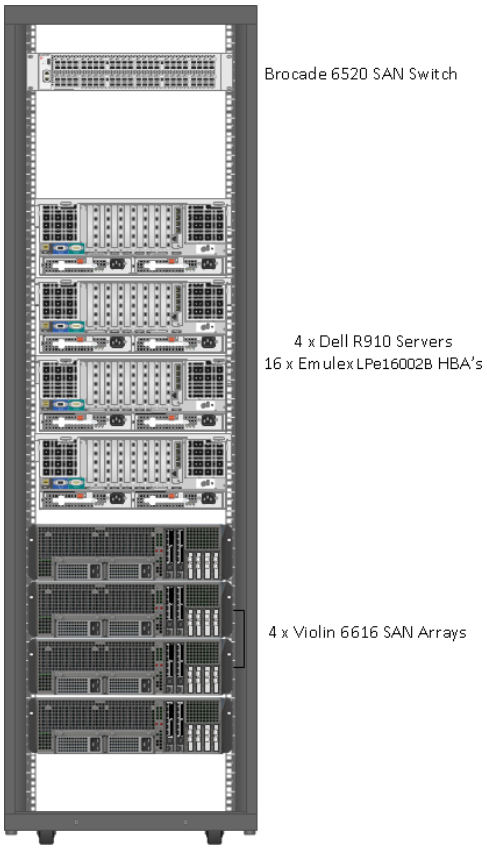
Lab Report: Designing a 2 Million+ IOPS Architecture
Brocade, Dell, Emulex and Violin Memory Systems Deliver 2 Million IOPS

Submitted By: George Crump – Lead Analyst Storage Switzerland
September 30, 2013

Executive Summary

Storage Switzerland was recently commissioned by Brocade, Dell, Emulex and Violin Memory to audit a lab test in which a storage infrastructure was developed that delivered more than two million 8-kilobyte I/O operations per second (IOPS). There are two key takeaways from this lab test. First, generating high performance requires an infrastructure, not just a fast storage device. Second, this type of performance testing is relevant to all data centers, even those that need more modest performance.

Testing Environment



The test environment consisted of four Dell PowerEdge R910 servers, each with four Emulex LightPulse Gen 5 Fibre Channel (FC) LPe16002B Host Bus Adapters (HBAs). Those cards were used to connect the servers to a single Brocade Gen 5 Fibre Channel 6520 Switch. The entire setup fit inside a standard rack with room to spare.

Brocade switches and Emulex HBAs leverage Gen 5 Fibre Channel with 16 Gbps performance. Gen 5 Fibre Channel is the purpose-built networking solution for flash storage, enabling breakthrough application performance, scalability, and availability.

Four Violin 6616 Flash Memory SAN Arrays were connected to the Fibre Channel Switch. The environment was tuned for performance; cost, while relevant, was a secondary concern. Clearly there are some environments that need and can justify this kind of 'extreme' configuration, but most probably cannot. The report will cover the importance of such a test to the latter group.

Testing Parameters

The standard benchmarking suites available were not designed for this type of high performance system. These tools expect storage systems comprised of hard drives and require configurations that are only logical in a hard drive-based world. Since the focus of the test was raw performance, Flexible I/O Tester (FIO), a standard and easy to access testing tool was selected as it is better suited to raw performance testing. A variety of jobs were run that ranged from 100% reads to various

read/write mixes. This report will detail a number of these scenarios because no two workloads are identical and the differences are important.

Overall, the results ranged from 2.3 million IOPS on 100% random read tests to a sustained 1.7 million 8-kilobyte IOPS with a 70/30 read/write mix.

Testing Infrastructure

The fact that the Violin Flash Memory Arrays are fast was not a surprise. The key takeaway was learning how important the environment was to delivering that performance. The report details the results of the testing with fewer arrays but it was the Gen 5 Fibre Channel infrastructure that allowed the creation of a shared environment that generated the combined IOPS result.

Powerful Dell servers were needed so that FIO could generate the data streams required to stress the environment. The Emulex HBAs played an obvious role by getting that data off the servers and onto the network, and probably most importantly, the Brocade switch made sure that traffic through the network allowed for optimal performance of the Violin arrays. Data was both read and written so the importance of the network cannot be overstated.

