violin SYSTEMS

White Paper

Deploying Oracle® on Violin All Flash Arrays

Guidelines for implementing Oracle databases on Violin All Flash Arrays

Version 1.0

Abstract

This document illustrates how to deploy Oracle on a Violin All Flash Array. Storage provisioning and database creation are covered. Also included are examples and explanations of performance tests and results.

Table of Contents

1. Overview

This document provides suggestions for deploying the Oracle database on a Violin All Flash Array. The information in the document reflects the following configuration for Oracle:

- Red Hat Enterprise Linux Server release 6.3 (64 Bit)
- Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 – 64bit Production
- Oracle Automatic Storage Management (ASM) Release 12.1.0.1.0 – 64bit Production
- Violin All Flash Array running vMOS 6.3 connected via fibre channel

1.1. Goals and Objectives

The goal of this document is to illustrate deploying Oracle on a Violin All Flash Array. Storage provisioning and database creation are covered. Also included are examples and explanations of performance tests and results, including HammerDB and SLOB 2—both freely available database load generation utilities.

Utilities referenced may be found at the following URLs:

- HammerDB: http://hammerora.sourceforge.net/
- SLOB 2: http://kevinclosson.wordpress.com/slob/

1.2. Intended Audience

This document is intended for Database Administrators (DBAs), Storage Administrators, and Linux Administrators considering deploying Oracle on a Violin All Flash Array. This document assumes some familiarity with Oracle database administration and Linux system and storage administration commands related to database installation.

2. Background: Flash for Databases

Enterprise database applications tend to be "latency-sensitive": while overall IO requirements are generally low in terms of IOs per Second (IOPS) and bandwidth, the latency of those IO requests determines application performance to a large degree. The physical design of a hard disk drive (HDD) leads to best performance for sequential data access—both reads and writes—while it performs worst for random IO workloads. Database applications tend to be highly random: many user sessions performing multiple activity types simultaneously. User-specific data requirements generate random IO requests while the user waits for the request to complete and the database to process the data and return it. The faster the storage can access requested data or acknowledge a write of new or changed data, the faster the application performs overall. Flash is a persistent solid-state storage device. As such, it is not bound by physical limitations of mechanical parts found on a HDD. Aggregated and managed properly, flash can provide orders of magnitude greater performance for databases (or any use case!) than arrays of spinning HDDs.

2.1. Flash Storage vs Hard Disk Storage

There are many storage vendors in the market today selling products they describe as "all flash arrays". The vast majority of these use a similar concept: replace a traditional HDD with an SSD and realize some performance improvement. This concept works to a point: the SSD itself performs IO requests much faster than a HDD, and some improvement is gained. However, the architecture quickly runs into a host of bottlenecks: traditional data pathways, controller heads, and RAID algorithms designed for spinning disks. These act to limit the speed of the SSD and exacerbate performance limitations related to flash management, namely garbage collection. Even new flash array technologies that employ SSDs fall victim to some or all of these bottlenecks, as they cannot manage flash at the chip level but must rely on each SSD to perform its own garbage collection as needed, often interrupting its ability to respond to IO requests and causing application performance to worsen and become less predictable. Violin produces purpose-built arrays designed from the ground up for flash with our Flash Fabric Architecture™, managing the flash at the lowest and most efficient level for maximum performance, scalability, and consistency. This, along with our highly available (HA) design, is what makes Violin a popular choice over other vendors for participation in TPC-type benchmarks.

http://www.violin-memory.com/products/performance-benchmarks/

2.2. Latency in Oracle Environments

Violin recommends reviewing Oracle's Automatic Workload Repository (AWR) reports to determine if IO is impacting your database. (Statspack reports may be used for customers deploying Oracle Standard Edition or who have not licensed the Oracle Enterprise Manager Diagnostics Pack for Enterprise Edition.) Top database IO wait events often include **db file sequential read** and **direct path read** on the read side, and **log file sync** and **log file parallel write** on the write side. If these wait events appear in the "Top 5 Timed Foreground Events" section of your AWR report ("Top 10 Foreground Events by Total Wait Time" for Oracle 12c), and moreover if they trump DB CPU as a percentage of database time, your database is suffering from inadequate IO performance. In an ideal world, your database would not wait for anything, and 100% of DB time would be spent on CPU. Anything less, and you are wasting money on Oracle core licensing that you will never get back. Addressing IO problems is often the greatest step you can take to realizing your full database performance potential and getting the best return on your software licensing investment. With the ability to perform over 1 million IOPS in a single 3U Violin All Flash Array, we can consistently provide IO latency at a fraction of a millisecond to Oracle's IO requests, improving application performance as much as 100X.

Below are before and after Top 5 listings from an actual Violin Proof of Concept. From the application perspective, Violin enabled the customer to process about 10X the number of records in about 1/10 the time.

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
db file sequential read	8,431,503	323,137	38	66.22	User I/O
log file sync	1,846,270	132,142	72	27.08	Commit
enq: TX - row lock contention	32,909	13,893	422	2.85	Application
DB CPU		13,063		2.68	
log file switch completion	4,677	1,801	385	0.37	Configuration

Figure 1: Top 5 timed foreground events on HDD-based SAN with cache

Top 5 Timed Foreground Events

Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		8,418		71.17	
db file sequential read	6,040,513	2,569	0	21.72	User I/O
log file sync	1,334,147	851	1	7.19	Commit
enq: TX - row lock contention	2,216	140	63	1.18	Application
library cache: mutex X	54,027	132	2	1.11	Concurrency

Figure 2: Top 5 Timed Foreground Events from same application on one Violin array

3. Optimum Storage Choice for Oracle

Violin offers a unique product in the all flash array market: a fully redundant, highly available array, truly designed from the ground up for flash. Its design and many patents make it the premier flash storage product, as evidenced by the myriad performance benchmarks that have used it. (See above links for benchmarks utilizing Violin All Flash Arrays.) As only the most robust, failure-tolerant, and performant storage products pass the rigorous tests included in such benchmarks, you must ask yourself: If your storage vendor's products have *not* been used in an official, *published* benchmark… is their product really as great as they say it is?

