Kathleen Locher

Dr. Foxwell

CS 671

May 1, 2008

Storage Management and File System Differences Between Container-based Operating System Virtualization Implementations

Abstract

Container-based Operating System virtualization provides the benefits of consolidation and isolation among systems while achieving significant performance increases over other types of virtualization. When choosing a virtualization environment ease of managing storage is one consideration in the final selection. This paper provides a comparison of the storage management abstractions and file systems available to three different open source container-based operating system virtualization platforms, Solaris Containers, Linux-VServer, and OpenVZ for Linux.

Container-Based Operating System Virtualization

Virtualization is a technique in which the physical characteristics of a computer are hidden from an execution context such as an application or an operating system. Virtualization aims to provide a way to utilize hardware more efficiently while easing administrative pains and increasing security. In order to meet these aims, flexibility, isolation and performance are major factors. Some types of virtualization stress the flexibility to run multiple kernels on dedicated resources, providing a high level of isolation in the process. In contrast to such systems,

container-based operating system (COS) virtualization trades some flexibility and isolation for better performance. COS virtualization relies on a single underlying kernel image to provide virtualization. In return, COS virtualization gains performance benefits. One study found that "container-based systems provide up to 2x the performance of a hypervisor-based [multiple kernel] system for server-type workloads" (Soltesz 13).

One of the driving forces behind virtualization is a desire to efficiently utilize computing resources. Powerful CPUs are often left idle when dedicated to a single system. Data centers seek to consolidate their systems to more efficiently use resources while preserving the isolation that led to hosting the processes on separate machines in the first place. One of the things which the now-consolidated systems share is storage. However, consolidating systems does not cause them to use any less space, and adequate thought must be given to growing space requirements for the systems. Adding additional virtual machines to a physical computer will also require additional storage. All of these changes must be planned for.

Because of the benefits of COS virtualization, several different implementations have emerged. These implementations have a variety of differences and trade-offs. Those trade-offs include the resources which the COS virtualization system can take advantage of and the administrative differences between systems. Solaris Containers, Linux-VServer, and OpenVZ are three examples of COS virtualization systems.

Solaris Containers are a superset of the technology known as Solaris Zones. Solaris Zones are the ability to create COS virtualization environments in Solaris. Containers are zones with resource management added in. This paper will discuss the use of Containers rather than plain Zones. Linux-VServer and OpenVZ both provide the concept of both security containment

and resource management under a single name for each product.

There are many considerations which must come into play when choosing a COS virtualization environment. A brief overview of the highlights of the various COS virtualization systems described in this paper can be found in Table 1 in the appendix. This paper provides data on administration of storage abstractions and file systems by three variants of COS virtualization environments to help to determine one aspect of the flexibility, ease-of-use, and power of each of the three environments. This paper will compare the configuration, installation, and management of the systems with regards to their usage of storage resources and its abstractions. The focus of the paper will be on the file systems and quota control tools available to the various systems from an administrative standpoint and the details of how the three systems utilize storage abstractions.

The Problem

COS virtualization is often used by companies such as web server hosts to efficiently utilize resources while providing isolation between customers' web servers. Isolation can not be effectively guaranteed without proving that a single virtualized environment cannot take all the disk space allocated to many environments. These guarantees must be made for a company to provide the security and isolation necessary to utilize and market virtualization.

As companies using COS virtualization grow and become more popular, their storage needs change. A single virtualized environment may need more storage than it was initially allocated. The addition of more virtualized environments may call for the addition of more space to the whole host node. These changes should be as simple as possible. The COS virtualization

environments support these changes to a greater or lesser extent through the use of advanced storage models and file systems.

Storage Models and Management

In the traditional file system abstraction, a single volume or device is assigned to a single file system. Once this assignment has taken place, it is requires a significant investment of time and administrative resources to increase the amount of storage available to a file system or make changes to the file system associated with a given volume. Several different kinds of answers have been provided to this problem.

On Linux, the Linux Volume Manager (LVM) attempts to solve this problem in a different way. Logical volume management looks at volumes in software terms rather than hardware terms. This allows volumes to be re-sized and moved. LVM also provides a snapshot capability. None of the features of LVM are automatic and many are heavy on administrative details. LVM requires many steps to make changes. However, LVM is far ahead of using physical volumes and regular storage systems.

