NetApp
Go further, faster

Technical Report

# Sizing Guidelines for SQL Server 2005/2008 for NetApp Solutions

John S. Parker, NetApp
June 2009 | TR-3779

## SQL SERVER SIZING GUIDELINES

This technical report describes the methodologies of sizing a Microsoft® SQL Server® 2005/2008 instance. It provides an overview of the Microsoft SQL Server architecture, guidelines to determine workload types, and explanations of gathering statistics with PerfMon and sqlstatspack.sql.

Microsoft GOLD CERTIFIED Partner

TABLE OF CONTENTS

# 1 INTRODUCTION

This technical report provides guidelines for the field to develop better sizing practices. This report aims to help anyone who is attempting to size NetApp® hardware for Microsoft SQL Server. It also covers procedures and methodologies as well as requirements for performance.

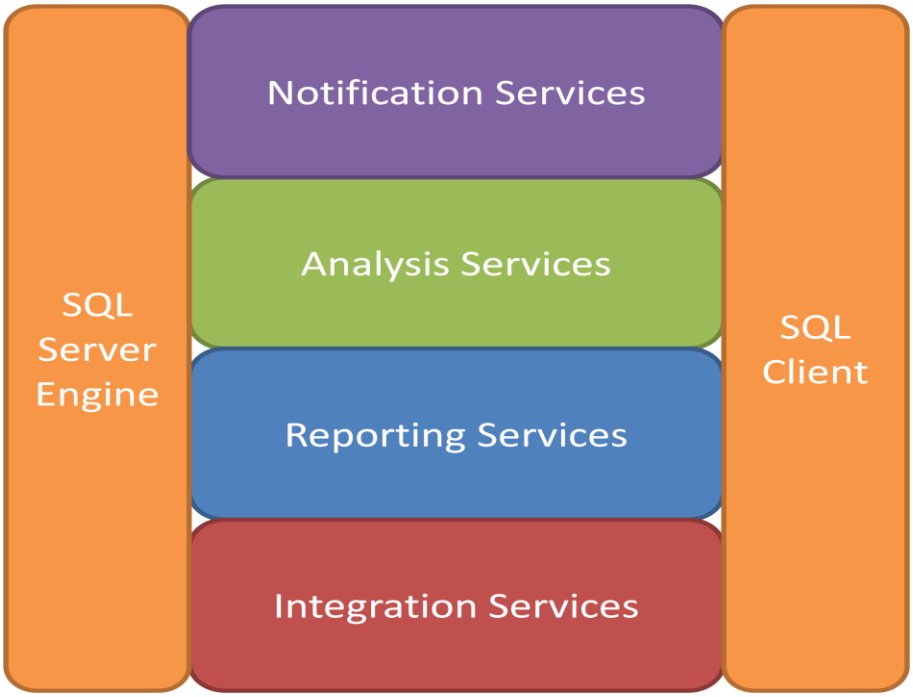
# 2 ARCHITECTURE AND COMPONENTS

SQL Server is a rich relational database engine that allows flexibility for both the small department and the large enterprise data warehouse. The flexibility of the architecture comes from the layering of components and services within the SQL Server product offering.

## 2.1 SQL SERVER COMPONENT MODEL

The SQL Server component model (Figure 1) helps you understand where the moving parts are concentrated when planning for SQL Server. The basic installation of SQL Server resides entirely on the local drive. The component locations can be moved around as needed.

**Figure 1) SQL Server model.**




The foundation of SQL Server is the database engine. The core engine comprises the initial 150MB needed from the operating system to start as well as the initial portion of memory. Chapter 2 (page 30–32) of the *SQL Server 2008 Administrator's Guide* describes in detail the need for high-speed drives and the performance characteristics of different drives. This guide does not recommend any particular type of drive, but describes factors you must consider when deploying SQL Server. This is important in helping customers understand that the storage designs for their SQL Server deployments are based on factors beyond the size of the SQL Server environment.

The ability to configure the location of the specific files allows for flexibility. When sizing the SQL Server components, the needs of the SQL Server engine, Analysis Services, Reporting Services, and in some cases the Integration Services must be addressed.