With advanced features such as patented vRAID protection, thin provisioning, online expansion, application-consistent snapshots and clones, per-LUN deduplication and compression, in-flight and at-rest encryption, long-distance replication, and fully non-disruptive upgrades (NDU), Violin Systems storage products offer all the data management capabilities your database needs. What's more, every gigabyte on a formatted array—over 40TiB usable in today's 3U chassis—is truly usable space. When creating a LUN, you need only specify the usable capacity. We handle all data protection aspects for you with patented algorithms optimized for flash at the lowest level. Any single LUN performs at the speed of the entire array, so there is no longer a need to define multiple storage spaces for data and indexes, temp space, or redo logs. Create any partitions you like, but do so only for your own logical separation requirements; the storage will perform the same regardless.

3.1. Oracle Performance Examples

So what do all Violin's distinct features mean to Oracle database applications? Fast execution? Blazingly fast. Consistent performance? Absolutely. Predictable run times? Finally!

Here is a screenshot from a demo Violin performed *live* at Oracle Open World 2013. HammerDB provided an Oracle workload similar to the TPC-C benchmark in which a single Violin array enabled a single 32-core server to deliver over 2.1 million transactions per minute at a rough cost of $0.31/transaction.

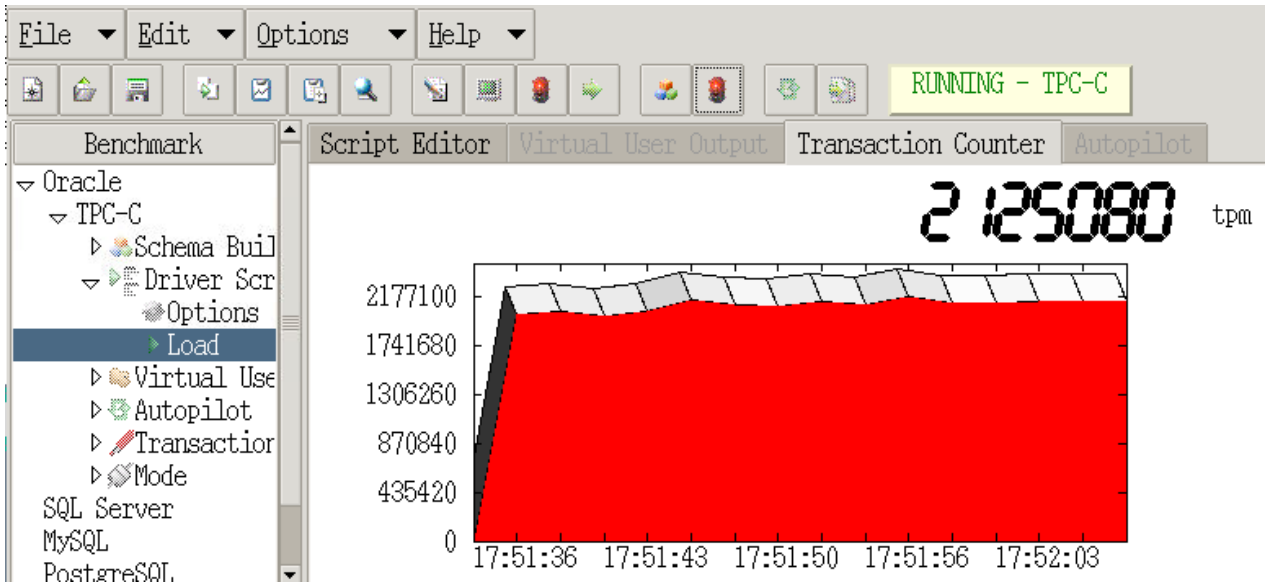


Figure 3: HammerDB 2.14 screen capture running OLTP workload on a single server attached to one Violin array

In the next figure, we see a screen capture from the web-based GUI monitoring tool while running SLOB 2 on a single server attached to a single Violin array. This test was run specifying a 20% update configuration, thus making it 80% read, 20% write. During this test, the array can be seen delivering over 250K IOPS and 2.1GB/s at less than 0.25ms average latency.

