XenServer Storage Overview

Overview

This document describes how XenServer provides and keeps track of the storage supplied to its guests. The first section is a reminder of how Linux looks at storage and the second section builds on that to explain XenServer storage. Basic knowledge of Linux is required, as some standard tools are used.

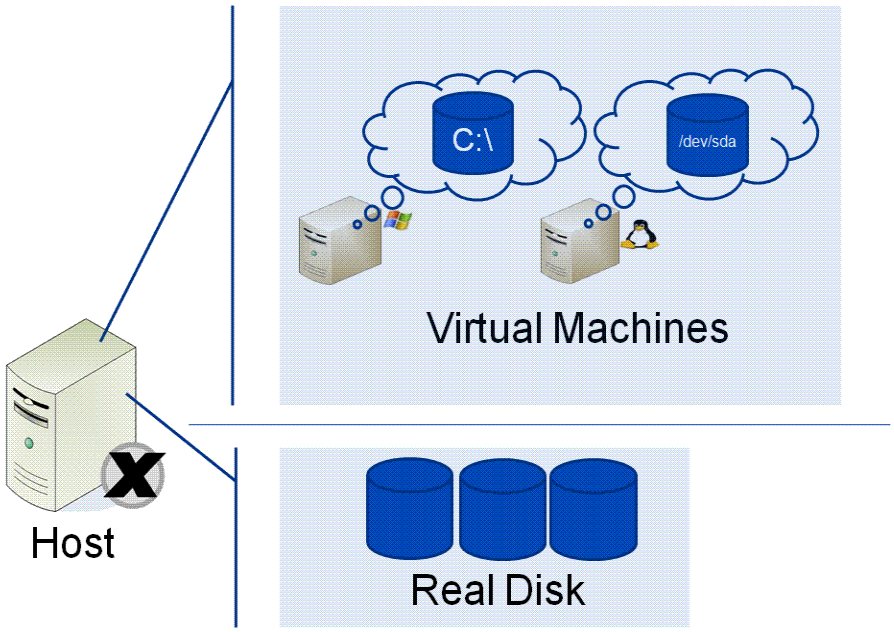
Target Audience

This document has been written for information technology (IT) specialists who are responsible for planning and designing infrastructures that include an existing storage infrastructure, and who therefore want to quickly understand the salient concepts and terminology. These specialists include consultants, internal IT architects, and any others who may need this information.

CITRIX®

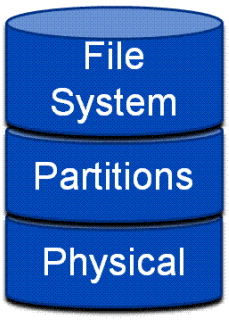
Table of Contents

Introduction



In the virtual world of a XenServer, the virtual machines (VMs) behave like real machines with locally attached disks. In reality, the XenServer host has allocated a chunk of real disk space and, through the hypervisor and Control Domain (Domain 0), has made them available as a disk resource to the VMs.

Linux and Physical Disks




```
mount /dev/sda1 /mnt/mymount
```

`/dev/sda1, /dev/sda2 etc`

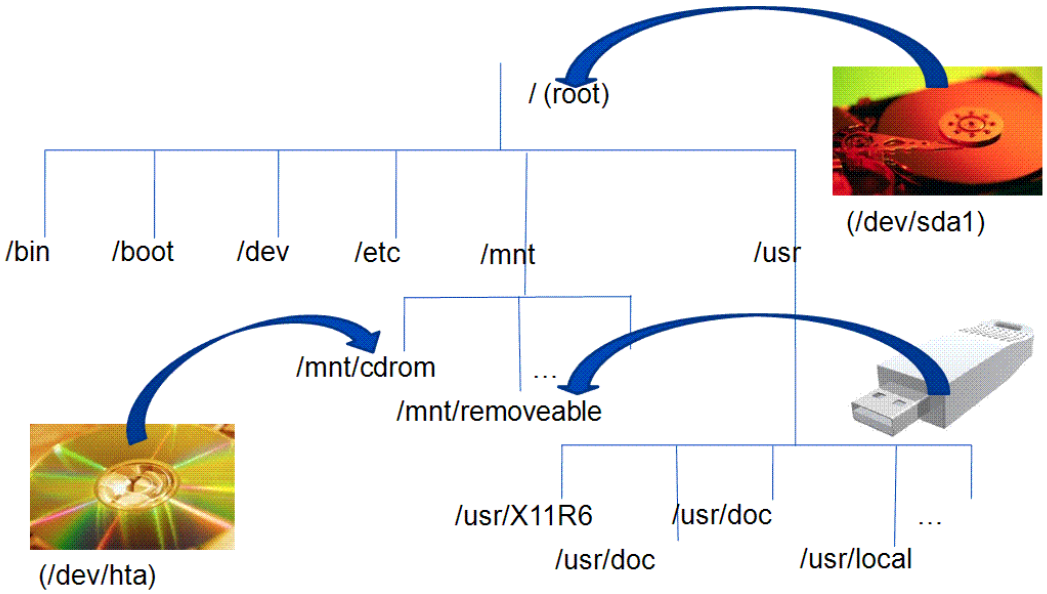
`/dev/sda`

Because the storage access is through the Control Domain, it is worth revising what we know about Linux and how it views disk storage. In Linux, typically the physical disk itself is split into partitions. Remember that all devices in Linux are accessed through device files.

