Delivering Low Cost High IOPS VM Datastores Using Openfiler* and Intel® SSD DC S3700 Series

Solutions Blueprint

November 2014

Revision 002

Document Number: 329238-002US

Configurations:
Hardware setup used for the Openfiler VM datastore SAN: Intel® Dual-Processor Xeon® S2600GZ Motherboard Based Server, 2x Intel® Xeon® E5-2680 8-core processors, 64GB of RAM in 8x 8GB Registered DDR3, 3x Intel® RS25DB080 dual-core 6Gbps RAID controllers with feature key and battery backup, 24x 2.5-inch drive 2U server chassis with passive backplane, 24x 400GB Intel® SSD DC S3700 Series solid state drives (data), separate boot/swap SSD., 1x Intel® x520-DA2 dual-port 10Gb Ethernet Adapter (copper Twinax SFP+ connections)

Hardware setup used to host the VMs: Intel® Dual-Processor Xeon® S2600GZ Motherboard Based Server, 2x Intel® Xeon® E5-2680 8-core processors, 128GB of RAM in 8x 8GB Registered DDR3, 1x Intel® x520-DA2 dual-port 10Gb Ethernet Adapter (copper Twinax SFP+ connections), Local SAS disk RAID1 for VMWare* ESXi 5.1 hypervisor

Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

For more information go to http://www.intel.com/performance.
Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Contents

Figures

Tables

Revision History

Revision Number	Description	Revision Date
001	Initial Release	September 2013
002	Minor edits & format changes	November 2014

§

1 Introduction

1.1 Audience and Purpose

The purpose of this document is to instruct IT professionals on how to construct a purpose-built high-IOPS/low-cost Storage Area Network (SAN) using off-the-shelf Intel components, the Openfiler* open source software stack, and VMware* ESXi 5.1. This software-based storage solution is specifically targeted at highly dense virtual environments that interleave VM I/O resulting in a mostly random I/O demand, also known as the 'IO Blender.' This document steps through basic installation and setup of Hardware (HW), Openfiler* Software (SW), and connection of NFS datastores at the VMware host system to NFS shares on an Openfiler server.

In addition, this document explains a best known method for hardware and NFS setup in Openfiler software for optimal performance as a VM datastore using Intel® 10Gb Ethernet controllers and Intel® Solid-State Drive Data Center S3700 Series (Intel® SSD DC S3700 Series) products.

Additional details on Openfiler may be found in the Openfiler installation manual and administration guide. See Section 1.5: Reference Documents for more information. Other Openfiler* manuals such as the Advanced iSCSI Target, or SAN Clustering Administration are available with the purchase of these add-on product modules.

Additional details concerning VMware setup may be found in the vSphere* 5.1 administrator's guide and related documents. In addition to installation guides and manuals, many helpful tips may be found in both Openfiler and VMware's community blog forums and support sites. See Section 1.5: Reference Documents for more information.

Note: All testing is based on internal measurements and setup performed by the Intel Non-Volatile Memory Solutions group specifically for this evaluation. Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

1.2 Prerequisites

This document assumes the reader has prior knowledge of hardware and systems, basic Linux* configuration and installation, and advanced knowledge of VMware ESXi and vSphere 5.1 interface specifically, storage configuration.

1.3 Solution Summary

The adoption of 10Gb Ethernet (10GbE) networking has allowed flexibility and efficiency in the data center. Businesses employing 10GbE can replace many 1Gb twisted pair cables connecting an individual server with only two 10GbE cables saving cost on both structured cabling and switch ports without sacrificing performance. Using NFS, iSCSI*, or Fibre Channel over Ethernet* (FCoE) protocols with 10GbE promotes converged network infrastructure and can eliminate the need for parallel storage networks. This document describes how to construct an NFS storage solution using off-the-shelf Intel® Xeon® servers, Intel® 10GbE cables, Intel® SDD DC S3700 series, and the Openfiler open source software stack. The solution described here is capable of supplying virtual machines (VMs) with 150K random 4K block mixed read/write IO operations per second (IOPS) for an approximate cost of $45K while using only 2 units of rack space and 650 watts of power. Building an equivalent IOPS solution from a simple array of 15K RPM 2.5-inch disks would require approximately 750 disks, 60U of rack space, 11.5 Kilowatts of power, and cost roughly

$175K for the disks alone. Similarly, a SAN solution capable of producing this type of sustained random I/O could cost over $1 per IOPS from a typical storage vendor. In both cases as long as the target usage does not exceed the proposed 18TB solution, it offers superior performance to the competing solution in IOPS per dollar and overall cost and upkeep.[1]

Table 1.1: General Assumptions for Solution Cost Calculation

Drive Type	IOPS	Web Cost	Power Consumption	Size
15k RPM SAS	200	$300	15W	600GB
Intel DC S3700 Series	75000	$1800	5W	800GB

1) 15k RPM drives cost and performance numbers based on an industry average of 200 random IOPS per 15k RPM drive at 15W of power. Cost numbers based on suggested retail pricing of Intel® SSD DC S3700 Series 800GB drive.

1.4 Terminology

Term	Description
Hypervisor	Software, also known as a virtualization manager or VMM (Virtual Machine Monitor) that manages virtual machines
IOPS	Input/output operations per second
LUN	Logical Unit Number
NFS	Network File System
SAN	Storage Area Network
TCO	Total Cost of Ownership
VM	Virtual Machine

1.5 Reference Documents

Document	Document Number/Location
Openfiler Installation Guide	http://www.openfiler.com/learn/
vSphere 5.1 Administration Guide	http://www.vmware.com/support/product-support/vsphere-hypervisor.html
Intel® Ethernet Converged Network Adapter Home Page	http://www.intel.com/content/www/us/en/network-adapters/converged-network-adapters.html
Intel® SSD DC S3700 Series Home Page	http://www.intel.com/content/www/us/en/solid-state-drives/solid-state-drives-dc-s3700-series.html
Intel® Xeon® E5 Family Home Page	http://www.intel.com/content/www/us/en/processors/xeon/xeon-processor-5000-sequence.html

2 Solution Overview

Intel® Solid-State Drives balance performance, endurance, and lower solution cost to meet virtualization and cloud demands.

2.1 Existing Challenges

Pressure to move toward cloud. Cloud computing offers a multitude of benefits. Balancing the enterprise private cloud with public offerings, IT organizations must increase internal density to compete with the public cloud's economy of scale.

Many VMs per datastore. As internal density increases, more and more VMs reside in single large Storage Area Network (SAN) based logical drives (LUNs). This interleaves I/O and produces, from the SAN's perspective, a random workload.

Doing more for less. All IT organizations are driven by budget constraints. Any solution that requires a smaller data center footprint, uses less power, and leverages hardware resources more efficiently helps accommodate these constraints.

2.2 Intel Solution

Intel® SSD DC S3700 Series. Deploying a software based NAS solution to handle many VMs using the Intel® SSD DC S3700 Series can provide a low-power, lower-cost solution relative to existing alternatives.

2.3 Solution Impact

150K Random I/O Operations per Second (IOPS). The Intel® SSD DC S3700 Series Openfiler based software SAN delivers 150k random 4k IOPS at a 90/10 read/write ratio, the equivalent of 700 15k hard disk drives (HDDs), at a power budget of 650 watts. This configuration is capable of driving hundreds of concurrent VMs.

Lower costs. When compared with the number of 15k HDDs required to produce 150k IOPS, the Intel® SSD-based solution requires[1] 650 watts vs. 11.25 kilowatts— 17x less power. When compared with the average SAN at $1 per random IOPS for 150k sustained workloads; this $45K solution comes in at only $0.30 per random IOPS, 70% less expensive. In addition, the 2U/18TB footprint is smaller than any comparable performing SAN, requires less cooling budget, and reduces overall complexity for simpler system management.

Table 2.1: General Assumptions for Solution Power and Cost Savings Calculations

Drive Type	IOPS Target	Drives Required	Web Cost	Power Consumption	$/IOPS
15k RPM SAS	150K	700	$210,000	11.5KW	$1.40
Intel DC S3700 Series	150K	24	$45,000[2]	650W	$0.30
	150K	2U vs. 40U	<25% Capex	17x less power	<80% cost/IOPS

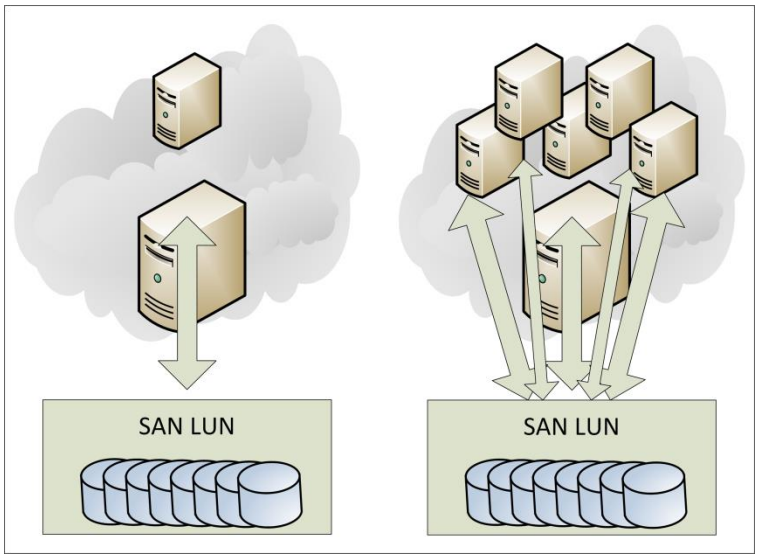
1) 15K RPM drives cost and performance numbers based on an industry average of 200 random IOPS per 15K RPM drive at 15W of power. Cost numbers based on suggested retail pricing of Intel® SSD DC S3700 Series 800GB capacity.