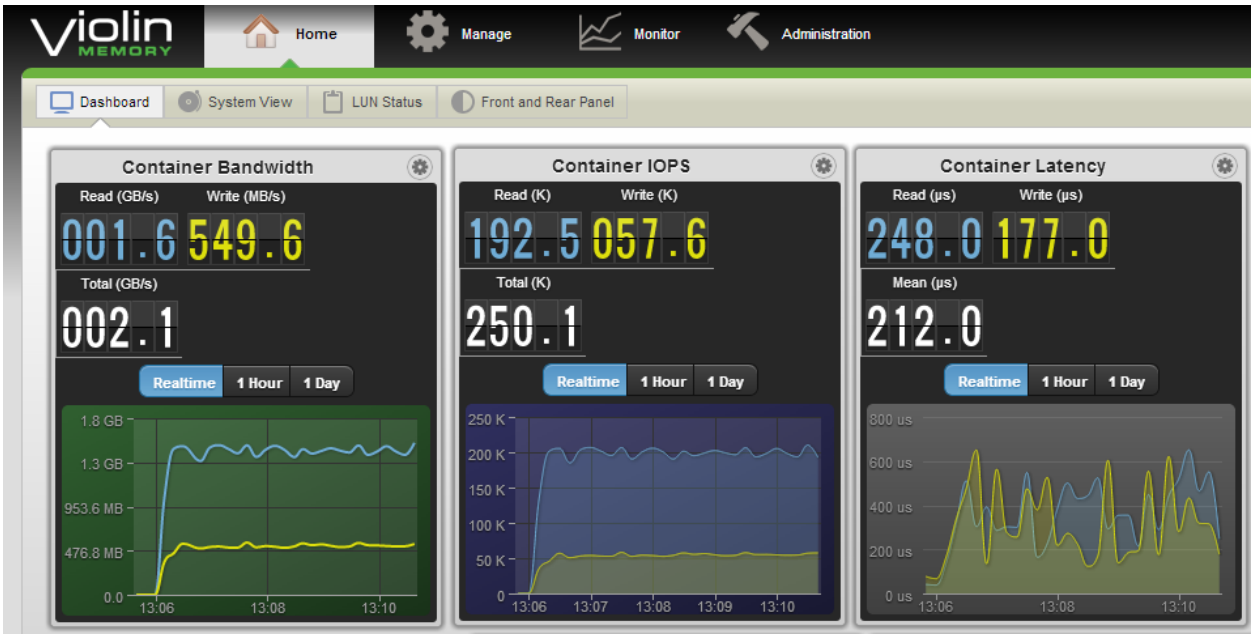


Figure 4: Violin GUI screen capture running SLOB2 workload on a single server attached to one Violin array

3.2. Deploying Violin All Flash Arrays

Violin Systems web-based GUI (CLI also available, along with REST API) makes administration of the array a snap, even when using thin provisioned LUNs or native snapshots of LUNs or LUN groups. We perform all data distribution and protection using patented algorithms for you so all you need to do is specify the size of each LUN. You can even create multiple same-size LUNs in a single step to make adhering to Oracle ASM best practices easy. Violin offers the option to present 4K sector LUNs to the OS and recommends doing so only if the OS fully supports 4K sector devices. (Most modern OS versions do, but check your vendor to be sure.) We also offer 512B emulation mode for OSs that do not support advanced format devices and recommend this if you are unsure of support in your environment. In either case, no changes are required to your database block size: choose your block size based on your data requirements, not on attributes of your storage.

Whether you employ Oracle ASM or file system (FS) to store your Oracle data files, Violin performs the same. Choose whichever suits your environment best without worry of full support. And, of course, logical volume managers (LVM) are fully supported.

Think of the Violin array as a single physical LUN that you may carve into as many (or few) logical LUNs as suits your requirements. Each LUN you create is capable of delivering the maximum performance of the entire array because each LUN is physically spread across every flash component in the array. Create many LUNs if you prefer: having IOPS, bandwidth, and latency metrics on a per-LUN basis within the Violin Systems GUI means you can track down hot data for your own auditing purposes, but you won't have to worry about relocating data to improve its performance or reduce its impact on other data. Also, in contrast to HDD-based and even some flash-based storage products, you can fill a Violin Systems LUN to 100% without worry of performance impact. No more short-stroking or wide-striping drives to achieve modest gains in performance at the cost of usable capacity. Our patented vRAID algorithm costs about as much capacity as RAID-5 (around 20%) and delivers blistering performance for all workloads.

Remember: Violin arrays are all flash, truly built from the ground up designed for flash. In fact, the only moving components in our array are the cooling fans. That means there is no such thing as "cold data" or data that must be retrieved from an under-performing storage tier, crippling performance for the end user and adding inconsistency to the application experience. And no data tiering means no cycles spent on determining hot data and relocating it onto a faster (or slower) storage tier. All your data will be lightning fast and consistent because it is all on flash.

3.3. Storage Cost vs Savings

One of the major inhibitors to wide adoption of flash to date has been the cost of acquisition. When it first hit the market, flash was much more expensive on a per-GB basis than high performance spinning disk solutions. Over the years, however, constant innovations in flash have improved the density, performance, and cost on a curve that closely resembles the way Moore's Law has applied to CPUs and RAM. Now, flash is equal to or less than the cost of high-end HDD systems even on raw per-GB costs. Add to that the fact that the performance of flash is such that you no longer need to over-deploy storage to meet your needs, and the true cost of flash becomes even lower. Our customers have achieved floor space, power, and cooling costs by over 90% compared to their HDD storage systems, while improving performance several times over and simplifying deployment and management.

Violin products offer features like thin provisioning, deduplication, and compression but with an advantage over many of our competitors: We allow you to deploy these features where they make sense and to leave them out of places they don't belong or aren't needed. Features like deduplication work very well in virtualized environments with many images of the exact same data, but they do not work nearly as well in database environments where every database block is unique. Similarly, it makes

little sense to use storage-based compression if your application already compresses data, such as Oracle's Advanced Compression option. Instead of forcing these features on all data sets, Violin allows enabling these features on a per-LUN basis so you can control their usage at a granular level.