## 2.2 DATABASE SIZING CONSIDERATIONS

There are several considerations to keep in mind when sizing a SQL Server environment. Keep track of the database (system and user databases) and transaction log files, file stream data, database compression (in SQL Server 2008), and the SnapInfo directory if you are using SnapManager® for SQL Server. Following are the additional considerations when configuring your SQL Server environment. For details on how to configure and deploy your SQL Server environment, see TR-3696: Microsoft SQL Server 2005 Engine: Storage Fundamentals for NetApp Storage Systems.

- Data files (tables and indexes) are the primary files that are used by the SQL Server storage engine. Each database might have multiple files and be spread across multiple LUNs and volumes.
- Transaction log files operate in a sequential manner. The backup of the transaction log is stored as a copy into the SnapInfo directory.
- Database compression is a new feature in SQL Server 2008 that allows the database to compressed, giving the database administrator the ability to stretch the growth of the storage system without severely impacting the database. The compression ratio varies depending on the actual data, so testing is important.
- Backing up tempdb is not required, as it is rebuilt when a SQL instance is rebooted. The size of tempdb is affected by features such as online database maintenance, version store, and database query plans.
- Filestream data (SQL Server 2008): This data type allows the extension of the database for large objects such as bitmap images, text files, music files, and video and audio files. These can then be managed through insert, update, delete, and select statements, as well as standard SQL backup procedures.
- SnapInfo—calculating the SnapInfo directory if using SnapManager for SQL Server. This can be accomplished using the following formula:

  SnapInfo_LUN_Size = ((DB_Size * Daily_Chg_Rate * Days_per_TA-Log_Snapshots) + system databases (master, model, MSDB, and so on) + SnapInfo metadata)

When sizing your SQL Server environment, any additional data that can be gathered helps define the profile of the system that you are sizing. The database sizer tool has specific data requirements. Having a good understanding of how the system is expected to perform and/or has performed in the past will help make sure of the viability of a sizing solution.

## 2.3 SQL SERVER FUNDAMENTALS GUIDELINES

Sizing is just one aspect of planning a new SQL Server environment. To better understand how to plan your layout for best performance, see TR-3696: Microsoft SQL Server 2005 Engine: Storage Fundamentals for NetApp Storage Systems. TR-3696 provides guidance on the layout for the tempdb and the system databases and recommendations for the aggregate, volume, and LUN configuration that will help you meet the performance objectives of the system you are planning.

# 3 WORKLOAD SIZING GUIDELINES

## 3.1 SQL SERVER I/O OVERVIEW

SQL Server is very sensitive to I/O latency issues due to the concurrent transactional nature of the SQL Server engine. SQL Server is built on a complicated system of row, page, extent, and table locks that provide transactional consistency throughout the SQL Server system. A poor I/O structure (for example, when I/O takes too long) causes resources to be held longer than necessary, resulting in blocking within the system. When these occur, it frequently is not obvious that the I/O subsystem is the root cause.

SQL Server customers new to NetApp might not understand the differences between RAID-DP® and RAID 10. SQL Server performance has always been about I/O. This performance can be improved by either increasing spindles or making the spindles go faster. This is where the traditional SQL environments have been slow to adopt RAID 5 or RAID 6. NetApp RAID-DP, a RAID 6 implementation, is a standard feature of Data ONTAP® and prevents data loss in the event of a second drive failure without excessive redundancy costs.

- **SQL Server reads**: When reading data from SQL Server, the client will first go to the buffer cache. If the data is not in the buffer cache, SQL Server will go to the I/O subsystem to retrieve the data. The statement will not complete until 100% of the data is read; hence, the user connection or process will be in an I/O wait state until completion.
- **SQL Server writes**: The user writes to the transaction log and the buffer cache. If the data to be modified is not already in the buffer cache, then it must be read into the buffer cache from the I/O subsystem. The buffer manager makes sure that the transaction log is written to first, before changes are written to the database. This is known as write-ahead logging. When the user makes the change and the `COMMIT` is executed, a log write occurs showing that the change took place, allowing the `COMMIT` to complete. Once the `COMMIT` is complete, the user process can continue on to the next stage or command without having to wait for the changes to be written to the disk. `ROLLBACK Transaction` follows the same process as the `COMMIT`, but in reverse. The buffer manager moves the data from the cache to disk. It keeps track of log sequence numbers (LSNs) for each log record.
- **Transaction log**: The SQL Server transaction log is a write-intensive operation that is sequential in nature. The transaction log is used to provide recoverability of data in case of database or instance failure.

