# logo-sql08.gif

# Consolidation Using SQL Server 2008

SQL Server Technical Article

**Writer:** Allan Hirt, Megahirtz LLC (allan@sqlha.com)

**Technical Reviewers:** Lindsey Allen, Madhan Arumugam, Ben DeBow, Sung Hsueh, Rebecca Laszlo, Claude Lorenson, Prem Mehra, Mark Pohto, Sambit Samal, and Buck Woody

**Published:** October 2009

**Applies to:** SQL Server 2008

**Summary:** This white paper is written for a technical audience to understand the various aspects and considerations for planning consolidation with SQL Server 2008 as well as administering the consolidated environment.

# Copyright

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, email 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, 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

[1 Introduction 1](#_Toc241980628)

[2 Consolidation Drivers 1](#_Toc241980629)

[3 Pre-Planning Steps 3](#_Toc241980630)

[3.1 Putting Together the Consolidation Team 3](#_Toc241980631)

[3.2 Creating the Guiding Principles 4](#_Toc241980632)

[3.3 Controlling Costs 5](#_Toc241980633)

[4 Planning for Consolidation 6](#_Toc241980634)

[4.1 Discovering the Environment 6](#_Toc241980635)

[4.1.1 Tools for Discovery 7](#_Toc241980636)

[4.1.2 Information to Gather 8](#_Toc241980637)

[4.2 Analyzing the Discovered Information 14](#_Toc241980638)

[4.3 Specific Considerations for SQL Server Consolidation 15](#_Toc241980639)

[4.3.1 Types of Consolidation 15](#_Toc241980640)

[4.3.2 Applications 21](#_Toc241980641)

[4.3.3 Choosing the Edition and Version of SQL Server for Consolidation 23](#_Toc241980642)

[4.3.5 Standardizing the Environment 25](#_Toc241980643)

[4.3.6 High Availability for Consolidated SQL Server Environments 25](#_Toc241980644)

[4.3.7 Capacity and Performance Planning for Consolidated SQL Server Environments 27](#_Toc241980645)

[4.3.8 Security 34](#_Toc241980646)

[4.3.9 Determining Database Consolidation Ratios 34](#_Toc241980647)

[4.3.10 Upgrading and Consolidating at the Same Time 35](#_Toc241980648)

[4.3.11 Other SQL Server Components 35](#_Toc241980649)

[4.4 Technologies for Consolidating SQL Server 36](#_Toc241980650)

[4.4.1 Backup and Restore 36](#_Toc241980651)

[4.4.2 Log Shipping 37](#_Toc241980652)

[4.4.3 Database Mirroring 37](#_Toc241980653)

[4.4.4 Detach and Attach 38](#_Toc241980654)

[4.4.5 Moving Objects Other Than Databases 38](#_Toc241980655)

[4.4.6 Resources for Moving Objects 39](#_Toc241980656)

[4.5 Putting the Plan Together 39](#_Toc241980657)

[4.5.1 Documenting the Technical Aspects of the Plan 39](#_Toc241980658)

[4.5.2 Staffing the Consolidation Effort 39](#_Toc241980659)

[4.5.3 Determining the Consolidation Timeline 40](#_Toc241980660)

[4.5.4 Devising the Communication Strategy 41](#_Toc241980661)

[4.6 Testing 41](#_Toc241980662)

[5 Administering a Consolidated SQL Server Environment 42](#_Toc241980663)

[5.1 Deployment Standardization 42](#_Toc241980664)

[5.2 Change Management 43](#_Toc241980665)

[5.3 Standard Administration 43](#_Toc241980666)

[5.3.1 Database Backups 43](#_Toc241980667)

[5.3.2 Index Rebuilds/DBCCs/Proactive Maintenance 44](#_Toc241980668)

[5.3.3 Monitoring 44](#_Toc241980669)

[5.3.4 Patching Servers and Instances in a Consolidated SQL Server Environment 45](#_Toc241980670)

[5.4 Constraining Resources 45](#_Toc241980671)

[5.4.1 Proper Configuration 46](#_Toc241980672)

[5.4.2 SQL Server 2008 Resource Governor 46](#_Toc241980673)

[5.4.3 Windows System Resource Manager 47](#_Toc241980674)

[5.5 Chargeback 48](#_Toc241980675)

[6 Conclusion 49](#_Toc241980676)

[7 Links for More Information 49](#_Toc241980677)

# 1 Introduction

Many companies are considering or have already implemented consolidation of computing resources, including Microsoft® SQL Server® instances and databases, in their organization. A consolidation effort is a complex task that requires information, a detailed plan and timeline for success, and a strategy for administering the consolidated environment. This white paper walks through the journey of gathering and analyzing the information to base all planning and implementation decisions on; how to plan, architect, and implement consolidation; and finally, the considerations for administering a consolidated SQL Server environment.

This white paper is based on the [Microsoft Operations Framework](http://technet.microsoft.com/en-us/library/cc506049.aspx) (MOF). MOF is an iterative approach to the IT lifecycle, and it is designed to bring together both the technical and nontechnical sides of implementation and administration.

**Note:** This white paper is written for resources within a company who are responsible for planning and executing SQL Server consolidation, as well as administering the consolidated solution. This list of people will vary from company to company, but at a minimum includes the DBAs and those responsible for storage, servers, networking, and applications. The white paper is not intended to be an overview of SQL Server consolidation. For a less technical, higher level paper on SQL Server consolidation, its drivers, and benefits, see the white paper [Consolidation with SQL Server 2008](http://download.microsoft.com/download/a/c/d/acd8e043-d69b-4f09-bc9e-4168b65aaa71/SQL2008SrvConsol.doc).

# 2 Consolidation Drivers

The main driver for any consolidation effort is *cost.* As budgets are being cut, IT is being forced to do more with less and maximize resources. The technical drivers for administrators are *consistency* and *control*. This white paper focuses mainly on consistency and control, but even if the person reading this is a DBA, he or she must realize that cost must factor into any architectural and technical decisions that are made for the consolidated environment. The end result of consolidation has two main outcomes:

* A SQL Server utility model that is agile, standardized, and easier to manage
* The costs of the tangible (air conditioning, power, servers, and so on) and the slightly harder to quantify (administrative costs, time it takes to administer, and so on) are reduced.

Table 1 presents a summary of the most popular drivers for SQL Server consolidation and which major categories they fall under. Following Table 1 is a more detailed explanation of what those drivers mean as related to SQL Server consolidation.

*Table 1. Consolidation drivers matched to their proper category*

|  |  |  |  |
| --- | --- | --- | --- |
| **Driver** | **Consistency** | **Control** | **Cost** |
| SQL Server sprawl | ■ | ■ | ■ |
| Lack of deployment standards | ■ | ■ | ■ |
| Capacity management |  | ■ | ■ |
| Licensing |  |  | ■ |
| Inconsistent administration | ■ | ■ | ■ |
| High availability |  | ■ |  |
| Virtualization | ■ | ■ | ■ |
| Supportability | ■ | ■ | ■ |
| Green IT |  | ■ | ■ |
| Poor server utilization | ■ | ■ | ■ |

SQL Server sprawl is a phenomenon that many customers face; it is the situation where SQL Server has blossomed in the environment and may have been installed as part of applications or other uses (such as development or test instances) to the point that it is not entirely known how much SQL Server is actually deployed. SQL Server sprawl results in problems such as issues due to varying standards of deployment and administration for each server, instance, or database. SQL Server sprawl also means that in some cases, a server hosting a SQL Server instance, an instance itself, or even a database has not been designed and/or deployed under the auspices of the DBAs. This leads to myriad problems that consolidation resolves, including:

* Supportability challenges, because DBA staff members may not be aware of the existence of an instance and its databases until they are asked to provide services to resolve an issue.
* Increased cost of administration due to having to support not only multiple versions of SQL Server, but also older versions of software and hardware platforms that are out of support from their vendors and pose significant upgrade challenges.
* The deployments may not be license-compliant for all the software that is in use, and the cost of becoming legal can potentially be quite expensive.
* Systems and applications that may have started out small and non-mission-critical may have changed over time to become central to the business, but the deployment of those servers is the same as when they were first deployed, so the hardware and architecture does not meet the availability and/or performance needs of the business.
* Poor server utilization and space usage in the data center. Large numbers of older servers consume space and other resources that may be needed for newer servers. The servers themselves may be either sitting nearly idle or completely maximized as a result of poor capacity planning or the lack of a hardware refresh during the lifetime of that deployment.

Green IT is also a growing trend. Green IT means a company is not only using fewer resources such as electricity to reduce costs through methods like consolidation, but also helping the environment as well as enhancing the image of the company in the process. A positive corporate image helps a company’s bottom line.

Finally, advances with server-based virtualization and the associated tools have accelerated the drive to consolidate over the past few years. Because virtualization is a specific technical topic related to SQL Server consolidation, it is addressed in more detail later in the section 4.3.1, “Types of Consolidation”.

# 3 Pre-Planning Steps

Before you can consolidate existing databases and instances, there are a number of tasks that must be performed. While these tasks are not necessarily technical in nature, such as putting the team together who will be responsible for consolidation, failure to complete these tasks will impact the success of the technical portions of a consolidation project.

## 3.1 Putting Together the Consolidation Team

Before any planning or consolidation work can be done, the right team must be in place. The makeup of the team will represent more than just the SQL Server DBAs; it will be composed of senior-level technical and nontechnical resources, and ideally, it will have representation from all groups that will be impacted by SQL Server consolidation. These groups include: the DBAs, all application owners whose application databases are being targeted for consolidation, security, network, Windows®, storage, and any other group which has direct or indirect responsibility for the day-to-day care and feeding of the SQL Server environment. SQL Server consolidation is not just a DBA effort. As the project goes on over time, it is important to note that the members of the team may change and the size of the team may even shrink. The team should have the right members for the right phases of the project. For example, in the early stages, there may be more senior members involved when the team is devising things like the guiding principles (see section 3.2 for more details on guiding principles), but some of those people may not need to be involved on a regular basis (if at all) after that work is complete.

The most important member of the consolidation team is the *executive sponsor*. There may even be more than one sponsor if the initiative has multiple stakeholders. Often, there may be a minimum of three sponsors: someone from the application or business side, someone from the technical/implementation side, and someone from operations. The executive sponsor is someone higher up in the organization, and that person might not be technical. His or her main responsibilities are to shepherd the project, own the overall vision and scope, and be the consolidation effort’s champion to upper management, who will deal with any political aspects. He or she is also a major force in securing the budget needed for the consolidation project.

Working closely with the executive sponsor will be the *project manager*. The project manager is to oversee the day-to-day operations of the consolidation project. This person should be a more senior who is technical, but also has the skills to interface with management. The project manager’s responsibilities include, but are not limited to: translating the technical requirements and needs into something management can understand, and translating the needs of the company to the technical people responsible for the actual consolidation planning and work and managing the timeline and all tasks associated with the timeline. Ultimately, from the technical side, the project manager owns the success or failure of consolidation.

The other members of the multidisciplinary team are most likely senior-level staff responsible for specific aspects of the consolidation effort. The problem with these members is that they may have day-to-day responsibilities beyond just being a member of the team responsible for consolidation. Consolidation is a serious investment by any company, and as such, the resources who will be tasked with leading the effort must have dedicated time to handle this task properly.

A big part of what the team must do is make those affected by consolidation feel comfortable with what will be going on over the coming months, and minimizing the risks and exposure to the business during the actual consolidation. Mitigating risk is a constant theme in IT and a key tenet of MOF. Even though the team may be empowered to carry out the task of consolidation, there will be a lot of worrying by many factions in the company (specifically, the business and application owners) until success is achieved and they realize that there has been no change to their daily workflow.

## 3.2 Creating the Guiding Principles

After the initial team is in place, their first task is to document the principles that will govern the consolidation effort. These are known as the *guiding principles*.

The guiding principles encompass both technical and nontechnical aspects of a consolidation effort, and they should be relatively high level, but have enough detail to direct the project in the right direction. At the onset of the project, there may not be enough information to make definitive statements, so the initial guiding principles may represent the desired state. As more information is gathered and the project proceeds, the guiding principles may need to be reevaluated if circumstances change; the document must be an evolving document, but changes should only be approved if they are necessary. Some examples guiding principles are listed here:

* Consolidation will happen in multiple stages, and it will be completed over the course of two years.
* The target platform will be SQL Server 2008 R2 running on Windows Server® 2008 R2. All applications must support this goal, and the teams responsible for the applications must do the work to ensure that there will be no major problems post-consolidation.
* Only applications and servers with less than 30 percent utilization will be targeted for consolidation.
* All applications based on SQL Server 2000 (and earlier) are consolidation candidates.
* Post-consolidation, all databases for new in-house developed applications must be designed with and deployed into the consolidated environment. If this is not possible, the application must justify why it cannot happen and will pay for any hardware that must be deployed. The same applies to vendor applications.
* The company will save 30 percent in physical costs for running the physical servers.
* All administrators must receive proper training for the new consolidated environment.
* The company must realize the return on the investment (ROI) made into consolidation within 3 years.
* Mission-critical applications will not be considered for the initial rounds of consolidation.
* The company must reduce its use of physical servers over the next year by 25 percent to reduce cost and management issues.
* All DBAs will be sent to SQL Server 2008 training over the next 12 months to prepare for the consolidated state.
* The company requires vendors to ensure that all aspects of an application are in a supportable state.
* Most Data Transformation Services (DTS) packages must be converted to SQL Server Integration Services.
* All older applications and their servers containing instances and databases that do not support SQL Server 2008 will be virtualized.
* Consolidation must support the company’s Green IT initiative.

## 3.3 Controlling Costs

Any project has a budget associated with it. The budget may or may not be flexible. Although the DBA executing a backup and restore for one of the applications may not be concerned about the cost of consolidation, the lead who is part of the planning must be aware of the impact that decisions play into the cost of not only the project, but also the ownership of the architecture going forward.

The budget of the actual consolidation project is based on the following factors:

* Software and licensing

Besides ensuring that the required licenses for things such as SQL Server, Windows, and any applications are made based on the target consolidation platform, additional tools may need to be purchased to assist with the consolidation. These costs must be in the budget for consolidation, because the goal is to do it in the fastest, safest way possible. For example, a company has hundreds of DTS packages, some complex. Assuming the target platform is SQL Server 2008, although there is a command line utility to run the DTS packages, the best thing to do is to convert them into something that will work better with SQL Server 2008 – an SSIS package. However, the cost in terms of time to complete that task can be measured in possibly man-months or years. A company named Pragmatic Works makes a program called DTS xChange to convert DTS packages (even complex ones) to SSIS. While the software will cost money, and it may not convert everything, if it converts 70 – 80 percent with no manual intervention, its cost is cheaper than hiring an army of consultants. Some work, whether done by consultants or experts in-house, may need to be done on the remaining 20 – 30 percent, but the workload is now reasonable and can be done to meet the consolidation timeframes.

* Hardware

Consolidation may require the purchase of new hardware; it should not be assumed that all existing hardware will be reused. How much and what kind of hardware that will be needed can only be determined after the data is analyzed during the discovery and analysis process.

* Staff augmentation

The resources who are assigned to the consolidation project may have other responsibilities, so it may be necessary to augment the staff to ensure that the project keeps going even if someone gets pulled away to deal with some current production issues. Sometimes the right expertise (especially when it comes to third-party applications and their implementations) may no longer exist within the company. Within the budget, some money should be allocated to staff augmentation. How large this amount is will be is determined by how confident the company is in their resources. Hire where needed. A good example is expertise in SQL Server 2008: Assuming that SQL Server 2008 is the desired platform to be used, bringing a consultant in-house who can not only assist in all phases of the consolidation, but also mentor the staff so they can administer it afterwards may be a worthwhile use of a portion of the budget.

From an ongoing cost perspective, a baseline cost of the environment must be documented and analyzed, if the baseline cost is not known already. The best time to do this is while you are gathering information to make consolidation decisions. The reason is that the business will want to realize the costs spent on consolidation, and show the ROI. The ROI should be achieved in a fairly short amount of time – on average 1- 3 years – otherwise the end may not justify the means. For example, it is easy to measure the electricity costs associated with servers, either through the overall bill to the data center or through specialized pieces of hardware devised for such a task. After consolidation, using those same tools, measure power consumption. Is it lower? If so, by how much? The numbers derived will show whether the desired target has been achieved.