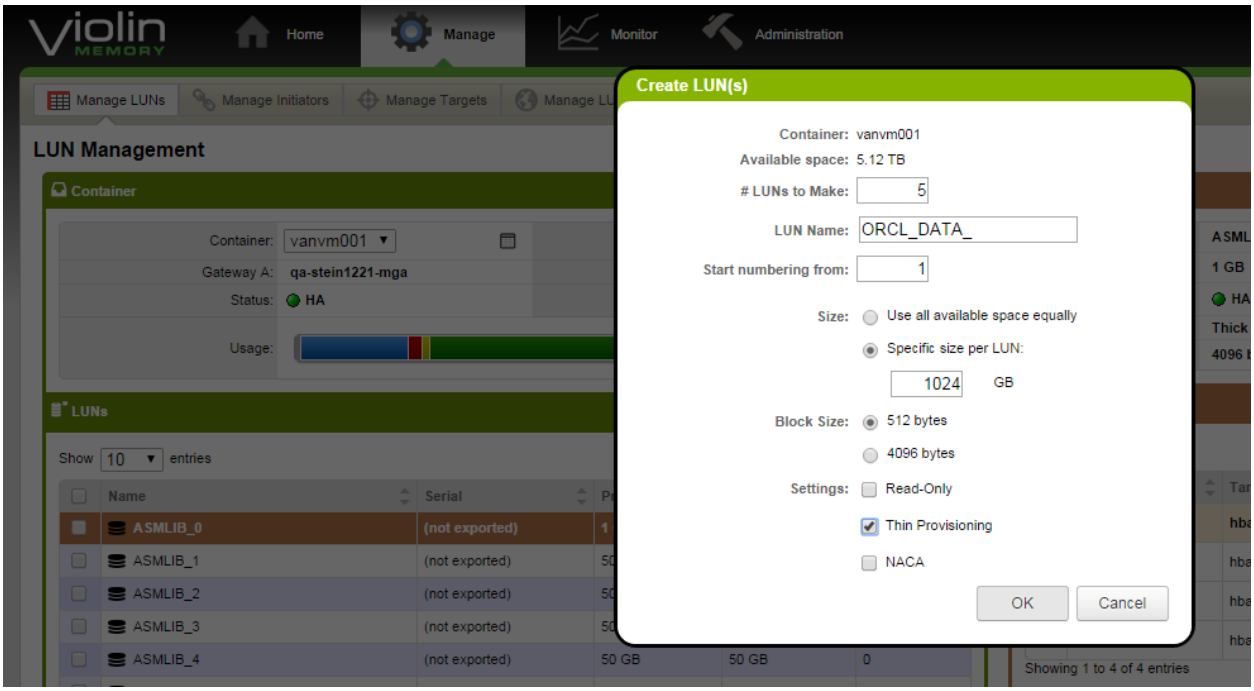
3.4. Database and Server Consolidation

With abundant performance for your transactional system, replicas for purposes of reporting and business analytics, Quality Assurance (QA) or Pre-Production / User Acceptance Testing (UAT) environments may become a thing of the past. Run reports against transactional data without worry of impact to that environment. Or, using Violin's nearly instant snapshots and clones, you could have up-to-date images of data for other environments captured at whatever frequency you like. Violin offers a robust REST API to allow automation of common tasks including provisioning storage and creating and exporting snapshots and clones. Our customers have found storage was not only a performance impediment but also a drag on the business: manual workarounds, extra hours and weekend maintenance operations, and stale data were the norm on legacy storage platforms. Violin revolutionizes the possibilities with our massive performance across all our product offerings. With storage performance no longer affecting your applications and latency removed from the picture for your database and applications, your true CPU requirements become apparent. This has opened up options for license reallocation and reduction for many Violin customers with significant cost savings that easily justify the advanced technology found in Violin All Flash Arrays.

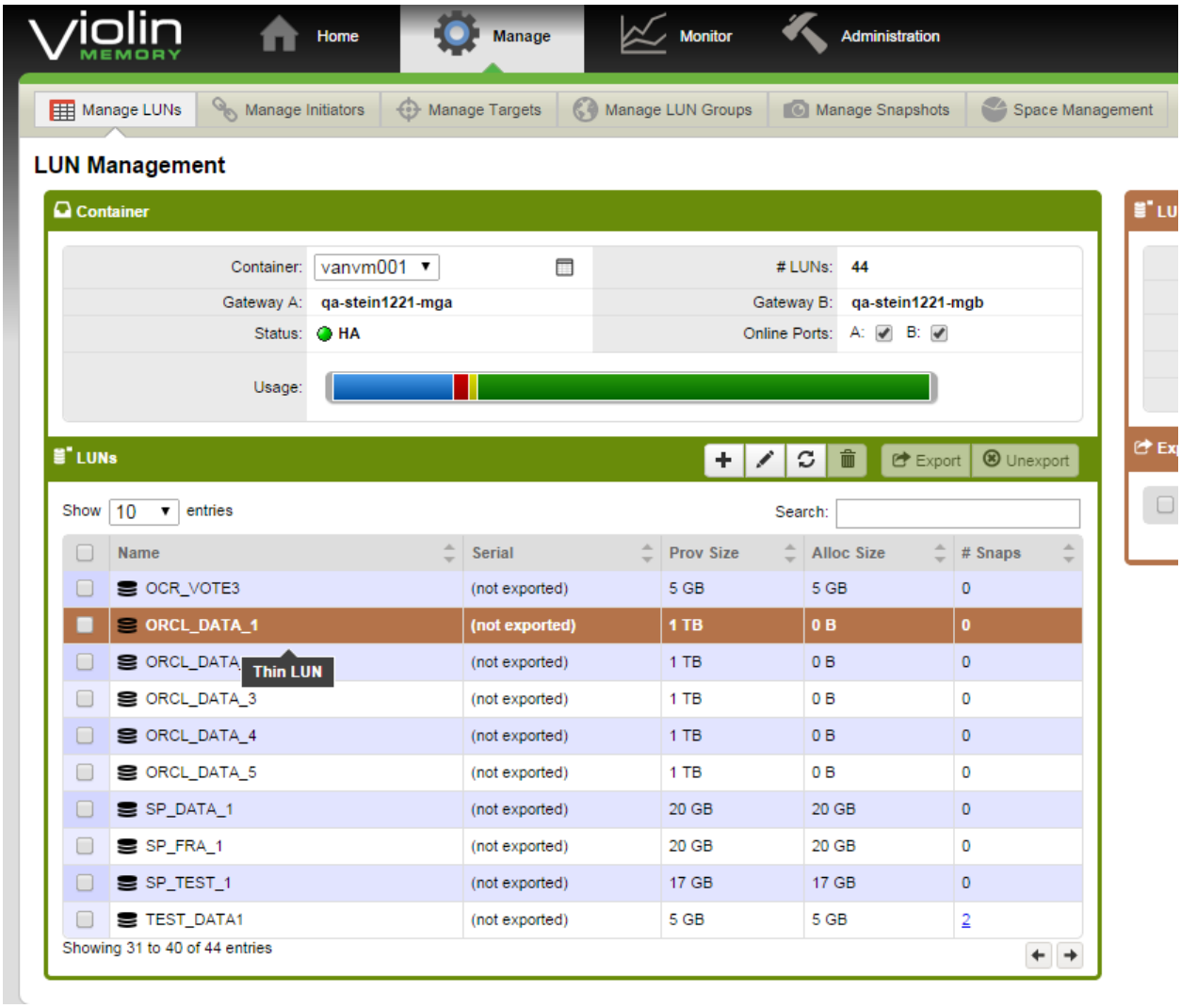
3.5. Using Violin LUNs, LUN Groups, and Snapshots for Oracle Database Clones