## 3.2 ONLINE TRANSACTION PROCESSING

The Online Transaction Processing (OLTP) database system is the SQL Server environment most concerned about getting the most number of transactions through the system in the least amount of time. Examples of different types of OLTP systems include Web order systems and manufacturing tracking systems. OLTP systems can have very large volumes of transactions per second, and for the OLTP system it is all about throughput. For these transactions to take place, SQL Server relies on an efficient I/O subsystem. According to the Microsoft SQL Server best practices article written by Mike Ruthruff, an OLTP transaction profile is composed of the following pattern:

- OLTP processing is generally random in nature for both reads and writes issued against data files.
- Read activity (in most cases) is consistent and point queries; it does not consist of large time-consuming queries.
- Write activity to the data files occurs during checkpoint operations (frequency is determined by recovery interval settings).
- Log writes are sequential in nature with a varying size, which is dependent on the nature of the workload (sector aligned up to 60KB).
- Log reads are sequential in nature (sector aligned up to 120KB).

## 3.3 DECISION SUPPORT SYSTEM

The SQL Server best practices article describes the nature of the decision support system (DSS) as follows:

- Reads and writes tend to be sequential in nature and are generally the result of table or index scans and bulk insert operations.
- I/O size varies but is generally larger than 8KB. Read-ahead is any multiple of 8KB up to 256KB (1024KB for Enterprise edition). Bulk load operations are any multiple of 8KB up to 128KB.

## 3.4 MIXED

In mixed environments you will need to take the blended approach for managing the I/O for SQL Server. The reads or writes work out to an average, which is closer to a ratio of 40% through 60% to 65% through 35%. Arranging the reads and the writes according to the proper mix of your environment will define the performance profile for the database system. Even though mixed environments vary, a 50-50 mix for reads and writes is a good place to start. Here, you are looking for the balance between reporting and online transactions.

**Table 1) Read/write percentages.**

| Database System Type | Read Percentage | Write Percentage |
|---|---|---|
| DSS | 80% | 20% |
| OLTP | 66% | 33% |
| Mixed | 50% | 50% |

# 4 APPROXIMATING DATA SIZES

## 4.1 ESTIMATING I/Os

Estimating the number of I/Os required for a system is crucial when sizing a database. This exercise helps the administrator understand how to keep the database instance performing within acceptable limits. You will have to estimate I/Os when you are not able to get the actual physical I/O numbers for the system. This is typically the case in new systems that are in the process of being constructed. Following are the formulas for estimating I/Os, as outlined in the *SQL Server 2005 Administrator's Guide*.

To estimate I/Os for a new database system without a system, do as follows:

1. Estimate the number of transactions for a given period size.
2. Multiply the number of transactions by the 0.85 saturation rate and then divide that by the number of seconds in a day.

   The seconds in a day are determined by the hours of operation for the database. If the database operates in a 24-hour environment, the number is 86,400. The formula for estimating the number of I/Os is:

   Total I/Os = (estimated number of transactions * 0.85)/seconds in a day

   For example, if there are 40,000 transactions on a system that is in operation 24 hours per day, the formula is as follows:

   (40,000 * 0.85)/86,400 = 0.3935 IOPS
3. After determining the I/Os required, determine what kind of system you are going to be deploying. If deploying an OLTP system, determine the read and write I/Os by multiplying the number of I/Os by the percentage of reads or writes. Table 1 lists the I/O percentages for each type of system.

   The formula I/Os in megabytes ((number of transactions * 0.85)/seconds in a day) * type % = I/O megabytes.

   For example, to determine the reads and writes for a DSS system, the formula is as follows:

   ((40,000 * 0.85)/86,400)* 0.80) = 0.3148MB reads

   ((40,000 * 0.85)/86,400)* 0.20) = 0.0787MB writes

## 4.2 GATHERING STATS

When sizing for an existing database environment, understanding the type of workload and interpreting the statistical data are helpful. The database sizer tool does not determine the type of system based on the data uploaded. By comparing the physical reads and writes to the numbers from Table 1, you can estimate the type of system you are working with. It is important to gather statistics during periods of peak stress on the system. The `sqlstatspack.sql` script will gather an average over the entire time frame that you are monitoring.

