napp-it

ZFS Hybridpools on a napp-it ZFS Storageserver

published: 2023-Nov-20

ZFS Hybridpools

1. About Hybridpools
2. Hybridpool Layout
3. Create and manage Hybridpools
4. more

1. About ZFS Hybridpools

ZFS Hybridpools are datapools buld from cheap and large disks for mass storage with additional fast SSD/NVMe disks to improve performance for some data (Tiering), read access (L2Arc caching) or data security on sync writes (Slog).

ZFS Data Tiering is a method to store some data on expensive/fast special vdevs (SSD/NVMe=Tier-1 storage) and other data on cheaper/ slower vdevs (disks=Tier-2 storage) in a hybrid pool. Usually you want newest/hot or performance critical data on the fast Tier-1 and older or uncritical cold data on the cheaper slower/cheaper Tier-2.

With ZFS you have three options for data tiering.

1.1 Blockbased tiering for small io based on physical data structures.

You can control onpool file blocksize with the ZFS recsize setting. This setting affects the maximal size of datablocks a file is split on a pool. A 200K file with 128K recsize is stored in two 128K blocs. If filesize is lower than recsize, blocksize is dynymically (beside draid) reduced to a smaller blocksize to reduce capacity waste. This means for a 200K file and a recsize of 256K or larger that only one 256k block is written. Small io and meta-data is the default ZFS tiering method when you use a hybrid ZFS pool. With a small blocksize like 8K only metadata and very small files are stored on the special vdev tier. A small blocksize setting of 64K will store most office files on the special vdev Tier. For larger files you will see no improvement due a special vdev hybrid pool as they are always stored on the slower Tier-2 vdevs. Metadata and small io is fine for special vdevs with 5-20% of pool capacity.

1.2 Blockbased tiering of whole filesystems based on ZFS small datablock and recsize settings

If you set the small blocksize property of a ZFS filesystem to the same or larger value as recsize ex the default 128, all files not only small ones are stored on the special vdev. For such filesystems performance is equal to a SSD/NVMe only pool but you can decide per filesystem if files should be stored on fast Tier-1 or slow Tier-2 storage. As all files and all snaps of such filesystem are on fast Tier-1 storage, you need a larger special vdev, say 20-60% of pool capacity.

Examples
File=smaller 128K, small blocksize=64K and recsize=128K:
file is stored on the tier-1 special vdevs (blocksize is dynamically reduced ex to 64K when filesize is < recsize)

File=128K or larger, small blocksize=64K and recsize=128K:
file is stored on the tier-2 vdevs (blocksize 128K)

This behaviour is very helpful as you can create a filesystem for hot/performance critical data and another for cold/ archival data based on small blocksize and recsize settings. This is the suggested tiering option on ZFS.

1.3 Manual tiering of hot/cold files

This is the classic tiering approach. You can decide where files are stored or you can move files between fast Tier-1 and slow Tier-2 with filepaths remains the same. This filebased tiering is not supported by ZFS although you can initially tier files due a recsize/small blocksize setting and move files when you switch recsize setting below or above the small blocksize value, then rename the file and copy it back again This results in a move between Tier-1 (special vdev) and Tier-2 (regular vdev) under the identical file path.

Main problem:
You can store files on a selected Tier or move a file between Tiers but any file edit will store the edited file due Copy on Write according to the then current recsize setting on one of the Tiers. Initial tiering of a file is not persistent on ZFS.

Manual tiering is therefor only an option if the default Tier is the special vdev. This means that all new or edited files are always stored on fast Tier-1 (recsize <=small blocksize). To limit the needed capacity of fast Tier-1 storage, you can move cold/older files some time after saving or last edit to Tier-2 vdevs. Hot/active data, metadata and small files is then always on Tier-1, larger and older files on Tier-2. This can be automated via admintools or planned jobs. Such a data tiering of cold data limits needed size of a special vdev Tier.

