Princeton Plasma Physics Laboratory

PPPL- 4454

PPPL- 4454

Utilizing ZFS for the Storage of Acquired Data

C. Pugh, P. Henderson, K. Silber, T. Carroll, K. Ying

September, 2009

Prepared for the U.S. Department of Energy under Contract DE-AC02-09CH11466.

Princeton Plasma Physics Laboratory
Report Disclaimers

Full Legal Disclaimer

Trademark Disclaimer

PPPL Report Availability

Princeton Plasma Physics Laboratory:

http://www.pppl.gov/techreports.cfm

Office of Scientific and Technical Information (OSTI):

http://www.osti.gov/bridge

Related Links:

U.S. Department of Energy

Office of Scientific and Technical Information

Fusion Links

UTILIZING ZFS FOR THE STORAGE OF ACQUIRED DATA*

C. Pugh, P. Henderson, K. Silber, T. Carroll, K. Ying
Information Technology Division
Princeton Plasma Physics Laboratory (PPPL)
Princeton, NJ
cpugh@pppl.gov

Abstract— **Every day, the amount of data that is acquired from plasma experiments grows dramatically. It has become difficult for systems administrators to keep up with the growing demand for hard drive storage space. In the past, project storage has been supplied using UNIX filesystem (ufs) partitions. In order to increase the size of the disks using this system, users were required to discontinue use of the disk, so the existing data could be transferred to a disk of larger capacity or begin use of a completely new and separate disk, thus creating a segmentation of data storage.**

With the application of ZFS pools, the data capacity woes are over. ZFS provides simple administration that eliminates the need to unmount to resize, or transfer data to a larger disk. With a storage limit of 16 Exabytes (10^{18}), ZFS provides immense scalability. Utilizing ZFS as the new project disk file system, users and administrators can eliminate time wasted waiting for data to transfer from one hard drive to another, and also enables more efficient use of disk space, as system administrators need only allocate what is presently required.

This paper will discuss the application and benefits of using ZFS as an alternative to traditional data access and storage in the fusion environment.

***Work Supported by U.S. DOE Contract No. DE-AC02-CH0911466**

Keywords-ZFS, data acquisition, storage, UNIX,

I. INTRODUCTION

In the beginning, system administrators created partitions and volumes. They saw the data storage capabilities and thought, this is good enough and they settled. Then came the revolutionary new file system called, ZFS and the system administrator saw the ease in which either large or small amounts of data could be efficiently stored, with minimal administration costs, maximal disk utilization, and the peace of mind for data integrity. Initially named the "Zettabyte File system," but now an orphaned acronym, ZFS is a relatively new file system that was designed by a team from Sun Microsystems [1]. This team broke away from what was commonly expected of file systems, and created a better system, from the ground up. Their new design features a seemingly limitless file system, which can be administered on the fly. With the ability to add space to a pool with one simple command, no longer does a system administrator have to guess how large a project will grow. No longer does a project have to be interrupted in order to grow a partition, or move data to a larger partition.

II. FINDING A SOLUTION

The administrators at Princeton Plasma Physics Laboratory (PPPL) were struggling with meeting the demands of ever growing acquired project data. Under their current system of exporting a UNIX file system (ufs) partition via Network File System protocol (NFS), any project at the lab could request a partition for the storage of acquired project data which could be automatically mounted on our internal "portal" and cluster nodes. Included in their request would be an estimate of the amount of data they planned to use. Quite often, many projects that were estimated to be "small" ended up growing exponentially in size. Even if a project on a shared partition exceeded their quota, under the previous system, they still had the ability to write to their disk if it had space left. This was because the quotas that were implemented were "soft quotas," meaning only a written warning was issued. Soft quotas were the only option, as hard quotas could only be implemented on a complete file system and applied to groups or users. Since theoretically, there was more space left on the partition, the project could consume the rest of the disk, causing other projects on the same partition to also run out of disk space. Projects would then have to either be moved to another larger partition (up to 500GB), or begin writing to a new partition. This created a segmented array of project disks. Some projects would have four or more project areas (e.g. gyro.1, gyro.2, gyro.3) and would have to set up links to keep their data organized. The administrators did not find it acceptable to serve out larger partitions, for fear of wasting valuable disk real estate, and concern for the size of backups.

In order to cope with this endless battle, the systems administrators began researching new solutions. Among the considerations for a solution were changing to a Logical Volume Manager (LVM), utilizing the new ZFS, or continuing to use the ufs while in inefficiently dishing out increasingly larger amounts of disk space. When the system administrators saw that even using LVM required unmounting the project disk in order to resize, it became clear this was not an acceptable solution.

