Performance of vSphere Flash Read Cache in VMware vSphere 5.5

Performance Study

TECHNICAL WHITE PAPER

vmware

Table of Contents

Introduction 3
vFRC Architecture Overview 4
Performance Tunables 5
Workload Characteristics 5
Cache Size 6
Cache Block Size 6
Flash Device Type 8
Performance Results 8
Decision Support System Database Workload (Swingbench DSS on Oracle 11g R2) 8
Test Bed 9
Results 9
DVD Store Benchmark (Microsoft SQL Server 2008) 12
Test Bed 12
Results 12
Accurate Replay of Enterprise I/O Traces 13
Performance Best Practices 15
Setting the Correct Cache Size 15
Setting the Correct Cache Block Size 16
Choosing the Right SSD Device 16
Cache Migration during vMotion 16
Conclusion 17
References 17

Introduction

VMware vSphere® 5.5 introduces new functionality to leverage flash storage devices on a VMware ESXi™ host. The vSphere Flash Infrastructure layer is part of the ESXi storage stack for managing flash storage devices that are locally connected to the server. These devices can be of multiple types (primarily PCIe flash cards and SAS/SATA SSD drives) and the vSphere Flash Infrastructure layer is used to aggregate these flash devices into a unified flash resource. You can choose whether or not to add a flash device to this unified resource, so that if some devices need to be made available to the virtual machine directly, this can be done.

The flash resource created by the vSphere Flash Infrastructure layer can be used for two purposes: (1) read caching of virtual machine I/O requests (vSphere Flash Read Cache) and (2) storing the host swap file. This paper focuses on the performance benefits and best practice guidelines when using the flash resource for read caching of virtual machine I/O requests.

vSphere Flash Read Cache (vFRC) is a feature in vSphere 5.5 that utilizes the vSphere Flash Infrastructure layer to provide a host-level caching functionality for virtual machine I/Os using flash storage. The goal of introducing the vFRC feature is to enhance performance of certain I/O workloads that exhibit characteristics suitable for caching.

In this paper, we first present an overview of the vFRC architecture, detailing the workflow in the read and write I/O path. We then show some of the workloads that perform better vFRC through detailed test results. We conclude the paper with performance best practices guidelines when using vSphere Flash Read Cache.

vFRC Architecture Overview

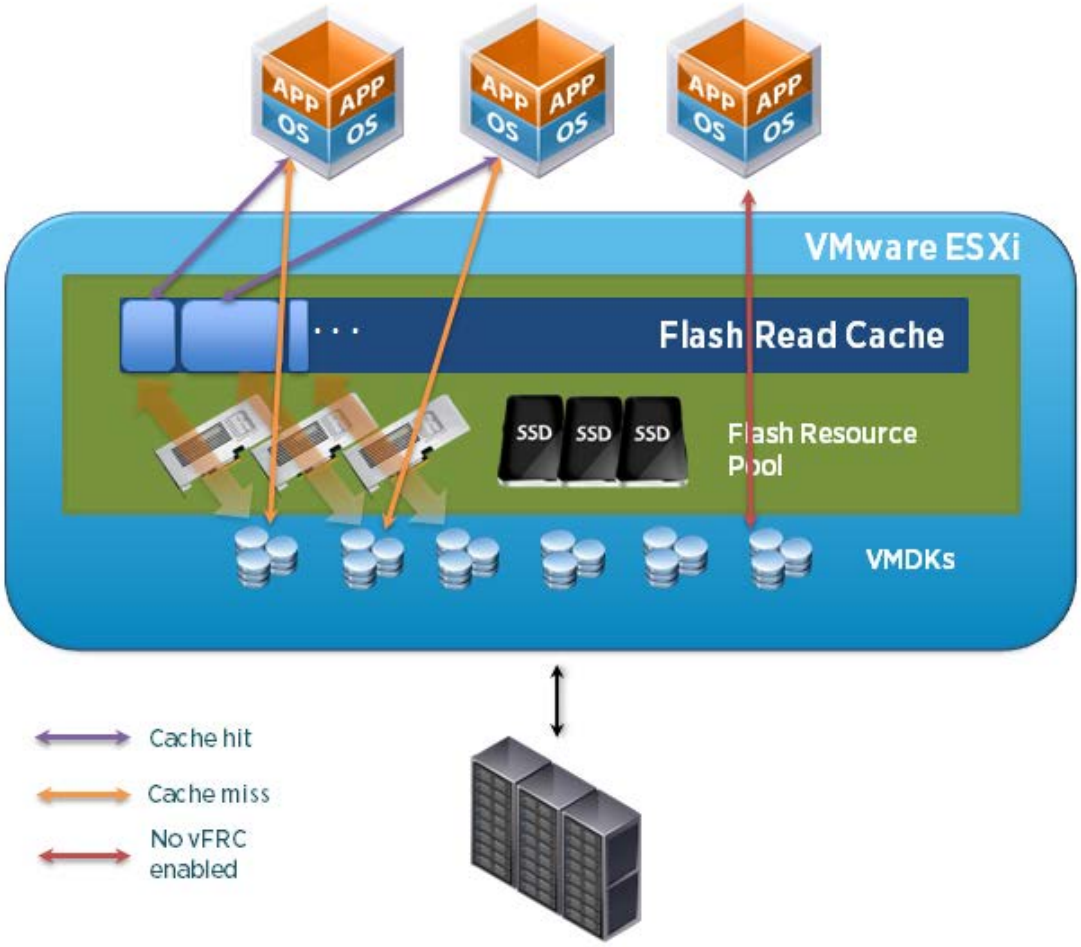

Figure 1. vSphere Flash Read Cache architecture

SSD drives and PCIe flash cards connected locally to the host server can be used to create a virtual flash resource. vFRC operates on top of the virtual flash resource and lets you provision space within the unified flash resource pool for their different workloads. This is illustrated in Figure 1, which shows how vSphere Flash Infrastructure and vFRC fit into the overall system architecture. vFRC interoperates well with other vSphere features like vMotion, snapshots, and suspend-resume.

Each workload exhibits different behavior and utilizes the cache differently. In order to enable the user to configure different amounts of cache space and cache configurations for different workloads, vFRC is enabled on a per-VMDK basis. Each VMDK can be configured with a certain size of flash cache with a certain cache block size. In later sections, we discuss the implications of cache block size on performance and some guidelines to configure them.

Once vFRC is enabled for a virtual disk, the cache is created when the virtual machine boots. vFRC is a write-through cache. This means that even though write I/O requests are cached by vFRC, I/O request completion status is sent to the guest virtual machine only after the data is written to physical storage. Because of this design, there is no change in the existing data reliability and availability guarantees.

