# logo-sql08.gif

SQL Server Technical Article

**Writers:** Emily Wilson, Mike Ruthruff, Thomas Kejser

**Contributor:** Juergen Thomas

**Technical Reviewers:** Sanjay Mishra, Prem Mehra, Lindsey Allen, Kevin Farlee, Ewan Fairweather, Ryan Stonecipher, Alexei Khalyako, Mike Weiner, Michael Thomassy, Wanda He, Kun Cheng, Jimmy May

**Published:** April 2010

**Applies to:** SQL Server 2008, SQL Server 2008 R2, SQL Server, 2005

Contents

[Introduction 4](#_Toc260219018)

[Characterizing Your Application I/O 4](#_Toc260219019)

[Common Types of Applications and Their General I/O Patterns 4](#_Toc260219020)

[OLTP 4](#_Toc260219021)

[Data Warehouse 5](#_Toc260219022)

[Log Workload 5](#_Toc260219023)

[tempdb Workloads 5](#_Toc260219024)

[Important Data Workload Metrics 5](#_Toc260219025)

[Reads and Writes 6](#_Toc260219026)

[I/O Size 6](#_Toc260219027)

[Collecting Data for Your Application 6](#_Toc260219028)

[Data to Collect 7](#_Toc260219029)

[Items to Note 9](#_Toc260219030)

[Setting Up a Test System at a Smaller Scale 9](#_Toc260219031)

[Examining the Data 9](#_Toc260219032)

[Reading the Performance Log Data 9](#_Toc260219033)

[Checkpoints 10](#_Toc260219034)

[Latencies 10](#_Toc260219035)

[Sizing Considerations for the Physical Storage Components 11](#_Toc260219036)

[Typical Sizing of Physical Disks 11](#_Toc260219037)

[Impact of RAID on Physical I/O 12](#_Toc260219038)

[What About the Cache? 12](#_Toc260219039)

[Read Cache 12](#_Toc260219040)

[Write Cache 13](#_Toc260219041)

[The Nondisk Components: Connectivity, Array Service Processors, and Internal Array Bandwidth 15](#_Toc260219042)

[Connectivity 15](#_Toc260219043)

[Array Service Processors 16](#_Toc260219044)

[Internal Array Bandwidth 17](#_Toc260219045)

[Examples of I/O Characteristics for Common Workloads 17](#_Toc260219046)

[Point-of-Sale System 17](#_Toc260219047)

[Decision Support System Query Workload 19](#_Toc260219048)

[Example: Translating the Data to Estimated System Requirements 21](#_Toc260219049)

[Summary 25](#_Toc260219050)

[Appendix A: An In-Depth Example of Calculating Physical I/O 26](#_Toc260219051)

[Appendix B: Other Sizing Considerations 28](#_Toc260219052)

[Thin Provisioning 28](#_Toc260219053)

[Snapshots and Clones 28](#_Toc260219054)

[Storage-Based Replication 29](#_Toc260219055)

[References 30](#_Toc260219056)

# Introduction

Understanding how to analyze the characteristics of I/O patterns in the Microsoft® SQL Server® data management software and how they relate to a physical storage configuration is useful in determining deployment requirements for any given workload. A well-performing I/O subsystem is a critical component of any SQL Server application. I/O subsystems should be sized in the same manner as other hardware components such as memory and CPU. As workloads increase it is common to increase the number of CPUs and increase the amount of memory. Increasing disk resources is often necessary to achieve the right performance, even if there is already enough capacity to hold the data.

Sizing storage systems for SQL Server can be challenging because I/O characteristics differ significantly between applications depending on the nature of the access patterns. The techniques in this paper will give you the tools you need to monitor and characterize I/O behavior of your SQL Server application as well as understand how this maps into a physical storage configuration. These techniques will provide a more in-depth understanding of common I/O patterns for SQL Server applications. This may be especially useful for ISVs or others who can accurately define their workload to characterize I/O within lab environments and use the information to provide more concrete deployment guidance to their customers.

This paper is meant not to serve as prescriptive capacity planning guidance for SQL Server but rather to provide an initial look at the tools and methodology for characterizing existing workloads and mapping those into physical storage requirements. This will be done using concrete examples to illustrate the application of the techniques.

# Characterizing Your Application I/O

The first step in being able to determine the requirements for your storage system is to understand the I/O pattern of your application. The frequency and size of reads and writes sent by the application are received and processed by the storage system. So, you need to understand their behavior and frequency in order to properly understand the requirements of that system.

# Common Types of Applications and Their General I/O Patterns

Different types of database applications have varying needs in retrieving and writing data. Understanding the archetypes for the most common database application workloads can be useful to understanding the likely behavior of your application.

## OLTP

Online transaction processing (OLTP) workloads tend to select a small number of rows at a time. These transfers come all over the data, and are each fairly small in size – typically between 8K and 64K. This causes the I/O pattern to be random in nature. The key metric in measuring performance of OLTP workloads is the number of I/Os per second (IOPs) that can be achieved while maintaining a healthy response time. When sizing a storage subsystem for this type of application, size based on the number of disks needed to support the required IOPs. Typical SQL Server I/O activities for OLTP workloads include queries with many seek operations, checkpoint activity that flushes dirty pages to disk periodically, and transaction log writes. Transaction log writes are particularly sensitive to I/O response times because log writes are done synchronously, which results in a direct impact on throughput of workloads performing writes to the database. This is further explained later in this document.

Point-of-sale systems are an example of a common OLTP application. Many applications have a mixed workload with both transaction processing and some level of real-time reporting. SAP and other enterprise resource planning (ERP) applications are examples of applications that represent a mixed workload.

## Data Warehouse

Data warehouse applications tend to issue scan-intensive operations that access large portions of the data at a time and also commonly perform bulk loading operations. These operations result in larger I/O sizes than OLTP workloads do, and they require a storage subsystem that can provide the required throughput. This makes the throughput or megabytes per second (MB/s) the critical metric, and ensuring there is enough bandwidth in the connectivity between server and storage is critical. Common data warehouse type applications include decision support applications.

## Log Workload

In addition to the reads and writes issued against user databases, I/O is issued against the transaction log. This I/O is highly sequential, consisting predominately of writes, and it requires a low latency for high scale transaction workloads. This makes for a very different pattern than the I/O being issued against the data files for either an OLTP or a data warehouse user database. Deployments that involve synchronous storage-based replication may introduce additional latency, impacting overall transactional throughput.

In addition to writes, there are other operations that also perform reads against the transaction log. For example, log backup operations, replication scenarios in which agent jobs read from the transaction log, database recovery, and some specific situations of database mirroring increase log workload.

## tempdb Workloads

tempdb is a system database used by SQL Server as a temporary workspace. Access patterns for tempdb may vary but are generally more like OLTP patterns. An in-depth discussion about the specific characteristics and sizing considerations for tempdb is beyond the scope of this document; however, there are several resources in the appendixes that can provide insight.

# Important Data Workload Metrics

The most important metrics to consider in your analysis are reads, writes, and I/O size.

## Reads and Writes

It is important to measure both the read and write workloads for your application. Reads and writes may have different characteristics depending on your application. They may also be different sizes, and of different proportions, depending on the application and where in the spectrum of workload types it falls.

For more information about how to measure reads and writes on an existing system or test load, see the “Examining the Data” section of this paper.

## I/O Size

In addition to the number of reads and writes being processed, the size of those reads and writes is important, and it has significant impact on the total throughput of the system. Larger transfer sizes are more efficient for operations that are reading large amounts of data. Disks are split into linear addressable regions; each unit in this space is 512 bytes, and such a unit is called a sector (note that contemporary high-end disks may allow you to define larger sectors). Because sectors close to each other in the linear space are also physically adjacent, it is faster to read two neighboring sectors than two sectors that are far apart.

