IT@Intel White Paper
Intel IT
IT Best Practices
Storage and Solid-State Drives
July 2012

Solid-State Drive Caching with Differentiated Storage Services

Our test results indicate that Differentiated Storage Services can significantly improve application throughput—resulting in a positive financial impact on storage costs.

Executive Overview

Intel IT, in collaboration with Intel Labs, is evaluating a new storage technology called Differentiated Storage Services (DSS).[1] When combined with Intel® Solid-State Drives, used as disk caches, we found that DSS can improve the performance and cost effectiveness of disk caching.

DSS uses intelligent I/O classification to enable storage systems to enforce per-class quality-of-service (QoS) policies on data, doing so at the granularity of blocks—the smallest unit of storage in a disk drive. By classifying disk I/O, DSS supports the intelligent prioritization of I/O requests, without sacrificing storage system interoperability. And when applied to caching, DSS enables new selective cache allocation and eviction algorithms.

In our tests, we compared DSS-based selective caching to equivalently configured systems that used the least recently used (LRU) caching algorithm and to systems that used no SSD cache at all. Our test results indicate that DSS can significantly improve application throughput—resulting in a positive financial impact on storage costs, in terms of cost per megabyte (MB) of each I/O operation per second (IOPS) delivered by the storage system; that is, U.S. dollars per MB per IOPS.

- DSS-based caching was capable of processing 1.5x to 6.8x more IOPS, depending on the workload and on whether the HDD was 7200 revolutions per minute (RPM) or 15K RPM.
- 7200 RPM HDDs with DSS caching outperformed 15K RPM HDDs without cache and 15K RPM HDDs with LRU.
- DSS caching reduced the cost per MB per IOPS by 66 percent compared to LRU-based caching, depending on the workload.

In our tests, DSS was tuned for general purpose file system workloads, where throughput is primarily limited by access to small files and their associated metadata. However, other types of workloads, such as databases, hypervisors, and big data, can also benefit from DSS by specifying application-specific DSS caching policies. For example, a database can specify caching policies for specific database queries. Intel Labs is exploring these and other potential applications of DSS technology.

Christian Black
Enterprise Architect, Intel IT

Michael Mesnier
Research Scientist, Intel Labs

Terry Yoshii
Enterprise Architect, Intel IT

---

[1] Mesnier, Michael, Jason Akers, Feng Chen, and Tian Luo. "Differentiated Storage Services." 23rd ACM Symposium on Operating Systems Principles (SOSP). October 2011.

Contents

IT@INTEL

The IT@Intel program connects IT professionals around the world with their peers inside our organization – sharing lessons learned, methods and strategies. Our goal is simple: Share Intel IT best practices that create business value and make IT a competitive advantage. Visit us today at www.intel.com/IT or contact your local Intel representative if you'd like to learn more.

BACKGROUND

As demand for data storage capacity at Intel continues to increase by about 35-percent annually, Intel IT—like the rest of the IT industry—is facing challenges related to data storage and caching.

- **Balancing throughput and cost.** Higher capacity utilization of a storage disk typically leads to decreased performance because of increased disk head movement. To avoid these performance issues, it is often tempting to purchase new disks before the old ones are completely filled. However, this solution, referred to as short-stroking, is not cost effective because it reduces effective utilization and increases storage costs.
- **Boosting storage performance.** Although processing capability—in particular, the number of cores per processor—has increased dramatically over the past several years, hard disk drive (HDD) storage performance has remained relatively flat. HDDs take milliseconds to access data, whereas processor performance can be measured in nanoseconds—a million times faster. Although a DRAM cache adequately supports large storage arrays, it cannot persistently store data without an expensive battery backup system. Additionally, the high cost of DRAM is a limiting factor in large-scale deployment.

High-performance solid-state drives (SSDs) that use flash memory technology support persistent storage and faster data access at a lower cost than DRAM, thereby addressing the storage performance problem. But, because the high cost of SSDs has prevented the mass replacement of HDDs, some form of caching is typically used, where a limited number of SSDs are used as a cache in front of conventional HDDs.

Caching is a common method of improving I/O performance in a storage system. The effectiveness of caching is largely based on the type of caching algorithm implemented and the performance of the physical caching device. There are many conventional caching algorithms, a few of which are described in Table 1.

Caching involves augmenting slower storage access with a small amount of high-speed storage that stores latency-sensitive data, effectively increasing the number of I/O operations per second (IOPS) produced by a given storage system. However, when the cache is under pressure, the caching algorithm may be unable to keep the correct data cached and instead may thrash—causing it to continuously exchange old data with new data, with the new data providing little or no value to the system.

