with the processor silicon. Hence new servers that the Intel Nehalem architecture will show up as NUMA architecture as well.

How Do Windows and SQL Server Deal with NUMA Architectures?

With Windows Server 2003, new APIs were defined that allowed hardware vendors to describe the hardware’s NUMA architecture to the Windows operating system. These APIs are summarized in the SRAT interface. On the application side, Windows offers
APIs for applications to retrieve this information in order to optimize the memory allocation and request scheduling.

SQL Server is an application that needs to optimize for NUMA architecture to obtain peak performance. These architectures have meanwhile become mainstream. In SQL Server 2000, the changes that could be made were not as drastic because only service packs could be used to deploy minimal changes: A complete redesign of the internal SQL Server memory manager was out of the question. With SQL Server 2005 under development at that time, the opportunity existed to completely redesign SQL Server’s memory management and implement it with all reasonable optimizations for NUMA architectures. At startup, SQL Server uses the Windows APIs to check the underlying hardware architecture. If the architecture is a NUMA architecture, SQL Server attempts to allocate memory evenly through the different nodes; it also keeps structures local to nodes and even partitions certain memory pools NUMA nodes. The aim behind this kind of node awareness is to minimize memory accesses across node boundaries. As some cases of high-end SAP customers proved, SQL Server 2005 was successful in this. In a particular case where a customer is running a 10-terabyte SAP ERP system on 32-socket NUMA architecture, the database response time after an upgrade from SQL Server 2000 to SQL Server 2005 was cut in half.

Check Whether SQL Server Runs on NUMA

In the past there always were cases where the server hardware didn’t tell Windows that it was running on NUMA architecture. For some hardware, the BIOS of the server needs to be set accordingly to provide Windows the correct information about the NUMA architecture of the server. If Windows has the information about the hardware NUMA configuration, the SQL Server error log should contain entries like the following:

Multinode configuration: node 0: CPU mask: 0x000000000000000c Active CPU mask: 0x000000000000000c.
Multinode configuration: node 1: CPU mask: 0x0000000000000003 Active CPU mask: 0x0000000000000003.
Multinode configuration: node 2: CPU mask: 0x0000000000000030 Active CPU mask: 0x0000000000000030.
Multinode configuration: node 3: CPU mask: 0x00000000000000c0 Active CPU mask: 0x00000000000000c0.

In this case, SQL Server starts up on 4 NUMA nodes (node 0 to node 3) with each node having 2 CPUs. In this actual case, SQL Server started up on a 4-socket AMD Opteron Dual-Core driven system. It is noteworthy that SQL Server put its first node on CPUs 2 and 3 and not on CPUs 0 and 1. The reason is that Windows already has structures that are allocated on node 0 only. SQL Server also has some structures that are allocated on SQL Server’s node 0 only. In order not to overload physical node 0, SQL Server places its logical node 0 on physical node 1.

In SQL Server 2008, two new DMVs were introduced that offer information on SQL Server’s NUMA configuration:

- sys.dm_os_memory_nodes
– Shows whether the allocation on different NUMA nodes is around the same. This DMV should show one row per NUMA node.

- `sys.dm_os_nodes`
  - Shows the NUMA nodes, the CPU affinity of the nodes, and the number of SQL schedulers within a node.

Other DMVs that show interesting information about number of CPUs and memory are:

- `sys.dm_os_info` (introduced in SQL Server 2005)
- `sys.dm_os_memory` (introduced in SQL Server 2005)

If you are sure that your hardware is a NUMA architecture, but SQL Server 2008 or SQL Server 2005 seems not to honor this, check with your hardware vendor to see whether the BIOS settings are correct. This was the issue in nearly every case we encountered in the past when SQL Server 2008 or SQL Server 2005 was not starting up in NUMA configuration.

### SQL Server Timers

Ever faster execution of queries due to faster processor technologies demanded the introduction of finer granular time measurements within SQL Server 2005. Instead of using the normal synchronized timer with a granularity of around 16 milliseconds, SQL Server 2005 introduced a high resolution CPU counter to deliver timing in microsecond granularity. Real Time Stamp Counter (RDTSC) instructions are used to acquire the 64-bit CPU tick count. This value is divided by the CPU frequency to convert the value to milliseconds. It all would work fine if there were not new features in processor technology that can lower the frequency of an individual CPU or tick counts on CPUs that could drift apart from each other. Because SQL Server worker threads can be scheduled on different CPUs by the operating system, some unwelcome side effects can appear. One of the most prominent before SQL Server 2005 SP2 resulted in this message:

```plaintext
2007-08-02 10:03:08.99 spid6s SQL Server has encountered 1 occurrence(s) of I/O requests taking longer than 15 seconds to complete on file [i:\d12\mssql\data\MST_data12.NDF] in database [MST] (6). The OS file handle is 0x0000000000000954. The offset of the latest long I/O is: 0x0000027faa6000.
```

Quite a few customers invested efforts with the storage vendors to figure out why their storage was slowing down. In fact, the scheduler that ran the SQL Server I/O thread was scheduled onto another CPU, and the tick counts of the CPUs were significantly different. To provide more helpful information for troubleshooting, SQL Server 2005 SP2 added more messages to the error log that indicated either drifts or nonsynchronized CPU behavior. The error log messages look like this:

```plaintext
Aug 15 2007 12:25AM CPU time stamp frequency has changed from 1346594 to 1197665 ticks per millisecond. The new frequency will be used.
Aug 15 2007 12:21AM The time stamp counter of CPU on scheduler id 7 is not synchronized with other CPUs.
```
If you receive one of these two messages following a message that notifies you of I/Os taking longer than 15 seconds, probabilities are high that there is no I/O problem, but the CPU tick counts drifted apart and the SQL Server I/O threads moved between different CPUs.

Several options can help minimize problems resulting from CPUs suddenly being out of synchronization.

**Figure 18: Set Power Schema to Always On**

One option is to affinitize the SQL Server schedulers by using the *affinity mask* and *affinity64 mask* options. These options can help you avoid worker threads ending up on different CPUs within one request. However, this is something that can interfere with the Windows operating system scheduling. And especially in situations where SAP applications run on the same server, this is something that we don’t recommend since it interferes with the Windows scheduling.

Another possibility is to disable power configurations that reduce the frequencies of CPUs. To do this, start powercfg.cpl by using the *Run* option. Then, in the *Power Options* dialog box, under *Power Schemes*, select *Always On*.

Whether the action of disabling frequency reductions (and thus consuming more power) is a wise choice in times when Green-IT is a major debate is up to the individual organization to consider.

A third possibility is to check with hardware vendors for BIOS patches that correct synchronizations between the CPUs. For example, IBM and AMD have released fixes for their X-series models and Opteron servers, respectively.

A fourth possibility is to ignore the messages and see the fact of CPU ticks drifting apart as simple annoyance that can sometimes give wrong timing indications or spill extraneous messages into the error log. You can choose to view these annoyances as secondary, as long as the performance of the system meets your organization’s needs.
In this case, you can even choose to switch off the new messages SQL Server 2005 SP2 introduced. To do this, set trace flag 8033 at startup.

In SQL Server 2008, the source of the timings changed again, making CPU ticks drifting apart less common. In most cases, SQL Server 2008 returned to the old timers and changed those so that the granularity became 1 millisecond. For the most part, the problems described in this section are limited SQL Server 2005 and SAP, as described later in the Windows section.

New SQL Server 2008 Features for SAP

SQL Server 2008 Database Compression

Types of SQL Server 2008 Database Compression

SQL Server 2008 introduced three basic features to become more space efficient when storing the data:

- Row compression
- Dictionary compression (part of page compression)
- Backup compression

All three use different methods and address different needs. This section discusses the first two types and how to implement them on an SAP database. Backup compression will be discussed in Part II of this white paper.

In general, both row compression and page compression require setting an option when creating or rebuilding a table or index. If you are working with SAP applications, however, the SAP application and installation are responsible for setting these options.

SQL Server 2008 Row Compression

The advantages of SQL Server 2008 row compression is solely focused on the representation of any data type storing numbers. SAP applications mostly assign a default value of 0 or 0.0 to these columns while defining the SAP schema in the database. Earlier versions of SQL Server stored all numeric data in a fixed length format (with the exception of vardecimal in SQL Server 2005 SP2). For example, an integer was represented by 4 bytes in the database files. Decimal columns consumed a fixed amount of space in the data files based on the exact definition of the decimal data type. SQL Server assigned the same 4 bytes whether the value associated with a column was the default value of 0 (or 0.0) or the annual state budget of your home country.

Row compression changes this significantly. The space used to store numeric data is now associated with the value rather than the data type. Optimization is done for the
specific default values of 0 and 0.0 so that only 4 bits of space is required. Other values will not require the full storage space of those data types either. The term *compression* is a bit misleading, because there are no algorithms involved as you might expect based on the way some file compression tools work. The term *row compression* describes a new, more efficient row storage format. As data rows are inserted, they are immediately stored in the new row format, after the table has been brought to the new compressed row format. Besides the change in storing all data types used to store numeric data, there are other changes to ROW Level compression. For more details please read this [article](#).

Depending on the SAP product and the nature of the data, net savings on storage can be expected to be between 15 percent and 40 percent. Net savings are defined as savings based on row compression only and not on the effect of data having been reorganized during the move into a row-compressed form. In observations of SAP customers using SQL Server 2008 row compression, no negative performance impact or significant increase in resource consumption was observed. In the SAP BW space, faster database response times and shorter query times were even observed, due to the reduction of I/O operations.

Row compression might sound a lot like *vardecimal*, which was introduced in SQL Server 2005 SP2. You can think of row compression as the consequential extension of *vardecimal* beyond the *decimal* data type, with other improvements that make it a better choice than *vardecimal*. *vardecimal* was restricted to the *decimal* data type only. The way defaults were handled was also improved with the row compression algorithms. Hence, if you are using SQL Server 2008, you should not use *vardecimal*, and you should move tables in the *vardecimal* format to row compression.

### SQL Server 2008 Dictionary Compression

SQL Server 2008 dictionary compression addresses a different data storage issue: duplication. Many organizations sort databases according to business keys, containing many rows store the same or similar values in the same columns. For example, in SAP applications, the MANDT/CLIENT column stores the same value for the entire data set, and the company code column (BUKRS) is often the same at least within one data page. Many data columns start with ‘2008’ as a prefix value. Dictionary compression, which is one of the steps of page compression, evaluates similarities in a page and in some cases replaces them with tokens. Dictionary compression looks for the following similarities within one data page:

- Columns that contain the same prefix; for example, all date values that start with ‘200811’ and differ only in the last digits
- Columns that contain duplicate values; for example, the MANDT column or BUKRS column

SQL Server inserts new rows into a table that has been has had dictionary compression applied without applying any compression to the new rows. When the page fills, SQL Server checks to see whether there would be enough space for more rows on the page if duplicate prefixes and duplicate values were stored once instead of every time they

*Microsoft Corporation ©2009*
occur. If this is the case, SQL Server 2008 replaces duplicate prefixes with a 1-byte token, and then it replaces entire duplicate values with a 1-byte token. The actual values are stored on the same page, but they are stored only once. If more than 128 values need to be replaced, the token becomes a 2-byte token. After this compression is complete, row insertion continues. This method has some advantages. Obviously one doesn’t perform the algorithms with every inserted row and thus saves some resources. For more information about page and dictionary compression, see Page Compression Implementation (http://technet.microsoft.com/en-us/library/cc280464.aspx) in SQL Server 2008 Books Online.