2) Cost includes small percentage for server chassis, board, controllers, and network

3 Solutions Implementation

Building a private cloud is a large and complicated task. One of the main challenges revolves around shared storage and SAN. A decade ago, servers were built with 3-8 disks in a Redundant Array of Independent Disks or RAID set, these disks serviced the operating system and application. In addition, high-performance storage needs were addressed with SAN and the only items placed in high-cost SAN were databases. Fast forward to 2013, the industry has only one or two local disks with a virtualization layer or hypervisor and all the virtual machines sitting in SAN. In essence, we have inverted the disk ratio where 10 years ago we had 3-8 disks for an OS/app stack now we have 3-8 OS/app stacks per disk in a SAN LUN.

One byproduct of this consolidation to VMs is a phenomenal ramp in the randomness of disk I/O. For example, say you have a single virtual machine (VM) connected to a SAN with 14x the number of disks in a RAID array as in your VM datastore. This configuration behaves exactly like a dedicated server and the disk I/O profile of the OS/app stack is identical. Now, you introduce a second server and have combined the two OS/app disk I/O profiles as they read and write to the same VM datastore and your disk I/O becomes 50% random when viewed from the perspective of the SAN. Continue to add VMs to this same datastore and with 10 or more active VMs, the I/O pattern is almost completely random as the individual VMs' I/O streams interleave.



This diagram shows the interleaving of I/O streams as multiple VMs from a single host process requests from a SAN LUN.

Random I/O does a few unsightly things to the typical SAN: it renders caching algorithms useless for predicting I/O, and with caching ineffective you revert to the least common denominator of about 200 IOPS per physical disk running at 15K RPM. Should IOPS demand from any particular VM reach 2.8K IOPs, we have exhausted the capability of our theoretical 14x disk array and every other VM in this datastore degrades because of a single 'noisy neighbor.'

Note: The typical VM datastore at 2-4TB in size can easily host 50-100 VMs at 40GB storage footprint per VM which makes I/O contention commonplace.

The solution, examined in this document, is to construct a SAN which is composed entirely of Solid-State Drives (SSDs). Still more expensive per drive than traditional spinning disks (HDDs) in most cases, the SSD has no moving parts and no seek time which makes random I/O a perfect fit for these drives. SSDs cost more in $ per Gigabyte (GB), however since they generate phenomenal volumes of IOPS per drive the cost per IOPS is dramatically less than traditional disks—by up to 400 times. For instance, a 400GB Intel® SSD DC S3700 Series drive can produce up to 75K random read IOPS at an approximate cost of $940, which yields up to 79 IOPS per dollar. Whereas a 15K RPM 600GB HDD can produce 200 random read IOPS at $300, or 0.67 IOPS per dollar. The bottom line in cloud infrastructure is that consolidating VMs produces more random IOPS, random IOPS devalue many caching algorithms, and hard disk drives alone cannot deliver these random IOPS efficiently. SSDs on the other hand continue to decline in cost per GB, excel at providing random I/O, run on approximately one third the power, and have now evolved with enterprise-class high-endurance features and consistency features.

It seems a logical conclusion that for high-density VM environments such as the ones found in enterprise IT data centers, an SSD-based SAN is the optimal solution. The solution examined in this blueprint shows an SSD based solution using all Intel components and Intel's enterprise class Intel® SSD DC S3700 Series with High Endurance Technology (HET) Multi-Level Cell (MLC) NAND, which allows for approximately 5 years of continuous writes to the drive at 10x drive overwrites per day.

3.1 Openfiler* Product Overview

Openfiler converts industry-standard x86/64 architecture systems into full-fledged NAS/SAN appliances, IP storage gateways, and provides storage administrators with a powerful tool to manage burgeoning storage needs. Openfiler allows storage administrators to make the best use of system performance and storage capacity resources when allocating and managing networked storage in a multi-platform environment.

Openfiler addresses key data storage concerns:

- *Reliability*—Openfiler supports both software and hardware RAID with monitoring and alert features, volume snapshots, and recovery
- *Availability*—Openfiler supports active/active high availability clustering, MPIO, and block level replication with WAN optimization
- *Performance*—Recent Linux kernel supports the latest CPU, networking, and storage hardware. Specifically this document explores the top end of performance using Openfiler and both Intel® Xeon™ processors and Intel® SSD Data Center Family.
- *Scalability*—Openfiler scales to 60TB+ with online file system and volume growth support

As a software-based solution, Openfiler can be downloaded and installed in minutes on any current Intel® Xeon-based server or platform. Openfiler may be used to build a storage gateway for existing deployed SAN or DAS storage systems. In this context, Openfiler can share existing storage capacity to additional clients, and provides flexible storage architectures for virtualized systems.

Openfiler provides both block-level and file-level storage networking capabilities, including:

- iSCSI
- FC
- FCoE
- CIFS
- NFS
- FTP
- HTTP/DAV

Openfiler's iSCSI target and FC target is optimized for supporting virtual image storage, with specific features for enhanced availability and performance.

Openfiler delivers an affordable high-availability and disaster recovery solution to enable business continuity in the event of a hardware component failure. Openfiler presents a unified storage paradigm by supporting both block-level and file-level storage networking protocols. Storage may be allocated across any or all protocols simultaneously, allowing data to be accessed from a heterogeneous base of network clients; UNIX*/Linux*, Windows*, or Macintosh* OSX*.

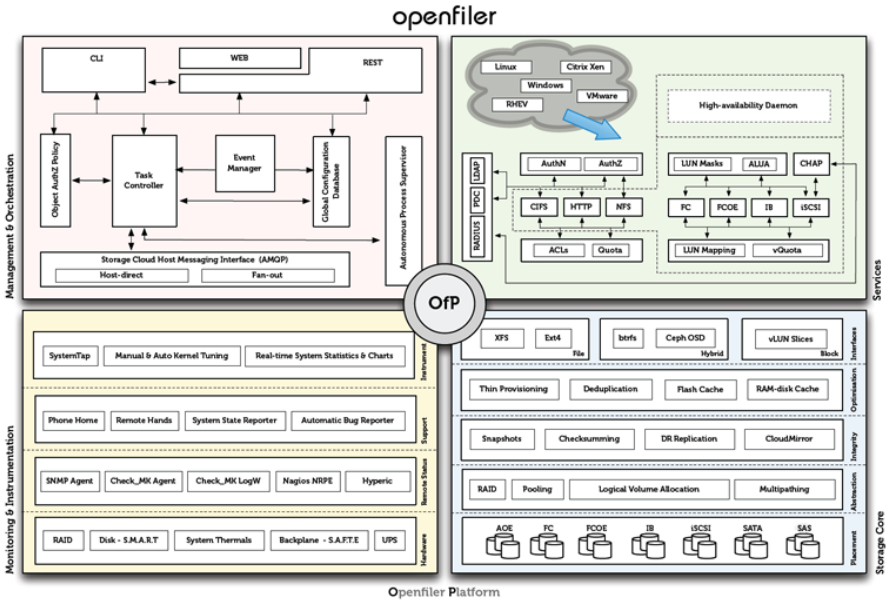
Openfiler's management capabilities are grouped into separate sub-interfaces by functional domain: networking, physical volumes, user and group authentication/authorization, system configuration, and status information. Volume management capabilities in the GUI include features such as dynamic volume aggregation, logical volume allocation and distribution, along with point in time copy (snapshot) management.

Building a storage router/gateway with Openfiler is simple. Most proprietary storage solutions require a license for each individual storage protocol. The Total Cost of Ownership (TCO) for a multi-protocol proprietary storage appliance increases significantly with each licensed protocol. Deploying Openfiler as a storage appliance gateway/router, either on bare-metal or as a virtual machine image, provides a cost-effective workaround to licensing storage protocols from proprietary storage vendors. In this scenario, the Openfiler accesses existing proprietary storage over Fibre Channel or other protocol and redeploys the storage capacity over any of the supported storage protocols.

Single Pane of Glass (SPOG) management delivers the ability to manage and monitor all facets of an Openfiler based SAN environment. An administrator is able to perform management tasks from one console to affect one or more Openfiler instances deployed locally or over a campus-area or wide-area network.

Administration access control confers the ability to delegate certain aspects of administrative tasks to multiple administrators based on a desired level of control. Administration delegation can be macro scale—manage an entire appliance or micro-scale—manage a single LUN on a single Openfiler appliance.

Figure 3-1. Openfiler* Architecture Diagram



3.2 Openfiler* Product Components

Block Device Management - any disk or raw volume is considered a block device in Openfiler. These can be physical block devices attached directly to the Openfiler host appliance, or remote block devices. Remote block devices accessed via iSCSI, FC, AoE or FCoE protocols are defined as virtual block devices. Openfiler can work with remote block devices to present to the upper layers of the Openfiler storage architecture.

