

# **Faster SQL Server Access with Solid State Disks**

---

By Woody Hutsell  
Texas Memory Systems



# Contents

---

Executive Summary.....	3
Perspectives From Industry .....	4
The Problem of I/O Wait Time .....	5
Traditional Approaches To Improving SQL Server Performance .....	6
Introduction to Solid State Disks .....	7
Identifying I/O Subsystem Problems.....	8
SQL Server Components That Should Be Moved to Solid State Disk.....	11
Identifying the Most Frequently Accessed SQL Server Tables.....	14
Customer Case Study: BetOnSports.com.....	15
Figure 1: Chart-- Maximum I/Os per Second.....	5
Figure 2: Processor Performance when Writing to Hard Disk.....	9
Figure 3: Processor Performance when Writing to Solid State Disk.....	10

# Section 1

---

## Executive Summary

This whitepaper discusses methods for improving SQL Server database performance, using solid state disks to accelerate the most resource-intensive data that slows performance across the board.

To this end, it discusses methods for identifying I/O performance bottlenecks and points out components that are the best candidates for migration to a solid state disk. An in-depth explanation of solid state disk technology and possible implementations are also included.

For more in-depth information, visit [www.superSSD.com](http://www.superSSD.com) or contact one of the following:

- Existing customers contact [support@superSSD.com](mailto:support@superSSD.com).
- Potential customers contact [sales@superSSD.com](mailto:sales@superSSD.com).

## Section 2

---

# Perspectives From Industry

*From chapter 6 of Tuning and Sizing NT Server by Curt Aubley, reprinted on Microsoft.com:*

"No matter how well you sized and tuned your NT Server disk subsystem, the disk subsystem is still composed of physical devices that are slow when compared to RAM. There may be times, regardless of how well you have load balanced and tuned your disk subsystem, when 'hot spots' still exist. 'Hot spots' are defined here as areas of the disk subsystem that application(s) consistently request services from which greatly impact the application's performance. Consider another technique to alleviate hot spot situations; try implementing RAM disks."

*From the Morgan Keegan System Area Network Conference, Solid State Caching, page 85-86:*

"Since 1964 the capacity of disk drives has increased by over 6,000 times while the average seek latency has only seen about a tenfold improvement. The continued increase in capacity without corresponding improvement in access times has caused a steady decline in the access density of disk media, which is the ratio of I/Os per second divided by the capacity of the disk drive. As a result many storage administrators are finding that in spite of the fact that they are adding storage capacity at an unprecedented pace, system performance does not improve. In other words, their storage infrastructures are I/O bound. One way to solve this dilemma is to spread the information access workload across a large number of systems. In many situations, however, a more efficient method of improving performance is the implementation of a solid-state disk (SSD) caching solution. SSD offers the fastest storage media available."

"However, we believe SSD caching will see increasing adoption in the system network as the optimal storage resource for certain data types requiring high levels of availability. The return on investment of SSD increases dramatically in a centralized storage configuration in which the cost may be amortized across a larger infrastructure and SSD can be added incrementally as needed..."

