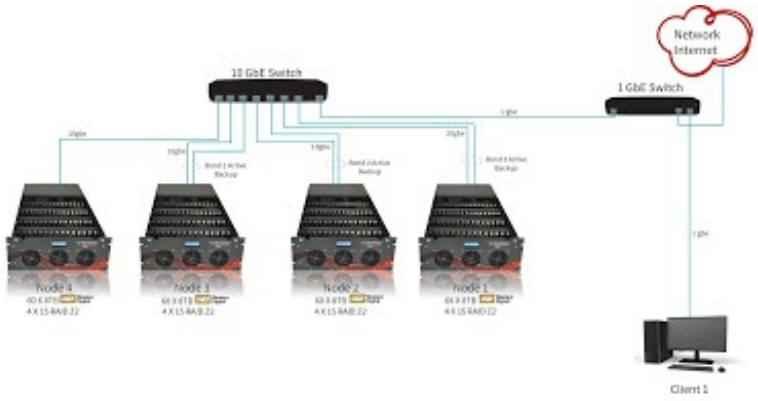
An Introduction to Clustering

45drives.blogspot.com/2016/11/an-introduction-to-clustering-how-to.html

How to build a Petabyte cluster using GlusterFS

with ZFS running on Storinator massive storage servers



If you checked out our recent video on clustering, you may have seen me keep a high-speed file transfer going, while I did just about everything I could think of that would give a system administrator nightmares. I was pulling out drives, yanking out network cables and hitting the power switch (virtually) on a server. This robustness is one of the key features of a storage cluster.

In this blog, I'm going to give you an overview of how we built that cluster, along with tons of detail on how I configured it. I'm hoping that when you see what I did, you'll see that clustering technology has improved and simplified to the point where even very small organizations can build affordable storage solutions that are incredibly robust and massively scalable.

Overview of Clustering

A cluster is a group of storage servers that work together as a single volume. Therefore, within an individual server, you can combine drives in various ways to create volumes. With clustering, you install and configure cluster software on top of each server, thus combining them to create a single storage space with larger aggregate capacity, or trading off some of your raw capacity for robustness via redundancy.

Here are the main elements of the cluster that I built:

Servers

I used four Storinator XL60 Enhanced massive storage servers (45drives.com). Each included:

- 64GB ECC RAM
- Intel E5-2620v4 processor
- 2 x R750 HBA hard drive interface cards
- X10-SRL Motherboard

Hard Drives and File System

The storage bays were filled with 60 8TB Western Digital Gold enterprise grade hard drives, giving each machine a raw capacity of almost half a petabyte!

It is important to note that when operating storage arrays of this magnitude, drive reliability becomes critical. Our experience has shown us that if you use low-reliability drives in large arrays in active storage applications, you will have a high level of aggravation. Issues range from major slowdowns from access retries on weak drives to frequent drive replacement and array rebuild. On the other hand, Western Digital Gold Drives, have proved to be rock-solid, and we have had zero complaints from users.

I set up my storage arrays with ZFS (I'm a huge fan!). ZFS has awesome features such as compression, snapshots and SSD caching with L2ARC (read caching) and ZIL (write caching). In service for many years, ZFS has proven to be highly robust with data loss almost unknown if implemented properly.

Cluster Software

I used GlusterFS as my cluster layer. This enabled scalability beyond one server, and let me create a high-availability, high-performance single storage volume.

Putting these elements together created an incredibly safe, reliable, large and really powerful storage volume. This allowed me to go on my 'rampage of destruction' in the video without interrupting the file transfer. Now let's get to the configuration specifics. I am going to break down the steps as follows:

1. ZFS configuration
2. Gluster volume configuration
3. Network setup
4. Mounting the volume

1. ZFS Configuration

Each node is configured in a 6 vdev x 10 drive RAIDZ2 configuration. This provides a usable capacity of 353TB per node. This configuration allows up to two disk failures per vdev, meaning up to 12 drives can fail without any data loss. Keep in mind, no more than two drives can go down per vdev.