- **Remote Block Replication** - Openfiler supports block device aggregation using RAID protocols by means of a software RAID management layer. Openfiler supports both synchronous and asynchronous block-level replication, with optional WAN optimization, supporting both high availability and disaster recovery. Block-level replication is done over the standard TCP/IP protocol, enabling data transfer over both local area and wide area networks.
- **Dynamic Volume Aggregation (DVA)** - before storage is presented to the export layers, a pool must be created of all the raw block storage; virtual block devices, RAID volumes, and replicated volumes are concatenated into unified storage pools. These storage pools can then be further sliced to be allocated as block or file system shares.
- **Logical Volumes** - Openfiler allows the administrator to partition a DVA instance into slices which can then be presented to clients over any of the supported storage networking protocols. These slices are referred to as logical volumes. Because these logical volumes can be migrated between physical storage locations, storage administrators can transparently manipulate the underlying physical block devices without affecting the structure of the exported volume.
- **Snapshots** – A snapshot volume is a frozen image of a logical volume. A logical volume can have one or more snapshot volumes, representing the state of the logical volume at the time the snapshot volume is created. With a snapshot volume, I/O on its source logical volume can continue uninterrupted, while a consistent backup of the snapshot volume takes place.
- **File systems** - Openfiler supports multiple on-disk file systems. Administrators can create a logical volume and apply a file system that meets the needs of specific application I/O profiles or performance parameters. The three major file systems supported by Openfiler are XFS, ext3/ext4, and BTRFS. These file systems are journaled, which enhances data security and reduces the need for regular file system checks.
- **High Availability** – Openfiler supports high-availability for mission-critical storage resource management. Storage exports can be made highly-available by taking advantage of an integrated cluster manager. Two or more Openfiler storage appliances can be configured in a cluster and combined with block replication to maximize uptime for client applications and business continuity.
- **Block Replication** - Openfiler supports block-level replication over standard TCP/IP connections. Replication is normally configured to be synchronous enabling data at the peer site to always be fresh. The Block Replication Driver is the equivalent of RAID 1, except that the data blocks for the mirror device are sent over a network to disks located in a remote server, and that reads are always satisfied from local disk. A Block Replication Driver can work over virtual private network (VPN) links for added security if a dedicated secure link is not available between the primary active server and secondary passive server.
- **WAN Replication** - Block Replication Proxy is required for situations where replication must happen over a slow Wide-Area Network (WAN) link. This allows bandwidth control and pending write coalescing to reduce bandwidth needs for asynchronous replication. The proxy stores copies of the written data locally, and asynchronously updates the remote site within the configured bandwidth constraints.

4 Test Configuration and Plan

4.1 Hardware Setup and Configuration

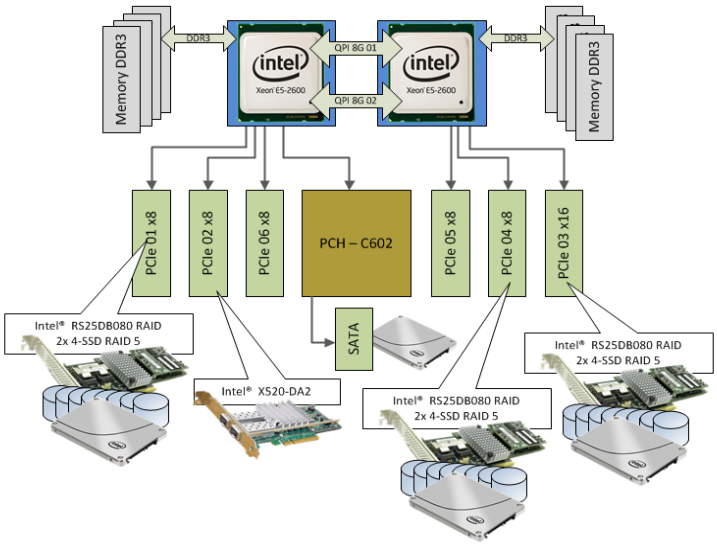
Hardware setup used for the Openfiler VM datastore SAN

- Intel® Dual-Processor Xeon® S2600GZ Motherboard Based Server
- 2x Intel® Xeon® E5-2680 8-core processors
- 64GB of RAM in 8x 8GB Registered DDR3
- 3x Intel® RS25DB080 dual-core 6Gbps RAID controllers with feature key and battery backup
- 24x 2.5-inch drive 2U server chassis with passive backplane
- 24x 400GB Intel® SSD DC S3700 Series solid state drives (data), separate boot/swap SSD.
- 1:1 SSD/SAS Channel Relationship
  (***Note****:* half-duplex *SAS expander not used as this can negatively impact SSD performance*)
- 1x Intel® x520-DA2 dual-port 10Gb Ethernet Adapter (copper Twinax SFP+ connections)

Hardware setup used to host the VMs

- Intel® Dual-Processor Xeon® S2600GZ Motherboard Based Server
- 2x Intel® Xeon® E5-2680 8-core processors
- 128GB of RAM in 8x 8GB Registered DDR3
- 1x Intel® x520-DA2 dual-port 10Gb Ethernet Adapter (copper Twinax SFP+ connections)
- Local SAS disk RAID1 for VMWare* ESXi 5.1 hypervisor

Figure 4-1. System Architecture Diagram



The above diagram shows the PCIe* to controller relationships between the three RS25DB080 RAID controllers, the X520-DA2 network controller, and the local SATA* controller for the VMware ESXi cloud OS. This configuration was chosen with three controllers to handle the I/O produced by multiple SSDs as this spreads I/O across both processors in this dual-socket system. Further performance gains are possible by using more powerful RAID controllers, or increasing the number of RAID controllers, and additional 10GbE Network cards to spread the I/O load even further across multiple CPUs and I/O paths. However, these types of expanded configurations were not tested in the scenario presented in this document.

4.2 Software Setup and Configuration

Software used in the high-density VM datastore SAN and host configuration and testing.

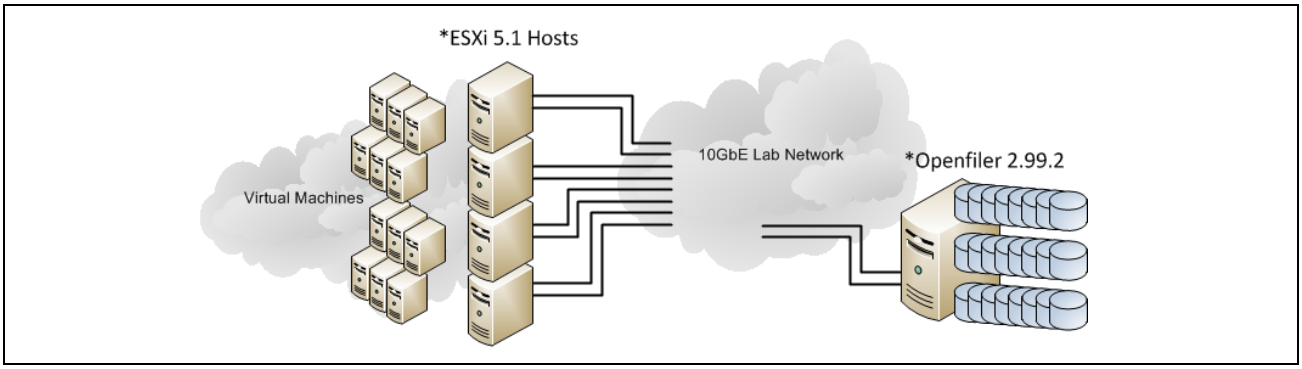
- 1x Openfiler* v2.99.2 with NFS shares
- 4x VMware* ESXi v5.1 build 838463
- 12x VMs with 4vCPU and 4GB RAM – All disks on NFS shares, 100GB test LUN (VMDK), Windows* Server 2008 R2 x64 with current patches, virus scanner, security compliance validation software
- 12x VMs running Iometer* with; 4x worker, 100% random, 4k block size, 4 queue depth, 90% read, 10% write workload with 2x VMs per NFS Datastore. This workload simulates several 100 VMs working on the same shared datastore.

