

Implementing EMC CLARiiON CX4 with Enterprise Flash Drives for Microsoft SQL Server 2008 Databases

Applied Technology

Abstract

This white paper examines the performance considerations of placing SQL Server 2008 databases on Enterprise Flash Drives (EFDs) versus conventional hard disk drives, as well as discusses the best practices for placing partial database containers on EFDs.

January 2010

Copyright © 2009, 2010 EMC Corporation. All rights reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com

All other trademarks used herein are the property of their respective owners.

Part Number h6080.2

Table of Contents

Executive summary	4
Introduction	4
Audience	5
Technology overview	5
Enterprise Flash Drives.....	5
SQL Server databases and EFDs	6
Database workloads that are the best fit for EFDs	6
Workload recommendations.....	7
Storage recommendations	7
Using Performance Monitor to identify I/O problems	8
Processor: %Processor Time.....	8
Physical Disk: Average Disk Queue Length.....	8
Physical Disk: Disk Bytes Per Second	8
Using SQL Server to identify I/O problems	9
Activity Monitor	9
Object Execution Statistics.....	9
Running a SQL query.....	12
Use Case #1: SQL Server OLTP workload comparison.....	12
Use Case #2: Short-stroked SQL Server OLTP workload	14
Use Case #3: Moving partial database to EFDs	14
Transaction logs on Flash? (or) not	15
SQL Server tempdb on Flash.....	15
SQL Server index on Flash	15
Use Case #4: DSS workloads on EFDs	16
Use Case #5: Hyper-V on EFDs	17
Impact of LUN cache settings.....	18
Conclusion	20
References	20

Executive summary

One of the major features that EMC introduced in the CLARiiON® CX4 series is the availability of Enterprise Flash Drives (EFDs). The revolutionary EMC® CLARiiON is the first midrange array with support for this emerging generation of drive technology. With this capability, EMC creates new ultra-performing “Tier 0” storage that removes the performance limitations of magnetic disk drives. By combining enterprise-class Flash drives optimized with EMC technology and advanced CLARiiON functionality, organizations now have a new tier of storage previously unavailable from any major midrange storage vendor.

EFDs dramatically increase performance of latency-sensitive applications. EFDs, also known as solid state drives (SSD), contain no moving parts and appear as standard Fibre Channel drives to existing CLARiiON management tools, allowing administrators to manage performance-critical applications without special processes, custom tools, or extra training. EFDs are ideally suited for applications with high transaction rates and those requiring the fastest possible retrieval and storage of data, such as currency exchange and electronic trading systems, or real-time data acquisition and processing. They also prove to be extremely good for highly read-intensive workloads like search engine databases. A CLARiiON CX4 with EFDs can deliver millisecond application response times and many times more I/O operations per second (IOPS) than traditional Fibre Channel hard disk drives. Even more importantly, for the same workload, EFDs can replace a large number of rotating drives and consume significantly less energy per IOPS, thereby significantly improving the upfront acquisition and ongoing maintenance costs.

This white paper examines some of the use cases and best practices for using EFDs with Microsoft SQL Server 2008® database workloads. It shows that EFDs deliver vastly increased performance to the database application when compared to traditional Fibre Channel drives, both in transaction rates per minute as well as transaction response time. It also discusses methods to identify the right database components to place on EFDs if the entire database cannot be placed on the EFDs.

Introduction

Database performance has long been constrained by the I/O capability of hard disk drives (HDD) and the performance of the HDD has been limited by intrinsic mechanical delays of head seek and rotational latency. In most cases, to overcome the limitations of HDDs, storage administrators were forced to use a larger number of drives and “short-stroke” them in order to achieve higher aggregate I/O. EFDs, however, have no moving parts and therefore no seek or rotational latency delays, which dramatically improves their ability to sustain a very high number of IOPS with very low overall response times.

Figure 1 shows the theoretical IOPS rates that can be sustained by traditional HDD based on average seek and latency times as compared to Flash drive technology. Over the past 25 years, the rotational speeds of HDDs have improved from 3,600 rpm to 15,000 rpm, yielding only four times the improvement in IOPS while other computer technologies, like CPU speeds, saw double-digit growth. Flash drive technology represents a significant leap in performance and can sustain over many times the IOPS of traditional HDD technology.

Enterprise Flash Drives Vs Traditional Disk Drives

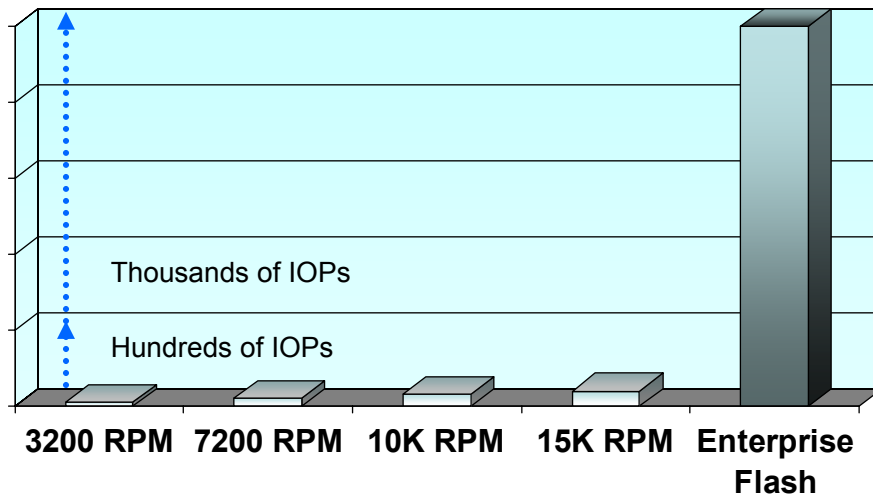


Figure 1. Relative IOPS of various drive technologies

The introduction of the industry's first EFDs into a midrange storage array enables EMC's CLARiiON disk arrays to meet the growing demand for higher transaction rates and faster response times. Companies with stringent latency requirements no longer have to purchase large numbers of the fastest Fibre Channel disk drives and only utilize a small portion of their capacity (known as short-stroking) to satisfy the IOPS performance requirements of very demanding random workloads.

Relational databases are often at the core of business applications. Increasing their performance, while keeping storage power consumption and footprint to a minimum, significantly reduces the total cost of ownership (TCO) and helps to alleviate growing data center constraints. The deployment of Tier 0 EFDs together with slower tiers such as Fibre Channel and SATA drives enables customers to structure the application data layout where each tier of storage meets the I/O demands of the application data it hosts.

Audience

This white paper is intended for SQL Server database administrators, storage architects, customers, and EMC field personnel who want to understand how the implementation of EFDs in SQL Server database environments can improve the performance of business applications.

Technology overview

Enterprise Flash Drives

The enterprise-class EMC EFDs supported by CLARiiON CX4 are constructed with nonvolatile semiconductor NAND Flash memory. Due to their solid state design, these drives are especially well suited for latency-sensitive applications that require consistently low read/write response times. EFDs are packaged in a standard 3.5-inch disk drive form factor used by existing CLARiiON disk drive array enclosures, making for simple integration with existing infrastructure.

