home rss search January 01, 2017

ZFS Raidz Performance, Capacity and Integrity

comparing speed, space and safety per raidz type

ZFS includes data integrity verification, protection against data corruption, support for high storage capacities, great performance, replication, snapshots and copy-on-write clones, and self healing which make it a natural choice for data storage.

The most asked question when talking about raids is, "Which raid is the best?" The best depends on what you are trying to achieve and what you are willing to give up. The question we should be asking ourselves is, "How important is our data ?"

Eventually you will lose a drive and depending on the raid solution will decide if data is lost. If the raid loses a single drive and all your files are gone could be a disaster. You may want to look at a safer raid configuration so the raid can lose two or more drives. The problem with higher data safety is you may have to give up speed or capacity or both.

Raids have three main benefits: Performance, Capacity and Integrity. Performance is how fast the raid will read and write data and is measured in megabytes per second as well as in milliseconds of latency. Capacity is how much does the raid hold. Integrity is how many disks can fail before all the data is lost. The problem is you may not be able to take advantage of all three benefits.

The ASCII triangle below shows all three properties. If you put your mouse cursor in the center of the triangle and move to one property, you see the other properties getting farther away. For example, raid0 is both fast and has the highest capacity, but absolutely no data integrity. Raid1 on the other hand has fantastic integrity and fast reads, but slow writes (multiple copies) and has limited capacity.

```
         Raids have three main advantages over using a sing
                Performance, capacity and data integrity.

                                capacity
                                   /\
                                  /  \
                                 /    \
                                /      \
                   performance /________\ integrity
```

What are the advantages and disadvantages of each raid type ?

- **raid0 or striping array** has no redundancy, but provides the best performance and additional storage. Any drive failure destroys the entire array so raid 0 is not safe at all. if you need really fast scratch space for video editing then raid0 does well.
- **raid1 or mirroring** simply mirrors the same data over every drive in the array. This is excellent redundancy as you can lose every drive except one and still have access to the data. A positive is the raid read speed is increased by every drive added to the array. The big negative is low capacity and slow write speed speeds. No matter how many drives are in the raid you have the total capacity of a single drive to use. Speed is reduced because every drive gets a complete copy of the same files. Mirroring is normally used for two(2) drives, not for 12 and 24 like in our tests due to the incredible amount of wasted space.
- **raid 2, raid 3 and raid 4** are not tested as they are no longer used by the IT industry. Raid2 uses an equal amount of disks as dedicated ECC drives. Raid 3 and 4 use a single dedicated parity drive. None of these raids are used in production anymore due to horrible random read and write performance.
- **raid5 or raidz** distributes parity along with the data and can lose one physical drive before a raid failure. Because parity needs to be calculated raid 5 is slower then raid0, but raid 5 is much safer. RAID 5 requires at least three hard disks in which one(1) full disk of space is used for parity.
- **raid6 or raidz2** distributes parity along with the data and can lose two physical drives instead of just one like raid 5. Because more parity needs to be calculated raid 6 is slower then raid5, but raid6 is safer. raidz2 requires at least four disks and will use two(2) disks of space for parity.
- **raid7 or raidz3** distributes parity just like raid 5 and 6, but raid7 can lose three physical drives. Since triple parity needs to be calculated raid 7 is slower then raid5 and raid 6, but raid 7 is the safest of the three. raidz3 requires at least four, but should be used with no less then five(5) disks, of which three(3) disks of space are used for parity.

- **raid10 or raid1+0** is mirroring and striping of data. The simplest raid10 array has four disks and consists of two pairs of mirrors. Disk 1 and 2 are mirrors and separately disk 3 and 4 are another mirror. Data is then striped (think raid0) across both mirrors. You can lose one drive in each mirror and the data is still safe. You can not lose both drives which make up one mirror, for example drives 1 and 2 can not be lost at the same time. Raid 10 's advantage is reading data is fast. The disadvantages are the writes are slow (multiple mirrors) and capacity is low.
- **raid60 or raid6+0** is a stripe of two or more raid6 volumes. You get the advantage of raid6 safety (lose two drives per raid6 array) and of raid0 striping read speeds. The negatives are the same as raid10.
- **raid70 or raid7+0** is a stripe of two or more raid7 volumes. Just like raid6, you take advantage of raid7 safety and raid0 striping read speeds, but lose capacity.

