Clusters de stockage : GlusterFS

David Delavennat

Centre de Génétique Moléculaire

Rencontres *Mathrice*, Octobre 2007

Plan

Plan

Redondant Array of Inexpensive Node

Définition

L'architecture **RAIN** (Reliable|Redundant|Random Array of Inexpensive|Independant Nodes) est au départ un sujet de recherche partant d'une vraie réflexion d'informatique théorique pour s'appliquer aux applications critiques des entreprises. Les chercheurs voulaient développer un modèle informatique distribué pour le stockage à base de **composants standards**. Le sujet a été étudié aux états-unis par **CalTech** (l'Institut de Technologie de Californie), le laboratoire **JPL** (Jet Propulsion) de la **NASA** et par le **DARPA** (Defense Advanced Research Projects Agency, département de la défense).

Redondant Array of Inexpensive Node

Description

- Distribution entre les noeuds assurée par des assemblages dits **Maximum Distance Separable Array Codes** permettant de calculer une répartition des données et d'assurer le recouvrement en cas de défaillance d'un élément de la chaîne.
- **Auto-reconfiguration** en cas de panne d'un constituant, d'un ajout ou d'un retrait d'un noeud du cluster.
- Aucune limite du nombre de noeuds.

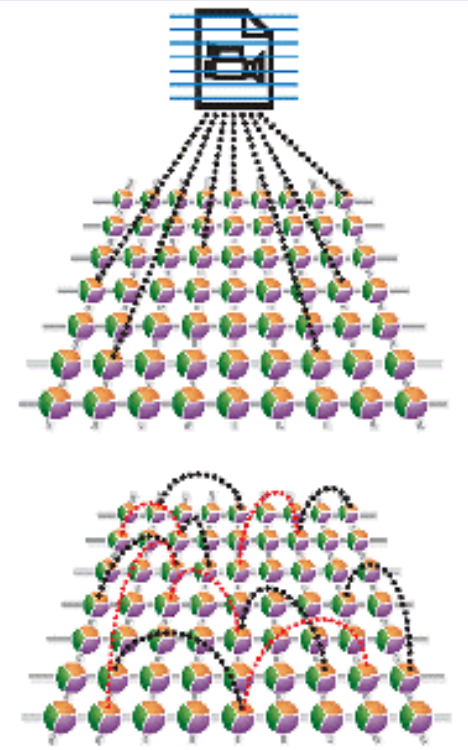
Redondant Array of Inexpensive Node

Description

- La couche RAIN fournit un **mécanisme d'équilibre de charge** au sein du cluster pour les requêtes entrantes et sa philosophie de redondance permet d'accepter plusieurs défaillances de plusieurs éléments de la configuration : noeud, interface ou lien réseau, switch, stockage ou noeud complet.
- Côté configuration, il peut être envisagé de déporter certains noeuds et de mixer des liens LAN et WAN pour rigidifier le cluster.

Redondant Array of Inexpensive Node

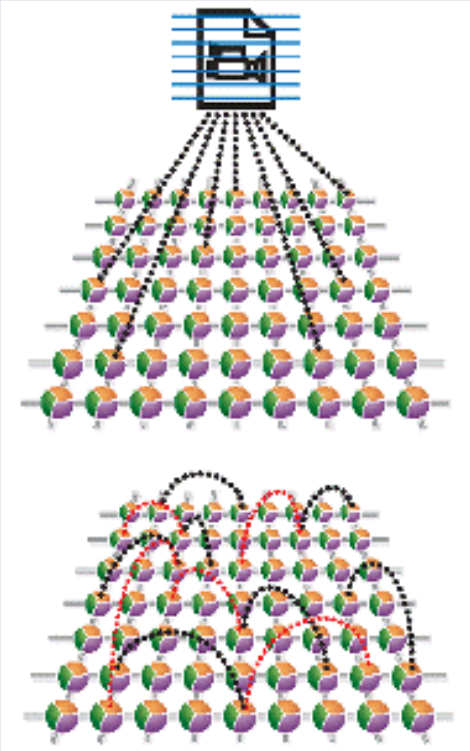
Description



- Gestion de fichiers, pas de disques.
- Evolutivité multi-dimensionnelle : performance, capacité, redondance
- Donnée éclatée/répliquée/distribuée entre plusieurs entités.
- Redondance assurée avec une finesse importante contrairement à une approche massive type RAID, coûteuse en reconstruction, qui fonctionne au niveau volume ou lun.

