Performance sucks

If your performance isn't what you expected. Here are some things you can try. Note that for some you might need to recreate your pool and not just edit it.

I myself found a lot of information at jrs-s.net and just by looking around.

- How do I change the options?
  - Example of setting options after creation
  - Example of setting options during pool creation
- Recordsize
- Atime
- Compression
- Ashift
- Write performance on NFS
  - Sync=always with SLOG
- Sources:

How do I change the options?

Example of setting options after creation

```
$ zfs set atime=off main # (Main is pool name)
```

Example of setting options during pool creation

```
$ zpool create -o recordsize=64k .....
```

Recordsize

If you have studied at TI UCLL, you have probably seen this in BS1. This is the block size.

When using a lot of images using the `qcow2` file format you are better of using a recordsize of 64k since that it also what `qcow2` uses.

> **Side note**
>
> If performance is very important, use `raw`. It is quite often much faster than `qcow2`. If you want to see how much visit this article.

Atime

This is literally `access time`. It stores when the file was last accessed. On VM images you don't really care when this was last accessed, so just set this to `off`. This is an operation less to do and could increase the performance.

Compression

Obviously, disable it right? **NO!** Actually compression can increase performance. You can read more on this article of Serve The Home. `lz4` is probably the best idea for balancing compression and performance. If you have files that are rarely accessed, something like archives where performance doesn't really matter. You should probably use zstd. For example, our archive of old VM's and containers are stored in `zstd` with level 7 (maximum). Yes, it uses around 70-80% CPU usage when writing files but we do get a compression ratio of 2+. (Files are uncompressed).

There are some exceptions though. Like enabling compression on a filesystem that only contains compressed files. No point in trying to do compression twice.

Ashift

An `ashift` value of 12 for 512 bytes sector sizes. (diskinfo -c da0, to see the sector size)

> This option needs to be set during the pool creation. Not afterwards

Write performance on NFS

Sources:

- https://jrs-s.net/2019/05/02/zfs-sync-async-zil-slog/
- https://www.servethehome.com/the-case-for-using-zfs-compression/
- https://www.ixsystems.com/blog/o-slog-not-slog-best-configure-zfs-intent-log/
- https://jrs-s.net/2019/04/03/on-zfs-recordsize/

You might notice that the performance on the VM's isn't what you expect. This is normal and difficult to fix with our budget. Why this happens is because the write are done in sync, meaning that if the VM writes something it will check if it was actually written to the disk and isn't still in something like memory of the FreeNAS box. If you were to disable sync and switch to async, you will notice it is much faster. Now you might ask, why don't we switch to async. Good question, the problem is that if a power failure for example happens, and the VM was writing to the disk. It might think that it has been written to the disk but in reality it was still in memory. You would have lost that data and corruption might occur. So disable sync is risky.

This video explains it pretty good:

Sync=always with SLOG

There is some conflicting information about this. At first people state that using sync=always will increase performance.



It doesn't do what you think it does. A lot of people think using a SLOG and setting sync to always will do this:

```
RAM ==> SLOG ==> HDD
```

That doesn't happen however. If you set `sync` to always, all writes will be written to the SLOG disk, async and sync.
This is what actually happens:

```
RAM ==> HDD
    ==> SLOG
```

So your data has to be written to the RAM and to the SLOG devices. When the pool goes offline instantly the data in RAM is lost. That's where the SLOG will kick in. After starting up, ZFS will ask the disk for it's contents and finish writes that where lost in the RAM.

So where is this useful, if you want to be 1000% sure that every write, async or sync is properly written to the disk. More information is here https://jrs-s.net/2019/05/02/zfs-sync-async-zil-slog/.