2. ZFS hybrid pool layout

A special vdev is not a cache device but a part of a pool. This means that a special vdev lost is a pool lost.
A special vdev can be a 2/3 way mirror not a Raid-Z. Use at least a 2way mirror, for very critical data use 3way.

The size of the special vdev (you can add several vdev mirrors) depend on the amount of data and snaps that you want to store on fast tier-1 storage compared to slow tier-2 storage. If you only want metadata and very small io on tier-1 with a small blocksize setting like 8K, around 10% of pool capacity for the special vdev is a starting point ex 2 TB SSD in a 20TB pool. You can add more special vdevs when more tier-1 capacity is wanted.

If you mainly want office files on tier-1 with a small blocksize of 128K and 256K recsize, prefer 30%+ tier-1 capacity (6TB+ in a 20TB pool). If you plan to use filesystems with large VM files on fast tier-1, consider more tier-1 capacity.

To reduce needed expensive tier-1 capacity, you can move cold data automatically (files created or last edited longer then a week or month ago) from tier-1 to tier-2. Any file editing will automatically move files back to tier-1 if filesystem recsize is <= small blocksize.

To reduce needed expensive tier-1 capacity, you should limit amount of snaps as they remain on tier-1 when data is on tier-1. If you want more snaps, replicate such filesystems quite often to a pure tier-2 filesystems with a more detailled snap history there.

A special vdev is a tiering method for data based on data blocksize. If you use ZFS realtime dedup you can add a special vdev for dedup tables to a pool (also 2/3 way mirror, say 5GB per TB dedup data).

You must care about capacity in tier-1. When full data land on slow tier-2.

3. Special vs other vdev types

A special vdev tier-1 is not a cache but the area where a whole file is stored so unlike a cache it improves read and write of whole files.

Another vdev type for hybrid ZFS pools is a L2Arc cache disk. Caching works on read last/read most ZFS data-blocks ex 128k not whole files. The cache content is a duplicate of pool data what means that a failed L2Arc does not affect pool integrity. As you need RAM to organize L2Arc, size should be no more than say 10x RAM. Main advantage of an L2Arc over the faster rambased ARC is persistency and the option to activate read ahead.

Another vdev type for hybrid ZFS pools is a Slog device. This is not a cache but a protection unit for the ram-based ZFS writecache. Think of it like a BBU unit on hardwareraid. It is never read beside a crash szenario where otherwise lost writes are written to pool on next reboot. A Slog lost does not affect pool integrity beside some data lost if it happens while the system is crashing with data still the rambased writecache. As an Slog must only log a few seconds of last writes, 10GB is enough for an Slog.

3. Create and manage a ZFS hybridpool

3.1 Create a new datapool

Use napp-it menu Pools > Create Pool
to make a pool from one or more Sata/SAS disks either as a single basic vdev or mirror/Raid-Z vdev
Avoid basic vdevs without redundancy. If such a vdev fails, the pool is lost.

3.2 Extend datapool

Use napp-it menu Pools > Extend Pools
to extend a pool by more vdevs like mirrors or Raid-Z. In a hybridpool you can add faster SSDs/NVMes vdevs for data tiering (special vdev mirror), dedup (dedup mirror), L2Arc readcache or Slog sync write protection.

As an option you can add hotspare disks that will replace faulted disks in a pool immediatly.

3.2 Control size and fillrate of Tier-1 special vdevs

Use menu Pools or „zpool list -v" to check special vdev size and used %

napp-it pro omnios ZFS appliance Pro v. 23.dev nov 17.2023

About Help Services System User Disks Pools ZFS Filesystems Snapshots Comstar Jobs Extensions VM Thre

home » Pools

» Create Pool » Extend Pool » Import » Export » Destroy » Shrink Pool » Features » History » Tiering » Poolinfo » Benchmarks » Encrypt Pool on fl