Much of this thrash occurs because conventional storage (block-level) caches do not adequately prioritize I/O in the storage system—and this is because today's block-based storage protocols consist of two basic commands: READ and WRITE. This API abstracts away useful information about what is being read or written, such as filenames, file sizes, metadata structures, application context, and user and process information. While the block protocol enables interoperability across storage systems,

Table 1. Popular Caching Algorithms

Caching Algorithm	General Description
Least Recently Used (LRU)	Discards the least recently used items first, based on age-bits that track what data was used and when it was used.
Segmented LRU (SLRU)	Divides the cache into a probationary segment and a protected segment. Within each segment, cache lines are ordered from the most to the least recently accessed. Data from misses are added to the probationary segment, while hits are moved to the most recently used end of the protected segment.
Most Recently Used (MRU)	Discards the most recently used items first. MRU algorithms are most useful in situations where older items are more likely to be accessed, such as when a file—particularly a large file—is being repeatedly scanned, and for random access patterns.

the intelligent prioritization of I/O requests is extremely difficult.

Intel IT, working with Intel Labs, is now evaluating Differentiated Storage Services (DSS)—a new I/O classification technology that addresses many of the limitations of conventional caching algorithms. DSS enables policy-based caching, whereby different caching policies are assigned to different classes of data. DSS is a storage (block-level) caching technology that is currently being applied in storage systems.

PRIORITIZING I/O WITH DIFFERENTIATED STORAGE SERVICES

DSS enables a unique form of data caching referred to as selective allocation and eviction.

- **Selective allocation.** Always allocates high-priority classes, such as file system metadata and database system tables; conditionally allocates low-priority classes, depending on cache pressure and cache contents. A tunable parameter determines the high-priority and low-priority classes.
- **Selective eviction.** Evicts in priority order (lowest priority first). For example, temporary database tables would be evicted before system tables.

DSS uses I/O classification to improve the performance—and therefore the cost effectiveness—of data caching, going far beyond the capabilities of the Segmented Least Recently Used caching algorithm, described in Table 1. By using I/O classification, DSS can enable storage systems to enforce per-class quality-of-service (QoS) policies, such as I/O priorities.

The specific I/O classification scheme depends on the origin of the I/O request. For example, the request could come from the file system, OS, database, hypervisor, or cloud storage server. Intel Labs is exploring how to best classify I/O for each of these application domains.

As shown in Figure 1, DSS classifies all I/O requests between a computer system and a storage system. Every I/O request contains a small classification field that uniquely identifies the I/O class. Examples of I/O classes include metadata, data, small file, database table, virtual machine disk image, and audio file. The classification field for Small Computer System Interface (SCSI) I/O is a five-bit number (the SCSI Group Number), which allows for 32 different classes of data. In this way, the storage system can determine the class of each block-level I/O request, and different QoS policies can be associated with each class of data. Policies may vary across storage system vendors; in our tests we evaluated storage system policies based on cache priority levels.

Before the storage system is brought online, the computer system must inform it of the various classes of data and their associated policies. For example, any I/O classified as metadata might be assigned a high-priority caching policy. The table on the left in Figure 1 lists sample I/O classes and their assigned policies. By separating policy from the classification mechanism in this manner, DSS enables OS interoperability across various storage systems, while still supporting the intelligent prioritization of I/O requests. For example, a file system can use the same I/O classification scheme across different storage systems, but it may need to assign policies to these classes differently. The policies that are assigned will depend on the capabilities of the storage system.

The methods used to classify data and assign policies are currently vendor-specific, although Intel is working with the industry to adopt a standard approach to these processes. Further, the storage system vendor may provide defaults, which are tunable by a system administrator.

Quality of Service (QoS) Policy Assignment

Highest priority = lowest number (0)

Scenario	I/O Classifications	QoS Policy Assignments
Database	System tables	Priority 0
	Temporary tables (on write)	Priority 1
	Randomly accessed tables	Priority 2
	Temporary tables (on read)	Priority 3
	Sequential tables	Bypass caching
	Index files	Bypass caching
Operating System	Metadata	Low latency
	Boot files	Low latency
	Small files	High throughput
	Media files	High bandwidth
File System	Metadata	Priority 0
	Journal	Priority 0
	Directories	Priority 0
	Files less than 4 kilobytes (KB)	Priority 1
	Files less than 16 KB	Priority 2
	Files less than 64 KB	Priority 3
	Files greater than 1 gigabyte	Lowest priority