Since there are no moving parts, EFDs use far less energy when compared to drives with rotating media and weigh less as well. In addition to consuming less energy, they produce less heat, which also translates to energy savings in terms of cooling costs. The real energy savings becomes evident when comparing

traditional disk drives to EFDs based on the comparable numbers of IOPS delivered per drive. This is because many more traditional drives are needed to deliver the performance level of a single EFD, so the power savings can be significant for environments with short-stroked FC drives.

EFDs are optimized for high levels of concurrency. This means that the workloads that push the EFDs hard enough will show significant performance gains. One way to increase the concurrency level is to consolidate the work done by many spinning drives into a small number of EFDs. Another way to achieve concurrency is to allocate LUNs for numerous I/O-intensive applications and databases on the same set of EFDs. The guidance of isolating spindles for certain database files does not apply to EFDs although paying attention to the I/O size and I/O type (read vs. write) can be important.

Most real world database workloads are not purely small I/O and read only in nature but typically have a mix of write I/O along with larger I/O sizes, which can benefit from EFDs. In general, EFDs provide the greatest impact when doing small read I/Os.

In the overall system architecture, EFDs only represents the I/O subsystem and are only one component that can affect the performance achieved by the system. Replacing traditional spinning disks with EFDs is the simplest way to eliminate the I/O bottleneck of the system provided there are no other bottlenecks such as SAN link speed. Optimizations to other components of the system may be necessary to fully realize the benefits of EFDs.

The EFDs were thoughtfully designed to seamlessly integrate with advanced features and capabilities of the CLARiiON platform. These include use of tools like Navisphere Analyzer, Navisphere Quality of Service Manager, and Virtual LUN technology.

CLARiiON storage systems currently support EFDs in three sizes; 73 GB, 200 GB, and 400 GB. All these drive sizes offer slightly different application performance characteristics depending on the nature of the workload. The 400 GB drives are specifically optimized for capacity requirements, while the 73 GB and 200 GB drives are optimized for increased write performance. The customer can choose any of these drives depending on their space and performance requirements.

SQL Server databases and EFDs

Database workloads that are the best fit for EFDs

It is very important to understand the load profile of an application before putting it on the EFDs. Most databases have different workload profiles during different times of the day so monitoring should be performed over a long period of time to ensure correct and accurate data points were obtained. The EFDs are suitable for highly read-intensive and extremely latency-sensitive applications and using these drives for solving the wrong problem may not justify their moderately high cost. It is important to understand the following storage-related terminology before identifying whether the EFDs are suitable for certain workloads.

Read hit: A read request from a database host can be served by storage immediately if it already exists in storage cache because of a recent read or write or due to prefetch. A read serviced from the storage cache without causing disk access is called a read hit. If the requested data is not available in storage cache, the CLARiiON must retrieve it from disk; this is referred to as a read miss.

Write cache: Most of the storage systems have big write side cache and all write IOPS from the host are generally written to cache and, unless cache is saturated, incur no delay due to physical disk access. CLARiiON storage arrays have write cache, which supports enabling and disabling write cache at the LUN level, if needed.

Short-stroked drives: Some extremely latency-sensitive applications use this technique on regular Fibre Channel drives to obtain low latencies. This is a technique where data is laid out on many partially populated disks in order to reduce the spindle head movement to provide high IOPS at a very low latency.

Workload recommendations

Workloads with low CLARiiON cache read-hit rates that exhibit random I/O patterns, with small I/O requests of up to 16 KB, and require high transaction throughput will benefit most from the low latency of EFDs.

Database and application managers can easily point to mission-critical applications that, if made much faster, will directly affect an increase in business revenue and productivity. In a similar way, the storage managers can point to these same applications, since, for performance planning purposes, they use a large number of “short-stroked” drives. When such applications are identified, EFDs can provide two very important benefits.

- A single EFD can replace many short-stroked drives by its ability to provide a very high transaction rate (IOPS). This reduces the total number of drives needed for the application, increases power saving by not having to keep many spinning disks, and eventually means reduced floor space in the data center as well.
- EFDs provide very low latency, so applications, where predictable low response time is critical and not all the data can be kept at the host or CLARiiON cache, will highly benefit from using such drives. EFD transfer rate is extremely high and data is served much faster than the best response time that can be achieved with a large number of short-stroked hard drives.

Storage recommendations

LUN cache settings

Due to the extremely high performance of EFDs, cache settings for EFDs do not follow traditional guidelines as explained below:

- EFDs are extremely fast, so when the read cache is enabled for the LUNs residing on them, the read cache lookup for each read request adds a significantly higher overhead as compared to FC drives, in an application profile that is not expected to get many read cache hits at any rate. Thus it can be faster to disable read cache to directly read the block from the EFD.
- In a real world scenario, where the CX4 is being shared by several applications and, especially, deployed with slower SATA drives, the write cache may become fully saturated, placing the EFDs in a force flush situation, which adds latency. In these situations, it is better to write the block directly to EFDs than to write the write cache of the storage system.

The cache setting for the storage system should be configured as depicted in Table 1.

Table 1. Recommended default cache settings

	SP read cache	SP write cache	LUN read cache	LUN write cache
FC Drive	ON	ON	ON	ON
EFD	ON	ON	OFF	OFF

These recommendations are by no means the absolute correct way to configure EFDs in a CLARiiON storage system. The recommendations were established to help ensure optimal performance for complex deployments with various drive types and mixed workload environments. For rare deployments with only EFDs configured serving a limited number of applications, enabling write cache provides an instant boost in performance for workloads that have moderate write activity. Even for mixed drive deployments, storage administrators and DBAs can experiment with enabling write cache on EFD LUNs to maximize performance. Of course, careful testing and benchmarking must be completed to ensure that other applications serviced by the storage system are not adversely affected by such changes.

In general, the following I/O patterns will usually experience significant performance gains with write cache enabled:

1. Write I/O ratio well above read I/O

-
2. Larger I/O write sizes for operations such as bulk inserts or backups and restores
 3. Sequential writes (logs, initial tablespace allocation, extending tables, and so on)

The I/O patterns listed here benefit greatly from the use of write cache. Note that cache activation can be scripted. In a case where the write cache is normally disabled for EFD LUNs, write cache can be activated for certain operational procedures that benefit. For example, a database that resides on EFDs provides more than adequate performance with write cache disabled during normal operation, but it suddenly requires the insertion of a large data set or restoration from an error. Enabling write cache for these specific events can dramatically reduce the data load or recovery window, allowing the database to return to normal operation in a shorter amount of time,

These guidelines apply to the 73 GB EFDs, 200 GB EFDs, and 400 GB EFDs. Note that the 400 GB EFDs can trail the 73 GB and 200 GB EFDs somewhat in write-intensive applications. Careful tuning and tailoring of the cache settings for the specific needs of each environment can improve upon the already phenomenal performance characteristics exhibited by EFDs.