Pools

Pool	VER	RAW SIZE/ USABLE	ALLOC	RES	FRES	AVAIL zfs [df -h/df -H]	DEDUP	FAILM	EXP	REPL	Alt	GUID
rpool	5000	39.5G/ **38.3G**	7.43G	-	-	**29.8G** [30G /32G]	1.00x	wait	off	off	-	9010640539027177116
tank	5000	23.2G/ **18.9G**	1.83G	-	-	**17.1G** [18G /19G]	1.00x	wait	off	off	-	16595921783538269064

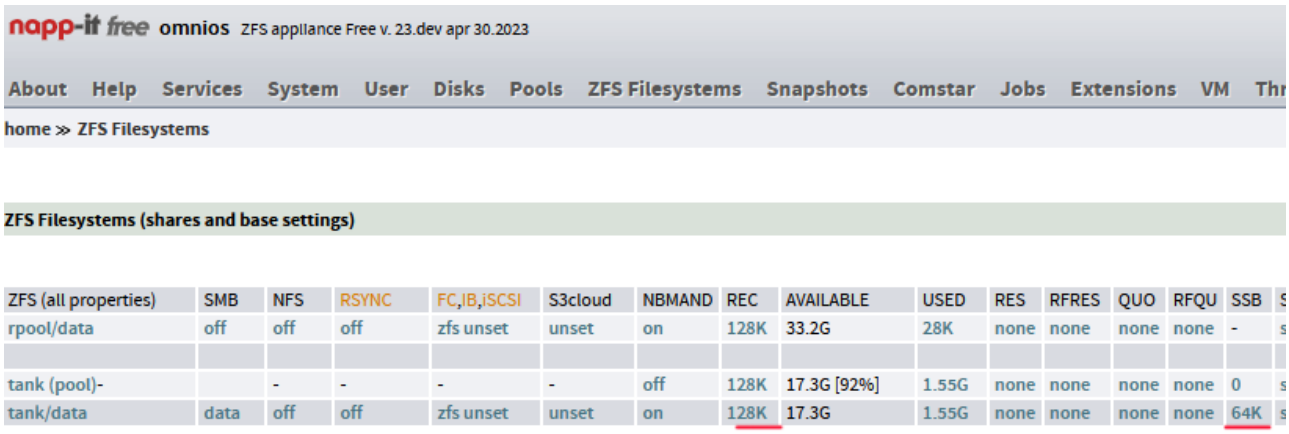
Info: RAW poolsize does not count redundancy, usable/available size is from zfs list, df -h displays size as a power of 1024 wheras df -H displays as a power of 1000

zpool status

```
  pool: rpool
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        rpool       ONLINE       0     0     0
          c1t0d0    ONLINE       0     0     0      42.95 GB       Virtual disk       6000C2994E97B10  S:0 H:0 T:0       -

errors: No known data errors

  pool: tank
 state: ONLINE
config:

        NAME          STATE     READ WRITE CKSUM
        tank          ONLINE       0     0     0
          c1t1d0      ONLINE       0     0     0      21.47 GB       Virtual disk       6000C29699D1895  S:0 H:0 T:0       -
        special vdev              4.03 GB   used: 96%
          mirror-2    ONLINE       0     0     0
            c1t2d0    ONLINE       0     0     0      2.15 GB        Virtual disk       6000C295E588A98  S:0 H:0 T:0       -
            c1t3d0    ONLINE       0     0     0      2.15 GB        Virtual disk       6000C29C45E20B9  S:0 H:0 T:0       -
          mirror-3    ONLINE       0     0     0
            c1t5d0    ONLINE       0     0     0      2.15 GB        Virtual disk       6000C2941246185  S:0 H:0 T:0       -
            c3t1d0    ONLINE       0     0     0      2.15 GB        VMware Virtual N   VMWARENVME0000   S:0 H:9 T:0       -
        cache
          c1t4d0      ONLINE       0     0     0      1.07 GB        Virtual disk       6000C2977DCD21D  S:0 H:0 T:0       -
```