Violin's web-based GUI makes LUN provisioning a snap. Whether working with a single LUN or a LUN Group, creating snapshots and clones of existing LUNs is easy to do. Here's an example run through to create a clone of an existing ASM diskgroup to be mounted on a second server.

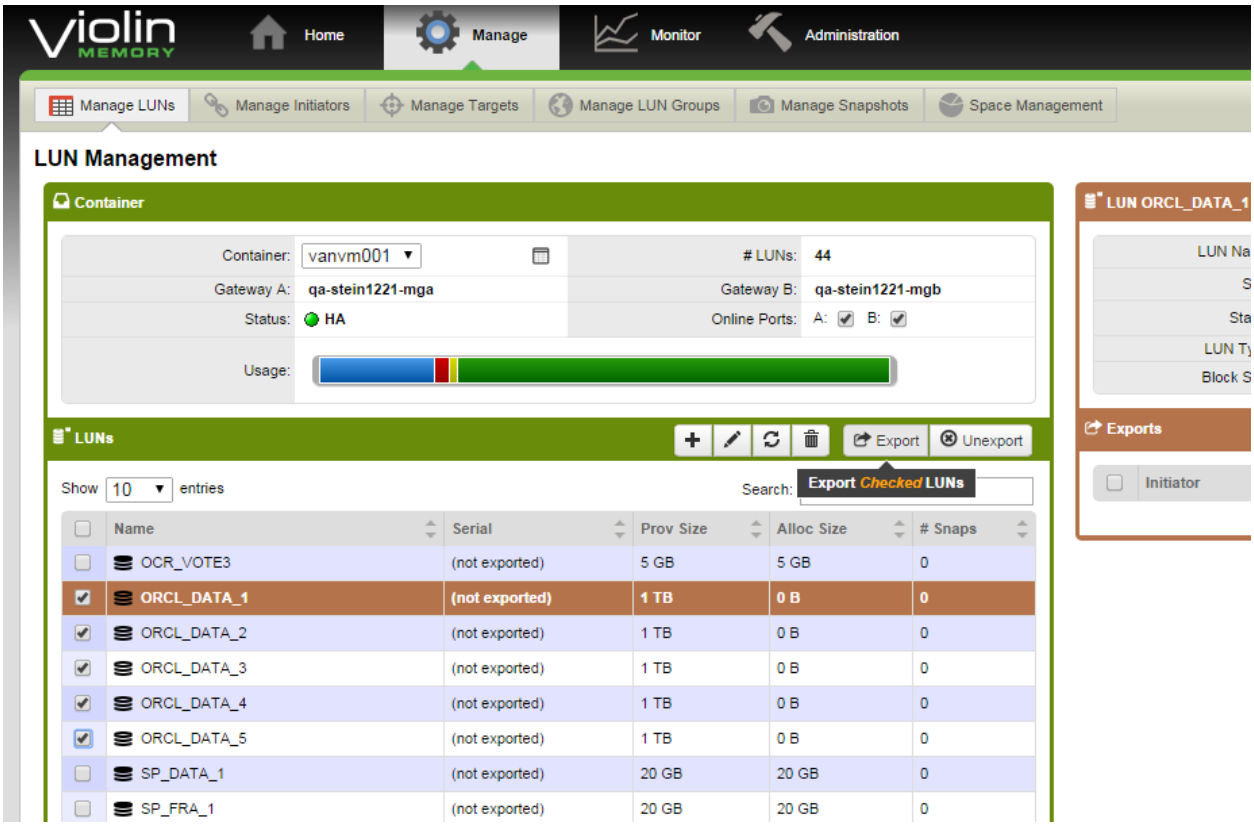
First, we create our DATA group LUNs in a single shot, specifying to create 5 LUNs with a name prefix string of "ORCL_DATA_" and starting index number of 1, each at 1TB and using thin provisioning.