Redondant Array of Inexpensive Node

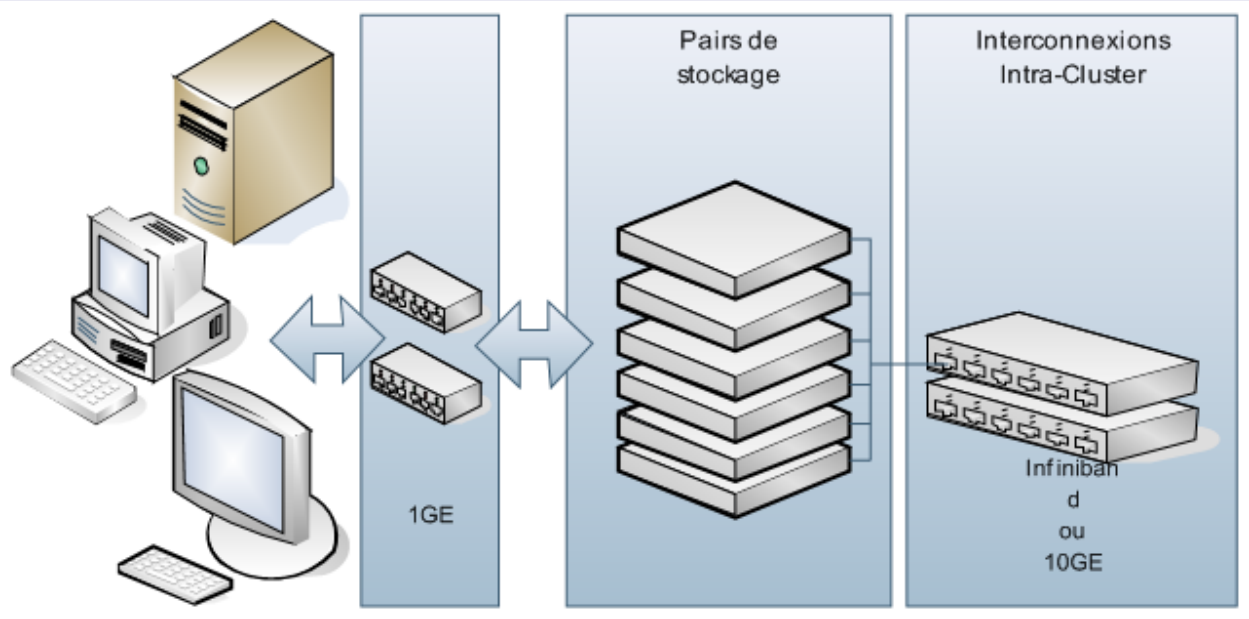
Description



- Répartition des fragments de fichiers sur plusieurs unités permettant d'aggréger la bande passante, de contrôler et limiter le temps de traitement.
- Petit et gros fichiers ne seront pas divisé dans le même nombre de fragments.
- Les datas ne peuvent plus vivre avec les meta-datas comme les systèmes de fichiers de conception "ancienne" le faisaient.