# 4 Planning for Consolidation

The technical portion for SQL Server consolidation starts with planning, which maps to the *Plan Phase* of MOF. This section will walk through the various tasks associated with planning a consolidated SQL Server environment and how to accomplish them.

## 4.1 Discovering the Environment

Information is the cornerstone of a consolidation effort; without it, it is difficult to make the decisions needed for the architecture and deployment of the new environment, including how much money for items will need to be accounted for in the budget. The information needed is a mixture of straightforward configuration and basics, in addition to performance data about the server, instance, and databases. Combining the configuration information with the performance data paints a holistic view of the current SQL Server deployments.

Some reading this document may already have configuration documentation as well as performance information about their deployments; if the existing information meets the needs for planning consolidation, do not gather additional information that is not needed. However, most will have varying degrees of documentation, or worse, none at all. This section is geared for the audience with less documentation about their configurations.

The goal of discovery is not to gather every bit of information or monitor every performance counter; make no mistake, there will be quite a bit of information needed. The information gathered should tell something about the environment that enables those doing the planning to make informed decisions. It may not be possible to get information for every server, instance, or database, but the objective should be to gather as much of what is required as possible, note where it was impossible to get, and move forward. Depending on the size of the environment, discovery could take a matter of days, or possibly weeks, so plan the timeline accordingly. Most of the information will only need to be gathered once; performance information, which needs more data points, is addressed in section 4.1.2, “Information to Gather”.

**Tip:** The performance information will be used as part of the capacity management planning (see Section 4.3.6) that will occur. Whether there is a physical or virtual component involved (see Section 4.1 for the different types of consolidation), the effort and information needed is going be the same, because the same capacity that would be needed for a physical server will still be needed for a virtualized server.

During the discovery process there may be roadblocks that appear. For example, a new SQL Server instance that the DBAs did not know about before is discovered, and the DBAs do not have a user or password to access the server. Time may be spent on such tasks as tracking down who owns the server, what application uses it, and getting the proper credentials to do further investigation and analysis. This is another reason why discovery can take longer than expected.

**Important:** Do not wait for every bit of information to come in before any analysis is done; even if most of the information is gathered during the discovery effort, other important bits of information may appear later during planning and will need to be taken into account. The information gathered initially helps to establish the first cut at what the consolidation candidates will look like.

### 4.1.1 Tools for Discovery

One of the challenging aspects of discovery is that there are lots of data points to gather, but there are only a handful of tools, and there are an even fewer number of tools that can assist in pulling all of the information together to present a cohesive view of the current SQL Server environment. Most discovery is done using a few different tools or methods, and then putting that information into a common format (such as a Microsoft Office Excel® spreadsheet or even a SQL Server database) for analysis.

Only a handful of tools from either Microsoft or third parties can perform the bulk of the work of discovery and some of the information gathering. Each tool usually can handle part of the work, but not the entire job. Some custom scripting or manual work may need to be done in addition to using tools and utilities if there is no other way to get the information. Some tools and methods that can be used are shown in Table 2.

*Table 2. Tools and methods to gather information*

|  |  |
| --- | --- |
| **Tool/method** | **Comments** |
| SQL Server system tables, functions, stored procedures, and dynamic management views (DMVs) | A wealth of information can be queried directly from SQL Server, including configuration and performance information. |
| Performance Monitor/System Monitor | For gathering performance-related information, if there is no centralized tool or it is not being used, the built-in tool in Windows can help gather the information. |
| The Windows PowerShell™ command-line interface and Windows Management Instrumentation (WMI) | Operating-system-level scripting can gather information about systems as well as SQL Server. |
| The [Microsoft Assessment and Planning Toolkit](http://technet.microsoft.com/en-us/library/bb977556.aspx) (MAP) | MAP is a good way to interrogate servers to see whether they are running any form of SQL Server, and to detect unknown installations. This tool will most likely be the starting point for most consolidation efforts. |
| Centralized monitoring tools such as Microsoft System Center Operations Manager, CA’s Unicenter, IBM’s Tivoli, and HP’s OpenView | Enterprise monitoring tools are generally used for monitoring health and/or performance of most servers in the environment. If the data exists, the DBAs should work with the monitoring administrators to get the data extracted from these systems instead of having it to capture it themselves. |
| Inventory software | If there are third-party or internal utilities used for inventory management, the information contained in those systems can be leveraged. As with centralized monitoring tools, the DBAs may need to work with other groups to gain access to the data. |

One tool that should be in your arsenal no matter what is MAP. While MAP may not gather everything, it is a quick way to see what is deployed in an environment to kick off the information-gathering process.

A lot of the information you need can be derived via SQL Server queries (system tables from versions of SQL Server before SQL Server 2005, and dynamic management views in SQL Server 2005 and later) as well as standard performance counters (many of which can also be gathered via SQL Server queries). Many of the items are one-time grabs, but when it comes to performance-related items, things are a bit different. To understand the performance profile of an application and its database(s), multiple data points are needed.

**Tip:** Where possible, DBAs should partner with the groups responsible for the other aspects of the physical environment and use existing monitoring tools to get the information needed if it is available. Do not reinvent the wheel. The best-case scenario is that information has been collected, stored, and available for years.

### 4.1.2 Information to Gather

This section covers the general items that should be gathered during the discovery phase. For more information about application information that is needed, see section 4.3.1, “The Application”.

**Tip:** Some of the information listed may not apply to every environment; only gather information that is needed, so add or remove information from these lists. They are meant to be a guideline as to what should be documented.

#### Physical Server/Windows Information

Table 3 shows information that should be gathered about the Windows servers and why each bit is relevant.

*Table 3. Windows and server information to gather*

|  |  |
| --- | --- |
| **Attribute** | **Comments** |
| Server name | The server name will be referenced throughout planning and implementation. |
| Operating system version, service pack level, and build number, 32-bit or 64-bit | The version of the server will help quantify not only what versions of Windows are deployed, but also any specific considerations that may need to be taken into account during consolidation. |
| Main purpose for the server | It is important to note what the server is actually being used for. For example, is it a development, test, staging, or QA server, and what application is using it? |
| If possible, list of all updates applied | Similar to knowing the version of the Windows, there may have been specific updates applied to resolve a particular issue that will need to be applied after consolidation. Updates include, but are not limited to, hotfixes, service packs, and other critical updates (including security). |
| Domain affiliation | This will tell what domain(s) the server belongs to. |
| Full networking/TCP/IP information | Networking information is important for configuration. |
| Memory configuration | This is the total amount of memory in the server. |
| Processor type and speed | This documents the current processor to map to a target platform. |
| Drive configuration | Knowing how the disks are configured is important, because if nothing is known about the server, drive configuration provides a map of where to get started looking for databases and backups. It also helps during the implementation plan to know the locations of where things are located to ensure they are migrated to the new environment. |
| List of all drivers on the system, including versions | This will document what drivers are in use, and it may impact the configuration and supportability. |
| Stand-alone or cluster node | Tells whether the server is part of a Windows failover cluster or not. |
| List of all software (including SQL Server) and utilities/administration tools (such as antivirus) installed | This is an inventory of what is configured besides Windows on the server. |
| Hardware vendor, physical server model number, serial number | Documents the vendor-specific information. |
| List of components and their vendor (such as the HBA card) | Documents all of the hardware inside the server. |
| Current support status for Windows | Documents whether or not the server is currently supported by Microsoft. |
| Current support status for the hardware and other software | Documents whether or not the server is currently supported by vendors other than Microsoft. |
| Any special configuration notes | Document any other configuration information specific to a particular server that may impact the plan or the consolidation. |

#### SQL Server Information

Table 4 is a sample list of items to record when documenting an installation of SQL Server or SQL Server Analysis Services.

*Table 4. SQL Server information to gather*

|  |  |
| --- | --- |
| **Attribute** | **Comments** |
| Instance name | This name will be a major reference point for the consolidation effort, whether the instance is being consolidated, or it is just the location of specific databases. If this is SQL Server 2000 or later, indicate whether it is the default instance or a named instance. |
| SQL Server major version, service pack level, and build number | This will help determine the best way to migrate, upgrade, or consolidate, because those tasks depend on the starting version. This information also helps quantify versions used in the environment. |
| List of all instance-level configuration parameters and current values (**sp\_configure**, mixed mode or Windows authentication, collation, and so on) | Helps to determine incompatibilities for combining objects and databases from multiple instances. |
| List of all databases and each one’s configuration parameters (recovery model, compatibility mode, file configuration and sizes, free space, collation, and so on) | Gets all database-related information; similar to instance-level information. This information helps quantify what is in the environment and the best paths to consolidation. |
| List of all instance-level logins and their rights and permissions; distinguish between SQL Server logins and Windows logins | Supports analysis of the security of the instance. |
| List of all database-level users, their rights and permissions, and what login they map to | Supports analysis of the security of the databases. |
| List of any custom objects/databases in the instance (for example, special DBA stored procedures) | Documents any nonstandard objects so they can be migrated to the target environment. |
| List of any and all DTS or Integration Services packages and their purposes | This will document and quantify Integration Services and DTS, and it will help the assessment of how much work will be needed to possibly convert them. |
| List of all jobs and their details, including job history | This will form the basis for any administration and maintenance that will be configured post-consolidation, if SQL Server Agent jobs are being used. Jobs are also used for nonadministrative tasks, so having them documented is crucial. |
| List of all maintenance plans and their details | This will form the basis for any administration and maintenance that will be configured post-consolidation, if maintenance plans are being used. |
| List of all special configurations (database mirroring, log shipping, replication, linked servers, and so on) | To ensure that configurations are re-created (if necessary) post-consolidation, they must be documented. |
| List of any third-party utilities, their versions, and support contact information | This is an inventory of any SQL Server-related tools used for administration, such as backup compression utilities. |
| Clustered or stand-alone; if clustered, name of the Windows cluster it is joined to as well as drive configuration and list of all nodes that can own the SQL Server resource | Information specific to a failover clustering instance. |
| Server name (stand-alone) | This is the name of the underlying Windows server if the instance is stand-alone. |
| Licensing model for SQL Server and current status with Microsoft | This will document whether per-processor or a server license plus individual client or device licenses are in use. If it is determined that an instance is not properly licensed, consolidation is a time to correct that issue to remain legal. |
| Current support status for SQL Server | This will document whether there is a current support contract for SQL Server and what its terms are. This is important for a variety of reasons, not the least of which is you may need support during the consolidation effort. It will also help determine the level of support needed for the new environment. |
| Any special configuration notes | This will document any other configuration information specific to that particular instance of SQL Server that may impact the plan or the consolidation. |

#### Application Information

Table 5 is a sample list of items to record when documenting an application. This information may not be discovered during the initial discovery phase, but from the DBA’s perspective, it provides the relevant information to be able make the proper decisions about the consolidation architecture and the plan to move the related application databases.

*Table 5. Application information to gather*

|  |  |
| --- | --- |
| **Attribute** | **Comments** |
| Application name | Like an instance of SQL Server, this application name will be referenced throughout the consolidation effort. |
| What the application does (summary) | It always helps to know what the application is used for and, at a high level, how it works. |
| Application version | This will determine a few things, for example, supportability against the target platform. |
| In-house developed or purchased | Was this application made by a third-party vendor or coded by developers? If the application was coded in-house, note the status of the code, who owns it, and where it is stored in the event it would need to be updated for SQL Server 2008. |
| Performance and availability service-level agreements (SLAs) | These will help in determining what can be mixed together, because mixed SLAs can be a bad thing in a consolidated environment depending on where databases and applications are consolidated. |
| Version(s) of SQL Server the deployed application version supports | For the currently deployed version (not one that is yet to be deployed or not purchased), what versions of SQL Server does it support? |
| Application owner with contact information | One of the most important things to document. |
| Type of workload (OLTP, DSS, and so on) for this application | The application workload is important when planning consolidation, because it will determine what can and cannot be combined. It is also important to note the **tempdb** usage of the application. |
| List of servers the application uses (application, database, web, remote, and so on) | This is an inventory of what is configured besides Windows on the server. |
| List of SQL Server databases the application uses along with the instance where each database is located | Allows mapping of applications to SQL Server instances and databases. |
| Any known special requirements for the application deployment (such as patch levels for Windows and/or SQL Server) | Documents any specific application requirements that may be needed in the target environment. |
| Any known restrictions that may affect the timing of the actual consolidation of the database(s) and/or instance(s) for that application | For example, different departments may have short windows for doing the work. An accounting department may have month-end, quarterly, or year-end closing activities that prevent any work from being done during specific dates. This will impact the schedule. |
| Current support status for the application | Documents whether or not the server is currently supported by vendors (if third-party software). |
| SQL Server features used by the application | Lists any features such as replication that the application leverages. |
| Any special configuration notes | Documents any other configuration information specific to that particular application that may impact the plan or the consolidation. |

#### Performance Information

After the basic information about the installations is gathered, performance information about the hardware, Windows, and SQL Server must be obtained. Performance information is the largest source of information that relates to the capacity planning that will occur. As noted earlier, whether the ultimate end state is virtual machines or consolidation on physical hardware, the sizing exercise is the same; a database or instance will need the same amount of resources.

Because there are hundreds of counters that can be gathered, Table 6 lists only the most common ones to gather for analysis. You can add to this list as appropriate for your organization, but only capture the counters that will actually be used during analysis; gathering information for information’s sake is not the goal of this exercise. Also realize that the amount of information goes up significantly with additional counters and the frequency with which they are captured.

Performance information must be gathered at different times, to provide a complete picture of the application as well as the server. Counters should be gathered when systems are at relative rest, on slow days, on busy days, and everything in between for reasonable time periods (for example, an hour each capture). Unlike the Windows and SQL Server information documented previously, if a one-time grab of information is performed, the information only represents that server at that particular moment. It is especially important to make sure that the server is monitored during the busy times, because that demonstrates the system under load. A physical or virtual server needs to be configured based on maximum usage, and if the wrong performance numbers are gathered, the sizing could be wrong and cause problems sooner rather than later in production.

*Table 6. Performance information to gather*

|  |  |  |
| --- | --- | --- |
| **Performance object** | **Counter** | **Comments** |
| **Memory** | Pages/sec | Shows whether memory is paging to disk |
| **Memory** | Available Mbytes | Displays the amount of memory |
| **Network Interface** | Packets received/sec | Per each network card, especially ones being used for SQL Server |
| **Network Interface** | Packets sent/sec |
| **Network Interface** | Bytes received/sec |
| **Network interface** | Bytes sent/sec |
| **Physical Disk** | %Disk Read Time | Configure per disk as well as \_Total; will show how I/O looks at a physical level for the disk configuration |
| **Physical Disk** | %Disk Write Time |
| **Physical Disk** | %Idle time |
| **Physical Disk** | Avg. Disk Bytes/Read |
| **Physical Disk** | Avg. Disk Bytes/Write |
| **Physical Disk** | Avg. Disk Read Queue Length |
| **Physical Disk** | Avg. Disk Write Queue Length |
| **Physical Disk** | Avg. Disk sec/Read |
| **Physical Disk** | Avg. Disk sec/Write |
| **Physical Disk** | Split IO/sec |
| **Process** | % Processor Time | Per process (focus on the SQL Server processes, but you may want to gather all processes); gives information about the processes running on the server. |
| **Process** | IO Read Bytes |
| **Process** | IO Read Operations |
| **Process** | IO Write Bytes |
| **Process** | IO Write Operations |
| **Process** | Page Faults/sec |
| **Process** | Thread count |
| **Process** | Working Set |
| **Processor** | % Processor Time | \_Total and per individual processor (especially with multiple cores); shows the performance of the processors |
| **SQL Server:Databases (2000,2005,2008)\*** | Transactions/sec | Select per database; tells how many transactions are happening in the database at that given time |
| **SQL Server:General Statistics (2000)** | User Connections | Gets the total amount of user connections in SQL Server |
| **SQL Server:Memory Manager (2000, 2005, 2008)** | Connection Memory(KB) | Displays the amount of memory consumed by connections |
| **SQL Server:General Statistics (2008)** | Active Temp Tables | Indication of **tempdb** usage (but not necessarily the only indicator) |
| **SQL Server:Memory Manager (2000, 2005, 2008)** | Target Server Memory (KB) | Displays the total amount of memory that can be used by the instance of SQL Server |
| **SQL Server:Memory Manager (2000, 2005, 2008)** | Total Server Memory (KB) | Displays the amount of dynamic memory being used by the instance of SQL Server |

\* Year numbers in this column indicate the version of SQL Server. For example, “2000” stands for SQL Server 2000.

