



Whitepaper

**Scaling SharePoint 2007:
Storage Architecture (R2)**

Russ Houberg

June 23, 2009

Table of Contents

| | |
|---|-----------|
| OVERVIEW | 3 |
| PURPOSE | 3 |
| BACKGROUND | 3 |
| INTRODUCTION | 4 |
| GENERAL RECOMMENDATIONS | 5 |
| <i>DAS vs SAN vs NAS</i> | 5 |
| <i>Disk Partition Alignment</i> | 5 |
| <i>Virtualization vs Bare Metal</i> | 6 |
| OPTIMIZING TEMPDB DATABASE PERFORMANCE | 9 |
| <i>Recommendations</i> | 9 |
| OPTIMIZING CONTENT DATABASE PERFORMANCE | 10 |
| <i>Recommendations</i> | 10 |
| OPTIMIZING SEARCH & SSP DATABASE PERFORMANCE | 12 |
| <i>Recommendations</i> | 12 |
| OTHER SHAREPOINT DATABASES | 14 |
| <i>Recommendations</i> | 14 |
| OTHER I/O PERFORMANCE CONSIDERATIONS | 15 |
| <i>Physical Volume File Fragmentation</i> | 15 |
| <i>Database Index/Statistics Fragmentation</i> | 15 |
| <i>WSS Column Indexes vs Large Content Databases</i> | 15 |
| IMPLICATIONS OF A VERY LARGE CONTENT DATABASE | 17 |
| MONITORING SHAREPOINT I/O PERFORMANCE | 18 |
| SQL SERVER MANAGEMENT FOR SHAREPOINT ADMINISTRATORS | 19 |
| <i>Move tempdb</i> | 19 |
| <i>Add tempdb data files and set all data file sizes (in KB)</i> | 19 |
| <i>Pre-Create a Database (for Content or Search databases)</i> | 20 |
| <i>Move a database file from a source volume to a destination volume</i> | 21 |
| <i>Redistribute content from a single large database data file to multiple data files</i> | 22 |
| SUMMARY | 23 |
| <i>Disk I/O Impact on SharePoint Performance</i> | 23 |
| <i>The Bottom Line</i> | 23 |
| APPENDIX A: REFERENCES | 24 |

Overview

Purpose

The purpose of this whitepaper is three fold. First, this document is intended to provide prescriptive guidance targeted at the storage subsystem for a highly scalable SharePoint implementation in order to facilitate architectural planning. Second, it is intended to provide techniques for monitoring scalability metrics for existing SharePoint implementations. Finally, it provides techniques for modifying storage architecture of SharePoint systems that are suffering from poor performance due to the fact that the I/O burden on the storage subsystem has exceeded its capability.

Background

Back in the 1990s, GM came up with a new slogan for their Pontiac sport sedan line of cars. The related commercial centered on a Hawaiian guy whose grandfather preached the “E Aho Laula” philosophy. This phrase translates to “Wider is Better”. His grandfather was talking about surfboards, Pontiac was talking about cars, and in this whitepaper, we’re talking about the Storage Subsystem for SharePoint.

Most experienced client server developers know that poor database architecture can absolutely wreck the performance of an application. A poor relational design, inefficient stored procedures, or even improperly indexed tables can have a dramatically negative effect on how SQL Server is able to serve data to your application. The same is true not only of logical design but also of physical storage implementation, particularly when large databases are present. Even with a properly architected database, the sheer storage requirements of large database data files cause SQL Server to spend significant additional time to seek the requested data.

While much of the specific detail in the prescriptive guidance of this whitepaper can be found in the [SharePoint Capacity Planning Guide](#), this document will strive to explain a little bit of the “how and why” that back up the recommendations.

Introduction

Many SharePoint instances that are in place today unfortunately “evolved” into production. Often times they exist as the result of an untrained person just clicking “next” through the WSS or MOSS installer application. That said, even an experienced SharePoint Administrator will often fail to get the most out of the farm they are building.

Today’s SharePoint Administrator is expected to be a platform expert, storage expert, SQL Server expert, security expert, IIS expert, software developer and more all rolled up into one person. It’s a lot to ask. So it’s understandable that even very experienced administrators can miss some very important points when architecting and implementing a SharePoint farm.

It’s not uncommon for a technically savvy business user to fire up desktop computer and install WSS. Over time, that install becomes very important for that person’s business unit. All the sudden you’re running a mission critical system on a low performance desktop computer under somebody’s desk with absolutely zero fault tolerance or disaster recovery consideration.

Another possibility might be that your company was cutting edge 2 years ago and installed the RELEASE version of SharePoint 2007... at the end of 2006. It was revolutionary, light years beyond what SharePoint 2003 had brought to the table. But being on that bleeding edge, nobody knew what they were doing. Prescriptive guidance didn’t exist back then. To save a buck or two SharePoint was installed on a small mirrored (system?) partition and your SharePoint farm performance has suffered.

The primary goal of this document is to provide specific SharePoint storage architecture guidance, discuss the reasons behind the recommended limits for SharePoint databases, and discuss how we might extend those limits by implementing SharePoint databases in a more granular way.

In the end, when you’re weighing the cost of new storage hardware with the performance requirements for your user base, just remember that when it comes to I/O performance... “Wider is Better”.