On the read I/O path, when a request arrives from the guest virtual machine to a vFRC-enabled VMDK, vFRC metadata is looked up to find if the entire data requested is available in the cache. If it is available, a cache read fetches the data from the flash device and the request is serviced. This is known as a *vFRC hit*. If some or all of the data requested is not available in the cache (which means that the data is accessed for the first time, or this data was available in cache before and was subsequently evicted), then the entire requested data is fetched from the VMDK and returned to the guest, while simultaneously writing those data to the flash cache. This operation is called a *vFRC miss,* and this leads to a subsequent cache fill operation. On the write I/O path, data is first written to the permanent storage (VMDK) and asynchronously written to the flash cache. vFRC is a volatile cache: a cold restart of virtual machine destroys the cache file and it will be recreated again on boot. Other scenarios when the cache will be destroyed include suspend-resume, vMotion of a virtual machine without migrating the cache, snapshot consolidation, snapshot revert, and so on.

Cache fills and cache evictions happen in the granularity of a *cache block size*. This value ranges from 4KB to 1MB to enable you to best configure your cache block size based on the I/O size of workloads. Even though cache fills and cache evictions happen in the granularity of a cache block size, actual read I/O serviced by the cache can be smaller than the cache block size. For example, if the cache block size is 64KB, and a 4KB read I/O request is issued by the guest virtual machine, and if the data is not available in the cache, a 4KB read is issued to the VMDK. When populating the cache, the vFRC algorithm looks for a 64KB region to place the new 4KB data. If no free space is available, a 64KB region is evicted and the space is used to hold the new 4KB data. The remaining 60KB region in the 64KB cache block is marked as invalid. The cache block size parameter therefore has profound effects on performance, which will be explained in detail in the following sections.

Performance Tunables

The performance of vFRC depends on a variety of factors like the workload, cache block size, cache size, and type of flash device used. It is important for you to understand how these factors affect performance. This understanding will set expectations for the amount of performance enhancement to be expected from vFRC. In this section, these factors and their impact on application performance are discussed in detail.

Workload Characteristics

A good understanding of the workload behavior and characteristics is the most important factor in deciding whether or not to enable vFRC because not all workloads will benefit from vFRC. vFRC caches data from both read and write I/Os, but write I/Os are always serviced by the underlying storage. Therefore, workloads that have a majority of reads can directly benefit from vFRC. Write-intensive workloads can also benefit from vFRC in some cases, although not directly. For example, consider two applications sharing a storage array and one of the applications is read-intensive and well suited for vFRC. If the other application is write-intensive, vFRC can improve the performance of the second application by decreasing the amount of I/O load in the storage array. This is because most of the I/O from first application will be serviced by the local flash storage, hence reducing load in the storage array.

In addition to being read-dominated, the workload access pattern should contain a frequently accessed working set to benefit from vFRC. Typically, when an I/O block is accessed the first time, it is brought to cache. Only when the same block is subsequently accessed, can it be serviced from the cache. If not, the block will stay in the cache for awhile and eventually be evicted to make space for other blocks. If the workload accesses only unique data blocks without any repeated accesses of any blocks, then vFRC merely stores data in flash only to evict it after some time and there will be only slight overhead due to adding an extra layer in the I/O path for zero benefit. Therefore, vFRC benefits workloads with high amounts of data re-access. These are generally termed *cache-friendly workloads*.

Cache Size

Configuring the cache size is important for optimal behavior of vFRC. The cache size should be big enough to hold the active working set of the workload. If the cache size is smaller than the active working set of the workload, useful cache blocks that may be accessed later will have to be evicted to hold other blocks. vFRC uses a replacement algorithm that favors retaining popular blocks for a much longer time and it is reactive to changes in workload characteristics. But if the cache size is not big enough to hold even the popular working set of the workload, this will result in increased cache misses and hence lower performance.

However, configuring abundant cache size for a workload whose active working set is much smaller than the cache size is also not good for performance. One obvious effect of this would be lack of flash space for other workloads, assuming there is only limited flash resource per server. Another instance of a performance issue because of higher-than-required cache size is during migration of the virtual machine. Because vFRC is implemented as thick cache files, migrating the cache would involve migrating the entire cache file (along with the unused portion of the cache). This will increase the vMotion duration to a long time, especially if tens of gigabytes of cache space is configured. Guidance on how to configure the cache size is discussed in the "Performance Best Practices" section.

Cache Block Size

Cache block size is the minimum granularity of cache fills and cache evictions. Having the optimal cache block size is critical to overall performance of vFRC. Because the metadata structures for vFRC are indexed by cache block size, the metadata footprint size depends on the cache block size. For good performance, vFRC places its metadata in the memory and therefore the cache block size has a direct correlation with memory usage. The higher the cache block size, the lower the amount of metadata is required for indexing those blocks and therefore results in a smaller memory footprint. Consequently, a smaller cache block size consumes a bigger memory footprint. Figure 2 shows the amount of memory consumed as a percentage of total cache size, for various cache block sizes. The figure shows that as the cache block size increases, the amount of memory required to store the metadata decreases.

Also, for higher block sizes, the number of I/Os required to access data from the cache is reduced. For example, if the cache block size is 4KB, there is a chance for a 256KB segment that is contiguous on physical storage to be scattered on the cache device. This is because the individual 4KB segments in the 256KB data might have been accessed at different times, ending up at different locations on the cache device. It is more efficient to access 256KB data in a single I/O than to issue multiple 4KB I/Os to the cache and aggregate them before serving them to the user. Therefore it is not efficient to have a smaller cache block size if the I/O size of the workload is larger.

