GLUSTERFS

PROJET DE SEMESTRE

Étudiant : Khaled BASBOUS

Professeur : Gérald LITZISTORF

HEPIA

15 avril 2011

GlusterFS est un logiciel open source distribué sous licence GPLv3. Ce logiciel implémente un système de fichiers distribué, répliqué, configurable capable de monter en capacité jusqu'à plusieurs pétaoctets sur du matériels usuels. GlusterFS est basé sur une architecture client-serveur où chaque serveur est une " brique de stockage ". Un volume de stockage est formé par l'ensemble de ces briques mis à disposition à travers un réseau. Les postes clients peuvent se connecter au cluster de stockage, à l'aide d'un client GlusterFS où par d'autres protocoles tel que CIFS, FTP pour accéder à l'ensemble de l'espace de stockage. Dans ce projet, j'ai commencé par étudier le fonctionnement de ce logiciel. Puis j'ai défini une méthodologie de benchmark pour vérifier les performances linéaires promise par GlusterFS et par la même occasion détecter les éléments limiteurs de ces performances.

CONTENU

1 ÉNONCÉ[1]

Dans le cadre de ce travail de semestre, j'ai étudié la solution GlusterFS, technologie innovante rachetée par RedHat pour la somme de 136 Millions de dollar américain, en octobre 2011. RedHat a acquis ce système pour l'intégrer à leur solution de cloud computing.

L'étude a porté, dans un premier temps, sur l'identification des principaux avantages de la solution GlusterFS. Puis dans un deuxième temps de comprendre son fonctionnement afin de préciser les principaux paramètres de configuration qui permettront de couvrir quelques cas de défaillance. À la fin de cette étude, je vais établir des scénarios de tests et j'effectuerai des mesures de performances.

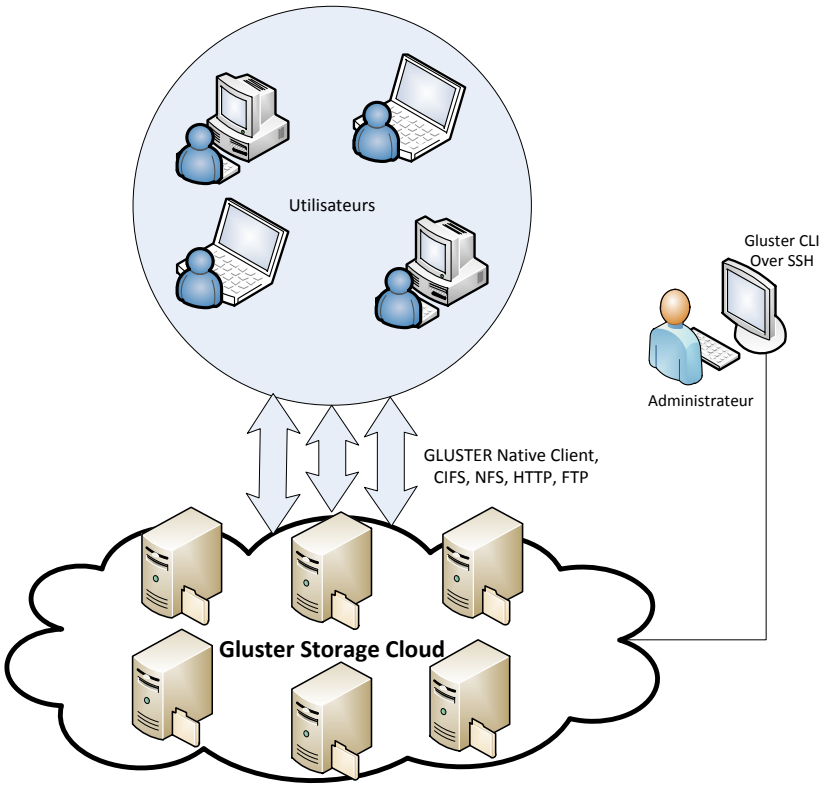

Figure 1 : Clients - Serveurs – Administrateurs

Dans cette figure, nous pouvons voir une ferme de serveurs GlusterFS appelée trusted pool ou cluster de stockage qui exportent un ou plusieurs volumes. Les clients qui souhaitent accéder au volume de stockage peuvent s'y connecter à travers le réseau à l'aide d'un logiciel client GlusterFS, Samba ou autres. Le client natif de Gluster est l'outil privilégié pour accéder au stockage. L'administrateur peut configurer cette ferme de serveurs en utilisant une connexion sécurisée SSH en ligne de commande ou bien en s'identifiant en tant que « root » sur un des serveurs GlusterFS.

[1] Lien pour l'énoncé officiel : http://www.tdeig.ch/kvm/Basbous_EPS.pdf

2 ANALYSE

Gluster simplifie fortement la sauvegarde et la gestion d'une grande quantité de donnée sur du matériels usuels. GlusterFS est un système de fichier distribué qui réunit l'espace disque, mis à disposition, de plusieurs serveurs en un pool de stockage.

Les fonctionnalités de Gluster sont nombreuses. Dans ce rapport je vais m'intéresser surtout aux points ci-dessous :

- Performance linéaire
- Espace de stockage capable de croitre jusqu'à plusieurs pétaoctets
- Haute disponibilité des données

Pour une liste plus complète des fonctionnalités offertes, veuillez-vous référer à l'annexe de ce document.

Les performances linéaires ne sont pas chose facile à réaliser. Avec les clusters de stockage, plus l'espace de stockage et le nombre de fichiers croit, plus les performances vont se dégrader. En effet, les clusters de stockages utilisent un serveur de métadonnées. Le nombre de métadonnées devient tel que cluster de stockage perd beaucoup en termes de performance à rechercher l'information concernant un fichier et à la modifier. Le serveur de métadonnées est un goulot d'étranglement au vu du nombre de nœud client qui veulent accéder à ce serveur. GlusterFS, lui ne gère pas les métadonnées, l'emplacement du fichier stocké sur le cluster est calculé à l'aide d'un algorithme nommé Elastic Hash. Donc, il n'y a pas de serveur de métadonnées. Les nœuds clients vont pouvoir calculer l'emplacement d'un fichier. Gluster réalise avec ce mécanisme une répartition de charge de calcul et par la même occasion supprime le goulot d'étranglement et le remplace par un calcul parallèle réalisé par les noeuds clients.

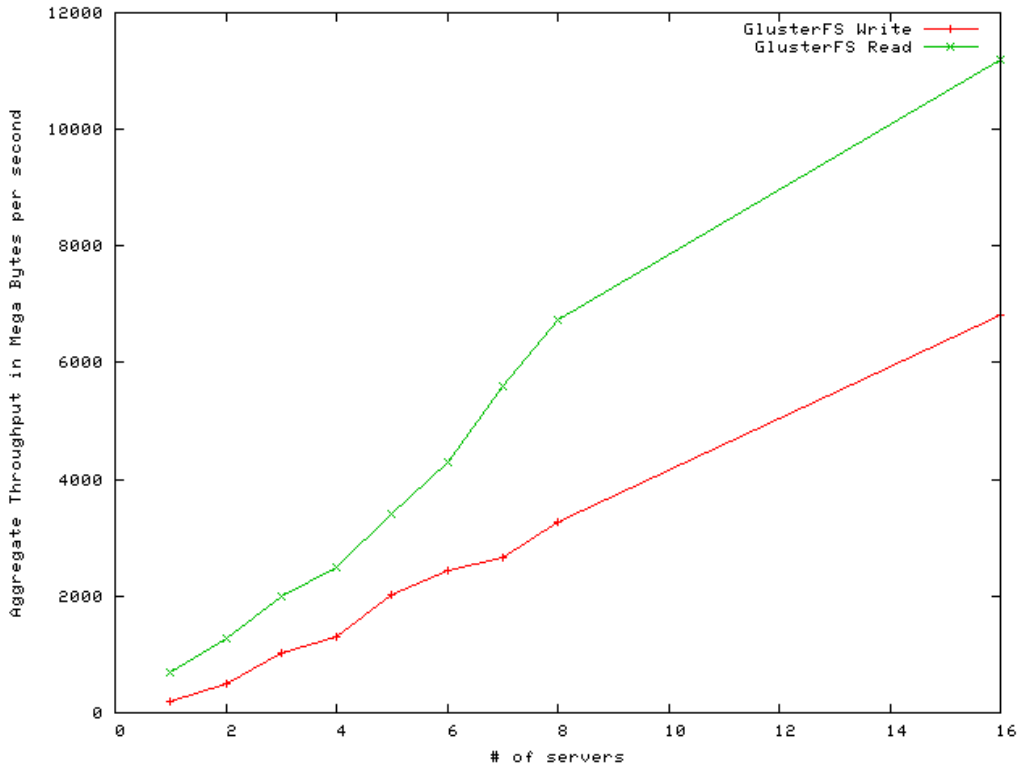

Figure 2 [2]- Performance GlusterFS avec 64 clients et en variant le nombre de serveur

