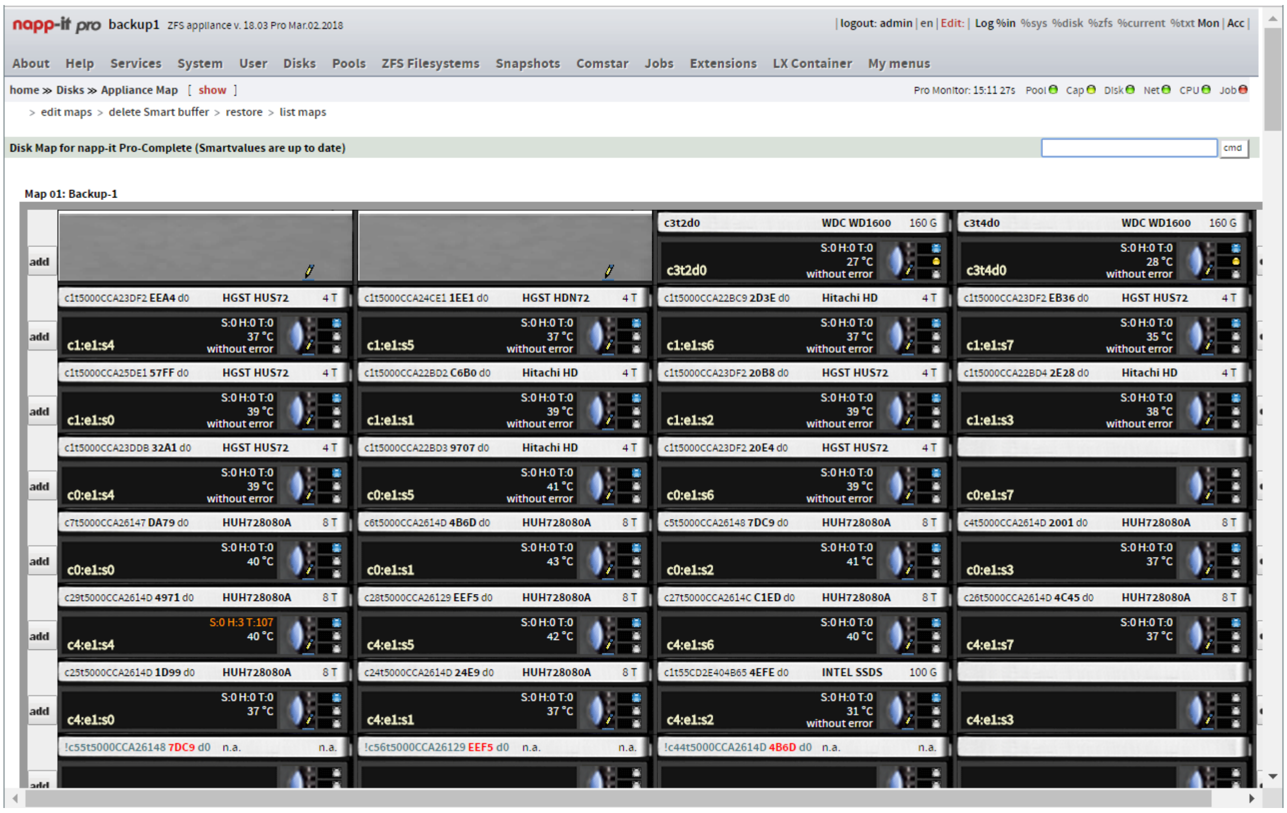
# Design principles for a napp-it ZFS Storageserver

published: 2018-Mar-15 (c) napp-it.org

Content:

# 1. Design principles for a napp-it ZFS Storageserver



napp-it pro backup1 ZFS appliance v. 18.03 Pro Mar.02.2018 | logout: admin | en | Edit: | Log %in %sys %disk %zfs %current %txt Mon | Acc |

About
Help
Services
System
User
Disks
Pools
ZFS Filesystems
Snapshots
Comstar
Jobs
Extensions
LX Container
My menus
home » Disks » Appliance Map [ show ]
Pro Monitor: 15:11 27s Pool Cap Disk Net CPU Job
> edit maps > delete Smart buffer > restore > list maps
cmd

Disk Map for napp-it Pro-Complete (Smartvalues are up to date)

Map 01: Backup-1

| | | | |
|---|---|---|---|
| | | c3t2d0 WDC WD1600 160 G<br>c3t2d0 S:0 H:0 T:0 27 °C without error | c3t4d0 WDC WD1600 160 G<br>c3t4d0 S:0 H:0 T:0 28 °C without error |
| c1t5000CCA23DF2 EEA4 d0 HGST HUS72 4 T<br>c1:e1:s4 S:0 H:0 T:0 37 °C without error | c1t5000CCA24CE1 1EE1 d0 HGST HDN72 4 T<br>c1:e1:s5 S:0 H:0 T:0 37 °C without error | c1t5000CCA22BC9 2D3E d0 Hitachi HD 4 T<br>c1:e1:s6 S:0 H:0 T:0 37 °C without error | c1t5000CCA23DF2 EB36 d0 HGST HUS72 4 T<br>c1:e1:s7 S:0 H:0 T:0 35 °C without error |
| c1t5000CCA25DE1 57FF d0 HGST HUS72 4 T<br>c1:e1:s0 S:0 H:0 T:0 39 °C without error | c1t5000CCA22BD2 C6B0 d0 Hitachi HD 4 T<br>c1:e1:s1 S:0 H:0 T:0 39 °C without error | c1t5000CCA23DF2 20B8 d0 HGST HUS72 4 T<br>c1:e1:s2 S:0 H:0 T:0 39 °C without error | c1t5000CCA22BD4 2E28 d0 Hitachi HD 4 T<br>c1:e1:s3 S:0 H:0 T:0 38 °C without error |
| c1t5000CCA23DDB 32A1 d0 HGST HUS72 4 T<br>c0:e1:s4 S:0 H:0 T:0 39 °C without error | c1t5000CCA22BD3 9707 d0 Hitachi HD 4 T<br>c0:e1:s5 S:0 H:0 T:0 41 °C without error | c1t5000CCA23DF2 20E4 d0 HGST HUS72 4 T<br>c0:e1:s6 S:0 H:0 T:0 39 °C without error | c0:e1:s7 |
| c7t5000CCA26147 DA79 d0 HUH728080A 8 T<br>c0:e1:s0 S:0 H:0 T:0 40 °C | c6t5000CCA2614D 4B6D d0 HUH728080A 8 T<br>c0:e1:s1 S:0 H:0 T:0 43 °C | c5t5000CCA26148 7DC9 d0 HUH728080A 8 T<br>c0:e1:s2 S:0 H:0 T:0 41 °C | c4t5000CCA2614D 2001 d0 HUH728080A 8 T<br>c0:e1:s3 S:0 H:0 T:0 37 °C |
| c29t5000CCA2614D 4971 d0 HUH728080A 8 T<br>c4:e1:s4 S:0 H:3 T:107 40 °C | c28t5000CCA26129 EEF5 d0 HUH728080A 8 T<br>c4:e1:s5 S:0 H:0 T:0 42 °C | c27t5000CCA2614C C1ED d0 HUH728080A 8 T<br>c4:e1:s6 S:0 H:0 T:0 40 °C | c26t5000CCA2614D 4C45 d0 HUH728080A 8 T<br>c4:e1:s7 S:0 H:0 T:0 37 °C |
| c25t5000CCA2614D 1D99 d0 HUH728080A 8 T<br>c4:e1:s0 S:0 H:0 T:0 37 °C | c24t5000CCA2614D 24E9 d0 HUH728080A 8 T<br>c4:e1:s1 S:0 H:0 T:0 37 °C | c1t55CD2E404B65 4EFE d0 INTEL SSDS 100 G<br>c4:e1:s2 S:0 H:0 T:0 31 °C without error | c4:e1:s3 |
| !c55t5000CCA26148 7DC9 d0 n.a. n.a. | !c56t5000CCA26129 EEF5 d0 n.a. n.a. | !c44t5000CCA2614D 4B6D d0 n.a. n.a. | |