4.3 Software Architecture and Network Configuration Diagram

Important notes:

- Openfiler configuration to distribute I/O across as many cores as possible within a system
  - Installed and configured NFS shares
  - 6x separate NFS shares – one for each 4-SSD/RAID5 controller LUN
  - 3x NFS shares per IP, dual port 10GbE
  - Scheduler algorithm In Openfiler OS for each SSD RAID set configured to 'noop'
  - Results: minimum of 12x multi-threaded processes that leverage multiple cores
- VMware ESXi 5.1* configuration for maximum performance
  - Hosts configured with 2x 10GbE connection
  - Half of test VMs on NFS datastores on 10GbE Port 1, other half on 10GbE Port 2
  - Validated communication across both 10GbE adapter ports to NFS shares

Figure 4-2. Network Configuration Diagram



5 Walk Through Setup and Configuration

5.1 Basic Installation – Openfiler

The first steps in creating an Openfiler VM Datastore are to assemble the hardware, install the requisite components in the server, configure the BIOS, set up the RAID controller(s) using the manufacturer's instructions, and install the Openfiler storage OS from CD or USB. In our installation, we used the RAID console available during POST for the Intel® RS25DB080 controllers. This allowed us to configure each of the 3 controllers we installed with two RAID 5 sets on each controller consisting of 4 SSDs each. Your setup will likely differ, please consult your manufacturers user guide for RAID setup. See Section 5.4 for SSD Setup Optimization; these may differ depending on your controller make & model. The complete explanation of a server build from the hardware up is beyond the scope of this document. Once the basic hardware setup is complete, including the RAID controller and SSD data LUNs:

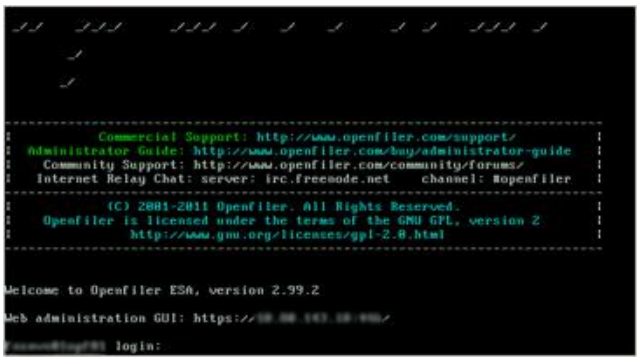
1. Insert Openfiler 2.99 disk or mount ISO through KVM/remote management and boot
2. Follow the on-screen instruction and Openfiler manual to complete the installation
3. When the installation is complete, click reboot to reboot the system, and don't forget to remove the CD ROM/ISO

5.2 Change Password and Update Openfiler

Once the base Openfiler installation is complete,

1. Open a web browser and browse to the address listed on Openfiler welcome screen (https://xxx.xxx.xxx.xxx:446)

Figure 5-1. Openfiler Welcome Screen



2. Accept the security mismatch for https certificate, a valid certificate from a trusted source can be installed later as necessary.
3. Log in to the web GUI with the default username/password (openfiler/password)

4. Under the Accounts Tab > Admin Password – Change the administrator password immediately!

***Note**: the web GUI is 'openfiler' + 'password', this is different from the console or PuTTY/SSH login that is root + <password set during installation>*

5. After changing password and validating (logout/login) use a terminal program like PuTTY to SSH to box using the root user and password

6. From an SSH terminal, run the code from the code snippet box below, replacing your local proxy configuration if necessary

Code Snippet:

```
[root@myopenfiler ~]# export
[root@myopenfiler ~]# http_proxy=http://<myproxy.proxy.url>:<portnum>
[root@myopenfiler ~]# export https_proxy=http:// <myproxy.proxy.url>:<portnum>
[root@myopenfiler ~]# export ftp_proxy=http:// <myproxy.proxy.url>:<portnum>
[root@myopenfiler ~]# conary updateall
```

7. Run a *[root@myopenfiler ~]#* **conary updateall** to update the Openfiler* software to the latest public release.

A direct Internet connection, if possible, is preferred over a proxy configuration as occasionally a proxy-based configuration will error out with a mismatch or file not found error. Simply retrying the update should resolve this. If you download Openfiler 2.99.1 this update may take a few attempts to get all the required updates. If you downloaded Openfiler 2.99.2 there are few to no updates depending on the system. Occasional proxy errors resolving hostnames/addresses and busy repositories may occur – retry until functional, deactivate 2nd NIC(s) while updating may help with disconnections.

8. Once the update is complete, reboot system: *[root@myopenfiler ~]#* **shutdown -ra now**.

Note: For newer RAID controllers like the Intel® RS25DB080 to be recognized, the system must be patched and updated from Openfiler 2.99.1 or installed with Openfiler 2.99.2. Additionally, the Openfiler community site can help in resolving RAID on drive controllers that for some reason are not recognized by Openfiler. These controllers tend to be extremely new and the drivers are typically available but not in the default Openfiler driver set.

5.3 Updating Complete – Add Optional Entitlements

Adding entitlements like VMware* certified iSCSI target and Openfiler* 3.0 beta access installs additional components and programs. This process will vary slightly depending on which pay-for features have been purchased from Openfiler and may include items such as LUN replication, HA/failover, and clustering controls. The method to install these additional services and plugins changes from time to time. Please consult your most recent Openfiler documentation for current installation instructions.

5.3.1 Adding a plug-in entitlement (paid-for Openfiler* options)

If you have purchased add-in components for Openfiler, these may be added to the server configuration using the following procedure.

1. Using VI* or another text editor, create a plain file called **esaupdate.openfiler.com** in the **/etc/conary/entitlements**, with syntax similar to the sample below (provided by Openfiler with purchase)

```
Code Snippet:

mkdir /etc/conary/entitlements
cd /etc/conary/entitlements/
vi esaupdate.openfiler.com
        ***ADD ENTITLEMENT TEXT*** & :wq! ***Write File***
cat /etc/conary/entitlements/esaupdate.openfiler.com
        <entitlement>
         <server>esaupdate.openfiler.com</server>
         <class>esa-3.0-class</class>
         <key> my business .com and authorization key for the entitlement I purchased goes here </key>
        </entitlement>
```

In addition to the esaupdate.openfiler.com file in the /etc/conary/entitlements folder there will be instructions from Openfiler for creating a **system-model** file in the **/etc/conary** folder with the purchase of any plugins.

2. Follow these instructions and again run a *[root@myopenfiler ~]#* **conary updateall** to update the system

This will take a bit and require multiple downloads and a few minutes.

3. Once the update is complete and your paid-for entitlements installed, reboot for good measure with a *[root@myopenfiler ~]#* **shutdown -ra** now or **reboot**

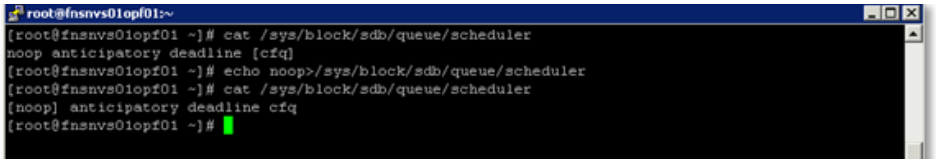
5.4 Setup Optimizations for SSD-based LUNs

Once the basics of Openfiler are completed, please keep in mind these specific optimizations pertinent to SSDs as you complete the configuration of partitions, LUNs, and shares on your Openfiler.

You can greatly improve random performance of the SSD-based RAID disk sets by setting the I/O scheduler to **noop**. In essence the *noop* I/O scheduler does not impose any algorithm such as *deadline/cfq/anticipatory* on the I/O queue. These I/O schedulers were intended for use with spinning disks and often increase the efficiency of standard hard drives. Unfortunately, they often hamper an SSD's performance so we have to turn these off and set the queuing scheduler to none (a.k.a *noop*). To accomplish this,

1. Using VI or another text editor, create a new file called **ssdOpts.conf** under the **/etc/bootloader.d** directory
2. Inside this new file place the following text: **add_options elevator=noop**
3. Save the file, then run **bootman** and reboot Openfiler.

***Note**: Another way to accomplish or check this would be to [root@myopenfiler ~]# **cat /sys/block/sdX/queue/scheduler** and look for the scheduler type in the return line. To set a block device to the noop scheduler run [root@myopenfiler ~]# **echo noop>/sys/block/sdX/queue/scheduler**. You can run this command for each device that is an SSD. This command will not persist after a reboot of the server and is recommended for testing only.*



In addition to the optimization of the OS scheduler in Openfiler, a number of changes can be made to the properties of the RAID controller disks to improve performance. These include:

- Adding the SSD feature key and battery backup to the controller
- Enabling individual disk cache
- Enabling write back cache
- Enabling adaptive read-ahead cache
- Adjusting the read patrol rate

Whether or not these options accelerate disk performance will completely depend on the storage workload that the Openfiler experiences. Testing with your expected workload is quite important to determine which of these options will benefit your workload and use case.

Since SSDs produce very high IOPS, higher core-count CPUs and more powerful drive controllers may increase performance. For instance, using a 24-port controller instead of an 8-port controller for 8 SSDs will result in higher per-drive IOPS due to the controller's increased capabilities and processing power. Finally, for the Intel® SSD DC S3700 Series drives, aligning the partitions to a 4k block size and 64 initial blocks can increase performance by up to a factor of two over non-aligned partitions. To accomplish this alignment:

- Use the **fdisk** or equivalent **parted** command at the Openfiler command line with the flag for 4k block size to create the initial partitions instead of the web GUI.