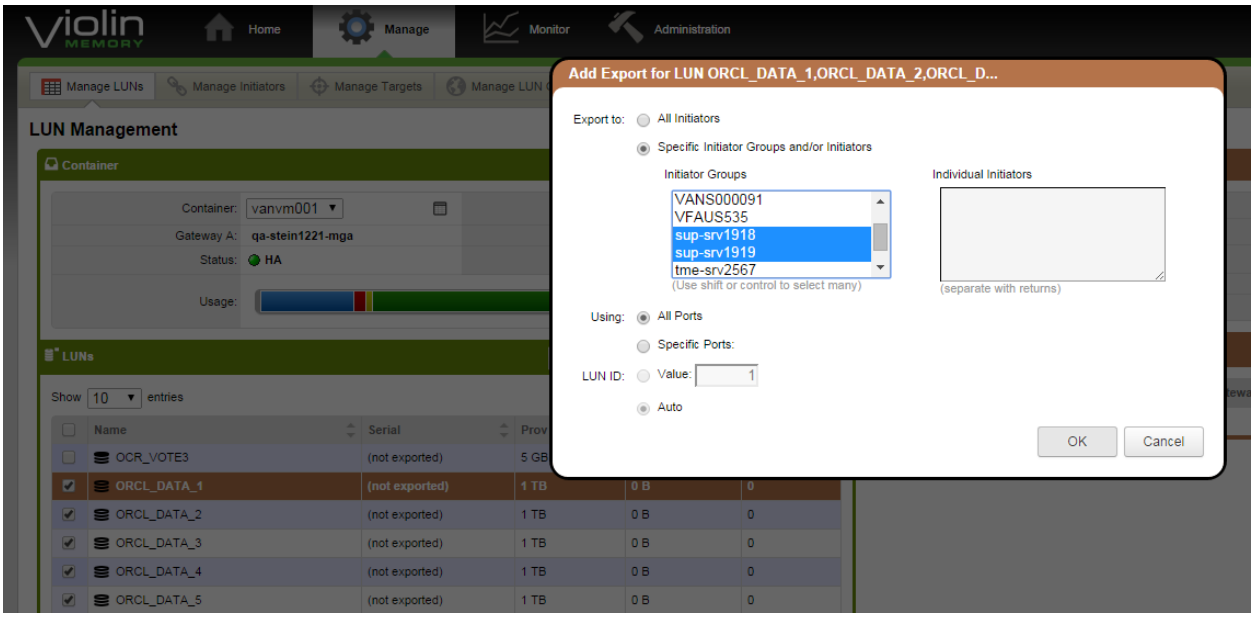


With our LUNs are created, we can see that they are presently consuming 0 Bytes, while they are able to consume up to 1TB each. We can see they are not exported to any hosts by default. We want to be selective in our presentation of these LUNs to help protect our data.



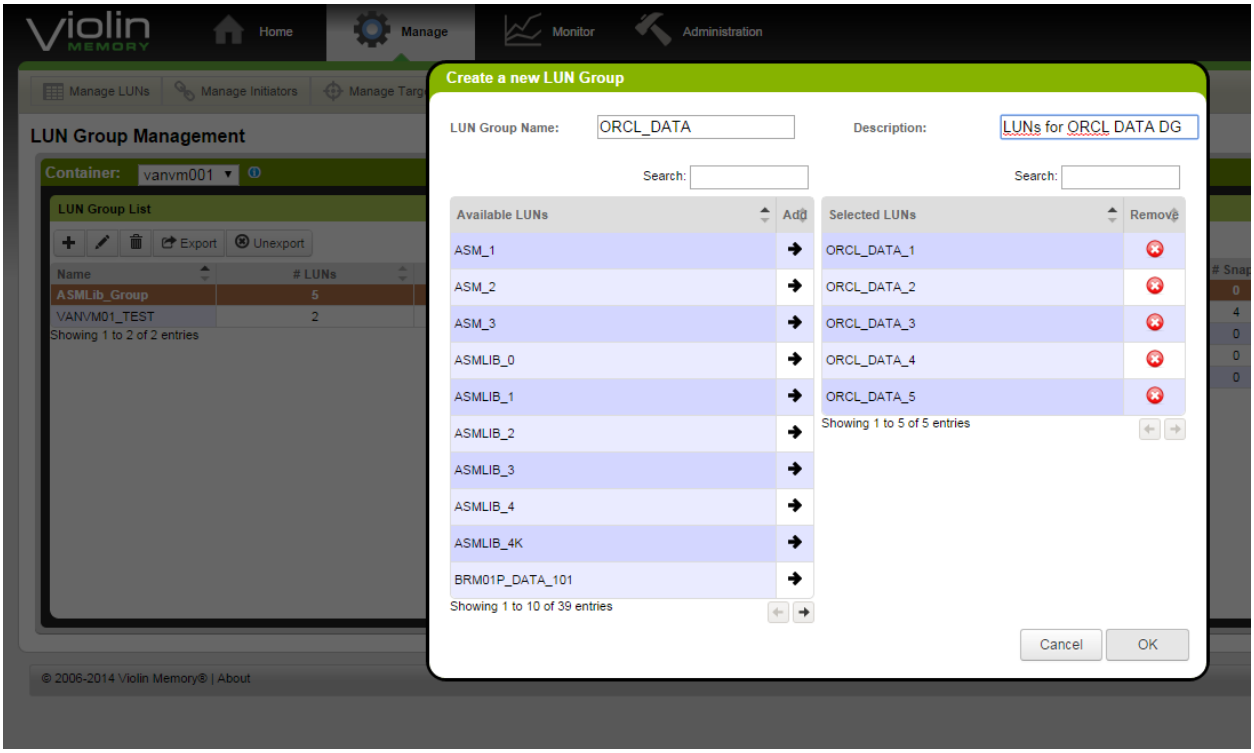
Now we select all 5 DATA LUNs and export them in a single operation to a pair of hosts, presumably a clustered pair of servers. These servers have been identified as an Initiator Group comprised of the WWN values of the HCAs of the servers in question. This makes exporting LUNs to a host on all initiator ports clear and straightforward. We could easily create an initiator group comprising all ports in the server cluster as well.