The test supports Storage Switzerland's long-standing position that performance tuning requires a holistic approach. The move to all-flash arrays has to be justified by delivering performance that enables the business to innovate. Putting very fast memory-based storage into a sub-optimal infrastructure will not deliver the return on investment (ROI) needed to justify the cost of flash. The whole environment should be flash optimized, not just the storage system.

Implementing the right network infrastructure ensures that when introducing flash performance into the storage layer new bottlenecks within the network layer do not occur. Brocade ensures flash storage speed is delivered fast, by doubling transfer rates from Gen 4 speeds and minimizing latency with its cut-through network architecture. Gen 5 Fibre Channel combines these attributes of low latency and unmatched IOPS to maximize application performance with flash storage. Extreme performance enables the deployment of more servers, desktops, and OLTP workloads without sacrificing reliability in the virtualized data center.

Testing Relevance

The most important part of the report is to explain the relevance of a system that can deliver two million IOPS. While extreme IOPS benchmarks like this sometime come under critisim but to be sure there are practicle real world use cases where this level of IOPS is required, High Perfromance Computing (HPC), and High Frequency Trading (HFT) are just two relevant examples that come to mind.

In fairness, many other environments don't need two million IOPS of performance explaining the relevance to those environments is a critical mission of this report. Without this explanation, the test was just an interesting science experiment, not a practical application of flash technology.

Most users are under the impression that storage system vendors have vast labs filled with technicians just running tests all day. In reality, this is rarely the case. Businesses in today's competitive climate are lean and focused on their customers as well as the projects that are most pressing to them. It is rare when the industry can pause long enough to go through the process of testing the limits of these devices, especially devices that can generate this level of performance.

Companies are certainly testing performance but this level of performance with this assembly of components is more rare. Doing so teaches manufacturers more about the technologies and pushes the envelope to benefit the products and storage designs the majority of their customers *will* buy.

The result is that tests like this become significant opportunities to verify what is theoretically possible and to learn when those theories aren't accurate. As noted above this test verified the importance of Gen 5 Fibre Channel infrastructure as well as the raw performance potential of the Violin Arrays.

Storage Swiss Take

The key takeaway from Storage Switzerland's audit of the Brocade, Dell, Emulex and Violin Memory lab test is that an extreme performance exercise can provide something for every data center. For those that need millions of IOPS (HPC, HFT and others), a system can be built to that level of performance. For those that don't, there is now a better understanding of optimal performance design at the 500K IOPS level.

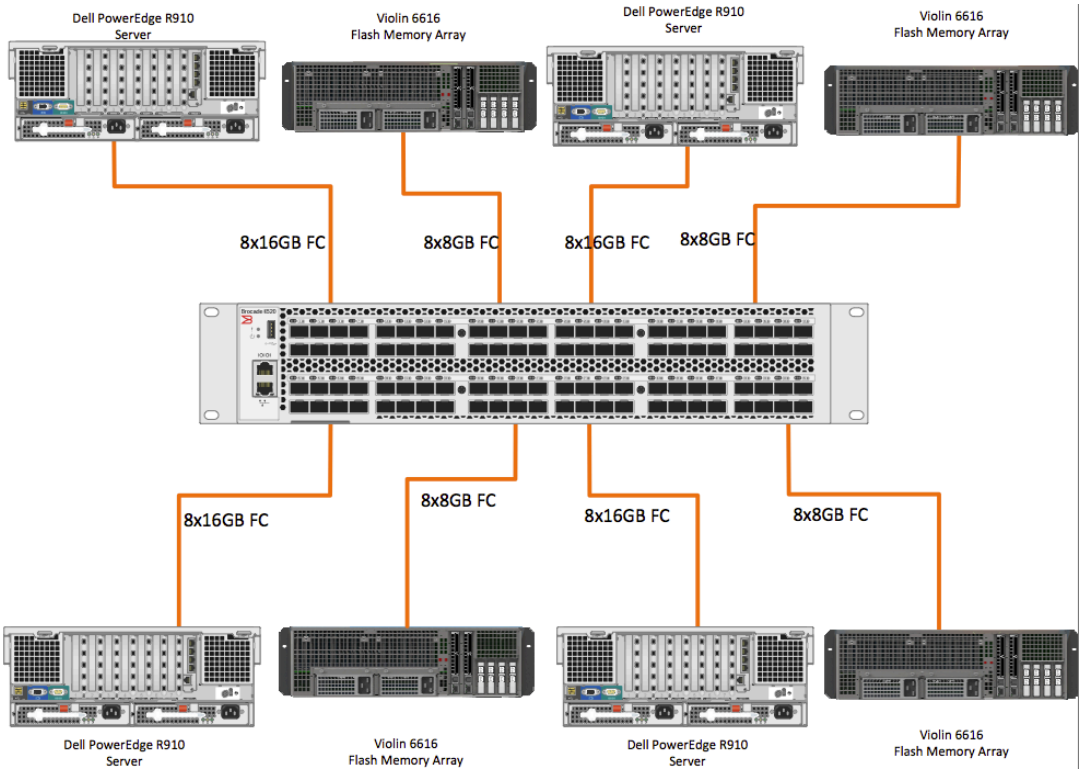
Lab Report: Designing a 2 Million+ IOPS Architecture