III. About ZFS

According to their research, the systems administrators found that the ZFS way of life would be the most ideal route. As the very first 128-bit filesystem, ZFS could provide seemingly limitless scalability. Some theoretical limits can help to put this into perspective. You could have 2^{48} files in any individual file system, 16 exabyte (10^{18}) file systems, 16 exabyte files, $3x10^{23}$ petabyte storage pools, 2^{48} files in a directory, 2^{64} devices in a storage pool, 2^{64} storage pools per system, 2^{64} file systems per storage pool. [2]

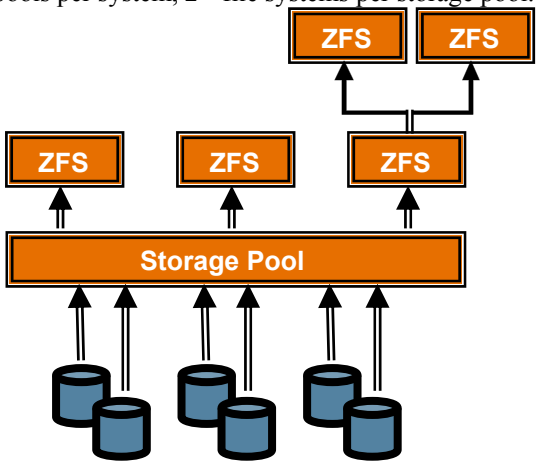

Figure 1. Simplistic diagram of ZFS pooled storage

With a completely unique set of simple commands, any administrator could quickly learn how to effectively administer the new data set. As illustrated in figure 1, ZFS utilizes pooled storage, where a disk or series of disks are added to a "pool" where the size of the pool is the sum of the sizes of the disks added. Since each nested file system can access the pooled storage, you can work creatively to ensure that disk real estate is efficiently allocated. The pools can be created with a few different options. Shown in examples below, the whole disk is used, with no software data redundancy. Since the SAN device PPPL uses had hardware RAID-5, the administrators feel RAID-5 is sufficient to ensure the data integrity. Other options to implement with ZFS are traditional mirroring, and the new RAID-Z. RAID-Z is similar to RAID-5 however, it avoids the RAID-5 "write hole" by using a "copy-on-write" policy.

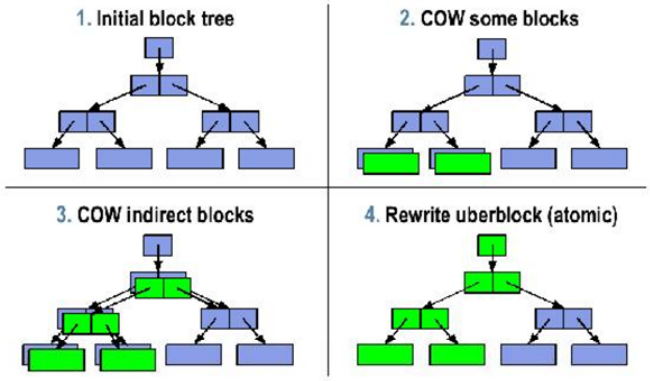

Figure 2. Copy-On-Write Transactions[5]

As seen in figure 2, this means that instead of overwriting old data with the new, a new location is written to, and then is the pointer to the old data is over written with a pointer to the new data. In this manner, only full-strip writes are performed, and therefore you either get all or nothing, for example, in the case of a power outage. [5] Mirroring or RAID-Z configurations also allow for unprecedented self-healing. ZFS has a checksum attached to every file, and gradually "scrubs" the data by traversing the metadata tree to read a copy of every block as shown in figure 3. If the scrub finds an error, it repairs the file with the clean redundant copy. By checksumming end to end in this fashion, ZFS detects "silent" data corruption such as bit rot, phantom writes, misdirected read or write, parity or driver errors. Bit rot can be detected on any file system, as this is the gradual decay of storage media, and is commonly checked for with SMART enabled disks. Phantom writes are where the write is dropped. Misdirected reads or writes happens when the disk accesses the wrong block. DMA parity errors between the array and server memory or from the driver are avoided since the checksum validates the data inside the array. Driver errors happen when the data winds up in wrong buffer inside the kernel. An accidental overwrite could occur when swapping to a live file system. In the case of non parity configuration, if an error is found, it will require administrator intervention, such as restoring from back up.

