Secure storage with GlusterFS

Bright Idea

You can create distributed, replicated, and high-performance storage systems using GlusterFS and some inexpensive hardware. Kurt explains. *By Kurt Seifried*

Recently I had a CPU cache memory error pop up in my file server logfile (Figure 1). I don't know if that means the CPU is failing, or if it got hit by a cosmic ray, or if something else happened, but now I wonder: Can I really trust this hardware with critical services anymore? When it comes to servers, the failure of a single component, or even a single system, should not take out an entire service.

In other words, I'm doing it wrong by relying on a single server to provide my file-serving needs. Even though I have the disks configured in a mirrored RAID array, this won't help if the CPU goes flaky and dies; I'll still have to build a new server, and move the drives over, and hope that no data was corrupted.

Now imagine that this isn't my personal file server but the back-end file server for your system boot images and partitions (because you're using KVM, RHEV, OpenStack, or something similar). In this case, the failure of a single server could bring a significant portion of your infrastructure to its knees. Thus, the availability aspect of the security triad (i.e., Availability, Integrity, and Confidentiality, or AIC) was not properly addressed, and now you have to deal with a lot of angry users and managers.

KURT SEIFRIED

Kurt Seifried is an Information Security Consultant specializing in Linux and networks since 1996. He often wonders how it is that technology works on a large scale but often fails on a small scale.

Enter GlusterFS

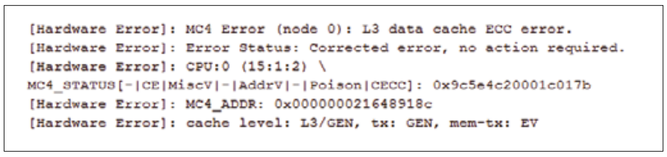
GlusterFS is a network/clustering filesystem acquired by Red Hat in 2011. In a nutshell, GlusterFS has a server and client component. The server is basically "dumb" (i.e., the metadata is stored with the file on the back end, simplifying the server considerably). You create a trusted pool of servers, and these servers contain storage "bricks" – basically disk space, which you can then export, either singularly or in groups (known as volumes) using a variety of replication, distribution, striping, and combinations thereof to strike the right balance of performance, reliability, and capabilities for the volumes you need.

The GlusterFS data can then be exported in one of three ways to clients, using the native GlusterFS client, which is your best bet for performance and features like automated failover, NFS (the GlusterFS server can emulate NFS), or CIFS (using Samba to export the storage). Of course, you could also mount the storage on a server and re-export it using other file-serving technologies.

The GlusterFS server storage bricks are just normal devices with a supported file system. XFS is recommended, and ext4 is supported (more on this later). With no need for special hardware, you could use leftover space on a device to create a partition that is then exported as a storage brick, by dedicating an entire device or making a RAID device and exporting it.

GlusterFS Installation

GlusterFS installation is pretty easy; Fedora, Debian, and several others include



```
[Hardware Error]: MC4 Error (node 0): L3 data cache ECC error.
[Hardware Error]: Error Status: Corrected error, no action required.
[Hardware Error]: CPU:0 (15:1:2) \
MC4_STATUS[-|CE|MiscV|-|AddrV|-|Poison|CECC]: 0x9c5e4c20001c017b
[Hardware Error]: MC4_ADDR: 0x000000021648918c
[Hardware Error]: cache level: L3/GEN, tx: GEN, mem-tx: EV
```

Figure 1: A CPU hardware error.

GlusterFS server and client by default. The upstream GlusterFS project also makes RPMs and DPKG files available [1], as well as a NetBSD package. I strongly recommend running the stable version if possible (3.3.x at the time of this writing), although the latest beta version (3.4) has some really cool features, such as Qemu thin provisioning and Write Once Read Many.

GlusterFS Setup and Performance

As with any technology, there are good ways and better ways to set up the servers. As far as filesystems go, you want to use XFS rather than ext4 if at all possible. XFS is far better tested and supported than ext4 for use with GlusterFS; for example, in March 2013, a kernel update broke GlusterFS on ext4 [2]. Also, if you're using XFS, you'll want to make sure the inodes are 512 bytes or larger because if you use ACLs on XFS, you'll want the inodes to be large enough to store the ACL directly in the inode; if it's stored externally to the inode, every file access will require additional lookups, thereby adding latency and affecting performance.