[2]Figure extraite du site web GlusterFS

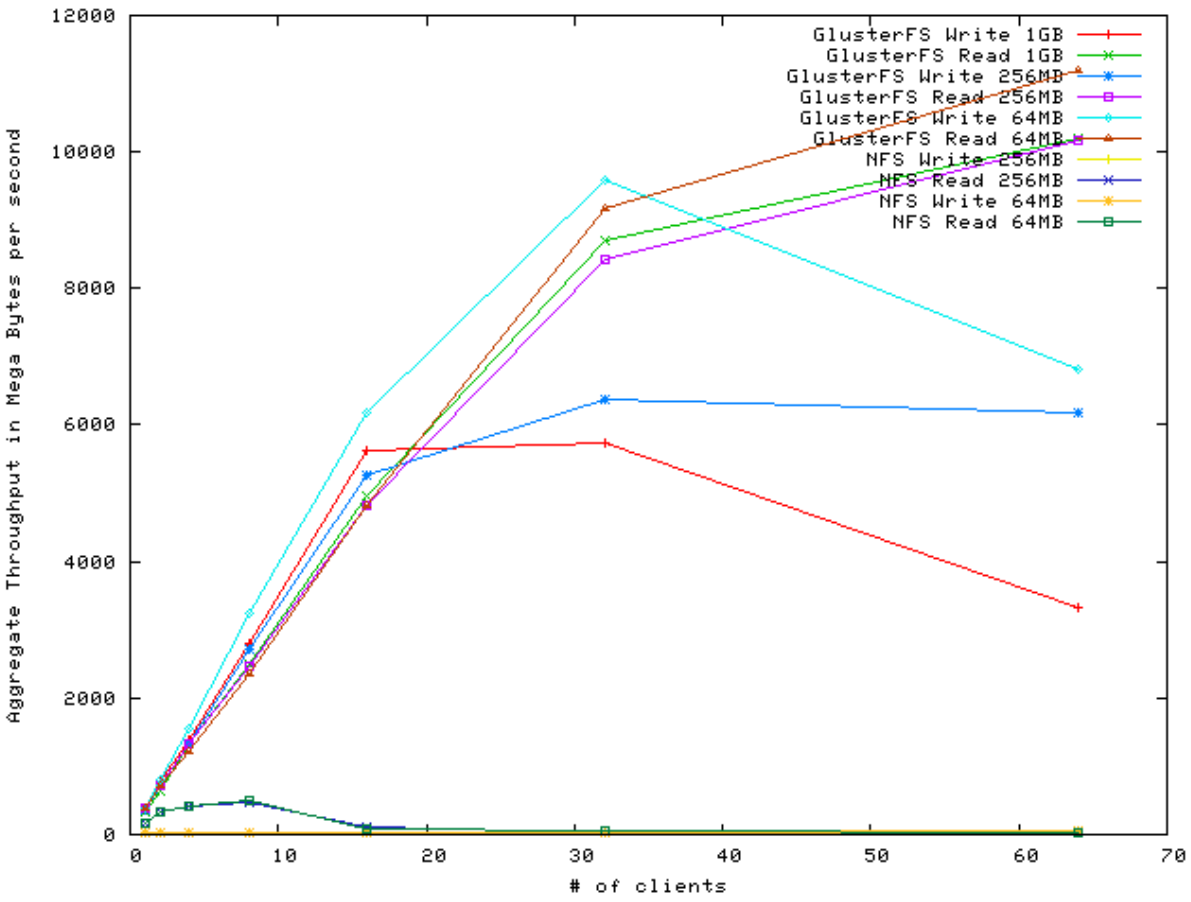

Figure 3 [3]- GlusterFS avec 16 briques Vs NFS

Dans la figure ci-dessus je m'intéresse aux données sur les serveurs GlusterFS. Nous constatons qu'à partir de 30 clients la pente des performances global du serveur GlusterFS diminue et peux même devenir négative avec l'utilisation de 16 briques. Ici, nous pouvons relever un problème de dimensionnement. Dans un environnement de production, il faut adapter au mieux le nombre de nœuds serveurs au nombre de nœuds clients.

L'architecture scale-out de GlusterFS nous permet de rajouter de l'espace disque en agrégeant des briques supplémentaire à un volume à chaud et sans interruption de service. Ainsi l'évolutivité de cette architecture est extrêmement intéressante.

La haute disponibilité des données peuvent être géré en réalisant des briques miroirs distribués sur plusieurs nœuds serveur. Si une panne logicielle ou matérielle survient sur un nœud, la brique miroir prendra le relai automatiquement et sans aucune intervention de l'administrateur.

Dans ce projet nous allons déployer GlusterFS sur une distribution Linux 64 bit, Fedora 16 en l'occurrence. GlusterFS va tourner en tant que tâche service, daemon, sur chacun des serveurs. Pour des raisons de simplicité, nous allons utiliser des disques-durs SATA sans la technologie RAID.

[3]Figure extraite du site web GlusterFS :
http://www.gluster.org/community/documentation/index.php/GlusterFS_1.2.1-BENKI_Aggregated_I/O_vs_NFSv4_Benchmark