Redondant Array of Inexpensive Node

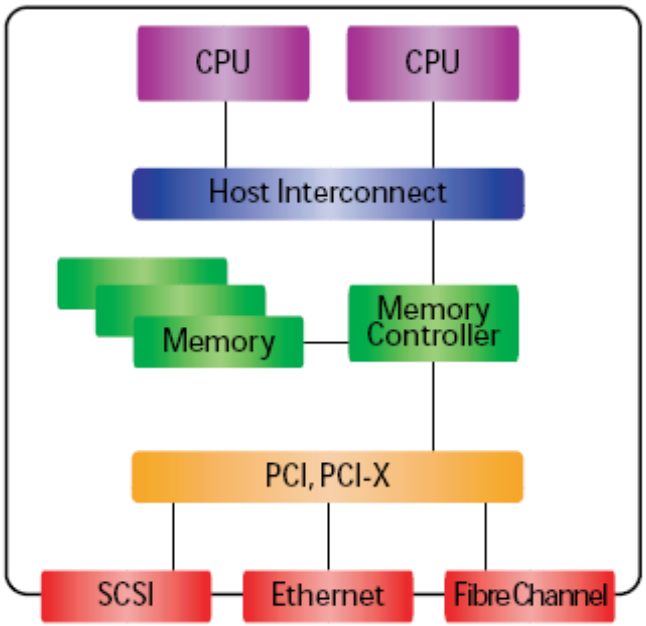
Schéma



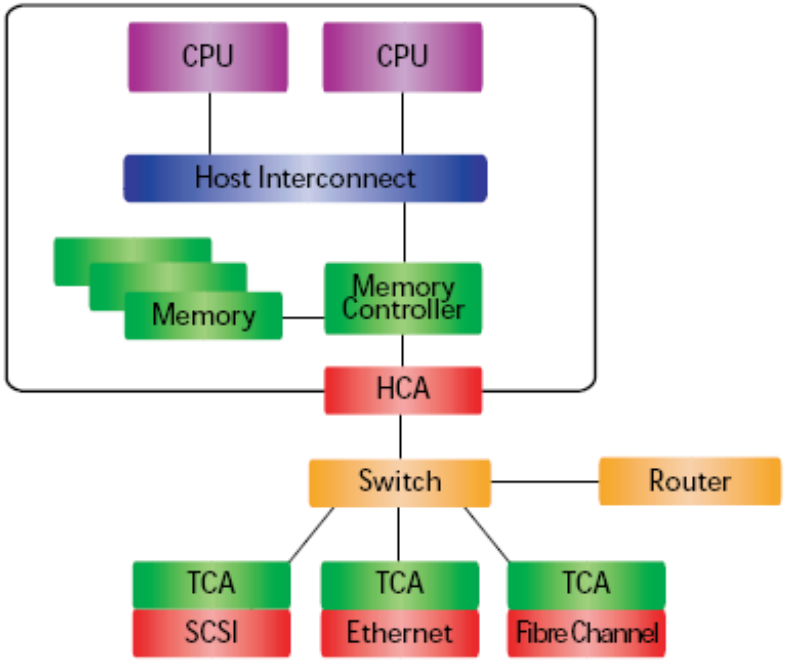
Plan

Infiniband

Architecture



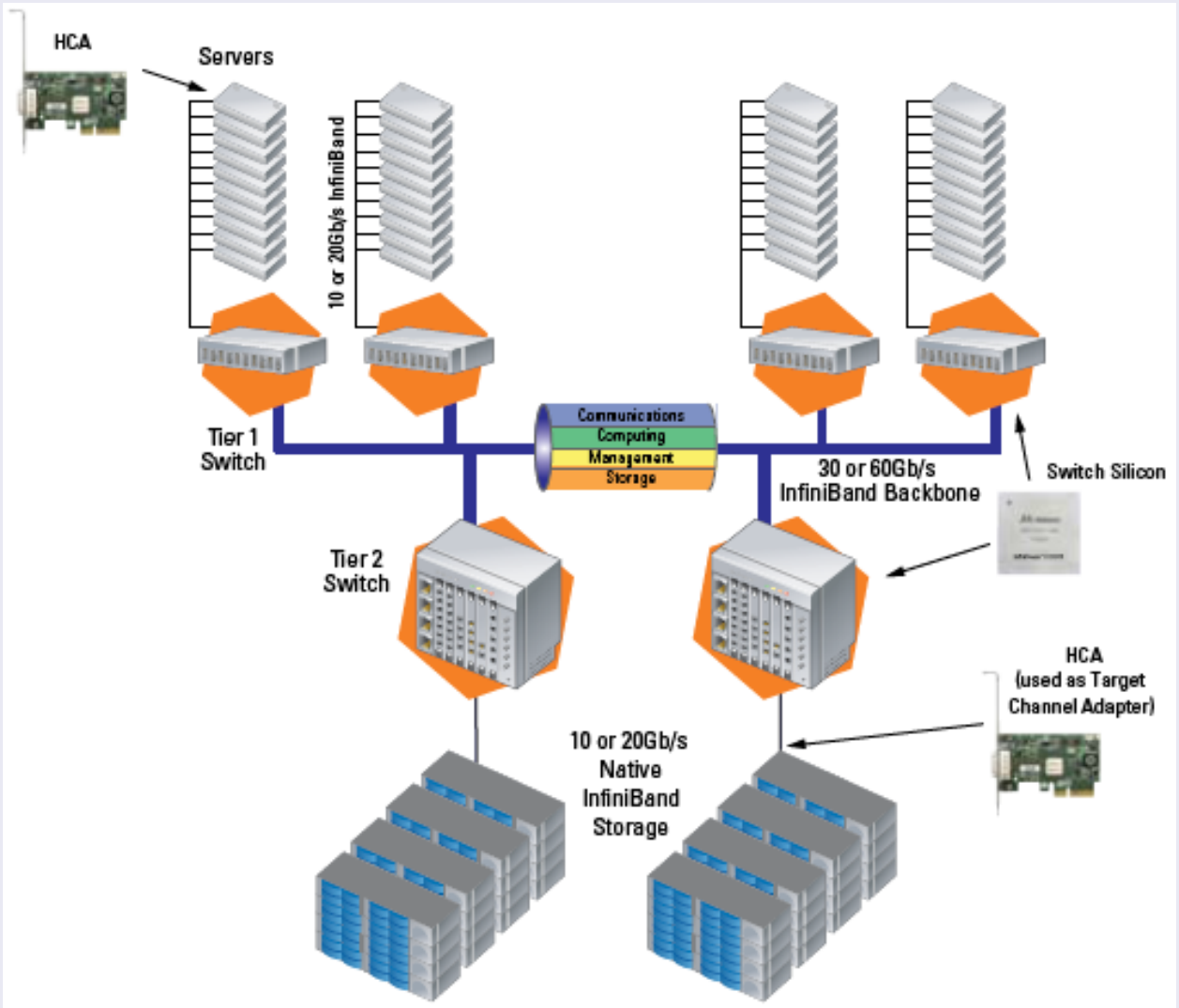
Example configuration of a PCI or PCI-X shared bus



