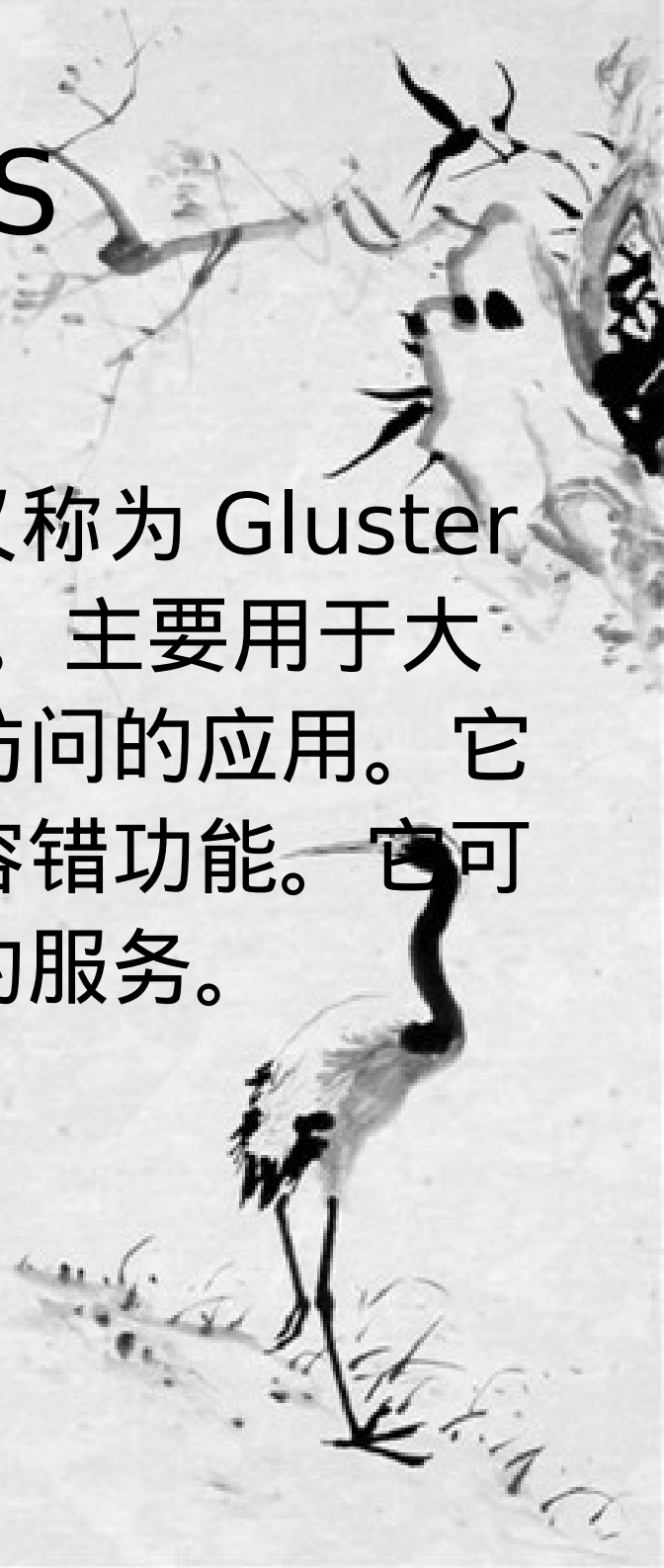
GNU/Linux-GFS

GNU/Linux-GFS

GFS:

Goole FS—google 文件系统，又称为 Gluster FS 。属于可扩展的分布式文件系统。主要用于大型的、分布式的、对大量数据进行访问的应用。它运行于廉价的普通硬件上，并提供容错功能。它可以给大量的用户提供总体性能较高的服务。