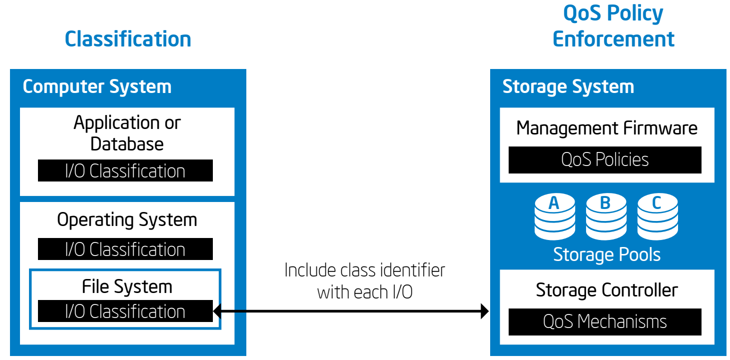


Figure 1. Differentiated Storage Services identifies the classes of I/O and the caching policy for each class.

Phase 1: MacroSetup

- Create 600,000 files in 20,000 directories. File sizes are selected from the SPECsfs* file size distribution. Sizes range from 1 kilobyte to 32 megabytes.

Phase 2: MacroTrans

- Perform 60,000 transactions against the file pool. The read-and-write request sizes are also chosen from the SPECsfs request size distributions.
- A transaction is defined as reading an existing file sequentially, where the file is chosen at random, or creating a new file, where the directory is chosen at random.
- We record the number of transactions per second or, equivalently, the time it takes for all 60,000 transactions to complete.

Phase 3: File System Utilities

These utilities consist of the following workloads:

- **MacroPrefind.** Search the file system for a non-existent file; walks all directory data
- **MacroArchive.** Back up one of the 20,000 directories; creates a large backup file
- **MacroFind.** Search the file system for a non-existent file; walks all directory data
- **MacroScan.** Antivirus scan

Figure 2. Test suite. Visit SPECsfs* for more information on file size distribution: www.spec.org

EVALUATING CACHING WITH DIFFERENTIATED STORAGE SERVICES

Working with Intel Labs, Intel IT ran multiple tests to evaluate the effectiveness of DSS when using SSDs as a disk cache. Our evaluation objectives included the following:

- Perform a technical assessment of the DSS capability compared to equivalent non-cached and LRU-cached configurations.
- Conduct a financial analysis of DSS technology.

Test Methodology and System Configuration

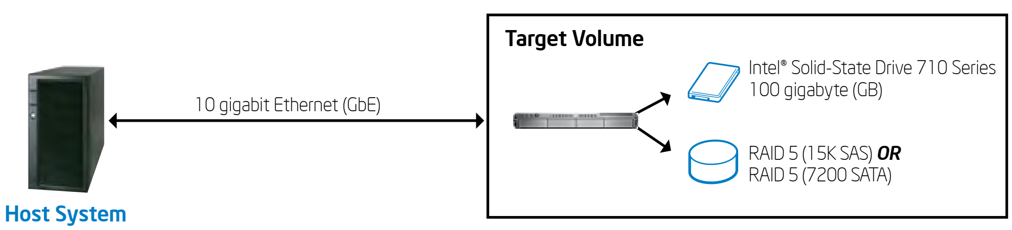
Using Intel® Solid-State Drives as the storage array cache, we evaluated the performance of common enterprise workloads, such as creating, reading/writing files, performing a directory search, and performing an antivirus scan. We ran the tests on both 15K and 7200 revolutions per minute (RPM) drives, and we compared the results from DSS, LRU, and non-cached systems.

To perform the tests, we executed a three-phase scripted test suite, described in Figure 2. Figure 3 specifies the test system configuration. The working set size—the total sum of all file sizes in the test set—was approximately 1 terabyte. We used an SSD cache size (100 gigabytes) that is 10 percent of the used disk capacity, which is in our experience the most cost-effective cache size for real-world applications.

We ran three complete sets of tests and averaged the results. For the performance portion of our analysis, we measured application running time (for example, the time to complete the 60,000 transactions, or the time to search the file system), and we also measured block-level I/O operations per second (IOPS). For the financial portion of our analysis, we calculated the average cost per megabyte per IOPS.