Back-end bus bandwidth considerations

Because of the high level of bandwidth delivered by a small number of EFD drives, the storage administrator must pay special attention to which back-end bus of the DAE tray that houses the EFDs is connected to. When possible, it is best to span EFD RAID groups across as many back-end buses as possible to fully utilize all available resources. With regard to an occasional surge in bandwidth usage during operations such as a backup or restore of the database, it is recommended to only place the EFDs on a bus with other drives with a usage pattern that can accommodate the large burst in bandwidth (MB/s). If special care is not taken into the placement of EFDs on the back-end bus, the bus can become saturated and the maximum performance benefits from EFDs cannot be realized.

Using Performance Monitor to identify I/O problems

The Windows operating system comes with the Performance Monitor tool that collects a large number of system metrics to help identify performance problems. This section describes statistics that can help identify a system with the I/O subsystems causing performance problems.

Processor: %Processor Time

The “Processor: % Processor Time” statistic shows how busy the system processor is. Although this statistic does not measure the I/O of the disk subsystem, it can be used to determine if there are I/O problems. For instance, if your system is busy performing transactions yet the “Processor: % Processor Time” is well below 100 percent, it is very likely that the system has I/O wait problems. A system with enough disk resources and a high volume of transactions will normally show a high value for the “Processor: % Processor Time.”

Physical Disk: Average Disk Queue Length

Microsoft recommends that the “Physical Disk: Average Disk Queue Length” should be a value of 2 per physical disk. In a RAID configuration, make sure the measurement is recalculated to reflect the actual number of physical spindles. The contents of a disk with a consistently high “Physical Disk: Average Disk Queue Length” reading is a promising target for migration to EFDs.

Physical Disk: Disk Bytes Per Second

The “Physical Disk: Disk Bytes Per Second” counter helps to visualize the level of performance delivered by the drives. If the “Disk Bytes Per Second” statistic remains steady and an increase in the “Average Disk Queue Length” is observed, there is a high chance that an I/O bottleneck is present. The contents of that drive may provide substantially better performance on EFDs.

Using SQL Server to identify I/O problems

SQL Server offers several ways to enumerate the amount of I/O on the different tables. This section describes several options from within SQL Server Management Studio to obtain I/O statistics and identify objects that can benefit from migration to EFDs.

Activity Monitor

From within SQL Server Management Studio, the Activity Monitor can be invoked by right-clicking the instance of interest from within Object Explorer and selecting Activity Monitor. Once open, the user has the option to expand and collapse Overview, Processes, Resource Waits, Data File I/O, and Recent Expensive Queries sections. The data displayed in each of the sections can be configured to be updated frequently and provides a “real time status” of the database instance. Figure 2 shows the Activity Monitor with the Overview expanded.

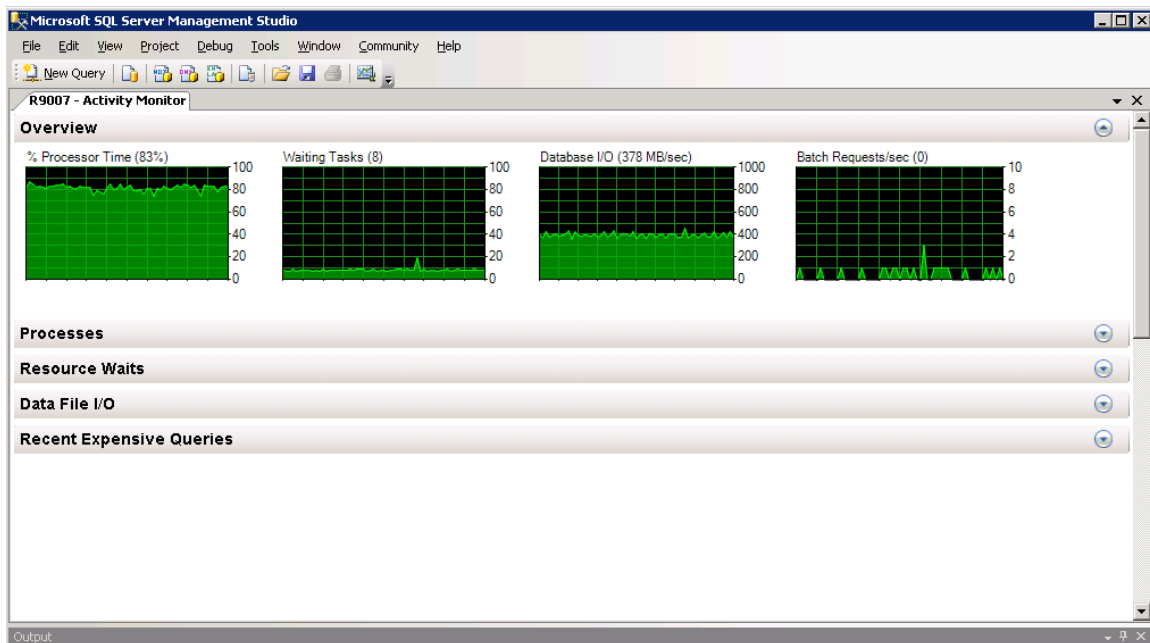


Figure 2. Activity Monitor

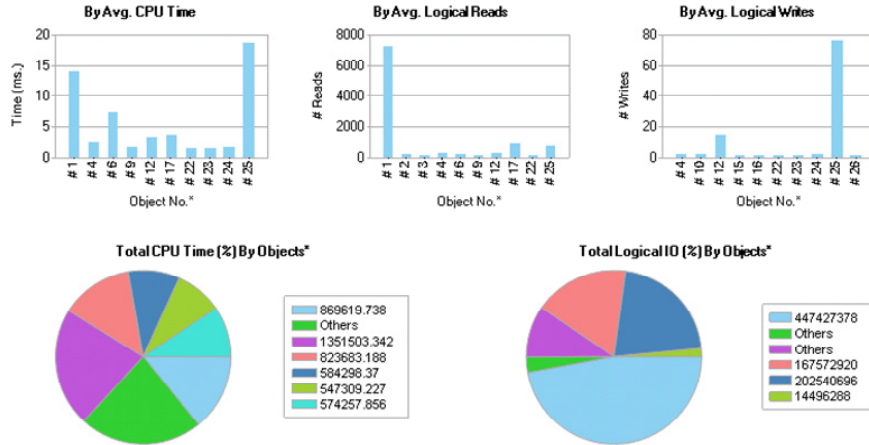
The various sections of the page provide useful information about SQL Server performance although the Data File I/O section can be used to quickly view which files have response times that are out of specifications and are busiest in terms of reads and writes. If multiple tables are stored within the files, the Recent Expensive Queries section may provide deeper insight in identifying the busiest tables.

Object Execution Statistics

In addition to Activity Monitor, SQL Server Management Studio offers many standard reports with valuable information although the Object Execution Statistics standard report is the key report that can be used to identify objects that can be moved to EFDs. At the top of the report, one can easily visualize which objects are CPU-intensive, read-intensive, and write-intensive.

This report provides detailed historical execution data for all currently cached plans for objects within the Database. This execution data is aggregated over the time during which the plan has been in the cache.