As an example, for the first Small Computer System Interface (SCSI) disk on the system, the device file name might be /**dev**/**sda** (SCSI disk “a”). Partitions of that disk will have device file names like /**dev**/**sda1**, /**dev**/**sda2**, and so on. The device /**dev**/**sda1** acts like an autonomous disk (except it cannot be sub-partitioned). This disk can then be mounted onto

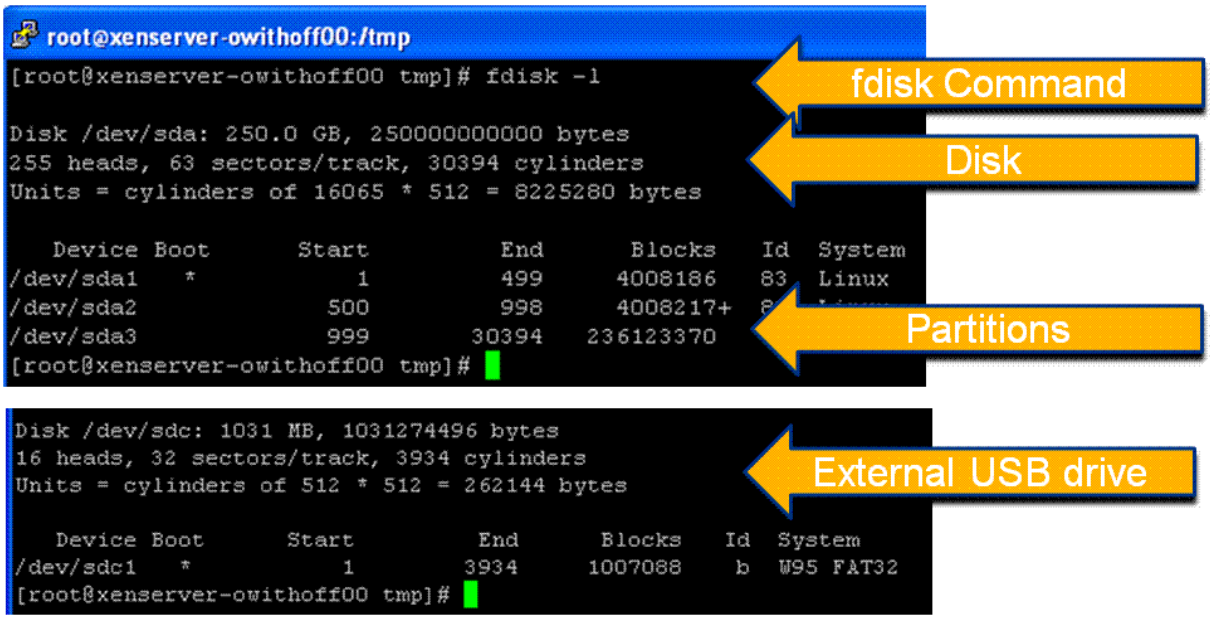
the Linux file system using the **mount** command. Typically when the system boots, a disk is mounted over "/", which is the root directory of the file system. For a disk to be mounted onto the file system, it must be formatted to have a file system placed on it. This is done using the **mkfs** shell command.

A disk or partition does not have to be formatted to be used in a file system; it can be used as a raw device. That is, any format you choose can be imposed on the disk device. Also, these concepts apply equally to external (for example, USB) disk drives that can be partitioned, formatted, and mounted onto an existing file system.



The Linux file system gives you a unified view of all storage and has a single root directory, indicated by a forward slash (/). Within the root directory is a hierarchy of files and directories. Parts of the file system can reside on different physical media, such as hard disks, flash disks, and CD-ROMs. If you are familiar with MS-DOS or Windows, note that you will not find drive letters in Linux. All disk drives and CD-ROM drives are part of a single file system hierarchy.

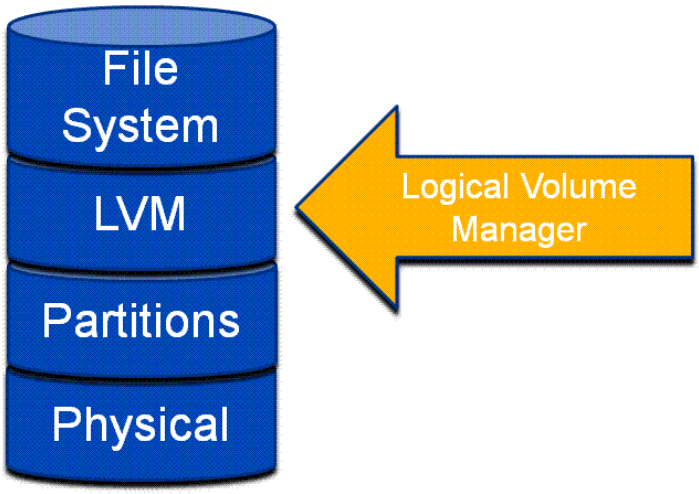
When you install XenServer, the default file system type used is **ext3**, which is actually sitting on top of the Logical Volume Manager (more on this later).