3.2 Control small blocksize and recsize of a filesystem

Use menu ZFS Filesystems and click on current recsize or small block size to modify
or use „zfs set"



ZFS (all properties)	SMB	NFS	RSYNC	FC,IB,iSCSI	S3cloud	NBMAND	REC	AVAILABLE	USED	RES	RFRES	QUO	RFQU	SSB
rpool/data	off	off	off	zfs unset	unset	on	128K	33.2G	28K	none	none	none	none	-
tank (pool)-		-	-	-	-	off	128K	17.3G [92%]	1.55G	none	none	none	none	0
tank/data	data	off	off	zfs unset	unset	on	128K	17.3G	1.55G	none	none	none	none	64K

3.3 Analyze file location

There is no direct method to list files on a special vdev or on other vdevs. But you can list recsize, all files with their onpool blocksize via zdb filesystem and the object/inode with ls -i example

```
fs get recsize,special_small_blocks tank/data;
zdb tank/data | grep plain;
ls -i /tank/data;
```

```
NAME       PROPERTY              VALUE                 SOURCE
tank/data  recordsize            256K                  local
tank/data  special_small_blocks  128K                  local

   Object   lvl   iblk   dblk  dsize  dnsize  lsize   %full  type

        2    1   128K    512      0    512    512    0.00  ZFS plain file
        4    1   128K    512      0    512    512    0.00  ZFS plain file
        6    1   128K    512      0    512    512    0.00  ZFS plain file
       60    1   128K    75K    44K    512    75K  100.00  ZFS plain file
       61    2   128K   191K   200K    512   191K  100.00  ZFS plain file
       63    1   128K    512     4K    512    512  100.00  ZFS plain file
       66    2   128K   256K  1.44M    512  1.50M  100.00  ZFS plain file
      148    1   128K    512     4K    512    512  100.00  ZFS plain file
      151    3   128K   128K   272M    512   284M   98.81  ZFS plain file
      153    1   128K  18.5K    20K    512  18.5K  100.00  ZFS plain file
      155    1   128K    512      0    512    512    0.00  ZFS plain file
      157    1   128K    512     4K    512    512  100.00  ZFS plain file

         61 191k.pdf
          66 PXL_20230825_100451098.jpg
         151 omnios-r151046.iso
          60 75k.pdf
```

The column dblk reports the datablocksize. As long as there is enough space on a special vdev, a dblk size lower or equal the special_small_blocks size indicates that the file is on the special vdev. The Object id is the inode number of a file that is reported via ls -i

To display files with their inode and last edit time you can use

```
find /tank/data -type f -exec stat -c “%i ‘%n‘ %z“ {} \; | grep -v ‘.\$‘

61 /tank/data/191k.pdf 2023-10-14 10:17:14.809267600 +0000
64 /tank/data/PXL_20230825_100451098.jpg 2023-11-19 08:30:08.980373695 +0000
60 /tank/data/75k.pdf 2023-11-13 14:17:07.002134000 +0000
153 /tank/data/Thumbs.db 2023-11-19 08:54:45.258000000 +0000
4 /tank/data/.$EXTEND/$QUOTA 2023-11-11 17:51:04.496913784 +0000
151 /tank/data/omnios-r151046.iso 2023-05-02 08:26:30.278452500 +0000
```

A move of files between Tiers is basically a command like

```
zfs set recsize && rename file file.tier && copy file.tier file && rm file.tier
```

If there remains file.tier, something happened during copy ->

```
rename file.tier file

+ some logs and checks with a script
```

4. more to come

check https://www.napp-it.org/manuals or
https://forums.servethehome.com/index.php?threads/real-filebased-data-tiering-on-zfs.42159/#post-400661 or
https://illumos.topicbox.com/groups/discuss/Ta9815f4d6c901308-M387286ac67f9181a0778c540/files-on-a-special-vdev-and-rule-based-data-tiering