If you design a ZFS storage system you must care about the same like with any other server system. You want reliable server class hardware, mainstream/ known to work parts for your OS of choice and you select the hardware according to your intended use case. As any powerful server system is now 64bit, does not matter if its Unix like BSD, OSX, Solarish a Linux system or Windows, you need at least 2GB RAM for the OS with a decent CPU.

## 1.1 What is so special with ZFS?

There is nothing in ZFS that requires extra hardware to be stable. ZFS works with cheap onboard Sata or simple raidless HBA disk adapters. Solaris, the source of ZFS claims 2 GB RAM as minimum does not matter a poolsize. A non-Solaris system may need a little more as the internal memory management of Open-ZFS is still Solaris based. But indeed there are reasons you want to add more resources like RAM to ZFS. For this it is good to compare how ZFS works in relation to a conventional filesystem and what are the consequenses.

If you use a conventional filesystem like FAT and want to edit a textfile ex to replace a „house" with „mouse" you can do this as a infile replacement that requires that you only need to write 1 Byte. If you extend the data and want a „houses" unstead of „house" you may need an extra 512B or 4k diskblock in the extreme together with an update of the metadata. This is very space efficient and fast.

ZFS works different as the above procedure has a risk. If a system crashes during a write, it can happen that the data update + metadata update is only partly written and not completed. This can result in a corrupted textfile or even a corrupted filesystem. These filesystems are not crash resistent. A offline fschk can party detect filesystem inconsistencies and hopely repair them. Sun developped ZFS to address this problem with a mechanism called CopyOnWrite. ZFS organises data in blocks ex 128k. This is the regular amount of data that can be read or written with a special behaviour. When ZFS updates the file, it does never overwrites current data. It writes the modified data and metadata as a new datablock. Only if this was successfully done, the modification becomes valid and the old datablock is marked as available for further writes. A write is done completely or not what makes ZFS crash resistent. A power outage during a write can only corrupt the currently written uncomplete file but never can corrupt the ZFS filesystem. As an extra, the former datablock that contains the former data state can be marked as readonly protected.  This is versioning and called a ZFS snap.

CopyOnWrite and ZFS Snaps gives you an extreme advantage regarding reliability and data security. The price is that you must read/ write much more data and you must expect a higher disk fragmentation as with the former in-file update. ZFS is slower therefor than a former filesystem if you only view the filesystem.

Every data on any filesystem is affected by silent data errors due radiation, a disk surface weakness or simply by chance. This effect happens with a statistical rate what means it becomes more propable with larger data arrays or over time. Another problem may be that data is changed in the chain OS -> disk driver > disk controller -> cables -> backplane -> disk for example due a driver, cabling or power problem. On a conventional filesystem such problems cannot be detected or repaired. You may only discover a problem when a binary file is corrupted (application crashes, damaged images). Sun developped ZFS to adress this problem. ZFS adds a checksum to every of its metadata or datablocks. Any read is verified and any problem can now be detected. This is called end to end data security as it detects all problems on the OS/ZFS subsystem level between data write and data read. The additional checksumming means some extra CPU load and more data. ZFS is slower therefor than a former filesystem.

A storage system needs more capacity than a single disk and requires redundancy to avoid a dataloss on a disk failure. This is why you use Raid. A conventional Raid, does not matter if hardware-raid, software-raid or mainboard-raid, often called fake-raid. All these Raid systems organise a Raid array and offer the whole array to the OS like a single disk but there is a risk with all these Raid arrays called write hole problem, see http://www.raid-recovery-guide.com/raid5-write-hole.aspx. For example if you look at a Raid-1 mirror. On a write both mirror parts must be updated sequentially by the Raid system. A crash during a write can result in a different state of both parts meaning a corrupted Raid, filesystem or file on one or both disks. On reads, you often read from one or the other half for a better performance. By chance you may read good or bad data. Neither the raid controller nor the OS can detect such a problem or decide with half is good or bad. A hardware Raid with Cache+BBU/ Flash-backup can reduce the problem. ZFS can detect these problems as it controls both disks, has checksums, read the other half on problems and repair the raid (self-healing).

## 1.2 What is ZFS?

ZFS is a „perfectly happy" package

**ZFS is a filesystem, just like ext4, FAT or ntfs.**
compared to the other it adds security with data/metadata checksums and CopyOnWrite with snapshots

**ZFS is a software Raid system with integrated Raid management**
Unlike a conventional Raid this gives ZFS access to every raid-disk and their individual checksums to detect errors. Together with CopyOnWrite on every single disk this avoids the write hole problem of conventional Raid.

**ZFS is a Volume manager**
You can create as many filesystems as you need. Unlike conventional partitions they have no fixed size. They can grow dynamically up to poolsize. Management is done via reservations and quotas (storage virtualisation)

**ZFS is share management (at least on Solarish)**
Sun added kernelbased iSCSI, NFS and SMB sharing within ZFS. iSCSI sharing is now separated in Comstar.
In napp-it you can still set iSCSI sharing as a ZFS filesystem property based on the Comstar service.

We have seen that ZFS must process and write/read more data and is slower compared to a conventional filesystem. This is all done with the idea of superiour data security in mind. But as you want also a superiour performance, you may want to compensate these performance disadvantages and Sun invented a lot not only to compensate the performance disadvantages but to overcompensate - with a proper hardware and setup.

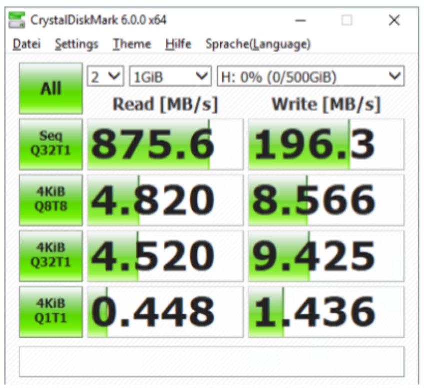
## 2. Hardware suggestions
see http://www.napp-it.org/doc/downloads/napp-it_build_examples.pdf

### 2.1 Mainboard for a ZFS storageserver

ZFS should work with most Intel or AMD server systems from HP, Dell, Lenovo or SuperMicro
If you build yourself or order from a SuperMicro reseller as build to order, prefer systems with Intel serverclass chipsets (c-xxx), ECC RAM, Intel 10G nics and LSI HBAs. CPU is not as relevant for performance than RAM.