We used cache statistics to calculate the hit rate for each cache size. The hit rate is the percentage of read requests that is already in the cache. A "miss" is defined as occurring when the desired data is not in the cache and must be retrieved from main storage. A higher hit rate indicates the cache is more effective. We also evaluated cache behavior, such as eviction of small files and metadata, and how that behavior affected throughput rates.

Table 2 illustrates the cache configuration we used in our tests. Both the LRU and DSS caches were write-allocate and write-back; we disabled read-allocate because the poor



Host System

- Intel® Xeon® processor E5-2680 (dual-processor with eight cores) with 32 GB RAM
- Microsoft iSCSI Initiator (10GbE)
- Intel's Differentiated Storage Services software (filter drivers that classify I/O from NTFS)

Storage System

- Intel Xeon processor E5-2680 (dual-processor with eight cores) with 32 GB RAM
- Linux* (Fedora 13)
- 10GbE iSCSI Initiator (Intel® Open Storage toolkit)
- Standard RAID Controller
  - Intel® Solid-State Drive 710 Series as cache
  - 5-disk RAID 5 (15K SAS) ***or*** 5-disk RAID 5 (7200 SATA)
  - 100-GB cache size (10 percent of a 1-terabyte working set)
- Intel's Differentiated Storage Services software (a cache that implements selective allocation and eviction)

Figure 3. Configuration of the test system.

locality of reference in the tested workloads (Figure 2) reduced the performance of both the DSS and LRU caches. For the DSS configuration, we classified all I/O from the file system using Windows* filter drivers, assigned a SCSI group number to each class, and gave cache priority to small files and their associated metadata.

The prototype iSCSI storage system we evaluated from Intel Labs runs on the Linux* target and contains the SSD caching driver, with the modifications needed to take advantage of I/O classification. In particular, the two new caching algorithms—selective allocation and selective eviction—were implemented. These algorithms inspect the relative priority of each I/O request and provide a mechanism by which NTFS caching policies can be enforced in the iSCSI storage system. The iSCSI storage system also contains an LRU mode of operation, which served as our baseline for comparison.

Performance Analysis Results

Our test results indicate that by applying QoS policies on data classes, DSS can significantly improve application throughput, with a positive financial impact on storage costs in terms of cost per MB per IOPS. Combined with high-performance SSDs, DSS caching produces significantly better application performance than standard caching algorithms prevalent in the industry today.

In our tests, the DSS solution outperformed both non-cached and LRU caching. DSS-based caching completed tests 1.5x to 5.6x faster than non-cached systems, and 1.5x to 6.8x faster than LRU caching, depending on the workload and on whether the HDD was 7200 RPM or 15K RPM.

Table 3 summarizes the performance analysis from our tests; Figures 4 and 5 show detailed performance analysis results.

As shown in Figure 4, DSS caching outperformed both non-cached and LRU-cached systems in all tests. In fact, LRU provided little benefit over a non-cached system because of the extremely random access characteristics of these typical file server workloads.

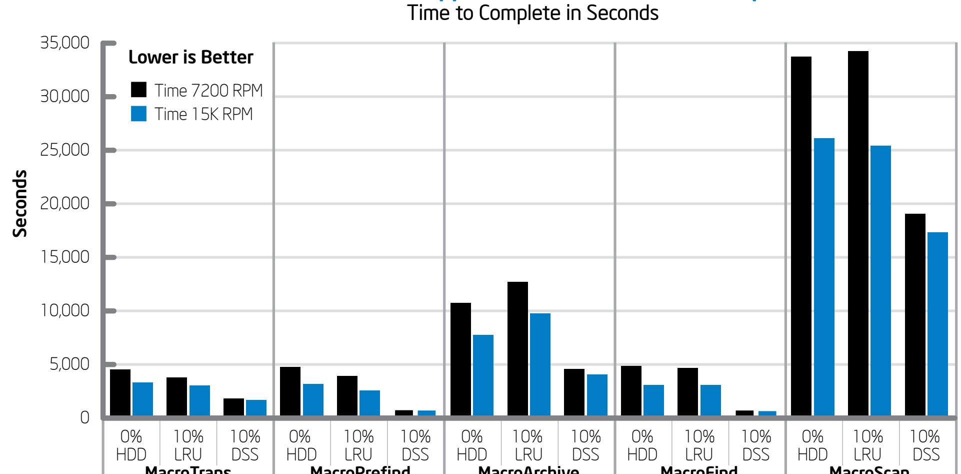
Table 2. Cache Configuration

I/O Task	Least Recently Used (LRU) Non-selective Caching	Differentiated Storage Services (DSS) Selective Caching
Allocation	Caches everything	Policy-based ▪ If more than 15 percent of the cache is clean, everything is allocated. ▪ If less than 15 percent of the cache is clean, only metadata and files that are less than 1 megabyte are allocated
Eviction	Eviction of LRU blocks	Policy-based ▪ Eviction of largest cached objects first ▪ LRU eviction within each I/O class
Classification	Not applicable	Configurable I/O classes

Table 3. Differentiated Storage Services Caching Performance Summary. Intel internal measurements, May 2011.