## Section 3

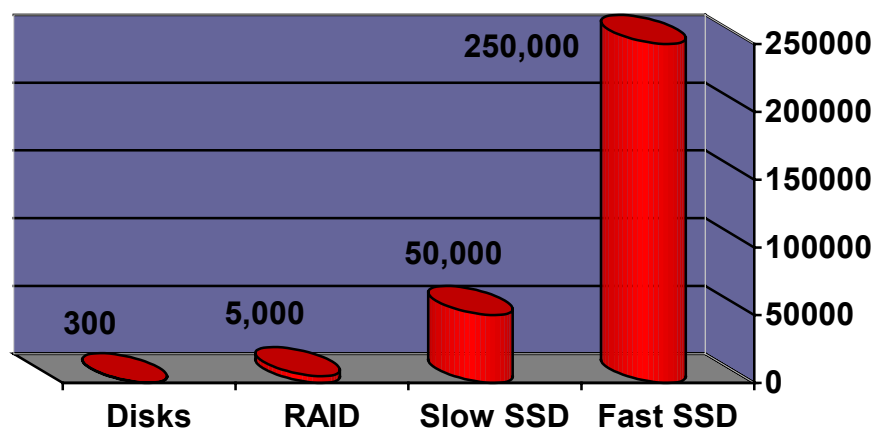
### The Problem of I/O Wait Time

Often, additional processing power alone will do little or nothing to improve SQL Server performance. This is because the processor, no matter how fast, finds itself constantly waiting on mechanical storage devices for its data. While every other component in the “data chain” moves in terms of computation times and the raw speed of electricity through a circuit, hard drives move mechanically, relying on physical movement around a magnetic platter to access information.

In the last twenty years, processor speeds have increased at a geometric rate. At the same time, however, conventional storage access times have only improved marginally. The result is a massive performance gap, felt most painfully by database servers, which typically carry out far more I/O transactions than other systems. Super fast processors and massive amounts of bandwidth are often wasted as storage devices take several milliseconds just to access the requested data.

When servers wait on storage, users wait on servers. This is I/O wait time. Solid state disks are designed to solve the problem of I/O wait time by offering 250x faster access times (.02 milliseconds instead of 5) and 40x more I/O transactions per second (250,000 instead of 5000) than RAID.<sup>1</sup>

Figure 1: Chart-- Maximum I/Os per Second



<sup>1</sup> The performance numbers cited here are those of the RamSan-320 solid state disk, sold by Texas Memory Systems.

## Section 4

---

# Traditional Approaches To Improving SQL Server Performance

Decreasing application performance under heavy user loads is not a new story for most enterprises. As the number of concurrent users increases, the response time to users worsens. The knee-jerk reaction to this problem is to look at two sources for database performance problems:

- **Server and processor performance.** When performance wanes, one of the first things that most IT shops do is add processors to servers or add servers to server farms.
- **SQL Statements.** Enterprises invest hundreds of thousands of dollars squeezing every bit of efficiency out of their SQL statements. The personnel required to painstakingly evaluate and iterate the code are a major cost for many IT shops.

In many cases, these likely sources for database performance problems are just masquerading the true cause of poor database performance: the gap between processor performance and storage performance. Adding servers and processors will have a minimal impact on database performance, and will compound the resources wasted as even more processing power waits on the same slow storage. Tuning SQL can result in performance improvements, but even the best SQL cannot make up for poor storage I/O. In many cases, features that rely heavily on disk I/O cannot be supported by applications. In particular, programs that result in large queries and that return large data sets are often removed from applications in order to protect application performance.

When system administrators look to storage, they frequently try these different approaches to resolving performance problems:

- **Increase the number of disks.** Adding disks to JBOD or RAID is one way to improve storage performance. By increasing the number of disks, the I/O from a database can be spread across more physical devices. As with other approaches, this has a trivial impact on decreasing the bottleneck.
- **Move the most frequently accessed files to their own disk.** This approach will deliver the best I/O available from a single disk drive. As is frequently pointed out, the I/O capability of a single hard disk drive is very limited.

The best solution to the performance gap is to implement solid state disks for the most frequently accessed database components.

## Section **5**

---

### **Introduction to Solid State Disks**

Strictly, a solid state disk (or SSD) is any storage device that does not rely on mechanical parts to input and output data. Typically, however, the term refers to storage devices that use RAM as the primary storage media. Data is stored directly on RAM chips and accessed from them. This generally results in storage speeds far greater than is even theoretically possible with conventional, magnetic storage devices. To fully make use of this speed, SSDs typically connect to servers or networks through multiple high-speed channels (SCSI on the low end, Fibre Channel and other more recent connections on the upper end).

What separates a solid state disk from conventional memory is non-volatility. An SSD typically includes internal batteries and backup disks so that, in the event of power loss or shutdown, the batteries keep the unit powered long enough for data to be mirrored onto the backup disks. Because of this, SSDs offer the raw speed of system memory without the disadvantage of losing data when powered down. Because of the lack of mechanical devices in the main data chain, SSDs typically have lower maintenance costs and higher reliability (including a higher MTBF) than conventional storage. They are generally cheaper to implement than solutions involving caching data to system memory, and can offer a superior variety of high-availability and management features.

