Princeton Plasma Physics Laboratory

PPPL- 4672

PPPL- 4672

Ensuring High Availability And Recoverability of Acquired Data

C. Pugh, T. Carrol and P. Henderson

August, 2011

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

Ensuring High Availability And Recoverability Of Acquired Data*

C. Pugh, T. Carrol, P. Henderson
Information Technology Division
Princeton Plasma Physics Laboratory
Princeton, NJ 08075
cpugh@pppl.gov

Abstract—**Every time one runs a shot, or simulation, exorbitant amounts of data are collected and sent off to live a life in storage. This data is important to our livelihood as a scientific research community, and to the goals of our mission of sustainable energy. Therefore it will behoove all to ensure the integrity of this data.**

Many mechanisms are available to store and ensure the availability of this data, from Hardware Raid, to Software Raid, and backups. Is the right amount of data redundancy being utilized in order to ensure data is safe? What are the scenarios in which these redundancies could fail? How can one ensure that each type of failure is accounted for with the least amount of overhead?

When using Hardware Raid on the storage networks, each Raid group is allowed a certain number of failures, before the whole group fails beyond recovery. Software Raid, specifically ZFS raid-z or mirroring, can check for "soft errors," and provide a way to recover, even if a hard disk fails or a device is prematurely removed. Finally, backups are only as good as the policy and resources provided to the system.

As with many engineering decisions, it is often not clear what the best solution is. Alone, each one of these mechanisms provides a certain level of data redundancy or availability. However, when one would combine these resources, it will ensure that no matter what scenario, data will be available and recoverable.

Keywords-data aquisition; raid; zfs; storage; backup; restore

I. INTRODUCTION

Former Executive and president of Hewlett-Packard, Carly Fiorina pointed out that, "The goal is to transform data into information, and information into insight." It is crucial to science to ensure that the data that is collected and produced is available to gain this insight. Therefore we need to ensure this data is properly replicated and stored using the most efficient methods available. Some things that should be taken into consideration are how the data is stored, and how it is backed up. It is important to remember, however, that certain risks, performance trade offs, and other options should be considered. This paper will explore different options available in order to ensure high availability and recoverability of data, specifically using ZFS/SAN solutions and disk-to-disk staging for backups.

II. DATA STORAGE

The first part of ensuring you have the data to transform, is to write it somewhere for safekeeping. In this case, let's consider using a SAN to provide shared storage, and ZFS pools on Solaris 10 servers.

A. Combining power of SAN and ZFS

Relying on hardware RAID at the SAN level does not account for such issues as user or controller error. For example, what happens with there is an accidental disk deletion? What happens if, for some reason, a controller loses the last configuration change?

Imagine this scenario. You create a non-redundant pool on a server, called TANK in the interest of maximizing your resources. You configure and present three virtual disks from your SAN. These virtual disks have a WWN ending in 4e:0a, 4e:0b and 4e:0c. Now that these disks are presented, you put TANK into production. Now, imagine some time later, you are informed by someone, that they no longer require their disks on another server, WAGON. Among the various disks they no longer need, you have one particular disk that has the WWN that ends in 4c:0b. Now, you may see that in a list of many long WWN's, "4c:0b" and "4e:0b" might look similar. It is fair to say, that as humans make mistakes, one of these disks could hastily be mistaken for each other, so imagine now, that "4e:0b" is deleted instead of "4c:0b". Now instead of cleaning up from an old server, you have now degraded your pool without a chance for recovery. In fact, if this exact scenario were to happen on TANK, unless a later patch fixed the bug, the whole server would become inoperable, and littered with I/O errors, requiring an immediate reboot before proceeding with restoring from backups.