	HDD without Caching (Baseline)	LRU Caching	DSS Caching	Result
7200 RPM: Averaged across all workloads at 10-percent cache (higher is better)				
Average Relative Application Performance	100%	99%	217%	DSS reduces average application completion times by 54 percent
Average IOPS	162	161	468	DSS delivers highest overall average I/O performance
Relative IOPS	1.0	1.0	2.9	DSS delivers 2.9x higher IOPS performance than non-cached and LRU-cached systems
15K RPM: Averaged across all workloads at 10-percent cache (higher is better)				
Average Relative Application Performance	100%	99%	176%	DSS reduces average application completion times by 43 percent
Average IOPS	231	228	556	DSS delivers highest overall average I/O performance
Relative IOPS	1.0	1.0	2.4	DSS delivers 2.4x higher IOPS performance than non-cached and LRU-cached systems

HDD - hard disk drive; **IOPS** - I/O operations per second; **LRU** - least recently used; **RPM** - revolutions per minute; **%** - percent



	MacroTrans 0% HDD	MacroTrans 10% LRU	MacroTrans 10% DSS	MacroPrefind 0% HDD	MacroPrefind 10% LRU	MacroPrefind 10% DSS	MacroArchive 0% HDD	MacroArchive 10% LRU	MacroArchive 10% DSS	MacroFind 0% HDD	MacroFind 10% LRU	MacroFind 10% DSS	MacroScan 0% HDD	MacroScan 10% LRU	MacroScan 10% DSS
7200 RPM	4,546	3,797	1,798	4,761	3,925	855	10,739	12,722	4,555	4,840	4,658	827	33,748	34,255	19,050
15K RPM	3,328	3,026	1,666	3,170	2,577	825	7,745	9,768	4,072	3,072	3,080	770	26,113	25,403	17,314

DSS - Differentiated Storage Services; **HDD** - hard disk drive; **LRU** - least recently used; **RPM** - revolutions per minute; **%** - percent

Figure 4. A 15K RPM drive with an Intel® Solid-State Drive 710 Series serving as a Differentiated Storage Services cache provides the best performance. Intel internal measurements, May 2011.

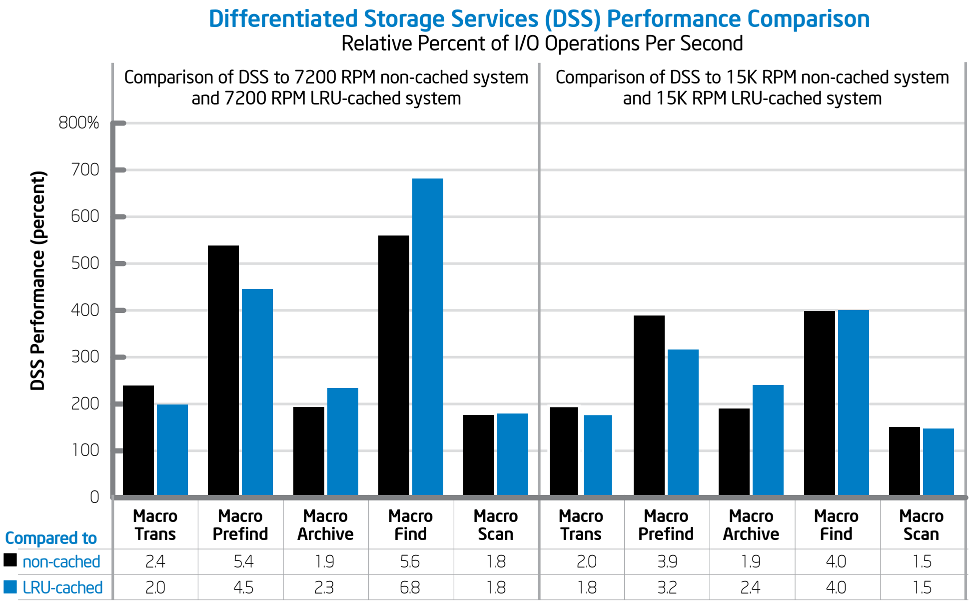


Figure 5. Differentiated Storage Services caching outperforms non-cached and least recently used cached systems for both 15K RPM and 7200 RPM drives. Intel internal measurements, May 2011.