**Tip:** Using SQL Server system tables, stored procedures, and dynamic management views (DMVs) can also gather further performance and other information about configurations. For example, the DMV **sys.dm\_io\_virtual\_file\_stats** (in SQL Server 2005 and SQL Server 2008) and the function **fn\_virtualfilestats** (in SQL Server 2000) can be used to gather disk information such as the total number of read and write I/Os used by a database since it was first configured. For more information, including a full list of DMVs that can be used, see [Dynamic Management Views and Functions (Transact-SQL)](http://msdn.microsoft.com/en-us/library/ms188754.aspx) in SQL Server Books Online. The link is to the SQL Server 2008 topic, but the topic contains a link to the SQL Server 2005 equivalent.

## 4.2 Analyzing the Discovered Information

Assuming that the gathering of information was successful, there may be quite a bit of data. It can be a daunting task to sift through and make sense of it. To assist in that process, the first thing that should be done is to put it into a format that makes sense. That may be a Word document, an Excel spreadsheet, or even a SQL Server database that can be queried and reported on. Group like information (for example: all disk related information in the same section in a Word document or on the same worksheet in Excel).

After the data is in a digestible format, the analysis work can begin. It is often the more static data that will control what and how things get consolidated, but the performance data also plays an important part when it comes to capacity planning. Seeing all of the data in one place can really make certain things stand out. For example, if 300 instances of SQL Server are discovered and documented, it will be easy to tell what mix of server and operating system versions are currently deployed. There may be some surprises as to how many versions are in play – with everything from SQL Server 6.5 through SQL Server 2008. Looking at that data leads to a fairly obvious outcome: The older ones may be great candidates for consolidation and further analysis. Tying that information to applications may also show and prove that some applications and hardware can be decommissioned since they may not be in use.

Another aspect of the basic data analysis is to identify duplicates, conflicts, or potential problems in the environment. For example, a database name of *mydb* is found on 10 different servers; which one is the one used in production by its associated application? Are some not used at all? The DBA will most likely need to ask others outside of the DBAs to gather this information, but the goal is to determine what the “database of record” for production is, its importance, and how it should ultimately be handled.

Performance data analysis is somewhat easier to look at. For example, if the processor utilization for a particular server averages 60 percent, with consistent spikes of 92 percent, that hardware may not be sized properly for the workload. I/O and memory numbers and information will tell similar stories. This information will form the basis of what can be consolidated together from a performance standpoint. Pay particular attention to I/O numbers for each database. What should emerge from the analysis are usage patterns for servers, instances, and databases.

With the data, another thing that should be established is patterns. If data is available dating back to when an application or database was first deployed, it makes a world of difference. One of the things that must be attempted to be established is a growth pattern. Later, as part of the sizing, someone is going to have to figure out server and disk capacity and sizing for the next few years as it relates to a database. If the only data point is where things are now, with no idea of how the database grew to this point, it is much harder to estimate where things will be in a few years. The biggest danger is that lack of information leads to guessing, and guessing can potentially lead to having an undersized system, or even put things back where they are now: The system is too large and underutilized. Consolidation should right-size the environment.

**Tip:** The best-case scenario for information would be if performance data was gathered from the day the system went into production and that performance data is still available. One thing that needs to be determined is growth – growth not only of usage of resources, but things such as database size. For example, if a database is growing 10 percent per month, that would provide definitive guidance on how to configure the sizing (but not the I/O) of the data and log files.

Analysis is an iterative approach, and it is more than just analyzing performance information; to be able to put a plan together for consolidation, the static and performance information that may not seem related must be linked. Basing decisions purely on the technical aspects such as processor, memory, and disk I/O can be misleading, because they do not tell you anything about service level agreements (SLAs), expected performance requirements, security, or any number of other aspects.

What should come out of the initial analysis is a list that represents the first cut of potential consolidation candidates, whether they are whole instances, physical servers, or even databases. This list will now be the basis for all work going forward on the consolidation effort. The list will be revised and refined until it becomes the definitive list for consolidation.

## 4.3 Specific Considerations for SQL Server Consolidation

There are many technical factors that influence both the plan and the architecture for a consolidated SQL Server environment. This section discusses those factors.

### 4.3.1 Types of Consolidation

When it comes to consolidating SQL Server, a few types of consolidation can occur to achieve the desired end state:

* Multiple schemas in a single database
* Multiple databases under a single instance of SQL Server
* Multiple instances on a single server (or cluster)
* Virtualized SQL Server deployments

These four options are listed in order of density. The general rule for SQL Server consolidation is that to achieve a higher density of consolidation with the lowest relative cost, isolation is sacrificed. Lower density with more isolation generally has higher operating costs.

Collapsing multiple databases into a single database has large tradeoffs that may not be acceptable to many DBAs, with the biggest two being security and a need to alter applications. There may be other problems, such as object naming conflicts and performance issues. Schema-level consolidation is an option that should be used carefully.

Historically, the next two types of consolidation have gone hand-in-hand. The traditional method for consolidation has been placing multiple databases under a single instance of SQL Server, and then configuring multiple instances on a single server (or cluster). Both have their individual tradeoffs, which are related. When multiple databases are combined under a single instance of SQL Server, usage is driven up, and isolation can occur at the database level, but there still is the potential for some conflicts around security and other areas, such as collation. Combining multiple instances on a single server maximizes utilization at the instance/server level and reduces the need to have multiple servers where there is enough capacity to run the workloads that may have individually run on their own underutilized servers.

Virtualization is a relatively new method of consolidating SQL Server, and until recently, it was mainly used for nondatabase servers. Virtualization provides complete isolation, because the virtualized server acts just like a physical server, and it has no dependencies on other virtualized servers. However, the virtualized servers all run under a single host (or multiple hosts with multiple virtualized servers), so underlying resources will be shared even though the virtualized servers think they are fully independent and have a full bandwidth of resources.

The rest of this section focuses on the comparison between physical consolidation using multiple instances or virtualization, because those are the two most common forms of consolidation with SQL Server 2008.

#### 4.3.1.1 Physical Consolidation Using Multiple Instances

SQL Server 2000 was the first version that featured the ability to use multiple instances. For more information about instances, see [Instance Configuration](http://msdn.microsoft.com/en-us/library/ms143531.aspx) in SQL Server 2008 Books Online. SQL Server 2008 supports up to 50 instances on a stand-alone server, and up to 25 in a clustered configuration. How many instances can actually be deployed is a function of capacity management. Many deployments today consume only 10, 20, or 30 percent of a server’s resources, so they could stand to be optimized and possibly combined, giving better overall hardware utilization. Multiple instances can enable this. Also, if servers are being over utilized, rearchitecting as well as consolidating on newer, faster servers can optimize utilization for underperformers.

An instance of SQL Server is mostly isolated: It uses its own set of binaries, system databases, memory, and so on. That means that multiple databases can coexist in their respective instances and not really know about each other. This is one of the biggest benefits of instances. There are some shared components, such as the connectivity layer and the SQL Server tools, but everything else at a physical server level is considered separated. Even if an application requires its own instance, it can be configured on a server (or cluster) with other instances, as long as the capacity and availability is properly accounted for. Using multiple instances is arguably the best way to maximize the usage of a server or cluster with physical consolidation, unless there is a desire to have one large, monolithic instance. Both single and multiple instances have their places, and that largely depends on the requirements of the application in all aspects.

An optimal physical SQL Server consolidation architecture is based on all of the principles presented in all of Section 4.3; there is no wizard that will always spit out the right results. Physical consolidation allows access to the full bandwidth of the underlying server (processor, memory, and I/O); the only limits are imposed by what resources are needed by what is installed on it.

Physical consolidation can be easier to deal with and translate requirements to – especially availability. Everyone understands the concept of a physical server and software to be configured on it, and everyone understands the traditional approaches to doing things like high availability, which may or may not be present in a virtual world. The downside to physical consolidation is usually the time spent preparing: the PO process, ordering hardware, waiting for the servers to come in, racking them, and so on. Physical servers also have costs such as electricity and air conditioning, so when a company is looking to reduce costs, minimizing the amount of new servers deployed will be important.

For physical consolidation efforts where a larger number of instances is desired, it is usually best to pursue the acquisition of a “big iron” server (64 processors, and more than 128 GB of memory), place one or more instances of SQL Server 2008, and consolidate to that platform. The initial cost may potentially seem high, depending on how much the hardware can be purchased for, but consolidation to SQL Server 2008 will make for a simple and clean consolidation design.

#### 4.3.1.2 Virtual Consolidation Using a Hypervisor

As noted earlier in Section 2, virtualization has become a big factor in the drive to consolidation. Each virtualization platform allows the host (also known as a *hypervisor*) to deploy a number of “guests”. These guests, otherwise known as virtual machines (VMs), act like a physical server with their own resources. These VMs are effectively isolated from the other VMs on the server to a point. Processor, memory, and disk I/O are controlled by the underlying host (even though the VM thinks it has its own dedicated hardware), but each VM runs in complete isolation from another in the same way that a physical server is independent from another; there is no interaction other than the hypervisor managing the underlying resources.

Microsoft’s virtualization platform is called Hyper-V™; it is an option of both Windows Server 2008 and Windows Server 2008 R2, and it is also available in stand-alone form as Microsoft Hyper-V Server. Microsoft also supports other virtualization platforms for deploying Microsoft products via the [Server Virtualization Validation Program](http://www.windowsservercatalog.com/svvp.aspx?svvppage=svvp.htm) (SVVP). SVVP is an official way for non-Microsoft vendors to validate their virtualization solution so that their customers have full Microsoft support, as if they had deployed on a physical server. If those responsible for virtualization are using a non-Microsoft hypervisor that will be used for virtualization, the DBAs must point those in charge of virtualization to the SVVP page to ensure that what is deployed is supported. For more information about the official Microsoft support policy for non-Microsoft virtualization platforms, see [Knowledge Base article 897615](http://support.microsoft.com/kb/897615/en-us). For more information about Microsoft’s general support policies for virtualization, see [Knowledge Base article 957006](http://support.microsoft.com/kb/957006), and for more information about specific SQL Server support policy for virtualization, see [Knowledge Base article 956893](http://support.microsoft.com/?id=956893). Table 7 shows some of the important differences in Hyper-V, depending on which version of Windows Server 2008 is deployed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Virtualization feature** | **Windows Server 2008 with Hyper-V** | **Windows Server 2008 Service Pack 2 with Hyper-V** | **Windows Server 2008 R2 with Hyper-V** |
| Logical processor support | 16 | 24 | 64 |
| Address space management | Software only | Software only | Hardware (Second Level Address Translation support) |
| Core Parking | No – Limited C states (for more information, see the Intel Web site) | No – Limited C states | Yes |
| IPv6 offloads | No | No | Yes |
| TCP Chimney Offload (VM Chimney) | No | No | Yes (off by default) |
| Jumbo frames | No | No | Yes |
| Virtual NIC interrupts | VP0 | VP0 | VP0 receive; distributed for send |
| I/O sizes (virtual SCSI) | 64 kilobytes (KB) | 64 KB | 8 megabytes (MB) |
| VHD block size | 512 KB | 512 KB | 2 MB |
| Hot-add storage | No | No | Yes |

Performance is arguably the number-one concern for virtualizing SQL Server deployments, because the total number of resources available to a virtual machine is limited by the capabilities of the hypervisor as well as the processor, memory, and I/O that are available to the hypervisor. With the proper planning, however, guest performance can have parity with that of its physical counterpart. Hypervisors support increasing numbers of virtual (or physical) processor usage, and they allow guests to access drives directly.

However, remember that the underlying operating system in the virtual machine does not know that it is actually a virtualized server; if four “processors” are allocated to that guest, it may not actually be four physical processors or cores from the hypervisor – just slices of them as needed, but the guest operating system thinks it actually has four physical processors. Memory is usually a 1:1 mapping.

Then there is the cost of virtualization: besides the direct needs of the guest, the hypervisor itself needs processor and memory for that virtual machine. All of this must be taken into account, and strict testing of performance should occur for mission-critical and performance-sensitive applications.

If you plan on deploying instances of SQL Server on Windows Server 2008 with Hyper-V, you should read the white paper [Running SQL Server 2008 in a Hyper-V Environment: Best Practices and Performance Considerations](http://download.microsoft.com/download/d/9/4/d948f981-926e-40fa-a026-5bfcf076d9b9/SQL2008inHyperV2008.docx), published by the SQL Server Customer Advisory Team. Even if you are not using Hyper-V, the white paper provides insights into virtualizing SQL Server installations that you may be able apply to the virtualization platform deployed in your environment. The bottom line that the white paper addresses is that most virtualization platforms limit the resources a guest can use; this limitation is one of the drivers to determine whether virtualization is right for a consolidation effort.

In addition to potential performance concerns, high availability can be a concern for virtual machines. What happens if the hypervisor fails? Microsoft and the other virtualization vendors have their own strategies for making virtual machines available. For example, one option in Windows Server 2008 R2 using Hyper-V is a feature called Live Migration (detailed information can be found in [this white paper](http://www.microsoft.com/DownLoads/details.aspx?familyid=FDD083C6-3FC7-470B-8569-7E6A19FB0FDF&displaylang=en)), which enables an administrator to move a virtual machine from one server to another, minimizing downtime to the end users who are accessing the virtual machine. SQL Server fully supports deployments that use Live Migration for virtual machines. However, database and instance availability is still a concern. Microsoft now supports the use of guests to be configured as a failover cluster in SQL Server. The other traditional methods for making databases available (log shipping, database mirroring, and replication) can all be employed.

**Tip:** If you want to cluster the guest virtual machines, spread the guests across at least two hypervisors to minimize a single point of failure for the virtualized cluster nodes.

One of the biggest challenges of virtualization is what led to consolidation in the first place: sprawl. It is very easy to deploy a virtualized server. This dark side of virtualization must be carefully managed; otherwise the entire effort of consolidation would wind up being for naught. A guest is just a virtual representation of a physical server; if a physical server is decommissioned, but there are just as many virtual servers out there, what has been saved? Nothing, but some hardware has actually been lost. Consolidation has to bring the *overall* cost down, not just remove physical servers.

Licensing is also something that must be considered with virtualization; cost is arguably the biggest driver for consolidation, and virtualization can drive up costs without proper licensing. Here is some important information for licensing as it relates to virtualization:

* Hyper-V is free and comes with a Windows license; no additional cost is needed for a virtualization platform or feature.
* If Hyper-V based virtualization will be used, Windows Server 2008 Datacenter provides unlimited virtualization rights. That means that all guests with a Windows Server operating system underneath the Windows Server 2008 Datacenter host are covered by the parent Windows license. That can be a big cost savings, depending on how much each individual server’s Windows license would be. While Windows Server 2008 Datacenter may seem expensive at a first glance, look beyond the initial number: long term, it may be cheaper to license Windows Server 2008 Datacenter than consolidate using any other edition with Hyper-V for the host and still have to pay for each individual guest’s Windows license.
* If Hyper-V will not be used for virtualization, the standard Windows licensing policies apply for each guest.
* With all virtualization platforms, SQL Server 2008 Enterprise allows the maximum supported number of guests to host an unlimited number of SQL Server 2008 Enterprise instances, assuming that the entire server containing the hypervisor is licensed. Like the Windows Server 2008 Datacenter option, this may wind up being cheaper than individually licensing each SQL Server installation used in a guest, because in that case, each virtual processor (assuming per-processor license) would need to be licensed.
* Licensing using the Server/Client Access Licenses (CAL) option means that each individual VM would need its own separate SQL Server licensing scheme that would need to account for licensing not only SQL Server for the VM, but each client or device accessing that SQL Server implementation. If the number of users or devices is not currently known, it will be just as difficult to license properly as a virtualized environment. This is why the per-processor SQL Server 2008 Enterprise license may be the better way to go for some environments.