Instead of requiring you to read individual sectors, a disk controller provides an abstraction that enables you to issue requests larger than one sector – typically all the way to 1-4 MB at a time for reads and 256 KB for writes. If you issue an I/O request larger than one sector, the disk services this request from adjacent sectors in the disk, which is faster than requesting the sectors individually. This means that larger I/O requests generally result in a need for higher throughput (MB/s). The size of the I/O transfers can also impact latency of the transfer, because larger I/O transfer sizes take longer to complete.

For more information about finding the average read size and write size, see the “Examining the Data” section of this paper.

# Collecting Data for Your Application

The best way to understand the access patterns for your application is through measurement. This can be done using the Performance Monitor tool in the Windows® operating system, commonly referred to as perfmon. It is also beneficial to collect this data from deployed applications at regular intervals to analyze trends and to have baseline measurements, which can be useful in investigating problems that are related to I/O performance.

When you collect data for sizing purposes, make sure that your data represents all expected deployment patterns, especially those for peak I/O loads of the production system. If the application is being characterized through testing in a lab environment, ensure that the scenarios being run represent the expected deployment scale and expected workload patterns. Data collected only during a maintenance window or off-peak hours will not be representative of your application’s regular workload. However, it is possible because that off-peak workloads will have very different I/O characteristics, data from these periods will provide valuable information about sizing. Hence, you need to collect both off-peak batch runs and user workloads. For example, an OLTP application may have one set of I/O characteristics during peak operations and another during maintenance activities and batch processing in off-peak hours.

## Data to Collect

There are numerous counters available in perfmon. We recommend that you collect data from the following counters when you are analyzing characteristics of a SQL Server workload. These disk-specific counters are listed under the LogicalDisk section of the available counters. (The PhysicalDisk counter object may show the same values if there is a 1:1 relationship between a logical volume and disk.) The LogicalDisk object reports volume letters or mount point names (rather than disk numbers) and it can provide a finer level of granularity if there are multiple volumes on a single Windows disk.

|  |  |
| --- | --- |
| **Counter** | **Description** |
| **LogicalDisk Perfmon Object** | |
| **Disk Reads/sec**  **Disk Writes/sec** | Measures the number of IOPs.  You should discuss the expected IOPs per disk for different type and rotational speeds with your storage hardware vendor.  Typical sizing at the per disk level are listed here:   * 10K RPM disk – 100 to 120 IOPs * 15K RPM disk – 150 to 180 IOPs * Enterprise-class solid state devices (SSDs) 5,000+ IOPs   Sizing is discussed at length later in this paper. |
| **Average Disk sec/ Average Disk sec/Write** | Measures disk latency. Numbers vary, but here are the optimal values for averages over time:   * **1 - 5 milliseconds (ms) for Log (ideally 1 ms or less on average)**   **Note:** For modern storage arrays, log writes should ideally be less than or equal to 1-2 ms on average if writes are occurring to a cache that guarantees data integrity (that is, battery backed up and mirrored). Storage-based replication and disabled write caching are two common reasons for log latencies in the range of 5 or more milliseconds.   * **5 - 20 ms for Database Files (OLTP) (Ideally 10 ms or less on average)** * **Less than or equal to 25-30 ms for Data (decision support or data warehouse)**   **Note:** The value for decision support or data warehouse workloads is affected by the size of the I/O being issued. Larger I/O sizes naturally incur more latency. When interpreting this counter, consider whether the aggregate throughput potential of the configuration is being realized. SQL Server scan activity (read-ahead operations) issues transfer sizes up to 512K, and it may push a large amount of outstanding requests to the storage subsystem. If the realized throughput is reasonable for the particular configuration, higher latencies may be acceptable for heavy workload periods.  If SSD is used, the latency of the transfers should be much lower than what is noted here. It is not uncommon for latencies to be less than 5 ms for any data access. This is especially true of read operations. |
| **Average Disk Bytes/Read**  **Average Disk Bytes/Write** | Measures the size of I/Os being issued. Larger I/Os tend to have higher latency (for example, BACKUP/RESTORE operations issue 1 MB transfers by default). |
| **Current Disk Queue Length** | Displays the number of outstanding I/Os waiting to be read or written from the disk.  Deep queue depths can indicate a problem if the latencies are also high. However, if the queue is deep, but latencies are low (that is, if the queue is emptied and then refilled very quickly), deep queue depths may just indicate an active and efficient system. A high queue length does not necessarily imply a performance problem.  **Note:** This value can be hard to interpret due to virtualization of storage in modern storage environments, which abstract away the physical hardware characteristics; this counter is therefore limited in its usefulness. |
| **Disk Read Bytes/sec**  **Disk Write Bytes/sec** | Measures total disk throughput. Ideally larger block scans should be able to heavily utilize connection bandwidth.  This counter represents the aggregate throughput at any given point in time. |
| **SQL Server Buffer Manager Perfmon Object**  The Buffer Manager counters are measured at the SQL Server instance level and are useful in characterizing a SQL Server system that is running to determine the ratio of scan type activity to seek activity. | |
| **Checkpoint pages/sec** | Measures the number of 8K database pages per second being written to database files during a checkpoint operation. |
| **Page Reads/sec**  **Readahead pages/sec** | **Pages reads/sec** measures the number of physical page reads being issued per second.  **Read-ahead pages/sec** measures the number of physical page reads that are performed using the SQL Server read-ahead mechanism. Read-ahead operations are used by SQL Server for scan activity (which is common for data warehouse and decision support workloads). These can vary in size in any multiple of 8 KB, from 8 KB through 512 KB. This counter is a subset of **Pages reads/sec** and can be useful in determining how much I/O is generated by scans as opposed to seeks in mixed workload environments. |

### Items to Note

If there is more than one instance of SQL Server running on your server, be sure that you select the instance in perfmon that corresponds to the instance that holds the particular database you are sizing.

If you are sizing a system for consolidation purposes, or in a shared storage environment, you should size at the aggregate level across all databases (possibly aggregating data from many systems).

### Setting Up a Test System at a Smaller Scale

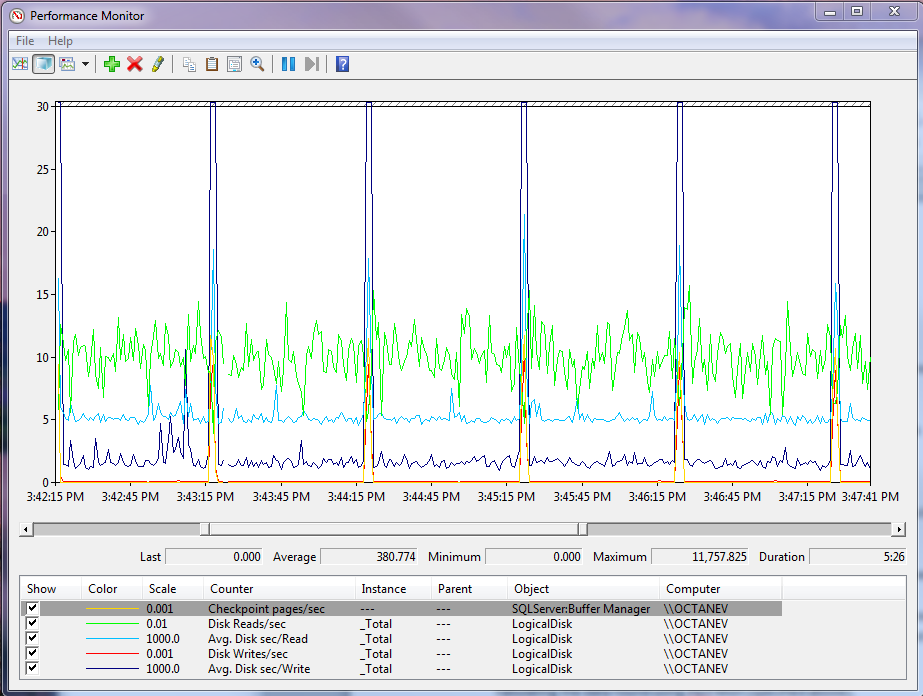
If the application that you are sizing is new, you should set up a test system or gather the data from an existing staging or test server in order to characterize the I/O workload. This can be done on a smaller scale; however, it is important to make sure it is proportional from both a hardware standpoint and a user standpoint. Choosing to use under-sized servers (causing the test system to be processor-bound) or a nonrepresentative workload will give you inaccurate data points. Smaller datasets that are not representative of the ultimate production dataset may be affected by caching layers both with the host and storage subsystem. In addition, test datasets that have a very different distribution of data values may result in inaccurate sizing conclusions.