*This work supported by the US DOE Contract No. DE-AC02-09CH11466

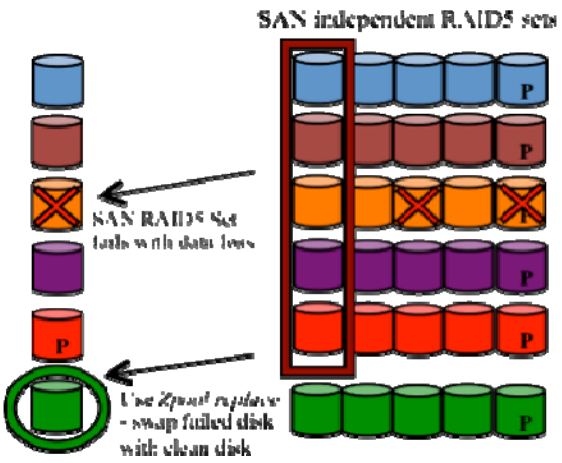

Figure 1. Hardware RAID combined with RAID-Z If a SAN RAID set experiences data loss, this scheme allows the RAIDZ set to recover simply by adding another virtual disk from a clean data set.

If either Raid-z or mirroring are employed, and one virtual disk that was presented to your pool is removed, your data is still intact. This is because you have a level of redundancy that protects your data. All you would have to do, once this issue is discovered, is simply create a new disk on the SAN, present it

to the degraded pool as a replacement, using *zpool replace*, for the "failed" disk with the WWN of "4c:0b". The Figure below shows an example of using both hardware RAID on the SAN, and also the added level of protection of using RAID-Z.

B. *RAID5 issues*

What about non-human errors? Sure, RAID5 devices recover one failed drive, but there are issues that you may not be aware of. The problem is that despite the improved reliability of modern drives and the improved error correction codes on most drives, and even despite the additional 8 bytes of error correction that EMC puts on every drive disk block (if you are lucky enough to use EMC systems), it is more than a little possible that a drive will become flaky and begin to return garbage, known as partial media failure. Now SCSI controllers reserve several hundred disk blocks to be remapped to replace fading sectors with unused ones, but if the drive is going these will not last very long and will run out and SCSI does NOT report correctable errors back to the OS! Therefore you will not know the drive is becoming unstable until it is too late and there are no more replacement sectors and the drive begins to experience partial media failure. [Note that the recently popular IDE/ATA drives may not include bad sector remapping in their hardware so partial media failure may be experienced even sooner.] When a drive experiences partial media failure, and this corrupted sector is written back, a corrupt parity will be calculated and then the RAID5 integrity is lost, as RAID5 does not check parity on read. Similarly if a drive fails and one of the remaining drives is flaky the replacement will be rebuilt with corrupted sectors, also, propagating the problem to two blocks instead of just one.

Furthermore, during recovery, read performance for a RAID5 array is degraded by as much as 80%. Some advanced arrays let you configure the preference more toward recovery or toward performance. However, doing so will increase recovery time and increase the likelihood of losing a second drive in the array before recovery completes resulting in catastrophic data loss. RAID10 on the other hand will only be recovering one drive out of 4 or more pairs with performance ONLY of reads from the recovering pair degraded making the performance hit to the array overall only about 20%! Plus there is no parity calculation time used during recovery - it's a straight data copy. [2]

C. *Performance*

It is important to balance performance tradeoffs with data integrity. As mentioned above, RAID10 is one way to prevent data loss. RAID10, in both hardware and zfs, is accomplished by first creating a RAID1 mirror set and then concatenating those sets into a RAID0 striped set. Below you will find a graph from a study that Simon Krenger performed. [1] It compares the performance of hardware RAID10 and zfs RAID-Z(10). As you can see, there is very little difference in performance, when using hardware RAID10 vs ZFS RAID-Z(10). The added flexibility and security of ZFS might be more important in your environment than pure performance. Overall, ZFS is 3-13% slower than a Hardware RAID, depending what load you apply to the file system, and the Hardware that you use. The higher than expected performance of RAID-Z could be attributed to the fact that it never has to do read-modify-write, like hardware RAID does.

But how does RAID10 compare to RAIDZ(1)? Recall that RAIDZ only sacrifices the capacity of one drive, whereas RAID10 is striped across mirrors, and therefore has an effective capacity of 50% of the assigned drives. In a test performed by Ben Tiefert, it was found that RAIDZ experienced a performance loss when performing small random reads, as compared to a ZFS RAID10 setup. This is because more devices had to participate in each individual read operation, reducing the speedup possible through parallel reads, as is possible with mirrors. In his test, Ben setup 20 drives into four RAIDZ virtual devices of five drives each. This had an overall parity-to-data ratio of 1:4, or 25%. This means that one fifth of our drive capacity is used for parity.