Be Safe, Not Sorry. When choosing a raid configuration you may look at raidz or RAID5 and see the speed and capacity benefits and decide it is a good choice. From real world experience we highly suggest NOT using raid5. It is simply not safe enough. We recommend using RAID1 mirroring or RAID6 (double parity) or even RAID7 (triple parity). The problem with raid5 is you only have one drive with parity. When one drive dies the raid5 array is degraded and you can not loose another drive before you lose the entire array; i.e. all your data is lost. What happens most of the time is one drive dies, you replace the drive and the array starts resilvering or rebuilding. This is when the other disks are being stressed and during this rebuild is when you have a very good chance (around 8%) that you you will lose another drive. For more information please take some time to read NetApp Weighs In On Disks which was written in 2007, but is still valid today since drive technology has not changed much.

Specification of the testing raid chassis and environment

All tests are run on the same day on the same machine. Our goal was to rule out variability in the hardware so that any differences in performance were the result of the ZFS raid configuration itself.

- FreeBSD 10.2, patched up to article publishing date
- Apply our FreeBSD Tuning and Optimization performance modifications.
- CPU: Intel E5-2630 single socket six(6) core
- RAM: Kingston 16gig of DDR3 1600MHz
- HBA: Avago Technologies (LSI) SAS2308 9207-8i in a PCI-E 3.0 x16 slot
- CASE: SuperMicro 4U, SuperChassis 846BE16-R1K28B
- OS Drive: Samsung 850 PRO 256GB
- RAID Drives: Western Digital Black 4TB 7200rpm SAS (WD4001FYYG), 24 drives
- MOTHERBOARD: Supermicro X9SRE
- SAS Expander: SuperMicro Back Plane, BPN-SAS2-846EL1
- Server Room Temperature: 21C, 71F, 294K at height of server
- Server Room Humidity: ~40% at front of chassis
- Server Room Air Flow: ~240 cubic meters of air per hour
- Server Sound Pressure Level: 73 dB @ 1 ft measured at the front of the chassis
- Server Chassis Vibration: less then 0.01 m/s2 measured at chassis drive back plane
- Benchmark Suite: Bonnie++ v1.97

Bonnie++ benchmarks

Bonnie++ is a benchmark suite aimed at performing a number of simple tests reporting hard drive and file system performance. The benchmark test uses database type calls to a single file to simulate creation, reading, and deleting of many small files.

The ZFS pools are created with LZ4 compression disabled so the Bonnie++ test data will be written directly to disk. Bonnie++ generates a 16 gigabyte file which was chosen because it could not fit into 16 gig of RAM or the ZFS ARC. Bonnie++ is using four(4) concurrent threads to better simulate real world server loads. A scripted while loop runs bonnie three(3) times and we report the median (middle) performance metric. After each bonnie run the system sleeps for 30 seconds so the load can subside.

Bonnie++ can do asynchronous I/O, which means the local disk cache of the 4k drive is heavily utilized with a flush between ZFS commits once every 30 seconds. Since the disk cache can artificially inflate the results we choose to disable drive caches completely using Bonnie++ in synchronous test mode only. Syncing after each write will result in lower benchmark values, but the numbers will more closely resemble a server which is heavily loaded and using all of its RAM.

```
bonnie++ -u root -r 1024 -s 16384 -d /storage -f -b -n 1
```

Quick Summery: ZFS Speeds and Capacity

The following table shows a summery of all the tests and lists the number of disks, raid types, capacity and performance metrics for easy comparison. The speeds display "w" for write, "rw" for rewrite and "r" for read and throughput is in megabytes per second.