## Examining the Data

### Reading the Performance Log Data

Look at the entire set of data captured in perfmon. Find a representative section of the data, after the test system caches are fully warmed and the system is in a steady state. Zoom in to that section of the data, and then capture the data for your calculations.

Figure 1 is representative of a typical OLTP workload.



**Checkpoint activity**

These are short-lived bursts of writes with a large amount of outstanding I/O requests. It is normal for latency to increase a bit during this operation.

**Latency (response time)**

Reads are in the 5-6ms range and writes 1-2ms with spikes to 20-30ms during checkpoints.

**Figure 1:** A typical OLTP workload

By selecting each line, you can find the average and burst (seen in perfmon as the maximum) read and write speeds for each logical volume. Note that if your data and log are separated onto separate LUNs, you will need to collect data for both logical disks.

### Checkpoints

A checkpoint is a forcing of data or log pages that have been modified to be written to disk. Checkpoints occur in five ways on a system:

1. On a periodic basis, depending on the amount of transaction log that has been generated since the last checkpoint.
2. When the transaction log is 70 percent full.
3. At a specified interval (that is, sp\_configure ‘recovery interval’). If this is set to 0, the behavior is determined by #1; otherwise it happens at the specific interval defined in the setting.
4. On demand (that is, using the CHECKPOINT command for a specific database).
5. Implicitly as part of certain operations with SQL Server (that is, BACKUP and creating database snapshots).

In perfmon, at the checkpoint counter, you should see regular bursts of activity, with little or no activity in-between. Under normal conditions these bursts should be short, generally less than 15 seconds if checkpoints are managed based on the transaction log volume written. If this is not the case, the system being examined is likely under-sized for the workload.

### Latencies

The latency (that is, the length of time from start to end for an individual I/O) should always be measured. The response time of the disks is the single most important measure of the health of an I/O subsystem, because it has a direct impact on performance of any workload that generates I/O.

# Sizing Considerations for the Physical Storage Components

## Typical Sizing of Physical Disks

In general, typical sizing numbers for disk are in the range of 100-130 IOPs per spindle on 10K RPM drives, and 150-180 IOPs per spindle on 15K RPM drives. These values are usually derived by running a workload against disks that are filled to 80 percent of the full drive capacity and then issuing random 8K I/O that has a ratio of 80 percent reads and 20 percent writes (which is typical for OLTP workload scenarios). For workloads that are more sequential in nature and have larger transfers, we have observed values in the range of 50-70 MB/s for each disk. Purely sequential workloads, with one thread reading from the drive, can deliver up to 125 MB/s on a 15K RPM drive. It’s important to note that these values can be significantly affected by the RAID configuration as well as the concurrency on the system; hence they should not be interpreted as an absolute measure.

In some cases you may find higher IOPs because:

* The disks are filled to only a fraction of their total capacity. Using only a small percentage of the capacity of a physical disk results in reduced seek times due to less physical movement of the disk head across the platter of the disk. This is commonly referred to as short-stroking the disk. Using a small amount of capacity of the disk can greatly increase the number of IOPs per disk. Figure 2 illustrates the effect of the total capacity used on the relative throughput per disk. As the number of disks increases, the size of the test data remains constant. Hence less physical capacity of each disk is used.

**Figure 2:** The total capacity used vs. the relative throughput per disk

* Most of the I/O is cached in the storage array cache. If a relatively small dataset is being used, or if the array has significant cache, you may see numbers higher than the standard sizing. This is especially true of very small datasets and for short bursts of writes that are absorbed by the cache.

If you observe IOPs numbers that are smaller than this, it is normally because:

* The same physical disks are being shared with another application with a significant enough workload to be affecting drive performance.
* The path to the drives (for example, fibers, cables, or storage processors in a SAN) cannot keep up with the activity that the drives can deliver.

## Impact of RAID on Physical I/O

RAID configuration affects the amount of actual physical I/O issued against the disks. When you measure I/O operations at the disk level, it is important to calculate the actual number of I/Os being generated and use that number for your calculations.

For the purposes of estimating actual physical I/O, consider the write penalty associated with the different RAID levels and use that in your sizing calculations. For sizing calculations assume that RAID 1+0 results in two physical writes for each logical write, RAID 5 results in 4 physical I/Os for each logical write, and RAID 6 results in 6 physical I/Os for each logical write. There are some exceptions to these rules, and some modern storage solutions provide the ability to mitigate the overhead via intelligence in the caching layer; however, we recommend sizing for the associated additional overhead each different RAID level incurs for writes.

## What About the Cache?

A common question we hear from customers is about the effect storage array cache has on SQL Server application workloads. Cache configurations and caching algorithms vary greatly between arrays, so generalizing can be difficult. In the context of SQL Server, we offer the following considerations.

### Read Cache

Read cache on storage arrays is normally used for prefetch purposes. When the array detects sequential I/O patterns, it may attempt to prefetch the next blocks of data into the cache so that I/O requests can be serviced faster. This results in increased transfer sizes within the storage system that are more efficient for this type of access pattern.

When SQL Server issues a scan, it does so using read-ahead operations. Read-ahead is very aggressive with respect to depth of the outstanding I/Os and the size of those, reading as much as 512 KB in a single I/O request. Read-ahead operations are not guaranteed to be truly sequential at the block level. Therefore, there is no guarantee that storage prefetch into read cache will benefit these operations. Read-ahead will read contiguous data pages; however, there can be gaps between these pages at the physical level for a number of reasons. These include interleaving with other objects in the file group and extent or logical fragmentation within an index. The sys.dm\_db\_index\_physical\_stats dynamic management view (DMV) provides information about the continuity of the underlying indexes. The following query and the data it returns provides an example of the information provided by this DMV for the clustered index of a table named ‘T1’.

SELECT database\_id, index\_id, object\_id

, avg\_fragmentation\_in\_percent, avg\_page\_space\_used\_in\_percent

, fragment\_count, avg\_fragment\_size\_in\_pages, page\_count

FROM sys.dm\_db\_index\_physical\_stats(db\_id(), object\_id('T1'), 1, NULL, 'SAMPLED')

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **database\_id** | **index\_id** | **object\_id** | **avg\_fragmentation\_in\_percent** | **avg\_page\_space\_used\_in\_percent** | **fragment\_count** | **avg\_fragment\_size\_in\_pages** | **page\_count** |
| 13 | 1 | 85575343 | 0.01 | 99.7158388930072 | 1685 | 72.5080118694362 | 122176 |

The practical benefit offered by the read cache for SQL Server depends heavily on the underlying physical organization of the data and can be very limited. We recommend that you favor write cache over read cache for SQL Server application workloads and maintenance operations such as BACKUP and CHECKDB.