Code Snippet/sample for fdisk:

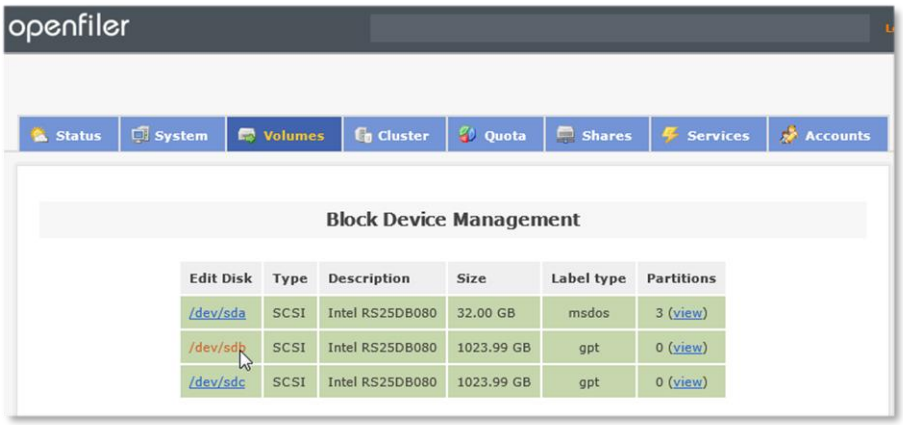
[root@myopenfiler ~]# fdisk –b 4096 /dev/sdX
c to turn off DOS compatability (if not set)
u to change units to sectors (if not set)
m to make a new partition offset starting at 0 or 64
w to write the partition table
q to exit fdisk
[root@myopenfiler ~]#

5.5 Creating Partitions, Volume Groups and Volumes

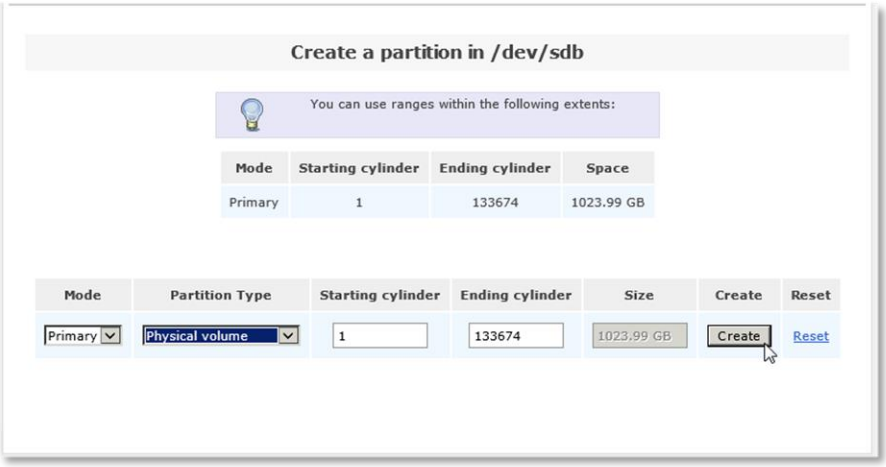
In this section, we partition (if not already partitioned) and format disks, then add and configure NFS shares through the Openfiler web GUI.

1. In the **Volumes** tab > **Block Device Management** create partition(s) by clicking on the **/dev/sdb** link.

Note: */dev/sda is normally the boot/swap/root device and should not be used for data storage or shares.*



2. Accept the settings and click **Create**

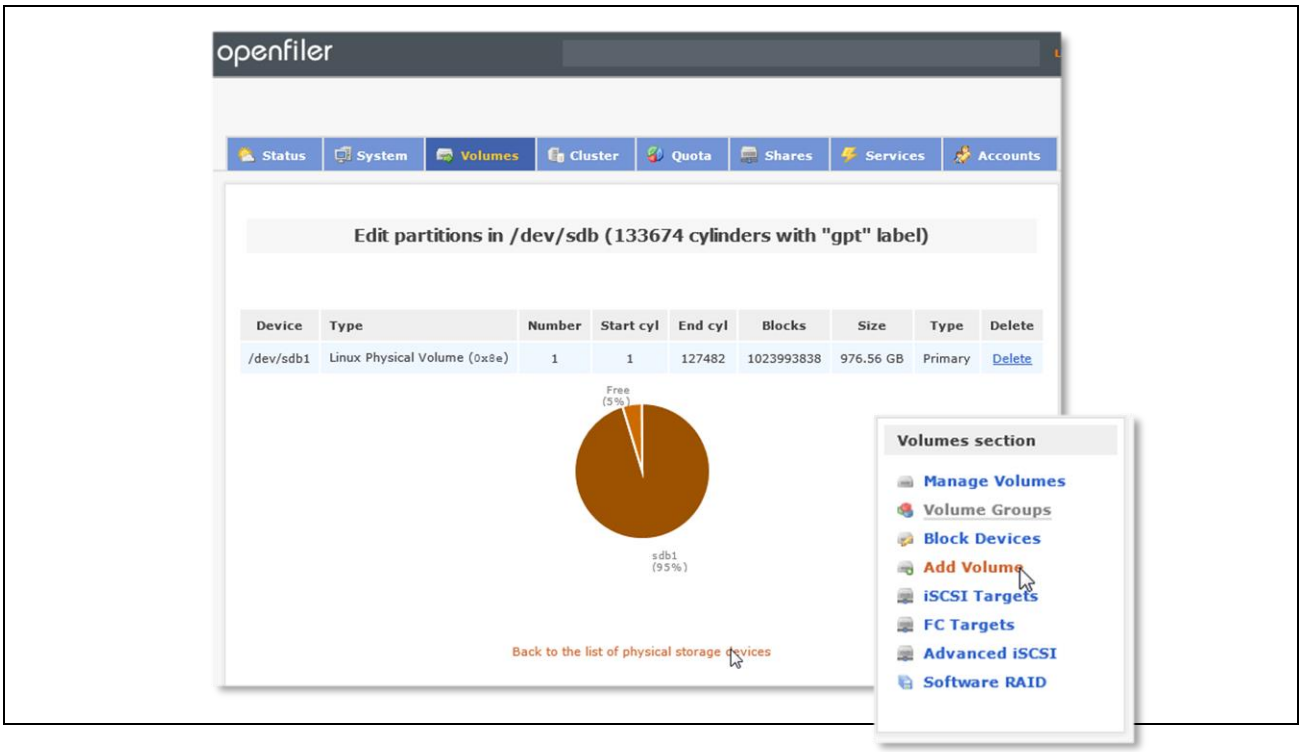


In this tested configuration, we used two logical disks per RAID controller, each logical disk being comprised of 4x SSDs in RAID5 at the hardware level.

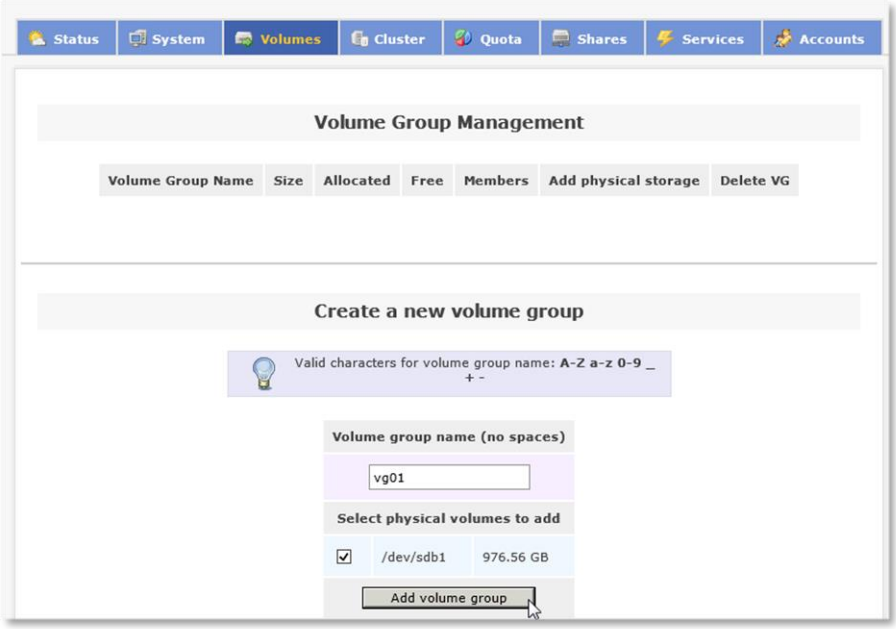
3. Once the partition is created, click on the **Back to physical devices list**

Once you have partitions in a block device, you can create a volume group.