PerfMon allows you to see the high-water marks for the time frame that you monitor the system. For more information about using PerfMon, see Appendix C. Enter the highest points of the reads and writes values in the database sizer tool.

# 5 BUILDING GROWTH AND CAPACITY MODELS

## 5.1 CAPACITY PLANNING

According to the *Microsoft SQL Server 2005 Administrator's Companion*, the fundamentals of sizing and capacity planning are based on queuing theory. Queuing theory has a few terms that you need to understand: service time and wait time, which is also known as queue time. The key formula is:

Response time = service time + queue time

Queuing theory has two key components according the *Microsoft SQL Server 2005 Administrator's Companion*:

- The chance of queuing increases exponentially as you near capacity of the resource.
- The response time is equal to the sum of the service time plus the queue time. (See Chapter 6 [pages 108–111] in the *Microsoft SQL Server 2005 Administrator's Companion*.)

Monitoring the I/O subsystem with PerfMon is an excellent way to develop a behavior profile for the I/O subsystem of the SQL Server. Physical disk counters are best for understanding your I/O.

- **Disk reads/sec**: Read IOPS for the disk drive or drives
- **Disk transfers/sec**: Total read plus write IOPS for each disk drive
- **Disk writes/sec**: Write IOPS for each disk drive
- **Avg. disk sec/read**: Read latency or average time for each read operation
- **Avg. disk sec/write**: Write latency or average write time for each operation
- **Average disk queue length:** Average of both read and write request that are in the queue

For a well-tuned I/O subsystem, the read and write latency should be within 5 to 10 ms. If the latency is 20 ms, the system might be experiencing an I/O issue. Latency exceeding 30 ms is in the unacceptable range. As all values are relative, the impact of measured latency depends on the type of operations taking place on your storage system and what they mean to your customers.

Questions to ask the customer about capacity planning:

- What is the size of the database or databases?
- How many users will be accessing the databases?
- When are the peak times for database activity?
- What types of activity will be taking place on each database? OLTP, DSS, batch jobs, misc.?
- What is the rate of growth of the data?
- What is the rate of growth users?
- Is this a dedicated SQL solution, or is it a multipurpose solution?
- What are the needs for high availability, scalability, and disaster recovery?
- Will the databases be replicated as part of the new strategy?

## 5.2 GROWTH MODELS

Growth models allow administrators to develop projections for the growth that databases will potentially see. The only growth model that is 100% accurate is the historical model, but this model only tells you where you have been and not where you are going. Hence, administrators need to take that historical data and plug it into the model that will best fit their requirements.

The limitation with any growth model is the amount of data that you have to plug into the model. The more data about the system that you can gather, the better your projection will be. This technical report will look at two different models: the linear growth and model and the geometric growth model (Hotek and Makin).

### LINEAR GROWTH MODEL

The linear growth model builds a straight line model for the future. It is based on the amount of growth in the system. As with all models, the need to determine the amount of growth that the system has historically seen over a given period of time is still an important aspect of system management.

The formula for the linear growth model is as follows:

Future usage = current usage + (growth amount * number of periods)

Using the formula determine the total size of the system as it currently stands. One procedure for gathering this information is by using `sp_helpdb`. `sp_helpdb`. This will give you the totals for each database. Next, take the sum of disk space used by all databases. The growth amount is found by tracking the size over a period of time. Take this growth amount and apply it for the number of months looking into the future.

For example, for a system that is currently 400GB in size with a growth amount of 50GB per month over a three-month period (Table 2), the formula is as follows:

Estimated future usage = 400GB + (50GB X 3) = 550GB in 3 months' time

**Table 2) Example 1.**

| Parameter | Metric |
|---|---|
| Current usage | 400GB |
| Growth amount | 50GB/month |
| Number of periods | 3 months |

The conclusions drawn from this exercise provide an idea of how much the capacity of the database system is expected to grow in the three-month time frame. This allowance for future growth gives an educated estimate of the sizing needs for a particular database.