With InfiniBand, I/O devices that traditionally reside within the server are moved to the network

Infiniband

Architecture



Infiniband

Débits

- 20Go/s point à point
- 60Go/s commutateurs à commutateurs
- 120Go/s en prévision

NOTE : Fiber-Channel tourne à 10Go/s maximum

Infiniband

Points Techniques

- Infiniband n'est pas concu autour des standards Ethernet/IP
- utilise la notion de **R**emote **D**irect **M**emory **A**ccess
- les cartes infiniband accèdent directement à la mémoire des noeuds distants
- accès en lecture comme en ecriture

Infiniband

Pile logicielle

- verbs : commandes élémentaires envoyées à la carte infiniband (bibliothèque ibverbs)
- **D**irect **A**ccess **P**rogramming **L**ibrary : interface de plus haut niveau. kDAPL en espace noyau, uDAPL en espace utilisateur
- **S**ocket **D**irect **P**rotocol : émulation de socket sur Infiniband
- **S**CSI **R**DMA **P**rotocol : accès distant par infiniband à des disques SCSI
- IPoIB : IP sur Infiniband

Plan

Traducteurs

Genèse

- Développement du noyau Hurd par la FSF à partir de 1990
- Paradigme objet
- Le **V**irtual **F**ile **S**ystem unix offre un système de nommage déterministe au travers de sa structure arborescente.
- Programme associé à un noeud/fichier du VFS.
- Comparable à un trigger de base de donnée.

Rappel : sous unix un répertoire est un fichier.

Traducteurs

Les classes de traducteur Hurd

- TrivFS → ne s'attache qu'a un seul noeud
- NetFS → s'attache à un noeud de type répertoire et crée une arborescence
- DiskFS → semblable à NetFS mais spécifiquement conçu pour un usage avec une unité physique

Traducteurs

Traducteur Hurd

- Il effectue une action en réaction à un évènement (ouverture, écriture, modification des droits...) lié au noeud auquel il est associé
- La pile TCP/IP, l'exécution de binaire ou le système de fichier ext2 fonctionne au travers de traducteurs
- Mode actif : association entre le traducteur et le noeud VFS jusqu'au prochain redémarrage
- Mode passif : $\sim$ démon/service