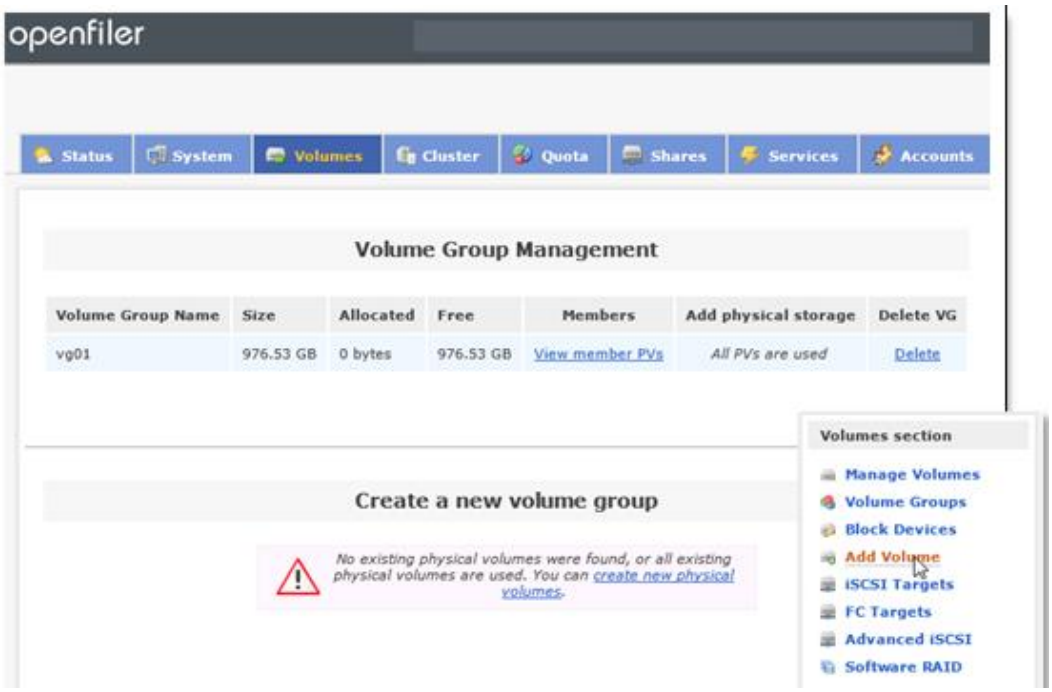
4. Click on **Add Volume** in the right hand menu in the volumes tab.



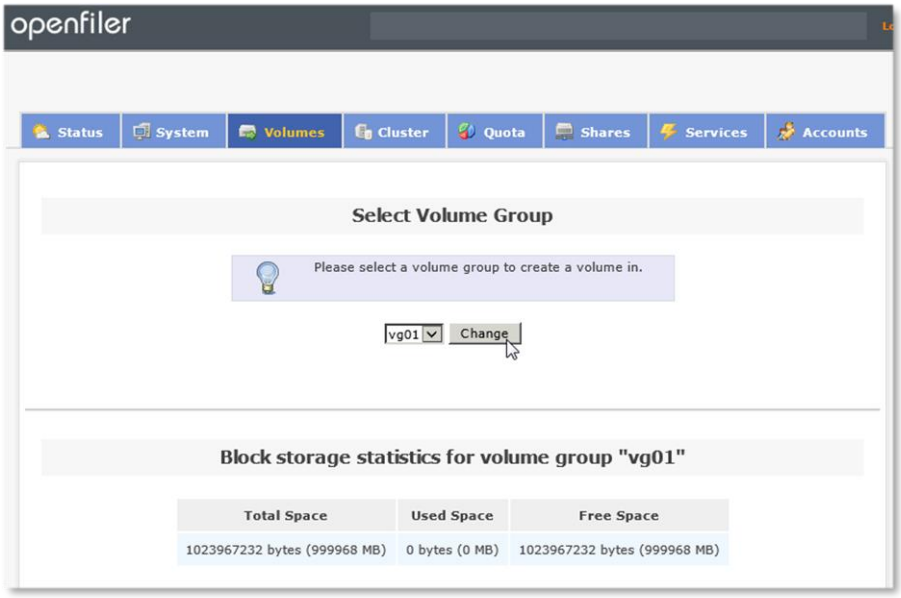
5. Then add a new volume group by giving it a name and selecting the appropriate partition. Click **Add volume group** to complete the operation.



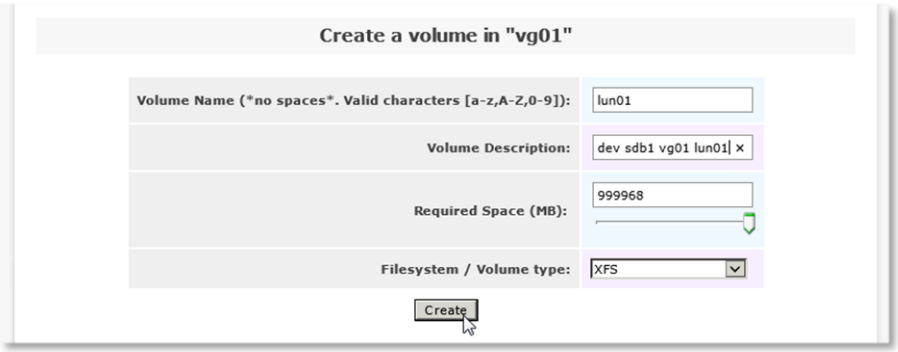
6. Click the **Add volume** link, select the appropriate volume, and scroll to the bottom of the page.



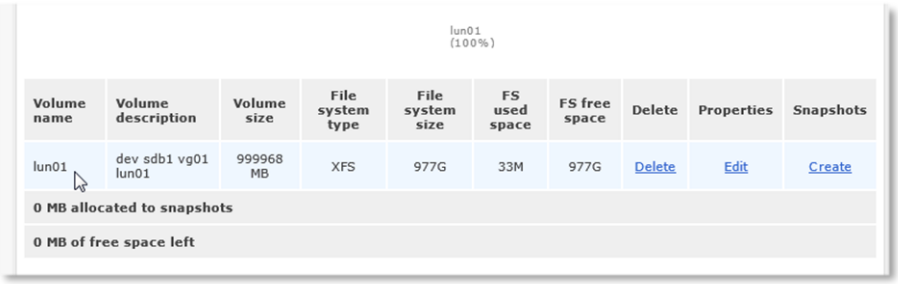
7. Click **change**



8. Add a volume name, description, space, file system format (XFS), and click **Create**

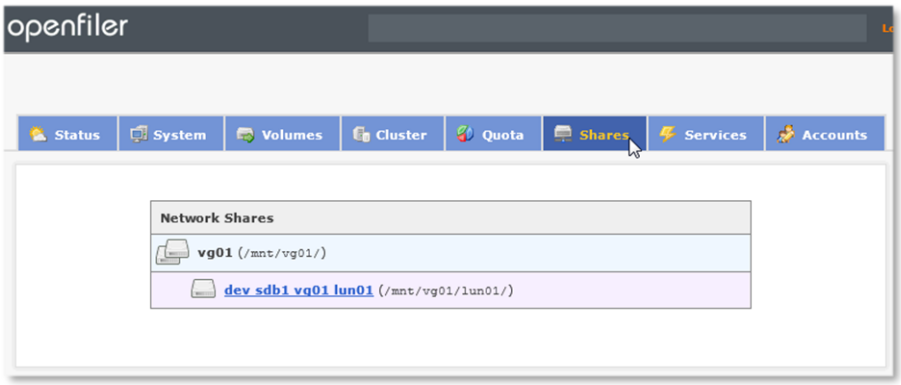


You should now see an XFS formatted partition with your specified size parameters.

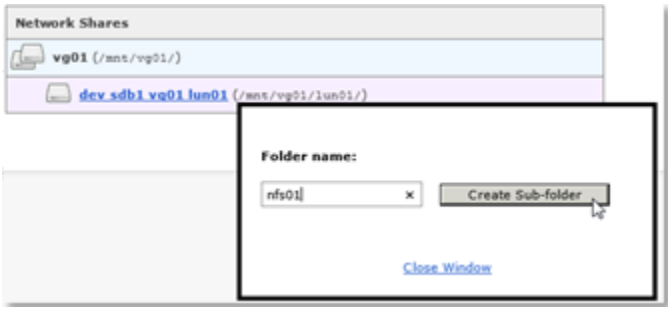


9. Click on the **Shares** tab in the GUI to continue.

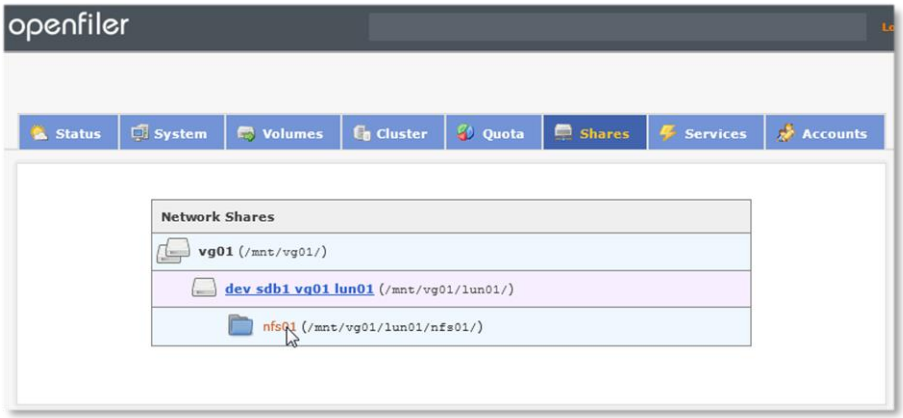
   Note: *According to Openfiler, XFS is the recommended format for partitions used for NFS exports*