I created four datasets on each storage pool to hold the "bricks" that Gluster will use to create the clustered volume. By default, no capacity quota is given to each dataset therefore, each data-sets capacity is reported the same as the parent zpool, 353TB. When the Gluster volume is created, the reported total capacity is four times larger than expected. *Pretty cool to see but not very helpful.*

To avoid this capacity confusion each brick was given a quota, the four of them add up to give the total capacity of the storage pool, so in my case 353 TB / four bricks equal 88TB.

I am currently experimenting with various brick counts and sizes. I started with the values above based on a document about GlusterFS recommendations. You can check out the document here.

There are many ZFS tweaks that could be applied to the zpool that will adjust various workloads. However, based on our own and others experiences with GlusterFS on top of ZFS I set the following tweaks by default:

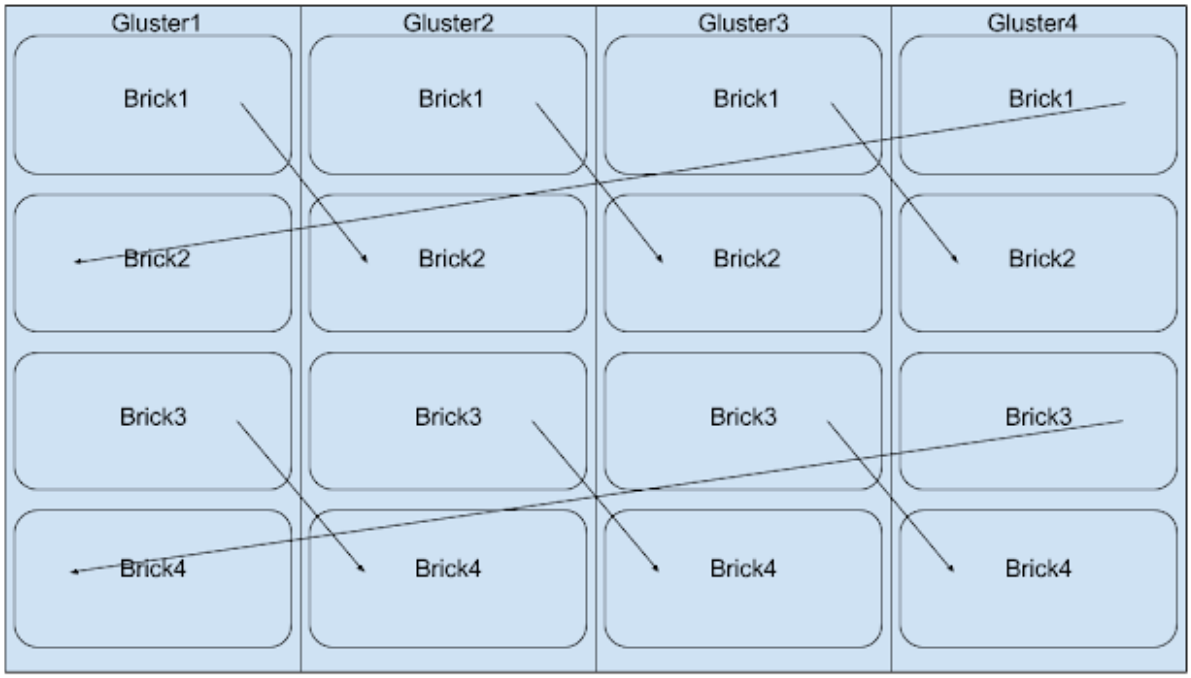
```
zfs set atime=off [volname]

zfs set xattr=sa [volname]

zfs set exec=off [volname]

zfs set sync=disabled [volname]
```

Once the pool was built, bricks were created, quotas and tweaks applied we were ready to build the Gluster volume. Which brings us to our second step.

2. Gluster Volume Configuration

I decided to build a distributed-replica volume with four bricks per node in a linked list configuration. The main benefit to a linked list topology is that you get replication. This ensures that if one node goes down the others in the cluster keep the volume available for the clients. In contrast to a traditional replicated volume where you would have to add two nodes at a time, this gives you the ability to expand your cluster one node at a time.