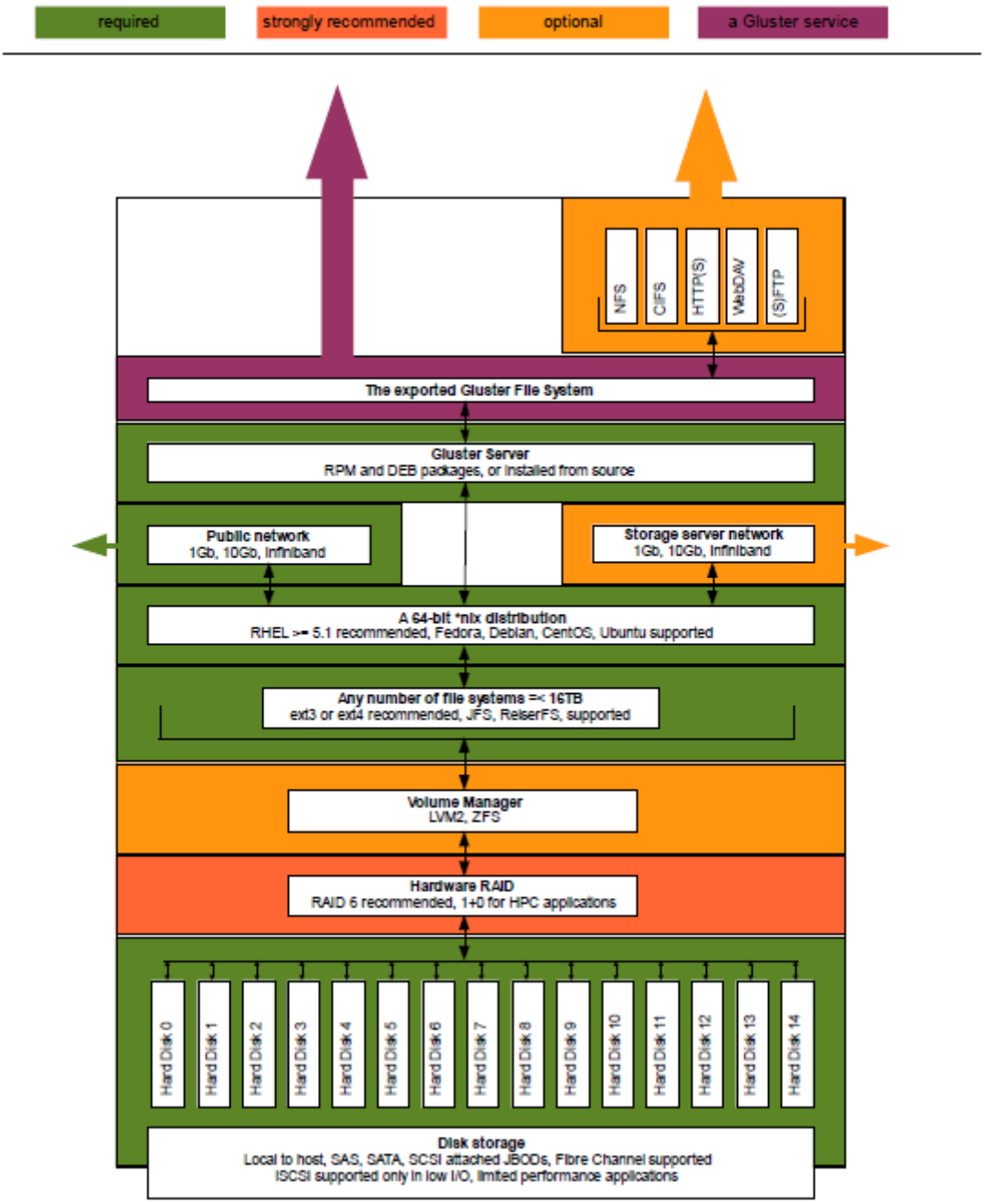

Figure 8 [4] *- Serveur GlusterFS*

Dans la figure, nous pouvons voir la vision de GlusterFS sur ce que pourrait être un serveur Gluster. Les couleurs nous donnent une idée de ce qui est requis, recommandée ou bien optionnel. Nous pouvons avoir un ordinateur avec plusieurs disques durs locale ou distant à celui-ci. Ces disques durs peuvent être gérés par un contrôleur RAID, pour réduire le risque de pertes de donnée et/ou pour augmenter la performance de lecture et écriture. Cet ordinateur est branché sur un réseau de stockage ou bien sur un réseau à multiple usage. Sur ce matériel est

[4] Figure extraite du document « introduction to GlusterFS » : http://www.google.ch/url?sa=t&rct=j&q=glusterfs%20introduction&source=web&cd=1&sqi=2&ved=0CCQQFjAA&url=http%3A%2F%2Fwww.think88.com%2FExamples%2FIntroduction_to_Gluster.pdf&ei=1VPuTpapOcPd4QTVjZGJCQ&usg=AFQjCNHdFD89OZkdaT0j9649Ys7IAyUKgA

installé un système d'exploitation Linux 64 bit. Le service Gluster serveur tourne sur cet OS linux afin d'exporter un système de fichier Gluster.

Pour simplifier la figure ci-dessus, je vais prendre l'exemple d'un serveur qui ne contient que l'essentiel pour faire tourner le daemon GlusterFS server. Dans la suite de ce rapport la figure de référence pour représenter un serveur est celle-ci.

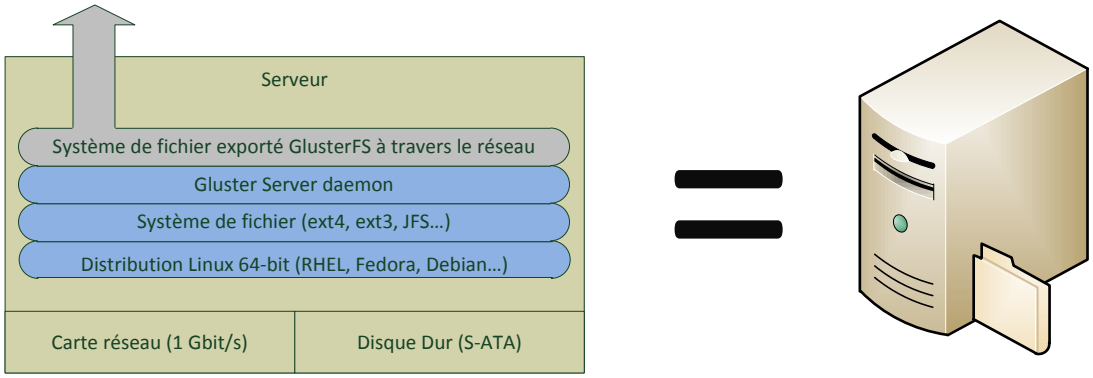

Figure 9 - Serveur simplifié

Dans la figure ci-dessus nous pouvons voir un ordinateur, qui dispose d'un disque dur et d'une carte réseau, capable de faire tourner une distribution Linux 64bit. Le disque dur est formaté par un système de fichier compatible linux. Sur celui-ci nous pouvons exécuter Gluster Server daemon pour partager une partie du disque dur, à travers le réseau sous forme d'une brique. Dans la figure, le PC avec un dossier à son coté est dans ce rapport considéré comme un serveur GluterFS. Cette représentation d'un serveur GluterFS a été utilisée en début de ce rapport. Dorénavant nous avons une idée plus précise de quoi est constitué un serveur de fichier GlusterFS avec les informations données tant au niveau hardware qu'au niveau software.

Un pool de stockage, ou un cluster de stockage est construit à l'aide d'un ensemble d'ordinateurs liés par un réseau afin de regrouper leurs espaces de stockages. Ceci permet d'exploiter une partie du disque dur et de la puissance de calcul de chacun d'entre eux afin de fournir un service en parallèle à leurs fonctions de base.

Pour construire un cluster de stockage, il faut exécuter Gluster File System Server sur chacun des ordinateurs que nous souhaitons lier au cluster pour partager un espace disque. Ensuite il faut créer un volume. Un volume GlusterFS est constitué de briques de stockage mis à disposition par chacun des serveurs. Une brique de stockage est un répertoire sur le serveur associé avec un volume. Les clients, ou le client du cluster va pouvoir monter un volume GlusterFS dans un répertoire pour qu'il puisse stocker ces données comme si c'était un espace de stockage local, un volume local à la machine.

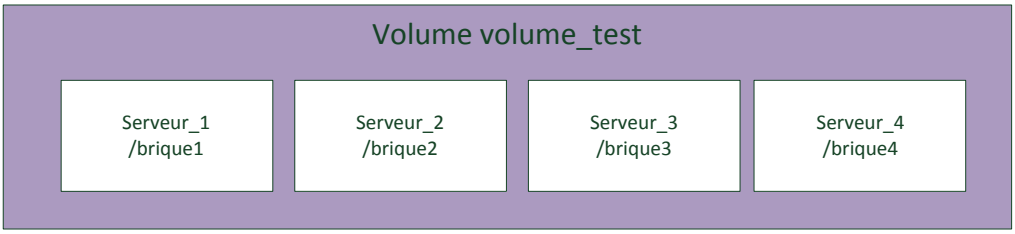

Figure 4 - Volume GlusterFS

Dans cette figure, un volume nommé volume_test est constitué de quatre briques, chaque brique est placée sur un serveur différent.
Le client natif GlusterFS peut à travers des traducteurs distribuer, répliquer ou découper les fichiers à stocker sur le volume distribué distant. Un traducteur est un code qui permet de faire un traitement et offre la possibilité de se connecter à une ou plusieurs briques d'un volume. Prenons l'exemple d'un volume distribué formé de trois briques. Chaque brique est stockée sur un serveur. Un client monte le volume dans un dossier local pour pouvoir y stocker ces fichiers et dossiers. Lors du montage le client natif va communiquer avec un serveur pour récupérer les propriétés du volume à monter (adresses des trois nœuds serveurs, nombre de briques, mode du volume distribué, ...). Une fois le volume monté le client peut y stocker par exemple un fichier.