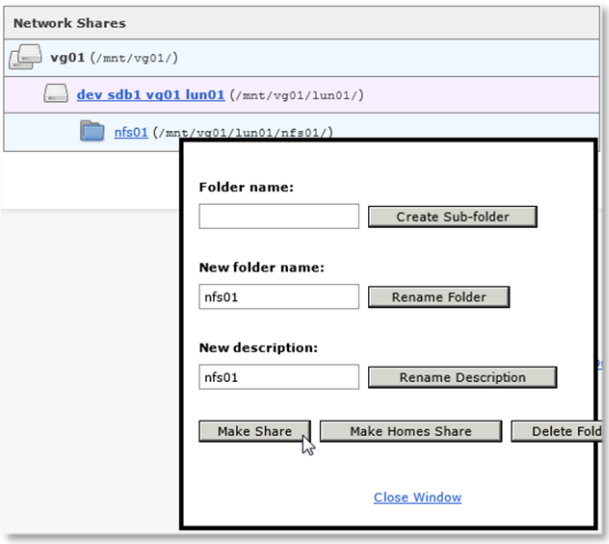
10. Create a Sub-folder by clicking on the Share name, typing in a name in the **Folder name** box, and clicking **Create Sub-folder**



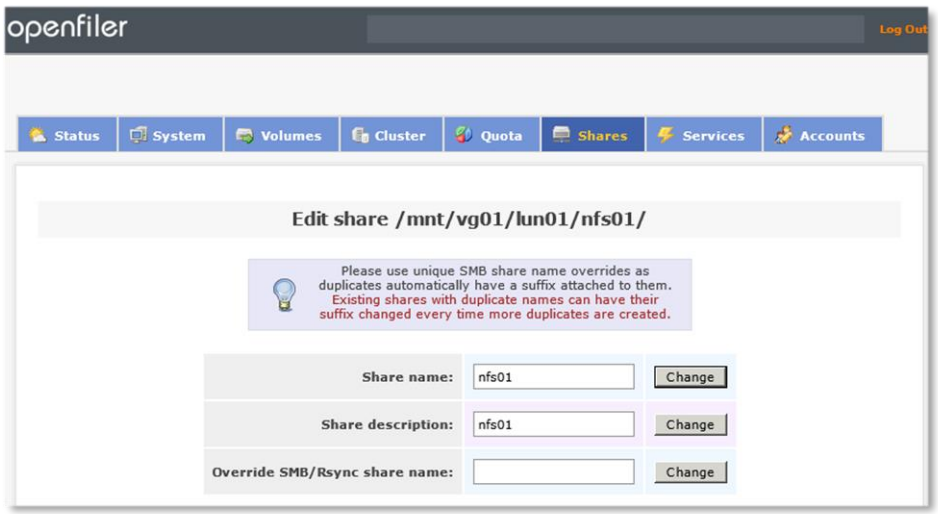
11. After creating the sub-folder, click the new sub-folder link



12. This will open the Make Share dialog. Enter any folder or description information, and then click **Make Share**



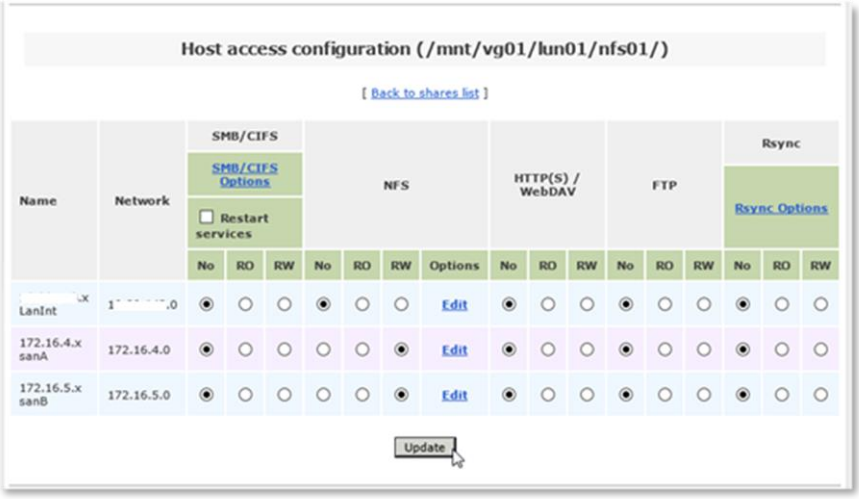
This will land you out in the 'make share space.'



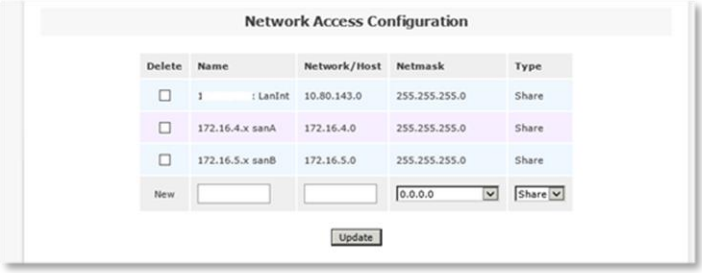
13. Scroll down the page half way, click on **Public guest access** and click **Update**. Since our test configuration is in an isolated lab, this setting is appropriate

Note: If you are setting up a production NFS share, please reference the Openfiler* manual for setting up appropriate access and accounts.

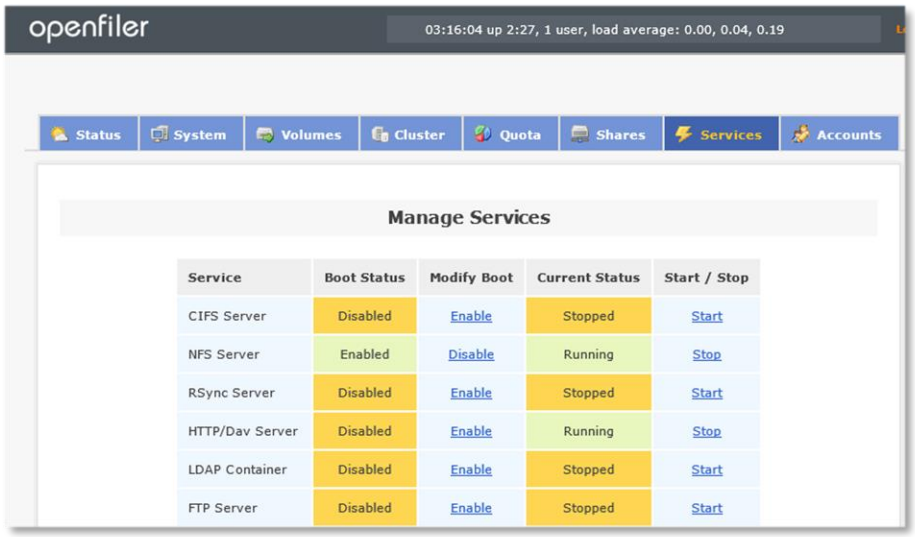
14. After setting share access, scroll down the page all the way and modify the **Host access configuration** to allow read/write (**RW**) on your networks that require NFS access, and click **Update**



15. If you have not yet configured **Network Access Configuration**, go back to the **System** tab in the GUI and at the bottom of the page, create the appropriate network entries.

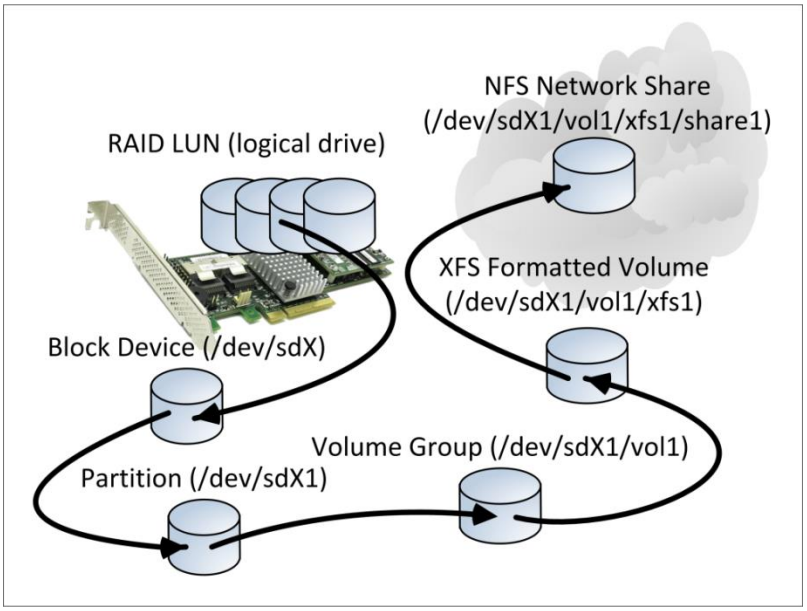


16. Once network access is configured, navigate over to the Services tab, enable the NFS service and click **Start**. If the service is already enabled and started, you will need to stop and restart the service for your changes to be reflected in network access rights.



This section covered the basic NFS export setup. You can go back and create more partitions, volume groups, and volumes if you have additional controllers and disks to configure. In our example, /dev/sdb contains 1 partition, /sdb1. That partition is put into a volume group, that volume group contains volumes, which in our case is an XFS formatted partition that we share out over the network.