ZFS uses variable-sized blocks of up to 1024 kilobytes. If data compression (LZJB or LZ4) is enabled, variable block sizes are used. If a block can be compressed to fit into a smaller block size, the smaller size is used on the disk to use less storage and improve IO throughput at the cost of increased CPU use for the compression and decompression operations. Take a look at our compression study further down this page.

When building a RAID it is common practice to use the "power of two plus parity" raid infrastructure to maximize parity striping, speed and well as capacity. When using ZFS the standard RAID rules may not apply, especially when LZ4 compression is active. ZFS can vary the size of the stripes on each disk and compression can make those stripes unpredictable. The current rule of thumb when making a ZFS raid is:

- MIRROR (raid1) used with two(2) to four(4) disks or more.
- RAIDZ-1 (raid5) used with five(5) disks or more.
- RAIDZ-2 (raid6) used with six(6) disk or more.
- RAIDZ-3 (raid7) used with eleven(11) disks or more.

Spinning platter hard drive raids

The server is setup using an Avago LSI Host Bus Adapter (HBA) and not a raid card. The HBA is connected to the SAS expander using a single multilane cable to control all 24 drives. LZ4 compression is disabled and all writes are synced in real time by the testing suite, Bonnie++. We wanted to test the raid configurations against the drives themselves without LZ4 compression or extra RAM to use for ZFS ARC. You can expect your speeds to easily increase by a factor of two(2) when storing compressible data with LZ4 enabled and when adding RAM at least twice as large as the data sets being read and written.

```
 config:

        NAME        STATE     READ WRITE CKSUM
        storage     ONLINE       0     0     0
          raidz2-0  ONLINE       0     0     0
            da0     ONLINE       0     0     0
            da1     ONLINE       0     0     0
            da2     ONLINE       0     0     0
            da3     ONLINE       0     0     0

 ##########################################################
```

```
5x 4TB, raidz1 (raid5), 15.0 TB,  w=469MB/s , rw=79MB/s

root@FreeBSDzfs:  zpool create storage raidz1 da0 da1 da

root@FreeBSDzfs: bonnie++ -u root -r 1024 -s 16384 -d /s
Version  1.97       ------Sequential Output------ --Sequ
Concurrency   4     -Per Chr- --Block-- -Rewrite- -Per C
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec 
FreeBSDzfs      16G           469287  69 79083  15      
Latency                        1202us    4047ms         


root@FreeBSDzfs:  zpool status
  pool: storage
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
```

Solid State (Pure SSD) raids

The 24 slot raid chassis is filled with Samsung 840 256GB SSD (MZ-7PD256BW) drives. Drives are connected through an Avago LSI 9207-8i HBA controller installed in a PCIe 16x slot. TRIM is not used and not needed due to ZFS's copy on write drive leveling.

```
1x 256GB  a single drive  232 gigabytes ( w= 441MB/s , rw=

2x 256GB  raid0 striped   464 gigabytes ( w= 933MB/s , rw=
2x 256GB  raid1 mirror    232 gigabytes ( w= 430MB/s , rw=

3x 256GB  raid5, raidz1   466 gigabytes ( w= 751MB/s , rw=

4x 256GB  raid6, raidz2   462 gigabytes ( w= 565MB/s , rw=

5x 256GB  raid5, raidz1   931 gigabytes ( w= 817MB/s , rw=
 5x 256GB  raid7, raidz3   464 gigabytes ( w= 424MB/s , rw=

 6x 256GB  raid6, raidz2   933 gigabytes ( w= 721MB/s , rw=

 7x 256GB  raid7, raidz3   934 gigabytes ( w= 591MB/s , rw=

 9x 256GB  raid5, raidz1   1.8 terabytes ( w= 868MB/s , rw=
10x 256GB raid6, raidz2   1.8 terabytes ( w= 806MB/s , rw=
11x 256GB raid7, raidz3   1.8 terabytes ( w= 659MB/s , rw=

17x 256GB raid5, raidz1   3.7 terabytes ( w= 874MB/s , rw=
18x 256GB raid6, raidz2   3.7 terabytes ( w= 788MB/s , rw=
19x 256GB raid7, raidz3   3.7 terabytes ( w= 699MB/s , rw=

24x 256GB raid0 striped   5.5 terabytes ( w=1620MB/s , rw=
```

Some thoughts on the RAID results...