The net savings in using dictionary compression with SAP databases could be in the range of 40 percent to 70 percent. Again it is highly dependent on the SAP application and the nature of the data. SAP Enterprise Portals with masses of BLOB/CLOB columns can show lower net savings. And in contrast to row compression, dictionary compression is expected to result in a 25-50 percent increase of CPU resources. In I/O bound SAP BW scenarios, however, faster report and query run times are still observed due to the massive reduction in I/O.

**SAP Supporting SQL Server 2008 Database Compression**

SAP supports SQL Server 2008 database compression methods on the data page layer and clustered index B-Tree level for all SAP products. Support for nonclustered indexes, however, is not yet implemented in SAP Basis and is a work item for the next nine months. This support policy by SAP is different from the SAP support of vardecimal, which has been restricted to SAP BW only.

SAP releases of NetWeaver 7.0 and Business Suite 2008 create the database schema with tables in the new SQL Server 2008 row format (row compression) only. SAP also supports the move to SQL Server 2008 database compression features for existing databases. For more information, see OSS Note #991014.

Because SAP is taking care of new installations, the next sections focus on moving an existing database from the old format to row compression or dictionary compression.

**Getting an Existing System to SQL Server 2008 Database Compression**

**Planning for Database Compression**

Despite the fact that usually only a small number of tables contribute to the majority of the volume, the goal is for nearly all tables in the SAP database to be compressed. If the database is consistently compressed, your system can take advantage of new SAP functionality without having to monitor which formerly empty tables are getting filled up by the new functionality. So we are shooting for a one time effort to get the database to a certain level of database compression. Nevertheless you will find quite a few SAP
customer examples documented already where only part of the tables got dictionary compressed. This has to do with the immediate requirements of saving space immediately and with CPU resource constraint on the other side.

Independent of the compression type, the fact remains that every table that gets compressed is rebuilt on the data layer and the clustered index B-Tree (if one exists). Hence additional resources in CPU, disk I/O, and transaction log space might be necessary. The table to be compressed can be rebuilt offline, which means the table is locked for modifications, or online, which means the table can be modified while it is being rebuilt. The table can be rebuilt with as many CPUs as you want to spend.

Customers have varying requirements for the availability of an SAP system. For example, in the easiest case you might have an SAP system with a database of a few hundred gigabytes and a user community that agrees on a downtime of up to one day. During this downtime, the compression against all tables can be performed offline using all of the resources of the database server. However, in business-critical situations, and with larger customers with SAP ERP systems with 10 terabytes or more and user communities that cannot accept downtime beyond the regular system maintenance, it is much harder to compress the whole database or even single large tables. Very often such systems are in use continuously, and compression must be performed online, with limited resources in order not to interfere with the regular business.

Another issue to be aware of is the usage of transaction log space of SQL Server. If you are able to perform the compression offline, choose the bulk-logged recovery model, which minimizes the volume in the SQL Server transaction log. However, this doesn’t reduce the volume in your transaction log backup, because all the changed pages are included in the transaction log backup, which can impact log shipping scenarios. Using Storage Replication of a SAN/NAS storage device an impact on the amount of data to be transferred also can be significant.

If you must perform the compression online, you might not be able to use the bulk-logged recovery model. For example, a point-in-time recovery might not be possible with bulk-logged recovery. Another reason might be that you use database mirroring, which supports only the full recovery model. So if you use the full recovery model, ensure that you have enough transaction log space. It needs to contain enough space for the largest table plus the business transactions that are executed while the largest table is rebuilt. It might be necessary to add additional storage space for the times compression is running. In such cases, you should create a second log file on the newly attached storage. This second transaction log file can be dropped after the activity is complete.

Two Tools for Database Compression

This article describes two different tools for database compression. The easiest one to use is the tool SAP provides as an attachment to OSS Note #991014. It imports an additional report that can be used to assemble the list of tables in a specific schema of the SAP database. It enables you to compress a selected list of tables or all tables. It enables compressing online and offline. If you have a database of a few hundred-gigabytes database and you can perform compression during downtime, it easy to use this tool; just select all the tables and then choose offline compression using all CPU resources on the table. Depending on the underlying hardware one easily should be
able to compress around 500GB-1TB database volume in an offline fashion within 24h using all the database server’s resources.

However, some scenarios are too complex for this first tool to be used effectively. For example, you may need to compress an SAP database with a few terabytes online, within selected time frames of low activity, like weekends. This scenario requires more testing, more planning, and a more complex tool designed for expert use. Microsoft provides a tool that you can use in these specialized scenarios. To download the Transact-SQL code for that tool, see the Running SAP Applications on SQL Server blog (http://blogs.msdn.com/saponsqlserver/archive/2008/11/24/procedure-used-for-applying-database-compression-to-microsoft-sap-erp-system.aspx).

SAP Deployed Tool

Let’s first go through the SAP provided tool. In OSS Note #991014, two transports are attached. Import these into the system in which you want to perform compression. After the transports succeed, log into the SAP system and lock the system for all other users. Call transaction SE38 and execute the report RSDD_MSSQL_CUBEANALYZE. Despite the name, it is a report for not only SAP BW, but all of SAP Basis, and the report is available in SAP Basis in general.

After the report starts, the following screen appears.

Figure 19: Entry screen for SAP Compression tool

Simply click Row/Decimal Compression.

The next screen looks like the following two figures.
The list of tables in the lower part is empty when the report is called for the first time.
In order to list all the tables the following steps need to be done:
1. Next to **n-largest Tables**, enter a number that is larger than the number of tables in your SAP database.

2. Click **Start Checks**.

3. In the popup that appears, click **Start immed**. A batch job starts, going through all the tables, checking their sizes, and displaying them in the list.

4. After the batch job finishes, click **Refresh Output** in order to get the new list of tables. The list of tables is sorted by size with the largest table listed first.

To perform a simple offline compression of all tables, perform the following steps:

1. Select all the tables.

2. Ensure that **Create ONLINE Index** is set to **inactv**.

3. Next to **Index MAXDOP**, enter the number of available CPU resources.

4. Click **Compress Selected Tables**. A message appears, asking you to either schedule a batch job or schedule immediately.

The tasks themselves are executed as a batch job in the background. The tool sorts the tables based on the size. It compresses the smallest tables first, creating space to compress the larger tables to avoid extending the database. For decompression, the order is reversed, with the largest table decompressed first. In all the cases where compressions were done, we never encountered a situation that required enlarging the database prior to the compression run.

You can also use the online method with the tool. There are tables in SAP databases, however, that can’t be transformed online due to restrictions in SQL Server. These are mainly tables containing **varbinary(max)** columns that can’t be compressed online. Usually there are a few hundred of these tables in SAP products.
You would need to make a second pass through the tool, selecting the noncompressed tables only and then compressing those tables offline. However, these tables are not usually too large, and you can compress them offline in a relatively short time without too much disruption.

Page Compression with the SAP Tool

As you can see in the screenshots shown in the previous section, there was no option to select page compression, which contains dictionary compression. If you want to use page compression, you must enable expert mode, as shown in the following figure.

Figure 23: Enable Expert Mode to get to page compression option

After expert mode is enabled, the screen looks like this.

Figure 24: Choose the compression method
More options appear:

- **Use Decimal Compression**: The `vardecimal` option introduced with SQL Server 2005 SP2. As discussed earlier, you should never use this option in with SQL Server 2008. This feature was only supported by SAP with SAP BW.

- **Use Row compression**.

- **Use Page compression**. Selecting this option and following the steps described earlier for row compression will compress the tables with dictionary compression.

- **Index Compression**: The option to compress all nonclustered indexes. At this point in time, you should not use this option, because there are potential problems during SAP Enhance Package Upgrades or Release Upgrades. It is expected that these problems will be resolved within the next nine months.

**Tables to Exclude from Dictionary Compression**

At the beginning the goal was stated to have all tables compressed using the same method. However, looking at how dictionary compression works and how SAP applications use some of the tables in their schema, it becomes obvious that some resources would be wasted compressing certain tables. The tables in discussion are the SAP tables used for the purpose of queuing data for a short amount of time. For example, in the tables VBHDR, VBMOD and VBDATA, data inserted could be deleted one second later. Hence any effort compressing would only waste CPU resources. The following tables should be excluded from dictionary compression:

- VBDATA
- VBHDR
- VBMOD
- ARFCSTATE
- ARFCSDATA
This list represents an initial recommendation. Dependent on the specific case there might be some more tables which are used heavily as short term queuing tables and hence shouldn’t be dictionary/page compressed.

A Second Tool to Use for Moving to Compression

The second tool is a stored procedure that was written and tested for those rare situations where database compression needed to be applied under extremely restrictive conditions in terms of time slices, resources, database volume, and absolute online conditions. The tool was used to move Microsoft’s SAP ERP system of 6.5 terabytes to row compression on all tables. We will refer to this migration as we discuss the tool. As mentioned earlier, the stored procedure can be found as an attachment of this blog entry.

The stored procedure does the following:

- Compresses or decompresses to or from row compression and to or from dictionary compression.
- Performs compression offline or online.
- Specifies the CPU resources.
- Bundles packages of tables to be compressed in batch_input tables.
- Keeps track of time spent and space reserved before and after the compression.
- Can compress or decompress data (clustered index or heap), data and all indexes, or nonclustered indexes only. By default, only data and clustered indexes are compressed or decompressed, which is highly recommended by SAP.
- If online compression cannot be performed for some tables, writes those tables into batch_input tables or, if a table is smaller than an adjustable threshold, compresses it offline immediately.
- Can be configured to run in verbose_only mode, generating the commands for compression or decompression of each table
- Can compress or decompress the tables within one schema only or all tables of the database, if the user executing it has sufficient permissions.

Example of Using the Stored Procedure to Compress

The SAP ERP system in which we used this tool for Microsoft displayed the following conditions, which are typical of large, complex SAP deployments:
The SAP ERP production system had about 6.5 terabytes of data, which needed to be compressed completely into row compression.

Compression activities could run only between 6 P.M. Friday evening and noon on Sunday.

Compression of data could be done online only, with only 2-4 CPU threads out of 16 available.

No additional downtime could be taken beyond the regular 2 hours per month.

Some weekends needed to be spared due to quarter-end reporting.

There were requests not to compress certain tables on specific weekends.

The largest table was 700 GB.

Because Microsoft uses database mirroring, full recovery model was a given. The recovery model couldn’t be changed to reduce the transaction log volume created.