Top Executable Objects



* See column 'Object No.' from table 'All Objects' for value of 'Object No.' It is the unique number assigned to each Object.

All Executable Objects

Shows statement wise execution statistics for all the executable objects.

Click + to expand

Object No.*	Object Name	Object Type	Avg. CPU Time (ms.)	Total CPU Time (%)	# Avg. Logical Reads	# Avg. Logical Writes	# Avg. Logical IO	Total Logical IO (%)
-------------	-------------	-------------	---------------------	--------------------	----------------------	-----------------------	-------------------	----------------------

Figure 3. Object Execution Statistics Report

Once an object of interest is identified, additional details from the report can be obtained by moving to the corresponding Object Number to view the various statistics. Table 2 describes the various top level statistics available from the report.

Table 2. Object Execution Statistics Report description¹

Column	Description
Object Number	A unique number within the report assigned to each object such as a stored procedure, that has a cached plan
Object Name	The “friendly” name of the object
Object Type	The type of the object within SQL Server
Avg. CPU Time (ms.)	The average CPU time taken by this object
Total CPU Time (ms.)	The total CPU time taken by this object
Total CPU Time (%)	The total percentage of the CPU time taken by this object
# Avg. Logical Reads	The number of average logical read operations taken by this object
# Avg. Logical Writes	The number of average logical write operations taken by this object
# Total Logical Writes	The number of total logical read operations taken by this object
# Avg. Logical IO	The number of average I/O operations taken by this object. Note that a SAN or other abstracted I/O subsystem can affect this number dramatically
# Total Logical IO	The number of total I/O operations taken by this object. Note that a SAN or other abstracted I/O subsystem can affect this number dramatically
Total Logical IO (%)	The total percentage of I/O operations taken by this object. Note that a SAN or other abstracted I/O subsystem can affect this number dramatically

By clicking on the “+” sign on the “# Avg Logical Reads” column shown by the red arrow in Figure 3, the report provides additional details as described in Table 3. As depicted in Figure 4, the expanded view

¹ Taken from <http://blogs.msdn.com/buckwoody/archive/2008/02/04/sql-server-management-studio-standard-reports-object-execution-statistics-database.aspx>

provides statistics on the total physical reads for that particular object. In this case, the high number of physical reads indicates that the database files accessed by this object require a lot of disk I/O and are a good candidate for EFDs.

Obj. CPU Time (ms.)	Total CPU Time (%)	# Avg. Logical Reads	# Total Logical Reads				# Avg. Logical Writes	# Avg. Logical IO	Total Logical IO (%)
	14.15	7,221.81	447,427,378				0.00	7,221.81	47.09
PU Time		# Avg. Logical Reads	# Total Physical Reads	# Last Logical Reads	# Min. Logical Reads	# Max. Logical Reads	# Avg. Logical Writes	# Avg. Logical IO	
		7,221.81	447,427,378	2,592	4	14,869	0.00	7,221.81	

I/O to drives




Figure 4. Expanded view of the Object Execution Statistics Report

Table 3. Object Execution Statistics Report description of expanded statistics²

Column	Description
SQL Statement	Part of the SQL Statement run against the object
# Executions (With Last Plan)	How many times this statement has executed against this object, within this plan
# Plans Generated	The number of plans that had to be generated for this SQL Statement
Avg. CPU Time (ms.)	The average amount of CPU time taken by this SQL Statement
Total CPU Time (ms.)	The total amount of CPU time taken by this SQL Statement
Last CPU Time (ms.)	The last amount of CPU time taken by this SQL Statement
Min. CPU Time (ms.)	The minimum amount of CPU time taken by this SQL Statement
Last CPU Time (ms.)	The last amount of CPU time taken by this SQL Statement
Min. CPU Time (ms.)	The minimum amount of CPU time taken by this SQL Statement
Max. CPU Time (ms.)	The maximum amount of CPU time taken by this SQL Statement
# Avg. Logical Reads	The average number of logical read operations taken by this SQL Statement
# Total Physical Reads	The total number of physical read operations taken by this SQL Statement. Note that a SAN can drastically affect this number
# Last Logical Reads	The last number of logical read operations taken by this SQL Statement
# Min. Logical Reads	The minimum number of logical read operations taken by this SQL Statement
# Avg. Logical Writes	The average number of logical write operations taken by this SQL Statement

² Taken from <http://blogs.msdn.com/buckwoody/archive/2008/02/04/sql-server-management-studio-standard-reports-object-execution-statistics-database.aspx>

# Total Logical Writes	The total number of logical write operations taken by this SQL Statement
# Last Logical Writes	The last number of logical write operations taken by this SQL Statement
# Min. Logical Writes	The minimum number of logical write operations taken by this SQL Statement
# Max. Logical Writes	The maximum number of logical write operations taken by this SQL Statement
# Avg. Logical IO	The average number of logical I/O operations taken by this SQL Statement
# Total Logical IO	The total number of logical I/O operations taken by this SQL Statement
# Last Logical IO	The last number of logical I/O operations taken by this SQL Statement
# Min. Logical IO	The minimum number of logical I/O operations taken by this SQL Statement
# Max. Logical IO	The maximum number of logical I/O operations taken by this SQL Statement

Running a SQL query

There are various options from within SQL Server that allows one to retrieve the I/O statistics such as `sp_monitor` and `fn_virtualfilestats` to the `sys.dm_io_virtual_file_stats()` dynamic management view (DMV) introduced in SQL Server 2005. From the <http://technet.microsoft.com/en-us/magazine/cc135869.aspx> site under August downloads, the code for “SQL Q&A: Finding Locks, Large Queries, I/O Statistics, and More” provides for an easy way to gather I/O statistics about the database files and filegroups. Running the code results in the output displayed in Figure 5. Special attention should be given to `NumberReads`, `NumberWrites`, `IOStallMS`, and `ISStallPCT`. Files with a large number of reads and writes are prime candidates for EFDs. In addition, the `IOStallMS` and `ISStallPCT` columns show the number of milliseconds (cumulative) waited for the I/O to be read or written to the file and the percentage that an I/O wait had to occur. Files that have a lot of reads and writes with a large `ISStallPCT` value should be the first candidates to be moved to EFDs.