### 2.2 RAM

For a stable 64bit OS you need around 2 GB. ZFS does not add any needs for stability, only for performance. As ZFS must write/read more data with a higher fragmentation you want RAM for performance reasons as write cache (default 10% of RAM/max 4GB) to transform many small random writes into one large sequential write and RAM as readcache to cache metadata and random reads. On a well designed ZFS system you can expect that most reads are processed from cache with a superiour performance even with slow disks. With less RAM you may fall back to pure disk performance. As data is spread over a Raid, this can mean that iops on small random reads is the limiting factor. While an average disk can offer around 200 MB/s sequentially (when you only must position a head and the data is streamed on a track) it offers a few MB/s or even KB/s on random write/reads and small datablocks.


CrystalDiskMark 6.0.0 x64
Datei Settings Theme Hilfe Sprache(Language)

2 | 1GiB | H: 0% (0/500GiB)

| All | Read [MB/s] | Write [MB/s] |
|---|---|---|
| Seq Q32T1 | 875.6 | 196.3 |
| 4KiB Q8T8 | 4.820 | 8.566 |
| 4KiB Q32T1 | 4.520 | 9.425 |
| 4KiB Q1T1 | 0.448 | 1.436 |

**Crystal Disk Mark for a Raid-0 array from 4 HGST HE8 disks**

While on sequential parallel reads you can achieve 875 MB/s
the 4KB value is dramatically lower especially with queue depth 1.

If these small data can be delivered from cache you can access them with RAM performance. The same with writes. Small random write is usually extreme slow. ZFS can use several GB RAM as writecache to transform many small random writes into a single large sequential write.

If you want ZFS to be fast you either need ultrafast disks with less RAM or with enough RAM ZFS is fast fast due rambased read and write caching even with slower disks. For a small home/media/backupserver 4-8 GB RAM is ok. For a multiuser system or with VMs use 16-128 GB RAM or more on some use cases.

The ZFS readcache is sophisticated. It works blockbased in a mixture of last read/ most read.

### Is ECC RAM required?

Without ECC a RAM problem on writes cannot be detected and can result in bad data on disk and on reads to bad data in RAM. This is not special to ZFS. ZFS uses all RAM so any bad RAM may affect you. But this is also not special to ZFS as any modern Server OS use RAM for caching with any filesystem. So yes, for a professional or production system ECC is required to detect and repair a RAM flipping on the fly just like you do with disks and RAID.

The often discussed question is, if non ECC RAM on ZFS is more critical than using non ECC RAM on older systems. My answer is no. While data corruption can occur with any filesystem, ZFS verifies data on reads due the checksums. This will result to checksum errors on reads and a disk/pool offline then. A „scrub to death" what means a continued repair with bad data is a myth, http://jrs-s.net/2015/02/03/will-zfs-and-non-ecc-ram-kill-your-data/

## 2.3 Disk controller, Raidcontroller, HBA

On a smaller system, onboard Sata/AHCI is perfect. For Sata hotplug you need mainboard support and must enable Sata hotplug in /etc/system (see System > Tuning). If you need more ports, add one ore more HBAs .

An often discussed question is using a Hardware-Raidcontroller either with its internal Raid and cache functionality or in a single Raid-0 manner where it can offer single disks to ZFS. The first is the worsest as this will introduce the write hole problem to ZFS (partly written write stripes or data/metadata on a crash). On problems ZFS can detect problems due its checksums but cannot repair as it has no access to the underlaying Raid or data. The controller cannot repair as it is not even aware of problems. As even in Raid-0 mode the full Raid fuctionality is active this can lead to a reduced performance or unwanted cache effects. As this mode is not typical for the Raidadapter the firmware may contain undetected bugs. From ZFS view you need the special driver for this adapter. As this driver is not typical, there may be still bugs in it or it may be not as performance optimized. Avoid Hardware raid adapters for ZFS!

If you use an HBA in IT mode, it simply gives all disks 1:1 to ZFS without a controller cache. The driver for an HBA in IT mode is usually maintained by the ZFS distribution itself ex Oracle or Illumos. This option is usually the fastsest and most stabelst option.

If you use an HBA in IR mode you need a different driver. It can offer basic Raid 1/10 functionality and is a good compromise if you do not want to disable a basic controller Raid capability. When ZFS is using such an HBA you must not activate the Controller Raid options. Use them for exampe only for ESXi or Windows. Only use ZFS as a pure software Raid. Only then you will get the whole package of ZFS security and performance.
Use Sata or HBAs, best in IT mode. HBA in IR mode is quite ok but prefer vendors with a good Solaris or OmniOS driver support like Broadcom/ LSI or Atto.

## 2.4 SAS vs Sata disks

If you use an HBA controller you can connect Sata and SAS disks. If you need more ports than the HBA has, you can add another HBA or use an Expander. On the Expander you can also use Sata or SAS disks so what to use?

There are some unique features of SAS like multipathing (a SAS disk has two ports that you can connect to a second HBA or Expander) to double performance or increase availability due the second data path. SAS has a more robust protocol that allows longer cables while Sata is limited to 1m. Whenever you need one of these features, SAS is mandatory. From history SAS is for professional use cases while Sata is for cheap desktop systems. In the past SAS was often much more expensive that Sata and sometimes this is the case even today. With Enterprise class disks you often get SAS at a very minimal premium over Sata.

If you setup a new system you have often the choice of Sata or slighty more expensive SAS disks. In such a case I would always recommend SAS especially when combined with an Expander. While modern expander chips and Sata disks work, there are reports from time to time that a semibad Sata disk on an Expander can initiate unwanted Expander resets or disconnects on good disks. This can make finding bad disks and outages more problematic than with SAS only setups. ZFS itself is not affected by such problems. Unlike hardwareraid ZFS has no problems with multiple disk disconnect/connects.

## 2.5 L2Arc readcache disk

RAM is the key feature for performance. So whenever you want more performance think of more RAM.
If you cannot add more RAM you can add an NVMe (or SSD) as readcache extension. This is not as fast as RAM cache (Arc) but faster than slow disks especially on metadata/ random reads. An extra is that you can enable read ahead on L2ARC disks when you add set zfs:l2arc_noprefetch=0 to /etc/system. This may improve performance on sequential reads. The size of an L2Arc should be around 5x RAM and never exceed 10x RAM as you need around 5% of the L2Arc size in RAM to organise it. The L2Arc should be at least as fast as your network. A failure of the L2Arc is uncritical for ZFS. To check if an L2Arc is worth, use arcstat after some days of warming up the cache.

## 2.6 Slog separate logging disk

To be fast, ZFS is using a quite huge rambased writecache of several Gigabytes. Every write is commited immediatly to the writing system and goes to this writecache and then after some time as a single large sequential write to the pool. On a crash, the content of the cache is lost. On a hardware Raid you would use a cache + BBU/ Flashbackup to protect the data in cache. On ZFS you simply enable sync write as a ZFS filesystem property. After this all writes continue to be written to the ramcache but additionally any single write commit results in a logging of this write either to a ZIL named device on the pool or alternatively to a dedicated separate logging device called Slog that can be much faster than the onpool ZIL regarding small random writes. If the Slog device is based on Flash you must check if the device has powerloss protection, ultra low latency and ongoing high write iops on small steady writes.