Below is an illustration showing the topology of a four node, four brick Gluster volume, the arrows indicate the replication pair of a brick. Gluster[1-4] are the host names of each node. This gluster configuration, paired with the ZFS pool mentioned above provides a usable capacity of 706TB. Using this configuration, should one node shutdown the other nodes will keep the volume up and running.



3. Network Setup

Each node uses network teaming in an "active-backup" configuration to provide fault tolerant interfaces. In our configuration, we are not using the two onboard 1gbe NICS, we are using the two 10gbe ports on an Intel X-540T2 addon card.

Therefore, each node has both 10gbe ports teamed together and plugged into an eight port Prosafe Netgear S708E 10gbe switch. Due to port restrictions, the fourth node only uses one of its 10gbe NICS. The remaining port on the switch is tied into our existing lab 1gbe network, for client access.

No matter the speed of your network, it is extremely helpful to have your Gluster nodes themselves connected to each other on a 10gbe backbone when using a replicated volume.

4. Mounting the Volume (Client Mounts)

Choosing a protocol, which enables clients to access the cluster data really depends on the application. There are a few options, here is a list of the main choices and what they offer:

Gluster Native Client

The Gluster Native Client (GNC) is a FUSE-based client running in user space. GNC is the recommended method for accessing volumes when high concurrency and high write performance is required. This is the mounting method we had used.

NFS

You can use NFSv3 to access Gluster volumes, and GlusterFS supports network lock manager (NLMv4). This allows NFSv3 client applications to do its own record locking of files on the NFS shared volume. GlusterFS also supports the newer "NFS-Ganesha", a user-level implementation of NFS server.

Samba

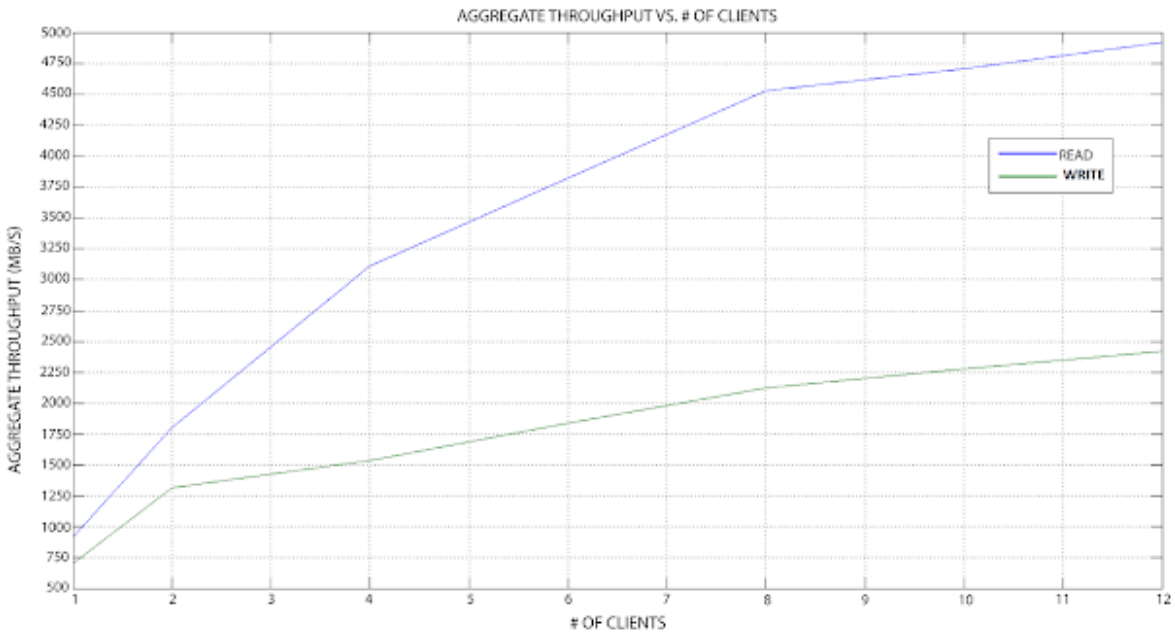
Samba exports are supported when using Windows or other SAMBA clients.