Benchmarks that generate over a million IOPS are not uncommon in the storage industry today but most of these tests are done in configurations like directly attached storage with illogical disk arrangements. These benchmark configurations would not be used by the typical data center. Two million-plus 8-kilobyte IOPS tests are much more rare, especially in a shared configuration.

A shared storage configuration was a key component of the testing because of the preponderance of application and hypervisor clusters in data centers today. Shared storage and its dependent infrastructure is a requirement. As a result the journey to a two million 8-kilobyte IOPS design resulted in valuable lessons for data centers that need a more common 500k or fewer IOPS.

As evident in this report, the simplicity of the storage and network infrastructure's design allows performance to scale despite the workload and read-write mix. This, in essence, gives business the flexibility to expand rapidly without needing wholesale infrastructure upgrades if the initial layout is planned and implemented correctly.

Logistics



Servers (I/O Generation)

Performing a test of this magnitude takes planning and cooperation between diverse companies. The tests were performed at Violin Memory's lab facility. The free and commonly available tool FIO was chosen to generate the I/O need to reach the two million 8k IOPS goal. While simple in use FIO provides the flexibility of I/O pattern shaping to simulate various kinds of data traffic.

Four Dell PowerEdge R910 servers were selected to run FIO. These powerful servers were required to generate enough I/O so that the IOPS goal could be met while still creating a realistic performance configuration. These servers were each configured with 512GB RAM and used four Intel E7-4870 processors, providing a total of 80 cores to drive I/O to the arrays.

Red Hat Enterprise Linux 6.4 and the native Linux SAN multi-path software were installed on each of the Dell PowerEdge R910s. The servers were configured to Violin Memory's recommendations for multipath.conf.

The role of the servers was simple: to generate as much I/O workload as possible. But to achieve that I/O requires a diligence in design. Making sure that the internal I/O bus of the server did not interfere with the movement of data in and out of the system was critical.

SAN Interconnect (I/O Delivery)

Inside each of these servers, four Emulex LightPulse LPe16002B Gen-5 Fibre Channel HBAs were installed. Gen 5 Fibre Channel HBAs are capable of providing up to 16 Gbps of bandwidth. The four HBAs were balanced across the four I/O controller hub (IOH) modules in the Dell servers. The Emulex HBAs were implemented so that each one had its own IOH.

With a performance demand this extreme even the I/O bus could be a bottleneck, distributing the load across I/O hubs eliminated that problem. In smaller systems with fewer IOHs, similar care should be taken to distribute the I/O load across all available hubs.

The Emulex HBAs and Dell servers were tuned and optimized according to manufacturer best practices by setting optimal HBA queue depths, customized udev rules (udev is a device manager for Linux kernals) for the Violin arrays and automatic CPU balancing of the Emulex HBAs. (See Appendix D)

Emulex product information can be found at the URLs below:
http://www.emulex.com/products/fibre-channel-hbas.html
http://www.emulex.com/artifacts/48b73ec7-7672-441f-ad3c-a5e5540b451c/elx_cr_all_hbaproductline.pdf

Connecting the above servers to the storage was the responsibility of a single Brocade Gen 5 Fibre Channel 6520 96-port switch. This 16 Gbps switch has multiple ASICs to control traffic flow to the servers via their HBAs distributed across these ASICs to once again distribute I/O load. There is one ASIC per port group. The system was configured so that the appropriate HBA and Violin flash Memory Array were on the same port group to reduce latency and IO conflict.