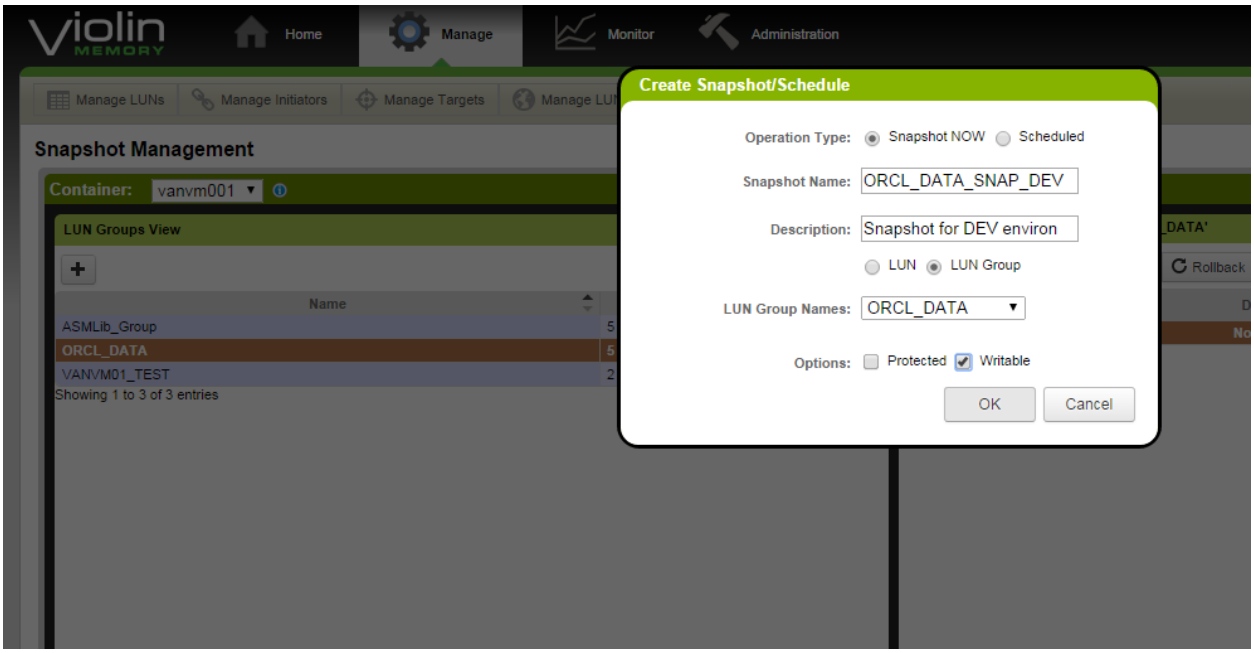


Now, rediscover LUNs on the target hosts. Different OSs use different commands for this, so be sure to check with your OS vendor support. Configure multipathing to enable the host to distribute IO calls across all available paths during normal operations—and to allow use of a subset of paths in the event of an outage of HCA, fibre channel switch, etc. To complete configuration, present the multipath LUNs to Oracle ASM or another host-side LVM. Begin loading your data onto the Violin storage and start to realize the benefits!

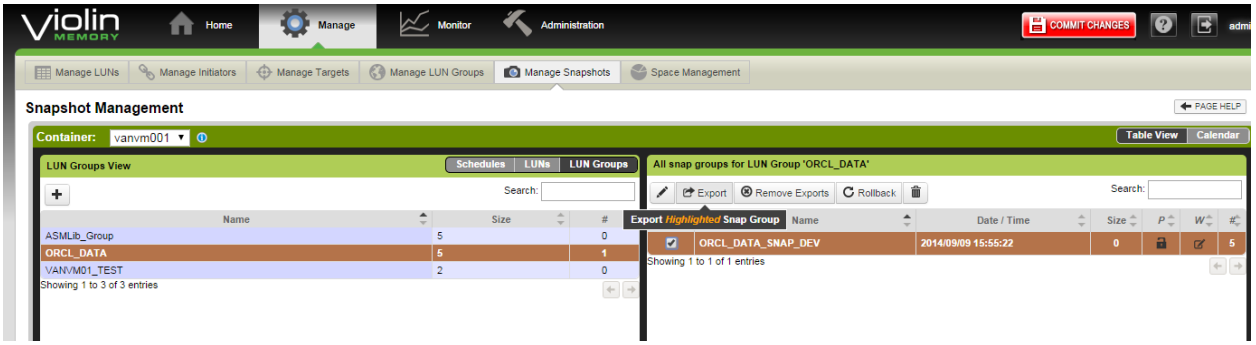
Next, create a LUN group with all your DATA LUNs. This will make the snapshot process clearer. Select all 5 LUNs created and place them in the group, here called "ORCL_DATA" for demonstration purposes.