As we begin, it is important to provide a bit of general guidance related to storage performance. After that, optimizations for the databases that most directly impact the functional performance characteristics of SharePoint will be discussed. The database sections attempt to provide prescriptive guidance for storage architecture in the “order of impact” to SharePoint performance.

The document will then continue by reflecting on the 100 GB content database size recommendation from Microsoft. Finally, a series of SQL scripts will be provided to help administrators implement the recommendations or modify existing databases as prescribed in this document.

General Recommendations

Before starting into specific database guidance, there are a few general recommendations that need to be addressed.

DAS vs SAN vs NAS

Network Attached Storage (NAS) – Just to get this out of the way... don't do it! It might even be prudent to include iSCSI based SANs in this category. Read and write I/O latency is typically going to be way too high on a storage subsystem based on NAS or iSCSI technology. Nothing is absolute so there may be exceptions, but typically these storage systems should be avoided.

Storage Area Network (SAN) – SANs are an interesting proposition. Management wants to buy them because they can spread the cost out over multiple business systems and thus multiple cost centers. But that same concept can also get you into trouble with SharePoint. The SharePoint storage platform will be highly taxed in a high scale SharePoint environment.

It is true that you get to spread I/O across just about every disk in that big box you scored, but what happens when a huge report calculation starts crunching a data warehouse (also on your SAN) or those regular backups of that massive Exchange storage introduce regular load on the SAN? Worse yet, what happens when a full crawl kicks off in SharePoint and now the CEO is experiencing email delays when waiting for that important message?

I would argue that a SAN can't be correctly justified in most SharePoint deployments. The SharePoint storage subsystem should be independent of all other business application storage subsystems if at all possible. But then without the cost sharing, the SAN solution basically becomes cost prohibitive in most cases. That leads me to what I would recommend in most cases.

Direct Attached Storage (DAS) – That's right folks. DAS can be justified much easier than a SAN solution. It's cheaper, easier to maintain, and the SharePoint/SQL Administrator has much finer control over LUN performance. With a SAN you typically select a performance "profile" and the SAN does all the black magic to make it work. With a DAS, you get direct control over the SCSI array such that you know exactly how many SCSI channels and disk spindles to work with and you have complete control over how they are allocated. DAS is simple and effective storage. Also, it will be less expensive than a SAN solution in most cases.

Also in existence are expandable storage arrays that are basically "advanced" DAS systems. These systems function like a DAS but they provide the added benefit of dual node (cluster) support. If you're considering clustering your SQL Server these enhanced devices may handle this requirement without the higher cost of a full blown SAN solution.

Disk Partition Alignment

Partition alignment occurs when the starting location of a partition is aligned with the stripe unit boundary in the disk partition that is created on a given volume. When a volume cluster is created OVER a stripe unit boundary instead of NEXT to the stripe unit boundary, the partition is said to be misaligned. This condition can cause **significant** I/O performance degradation, sometimes requiring two I/O operations where there should have only been one!

Windows Server 2003 has been a great platform for SharePoint in recent years. But these days Windows Server 2008 is a much more wise choice. One of the big reasons for this is this concept of partition alignment, particularly on the SQL Server. Partition alignment in Windows Server 2003 is often times a manual effort. Rather than rehash existing information, it's better to just reference the source material directly. Please read this [Microsoft KnowledgeBase article](#) regarding partition alignment in Windows Server 2003.

Now that Windows Server 2008 is (or should be) the standard deployment platform, this is not as big of an issue because Windows Server 2008 facilitates partition alignment out of the box.

Another amazing resource that covers this concept in excellent detail is [this SQL Server technical article](#) on MSDN. Jimmy May and Denny Lee do a fantastic job of covering this topic in such a way that the "jack-of-all-trades" server administrator can understand!

Disk partition alignment is a far more common problem than many administrators realize. It is common to encounter a SharePoint System with a higher level of technical specs but for some reason it just seems to underperform compared to a lesser system. Disk partition alignment, particularly on the SQL Server or file server, is one explanation as to why this occurs.

Virtualization vs Bare Metal

Virtualization... The heaven sent answer to scalability, high available, and just plain getting the most bang for your server buck! Everybody wants to virtualize these days, and with good reason as ROI is easy to prove.

Ok, so virtualization isn't "directly" related to Storage Architecture, but it does have an impact on IOPS. Just like with CPU and Memory resources, there is typically a performance penalty to storage I/O in a virtualized environment. So we'll use that as an excuse to touch on the subject.

The truth is that every SharePoint server in the farm "can" be virtualized, but that doesn't necessarily mean that you should. It's a classic case of "it depends". So rather than just expressing an opinion here, let's look at some scenarios of when it's a good idea to virtualize and when it's not.

Scenario 1: An organization has one or more small WSS farms for departmental portals. There isn't really all that much content but the portals are used regularly by hundreds or thousands of employees.

- The SharePoint Web Front End (WFE) servers can definitely be virtualized. Even if there is enough traffic to warrant multiple load balanced WFEs per farm, virtualization presents a nice story here because the processing overhead that it adds does not significantly affect the end user experience.
- The SQL Server is a toss-up based on exact circumstances, particularly the number of users and quantity of content. If at least one of these factors is fairly low (maybe a couple hundred users or maybe the total corpus is only 80GB in size), then the SQL Server "could" be virtualized. But if there are thousands of users hitting millions of content items in ½ of a terabyte or more, then it will always be better to leave SQL Server as bare metal.