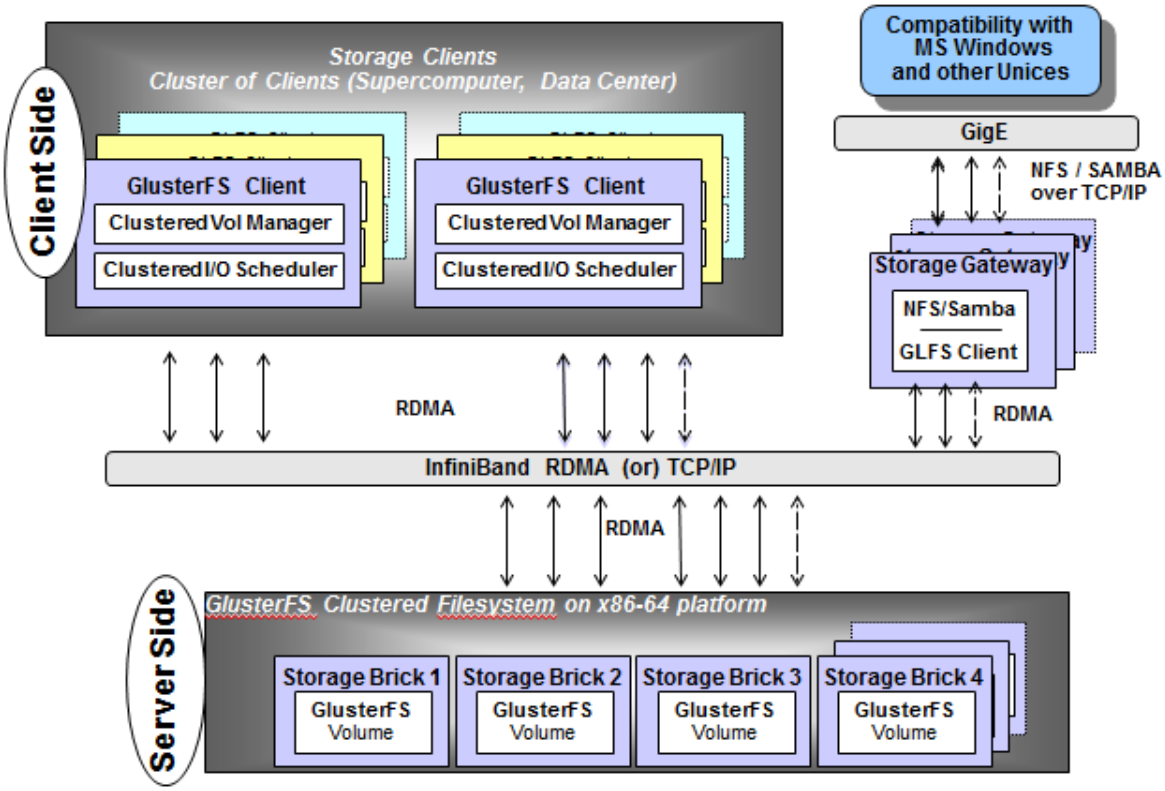

Figure 5 - Vue d'ensemble[5]

Dans cette figure nous avons le regroupement du coté client avec le coté serveur. À droite nous avons une représentation d'un client spécial. Celui d'un client qui exécute en natif GlusterFs client et exécute un autre logiciel qui va permettre d'exporter ce volume à l'aide d'autre protocoles en tant que serveur. Par exemple Samba qui est un protocole compatible windows et qui va permettre aux utilisateurs windows d'accéder à cette espace de stockage, au volume

[5] Figure extraite du document « introduction to GlusterFS » :
http://download.gluster.com/pub/gluster/glusterfs/talks/Z/GlusterFS.odp

partagé. Attention, la comptabilité avec d'autres protocoles à un prix. En effet, le client qui joue le rôle de passerelle entre GlusterFS et d'autre protocole peut être débordé au vu du nombre de clients qui se connecte à celui-ci et ainsi représenter un goulot d'étranglement. Dans ce projet, je m'intéresse uniquement au client natif GlusterFS.

Pour mieux illustrer ceci, je vais donner un exemple de comment GlusterFS stocke les informations liées à un volume distribué crée sur 4 serveurs. Les informations ci-dessous sont obtenues à l'aide de la commande gluster volume info exécuté sur l'un des serveurs du volume :

```
# gluster volume info
Volume Name: volume_test
Type: Distribute
Status: Created
Number of Bricks: 4
Transport-type: tcp
Bricks:
Brick1: Serveur_1:/brique1
Brick2: Serveur_2:/brique2
Brick3: Serveur_3:/brique3
Brick4: Serveur_4:/brique4
```

Donc nous voyons ci-dessus que le volume est de type distribué et est composé de quatre briques chacune provenant d'un des serveurs du cluster. Le transport des données stockées se fait à l'aide de paquets TCP sur le réseau.

Par exemple lorsque nous avons créé un volume distribué, la distribution des fichiers sur les briques du cluster se fait à l'aide d'un traducteur de distribution. La taille d'un fichier maximum est plus petite ou égale que la taille d'une brique. Pour limité la taille d'une brique de stockage, nous pouvons crée un volume logique d'une taille limité sur le serveur qui contient cette brique. Différents outils peuvent être utilisés pour crée ce volume local tel que LVM ou bien l'outil de gestion de Gluster. Pour limiter la taille d'un volume ou d'un dossier dans le volume il y a la possibilité de mettre des quotas qui sont implémenté aussi par des traducteurs GlusterFS.

Le serveur qui va contenir le fichier est déterminé à l'aide du hachage du nom du fichier à stocker. Si le nom du fichier change, un pointeur sur l'ancien emplacement du fichier est stocké sur le serveur pointé par le hash du nouveau nom. Ceci est fait pour éviter de déplacé le fichier à chaque changement de nom, surtout si le fichier est volumineux. Dans le cas où un fichier n'est pas présent à l'emplacement où l'algorithme de hash indique, alors une requête de recherche est lancée induisant un temps de latence.

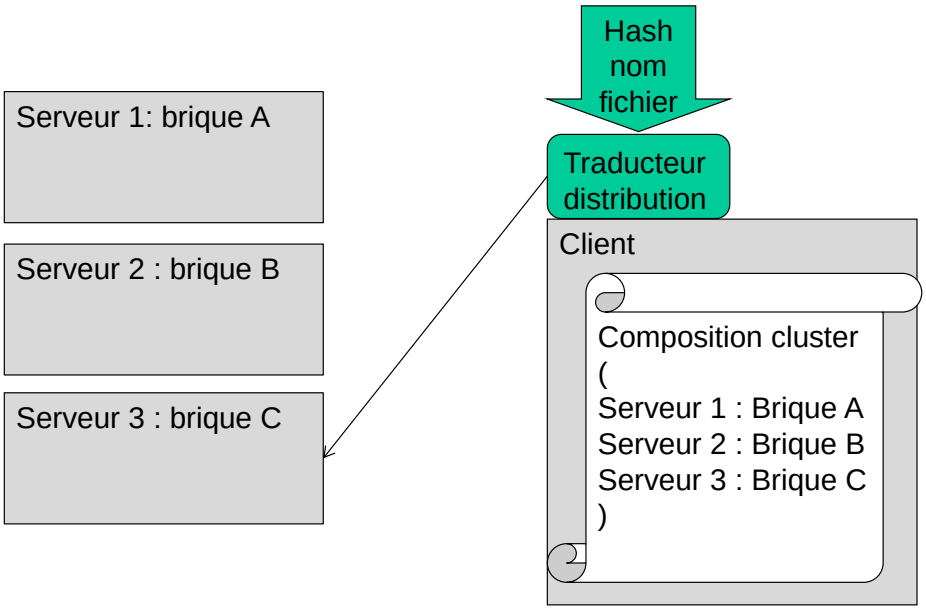

Figure 7 – Distribution fichiers

La distribution est une technique qui ne garantit pas l'accès à l'ensemble des fichiers stockés dans le cas d'une défaillance d'un des serveurs. Pour résoudre ce problème de disponibilité, il faut utiliser le traducteur de réplication au prix d'une performance d'écriture moindre. Plus on ajoute de serveurs, plus nous augmentons les performances de notre cluster en terme d'accès aléatoire aux fichiers.

Un disque dur S-ATA est compatible GlusterFS. Les performances de GlusterFS sont liées non seulement à la vitesse d'écriture/de lecture des disques durs mais elles sont aussi liées au débit du réseau et à la puissance des serveurs. La puissance du serveur n'intervient pas fortement, mais peu affecté les performances dans le cas où nous déployons GlusterFS serveur sur un ordinateur fortement sollicité ou sur un ordinateur qui ne dispose pas d'une puissance suffisante pour faire tourner une distribution linux convenablement. Si une connexion 1 GB sur ethernet est insuffisante, GlusterFS supporte aussi les connexions 10 GB sur ethernet et les connexions allant jusqu'à 40 GB sur InfiniBand.

Comme dans RAID 0, nous pouvons configurer un volume agrégé par bandes (striped volume). C'est-à-dire que lors du stockage d'un fichier, celui-ci sera découpé en bandes qui seront stockées sur les n briques de stockage. Pour avoir des bonnes performances avec cette configuration, nous ne devrons l'utiliser que dans un environnement où l'accès aux fichiers est hautement concurrent avec de gros fichier. Prenons l'exemple d'un système de distribution de vidéo à la demande. Les films les plus récents sont vus par beaucoup de clients en même temps, la configuration d'un striped volume sera bénéfique dans ce cas.