Now, when performing large writes, in this example, a 100GB file, a performance gain was demonstrated. The yield was a write performance of 669 MB / sec; faster than the RAID10 result of 458 MB / sec. This can be attributed to spreading the workload over more devices, as only 6.25 GB was written to each drive. The stripe of mirrors required that 10 GB be written to each drive. The limiting factor for throughput was the PCI-X bus, which wrote 836 MB / sec of total information (data + parity) in order to support the payload of 669 MB / sec of data. (669 x 1.25 due to a 4:1 data to parity ratio.) In the table below, you can see the results of Ben's test. [8]

ZFS RAID TYPE	Disk setup	Test	I/O Summary			
raidz	5x4	rand-read1	273ops/s	4.3mb/s	14.6ms/op	114us/op-cpu
raid10	10x2	rand-read1	412ops/s	6.4mb/s	9.7ms/op	65us/op-cpu

TABLE I. I/O SUMMARY RESULTS FOR RAIDZ VS RAID10 PERFORMANCE TEST

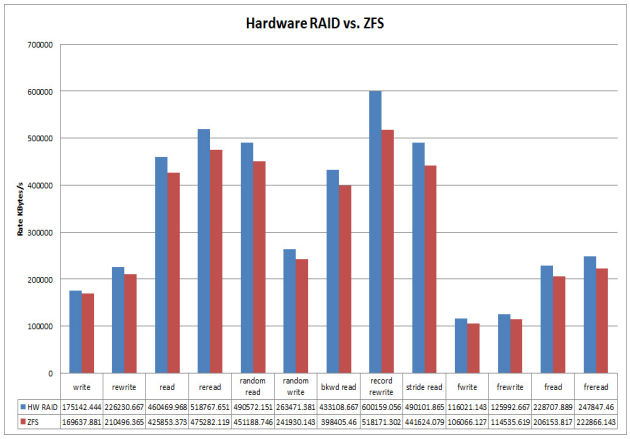

Figure 2. Comparing Hardware RAID10 with ZFS RAID10

D. *Resiliancy*

One of the main bonuses to using ZFS is data checksumming to prevent silent data corruption. Silent data corruption can be caused by any number of factors, including, but not limited to current spikes, disk firmware bugs, bit rot, and/or cosmic radiation. Data Integrity is a high priority in ZFS because none of the currently widespread file systems nor Hardware RAID provide sufficient protection against such problems. This is due to a fatal flaw known as the RAID-5 write hole. Whenever you update the data in a RAID stripe you must also update the parity, so that all disks XOR to zero -- it's that equation that allows you to reconstruct data when a disk fails. The problem is that there's no way to update two or more disks atomically, so RAID stripes can become damaged during a crash or power outage. [3]

However, ZFS, when used with redundancy such as RAID-Z or mirroring, has the capability to check for silent data corruption, AND fix it. Whenever you read a RAID-Z block, ZFS compares it against its checksum. If the data disks didn't return the right answer, ZFS reads the parity and then does combinatorial reconstruction to figure out which disk returned bad data. It then repairs the damaged disk and returns good data to the application. ZFS also reports the incident through Solaris