Table 8 summarizes the differences between using multiple SQL Server instances and virtual machines for consolidation.

*Table 8. Comparison of multiple instances with virtual machines*

|  |  |  |
| --- | --- | --- |
| **Functionality** | **Multiple instances** | **Virtual machines** |
| Isolation | Shared Windows installation | Dedicated Windows installation (virtual machine acts like a physical server) |
| Processor resources | All processors available to Windows  Possible hot-add | Maximum allowed by the hypervisor  May need to power down the VM to change processor configuration (hypervisor-dependent) |
| Memory | All memory available to Windows  Standard management in SQL Server  Possible hot-add | Limited both by maximum seen by the hypervisor as well as limitations for maximum memory that can be configured on a VM  Statically allocated to a VM  May need to power down the VM to change memory configuration (hypervisor-dependent) |
| Storage | Connected to physical disks | Pass-through/direct access from a VM to physical disks  Virtual disks |
| Resource management | WSRM (process level)  SQL Server Resource Governor | Hypervisor tools for VM  SQL Server Resource Governor (for SQL Server in the VM) |
| Number of instances/virtual machines | 25 (cluster), 50 (stand-alone) | Determined by available resources |
| Support | Normal support rules | Hypervisor must be Hyper-V or part of the SVVP  Only supports SQL Server 2005, SQL Server 2008, and SQL Server 2008 R2 |
| High availability | Normal availability techniques | Host-level: Failover clustering (Windows), Live Migration, other vendor specific (such as V-motion for VMware)  VM-level: failover clustering (of the VMs – SQL Server 2008 only)  Database Level: database mirroring, log shipping |

The question remains: Where does virtualization fit into consolidation? First and foremost, using virtualization to consolidate development and test environments is one of the best things that can be done. Many already do this. Another use of virtualization was already mentioned: legacy boxes running older versions that cannot be decommissioned or upgraded. To assist in that effort, a process known as physical to virtual (P2V) can be employed to virtualize the existing server. This process will not change any of the current application or database server configurations.

**Tip:** Remember to gracefully shut down SQL Server prior to running the utility used for P2V so that there is no risk of any data or log files being open.

In a production environment, things are less concrete. Virtualization can be and should be done in production, but it will depend on some factors. IT loves virtualization because it is easy to deploy another “server” without having to purchase hardware. But as outlined here, there can be limitations for a guest VM such as being limited to four “processors” or a certain amount of memory, even though the hypervisor has more. What it boils down to is what workloads are being run, and whether a VM gives the performance and growth needed.

There are ways to optimize a VM, such as using pass-through disks to talk directly to a disk subsystem, but remember that the I/O is still being shared with the rest of the VMs through the hypervisor. Avoid overcommitting processor resources, and size memory properly.

The key to success with virtualization in production is testing. Do not put a deployment in jeopardy by just assuming virtualization will work. If the deployment does not succeed, you will have to pull it out of virtualization and onto physical hardware.

Assuming that any issues can be worked through, virtualization brings management (such as Live Migration) and standardization benefits that may not be as easy to achieve in some physical implementations.

### 4.3.2 Applications

Although technically they do not fall under the SQL Server category, the applications associated with the databases and instances control consolidation, more than any other technical aspect.

Every application is different; it is virtually impossible to take a broad stroke and make generalizations that apply to every application. However, the following points should help you determine what to do with the application as it relates to consolidating the SQL Server backend:

* If the application was purchased, third-party vendors are often very specific about what they will and support as it relates to their applications. This includes but is not limited to things such as the version of SQL Server, down to the patch level, that can be deployed for a specific version of the application in question and whether the application database can be contained in an instance with – or even on the same server as – other application databases. This is why documenting the version of the application is crucial; without that information, not much about what can and cannot be done to consolidate can be determined.
* If the version of the application that is deployed does not support the target SQL Server version (in this case, SQL Server 2008), a shared instance with other application databases, or virtualization, there are only a handful of options for consolidation:

1. Is there a later version of the application that does support the target environment? If the answer is yes, there are a few factors that influence whether or not it is possible to deploy the later version of the application. A new version generally has cost associated with it. If there is no additional budget to purchase the later version of the application, it may not be possible to upgrade the application. This means that consolidating it using an unsupported (by the vendor) version of SQL Server would put supportability at risk. Consolidation for this application may need to be considered for a later time.

Even if there is budget to purchase the application, there are other changes that come along with deploying a new version of the application: data migration and upgrading, ensuring that application functionality is the same and that there will be minimal to no impact on end users if the later version is deployed,

1. If there is not a later version of the application that supports the desired SQL Server backend (virtualization or multiple databases per instance), never put supportability of an application at risk for the sake of consolidation. If the vendor does not support the target consolidation architecture and platform, do not consolidate that application’s database to it.
2. If the application is so old (for example, one that uses a SQL Server 6.5 or SQL Server 7.0 backend) and no one knows anything about the configuration because it was in place long before anyone currently at the company was hired, and it cannot be upgraded to SQL Server 2008, consider virtualizing SQL Server with a P2V tool that converts the physical server into a virtual.

* Microsoft provides a feature in SQL Server called a compatibility level. This can be set per database. Its purpose is to tell the database engine that where it is possible, to try to act like an older version of SQL Server for certain behaviors that may have changed. However, it is not emulation: The database and the instance will not be transformed into that previous version of SQL Server. For example, if the database is running under SQL Server 2008, it is still a SQL Server 2008 database, no matter what compatibility level is set. Compatibility level can be used like a crutch to avoid testing as well as fixing issues. For example, if you have some databases set to 6.5 compatibility under SQL Server 2000, that level is not supported with SQL Server 2008, so the application now must be tested against a later compatibility level supported by SQL Server 2008. The recommendation is to always set compatibility level to the version of SQL Server you are deploying with, and only change it if there are issues.

**Note:** With SQL Server 2008, Microsoft supports only three compatibility levels: 80 (SQL Server 2000), 90 (SQL Server 2005), and 100 (SQL Server 2008). Some databases under SQL Server 2000 and SQL Server 2005 may have older compatibility levels.

* Most applications have application-specific tasks that must be performed before and after consolidation, such as running a script to update configuration parameters. These are most often not DBA-related tasks, but it is crucial to be aware of them, so they can make it into the plan and the timeline. It is possible to break an application if certain tasks are not performed, or more specifically, not performed in the right order. Some applications may even have a documented and supported method for migrating databases to a new server. If this is the case, follow the instructions for that application to successfully move a database and its objects from Server A to Server B.
* Determine whether the application has any hardcoded aspects such as instance names, database names, IP addresses, usernames, passwords, or anything else that will break the application if the underlying servers are changed. If these names, IP addresses, and so on, cannot be changed, other methods such as using DNS aliases (CNames) must be tested and used to obscure the change. Even if some of the aforementioned aspects are not hardcoded, they must still be documented, accounted for, and the right method determined to modify the information to point to a new application, Web, or database server. Changing the information may also mean that services that utilize the information must be restarted.
* When analyzing applications, DBAs need to determine whether the application cares whether or not a default or a named instance of SQL Server is used. This application consideration has a few aspects to it. Older applications sometimes only supported default instances of SQL Server. Even with SQL Server 2008, there can only be one default instance per server or cluster, and the rest can be named instances. This affects the architecture that will be deployed. Couple this with a possible dependency on a name or IP address as listed in the last bullet, and moving an application to point to a new SQL Server instance is not a straightforward task. It may be fraught with risk.
* Consider assigning a risk factor to each application and its databases. A low risk factor indicates that an application and its databases do not have significant challenges when it comes to items such as supportability of the target platform, scheduling, or impact to the business, making them better consolidates for consolidation. Ones that have a high risk factor can be much more difficult to consolidate, and indicates that issues such as cost, complexity (which would include the difficulty level of tasks such as recoding to work against SQL Server 2008, mission criticality to the business, and so on), or coordinating a time when the application could be taken offline for consolidation may require significant work. If you use the list of items in this section, it should become clear whether an application is easy or hard to move. Even if an application is an easy move from a technical standpoint, the application may have other factors that make it difficult to consolidate. Issues such as scheduling (such as the accounting example mentioned earlier), lack of expertise in-house and no budget for a consultant, or something similar do not mean that the application and its databases should not be consolidated. A higher risk factor just means that more attention needs to be paid, and it may just affect the order or timeframe in which the consolidation is performed. There is a chance that the risk factor may knock something out of consideration for consolidation, but that is the point of the data analysis and devising the candidate list.
* Because the goal of consolidation is to reduce cost as well as improve agility, for some older applications, it may be determined after they are discovered that they are no longer in use. Until the information gathered is in a digestible format, it is impossible to tell whether or not an application and its databases could potentially be decommissioned. Determining that an application can be decommissioned saves time and work, and it definitely lowers the cost to the business because not only fewer resources are being used, but there is less to support. If an application will be decommissioned, make a final state backup of the databases and application configuration, and then proceed to decommission both the application and the hardware associated with it.

### 4.3.3 Choosing the Edition and Version of SQL Server for Consolidation

Various editions of SQL Server can be deployed, but which one is the right one for a particular consolidation effort? This section discusses some decision points that can help you choose the right version of SQL Server for your organization.

#### 4.3.3.1 SQL Server Edition and Version

Many reading this white paper may still have a significant number of SQL Server 2000 instances deployed. That means technically, consolidation can be done using either SQL Server 2005 or SQL Server 2008. We recommend that you use SQL Server 2008 unless your applications or databases need to reside on SQL Server 2005. The goal should be to configure the end state as one that will be supported for quite some time. If there are applications or databases that do not support SQL Server 2008, consider deploying limited amounts of SQL Server 2005. Section 4.3.2 details methods on how to approach applications that only supports versions older than SQL Server 2008.

SQL Server 2008 comes in various editions (Standard, Enterprise, Workgroup, and so on). To achieve maximum density and scalability, especially if you are consolidating using multiple instances, the recommendation is to deploy SQL Server 2008 Enterprise. SQL Server 2008 Enterprise also has features such as backup compression that can provide a huge cost savings. From a licensing perspective, section 4.3.3.3 discusses how the edition chosen can affect the overall cost of an implementation, especially if virtualization is the method for consolidation.

**Tip:** For more information about the differences between the editions of SQL Server 2008, see [Microsoft SQL Server 2008 Editions](http://www.microsoft.com/sqlserver/2008/en/us/editions.aspx).

#### 4.3.3.2 32-Bit vs. 64-Bit

All SQL Server 2008 deployments should be done using a 64-bit platform where possible. Outside of some performance differences, SQL Server is SQL Server whether it is 32-bit or 64-bit – the on-disk formats for database files are the same, managing instances and databases are the same, and so on. The same could be said for Windows; a 32-bit version of Windows looks and feels like a 64-bit one from an interface perspective, and the tasks are the same.

**Note:** Starting with Windows Server 2008 R2, Microsoft will only ship 64-bit server operating systems. Windows Server 2008 is the last 32-bit server operating system from Microsoft. This means that if the group responsible for Windows has not started to think about 64-bit deployments, they should. The Windows group needs to have the following in place:

* A standard 64-bit build of Windows
* 64-bit equivalents for all software (backup agents, anti-virus, monitoring, and so on)
* 64-bit drivers for all hardware

When it comes to performance, do not assume that 64-bit means twice the performance as 32-bit. In some cases it may be dramatically higher, and in other cases, 64-bit may only be just as fast as 32-bit. All of the factors outlined in Section 4.3 apply. Also, poorly coded and written applications may show minimal improvement in a move to a 64-bit backend, but the real improvements come only from fixing the application. Think of consolidation as an opportune time to visit these sorts of issues with each application that is earmarked.

**Tip:** Be careful; with SQL Server 2008, installing a 32-bit instance using WOW under 64-bit is not supported. So if 32-bit SQL Server is needed, deploy it on a 32-bit version of Windows.

#### 4.3.3.3 Licensing for Consolidated Environments

Consolidation should provide many benefits, not the least of which should be a more manageable approach when it comes to licensing Windows and SQL Server. The reason that licensing matters to a technical audience is that whatever the final architecture is, it needs to be properly licensed. This goes back to the earlier budget discussion. Licensing is but one aspect of the overall budget, and those responsible must know what their limitations are so they can work within what is possible. Also consider the long term cost for the environment as it is being designed: what may be relatively cheap now could be very expensive later if the wrong licensing model is chosen.

**Tip:** For more information about how to license a deployment, see the [SQL Server](http://www.microsoft.com/sqlserver/2008/en/us/how-to-buy.aspx) and [Windows](http://www.microsoft.com/windowsserver2008/en/us/how-to-buy.aspx) Web sites. The policies vary from edition to edition, and the local Microsoft licensing team or a reseller should also help answer any questions.

### 4.3.5 Standardizing the Environment

A major component of the SQL Server architecture that will be designed is its standardization. During this planning phase of the project, review the various deployments already in use and come up with a single, or at least a limited, number of configurations. Besides the version of Windows and SQL Server, other aspects of the deployments are easy targets for standardization. For example, ensure that each server has the same drive configuration. That way, whenever a DBA has to access the server itself, he or she knows where to find backups, program files, and other SQL Server-related objects. Every aspect of a server deployment – down to the **sp\_configure** settings – should be written in a separate document from the plan. That document will be used for any new server that will be configured and added after the consolidation effort.

### 4.3.6 High Availability for Consolidated SQL Server Environments

Consolidation should increase, not decrease, availability of SQL Server databases. This section walks through the considerations and options for making a consolidated SQL Server environment highly available. This section assumes basic knowledge about the various availability features of SQL Server. If these features and how they work are unfamiliar, consult SQL Server 2008 Books Online for more information.

#### 4.3.6.1 Service Level Agreements

Before talking about any availability technologies, service level agreements (SLAs) must be discussed. SLAs are agreements between the various corporate entities (DBAs and IT, IT and the application owner, etc.) which define how available a solution must be. If these are not currently defined, they must be defined prior to consolidation. If they are defined, they should be discovered and documented during the information gathering. After this information is known, it is easier to classify applications and figure out where to place them in the new consolidated environment. The reason for this is that many applications with have different SLAs, and applications with wildly varying SLAs (for example, one application that has an availability SLA of one hour and another that has an SLA of two days) should most likely *not* be combined under the same SQL Server instance. Why? Which SLA will be the one that will be the one that is managed to? If multiple databases exist under a single instance, and the one that has the two day SLA brings the instance down, the application with the one hour SLA cannot afford to be down for two days. SLAs that are similar should be combined.

#### 4.3.6.2 Failover Clustering

Failover clustering is arguably the most popular SQL Server availability feature. Failover clustering is not a scale-out feature; it is used purely for availability. Deploying a failover cluster cannot be done after the fact; it is a decision that must be made prior to installing a SQL Server instance. Failover clustering provides availability an entire instance and everything contained inside of it. Because failover clustering provides full instance-level protection, this means that in the event of a failover, all objects (databases, users, logins, SQL Server Agent jobs, and so on) exist after the instance is restarted on another server after the failover. The benefits of this failover are immediately realized in a consolidated environment: when there are *N* numbers of databases for *A* number of applications, the only thing that will need to be taken account after failover is whether the application (client or server) needs to reconnect to the database server if it was not coded to be cluster-aware. The instance’s name and IP address remain the same; there is no need to worry which node of the cluster the instance resides on from an application perspective. Failover clustering plays into the overarching theme of consolidation in that the backend should remain relatively transparent to applications and end users. Failover clustering is supported for guests running under a hypervisor as discussed in [Knowledge Base article 956893](http://support.microsoft.com/kb/956893/en-us).

**Tip:** For more information about SQL Server 2008 failover clustering, read the book *Pro SQL Server 2008 Failover Clustering* (Allan Hirt, Apress, 2009) as well as the [white paper](http://download.microsoft.com/download/6/9/D/69D1FEA7-5B42-437A-B3BA-A4AD13E34EF6/SQLServer2008FailoverCluster.docx) on failover clustering SQL Server 2008 published by Microsoft.