Table 4. Financial Summary. Intel internal measurements, May 2011.

	HDD without Caching (Baseline)	**LRU Caching**	**DSS Caching**	**Result**
7200 RPM: Averaged across all workloads at 10-percent cache				
Average Cost per Megabyte per IOPS	USD 0.88	USD 1.43	USD 0.49	DSS provides the best cost per performance value
Cost Reduction Over Non-cached System	0	-63%	44%	DSS delivers a 44-percent reduction compared to a non-cached system
Cost Reduction Over LRU-cached System	38%	0	66%	DSS delivers a 66-percent reduction compared to LRU caching
15K RPM: Averaged across all workloads at 10-percent cache				
Average Cost per Megabyte per IOPS	USD 5.47	USD 6.81	USD 2.80	DSS provides the best cost per performance value
Cost Reduction Over Non-cached System	0	-24%	49%	DSS delivers a 49-percent reduction compared to non-cached system
Cost Reduction Over LRU-cached System	20%	0	59%	DSS delivers a 59-percent reduction compared to LRU caching

DSS - Differentiated Storage Services; **HDD** - hard disk drive; **IOPS** - I/O operations per second; **LRU** - least recently used; **RPM** - revolutions per minute; **USD** - U.S. dollars

In particular, 7200 RPM HDDs with DSS caching outperformed both 15K RPM HDDs without caching and 15K RPM HDDs with LRU caching. The 15K RPM HDD with DSS caching reduced application completion times by 43 percent, and the 7200 RPM HDD with DSS caching reduced those times by 54 percent.

For example, for the MacroScan workload on 7200 RPM drives, the DSS time reduction was greater than four hours—the non-cached and LRU-cached systems took about nine-and-a-half hours to complete the workload, while the DSS-cached system completed the workload in a little over five hours.

We achieved similar results with the 15K RPM drives: the non-cached and LRU-cached systems took a little over seven hours to complete the workload, while the DSS-cached system reduced this time by more than two hours, down to just under five hours.

Figure 5 illustrates that the DSS caching solution outperformed non-cached and LRU caching for both 15K RPM and 7200 RPM drives. For 15K RPM drives, the DSS performance was 1.5x to 4.0x better than the non-cached system, and 1.5x to 4.0x better than LRU caching. For 7200 RPM drives, the DSS performance was 1.8x to 5.6x better than non-cached systems and 1.8x to 6.8x better than LRU caching.

Financial Analysis Results

In addition to analyzing the performance of DSS-based caching compared to LRU caching and a non-cached system, we determined the cost per MB per IOPS for each workload, on both 7200 RPM and 15K RPM disks. Table 4 provides a summary of our financial analysis; Figures 6 and 7 show detailed financial results.

The standard method of calculating storage cost is U.S. dollars per MB. This formula works well but does not take into consideration the performance delivered by the storage system being purchased. Dividing the cost-per-capacity figure by IOPS—that is, by I/O throughput—yields a number that reasonably reflects the cost of the storage solution being purchased. Because the test environment remained unchanged except for the disk components, we used only the HDD and SSD as the cost for this analysis.[1]

- For the cost of the HDD without caching, we factored in only the cost of the HDDs.
- For the cost of LRU caching and DSS caching, we added in the additional cost of the Intel SSD 710 Series.

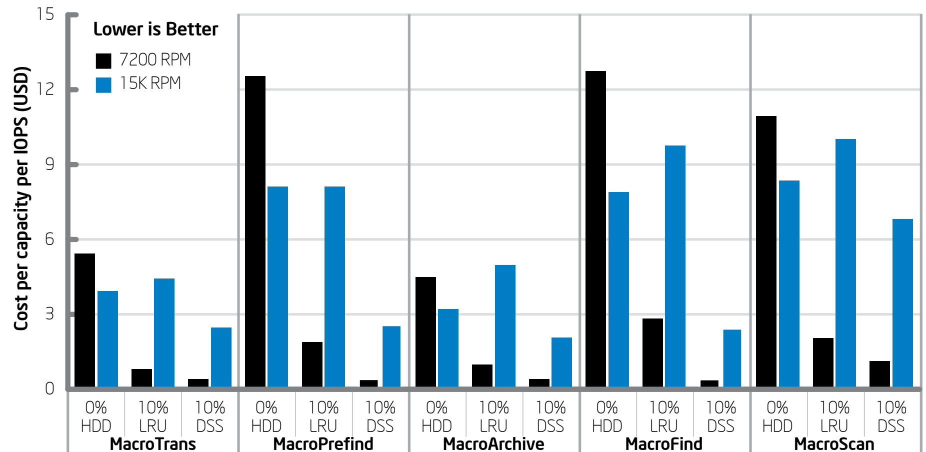
As shown in Figure 6, our tests indicate that DSS significantly reduced the cost storage when compared to the cost of the HDD only and the cost of HDD with LRU cached storage. The 7200 RPM HDD with DSS caching provided the lowest cost per MB per IOPS across all workloads. And as discussed previously, this configuration produced higher performance than the 15K RPM HDD only and the 15K RPM HDD with LRU cached storage.