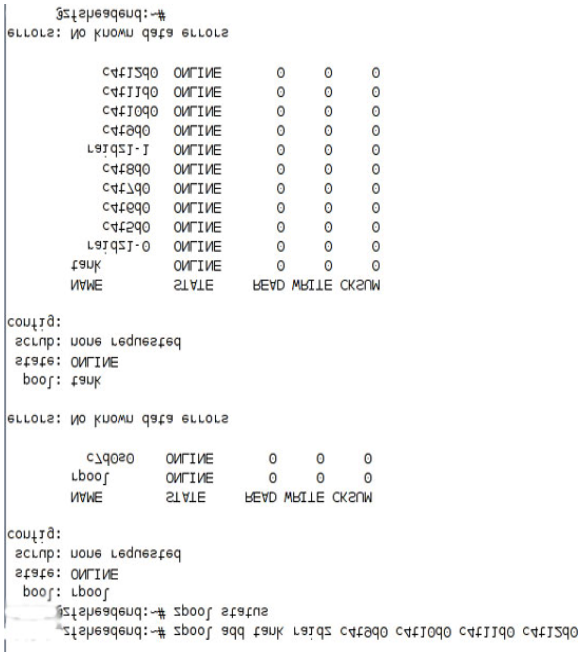

Figure 3. Example of how to create a nested RAID-Z set

FMA so that the system administrator knows that one of the disks is silently failing. Another thing to consider is that RAID-Z is a data/parity scheme like RAID-5, but it uses dynamic stripe width. Every block is its own RAID-Z stripe, regardless of blocksize. This means that every RAID-Z write is a full-stripe write. This, when combined with the copy-on-write transactional semantics of ZFS, completely eliminates the RAID write hole.

One thing to consider is that, while RAID-Z may show high performance in writing speed, data integrity, etc. there may be a read performance issue. If there are many small random reads from many streams and the dataset is large enough where the cache hit ratio is really small. Therefore, another solution could be to make a pool with many nested RAID-Z sets. The figure below is an example of how to create a nested RAID-Z set. Of course this would mean less available storage, but better performance (in terms of IO/s). Therefore, if you want to use RAID-Z in your environment, first carefully consider your workload. If many nested RAID-Z sets in one pool is not a viable solution for you, perhaps one of the other RAID algorithms offered by ZFS could be.

III. DATA RECOVERY

We have discussed different ways of preventing data loss, however, it is important to be prepared, in the event of a loss. Loss can occur accidentally, due to human error, or due to mechanical error, etc.

A. *Disk to Disk Backup Staging*

Any good storage plan needs a good backup plan. Magnetic tape has been the backup medium of choice for a long time. The advantage of tape is cost; it's less expensive than other storage options. However, the tradeoff is performance. As the amount of data that organizations have and need to back up has grown, the amount of time it takes to back up all that data to tape has become increasingly inconvenient. Likewise, finding data on tape is a time-consuming process. With massive amounts of data being stored, it becomes difficult to ensure that backups go according to schedule.

In order to alleviate the stress of waiting for tapes to become available for a backup to take place, one could employ the use of disk to disk backup staging. Disk to disk backup has the benefit of faster backups and restores, occurring at disk speed. The backups are done on the disk staging area, where they are kept short-term. This aids in ensuring data is properly backed up. The figure below shows a simplification of the process of disk to disk backup. First the data is staged to fast SAS disks on a SAN. Once all data is backed up on the Staging Area, a backup image tar file can be written to tape. Therefore, time is used more efficiently, as one does not waste time looking for a tape, only to find out that the system cannot be backed up for reasons such as: the system is off, or improperly configured.

Another added benefit to disk to disk staging is the added benefit of temporary on disk backups. Since image tar files are kept until space is needed, recent backups are readily available for restores, making recent accidental deletions easy and quick to restore. This is achieved by using "High and Low water marks." A high water mark dictates when oldest tar files are purged down to the low water mark.

B. Verified Backups

Having backup policies mean nothing if they are not enforced. It is sometimes difficult to keep up with numerous backup policies, especially if there are multiple admins, who might also be spread thin, on other tasks. Therefore, it is important to have a procedure of checking, or verifying backups, in place. One such method is to use a script to traverse through the backup policies, checking to see first, is a given filesystem COVERED by a policy, and another to reveal if that policy is even ACTIVE! There are some built in binaries that, when used together, can help you perform sanity checks to ensure that your data is covered. Briefly, here are some of the commands you should be interested in and where they are located. These commands can be used together in a script to ensure everything is covered in a policy.

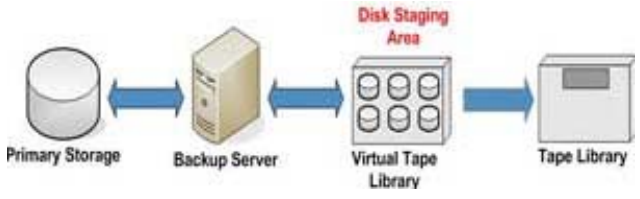

Figure 4. Disk to Disk Staging [4]

- **bperror** - Display NetBackup status and troubleshooting information or entries from the NetBackup error catalog.

  `/usr/openv/netbackup/bin/admincmd/bperror`

- **bpmedialist** - Display NetBackup media status.