Scenario 2: An organization has a medium MOSS farm that serves as the corporate intranet and collaboration portal. They have 500 users that hit the load balanced WFEs (for high availability). The content from five (5) departmental site collections (each in their own content database) results in a corpus size of 3 million documents that consumes about 350GB of storage space.

- The SharePoint Web Front End (WFE) servers can definitely be virtualized. The key here is that for there to be true high availability, the two WFE guests must execute from separate virtual machine host servers. If the WFEs also serve as Query Servers (and in this case, I think they should), then the data volume for the content index share should be optimized for high read performance (like RAID 5).
- The Index Server is going to be taxed, but it can be virtualized. It will be necessary to assign as many CPU cores to this machine as possible. Note that some virtualization products do not show improved performance beyond a certain number of cores. My experience is that anything more than 4 cores is a waste of allocation in most cases. Also, a decent quantity of RAM must be allocated to this server, probably 8GB or so. Finally, the storage volume for the content index should be optimized for high read and write performance (like RAID 10).
- Interestingly enough, the SQL Server could be virtualized in this environment. Again, potentially 4 CPU cores and 8GB RAM should be assigned. Also, the server will be fairly taxed during full crawl operations, so be sure to schedule that during off hours. But there should be enough horsepower to complete the full crawl in an evening or weekend before end users are again hitting the system. End user experience will still be solid during normal portal surfing. It is important to follow the database storage architecture outlined in this whitepaper in order to optimize performance.

Scenario 3: An organization has a medium or large MOSS farm that serves as the corporate intranet, collaboration portal, and document imaging repository. They have between 50 and 5,000 concurrent users that hit two (2), three (3), or (4) load balanced WFEs (for high availability and high performance). The content from five (5) departmental site collections (each in their own content database) results in a collaboration corpus size of 3 million documents that consumes about 350GB of storage space. The imaging repository consists of 15 million documents that are being migrated in from a legacy repository and a day-forward growth rate of 8,000 documents per day (2.1 million documents per year).

- The SharePoint Web Front End (WFE) servers can be virtualized. The key here is that for there to be true high availability, the WFE guests must execute from separate virtual machine host servers. I would expect to see 2 CPU cores and somewhere around 4GB of RAM for each virtualized WFE. Storage isn't tremendously important because in this type of farm, the Query Server role (and thus the content index) would likely be separated from the WFE role.
- The Query Server(s) can be virtualized. Again 2 CPU cores and between 4GB and 8GB RAM should be allocated. The key is that the storage volume that will contain the content index share should be optimized for read performance (RAID 5).
- The Index Server can DEFINITELY NOT be virtualized. This machine is going to get hammered during a full crawl. Given the fact that the Index Server in a MOSS 2007 farm is the only machine for which "scale out" is not an option (for a given content source), this server needs to be a monster PHYSICAL machine. Do not pass Go, do not collect 200. Proceed directly to your favorite hardware vendor and purchase a

Dual-Quad Core (8 CPU cores, 16 cores if you can get it cheap enough) and start with 16GB of physical RAM. A large corpus is extremely hard on the index server as each document must be cracked open and indexed individually and on 20 million or so documents, this can take quite some time. If the Index Server is virtualized in this environment, then failure is imminent as search results WILL be untrustworthy. If in future versions of SharePoint, Microsoft allows for “scale-out” options, then this recommendation may be revisited.

- The SQL Server should NOT be virtualized. Again, during full crawl operations the Index Server and SQL Server(s) work together to accomplish the crawl. Having as many CPU cores as possible for thread processing and as much RAM as possible for caching is imperative. In fact, in addition to being physical machines, it might be prudent to have two (2) SQL Servers. Place the SharePoint Configuration DB and content DBs on one server and the SSP and Search databases on the other. Both should have at least 8 CPU cores and 16 GB of RAM. All other storage architecture guidance in this whitepaper should be followed for the SharePoint databases.

Virtualization Performance Tips: If you do decide to virtualize, here are a few tips that can save you some performance pain:

- Try to allocate RAW LUNs for the high performance volumes that are necessary for the SQL Server databases and content indexes. I understand that this may affect your ability to “snapshot” a VM but it is the only way to fly if performance is important.
- In the absence of RAW storage volumes, you should pre-allocate the storage space and never use dynamically expanding disks. During performance tests, pre-allocated storage space performs very closely to raw storage volumes whereas the performance of dynamically expanding disks is quite terrible.
- If you’re looking to “try out” virtualizing your SQL Server before reverting to bare metal, then when installing SharePoint, use a SQL Alias. After you build out your virtual SQL Server, build out your base SharePoint servers. But before you install SharePoint bits, use the SQL Server Configuration Manager to create an “alias” to your SQL Server. That way, if the virtualized SQL Server underperforms, you can stand up a physical SQL Server (with a different name). Then when you’re ready to switch over, you just move the databases to the new server and update the alias on all of the SharePoint servers.

Optimizing TempDB Database Performance

The next few sections of this whitepaper will concentrate on optimizing performance for each type of database (or log) that affects SharePoint performance. The databases will be discussed in the order of impact with the first and most important database being the tempdb.