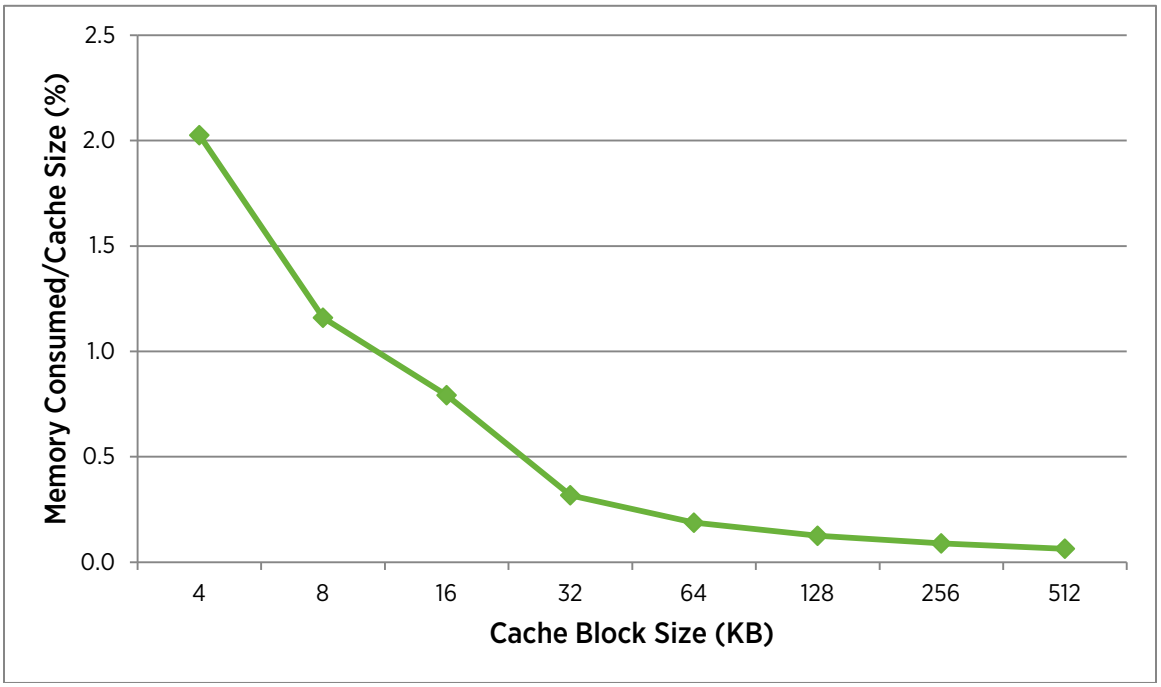

Figure 2. Memory consumption with respect to cache block size

However, higher cache block sizes are not better in terms of performance and efficient management of cache space. As cache evictions and cache fills happen in the granularity of a cache block size, if the cache block size is much higher than the typical I/O size, there might be a situation where an additional amount of already cached data needs to be evicted to store a small amount of new data. For example, consider a cache with a cache block size of 64KB. Assuming there are no free blocks in the cache to hold new data, when a 4KB I/O arrives from the guest VM, if there is a cache miss, 64KB of cached data would have to be evicted to hold the new 4KB data. This leads to sub-optimal management of cache space and may reduce the overall cache hit rate for the workload.

In Figure 3, the importance of choosing the right block size for vFRC is illustrated. The graph shows performance differences when running a Hardware Monitoring Server workload. Details of this workload are discussed in the performance results section. I/O trace for this workload is publicly available from Microsoft Research Cambridge and are widely used in the storage research community [8]. The Hardware Monitoring Server I/O traces were replayed in our system with various configurations, namely baseline case (absence of vFRC), and all other cases are with vFRC enabled and with different cache block sizes: 4KB, 8KB, and so on. We have plotted the average per-request latency during replay of the I/O trace under different configurations. For this particular example, the 4KB cache block size shows the most benefit. This is because the most dominant I/O size for this workload is 4KB and therefore the cache block size matches well with the I/O size. Larger than optimal cache block sizes show degraded performance for this workload because, with larger cache block sizes, the eviction granularity is also greater. Therefore, when a 4KB I/O is issued, to do a cache fill of 4KB, larger amounts of data are evicted. This leads to an increased rate of cache misses and hence there is a decrease in performance. More details about how to set up an optimal block size for vFRC are covered in the "Performance Best Practices" section of this paper.

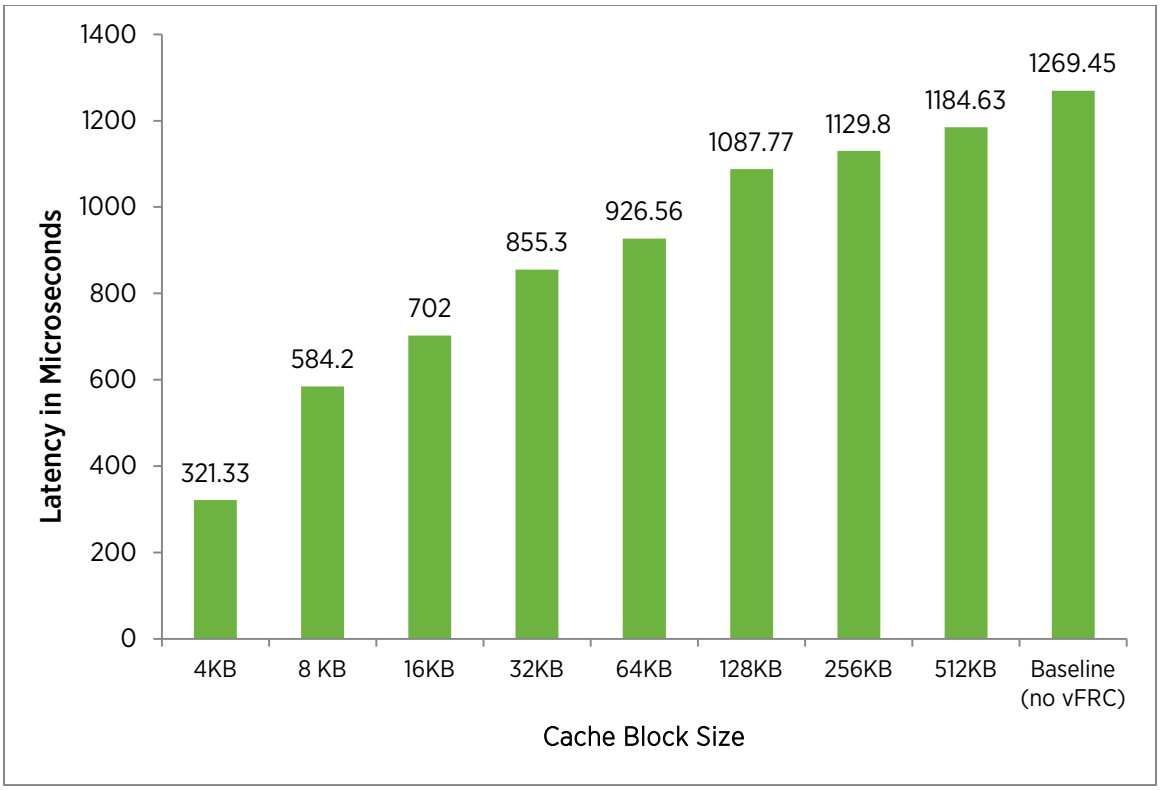

Figure 3. Impact of cache block size on application performance for the Hardware Monitoring Server workload

Flash Device Type

Not all flash devices perform the same way. On a higher level, PCIe flash cards perform very differently compared to a SATA/SAS SSD drive. PCIe flash cards usually perform many times better than a commodity SAS SSD drive. For example, a Micron P320h PCIe flash drive is rated to service a sustained random read IOPS of around 750K [1], while an Intel 6Gb/s SATA 320 SSD is rated to service at 39.5K IOPS for random reads [2]. Similarly, there are two basic types of flash devices: Single-Level Cell (SLC) and Multi-Level Cell (MLC). MLC packs more bits per cell and hence offers higher capacities, while SLC stores data in individual cells and therefore is expensive and has a smaller capacity. Consequently, SLC flash performs far better than MLC flash.