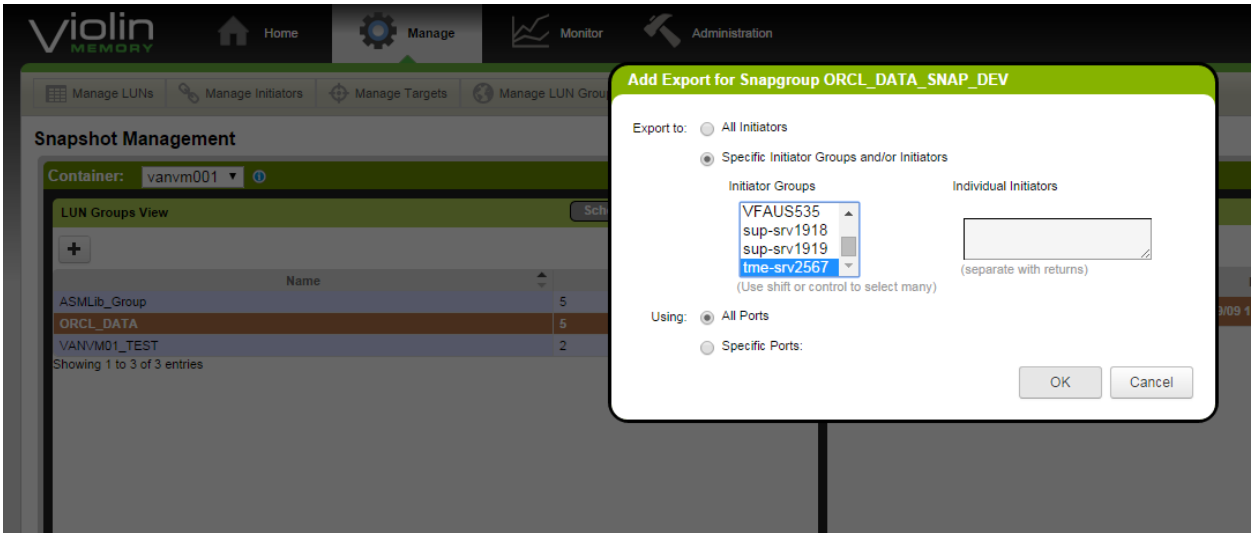
Now, you may take a one-time snapshot or schedule recurring snapshots of the LUN group. Here, we create a writeable snapshot of the LUN group in a single step for presentation to our Dev environment.



We must export the snapshots, which will appear to the Dev host as LUNs themselves, although we know they are simply writeable snaps of our primary data set.



Export the LUN group snapshot to the Dev server.



Now on the Dev server, rescan the HCAs for new LUNs. Configure multipathing, and present the multipath devices to Oracle ASM or the Dev server LVM software to be able to mount the disk group on the Dev server. Media recovery will likely be required, as this process does not constitute an application-consistent snapshot. Look to our Concerto product to include fully application-consistent snapshots and replication for Oracle, MicroSoft SQL, and other enterprise applications.

4. Customer Experience and Economics

One of the most common use cases for Violin All Flash Arrays is for Oracle Databases. With hundreds of Violin All Flash Arrays deployed, we have several who have agreed to share some information on their deployment. We have many more who feel the advantage they gain from Violin on their critical applications based on Oracle is a competitive advantage, and they don't want to discuss any details.

One example would be Sinopec, one of the largest companies on Earth. They are a petroleum company based in China that uses Oracle for several applications including SAP. Sinopec's Oracle database was suffering delays in daily transaction and nightly batch jobs that were taking too long. They needed a high-performance, yet compatible solution with their existing environment to accelerate OLTP transactions (POS, billing, oil card value re-charge) during business hours and accelerate nightly batch jobs (account settlement/reporting). The results were profound. OLTP applications ran faster by 3X over legacy storage and batch job time shortened from 23000 to 6700 seconds, a 70% improvement.

GFI, a trading services company in New York City is another example. Limited technology was preventing GFI from launching a new trading application which required a reduction in hops demanding the best latency storage solution available. GFI also needed to build a new storage tier to support their virtualized environment where the IO blender (where virtualization makes even sequential workloads act like random workloads) challenge was a concern. GFI found a 4X performance increase from 1,000 transactions per second to over 4,000 transactions per second for their trading platform. They also found that virtualizing all applications gave them a competitive advantage to the market, by being faster and able to offer greater services.

Finally, Darden is a hospitality company based in the US. They use analytics to drive menu customization, promotions and guest experience at their over 1,500 restaurants. They needed to accelerate Oracle Data Warehouse with SQL feeds while scaling IT during an expansion into new markets. What Darden found was 50% savings in total cost of hardware, software and support with Violin All Flash Arrays. They also got high availability and performance that included sustained random I/O that was 4x faster than the existing environment.

4.1. No-Cost Consultation and Analysis

Violin Systems employs application experts in most popular areas, including Oracle Database. We offer free, no pressure consultation services to gain understanding of your current environment and its challenges. We will happily assist you in gathering AWR or Statspack reports, analyze them, and present our analysis to you along with improvement estimates based on years of experience with customer implementations of Violin products. Reach out today to learn how Violin works and how our products can revolutionize your business.

www.violin-memory.com/company/contact-us/

About Violin Systems

Business in a Flash. Violin Systems transforms the speed of business with high performance, always available, low cost management of critical business information and applications. Violin's All Flash optimized solutions accelerate breakthrough CAPEX and OPEX savings for building the next generation data center. Violin's Flash Fabric Architecture™ (FFA) speeds data delivery with chip-to-chassis performance optimization that achieves lower consistent latency and cost per transaction for Cloud, Enterprise and Virtualized mission-critical applications. Violin's All Flash Arrays and Appliances, and enterprise data management software solutions enhance agility and mobility while revolutionizing data center economics. Founded in 2005, Violin Systems is headquartered in San Jose, California.

For more information, visit www.violin-systems.com. Follow us on Twitter at twitter.com/violinsystems.