### GEOMETRIC GROWTH MODEL

Exploring the geometric growth model allows for finer tuning of changes in the system. One way to look at the growth rate is as a percentage of growth over a period of time. The sampling of that time frame for total database size then compares the differences from the last sampling period. If there is no historical sizing data, then you can look at the difference between the `backup size` for each database over the history of backups in the msdb database. As long as the databases in the system have been backed up regularly, you will be able to estimate the growth rate over a given sample period.

To estimate the growth, insert the current usage and growth rate in to the following formula along with the number of months for the projection and you can get a geometric view of the rate of growth.

Future usage = current usage * (1 + growth rate/current usage) $^{\text{number of periods}}$

For example, applying the values from Table 2, the formula is as follows:

Estimated future usage = 400 * $(1 + 50/400)^3$ = 569.53GB in 3 months' time

# 6 PHYSICAL LAYOUT FOR SQL SERVER DATABASES AND STORAGE (MAP TO THE STORAGE FUNDAMENTALS)

The planning process of a new system requires examining the layout of the SQL Server instance and databases. This exercise is helpful for the database and storage administrators to plan what their new system will look like in the future. For guidance on how to lay out the aggregates, volumes, and LUNs, see the technical report TR-3696: Microsoft SQL Server 2005 Relational Engine: Storage Fundamentals for NetApp Storage Systems. This technical report describes how to configure the SQL Server instance for best performance.

# 7 SIZING EXAMPLE

Now that we have taken all these things into account, here is an example of sizing a SQL Server database for NetApp storage. The first step is to gather data that will be applicable to the database sizer tool.

Start by gathering information such as database reads and writes and database sizes. This example is a new system, requiring the use of sizing estimations.

1. Determine the number of database transactions. This database will be in operation 24 hours a day and is estimated to have 40,000 transactions in a 24-hour period. These transactions include all inserts, updates, and deletes that will happen in the database.
2. Determine reads and writes. Enter the estimated 40,000 transactions into the following formula:
   Total I/Os = (estimated number of transactions * .85)/seconds in a day

   That is:

   (40,000 * 0.85)/86,400 = 0.3935

   This gives us the total I/Os in megabytes for a 24-hour period.

   Apply this result this to the following formula based upon the type of database system DSS, OLTP, or mixed:

   ((Number of transactions * 0.85)/seconds in a day) * type % = I/O megabytes

   For example, for a DSS system the calculations are as follows:

   ((40,000 * .85)/86,400)* 0.80) = 0.3148MB reads

   ((40,000 * .85)/86,400)* 0.20) = 0.0787MB writes
3. Determine the size of the databases and transaction logs. Now that the I/Os for the system have been determined, as shown in step 2, the sizes of the database and log are required. The database size used here is 500GB, and the transaction log is approximately 15% of the database size, which is 75GB.
4. Determine the number of transaction logs to be kept online. With our transaction rate we estimate to keep seven transaction logs online.
5. Determine the number of Snapshot™ copies to retain. In this example, the backup retention policy will go back 14 days for full database backups. So we will keep 14 Snapshot copies available for restore at any time.
6. Determine the growth of the database for the year. As this is a new system, it is required to interpolate some data to plug into one of the growth models. To calculate the annual change rate of data, the formula for writes per megabyte per second is modified to determine the percentage daily change rate, which is then multiplied by 365 to get the percentage annual change of data.
   The formula modified to calculate the percentage daily change rate is:

   ((Estimated transactions * 0.85) * type %)/database size in megabytes) = daily change rate (%)

   (Daily change rate * 365) = annual growth rate (%)

   If the annual growth rate is less than 10%, then use 10%.

   For example:

   Daily write change rate: ((40,000 * 0.85) * 0.20)/500,000 = 0.0136%

   Annual writes growth rate: (0.0136 * 365) = 4.964%. Use 10%, as calculated rate is less than 10%.