It is therefore important to pick the right flash device for your workloads after taking into account the cost versus the benefit of using any particular type of flash device. vFRC performance can vary across a wide spectrum depending on what flash device is used.

Performance Results

In this section, we provide performance results for some workloads that benefit from vFRC.

Decision Support System Database Workload (Swingbench DSS on Oracle 11g R2)

Decision Support Systems (DSS) [3] are a set of business applications and processes that provide answers in response to various queries regarding the business in order to help make key business decisions.

Test Bed

Swingbench DSS is a part of Swingbench 2.4 [4] that issues DSS-like queries on a schema named Sales History. The Swingbench benchmark program runs on a client virtual machine, which runs Red Hat Enterprise Linux 6.4. The backend database virtual machine is a Windows 2008 server running remotely in a different ESXi server. This virtual machine runs an Oracle 11g R2 database optimized for data warehousing applications. Swingbench creates the Sales History schema and populates it with data before issuing queries. The following figure shows the test bed setup.

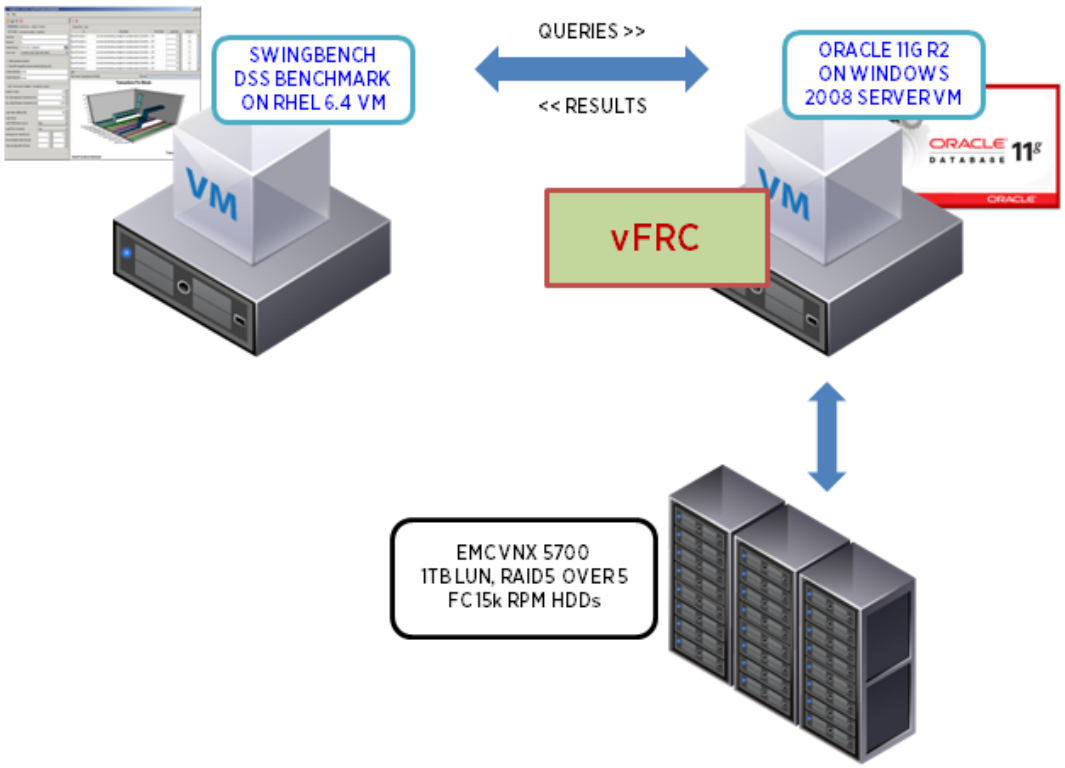

Figure 4. Swingbench benchmark test bed architecture

The backend database virtual machine consists of 8 vCPUs and 8GB memory with two virtual disks, a 60GB disk containing the operating system files and a 40GB eager-zeroed-thick VMDK for holding the database. The database VMDK was created on a 1TB RAID-5 volume consisting of 5 15,000 RPM Fiber Channel hard disks on an EMC VNX5700 storage array [5]. The flash device used for this run is an Intel SAS MLC 200GB SSD drive below an HP SmartArray P410 local RAID controller with 512MB on-board memory cache. The Sales History database is 15GB in size and an 8GB vFRC was configured with a default cache block size of 8KB. The cache block size was set as 8KB because this application predominantly issues 8KB I/Os.

Results

The Swingbench DSS benchmark provides metrics like total transaction count, transaction counts for each type of query, average transactions per minute, and response time information. Figure 5 shows a 47–145% improvement in terms of transaction count depending on the particular type of transaction.