### Write Cache

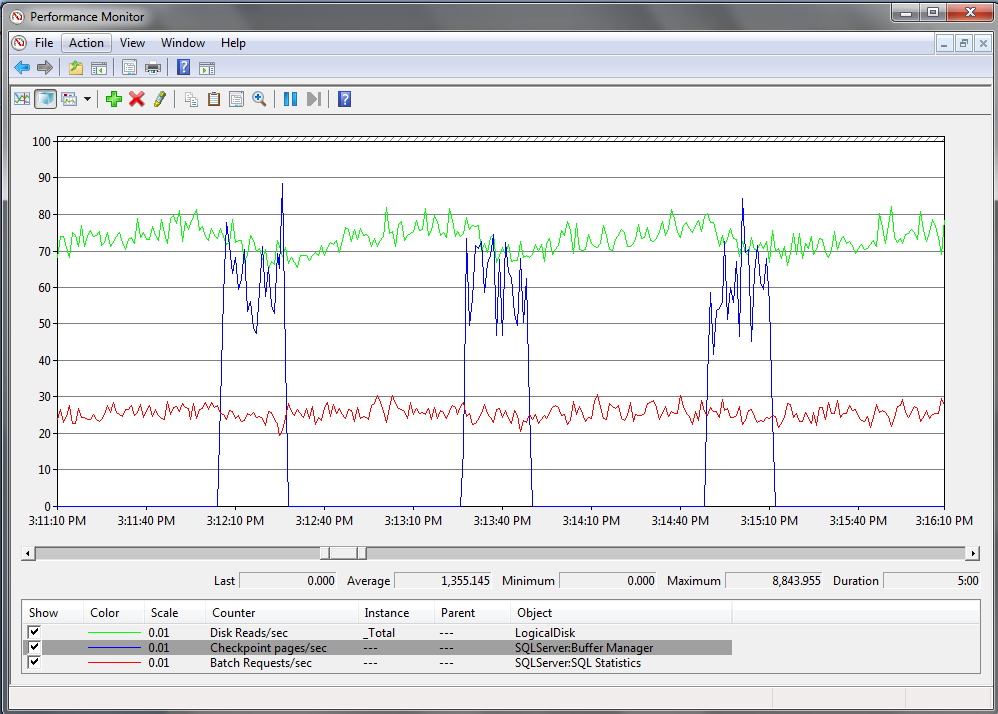
Write cache on storage arrays is provided to help maintain low latency for write requests being issued against the array. Most modern arrays have a cache that is battery backed and mirrored with storage vendors guaranteeing that a write to cache will make it to the physical media. This is something you should always confirm on any storage using cache for writes to ensure durability of SQL Server I/O.

There are two main benefits write cache provides to SQL Server workloads:

* The ability to maintain very low latency on writes to the transaction log.
* The ability to absorb periodic bursts of write activity issued by checkpoint operations.

Write cache plays an important role on transactional workloads for these reasons, and we recommend heavily favoring write cache whenever possible. We recommend configuring at this 80/20 ratio of writes to reads (or even 90/10). Configuring cache at this level is not possible on all storage arrays. In addition, if the storage array is shared across many servers, such tuning may not be feasible, because cache is a shared resource.

While write cache is a good thing for the reasons listed earlier, it’s important to keep in mind that at some point writes must be written (that is, destaged) to physical media. Figure 3 illustrates the background effect that the destaging of these writes has on workload.

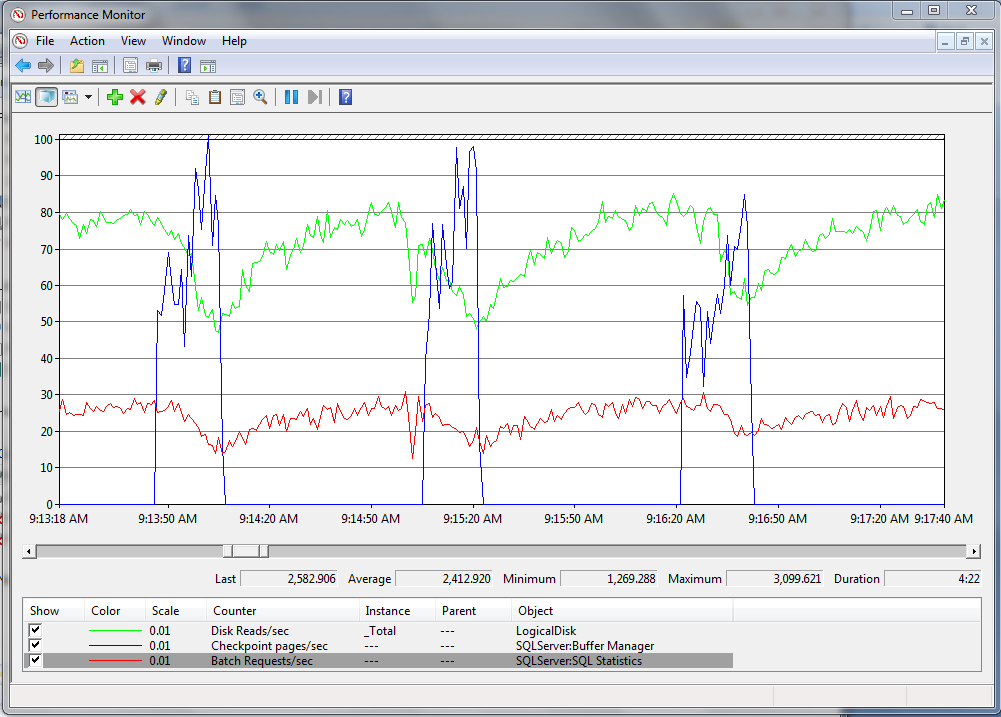


There is a drop in read IOPs and Batch Request/sec after all writes from the checkpoint have *completed*. With more cache the drops are less pronounced.

Figure 3: The effect of destaging on a machine with 8 GB cache, 90 percent allocated to writes

Notice the drop in read throughput after each checkpoint occurs. This is a result of the additional I/O happening in the background on the array as the writes are destaged from cache to the physical disks. Destaging operations cannot be captured by perfmon, because they occur only at the physical level, but perfmon can illustrate the impact on the workload.

The example in Figure 3 is taken from an array with 8 GB of usable cache, 90 percent of which is allocated to writes. On a configuration with less usable cache, the drops in throughput are more pronounced. The example shown in Figure 4 is taken from the same configuration configured with 2 GB of usable write cache.



There is a drop in read IOPs and Batch Request/sec after all writes from the checkpoint have *completed*.

Figure 4: The same system as in Figure 3, with 2 GB of write cache

The bottom line when it comes to cache is this: Be very careful when you make sizing decisions based on the benefit you expect cache to provide. Especially in shared storage environments, this resource has the potential to be shared across many servers.

## The Nondisk Components: Connectivity, Array Service Processors, and Internal Array Bandwidth

Sizing storage systems is much more than just sizing the disks. Every component in the path to the physical disk has an associated throughput limit. Every component has the potential to become the bottleneck. Considering the bandwidth offered by the connectivity between server and storage as well as the capacity of the service processors within the array is an important part of sizing. This is especially true for throughput-intensive workloads such as data warehousing.

In-depth discussion of each of these is beyond the scope of this document; however, when sizing, take the time to understanding the theoretical throughput limitations, keeping in mind that theoretical limits are rarely achieved. Each of the components discussed in this section is in the path from the server to the disk, which has a fixed amount of throughput.

### **Connectivity**

More and more we are observing connectivity, specifically bandwidth, becoming a bottleneck for throughput-intensive SQL Server workloads. It is important to consider and understand the limits of the components involved.

Mainstream Fibre Channel host bus adapters (HBAs) are rated at a theoretical throughput of 4 gigabits per second (Gbps) to 8 Gbps. For most SQL Server workloads, the actual achievable throughput is slightly lower than this. Most commonly we have observed approximately 360 MB/s for a 4 Gbps port; however, for a finely tuned configuration, throughput values may be higher. When you look at connectivity, consider the following:

* The throughput capacity of the HBAs or network interface cards (NICs)
* The throughput capacity of all switch ports in the connectivity layer, and whether those are shared with other servers on the storage network
* The throughput capacity of the Fibre Channel or iSCSI ports on the storage array
* The number of paths between the server and the storage

As a practical reference, the following table lists some practical observed limits of typical components in the connectivity layer for both Direct Attached Storage (DAS) and Storage Area Network (SAN) storage.