The counters for `fn_virtualfilestats` are reset each time the database instance is restarted so the easiest way to start monitoring the activity of the database files is right after the database instance is restarted. Of course, with a production database, it is not always feasible to bring the database instance down so some calculations with the difference between the `fn_virtualfilestats` readings must be performed to determine the busiest files during the monitoring period.

	dbname	FileName	FilePath	FileSize	NumberReads	NumberWrites	BytesRead	BytesWritten	PerCentReads	PerCentWrites	IOStallMS	ISStallPCT
40	tpce	Customer1	C:\TPCEM...	1109760	713961	263524	8949230...	16947421184	73	26	10813011	11
41	tpce	Customer7	C:\TPCEM...	1108480	714038	221082	8914622...	16298508288	76	23	8022028	8
42	tpce	Customer5	C:\TPCEM...	1108480	714663	238489	8955555...	16597868544	74	25	13567930	14
43	tpce	Customer2	C:\TPCEM...	1108480	717312	267630	8969967...	16976166912	72	27	13169806	13
44	tpce	Broker5	C:\TPCEM...	4381440	9922061	1233872	5072995...	89450987520	88	11	177051...	15
45	tpce	Broker2	C:\TPCEM...	4380160	9998102	1231210	5068539...	89440804864	89	10	185061...	16
46	tpce	Broker6	C:\TPCEM...	4381440	10014185	1239434	5069833...	89377423360	88	11	192127...	17
47	tpce	Broker1	C:\TPCEM...	4381440	10025210	1229382	5067213...	89486860288	89	10	223773...	19
48	tpce	Broker4	C:\TPCEM...	4381440	10063300	1234618	5074104...	89429925888	89	10	186767...	16
49	tpce	Broker3	C:\TPCEM...	4380160	10089614	1235968	5074533...	89425051648	89	10	167258...	14
50	tpce	Broker7	C:\TPCEM...	4381440	10132417	1236874	5074632...	89342091264	89	10	124404...	10
51	tpce	Broker8	C:\TPCEM...	4381440	10138471	1236782	5077758...	89394462720	89	10	131270...	11

Figure 5. Output of the `fn_virtualfilestats` query used to identify read- and write-intensive database files

Use Case #1: SQL Server OLTP workload comparison

EFDs are extremely good for highly read-intensive applications and are moderately fast for applications with a read/write mix. In either case they provide good TCO because of the other benefits they bring (such as a huge reduction in energy costs and significant improvements in latency) along with the performance benefits. A typical SQL Server OLTP application will have some write activity because of DML operations like updates and inserts.

To compare the performance of EFDs with hard disk drives in OLTP environments, two identical SQL Server 2008 databases using NTFS were deployed each onto a single RAID 5 (5+1) group. This can be considered an “apples to apples” comparison because the first RAID group was created using 73 GB EFDs while the second RAID group was created using 300 GB 15k rpm Fibre Channel drives. The number of

LUNs, number of file groups, and file layout were identical for both configurations. An OLTP workload with a 85/15 read/write ratio was used to generate identical workloads against both databases, which used 64-bit Windows Server 2008 as the operating system. The LUNs created from the FC drives had both read and write cache enabled while the LUNs created from EFDs had both read and write cache disabled.

The results are shown in Figure 6.

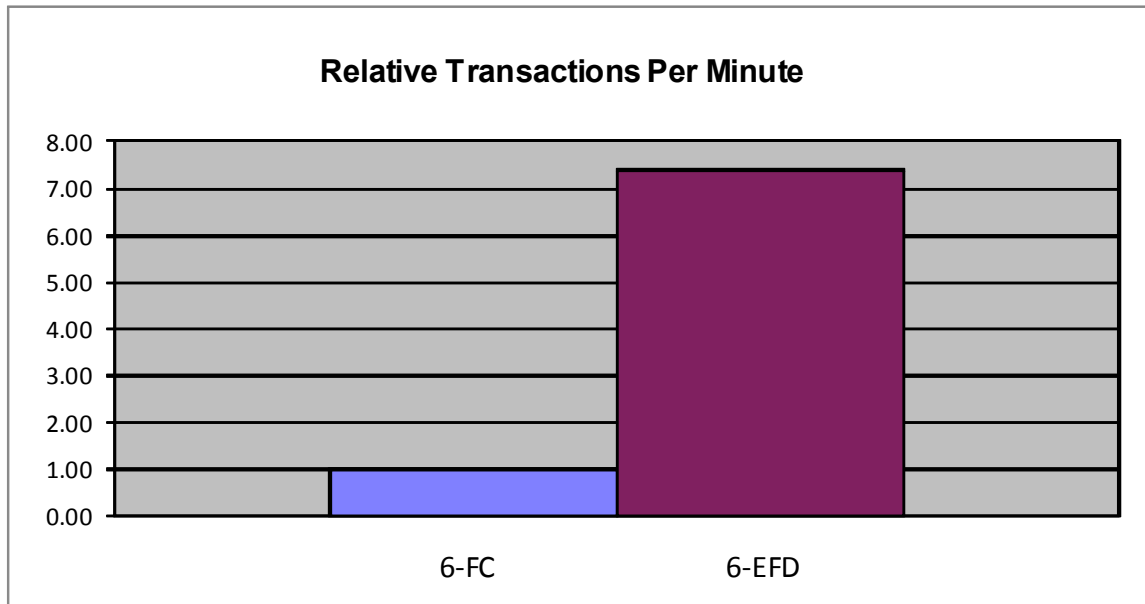


Figure 6. Transactions per minute comparison of 6 Flash drives vs. 6 FC drives

As Figure 6 shows, the EFDs gained roughly a seven-fold improvement in sustained TPM over traditional FC drives, which had the additional benefit of array read and write cache enabled. In addition it is interesting to note that the response time on the Flash drive was also one-seventh of that observed on traditional drives.

This use case points to an important fact that changing only the disks in the configuration can result in a significant improvement in IOPS and response time. However, with any tuning effort it is extremely important to understand the real issue with the application. Improving just the I/O by a factor of “X” may not always result in a multi-fold improvement in overall system response. Often times, the storage bottleneck has been simply removed to uncover another bottleneck somewhere else. The overall improvement in the application will be related to how badly it was impacted by the storage bottleneck that was just removed by moving to EFDs. In this case, the CPU of the server was running at 100 percent during the test so getting a larger server may yield better than the seven-fold increase observed.

Use Case #2: Short-stroked SQL Server OLTP workload

While Use Case #1 clearly shows the performance benefits of EFDs, oftentimes DBAs and storage architects may not replace Fibre Channel drives with an equal number of EFDs. For most mission-critical databases, the disk used to store database files is usually short stroked to sustain more IOPS with lower response times. In Use Case #2, the same set of data is now spread across 75 FC drives in comparison to using only six EFDs. The same OLTP workload is driven against the LUNs created from the FC drives, which have read and write cache enabled, and the LUNs created from EFDs, which have read and write cache disabled.