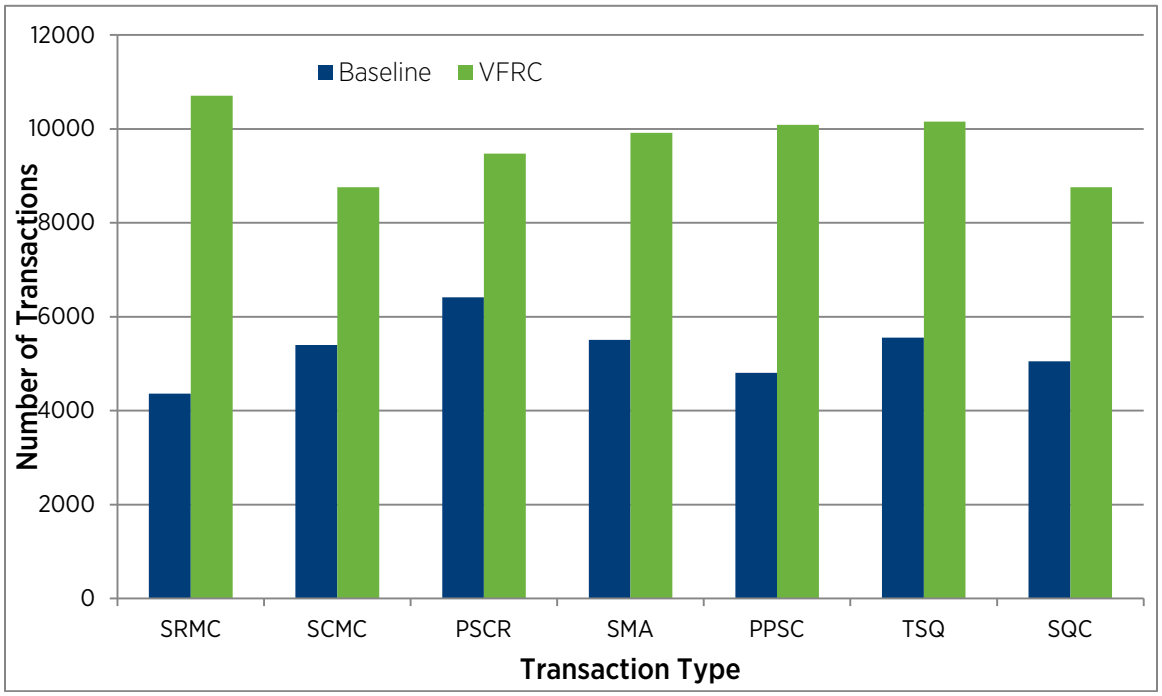

Figure 5. Transaction count for Swingbench DSS workload

Figure 6 and Figure 7 plot the transactions per minute (TPM) value for the vFRC-enabled case versus the baseline where there is no vFRC and I/Os are serviced by the storage array. Overall, both TPM and average response time metrics for the vFRC-enabled case are about 2x better than the baseline case.

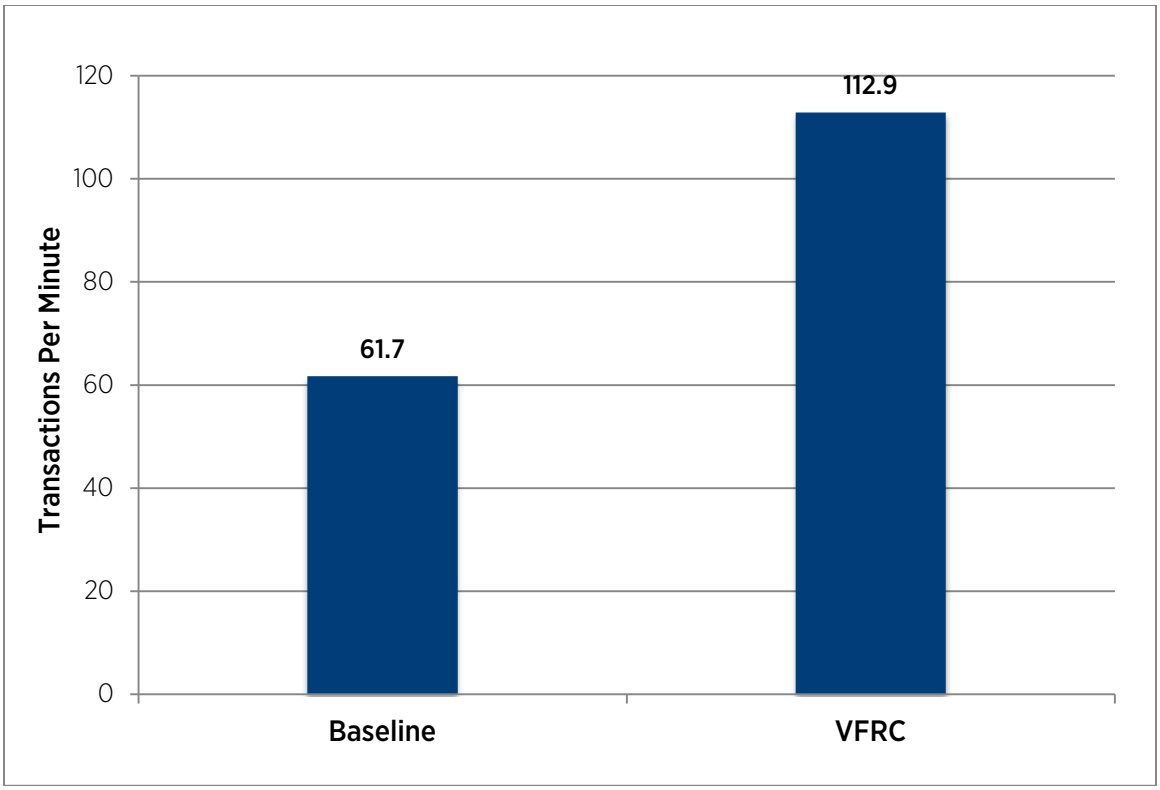

Figure 6. Swingbench DSS workload throughput comparison

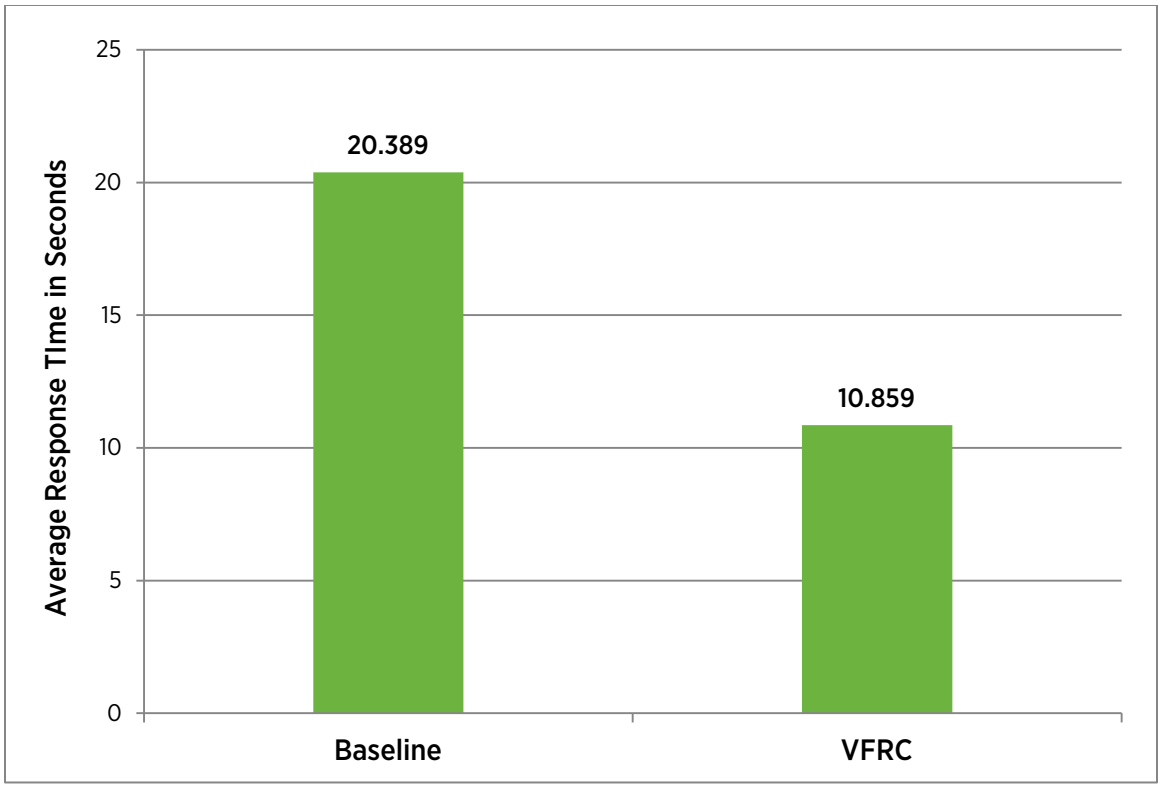

Figure 7. Swingbench DSS latency comparison

The performance improvement is primarily due to the high amount of repeated accesses to a smaller data footprint. The cache hit rate achieved during this run was about 89%.