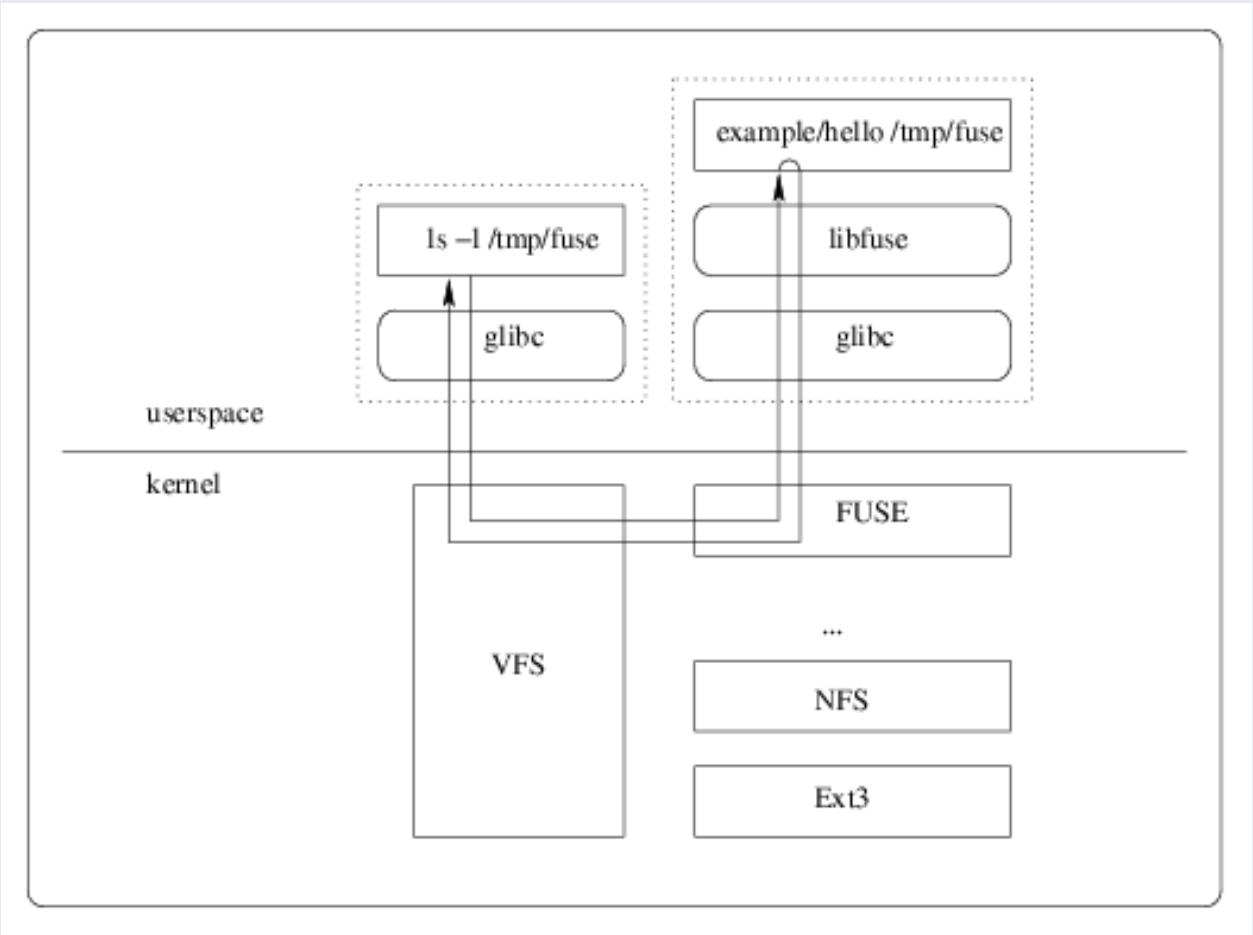
Traducteur

Filesystem in **Use**rland

- API simple
- Portable (pas besoin de patch noyau)
- La bibliothèque et le module noyau de FUSE communiquent au travers du fichier spécial /**dev**/**fuse**.
- Interface entre le paradigme unix et le paradigme hurd