A good Slog must combine ultra low latency, high write iops with small random 4k writes under steady load and a powerloss safe behaviour. In the past the best Slogs are DRAM based like a ZeusRam or DDRdrive. The size of the Slog should be at least around 10s of possible writes from your network on Solaris and around 8GB on Illumos systems so you never need more than 10-20GB. The Slog is only for logging. The content is only read after a crash to restore the lost data from the rambased writecache. Its NOT a write cache as ZFS does this in RAM. A failure of the Slog is uncritical as ZFS reverts then to the onpool ZIL for logging. On a diskbased pool an Slog NVMe/SSD is always a good idea to improve sync. On an SSD pool you only need an Slog to add Powerloss Protection when the pool SSDs lack this or when the Slog is much faster than the combined write performance of a whole SSD pool. Sync write and an Slog are not for performance but security. In any case sync write is slower than non-sync write even with the fastest Slog.

The current best of all Slog devices are based on Intel Optane like the 900P or 4800X. The small Optane 16/32GB cache moduls are too slow. The new 800P/58GB seems a good compromise between cost and performance. Compared to traditional Flash, Optane works more like RAM and is not affected by blockwise access only with erase prior write/ Trim/ Garbage Collection or steady write performance degrations what makes them perfect for an Slog. You can use Optane additionally for concurrent read/write workloads like bootdevice, Slog and L2Arc.

**Optane comparison,** https://www.servethehome.com/intel-optane-800p-58gb-and-118gb-m-2-ssd-modules-released/

| Optane model | | Cache 16GB | Cache 32GB | 800P | 900P | 4800x |
|---|---|---|---|---|---|---|
| vs Intel Sata DC 3700 | 200GB | | | | | |
| Interface | Sata | PCI2 3.0 x2 | PCI2 3.0 x2 | PCI2 3.0 x2 | PCI3 3.0 x4 | PCIe 3.0 x4 |
| Powerloss protection | yes | no/? | no/? | no/? | no/? | guaranteed |
| Sequential read | 500MB/s | 900 MB/s | 1350 MB/s | 1450 MB/s | 2500MB/s | 2400 MB/s |
| Random read | 75k iops | 190k iops | 240k iops | 250k iops | 550k iops | 550k iops |
| Write Latency | 65us | 30us | 18us | 18us | <10us | <10us |
| Sequential Write | 365MB/s | 145 MB/s | 290 MB/s | 640 MB/s | 2000 MB/s | 2000 MB/s |
| Random Write | 32k iops | 35k iops | 65k iops | 140k iops | 500k iops | 500k iops |
| Endurance | 10 DWPD | 182TB | 182TB | 365TB | 10 DWPD | 30 DWPD |

The Optane comparison makes it clear. If you want ultimate sync write performance with a guaranteed powerloss behaviour, you need the 4800X as an Slog. This is the enterprise model in all critical environments.

As there is no DRAM cache on Optane you may expect a good powerloss behaviour even on the smaller models although this is not guaranteed by Intel. The Optane 900P gives quite the same performance at a fraction of the price what makes it the best for a high-performance lab environment. For an average lab environment, the 800P/ 58GB seems ideal due the price and good Slog performance, better than any SSD that we preferred in the past like the Intel DC3700. The smaller cache models are simply too slow.

Intel DC3700 Sata, the former number 1 of all affordable Slog options seems no longer an alternative.
Especially the upcoming Optane 800P-64 seems the best option now (as long as you hope for a good enough powerloss behaviour of the Dram free Optanes without a Powerloss Protection guarantee from Intel)

## 2.7 Mechanical disks

If you need capacity for a decent price, you use disks. Disks are available up to 12TB. Beside capacity the most important spec is sequential performance. Typically this is in the range 150-300 MB/s. This is the performance when you read or write to a disk as a datastream. In such a case you do not need to reposition the heads beside on next track so sequential performance is limited by the disk rpm (typically 5400rpm or 7200rpm) and data density.

Sadly you can achieve a mostly sequential workload only when a single user writes a videostream to an empty disk and plays then the video. After some time of usage the disk become fragmented. In a Raid data is spread over the whole array so random read/write performance is the limiting factor. In a worst case scenario where for example small 4k datablocks are spread over the disk, you must reposition the heads for every io and then wait until the data is under the head. In such a scenario you can multiply a max of 100 raw iops of a disk with these 4k blocks what gives you a 400 Kilobytes/s random data rate from a single disk. Real world datarates are between these two extremes.  In a Raid array, the sequential performance can scale with number of datadisks while iops can scale with number of vdevs.  If ZFS has full disk control (no raidadapter with cache between), disks allow secure sync writes but performance is then a fraction of the sequential value (expect 10%). My preferred choice for disks is the HGST HE Ultrastar that is available with an Sata and SAS interface. In any case buy 24/7 models. In a larger backplane prefer enterprise disks that that are prepared for the vibrations in such an environment.

## 2.8 Flash SSD

SSDs offer a slightly higher sequential performance (up to 500MB/s) but much higher iops values. While a mechanical disk can have up to 100 iops, even a cheap desktop SSD is advertized with 20-100k iops. The problem with these adverized values is that they are achieved mainly on reads while write iops are much lower and only achieveable on a new or secure erased SSD for a short time. During steady writes the iops of consumer SSDs can go down to a fraction ex from advertized 50k iops to 5k iops. The reason for this is that SSDs are organized in pages like 4Kbyte and blocks build for example from 128 pages. You cannot write data to a page directly. A block is the smallest entitity that can be written and it must be empty to do.  This means you can write directly only on a new, securely erased ot trimmed disk directly. If there is old data on a page withing a block, you must read the old data, erase the block and write the new datablock as a whole. If you want a good performance even after some time of steady write you either need a trim capable system (this negatively affects performance undera  steadyload) or an efficient Garbage Collection that is done by the SSD firmware combined with a low latency/ high iops system. Only the best SSDs like an Intel DC 37x0 can hold 30-80k iops even on steady writes. As SSDs often have an internal ramcache or as the internal Garbage Collection is not powerloss safe, SSDs should not be used as an Slog device (to protect the ZFS rambased writecache) unless they guarantee powerloss protection, mainly a feature of Enterprise SSDs like the Intel DC line or Samsung PM/SM SSDs.

## 2.9 Flash NVMe

Flash NVMes are similar to SSDs. The main difference is that they do not use an Sata/SAS interface with a performance of around 500 MB/s but PCIe x2 or x4 what allows them 1-2 GB/s sequentially and slightly higher iops values due a lower latency. Beside that they are based on Flash with the same restrictions like SSD. NVMe are available as a PCI-e adaptercard, a small M.2 card to place it directly to a mainboard and removable U.2 for backplanes.