DVD Store Benchmark (Microsoft SQL Server 2008)

DVD Store [6] is an online e-commerce workload generation application on a backend database. This workload is a type of database transaction workload with about a 60% read ratio. The access pattern is mostly random and the active working set covers almost the entire database.

Test Bed

The test bed consists of a single virtual machine that acts as both the client (workload generator) and the server (Microsoft SQL Server 2008 database). The virtual machine consists of 1 vCPU and 4GB memory with three virtual disks, a 40GB disk for the guest operating system, a 25GB disk for the database and a 10GB disk for database logs. The database size used was 15GB and the benchmark was run for 2 hours. A Micron PCIe flash card was used to create vFRC for the database VMDK. The backing storage array is an EMC VNX5300 and the volume is RAID-5 over 10 SAS 10,000 RPM hard disk drives. The workload was I/O bound with CPU utilization being consistently below 70% with a single vCPU.

Results

Figure 8 shows the "Orders per minute" metric from the benchmark, which is a measure of application throughput.

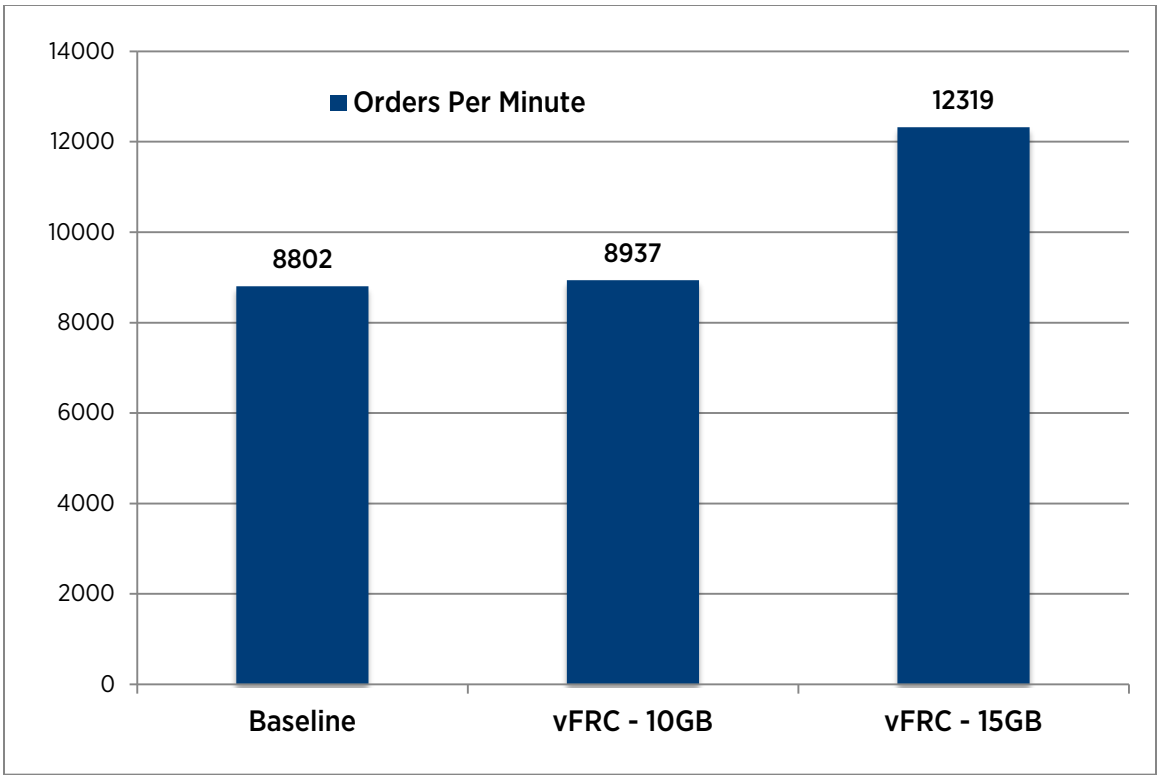

Figure 8. Throughput comparison of DVDstore benchmark

In Figure 8, the baseline case is when no vFRC is configured for the virtual machine. This means I/Os from the virtual machine go directly to the backend storage array. The other two cases are with vFRC enabled and configured with different sizes.

When the cache size is 10GB, vFRC performance is almost the same as the baseline and the improvement is very minimal. Given that the database is 15GB in size, even a 10GB flash cache doesn't improve the performance substantially because DVD Store issues mostly random I/Os covering the entire database. Therefore, there is very little block re-use in the workload to make caching useful. However, when the entire working set is brought to cache in the case of the 15GB vFRC size, we see about 39% improvement in orders per minute.

In general for online transaction processing (OLTP) workloads, the active working set spans almost the entire database and the workload is mostly random. Therefore vFRC would provide benefit in these cases when the cache size is configured carefully to hold the entire working set.

Accurate Replay of Enterprise I/O Traces

We consider two enterprise server-level I/O traces that are available publicly and are used extensively in storage research. These traces are collected from Microsoft Research Cambridge [7] and are also maintained in SNIA IO Trace Repository. These traces are a list of all I/O requests that were received by MSR's servers and we use these traces for performance evaluation by means of replaying all these requests in our setup while preserving the timing and access characteristics of the trace accurately using IOAnalyzer [8]. The IOAnalyzer virtual machine consists of 1 vCPU, 2 GB memory, and two eager-zeroed-thick virtual disks. The first VMDK holds the Ubuntu Linux operating system and the trace was replayed in the second VMDK, of size 100GB. Details about these traces are provided in Table 1.