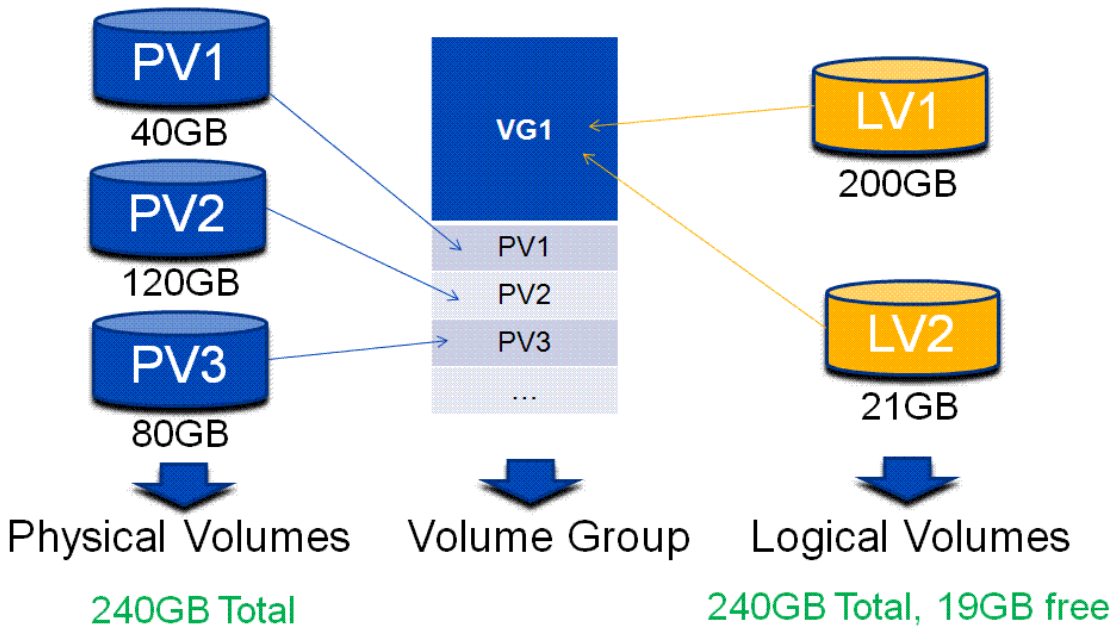
How does Linux let us know which disks it has detected? The **fdisk** command is the partition table manipulator for Linux, but it can be destructive when used improperly. You can, however, use **fdisk –l** to check which disks and partitions on those disks Linux has detected. Remember that you can read the manual pages for Linux commands using the **man** command. For example, **man fdisk**.

The bottom part of the following graphic shows the fdisk output after an external flash disk has been inserted into the XenServer system.

Linux and Physical Disks



Linux can also add another abstraction layer to manage the different disk volumes. We have seen that you can split a disk up into partitions, but how do you spread one partition along many disks?

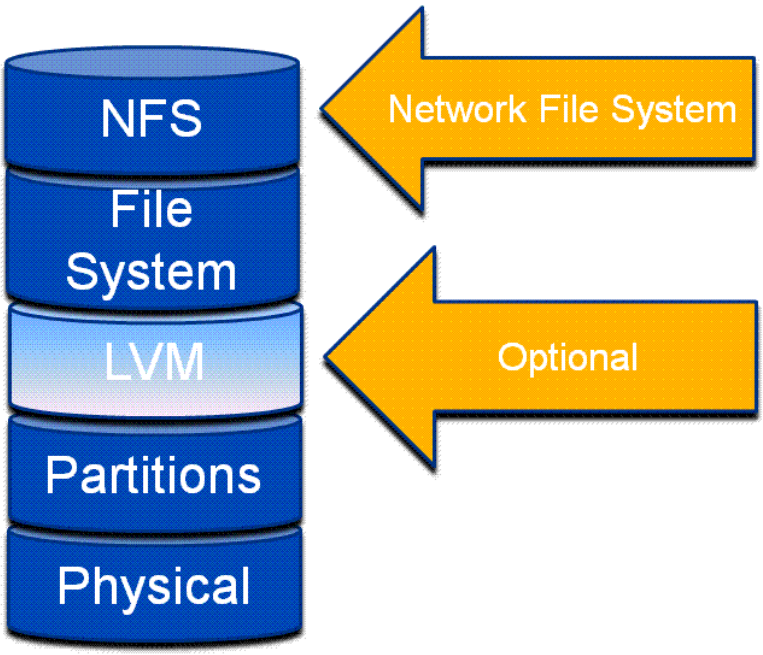


The Logical Volume Manager (LVM) gives Linux the ability to abstract physical volumes into logical volumes. In this example you see all of the physical volumes (PVs) as part of a single Volume Group (VG). The Logical Volume Manager can support one or more VGs, and a VG acts as a pool of storage from which you can create Logical Volumes (LVs). In this example there are two LVs that can now be used as though they were PVs.

Because a PV is represented in Linux as a device file (for example: /dev/sda), that device file may represent directly attached storage (local storage) or a Logical Unit Number (LUN) on a Storage Area Network (SAN). The LVM can therefore be used to manage local or SAN-based Storage Repository (SR) targets.

As with PVs, LVs do not have to be formatted to be part of a Linux file system; they can be used as raw storage, which is how XenServer uses them. When the LVM is being used to support SRs, you will find one LV on the XenServer for each VDI (more on SRs and VDIs later).