Fuse

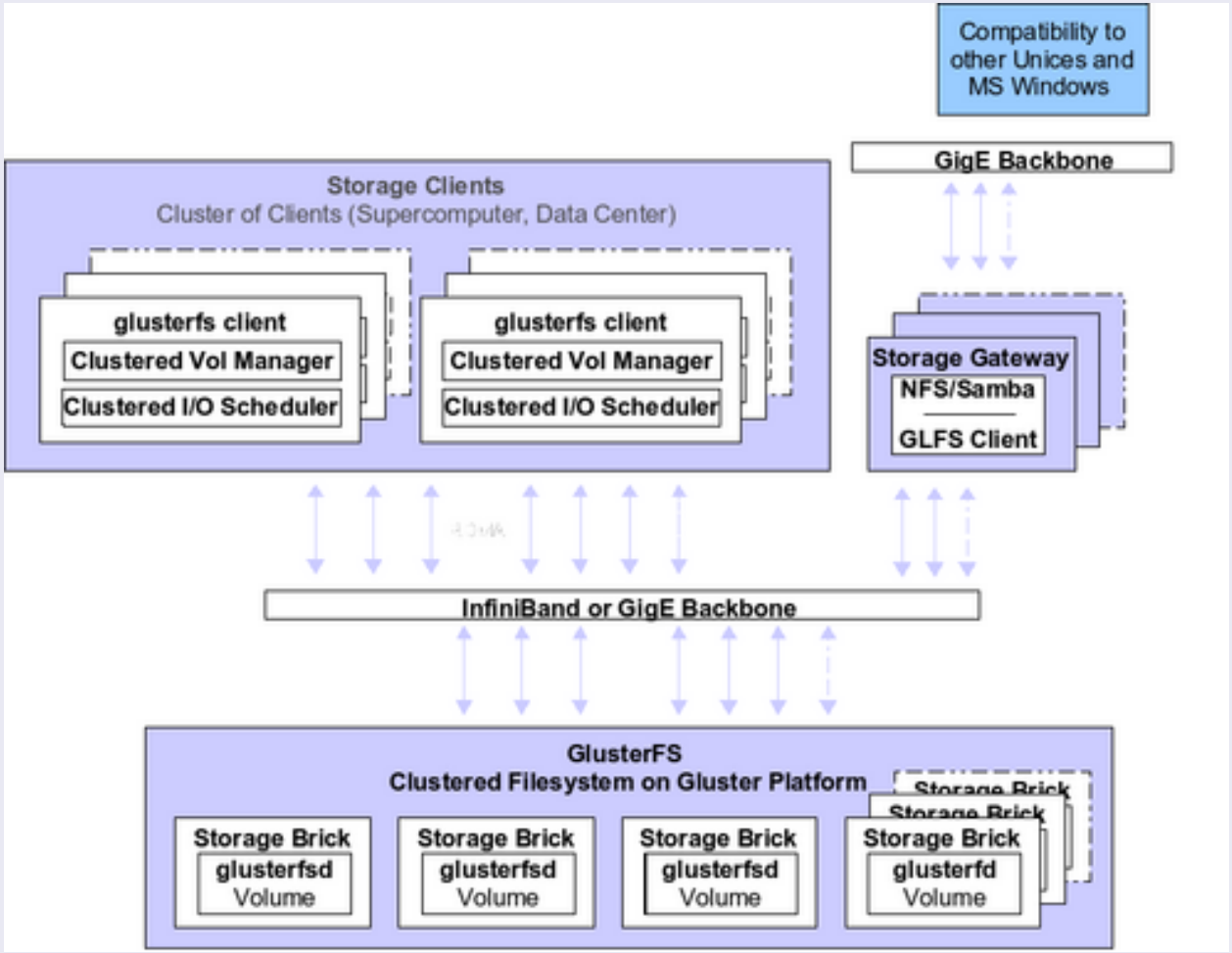
Exemple



Plan

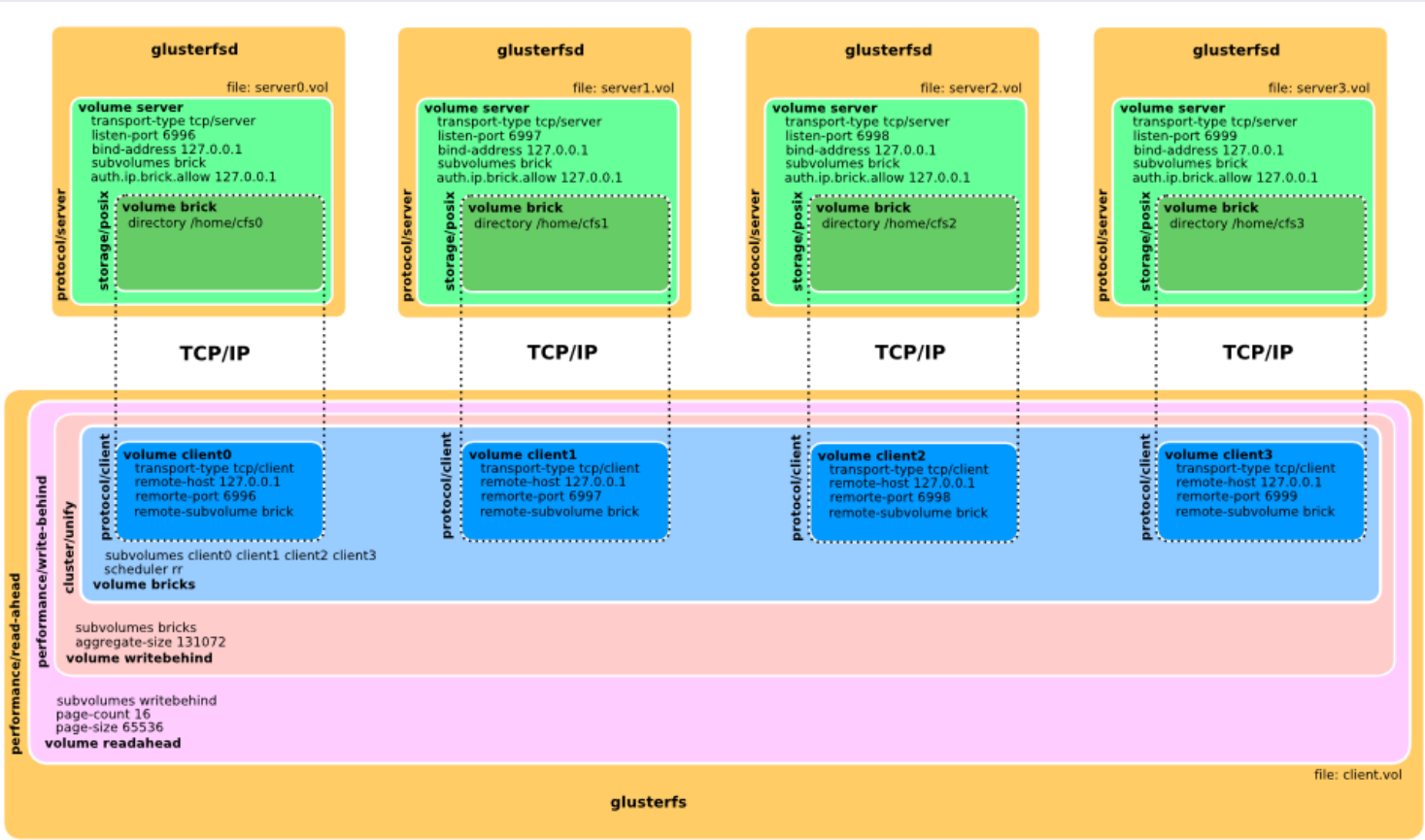
GlusterFS

Architecture



GlusterFS

Architecture



Configuration example of glusterFS with 4 storage nodes and 1 client node

Traducteur de stockage

Posix

- GlusterFS s'appuie sur un système de fichier traditionnel (Ext3 ou XFS)
- Le traducteur POSIX fait le lien entre le serveur GlusterFS et le système de fichier sous-jacent

```
1 volume posix1
2   type storage/posix
3   option directory /home/export
4 end-volume
```

Traducteurs d'agrégat

Automatic-File-Replicator

- réplication de fichiers en fonction d'un motif
- la réplication est faite dans l'ordre de déclaration des sous-volumes
- le système de fichier sous-jacent doit supporter les attributs étendus

```
1 volume afr
2   type cluster/afr
3   subvolumes brick1 brick2 brick3 brick4
4   option replicate *.html:2,*.db:1,*:3
5 # *.html   => brick1, brick2
6 # *.db     => brick1
7 # le reste => brick1, brick2, brick3
8 # rien sur brick4
9 end-volume
```

Traducteurs d'agrégat

Unify

- unifie plusieurs bricks en un seul volume
- la distribution des fichiers se fait selon l'ordonnanceur choisit

```
1  volume unify
2      type cluster/unify
3      subvolumes brick1 brick2 brick3 brick4 brick5
           brick6 brick7 brick8
4  # ne doit pas être un 'subvolumes'
5      option namespace brick-ns
6  # ordonnanceur round-robin
7      option scheduler rr
8  end-volume
```

Traducteurs d'agrégat

Stripe

- découpe les fichiers en block de taille block-size et les répartit sur les sous-volumes en fonction d'un motif
- le système de fichier sous-jacent doit supporter les attributs étendus

```
1 volume stripe
2     type cluster/stripe
3     subvolumes brick1 brick2 brick3 brick4