## Section **6**

---

### **Identifying I/O Subsystem Problems**

While a solid state disk can be used to speed up almost any SQL Server database, it is most needed in installations where your servers are experiencing I/O wait time. It is important to note that there are a number of elements involved in server I/O. The PCI (or other) bus, host bus adapter, interface, storage network switch, RAID controller, and hard disk drives are all involved in every I/O between server and storage. Theoretically, any one of these points can cause an I/O bottleneck. In practice, however, the hard disk drives are the most likely culprit. Simply put, every component in the I/O process is solid state except for the hard disk drives. Therefore, when I/O wait time is identified the most likely cause is the hard disk drives. Adding a solid state disk drive can eliminate I/O wait time.

Looking at operating system performance is the best way to identify I/O wait time. The tools to evaluate operating system performance vary by operating system.

For Microsoft Windows operating systems the best tool for system performance analysis is the Performance Monitor. Unfortunately, the Performance Monitor does not provide the actual I/O Wait Time statistic. Performance Monitor does include actual processor performance levels. By looking at Processor: % Processor Time it is possible to see the actual work being done by the processor. If you know that your system is being hit hard by transactions and yet your % Processor Time is well under 100% it is possible to infer an I/O wait problem. Systems that implement solid state disk show high % Processor Time numbers.

For example, two screen shots are included from Windows Performance Monitor. The tested system is running Windows NT.

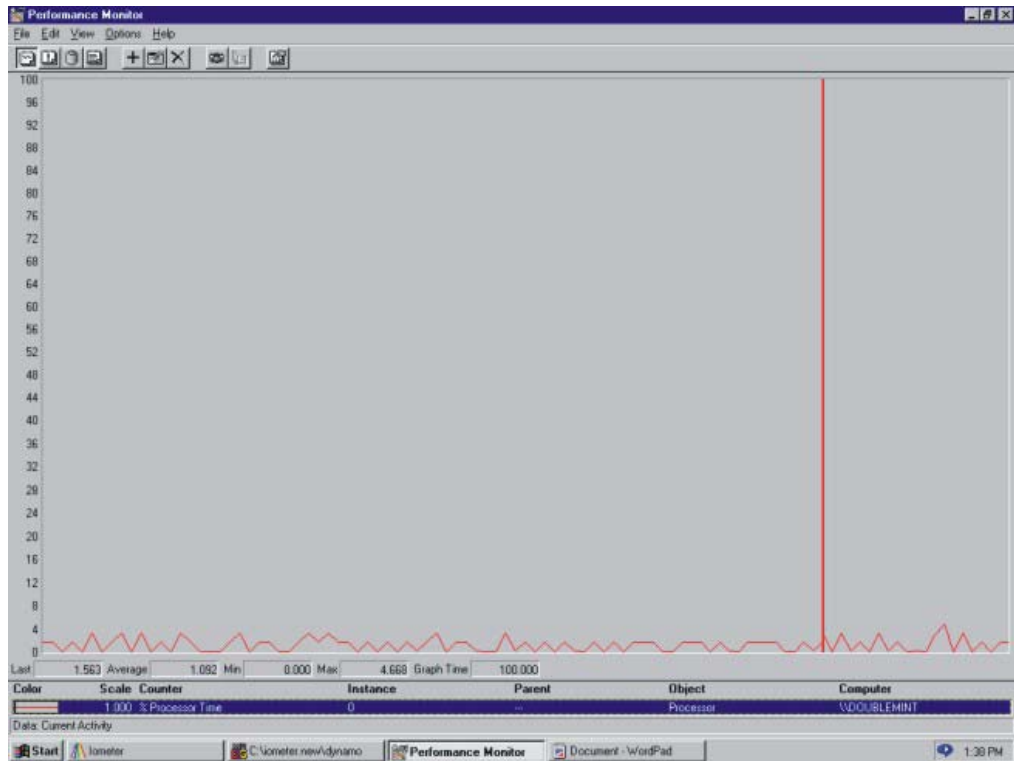


Figure 2: Processor Performance when Writing to Hard Disk

The first screen shot, Figure 1, shows the Processor: % Processor Time for a Windows NT system running Intel’s IOMeter performing 100% random writes to a hard disk drive. In this exhibit, you can see that the processor utilization averages around 1.8%. However, if you were to try to run additional applications on this system, the processor utilization would only marginally increase because the processor is tied up waiting on I/O from the hard disk drive. In this example, IOMeter shows that on average there were 150 writes per second (150 IOPS) to the disk drive.