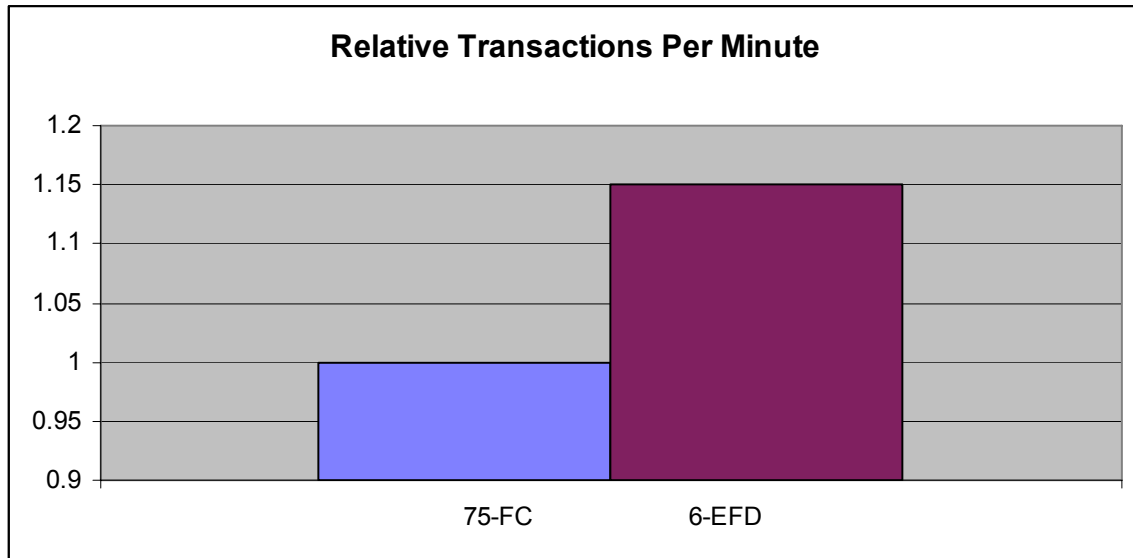


Figure 7. Relative transactions per minute comparison of 6 Flash drives vs. 75 HDDs

The EFDs show a relative improvement of 15 percent in overall transactions and 25 percent improvement in the latency. The overall efficiency factor is calculated by the following formula:

$$\text{Per Disk Efficiency} = \text{Disk reduction factor} * \text{Performance improvement factor}$$

$$\text{Per Disk Efficiency of above test} = (75 / 6) * 1.15 \approx 14 \text{ times}$$

This is a huge improvement in performance given the small investment made in a few EFDs. What gets ignored here is the improvement in the latency, which also results in an improved user experience. Also ignored is the fact that with the six EFDs, the server exhibited 100 percent processor utilization. The EFD results could very well be much larger but the bottleneck now seems to be the processor in the server. With the 75-FC case, the server utilization was already above the 80 percent level so it can be assumed the EFDs helped to eliminate the I/O subsystem as the bottleneck for maximum performance.

Use Case #3: Moving partial database to EFDs

The temptation may be to move the entire database to EFDs for the sake of simplicity and the intent of yielding the maximum benefit from EFDs; however it may not be economically viable to move the entire database because of constraints like the size of the database. This section discusses which parts of the database should be moved to EFDs to achieve the maximum benefit from such a huge investment. Keep in mind that EFDs are highly optimized for high concurrency so any capacity savings as a result of not moving nonessential components to EFD can be reserved for other I/O-intensive applications.

Transaction logs on Flash? (or) not

One of the first components that many may target for migration to EFDs is the SQL Server transaction logs. Since data must first be written from the data cache to the transaction log before writing of the actual data when a transaction is committed, one would assume that having very fast performance for the transaction log will automatically boost the overall performance of the database. Testing has shown that moving transaction logs to EFDs results in a slight performance degradation. It is better to leave them on the write cache backed Fibre Channel drives rather than moving them on to EFDs, thereby using EFDs for other I/O-intensive parts of the database like indexes or data.

In addition, since all database changes are recorded in the transaction logs, its size can grow large fairly quickly for large transactions and instances where backups and checkpoints in log truncate mode are not performed frequently. Because database performance comes to a grinding halt when the transaction log is full, the space allocated for transaction logs tends to be quite large, which can consume valuable capacity of the EFDs.

SQL Server tempdb on Flash

SQL Server uses this space mainly for data aggregations, complex queries, joins, and index creations. SQL Server typically does large sequential I/Os against the tempdb and Microsoft recommends having .25 to 1 temporary database file per core, which may or may not reside on the same physical spindles. On high-end servers with multiple quad-core processors, I/O access to tempdb files can be largely random in nature when multiple tempdb files reside on the same set of spindles. Even though EFDs do not provide as much benefit for large random I/O as they provide to small random operations, still they are far ahead of what regular rotation Fibre Channel drives can deliver. Depending on the availability of space on EFDs and the types of transaction mix, SQL Server applications can benefit from moving the tempdb to EFDs. These files should only be moved to EFDs after all the I/O-intensive parts have already been moved to EFDs.

SQL Server index on Flash

SQL Server uses clustered and nonclustered indexes to help speed up access to database records. SQL Server normally updates the indexes when adding records or modifying data. For OLTP workloads, the large numbers of transactions will frequently reference the index to find the corresponding data in the tables. Because of the large number of I/O requests to the index, the I/O profile is usually small and random in nature. This makes indexes an ideal candidate for EFDs. Because clustered indexes modify the underlying storage structure, moving the clustered indexes equate to move the data as well. Therefore only the nonclustered indexes should be moved to EFDs if storage capacity permits.

The data in Figure 8 shows performance improvements that partial database movements to EFDs can achieve. Various components from a 600 GB OLTP database deployed on 45 x 15k rpm Fibre Channel spindles were moved in isolation one after another to an NTFS file system created on six EFDs. Again LUNs created from the FC drives had read and write cache enabled while LUNs created from EFDs had read and write cache disabled.

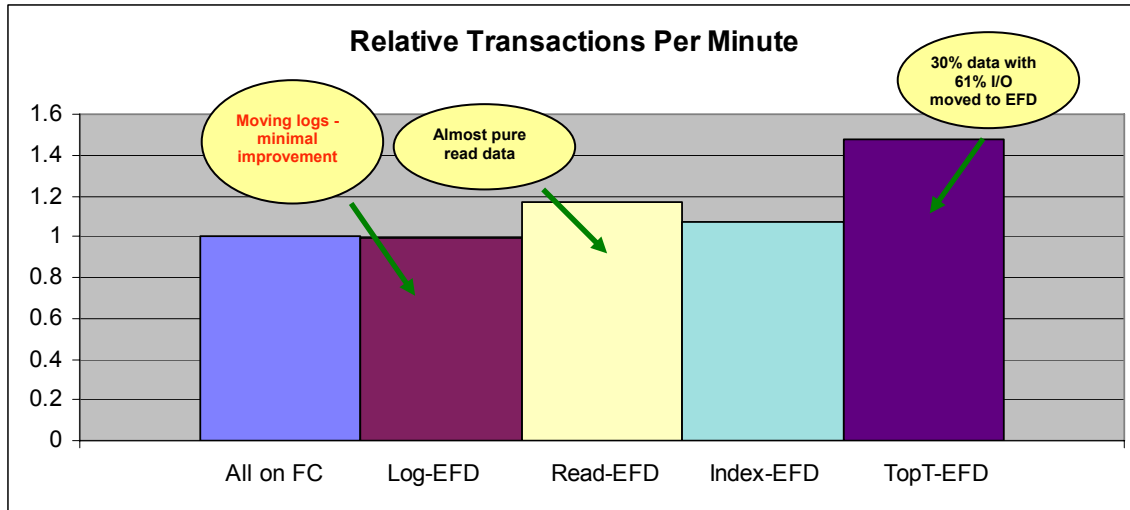


Figure 8. Performance benefits from moving a partial database

The data in Figure 8 shows that transaction logs files are not ideal candidates for EFDs. There was a very slight degradation in performance when moving logs on to EFDs. The logs can safely be left on their own set of Fibre Channel drives and EFDs can be used for moving other latency-sensitive parts of the database to maximize the benefit.