Raid 6 or raidz2 is a good mix of all three properties. Good speeds, integrity and capacity. The ability to lose two drives is fine for a home or office raid where you can get to the machine in a short amount of time to put in another disk. For data center use raidz2 (raid6) is good if you have a hot spare in place. Otherwise, raidz3 (raid7) might be better for data center use as it might take an admin a week to get there to replace a disk. Of course, if you use raidz2 (raid6) with a hot spare you might as well use raid7. Raid 7 would offer the same total amount of space and allow the raid to lose ANY three drives.

We found that no matter which spinning hard drive we used, the speeds and latencies were about the same. You do not have the get the most expensive SAS drives like we used. Look for any manufacture which label their drives as RE or Raid Enabled or "For NAS systems." Some Western Digital Red 3TB (WD30EFRX) hard drives performed just as well. 7200 rpm drives are around 10 percent faster then 5400 rpm drives, but the 7200 rpm drives run hotter. We recommend staying away from power saving or

ECO mode drives like the Western Digital Green (WD30EZRX) series. These green drives seem to perform poorly in a raid configuration due to their ability to spin down, go into sleep mode and long SATA timings. If you are looking for all out speed, the best performance increase you are going to get is to switch from spinning hard drives to SSDs and/or enable ZFS compression.

Reliability of the pool is directly proportional to the number of disks, quality of the hardware and re-silver time. As the number of disks increase the reliability decreases because there is a higher chance for another disk to fail. Using enterprise SAS drives which have a higher meantime between failure should increase reliability compared to using desktop based personal drives. The longer the raid takes to resilver the higher the chance is another drive will fail during the re-silver process. To deal with the chance of pool failure with more disks we should increase the amount of parity disks. Higher parity count will reduce the MTTDL, Mean Time To Data Loss.

Some Random Observations: Always use LZ4 compression and stick as much RAM as you can afford into the machine. When deciding how many drives to use, drive types do not matter, SAS or SATA are both fine. The more disks in a pool, less space lost to parity equals greater space efficiency. Hard drives die in batches; try to vary the manufacturing date of the drives or buy from different companies. In general, mirrors are better for random IOPs. IOPs do not seem to differ between raidz, raidz2 or raidz3.

To compress, or not to compress, that is the question

Yes, enable LZ4 compression. The command "zfs set compression=lz4 tank" will enable compression on the "tank" pool. The long answer is also yes, and you want to be aware of the positives and negatives to make an informed decision.

Spinning drives in general are really, really slow. The OS wastes a significant amount of time just waiting to get data to and from the hard drives. The main argument against compression on the fly is it uses CPU time when reading or writing to the array and also adds latency. If you have been using computers for a while you might remember "Stacker" from the 1990's. Ideally, we want the system to contact the drives as little as possible because they are so slow.

Compression can reduce the amount of blocks the drive needs to access and the result is faster speeds at the cost of CPU time. What you gain is less blocks read or written to for even moderately compressible files meaning you have effectively increased the aerial density of the drive platters. Compression allows the drive heads to move over a smaller area to access the same amount of data.

The developers of ZFS offer lz4 compression which is disabled by default. Take note that ZFS and compression use multiple CPU cores so the more cores and the faster they are the faster compression will be. With LZ4 compression enabled, the system will use around 20% more CPU time to compress and uncompress files compared to handling raw data. On average lz4 will compress data to a maximum of 2x to 3x.

In our tests we show single drive reads go from 150MB/sec to 1,174MB/sec of compressed Bonnie++ test data with lzjb. Even compressed files like bzip2, zip, mp3, avi, mkv and webm will not waste much CPU time as lz4 is a very inexpensive compression method.