Figure 2 shows the exact same system, exact same access specifications in IOMeter running against a Texas Memory Systems RamSan solid state disk. In this example, the processor averages 68% utilization. The IOMeter shows that 27,000 writes per second are going to the RamSan (27,000 IOPS). As it turns out, this IOPS number is a limitation of the host bus adapter used in the demonstration. Nonetheless, it is easy to observe how a solid state disk can improve processor utilization for a Windows based system.

In addition to processor indicators, Microsoft recommends looking at the Current Disk Queue Length and % Disk Time Counters to detect bottlenecks in the disk subsystem. If these values are consistently high consider moving files that are located on that disk to the solid state disk. A Disk Queue Length greater than 3 indicates a problem.

If you are using a RAID device, the % Disk Time counter can indicate a value greater than 100%. If it does, use the PhysicalDisk: Avg Disk Queue Length counter to determine the number of requests.

If you have more than one logical partition on the same disk, use the Logical Disk counters instead of the Physical Disk counters.

Using these tools, I/O wait time for a Windows based system can be observed.

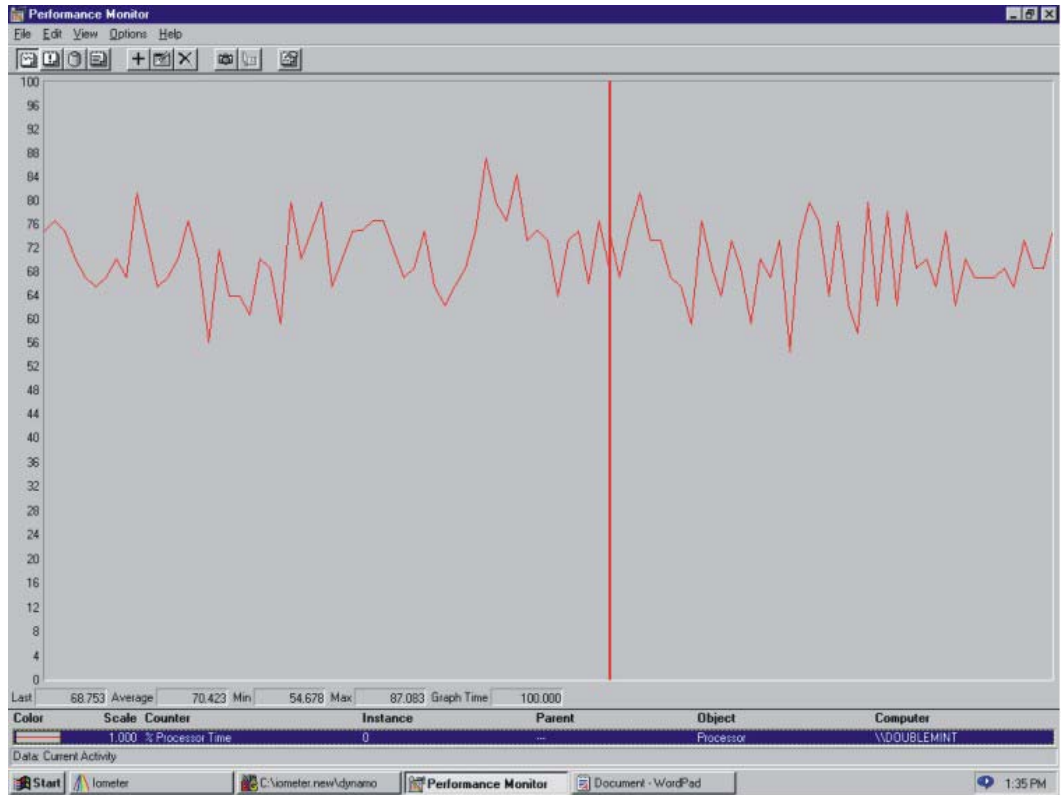


Figure 3: Processor Performance when Writing to Solid State Disk

## Section 7

---

# SQL Server Components That Should Be Moved to Solid State Disk

Once you determine that your system is experiencing I/O subsystem problems the next step is to determine which components of your SQL Server database are experiencing the highest I/O and in turn causing I/O wait time. The following database components should be looked at:

Entire Database. Some databases should have all of their files moved to solid state disk. These databases tend to have at least one of the following characteristics:

- **High concurrent access.** Databases that are being hit by a large number of concurrent users should consider storing all of their data on solid state disk. This will make sure that storage is not a bottleneck for the application and maximize the utilization of servers and networks. I/O wait time will be minimized and servers and bandwidth will be fully utilized.
- **Frequent random accesses to all tables.** For some databases, it is impossible to identify a subset of files that are frequently accessed. Many times these databases are effectively large indices.
- **Database performance is key to company profitability.** There is some subset of databases that help companies make more money, lose less money, or improve customer satisfaction if they process faster. Solid state disk can help make these companies more profitable.

Transaction Logs. Transaction logs are one of the most important factors in the write performance for SQL Server databases. Whenever a database write occurs, SQL Server creates a transaction log entry. For each SQL Server database there is at least one transaction log (\*.ldf). The operation is considered committed once the write to the transaction logs is complete.