   Daily read growth rate: ((40,000 * 0.85) * 0.80)/500,000 = 0.0544%

   Annual read growth rate: (0.0544 * 365) = 19.856%. Use 19.856%, as calculated rate is greater than 10%.
7. It is recommended to keep the data and logs in separate volumes and in some instance separate aggregates. This depends of how busy your database is; try to keep your aggregates as large as possible to get the advantage of making the most disks available to the database.
8. Once you have these key pieces of information, you then need to set a few general items such as storage system platform, maximum number of heads, drive type, and so on.

| General Storage System Specifications | |
|---|---|
| Data ONTAP Version | 7.3x |
| Storage System Platforms | Any |
| Max # of Heads | 10 |
| CPU Headroom % | 30 |
| Drive Types | Any |
| RAID Type | RAID-DP |
| RG Size | 16 |
| # Spares | 1 |
| Full Shelf? | N |
| Disk Headroom % | 10 |
| Local Cluster | No |
| CFO Performance | Degraded |
| MetroCluster | No |
| # of Accelerator Modules/Head | 0 |
| CFO Performance | Degraded |

**Note:** The above table lists just a sample of values that need to be entered into the database sizer tool. This can be customized to fit the needs of the database solution.

9. Additional points to note are the use of SnapMirror®, SnapDrive®, and SnapManager for SQL Server.
10. Check the database sizing input form to make sure that you have gathered all required data. When you have gathered the required information, enter it into the database sizer tool.

# APPENDIXES

## APPENDIX A DATABASE SIZING INPUT FORM

This spreadsheet helps with the calculations for I/O and helps organize your data before entering them into the database sizing tool. https://fieldportal.netapp.com/ci/files/18388.xlsx

**Note**: It is not required to use all of the spreadsheet.

## APPENDIX B STATSPACKSTATSPACK4SQL3.SQL

http://perf-build1.eng.netapp.com/DBSizer/SQLMon.jsp

```
/*
##########################################################################################
######################
## Original Create Date: 6/15/05 Original Author: Jonas Irwin
## Current Version: 2.0
## Current Author: Mike Flannery
##
## Purpose:
## Grab IO statistics for SQL Server databases and data files over operator
specified time interval
## Mods:
## 12/20/07 - Flannery - Added UNION at end to make the output more end user
readable
## 12/20/07 - Flannery - Tested with SQL Server 2005. Made tables and columns
case consistent
## 3/14/08 - Flannery - Tested with SQL Server 2008
## 3/14/08 - Flannery - Changed SQLIO table from permanent to variable table
## 8/01/08 - Safa – Adjust case in Column calls
## 8/01/08 - Safa – Validate on SQL 2000 and 2005
##########################################################################################
######################

*/

DECLARE @total int
DECLARE @now datetime
select @now = getdate()
select @total = 0

DECLARE @SQLIO TABLE
 (dbname varchar(128) ,
 fname varchar(2048),
 startTime datetime,
 noReads1 int,
 noWrites1 int,
 BytesRead1 bigint ,
 BytesWritten1 bigint ,
 noReads2 int,
 noWrites2 int,
 BytesRead2 bigint ,
 BytesWritten2 bigint,
 endtime datetime,
 deltawrites bigint,
 deltareads bigint,
 timedelta  bigint,
 fileType  bit,
 fileSizeBytes bigint
```

```
 )

USE master

--grab baseline first sample
INSERT INTO @SQLIO
SELECT
 cast(DB_Name(a.DbId) as varchar),
 b.filename,
 getdate(),
 cast(NumberReads as int),
 cast(NumberWrites as int),
 cast(a.BytesRead as bigint),
 cast(a.BytesWritten as bigint),
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 'filetype' = case when groupid > 0 then 1 else 0 end,
 cast(b.size as bigint) * 8192
FROM
 ::fn_virtualfilestats(-1,-1) a ,sysaltfiles b
WHERE
 a.DbId = b.dbid and
 a.FileId = b.fileid
ORDER BY 1

/*DELAY AREA - change time at will */
WAITFOR DELAY '000:00:10'


UPDATE @SQLIO
set
 BytesRead2=cast(a.BytesRead as bigint),
 BytesWritten2=cast(a.BytesWritten as bigint),
 noReads2 =NumberReads ,
 noWrites2 =NumberWrites,
 endtime= getdate(),
 deltawrites = (cast(a.BytesWritten as bigint)-BytesWritten1),
 deltareads= (cast(a.BytesRead as bigint)-BytesRead1),
    timedelta = (cast(datediff(s,startTime,getdate()) as bigint))

FROM ::fn_virtualfilestats(-1,-1) a ,sysaltfiles b
WHERE  fname= b.filename and dbname=DB_Name(a.DbId) and
 a.DbId = b.dbid and
 a.FileId = b.fileid


/*dump results to screen - individual results
SELECT
 'Transaction Log Size',
 sum(cast(b.size as float) * 8192)/1024/1024
FROM
 ::fn_virtualfilestats(-1,-1) a ,sysaltfiles b
WHERE
 a.DbId = b.dbid and
 a.FileId = b.fileid and
 groupid = 0
union
```

```
SELECT
 'Database Size',
 sum(cast(b.size as float) * 8192)/1024/1024/1024
FROM
 ::fn_virtualfilestats(-1,-1) a ,sysaltfiles b
WHERE
 a.DbId = b.dbid and
 a.FileId = b.fileid and
 groupid > 0
union
select
 'Read IO Percent',
 (sum(cast(deltareads as float))/(sum(cast(deltawrites as
float))+sum(cast(deltareads as float))))*100
from @SQLIO
union
select
 'Database Read Rate',
 sum(cast(deltareads as float))/max(cast(timedelta as float))/1024/1024
from @SQLIO
union
select
 'Database Write Rate',
 sum(cast(deltawrites as float))/max(cast(timedelta as float))/1024/1024
from @SQLIO
union
select
 'Log Rate',
 (sum(cast(deltawrites as float))/max(cast(timedelta as float))/1024/1024)*3
from @SQLIO
where fileType=0
*/
/*dump results to screen - sizer appropriate results */
select * from @SQLIO
```