#### 4.3.6.3 Database Mirroring

Database mirroring provides database-level protection. This not only means that all objects outside of the database (SQL Server logins, SQL Server Agent jobs, and so on) must be accounted for in a server switch, but that availability is being managed differently for each database. With the improvements in hardware and to SQL Server 2008, it is possible to mirror more databases than was possible in SQL Server 2005. However, there is a practical limit, because each mirroring session consumes resources such as threads. Because consolidated instances will contain quite a few databases, testing must be done to see how many databases can be mirrored for a given instance of SQL Server.

SQL Server 2008 also has one new improvement to database mirroring that has a direct implication in a consolidated environment. The log stream between the principal and the mirror can be compressed. Not only will this allow more databases to be mirrored per instance, but it will also reduce the network resources needed for databases participating in a mirroring session.

#### 4.3.6.4 Log Shipping

Log shipping is similar to database mirroring – it provides database-level protection. However, unlike database mirroring, there are no real restrictions on the number of databases per instance whose transaction logs can be shipped to a warm standby server. Log shipping has always been a DBA-friendly option; it is easy to deploy and maintain, because it is based on backup and restore. Like database mirroring, it means that availability is being managed for each individual database and all objects residing outside the database must be accounted for. A DBA has to rationalize in a consolidated environment if he or she wants to monitor many log shipping plans.

### 4.3.7 Capacity and Performance Planning for Consolidated SQL Server Environments

One of the hardest things to do when devising the consolidated environment is to ensure that it is highly performing and has enough growth for the future. While it is always a challenge even in the best of circumstances where things are 100 percent isolated, less isolation means there is also less room for error. This section talks about the various considerations to ensure that the consolidated environment will meet the performance and capacity needs, and make deployments more agile.

**Tip:** An excellent resource for capacity management is the SQL Server Books Online topic [Maximum Capacity Specifications for SQL Server](http://msdn.microsoft.com/en-us/library/ms143432.aspx). This topic details everything from how much memory can be used to the number of bytes used for a data type and everything in between. Assuming that the right information was collected, the information contained in this topic enables DBAs to figure out how much capacity will be needed based on the information gathered matched to the right attribute listed in the topic. This is “simple math”.

#### 4.3.7.1 Processor

Some databases and instances that are consolidation candidates may be sitting on older servers. These servers have processors that are not necessarily analogous to today’s modern multiple-core processors, which can either be configured with a 32-bit or a 64-bit operating system. There is no calculator or utility that will convert an old 32-bit Pentium III running at 80 percent capacity into a utilization number for a newer 64-bit multi-core processor. So how is processor utilization determined?

Today’s multi-processor, multi-core systems are much more capable than big machines bought even three to five years ago. If a particular existing instance is currently averaging 20 percent of a slower, older single-core processor, it is a reasonable assumption that it will be more efficient on a modern multi-core, faster processor. How much faster is difficult to say; the only definitive way is to test under actual workload. Even if the assumption is made that 20 percent of a single core old processor equates to 20 percent of one core of one processor in a newer system, that may be overkill, but it still leaves plenty of capacity for other things on that server. Unfortunately this exercise is not an exact science unless testing is done.

Depending on the density of consolidation you want and the workloads that the server may have to run, a large server with many physical processors may be needed. For example, the desire to host 20 VMs will take some computing power to ensure all of the guests have proper performance. The only way to know for sure is to test actual workloads on a configured server. The cost must be balanced against the benefit: If the costs to buy processors for one server are higher than what is gained, it may be better to just buy another physical server.

**Tip:** For information about how to use WSRM to constrain processor resources with a SQL Server instance, see Section 5.4.3. Another tool that can be used for physical, but not virtual, deployments of SQL Server is the affinity mask option in SQL Server. For more information, see [affinity mask Option](http://msdn.microsoft.com/en-us/library/ms187104.aspx) in SQL Server Books Online.

#### 4.3.7.2 Memory

When it comes to keys to SQL Server performance, memory is a close number two to disk I/O (some may argue they are both equal). The good thing is that these days, memory is fairly cheap from a cost perspective. However, containing costs is part of the consolidation process, so sizing servers appropriately is crucial. Too much excess capacity is wasted money, and too little capacity means money needs to be spent fixing the problem.

Configuring memory, like processor, is not as straightforward. It is easy to measure at a server and instance level, but not at a per-database level. As noted in the SQL Server Books Online topic “Maximum Capacity Specifications for SQL Server” linked earlier, some objects (such as a user connection) consume a fixed amount of memory. These numbers must be worked out, factoring future growth based on the information gathered during discovery, to get a whole picture of how much a database and instance of SQL Server will really need when it comes to memory. Remember to take into account leaving memory for Windows and other processes and applications running on the server. DBAs should work with the team responsible for Windows to ensure that SQL Server will not starve the underlying operating system; if that happens, nothing will work. In the past, the assumption was that leaving 1 gigabyte (GB) for Windows was fine, but when a larger system is deployed, the overhead may go up to 2, 3, 4, or more GB of memory required.

There are three methods of configuring memory for an instance of SQL Server:

* Set a fixed maximum amount of memory
* Set a fixed minimum amount of memory, and let it all be dynamic above that
* Let all memory be dynamic

The best method will always be to assign a fixed maximum amount of memory to an instance with the **max server memory** option of the system configuration stored procedure **sp\_configure**, but this method can lead to a perception of wasted resources. In the old days of 32-bit Windows 2000 Advanced Server and SQL Server 2000 Enterprise Edition, if you wanted to scale beyond 4 GB of memory, the only way to make things work was to assign a fixed amount of memory to an instance of SQL Server. 8 GB of memory was about the maximum people would configure servers with, so to get even two instances on Windows 2000 Server, setting a fixed maximum meant each instance needed to be 3.5 GB or less to account for the overhead of Windows as well as both instances so they did not impede on each other. The story is really not different today, but servers can contain much more memory, and buying additional memory is a much more trivial concern in the grand scheme of a major project like consolidation.

Setting a fixed minimum amount of memory can work; in cases where failover clustering is deployed and a failover occurs, or on a stand-alone server with multiple instances, they need to adjust to each other while maintaining that minimum memory. How long that process takes is unknown, and if it does not adjust to the satisfaction of those using the instance (that is, there are help desk calls that performance is slow), failover is not a success. The only way reduced capacity would be acceptable is if there is a performance SLA in place that states that in a failover, there will be reduced capacity, and it is agreed to by the business and/or application owner.

The last option – letting all memory be dynamic – is generally a bad idea. Consolidation is all about standardization, consistency, and control. Dynamic memory is none of those things. Dynamic memory can work because the SQL Server instances will eventually adjust; that is not the issue. The issue is that most users would rather have consistency and predictable performance.

#### 4.3.7.3 Storage

Last, but certainly not least, is the storage configuration for SQL Server. There are two aspects to configuring storage: the actual space needed for things like data and log files as well as backups, and the I/O requirements. When most IT workers deploy a server, they are asked the following question: “How much space do you need?” This is a completely legitimate question and it must be answered, but it only tells half of the story. Disk I/O is an afterthought. Both are equally as important. This section talks about how to approach both, but it does not go into every aspect of disk configuration. Only topics that directly relate to consolidation are addressed.

**Tip:** Two resources for diving deeper into disk configuration with SQL Server are the white papers [Predeployment I/O Best Practices](http://technet.microsoft.com/en-us/library/cc966412.aspx) and [Disk Alignment Best Practices for SQL Server](http://msdn.microsoft.com/en-us/library/dd758814.aspx). Another good resource is [Jimmy May’s blog](http://blogs.msdn.com/jimmymay/default.aspx).

##### 4.3.7.3.1 Storage Capacity

Coming up with a first stab at how much space is needed for a consolidated SQL Server deployment is a straightforward task. Total the amount of space used (the information should have been part of discovery) by the current databases (both data and log file) as well as backups, and then incorporate some sort of reasonable multiplier to account for growth. However, that number does not take into account any new deployments or applications that may be deployed in the new shared model. A big mistake that many make in consolidation is that they don’t account for longer-term growth; just what is used now and a little beyond it. A good “fudge factor” would be to double the capacity used today.

Backup storage is often overlooked in storage capacity discussions. Backups, like I/O, are just as important a component to the storage story as the data and log files are. Backups can be the same size as the data itself, and assuming the applications will be in place for quite some time with no data archiving, full backups can get to be large. Add to that transaction log backups, and top it off with a corporate retention policy, and backups could wind up consuming as much as, if not more than, main data and log files. Understanding aspects like retention policies are crucial in planning storage for consolidation.

##### 4.3.7.3.2 Disk Performance

The number-one key to SQL Server performance is disk I/O. Unfortunately, most underlying storage subsystems have been optimized for space and not I/O capacity. Many storage engineers configure a SAN as one big chunk of disk that does spread the I/O load, but is not optimal for any one workload; it is a completely shared resource. Deploying mission-critical, highly performing SQL Server can be challenging. How do DBAs deal with shared storage and no way of controlling I/O or low level configuration aspects like LUN sizes and RAID types? The answer is simple: information.

During discovery, I/O information for each database should have been gathered. Like storage space, disk I/O is somewhat of a math problem, in the sense that the results from **dm\_io\_virtual\_file\_stats** or fn\_virtualfilestats can be used to see how many I/Os (both read and write) were done since the database has been in place (divide by the number of years, months, days, and so on). Couple that with the performance stats captured during the busy, slow, and normal times, and a picture will emerge of what disk I/O looks like for a given application and database. Map the disk I/O to the number of transactions per second, and an even clearer picture emerges. Work with the storage administrators so that both the space and I/O needs of the consolidated environment are met; go to them with both calculations, not just one.

After the basic numbers are derived, as candidates are being sorted out and combinations put into place, these numbers will guide if combining databases A and B make sense. Starving one database at the expense of another is not the way to deploy consolidation. Consolidation is a balanced approach, so mixing I/O loads of heavy to light is the way to go.

If you are using virtualization, and if the hypervisor supports it, use physical disks (through some sort of pass-through feature) instead of virtual disks (such as the VHDs in Hyper-V that are used for storing the guest operating system). The performance increase can be significant. Whether the configuration is physical or virtual, it is always recommended not to share underlying spindles with other servers, applications, and workloads. As mentioned at the beginning of this section, that may not be possible, but it is still a best practice. The two main things to worry about from a performance standpoint are transaction logs and **tempdb**.

This method is not an exact science, and it will definitely be trial and error, but it will result in a better consolidated environment than if the effort to obtain I/O information is avoided. Post-consolidation, make sure that I/O latency and performance is monitored.

##### 4.3.7.3.3 File Placement

In the consolidated world, it is unrealistic to expect that each and every database will have its data and log files always separated and isolated. If you have 50 databases per instance, that is a minimum of 100 files (50 data, 50 log), and attaching 100 drives to a server is unnecessary overhead. File placement is a direct result of both storage and I/O management, because you know which databases have conflicting I/O patterns or consume the most space.

Combining files on the same drives is most likely what will happen with some of the databases. At a bare minimum, interleave data and log files across drives so there is not a single point of failure. Table 9 shows an example of five databases striped across two different drives.

*Table 9 – Example of database striping*

|  |  |  |
| --- | --- | --- |
| **Database** | **Drive A** | **Drive B** |
| Database 1 | Data | Log |
| Database 2 | Log | Data |
| Database 3 | Data | Log |
| Database 4 | Log | Data |
| Database 5 | Data | Log |

Remember to also take backup placement into account.

##### 4.3.7.3.4 tempdb

One thing that should be explicitly pointed out from a capacity as well as a performance standpoint is **tempdb**. **tempdb** is a completely shared resource for a single instance of SQL Server; each instance has its own, so if there are multiple instances per server or cluster, there is not a single shared **tempdb** for all of them. The reason this is being mentioned is that some applications use **tempdb** heavily, others less so. SQL Server 2008 also uses **tempdb** quite a bit (example: sorting space for index rebuilds), so sizing **tempdb** properly to handle a combined workload from multiple applications and databases is something that must be done, and it can only be done properly by testing. Hopefully during the discovery phase that information was gathered around the usage of **tempdb**, including the sizing and I/O usage of it from an instance where a soon-to-be consolidated database was living. Take those numbers and do some rudimentary addition to come up with sizing numbers for a consolidated **tempdb**, then test it with workloads. Also be sure to check that databases with different collations are not consolidated under the same instance; for more information, see [Mixed Collation Environments](http://msdn.microsoft.com/en-us/library/aa214897(SQL.80).aspx) in SQL Server 2000 Books Online. For information on SQL Server 2008 collations, see [Collation and Unicode Support](http://msdn.microsoft.com/en-us/library/ms143503.aspx) in SQL Server 2008 Books Online. There is also an excellent [white paper](http://technet.microsoft.com/en-us/library/cc966545.aspx) on working with **tempdb**. Although these two documents were written for earlier versions of SQL Server, the information they provide applies to SQL Server 2008 as well.

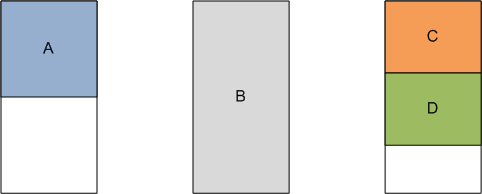
**Tip:** To avoid spinning up unnecessary I/O operations, size all databases, including **tempdb**, appropriately from the start, even if that space will not be used. Set reasonable automatic growth if you want, but the goal should be for databases not to have to grow. The only downside is that if the database needs to be restored elsewhere, that same amount of space will be needed, so come up with a sizing scheme that minimizes growth to keep the files from expanding every time an operation occurs.

#### 4.3.7.4 Failover Clustering, Performance, and Disk Configuration

All of the aspects discussed in this section are valid for a clustered implementation of SQL Server 2008; however there is one main difference that does not exist for a stand-alone server: failover. Failover complicates the performance architecture. What everyone wants after a failover is no performance degradation, no matter how many instances are running on a particular node in the cluster. That is easier said than done. Some of the controls and utilities to enforce policies to help performance are documented later in section 5.4. This section talks about the general concepts of how to approach the failover condition, and more specifically, the worst-case failover condition.

**Note:** If the deployment is not clustered, similar principles still apply for combining; the failover condition is unique to clusters and needs calling out.

Consider the following example: a three-node failover cluster is deployed with a total of four instances. Each node has a total of 16 GB of memory. Instance A requires 8 GB of memory, Instance B requires 12 GB of memory, Instance C requires 6 GB of memory, and Instance D requires 6 GB of memory. The initial design did not take a failover condition into account, and it may look something like Figure 1, where Instance A is on Node 1, Instance B is on Node 2, and Instances C and D are on Node 3.



**Figure 1**. Example three-node cluster with four instances of SQL Server

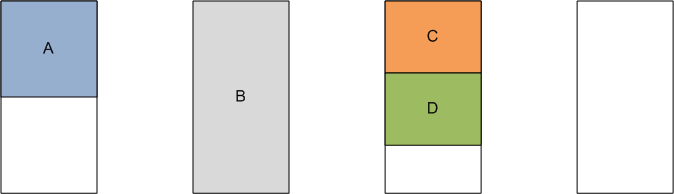
So what is wrong with Figure 1? Consider failover – it is essentially a math problem. If Node 2 fails, what node is Instance B going to fail over to? It will attempt to move to either Node 1 or Node 3, but adding 16 GB plus any processor and I/O requirements to two nodes that do not have the capacity will result in Instance B not being able to start. That defeats the whole purpose of clustering.

Now that the problem is understood, there are a few different approaches to having a clustered configuration of SQL Server with multiple instances where contention is minimized in the failover condition. There are really two failover conditions – worst case, where all instances move to a single node, and the normal condition, where a subset of instances moves to another node. Both need to be accounted for.

In this example, the three nodes cannot handle the worst-case scenario, and they can barely handle the normal failover condition. Instance C or D could fail over to Node 1, but not both at the same time. In a perfect world, nothing would ever fail over, so if that happens, there will never be contention. That is not realistic. The ability to fail over is not just for disaster purposes; instances can be manually moved to allow maintenance to be performed on the node itself, minimizing downtime. If there is the capacity on another node, the instance moved should not have to fail back.