The Enterprise Volume Management System (EVMS) is another volume manager for Linux. While LVM expects to be dealing with raw devices, EVMS recognizes a variety of disk partition formats as well as software RAID configurations and LVM-created logical volumes. EVMS attempts to unify Linux tools such as md - the software RAID manager - LVM, and others under one umbrella. EVMS provides a snapshotting capability as well as some attempt at recovery of data through bad block relocation. Bad block relocation “monitors its I/O path and detects write failures (which can be caused by a damaged disk). In the event of such a failure, the data from

that request is stored in a new location" ("EVMS User Guide"). EVMS can create LVM volumes as well as EVMS volumes, and therefore provides a nice tool for creating volumes on which Linux-VServer or OpenVZ can reside. Both EVMS and LVM support a variety of different file systems with a multitude of advanced features.

Most recently, Solaris provided an entirely new file system as an answer to these kinds of planning requirements. The Zettabyte File System (ZFS) usage a storage model which completely eliminates the idea of volumes. Instead, ZFS provides a storage pool from which many file systems can draw. The file systems overlaying this pooled storage, known as a zpool, grow and shrink automatically – within the constraints of set quotas and hardware limitations. ZFS utilizes copy on write (CoW) behavior to ensure data consistency. Because ZFS makes a copy of data when doing a write, creating a snapshot of a file system is more efficient on ZFS than not creating one. ZFS is built with consistency in mind. ZFS zpools are usually created with some form of redundancy. While a single zpool can be created with a single device, it is most often created using a mirror or a RAID. ZFS utilizes a redundant array known as RAID-Z. RAID-Z provides the parity of a higher-order RAID system like RAID 6 or RAID 6, but with a variable stripe width.

File Systems

A ZFS file system is mounted on top of a zpool and utilizes all the features of the next-generation model provided by pooled storage. ZFS also provides CoW behavior to ensure data consistency. ZFS is a 128 bit file system which allows for growth beyond foreseeable needs and even known hardware capacity ("128-bit storage: are you high?"). On top of all of these

features, when provided with a mirror or RAID, ZFS is a self-healing file system which will detect data corruption which often occurs silently and heal the data from the backup copy.

Several different advanced file systems for the Linux platform can be mounted on top of a logical volume created by LVM or EVMS. The default file system for many Linux distributions is ext3, an update of ext2. Ext3 is fully compatible with the ext2 file system, but adds journaling. Because of the compatibility, ext3 has a full featured and mature tool set. Ext3 journals not only file system meta data, but also file contents. While ZFS provides consistent files by using copy on write behavior and ext3 by journaling, none of the other file systems discussed in this paper provide the same level of file consistency.

XFS is an open source file system available from SGI for Linux. Like ZFS, XFS was designed with scalability in mind. XFS was created in the early 1990s and was designed even at that time to handle terabytes of data and exobyte-sized files. XFS is a traditional file system in that it utilizes the concepts of partitioning and volumes. However, XFS uses balanced B trees for many of its storage data structures in order to provide speed gains. XFS also provides a write-ahead log to help ensure data consistency on disk. The log provided by XFS journals only file system meta data, not file contents.

The journaled file system (JFS) is a very similar file system to XFS created by IBM. JFS was initially designed in the early nineties like XFS, but went through a serious re-design in the late 1990s. JFS incorporates many of the features provided by XFS, including a similar data structure and journaling. In a variety of benchmarks, JFS and XFS often trade places on who is the fastest. JFS is known for low CPU utilization in benchmarks.

One of the more controversial file systems in use on Linux system is ReiserFS. ReiserFS

was created by Namesys, a company which has since fallen on hard times. Due to this, continued work on ReiserFS has ground to a halt. Reactions to ReiserFS are mixed – for some users it is both stable and fast, while others have experienced stability issues with the file system. The OpenVZ user's guide specifically recommends against the use of ReiserFS as the file system for the partition used for virtualized environments, stating that " it is proved to be less stable than the ext3, and stability is of paramount importance for OpenVZ-based computers" (25).

