# Our GlusterFS Experiences



By Andrew Boag

*NOTE: This blog does not aim to explain the workings of GlusterFS, nor provide you with setup and configuration advice. There are plenty of great resources out there on the net to explain these things.*

Catalyst has been working with GlusterFS (www.gluster.org) – also known as 'gluster' – since late 2012 when we built a large enterprise LMS application stack in AWS for one of our Australian University clients.

Before this point, like many in the Linux world, Catalyst had utilised NFS heavily in order to present a mounted file system to a number of web or application servers over the network. While NFS has its challenges, it is a very known quantity that we are extremely comfortable working with.

The background for the choice to try GlusterFS was that it is considered bad form to use an NFS server inside an AWS stack. As this is your “single point of failure” which the AWS Solutions Architects (SA) love to circle and critique on the whiteboard when workshoping stack architecture.

So after some investigation, experimentation and consultation, we decided upon GlusterFS as the file storage solution for our application.

**What other network/clustered/distributed file systems have we worked with?**

Catalyst had experimented with:

- Samba and NFS used heavily
- Lustre - no production use
- OCFS2 - we used in production but ran away after a tragic experience
- DRBD - still using in places. Useful tool
- S3FS - no use for our workloads
- CEPH - the Catalyst Cloud makes heavy use of ceph with great success - see https://catalyst.net.nz/what-we-offer/cloud-services/catalyst-cloud

**What versions of GlusterFS have we worked with?**

We began with 3.2 and worked through 3.3, 3.4, 3.5 and finally 3.6. GlusterFS was running on a number of dedicated EC2 instances, running Ubuntu LTS.

**What were the biggest challenges to adopting GlusterFS?**

Adopting new file system technologies is always an interesting exercise, as assessing a new type of file system solution is not typically part of the application build cycle. Generally, the file system was "just there" and we knew how to manage it, configure it and diagnose

performance issues.

However, with GlusterFS we had to quantify what our expectations were from the file system and then map this into a round of acceptance testing. This was chiefly achieved by a heavy period of application load testing using the powerful open source “Bees with Machine Guns” tool set – https://github.com/newsapps/beeswithmachineguns – essentially DOS tool that doubles as a load testing instrument.

Some of the decisions around the correct infrastructure choice – meaning AWS EC2 instance size and EBS block storage details – were quite arbitrary and we tended towards the traditional approach of over-provisioning capacity.

**What is GlusterFS good for?**

GlusterFS is very suitable for large static files that don't change. In this case large meaning megabytes and above. Examples: media files, document assets, images.

And it's much better if these files don't change i.e. are not opened and re-saved with the same file name as you might with a word document on an office shared drive.

Large immutable (write once, never change) files were a good fit for GlusterFS. But it is important that whichever application is writing to gluster sticks to this rule.

**What was Gluster bad for?**

GlusterFS was not a good solution in the case that the web servers were writing small files (meaning small number of kilobytes) often that change a lot e.g. session.xml being read, updated re-saved, read, updated, re-saved etc.

In our case the culprit was Moodle session files. These small data pieces were being constantly read and updated as part of the session management for our Moodle users.

Without diving into too many details, the problems are generally race-condition related and can result in what is referred to as a “split brain” scenario where the two (or more) GlusterFS servers disagree with the state of a particular file. Not good.

GlusterFS takes some steps to try to automatically resolve split brain issues but there is no perfect way to always determine which of the versions of a file is correct.

**What are some of the gotcha's with GlusterFS?**

You don't get fully redundant file storage for nothing. Some gotcha's we discovered:

- Backup and restoration is a very different challenge to how these things work with a more traditional network file system like NFS. It's all possible but we had to dream up some new bespoke strategies.
- The underlying files on a gluster file system are only visible by mounting externally. You can't ssh onto the gluster server and easily poke around in a local copy of the file system that looks the same as the one you get when you mount externally. This is annoying if the wheels come off (slow performance) and you just want to get in and wrestle some files out.

- Split brain can be bad

**Have we had any nightmares with GlusterFS?**

Yes, we have. Under certain application load patterns the mounted gluster file system can become quite fragile and suffer from frightening performance degradation.

There have been times when even performing an 'ls' within a mounted directory could cause near meltdown.

We put quite a few automated processes in place to monitor, detect and resolve the issues that we discovered through experience.

**What have we learnt?**

The major lesson that we have taken from our exposure to GlusterFS and the general world of clustered file systems is that Clustered/Network file systems are not a one-to-one replacement to a singular local file system on the same machine as your application. And if you are going to experiment with GlusterFS you should think long and hard about the usage patterns that your application is going to expose the file system to, and create automated test cases around these.

And secondly, it is more and more clear to us that in the world of “cloud applications” we should always be trying to move stuff out of a traditional file system. Using caching services, object storage, noSQL solutions and the good old database to hold state and assets.

Still, GlusterFS is still one of the most mature clustered file systems out there. And is certainly worth a look if it might fit your needs. Get in touch if you want some help!

*Thanks very much to Jordan Tomkinson for all his hard work with GlusterFS over the years and for the help with this article.*