Network File System – NFS



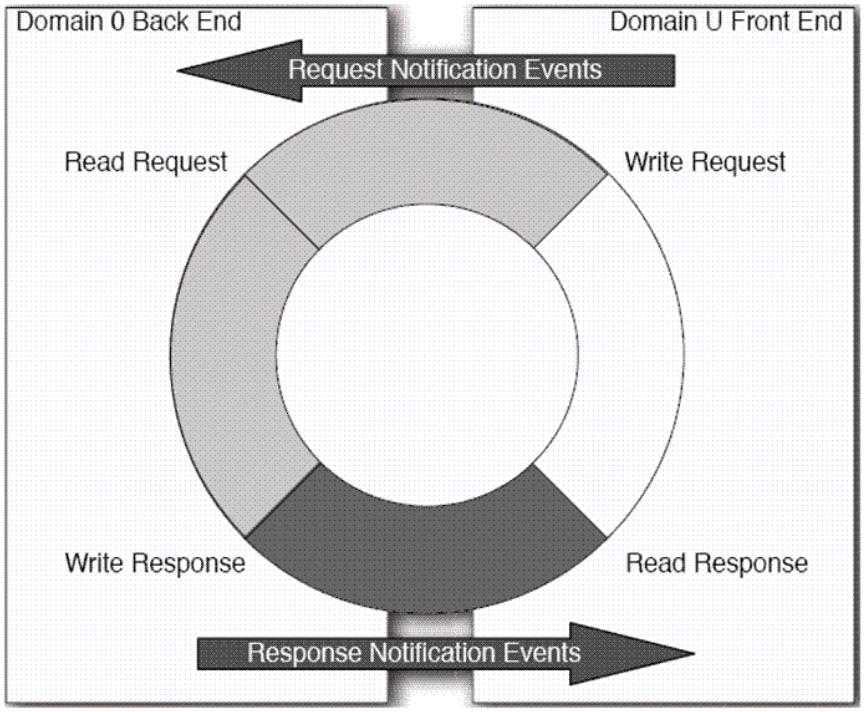
Network File System (NFS) is a network file system protocol originally developed by Sun Microsystems in 1983, allowing a user on a client computer to access files over a network as easily as if the network devices were attached to the computer's local disks. It is equivalent to file shares in Windows. Early versions of NFS used UDP/IP, but version 3 (v3) uses TCP/IP. Both are supported by XenServer; v3 is supported by default. NFS is also becoming ubiquitous on Network Attached Storage (NAS) appliances.

A NAS (or filer) is a preferable solution, because they have a high degree of fault-tolerance and can provide write acknowledgements without the write being physically on disk. In this case, the filer should be set to present the mount point (share) as an asynchronous mode export. Without this, the XenServer will wait for a write acknowledgement from the NFS service, which will adversely affect the speed of the VM.

You should not set your filer to present the mount point (share) as an asynchronous mode export if it cannot guarantee data integrity in the case of the failure of one of its components (for example, a disk).

You can configure an NFS server as a virtual machine if you want to experiment. There are tools like OpenFiler (http://www.openfiler.com), which can be installed as a VM or on a bare-metal computer and will support multiple protocols for access to storage (including software iSCSI).

The XenServer Storage driver model

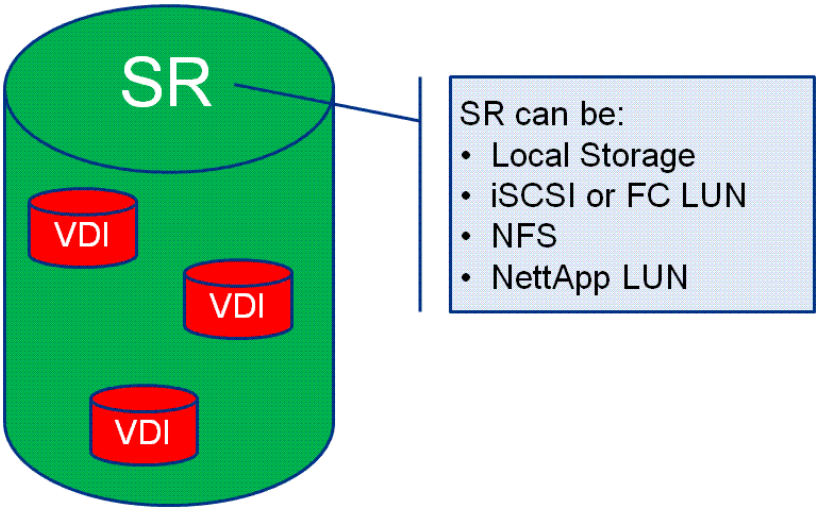


From *The Definitive Guide to the Xen Hypervisor* (Prentice Hall ISBN: 013234971X)

The Xen model for device access makes it very efficient both at runtime and when introducing new device support. The Xen hypervisor essentially knows nothing about the real hardware, but uses a special guest (which used to be called Domain 0, but is now called the Control Domain) that talks to the real network and storage. The advantage is that support for new hardware is placed in the Control Domain, which is a standard Linux guest. Theoretically (although not supported) you could install new storage device drivers into the Control Domain to support different or new kinds of devices. As long as these devices can be enumerated to the hypervisor as standard Linux device drivers, the hypervisor should have no trouble picking up that support.

For any guest (except the Control Domain), the device drivers they use are usually the ones supplied as part of the XenServer tools installation. These drivers are designed to communicate with the hypervisor, which in turn communicates with the Control Domain. The guest device drivers (except the Control Domain) for storage end up being a very quick communication mechanism to the hypervisor through a shared circular buffer. In Xen, this is known as the Split Driver Model, where the back-end of the driver is contained in the hypervisor and the front-end in the guest.

