January 01, 2017

ZFS Health Check and Status

monitor for degraded or over capacity volumes and disk errors

ZFS on FreeBSD is one of the best file systems we have ever used. But, once we setup the machine we asked the same question every other ZFS admin has asked themselves. How do we tell when there is a problem with our ZFS volumes? To monitor the ZFS filesystems we wrote the following shell script.

The ZFS Health shell script will use standard zpool arguments to look at the state of the volumes and drives. We currently check for the following default conditions:

- **Health** - Check if all zfs volumes are not degraded or broken in any way.
- **Capacity** - Make sure all pool capacities are below 80% for best performance.
- **Errors** - Check the columns for READ, WRITE and CKSUM (checksum) drive errors.
- **Scrub Expired** - Check if all volumes have been scrubbed in at least the last 8 days.

If any problems are found an email is sent to "root" with the current status of the volumes. You can then take action as needed.

We usually try to avoid any graphical user interfaces (GUI) and appreciate the flexibility a command line program gives over a GUI solution. Using some simple shell scripting we can find out the health of the RAID, email ourselves about problems and work with failed drives.

The ZFS Health Check script

You are welcome to copy and paste the following script. We call the script "zfs_health.sh", but you can use any name you wish. Take a look at the section for "scrubExpire" and set the script to either use FreeBSD or Ubuntu's version of the date script format. We commented every method so take a look at the script before using it and make any changes you feel necessary for your environment.

```
    scrubRawDate=$(/sbin/zpool status $volume | grep scrub | awk '{pr
    scrubDate=$(date -j -f '%Y%b%e-%H%M%S' $scrubRawDate'-00(

    if [ $(($currentDate - $scrubDate)) -ge $scrubExpire ]; then

      emailSubject="`hostname` - ZFS pool - Scrub Time Expired. Scru
      problems=1
    fi
  done
fi



# Email - On any problems send email with drive status information an
# capacities including a helpful subject line. Also use logger to write th
# email subject to the local logs. This is also the place you may want to
# any other notifications like playing a sound file, beeping the internal
# speaker, paging someone or updating Nagios or even BigBrother.
```

```
if [ "$problems" -ne 0 ]; then
```

What does an email from the ZFS health script look like ?

The shell script will check the status of ZFS volumes, also called virtual drives, and if the volume is degraded or reporting errors an email is sent. The output of the email will show the capacity of the volumes fist and the full "zpool status" of all the volumes below. The email subject will give a clear indication as to the cause of the email report so you know what to look for.

For example, this email was sent out because the raid is fine, but the three(3) drive mirrored "tank" volume is at 97% capacity. By default the script is setup to send email if any volume is over 80% full. You can change the script to report on any upper limit you think is good for your environment.

```
        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada3    ONLINE       0     0     0
            ada2    ONLINE       0     0     0
            ada1    ONLINE       0     0     0

errors: No known data errors


  pool: root
```

```
 state: ONLINE
  scan: scrub repaired 0 in 0h0m with 0 errors on Sun Mar  3 01:20:38 2
config:

        NAME        STATE     READ WRITE CKSUM
        root        ONLINE       0     0     0
         gpt/root   ONLINE       0     0     0
```

Use cron to run the ZFS health check script

We run the script as an unprivileged user in a cron job. This way when the raid has an issue we get notification. The following cron job will run the script a few times a day when we know we should have time to respond to the error. As long as the raid is having an issue we will get emails. We see this as a reminder to check on the raid if it is not fixed in a reasonable amount of time. Cron will check the health of the ZFS volumes at 7am, 11am and 4pm every day.

```
SHELL=/bin/bash
PATH=/bin:/sbin:/usr/bin:/usr/sbin
#
#minute (0-59)
#|   hour (0-23)
```

```
#|  |    day of the month (1-31)
#|  |    |  month of the year (1-12 or Jan-Dec)
#|  |    |  |  day of the week (0-6 with 0=Sun or Sun-Sat)
#|  |    |  |  |  commands
#|  |    |  |  |  |
# zfs health check and report
00 7,11,16 *   *   *   /home/calomel/zfs_health.sh
```

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

Questions?

Any tips to speed up and optimize ZFS performance ?