METRICS	HARDWARE MONITORING SERVER	PROXY SERVER
Workload Description	Trace collected from servers that logs data from multiple hardware monitoring programs across a datacenter - collected at Microsoft	Web Proxy Server collected at Microsoft
Read Write Ratio	95% reads	67% reads
Total Number of requests	~600k	~5 Million
Dominant I/O Size	4KB	4KB

Table 1, Description of enterprise I/O traces used for performance comparison

Figure 9 shows the performance in terms of average request latency for the Hardware Monitoring Server workload and Figure 10 shows the performance benefits of vFRC for the Web Proxy Server workload. Both workloads exhibit read-intensive behavior and their access patterns are very well suited for vFRC. There is a high level of block re-use in these workloads as is evident from the high cache hit ratio obtained from vFRC statistics. The average per-request response time for these workloads has improved by 2–3x compared to the baseline when no vFRC is enabled.

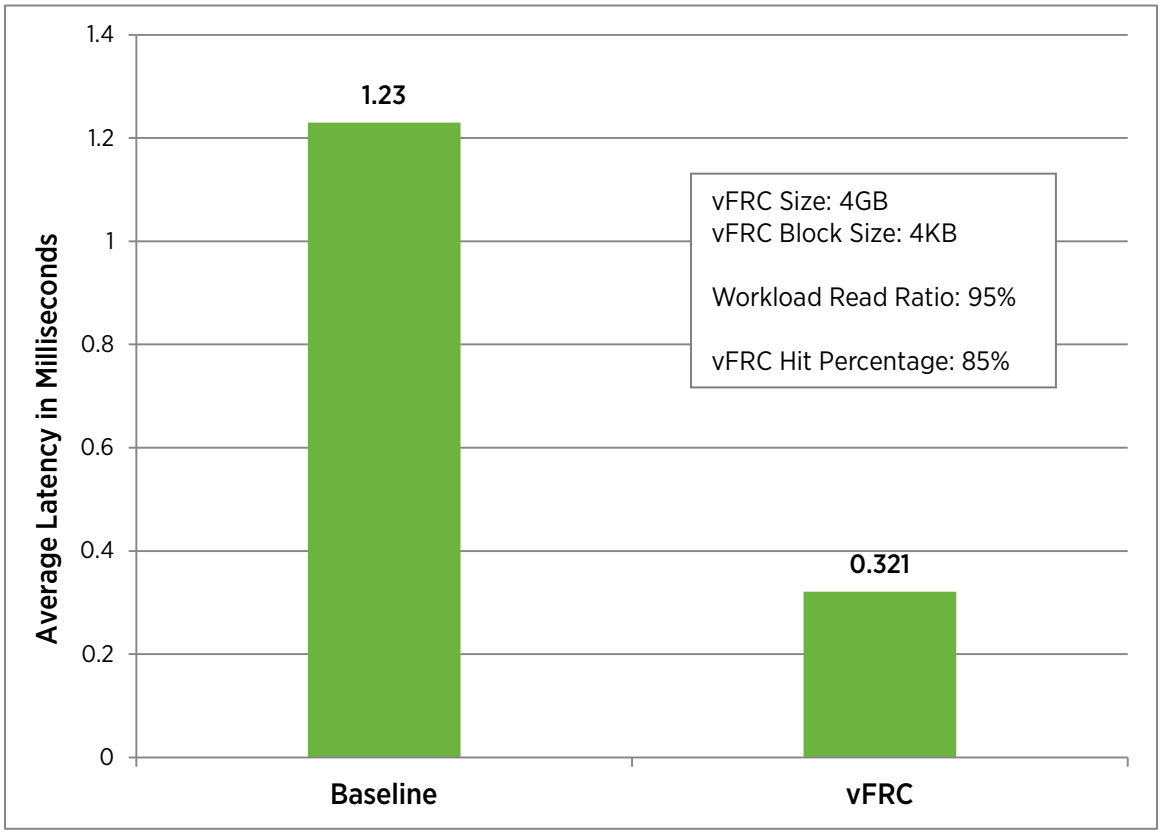

Figure 9. Comparison of average latency per request (Hardware Monitoring Server workload)

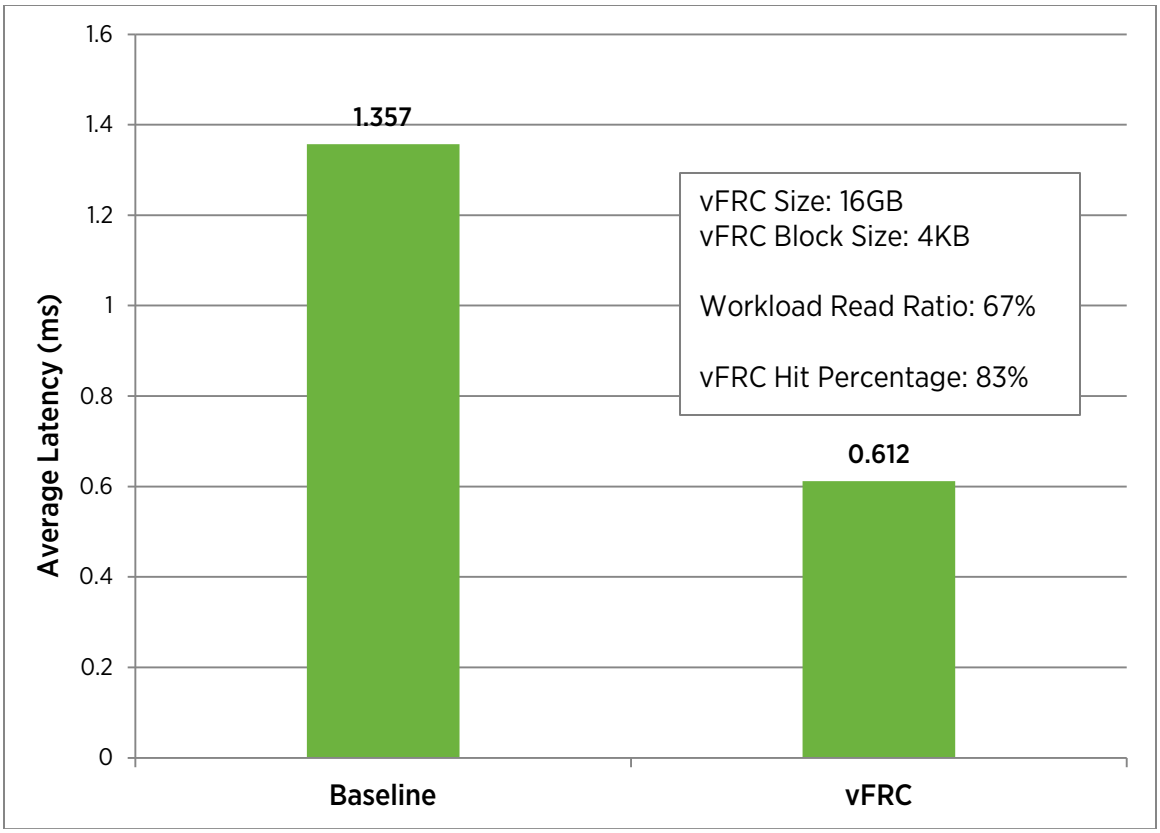

Figure 10. Comparison of average latency per request (Proxy Server workload)

Performance Best Practices

Setting the Correct Cache Size

A good understanding of the workload is required to set the optimal cache size. As discussed in the section "Performance Tunables," a less than optimal cache size leads to more evictions and fewer cache hits, while having a higher than optimal cache size impacts the time to migrate the cache during vMotion.