Il est important de noter que les clients natifs GlusterFS ne communiquent pas entre eux mais communique uniquement avec les serveurs GlusterFS. GlusterFS utilise son propre protocole de communication qui est en unicast uniquement. Voici une capture Wireshark que j'ai réalisé en

créant un fichier sur un volume GlusterFS. Nous voyons que le client qui envoie une requête « CREATE CALL » pour demander au serveur de crée un fichier. Le serveur répond au client en acquittant la création du fichier avec la requête « CREATE REPLY ». Puis il y a le transfert des données du fichier envoyé par le client vers le serveur. Et une terminaison de l'opération par un « REALEASE REPLY ».

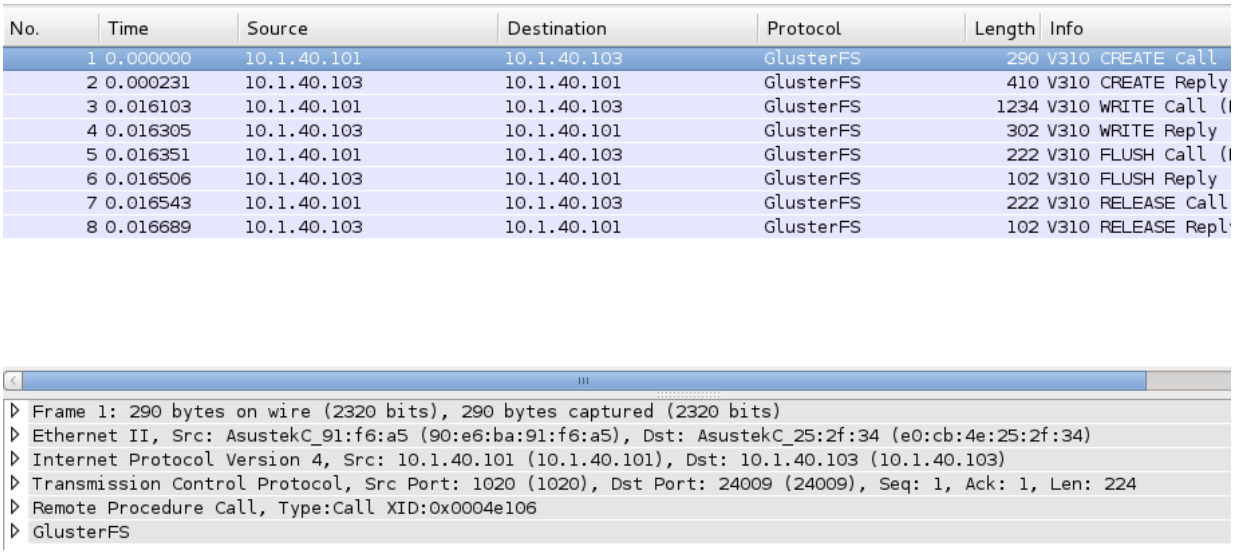

Figure 6 - Capture Wireshark sur un volume distribué

3 RÉALISATION

3.1 HARDWARE

SERVEURS & CLIENTS[6]

- Carte mère ASUS
- Processeur Core2Duo 3 Ghz, 6Mb de cache, FSB de 1333Mhz
- Disque dur de 320GB en S-ATA
- RAM (2x2GB en DDR800)

RÉSEAUX

- Switch Extreme Networks Summit 400-48t – 48 ports x 1Gbps
- Switch Netgear 5 ports x 1 Gbps

3.2 SOFTWARE

DISTRIBUTION LINUX

- Fedora 16 64bit

GLUSTERFS

- Glusterfs v3.2.4
- FUSE v2.8.6

OUTILS DE MESURES

- Wireshark v1.6.5-1.fc16.0.gluster.x86_64[7]
- dd
- df
- du
- time
- IOzone
- Fileop
- Iostat
- Iperf

[6] PC Asus monté par José Tavares pour l'étude de VMware ESXi : http://www.tdeig.ch/vmware/Montage_PC_Gigabyte.pdf

[7] http://repos.fedorapeople.org/repos/devos/wireshark-gluster/fedora-16/

4 TESTES

4.1 MÉTHODOLOGIE DE MESURE

Je vais d'abord procéder à la mesure des performances du disque dur en local à l'aide du logiciel IOzone. Cet outil va me permettre de générer et de mesurer plusieurs opérations sur l'espace de stockage d'un volume local ou distant. Ce premier test va me permettre d'avoir une idée précise des performances de mes disques durs utilisés durant le projet.

Dans un deuxième temps, je vais mesurer le débit réel de mon réseau en utilisant la commande iperf du monde linux. Le débit de mon réseau est l'élément principal qui va limiter le débit global de mon ou mes volumes distants. La commande iperf est de type client/serveur. Donc j'aurai besoin de lancer la commande iperf client sur un PC et la commande iperf serveur sur un autre PC relié sur mon réseau. Pour vérifier que le switch que j'utilise gère bien le débit Gigatbit sur chaque interface. Pour ce faire je lancerai ce test sur un nombre pair d'interfaces du switch en même temps.

Au final je vais mettre en place des volumes distribués ou des volumes répliqués GlusterFS. J'exécuterai ensuite les mêmes testes IOzone que j'ai lancé avant sur le disque local. Ce dernier point est la finalité de mes tests préalables. Je relèverai aussi l'utilisation des ressources RAM et CPU des serveurs durant les activités d'écriture et de lecture. En effet, comme GlusterFS est un système qui est lancé sur n ordinateurs qui peuvent être aussi utilisé en parallèle pour d'autres fonctions, ces données peuvent s'avérer très utiles pour des questions des dimensionnements des PC serveurs. Une autre mesure intéressante est celui du débit transférer sur le réseau par rapport à la taille réelle du fichier.

4.2 PERFORMANCE DU DIQUE DUR EN LOCAL

VITESSE DE LECTURE

Test du débit de lecture séquentielle en local. Avant tout je vide le cache avec la commande :

```
# echo 3 > /proc/sys/vm/drop_caches
```

Lecture du d'un fichier de 1 GB :

```
# time dd if=/home/labotd/lecture_1GBX1 of=/dev/null bs=1000000000
count=1
```

Résultat :

```
1000000000 octets (1,0 GB) copiés, 10,099 s, 99,0 MB/s
```

VITESSE D'ÉCRITURE

Test du débit d'écriture séquentielle d'un fichier de 1 GB en local :

```
# time dd if=/dev/zero of=/home/labotd/ecriture_1GbX1  bs=1000000000
count=1
```

Résultat :

```
1000000000 octets (1,0 GB) copiés, 10,3226 s, 96,9 MB/s
```

RÉSULTAT IOZONE

```
./iozone -i 0 -i 1 -i 2 -s 4g -Rb /home/resultat_4g.xls -f /file4g
```

Taille fichier [GB]	Écriture [MB/sec]	Réécriture [MB/sec]	Lecture [MB/sec]	Relecture [MB/sec]	Lecture aléatoire [MB/sec]	Écriture aléatoire [MB/sec]
4	107.845	108.015	116.574	119.423	12.248	24.896

Après les tests de performance de mon disque local avec le logiciel IOzone, je relève que le disque dur est capable de lire et écrire à un débit de 100 MB/s. La lecture et l'écriture aléatoire est presque quatre fois plus lente qu'une lecture/écriture séquentielle.

4.3 PERFORMANCE DU RÉSEAU

SWITCH NETGEAR

Résultats obtenu avec la commande iPerf exécuté deux fois en mode serveur sur deux machines et deux fois en mode client sur deux autres machines. Toutes les machines sont connectées sur les différents ports du switch Netgear :

Serveurs 1 :

0.0-10.0 sec 1.09 GBytes 937 Mbits/sec

Serveurs 2 :

0.0-10.0 sec 1.09 GBytes 939 Mbits/sec

Les résultats sont très proches du débit maximum de chaque port du switch Netgear.