## 2.10 Intel Optane NVMe

Intel Optane NVMes are the first of a new generation storage technology. Unlike Flash their cells are adressed like RAM - not in large blocks, no need for Trim or Garbage Collection, no performance degration on steady writes. You can use them for concurrent read/write workloads without problems. They offer ultra low latency (10us) and up to 500k iops. This makes them the perfect choice as an Slog (optionally paired with an L2Arc cache) or for critical, performance sensitive workloads with small datablocks or datafiles ex databases,  mailserver, VM storage etc. Currently you can buy the enterprise model P4800X with guaranteed powerloss protection, the nearly as good 900P without this guarantee (but as they do not use a ramcache you have a good chance that they operate well on a power outage) and the small 16/32GB cache moduls that are much slower than the 4800/900P option.

## 3. ZFS Pools

### Rpool (ZFS boot filesystem)

For your operating system you should use an extra bootdisk to make OS and data seperate. As you are using a full featured regular Enterprise Unix and not a stripped down NAS distribution, you can at least count around 15GB for a minimal/ text installation of OmniOS/OI or Solaris. For updates you need extra space for the update files and for bootenvironments that preserves the state of the OS prior updates as bootable snaps. Count at least 5 GB for this. Then you should add space for swap and dump, around 3/4 of RAM see https://docs.oracle.com/cd/E23824_01/html/821-1459/fsswap-31050.html. This means around 30 GB for a bootdisk on a 16GB system. Prefer SSDs as bootdevice. My favourites are Intel DC 35x0 with 80 or 120GB due their reliability and powerloss protection (PLP).

While with a newer OmniOS you can use any name for the bootpool, napp-it expects the default name „rpool". It is possible to use rpool also for data ex in a single disk system, but this is not suggested and supported.

### Datapools

The basic idea behind ZFS is to create a datapool from all disks. Then you create filesystems on the pool. They do not have a fixed size like traditional partitions and can grow up to the whole poolsize. To manage space you can set reservations (this space is guaranteed to a filesystem) and quotas (cannot use more). This concept is called Storage Virtualisation. If you need more capacity, you can simply extend the pool by adding new vdevs.

You create a pool in menu Pools > Create where you can select the disks for and the type of the first vdev. If you want to increase capacity, select Pools > Extend and select disks and type of the next vdev.
A pool is usually either optimized for capacity (build from disks) or performance (build from NVMe/SSD). With enough RAM many workloads can be processed from RAM so disk performance is not as important. You can also add an Slog and an L2Arc to improve performance of a slow diskbased pool.
Quite often a dual pool layout is the optimal solution. One large pool build from disk for filer/backup use and one smaller high performance pool build from NVMe/SSD used as VM storage or for databases. For your operating system you need a bootable ZFS pool.

### ZFS datapool layout

A ZFS pool is build from one or more vdevs. Every vdev is a raid and build from disks. A single vdev can be created as a basic vdev from a single disk, as n-way mirror or as a Raid Z(1-3) that works similar to a traditional Raid 5/6 but without the write hole problems of a conventional Raid. If you add another vdev to a pool, you increase pool capacity and performance. A pool from 2 x Raid-Z1/2 vdevs is similar to a traditional Raid 50/60. If you add vdevs to a pool, they are combined in a Raid-0 manner. Unlike Raid 10/50/60 you are not limited to two vdevs but can create a pool from as many vdevs as you want. Due the Raid-0 over vdevs the overall pool reliability is identical to the weakest vdev. If any vdev fails completely the whole pool is lost.

### Basic vdev

A basic vdev is build from a single disk. If you create a pool from basic vdevs you have a pure Raid-0 setup. If you add a basic vdev to a pool with higher Raid-levels and this disk fails, the pool is lost a there is no redundancy.

### Mirror/ Raid-1 vdev

A mirror vdev is build from two or more disks, typically you do not use more than 3. (2-way or 3-way mirror).
If you create a new pool from a mirror vdev you havy a traditional Raid-1 system.  If you add another mirror vdev you get a Raid-10 system. A third and more would result in a Raid-100 or Raid-1000 but these names are not yet defined Raid names so we talk about a multi Raid-10. Multi Raid-10 Pools are used when you want the best performance or an easy expandability of a pool. The sequential write  performance and write iops of such a pool scale with number of mirrors. The sequential read and read iops performance scale with 2x number of mirrors (2 way mirror) or 3x number of mirrors with 3-way mirrors.
A 3-way mirror is good if you do not need more capacity but want a superiour datasecurity where 2 disks can fail per mirror without dataloss and a superiour readperformance (3x of a single disk).

## Raid-Z1 vdev (similar to Raid-5)

A Z1 vdev can be build from two or more disks, typically you use it for 3-5 disks. With two disks a mirror is a better option.  On a Z1, data is striped over all disks. A redundancy is added to the data to allow one disk to fail without dataloss. If one disk fails, the vdev remains intact in a degraded state. If a second disk fails completely the pool is lost. If you only encounter a readerror then, only the data related to that error is lost, not the whole array. This is different to a Raid-5 where such a second error results in a lost array. The sequential performance of a Z1 equals to the added sequential performance of all datadisks (without counting the redundancy disk). As every head must be positioned on an io to a different area of a disk, iops is equal to one disk (50-100 raw iops). With more disks or high capacity disks with a long resilver time you should not use Raid-Z1 but use Z2.

## Raid-Z2 vdev (similar to Raid-6)

A Z2 vdev can be build from three or more disks, typically you use it for 6-12 disks. On a Z2, data is striped over all disks. A redundancy is added to the data to allow two disks to fail without dataloss. If up to two disks fail, the vdev remains intact in a degraded state. If a third disk fails completely the pool is lost. If you only encounter a readerror then, only the data related to that error is lost, not the whole array. This is different to a Raid-6 where such a third error results in a lost array. The sequential performance of a Z2 equals to the added sequential performance of all datadisks (without counting the redundancy disks). As every head must be positioned on an io to a different area of a disk, iops is equal to one disk (50-100 raw iops). With more disks consider to use Z3.

## Raid-Z3 vdev (no convential Raid of this type)

A Z3 vdev can be build from four or more disks, typically you use it for 10-16 disks. On a Z3, data is striped over all disks. A redundancy is added to the data to allow three disks to fail without dataloss. If up to three disks fail, the vdev remains intact in a degraded state. If a forth disk fails completely the pool is lost. If you only encounter a readerror then, only the data related to that error is lost, not the whole array. The sequential performance of a Z3 equals to the added sequential performance of all datadisks (without counting the redundancy disks). As every head must be positioned on an io to a different area of a disk, iops is equal to one disk (50-100 raw iops). .