The data recovery site needed frequent, regular transaction log backups.

Progress needed to be monitored, because log shipping was used as well and we needed to know how far the copies of the transaction log backups to the secondary fell back. (Transaction log backups were executed every minute.)

Steps Required to Get to the Compression Phase

Before you can perform the actual compression of a production SAP ERP system, you may need to follow additional steps:

- Add an additional SAN partition to the database server of the test system on the principal, mirror side, and log shipping destination in case the full recovery model is required (as it is with database mirroring).

- Add a second transaction log file on the new partition on the principal side and grow it to a size of a few hundred gigabytes in order to have enough space in the transaction log. The new log file is automatically created on the mirror and on the log shipping destination. For the purpose of adding the additional file, switch synchronous database mirroring to asynchronous. After the extension of the transaction log, switch database mirroring back into synchronous mode again.

- Run the `sp_use_db_compression` stored procedure out of the SAP ERP schema under the conditions that are close to production on the test database server with this call:

  ```sql
  sp_use_db_compression 'ROW', @online='ON', @maxdop=2
  ```

  This call leverages two CPU threads on the server. The stored procedure opens a cursor over all tables in the specific database schema, joins data of those tables against some system tables, and then works through the list of tables. The procedure has a set of tables that are excluded (tables starting with ‘sap’ or ‘sp’) and are not compressed.

  The activity is reported in the table `sp_use_db_compression_table`. In this table, run times and data sizes before and after the compression are reported (see more in the monitoring section). While the process is running, operations on the test system continue, including stress tests and other tests, which should continue in order to simulate a production environment. Depending on the volume of the database, the resources used, and the workload, this first run can take several days. The table `sp_use_db_compression_batch_input` one contains entries like this.
These tables reported with the status TO_BE_DONE are some of the larger tables with `varbinary(max)` columns. As mentioned earlier, there are restrictions in SQL Server that prevent tables with `varbinary` columns from being rebuilt online. While the online compression of tables is done, these tables are skipped, and a row table is entered for each table into the `sp_use_db_compression_batch_input` table with the Online column set to OFF.

During downtime, run the stored procedure again with the following syntax:

```
sp_use_db_compression 'ROW', @maxdop=0, @batch_input=1
```

If you set the `@batch_input` parameter set to 1, the stored procedure works only on the set of tables listed in the table `sp_use_db_compression_batch_input`, and only if the Status column for the table is set to TO_BE_DONE.

At the end of a run through a test or sandbox system with data that simulates a production environment, you should have a pretty comprehensive idea of the run time, even with more strenuous conditions than you might expect in production. Based on time periods available for compressing the production system and the time it took to compress the test system, decisions about time periods and scheduling can be made. The exact time to compress for every single table is available in the table `sp_use_db_compression_table`. Using this data, you can create schedules for the longest-running table compressions. The table `sp_use_db_compression_table` contains a list of compressed tables. You can use the following SELECT statement to return the list in reverse order of compression:

```
SELECT * FROM sp_use_db_compression_table ORDER BY DATE DESC
```

The following statement returns the sum of the time taken to compress the whole test system in milliseconds:

```
SELECT SUM(Compression_Time) FROM sp_use_db_compression_table
```

**Building the Packages for the Different Weekends**

After agreeing on specific time periods that can be used to run online compression, the next step is to build the packages of tables that should be compressed in each time period. The most important rule is: The available free space in the database has to be at least the size of the table to be compressed. The easiest way to achieve this is to start with the smallest tables and work your way up to the largest table. Why is this? Tables in the database are typically fragmented and are rebuilt during the compression. The released fragmentation space plus the space saved by the compression will increase the available free space.
In the case of the Microsoft’s SAP ERP system, the first package of tables for the first 30-hour compression time period contained all but around 70 of the largest tables. The other 5 packages contained fewer and fewer tables up to the last one, which contained only one single table. As an example, the format and content of one of the packages looked like this.

<table>
<thead>
<tr>
<th>Tab_Name</th>
<th>Target_Type</th>
<th>Compression_Type</th>
<th>Online</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPUMAWIP</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
<tr>
<td>VBAP</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
<tr>
<td>HRP1001</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
<tr>
<td>VBPA</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
<tr>
<td>SWFREVLOG</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
<tr>
<td>GLPCA</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
<tr>
<td>PPOIX</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
</tbody>
</table>

The different packages had names like 'sp_use_db_compression_batch_input_week<x>'. Each weekend prior to the compression work, renames to the name sp_use_db_compression_batch_input had to be executed.

It is important to keep the content of table sp_use_db_compression_batch_input after the compression is finished to identify tables that must be compressed offline. The last step in the compression process is to execute offline compression for all remaining tables. All the rows of the different batch_input tables that have a value of TO_BE_DONE in the Status column and with a value of OFF in the Online column need to be combined into the final sp_use_db_compression_batch_input table. Then you can use regularly scheduled downtime to compress those tables to row compression.

### Monitoring While Compression Is Running

There are two possibilities for monitoring while the compression procedure runs. The first one is to issue the following query. Its output is similar to that shown in figure 25:

```sql
SELECT * FROM sp_use_db_compression_batch_input ORDER BY DATE DESC
```
Figure 25: Statistics list on already compressed tables

The first row contains NULL values in columns like **Compression_Time**, **Row_Count**, **AfterReserved**. This is an indication that this table is currently being worked on.

The second possibility applies to the case where **sp_use_db_compression** is executed with '@batch_input=1' set as a parameter, you can use the following statement for monitoring:

```
SELECT * FROM sp_use_db_compression_batch_input
```

The result might look like the following table.

<table>
<thead>
<tr>
<th>Tab_Name</th>
<th>Target_Type</th>
<th>Compression_Type</th>
<th>Online</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPUMAWIP</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>DONE</td>
</tr>
<tr>
<td>VBAP</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>DONE</td>
</tr>
<tr>
<td>HRP1001</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>IN_WORK</td>
</tr>
<tr>
<td>VBPA</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
<tr>
<td>SWFREVLOG</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>DONE</td>
</tr>
<tr>
<td>GLPCA</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
<tr>
<td>PPOIX</td>
<td>DATA</td>
<td>ROW</td>
<td>ON</td>
<td>TO_BE_DONE</td>
</tr>
</tbody>
</table>

A value of **IN_WORK** in the Status column indicates that the table is currently being compressed.

**Experiences with Row Compression So Far**

From a space savings point of view, the Microsoft SAP ERP system experienced savings of 29 percent. This included effects of database reorganization. However, a lot of the content in the major tables was not very fragmented, because of a Unicode Migration that took
place in February 2007, leaving the database in a completely reorganized state. Nevertheless, 29 percent translated into more than 1.76 terabytes of free space released. Given the fact that copies of the production database in the Microsoft/SAP landscape are stored for around 10 times for test, HA and DR purposes the overall saving accumulated to more than 17 terabytes. There was a wide bandwidth of compression efficiency on different tables. Some tables, like ANLC, compressed down to one quarter of their original size. Other tables were cut to half of their original volume. We saw a great effect on one of our customer tables, which was 700 GB with 1.6 billion rows. After row compression was applied, the table was only 280 GB.

This reduction of space also means that new storage investments can be pushed out further, which translates into savings of around 500,000 USD for the current fiscal year.

Other positive side effects were:

- A reduction of I/O. A detectable improvement of the database response time, however, was not immediately observed. Reasons are manifold starting with the fact that the cache hit ratio of this particular SAP ERP system was beyond 99 percent before compression was applied. Another reason was that the database response time of the SAP ERP system on a daily basis varied by a range of 25-30 percent anyway, depending on load and season: such high variability inhibits statistical significance.
- Smaller online backups and differential backups. Thanks to SQL Server 2008 backup compression (which will be described in Part II of this white paper), backups were reduced dramatically before compression was applied. However after row compression was applied, we saw further reductions.
- No additional CPU resource consumption was observed.

Other great customer experiences can be found here.

**Transparent Database Encryption**

SQL Server 2008 introduces a feature called Transparent Data Encryption (TDE). SQL Server 2005 introduced Data Encryption, which could be used to encrypt data of certain columns in tables. However, we received the following feedback from customers and partners, especially from larger ISVs and customers:

- An encryption technology should be transparent to the applications. The emphasis of such a technology would be less on protecting the data from user access, and more on protection from media loss (backup tapes or disks) or media theft. This was an aspect the solution we introduced in SQL Server 2005 only could provide partially.
- Data encryption should cover the entire database.
- Data encryption should be extended to database backups.
Therefore the focus developing of the new encryption improvement was on how to encrypt a whole databases in the most secure and efficient way. Theft of media or copy of just the database files should not result in success in reading the data contained in the database. Our response to these requests and efforts is the Transparent Data Encryption (TDE) feature in SQL Server 2008.

How Does TDE Work?

If TDE is enabled on a database, all pages that are getting written to data files are encrypted in the I/O path together with the calculation of the checksum of a page. TDE works in a similar way for reading data. In the I/O path where the checksum of a page is calculated and compared to the checksum noted in the page header, decryption takes place. The data is not encrypted at all when the data pages are in the SQL Server buffer pool. Therefore all existing indexes can be used, and query performance is not affected by the fact that the data is stored encrypted. The only area where run time of a query could increase is when the data to serve the query needs to be loaded from the data files. In that case, you could see an increase in CPU use or run time. However keep in mind that under normal operational conditions, especially with OLTP type of applications, we are shooting for data cache hit ratios of 98 percent or higher. This means that for every 50 pages, 1 page or less should be read from disk. Ideally, physical I/O is only required on 1 page or less for every 100 pages.

Additionally when enabling TDE on a database, SQL Server is taking an additional step to encrypt all the existing pages in the database. This means that every page that hasn’t been touched by data modification operations, since enabling encryption, is encrypted immediately after enabling encryption on a database.

Encrypting a Database – Steps Demonstrated

There are two ways to encrypt a database. The first one which is to create a key in an Extensible Key Management (EKM) system. For more information, see Understanding Extensible Key Management (http://technet.microsoft.com/en-us/library/bb895340.aspx) in SQL Server Books Online. For SAP databases we would recommend usage of such an EKM system.

As a second option, local security certificates can be used. To do this, follow these steps:

1. Create a master key that applies within the entire SQL Server instance. For example:

```sql
use master;
go
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'strong password';
go
```

This creates the database master key (DMK), which again is used to encrypt certificates to be created for database encryption. The master key itself is protected by the Service Master Key, which is automatically generated when SQL Server is installed. Be aware that the password needs to accommodate the
default Windows password policies if those are enabled on the server you are creating the certificate on.

2. Create a certificate that is also valid for the entire SQL Server. For example:

```sql
use master;
go
CREATE CERTIFICATE MyServerCert WITH SUBJECT = 'My DEK Certificate';
go
```

Because no specific encryption password is used, the DMK is used as encryption password.