The tempdb database is used by SQL Server to store intermediate results during query and sort operations. SharePoint, in particular, uses the tempdb extensively during query operations. This means that poor tempdb performance can have a dramatic impact on the SharePoint end user experience

The following recommendations are listed in order of least resource intensive (read lower cost) to most resource intensive. Implementing all recommendations will yield the highest performing configuration.

Recommendations

- Pre-size your tempdb. Since autogrowth will cause this database to grow significantly anyway, this is essentially a free option. So there is no reason not to do this! Dynamic data file extents are an expensive SQL Server operation and lead to fragmentation. When you consider that every time SQL Server is restarted, the tempdb is reset to its original size, you can see how SQL Server can spend a lot of time extending the tempdb. The tempdb size total size should be preset to 25% of the predicted size of your largest database. "Autogrowth" should remain enabled. If the tempdb data file(s) autogrow beyond your predicted size, then they should be permanently resized again taking into account maximum observed growth.
- Add tempdb data files such that the number of data files is equal to or less than the number of CPU cores present in the SQL Server machine. There should be greater than one file for multi-core machines. Each data file (primary .mdf and additional .ndf(s)) should be pre-sized to be equal in size.
 - The number of data files can be calculated as between $\frac{1}{4}$ to $\frac{1}{2}$ the number of CPU cores. Typically, we see 2 data files being sufficient for a 2 or 4 core SQL Server, and 4 data files being ideal for 8 or 16 core SQL Server.
 - The size of each data file can be calculated using this formula: $[\text{MAX DB SIZE (KB)}] \times [.25] / [\# \text{ DATA FILES}] = \text{DATA FILE SIZE (KB)}$
- Place the tempdb database file(s) on a RAID 10 logical unit if at all possible.
- With a multi-core SQL Server, you should have multiple data files. If possible, separate each data file to separate logical units consisting of unique physical disk spindles. The tempdb log file should also be placed on a (preferably unique) LUN of RAID 10 space.
- Do not allow other data files from other databases to be placed on any of the tempdb LUNs if at all possible.

Optimizing Content Database Performance

SharePoint Content databases contain the bulk of all content in SharePoint instance. In WSS v3.0 (and MOSS), not only is all of the content metadata stored in the database, but the binary files themselves related to document library records are also stored in the database. That statement should be qualified with the knowledge that a hotfix (and SP1) is available that will allow the externalization of content.

The point, however, is that SharePoint content databases grow very rapidly when used in document imaging system or in file share replacement scenarios. With this in mind it is important to consider growth and overhead factors when determining how much content will eventually be stored in a given content database.

Given the wide range of file plan variables for any given organization, the formulas for predicting content database size are a moving target and as such are outside the scope of this document. However, once raw file storage quantities are projected, the following formula may be used to estimate database overhead:

Database Overhead Formula:

Low: $1.2 * [\text{Raw Storage Size}] = \text{ContentDB Size}$

High: $1.5 * [\text{Raw Storage Size}] = \text{ContentDB Size}$

Once the storage requirements have been determined, once again attention should be paid to maximizing I/O performance of the content databases.

The following recommendations are listed in order of least resource intensive (read lower cost) to most resource intensive. Implementing all recommendations will yield the highest performing configuration.

Recommendations

1. Pre-construct and pre-size your content databases. "Autogrow" should be left on. The content databases should be monitored so that the databases don't exceed predetermined limits.
2. Place the content database file(s) on a RAID 10 logical unit(s) if at all possible. If this is not possible, then RAID 5 is an acceptable alternative.
3. With a multi-core SQL Server, the content database should consist of multiple data files.
 - o The number of data files can be calculated as between $\frac{1}{4}$ to $\frac{1}{2}$ the number of CPU cores. Typically, we see 2 data files being sufficient for a 2 or 4 core SQL Server, and 4 data files being ideal for 8 or 16 core SQL Server.
 - o The size of each data file can be calculated using this formula: $[\text{MAX DB SIZE (KB)}] \times [.25] / [\# \text{ DATA FILES}] = \text{DATA FILE SIZE (KB)}$
4. If possible, place each data file onto separate LUNs consisting of unique spindle sets.

5. Place the log files for all content databases on a shared LUN consisting of its own set of isolated spindles. Log files should be pre-sized in 4GB or 8GB segments to prevent a form of fragmentation. There isn't a rule of thumb to predict log file sizes as it is dependent on backup operations of a given organization. Start with 4GB to 8GB per content database depending on the database size and then monitor frequently along with backup timing. If a pre-sized log begins to grow, then pre-size again, adding an additional 4GB or 8GB chunk as necessary or execute backups more frequently.
 - **NOTE:** For massive (repeatable) content database loading operations (migrations, etc), consider using taking a full backup and then switching to "Simple" recovery model for that content database until the mass upload is complete. Then switch back to "Full" recovery model and take another full backup. This will prevent unnecessary and often times "out of control" log growth. If the load operation is not repeatable upon failure, "Full" recovery model is still the best choice even during the load operation.
6. Don't allow data files from different content databases to share LUN space. If this is not possible then stripe the data files from multiple content databases across multiple logical units. So if you have 2 content databases and 4 LUNs, then both database "01.mdf" data files would be on LUN1 and then both database "02.ndf" files would be on LUN2, the database "03.ndf" files would be on LUN3, and finally the "04.ndf" files for both databases would be on LUN4.