4     option block-size *avi:10MB,*mpg,*dat:100MB
5 # *avi => decoupé en block de 10Mo,
6 # *dat => decoupé en block de 100Mo
7 # *mpg => découpé en block de 128Ko (défaut)
8 end-volume
```

Traducteurs d'ordonnancement

Adaptive **L**east **U**sage

C'est l'ordonnanceur le plus puissant disponible. Il s'adapte dynamiquement aux différentes caractéritiques spécifiées.

- disk-usage
- read-usage
- write-usage
- open-files-usage
- disk-speed-usage $\longrightarrow$ peu utile, cette valeur étant constante

Traducteurs d'ordonnancement

Adaptive Least Usage

```
volume bricks
 type cluster/unify
 subvolumes brick1 brick2 brick3 brick4
 option scheduler alu   # use the ALU scheduler
 option alu.limits.min-free-disk  5%
 option alu.limits.max-open-files 10000
 option alu.order disk-usage
 option alu.disk-usage.entry-threshold 2GB
# 2048-60=1988
 option alu.disk-usage.exit-threshold 60MB
 option alu.stat-refresh.interval 10sec
 option alu.stat-refresh.num-file-create 10
end-volume
```

Traducteurs d'ordonnancement

Non-Uniform Filesystem Scheduler

- utilisé en environnement HPC
- donne une priorité d'accès au système de fichier local par rapport à celui des autres noeuds

```
1  volume posix1 # stockage local au noeud
2    type storage/posix
3    option directory /home/export
4  end-volume
5  volume bricks
6    type cluster/unify
7    subvolumes posix1 brick2 brick3 brick4
8    option scheduler nufa
9    option nufa.local-volume-name posix1
10   option nufa.limits.min-free-disk 5%
11 end-volume
```

Traducteurs d'ordonnancement

Random

```
1 volume bricks
2   type cluster/unify
3   subvolumes brick1 brick2 brick3 brick4
4   option scheduler random
5   option random.limits.min-free-disk 5%
6 end-volume
```