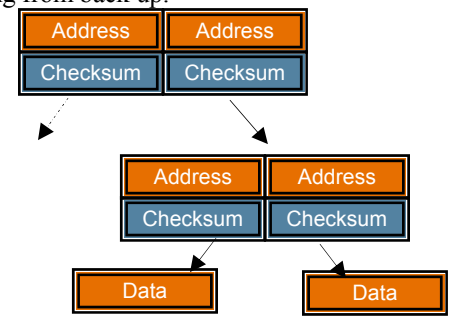

Figure 3. ZFS self validating metadata tree

Another useful feature of ZFS is its ability to transparently compress a file system. Fast cameras that collect data can use around 1GB per shot. When you multiply this by three cameras and thousands of shots over the lifetime of a project, the numbers can sure add up! Transferring, accessing or storing files this large can pose a tiresome challenge. With ZFS compression, the file size can be reduced by 2-3 times, depending on the algorithm used. The available compression algorithms are lzjb, gzip, gzip-N. As with most of the features of ZFS, compression can be toggled dynamically, on a per-file system basis. If you simply need to archive data for future use, compression is certainly the way to go. In fact, by using compression thoughtfully, one can improve a common bottleneck, the disk I/O. [3] Of course, with compression you sacrifice some performance, however with the server that was chosen for implementation, cpu performance is not an issue. If you are intending on running a cpu intensive program, such as camera diagnostics however, compression may not be the best idea, performance wise. Experiment before you deploy.

IV. Implementation

In order to implement the ZFS solution the team of administrators decided to purchase new Sun Servers with Solaris 10, which included the ZFS software. Three Sun Microsystems Sun Fire X4240's were purchased, with an optional Qlogic Corp qla2432 fiber card. The X4240 offers high performance AMD Quad Core cpus and high memory capabilities. [4] These servers were then connected via fiber to the existing Storage Area Network (SAN) device. A series of commands were used to verify the connectivity to the SAN. The first command shows the information about the card attached to the server.

```
# fcinfo  hba-port
```

This information can be used to properly set up the zones on the SAN device. Once the zones were set up properly, the server was checked to verify connectivity to the SAN by issuing the command:

```
# luxadm –e port
```

This verified that the SAN and the server had communication. Once this was established, the Solaris Multipathing software was enabled by issuing the command:

```
# stmsboot –e
```

When all steps in setting up the server and SAN were complete, disks could be presented to the new server from the SAN. Traditionally, the administrators would use shared 500GB partitions for project disks. However, it was decided that with the abundance of inexpensive disk space available today, 1 to 2 TB disks would be added to the pools.

V. Example Use of ZFS

Once you have your environment set up, the creation of ZFS pools, or zpools is simple. One can think of a zpool as a virtual disk, as the size of the pool is the sum of the sizes of the disks attached. The command to both create a new pool, "exampool" and add disks, "c1t2d0" and "c1t3d0" to the pool is as follows:

```
# zpool create exampool c1t2d0 c1t3d0
```

This command automatically creates the mount point /exampool and mounts the pool there. The mount point can be changed if that is what you require. Once the pool is created, a file system needs to be established.

```
#zfs create exampool/fusion
```

Now, /exampool/fusion is not only a new directory, but it is its own file system that can use whatever disk space is available in the pool it which it is contained. Since each file system can contain other "nested" file systems, it is ideal to give each user, group or project their own file system. The child file systems will inherit options, such as quotas, compression, reservations or ACL's from the parent filesystem. For example, a group could have the need for a new project disk for archiving data, and running camera diagnostics:

```
#zfs create exampool/fusion/archive
#zfs create exampool/fusion/CameraDiags
```

Since /exampool/fusion/archive will be used for archiving data, it is a good idea to turn on file compression. However, compression will not be implemented on the CameraDiag file system, since this requires high performance. Compression can be dynamically set, like every other option. However, realize that if you have data on the disk before you enable compression, the data will not automatically become compressed. The data would have to be moved off, then back on in order to become compressed.

```
#zfs set compression=gzip exampool/fusion/archive
```

Each nested file system belonging to "exampool" can by default, enjoy all the space that is contained in the pool. If however, you would like to ensure that a project only uses a certain amount of disk space, you can implement a quota that affects all file systems in the /exampool/fusion file system:

```
#zfs set quota=1TB exampool/fusion
```

Quite the opposite of quotas, if you would like to ensure that a certain amount of disk space is always available, you can set aside a reservation for the file system: The amount of space referred to by CameraDiags will reflect the size of the reservation. Therefore, you cannot set a reservation to a size greater than what is available in the pool.

```
#zfs set reservation=50GB exampool/fusion/CameraDiag
```

The filesystem, including all the data in the nested file systems in /exampool/fusion is now limited to using only 1TB of the pool. Remember though, this can be changed dynamically at any time, by reissuing the above command. Quotas can be used to have an idea of how much space a project expects to use, so that pools can be created with the scale of the projects in mind. Also remember, that if the pool begins running short on available space, more disks can be easily added to the pool:

```
#zfs attach exampool c1t4d0 c2t1d0
```