Table 2 in the appendix shows a comparison of the features of the file systems described. Because ZFS does not exist natively on Linux, it is impossible to provide a fair benchmark comparison between all the file systems. Therefore, a comparison of the performance of the various file systems is not provided.

ZFS and Containers on OpenSolaris

ZFS and Containers are closely integrated in OpenSolaris. A ZFS file system can serve as the root file system for a Container. ZFS quota tools can be used to provide the disk limits for a containers' file systems. Perhaps the most exciting feature of ZFS inside Containers is that the Container administrator can configure quotas and create ZFS file systems inside the Container without being the operating system root user. Much of the configuration information below is based off of the Sun How-To document "Solaris Operating System – Managing ZFS in Solaris 10 Containers."

As discussed earlier, a ZFS file system is mounted on top of a zpool. A zpool is created by giving it access to some sort of storage object, from an entire redundant array of inexpensive disks (RAID) to a single file. Most zpools are created in such a way as to provide some sort of

redundancy. An example command to create a zpool is `zpool create zonepool mirror disk_1 disk_2`, where the disk names are provided by the operating system. The pool can either be shared with the host operating system, known as the global zone in Solaris terms, or separate pools can be created. However, with ZFS there is not a need to create multiple pools in the same way there is to provide partitions under traditional storage management.

The command `zpool list` can be used to check that the zpool has been created. In order to then create a ZFS file system for use with a container, use the command `zfs create zonepool/zonefs`. ZFS provides its own quota tools as part of the `zfs` command. In order to set a quota using ZFS on the zonefs file system in the zonepool pool, use `zfs set quota=SIZE zonepool/zonefs`. This command will set the maximum size of the zonefs file system to the specified size. To see this, the `zfs list` command can be used to show existing ZFS file systems (it shows the zpool as well) and their properties.

Once a file system is created that can be used for a container, the next step is to create the container itself. The container is created by making a zone. This zone must live in a directory. Once an appropriate directory is selected, the `zonecfg` command is used to create the zone. In order to specify the zone name, use the -z option. The command looks something like the following, where # represents the command prompt.

```
# zonecfg -z cs671
cs671: No such zone configured
      Use 'create' to begin configuring a new zone
      zonecfs:cs671< create
      zonecfg:cs671< set zonepath=/zones/cs671
      zonecfg:cs671< add dataset
      zonecfg:cs671:dataset< set name=zonepool/zonefs
      zonecfg:cs671:dataset< end
      zonecfg:cs671< commit
      zonecfg:cs671: exit
```

The `zonecfg` command above is followed by a series of subcommands appropriate to

the creation of a very simple zone. Further resource configuration of a zone is done by using a resource pool created with the `pooladm` and `poolcfg` commands. This kind of configuration is the resource management function which takes a zone from a simple zone to a container.

The interesting part of the creation of this zone is the addition of the dataset. The dataset references the ZFS file system created above. The dataset resource type is specific to ZFS. Other file systems use the fs resource type. When the zone created above is booted, the ZFS file system will be available for use. If the dataset had not been added to the Container, users and administrators within the container would have no idea of its existence.

To take the zone from the created state to the installed state, use the `zoneadm` command. Provide the options `zoneadm -z cs671 install`. This will install the zone so that it is ready for use, but not boot it. Booting is done via `zoneadm -z cs671 boot`. In order to connect to a zone, use the `zlogin` command. Using `zlogin -C cs671` logs into the zone's console. On first login, several system questions are presented, including terminal selection and networking issues (if networking is configured). The root password for the zone is also provided on first log in. Running a `zfs list` inside of the zone will show the zonepool and zonepool/zonefs pool and file system. The zonefs file system is constrained as configured. The available size for the zonefs pool is the size (shown as SIZE above in the `zfs set` command) as specified.

Once logged in to the zone, a zone administrator has the ability to create additional file systems. If the container provides an isolated environment for a web server, the zone may contain apache files. Apache integrates well with containers. Apache requires the following directories to be in place: /usr/apache2, /etc/apache2, and /var/apache2 (“Zones for ASF Projects”). The zone administrator can create a ZFS file system to contain these files. The file

system is created on top of the existing zonefs system using the same commands as before: `zfs create zonepool/zonefs/apache_binaries`. The zone administrator can also provide a quota for the apache binaries folder in order to prevent it from using all of the space allocated to zonefs. This is done using the `zfs set` command as shown above. The size for the apache_binaries file system cannot be larger than the size for the zonefs file system.