3. The third step is to link the master key, the certificate, and the database. This is done by creating an encryption key for the specific database to be encrypted. For example:

```sql
use my_database;
go
CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_128 ENCRYPTION
BY SERVER CERTIFICATE MyServerCert
```

Possible algorithms include AES_128, AES_192, AES_256, and TRIPLE_DES_3KEY. For more information about those different algorithms, see the Wikipedia articles Advanced Encryption Standard (http://en.wikipedia.org/wiki/Advanced_Encryption_Standard) and Triple DES (http://en.wikipedia.org/wiki/TripleDES).

Please see later results and comments about the choice of encryption algorithms in specific sections and measurements.

4. The next step is to enable encryption on the database with this command:

```sql
use mydatabase;
go
ALTER DATABASE mydatabase SET ENCRYPTION ON;
go
```

5. If the local security certificate is used, back up the certificate. If possible, the backup location should be on a different server. If you cannot retrieve this certificate, the data is lost. It doesn’t matter whether there is a valid backup of the encrypted database. In order to restore the backup, a valid certificate is needed. If the only location of the certificate is the production SQL Server instance and the server goes down or the SQL Server instance is lost, the backup is not worth a cent. Therefore it doesn’t make sense to keep the backups of the certificates on the same server; they should be stored on a different computer. The command to backup the certificate looks something like this:
Commands that lock the database can block the ALTER DATABASE command, which is the last command in the encryption or decryption process. For example, if a BACKUP DATABASE command is running, ALTER DATABASE commands are blocked. All DDL commands that affect the structure of the database itself (not the objects within the database) will block the start of encryption.

The first two commands (creating a DMK and creating the certificate) need to be executed in the master database. The third command (defining the algorithm using the certificate) searches the master database for the certificate.

### What Happens When TDE Is Enabled on a Database

The interesting question is what is happening after the last command of enabling encryption against the database is issued. Here the answers:

- The command runs immediately if it is not blocked by a backup. Hence the command can run while applications are online.
- Every page that is written down after TDE is complete is encrypted when it is written to the database.
- Background tasks kick in to read all the database pages and encrypt those pages. This is the most interesting part of the TDE in terms of immediate system impact. We’ll look at this in more detail later in this section.
- tempdb is encrypted with the strongest encryption key available. Regardless of the algorithm chosen for the user database, tempdb is encrypted in AES_256. This encryption can have consequences for other databases that use tempdb intensively. For example, if multiple databases are consolidated on one SQL Server instance, you should not apply TDE if any of the database workloads demand high tempdb usage. TDE should not be applied to databases that experience OLAP-type workloads, because OLAP operations rely heavily on tempdb use. These applications can require intensive scanning of large amounts of data, which can cause intensive I/O load. Because all of the pages read need to be decrypted, the I/O latency can be increased.

### Large SAP ERP Database Encrypted

In order to demonstrate the impact of the background tasks performing the initial encryption of the complete content of the database, we’ll look at a real-world example of encrypting an SAP ERP database with the following characteristics:

- HP DL585 – 4 socket Dual Core Opteron 2.2GHz, 48 GB RAM
- SAN storage Clarion 128 spindles configured as 16 LUNs mounted as mount points for data files
- 20 spindles for the log file
- The database itself has 32 data files, with two data files for each LUN

After encryption is enabled, the first noticeable action is that a background task spins off one thread per LUN. These threads hold database files of the database to be encrypted. One additional thread is spun off as coordinator. Each single of this thread is hard assigned to a particular LUNs or volumes. Different performance characteristics on those LUNs or volumes affect the balancing of the run time of each of the threads performing the initial encryption. This means that some background threads would finish sooner than others.

For example, the following query might run on 16 LUNs.

Query1:

```sql
SELECT session_id, status, command, reads as 'physical_reads', logical_reads, cpu_time, total_elapsed_time FROM sys.dm_exec_requests;
```

The following table displays results.

<table>
<thead>
<tr>
<th>session_id</th>
<th>status</th>
<th>command</th>
<th>physical_reads</th>
<th>logical_reads</th>
<th>cpu</th>
<th>total_elapsed_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>....</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>background</td>
<td>BRKR TASK</td>
<td>1</td>
<td>80</td>
<td>68734</td>
<td>138064053</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>1</td>
<td>565792</td>
<td>0</td>
<td>57946114</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>21146853</td>
<td>42684134</td>
<td>2231907</td>
<td>57946073</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>20758983</td>
<td>44089822</td>
<td>2209890</td>
<td>57946073</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>20797384</td>
<td>44163226</td>
<td>2217640</td>
<td>57946073</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>14833083</td>
<td>29749702</td>
<td>2300734</td>
<td>57946073</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>14955193</td>
<td>29997836</td>
<td>2308906</td>
<td>57946073</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>14874839</td>
<td>29833935</td>
<td>2313094</td>
<td>57946073</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>14440627</td>
<td>28962369</td>
<td>2251875</td>
<td>57946073</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>14803619</td>
<td>29691943</td>
<td>2298812</td>
<td>57946073</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>14994030</td>
<td>30076066</td>
<td>2319968</td>
<td>57946073</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>14895896</td>
<td>29881675</td>
<td>2310187</td>
<td>57946072</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>14501903</td>
<td>29081588</td>
<td>2260343</td>
<td>57946072</td>
</tr>
<tr>
<td>30</td>
<td>background</td>
<td>ALTER DATABASE E</td>
<td>20931868</td>
<td>44163124</td>
<td>2211218</td>
<td>57946069</td>
</tr>
</tbody>
</table>
The first thread with a session_id of 30 is the coordinator thread, which consumes CPU even though it performs almost no I/O. The other 16 threads execute I/O by reading, encrypting, and writing pages.
The fewer LUNs over which the database is spread, the fewer threads that work on the background encryption.

If 16 threads are working in the background, the CPU impact might look like this.
In this example, with more than half of the CPU resources eaten up, you would not necessarily be able to keep the associated SAP application online for normal user workload. The background tasks worked for around 18 hours to encrypt all of the pages.

While the background processes continue to work, this query can be used to get an overview of the progress of work:

```sql
SELECT DB_NAME(database_id), encryption_state, key_algorithm, key_length, percent_complete FROM sys.dm_database_encryption_keys
```

The first column will show the database name. The second column will have a value from 1 to 5 where the values stand for:

1. Unencrypted
Columns 3 and 4 will show the algorithms used for encryption. These are dependent on what was defined in the command CREATE DATABASE ENCRYPTION KEY. Possible values are AES in the length of 128, 196, and 256, or TRIPLE_DES_3KEY. Finally, the last column will show a percentage of progress toward completion. If encryption was just enabled on the database, the status of the database changes from 1 to 2, and the value in the column \textit{percentage_complete} increases. After the whole database is encrypted, the status changes to 3, and \textit{percentage_complete} is reset to 0.

In addition to the CPU workload, there is I/O workload. In this particular case, we were looking at around 400 to 500 I/O operations per second reading 256 KB chunks out of the data files, which added up to a load of around 100 MB per second. On the write side the encrypted pages are written down by either checkpoint or lazywriter.

**Small SAP ERP Database Encrypted**

We also encrypted a smaller SAP ERP database. The configuration of this small database looked like:

- HP DL585 – 4 socket Dual Core Opteron 2.2GHz, 48 GB RAM
- Direct attached storage (DAS) with one LUN for 6 data files on 30 spindles
- 10 spindles for log file

We made the following observations:

- Because one LUN was available for the data drives, there was one background thread at work plus a coordinating thread.
- In terms of CPU, the resource of half a CPU core was acquired by the one background task that encrypted all the pages.
- Given a volume of 30 MB/sec read rate, it took the following amounts of time to move a database from a nonencrypted state to a fully encrypted state:
  - 100 GB database volume ~1 hour with AES algorithms
  - 100 GB database volume ~2.5 hours with TRIPLE_DES algorithms

We saw the same workload conditions when the database was decrypted from an encrypted state.

**Impact of the Process of Encryption**

Besides the CPU impact discussed in the large SAP database example, there are other effects you should consider.
Storage Replication on Hardware Level

When encryption is enabled and the background tasks start to encrypt pages, it may look as if every single page is being changed. This means that the hardware replication will be flooded with pages to be replicated. In our large SAP example, even using commodity hardware, a considerable write load of over 100 MB per second was achieved. Given the fact that normal OLTP systems under heavy load might read 200-300 MB per second, they hardly write around 100 MB per second to disks. We recommend that you perform thorough tests to see whether the hardware replication can stand such a throughput. If the required throughput can’t be achieved and the storage replication develops to a bottleneck that manifests itself in high I/O response time, there will be an immediate negative performance impact on the encrypted system.

SQL Server Database Mirroring or Log Shipping

If your system uses database mirroring or log shipping, there are additional steps to take while enabling Transparent Data Encryption (TDE). These features also can have a strong impact during the phase where the complete database is encrypted (right after TDE is enabled on a database).

Additional Steps When Using Database Mirroring and Log Shipping

In the sequence of commands to enable TDE on a database, additional steps are required between steps 3 and 4 (here). Before TDE is enabled on a principal database or a log shipping source, the mirror server or log shipping destination(s) needs to be provided with the master key and the certificate. First, generate a master key on the secondary or mirror server as well.

```sql
use master;
go;
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'strong password#1';
go;
```

The password doesn’t have to be the same as the one of the master key for the principal instance. This key is used to encrypt the certificate you want to create on the secondary or mirror server as well.

Next, return to the primary or principal server to back up the certificate used for encryption, using the following command:

```sql
use master;
go;
BACKUP CERTIFICATE MyServerCert TO FILE = 'path_and_file_name.cer'
    WITH PRIVATE KEY ( FILE = 'path_and_file_name.pvk' ,
    ENCRYPTION BY PASSWORD = 'strong passwd#2' );
```
The third step is to copy the certificate file and the key file to the mirror and then go to the mirror to create a certificate using these two files. The command would be:

```sql
use master;
go;
CREATE CERTIFICATE MyServerCert
  FROM FILE = 'path_and_file_name.cer'
  WITH PRIVATE KEY (FILE = 'path_and_file_name.pvk',
                  DECRYPTION BY PASSWORD = 'strong passwd#2');
go;
```

The passwords that are used to back up a certificate or to encrypt a private key file are checked against the Windows policies for strong passwords. Also be aware of the slight difference of terms used in the two last commands. In the backup command of the certificate, the password is defined for 'Encryption', whereas in the command to create the certificate out of the backed up files, the same password is used for 'Decryption'. If this is done, you can continue on the principal server issuing the last command of enabling encryption.

### Impact During Initial Database Encryption

In both cases, an eventual bottleneck on the mirror or the log shipping destination will not cause an impact to the principal or primary server. Unlike storage replication, which needs to transfer the complete encrypted pages, the SQL Server transaction log contains only a short entry for each page, recording that the page was encrypted. The pure volume of transaction log records that needs to be transferred to the mirror or the backups of the transaction logs will not cause an issue due to its volume. However, a bottleneck is expected when those log records are restored, either by sending the records via database mirroring or by backing up the transaction log to the mirror or the log shipping destination.