As shown in Figure 7, DSS caching reduced the cost for all relevant workloads. Our tests indicated that DSS reduced the cost of higher-performance 15K RPM storage by up to 75 percent over standard LRU caching and up to 69 percent over non-cached 15K RPM HDDs. DSS reduced the cost of 7200 RPM storage by up to 85 percent over standard LRU caching and up to 79 percent over non-cached 15K RPM HDDs. The highest cost reduction was for directory search operations, because the caching and eviction policies for our tests were set to provide optimal performance for file metadata reads such as finds and searches.

In summary, replacing 15K RPM drives without cache, or 15K RPM drives with LRU caching, with 7200 RPM drives configured with DSS caching is an extremely cost-effective solution that can also provide much higher performance. Our financial analysis does not include the power and cooling savings offered by 7200 RPM disks with SSD cache, and we did not factor in the time-saving benefits of completing application jobs in less time, because these factors may vary significantly by each specific application and by geographic location. However, the data and simple analysis from the evaluation of DSS caching technology is very compelling from both a technical and a business perspective.

Cost Analysis at 10-Percent Cache

Cost = Cost of HDD + SSD (for LRU and DSS) / Total disk capacity purchased / IOPS



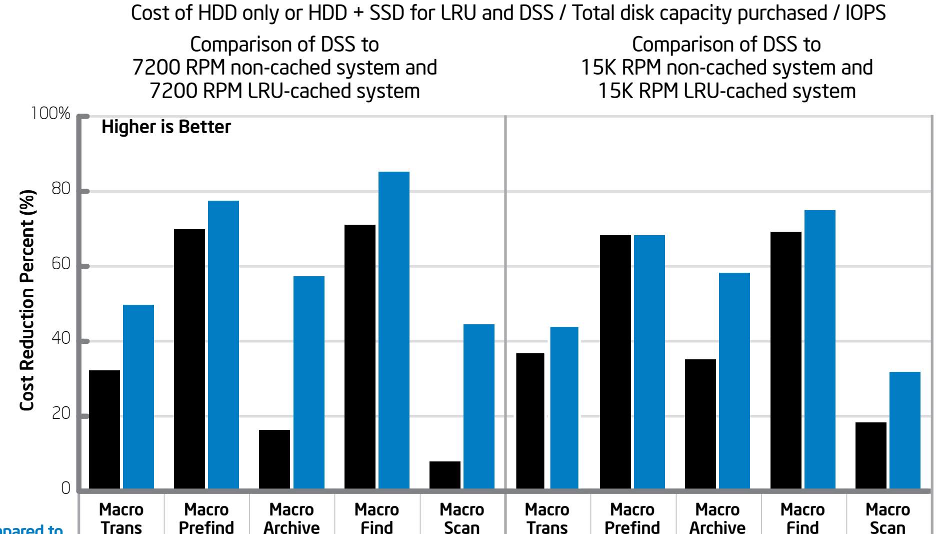
	MacroTrans 0% HDD	MacroTrans 10% LRU	MacroTrans 10% DSS	MacroPrefind 0% HDD	MacroPrefind 10% LRU	MacroPrefind 10% DSS	MacroArchive 0% HDD	MacroArchive 10% LRU	MacroArchive 10% DSS	MacroFind 0% HDD	MacroFind 10% LRU	MacroFind 10% DSS	MacroScan 0% HDD	MacroScan 10% LRU	MacroScan 10% DSS
7200 RPM	5.44	0.82	0.41	12.54	1.89	0.42	4.49	0.99	0.42	12.74	2.83	0.41	10.95	2.05	1.14
15K RPM	3.93	4.43	2.47	8.13	8.13	2.58	3.21	4.98	2.08	7.91	9.77	2.44	8.36	10.02	6.82

DSS - Differentiated Storage Services; **HDD** - hard disk drive; **IOPS** - I/O operations per second; **LRU** - least recently used; **RPM** - revolutions per minute; **SSD** - solid-state drive; **USD** - U.S. dollars; **%** - percent

Figure 6. A 7200 RPM drive with an Intel® Solid-State Drive 710 Series serving as a Differentiated Storage Services cache provides the best value. Intel internal measurements, May 2011.