The mount point for the apache_binaries will not be correct at this point. As described above, the files on this system should be located at /usr/apache2. The zone administrator can change the mount point for the file system he or she has created using `zfs set mountpoint=/usr/apache2 zonepool/zonefs/apache_binaries`. All of this configuration can be done within the zone by an administrator assigned to that zone. The zone administrator may wish to have back-ups of these files. If the /var/apache2 file system was zonepool/zonefs/apache_var in the ZFS hierarchy, a snapshot is created with the command `zfs snapshot zonepool/zonefs/apache_var@1st`. This creates a snapshot entitled 1st. This snapshot is available under .zfs/snapshot/1st.

A web hosting business could use ZFS and Containers as described above to host many web sites on the same box. Because ZFS can grow to almost unimaginable sizes, the storage will continue to scale. If the web hosting business grows beyond the initial capacity of the storage assigned to it, the pooled storage makes adding additional devices to the pool simple. The example pool created above is a mirrored pool. To add storage to it, the additional devices must also be mirrored. Once the devices are available to the global zone, where zonepool is configured, one simple command adds these devices to zonepool: `zpool add zonepool mirror disk_3 disk_4`. In this way, the administrator of the global zone for the web hosting

company can continue to add storage to the pool used by the containers which host web servers. Each web server's ZFS file system will use its allocated amount of the pool, if a quota is specified. No additional storage management is required. The flexibility provided by ZFS for container management makes it an attractive option for managing storage abstractions within COS virtualization environments.

Linux-VServer, LVM, and ext3

In many benchmarks, XFS, JFS, and ReiserFS trade places as the best performers. However, ext3 is one of the most commonly used file systems on Linux. For the purposes of illustrating the management of file systems with Linux-VServer, the example will utilize the ext3 file system for Linux.

Linux-VServer initially provided a rather crude way of separating the view of the file system provided to each COS environment, known as a virtual private server (VPS). To begin with, VServer used chroot() to set the root for each VPS. The chroot() command changes the location in the file system perceived as root. Unfortunately, chroot() is not very secure and the "root" set by that command can be escaped. To work around this, VServer utilized a chroot barrier which could not be escaped by traditional means. In later versions of VServer, a more elegant implementation of securing file systems views per VPS has emerged. This change used private namespaces per VPS.

Linux-VServer is somewhat integrated with a variety of techniques for limiting disk usage by a VPS. Vserver uses standard Linux tools provide both disk limits and quotas. Disk limits constrain a VPS to a certain amount of inodes as well as a certain amount of disk space

total. Space can also be reserved for the roor user. If a partition is shared between many different guests, context disk limits may be used. Context disk limits, which are used for VPS-es on a shared partition, can only be used with context ID (XID) hashes attached to the files. This is supported in full by only the ext2 and ext3 file systems for Linux, but is supported on some level by all file systems supported by Linux-VServer. With context disk limits

> "the number of inodes and blocks for each filesystem is accounted, if an XID-Hash was added for the Context-Filesystem combo. Those values, including current usage, maximum and reserved space, will be shown for filesystem queries, creating the illusion that the shared filesystem has a different usage and size, for each context" ("Paper – Linux-VServer").

Quotas require virtual root support. Virtual root support allows VPS-es to access virtual block devices instead of real physical block devices as root. Without virtual root support, quotas fail. A tool exists to provide a virtual block device as the mount point for a partition. Once the virtual device is configured, the VPS also needs some configuration to provide quota support to it. From there, the VPS needs access to the virtual block device, and then is ready to go. To use quotas inside of a VPS, the VPS must be located on its own partition. The partition itself acts as a disk limit in this case.