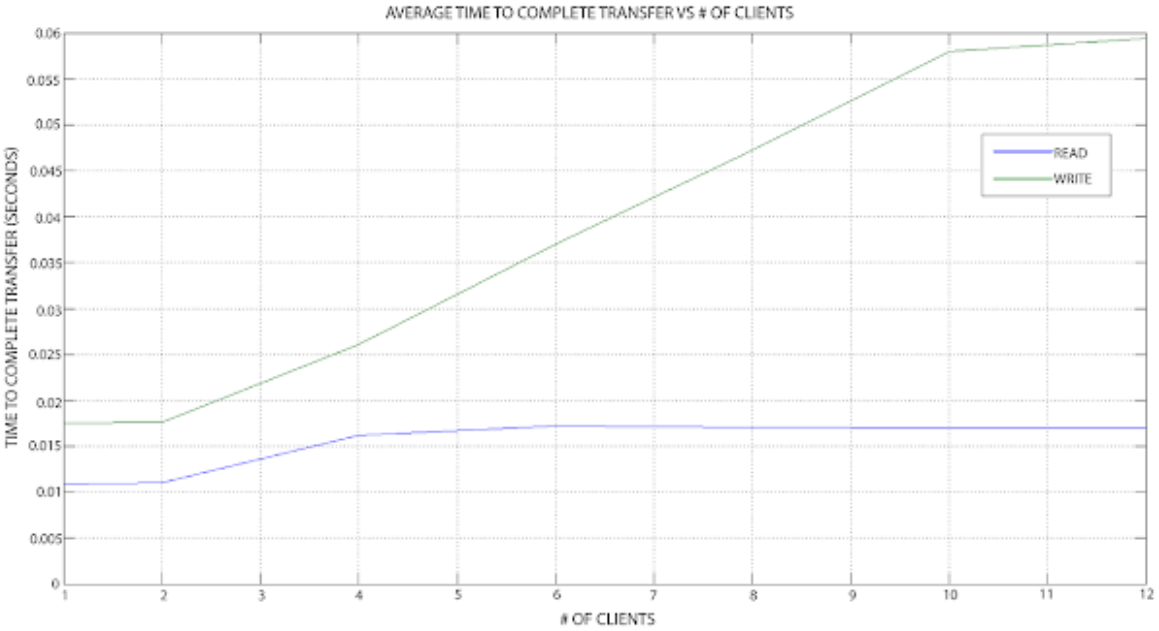
NOTE: NFS and Samba sharing: Unlike the native client, if a node goes down, the clients who are accessing that particular node will not be automatically reconnected to a different node in the cluster. While the volume is replicated and still online, when using traditional Samba or NFS shares, as far as your user is concerned the volume is not accessible at all. This basically defeats the purpose of the distributed replicated volume.

To solve the problem, developers from the Samba project have created a solution called Clustered Trivial Database (CTDB). CTDB is a simple clustering daemon that adds virtual IP addresses and a heartbeat service to each node, allowing automatic failover for clients accessing the share should one of the nodes go down. CTDB allows support for both NFS and Samba

Performance Measurements

- One graph shows the aggregate throughput seen by all the clients, and the other shows the average time to complete the file transfer across all clients.
- Each client executed a script in parallel that wrote 10,000 files of 10MB at a block size of 64kB.
- The results collected were graphed using MatLab.





Based on the results found in the lab, this configuration has the following performance stats:

Maximum Read aggregate throughput of 5000MB/s (Aggregate)
Maximum Write aggregate throughput of 2500MB/s
Time to complete reading files plateau at 17ms
Time to complete writing files plateau at 58ms

NOTE: Keep in mind all nodes are on the same 10gbe backbone

Conclusion

Previously, clustering was difficult, finicky, and expensive. However, low-cost, massive open storage platforms and rock solid servers and file systems (such as CentOS Linux and ZFS) grouped with powerful, reliable, and easy to configure clustering software (such as GlusterFS) have changed all that. It is now reasonable for small and medium-sized organizations to take advantage of clustering and create ultra reliable, fast and cost-effective storage systems.