Depending on the database in question, tables might exist that require mostly reads from the various transactions that are executed. In this test, a read-intensive table was moved off to EFDs and a 17 percent gain was observed when only a small percentage of the data was moved to EFDs (less than 3 percent). Although not shown, moving only portions of a busy table onto EFDs can also provide additional benefits.

Moving the nonclustered indexes to EFDs resulted in a moderate increase in the observed transaction rate. The indexes constituted about 12 percent of the database size and received about 8 percent of the I/Os and provided about a 7 percent gain in overall performance. If a database application makes heavy use of nonclustered indexes that are small in relation to the overall database size, nonclustered indexes should be considered for EFDs for increased performance.

The final scenario involved moving the busiest table, which accounted for 61 percent of the total I/O, and resulted in about 48 percent improvement in the observed TPM. It is important to note that a total of 30 percent of the data was moved to achieve the 48 percent increase. Therefore it is essential to use the tools described in this paper to find the busy table(s) to migrate to EFDs for the best return on investment (ROI).

Use Case #4: DSS workloads on EFDs

EFDs also provide better performance over sequential workloads with larger I/O sizes such as those seen in DSS deployments. The nature of DSS deployments is generally sequential in nature because of table or index scans and bulk insert operations. With read ahead engaged at the database engine level, I/O sizes can be as large as 256 KB and reach 1024 KB for the Enterprise Edition. Bulk load operations can write I/O sizes up to 128 KB.

Even with multiple concurrent DSS queries running, the SQL Server Enterprise Edition database engine employs an advanced scanning mechanism (the merry-go-round scan) that allows multiple tasks to share a full table scan, creating a more sequential I/O pattern. In essence, if the database engine detects that a second query plan is performing a full table scan on a table that is already being scanned by another query, the database engine joins the two scans together, eliminating the need to transfer the same set of data multiple times and reducing randomness in the I/O pattern. This does not mean that a true sequential I/O

pattern will always be seen at the disk level since multiple tables can reside within a single data file. EFDs are the undisputed performance leaders for OLTP workloads, but testing has shown that EFDs provide superior performance for DSS type transactions as well. Although the performance gain factor for DSS is not as high when compared to OLTP, the performance difference is still significant. Figure 9 shows completion times for three DSS type queries on the different drive types.

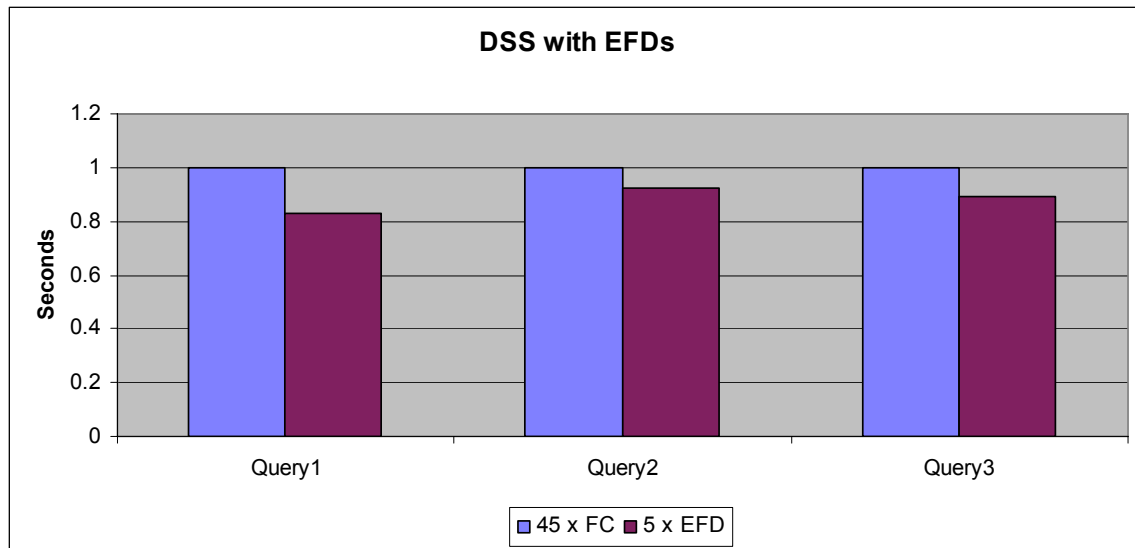


Figure 9. DSS query completion time

For each query type, the database deployed on EFDs completed 7 to 15 percent faster than a similar database deployed on FC drives. The results show that EFDs can be effectively used for DSS queries with short completion windows.

Use Case #5: Hyper-V on EFDs

Virtualization is revolutionizing the technology landscape by allowing companies to consolidate a large number of underutilized servers into a smaller set, increasing individual server utilization while reducing overall power consumption requirements. Although the number of physical servers has decreased, the provisioning of LUNs at the storage level may not necessarily be simplified if servers hosting high performance databases are being virtualized. The temptation with virtualization is to create multiple virtual hard disks on a single LUN. The virtualization of high performing SQL Server database instances will still require careful layout of the data. EFDs can alleviate this situation and make deployment of virtual machines more attractive since the performance capabilities of the EFDs are well suited for high levels of concurrency, enabling the sharing of spindles across multiple virtual machines. Figure 10 shows the performance achieved with a Hyper-V virtual machine deployed on advanced EFDs and traditional FC drives.