The transaction logs are a source of constant I/O during database operation. It is important that the transaction logs are stored on the fastest possible disk. Writing a transaction log to a solid state disk is a natural way to improve overall database performance.

The Microsoft web site: [MSDN Home](#) > [MSDN Library](#) > [Microsoft SQL Server](#) > [Database Design](#) > [Physical Database Design](#) has the following to say about optimizing performance of transaction logs:

“Create the transaction log on a physically separate disk or RAID (redundant array of independent disks) device.”

Texas Memory Systems recommends that the separate disk be a RamSan-320 solid state disk.

Temporary Database (tempdb). Tempdb is used to support temporary data during certain SQL Server operations. The tables support complex queries, joins and index creations. Because tempdb supports many kinds of operations it is essential that they be located on the fastest possible storage.

When complex operations occur they will complete more quickly if the tempdb is moved to solid state disk. Because the I/O to the tempdb can be frequent, disk drives cannot easily handle them.

The Microsoft web site: [MSDN Home](#) > [MSDN Library](#) > [Microsoft SQL Server](#) > [Database Design](#) > [Physical Database Design](#) has the following to say about optimizing performance of tempdb:

“Place the **tempdb** database on a fast I/O subsystem to ensure good performance. Stripe the **tempdb** database across multiple disks for better performance. Use filegroups to place the **tempdb** database on disks different from those used by user databases.”

Indexes. An index is a data structure that speeds up access to database records. An index is usually created for each table in a database. These indexes are updated whenever records are added and when the identifying data for a record is modified. When a read occurs an index is consulted so that SQL Server can quickly get to the correct record. Furthermore, many concurrent users may read any index simultaneously. The activity to the disk drive is characterized by frequent, small, and random transactions. Under these conditions, disk drives are unable to keep up with demand and I/O wait time results.

By storing indexes on a solid state disk, performance of the entire application can be increased. For on-line transaction processing (OLTP) systems with a high number of concurrent users, this can result in faster database access. Because indexes can be recreated from the existing data, they have historically been a common SQL Server component to be moved to solid state disk.

The Microsoft web site: [MSDN Home](#) > [MSDN Library](#) > [Microsoft SQL Server](#) > [Database Design](#) > [Physical Database Design](#) > [Data Placement Using Filegroups](#) has the following to say about allocating indexes to their own Filegroup:

“By default, indexes are created on the same filegroup as the base table on which the index is created. However, it is possible to create nonclustered indexes on a filegroup other than the filegroup of the base table. By creating the index on a different filegroup, you can realize performance gains if the filegroups make use of different physical drives with their own controllers.”