For Openfiler* the storage hierarchy looks something like this:

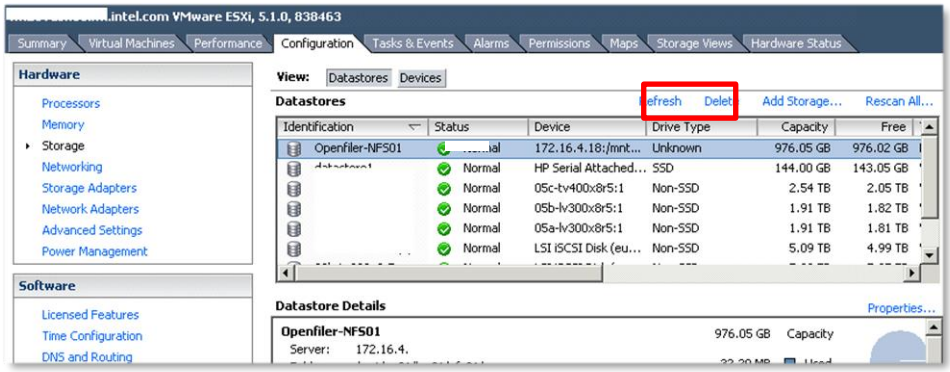


5.6 VMware NFS Configuration

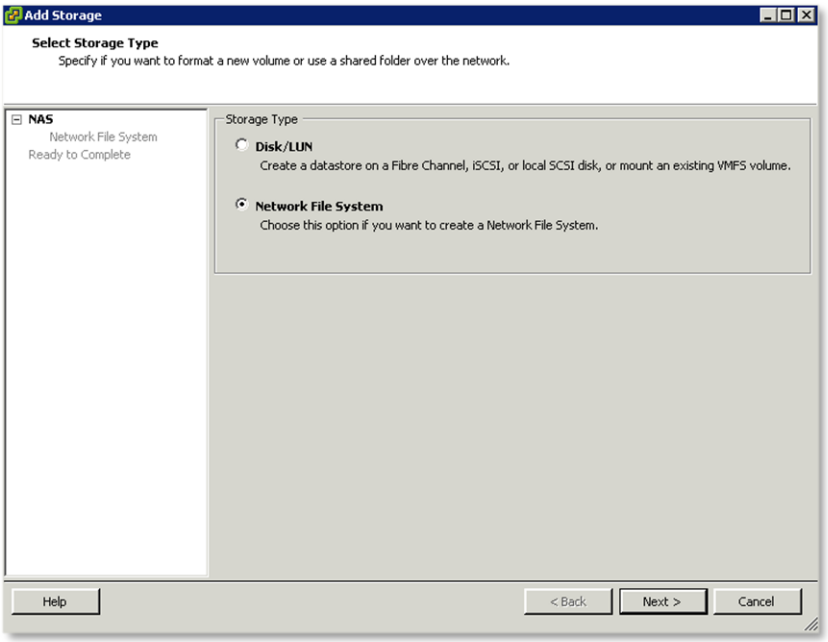
After the NFS export configuration is complete on the Openfiler:

1. Open the vSphere* client console and add an NFS mount point in the VMware host. For our example we used the following private 172.16.x.x address and the mount point listed in the Openfiler console we created in the previous steps.

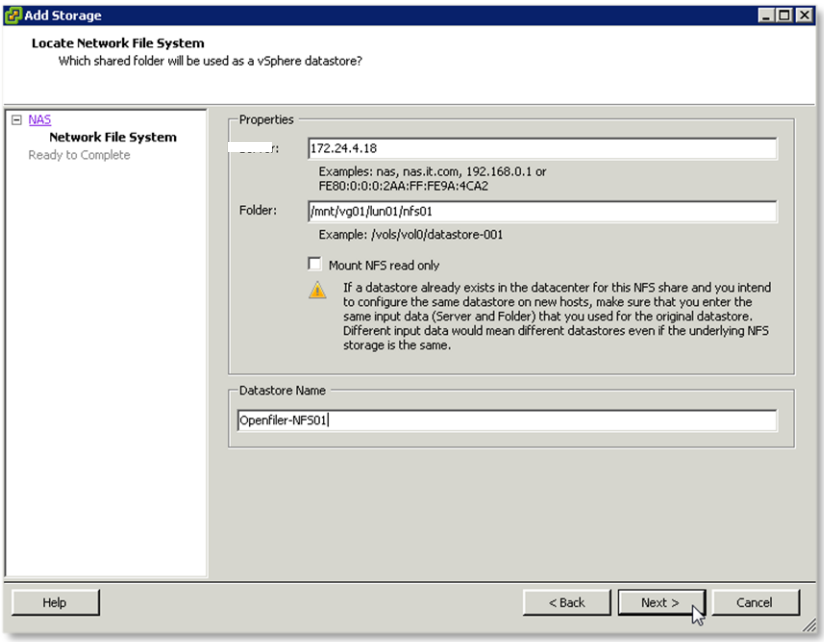
2. On the **Configuration** tab of a selected host, click on **Add Storage...**



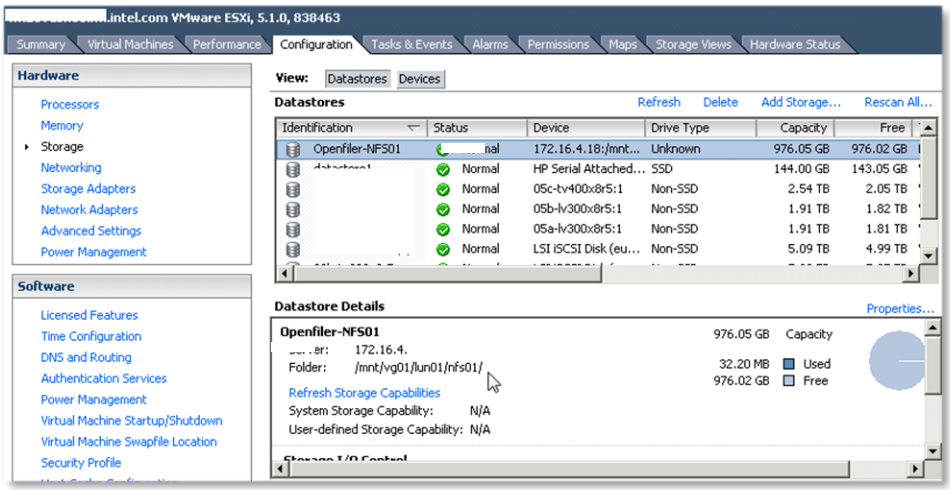
3. Click the **Network File System** button if it is not selected, and click **Next**



4. Follow the series of instructions for adding a network file system mount, IP address, and path.



Once this process is finished, you should end up with something similar to the screen shot below, which shows an NFS datastore in the VMware storage configuration tab.



5.7 Review Optimizations for SSD-based LUNs

Now that the Openfiler is set up with an NFS export and the VM host is connected to that NFS mount point, don't forget about those optimizations in Section 5.4.

- Set the I/O scheduler to **noop for SSD based luns**
- Add the SSD feature key and battery backup to the controller
- Enable individual disk cache in the controller
- Enable controller write back cache
- Enable controller adaptive read-ahead cache
- Adjust the controller read patrol rate
- Use the parted command to align SSD partitions to 4K

This completes the setup of both VMware and Openfiler, you system should be ready for testing.

5.8 A Note on Calculating SSD Endurance in RAID

When using RAID5 and SSDs, the endurance of the array = (N-1) where N is the number of disks in the RAID5 array. Table 3-1 shows an example calculation for a 400GB Intel® SSD in 5x disk RAID5 set. In this table, we use the formatted capacity of the drives in Bytes using base 2 (/1024) instead of base 10 (/1000). This calculation provides a more accurate representation of what an end-user will see in a formatted volume or RAID controller interface.

Table 5-1. Sample Endurance Calculation – 5x Disk RAID5 Set

Per Drive GB Usable Capacity	390 GB
Per Drive Overwrites/Day	10 Overwrites
Per Drive Lifetime (years)	5 Years
Per Drive Lifetime (petabytes) = Usable capacity *overwrites/day *days *years/1024 (TB)/1024 (PB)	6.8 Petabytes
Number of disk in RAID set	5 Disks
Equation: Lifetime of 5 disk RAID5 = (5-1) *endurance (6.8 PB) or 27.2 petabytes of write activity.	

5.9 Configuration and Setup Results

5.9.1 Results

- 4x hosts running against 6x NFS targets with 12x VMs
- Each VM configured with a 100% random 4k workload of 90% read 10% write
- Sustained 150K 100% Random IOPs across a 10GbE network at 2.5ms latency
- Power footprint of 650 watts, rack space of 2U, and estimated retail pricing under $45K

5.9.2 Conclusion

The specific configuration of Intel® Xeon™ processors, Intel® X520-DA2 10GbE, Intel® 6Gb SAS RAID, and Intel® SSD DC S3700 Series solid-state drives with the Openfiler 2.99.2 software stack using the methods described in this paper provides a high-IOPS solution capable of hosting hundreds of virtual machines. Given the higher endurance of the Intel® SSD DC S3700 Series, the solution presented here will easily last through 24-hour operation with a 30% write workload for 8-10 years. Finally, the small power and datacenter footprint plus low-cost per IOPS when compared to traditional SAN supplies the business value required to pursue this non-traditional software based storage solution. With the price of SSDs continuing to decrease as the size increases, the continuous move toward the cloud and VMs, and increasing demand for IT services, the enterprise cloud community must look at alternative methods for supplying the datacenter with IOPS. Intel® SSDs and platforms will play an integral part in changing the face of storage in the datacenter.

§