In FreeBSD 10 both lzjb and lz4 are available in ZFS. LZJB is fast, but LZ4 is faster with 500MB/sec compression and 1.5GB/sec decompression speeds. This makes LZ4 50% faster on compression and 80% on decompression compared to LZJB. In FreeBSD 9.1 you may need to use "zpool set feature@lz4_compress=enabled poolname" to enable lz4 on your pool as lz4 is not backwards compatible with non-lz4 versions of zfs.

LZ4's performance on incompressible data is equally impressive. LZ4 achieves higher performance by incorporating an "early abort" mechanism which triggers if LZ4 can not reach a minimum compression ratio of 12.5%. LZ4 is perfect for admins who would like to use compression as a default option for all pools. Enabling LZ4 on all volumes means compression will be used where appropriate, while making sure the algorithm will not waste CPU time on incompressible data and reads will not incur a latency penalty due to transparent decompression. Just incredible.

The following table shows a few raid types without compression and then with lzjb and lz4 compression enabled. Bonnie++ tests use database like files which compress fairly well. You will notice as we add more physical disks, compression makes less of an impact on performance. In fact, with lz4 compression we max out read and write speeds with only three disks in raidz (raid5). Our suggestion is to enabled compression on all arrays because the amount of throughput you gain outweighs the CPU usage and you get to compress data which can save space.

```
24x 2TB  raid5, raidz1   40  terabytes
 # zfs set compression=lzjb tank
  Version  1.96       ------Sequential Output------ --Se
  Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per
  Machine        Size K/sec %CP K/sec %CP K/sec %CP K/se
  calomel.org     80G           478082  87 382614  79
  Latency                         990ms    1013ms


24x 2TB  raid5, raidz1   40  terabytes
 # zfs set compression=lz4 tank
  Version  1.96       ------Sequential Output------ --Se
  Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per
  Machine        Size K/sec %CP K/sec %CP K/sec %CP K/se
  zfsFBSD10       80G           504239  87 431002  81
  Latency                         514ms     844ms

24x 2TB  raid0 stripe  5.5 terabytes SSD
 # zfs set compression=off tank
  Version  1.96       ------Sequential Output------ --Se
  Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per
  Machine        Size K/sec %CP K/sec %CP K/sec %CP K/se
  calomel.org     80G           587993  97 359645  65
  Latency                         109ms     827ms
```

```
24x 2TB  raid0 stripe  5.5 terabytes SSD
 # zfs set compression=lzjb tank
  Version  1.96       ------Sequential Output------ --Se
  Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per
```

All SATA controllers are NOT created equal

The performance of your RAID is highly dependent on your hardware and OS drivers. If you are using the motherboard SATA connectors, they are not going to perform as well as a SATA expander or a dedicated raid card. Just because the SATA port says SATA 6 and comes with fancy cables does not mean the port can move data quickly. The onboard chipsets are normally the cheapest silicon the manufacturer can get away with. This sentiment is true for most of the substandard on motherboard network ports too.

On mailing lists and forums there are posts which state ZFS is slow and unresponsive. We have shown in the previous section you can get incredible speeds out of the file system if you understand the limitations of your hardware and how to properly setup your raid. We suspect that many of the objectors of ZFS have setup their ZFS system using slow or otherwise substandard I/O subsystems.