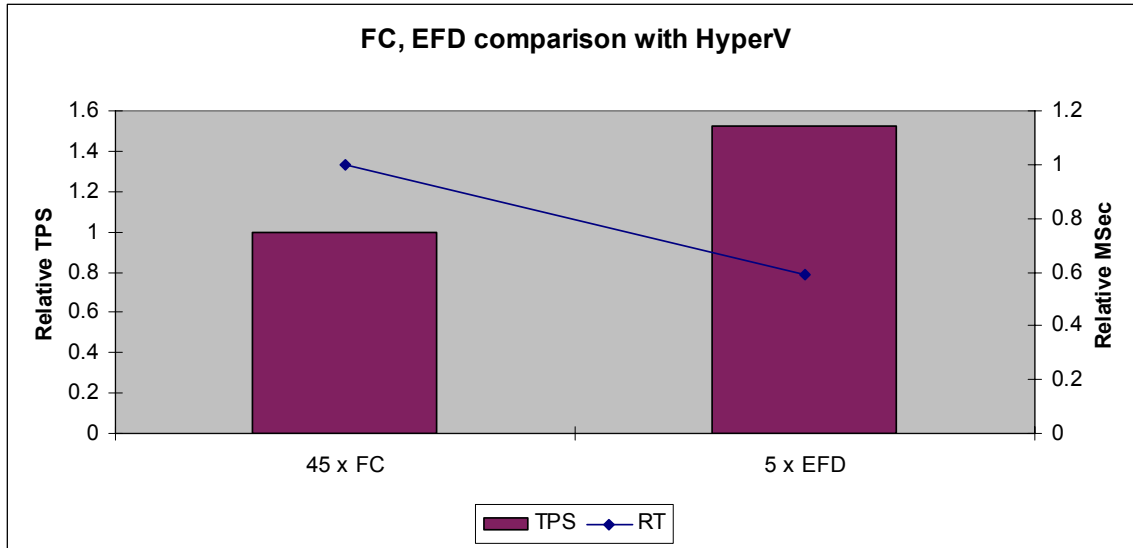


Figure 10. Hyper-V performance with EFDs

The FC configuration used 45 x 15k rpm drives in comparison to the five EFDs. Even with a smaller drive count, the EFDs provided 40 percent reduced latency and 52 percent increased transaction rate. The version of Hyper-V used only allowed for assigning four virtual processors to the virtual machine and processor utilization was at 100 percent for the EFD test case. Perfmon and Navisphere Analyzer data indicated that the EFDs were capable of servicing many more I/O requests than those driven by the virtual machine (given more assignable physical resources).

Overall, EFDs are a good complement to virtualization technology by allowing the consolidation of physical servers and enabling sharing of disk resources across multiple virtual machines. In addition, the lower energy requirement of EFDs aligns with the reduction of power consumption messaging of virtualization technology.

Impact of LUN cache settings

EMC CLARiiON storage arrays support enabling and disabling both read/write caches at LUN granularity. The default recommendation is to turn off both read and write caches on all the LUNs that reside on EFDs for following two reasons:

- EFDs are extremely fast, so when the read cache is enabled for the LUNs residing on them, the read cache lookup for each read request adds a significantly higher overhead as compared to FC drives, in an application profile that is not expected to get many read cache hits at any rate. Thus it is faster to directly read the block from the EFD.
- In a real-world scenario, where the CX4 is being shared by several applications and especially when deployed with slower SATA drives, the write cache may become fully saturated, placing the EFDs in a force flush situation, which adds latency. In these situations, it is better to write the block directly to EFDs than to the write cache of the storage system.

Even though the standard recommendation is not to enable caches for the LUNs residing on EFDs, DBAs and storage administrators can still choose to enable write cache for the LUNs residing on EFDs if they are aware of the implications. This may help them to get the maximum benefit from EFDs in some dedicated environments where the storage system is not shared across many applications. A careful analysis and benchmarking are highly recommended before deviating from the default settings.

Figure 11 shows that enabling write cache has a positive performance impact on 73 GB, 200 GB, and 400 GB EFDs. The improvement obtained by just enabling write cache for the EFDs cannot be guaranteed for

every application. It is heavily dependent on the nature of the application and its data access patterns. It was also noticed in our testing that when the database was deployed on EFDs with write cache enabled, EFDs on average delivered almost 1,600 percent improvement over FC drives on a per-disk efficiency basis.

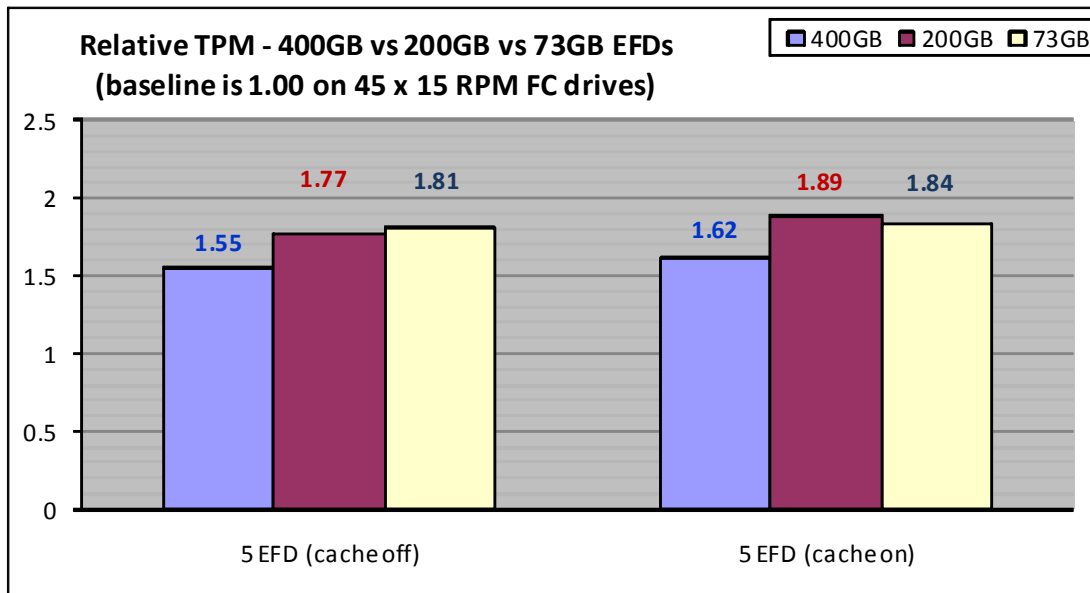


Figure 11. Relative transactions per minute comparison of 5 EFDs vs. 45 FC drives. Per-disk efficiency with cache on $\approx (45 / 5) * 1.78 \approx 16$ times

Conclusion

Incorporation of EFDs into CLARiiON CX4 with FLARE® 28 firmware provides a new Tier 0 storage layer that is capable of delivering very high I/O performance at very low latency, which can dramatically improve OLTP throughput and maintain very low response times. With comprehensive qualification and testing to ensure reliability and seamless interoperability, Tier 0 is supported by key CLARiiON software applications that enable advanced management tools.

Magnetic disk drive technology no longer defines the performance boundaries for mission-critical storage environments. The costly approach of spreading workloads over dozens or hundreds of underutilized disk drives is no longer necessary. With EFDs, a small number of drives can be used to replace many more FC drives and achieve comparable if not better performance. Tools from the Windows operating system and within SQL Server Management Studio can be leveraged to identify parts of the database that are best suited for this revolutionary new technology. I/O subsystem problems can be further diagnosed using the versatile Navisphere Analyzer tool. It is important to keep in mind that the I/O subsystem is only one of the potential bottlenecks preventing optimal SQL Server performance and a well-balanced environment is essential for the results.

CLARiiON now combines the performance and power efficiency of Flash drive technology with traditional disk drive technology in a single array managed with a single set of software tools, to deliver advanced functionality, ultra-performance, and expanded storage tiering options for any deployment needs.

References

The following can be found on Powerlink®, EMC's password-protected customer- and partner-only extranet. Access to the following may depend on your login credentials.

- *EMC CLARiiON CX4 Model 960 Networked Storage System* specification sheet
- *EMC CLARiiON CX4 Series Ordering Information and Configuration Guidelines*

The following white paper can also be found on EMC.com:

- *Introduction to the EMC CLARiiON CX4 Series Featuring UltraFlex Technology*