4.4 PERFORMANCE DU CLUSTER

ARCHITECTURE DISTRIBUÉE

UN SERVEUR, UNE BRIQUE, UN CLIENT

J'ai installé GlusterFS au préalable sur le PC que je souhaite utilisé en tant que serveur GlusterFS. Je lui ai assigné l'ip 192.168.1.10 Voici le code du déploiement de l'architecture :

```
# ifconfig eth0 192.168.1.10

# glusterd start

# mkdir /gluster/brique1

# gluster volume create volume_une_brique
192.168.1.10:/gluster/brique1

# gluster volume start volume_une_brique
```

Montage du volume sur un pc autre que le pc serveur. Il faut assigner une IP. Et crée un dossier destiné à accueillir le volume GlusterFS volume_une_brique. Le code ci-dessous est exécuté dans la console du pc client.

```
# mkdir /mnt/glusterfs

# mount -t glusterfs 192.168.1.10:/volume_une_brique /mnt/glusterfs
```

Test du débit de lecture séquentielle :

Je vide le cache du client avec la ligne de commande qui suit :

```
# echo 3 > /proc/sys/vm/drop_caches
```

Je fais une lecture séquentielle avec la commande dd et je la chronomètre avec la commande time :

```
# time dd if=/mnt/glusterfs/lecture_1gbX1 of=/dev/null bs=1000000000
count=1
```

Résultat :

```
1000000000 octets (1,0 GB) copiés, 9,65578 s, 104 MB/s
```

Test du débit d'écriture séquentielle :

```
time dd if=/dev/zero of=/mnt/glusterfs/lecture_1gbX1 bs=1000000000
count=1
```

Résultat :

```
1000000000 octets (1,0 GB) copiés, 9,538 s, 105 MB/s
```

```
Résultat IOZone :

./iozone -i 0 -i 1 -i 2 -s 1g -Rb /gluster/test_s1Gb_i012 -f
/gluster/2S2B/C
```

Taille fichier [GB]	**Écriture [MB/sec]**	**Réécriture [MB/sec]**	**Lecture [MB/sec]**	**Relecture [MB/sec]**	**Lecture aléatoire [MB/sec]**	**Écriture aléatoire [MB/sec]**
1	68.987	68.901	105.031	105.041	7.988	16.003

Avec la mise en place d'une brique nous pouvons comparer ces valeurs avec les valeurs obtenues pour la lecture sur le disque dur en local. Les performances d'écriture est passé de 108 MB/s en écriture locale à 69 MB/s en écriture distante. Alors que la lecture de fichier a perdu en débit moins que l'écriture. La lecture est passée de 116 MB/s à 105 MB/s. La lecture aléatoire a baissée de 4 MB/s et l'écriture de 8 MB/s.

DEUX SERVEUR, DEUX BRIQUE, DEUX CLIENT

J'ai assigné l'ip 192.168.1.10 au serveur 1 et l'ip 192.168.1.11 au serveur 2. Voici le code du déploiement de l'architecture :

Sur serveur 1 :

```
# ifconfig eth0 192.168.1.10

# glusterd start

# mkdir /gluster/brique1
```

Sur serveur 2 :

```
# ifconfig eth0 192.168.1.11

# glusterd start

# mkdir /gluster/brique2
```

Sur serveur 1 :

```
# gluster peer probe 192.168.1.11

# gluster volume create volume_deux_brique
192.168.1.10:/gluster/brique1 192.168.1.11:/gluster/brique2

# gluster volume start volume_deux_brique
```

Montage du volume sur un pc autre que les pc serveurs. Il faut assigner une IP. Et crée un dossier destiné à accueillir le volume GlusterFS volume_deux_brique. Le code ci-dessous est exécuté dans la console du pc client.

```
# mkdir /mnt/glusterfs

# mount -t glusterfs 192.168.1.10:/volume_deux_brique /mnt/glusterfs
```

Client 1 :

echo 3 > /proc/sys/vm/drop_caches

time dd if=/gluster/glusterMount/B of=/dev/null bs=1000000000 count=1

1+0 enregistrements lus

1+0 enregistrements écrits

1000000000 octets (1,0 GB) copiés, 11,5191 s, 86,8 MB/s

real 0m11.642s

user 0m0.000s

sys 0m0.689s

Client 2

echo 3 > /proc/sys/vm/drop_caches

time dd if=/gluster/glusterMount/A of=/dev/null bs=1000000000 count=1

1+0 enregistrements lus

1+0 enregistrements écrits

1000000000 octets (1,0 GB) copiés, 10,0735 s, 99,3 MB/s

real 0m10.136s

user 0m0.000s

sys 0m0.684s

La vitesse de lecture global du volume est de : **186.1 MB/S**

Client 1 :

time dd if=/dev/zero of=/gluster/glusterMount/A bs=1000000000 count=1

1+0 enregistrements lus

1+0 enregistrements écrits

1000000000 octets (1,0 GB) copiés, 9,538 s, 105 MB/s

real 0m9.568s

user 0m0.000s

sys 0m1.126s

Client2 :

time dd if=/dev/zero of=/gluster/glusterMount/A bs=1000000000 count=1

1+0 enregistrements lus

1+0 enregistrements écrits

1000000000 octets (1,0 GB) copiés, 9,65578 s, 104 MB/s

real 0m9.729s

user 0m0.000s

sys 0m0.754s

La vitesse d'écriture global sur le volume est de : **209 MB/S**

Résultat IOZone :

```
./iozone -i 0 -i 1 -i 2 -s 1g -Rb /gluster/test_s1Gb_i012 -f
/gluster/2S2B/C
```

Taille fichier [GB]	Écriture [MB/sec]	Réécriture [MB/sec]	Lecture [MB/sec]	Relecture [MB/sec]	Lecture aléatoire [MB/sec]	Écriture aléatoire [MB/sec]
1	117.836	126.795	206.989	208.258	15.153	31.960

Avec deux briques, les performances global d'écriture / lecture ont doublé selon nos attentes.

QUATRE SERVEUR, QUATRE BRIQUE, QUATRE CLIENT

Résultat IOZone :

```
./iozone -i 0 -i 1 -i 2 -s 1g -Rb /gluster/test_1G_i012 -f
/gluster/glusterM/txt
```

Taille fichier [GB]	Écriture [MB/sec]	Réécriture [MB/sec]	Lecture [MB/sec]	Relecture [MB/sec]	Lecture aléatoire [MB/sec]	Écriture aléatoire [MB/sec]
1	203.242	207.784	327.913	355.410	22.678	49.587

Avec les quatre briques la linéarité des performances est très claire. C'est au final comme si nous avions n briques qui fonctionnent en parallèle.

4.5 GRAPHIQUE DES TESTES

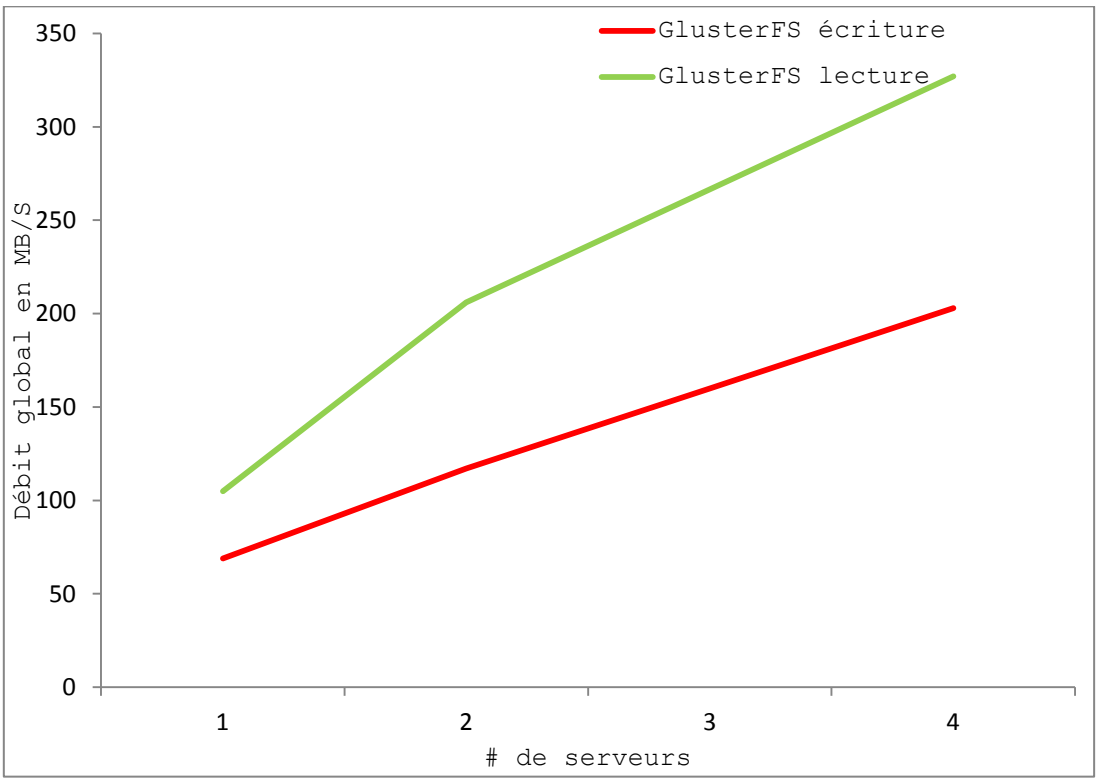

Figure 10 – Linéarité des performances architecture distribué

Dans la figure 10, nous pouvons relever la linéarité des performances et la comparer avec les performances promises. Celle-ci augmente linéairement en fonction du nombre de serveurs. Nous pouvons constater que ce graphique est très proche de celui présenté par la communauté GlusterFS, mis en référence dans ce document à la figure 2.