Here we test the SATA throughput of the same physical Western Digital Black 2TB SATA6 (WD2002FAEX) spinning hard drive and Samsung 840 PRO 256GB SSD on three(3) different interfaces using FreeBSD. We look at the SATA 6 Gbit/s port of a common gaming motherboard by Asus, a Supermicro server motherboard and an LSI MegaRAID raid card. All machines have at least a six(6) core CPU and 16 gig of ram. According to Western Digital the Black series 2TB (WD2002FAEX) should be able to theoretically sustain 150MB/sec sequential read and writes. The Samsung 840 Pro is rated at 540MB/sec reads and 520MB/sec writes. Notice the throughput between motherboard based SATA ports (sata 3 versus sata 6) and a dedicated card.

```
1x 2TB a single drive - 1.8 terabytes - Western Digital

 Asus Sabertooth 990FX sata6 onboard ( w= 39MB/s , rw= 2
 SuperMicro X9SRE sata3 onboard      ( w= 31MB/s , rw= 2
 LSI MegaRAID 9265-8i sata6 "JBOD"   ( w=130MB/s , rw= 6

1x 256GB a single drive - 232 gigabytes - Samsung 840 PR

 Asus Sabertooth 990FX sata6 onboard ( w=242MB/s , rw=15
 LSI MegaRAID 9265-8i sata6 "JBOD"   ( w=438MB/s , rw=23



# The raw bonnie output on FreeBSD 9.1

1x 2TB   a single drive  1.8 terabytes (Asus Sabertooth
  Version  1.96       ------Sequential Output------ --Se
  Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per
  Machine        Size K/sec %CP K/sec %CP K/sec %CP K/se
  zfsASUS         80G           39139   11 25902  8
  Latency                       4015ms     3250ms

1x 2TB   a single drive  1.8 terabytes (Supermicro X9SRE
  Version  1.96       ------Sequential Output------ --Se
  Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per
  Machine        Size K/sec %CP K/sec %CP K/sec %CP K/se
  zfsSuper        80G           31342   6 22420   4
  Latency                       4709ms    4699ms
```

Want more speed out of FreeBSD ? Check out our FreeBSD Network Tuning guide where we enhance 1 gigabit and 10 gigabit network configurations.

How do I align sectors to 4K for a ZFS pool ?

By default, older hard drives have a sector size of 512 bytes. Sectors sizes changed when drives became large enough that the overhead of keeping track of sectors consumed a high percentage of storage space. Many modern drives are now specified as advanced format drives which means they have 4096 byte sector sizes (4KiB). Modern 4K drives includes all SSDs and most 2TB and above magnetic drives. You can check on the manufacture website for your drives "sector size" or "sector alignment".

ZFS will query the underlying device to understand how large the sectors are and use this information to determine the size of its dynamic width stripes. A fine idea unless the hardware lies. Today, drive hardware lies more often than not. Drives claim to have a logical sector size of 512 bytes (ashift=9 or 2^9=512) while the physical sectors are 4Kib (ashift=12 or 2^12=4096). Since the drive lied, ZFS will incorrectly make stripes aligned to 512 bytes. This means stripes will almost always be non-aligned forcing the underlying device to use internal processing time which will slightly degrade write performance. We say slightly because in practice a misaligned drive only writes less then 10% slower then a properly aligned drive. Still, every byte per second counts right?

So, we are forced to manually tell ZFS exactly how we want the sectors aligned since we can not trust the hardware to tell us the truth. The procedure is not hard, but it is a little abstract. Lets setup a single drive in the ZFS pool called "tank." Using gnop we can create .nop devices to force ZFS to align physical sectors to 4K no matter what the hard drives tell us.

```
# Setting up a single drive ZFS pool called "tank" which i

# If you already have an unaligned pool you need to
# destroy it. Backup your data if need to.
 zpool destroy tank

# A 4K gnop device needs to be created which will be chair
# This will create the "tank" pool with *.nop devices ford
# physical sectors.
 gnop create -S 4096 /dev/mfid0
 zpool create tank /dev/mfid0.nop

# Next, the pool needs to be exported and the temporary gr
# devices removed
 zpool export tank
 gnop destroy /dev/mfid0.nop

# Now, the aligned pool can be imported from the "raw" dev
 zpool import tank

# All done. Query ZFS to make sure the ashift equals 12 wh
# is 4K aligned. If you see "ashift: 9" then you are 512b
 zdb -C tank | grep ashift
      ashift: 12
```

Performance of 512b versus 4K aligned pools

So, what performance difference will I see for a properly aligned 4K sector sized ZFS pool? Truthfully, we did not see much difference with one or two spinning disks. Normally less then a 10% speed variation. With alignment you may see a slight increase in hard drive speeds and your setup will be more efficient because the hard drive itself will not have to do any sector splitting or read-modify-write operations.