In this case, Texas Memory Systems recommends moving the filegroups that contain the indexes to solid state disk.

Frequently Accessed Tables. Sometimes, just 5-10% of tables account for a large percentage of all database activity and thus I/O to storage. When a large number of users hit a table, they are likely going after different records and different attributes. As a result, the activity on that table is random. Disk drives are notoriously bad at servicing random requests for data. In fact, the peak performance of a disk drive drops as much as 95% when servicing random transactions. When a table experiences frequent access, transaction queues develop where other transactions are literally waiting on the disk to service the next request. These queues are another sign that the system is experiencing I/O wait time.

It makes sense to move the frequently accessed tables to a solid state disk. Solid state disk performance is not impacted if performance is random. Additionally, solid state disks by definition have faster access times than disk drives. Therefore, application performance can be improved up to 10x if frequently accessed tables are moved to solid state disk

SQL Server supports moving specific tables to specific filegroups. This feature allows you to easily move the most frequently accessed tables solid state disk.

## Section 8

---

# Identifying the Most Frequently Accessed SQL Server Tables

The SQL Server Profiler is a tool that allows database administrators to monitor events in SQL Server. The Profiler monitors SQL Server activities by tracing the selected events. Among the events that can be monitored using the SQL Server Profiler are events related to database objects. Database objects include: databases, tables, indexes, views, and stored procedures. The SQL Server Profiler specifically supports monitoring when a table is opened. Tables are opened during "Select," "Insert," and "Delete" statements. By using the Profiler, a database administrator will be able to monitor the number of times all tables are opened during the trace period. The results of the trace can be sent to a table for further analysis to determine which tables were most frequently opened during the trace period and thus which tables are the best candidate for placement onto a solid state disk.

Microsoft cautions that "Object" related traces could impose a significant overhead on database performance because of the frequency that objects are accessed. If you have a test environment that models production activity and transaction mixes, this may be the best place to use an SQL Server Profiler "Object" trace. Otherwise, make sure you do not use the Profiler during peak transaction periods unless you are willing to take the performance hit.

More information on SQL Server Profiler and database objects can be found at: [MSDN Home](#) > [MSDN Library](#) > [Microsoft SQL Server](#) > [Monitoring Server Performance and Activity](#) > [Monitoring with SQL Profiler](#) > [Monitoring with SQL Profiler Event Categories](#) > [Objects Event Category](#)

## Section 9

---

# Customer Case Study: BetOnSports.com

BetOnSports.com is the world's largest, legal and licensed sportsbook. With 12 years of experience, it has positioned itself as the highest quality and most reputable service in online sports betting and has been covered by the Wall Street Journal, CNN, and numerous trade review organizations.

BetOnSports Gaming prides itself on the quality of its customer service. To keep customers happy and business flowing, IT and telephone systems must be able to accept short bursts of massive amounts of traffic—especially as major sporting events take place. For them, every busy signal and web error is lost business.

BetOnSports Gaming contacted Texas Memory Systems during the height of the American football season. College bowl games were about to take place and the NFL playoffs were just around the corner. They needed hardware right away that could accelerate their database to accept as many online transactions as their customers could provide.

The primary application database was an OLTP system running SQL Server on a Dell PowerEdge system. The database accepted new transactions from customers and quickly moved them to central archive. Because of this, the database itself was quite small—small enough for the entire database to be stored on a single RamSan solid state disk.

“I was impressed with the easy installation,” said Tony Perez, IT Manager at BetOnSports.com. “I installed the RamSan and saw markable stability and faster delivery of data throughout the system architecture.”

BetOnSports met its solution goals by enabling all customers' transactions to be carried out by the database. During the busiest periods of activity, the RamSan ensured that no business was lost by slow storage. Additionally, users experienced faster response times from the website and fewer dropped or lost transaction attempts.

“Because of the quick time-to-implementation, we were ready for the big college bowl games,” said Perez. “The return on investment was immediate and undeniable.”