The issue is that encrypting a page requires the same resources on the mirror or log shipping destination as it does on the principal or primary server. The mirror or secondary server is instructed to encrypt the pages, and then the recovery thread or threads read the page and encrypt it. As on the primary or principal server, this operation costs a lot of CPU resources on those servers.

But more importantly, the number of threads performing the encryption on the mirror or secondary side is likely to be different from the number of threads on the principal/primary side:

- For log shipping, only one thread restores the shipped transaction log records. Depending on the number of threads executing the encryption on the primary side, the secondary server may fall behind. For example, in our large SAP example, the encryption on the primary using 16 background threads took 18 hours. A single thread would take a much longer time to perform the same amount of work on the secondary side. In this case, it might be more reasonable to stop log shipping before enabling encryption. After encryption succeeds on the principal, then you can synchronize the secondary with a backup of the primary again.
Database mirroring uses parallel threads to restore the transaction log records that were shipped to the mirror instance. By default it uses one restore thread for every 4 CPU threads. In our large SAP ERP encryption example, this still amounted to 16 threads encrypting on the principal, but only 2 threads were restoring on the mirror side.

If your system encounters an imbalance like the one we encountered, you should consider using trace flag 1485 on the mirror instance. This trace flag specifies one restore thread for every CPU thread. In our large SAP example, a 1:1 ratio between encrypting threads on the principal and restoring thread on the mirror could have been achieved with this traceflag. Reason was that the mirror server did have 16 CPUs (equals the # of threads which performed the initial encryption). This traceflag should be configured as a startup trace flag on the mirror and the principal (in case of an automatic failover) before encryption is started. After setting the trace flag as startup trace flag, the SQL Server instances need to be restarted.

A large imbalance between the threads on the principal/primary and the mirror/secondary can result in a backlog in restore activity. In the case of synchronous mirroring with or without failover, this backlog can result in the inability to failover automatically or manually, because restoring already-received transaction log records could take another few hours. In the case of log shipping, restoring the transaction log backups can fall behind; you would not be able to manually fail over before the restore caught up.

**Impacts of SQL Server 2008 TDE on Production Systems**

**Impact on Backup/Restore**

The most severe impacts on backup and restore of an encrypted database are presented here in order of severity:

- The database backup of an encrypted database will contain encrypted data as well. There is no decryption taking place. However, the pages will be stored on backup as they got read from the data files. Means in an encrypted state. There is no way around this; this was one of the design goals.

- To restore a database from an encrypted backup, you must first restore the certificate for the specific database to the instance the restore should take place against. If this is the original instance, there shouldn’t be a problem (unless the certificate wasn’t dropped from the master database). However, if the database is being restored in order to create a sandbox system or to synchronize the test system, the original certificate is needed on that instance. The database restore itself would progress even without the certificate, but the recovery of the database would fail. When the SQL Server instance tries open the database to start the recovery process, an error is issued stating that the database couldn’t be opened due to a missing certificate.

- If you use SQL Server 2008 or third-party tools to perform backup compression, the results of the compression will be very different compared to backup compression of nonencrypted databases. Backups of encrypted databases are difficult to compress, regardless of the tool used. By design, encryption destroys...
the regular patterns that respond well to compression. Intensive tests with
different databases confirmed that the compressed backups of encrypted
databases using SQL Server 2008 backup compression and several third-party
tools yielded backups that were more or less in the range to the origin non-
compressed backup. Hence using backup compression against an encrypted
database can be regarded as waste of CPU and time resources.

Impact on DBCC Checks

DBCC CHECKDB can cause severe impact when it is run on an encrypted database. We
compared execution times for DBCC CHECKDB on an encrypted database and on the
same database without encryption. When we tested CHECKDB WITH PHYSICAL_ONLY,
an I/O bottleneck was reached before the CPU resources were fully leveraged. In our
small SAP ERP example, an I/O bottleneck was hit at around 100 MB per second of
volume. Therefore the actual run time of the CHECKDB command was the same in the
encrypted as well as nonencrypted test. However, it revealed that the CPU consumption
was very different. For the nonencrypted database, only 10 percent CPU resources were
used, but for the encrypted database, around 28 percent of CPU resources were used.

Loading Data into Encrypted Databases

For covering the case of bulk loading data into an encrypted SQL Server database, a
test was conducted with a small SAP BW table of 500 MB volume. We used bcp to load
this data. The ratio between the loads into a nonencrypted and encrypted database with
one of the 4 different algorithms is more or less the same in nearly all the cases.

<table>
<thead>
<tr>
<th>Load of 500MB using minimal logging</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Table: Heap</td>
<td>Heap</td>
<td></td>
</tr>
<tr>
<td>Recovery Model:</td>
<td>Bulk Logged</td>
<td></td>
</tr>
<tr>
<td>Batchsize:</td>
<td>10000 rows</td>
<td></td>
</tr>
<tr>
<td>Load time in seconds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Encryption</td>
<td>12.7</td>
<td></td>
</tr>
<tr>
<td>AES_128</td>
<td>22.9</td>
<td></td>
</tr>
<tr>
<td>AES_192</td>
<td>25.2</td>
<td></td>
</tr>
<tr>
<td>AES_256</td>
<td>26.6</td>
<td></td>
</tr>
<tr>
<td>TRIPLE_DES</td>
<td>75.2</td>
<td></td>
</tr>
</tbody>
</table>

Disregarding the impact of TRIPLE_DES and looking instead at the AES algorithms, you
can assume that bulk loading data into an encrypted database doubles the run time.
This also indicates that in scenarios where it is important to load data quickly, TDE is
not an optimal choice. If you do want to use TDE, it requires some major refactoring of
the load processes and methods.
Impact on Checkpoint or Lazywriter

Because they use asynchronous I/Os, we found that encryption had less impact on checkpoint and lazywriter tasks. You should be able to compensate for latency extensions by issuing more I/O requests.

However, CPU consumption by checkpoint or lazywriter increased up to 20 times. In particular, the checkpoint thread and the I/O completion thread consumed large amounts of CPU while they executed checkpoint tasks.

Impact on Table Scans

In order to test a scenario reflecting an extreme case in the workload of a single-threaded scan, we scanned a 30-GB fact table into a completely cold SQL Server buffer pool. While a buffer pool without a single page of the fact table is unlikely, a fact table in an OLAP schema is likely to be fully scanned, more or less. Not having ideal aggregates or indexed views prepared usually end in such cases.

Again we disregarded the TRIPLE_DES algorithm and focused instead on the AES_256 algorithm. The I/O volume achieved per second was reduced from 210 MB per second to 60 MB per second using the AES_256 algorithm. To compensate for this throughput decrease, you would need to leverage parallel query execution. Even with parallel query execution, CPU consumption would increase with encryption, and overall throughput would still decrease.

Homogeneous Database Copies

One way to make a homogeneous database copy is to copy the database files of a database to the new server and attach them to the new instance. As described in “Impact on Backup and Restore,” you need to install the certificate for the database to be attached on this new instance. Otherwise SQL Server issues an error stating that it can’t open the database because it can’t find the certificate.

Encryption Conclusions

This section summarizes the lessons learned and other main points of this section:

- One of your first considerations regarding administration of certificates and keys should be to avoid data loss due to the loss of a certificate or key. If you are considering TDE for SAP, you should also consider using an EKM system.
- The certificate needs to be installed on the mirror side of a database mirroring scenario or log shipping destination before encryption is enabled on the main database.
- The processes that kicks in immediately after encryption is enabled may be able to run while the database is online. It depends on the number of LUNs the data files of the database are located on and the existing CPU resources available compared to the number of LUNs and the I/O throughput achievable. Unfortunately the only way to control the number of threads that are issued is the number of LUNs the database is spread over.
If you are encrypting a smaller database, the overhead of the TRIPLE_DES algorithm can be up to three times higher than that of the AES algorithm.

Use startup trace flag 1485 if you want to encrypt a system that uses SQL Server database mirroring. This trace flag prevents the mirror from falling behind as encryption is applied.

With log shipping, in extreme cases we recommend that you synchronize the secondary with a new backup of the primary after the primary is encrypted.

Database consistency checks will consume nearly three times more CPU resources. Hence the ability to run those on production database might be more restricted.

As TDE takes on resources or adds I/O, latency needs to be tested very thoroughly. On a case-by-case basis, you need to determine whether these impacts are acceptable for your particular systems and Service-Level Agreements (SLAs).

It is highly likely that OLAP and data warehousing systems will be impacted in a more severe way in terms of response times than OLTP applications, due to latencies in the I/O path and increases in resource consumption.

Scenarios where time-critical mass loads of data are executed will be impacted severely by the usage of TDE.

Backup compression by SQL Server 2008 or other third-party tools will not show significant compression rates with encrypted databases. Especially with large databases in the terabytes space, this could cause significant costs in either providing more disk space for backups or longer usage of backup robots for one particular database backup.

On small servers with one or two CPU threads, checkpoint activity can result in short burst of overly high CPU consumption, which could be experienced as hiccups in response times.

### Resource Governor

SQL Server 2008 introduced Resource Governor Functionality. The Resource Governor allows limiting resources taken by a class of queries. It gives the possibilities to define different resource pools and their limits in resource consumption. Connections then can be assigned to those pools. Queries issued over these connections are then governed by the resource limits of the specific resource pool. The feature itself consists of three different components:

**Resource pools:** SQL Server already contains two predefined resource pools, the internal and the default pool. The internal pool is used for internal functions like checkpoint writer and lazywriter, and its CPU and memory limits can’t be changed. All resources are always available to threads that use this pool. New pools can be created at any point in time. The resources limitations that can be associated with these pools are CPU resource consumption and memory consumption, where minimum and maximum can be set. Changes can be made to the default pool as well, though we do not recommend this if you are running SAP applications.

**Workload Pools:** A workload group serves as a container for session requests that are similar according to the classification criteria that are applied to each request. A
workload group allows the aggregate monitoring of resource consumption and the application of a uniform policy to all the requests in the group. A group defines the policies for its members. Resource Governor predefines two workload groups: the internal group and the default group. Users cannot change anything classified as an internal group, but can monitor it. Requests are classified into the default group if the following conditions exist:

- There are no criteria to classify a request.
- There is an attempt to classify the request into a nonexistent group.
- There is a general classification failure.

Resource Governor also provides DDL statements for creating, changing, and dropping workload groups. User-defined workload pools can be assigned to different resource pools, whereas the internal pool and default pool have a fixed assignment to the internal and default resource pool.

**Classifications:** Resource Governor supports the classification of incoming sessions. Classification is based on a set of user-written criteria contained in a function. The results of the function logic enable Resource Governor to classify sessions into existing workload groups. To use classification, you write scalar function that contains the logic that you want to use to assign incoming sessions to a workload group.