**Note:** These are approximate numbers based on our experience. They are provided for reference and for use in identifying when connectivity limits are reached. They should not be considered exact. Vendor documentation should be used to determine the limits of any specific hardware.

|  |  |
| --- | --- |
| **Component** | **Limits** |
| SAS cable | Theoretical: 1.5 gigabytes per second (GB/s)  Typical: 1.2 GB/s |
| PCI-X v1 bus | X4 slot: 750 MB/s  X8 slot: 1.5 GB/s  X16 slot: roughly 3 GB/s |
| PCI-X v2 bus | X4 slot: 1.5 – 1.8 GB/s  X8 slot: 3 GB/s  Note: Be aware that a PCI-X bus can be v2 compliant but still run at v1 speeds. |
| Fibre Channel HBAs, switch ports and front end Fibre Channel array ports | 4 gigabits per second (Gbps): 360-400 MB/s  8 Gbps: Double the speed of 4 Gbps  **Note:** Make sure to consider the limits of the PCI bus. An 8-Gbps card requires a PCI-X4 v2 slot or faster. |

### **Array Service Processors**

Service processors on a storage array also have a limited amount of throughput they can offer. This can vary significantly between classes of arrays and between generations. On storage arrays with a very large number of physical disks, it is not uncommon for the service processors to become a bottleneck. In many cases these cannot offer the aggregate amount of throughput the disks can deliver. We recommend that you check with the storage vendor to determine theoretical throughput capacity. When you look at the service processors, consider the following:

* The number of service processors in the array and their associated throughput capacity.
* How the LUNs are balanced across these service processors. An imbalance of LUNs across the service processors can result in overloading a specific service processor. This is a very important consideration when configuring storage.

### **Internal Array Bandwidth**

Each array has a fixed amount of internal bandwidth. Ensuring disks are balanced across backend buses will result in the best performance. Make sure to consider:

* Throughput capacity of the backend buses on the array.
* How the physical disks being used are balanced across those buses.

One interesting thing to note is that as SSD becomes more prevalent, it is likely that bottlenecks will be pushed from the physical components to the components above. Due diligence in ensuring the above components are optimized and configured correctly will become critical in achieving potential of SSD devices.

This section covered some of the considerations for sizing the physical hardware. In today’s virtualized storage environments, people who manage and deploy SQL Server applications often have limited knowledge of the physical configuration, which is commonly managed by a different set of people. This separation of duties makes troubleshooting performance problems related to I/O very difficult. In addition, there are considerations related to sizing if advanced array technologies are used. Some of these are discussed in the appendixes of this paper.

# Examples of I/O Characteristics for Common Workloads

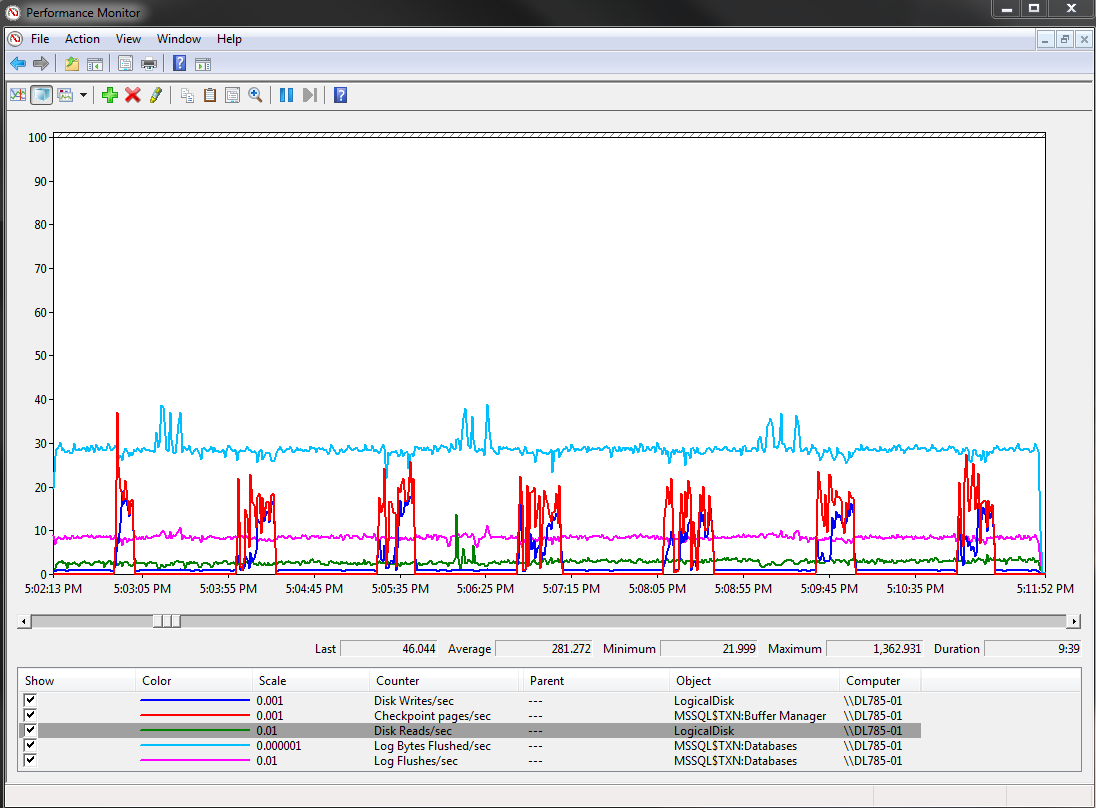
The following provides some specific examples of a workload running on a specific storage configuration. These are provided as illustration and example of applying the concepts in this paper.

## Point-of-Sale System

A point-of-sale application is a common type of OLTP application. The characteristics of this application are heavy writes with a relatively low number of reads. The following configuration was used in testing a workload of this type in the SQL Server Customer lab. This storage system was configured as follows:

* Storage: EMC® CLARiiON® CX4-960
* SQL Server data files: Single Meta LUN, 48 spindles configured RAID 1+0
* SQL Server log file: Single LUN, 4 spindles configured RAID 1+0
* 8 GB of usable cache configured as 90/10 write-to-read ratio

Here is data collected from that system.



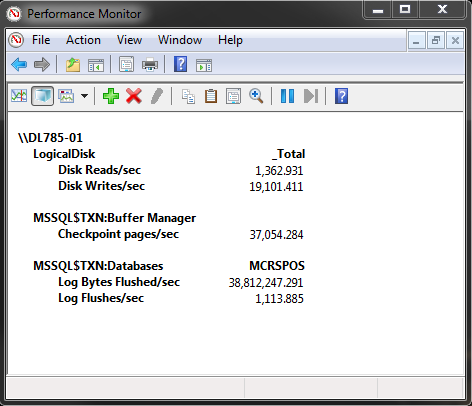
Checkpoints – spikes to 19,000 IOPs and roughly 10 seconds in duration

Log - roughly 900 IOPs (roughly 30 MBps)

Relatively low reads of 200-300 IOPs

**Figure 5:** Perfmon data on a point-of-service system

Figure 6 shows the maximum observed values.