## APPENDIX C PERFMON

PerfMon is a tool that is part of a Windows® server system. This tool helps you easily monitor SQL Server to gather I/O reads and write activity required for the database sizer tool.

Instructions for running PerfMon:

1. Start the PerfMon tool. The PerfMon tool can be used to monitor the SQL Server instance from the local instance or from a remote system. The remote system does not require SQL Server to be installed on it. However, both operating systems should have the same operating system service packs installed.
2. Select the following PerfMon settings:
   - LogicalDisk
     - Avg. disk queue length
     - Current disk queue length
     - Disk reads/sec: Enter this number in the database sizer tool.
     - Disk writes/sec: Enter this number in the database sizer tool.
   - PhysicalDisk
     - Disk reads/sec: Read IOPS for the disk drive or drives
     - Disk transfers/sec: Total read plus write IOPS for each disk drive
     - Disk writes/sec: Write IOPS for each disk drive
     - Avg. disk sec/read: Read latency or average time for each read operation
     - Avg. disk sec/write: Write latency or average write time for each operation
     - Average disk queue length
   - Processor

Percentage processor time (for each processor)

- SQLServer: Transactions

  Transactions

- SQLServer: Wait statistics

  Network I/O waits

  Page I/O latch waits (average)

  Page I/O latch waits (current)

  **Note**: The only items that are really needed for sizing are the LogicalDisk: Disk reads/sec and the LogicalDisk: Disk writes/sec. The rest of the information will help to give an idea of overall performance and health of the system.

3. After setting the desired PerfMon counters, start monitoring.
   **Note**: When monitoring, attempt to capture data during the peak so that you can capture the desired performance for the future system.

## APPENDIX D ADDITIONAL REFERENCES


Garvey, Brad, Space Reservation for Microsoft Exchange Server Using a NetApp SAN or IP SAN: TR-3758i, March 2009, http://my.netapp.com/psweb/jsp/DownloadContent.jsp?dDocName=P_053282.

Hotek, Mike, and Makin, J.C., MCITP Self-Paced Training Kit (Exam 70-443): Designing a Database Server Infrastructure Using Microsoft SQL Server 2005, Microsoft Press, 2007.

Isakov, Garcia, Misner, Patel, and Whalen, Microsoft SQL Server 2005 Administrator's Companion, Microsoft Press, 2006.

McPhail, Robert, Microsoft SQL Server 2005 Relational Engine: Storage Fundamentals for NetApp Storage Systems, TR-3696, July 2008, http://media.netapp.com/documents/tr-3696.pdf.

NetApp database sizing tool, http://perf-build1.eng.netapp.com/DBSizer/UnifiedDatabase.jsp.

Ruthruff, Mike, SQL Server Best Practices: Predeployment I/O Best Practices, June 2008, http://technet.microsoft.com/en-us/library/cc966412.aspx.

Considerations for Transactional Replication, http://msdn.microsoft.com/en-us/library/ms151254.aspx.