DSS Solution Cost Reduction

Cost of HDD only or HDD + SSD for LRU and DSS / Total disk capacity purchased / IOPS



Compared to	Macro Trans	Macro Prefind	Macro Archive	Macro Find	Macro Scan	Macro Trans	Macro Prefind	Macro Archive	Macro Find	Macro Scan
non-cached	32.3%	69.9%	16.3%	71.1%	79.0%	37.2%	68.3%	35.2%	69.2%	18.3%
LRU-cached	49.7%	77.5%	57.3%	85.3%	44.5%	44.2%	68.3%	58.3%	75.0%	31.9%

DSS - Differentiated Storage Services; **HDD** - hard disk drive; **IOPS** - I/O operations per second; **LRU** - least recently used; **RPM** - revolutions per minute; **SSD** - solid-state drive; **%** - percent

Figure 7. Differentiated Storage Services caching reduced the cost for all relevant workloads. Intel internal measurements, May 2011.

[1] The cost of the enterprise hard disk drives and the Intel® Solid-State Drive 710 Series were derived from current (as of May 2012) online pricing.

Beyond Caching: Broadening the Application of Differentiated Storage Services (DSS)

For our tests, the solution was specifically tuned for a general file system workload, which is why workloads with large amounts of metadata showed the greatest benefit from DSS caching. However, the advantage of DSS is that its policies can be optimized for larger files as needed. This is the first time we've seen an interface that lets the OS effectively pass "hints" to a storage system with each I/O request, without sacrificing interoperability.

Our evaluation of DSS for use in data caching is only the beginning of the possible applications of this new technology. DSS has great potential in a variety of other uses, such as database access, virtualization, security, cloud storage, big data, reliability, and application acceleration engines. Intel Labs is already evaluating DSS in the context of large-file solutions, such as databases, which store tables as large files, and hypervisors, which store virtual machine disk images as large files.

We are also exploring multi-level caching solutions, where DSS could be used to implement multiple caching layers. For example, non-volatile memory could be used as an even faster cache on top of an existing solid-state drive plus a hard disk drive configuration. This type of multi-level cache could result in even greater performance and financial benefit.

CONCLUSION

DSS enables a unique form of data caching that prioritizes I/O by creating classes of I/O requests and applying policies to these classes. DSS enables interoperability across storage systems, while supporting intelligent prioritization of I/O requests.

When we compared DSS-based caching to equivalently configured systems that used the LRU caching algorithm and no cache at all, we found that DSS can significantly improve application throughput—resulting in a positive financial impact on storage costs, in terms of cost per MB per IOPS. Combined with high-performance SSDs, DSS caching produces significantly better application performance than standard caching algorithms prevalent in the industry today.

- In our tests, DSS-based caching was capable of processing 1.5x to 6.8x more IOPS, depending on the workload and on whether the HDD was 7200 RPM or 15K RPM.
- 7200 RPM HDDs with DSS caching outperformed 15K RPM HDDs without cache and 15K RPM HDDs with LRU.
- DSS caching reduced the cost per MB per IOPS by 66 percent compared to LRU-based caching, depending on the workload.

Because DSS policies can be adjusted as necessary, other applications, such as databases, virtualization, security, cloud storage, big data, reliability, and application acceleration engines could all benefit from this technology.

RELATED READING

- Michael Mesnier, Jason Akers, Feng Chen, and Tian Luo. "Differentiated Storage Services." 23rd ACM Symposium on Operating Systems Principles (SOSP). October 2011.

ACRONYMS

DSS	Differentiated Storage Services
HDD	hard disk drive
IOPS	I/O per second
LRU	least recently used
MB	megabyte
MRU	most recently used
QoS	quality of service
RPM	revolutions per minute
SCSI	Small Computer System Interface
SFS	system file server
SLRU	segmented LRU
SSD	solid-state drive

For more information on Intel IT best practices, visit www.intel.com/it.