**Figure 6:** Maximum values observed on the example point-of-sale system

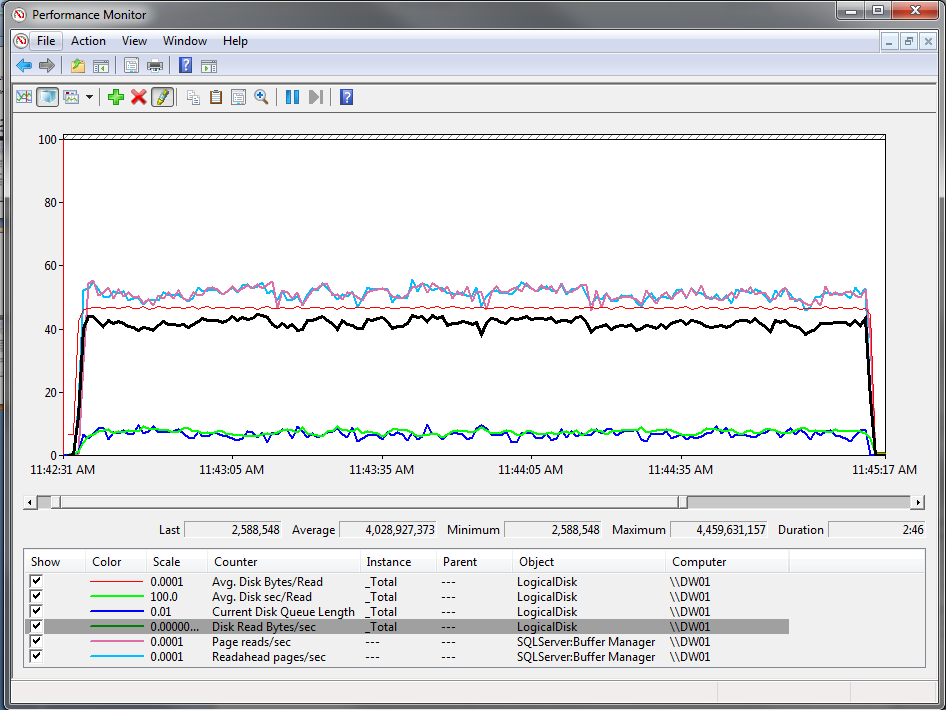
In sizing the point-of-sale system, spindle count is important. The burst of writes can be absorbed by the cache and destaged in the background without affecting the workload, because the other I/O is relatively light. Between each checkpoint operation, there is nearly a minute of very low I/O activity, giving the disks enough time to absorb the writes. Even though 48 disks is less than what would be calculated using sizing numbers noted earlier in this paper, for this particular workload it is acceptable. Sizing assumptions were made in this case based on the fact that the array was dedicated to the database server and there was no sharing of any components in the entire path to the disks (cache, service processors, or connectivity). In addition, the assumption was made that for this scenario the access pattern would be heavy checkpoints with low I/O activity otherwise. In production, this system may need to be sized to accommodate additional I/O that may be introduced by other operations such as online reporting.

## Decision Support System Query Workload

The data in the following example was captured from a system configured as follows:

* Storage: HP MSA60:
  + 5 x HP SAS P800 controllers with 512 MB cache on each controller.
  + Each controller is attached to an MSA60 shelf.
* SQL Server data files: 24 LUNs, one RAID 1 pair for each LUN with 48 disks total. RAID 1 is chosen in this case because it offers the best sequential performance. Striping across the LUNs is achieved via SQL Server data files.
* SQL Server log file: 1 LUN, single RAID 1 pair.

Here is an example of what the data looked like on one such system.



Current Disk Queue Length = ~ 670

(But reasonable response times given size and depth of outstanding I/O)

Disk Read Bytes / sec = ~ 4 GB/s

This is reasonable given the configuration

Readahead pages/sec is almost the same as Page reads/sec.

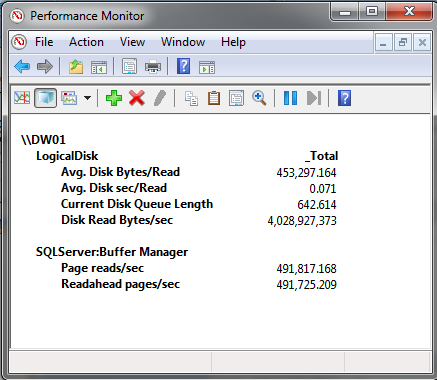
Avg.Disk Bytes/Read =~ 500 KB, representing close to the maximum size of a Readahead operation. This indicates good continuity of the data pages at the physical level.

**Figure 7:** Perfmon data for decision support system

Figure 8 illustrates the averages for the duration of the test. Notice that latency is relatively high compared to the recommended latency values discussed earlier in this paper. This is due to:

* The size of the I/O transfers.
* The amount of outstanding I/O being pushed against the system.

When sizing on and interpreting this data, consider the aggregate throughput achieved. This throughput is nearly 160 MB/s for each LUN (RAID 1 pair), which is not the maximum capacity from two 15K rpm spindles. This is an example of a workload where considering the nondisk components and aggregate bandwidth is critical. In this particular example, configuration of the connectivity—specifically the PCI configuration—was the limiting resource.



**Figure 8:** Aggregate throughput observed on a data warehouse system

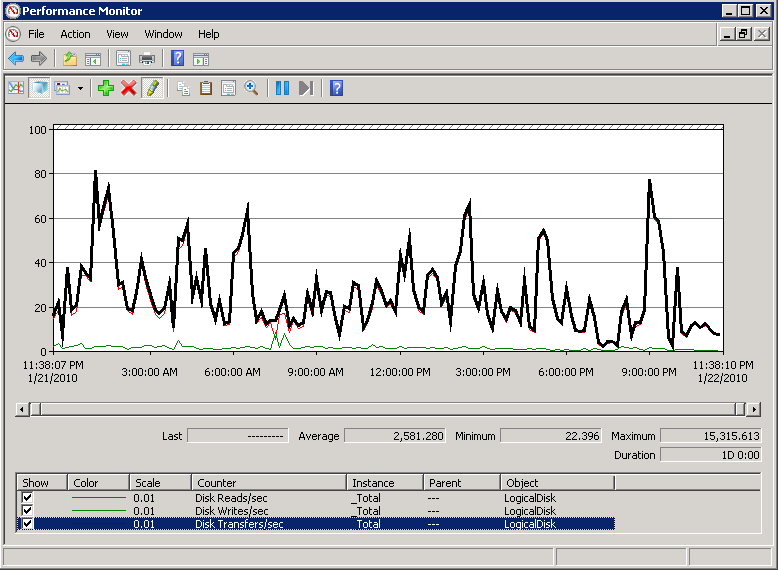
# Example: Translating the Data to Estimated System Requirements

This section walks through an example of Microsoft’s SAP ERP system, looking at workload over a 24-hour period and translating the findings into sizing requirements.

At a high level, the process of analysis consists of these steps:

1. Look across the entire 24-hour period, measuring:
   * Sustained maximum IOPs and throughput.
   * Peaks.
   * Read/write ratio.
   * Average sizes for the I/O transfers.
2. Zoom in on the busy portions of the time period and observe in more detail the access patterns for that time period.
3. Map this data into a proposed configuration with respect to number of disks and bandwidth needed to support the workload.

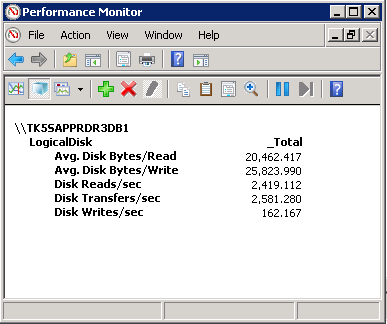
The first step is to look at the entire time period.



Sustained periods of 8,000 IOPs (with bursts to 15,000 IOPs).

Writes are less than 10 percent of the total IOPs on average.

Periods of heavy write activity are in the 2,000 IOPs range.