The design relies on the fact that a user-written function distinguishes incoming connections based on different criteria (such as user name, server name, or application name) in order to assign them to different workload pools. If none of the qualifiers hit, the connection is assigned to the default pool. After a connection is assigned to a workload pool, the assignment can’t be changed unless the classification function is changed and the connection is re-established. On the other hand, a workload pool can be assigned to a resource pool.


It is obvious that resource dimensions like I/O or network traffic are not covered in this version of Resource Governor. These dimensions only can be indirectly controlled by controlling CPU or memory for the jobs or requests.

How does this apply to run SAP applications? We actually recommend not using SQL Server 2008 Resource Governor with connections and queries issued by SAP applications, because it can interfere with activity that is issued by SAP ABAP or Java application instances. However, there might be areas outside of the requests issued by SAP that you could tackle with Resource Governor. For example, it works well with SQL Server 2008 backup compression or CHECKDB. In [How to: Use Resource Governor to Limit CPU Usage by Backup Compression (Transact-SQL)](http://technet.microsoft.com/en-us/library/cc280384.aspx) in SQL Server Books Online, an example is shown how to use Resource Governor to limit the amount of CPU spent with SQL Server 2008 backup compression. The great example described also demonstrates the concepts of Resource Governor effectively.
SQL Server 2008 Auditing Functionality

SQL Server 2008 includes a new feature, SQL Server Audit that you can use to audit user activity. This functionality was not designed for auditing database access of different users using SAP applications; it was designed for logging access and activities of different database users accessing a database or specific tables within a database schema. As described earlier in this paper, SAP applications establish their connections to SQL Server with one specific user’s context. The real SAP user behind the requests issued over the connections is hidden from SQL Server by the SAP applications. Hence there is no possibility to detect or log the activity of a specific user who logged into an SAP application. However, SQL Server 2008 Audit functionality can help you address the compliance and auditing concerns in situations where users who are not SAP users are accessing the database directly through means other than one of the SAP applications. For example, an HR department might have concerns about database administrators directly accessing the SAP database (which no doubt is a necessity of their job) and accessing sensitive data. At the end it boils down to be able to monitor the activities of operators or even DBAs on SAP databases. For this purpose the new audit functionality was created as a first class server object. This means that instead of using SQL Server Profiler to audit, you define audit sessions and audit rules by simple Transact-SQL statements. No additional tools are necessary. The functionality is completely integrated into the SQL Server Database Engine. In former SQL Server releases one needed to look in different places or use different features and construct to get the auditing data one can get now in one place. A lot of data needed to be gathered with SQL Server Profiler before. For specific cases one even needed to use triggers. All these trace data now can be collected via the SQL Server 2008 Auditing framework.

Figure 27: Move capabilities of different tools into one SQL Server Database Engine functionality to audit activities

However, to make auditing of non-SAP database accesses successful in an SAP landscape, you have to make a few changes. There are limitations to the selectivity of the tool. They are related to the permission context within which SAP establishes the connections to SQL Server (explained here). Hence all requests SAP ABAP based applications issue are issued under the ‘db_owner’ database role. Unfortunately, that is the same role a DBA would be mapped into. Hence it means that you must collect every activity on the database executed to one or multiple tables under ‘db_owner’ and then filter out the activities issued by a certain server-principal (login) that isn’t the SAP...
This also means that it is in no way advisable to audit activities on all tables of an SAP database. The risk is that tremendous numbers (gigabytes) of entries to the auditing log are written and tremendous resources are consumed for doing so. So keep the number of tables to be audited in a small number.

Here are the concepts and objects of SQL Server 2008 Auditing.

**Audit Session:** Is the umbrella object of an auditing session. All specification of the particular auditing session relates to this object. This object is valid on a SQL Server instance level. It is possible to have multiple different auditing sessions within one SQL Server instance. The destination for the auditing data is defined when the object is created.

In case of writing into a file, attributes like maximum file sizes, number of rollover files, and whether the volume should be allocated in advance need to be determined. The maximum space can be set to unlimited. If you set the maximum space to unlimited, however, preallocation can’t take place.

Two other extremely important attributes need to be set in this auditing session definition. The first attribute, the QUEUE_DELAY option, controls whether the auditing events need to be reported into the destination synchronously. You make this decision based on whether your organization can afford delays in having the events written into the auditing destination. This attribute is set via the QUEUE_DELAY option. By default, SQL Server 2008 continues its normal operations if a failure occurs. But you do have the option of changing this default behavior.

A statement to define such an auditing session could look like:

```
CREATE SERVER AUDIT Audit_DBA_Activity
    TO FILE (FILEPATH = '\SQLPROD_1\Audit\',
             MAXSIZE = 100MB) with
             (QUEUE_DELAY=1000);
```

This command created an umbrella auditing session that writes the data to files on another server. The files are rolled over after 100 MB with no limitation to the maximum number of files. Reporting of auditing events can be delayed by 1000 milliseconds or 1 second. If the auditing session runs into error conditions, the SQL Server instance will continue to work. Writing into the local Windows Security Event Log could look like:

```
CREATE SERVER AUDIT Audit_DBA_Activity
    TO SECURITY_LOG with (QUEUE_DELAY=1000);
```

**Note:** The context SQL Server is running with requires the permission of Generate Security Audits in the User Rights Assignment under Local Security Policy.

Because auditing on some tables is enabled on all activities db_owner mapped users perform, it is not advisable to use anything else than a disk location, because a huge amount of data can be logged. Therefore you should always start by using a local file as your audit target. If that does work fine then and you have an idea on the data volume you can try a file on another server.

**Server-wide audit specification:** Is a specification of events that should be tracked on a SQL Server instance basis. The specifications are assigned to the umbrella object of the auditing session. One server-wide audit specification is allowed. For more information, see [SQL Server Audit Action Groups and Actions](#).
For SAP, all events in regards to changes on the auditing specification need to be added into the server-wide auditing specification. Therefore the minimum declaration needs to look like this:

```sql
CREATE SERVER AUDIT SPECIFICATION Audit_DBA_SERVER_Specification
FOR SERVER AUDIT Audit_DBA_Activity
  ADD (AUDIT_CHANGE_GROUP);
GO
```

For SAP, if you want to see whether DBAs change permissions on server-wide objects or database-wide objects, the specification could look like this:

```sql
CREATE SERVER AUDIT SPECIFICATION Audit_DBA_SERVER_Specification
FOR SERVER AUDIT Audit_DBA_Activity
  ADD (AUDIT_CHANGE_GROUP),
  ADD (SUCCESSFUL_LOGIN_GROUP),
  ADD (LOGOUT_GROUP),
  ADD (LOGIN_CHANGE_PASSWORD_GROUP),
  ADD (SERVER_ROLE_MEMBER_CHANGE_GROUP),
  ADD (SERVER_PRINCIPAL_CHANGE_GROUP),
  ADD (DATABASE_PRINCIPAL_CHANGE_GROUP),
  ADD (SERVER_OBJECT_PERMISSION_CHANGE_GROUP),
with (STATE=OFF);
GO
```

The preceding list of groups audited is not a perfect list; these are just examples. You can compare it to the list of groups possible in SQL Server Books Online and add other groups as needed.

Specific compliance requests might require the addition of other activities to be audited. The preceding query would return SAP applications establishing connections and logging into the server. However, it is not usual for SAP applications to create new users, change passwords or change permissions to objects. Because these activities usually are not done by SAP, they might be events that one wants to record or audit.

**Database-wide audit specification:** This type of specification defines audit specifications in the scope of a certain database. This audit specification is assigned to the general audit session as well. You can select special audit groups that generally apply to all objects in the database. These could catch events like changes on permissions of objects. An example might look like this:

```sql
CREATE DATABASE AUDIT SPECIFICATION Audit_DBA_DB_Specification
FOR SERVER AUDIT Audit_DBA_Activity
  ADD (DATABASE_PRINCIPAL_CHANGE_GROUP),
  ADD (SCHEMA_OBJECT_CHANGE_GROUP)
with (STATE=OFF);
```

In this case, all changes to permissions of database users would be recorded as well as changes to permissions on objects within the schemas of the database. This audit
specification must be executed within the database it should apply to, and it works within the frame of the database only.

However, you can also try to audit activity on specific tables by specific database_principals (users). As mentioned earlier, SAP connections are established with a server_principal (login) which is mapped into the ‘dbo’ database_principal (user). This is unavoidable because SAP applications require certain sets of permissions that are easiest to get as db_owner. In a simple case such a specification can look like this:

```sql
CREATE DATABASE AUDIT SPECIFICATION Audit_DBA_DB_Specification
FOR SERVER AUDIT Audit_DBA_Activity
ADD (SELECT, INSERT, UPDATE, DELETE, EXECUTE
    ON ssq.HRP1001 BY dbo)
with (STATE=ON);
```

In this case, all activities on table HRP1001 in the schema ssq are audited and recorded. The specification audits activities issued by the SAP ABAP based application as well as the activities from anybody else. Therefore you should be extremely careful using such kind of auditing. The impact in performance and resource consumption can increase with a growing number of tables to be audited, and the amount of data to be filtered and audited can grow into enormous amounts.

**Important:** Never try to audit the whole SAP database. Based on our own experience, we can tell you that it completely brings down the throughput of the system. This is why the syntax to audit the whole database is not included in this paper.

Let’s go through it in theory and discuss some aspects of it. The first issue, which is unavoidable at this point in time, is the fact that DBAs are usually required to be members of the **sysadmin** server role. This means the DBAs have all the power and can enable, disable, and view the results of an audit session. So you need to audit who applies changes to the auditing rules and who stops the auditing session.

The next discussion is around how to read and filter the data. The data can be read with a T-SQL statement like this very easily:

```sql
SELECT * into audit_dba_activity FROM
    sys.fn_get_audit_file('C:\temp\Audit_DBA_Activity.sqlaudit',
    default,default);
```

This statement places data in a table and analyzes it according to events and activities. All DDL and DML statements are shown. However, parameter values for parameterized statements as with SAP applications might not be displayed, in contrast to nonparameterized statements, where all parameter values are visible in the audit trace.

Another possibility for resolving some of the classes named in the trace could be to perform a SELECT like this:

```sql
SELECT a.name AS Action,c.class_type_desc AS ObjectType,
    f.server_principal_name,f.schema_name,f.OBJECT_NAME,f.statement
FROM fn_get_audit_file(@filepattern,NULL,NULL) AS f
JOIN sys.dm_audit_class_type_map c ON f.class_type = c.class_type
```
A result like this could be shown.