The performance difference seems to also depend on the type of drives you use. Green or ECO drives perform poorly even with 4K alignment compared to RE (Raid Enabled) drives. To make things more complicated some drives are simply slower then others even from the same manufacturer or family of drives. The only way to make sure you are getting the most out of your equipment is to format the drive as 512b and 4K and do the bonnie++ test on each. If you have many drives, test each one individually and put the "fast" ones in one raid and the "slower" ones in another. Mixing fast and slow drives will make the entire setup slow. It is a pain, we know, but the individual test method works reliably.

For a a new boot drive or raid we still suggest aligning the sectors. If you have a unaligned drive and want to wipe the data and re-align, keep these numbers in mind, test your own equipment and make an informed decision.

```
Western Digital Black 2TB (WD2001FYYG)
 512b 1x 2TB   a single drive  1.8 terabytes ( w=130MB/s
 4K   1x 2TB   a single drive  1.8 terabytes ( w=131MB/s

Samsung 840 PRO 256GB (MZ-7PD256BW)
 512b 1x 256GB a single drive  232 gigabytes ( w=446MB/s
 4K   1x 256GB a single drive  232 gigabytes ( w=438MB/s

##### 512 byte UN-aligned sectors Western Digital Black
#
# zdb -C tank | grep ashift
               ashift: 9
  Version  1.96       ------Sequential Output------ --Se
  Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per
  Machine        Size K/sec %CP K/sec %CP K/sec %CP K/se
  calomel.org     80G           130463  23 65792  11
  Latency                       11645ms    4845ms
```

```
##### 4K aligned sectors Western Digital Black 2TB
#
# zdb -C tank | grep ashift
                ashift: 12
  Version  1.96       ------Sequential Output------ --Se
  Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per
  Machine        Size K/sec %CP K/sec %CP K/sec %CP K/se
  calomel.org     80G           131335  23 66141  11
  Latency                       13651ms    5009ms

##### 512 byte UN aligned sectors Samsung 840 PRO 256GB
```

Your firewall is one of the most important machines on the network. Keep the system time up to date with OpenNTPD "how to" (ntpd.conf), monitor your hardware with S.M.A.R.T. - Monitoring hard drive health and keep track of any changed files with a custom Intrusion Detection (IDS) using mtree. If you need to verify a harddrive for bad sectors check out Badblocks hard drive validation/wipe.

Questions?

Do you have a ZFS Health Script we can use ?

Take a look at our ZFS Health Check and Status script. The script checks disk and volume errors as well as the health of the pool and even when the last zpool scrub was done.

Why is ZFS import not using GPT labels ?

When importing a volume you must point ZFS to the directory with the GPT labels using the "-d" argument. For example our GPT labels start with data1-XX and under Ubuntu the device labels are in /dev/disk/by-partlabel/ .

```
# import pool with GPT labels on Ubuntu for example.

root@zfsRAID:~#   zpool import -d /dev/disk/by-partlabel

root@zfsRAID:~#   zpool status
  pool: data1
 state: ONLINE
  scan: scrub repaired 0 in 0h0m with 0 errors on Mon Ap
config:

        NAME          STATE     READ WRITE CKSUM
        data1         ONLINE       0     0     0
          raidz2-0    ONLINE       0     0     0
            data1-00  ONLINE       0     0     0
            data1-01  ONLINE       0     0     0
            data1-02  ONLINE       0     0     0
            data1-03  ONLINE       0     0     0
            data1-04  ONLINE       0     0     0
            data1-05  ONLINE       0     0     0
            data1-06  ONLINE       0     0     0
            data1-07  ONLINE       0     0     0
            data1-08  ONLINE       0     0     0
            data1-09  ONLINE       0     0     0
            data1-10  ONLINE       0     0     0
            data1-11  ONLINE       0     0     0
```

```
# If you do a straight zpool import ZFS will mount the d
# raw device names which we do not want.
```