One way of managing VPS storage is to use one LVM partition per VPS. In order to do this, the `fdisk` command must be used to create a new partition for use with LVM. Alternatively, EVMS can be used to create an LVM compatible volume. From there, several steps allow an administrator to place a VPS on the logical volume. Much of this information is derived from the web page "Howtos Linux-VServer with LVM and Quotas." In order to create a

VPS on an LVM volume group, the following series of commands can be used:

```
pvcreate /dev/volume_name
vgcreate group_name /dev/volume_name
lvcreate -L<size> -n volume_name group_name
```

The first command creates a volume for use. The second command creates a volume group. A volume group can span more than one physical device, but only one is shown here. A volume group is analogous on some level to a ZFS zpool. The third command creates a logical volume which a limit of the specified size and the specified name in the group. This logical volume is where the file system for the VPS will be placed. The next step is to create the file system itself. The following commands create an ext3 file system on the just-created logical volume, then mounts the file system at the location which Linux-VServer expects root directories.

```
mkfs -t ext3 /dev/group_name/volume_name
mount /dev/group_name/volume_name /var/lib/vservers/vps_name
```

After running the above commands to create an appropriate root file server for a VPS, the `vserver` command with the build option can be used to actually create the VPS. This command has a wide variety of options. The basic configuration requires a name for the VPS, a hostname, a network interface, and the distribution of Linux being used for the VPS. Once the VPS is created, a small amount of further work needs to be done to provide the VPS access to the file system on the logical volume. In order to do this work, the VPS must be stopped and the partition must be unmounted. First, a vroot block device must be created for the VPS. This can be done using the `mknod` utility. Because the partition is the vroot for a VPS, there are several commands which must be run every time the partition is mounted.

```
mount -o usrquota,grpquota /dev/group_name/volume_name
      /var/lib/vservers/vps_name
rm -f /var/lib/vservers/vserver1/dev/hdv1
vrsetup /dev/vroot/vps_name /dev/group_name/volume_name
```

```
cp -fa /dev/vroot/vps_name /var/lib/vservers/vps_name/dev/hdv1
```

This command will mount the file system with quotas enabled on the block device. This command will remove the device which the verser is expecting to be its root. This command will attach the virtual root device to the logical volume created with LVM. This command will copy the virtual root into the location expected by the VPS.

Once an LVM partition is set up, standard Linux quota utilities can be used to provide quotas for the VPS. In order to do this, a capability flag must be set on the VPS. This is done by adding the quota_ctl flag to /etc/vservers/vps_name/ccapabilites. After the quota capability has been added to the VPS, quotas can be set within the VPS. Note that mounting the file system with the usrquota (user quota) and grpquota (group quota) options as shown above is critical for this operation. Quotas on Linux using these tools allow the utilization to be specified for users and groups. There is a significant amount of additional administrative work compared to ZFS on Containers, but the end result is that disk utilization can be capped within a COS virtualization environment under both platforms.

The same scenario of a group of VPS-es needing to grow past current storage capacities can easily occur on this setup. Based on the restriction that quotas on Linux-VServer work best with one VPS per volume, adding more space would most likely require creating additional logical volumes on new hardware. Because of the logical volume set-up, careful planning must be done in advance in order to make sure that the size of each logical volume is appropriate. Logical volumes can be grown or shrunk and physical volumes can be removed from logical volumes if the initial calculations are incorrect.

OpenVZ Resource Controls

OpenVZ is an open source COS virtualization environment for Linux that is somewhat similar to Linux-VServer. However, there are many philosophy differences between the two. Where Linux-VServer exists as a lean patch to the vanilla Linux kernel and co-exists easily with other kernel patches, OpenVZ makes enough modifications to the kernel that it is difficult to add anything to a standard OpenVZ configuration. OpenVZ focuses on ease of administration. Different Linux distributions can be installed on a single OpenVZ hardware node (HN), which is the computer that VPS-es exist on in OpenVZ terms. These different distributions are installed via pre-configured templates. OpenVZ also builds resource controls in up front, requiring administrators to do careful capacity planning in advance, and therefore preventing rouge VPS-es from taking down the system unexpectedly at a later date.