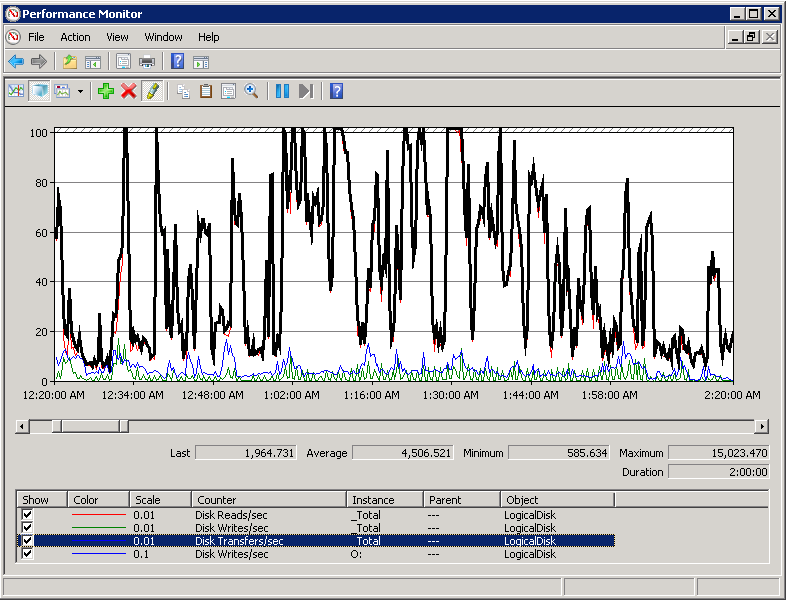


Average I/O sizes are <32K

**Figure 9:** Perfmon data on SAP system over a 24-hour period

For this time period there are sustained averages of 8,000 IOPs with spikes in the range of 15,000 IOPs. Writes account for a small percentage of the total overall workload. The average transfer sizes are in the range of 20-25 KB. You can size based on the assumption that there will not be additional physical I/Os for a single logical I/O. These I/Os will not span stripe boundaries. The goal is a system that can maintain good performance (healthy latencies) during periods of high activity. Consider sustained maximums, peaks, and projected growth when you determine the storage configuration.

The next step is to zoom in on one of the busier time ranges and observe I/O behavior over that range.



Sustained periods of more than 10,000 IOPs during night time activity.

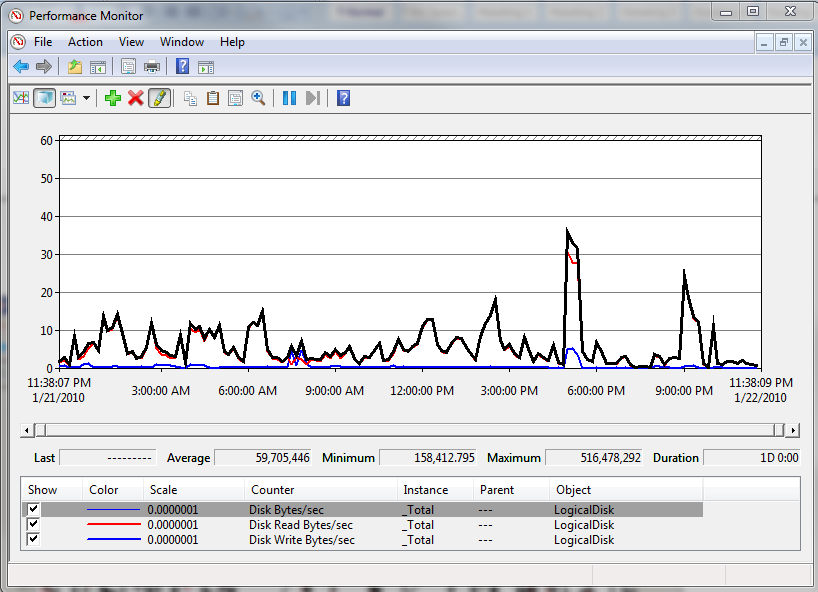
Writes (green) represent checkpoint activity and log writes with sustained maximums near 1,000 IOPs

Log writes (blue) are relatively low – averaging less than 100 IOPs.

**Figure 10:** Perfmon data from a busy period on an SAP system

The data shown in figure 10 is consistent with the broader time range. It also provides additional insight into the breakdown of the writes as they relate to checkpoint activity versus the transaction log.

This data provides information about sizing from an IOPs perspective. To complete the picture, look at the throughput in the same manner to determine the bandwidth necessary in the nondisk components to support the workload. Figure 11 shows the same workload measuring the total throughput in bytes per second, both aggregated and broken down by read and write.



Sustained periods of approximately 400 MBps.

Maximum Peaks of approximately 500 MBps.

**Figure 11:** Perfmon data from a busy period on an SAP ERP system

To support the throughput required by the system, you need a minimum of two 4-Gbps paths to the storage array. This is because the sustained throughput is 400 MB/s with peaks to 440 MB/s. As discussed earlier, the practical limit of a 4Gbps connection is about 360 MB/s. In production deployments, there should always be redundant paths in the connectivity layer.

The next step is to use this information to size the underlying storage system.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **I/O type** | **Average IOPs** | **Sustained maximum IOPs** | **Peak IOPs** | **Sustained maximum MB/s** | **Peak MB/s** | **Average I/O size** |
| Data file reads | 2,500  (over a 24-hour period)  4,500  (during busy times) | 10,000 | 15,000 | 400 MB/s | 440 MB/s | 20K |
| Data file writes | Less than 200 | 1,000 | 1,900 | 70 MB/s | 85 MB/s | 25K |
| Log file writes | Less than 200 | 200 | 200 | 40 MB/s | 16 MB/s | 16K  (60K maximum) |

After the workload is characterized, you can map this into a physical configuration. This example shows how to size based on the sustained maximums. After you make this determination, the next step is to consider peaks and growth.

When sizing the log disks, it is acceptable to size on throughput because of the sequential nature of the transaction log. In addition, the configuration utilizes dedicated disks for the transaction log with no sharing between log and data I/O. Our observation is that a pair of RAID 1 disks can sustain in the range of 70 MB/s write throughput. However, keep in mind the fact that both capacity and the impact of log backups, as well as other processes that read the log, influence sizing decisions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sizing for** | **Total IOPs (data) and MB/s (log)** | **RAID adjusted** | **Assumptions** | **Required number of disks** |
| Data files | 11,000 IOPs | 12,000 = (10,000 reads) + (1,000 \* 2)  RAID 1+0 | 180 IOPs per 15K disk | Approximately 67 |
| Log files | 15 MB/s | 15 MB/s \* 2  Raid 1 +0 | 70 MB/s per 15K disk | Approximately2 |

In this case, a 68-disk RAID 1+0 configuration is needed to support the workload for data files, and a minimum of 2 disks is needed for the log. To use peak values for sizing, increase the disk requirements by 25 percent, because the sizing is based on 12,000 physical I/Os. In production, this system is currently sized with 128 15K disks configured in RAID 1+0. Our example shows how one would size based on the data captured.

# Summary

This paper provides an initial look at the tools available to characterize a SQL Server workload and considerations for then sizing a storage system for that workload. Sizing storage systems for SQL Server is a complex task because of the fact that characteristics can vary greatly between applications. All too often our team encounters systems that suffer poor I/O performance because they were not sized properly. Many times this is because only capacity requirements, with respect to the size of the database(s), were considered for sizing. I/O performance is a critical component of any DBMS; this is especially true for systems with datasets that are much larger than the physical memory available for their servers. We hope that these methodologies and examples provide helpful insight into sizing storage subsystems. We plan to expand further on these topics in future articles.

# Appendix A: An In-Depth Example of Calculating Physical I/O

The example in this appendix illustrates the effect of RAID level on physical I/O by comparing the difference between RAID 1+0 and RAID 5. It also demonstrates how other components, such as storage array cache, come into play.

This section also provides a sample methodology for factoring in the effect cache can have on physical I/O. The differences in the impact of RAID level on writes are largely consistent across any storage array. The impact of cache varies based on the nature of access patterns and the type of array being used.