The Brocade switches provide a cut-through architecture which was another key point in this test. With a cut-through architecture the switch has to examine only the first 128 bytes of the packet header to determine where the packet needs to be sent. The packet can then be forwarded to the destination before the switch receives the entire packet, resulting in very low latency.

Once again, in smaller systems distribution of performance demanding loads across switch ASICs is important to deliver maximum performance.

All of the ports on the switch were included into a single Fibre Channel zone. Port masking was done at the array to control host access to the LUNs.

It is important to reiterate that a single Brocade switch was all that was needed for this test to meet its goals. In a more traditional data center configuration, where a minimum of two switches would have been used for high availability, even more I/O could have been supported at the environment's core.

Brocade product information can be found at the URLs below:
http://www.brocade.com/products/all/switches/product-details/6520-switch/index.page
http://www.brocade.com/forms/getFile?p=documents/data_sheets/product_data_sheets/6520-switch-ds.pdf

SAN Storage (I/O Turnaround)

The SAN storage consisted four Violin Memory 6616 SAN arrays. It is important to note the holistic nature of this design. Installing the Violin Memory 6616 into a non-optimized infrastructure would have likely improved performance but certainly would not have allowed the arrays to achieve maximum performance. Only by being a member of a highly tuned storage infrastructure were these arrays allowed to reach their full potential.

The Violin Memory 6616 SAN arrays were configured with 64 x 256GB SLC DIMMs and were formatted at 65% of capacity to provide the best balance of capacity and performance. The extra capacity allowed the Violin Memory array to maintain consistent performance while keeping pace with garbage collection routines, important when the array was stressed with heavy write traffic. The total useable capacity of the configuration was 31.3 TBs, approximately 8 TBs per array.

More information about Violin 6616 SLC Flash Memory arrays can be found at these URLs:
http://www.violin-memory.com/products/6000-flash-memory-array/
http://www.violin-memory.com/wp-content/uploads/Violin-Datasheet-6000.pdf

Each server was presented with a total of 16 LUNs, each having four paths to the array using a round-robin load balancing algorithm within the multi-path software. Round-robin is a typical setting for optimizing storage performance in clustered application and clustered hypervisor environments.

The 16 LUNs on each host were allocated across the four arrays so each array was involved in each server's I/O workload. In the largest tests utilizing four host servers and four arrays, 64 LUNs were active. Each LUN exported from the Violin arrays was further balanced across the HA port pairs on the arrays and the HBA port pairs on each host.

Test Capture

Violin Symphony was used to capture the test results . This software displays the aggregate performance of all the arrays in near real-time. Violin Symphony provides a single pane of glass view into the Violin Memory storage infrastructure. It provides customized dashboards, in-depth performance and health monitoring, automated operations and a consolidated management interface across hundreds of Violin Memory arrays.

Design Summary

The design of the test environment was optimized to spread the load as intelligently as possible across the available components. This is a key lesson for any sized environment: make sure that all the ports and I/O hubs are leveraged equally in the design so that there is limited overlap between performance-demanding application I/O paths. Only share the I/O path with less demanding applications.

The environment, with the exception of the switch, was highly redundant. The switch had redundant components throughout and was only exposed to an unlikely catastrophic failure. A second switch would have provided that redundancy plus potential for additional load balancing.

Again, while this design was intended to set a record, there was nothing "unnatural" about the design and it would be suitable, with adjustments for performance needed, in any sized data center.

Testing

Testing was done with the benchmarking tool Flexible I/O Tester, more commonly known as FIO. FIO is a tool that can spawn a number of threads or processes performing a particular type of I/O action as specified by the user. It allowed our test to perform read-only, write-only and mixed workloads at a very high rate.

FIO also eliminates the potential bottleneck and latency that an application or even other benchmarking tools may introduce into the test bed. Again, the purpose of this test was to design an environment that could sustain more than two million IOPS and, not to troubleshoot other applications. While these are real-world concerns and part of the holistic conversation, they require the inclusion of application owners. The goal of this test was to focus on infrastructure design.