OpenVZ has specific requirements for the host operating system. Because of security concerns, a default OpenVZ host operating system set-up is minimal. The main requirement for the host operating system is that it have a /vz partition which is as large as possible. The /vz partition is where VPS-es go. The /vz partition can be an LVM partition for ease of expansion. No comment is made on a choice of ext3, XFS, or JFS in the OpenVZ User's Guide. Along with a supported Linux distribution as the host operating system, OpenVZ requires an OpenVZ kernel and several user tools to be downloaded after the host operating system has been installed.

Creation of a VPS is done by installing an OS template. There are two different ways to install a template. The first way is to download a template metadata file via `rpm` and use the `vzpkgcache` utility provided by OpenVZ to create a cache of the metadata for that template. Template metadata includes the packages which make up a template, a location to download those packages, scripts needed to run during installation, OpenVZ specific information, and other

files. An even easier way to create a VPS on OpenVZ is to download a precreated OS template cache tarball. There are tarballs for various versions of Fedora, Debian, Gentoo, Mandriva, and Ubuntu among others. Template caches are installed into the /vz/templates/cache directory in the host operating system.

Once an administrator has installed the appropriate template cache, the next step is to create the VPS itself. This is done using the `vzctl create` command. This command takes an ID for the VPS and an OS template. The ID should be greater than 100 due to OpenVZ concerns and unique on any given HN. The `vzlist` command can be used to see the IDs in use on a given HN. The `vzpkgls` command can be used to list all the OS templates on a given OpenVZ host operating system. The `vzctl` command takes an optional configuration flag which will specify a file with several parameters for the VPS. This enables an administrator to put all configuration for identical VPS-es in one place. A VPS on OpenVZ must merely be created and can be started once creation is done.

```
vzctl create 101 –ostemplate TEMPLATE –config file.name
vzctl start 101
```

Resource configuration is carefully built into OpenVZ. The highest level of quotas are per-VPS disk quotas. These are known as first-level disk quotas for OpenVZ. Using Containers and ZFS these quotas can be achieved by setting the quota of the ZFS file system. Using Linux-VServer the recommended way to set something analogous to an OpenVZ first-level quota is putting the VPS on a separate partition or using an XID-hash on the file system.

By default, first level quotas are on in OpenVZ. This can be changed in the global OpenVZ configuration file. First level disk quotas can be turned on and off per VPS in the VPS configuration file. The parameters which can be set for a disk quota for a given VPS are the

amount of disk space available to that VPS, the number of inodes available for the VPS, and the grace period during which a VPS may be over its limits in seconds. An example to set the VPS 101 to a disk quota of 5 gigabytes and 450,000 inodes is shown below. The grace period for VPS 101 will be 10 minutes. The –save options persists the configuration changes.

```
vzctl set 101 –diskspace 5000000:5100000
vzctl set 101 –diskinodes 450000:451000
vzctl set 101 –quotatime 600 –save
```

OpenVZ utilizes the same standard Linux quota tools that Linux-VServer uses to quota users and groups. In OpenVZ these quotas are known as second-level quotas. A parameter in the VPS configuration file sets the number of user and group IDs which can have quotas assigned to them. This control parameter is a first-level quota parameter. Assigning that value to zero turns off second level quotas for the VPS. Assigning it to a value greater than zero allows up to that many user or group IDs to be assigned a quota. The OpenVZ User Guide says "this value must be greater than or equal to the number of entries in the VPS /etc/passwd and /etc/group files. Taking into account that a newly created Red Hat Linux-based VPS has about 80 entries in total, the typical value would be 100" (51).

The OpenVZ tools are more obvious than those of Linux-VServer. The ease of use is on par with Solaris Containers and ZFS. In a situation where an OpenVZ standard configuration makes sense, it is a viable choice.

In the example of the web server needing to grow the available storage for VPS-es, the case is easy if the /vz volume is an LVM volume. Standard LVM commands extend the volume. If the volume group supporting the logical volume is named vz_group, the new device is located at /dev/disk_1, and the logical volume is located as /dev/vz_group/vz, the following commands would add disk_1 to the volume group and extend the logical volume to utilize its size.

```
Vgextend vz_group /dev/disk_1
lvextend -L+DISK_SIZE /dev/vz_group/vz
```

LVM does not know how to automatically extend one of the many file systems which may be mounted on it. However, all of the advanced file systems for Linux described above can grow in size using a variety of different commands. Once the logical volume is extended, the file system on top of it must also be extended. Each advanced Linux file system described in this paper can grow in size. For example, XFS grows using `xfs_growfs /vz`. Note that some Linux kernel versions have problems with the syntax used to resize a JFS file system. This may be a drawback when selecting a file system for a /vz partition.