<table>
<thead>
<tr>
<th>Action</th>
<th>ObjectType</th>
<th>server_principal_name</th>
<th>schema_name</th>
<th>OBJECT_NAME</th>
<th>statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>TABLE</td>
<td>SAP\sapadm ssq</td>
<td>HRP1001</td>
<td></td>
<td>select * from ssq.HRP1001</td>
</tr>
<tr>
<td>ALTER</td>
<td>DATABASE AUDIT SPECIFICATION</td>
<td>SAP\sapadm</td>
<td>Audit_DB_DB_</td>
<td>ALTER DATABASE AUDIT SPECIFICATION [Audit_DB_DB_Specification]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specification</td>
<td>WITH (STATE = ON)</td>
<td></td>
</tr>
<tr>
<td>ALTER</td>
<td>DATABASE AUDIT SPECIFICATION</td>
<td>SAP\sapadm</td>
<td>Audit_DB_DB_</td>
<td>ALTER DATABASE AUDIT SPECIFICATION [Audit_DB_DB_Specification]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specification</td>
<td>WITH (STATE = OFF)</td>
<td></td>
</tr>
<tr>
<td>SELECT</td>
<td>TABLE</td>
<td>SAP\sapadm ssq</td>
<td>HRP1001</td>
<td></td>
<td>select * from ssq.HRP1001</td>
</tr>
</tbody>
</table>

As you can see, in the SELECT statement there are two DMVs that encode the different class types and actions into more descriptive names.

The next point to discuss is where to keep the auditing data. You can keep the data as a file on the database server or on another server, or you can write it into the Windows Security Log. As mentioned earlier, substantial data can be collected. Hence it might make sense to start with a local file to get an idea about the resources and space required. If the auditing data will be kept on the same server, the DBAs need to be restricted in their access on the Windows operating system. The common practice of adding the DBA accounts to the Windows Administrator group certainly wouldn’t work anymore. You would need to assign the DBAs to another group and then restrict this group’s access to the auditing files.

If the requirements in volume per second and space are clear, you should move the auditing records to a server that is not accessible to the DBAs or the users who are being audited. This action helps prevent an unauthorized user from deleting the trace on the file level.

However, there is nothing to prevent a DBA who is in the sysadmin role from reading the auditing trace with SQL Server methods using a query as shown earlier. Because SAP requires xp_cmdshell to be enabled, and because xp_cmdshell uses the SQL Server Service Account to execute operating system commands, a DBA could check on the file using Windows commands and even delete files that have been rolled over (local or remote).
For SAP applications, the SQL Server Audit feature doesn’t provide complete coverage for monitoring members of the **sysadmin** role. However you can use this powerful tool to create audit traces on users who have more limited permissions.

**Star Join Improvement**

In order to speed up typical SAP BW queries, SQL Server 2008 introduced improvements in the way star join queries are processed. To use this functionality, ensure that the **max degree of parallelism** option is set to 2 at least. If you set the degree of parallelism away from 1 in the SQL Server instance, configuration can be too generous on the other side and can have too dramatic an impact in resource consumption. In order to use this feature, SAP provides changed program coding for SAP BW where a hint to use more than one CPU is given to very selective amount of queries. For more information about how to influence the degree of parallelism for specific queries, see OSS Note #1126568.

We recommend starting your first tests with setting a degree of parallelism of 2 with the method described in the SAP OSS note. Usually the biggest percentage gain in run time is seen when a query that was executed in a single thread starts executing with two threads (or CPUs). If you use more than two threads, you could experience unpredictable response times if many users run reports where queries are executed in multiple threads. This situation could provoke CPU resource loads that cause SQL Server to stop executing queries in multiple threads. This again would lead to nondeterministic response times, because some queries might be executed in multiple threads and some others not. This was the initial reason that SAP and Microsoft recommended setting **max degree of parallelism** to 1 when running SAP applications.

**Windows Server Configurations**

We today assume that the majority of SAP customers running on Windows have made the transition to Windows Server 2003. Windows Server 2003 was a large jump in quality and scalability compared to Windows Server 2000. 64-bit releases also are only available on Windows Server 2003 and Windows Server 2008. Within Windows Server 2003, we recommend that you apply SP2 to your database server and application servers.

SAP meanwhile announced General Support for Windows Server 2008. You might need to download a new JVM from SUN using Windows Server 2008. Windows Server 2008 SP2 will be supported with NetWeaver 7.00 based SAP applications and later releases of SAP software only. One of the main reasons for restricting Windows Server 2008 so narrowly is the fact that the complete cluster setup changed in Windows Server 2008 compared to Windows Server 2003. Hence SAP needed to make changes in their Installers to accommodate the changes in the Windows cluster setup that would have been extremely difficult to make for older SAP releases. For more information about the state of Windows Server 2008 support, see OSS Note #1054740.
Patches Required for Windows Server

Windows Server 2003 SP2 requires one patch, which can very positive impact on performance of the application server side of SAP applications. The patch should correct the following behavior.

SAP creates a large memory mapped file that assembles Extended Memory. The different SAP work processes get the memory for their working set out of the Extended Memory by mapping segments (4 MB segments on 64-bit) into the particular work process when a user request requires space to, for example, build internal tables with results of database requests. It can happen that dozens or several hundred of these segments get mapped into the work process, depending on the volume of data which is handled in the user request. When the user context gets rolled out of an SAP process again after the request’s execution finished, the SAP process unmaps the memory segments it borrowed from the SAP Extended Memory. For the Windows Memory Management, this unmap operation leaves memory segments that were changed without associated processes. These two facts are reason that Windows Memory Management pages out these memory segments preventively to the Windows page file, if there is CPU available. Because of this behavior, a computer running an SAP ABAP application can exhibit a very high rate of page-out operations towards the Windows page file. The higher the workload, the higher the page-out rate will be. In extreme cases, it could be as high as writing a constant stream of 20 MB per second to the Windows page file.

However, do not let this fact lead you to incorrect conclusions. The page-out rate in such cases itself does not tell you anything about experiencing a situation of memory pressure. In order to evaluate whether there is memory pressure, you need to evaluate the page-in rate from the Windows page file. If the page-in rate is high as well, you can suspect memory pressure. Nevertheless, with increasing throughput that can be handled by commodity hardware, the I/O rate caused by this page-out sometimes hit values that severely impacted the performance of the application instances. In order to reduce the page-out rate by a dramatic portion, a patch was released for Windows Server 2003. For more information about how to get this patch, see SAP OSS note #1009297. Our experience with this patch was extremely positive. Therefore we recommend that you deploy this fix on all servers running SAP application server instances.

Windows Server Configurations

In case of Windows Server 2008 and Windows Server 2003, there are not a lot of configuration changes which are required for running SQL Server or SAP applications. Some of these attributes are set automatically when you install SQL Server or SAP. However, for the sake of completeness, we list them all here.

Set Advanced Server Options

In Windows Server, the following options should be set. In the System Properties dialog box, on the Advanced tab, under Performance, click Settings. In the Performance Options dialog box, on the Visual Effects tab, click Adjust for best performance.
Because we are talking about database servers or SAP application servers, sophisticated graphics performance is not required. The base settings for visual effects are sufficient.

Next, check the advance performance options. In the **Performance Options** dialog box, on the **Advanced** tab, under **Processor scheduling**, click **Background services**. Under **Memory usage**, click **Programs**.
For SAP applications as well as SQL Server, the settings should be for background processes scheduling and adjustment for programs in the memory section. Setting **Memory usage** to **Program** is especially important for the SQL Server Relational Engine, which does not use the Windows File Cache.

Another setting in Windows Server 2003 which that no longer exists in Windows Server 2008 controls throughput for network applications. The following options should be set in Windows Server 2003. In Windows Server 2008, this optimization is performed automatically.

In **Network Connections**, expand **Network Tasks**, and then right-click LAN or High-Speed Internet. In the **Local Area Connection Properties** dialog box, click **File and Printer Sharing for Microsoft Networks**, and then click **Properties**. In the **File and Printer Sharing for Microsoft Networks Properties** dialog box, click **Manage data throughput for network applications**.

![Figure 30: Maximize for network applications](image)

A Change to the Windows Server 2003 Registry

Another common scenario in the SAP space today is to perform a backup to local disks and then have an agent of a centralized backup infrastructure pick up the backup files and copy those to the backup infrastructure. However, having full database backups on
the production database server also triggers other scenarios where the backup files should be copied manually to other destinations, such as building up a sandbox system in production size or synchronizing a test system or development system with the production volume database. If you are working with databases of a few terabytes, you probably want to finish such a copy pretty quickly. You can use database backup compression as offered with SQL Server 2008 or a third party. You can also create multiple backup files, so that you can start these copies in parallel, which would lessen copying time.

If the server the copy is going towards is remote, such as on your disaster recovery site, the copy may take longer. You can use multiple parallel copy jobs to accelerate the copy using a tool such as the Robocopy utility.

We have had customers experience problems with large file copies, where the whole database server system seems to freeze, even preventing access through Terminal Server. Customers had to resolve the situation by taking the server down. In post-mortem analysis, we analyzed minidumps SQL Server wrote during the time the server was unresponsive. These minidumps show memory listings like the following:

- MemoryLoad = 99%  
- Total Physical = 73724 MB
- **Available Physical = 3 MB**
- Total Page File = 104552 MB
- Available Page File = 55236 MB
- Total Virtual = 8388607 MB
- Available Virtual = 8312979 MB

Despite the fact that SQL Server is usually configured with the Lock Pages in Memory option, it looked as if SQL Server wasn’t getting enough memory. Low physical memory could trigger all kinds of issues, given that the Lock Pages in Memory option applies only to the SQL Server buffer pool (including associated cache areas that are taken out of the buffer pool) and not to what we call mem-to-leave (direct memory allocations of SQL Server). When we checked SQL Server DMV `sys.dm_os_ring_buffers` in another case where we still had access to the server, it suggested that SQL Server tried to reduce its memory footprint rapidly as less and less physical memory was available. Windows was also prevented from being able to authenticate new users, which explained why SQL Server wouldn’t accept new connections. For more information about this behavior, see **You may experience a decrease in overall system performance when you are copying files that are larger than approximately 500 MB in Windows Server 2003 SP1 or on Windows Server 2003 SP2** (http://support.microsoft.com/kb/920739/) in the Microsoft Knowledge Base.

This article describes a scenario like ours, where large files are copied over more or less slow network lines. Copying a few backup files might create a scenario where the Windows Cache Manager can’t flush the pages fast enough to the ‘slow disk’. Because dirty pages in the Cache Manager are limited to half of the memory, which in today’s case of database server is 32 or 64 GB or even more, the Cache Manager could expand and eat up all the memory until a system is as good as starved for memory. To avoid such situations, set `SystemCacheDirtyPageThreshold` to 3 GB via registry subkey as described in the Knowledge Base article. Several customers running into a similar problem changed this setting as recommended here to 3 GB, which resolved the problem.

Windows Server 2003 Scalable Network Pack

Windows Server 2003 SP2 introduced a whole set of new network scalability features for x64 and x86, the Windows Server 2003 Scalable Network Pack. For IA64 these features got introduced with Windows Server 2008. For Windows Server 2003 SP1, the package can be downloaded from http://support.microsoft.com/?kbid=912222

These features can include the following:

- **TCP Chimney Offload.** TCP Chimney Offload provides automated, stateful offload of Transmission Control Protocol (TCP) traffic processing to a specialized network adapter implementing a TCP Offload Engine (TOE). For long-lived connections with large-sized packet payloads, like those associated with file server, backup and storage workloads, and other content-heavy applications, TCP Chimney Offload greatly reduces CPU overhead by delegating network packet processing tasks, including packet segmentation and reassembly to the network adapter. By using TCP Chimney Offload, you can free up CPU cycles for other application tasks, such as supporting more users sessions or processing application requests with lower latency.