XenServer Storage Objects

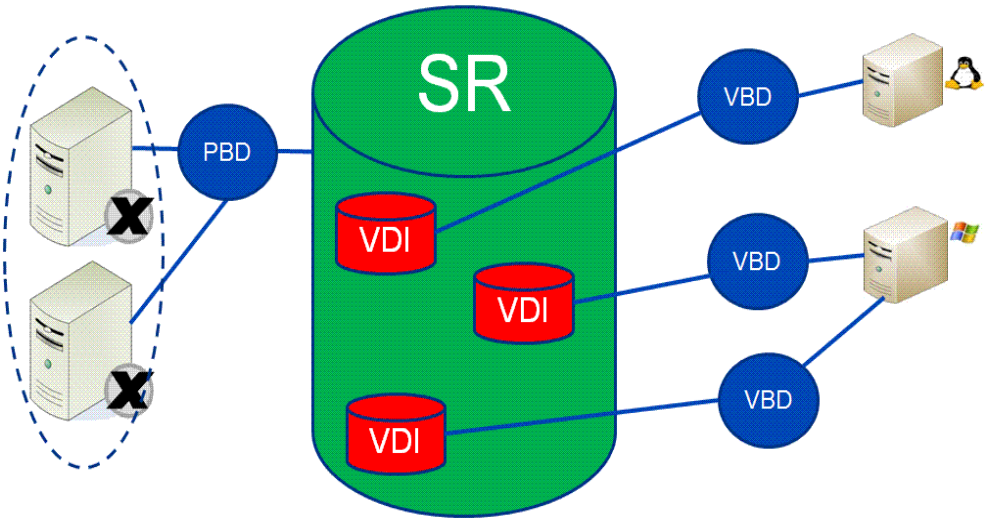


A key object in XenServer storage is a Storage Repository (SR). This is the physical on-disk structure or format imposed on the available physical storage by XenServer. XenServer allocates Virtual Disk Images (VDI) on an SR, which is what a VM sees as a physical disk.

Generally, the target storage for an SR can be one of:

- Local storage on the XenServer host. XenServer supports two types of local storage: The ext file system, and Logical Volume Manager.
- A LUN on an iSCSI SAN, or a LUN on a Fibre Channel (FC) SAN. Managed separately from XenServer, these will appear to XenServer as normal device files, on which an SR can be imposed. In these cases LVM is also used.
- Storage through a NetApp or Dell EqualLogics storage appliance. Management is supported directly through the snap-in architecture of the storage administrator in the XenCenter.
- NFS (more on NFS later)
- udev: externally plugged-in devices, such as external hard drives and flash drives.

Each XenServer host can use multiple SRs and different SR types simultaneously. These SRs can be shared or dedicated between hosts. Shared storage facilitates shared SRs, and is available to more than one XenServer host. If the SR is shared, a VDI can be started on any XenServer host in a resource pool that can access that SR.



- The Storage Repository (SR) is a format imposed by XenServer on available storage. The SR format, or type, will vary depending on the physical storage and the choices the administrator makes when creating it.
- The Physical Block Device (PBD) is what describes the SR so that it can be plugged into the XenServer (perhaps by multiple XenServers in a pool that are sharing the storage)
- The Virtual Disk Image (VDI) is a piece of an SR that is carved out to be a virtual disk for a VM. The format of the VDI depends on the type of the SR it is contained in.
- A Virtual Block Device (VBD) describes the VDI so that it can be plugged into a VM.

XenServer Storage Attributes

- **Shared**: This means that the storage is based on a Storage Area Network (SAN) or NFS, and so can inherently be shared by multiple XenServers in a pool. This is essential for VM performance.
- **Sparse allocation**: The expansion of a VDI file is allocated as the VM writes data to it (the VM is writing to what it thinks is a local drive). The VM VDI files take up only as much space as is required. For example, if a 20 GB VDI is allocated for a new VM and an operating system is installed, the VDI file will only reflect the size of the operating system data that has been written so far (plus some format overhead), not the entire 20 GB.
- **Resizable VDI**: Used on a detached VDI to increase the size of that VDI. The VM operating system should have the capability to extend its format into the new space.
- **Fast Cloning**: Where a new VDI is created for a VM, but is in essence a pointer to an existing VDI. Any blocks that are read from the new VDI will actually come from the old VDI, but any changes to blocks or newly added blocks are put in the new VDI. This means a VM can be cloned almost instantaneously because the space is allocated on an as-needed basis. For VHD types (next section) of VDIs, this can lead to long chains if this process is repeated many times.

VDI Formats

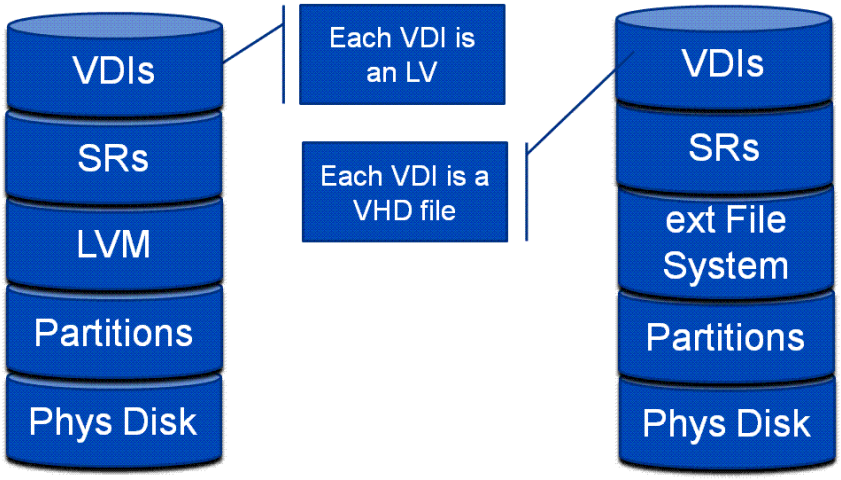
The format of a VDI will be one of three types depending on the type of SR it is contained in and which features are needed:

- **VHD**: Virtual Hard Disk is a technology used in Microsoft Virtual PC (originally created by Connectix, then bought by Microsoft). This format is also used by Microsoft's Hyper-V technology and is how XenServer supports VMs created by Hyper-V. These are essentially flat files stored either on an NFS SR or a local storage external SR. The VHD format supports sparse allocation and fast cloning. It is also shareable when the containing SR is of type NFS. VHD files may also be chained, allowing two VDIs to share common data. In cases where an NFS-based VM is cloned, the resulting VMs will share the common on-disk data at the time of cloning. Each will

proceed to make its own changes in an isolated copy-on-write (CoW) version of the VDI. This feature allows NFS-based VMs to be cloned quickly from templates, facilitating very fast provisioning and deployment of new VMs. The VHD format also supports snapshots without the need for support from the storage back-end.

- **LVM**: The Logical Volume Manager is used on a raw block device (either local or SAN-based), and a VDI ends up being a Logical Volume (LV) in a Volume Group (VG). This format supports VDI resizing. It is also sharable when the storage originates from a SAN.
- **Supported storage appliances**: For the NetApp appliance, the VDI ends up as a LUN in a NetApp volume. The Dell EqualLogics filer will also store a VDI as a LUN. This format supports sparse allocation, VDI resizing, fast cloning, and is shareable. Explicit support for third-party storage appliances is provided through the XenServer Storage Plug-in architecture. OEM editions of XenServer may support different storage appliances out-of-box.

Local Storage



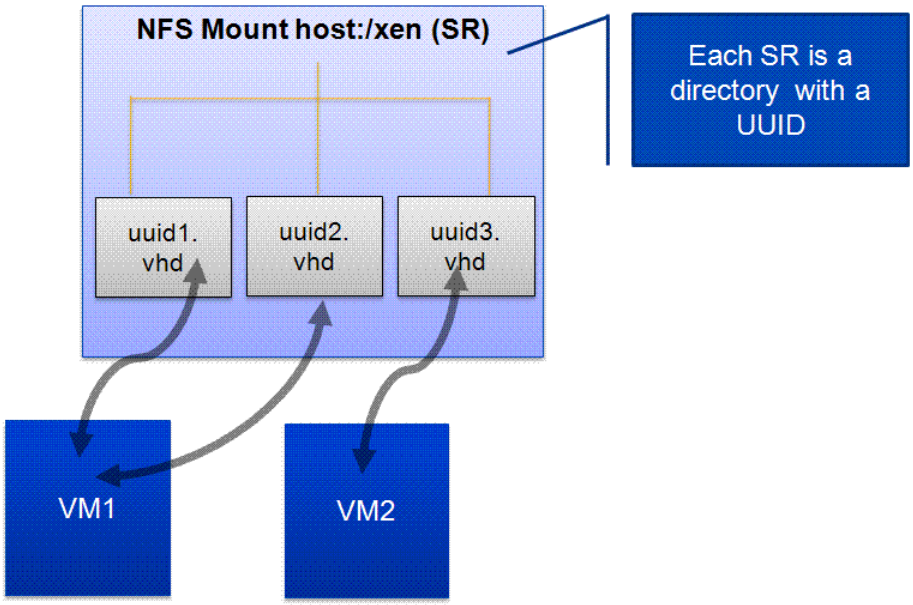
The disks that are installed in the XenServer host or belong to a Directly Attached Storage (DAS) system are considered local storage by XenServer: This storage is only available to that one XenServer host, thus SRs created on them will not be available for sharing.

Two types of SRs are supported on local storage: **ext** file system and **LVM**. Ext is used when you need a VDI of type VHD on local storage (a VHD is a flat file). This use case has to be implemented using the command line interface (CLI); there is no interface for it in XenCenter (use **xe help sr-create** at the CLI for more details). Also see *How To Change the Default Storage Repository to File-based VHD-on-EXT3*: http://support.citrix.com/article/ctx116324.

Note: You will see that an ext3 file system is installed by default along with the LVM on any XenServer machine. The file system in this case is not used to support SRs, but to support the Linux Control Domain and Hypervisor. As shown above, you can use **fdisk –l** to show the physical disks, and how those disks are partitioned. For example, a computer has a single 250 GB disk that corresponds to the Linux device file /**dev**/**sda**. You can then use the **df** command to show that only the first partition (in this case /**dev**/**sda1**) is mounted over "/". This command also shows the amount of free space. In this example, a default installation of XenServer on a 250 GB drive only allocated 4GB of disk for its supporting ext3 file system.

Where has the rest of the local storage gone? If you use the **vgdisplay** (Volume Group Display) command, you will see that the remainder has been allocated to the LVM to support local storage SRs.