Multiple hosts executing FIO against 1-4 Violin Memory arrays were tested. The results were captured to demonstrate how the shared storage infrastructure and the storage systems themselves scaled IOPS and bandwidth while still displaying low latency.

It is noteworthy that reported latencies for 4-array tests were between 0.2 to 0.71ms. These latencies are approximately 90% lower than traditional storage. The benefits of these lower latencies are:

- Significantly less I/O wait
- Improved application performance
- Greater workload density as measured by threads or virtualized workloads.

An example parameter file for FIO is available in Appendix A. It contains the list of devices for the I/O workload to be run against for a given host as well as the number of concurrent jobs and outstanding queue depth.

During the testing, optimal values for each configuration were established for each variation, depending on number of host/array tests. Twenty concurrent FIO jobs were found to be the optimal number to allow for all the processes to run without going into a wait state during the test jobs. The Linux TOP command was run during the test runs to confirm.

Test Results

Using FIO, the test jobs performed ranged from one server with one array to four servers with four arrays. The IOPS of the combined storage system is what was reported. Also, tests ranged from 100% reads to a mixed read/write workload. Finally, two block sizes were used, 4K and 8K. The granularity of the testing should allow an IT planner to select the read/write mix and block size that most closely matches their data center.

The job-by-job test results can be found in Appendix B. In summary those tests will show a range of performance from 399K IOPS (three servers, one array, 60% writes), to 2.3 million IOPS (four servers, four arrays, 100% random reads) and 2.9 million IOPS (four servers, four arrays, 100% sequential reads).

It is also important to note that several times the aggregate bandwidth exceeded 17GBps, or the equivalent of 4 DVDs per second. Often the focus of all-flash storage systems is IOPS performance, but they can also deliver excellent bandwidth. This testing showed that the infrastructure and the memory arrays themselves were capable of very high throughput as well as IOPS.

While plenty of random read-heavy environments exist, many data centers require a mix of random reads and writes. For those data centers the results of the four server-four array configuration with 70%/30% random read/write load may be the most interesting. This test came in at 1.7 million IOPS. This worst case test, for flash storage, also highlights the importance of flash array design. With this heavy of a workload the flash array needs to have plenty of flash controller intelligence to maintain performance while dealing with flash management functions like garbage collection.

Objective Attainment

IOPS

The clear focus of this test was not just to achieve an attention-grabbing IOPS result, although it did that. The objective was also to understand how architectures need to be crafted for maximum performance. When a data center has an application or environment capable of generating hundreds of thousands of IOPS, the entire infrastructure must be examined. The storage system must of course be designed to handle the type of workload required. Violin's array showed the ability to handle a wide range of I/O types and sizes. Then the infrastructure must be structured in a way that I/O can be distributed as broadly as possible. This test showed that Brocade's switch and Emulex's HBAs were able to do exactly that.

Scale

These tests also showed that the this environment was designed to scale. No changes were required in the network infrastructure configuration as it scaled IOPS. Again, focusing on the most difficult test, 8K blocks with 60% random writes and 40% random reads, showed that as each Violin array was added the infrastructure (servers, HBA, switch) were able to scale to deliver that performance. In this test one array generated 399K IOPS, two generated 777K IOPS, three generated 1.3 million IOPS and four generated 1.6 million IOPS.

Relevance

It is important to note that there are environments that clearly need high performance; the need for 2M IOPS should not be overlooked. Remember that just a few years ago, benchmarks that achieved 300k+ IOPS were considered unrealistic, and today, applications easily surpass that requirement.
The number of data centers that need millions of IOPS will increase steadily. This will be driven by the following factors:

1. Virtualized servers and private cloud deployments- as customers begin to leverage their Fibre Channel infrastructure for private cloud implementations, there will be an increase in the workloads demanding access to fast shared storage. The value of using Violin flash Memory Arrays with Gen 5 Fibre Channel increases the VM density per physical server for greater server consolidation.

2. VDI - housing desktop images and swap files on shared storage will dramatically increase the traffic patterns and demand for IOPS. This architecture supports a higher number of virtual desktops and overcomes performance challenges such as boot storms and traffic spikes common in VDI environments.