ZFS on FreeBSD 9.1 is incredibly fast by default. The internal auto optimization is efficient and the developers should be applauded for their work. For efficiency, we recommend installing at least 8 gigabytes of system memory to cache frequently requested data in RAM and then use the zfs-stats script from ports to monitor the ARC cache hit ratio. Spinning disks are very slow and if all the cached data does not fit in RAM then add a Samsung 840 Pro SSD drive as L2ARC cache or add a ZFS Intent Log (ZIL) drive to reduce spinning disk I/O. If your disk controller is too slow (we are looking at you on-board SATA ports) consider an LSI MegaRaid card. Our LSI MegaCLI Scripts and Commands guide has a ZFS setup at the bottom of the page showing our production ZFS RAID reading at 660MB/sec. Also make sure compression and

deduplication are off. In any applications, like Nginx, disable sendfile and mmap to avoid redundant data caching.

That said, we do make some changes to ZFS. Set the checksum method to fletcher4, enable lzjb compression and turn access time stamps off (atime). The following will set these options on the "tank" zfs volume. Use "zfs get all tank" to verify.

```
zfs set atime=off tank
zfs set checksum=fletcher4 tank
zfs set compression=lzjb tank   ### for ZFS v5 pool v28
zfs set compression=lz4 tank    ### for ZFS v5 pool v5000
```

```
zpool set autoexpand=on tank
zpool set autoreplace=on tank
zpool set listsnapshots=on tank
```

We DO NOT EVER recommend turning off checksums. Checksums (i.e. checksum=fletcher4) are used for data to metadata integrity verification and self-healing (scrub, mirror, raidz, raid-z2, raid-z3) of the volume. If you are turning off checksums you will get a slight speed boost, but the integrity of your data is reduced substantially. For more tuning tips and insights check out the ZFS Evil Tuning Guide.

How can I make rolling ZFS snapshots ?

ZFS snapshots are read-only copies of a file system or volume. Snapshots are created in seconds and they initially consume no additional disk space within the pool. As data is added and deleted the active volume changes and the snapshot consumes disk space by continuing to reference the old backup copy of the data. ZFS snapshots are a very efficient way to make multiple time stamped copies of an entire file system without consuming double the space.

In FreeBSD ports or through packages you can install "zfSnap". zfSnap is a simple shell script to make rolling zfs snapshots using cron. The main advantage of zfSnap is it's written in pure /bin/sh so it doesn't require any additional

software to run and all information about the snapshot is in the snapshot name itself.

Once you have zfSnap installed just setup a cron job as root. In this example we take a snapshot of the root file system and the tank raid array every Sunday at 6am. The snapshot is set to expire in one month. After a backup is made the script then deletes any snapshots which have since expired. The entire processes on both volumes normally completes in less then a second. Note you can also execute this same command before working on the system. Before upgrading packages we make a quick snapshot with an expiration time of 2 days (--2d) in case we make a mistake.

```
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin
#
#minute (0-59)
#|   hour (0-23)
#|   |    day of the month (1-31)
#|   |    |   month of the year (1-12)
#|   |    |   |   day of the week (0-6 with 0=Sun)
#|   |    |   |   |   commands
#|   |    |   |   |   |
#### zfs snapshots
00   6    *   *   0   /usr/local/sbin/zfSnap -d -s -S -a 1m root tank
#
```

You can look at all the snapshots using zfs list. Here we see an example of the zfSnap file naming syntax. We have the name of the volume, "@" sign, the date/time and at the end is "1m" which stands for the expiration time of one month. It is not shown here, but after a month you should see four(4) complete snapshots of your data for each separate volume (root and tank).

```
calomel@RAID:  zfs list -t filesystem,snapshot
NAME                        USED  AVAIL  REFER  MOUNTPOINT
tank                       80.0G  17.7T  80.0G  /tank
tank@2033-03-03_06.00.00--1m     0      -  80.0G  -
root                       7.53G   286G  6.49G  /
root@2033-02-24_06.00.00--1m  2.01M      -  6.33G  -
root@2033-03-03_06.00.00--1m  2.36M      -  6.49G  -
```

That is about it. You now have a cron job creating rolling snapshots of your system. If you have a problem where a file is deleted or missing you can go to your snapshot to recover the file. zfSnap is simple insurance against data loss.