## Number of disks per Raid-Z vdev/ Golden Number Rule

With a fixed blocksize of say 128k you may want to spread this over the disks in a Raid-Z without a rest (garbage). This would require that your number of datadisks is a power of 2 (128KB spread over 4 disks mean 32KB per disk. When the disks use a physical blocksize of 4KB, you need exactly 8 blocks). Nowadays we mostly enable lz4 compress. As this works on a datablock level we always have a variable blocksize. This means this rule is now senseless, use as many disks per vdev as is optimal for your jbod case or workload, see
https://www.delphix.com/blog/delphix-engineering/zfs-raidz-stripe-width-or-how-i-learned-stop-worrying-and-love-raidz

## Number of disks per Raid-Z vdev/ resilver time

You should no use too many disks per vdev. The reason is simply resilver time. This is the time until a failed disk is replaced by a good one. The time is related to capacity (scale with number of disks) and iops (equal one disk) as you must travers the whole metadata of the pool to decide if data was on the bad disk, then read it from the redundancy and write it to the new disk. From a practical vie I would not use more than say 16 disks per vdev. Only Solaris is faster due sequential resilvering, see https://blogs.oracle.com/roch/sequential-resilvering

## Typical vdev setups

| | |
|---|---|
| SoHo Filer: | single vdev mirror or Raid-Z2/Z3 |
| VM Datastore: | single vdev mirror or Raid-Z2 of NVMe/SSDs or multi Raid-10 of disks (+Slog with disks) |
| Large filer: | multi RaidZ1/2/3 |
| Petabyte filer: | multiple Raid-Z2/3 with 8-16 disks per vdev |

## Pool Scrub/ repair a Pool

ZFS was developped to be crash resistent due CopyOnWrite. A write is done completely (write data, update metadata) or not at all. A corrupt filesystem due a crash on write should not happen unlike you discovered with older filesystems. As you have a second copy of metadata on a different area of the pool, ZFS is ultra robust and the chance of a corrupted filesystem is very very minimal.

But there can always be an error on writing due a bad cable or PSU or you have silent data errors on a pool due radiation, weak surfaces, bad Flash chips or simply by chance. For this your ZFS pool has redundancy. As every datablock has checksums and ZFS reads them per disk and not per Raid, it detects all errors, errors that a conventional Raid will never be able to detect and repair the affected datablock and the Raid on access. Usually this on access method is sufficient but maybe you want to get informed early if error rate goes up indicating a future failure of a disk. For this you can start a online Scrubbing. This reads and verifys all data on a pool and repair data, metadata or Raid when an error is found (and ZFS finds them from time to time, good to have ZFS then).

This online Scrubbing is different to a Filecheck that you have done on ext4 or ntfs. A filecheck must be done offline and it cannot repair data (no checksum to detect errors) but tries to repair a corrupted filesystem. After a filecheck the filesystem may be consistent but full of damaged file contents without further notice. This is why ZFS has no and need no Filecheck as this is done online on every read.

With regular NAS disks a Scrub should be done once a month, with Enterprise disks less often. A Scrub takes a similar time as a disk replace and can last one or two days on heavy load even longer.  Scrub is a low priority process. During high load load it runs longer as fileservices have a higher priority. Despite this you should run in a low priority time ex Saturday/ Sunday.

Despite: On a real disaster, you need a disaster backup on a different media. This is the case even with ZFS.

## 3.1 ZFS filesystem

A ZFS pool is also a ZFS filesystem. In a simple config you may store data directly to your datapool but you should not. Use the toplevel pool-filesystem only as a container for user-filesystems and as a place where you can set defaults for filesystems (example compress) or basic reservations.

I suggest that you create an additional parent filesystem like data on a pool like tank with all user filesystems like invoices, video, audio or pictures below. The reason is replication where you can replicate a filesystem only below a target filesystem. If you want to preserve the source structure on a target system this is only possible if you for example replicate tank/data recursively to the toplevel filesystem (pool) ex backup.  This will result in the intended backup/data structure on the backup system and you have the option to replicate it back with identical structure.

## 3.2 How many ZFS filesystems should I create?

ZFS allows to create as many filesystems as you want. You can for example create a filesystem per user or per use case what can mean hundreds. From a practical view, create only as many filesystems as you need for SMB/ NFS shares (as they are filesystem properties) with simple folders below or as many filesystems as you want for independent replications or an individual snap history (snaps are also a filesystem property) or when other ZFS properties should be different between filesystems.

## 3.2 Shares (iSCSI, NFS, SMB)

You mainly use napp-it for filesharing. Sun developped Solaris and ZFS with in ZFS and the kernel embedded sharing services for iSCSI, NFS and SMB based on pure Sun technologies as ZFS properties. Sun invented NFS and integrated its own SMB server into ZFS (Solaris only). This means sharing is zero config without compatibility or performance problems that you may discover with 3rd party services like SAMBA that must work on any X platform. You only need to enable the property. For a greater flexibility with enterprise class iSCSI demands Sun later added the powerfull Comstar framework for iSCSI. Napp-it still allows to enable iSCSI as a filesystem property but based on Comstar as the background technology.

### iSCSI

If you want to offer a fraction of a ZFS pool to a client you use iSCSI.  iSCSI Targets are blockdevices and can be used like a local disk. They are be based on a RAW disk, a file or mostly a zvol, a ZFS filesystem what gives them full ZFS flexibility like dedicated ZFS properties ex sync, snaps and replication.

Advantage: low overhead (fast), multipath options and you can format it with a client filesystem ex ext4, ntfs but with ZFS as background technology that adds checksums and CopyOnWrite

Disadvantage: Without a Cluster software you can use iSCSI targets only by one client at a time (no concurrent access). As you can snap only the whole filesystem, you can only restore the whole target not a single file.

Performancewise iSCSI is mostly not faster than NFS or SMB when you use similar settings especially regarding sync. If you force sync for the ZFS zvol filesystem, you always use sync on iSCSI. With sync=default you can set the sync write behaviour also via the target writeback setting (writeback=off forces sync write then). For most use cases prefer NFS over iSCSI for the simple reason: Simplicity with similar performance. In an ESXi AiO setup with a staggered bootup of a storage VM and then the other VMs booting from a network datastore on ZFS, NFS allows a automatical reconnect of NFS datastores. All VMs can start then automatically. without enabling the datastore manually.

### NFS v3

Sun invented the network filesystem NFS to mount a remote folder within the local filesystem with performance as the main and only concern. This is why NFS (v3) comes without authentication or authorisation. Depending on an OS, new files are created as the nobody or the client uid. Access can be restricted only based on ip as a share setting or with the help of a filewall. Main usecase of NFS v3 is VM storage ex for ESXi or other virtualisation platforms as it offers performance and easyness. You can place several VMs on NFS shares with shared access to each. If you enable SMB sharing to the same filesystem can can use Windows and Previous versions to backup/ move/ copy/ clone a former version of VMs or single files even when VMs are running.

As there is no authentication with NFS you should set file permissions recursively to everyone@=modify. If you want to share additionally via SMB you should restrict permissions there with share level permissions only. The newer NFS versions 4.0 and 4.1 offer authentication but when you need this feature you should think of SMB as this offers a mode advanced featureset.

SMB

SMB is the most important filer service. Originally it is the file service protocol of Windows and Microsoft still sets the standards like SMB 1, SMB 2.1 or SMB3. As it is available on all platforms, use it when you want access restrictions, performance and features. Over SMB you have access to features like „previous versions". On ZFS this gives you access to ZFS snaps. Only on Solaris snaps and shares are both properties of a ZFS filesystem. This is the reason why this feature works there out of the box without any extra setting. As the Solaris ZFS permissions are based on NFS4 permissions they work very similar to ntfs permissions including the option to inherit permissions to files or folders or both or add SMB groups to SMB groups. Permissions on Solaris are related to Windows SID. This allows that you move a ZFS pool on Solarish to another AD member server with AD permissions intact.

