Confusion with used/free disk space in ZFS

oshogbo.com/blog/65

May 16, 2019, 3:55 p.m.

I use ZFS extensively. ZFS is my favorite file system. I write articles and give lectures about it. I work with it every day. In traditional file systems we use *df(1)* to determine free space on partitions. We can also use *du(1)* to count the size of the files in the directory. But it's different on ZFS and this is the most confusing thing EVER. I always forget which tool reports what disk space usage! Every time somebody asks me, I need to google it. For this reason I decided to document it here - for myself - because if I can't remember it at least I will not need to google it, as it will be on my blog, but maybe you will also benefit from this blog post if you have the same problem or you are starting your journey with ZFS.

zpool

Let's create some test pool:

```
# mdconfig -s 1G
# mdconfig -s 1G
# mdconfig -s 1G
# zpool create ztest raidz1 /dev/md0 /dev/md1 /dev/md2
# zpool list
NAME    SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP
HEALTH  ALTROOT
ztest  2.75G   431K  2.75G        -         -     0%     0%  1.00x  ONLINE  -
```

Does it mean that we can store 2.75GBs on the pool? Unfortunately not. zpool under column FREE reports to us the free bytes in the pool. This means that it doesn't count the data redundancy in it. So, every time we write data on the disk a parity data will be written to the pool. In the case of RAIDZ1, the size of the one disk will be used for the parity data. The SIZE value reports the size of the whole pool (so all the disks in the pool).

zpool shows the total bytes of storage available in the pool. This doesn't reflect the amount of data you can store on the pool. To figure out that you should refer to the AVAIL space from the zfs.

```
$ zfs list
NAME                USED  AVAIL  REFER  MOUNTPOINT
ztest               261K  1.71G  29.3K  /ztest
```

What is interesting is in the case of a mirror it will show the size of a single disk.

```
NAME   SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP
HEALTH  ALTROOT
ztest   960M  87.5K   960M      -        -     0%     0%  1.00x  ONLINE       -
```

zfs

```
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
zroot                146G  49.3G  14.4G  legacy
zroot/home           2.0G  49.3G    12K  /home/
zroot/home/def       1.0G  49.3G   1.0G  /home/def
zroot/home/oshogbo   1.0G  49.3G   1.0G  /home/oshogbo
```

The *zfs(1)* command shows us the used and available space per dataset. The used space (USED column) is hierarchal. It means that the size of the *zroot/home/oshogbo* (1GB) is also added to the *zroot/home* (2GB). *zroot/home* contains 2GB because both *zroot/home/oshogbo* and *zroot/home/def* use 1GB and it probably doesn't contain data by its own.

The available space (AVIL) means how much data we can actually write to the dataset. This value refers to the size of data stored after compressions, deduplication, and all the RAIDs stuff.

The available space in our example is exactly the same because all datasets have access to the whole pool. This value may be changed per dataset, for example using quotas and reservations.

The reference data (REFER) means how many data are REFERENCED to the particular dataset (not stored in the dataset). The *zroot/home* refer to 12KBts of space. In this space there is only some metadata, as it is not real data that is stored there. Those data basically say that such a file system exists. Let’s look at the example below:

```
NAME                 USED  AVAIL  REFER  MOUNTPOINT
ztank/test          11.0G   614G  11.0G  /test
ztank/mytests           0   614G  11.0G  /mytests
```

The *ztank/test* is using 11.0G and it has REFERance 11.0G. The *ztank/mytest* REFERENCE to 11.0G but is using 0 storage space. How is that possible? This is because the *ztank/mytest* is a clone of the *ztank/test*. It means that if we would like for example to send *ztank/mytest* to the file, the created file will have 11.0GB size, but physically on our disks *ztank/mytest* doesn't use any blocks.