Conclusions

When it comes to ease of storage administration with COS virtualization environments, the integration between Solaris Containers and ZFS is untouched by the comparable open source offerings on Linux. Linux-VServer has convoluted and disorganized documentation. Finding out how to do anything with storage management and then doing it is very difficult and time consuming. OpenVZ provides a similar ease of administration, but at the cost of tying the administrator into the OpenVZ mindset. Also, the file systems on which VPS-es for an OpenVZ environment can be mounted are neither as powerful or as flexible as ZFS. While ext3, ResierFS, XFS, and JFS can all grow to meet increased storage needs and LVM provides flexibility in partition management, it takes several commands to state the semantic intent that the zpool and zfs commands can express in one. Many concerns obviously relate to the selection of a COS virtualization environment, but where file systems and storage are a major issue, ZFS's tight integration with Containers makes the Solaris 10 or OpenSolaris environment a clear

winner.

Appendices

Table 1: Container-based Operating System Virtualization Comparisons

Data for this table came from the "OpenVZ User's Guide" for OpenVZ. For Linux-VServer the data was provided by the Linux-VServer white paper at "Paper – Linux VServer." Data on Solaris Containers was provided by "Solaris Operating System – Managing ZFS in Solaris 10 Containers" and "Solaris Operating System – Moving a Solaris Containers How To Guide."

	Solaris Containers	Linux-VServer	OpenVZ
Disk Quotas	Supported by ZFS or the Solaris Volume Manager (SVM)	Supported by partitions or standard Linux quota tools	Supported by OpenVZ and standard Linux quota tools
Memory Quotas	Yes	Hard only	Yes
CPU Quotas	Yes	Hard only	Yes
Network Quotas	In beta	No	Yes

Table 1: Container-based Operating System Virtualization Comparisons

Table 2: File System Comparisons

The data for ZFS came from "ZFS: The Last World in File Systems" and "ZFS FAQ at OpenSolaris.org." The data for ext3 was provided by "". The data for XFS was provided by "SGI Developer Central Open Source | XFS" and "Open Source XFS for Linux." The data for JFS sizes came from an IBM "JSF Overview." Information for the rest of the JFS data in the table came from a variety of comments on Linux mailing lists. The demise of Namesys has made finding original sources of data on ReiserFS difficult. However, Google cache provides a cached version of the Namesys FAQ on ReiserFS, which was consulted to provide information on that file system. Ext3 data was provided by "Design and Implementation of the Second Extended

Filesystem" and "GNU ext2resize – an ext2 filesystem resizer." This data is current for ext3 because ext3 is an extension of ext2 to provide journaling only.

	ZFS	ext3	XFS	JFS	ReiserFS
Supported Operating Systems	OpenSolaris, Solaris 10, OS 10.5, Linux (via FUSE)	Linux, BSD, Windows (via an IRS)	Linux, IRIX	Linux, AIX, OS/2	Linux
Maximum File System Size	16 EiB	4 TB	18 million terabytes	4 PiB	2^{32} – 3 files
Maximum File Size	16 EiB	2 GB	9 million terabytes	Based on the virtual file system	2^{60} bytes
Meta Data Consistency	Provided by copy-on-write	Provided by a journal	Provided by a journal	Provided by a journal	Provided by a journal
File Content Consistency	Provided by copy-on-write	Provided by a journal	No	No	Available as a patch
Grow File System	Automatic	Via ext2resize	Online only using xfs_growfs	Online only	Online or offline
Shrink File System	Automatic	Via ext2resize	No	No	Offline only, does not resize underlying partition
Quota Support	Provided by the zfs command.	Using standard Linux quota tools.	Use xfs_quota or specify quota options when mounting	Yes	Available as a patch for the Linux kernel

Table 2: File System Comparisons

Works Cited

Best, Steve. “JSF Overview.” Online. Internet. http://www-128.ibm.com/developerworks/library/l-jfs.html. 1 January 2000. IBM. 21 April 2008.