NFS based Storage



In brief:

- All servers connect to the same NFS share
- VDIs are files on the NFS share
- Only the server running a VM connects to the individual virtual disk for that VM
- The pool master coordinates which servers connect to which files

The NFS service could be hosted by a NAS Filer (such as a NetApp appliance) that supports NFS, or perhaps it could be a Linux VM hosting an NFS share. Because the NFS share is accessible over a network, this means it is shareable among XenServers in a pool. VHD is the only VDI type supported on an SR based on NFS. The VDI is allocated as sparse, so the administrator needs to ensure that there is enough space on the share to cover the maximum size of the VDI.

Usually the NFS shares need to be exported from the server. See this article for Linux: http://nfs.sourceforge.net/nfs-howto

If you are using a filer, the filer admin interface will probably do this automatically behind the scenes. The XenServer host (which is the NFS client in this case) just needs to know the IP address of the NFS server and the (Linux style) path of the exported directory. You can use the **sr-probe** command to probe the IP address for a list of valid shares.

When you create an NFS-based SR, a directory will be created in that share with a name that is the Universally Unique Identifier (UUID) assigned to that SR. If created using the **sr-create** command in the CLI, the UUID will be returned when complete.

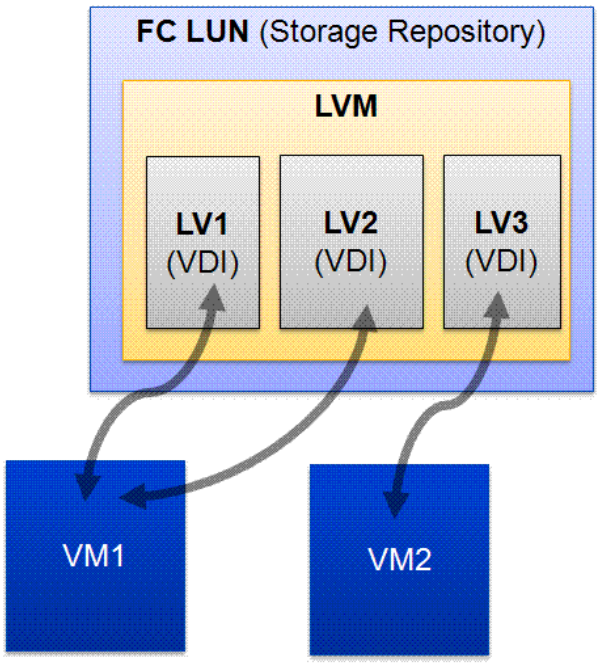
Note: You may have to re-synchronize any XenCenter sessions if you create an SR through the CLI.

Any VDIs that are subsequently created in that SR will appear under the directory (mentioned above) as a flat, sparsely allocated VHD file (that is, with a **.vhd** extension). The name of the file will be the UUID allocated to that VDI.

To import a VHD file, place the file into the directory that has the UUID of the SR that will contain it. Make sure you give the file a UUID (of your making, or see http://createguid.com). You can then use **xe sr-scan…** to have the imported file recognized by XenServer.

By using the **mount** command at the CLI, you will be able to see that your XenServer has actually mounted the remote directory corresponding to the SR into its local file system.

Fibre Channel- and iSCSI-based Storage



In brief:

- All servers connect to the same LUN
- The LUN is partitioned into **logical volumes**
- Only the host running a VM connects to the individual virtual disk for that VM
- The pool master coordinates which hosts connect to which VDIs

SAN storage support comes as one of three types:

- **An iSCSI Software initiator**: Uses Open-iSCSI software to facilitate iSCSI initiation.
- **An iSCSI hardware initiator**: Mitigates the overhead of iSCSI and TCP processing and Ethernet interrupts, and therefore may improve the performance of servers that use iSCSI. An iSCSI host bus adapter (HBA) implements a hardware initiator and is typically packaged as a combination of a Gigabit (or 10 Gigabit) Ethernet NIC, some kind of TCP/IP offload technology (TOE) and a SCSI bus adapter (controller), which is how it appears to the operating system.
- **Fibre Channel HBA**: The only way to connect to a fibre switch. Note, though, that because these are specialized networks, that only certain HBAs from Qlogics and Emulex are supported.

For both software and hardware iSCSI, the XenServer has a built-in unique initiator name in the form of an iSCSI Qualified Name (IQN). This can be changed through the general tab in XenCenter, or through the CLI.

SAN-based storage is similar to local storage, in that the LVM is leveraged on the LUNs that are annexed by XenServer for use as an SR. Once you go through the **sr-probe** > **sr-create** cycle, XenServer ends up with a device file that makes the LUN look to Linux like a regular, locally attached block device. On the example system described above, after creating an **lvmoiscsi** (LVM over iSCSI) SR associated with a LUN contained on an OpenFiler appliance, it found a new block device /**dev**/**sdc**. By using the shell command **fdisk –l** on the XenServer, it showed this device, and that it did not contain any valid partitions. You may also find many soft links to this block device, especially in the directory /**dev**/**disk**, which enumerates the same block device in different ways.