Optimizing Search & SSP Database Performance

The SharePoint search database is a very important and often ignored component of SharePoint scalability. Consider the fact that stored in the search database is every metadata value for every document, in every library, in every site, in every site collection, in every web application that is served by the related Shared Service Provider. Also stored in the search database are the history log, search log, crawl statistics tables, links (anchor) tables and statistics tables.

Often times a Shared Services Provider serves many site collections consisting of many content databases. If each of those content databases are 100GB or more in size, then it must be considered that the sum total of the metadata along with the related anchor and crawl information can grow substantially. As an example a single search database that services queries and full crawls of 50 million content items (Microsoft recommended limit for an Index Server) can have a search database that is over 500GB in size!

Use the following formula to calculate how much disk space is needed for the search database:

GB of disk space required = Total_Content_Size (in GB) x File_Size_Modifier x 4

where File_Size_Modifier is a number in the following range, based on the average size of the files in your corpus:

- 1.0 if your content consists of very small files (average file size = 1KB).
- 0.12 if your content consists of moderate files (average file size = 10KB).
- 0.05 if your content consists of large files (average file size 100KB or larger)

As for the SSP database, I didn't realize its importance for a long time. As it turns out, it's integral during both crawl processing and query operations. This is because, among other things, mapping of managed properties to crawled properties is stored in the SSP database. So this database is hit a lot more than I realized. So I'm moving the "SSP" database out of the "base" database list and to a more important location. The storage architecture for the Search database can also include the SSP database. The SSP database typically is not very large but can be impacted by personalization services (number of user profiles) as well as whether or not your farm takes advantage of the BDC, Excel Services or Project Server. Typically, the SSP database size won't exceed 4GB.

For these reasons, a robust I/O subsystem is crucial for the search and SSP databases. The following recommendations are listed in order of least resource intensive (read lower cost) to most resource intensive. Implementing all recommendations will yield the highest performing configuration.

Recommendations

1. Add a 2nd file group to your search database! This is an option that wasn't supported when the "first version" of this whitepaper was released. But Microsoft now acknowledges that the Search database supports 2 distinctly different operations, (1) crawl processing and (2) end user queries. That means there can be physical separation of database structures in this database. Rather than to rehash existing content, for large MOSS implementations (millions or 10s of millions of documents), [follow this procedure](#) to add a 2nd file group! If your search database was I/O bound before, then crawl and search performance will be improved!

2. Pre-construct and pre-size your search file group files and SSP database files. "Autogrow" should be left on. These databases should be monitored so that the database file groups don't exceed predetermined limits.
3. Place the search and SSP database file(s) on RAID 10 logical unit(s). RAID 10 is practically a requirement for large scale systems. The search database is VERY read and write intensive. RAID 10 is STRONGLY recommended.
4. With a multi-core SQL Server, the search database should consist of multiple data files for each file group.
 - The number of data files can be calculated as between $\frac{1}{4}$ to $\frac{1}{2}$ the number of CPU cores. Typically, we see 2 data files being sufficient for a 2 or 4 core SQL Server, and 4 data files being ideal for 8 or 16 core SQL Server.
 - The size of each data file can be calculated using this formula: $[\text{MAX DB SIZE (KB)}] \times [.25] / [\# \text{ DATA FILES}] / [\# \text{ FILE GROUPS (2)}] = \text{DATA FILE SIZE (KB)}$
 - The size of each SSP data file should be 1GB unless your SSP database is abnormally large.
5. Do not place any other database data files on any logical unit where search database data files reside. If possible, even try to ensure that the RAID 10 logical units for the search database data files do not share their physical spindles with other databases. **NOTE** that for the separation of the two search database file groups to be effective, they must be located on DIFFERENT SPINDLE SETS!
6. Place the search and SSP database log file on its own LUN. This LUN should be RAID 10.
 - The search DB log file should be pre-sized in 4GB or 8GB segments to prevent a form of fragmentation. There isn't a rule of thumb to predict log file sizes as it is dependent on backup operations of a given organization. Start with 4GB to 16GB depending on corpus size and then monitor frequently along with backup timing. If the log size is exceeded, then pre-size again, adding an additional 4GB or 8GB chunk or execute backups more frequently.

Other SharePoint Databases

The remaining SharePoint “base” databases are not nearly as read or write intensive as the tempdb, content, ssp, or search databases. These databases include:

- SharePoint Configuration Database
- SharePoint Admin Content Database
- SSP Admin (Site Collection) Content Database

Finally, resources can be shared! Following these recommendations will yield a good cost vs performance balance for these lesser taxed databases:

Recommendations

1. There is no need for multiple data files per database.
2. All database data files for these databases can share a logical unit on shared physical disk spindle resources.
3. The log files for these databases can share a logical unit on a shared physical disk spindle.

Other I/O Performance Considerations

Physical Volume File Fragmentation

Physical volume file fragmentation is often ignored and can play a key role in any observed database performance degradation. Once database sizes have been determined, the actual size of the logical units being created must be considered. Unfortunately, high speed/fault tolerant storage isn't cheap so this is an important decision.