  `/usr/openv/netbackup/bin/admincmd/bpmedialist`

- **bpps** - A script to determine which NetBackup processes are active on a UNIX system.

  `/usr/openv/netbackup/bin/bpps`

- **cleanstats** – A script tocheck the status of the cleaning tape(s) and thenumber of hours drives have been used since last cleaning.

  /usr/openv/netbackup/bin/goodies/cleanstats

Once you are sure that your data is covered by a policy, you should also ensure your backups are valid! NetBackup can verify backup images (one at a time), to guarantee that you can restore it. On a command line one could use the command bpverify. bpverify verifies the contents of one or more backups by reading the backup volume and comparing its contents to the NetBackup catalog. This operation does not compare the data on the volume with the contents of the client disk. However, it does read each block in the image, thus verifying that the volume is readable. NetBackup verifies only one backup at a time and tries to minimize media mounts and positioning time.

```
# bpverify -client <CLIENTNAME> -st FULL
```

If you run bpverify without any option, it will verify all taken images from all clients. Additionally, you can choose only the client <client> and only type <FULL>

IV. CLOSING

A. Further Considerations

This is only a few ideas and suggestions to use with SAN/ZFS configurations. Other implementations, such as SAMFS could be used in different ways that have not been explored here.

One of the major milestones for ZFS Storage appliance is data depuplication. Using data depulication in addition to normal backup procedures can help in maximizing backup efficiency. In this way, one would have to back up less, therefore again using time more efficiently. [5]

In addition to RAIDZ(1), ZFS also offers RAIDZ2, and RAIDZ3. RAIDZ2 is like RAID6, where you get double parity and can tolerate up to two disks failing. Performance is similar to RAIDZ. RAIDZ3 has a third parity point, allowing a toleration of up to 3 disks failing. Performance is similar to RAIDZ and RAIDZ2 [6]

B. Conclusion

Many design trade offs should be considered when dealing with massive amounts of data storage. Space, performance, and Mean Time To Data Loss (MTTDL) must be considered and balanced according to use and resources available. As with many engineering decisions, it is often not clear what the best solution is. Alone, each one of these mechanisms provides a certain level of data redundancy or availability. However, when one would combine these resources, it will ensure that no matter what scenario, data will be available and recoverable.

However, if the resources are available it seems that the best combination of performance and resiliency comes from Striping and mirroring everything. This can be done either on the SAN side, or the ZFS side. If you are using the SAN for servers other than ZFS file storing, then perhaps it is prudent to stripe and mirror at the SAN level. This would also mean less management required as the disks would only have to be striped and mirrored as they are added to the storage pool.

It is also important to remember to have regular backups available. Having regular backups means having verified working policies in place, with necessary offsite duplicates available in case disaster recovery is needed. These working backups need to be completely performed on schedule, in order to ensure a backup is available when needed. Using methods such as disk to disk staging, ensures that all backups are quickly performed, and can be written to tape as they become available.

REFERENCES

[1] Krenger, S, "ZFS vs. Hardware RAID (RAID 10) " http://www.krenger.ch/blog/zfs-vs-hardware-raid-raid-10/

[2] Kagel, A, "RAID5 versus RAID10" http://www.miracleas.com/BAARF/RAID5_versus_RAID10.txt

[3] Bonwick, J "RAID-Z" http://blogs.oracle.com/bonwick/entry/raid_z , Nov 18, 2005

[4] "Disk to Disk Backup (D2D)" http://www.storage-backup-archive.com/disk-to-disk-backup.html

[5] Bourbonnais, R. , "Dedup Performance Considerations "http://blogs.oracle.com/roch/

[6] "ZFS RAID levels" http://www.zfsbuild.com/category/raid

[7] "VERITAS NetBackup™ 6.0 Disk-Based Data Protection" http://eval.symantec.com/mktginfo/products/Datasheets/Data_Protection/nbu_6_0_dbdp_dsht.pdf

[8] Tiefert, B. "ZFS Performance – RAIDZ vs RAID10" , July 10, 2009 http://www.stringliterals.com/?p=161

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: 609-243-2245
Fax: 609-243-2751
e-mail: pppl_info@pppl.gov
Internet Address: http://www.pppl.gov