4.6 DÉBIT UTILE SUR LE RÉSEAU

Dans ce sous chapitre, je réalise une mesure avec Wireshark et avec le menu statistiques de celui-ci, je relève le nombre de kilo-octets transféré sur le réseau. Ce test permet de relever la différence entre la taille du fichier, débit utile, et le débit réellement transféré sur le réseau.

taille fichier [KB]	taille transfert [KB]	delta [KB]	Données envoyés /taille de fichier [%]
1	2.884	1.884	188.4
10	12.152	2.152	21.52
100	104.004	4.004	4.004
1’000	1'020.692	20.692	2.0692
10’000	10'173.058	173.058	1.73058
100’000	100'330.748	330.748	0.330748

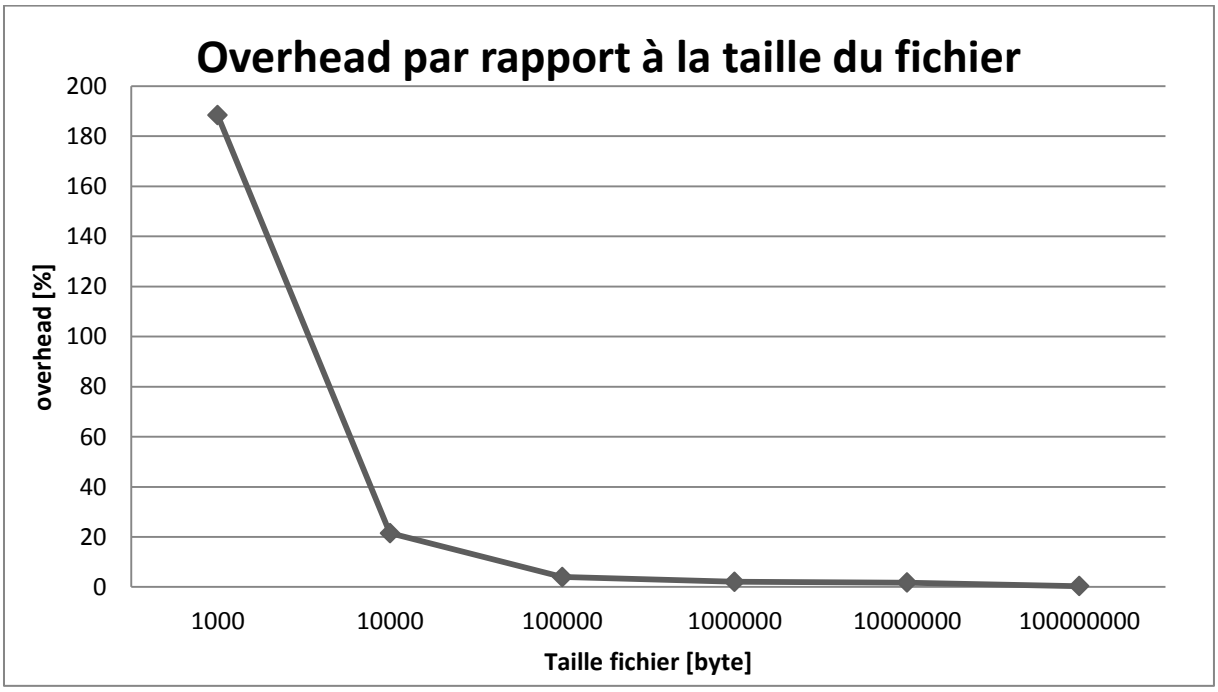

Figure 11 –impacte des en-têtes

Nous pouvons constater que pour des fichiers de 1000 octets nous avons 1884 octets transféré en plus sur le réseau pour les entêtes Ethernet, IP, TCP et GlusterFS en plus des données du fichier. Mais très rapidement avec l’augmentation de la taille du fichier, les entêtes deviennent négligeables.

5 DIFFICULTÉS RENCONTRÉS

5.1 INSTALATION ET EXÉCUTION DE GLUSTERFS

Un des packages de GlusterFS n'était pas installé sur Fedora 16 du laboratoire. Le daemon GlutsterFS n'est pas disponible. Voici quelques commandes qui peuvent résoudre le problème :

```
# yum reinstall fuse.x86_64

# yum reinstall fuse-libs.x86_64

# yum reinstall glusterfs

# yum reinstall glusterfs-fuse

# yum reinstall glusterfs-vim

# yum reinstall glusterfs-devel

# yum reinstall glusterfs-server
```

La commande chkconfig permet de lancer le daemon GlusterFS au démarrage de l'ordinateur. Ici je supprime le glusterd du démarrage et je le remets. Puis je redémarre le service glusterd.

```
# chkconfig --del glusterd

# chkconfig glusterd on

# service glusterd stop

# service glusterd start
```

5.2 BENCHMARK

La méthodologie de mesure n'est pas anodine. Tout d'abord j'ai commencé à exécuter des tests à l'aide de la commande basique dd. Cette commande exécute des écritures et des lectures séquentielles basiques. Le problème avec ces opérations séquentielles lancées avec la commande dd, c'est qu'elle ne représente pas tous les types d'accès possible sur des fichiers. Par exemple, lors de la lecture d'un fichier, il est possible de récupérer des blocs de données non-séquentielles du fichier. Pour prendre en compte différent type d'accès, j'ai exécuté un banc de test en utilisant le logiciel IOzone. En comparant les résultats obtenus avec IOzone pour une écriture séquentielle sur le disque local et les résultats obtenus lors d'un projet précédent [8]réalisé sur le même matériel, les résultats obtenus sont différents. Cette différence est dû en partie à la mémoire cache qui se situe à différents niveaux, tant en mémoire RAM, en mémoire cache du processeur et encore ailleurs.

[8] Résultat test en page 13 dans ce rapport : http://www.tdeig.ch/vmware/Stockage_VM.pdf

6 CONCLUSION

6.1 TECHNIQUE

Durant ce projet, j'ai pu étudier le fonctionnement de GlusterFS et j'ai pu le déployer avec succès sur cinq machines. Les performances linéaires mis en avant par les développeurs de Gluster ont pu être testé et validé par les benchmarks IOZone. Cependant, l'écriture et la lecture aléatoire de fichier est d'environ dix fois moins rapide que la lecture d'un fichier séquentiellement en laissant la configuration par défaut de GlusterFS. L'ajout de brique de stockage à chaud est une fonctionnalité qui permet d'augmenter l'espace de stockage à volonté. Bien évidemment, je n'ai pas pu monter au pétaoctets en termes d'espace de stockage sur les machines exploitées mais j'ai augmenté la taille des volumes selon mes besoins sans que ceci me pose problème. En ce qui concerne la haute disponibilité des données, si une brique n'est plus disponible la brique miroir prend le relai mais avec un temps de latence d'environ vingt secondes. Ce temps de latence est dû au faite que le client essaye de communique avec la brique qui n'est plus disponible. Après un délai, le client va demander la suite du fichier à la brique miroir.

La solution GlusterFS s'acquitte d'un serveur de central. Chaque client et chaque serveur connait comment est composé le cluster de stockage (nombre de briques, volume distribué, adresses IP des serveurs GlusterFS, attributs du volume,…). Le client est dit intelligent, il est capable de retrouver l'emplacement du fichier et de l'utiliser afin d'accomplir l'action voulue. GlusterFS n'est pas un système de fichier en soi. Il ne travail pas au niveau bloque de donnée, bas niveau, mais au niveau fichier. C'est ainsi que les serveurs peuvent posséder des systèmes de fichiers les uns différents des autres.

Les performances d'une brique est principalement limitée par le disque dur, la carte réseau et le réseau utilisé. Dans le cadre des tests réalisé durant ce projet, j'ai obtenu une performance globale de 327 MB/s en débit de lecture à l'aide de quatre serveurs GlusterFS qui tourne sur du matériel usuel.