3. Scale-up databases - the proliferation of mobile devices with new applications that leverage social media and related chat attributes will result in high volumes of

small packet transactions to back-end data stores. The speed of flash memory arrays and Gen 5 Fibre Channel increases transaction rates, reduces response times and enables hyper-scale OLTP and OLAP deployments for data-driven applications.

Even for more mainstream environments that are years away from needing millions of IOPS, there is value in understanding the ramifications of this test. It emphasizes the importance of design, strategic placing of hardware components and the realization of IO mix. The appendix shows a more modest set of configurations, which provide relevance for data centers that may not need two million IOPS today.

For example: the 30% random write, 70% random read test done with a single array and three servers generated 457K IOPS; a realistic number that more and more data centers will need in the coming year. These numbers allow new levels of virtual machine density which can deliver new ROI in virtual server, virtual desktop and clustered database environments.

It is Storage Switzerland's belief that performance beyond 400K will quickly become a standard requirement as virtual desktop and virtual server infrastructures continue to increase in density (number of virtual instances per physical host).

A related point is that while a specific application may not use two million IOPS, the underlying storage array may still need to deliver far more IOPS than the demand of a single application. This is because it is common for multiple applications and test and development environments to access the same, centralized SAN. In these cases, having additional storage bandwidth (16 GB/s) both accelerates these other operations and ensures that the impact on the primary application is minimized. With this set-up, one terabyte of data can be read in about one minute, making these I/O intensive tasks both faster and less impactful on normal operations.

Obviously, the smaller environment will require less memory arrays, smaller switches, less HBAs and fewer hosts and the cost of a 500K IOPS system quickly becomes affordable to the mid-sized data center. Most importantly, the opportunities this kind of performance presents to that data center can be significant. The ability to scale up databases and hypervisor hosts can eliminate physical server sprawl.

As a final note, many recent product announcements tout the introduction of various all-flash storage arrays. Most of these all-flash arrays peak at 150K 8-kilobyte IOPS, making them attractive for today’s mid-market range. While these arrays appear to have lower costs per gigabyte than the high-performance Violin flash Memory Array tested for this report, they do have higher total cost of ownership once long-term sustainability and scalability is factored into the equation.

Storage as the Enabler

In most clustered environments, storage is seen as the bottleneck. It limits how many users and applications can support and how many virtual instances a hypervisor can support. Based on our research, a properly designed, high performance infrastructure can support 5 to 10 times as many users or virtual instances per physical server/host compared to a typical environment. The ability to eliminate future host purchases or at least slow the rate of host deployment can mean a much more cost-effective and easier-to-manage application or hypervisor environment.

Storage Made Simple

One of the key storage challenges is the complexity in designing and, more importantly, tuning an environment in order to respond to the changing requirements of applications and hypervisors. Another key takeaway of this test is the scalability of the design. As the FIOs operation load was scaled, no changes to the environment were needed.

It eliminates the need for a "storage performance tuning specialist" to get involved when applications are deployed or expanded. Complex designs that distribute storage across memory-based and hard-disk based storage devices are replaced by a single set of policies. LUNs are assigned to hosts, and for the most part, no future tuning is needed.

Summary

The goal of this test was to set a new standard for IOPS, which it does. For HPC and HFT environments that are always in search of more IOPS, that search may be over. The result of achieving that goal though has far reaching implications for data centers that don't need millions or even half a million IOPS. It clearly demonstrates that the design of the storage infrastructure is a critical component to making sure a flash investment achieves its full potential and corresponding ROI. The lessons are applicable to data centers that need a few hundred thousand IOPS of performance as well as those that need millions.