- **Receive-Side Scaling.** Receive-Side Scaling enables the processing of inbound (received) networking traffic to be shared across multiple CPUs by leveraging new network interface hardware enhancements. Receive-Side Scaling can dynamically balance the inbound network traffic load as either system load or network conditions vary. Any application that has significant inbound networking traffic and runs on a multiprocessor host, such as a Web or a file server, can benefit from Receive-Side Scaling.

- **NetDMA.** NetDMA enables memory management efficiencies through direct memory access (DMA) offload on servers equipped with supportive technology, such as Intel’s I/O Acceleration Technology (I/OAT).


TCP Chimney Offload

All of the features of the Scalable Networking Pack were implemented into Windows Server 2008 as well. However, one of the features, TCP Chimney Offload, can sometimes create problems with certain network cards and drivers. Microsoft documented these problems and provided suggestions in Knowledge Base article 943861. For more information, see [Error message when an application connects to SQL Server on a server that is running Windows Server 2003: "General Network error," "Communication link failure," or "A transport-level error"](http://support.microsoft.com/kb/942861) in the Microsoft Knowledge Base.

In the SAP space, TCP Chimney Offload caused problems that resulted in unmotivated connection breakdowns between the SAP application servers and database servers.
However a lot of problems were not only reported out of the SAP pace, but over the complete bandwidth of applications connecting to SQL Server. Therefore TCP Chimney Offload was disabled by default on Windows Server 2008. Because it does not benefit performance for the majority of SAP applications, we recommend that you disable it for Windows Server 2003 as well. In order to do so, follow these steps:

1. Click **Start**, click **Run**, type `regedit`, and then click **OK**.
2. Locate the following registry subkey:
   ```plaintext
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
   ```
3. Right-click **EnableTCPChimney**, and then click **Modify**.
4. In the **Value data** box, type `0`, and then click **OK**.

Please note that the registry location for Windows Server 2008 changed to:
```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NDIS\Parameters
```

TCP Chimney Offload should be disabled on the database server and on the application servers.

**Receive Side Scaling**

Because of the architecture of NDIS 5.1 miniport drivers, a network adapter in a server with multiple CPU threads running Windows Server 2003 is always associated with a single CPU thread. NDIS 5.1 allows a single deferred procedure call (DPC) to execute at any given time for each network adapter. Any network packets received from the
network on a particular network adapter trigger an interrupt to the host processor and eventually cause a DPC to execute on one of the system’s CPU threads, typically on the processor that was interrupted. The network stack processes all the received packets in the context of this DPC.

In large SAP systems that have many separate SAP application instances on dedicated application servers that issue thousands of requests against a dedicated database server, this single CPU assignment could represent a bottleneck. In addition, some processors and systems route all interrupts from a single device to one specific processor, which also increases the potential for a scalability bottleneck.

In such situations a single CPU thread needs to handle all the traffic received by the network adapter, regardless of whether other CPUs are available. With the changes deployed in Windows Server 2003 SP2 and Windows Server 2008, a network adapter is not associated with a single CPU anymore. Instead, the processing for incoming traffic is distributed among the processors on the computer. This new feature is called Receive-Side Scaling. It allows much more traffic to be received by a network adapter on a high-volume server. NDIS 5.2 and Receive-Side Scaling enable multiple DPCs on different processors for each network adapter, while preserving in-order delivery of messages on a per-stream basis. Receive-side Scaling also supports dynamic sharing inbound network processing across multiple processors.

With Receive-Side Scaling, a multiprocessor computer can now handle more incoming traffic. To take advantage of this new feature, compatible network adapters that can utilize the new architecture provided need to be installed. Receive-Side-Scaling-capable network adapters are available from many network adapter vendors.

By default, Receive-Side Scaling can use up to 4 CPU threads. However, some NIC vendors might set the number of CPUs differently. In the current SAP deployments, the 4 CPUs were sufficient.

If you do need to change this default value, you can do so in the following registry subkeys:

- **Windows Server 2003**: MaxNumRssCpus in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`
- **Windows Server 2008**: MaxNumRssCpus in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NDIS\Parameters`

For more information, see [Setting the Number of RSS Processors](http://msdn.microsoft.com/en-us/library/bb734079.aspx).

Usually the CPUs used are on the first CPU thread (CPU 0). The location can be changed by setting RSSBaseCPU. This can be done in the following subkeys:

- **Windows Server 2003**: RssBaseCpu in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`
- **Windows Server 2008**: RssBaseCpu in `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NDIS\Parameters`

For more information, see [Reserving Processors for Applications](http://msdn.microsoft.com/en-us/library/bb734080.aspx) in the Windows Driver Kit.

Whether these registry keys work also depends on the particular NIC driver. For example, Broadcom adapters were known for a while having ignored these registry keys. In the past customers didn’t report any issues around Receive-Side Scaling.
However, you should start with these explanations when you interpret information like that shown in figure 32.

Figure 32 shows a highly loaded database server in a three-tier SAP configuration. Especially for larger SAP three-tier implementations with a high volume of network traffic, Receive-Side Scaling provides increased scalability. The first four CPUs in the figure show extreme high system CPU usage by working through thousands of DPCs per second.

![Figure 32: Demonstration of Receive-Side Scaling using 4 CPU threads starting from CPU 0 on](image)

**Timer Settings for SAP Applications**

SAP has used the high-frequency RDTSC timers for a long time. With today's NUMA type of hardware architecture SAP applications can report issues with the timers. The most common issue becomes visible when you look at the developer traces of the work processes (for example, C:\usr\sap\E62\D01\work\dev_w0). Entries there reveal that the statistics end time is before the statistics start time. A request was started in a work process that ran on one CPU. While the request was being worked on, Windows scheduled the work process on another CPU, which had a delta in tick counts to the first one (in this case a lower tick count). Hence when the time between the start and end times for the request was calculated, the end time appears to have been earlier than the start time. (Statistics like this are invalid and will not be counted in ST03 statistics or STAD).

When an SAP application instance starts, the tick counts of the different CPUs are checked in order to figure out whether there is a certain percentage of drift. If this is the case, the SAP application instance will start and use the QueryPerformanceCounter in Windows. Because this counter is slower than the RDTSC counter, it is not used by default, but only when drift at startup time of the instance is noticed. After the instance is started, no additional checks are done. Because an SAP application instance often starts after a server reboot, the tick counts of the CPUs not yet far apart from each other, so SAP does not detect sufficient drift. However, after a few days of operation,
the developer traces might be full of these messages. For such cases, SAP provides a way to set an environment variable that will direct SAP to use QueryPerformanceCounter in Windows when the instance starts.

For more information, see OSS Note #532350.
# Appendix

## Important SAP OSS Notes Related to SQL Server or SAP Basis Architecture

<table>
<thead>
<tr>
<th>Note ID</th>
<th>Note Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>62988</td>
<td>Service Packs for MS SQL Server</td>
</tr>
<tr>
<td>98051</td>
<td>Database Reconnect: Architecture and Function</td>
</tr>
<tr>
<td>159316</td>
<td>Reorganizing tables under SQL Server</td>
</tr>
<tr>
<td>327494</td>
<td>Configuration Parameters for SQL Server 2000</td>
</tr>
<tr>
<td>532350</td>
<td>Incorrect statistical data due to unsynchronized timers</td>
</tr>
<tr>
<td>542468</td>
<td>Activate manual statistics generation on infocubes in BW</td>
</tr>
<tr>
<td>600027</td>
<td>Installing the correct MS SQL Server Code Page</td>
</tr>
<tr>
<td>639702</td>
<td>JDBC driver for MS SQL Server</td>
</tr>
<tr>
<td>652634</td>
<td>FOR ALL ENTRIES performance with Microsoft SQL Server</td>
</tr>
<tr>
<td>666805</td>
<td>MS SQL Server: Migration from 32-Bit to 64-Bit</td>
</tr>
<tr>
<td>683447</td>
<td>SAP Tools for MS SQL Server</td>
</tr>
<tr>
<td>767691</td>
<td>Migration of SAP Systems to/from MS SQL Server</td>
</tr>
<tr>
<td>799058</td>
<td>Setting up Microsoft SQL Server 2005</td>
</tr>
<tr>
<td>879941</td>
<td>Configuration Parameter for SQL Server 2005</td>
</tr>
<tr>
<td>905631</td>
<td>SAP Release Planning for SQL Server 2005</td>
</tr>
<tr>
<td>924288</td>
<td>Installation of 6.x based SAP systems on MS SQL Server 2005</td>
</tr>
<tr>
<td>941595</td>
<td>Download J2SE 1.4.2 for the x64 platform</td>
</tr>
<tr>
<td>965145</td>
<td>Installation of 46C Systems on MS SQL Server 2005</td>
</tr>
<tr>
<td>965908</td>
<td>SQL Server Database Mirroring and SAP Applications</td>
</tr>
<tr>
<td>1002587</td>
<td>Flat Memory Model on Windows</td>
</tr>
<tr>
<td>1009297</td>
<td>Windows Server 2003 Family: High Paging Rates</td>
</tr>
<tr>
<td>1054740</td>
<td>SAP system installation on Windows Server 2008</td>
</tr>
<tr>
<td>1054852</td>
<td>Recommendations for migrations to SQL Server</td>
</tr>
<tr>
<td>1109274</td>
<td>New JDBC Driver for NetWeaver 7.0 and higher</td>
</tr>
<tr>
<td>1126568</td>
<td>Enable Star Join Optimization with SQL Server 2008</td>
</tr>
<tr>
<td>1152240</td>
<td>Setting up Microsoft SQL Server 2008</td>
</tr>
<tr>
<td>1238993</td>
<td>Proportional File Autogrowth with SQL Server 2008</td>
</tr>
<tr>
<td>1241751</td>
<td>SQL Server Minimal Logging Extensions</td>
</tr>
</tbody>
</table>
Related Links and Online Resources

Microsoft SQL Server 2008:

Windows Server 2008:

Microsoft – SAP Customer Information Center:
http://www.microsoft.com/sap

SAP and SAP ERP on SQL Server 2008 courses for administrators:
http://www.microsoft.com/sap/events.aspx

SAP AG:
http://www.sap.com/index.epx

SAP NetWeaver:

SAP SDN SAP on SQL Server:
https://sdn.sap.com/irj/sdn/mssql

Blog on SAP running on SQL Server:
http://blogs.msdn.com/saponsqlserver

Note that the SAP OSS Notes and SAP Product Support Matrix are only available to registered customers of SAP AG.

For more information:
http://www.microsoft.com/sql/

Did this paper help you? Please give us your feedback. On a scale of 1 (poor) to 5 (excellent), how would you rate this paper?