At a high level, the first approach is more straightforward: have enough memory, processor, and I/O capacity on each node to be able to run all instances of SQL Server. Memory-wise, that means that each node would need 32 GB of memory plus whatever is needed for Windows and an additional amount for growth; assume therefore that each node will need at least 40 GB of memory to handle the worst-case failover condition. The process is similar for both processor and memory: What is needed is the sum of all instances for what they need now plus what Windows needs, in addition to allowing some overhead for future growth. The problem with this approach to some is that there may be the perception of “wasted resources”. For example, Instance A only consumes 8 GB of memory, meaning that minus what Windows uses, there is a lot of capacity not actively being used. That may bother some in the IT shop, but remember why the solution was deployed: to increase availability and to ensure that after a failover, performance remains the same.

Another approach which has been around for quite some time is known as deploying an N + *m* cluster, where N is the number of nodes, and *m* is one or more dedicated failover nodes. **Figure 2** shows an N + 1 cluster.



**Figure 2**. N + 1 failover cluster

What does the + node (or nodes) bring to the table? The answer is surprisingly simple: The reality is that if more than one node fails, there is most likely a fundamental problem with the cluster that will affect all of the nodes. If the desire is to maximize usage while also increasing availability, adding a dedicated failover node (or nodes – up to the maximum limit of nodes that Windows supports can be added) is a middle ground. Yes, the node will just sit there doing nothing, but if a failover occurs, each instance can be configured to fail over to the additional node first, and the others after. This is known as setting failover and failback policies in Windows, and it is only done with more than two nodes. The problem still exists if Instance B needs to fail over to another node and there is not the capacity, but failover should be an exception state, not a consistent event. Having one or more dedicated failover nodes is a fairly cheap exercise, because the cost of hardware as an insurance policy is generally less than the cost of the organization being down for an extended time.

There are both instance-level tweaks (such as setting processor affinity or setting a fixed amount of memory) as well as utilities documented later in Section 6.4.3 to assist with ensuring performance in a failover condition.

From a disk configuration standpoint, there is one item that those consolidating with failover clustering should be aware of: A SQL Server 2008 failover clustering instance still requires the use of at least one drive letter. Mount points can be used, but they must be mounted under a drive letter. The use of a drive letter clearly is a factor in determining how many instances of SQL Server 2008 can be deployed in a clustered configuration.

**Note:** Although not explicitly called out in this section, the processor has the same issues post-failover that memory does; there needs to be enough processing power to take on the additional load of another instance.

### 4.3.8 Security

Using the information gathered during the discovery phase, analyze the permissions for users, logins, and objects for the databases targeted for consolidation. Associate the SQL Server objects with the proper application, and specifically identify objects that require escalated privileges such as logins configured as system administrator, because these may be potential problems. Also identify anomalies such as developers having access to the production environment in any capacity. DBAs should view the new, clean, consolidated environment as just that: a clean slate. Do not carry over the problems of the old environment; otherwise the issues (such as failing SOX or HIPAA audits) will follow. This analysis speaks to the drivers of control and consistency.

To ensure that the new environment is secure, security standards for applications that will be deployed in the consolidated environment must be devised and published so that all applications will work in the new environment. Consider cleaning up domain access. The current SQL Server implementations may be joined to many different domains where you need to worry about many different logins. You should attempt to simplify access and consolidate security during database consolidation.

When applications and databases are identified as consolidation candidates, and the initial designs are made, there could be more than just problems with conflicting performance and workloads; there could be security conflicts. For example, if two databases (A and B) have users named RANDY that map to a single SQL Server-level login that is also named RANDY, RANDY from Database A and Randy from Database B may be granted the same level of access. If RANDY is a system administrator for an application, that translates into a conflict because the login that did not need system administrator privileges will now be escalated. If the RANDY user from the previously non-escalated application accesses SQL Server, the person will be able to do whatever they want within the instance.

Security also plays into standardization – by devising standards that all applications must adhere to in a consolidated world, the DBAs gain control of the environment. Developers should not have sa privileges in production. Similarly, packaged applications should also not need escalated privileges. In a consolidated environment, applications need to be islands in a bigger sea. If one violates the standards and has escalated privileges, is it violating any other rules in the environment? Can the other applications – and their owners – accept that another application may be able to read their data?

### 4.3.9 Determining Database Consolidation Ratios

Related to the discussion of physical vs. virtual consolidation is the concept of *database consolidation ratios*. Some reading this paper may have heard from those who have successfully consolidated environments a statement like, “We have achieved a 50:1 consolidation ratio.” What this means is that for every instance of SQL Server (or possibly physical server, depending on the unit of measurement), there are either 50 databases per SQL Server instance, or for every 50 of the older servers, there is one new server. Going back to ROI for a moment, this is one way to quantify how much is being saved via consolidation.

Coming up with how much can be consolidated on one server is completely a function of capacity management combined with figuring out complimentary workloads. For example, placing 10 DSS databases under the same instance, or even on the same disk, makes no sense, because there is a high change there will be performance conflicts. Where possible, mix workload types, and combine databases if it makes sense to do so. Database size may not relate to how much it is used; there are many small databases that are used heavily, and there are large databases in the hundreds of gigabytes or terabyte range that are barely used at all.

**Important:** Be careful about overconsolidation, such as ratios of 200:1. The end architecture must be easy to administer. Overconsolidation can lead to taking things out of consolidation or moving them around, which is not the desired end result.

### 4.3.10 Upgrading and Consolidating at the Same Time

If the goal is to consolidate from SQL Server 2000 or SQL Server 2005 to SQL Server 2008, the databases will be upgraded when they are restored under a SQL Server 2008 instance. While this is fully supported and encouraged, appropriate steps must be taken to ensure that the application is fully tested against SQL Server 2008 prior to the consolidation. This application testing is not a DBA task, but it is crucial for ensuring that the application will run correctly post-consolidation. This testing is not a test of the move process, but purely of functionality. For more information about all of the considerations of upgrading applications and databases to SQL Server 2008, see the [SQL Server 2008 Upgrade Technical Reference Guide](http://www.microsoft.com/downloads/details.aspx?FamilyID=66d3e6f5-6902-4fdd-af75-9975aea5bea7&displaylang=en).

### 4.3.11 Other SQL Server Components

Depending on what has been deployed as part of each solution, there may be additional SQL Server components, such as SQL Server Reporting Services and SQL Server Integration Services, that are part of the overall architecture. In the new consolidated world, will these components be shared by multiple instances and databases? Will each application still use its own Reporting Services or Integration Services installation? Do the shared components need to be upgraded, or can they be left alone? Part of the answer to these questions goes back to capacity management: If the load is not heavy, it should be possible to reuse these components for multiple databases or applications.

This may mean that part of the application needs to be reconfigured. In some cases, it may be best for an application to avoid sharing some components. For example, if there is an application that must be SOX and HIPAA compliant and uses a secure Reporting Services deployment, it may be that the deployment should not be shared with an application that does not have the same stringent security.

Another more practical example is the use of replication. Many current customers use some form of replication, which has various components – a Publisher, a Distributor, and finally, a Subscriber. It is pretty clear where the Publisher will live – it is where the source database will exist. However, what Distributor to use is now a question mark, because with many databases consolidated under one or a few instances or servers, it may not make sense to have Distributors for each replicated database. If fewer Distributors, or even a single centralized Distributor, are going to be used for replication in the consolidated environment, make sure that there are no network issues, because the replicated transactions may also now be traveling farther so they can be replicated to a Subscriber. Disk I/O issues will also need to be dealt with, because multiple publications will be sharing resources.

If Analysis Services is going to be consolidated as well, there are a few guidelines to follow:

* If you are placing Analysis Services on the same server with a relational warehouse instance, use the Shared Memory network protocol to avoid a network bottleneck for processing, because it can affect query performance.
* Use separate query and processing servers, and use multiple query servers for availability and performance.
* Analysis Services requires approximately 1.5 times the number of threads needed for the relational engine. So the threads on the relational engine may need to be increased under heavy load.

## 4.4 Technologies for Consolidating SQL Server

Assuming an application does not have a special utility for moving data from one instance to another, it is the DBA’s responsibility to move the data using one of the technologies listed in this section. Unfortunately, there is not a clear-cut winner, and different options can be used for different applications and databases depending on any number of factors, such as the window to make the switch, network latency, and size of the data itself.

There are two utilities which are not talked about in this section for moving databases: SQL Server Integration Services (SSIS) (and to some extent, Data Transformation Services [DTS]) and the command-line utility BCP. The reality is that SSIS and BCP are good at what they do – move data – but they are better for loading and maintaining warehouses, not re-creating an environment. SSIS is also more of a programming tool starting with SQL Server 2005, and not all DBAs are able to write SSIS packages.

**Tip:** No matter what method is chosen to move the databases to the consolidated environment, always make a final full backup of all databases (including system databases) before you decommission the server and/or instance. Depending on the number of databases, this could take up a considerable amount of disk space, but it gives you a fallback plan in the event that the server or instance is immediately decommissioned and a problem occurs post-consolidation. With these backups, it is possible to re-create the old environment based on backups.

### 4.4.1 Backup and Restore

Taking a full database backup, copying it, and restoring it on a target server is still the most common method of transferring data from one environment to another. For a small number of databases where each database’s size is manageable, using backup and restore may be the easiest method to use. The challenge of performing consolidation in this manner is that there are only two ways to do this: serial, meaning that after one database is moved, you move onto the next; and parallel, meaning that multiple databases are moved at the same time. Both have their pros and cons. Moving databases in serial allows for less possible errors, because no new work is started until the existing work is finishes; there are fewer distractions. In environments where there may only be one or two people performing the task, this may be the way to go, but it may also take longer. Assuming that there are enough qualified resources on the ground to do the work, moving databases in parallel certainly maximizes time but may increase risk, because there is not a singular focus. The order of databases may also need to be tightly coordinated, as one database may need to be moved before another for any number of reasons. If there is only a single resource doing work in parallel, it is sometimes very easy to lose track of where you are and what you are doing, leading to mistakes. Mistakes that need to be corrected extend downtime. Whichever approach is followed, make sure the appropriate steps are put in place to ensure that risk is minimized.

When it comes to very large databases (VLDB) with sizes in the hundreds of gigabytes or over one terabyte, the time it takes to back up, copy, and restore a database can be measured in hours or even days. Waiting until the time for the cutover to the consolidated server may not work, because you may only have a window of a few hours. If there is an upgrade to SQL Server 2008 in the process, the database will also be upgraded during the restore process. The upgrade process may extend the restore time, depending on the amount of work SQL Server needs to do to update it.

### 4.4.2 Log Shipping

Log shipping is a very effective way to migrate data. Log shipping is normally used as a high availability or disaster recovery technology, but it can also be used to minimize downtime during a server switch for consolidation as well. Log shipping is based on backup and restore; in log shipping you take a point-in-time full backup of the source database (known as the primary) and then restore it to a destination server (in this case, the one where you will be consolidating), using the WITH STANDBY or WITH NORECOVERY option to allow transaction logs backups to be applied to it. The destination is called a secondary or a warm standby. You then ensure that transaction log backups of the primary database are made on a scheduled basis, copied to the standby server, and subsequently applied.

When it comes time to bring the standby database online, you only need to worry about making sure the last transaction log backup was made on the primary and copied, and that the transaction log backups are applied. If you have been shipping transaction logs the entire time, this may only mean minutes of time to do the database portion of the switch. One of the best things about log shipping is that you can frontload a lot of the work, especially if you’re talking about a VLDB. You can configure log shipping as soon as the new consolidated hardware is configured with Windows and SQL Server, so you can do this preparation work weeks or even months in advance.

SQL Server 2000, SQL Server 2005, and SQL Server 2008 all have built-in log shipping, but they are incompatible; you cannot use them unless you are going to consolidate to the same version of SQL Server (such as SQL Server 2005 to SQL Server 2005). You can use your own implementation of log shipping, which is easy to code, or a version-agnostic one such as the free available-for-download one written to accompany the book *Pro SQL Server 2005 High Availability* (Allan Hirt, Apress, 2007). If you are upgrading to SQL Server 2008 as your consolidated platform, the databases will be upgraded as part of log shipping.

From a DBA perspective, log shipping is easy to implement and easy to manage. It is not complex. Outside of folder permissions to ensure that the transaction log backup files can be written to the destination, assuming you use an automated version of log shipping, it should “just work”, and the role change should be able to be measured in minutes, not hours.

### 4.4.3 Database Mirroring

Database mirroring can also be used to migrate databases from A to B. However, it is limited in the sense that it is edition-specific and version-specific; you can mirror SQL Server Standard to SQL Server Standard and SQL Server Enterprise to SQL Server Enterprise, and in limited scenarios, you can use database mirroring to upgrade a mirrored configuration from SQL Server 2005 to SQL Server 2008.

Like log shipping, database mirroring can allow you to frontload a lot of the work, such as the initial backup, copy, and restore of the database. The cutover to the consolidated server for that particular database should be completed during a short outage instead of a prolonged one. Configuring database mirroring is a touch more complex than log shipping, because it uses more resources; you need to ensure that the network and disk I/O on each side are optimal to handle the task. Database mirroring may not be for everyone, but it may be worth a look for databases that cannot tolerate extended downtime.

### 4.4.4 Detach and Attach

Another option in the DBA’s arsenal is detaching a database, copying the data and log files, and then attaching them on the destination consolidation server. Detach and attach has been a feature in SQL Server since SQL Server version 7.0, so it is mature. When you detach a database, it is completely disconnected from the instance, as if it has been dropped; it no longer shows up as a usable database. This poses minimal risk to the original instance, because the database no longer technically exists (until it is reattached). However slim the chance may be, things can go wrong in attaching the files back to the original instance.

The use of detach and attach for moving databases to a consolidated environment should not be your first choice. You cannot complete any work ahead of time; the move can only happen at the time of consolidation. Add to that the fact that when you move the database files, you move their actual size; there is no compression involved unless there is some sort of hardware compression on the network itself. This means that VLDBs will take a long time to go from A to B, so unless the window for the move is large, detach and attach should be thought of as a last resort option.

**Tip:** As noted in the SQL Server 2008 Books Online topic [Deprecated Database Engine Features in SQL Server 2008](http://msdn.microsoft.com/en-us/library/ms143729.aspx), the stored procedure **sp\_attach\_db** has been deprecated as of SQL Server 2005 and was replaced with the command **ALTER DATABASE … FOR ATTACH**. Detaching is a feature set to be removed in a future version of SQL Server. You should not rely on the use of **sp\_attach\_db** if using this functionality. For information about how to use the FOR ATTACH clause of ALTER DATABASE, see [ALTER DATABASE (Transact-SQL)](http://msdn.microsoft.com/en-us/library/ms174269.aspx) in SQL Server Books Online.

### 4.4.5 Moving Objects Other Than Databases

All objects outside of the database are also available on the consolidated server. Some objects are more straightforward to move than others. The documentation created during discovery and updated throughout the planning process is the blueprint for what objects need to be moved where. Missing an object may not appear to be a problem, but the first time someone goes to access a bit of functionality that is dependent upon that object, issues can occur. DBAs can see objects in Enterprise Manager or SQL Server Management Studio, but they have no clue why they are there or what they are used for. That is why the application owners must be involved in the process. Here is a partial list of objects that will possibly need to be migrated to the consolidated environment:

* Linked servers
* SQL Server Agent jobs
* Replication configuration
* DTS and/or SSIS packages
* SQL Server-level logins
* Any non-system stored procedures, functions, and so on that reside outside of the user databases

Many of these objects may require an application-specific way to be migrated, and it may not be as simple as having the DBA script and then re-create it in the target consolidated instance. If the objects do not have any application-specific dependencies, SQL Server processes can be used. It should also be noted that both the SQL Server 2005 and SQL Server 2008 versions of Integration Services have object transfer tasks, which more skilled DBAs (or developers) can use to create Integration Services packages to move some objects.

### 4.4.6 Resources for Moving Objects

Here are some useful articles in the Microsoft Knowledge Base that may help you with moving objects and databases:

* [http://support.microsoft.com/kb/314546/](http://support.microsoft.com/kb/314546) “How to move databases between computers that are running SQL Server”
* <http://support.microsoft.com/kb/246133/> “How to transfer logins and passwords between instances of SQL Server”
* [http://support.microsoft.com/kb/240872/](http://support.microsoft.com/kb/240872) “How to resolve permission issues when you move a database between servers that are running SQL Server”
* <http://support.microsoft.com/kb/918992/> “How to transfer the logins and the passwords between instances of SQL Server 2005”
* [http://support.microsoft.com/kb/224071/](http://support.microsoft.com/kb/224071) “How to move SQL Server databases to a new location by using Detach and Attach functions in SQL Server”

## 4.5 Putting the Plan Together

After clear decisions have been made around architecture and other aspects of the consolidation, it is time to put the plan together. The plan is a living document, because it will change based on factors that may appear at a moment’s notice. The plan must include not only the obvious technical aspects, such as the steps, in order (even if the tasks are performed by different groups), needed to accomplish a task, but also any other detail related to the consolidation. It should be the master reference.