## 4. napp-it is installed, what now?

After installing napp-it you must reboot or check if the current bootenvironment is the default (menu Snapshots). Then you should reset the root password with a new or same password to create an additional SMB password.

Next step is to create a datapool example tank from an initial vdev in menu Pools > Create from your disks with a workload in mind, example a Raid-Z (1-3) for capacity reasons or a mirror for performance reasons. You can then extend the Pool in menu Pool > extend. This means adding mode vdevs like another mirror for a Raid-10 or a Raid-Z. If you have a spare disk you can assign it as a hotspare. If you have more spare disks use them as cold spare as a flaky pool may otherwise eat up spare disks. This can be fixed with Disks > Remove but state is a little confusing.

Next step is creating user filesystems. I suggest to create a parent filesystem like tank/data and then your file-systems like vm, sales, videos etc below to allow a recursive/ complete replication of data to another pool at root level example backup with the filesystem structure intact (ex backup/data then).
For an SMB filer use, set SMB to on (menu ZFS filesystems, click on off to enable). If you create a new filesystem this is also an option in the create menu. If you want anonymous access to an SMB share you must check if you have a user „guest" and enable the guest account with the command „smbadm enable-user guest" at console or the napp-it cmd form field (below the status ample). The guest user is automatically created with the current wget installer. If you want restricted access, do not enable SMB guest sharing, add users in menu „Users", then connect the SMB share from Windows as user „root" and set the permissions from Windows with the security tab. These permissions are then active when you connect as a user. If a user has same name/pw on Solarish like on Windows (s)he can connect without entering name/pw.

Solarish uses NFS4 permissions. They are quite identically to Windows ntfs permissions with all the advanced ntfs options and inheritance settings together with Windows SID to keep permissions in an AD environment intact when you move a pool to another Solarish AD member server.
Optionally share ZFS filesystems in menu „ZFS filesystems" via NFS or iSCSI. If you want to share a filesystem via NFS ex for ESXi, set all file permissions recursively to everyone@=modify. You can restrict access via ip or firewall only. If you share via NFS or iSCSI for applications/VM with older non crash resistent filesystems or for databases where you need a transaction safe behaviour, enable sync on NFS or disable writeback for iSCSI (mean the same).

### 4.1 Essential Job settings

At this point, you should set some jobs to be prepared for a crash or to activate versioning so you can go back to a former disk or file state. Goto menu Jobs and
- enable AutoJob (Job > Autojob), set to every 5 or 15 min
- set a Backup Job ex once a day. This will backup your napp-it settings like /var/web-gui/_log to a datapool
  If you need to reinstall napp-it from scratch you can restore manually or via menu Users > Restore (Pro)
- set the clock, create an other job with command „ntpdate pool.ntp.org", ignore return values
- set a scrub job for your datapool ex once a month (ex first saturday) to check for silent errors
- set an email alert Job (set mailserver settings in About > Settings) in napp-it free or Job > Reports in Pro
- set Autosnap Jobs for versioning ex (keep sets number, hold minimum days, both limits are and related)
  - A job to keep 4 snaps for the last hour (snap every 15 min, keep 4, hold=„keep value empty")
  - A Job to keep 24 snaps for current day (snap every hour, keep 24, hold=„keep value empty")
  - A job to keep a snap for every day on current year (snap everyday 0:00, keep=365, hold=„keep value empty")

You can then go back to these former snappoints with Windows and „Previous Versions" (this is working out of the box without the need to setup anything) or in a share when you open the hidden folder \\share\.zfs\snapshot

Info: ZFS snaps are readonly by default. After creation you cannot modify, you can only destroy as Solarish admin but this is not possible over SMB so even if a Ransomware get admin permissions on Windows, it cannot destroy ZFS snaps like it can do with Windows Shadow Copies. Versioning with Snaps is the first and most important way to protect your data. In most cases this allows a data recovery - you must care additionally only for a real disaster like fire, flash, sabotage or theft. This requires a backup to one or more another server or offline disk set.

## 4.2 ZFS backup

Prior ZFS backup meant a full copy of all files to disks or tapes. After an initial full backup you create ex daily incremental backups for some time to limit amount of data. After some time ex a week you create a new full backup to a new disk or tape set with additional incremental backups then . The former set can be places externally.

There are some problems with this procedure. You cannot usually backup open files or you cannot restore a single file directly but must propable parse a full backup and the following incrementals. Main problem is that it is slow. It is not capable for short delay backups with open files on a high load server. Backups lack the datasecurity that you have with ZFS checksums and and ZFS Raid redundancy. This is why it is not the way you do backups with ZFS.

On ZFS your suggested backup procedure is different and is based on versioning on the main filer
plus a disaster backup on external backup systems.

### ZFS versioning (on the data filer directly)

With local snaps and a decent pool configuration ex based on Raid-Z2 where any two disks are allowed to fail, you are well prepared to regain a former data state when disks fail, Ramsomware affects you, you accidetially deleted a file or if you want to go back to a former state of a filesystem for whatever reason. If you follow the suggested AutoSnap options, you can restore any file or folder directly on the source filer from a state every 15 min in the last hour, a state every hour for current day or a daily for last year. You propably will never need anything else.

### Disaster backup

But then a real disaster affects you. Someone has stolen your server or a fire destroyed your server. For this you need at least a disaster backup. This is a 1:1 physical copy of your whole server that you should keep up to date at least on a daily or weekly base, on a critical system down to a minute delay.

Your main disaster backup target is a ZFS pool. You can use a removable diskset with a ZFS pool ex a basic vdev, mirror or Raid Z (1-3) on it. Such a set or several sets can be placed in a safe for a disaster backup. You then simply keep you local filesystem in sync with the backup pool using ZFS replication. Replication is based on snaps and include open files and can be run even on a highload Petabyte system with a delay down to a minute. The versioning that you have done formerly with incremental backups are done via ZFS snaps. Your ZFS backups now have data validity due checksums, data security due redundancy (autorepair of silent data errors) and versioning due readonly snaps. ZFS cannot be modified after creation time, not even by an admin (audit-proof).

As incremental replications are based on CopyOnWrite snaps, they only consume the space of modified datablocks. Even thousands of incremental replication snaps do not require extra space unless data is on the source system is not modified.

In a production setup, you use a dedicated backup server on a different location/ room/ building with a backup pool for backup and failover if the main systems fails. Then replicate your main pool over the net. After the initial replication the following incrementals are very quick so you can start the replication job to keep the systems in sync down to a minute delay.

In a critical environment where you always want a backup even if the main backup system is under maintenance, use a second backup system. If you have enough free disk bays, such a backup system can also be used as a spare system for the main filer with original data (move the pool, import and enable the service over the former ip).

If your backup system has enough capacity, you can hold there not only the 1:1 disaster backup but also former versions. This can be done with replication settings keep and hold and options like hours:48,days:32,months:12, years:2 what means means keep one replication/target snap per hour for last 48 hours, one per day for last 32 days, one per month for last 12 months and one snap per year for last two years.

With local snaps, one or more backup server and backups on external media you should be prepared for any sort of troubles. If a backups system can be used as a failover system you are online after a crash in a very short time.