6.2 PERSONNELLE

GlusterFS est une solution stable, proposant énormément de possibilité intéressante pour une entreprise moderne. Ce n'est pas par hasard que Red Hat, le plus importants distributeur de système GNU/Linux aux entreprise à décider d'intégrer et de supporté GlusterFS. Après cette étude je conseille vivement l'utilisation de ce produit dans un environnement de production. Il est clair que durant ce projet de semestre j'ai testé que quelques une des possibilités de ce logiciel pour des questions de temps. L'utilisation de ce logiciel nécessite une bonne connaissance du système d'exploitation Linux. Heureusement que j'avais déjà utilisé et lu un livre sur Linux il y a quelque temps. Le professeur Litzistorf m'a donné un cours de mise à niveau sur Linux ce qui m'a bien aidé à me remémoré quelques commandes bien utiles. Le laboratoire sur le gestionnaire de volume logique m'a beaucoup aidé pour ce projet ainsi que l'utilisation des analyseurs de protocoles dans divers cours dispensé dans le cursus Bachelor.

Au final, l'administrateur qui doit gérer GlusterFS à besoin de bien lire et comprendre ce qu'est GlusterFS. L'utilisation du serveur et client Gluster se révèle ensuite simple dans leurs utilisations de base.

Ce document ainsi que d'autres rapports réalisé dans le cadre du laboratoire de transmission de donnée sont disponible sur le site internet www.tdeig.ch/linux.

7 ANNEXES

7.1 FONCTIONNALITÉS DE GLUSTERFS

Les fonctionnalités de Gluster sont nombreuses, en voici quelques exemples :

- Haute disponibilité des données
- Espace de stockage capable de croitre jusqu'à plusieurs pétaoctets
- Performance linéaire
- Géo-réplication des données
- Accès possible à l'aide de protocoles tel que CIFS, NFS, FTP, HTTP(S)
- Augmentation de l'espace de stockage à la volé (depuis la version 3.1.0)
- Pas de métadata grâce à l'algorithme élastique de hachage

7.2 DEPLOIMENT DE GLUSTERFS

Gluster peut être déployé de deux manières. La première méthode est d'installer le logiciel GlusterFS sur une distribution linux 64 bit. La deuxième méthode est de déployer une appliance qui intègre GlusterFS sur VMware ESX ou linux KVM. GlusterFS recommande du RAID 10 sur les machines qui vont servir de serveur.

7.3 CONFIGURATION DU SYSTÈME[9]

CONFIGURATION SERVEURS

Installation de GlusterFS sur Fedora

```
# yum install glusterfs glusterfs-server  glusterfs-fuse
```

Démarrage du daemon glusterd

Pour construire un cluster de stockage, lancer le daemon glusterd sur chaque noeud destiné à contenir une ou plusieurs briques de stockage.

```
# chkconfig glusterd on

# service glusterd restart
```

Glusters offre une commande CLI interactive connu sous le nom de Gluster Console Manager qui simplifie la configuration et le gestion de l'environnement de stockage.

Gluster permet la gestion du cluster depuis n'importe quel nœud serveur du cluster de stockage ou par une connexion sécurisée distante à un des serveurs.

Pour exécuter une commande depuis le terminal nous pouvons taper :

[9] Gluster FS Administration Guide : http://download.gluster.com/pub/gluster/glusterfs/3.2/Documentation/AG/pdf/Gluster_File_System-3.2.5-Administration_Guide-en-US.pdf

```
# gluster COMMANDE
```

ou bien nous pouvons utiliser Gluster CLI en mode interactif comme suit

```
# gluster

gluster> COMMANDE
```

Pour afficher l'aide de Gluster :

```
gluster> help
```

Maintenant nous allons mettre en relation un ensemble de nœuds pour crée le cluster de stockage. L'ensemble de ces serveurs est appelé dans la documentation GlusterFS: Trusted Storage Pool.

Quand nous lançons le premier serveur, le pool de serveurs est composé seulement de ce serveur. Pour ajouter d'autres serveurs au pool, il faut envoyer une demande à chaque serveur depuis un des serveurs du pool.

```
# gluster peer probe IP_SERVEUR_INTÉROGÉ
```

Pour afficher les serveurs du cluster de stockage actuellement dans le pool ainsi que leurs états :

```
# gluster peer status
```

Pour enlever un serveur du pool, il faut taper cette commande :

```
# gluster peer detach IP_SERVEUR
```

Attention, les fichiers et les dossiers stockés sur le serveur détacher ne peuvent plus être accessible à moins de les redistribués sur les autres nœuds.

Maintenant que les nœuds du cluster sont lié, nous pouvons crée un volume. Pour créer un volume distribué où les fichiers sont répartis sur les briques de stockage il faut taper cette commande :

```
# gluster volume create NEW-VOLNAME [stripe COUNT | replica COUNT]
[transport tcp | rdma | tcp, rdma] NEW-BRICK1 NEW-BRICK2 …
```

Exemple :
Sur serveur 1 :

```
# mkdir /exp2
```

Sur serveur 2 :

```
# mkdir /exp2

# gluster volume create test-volume serveur1:/exp1 serveur2:/exp2 …
```

Nous pouvons aussi spécifier plusieurs options sur ce volume :
Exemple :

```
# gluster volume set test-volume auth.allow 10.1.40.*
```

Cette commande permet d'autoriser l'accès de ce volume au client avec une ip qui commence par 10.1.40

Maintenant que le volume est créé, il ne faut pas oublier de le démarrer pour pouvoir y accéder.
```
# gluster volume start NOM_VOLUME
```

Pour vérifier l'état d'un volume nous pouvons exécuter cette commande :
```
# gluster volume info NOM_VOLUME
```

Désormais, le volume est configuré et accessible depuis un poste client. Nous pouvons monter le volume sur le poste client.

Nous venons de mettre en service un système de fichiers distribué. Pour garantir l'accès aux données dans le cas d'une défaillance d'un des serveurs, le traducteur de distribution est très souvent accompagné du traducteur de réplication.

```
# gluster volume create test-volume replica 2 transport tcp
serveur1:/exp1 serveur2:/exp2 serveur3:/exp3 serveur4:/exp4
```

CONFIGURATION CLIENT

Montage manuel du volume
```
# mount -t glusterfs ip_d'un_des_seveurs:/NOM_VOLUME /REP_DU_MONTAGE
```

Exemple
```
# mount -t glusterfs 10.1.40.98:/test-volume /mnt/glusterfs
```

Montage automatique du volume
Il faut éditer le fichier /etc/fstab et ensuite ajouté cette ligne au fichier fstab :

```
Ip_d'un_des_serveurs:/NOM_VOLUME   REPERTOIRE_DU_MONTAGE   glusterfs
defaults,_netdev 0 0
```

DOSSIER UTILES SUR SERVEUR GLUSTERFS

- Fichiers log :
  /var/log/glusterfs
- Stockage des peers et information constitution volumes :
  /var/lib/glusterd
- Documentations et exemples volumes :
  /usr/share/doc
  /etc/glusterfs

8 LIENS ET RÉFÉRENCES

Site officiel de GlusterFS. Documentations de la solution :

http://www.gluster.org

Présentation sur RAIN (Reliable Redundant Random Array of Inexpensive Independant Nodes) et les traducteurs utilisés dans GlusterFS :

http://www.mathrice.org/rencontres/octobre.2007/support-GlusterFS.pdf

Article extrêmement intéressant et complet sur GlusterFS :

http://www.linux-mag.com/id/7833/

Comparaison des solutions de stockage distribué - Institut Universitaire de Technologie Nancy-Charlemagne :

http://www.generation-linux.fr/dl/stockage-distribue_rapport.pdf

Mise en place d'un cluster GlusterFS et test de celui-ci avec la commande dd :

http://prefetch.net/blog/index.php/2011/11/13/creating-clustered-file-systems-with-glusterfs-on-centos-and-fedora-linux-servers/

Test de GlusterFS en créant des dossiers imbriqués et fichiers avec l'outil fileop fourni avec le logiciel IOzone :

http://blog.yacoubi.fr/index.php?tag/glusterfs

Package Wireshark avec décodeur GlusterFS :

http://repos.fedorapeople.org/repos/devos/wireshark-gluster/fedora-16/

Site officiel du logiciel de benchmark IOzone :

http://www.iozone.org

Site du laboratoire de transmission de donnée HEPIA :

http://www.tdeig.ch/