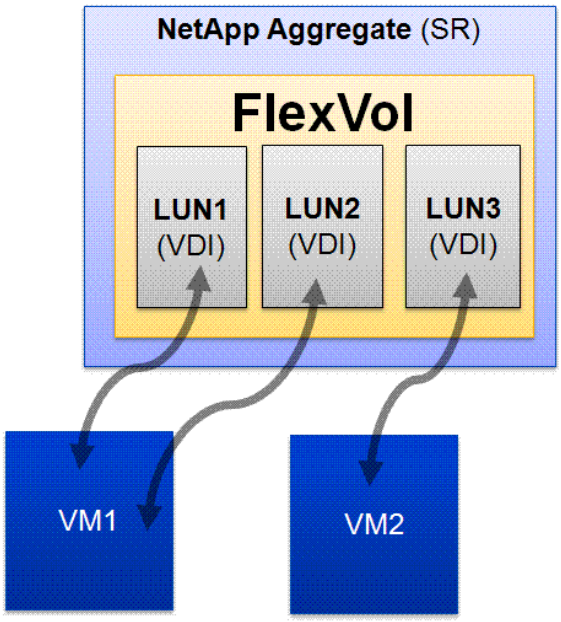
As an example, using the **sr-list** CLI command can show the block device file name for an SR based on a SAN LUN. Part of the output may look like the following:

```
content-type ( RO): user
      shared ( RW): false
other-config (MRW):
   sm-config (MRO): devserial: scsi-14f504e46494c45006b586d7658482d434f58662d34503039;
multipathable: true
       blobs ( RO):
```

In the example system, the sm-config item shown corresponds to this file: **/dev/disk/by-id/scsi-14f504e46494c45006b586d7658482d434f58662d34503039**, which itself is a soft link to **/dev/sdc**.

Using the **pbd-list** command will show all the attributes needed by XenServer to connect to the LUN. This includes the SCSI ID, which turns out to be the same large number in the 'device by-id' name in Linux (above).

NetApp-based Storage



In brief:

- XenServer provisions raw iSCSI LUNs using NetApp Application Programming Interface (API)
- Each NetApp **aggregate** contains one or more SRs
- Each aggregate contains one or more **FlexVols** containing the LUNs
- Hosts connect to LUN through iSCSI and map it into VM

In contrast to generically supported SAN storage, the VDIs on a NetApp appliance are implemented using a raw iSCSI LUN (as opposed to LVM). In theory this means a LUN that contains a VDI can be mapped to a real physical machine as well as the VM that created it. This will only work exclusively though; it has to be one or the other and is an unsupported configuration.

Multipath I/O

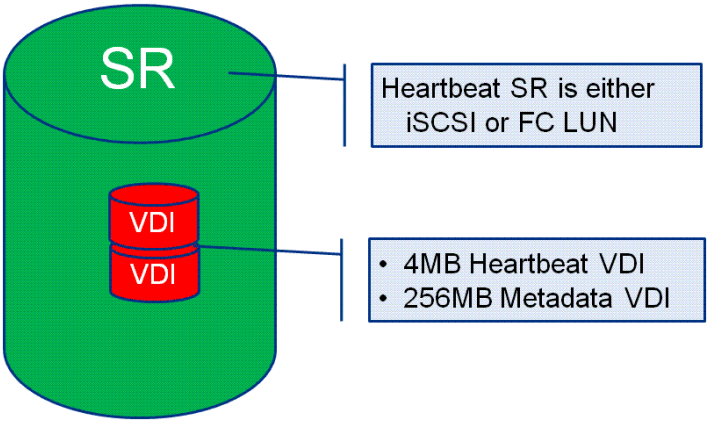
Multipath I/O is a fault-tolerance and performance enhancement technique where there is more than one physical path between a XenServer host and its SAN-based mass storage devices.

A simple example would be a disk connected to two FC ports. Should one controller, port, or switch fail, XenServer can route I/O through the remaining controller transparently to the application, with no changes visible to the applications, other than incremental latency.

This is achieved by use of the Device Mapper kernel code. The Device Mapper (DM) is typically used to map block devices together, with some filter code in between. For example, the LVM uses Device Mapping extensively to transparently map multiple block devices to a single block device.

For Multipath I/O, if the target Linux device file is one created by the DM, read and writes can be filtered through the Multipath I/O code to determine the best device path in reality. The DM also allows for the stacking of mapped devices on top of others, so that Multipath I/O paths may lead to a logical volume, which itself is a mapped device. See: *Multipath support in the device mapper*: http://lwn.net/Articles/124703

Storage & High Availability



The XenServer High Availability(HA) feature employs two mechanisms: A networking heartbeat and a storage heartbeat. When you enable HA in a pool, you must nominate an iSCSI or FC storage repository to be the heartbeat SR. XenServer automatically creates a couple of small virtual disks in this SR. The first disk is used by every physical host in the resource pool as a shared quorum disk. Each host allocates itself a unique block in the shared disk and regularly writes to the block to indicate that it is alive.

More Information

http://support.citrix.com/article/ctx118397

http://community.citrix.com/blogs/citrite/anilma/2008/09/17/Peeking+under+the+hood+of+High+Availability

Notice

The information in this publication is subject to change without notice.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. CITRIX SYSTEMS, INC. ("CITRIX"), SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR DIRECT, INCIDENTAL, CONSEQUENTIAL OR ANY OTHER DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS PUBLICATION, EVEN IF CITRIX HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES IN ADVANCE.

The exclusive warranty for Citrix products, if any, is stated in the product documentation accompanying such products. Citrix does not warrant products other than its own.

Version History			
Author	Version	Change Log	Date
Olivier Withoff Principal Technical Readiness Engineer Worldwide Field Readiness and Productivity	1.0	Initial document	November 2008

851 West Cypress Creek Road Fort Lauderdale, FL 33309 954-267-3000 http://www.citrix.com