You also need to ensure the system running the GlusterFS Server has enough RAM. GlusterFS aggressively caches data, which is good for performance, but I have seen cases in which this caused the system to run short on memory, invoking the OOM killer and killing the largest process (in terms of memory use), which was invariably `glusterd`. The same goes for clients, which will need at least 1GB of RAM. I found in production that cloud images with 640MB of RAM would consistently result in the GlusterFS native client being killed by the OOM killer.

As far as storage, I recommend setting up one storage brick per device, or ideally using RAID 5/6/10 devices for storage bricks, so a single failed disk doesn't take out the entire brick on that server. If possible, you should not share devices between GlusterFS and the local system. For example, if you have a locally I/O-intensive process (e.g., log rotation with compression), this could affect the performance of `glusterd` on that server and, in turn, affect all the clients using the volume that include that storage brick.

Setup of GlusterFS and volumes is quite easy, assuming you have two servers with one disk each that you want to make into a single volume, replicated (basically RAID 1, a mirror). From the first server, run:

```
# gluster peer probe ip.of.server.2
# gluster volume create vol1 replica 2 ⮯
  transport tcp 10.2.3.4:/brick1 ⮯
  10.3.4.5:/brick2
# gluster volume start vol1
```

On the client(s), mount the filesystem:

```
# mount -t glusterfs -o acl ⮯
  10.2.3.4:/vol1 /vol1
```

Replicated volumes are pretty simple: Basically it's RAID 1, and the number of bricks should be equal to the replica count. Plans have been made to support more flexible RAID setups (similar in nature to RAIDZ1/2/3, which allows you to specify that you want data replicated to withstand the loss of one, two, or three drives in the group). You can also create distributed volumes; this is most similar to JBOD (Just a Bunch of Disks), in that files are written in their entirety to one of the bricks in the storage volume. The advantage here is you can easily create very large storage volumes.

A third option is distributed volumes; these are basically RAID 0-style volumes. Here, files are written in pieces to each storage brick, and for write and read access, you can get very high volumes (e.g., 10 servers writing one tenth of the file each compared with a single server handling all of it). Finally, you can mix and match these three types. For example, you could create a distributed striped volume for extremely high performance, a distributed replicated volume for extremely high availability, or a distributed striped replicated volume for both performance and reliability.

A note on GlusterFS security: Currently, GlusterFS only supports IP/port-based access controls; in essence, it is meant to be used with a trusted network environment. A variety of authentication and encryption options (e.g., Kerberos and SSL) are being examined; however, nothing had been decided at the time of this writing. I suggest you use firewall rules to restrict access to the clients, and if you have clients with different security levels (e.g., trusted internal clients and untrusted external clients), I advise using different storage pools for each.

Other Cool GlusterFS Tricks

Because you can add and remove storage bricks from a volume, you can add entirely new servers and remove existing servers. Thus, you can remove dying hardware from your storage pool with minimal interruptions. Because the back-end data storage is simply a filesystem like XFS or ext4, and the metadata is contained within the files, you can easily back up your GlusterFS data on the storage servers. You also can monitor the contents of files easily by, for example, scanning for viruses or rootkits without having to depend on the clients mounting and using the data.

GlusterFS can also provide data storage to OpenStack Swift, Glance, and Cinder clients, and it can provide Hadoop HDFS storage, among other capabilities. Because GlusterFS uses a plugin architecture, in the longer term, it will be possible for more capabilities to be added and for GlusterFS to be extended – for example, by adding file encryption or multitenancy support per volume.

The future of storage on Linux is bright. Several solutions (e.g., GlusterFS, Ceph) that are under very active development allow you to create distributed, replicated, and high-performance storage systems with the use of cheap hardware. If you aren't already using these, I suggest you investigate them. Not having to worry about disk or server failure bringing down your storage back end is a good thing indeed. ■■■

INFO

[1] GlusterFS download: *http://www.gluster.org/download/*

[2] A kernel change breaks GlusterFS: *http://lwn.net/Articles/544298/*