If we were to start writing to the dataset *ztank/mytest* the USED and REFER amount will be increased:

```
NAME                 USED  AVAIL  REFER  MOUNTPOINT
ztank/mytests        1.19G   612G  12.2G  /mytests
ztank/test          11.0G   612G  11.0G  /test
```

What if we were to remove the data from dataset *ztank/mytest* which refers to the *ztank/test*? The USED value wouldn't change because the data wasn't freed from ztank/mytest, but the reference count will drop.

```
NAME                 USED  AVAIL  REFER  MOUNTPOINT
ztank/mytests         1.51G   612G  1.55G  /mytests
ztank/test           11.0G   612G  11.0G  /test
```

And the last thing that would happen if we freed some space in *ztank/test*? *ztank/test* is we would have a snapshot because *ztank/mytest* was created from it.

```
NAME                 USED  AVAIL  REFER  MOUNTPOINT
ztank/test           11.0G   612G  48.9M  /test
```

The snapshot is using and REFERing to the 11.0GB of data. As mentioned before the USED is hierarchal and means that it counts all datasets and snapshots. This means that 11.0G used by the `ztank/test` is a value of the all underlying datasets and snapshots. If we were to rollback to the state of test snapshot:

```
NAME                 USED  AVAIL  REFER  MOUNTPOINT
ztank/test@test         0      -  11.0G  -
```

It will turn out that the snapshot doesn't use any space because all of our data is stored in the dataset:

```
NAME                 USED  AVAIL  REFER  MOUNTPOINT
ztank/test           11.0G   612G  11.0G  /test
```

To see more details about used space we can run the `zfs list -o space` command.

```
zfs list -o space
NAME                 AVAIL    USED  USEDSNAP  USEDDS  USEDREFRESERV
USEDCHILD
ztank/mytests          612G   1.51G         0   1.51G              0         0
ztank/test             612G   11.0G         0   11.0G              0         0
```

The USED and AVAIL columns we know already.
The USEDSNAP is a space used by the snapshots. If we removed a file like previously this value would go up to 10.9G.

```
NAME                 AVAIL    USED  USEDSNAP  USEDDS  USEDREFRESERV
USEDCHILD
ztank/mytests          612G   1.51G         0   1.51G              0         0
ztank/test             612G   11.0G     10.9G   48.9M              0         0
```

The USEDDS column show the size of files in the dataset - only files without snapshots, reservations etc.
The USEDREFRESERV value is reporting the space used by refreservation for this dataset.
The USEDCHILD value is reporting the space used by its children. If we would go back to this example within the hierarchy:

```
NAME                 AVAIL    USED  USEDSNAP USEDDS  USEDREFRESERV
```

```
USEDCHILD
zroot/home            49.3G    2.0G         0      0             0      2.0G
zroot/home/def        49.3G    1.0G         0   1.0G             0        0G
zroot/home/oshogbo    49.3G    1.0G         0   1.0G             0        0G
```

We see that *zroot/home* does not USEDDS any of the data and its child (USEDCHILD) is using 2.0GBs.

df

The *df(1)* output may be a little bit confusing.

```
Filesystem          Size Used Avail Capacity  Mounted on
zroot               64G  14G   49G      23%  /
zroot/tmp           51G 1.9G   49G       4%  /tmp
zroot/usr           87G  38G   49G      44%  /usr
zroot/var           54G 5.0G   49G       9%  /var
```

Normally *df(1)* reports the size of all of the filesystem in the operating systems. The problem is that all the filesystems (datasets) are using the same pool of data and all of it is available to any of the filesytems. So, if we were to add it up as we use to it would turn out that our disk is much bigger.

This is also the reason why we can't depend on the capacity value. You also may notice that the size of the file system shrinks as space is used up and grows when space is freed. This tool will give us incorrect answers. This may also confuse other tools and windows machines while we mount datasets via SAMBA.

The situation in which we can depend on the df is the used size. This value correspondents to the REFER value of the zfs list, and also to determining where the mount points of datasets are.

du and ls

Let's examine this example:

```
# zfs list
NAME              USED AVAIL REFER MOUNTPOINT
ztank/test        23K   625   23K /test

# du -h file3
512B    file3

# ls -lah
-rw-r--r--  1 root wheel   1.0G May 12 13:40 file3
```

The zfs list says that we are using 23KB of data. *du(1)* is saying a few bytes and *ls(1)* is reporting a GB. The case is the written file is compressed or full of zeros which ZFS also compress.

The *du(1)* tool reports how many bytes are used to store the contents of the files after compression, dedupe and so on.

The *ls -l* shows the real size of the file. If you plan to copy a file to a different FS without compression you need to prepare to have enough disk size.

Summary

The understanding of how ZFS is uses space and how to determine which value means what is a crucial thing. I hope thanks to this article I will finally remember it!

⇐ BSD PL #12 Resuming ZFS send ⇒