Bonwick, Jeff. “128-bit storage: are you high?” Online. Internet. http://blogs.sun.com/bonwick/entry/128_bit_storage_are_you. 25 September 2004. Jeff Bonwick. 21 April 2008.

Bonwick, Jeff and Moore, Bill. “ZFS: the Last World in File Systems.” Online. Internet. http://www.opensolaris.org/os/community/zfs/docs/zfs_last.pdf. Sun Microsystems, Inc. 21 April 2008.

Card, Remy, et. al. “Design and Implementation of the Second Extended Filesystem.” Online. Internet. http://e2fsprogs.sourceforge.net/ext2intro.html. Ext2fs. 28 April 2008.

Dierkes, William Joseph. “Howtos Linux-VServer with LVM And Quotas.” Online. Internet. http://www.5dollarwhitebox.org/wiki/index.php/Howtos_Linux-Vserver_With_LVM_And_Quotas. 12 January 2008. William Joseph Dierkes. 14 April 2008.

Dilger, Andreas and Buytenhek, Lennert. “GNU ext2resize – an ext2 fielsystem resizer.” Online. Internet. http://ext2resize.sourceforge.net/. Andreas Dilger and Lennert Buytenhek. 28 April 2008.

Lewis, AJ. “LVM HOWTO.” Online. Internet. http://www.tldp.org/HOWTO/LVM-HOWTO/. 27 November 2006. Rackable Systems, Inc. 24 April 2008.

Linux-VServer. “Paper – Linux-VServer.” Online. Internet. http://linux-vserver.org/Paper. 1 January 2008. Linux-VServer. 8 April 2008.

Linux-VServer. “XFS – Linux-VServer.” Online. Internet. http://linux-vserver.org/XFS. 26 October 2007. Linux-VServer. 16 April 2008.

Lorenze, Christine, et. al. “EVMS User Guide.” Online. Internet. http://evms.sourceforge.net/user_guide/. 18 January 2005. IBM. 14 April 2008.

Namesys. “FAQ.” Online. Internet. http://64.233.169.104/search?q=cache:r-vvdhBMOZAJ:www.namesys.com/faq.html 15 October 2003. Namesys. 21 April 2008.

Silicon Graphics, Inc. “Open Source XFS for Linux.” Online. Internet. http://oss.sgi.com/projects/xfs/datasheet.pdf. July 2006. Silicon Graphics, Inc. 21 April 2008.

Silicon Graphics, Inc. “SGI Developer Central Open Source | XFS.” Online. Internet. http://oss.sgi.com/projects/xfs/faq.html. Silicon Graphics, Inc. 21 April 2008.

Soltesz, Stephen, et. al. "Container-based Operating System Virtualization: A Scalable, High-performance Alterative to Hypervisors." Online. Internet. http://www.cs.princeton.edu/~mef/research/vserver/paper.pdf. 19 January 2007. Marc E. Fiuczynski. 8 April 2008.

Sun Microsystems, Inc. “Solaris Operating System – Managing ZFS in Solaris 10 Containers.” Online. Internet. http://www.sun.com/software/solaris/howtoguides/zfshowto.jsp. Sun Microsystems, Inc. 23 April 2008.

Sun Microsystems, Inc. “Solaris Operating System – Moving a Solaris Containers How To Guide.” Online. Internet. http://www.sun.com/software/solaris/howtoguides/moving_containers.jsp. Sun Microsystems, Inc. 28 April 2008.

Sun Microsystems, Inc. “ZFS FAQ at OpenSolaris.org.” Online. Internet. http://www.opensolaris.org/os/community/zfs/faq/. 6 March 2008. Sun Microsystems, Inc. 21 April 2008.

SWSoft, Inc. “OpenVZ User's Guide.” Online. Internet. http://download.openvz.org/doc/OpenVZ-Users-Guide.pdf. 2005. SWSoft, Inc. 24 April 2008.

The Apache Software Foundation. “Zones for ASF projects.” Online. Internet. http://www.apache.org/dev/solaris-zones.html#document. 2008. The Apache Software Foundation. 23 April 2008.