Appendix A – FIO Parameter File

An example FIO parameter file for four host/four array workload is as follows

```
[global]
ioengine=libaio
direct=1
numjobs=20
iodepth=20
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_7BA0508D25903670
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_D96F714A875F17B7
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_4FC19BF011F1FD0D
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_7BA0508D83C53628
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_D96F714A210A17EF
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_A1B480EDFF84E610
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_4FC19BF0B7A4FD55
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_A1B480ED59D1E648
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_D96F714A516C6752
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_D96F714A741575D6
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_A1B480ED29B796F5
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_7BA0508DF3A34695
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_4FC19BF0C7C28DE8
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_7BA0508DD6DA5411
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_4FC19BF0E2BB9F6C
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_A1B480ED0CCE8471


[seq-read-8k]
wait_for_previous
description=Sequential Read 8K
ramp_time=120
runtime=800
rw=read
bs=8K
time_based

[ran-read-8k]
wait_for_previous
description=Random Read 8K
ramp_time=120
runtime=1800
rw=randread
bs=8K
norandommap
time_based

[mixed-read-8k]
wait_for_previous
description=Mixed I/O 4K
ramp_time=120
runtime=1800
rw=randrw
rwmixread=70
bs=8K
norandommap
time_based

[mixed-read-8k]
wait_for_previous
description=Mixed I/O 4K
ramp_time=120
runtime=1800
rw=randrw
rwmixread=40
```

```
bs=8K
norandommap
time_based
```

When the benchmark was against less than four arrays, the fio parameter file was edited as follows

```
[global]
ioengine=libaio
direct=1
numjobs=20
iodepth=40
#filename=/dev/mapper/SVIOLIN_SAN_ARRAY_7BA0508D25903670
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_D96F714A875F17B7
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_4FC19BF011F1FD0D
#filename=/dev/mapper/SVIOLIN_SAN_ARRAY_7BA0508D83C53628
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_D96F714A210A17EF
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_A1B480EDFF84E610
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_4FC19BF0B7A4FD55
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_A1B480ED59D1E648
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_D96F714A516C6752
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_D96F714A741575D6
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_A1B480ED29B796F5
#filename=/dev/mapper/SVIOLIN_SAN_ARRAY_7BA0508DF3A34695
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_4FC19BF0C7C28DE8
#filename=/dev/mapper/SVIOLIN_SAN_ARRAY_7BA0508DD6DA5411
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_4FC19BF0E2BB9F6C
filename=/dev/mapper/SVIOLIN_SAN_ARRAY_A1B480ED0CCE8471
```

Appendix B – Detailed Job Results

Four Array Results

Workload	# Arrays	IOPS (K)			Latency MS	Bandwith GB/s
		Read	Write	Total		
4k 100% seq read	4	2980	0	2980	0.234	11.38
8k 100% seq read	4	2304	0	2304	0.414	17.17
4k 100% random read	4	2960	0	2960	0.215	11.28
8k 100% random read	4	2330	0	2330	0.402	17.53
8k 60/40 W/R	4	637	956	1593	0.712	12.13
8k 30/70 W/R	4	1200	515	1715	0.674	13.08

One, Two, and Three Array Results (* Tests run against a single array utilized only three of the hosts.)

Workload	# Arrays	IOPS (K)			Latency MS	Bandwith GB/s
		Read	Write	Total		
4k 100% seq read	1*	870	0	870	0.100	3.32
4k 100% seq read	2	1720	0	1720	0.195	6.55
4k 100% seq read	3	2530	0	2530	0.216	9.6
8k 100% seq read	1*	525	0	525	1.950	4.02
8k 100% seq read	2	1080	0	1080	1.400	8.31
8k 100% seq read	3	1630	0	1630	0.575	12.53
4k 100% random read	1*	838	0	838	0.100	3.2
4k 100% random read	2	1690	0	1690	0.151	6.45
4k 100% random read	3	2550	0	2550	0.200	9.55
8k 100% random read	1*	558.7	0	558.7	0.169	4.28
8k 100% random read	2	1080	0	1080	0.231	7.93
8k 100% random read	3	1559	0	1559	0.265	12.38
8k 60/40 W/R	1*	160	239	399	2.490	3.04
8k 60/40 W/R	2	311	466	777	6.220	5.95
8k 60/40 W/R	3	512	769	1281	0.742	9.77
8k 30/70 W/R	1*	320	137	457	0.313	3.5
8k 30/70 W/R	2	468	248	716	0.628	5.46
8k 30/70 W/R	3	912	390	1302	0.641	10.16
8k 30/70 W/R	4	1200	515	1715	0.674	13.08