Traducteurs d'ordonnancement

Round-Robin

```
1 volume bricks
2   type cluster/unify
3   subvolumes brick1 brick2 brick3 brick4
4   option scheduler rr
5   option rr.limits.min-free-disk 5%
6   option rr.refresh-interval 10
7 end-volume
```

Traducteurs d'export

Serveur

```
volume server
  type protocol/server
  option transport-type tcp/server
# option transport-type ib-sdp/server
# option transport-type ib-verbs/server
# option ib-verbs-work-request-recv-size   1048576
# option ib-verbs-work-request-recv-count  16
# option ib-verbs-work-request-send-size   1048576
# option ib-verbs-work-request-send-count  16
# option bind-address 192.168.1.10
# option listen-port 6996
# option client-volume-filename /etc/client.vol
  subvolumes brick1 brick2
  option auth.ip.brick1.allow 192.168.*
  option auth.ip.brick2.allow 192.168.*
end-volume
```

Traducteurs d'export

Client

```
1  volume client1
2    type protocol/client
3    option transport-type tcp/client
4  # option transport-type ib-sdp/client
5  # option transport-type ib-verbs/client
6  # option ib-verbs-work-request-recv-size   1048576
7  # option ib-verbs-work-request-recv-count  16
8  # option ib-verbs-work-request-send-size   1048576
9  # option ib-verbs-work-request-send-count  16
10   option remote-host 192.168.1.10
11   option remote-port 6996
12   option transport-timeout 30
13   option remote-subvolume brick
14 end-volume
```

Traducteurs divers

Posix-locks

- support des verrous POSIX (verrou fcntl) indépendamment du stockage sous jacent.
- s'utilise coté serveur en surcouche du traducteur de stockage POSIX.
- fournit des verrous de type 'consultatif' (advisory) et 'obligatoire' (mandatory)

```
1 volume locks
2   type features/posix-locks
3   subvolumes brick
4 end-volume
```

Traducteurs divers

Trash

```
1 volume trash
2   type features/trash
3   option trash-dir /.trashcan
4   subvolumes brick
5 end-volume
```

Traducteurs de performance

Read-Ahead

- pré-charge une séquence de blocks selon des prédictions de besoin → accélère les lectures consécutives
- aggrège les entrées/sorties afin de réduire la charge réseau et la charge disque

NOTE : Ce traducteur marche bien avec le transport infiniband. Sans read-ahead on peut saturer des interfaces GigaEthernet.

```
1 volume readahead
2     type performance/read-ahead
3     option page-size 128kB          # 256Ko par défault
4     option page-count 16            # 16
5     option force-atime-update off # off
6     subvolumes <x>
7 end-volume
```

Traducteurs de performance

Write-Behind

En général les opérations d'écriture sont plus lentes que les opérations de lecture. write-behind aggrège les opérations d'écriture en arrière plan de manière non bloquante. aggregate-size définie la taille de block à aggréger avant écriture. Différentes valeurs sont à tester afin de trouver la taille optimale en fonction de l'environnement (mémoire, transport, charge).

```
1 volume writebehind
2     type performance/write-behind
3     option aggregate-size 1MB      # 0bytes par défaut
4     option flush-behind on         # off
5     subvolumes <x>
6 end-volume
```

Traducteurs de performance

Threaded-IO

- optimise l'usage des ressources lors des lectures/écritures asynchrones.
- le CPU, la mémoire, et le réseau ne sont pas solicités lorsque le serveur est bloqué sur des opérations DMA disques.
- les temps de blocage pour achèvement de certaines opérations sont utilisés pour gérer de nouvelles requettes.

```
1 volume iothreads
2     type performance/io-threads
3     option thread-count 4         # 1 par défaut
4     option cache-size 32MB        # 64MB
5     subvolumes <x>
6 end-volume
```

Traducteurs de performance

IO-Cache

```
volume io-cache
  type performance/io-cache
  option cache-size 64MB                  # défaut 32Mo
  option page-size 1MB                    # 128Ko
  option priority *.h:3,*.html:2,*:1 # *:0
  option force-revalidate-timeout 2  # 1
  subvolumes <x>
end-volume
```

Traducteurs de performance

Stat Pre-fetch

- récupère les informations de tous les fichiers du répertoire en une seule opération
- améliore les temps de réponse des commandes interactives

```
1 volume stat-performance
2   type performance/stat-prefetch
3   option cache-seconds 1 # Timeout.
4   subvolumes <x>
5 end-volume
```

Plan

Exemples

Clustered Mode

- deux briques configurées en mode cluster AFR
- chaque brique mirror l'autre

cf fichiers de config...