If you follow my suggestion to keep critical services like storage (or a VM server like ESXi) as simple as possible you must not care about a crash. Even if a server/ bootdisk fails completley you must only re-install the OS with napp-it, restore the napp-it setting, import the pool and you are online again. You do not need a backup of your operating system as this is trivial. Time to regain storage services from scratch is only a few minutes.

## 5.0 Encryption

in the light of the new data security rules in the EU (DSGVO) with detailled rules about protection and handling of personal data, a state of the art technical procedure like encryption to protect data and backups seems mandatory to protect them in case of a theft or a hack. Currently Oracle Solaris has ZFS encryption with a key per filesystem and encrypted replication to an encrypted backupsystem. A similar or identical ZFS Encryption in Open-ZFS (ex OmniOS and Openindiana) based on the nearly ready state in the common source OpenSolaris is nearly ready.

## 5.1 Expect new Open-ZFS options in 2018/19

- ZFS Encryption, see https://www.youtube.com/watch?v=frnLiXclAMo&feature=youtu.be
current state, https://github.com/openzfs/openzfs/pull/489

- vdev Removal, https://www.delphix.com/blog/delphix-engineering/openzfs-device-removal
This will allow to shrink a pool say from 3vdev to 2 vdev

- Raid-Z expansion, http://open-zfs.org/w/images/6/68/RAIDZ_Expansion_v2.pdf

more
http://open-zfs.org/wiki/Projects

## 6.0 Problem solving

### Disk related

If a disk in a ZFS raid fails, the pool goes to a degraded state. If more disks fail than the redundancy level allows, the Pool goes to a offline state. Whenever enough disks come back the pool is degraded or online again. If a degraded error remains although all disks are back online, try a Pool > Clear error. Even a flaky backplane with ongoing disconnect/ connect will not affect ZFS unlike a hardware raid where this can lead to a lost array.

If you swap disks to a new disk bay, ZFS detects the new disk location automatically if the disk is using a WWN notation. If you use port based names like c1t1d0 (ex onboard Sata) a disk is not detected as a pool member and the pool goes to degraded. A pool > Export followed by a Pool > Import under same name fixes this.

If you have several hot spare disks on a flaky backplane it can happen that a failure initiates a disk replace with a hotspare. The following error adds the next until all hotspares are in a replace state. A pool status can be very confusing then. You can cancel a replace with a Disks > Remove for the hotspare.

If you cannot use a disk as it has a ZFS label on it (you have forgotten to destroy a pool) you must re-initialize the disk with another filesystem (can also be done on another OS)

Always prefer to use a whole disk for ZFS. Using an Optane as Bootdevice, L2Arc and Slog may be an exception.

Modern HBAs detect disks via their WWN identifications ex c1t5000CCA23DF2EB36d0 and not based on a controller port ex c1t2d0 (first controller, second diskport). This gives a huge advantage in setups with many disks as a WWN number is a property set by the manufacturer like a Mac address of a nic. You can move such disks between controllers or servers and the number keeps the same and ZFS will find all disks in a ZFS Raid automatically. But if such a disk fails in a backlane it may be hard to find without a list that translates WWN ans slots or a map. It is strongly suggested that to create such a list with WWN, serial and slot when you insert the disks. With napp-it you should create a disk map (see page 3 of this document), print it out and place it on top the server. Disk > Map is a Pro feature but you can create such a map during the evaluation period and print it out.

**Vdev related**

If you accidentially add a single basic disk or another unwanted vdev type to a pool you can currently no remove the new vdev. A fix means a Pool backup, proper recreation and restore. There is work to add a vdev remove to Open-ZFS, possibly availabe in 2018

You have for example a 6 disk Raid-Z2 vdev and wants to increase capacity by adding a new disk to the same vdev. This is currently not possible. To increase capacity, you must either replace all disks in a vdev or add a new vdev. There is work to add a vdev  extend to Open-ZFS, possibly availabe in 2018

**Pool related**

If you have problems importing a pool, you can try to import with a different name, readonly, with a missing Slog, without mount or in a former state prior last writes if they introduce troubles (Menu Pool > Import)

On a Pool > Import all disks are read to show possibly importable pools. The menu Pool > Import showa all of them with their state. Especially if you have not destroyed a pool correctly prior re-using disks you must care that you select the proper Pool to import.

If you want to rename a pool, you can do this with a Pool > Export and a Pool > Import with a different name.

If you want to import a Pool on another OS or system, you must care that the new system supports the ZFS version. Between Open-ZFS systems you must care that the new system supports all activated features. Problems can result with different disk partition chemas but in general a Pool move even between platforms is possible.

A pool move between a genuine Oracle Solaris ZFS and Open-ZFS is only possible with an old ZFS 28v5 pool. As this ancient ZFS version does not offer newer Solaris nor Open-ZFS features, this is not an option. When a replication also give problems you must use tools like rsync to transfer data.

**iSCSI related**

If you want to move an iSCSI target to a new Pool, there are different options

Mostly you create an iSCSI target in menu ZFS Filesystem as a filesystem property. This creates a zvol with the LU GUID in its name. To transfer such a target you must only replicate the zvol to a new filesystem and activate iSCSI sharing in menu ZFS filesystem to have it active again with previous GUID

If you create a logical unit and target manually in menu Comstar, the GUID is not part of the zvol. When you move this zvol to a new location, you can imprt the zvol. You must enter the old or a new GUID then. Optionally save/restore Comstar setting for this.

**napp-it related**

If you reinstall napp-it from scratch (via OS setup and wget installer) and want to keep keys, groups and jobs you can restore the content of /var/web-gui/_log manually or via Users > Restore (Pro). If you have created a Backup Job in menu Jobs you find a backup of this folder on your datapool.

If you reinstall napp-it without a jobs backup and want to continue a replication, you can create a new replication job with former source, target and jobid settings. Get the former job-id from a snap listing.

**Perfomance related**

If your appliance performance is not as expected, you should first check local pool performance.
Run menu Pools > Benchmark on current napp-it. This executes random and seqential write performance tests with and without sync and then random and sequential tests on read. Compare your results with http://napp-it.org/doc/downloads/optane_slog_pool_performane.pdf

If the performance is as expected, test the network ex via iperf3. Part of current napp-it is a iperf3 server that you can start in menu Services > Iperf server. You can use a current napp-it also as a iperf3 client in menu System > Network Eth > Iperf Client. For multiple parallel clients you can start multiple iperf3 server on different ports at console and connect each client with an iperf server on a different port.

## 7.0 Add some tools that you want for easy management

For system maintenance you should use the following tools
- midnight commander, a console based filemanager to move/copy data locally
  On current OmniOS the mc from the OmnisOS repo at Uni Maryland can give problems.
  You can download binaries from http://napp-it.org/doc/downloads/mc.zip.
  Copy files to / and set /usr/bin/mc executable (/usr/bin/chmod 755 /usr/bin/mc )

- WinSCP, a free Windows tool to edit or manage files on Unix
- Putty, a remote console for Windows

- best stability with napp-it and realtime html-5 graphs gives a Chromium based browser

## 8.0 Read the manuals for napp-it and Solaris admin
Get them at http://napp-it.org/manuals/index_en.html