ZFS performance tuning tips

Tune L2ARC for backups

When huge portion of data are written (new backups) or read (backup verify) L2ARC is constantly written with current data. To change this behaviour to cache only Most Frequent Use:

/etc/modprobe.d/zfs.conf

```
options zfs l2arc_mfuonly=1 l2arc_noprefetch=0
```

Explanation:

- l2arc_mfuonly Controls whether only MFU metadata and data are cached from ARC into L2ARC. This may be desirable to avoid wasting space on L2ARC when reading/writing large amounts of data that are not expected to be accessed more than once. By default both MRU and MFU data and metadata are cached in the L2ARC.
- l2arc_noprefetch Disables writing prefetched, but unused, buffers to cache devices. Setting to 0 can increase L2ARC hit rates for workloads where the ARC is too small for a read workload that benefits from prefetching. Also, if the main pool devices are **very slow**, setting to 0 can improve some workloads such as **backups**.

I/O scheduler

If whole device is managed by ZFS (not partition), ZFS sets scheduler to `none`. For rotational devices, there is no sense to use advanced schedulers `cfq` or `bfq` directly on hard disc. Both depends on processes, processes groups and application. In this case there is group of kernel processess for ZFS.

Only possible scheduler to consider is `deadline` / `mq-deadline`. `Deadline` scheduler group reads into batches and writed into separate batches ordering by increasing LBA address (so it should be good for HDDs).

There is a discussion on OpenZFS project to do not touch schedulers anymore and let it to be configured by admin:

- Set "none" scheduler if available (initramfs) #9042
- https://github.com/openzfs/zfs/commit/42c24d90d112b6e9e1a304346a1335e058f1678b

Postgresql

See Archlinux wiki: Databases

```
zfs set recordsize=8K <pool>/postgres
```

```
# ONLY for SSD/NVM devices:
zfs set logbias=throughput <pool>/postgres
```

reduce ZFS ARC RAM usage

By default ZFS can sue 50% of RAM for ARC cache:

```
# apt install zfsutils-linux

# arcstat
    time  read  miss  miss%  dmis  dm%  pmis  pm%  mmis  mm%  size     c  avail
16:47:26     3     0      0     0    0     0    0     0    0   15G   15G   1.8G
```

```
# arc_summary

ARC size (current):                                    98.9 %   15.5 GiB
        Target size (adaptive):                       100.0 %   15.6 GiB
        Min size (hard limit):                          6.2 %  999.6 MiB
        Max size (high water):                           16:1   15.6 GiB
        Most Frequently Used (MFU) cache size:         75.5 %   11.2 GiB
        Most Recently Used (MRU) cache size:           24.5 %    3.6 GiB
        Metadata cache size (hard limit):              75.0 %   11.7 GiB
        Metadata cache size (current):                  8.9 %    1.0 GiB
        Dnode cache size (hard limit):                 10.0 %    1.2 GiB
        Dnode cache size (current):                     5.3 %   63.7 MiB
```

ARC size can be tuned by settings `zfs` kernel module parameters (Module Parameters):

- `zfs_arc_max`: Maximum size of ARC in bytes. If set to 0 then the maximum size of ARC is determined by the amount of system memory installed (50% on Linux)
- `zfs_arc_min`: Minimum ARC size limit. When the ARC is asked to shrink, it will stop shrinking at `c_min` as tuned by `zfs_arc_min`.
- `zfs_arc_meta_limit_percent`: Sets the limit to ARC metadata, arc_meta_limit, as a percentage of the maximum size target of the ARC, `c_max`. Default is 75.

Proxmox recommends following rule:

```
As a general rule of thumb, allocate at least 2 GiB Base + 1 GiB/TiB-Storage
```

Examples

Set `zfs_arc_max` to 4GB and `zfs_arc_min` to 128MB:

```
echo "$[4 * 1024*1024*1024]" >/sys/module/zfs/parameters/zfs_arc_max
echo "$[128     *1024*1024]" >/sys/module/zfs/parameters/zfs_arc_min
```

Make options persistent:

```
options zfs zfs_prefetch_disable=1
options zfs zfs_arc_max=4294967296
options zfs zfs_arc_min=134217728
options zfs zfs_arc_meta_limit_percent=75
```

and `update-initramfs -u`