Logical units should be large enough to contain the data that will be stored plus at least 25%. This is to allow for enough space for a defragmentation routine to optimize file structures. 50% may even be a safer recommendation as there are always storage requirements that aren't thought of in the beginning.

You can manually execute defragmentation operations as database performance begins to degrade or you can implement a server class defragmentation solution on a scheduled basis.

Database Index/Statistics Fragmentation

Out of the box, SharePoint implements a timer job called SPDatabaseStatisticsJobDefinition that is supposed to maintain the efficiency of statistics in SharePoint databases. However, this job doesn't do much for indexes and thus isn't always as effective as a good quality maintenance plan. Also, large or long running operations may necessitate the need for an index rebuild upon completion. For example, after a large content migration into SharePoint it would be a good idea to rebuild the indexes.

So in order to combat fragmentation in SharePoint databases, Microsoft released [KB article 943345](#). The stored procedure provided by this KB article provides a no-nonsense approach to database maintenance. You don't have to know a whole lot other than to just run the stored proc against your SharePoint databases regularly.

For more thorough maintenance plan direction including information regarding DBCC checking your SharePoint databases (an absolute requirement!) you will definitely want to read [this excellent whitepaper](#) written by Bill Baer from Microsoft. Bill includes step by step instructions for creating maintenance plans for SharePoint databases. Bill and [his blog](#) are a treasure trove of excellent information regarding SharePoint "IT Operations". **NOTE** however, that the stored procedure in Bill's whitepaper is a little outdated. You'll still want to use the stored proc in the KB article as it is more current. The newer stored procedure also includes provisions for index "fill factor" which is an important factor in preventing future fragmentation.

Finally, if you take the quick and easy path and use the standard maintenance plan wizard, there is an important article warning here (SQL Server 2005 only). If SQL Server 2005 SP2 is not installed, the maintenance plan may break the search database. See this KB article for more information: <http://support.microsoft.com/default.aspx/kb/930887/en-us>. Also SEE APPENDIX.

WSS Column Indexes vs Large Content Databases

In SharePoint, it is possible to create column indexes in document libraries. These indexes can dramatically improve the load speed of a library view if that view contains an indexed column in the filter criteria. Also, any programmatic searches that use the Site Data Query engine to retrieve results will also benefit from having an indexed column in the CAML "where" clause.

These indexes aren't true SQL Server table indexes, but rather pseudo-indexes that are table driven. Because of this, there is an impact to the content database as indexes are created and deleted. A large content database consisting of several hundred GB can result in a table locking situation during the manipulation of the column indexes.

While the index modification routines are building or destroying indexes, other SharePoint users are effectively locked out of any browse operations of any sites that use the same content database where the indexes are being modified. If the content database is large enough or if the I/O for the content database is poorly optimized, it is possible that an index add/delete operation may even time out in the browser. If this happens, allow time for the index operation to complete, possibly as long as 15 to 30 minutes after the timeout occurs. Once browsing of the site returns to normal the operation will likely be finished.

This issue is one of the few functional limitations of having a very large content database. The other limitations are related to the ability to backup and restore a site collection through the UI.

Implications of a Very Large Content Database

Microsoft suggests that the optimal maximum content database size should be 50GB. They also recommend that for most SharePoint implementations, the solution should not include any content databases larger than 100GB. This is commonly referred to as the “100GB content database size limitation”.

In fact, this is not a true limitation but rather a recommendation. SQL Server databases have been scaling far beyond 100GB for years now. Practically speaking, the recommendation is based primarily on three significant factors:

1. Service Level Agreement (SLA) requirements for a given organization may dictate that backup operations for the SharePoint databases must be executable in a limited amount of time. The size of the content databases will have a direct impact on how long it takes to execute that backup.
2. Large List contention. Microsoft recommends that lists/libraries should be limited to 5 million documents. With a good I/O subsystem and careful library view design, I've seen libraries that run comfortably up to 10 million. Beyond that we begin to see performance degradation due to table/index design and all of the joins that take place to return list item data.
3. The storage subsystem must be robust enough to handle the disk I/O requirements of the SharePoint solution that it serves.

As long as a given organization is able to mitigate these three considerations, then the content databases can be allowed to grow somewhat beyond the 100GB recommended limit. Real world implementations have seen successful SharePoint deployments that have implemented database sizes of 100GB, 150GB, 200GB, 250GB, 300GB, 350GB and even 400GB. It is important to note that some of the very large content databases mentioned here were used by very few people. Thus issues of I/O contention were mitigated by a relatively few number of database requests.

In order to mitigate the SLA requirements, a solution might employ a high speed disk-to-disk backup solution, database mirroring, or database log shipping to a disaster recovery site.

In order to mitigate large list contention make sure that if a site has a lot of content, that content should be broken up into libraries such that the 5 – 10 million document count per library won't be exceeded. Also, ensure that library views do not attempt to display more than 2,000 items.

The combination of a strong storage subsystem and the recommendations in this whitepaper will go a long way to mitigate the I/O requirements of very large content databases (over 100GB) in a given SharePoint solution.