Appendix C – Violin Multipath.conf settings and udev rules

```
## Use user friendly names, instead of using WWIDs as names.
defaults {
    max_fds     262144
    user_friendly_names no
}

blacklist {
    device {
        vendor "ATA"
        product "WDC"
    }

    device {
        vendor "ATA"
        product "LOGICAL VOLUME"
    }
    device {
        vendor "DELL"
        product "PERC H700"
    }
#    devnode "^sd*"
}
# blacklist_exceptions {
#    devnode "^sd([x]|ab)"
#    devnode "^sd([t]|af)"
# }

devices {


  device {
      vendor         "VIOLIN"
      product         "SAN ARRAY"
      path_grouping_policy  group_by_serial
      getuid_callout     "/sbin/scsi_id --whitelisted --replace-whitespace --page=0x80 --
device=/dev/%n"
      path_checker      tur
      path_selector      "round-robin 0"
      hardware_handler    "0"
      features        "0"
      failback        immediate
      rr_weight        uniform
      no_path_retry      fail
      rr_min_io        1
      fast_io_fail_tmo    5
      dev_loss_tmo      1
 }
}
```

/etc/udev/rules.d/12-violin.rules

```
# UDEV rules for RH6 and OL6 with Violin Memory

# Set scheduler and queue depth for Violin SCSI devices
KERNEL=="sd*[!0-9]|sg*", BUS=="scsi", SYSFS{vendor}=="VIOLIN", SYSFS{model}=="SAN
ARRAY*", RUN+="/bin/sh -c 'echo noop > /sys/$devpath/queue/scheduler && echo 4096 >
/sys/$devpath/queue/nr_requests'"

# Set rotational, scheduler and queue depth for Violin multipath devices
KERNEL=="dm-[0-9]*", ENV{DM_UUID}=="mpath-SVIOLIN*", RUN+="/bin/sh -c 'echo noop >
/sys/$devpath/queue/scheduler && echo 32768 >
```

Appendix D – Emulex Tuning parameters

/etc/modprobe.d/ elx-lpfc.conf

```
# This file is used to configure lpfc parameters and aliases
# Emulex lpfc options
options lpfc lpfc_lun_queue_depth=128 lpfc_use_msi=2 lpfc_fcp_wq_count=16
lpfc_fcp_eq_count=7
```

/etc/init.d/set_smp_affinity_lpfc – This is enabled via chkconfig to run at startup

```
#!/bin/bash
#
# set_smp_affinity_lpfc
#
# chkconfig: 2345 55 25
# description: workaround for SMP CPU affininty issue on RHEL6
#

### BEGIN INIT INFO
# Provides: set_smp_affinity_lpfc
# Required-Start: $local_fs $network $syslog
# Required-Stop: $local_fs $syslog
# Should-Start: $syslog
# Should-Stop: $network $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: workaround for SMP CPU affininty issue on RHEL6
# Description:    workaround for SMP CPU affininty issue on RHEL6
### END INIT INFO

. /etc/rc.d/init.d/functions

[ -f /etc/sysconfig/set_smp_affinity_lpfc ] && . /etc/sysconfig/set_smp_affinity_lpfc

RETVAL=0
prog="set_smp_affinity_lpfc"
lockfile=/var/lock/subsys/$prog

start()
{
 [ -x /etc/sysconfig/set_smp_affinity_lpfc.pl ] || exit 5
 echo -n $"Starting $prog: "
 /etc/sysconfig/set_smp_affinity_lpfc.pl && success || failure
 RETVAL=$?
 [ $RETVAL -eq 0 ] && touch $lockfile
 echo
 return $RETVAL
}

stop()
{
 echo -n $"Stopping $prog: "
 [ $RETVAL -eq 0 ] && rm -f $lockfile
 echo
}

rh_status() {
 status -p $PID_FILE openssh-daemon
}

rh_status_q() {
 rh_status >/dev/null 2>&1
}


restart() {
 stop
 start
}

case "$1" in
 start)
  rh_status_q && exit 0
  start
  ;;
```

```
 stop)
  if ! rh_status_q; then
   rm -f $lockfile
   exit 0
  fi
  stop
  ;;
 restart)
  restart
  ;;
 *)
  echo $"Usage: $0 {start|stop|restart}"
  RETVAL=2
  ;;
esac
exit $RETVAL
```