Ideally, the cache size should be just big enough to hold the repeatedly used blocks in the workload. We call this the *active working set*. However, it is non-trivial to obtain the active working set of the workload because typical workloads show variations with respect to time. The active working set may change over the course of the workload. Therefore, you can approximate the right cache size by following these guidelines:

- To start with, during vFRC creation, specify an approximate value, for example 20% of the database size or VMDK size.
- Collect vFRC statistics using esxcli to see cache utilization in real-time. vFRC statistics can be collected once the application passes the initial stage where the cache gets warmed up and the workload stabilizes. The *numBlocks* field in the statistics represents the total number of blocks in the cache when created. For example, if a 1GB cache was created and 8KB cache block size was used, this value will be 131072. The *numBlocksCurrentlyCached* field represents the number of blocks that actually hold some data.

- While running a workload, if *numBlocksCurrentlyCached* is less than *numBlocks,* this means that the cache is over-provisioned and it can be reduced.
- If the two fields *numBlocksCurrentlyCached* and *numBlocks* are equal, then the cache size may either be correct or under-provisioned.
- The *Evict:avgNumBlocksPerOp* field in the statistics represents the average amount of data that has been evicted so far. If this value is very high, there is a possibility that the cache size is under-provisioned. However, when the cache hit percentage value, represented by the field *vFlash:cacheHitPercentage,* is very high, then the cache size may just be right.
- At this point, when there are more evictions, try to increase or decrease the size of cache and monitor the change in evictions and cache hit percentages. After decreasing the cache size, if eviction increases and cache hit percentage decreases, then more cache size is required. Similarly, after increasing the cache size, if the cache hit percentage stays the same, then the cache size may be decreased.
- Such experiments with cache size while closely monitoring the vFRC statistics will help in settling on a reasonably optimal cache size. This must be done once for every new workload.

Setting the Correct Cache Block Size

As already covered in the section "Performance Tunables," the cache block size impacts vFRC performance. The best way to choose the best cache block size is to match it according to the I/O size of the workload. VscsiStats [9] may be used to find the I/O size in real-time when running the workload. This utility outputs an *IOLength* histogram that can be used to find the most dominant I/O size of the workload. The cache block size of vFRC can be configured to match this value. In general, vFRC performs better if the cache block size either matches or is less than the I/O size of workloads. However, configuring cache block size to be less than the dominant I/O size leads to increased memory consumption and more I/Os issued to the cache, possibly resulting in lower performance.

Choosing the Right SSD Device

vFRC performs best in PCIe flash cards compared to SAS/SATA SSD drives. Even among PCIe devices, the ones with a higher device queue depth like 256 perform better with vFRC because the device can handle more I/Os than a typical device queue depth of 32.

Cache Migration during vMotion

By default, vMotion of a vFRC-enabled virtual machine migrates all caches associated with the virtual machine. This feature helps in maintaining the warm cache even during and after the vMotion process. The application workload will therefore achieve the same amount of cache hit rate during vMotion. However the entire cache will be migrated over the network and therefore the time taken for vMotion will increase depending on the number of caches and the size of those caches. There is also an option to drop the cache during vMotion. If this option is chosen, the virtual machine migration happens without the cache contents, and after vMotion completes, the cache is warmed up again in the destination host. While this makes the vMotion time to be shorter, the application may see a dip in performance for a brief period of time until the cache gets warmed up again.

To choose the right policy for cache migration during vMotion, you must understand the trade-off between the policies. Cache migration maintains the cache contents without the application perceiving any temporary dip in performance, while increasing the vMotion time and consuming network bandwidth. Whereas, dropping the cache makes vMotion complete faster, while the application may have temporary performance degradation until the destination cache gets warmed up again. Choose the right policy based on the criticality of consistent application performance, utilization of network bandwidth, and the expected duration of the vMotion process.

Conclusion

In this paper, we present an overview of vSphere Flash Read Cache architecture along with the read-write workflow of vFRC, and various tunables in hardware and software that can have a significant impact on vFRC performance. We show the performance results for database workloads and some widely used enterprise server I/O traces. Finally, we provide some performance best practices about setting the cache size, cache block size, and choosing the right kind of flash device. Our test results show that vFRC can help improve the performance of certain applications by a factor of 2–3x, but in order to achieve good performance results with vFRC, you need a good understanding of the workload so you can correctly configure the cache.

References

[1] Micron P320h PCIe Flash Data Sheet
http://www.micron.com/my/login?returnUrl=http://www.micron.com/parts/solid-state-storage/ssd/mtfdgal175sah-1n3ab

[2] Intel 320 SATA SSD Data Sheet
http://www.intel.com/content/dam/www/public/us/en/documents/product-specifications/ssd-320-specification.pdf

[3] Decision Support Systems (DSS)
http://www.journals.elsevier.com/decision-support-systems/

[4] Swingbench 2.4
http://www.dominicgiles.com/swingbench.html

[5] EMC VNX 5700 Storage array data sheet
http://www.emc.com/collateral/software/specification-sheet/h8514-vnx-series-ss.pdf

[6] DVD Store Benchmark
http://en.community.dell.com/techcenter/extras/w/wiki/dvd-store.aspx

[7] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. 2008. Write off-loading: Practical power management for enterprise storage. Trans. Storage 4, 3, Article 10 (November 2008), 23 pages. DOI=10.1145/1416944.1416949
http://doi.acm.org/10.1145/1416944.1416949

[8] IOAnalyzer 1.5.1
http://labs.vmware.com/flings/io-analyzer

[9] vscsiStats
http://communities.vmware.com/docs/DOC-10095

About the Author

Dr. Sankaran Sivathanu is a senior engineer in the VMware Performance Engineering team. His work focuses on the performance aspects of the ESXi storage stack and characterization/modeling of new and emerging I/O workloads. He has a PhD in Computer Science from the Georgia Institute of Technology.

Acknowledgements

The author thanks Edward Goggin, Julie Brodeur, Kiran Madnani, Shilpi Agarwal, Thiruvengada Govindan Thirumal, and Todd Muirhead for their reviews and contributions to the paper.

VMware, Inc. **3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com**