Monitoring SharePoint I/O Performance

Microsoft provides a handy tool called Performance Monitor to allow the monitoring of statistics that will help tune an application. When monitoring the disk I/O performance for SharePoint, we have a simple indicator.

To get a quick indication of whether or not the storage subsystem is keeping up with demand, open up Performance Monitor and watch the Average Disk Queue Length counter.

- Monitor the “total” value first.
 - o If ADQ stays in the decimal range, then I/O performance is excellent
 - o If ADQ stays in the low single digits, the I/O performance is acceptable
 - o If ADQ regularly averages in the low double digits, then I/O performance is degraded and users may be occasionally affected.
 - o If ADQ regularly averages in the triple digits or higher, then I/O performance is definitely suffering and significant user impact is likely.
- If the “total” value exceeds recommended limits, then add counters for individual storage volumes to determine the offending volume. If the recommendations of this whitepaper were followed, it should be possible to determine offending database(s). Once that is determined, it may be possible to manipulate the storage subsystem or SQL Server to relieve the identified I/O stress (by adding additional LUNs of unused spindles).

Other helpful I/O counters are:

- Logical Disk: Disk Transfers/sec
- Logical Disk: Disk Read Bytes/sec & Disk Write Bytes/sec
- Logical Disk: Average Disk sec/Read (Read Latency)
- Logical Disk: Average Disk sec/Write (Write Latency)
- Logical Disk: Average Disk Byte/Read
- Logical Disk: Average Disk Byte/Write
- Logical Disk: Current Disk Queue Length
- Logical Disk: Average Disk Reads/Sec
- Logical Disk: Average Disk Write/Sec

Microsoft also provides a tool called SQLIO which can be used to determine the I/O capacity of a given SQL storage configuration. The SQLIO tool can be found here:

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

SQL Server Management for SharePoint Administrators

Move tempdb

After executing this script and restarting SQL Server, the tempdb data and log files will be initialized in the new location.

```
USE master  
GO
```

```
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev, FILENAME = '[New data folder  
path]tempdb.mdf')  
GO
```

```
ALTER DATABASE tempdb MODIFY FILE (NAME = templog, FILENAME = '[New log folder  
path]templog.ldf')  
GO
```

(Restart SQL Server)

Add tempdb data files and set all data file sizes (in KB)

After executing this script and restarting SQL Server, the existing tempdb data file will be set to the new size and the new tempdb data files will be initialized in the specified locations and set to the specified size.

```
USE [master]  
GO
```

```
ALTER DATABASE [tempdb] MODIFY FILE ( NAME = N'tempdev', SIZE = [Calculated]KB )  
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev_02', FILENAME = N'[Second data  
file path]tempdev_02.ndf' , SIZE = [Calculated]KB , FILEGROWTH = 10%)  
GO
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev_03', FILENAME = N'[Third data file  
path]tempdev_03.ndf' , SIZE = [Calculated]KB , FILEGROWTH = 10%)  
GO
```

```
.  
. .
```

```
ALTER DATABASE [tempdb] ADD FILE ( NAME = N'tempdev_0N', FILENAME = N'[Nth data file  
path]tempdev_0N.ndf' , SIZE = [Calculated]KB , FILEGROWTH = 10%)  
GO
```

(Restart SQL Server)

Pre-Create a Database (for Content or Search databases)

The number of data files, size of the data files, and location of the data files for the database should be determined using information in this whitepaper. Once that is determined, plug those values into the script below. Once this script is executed, a properly distributed blank database will exist. Simply choose the database name of the (blank) database that this script created when configuring SharePoint through the SharePoint Central Administration or when using STSADM.

```

CREATE DATABASE [ContentDBName] ON PRIMARY
( NAME = N'ContentDBName_01', FILENAME =
N'D:\MSSQL\DATA\ContentDBName_01.mdf' , SIZE = 2048KB , FILEGROWTH =
1024KB ),
( NAME = N'ContentDBName_02', FILENAME =
N'E:\MSSQL\DATA\ContentDBName_02.ndf' , SIZE = 2048KB , FILEGROWTH =
1024KB )
LOG ON
( NAME = N'ContentDBName_log', FILENAME = N'C:\Program Files\Microsoft
SQL Server\MSSQL.1\MSSQL\DATA\ContentDBName_log.ldf' , SIZE = 1024KB ,
FILEGROWTH = 10%)
COLLATE Latin1_General_CI_AS_KS_WS
GO
EXEC dbo.sp_dbcmptlevel @dbname=N'ContentDBName', @new_cmptlevel=90
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [ContentDBName].[dbo].[sp_fulltext_database] @action = 'disable'
end
GO
ALTER DATABASE [ContentDBName] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [ContentDBName] SET ANSI_NULLS OFF
GO
ALTER DATABASE [ContentDBName] SET ANSI_PADDING OFF
GO
ALTER DATABASE [ContentDBName] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [ContentDBName] SET ARITHABORT OFF
GO
ALTER DATABASE [ContentDBName] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [ContentDBName] SET AUTO_CREATE_STATISTICS ON
GO
ALTER DATABASE [ContentDBName] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [ContentDBName] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [ContentDBName] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [ContentDBName] SET CURSOR_DEFAULT GLOBAL
GO
ALTER DATABASE [ContentDBName] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [ContentDBName] SET NUMERIC_ROUNDABORT OFF
GO