Finally, if you did want to move data from one disk to another, an easy option is to clone. Cloning is a writable copy of a snapshot. They are quick to create, and use no additional space when created as it is populated with the files of the original system. Remember, this is a copy-on-write system.

```
#zfs  snapshot exampool/fusion@090605
#zfs clone exampool/fusion@090605 exampool/plasma
```

VI. Known Issues

When experimenting with and testing the implementation of ZFS, the administrators at PPPL encountered a few minor setbacks. The first issue encountered was the project disks randomly unmounting, and/or not being available for mounting for brief intermittent intervals. It was found that this was caused because of excess traffic created by the Name Server Caching Daemon (NSCD). Nscd is a daemon that provides a cache for the most common name service requests. It turned out that as the number of shares, hosts in the access lists or number of clients increased, a serious bottle neck occurs, thereby reducing the application performance. [7] The problem was therefore solved by disabling the NSCD.

Another consideration is the management of the performance of NFS with ZFS. NFS was developed with the needs of file systems from 20 years ago. In the past, only a few directories were shared out. With ZFS, the number of shared exports can easily creep in to the double or even triple digits! Since theoretically, ZFS can perform at the platter speed of your hard drives, NFS becomes the limiting factor when it comes to the performance of ZFS. This is why you will not find performance evaluations of ZFS in this paper.

Another observation was that ZFS has Access Control List's (ACL's) turned on by default. Usually this does not cause a problem, as the default settings are the same owner and group detailed in the long listing of the file. However, this ACL became a concern to some users at the lab. There was no known way of turning off ACL's on the server side, as the only options were to clear out the ACL, but not remove it from the file. The solution used was to disable ACL in the automounting options (noacl).

Finally, care should be exercised when forcing a scrub. While a normal scrub happens gradually over time, it seems by forcing a scrub, performance degrades, and the users may notice this degradation. Therefore it is suggested that if a forced scrub is desired, a cron job should be scheduled to perform the scrub on an off-peak time.

VII. Final Remarks

Currently, ZFS has been ported for use not only on Solaris 10, but also FreeBSD [8] and Mac OS X Server (Snow Leopard 10.6) [9]. Due to licensing restrictions, ZFS is only available on one Linux flavor, FUSE. In Jan 2009, a similar file system called BTRFS was merged into the Linux kernel 2.6.29. [10]

The information included here is just a sampling of what ZFS has to offer. These were the options and information that were relevant to the implementation and administration at PPPL. If you are interested in implementing this at your institution, let this be an inspiration to your administrators. ZFS is far from intimidating, and is simple to set up and administer. Your organization will enjoy the benefits of a dynamic file system, with limitless storage capabilities and the reassurance of data integrity. If you would like to learn more, the sites listed offer a wealth of knowledge to assist you on your road to freedom.

References

[1] Sun Microsystems, "ZFS FAQ at OpenSolaris.org", http://opensolaris.org/os/community/zfs/faq/#whatstandfor

[2] A. Rich, "ZFS, Sun's Cutting-Edge File System," http://www.sun.com/bigadmin/features/articles/zfs_part1.scalable.jsp

[3] "Understanding ZFS compression," http://www.cuddletech.com/blog/pivot/entry.php?id=983

[4] Sun Microsystems "Sun Fire X4240 Server," http://www.sun.com/servers/x64/x4240

[5] J. Bodwick, B. Moore, "ZFS: the last word in file systems," http://www.sun.com/software/solaris/zfs_lc_preso.pdf

[6] C. Swearingen "Using ZFS: How to Solve Administration Problems with ZFS" http://www.sun.com/software/solaris/learning_center/linux_mag_zfs.pdf

[7] T. Haynes, D McCallum, "The Management of NFS Performance With Solaris ZFS," http://developers.sun.com/solaris/articles/nfs_zfs.html

[8] ZFS on Linux/FUSE: http://zfs-on-fuse.blogspot.com

[9] ZFS on Apple Mac: http://developer.apple.com/adcnews

[10] S. Kerner, "A Better File System for Linux?" http://www.internetnews.com/dev-news/article.php/3781676/A+Better+File+System+for+Linux.htm

The Princeton Plasma Physics Laboratory is operated
by Princeton University under contract
with the U.S. Department of Energy.

Information Services
Princeton Plasma Physics Laboratory
P.O. Box 451
Princeton, NJ 08543

Phone: 609-243-2750
Fax: 609-243-2751
e-mail: pppl_info@pppl.gov
Internet Address: http://www.pppl.gov