### 4.5.1 Documenting the Technical Aspects of the Plan

The part of the document that many reading this white paper will be concerned about is the section that directly deals with all technical aspects: the backend architecture, the steps to consolidate (such as how to configure log shipping for the databases), and so on. That part of the document must be extremely detailed and thorough; there should be no ambiguity. A junior resource should be able to pick the plan up and execute it if need be.

### 4.5.2 Staffing the Consolidation Effort

The plan must also document who will be responsible for working the consolidation effort. Staffing needs to take into account not only the DBAs, but also the other administrators, as well as application experts (such as testers). It may not matter if the resource is physically “in the building” or elsewhere remotely connected. Every consolidation date (assuming a phased approach) should have a mixture of both senior-level and junior-level resources on a rotating schedule, because no one worker should get burned out. Post-consolidation, extended support hours (up to 24x7) need to be in place for a predetermined time period (a few days to a few weeks, depending on the needs of your organization). Even the best consolidation efforts will have problems. Extended support hours and the corresponding extra staff available to help troubleshoot any unexpected problem leads to a smoother transition and creates an air of confidence for future executions.

Although it technically is not a staffing issue, it is always best to let vendors (such as Microsoft and any application vendors) know that the work is happening. This will ensure that if a major problem occurs, they are already aware of what is going on and can help out much more quickly; you won’t need to bring them up to speed. While it may burn a support incident, having peace of mind for such a major investment by the company is worth the cost.

### 4.5.3 Determining the Consolidation Timeline

Chances of successful consolidation will be higher if it is completed in phases rather than one big bang. The plan’s timeline should represent a phased approach. It is easier to move smaller amounts of applications and databases over a period of time than it is to do everything in one shot; start with instances and databases that have lower risk and are going to be easier to consolidate. There is nothing like having solid success from the start. The consolidation process should be a manageable one for all parties involved, and doing too many things at once could complicate execution as well as post-consolidation troubleshooting.

The timeline is often controlled by the needs of the applications and the groups using them. Very few want to be the first to go; the first round may be a mixture of volunteers and nonvolunteers. Also, there are always some applications in every environment which can only be moved, upgraded, and so on during a specific window of time. A good example would be a financial application used by the accounting department that is an integral part of monthly, quarterly, and year-end closing. Chances are the application cannot be touched for two or three weeks a month (the week of close, and maybe the week before and after). Assuming the accounting application and its database are to be consolidated, the timeline needs to be negotiated and incorporated into the phase planning as per the result of the negotiations. For every application user testing needs to be guaranteed by the application owner and incorporated into the plan as a major step.

There will also be applications that may seem impossible to schedule the consolidation. At some point, someone with the proper authority will have to make that call and schedule a date. Try not to let the application owner stall the move. Even applications with SLAs that say no downtime may have to take some downtime to be consolidated if they are targeted.

For scheduling purposes, the best thing to do is move “like” (that is, complimentary) or associated applications and databases at the same time. For example, if you have a reporting application that has five sources, move all of those databases at the same time. It will reduce troubleshooting, because the ecosystem that supports those applications is intertwined (and hopefully well documented by the time the discovery phase is complete).

Finally, the end-to-end effort may encompass quite a long time for a phased approach. The human toll that consolidation will take on the staff must be accounted for. If you have hundreds or thousands of applications and databases, the consolidation process may burn out employees. The schedule needs to think about ensuring that the staff can manage the consolidation without feeling overwhelmed.

### 4.5.4 Devising the Communication Strategy

The plan needs to define a communication schedule that includes regular communiqués that start a few months out and become more frequent as the date for the consolidation move gets closer. The communiqués are sent out from the central team to the groups that will be affected. Each group should get detailed and specific instructions of what they need to be aware of, what they should plan for, what they need to do (such as powering down their machines on the night that the consolidation happens), and when it needs to happen. Here is a sample timeline:

* For the initial stage of the project, when users will not be impacted, the information is high level and more of a message that a change is coming.
* For the week or two leading up to the consolidation, communicate more frequently. Usually the communications will be every other day, and then every day for the few days leading up to the event. These will be reminders and helpful information. Reinforce anything they must do before the cut-off day.
* Send out a final reminder just before the cut-off time.
* Most importantly, send out a final message when everything is back up and running with information on how to report problems if they are encountered. The goal is to make those who will be affected by the consolidation (directly or indirectly) feel comfortable and that their data is safe. If everything goes as planned, when they come in on Monday morning, they should not notice anything different.

For the actual consolidation effort execution, set up a master bridge line that will stay active from the time the consolidation starts until the main part of the consolidation is complete. At a minimum, the bridge must remain open until extended support effort is finished. The bridge line allows the IT staff (including DBAs) a guaranteed way of reaching those who need to know what is going on – good or bad. The same bridge line can be used by end users, but it is most likely best to use existing help desk procedures for support, and notify those staffing the help desk to contact the bridge line to escalate possible consolidation related-issues end users may be experiencing.

## 4.6 Testing

Testing is the only way to ensure success of the consolidation effort. Testing needs to occur at all stages of consolidation. This section details how testing must be thought about and done for all SQL Server consolidation efforts.

* When the new hardware is configured with an operating system, it should be fully verified and tested before SQL Server is installed. This includes running utilities like [SQLIO](http://www.microsoft.com/downloads/details.aspx?familyid=9a8b005b-84e4-4f24-8d65-cb53442d9e19&displaylang=en) and [SQLIOSim](http://support.microsoft.com/kb/231619). While SQLIO and SQLIOSim are not actual SQL Server workloads, they will show the limitations of the configured hardware. This is one of the few, if only times, that you will be able to tweak the configuration without impacting availability.
* As soon as possible, those responsible for the application should test against an environment that is configured like the proposed consolidated environment. Application-level testing is not something a DBA would do. However, a DBA can use information gathered during testing, such as how it uses the database, and its performance on the new target platform. The application testing at this stage also is used to identify problems and mitigate risk to take account in the final plan. If new hardware is being purchased, it is always a good idea to use that hardware for testing prior to it being turned into production. In addition to functionality tests, there should be load tests to ensure that the system will be able to carry the combined consolidated workload.
* Before any plans can be tested, the actual technologies that will be used for the plan should be put through their paces. DBAs and any other administrators responsible for part or all of the plan must be familiar with how the technologies involved work. For example, if the plan calls for using log shipping, the person responsible for executing the role change from the old server to the new server should know how to perform that task. The first time they see it should not be either during a dry run or worse, the night of the database/instance move.
* After those responsible for the technical work are comfortable with the technologies and the plans are considered complete, a dry run should occur. This also assumes that there is new hardware and the dry run will not impact production in a negative way. The goal of the dry run is to test the plan itself and document any gotchas and exceptions; you may even need to fix the plan, if aspects are missing or not working. Plan the timing of your dry run with care: The first time the consolidation is executed should not be the night of the planned consolidation. The dry run is arguably the last time to fix any problems with the plan. If not enough time is allowed between the dry run and the actual consolidation, it may be too late to address issues that arise. If a major problem is discovered for which a fix is not immediately available, it may delay the consolidation timeline.
* During the consolidation effort itself, testing must occur both at the server and application levels. Each stage will have its own set of tests to validate that the steps were performed correctly. The final stage of testing will be when the DBAs and all of IT turn the environment over to the application owner for final validation.

# 5 Administering a Consolidated SQL Server Environment

After the consolidation is complete, DBAs must manage the new environment. While what needs to be done is basically the same as before, because everything has changed, all processes must be rethought and reworked so they work as well, or better than, they did in the past. By centralizing the databases through consolidation, a DBA’s life should be easier, because he or she is not looking in many places to perform maintenance or check the status of SQL Server Agent jobs. However, DBAs should realize that they are most likely managing nearly the same amount of databases, just in a more centralized and easy-to-digest fashion.

## 5.1 Deployment Standardization

Going back to the earlier top-level drivers, control and consistency are the cornerstones for DBAs in a consolidated environment. Where there may have been a lack of standardization in the past, consolidation should solve that problem. In theory that is true, but it will take work to maintain. Besides standardizing how things are done in the environment, there must be enforcement of how applications and databases are deployed. That means that before any server is bought for a new application or deployment, the standard should be that its databases should always go into a consolidated environment unless there are extenuating circumstances. Methods should be put in place for detecting rogue installations and shutting them down or correcting them to conform to the deployment standards, such as running MAP or another discovery tool periodically.

## 5.2 Change Management

Before consolidation, most applications were isolated. In most cases, an issue affecting one application would not affect another. In a consolidated environment, the probability of one database affecting others is significantly higher due to the shared resource model. This means that any update – however minor it may seem – must be given have proper scrutiny. The scrutiny will come through two tried-and-true processes: testing (already covered in numerous places in this paper) and change management.

The best defense is a strong offense. While most companies have a change management process, others do not, or if they do, it may not be robust. In the new world order, no change should be applied without being run through a formal change-management process. All changes should have a way of being rolled back without affecting any other database. If the existing change management process works, do not change something that is not broken. If there is no change management process, put one in place before attempting consolidation.

## 5.3 Standard Administration

Post-consolidation, everything that DBAs did before must be done again: backups, index rebuilds, running DBCCs for health checks, and so on. Basic administration best practices are outside the scope of this white paper, and there are many good resources (both free and ones that would need to be paid for) that espouse the best practices for maintaining databases and instances. The challenge is that some of those tasks need to be changed or redesigned, because as with everything else related to a consolidated environment, doing something to one database can affect another.

### 5.3.1 Database Backups

When smaller numbers of databases (or even a single database) were housed on isolated instances, it was easy to have every full database backup job run at the same time on every instance in the environment. Now with 10, 20, 30, or more databases housed in the same instance, if all of the full backups kick off at the same time, disk I/O can be saturated. Database backups should be staggered to minimize contention. Use the information gathered in discovery about the backup jobs for each database as they existed in their original configuration to determine the schedule. It will take a little while to figure out the right schedule. As for transaction log backups, the same principle applies: Stagger them, but still do them.

Remember that the databases will grow over time (assuming that there is no process for pruning the data in the database), so the schedules will need to be monitored for contention and altered as necessary.

**Tip:** SQL Server 2008 Enterprise includes backup compression as a standard feature. Use this new feature to generate smaller database backups, speed up backups, and reduce I/O contention.

### 5.3.2 Index Rebuilds/DBCCs/Proactive Maintenance

The same general rules that apply to database backups apply to other common administrative tasks such as index rebuilds and proactive maintenance tasks such as running DBCC CHECKDB. Rebuilding all the indexes at the same time is generally a bad idea on a consolidated instance of SQL Server. Take the time to figure out the schedule of maintenance for each database in every consolidated instance. As with backups, use the information from discovery process to see how long each of the maintenance tasks took before, and schedule them accordingly in the new consolidated environment.

### 5.3.3 Monitoring

Monitoring of the environment also must change post-consolidation, especially if the environment was poorly monitored before consolidation. With multiple databases per instance and server, the risks are much larger for a consolidated environment. A problem with one database can affect others, in addition to the availability and/or performance of one or more applications. Two types of monitoring must be in place for a consolidation environment: standard and performance. All monitoring should be approached from a proactive, not reactive, standpoint.

You should store the information gathered from monitoring to establish baselines and growth patterns. As you may have noticed during the information-gathering phase for consolidation effort, sometimes things are harder without that information. If that data is monitored and analyzed on a continual basis, further opportunities for optimization and/or adjusted consolidation are possible.

**Note:** If the deployment is done using virtualization, the guest VMs need to be monitored as a server would be normally. Additionally, the hypervisor and the guest from the hypervisor view need to be monitored to ensure that the VMs are running well under the hypervisor. There are then three layers of monitoring: inside the VM (the same as if the VM was a physical server), the hypervisor, and the VM itself under the hypervisor. For example, on the hypervisor (assuming Hyper-V), monitor both the % Processor Time for the hypervisor as well as both of the following counters from the Hyper-V Logical Processor category: Total CPU Time (which covers the entire server) and the CPU time for each guest virtual processor.

#### 5.3.3.1 “Standard” Monitoring

Standard monitoring is the first line of defense, and it ultimately provides some of the most important information that DBAs and all administrators need to know. Implementing standard monitoring generally encompasses tasks like monitoring the log and status of SQL Server Agent jobs to look for problems, or being notified that a task was completed (either successfully or unsuccessfully). The best use of this type of monitoring is to filter on certain critical events (such as an 824 error in SQL Server) or specific messages or strings (such as “failure”). Most monitoring software has ways to notify administrators with these informational or status messages. SQL Server 2008 also has various ways of notification, such as utilizing the Database Mail feature for alerting. Be careful as to how much is reported back to DBAs and other administrators – too many messages with irrelevant information can be become noise, and then they will be ignored. When a message comes in, it should have relevance and importance.

#### 5.3.3.2 Performance Monitoring

Maintaining performance will always be an ongoing concern in every environment whether or not it is consolidated. The problem is that an administrator needs to know what normal usage of the system looks like at rest and when it is busy. Administrators need to be able to identify when things look wrong, even if it is just a slight deviation from normal. Again, this is being proactive; at the first sign of trouble, administrators can take immediate actions.

Post-consolidation, where the lines of delineation are not black and white, DBAs need to not only do the normal preventative maintenance outlined earlier in Section 6.3, but also be more aware of what is going on in the environment. The performance counters or DMVs that will be used are going to be the same before and after consolidation.

To assist in identifying a problem, thresholds for the counters that are being monitored should be defined, and if a threshold is exceeded, or if it is exceeded a certain number of times (one brief spike may be acceptable sometimes), administrators should be notified immediately.

**Tip:** Both standard and performance monitoring can be done using centralized monitoring tools such as System Center Operations Manager, Tivoli, and Unicenter. Many companies already use these tools, but they may not be monitoring SQL Server, and DBAs may not have access to the information.

### 5.3.4 Patching Servers and Instances in a Consolidated SQL Server Environment

It is inevitable that both the underlying servers and the SQL Server instances will need some sort of update installed during the lifecycle. Updates and patches should be inspected with more scrutiny than ever before in a consolidated environment. Consider this example: An application encounters a specific issue that is fixed by a SQL Server hotfix. The database used by that application is located in an instance with 39 other databases, and applying that hotfix will have a direct impact on the behavior of every application served by those 39 databases. Depending on the critical nature of the problem, there may be no choice but to apply it. However, it is may also be possible to wait for a cumulative update or a service pack to apply the fix, because it will have been fully tested in conjunction with all of the other fixes, and it will reduce risk to those other 39 databases.

Similarly, patching the operating system if there are multiple databases and instances must also be done with more care by the non-DBAs. “Just rebooting” a server to install a security patch now affects multiple applications all at the same time, not just one.

## 5.4 Constraining Resources

Post-consolidation, one of the major tasks for a DBA is to ensure the performance of the SQL Server environment. A concern of application owners and IT alike after consolidation is ensuring that a single instance, database, application, or query will not adversely affect another from a performance standpoint. This is a challenge in a nonconsolidated environment as well, but the effects are more glaring when you have 10, 20, or 100 other databases housed in a single instance.

While a topic such as constraining resources may seem more appropriate under performance, it is really an administrative task that is implemented after consolidation is done, and only if it is needed. Never use a potential tool for constraining resources if there are no performance issues; that could cause performance problems where there were none before. This section discusses the options that are available for constraining resources post-consolidation. There are three options that you can use to manage the performance characteristics. You may also want combine these options to suit the needs of your environment.

* Configure SQL Server, Windows, and hardware properly. This may include, but is not limited to, using options in SQL Server such as max server memory.
* Use the new Resource Governor feature of SQL Server 2008.
* Use Windows System Resource Manager (WSRM).