```

```

ALTER DATABASE [ContentDBName] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [ContentDBName] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [ContentDBName] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [ContentDBName] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [ContentDBName] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [ContentDBName] SET READ_WRITE
GO
ALTER DATABASE [ContentDBName] SET RECOVERY FULL
GO
ALTER DATABASE [ContentDBName] SET MULTI_USER
GO
ALTER DATABASE [ContentDBName] SET PAGE_VERIFY CHECKSUM
GO
USE [ContentDBName]
GO
IF NOT EXISTS (SELECT name FROM sys.filegroups WHERE is_default=1 AND
name = N'PRIMARY') ALTER DATABASE [ContentDBName] MODIFY FILEGROUP
[PRIMARY] DEFAULT
GO

```

Move a database file from a source volume to a destination volume

Often times SharePoint solutions are implemented without properly planning for storage requirements resulting in “out of disk” errors. When this occurs, it may be necessary to create a new, larger volume and move the database to that new volume. The procedure for this is as follows:

1. Ensure a validated backup of the database has just been executed.
2. Execute a full DBCC check of the database in question. Structural errors will impact your ability to detach and re-attach the database successfully!
3. Detach the database from SQL Server. You may need to check the “drop existing connections” checkbox.
4. Move the database content files and/or the log file from the source volume to the destination volume.
5. Re-attach the database to SQL Server. If using the SQL Management Studio, select the database “mdf” file from its current/new location. If any of the related database files show a message status of “Not Found”, simply locate that file in its new location.
6. Verify that the database is accessible.
7. Alternatively, you can move the database backup from step 1 to the new server and then restore the database to that server. Depending on the database size, this may be a slow operation. But the benefit is that you’re also validating the restore at the same time! A DBCC CheckDB should be run after the restore to ensure the database is sound.

Redistribute content from a single large database data file to multiple data files

Often times SharePoint solutions are implemented without properly planning for storage requirements resulting in disk I/O issues. In order to move from the out of the box configuration to a more highly scalable configuration, it may be necessary to redistribute content from a single content database data file to multiple data files. The procedure for this is as follows:

1. Ensure a validated backup of the database has been recently executed.
2. Execute a full DBCC check of the database in question. Any structural defects in the database will manifest as errors during SHRINKFILE operations. You should ensure that you have “clean” database before beginning redistribution operations.
3. Add additional data files to the primary file group of the database using the recommendations in this whitepaper.
4. Execute this script:

```
USE [Content_DB_Name]
```

```
GO
```

```
DBCC SHRINKFILE (N'Content_DB_Primary_File_Logical_Name' , EMPTYFILE)
```

```
GO
```

This procedure will cause ALL of the content from the original (mdf) file to be equally distributed to the other (new ndf) files in the filegroup. This script may take SEVERAL HOURS to complete if the source database file is extremely large. Typically, I will “cancel” the script execution when the new “ndf” files have filled up to the point where they are roughly equal in size to the “mdf” file, although this is optional.

Summary

Disk I/O Impact on SharePoint Performance

SharePoint scalability and performance is an exercise in resource balancing. By following the recommendations of the SharePoint Capacity Planning tool provided by Microsoft, the topology necessary to support specified user requirements are clear. However, this tool explicitly states that the storage subsystem is not considered in the results.

So while the server infrastructure may be in place for a properly architected farm, it is easy for the storage subsystem to become overwhelmed by that those servers resulting in unexpected poor performance

A strong and expandable storage subsystems as well as regular maintenance are the keys to current and future performance of large scale Microsoft SharePoint implementations. The ability to spread content across as many disk spindles as possible is paramount.

The Bottom Line

The scalability of a software product is determined by that software's ability to consume additional hardware and, in turn, produce improved performance relative to the hardware increase.

The focal point of this whitepaper is that SharePoint is highly scalable, largely due to its ability to use additional storage capacity and servers and, in return, produce the ability to store more content and serve more users.

Using these techniques, a SharePoint farm has been scaled and tested to at least 50 million content items which is the recommended guideline that a farm Index Server should support.

Even further, with the implementation FAST technology to handle Index/Search services, it is likely that a single SharePoint farm that implements these techniques will scale far beyond 50 million items.

Just remember that when it comes to any SharePoint database storage solution, try not to spare expense. "E Aho Laula"...Wider is better.

Appendix A: References

Capacity Planning for tempdb

<http://msdn2.microsoft.com/en-us/library/ms345368.aspx>

Physical Storage Recommendations (Office SharePoint Server)

<http://technet.microsoft.com/en-us/library/cc298801.aspx>

Estimate Performance and Capacity Requirements for Search Environments

<http://technet.microsoft.com/en-us/library/cc262574.aspx>

Performance Recommendations for Storage Planning and Monitoring

<http://go.microsoft.com/fwlink/?LinkID=105623&clcid=0x409>

“Office SharePoint Server 2007 – Administrator’s Companion”

Bill English with the Microsoft SharePoint Community Experts, Microsoft Press

How to Defragment Windows SharePoint Services 3.0 and MOSS Databases

<http://support.microsoft.com/default.aspx/kb/943345/>

Database Maintenance for Office SharePoint Server 2007

<http://technet.microsoft.com/en-us/library/cc262731.aspx>