GlusterFS: Changing bricks without downtime or data loss

140139

16. november 2014

Sammendrag

SLA

Innhold

1 Introduction

GlusterFS is an open source distributed file system with huge a scalability infrastructure, allowing up to 72 brontobytes of data [2]. Gluster volumes are a collection of bricks, it is where most of the configurations happen. A brick is an export directory inside a server, it is the basic storage unit [1]. Inside a volume, the brick is defined by 'server:/path_to_brick'.

With this project I aim to show how to replace and expand a GlusterFS volume without having downtime and avoiding slowness of the system. The scenario where the experiments are running consists of 4 virtual machines running Ubuntu 14.04, 2 Gigabytes of RAM and 2 single core CPUs at 2,1 GHz. Each of these machines got two extra block devices that are utilized as bricks, each one having 50 Gigabytes.

2 SLA/RECSPEC

By doing this project I wish to learn as much as possible about GlusterFS and file systems in general. I see this as an opportunity to acquire practical knowledge, in a real scenario that otherwise would be very hard to get. Also, besides helping me to develop myself as an capable system administrator, this project will have direct positive effect on the university infrastructure, by expanding GlusterFS effective storage space and proving its scalabillity.

To accomplish this, I will start by simulating the real scenario on virtual machines and learning how GlusterFS works. I intend to repeat this process as many times as needed to get a very good understanding of the brick substitution process.

3 Survey of similar projects

4 Description of your experiment

Before doing any Gluster related operation, it was necessary to create a compatible file system in every brick. The recommended file system by the Gluster team is XFS as it is specially good at parallel operations, due to its design [3]. Also, the recommended inode size is 512. This step was done by running the following commands in each of the nodes.

```
#mkfs.xfs -i size=512 /dev/vdb
#mount /dev/vdb /mnt/brick0/
#mkfs.xfs -i size=512 /dev/vdc
#mount /dev/vdc /mnt/brick1/
```

The next step consisted on creating the Gluster volume, called CinderSAN, and starting it, from any Gluster server of choice. For this project, all the possible gluster operations were done from *node01*.

```
#gluster volume create CinderSAN replica 2 \
node01:/mnt/brick0/brick node02:/mnt/brick0/brick \
node03:/mnt/brick0/brick node04:/mnt/brick0/brick
```

```
#gluster volume start CinderSAN
```

The option *replica 2* means it will replicate any written data to two disks. After having the volume created and started without errors, the next step is to mount it from the client.

```
mount -t glustefs node01:/CinderSAN /mnt/gluster/gluster0/
```

5 Results and discussion

6 Security aspects

7 Conclusions

Referanser

[1] GLUSTERFS. Glossary - gluster, 2014. [Online; accessed 11-November-2014]. 1

[2] GLUSTERFS. Gluster, 2014. [Online; accessed 11-November-2014]. 1

[3] SGI. Xfs: A high-performance journaling filesystem, 2012. [Online; accessed 16-November-2014]. 1