**Note:** Technically, using virtualization is another form of constraining resources, because a VM provides isolation for the guest operating system running SQL Server, but it is not discussed here because the pros and cons of virtualization and what it means from a performance standpoint were discussed earlier in this paper.

### 5.4.1 Proper Configuration

It should go without saying that if the consolidated architecture was planned an implemented correctly using a lot of the information in the whole of section 4 (especially 4.6), there should be no performance issues. It is much cheaper to get things right from the start than it is to fix performance issues only after they cause downtime. The right configuration boils down to three main points talked about throughout this document:

* Right disk configuration to optimize I/O throughput
* Enough memory on the physical server and for each SQL Server instance
* Enough processor power

### 5.4.2 SQL Server 2008 Resource Governor

Resource Governor is a new feature in SQL Server 2008 that provides the ability for the DBA to control what goes on within an instance. A key concept is that Resource Governor only knows about a single instance of SQL Server; if you deploy multiple instances, you will have to tweak Resource Governor in each instance separately. Resource Governor is useful for the following scenarios:

* Controlling runaway queries within an instance.

When multiple applications are now using a single instance of SQL Server, there is always the chance that one could have a query issued that consumes all of the given resources for an instance. Resource Governor can prevent that from happening.

* Managing differing workload executions within a single instance.

In the consolidated world, there will always be differing workloads under an instance with multiple databases. Resource Governor can help manage the different workloads by defining different resource pools and resource groups to have more predictable executions such that one database’s workload will not starve another.

* Assigning priorities to workloads.

An obvious advantage to knowing the workloads is to assign priorities for them. For example, the accounting department’s database may have been consolidated with others, but they need additional resources at month, quarterly, and year end. Resource Governor can help assign priorities to workloads to give some preferences.

By default, Resource Governor is disabled. To enable it, under the Management folder in SQL Server Management Studio, right-click, and then click **Enable**. It can also be turned on via Transact-SQL with the command ALTER RESOURCE GOVERNOR RECONFIGURE.

**Important:** One thing that the current version of the Resource Governor does not support is handling disk I/O allocation. So if a particular query or workload currently consumes a lot of I/O, you will need to mitigate it in more traditional ways, such as having the proper amount of I/Os at the disk subsystem level.

For more information, see [Managing SQL Server Workloads with Resource Governor](http://msdn.microsoft.com/en-us/library/bb933866.aspx) in SQL Server 2008 Books Online as well as the white paper [Using the Resource Governor](http://msdn.microsoft.com/en-us/library/ee151608.aspx).

### 5.4.3 Windows System Resource Manager

Windows System Resource Manager (WSRM) is an optional (and free) component of Windows Server 2003 and Windows Server 2008; the WSRM utility is not specific to SQL Server. WSRM can manage multiple aspects of the performance of a process, but it should only be used to constrain the percentage of processor used for SQL Server instances. WSRM will allow you to configure only up to 99 percent of the available CPU in any server; it reserves at least 1 percent for Windows.

**Tip:** For more information about WSRM, see “Windows System Resource Manager” [at this link](http://technet.microsoft.com/en-us/library/cc755056.aspx).

One of the most common uses of WSRM for SQL Server is when multiple instances are deployed, especially in a clustered condition. Earlier the failover condition was discussed, and concerns around ensuring that the instances and databases remain highly performing after failover were part of the discussion. Because there is no built-in mechanism within SQL Server to constrain processor to a percentage, WSRM can be used to provide that functionality if it is needed.

Consider this example: A stand-alone physical server has three SQL Server 2008 instances (A, B, and C) configured on it. At the end of the month, instance C is used by the accounting department for month-end closing. During this time instance C monopolizes the server. At all other times of the month, the three instances coexist peacefully. Moving Instance C to another server (physical or virtual) is not an option. End of the month it needs to be ensured that Instances A and B can server their respective application reasonably. Using WSRM you can create a Resource Allocation Policy that only kicks in during the last week of the month. This policy will give Instance C up to 50 percent (or whatever you decide) of the CPU, and you can divide the remaining percentage available between the other two instances (accounting for Windows overhead as well). Although Instances A and B may still be affected, they may not be affected as much, and Instance C may process a little slower but the end result is that *everyone* is still productive. The percentage mentioned is not a hard cap; it is a soft cap. What that means is if Instance A and B are not utilizing the 50 percent allocated to them, WSRM will allow Instance C to exceed 50 percent.

**Important:** If you decide to use WSRM and want to use it on a cluster, you must create the Resource Allocation Policy on each node of the cluster. Creating it on one node only will not work across all nodes of the cluster.

After a WSRM policy is enabled, it can be monitored by a set of counters. Two counters that you should look at if using WSRM are Target Managed CPU% (per process managed) and Actual Managed CPU% (per process managed).

## 5.5 Chargeback

A unique challenge that some IT environments have to deal with is *chargeback*. Chargeback is the ability to measure the use of resources (I/O, memory, processor) for a given server, instance, and/or database, and then turn them into some sort of cost model to charge customers (external or internal) for that utilization. There are two main approaches to chargeback:

* Chargeback by utilization, which has challenges (see the rest of this section)
* Chargeback by allocation, which ensures that resources are paid for no matter how it is utilized

Unfortunately, there is no formal documented process or feature built into SQL Server to assist with chargeback. SQL Server and Windows can provide a lot of information via features such as performance counters and DMVs. However, it may not always be able to provide all the information that you may be looking for with the appropriate granularity.

For chargeback, most companies are looking for measurements for some or all of these aspects of the SQL Server environment:

* Processor utilization
* Memory usage
* Disk I/O
* Disk space used
* Network usage
* Application usage as it relates to SQL Server (such as **tempdb**, number of users, number of transactions, and so on)

Some of these aspects are easier to measure than others, and they may not be as granular as desired for chargeback. For example, processor utilization is easy to measure at an instance level, but not per-database. While it is possible to run server-side SQL Server Profiler traces to get individual statistics about query usage per user and tie it back to a database, that process itself can add some overhead and can be complicated and difficult maintain. Monitoring CPU usage may be a good solution if chargeback is at the instance level; however, it may not work as well for database-level chargeback.

WSRM can also assist when it comes to CPU-based chargeback, but it may not provide the granularity of information you are looking for. WSRM stores accounting information on resource utilization. You can then use the information stored to show CPU utilization of that particular instance.

Chargeback based on disk statistics is much easier. The SQL Server dynamic management view **sys.dm\_io\_virtual\_file\_stats**, which is available in both SQL Server 2005 and SQL Server 2008, reports I/O statistics. Information derived using the execution of this DMV can be used for chargeback.

The white paper [Planning for Consolidation with Microsoft SQL Server 2000](http://technet.microsoft.com/en-us/library/cc966486.aspx) included scripts to provide some chargeback information. These have not been updated since the paper was published, but they may prove to be a good starting point for your own custom chargeback scripts for SQL Server 2008. Also, Buck Woody recently posted a two-part series on chargeback ([Part 1](http://www.informit.com/guides/content.aspx?g=sqlserver&seqNum=311) and [Part 2](http://www.informit.com/guides/content.aspx?g=sqlserver&seqNum=312)) based on SQL Server 2008. The code samples in Part 2 may also be useful.

Whatever is ultimately used for chargeback, it should be easy deploy and maintain, and it should not be a burden either from an overhead or administrative perspective.

**Important:** Think about how chargeback may or may not work for virtualized servers. In a virtualized environment, chargeback is even more obscured because the virtualized server (and its subsequent instances and databases) are already using a slice of the host CPU, memory, and I/O. What is measured in the virtual machine may not be completely accurate, because what is measured is relative to the virtual machine, not the host.

# 6 Conclusion

Done properly, SQL Server consolidation can ease administration, reduce costs, increase capacity and availability, and make your deployments more agile. Whether you are consolidating on a physical or a virtual platform such as Hyper-V, there is not a one-size-fits-all approach; the eventual architecture will probably be a hybrid of different approaches based on the various factors discovered. Make no mistake about it; virtualization is here to stay and a valid option for consolidation. Due to the scalability and availability enhancements to the combination of SQL Server 2008 and Windows Server 2008, it is a good choice as a platform for consolidation. While there is no wizard to magically consolidate existing databases and instances, with the right information and plan, the process should be smooth and the end result optimal, resulting in an architecture that will serve the business’ growth for years to come.

# 7 Links for More Information

This section contains the embedded links used throughout the paper and documents other sources of information that were not embedded but can also serve as references for a SQL Server consolidation effort. Following the table are some links to general information about SQL Server and information about providing your feedback.

*Table 10. List of links*

|  |  |  |
| --- | --- | --- |
| **Section** | **Detail** | **Link** |
| N/A | White paper, *SQL Server Consolidation at Microsoft* | <http://technet.microsoft.com/en-us/library/dd557540.aspx> |
| N/A | White paper, *Green IT in Practice: SQL Server Consolidation in Microsoft IT* | <http://msdn.microsoft.com/en-us/architecture/dd393309.aspx> |
| N/A | SQLCAT Blog Post, *Useful links for upgrading to SQL Server 2008* | <http://sqlcat.com/msdnmirror/archive/2009/03/27/useful-links-for-upgrading-to-sql-server-2008.aspx> |
| 1 | Microsoft Operations Framework | <http://technet.microsoft.com/en-us/library/cc506049.aspx> |
| 1 | White paper (higher level), *Server Consolidation with SQL Server 2008* | <http://download.microsoft.com/download/a/c/d/acd8e043-d69b-4f09-bc9e-4168b65aaa71/SQL2008SrvConsol.doc> |
| 4.1.1 | Microsoft Assessment and Planning Toolkit | <http://technet.microsoft.com/en-us/library/bb977556.aspx> |
| 4.1.2 | SQL Server Books Online topic, “Dynamic Management Views and Functions (Transact-SQL)” | <http://msdn.microsoft.com/en-us/library/ms188754.aspx> |
| 4.3.1.1 | SQL Server Books Online topic, “Instance Configuration” | <http://msdn.microsoft.com/en-us/library/ms143531.aspx> |
| 4.3.1.2 | Server Virtualization Validation Program | <http://www.windowsservercatalog.com/svvp.aspx?svvppage=svvp.htm> |
| 4.3.1.2 | Whitepaper, *Windows Server® 2008 R2 Hyper-V™ Live Migration* | http://www.microsoft.com/DownLoads/details.aspx?familyid=FDD083C6-3FC7-470B-8569-7E6A19FB0FDF&displaylang=en |
| 4.3.1.2 | Knowledge Base Article 897615 “Support policy for Microsoft software running in non-Microsoft hardware virtualization software” | <http://support.microsoft.com/kb/897615/en-us> |
| 4.3.1.2 | Knowledge Base Article 957006 “Microsoft server software and supported virtualization environments” | <http://support.microsoft.com/kb/957006> |
| 4.3.1.2 | Knowledge Base Article 956893 “Support policy for Microsoft SQL Server products that are running in a hardware virtualization environment” | <http://support.microsoft.com/?id=956893> |
| 4.3.1.2 | White paper, *Running SQL Server 2008 in a Hyper-V Environment: Best Practices and Performance Considerations* | <http://download.microsoft.com/download/d/9/4/d948f981-926e-40fa-a026-5bfcf076d9b9/SQL2008inHyperV2008.docx> |
| 4.3.3.1 | SQL Server 2008 edition comparison Web page | <http://www.microsoft.com/sqlserver/2008/en/us/editions.aspx> |
| 4.3.3.3 | How to Buy SQL Server 2008 Web page | <http://www.microsoft.com/sqlserver/2008/en/us/how-to-buy.aspx> |
| 4.3.3.3 | How to Buy Windows Server 2008 Web page | <http://www.microsoft.com/windowsserver2008/en/us/how-to-buy.aspx> |
| 4.3.6.2 | Knowledge Base Article 956893 | See earlier link |
| 4.3.6.2 | White paper, SQL Server 2008 Failover Clustering | <http://download.microsoft.com/download/6/9/D/69D1FEA7-5B42-437A-B3BA-A4AD13E34EF6/SQLServer2008FailoverCluster.docx> |
| 4.3.7 | SQL Server Books Online topic, “Maximum Capacity Specifications for SQL Server” | <http://msdn.microsoft.com/en-us/library/ms143432.aspx> |
| 4.3.7.1 | SQL Server Books Online topic, “affinity mask Option” | <http://msdn.microsoft.com/en-us/library/ms187104.aspx> |
| 4.3.7.3 | White paper, *Predeployment I/O Best Practices* | <http://technet.microsoft.com/en-us/library/cc966412.aspx> |
| 4.3.7.3 | White paper, *Disk Alignment Best Practices for SQL Server* | <http://msdn.microsoft.com/en-us/library/dd758814.aspx> |
| 4.3.7.3 | Blog, Jimmy May | <http://blogs.msdn.com/jimmymay/default.aspx> |
| 4.3.7.3.4 | White paper, *Working with tempdb in SQL Server 2005* | <http://technet.microsoft.com/en-us/library/cc966545.aspx> |
| 4.3.7.3.4 | SQL Server Books Online topic, *“Collation and Unicode Support”* | <http://msdn.microsoft.com/en-us/library/ms143503.aspx> |
| 4.3.9 | White paper, SQL Server 2008 Upgrade Technical Reference Guide | <http://www.microsoft.com/downloads/details.aspx?FamilyID=66d3e6f5-6902-4fdd-af75-9975aea5bea7&displaylang=en> |
| 4.4.4 | SQL Server Books Online topic, *“Deprecated Database Engine Features in SQL Server 2008”* | <http://msdn.microsoft.com/en-us/library/ms143729.aspx> |
| 4.4.4 | SQL Server Books Online topic, “ALTER DATABASE (Transact-SQL)” | <http://msdn.microsoft.com/en-us/library/ms174269.aspx> |
| 4.4.6 | Knowledge Base Article 314546 “How to move databases between computers that are running SQL Server” | <http://support.microsoft.com/kb/314546> |
| 4.4.6 | Knowledge Base Article 246133 “How to transfer logins and passwords between instances of SQL Server” | <http://support.microsoft.com/kb/246133/> |
| 4.4.6 | Knowledge Base Article 240872 “How to resolve permission issues when you move a database between servers that are running SQL Server” | <http://support.microsoft.com/kb/240872> |
| 4.4.6 | Knowledge Base Article 918922 “How to transfer the logins and the passwords between instances of SQL Server 2005” | <http://support.microsoft.com/kb/918992/> |
| 4.4.6 | Knowledge Base Article 224071 “How to move SQL Server databases to a new location by using Detach and Attach functions in SQL Server” | <http://support.microsoft.com/kb/224071> |
| 4.6 | SQLIO utility | <http://www.microsoft.com/downloads/details.aspx?familyid=9a8b005b-84e4-4f24-8d65-cb53442d9e19&displaylang=en> |
| 4.6 | SQLIOSim utility | <http://support.microsoft.com/kb/231619> |
| 5.4.2 | SQL Server Books Online topic, *“Managing SQL Server Workloads with Resource Governor”* | <http://msdn.microsoft.com/en-us/library/bb933866.aspx> |
| 5.4.2 | White paper, *Using the Resource Governor* | <http://msdn.microsoft.com/en-us/library/ee151608.aspx> |
| 5.4.3 | Documentation, Windows System Resource Manager | <http://technet.microsoft.com/en-us/library/cc755056.aspx> |
| 5.5 | White paper, *Planning for Consolidation with Microsoft SQL Server 2000* | <http://technet.microsoft.com/en-us/library/cc966486.aspx> |
| 5.5 | Article, “SQL Server Chargeback Strategies, Part 1” | <http://www.informit.com/guides/content.aspx?g=sqlserver&seqNum=311> |
| 5.5 | Article, “SQL Server Chargeback Strategies, Part 2” | <http://www.informit.com/guides/content.aspx?g=sqlserver&seqNum=312> |

<http://www.microsoft.com/sqlserver/>: SQL Server Web site

<http://technet.microsoft.com/en-us/sqlserver/>: SQL Server TechCenter

<http://msdn.microsoft.com/en-us/sqlserver/>: SQL Server DevCenter

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

* Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
* Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

[Send feedback](mailto:sqlfback@microsoft.com?subject=White%20Paper%20Feedback:%20Consolidation%20Using%20SQL%20Server%202008).