This example was captured from SQL Server lab testing in which we compared spinning media with SSD. Our goal was to measure and compare the relative throughput of spinning media with that of enterprise-class SSD. The results were captured by running an OLTP workload at high concurrency on the server. In the case of SSD the workload was CPU bound, and for the traditional drive configuration the workload was disk bound.

Here are some considerations to bear in mind as you look at the data:

* Tests were first run against the SSD configuration. Through experimentation it was determined that using four physical SSDs could provide enough I/O throughput for the workload. Using this configuration the CPU resources of a 16-physical-CPU-core server could be exhausted.
* The amount of IOPs achieved through this experiment was used to estimate sizing of the number of physical disks needed to support the same workload with traditional Fibre Channel drives. When estimating this number we assumed a much high per disk IOPs than standard sizing because only a small fraction of the total capacity would be used, resulting in very little seek times at the physical disk level (short stroking).
* RAID 5 was chosen for the SSD test with the assumption that in actual deployments this is more realistic from a cost perspective and it is a feasible option given the significant increase in IOPs per SSD. RAID 1+0 was also selected for the spinning media tests because this has been recommended practice for high throughput OLTP applications on traditional spinning media.

Here are the results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Disks | Number of disks | RAID level | Total logical I/Os  (as measured via perfmon) | Total physical I/O  (RAID adjusted) | I/Os per device |
| SSD | 4 | 5 | 12,044 | 17,876 | 4,469 |
| Traditional 15K Fibre Channel drives | 34 | 1+0 | 12,362 | 14,465 | 425 |

For more information about these tests and how the number of actual physical I/Os were calculated, see the white paper [EMC Symmetrix DMX-4 Enterprise Flash Drives with Microsoft SQL Server Databases](http://www.emc.com/collateral/hardware/white-papers/h6018-symmetrix-dmx-enterprise-flash-with-sql-server-databases-wp.pdf) (http://www.emc.com/collateral/hardware/white-papers/h6018-symmetrix-dmx-enterprise-flash-with-sql-server-databases-wp.pdf). This testing was done in partnership with EMC using a DMX-4 storage array.

Logical I/O was monitored by using perfmon on the server. These measures were used in combination with cache hit rates (as measured within the storage array) to determine the actual number of physical IOPs. We used the following formula:

Physical\_IO = ((Total\_IO \* Read\_IO%) - Read\_CacheHits) + ((Total\_IO \* Write\_IO%) \* RAID\_Factor)

Where the components are defined as follows:

* Physical\_IO: The actual I/O at the physical disk level.
* Total\_IO: The total logical I/O measured via perfmon.
* Read\_IO%: The percentage of the total I/O that was reads.
* Read\_CacheHits: The number of reads that were serviced from the array cache without needing to read from the physical disk. In this particular scenario, we obtained this measurement through use of advanced storage array specific monitoring tools. If this data is not available, we recommend that you calculate based on the worst-case scenario, which would be zero cache hits.
* Write\_IO%: The percentage of the total I/O that was writes.
* RAID\_Factor: The additional I/O required for write operations (2 for RAID 1+0 and 4 for RAID 5).

During testing, we measured the cache hit ratio for reads by using array monitoring utilities.

In addition to the RAID level, other things can impact the amount of actual physical I/O that occurs. These can include:

* Reads that are serviced from the storage array cache without needing to go to the physical disk.
* Writes that are coalesced within the storage array cache into fewer physical transfers.
* The size of I/Os. If large I/Os that span strip boundaries on the disks are issued, the actual number of physical I/Os being generated will be higher than the logical I/O reporting from Windows. In this example, I/O transfer sizes were always less than or equal to 64K, so this was not a consideration.

# Appendix B: Other Sizing Considerations

As storage technologies evolve, storage arrays provide advanced features that can impact sizing decisions. This appendix contains a few examples of these. This is not an exhaustive list or an in-depth explanation of the technologies but rather a list of some common technologies that can affect sizing decisions.

## Thin Provisioning

Thin provisioning is a storage technology that enables an array to provision a LUN of a certain size without having the actual physical storage allocated. The physical storage is allocated on demand as more capacity of the LUN is used. The most common deployment practice is to allocate LUNs from a pool of physical disks that are shared by any number of servers. This consolidation approach enables more efficient use of storage.

The sizing concepts discussed in this paper are no different for deployments using thin provisioning with respect to sizing for the required IOPs, bandwidth needed, and RAID considerations. Consider the following when using thin provisioning:

* To take advantage of physical capacity being allocated on demand, 1) NTFS volumes must be formatted using the quick option, and 2) SQL Server databases must be created using instant file initialization.
* Using thin provisioning results in shared disks between any server and the application using the resources. Sizing needs to be done at an aggregate level, considering all consumers of the storage. Pay special attention to latency in these scenarios. Observing high latencies with no change in your particular application’s I/O characteristics generally indicates competition for resources with other users of the storage.
* Thin provisioning can result in much less continuity of data at the physical level because the storage is only committed as it is used. This means that regardless of host access patterns, access patterns at the physical level will be random in nature. This may be suboptimal for scan-intensive workloads.

## Snapshots and Clones

Many arrays now provide the ability to create snapshots or clones of LUNs. In addition, these technologies integrate with SQL Server backup functionality and provide the ability to back up large databases in a very short period of time. If you are working with these technologies, two main considerations have the potential to impact I/O performance:

* Snapshots are generally copies of LUNs that do not contain a full physical copy of the data. Instead the snapshot LUNs maintain the original data when any modifications are made on the source LUNs. Applications using the snapshot LUNs affect the source LUNs, because requests for any data that has not changed since the snapshot was created are directed to the original LUNs.
* Clones are full copies of source LUNs maintained by the storage system. Mirroring of source LUNs can affect performance of the source LUNs, because any modifications to the source LUNs must also take place on the cloned LUNs. This is especially true if the clone LUNs utilize fewer physical disks or are configured with a RAID level that introduces more overhead for writes.

**Note:** Terminology and implementations vary among vendors.

## Storage-Based Replication

It is becoming more and more common for SQL Server application deployments to utilize storage-based replication technologies for high availability and disaster recovery purposes. Storage-based replication can introduce significant effects on the application. This is especially true for deployments that use synchronous replication, because the network latency between storage arrays directly affects latency of I/O issued by SQL Server.

# References

[SQLCAT.com](http://sqlcat.com/Default.aspx)

[Storage Top 10 Best Practices](http://www.microsoft.com/technet/prodtechnol/sql/bestpractice/storage-top-10.mspx)

[SQLIO download page](http://www.microsoft.com/downloads/details.aspx?familyid=9a8b005b-84e4-4f24-8d65-cb53442d9e19&displaylang=en)

[SQLIOSim tool KB article](http://support.microsoft.com/kb/231619)

[SQL Server Storage Engine Team Blog](http://blogs.msdn.com/sqlserverstorageengine/)

[SQL Server I/O Basics](http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/sqlIObasics.mspx)

[SQL Server I/O Basics – Chapter 2](http://technet.microsoft.com/en-us/library/cc917726.aspx)

[Predeployment I/O Best Practices](http://technet.microsoft.com/en-us/library/cc966412.aspx)

[Disk Partition Alignment Best Practices for SQL Server](http://msdn.microsoft.com/en-us/library/dd758814.aspx)

[EMC Symmetrix DMX-4 Enterprise Flash Drives with Microsoft SQL Server Databases](http://www.emc.com/collateral/hardware/white-papers/h6018-symmetrix-dmx-enterprise-flash-with-sql-server-databases-wp.pdf)

[Implementing EMC CLARiiON CX4 with Enterprise Flash Drives for Microsoft SQL Server 2008 Databases](http://www.emc.com/collateral/hardware/white-papers/h6080-clariion-cx4-enterprise-flash-sql-server-2008-wp.pdf)