GNU/Linux-GFS

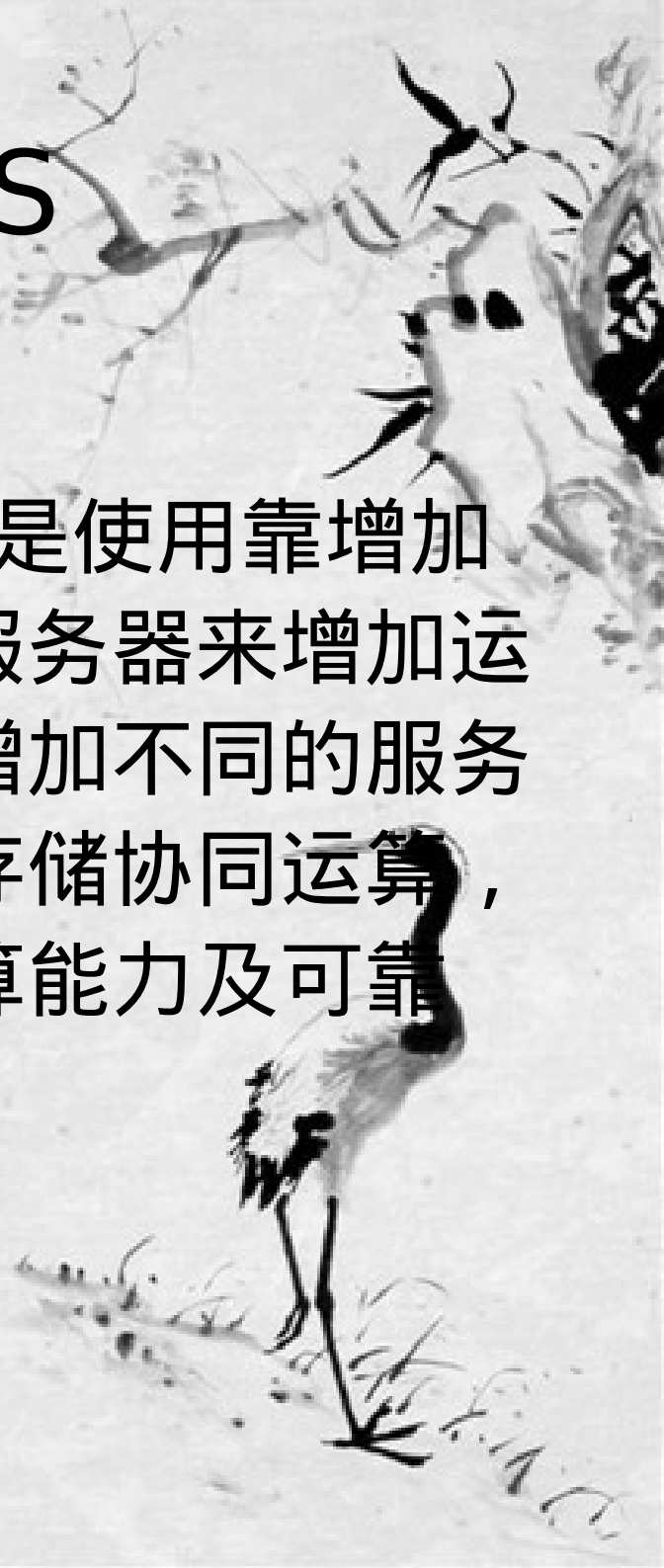
GFS:

属于 Scale-Out(横向扩展) 存储解决方案 Gluster 的核心，它是一个开源的分布式文件系统，具有强大的横向扩展能力，通过扩展能够支持数 PB 存储容量和处理数千客户端。 GlusterFS 借助 TCP/IP 或 InfiniBand RDMA 网络将物理分布的存储资源聚集在一起，使用单一全局命名空间来管理数据。 GlusterFS 基于可堆叠的用户空间设计，可为各种不同的数据负载提供优异的性能。

GNU/Linux-GFS

Scale Out(横向扩展) :

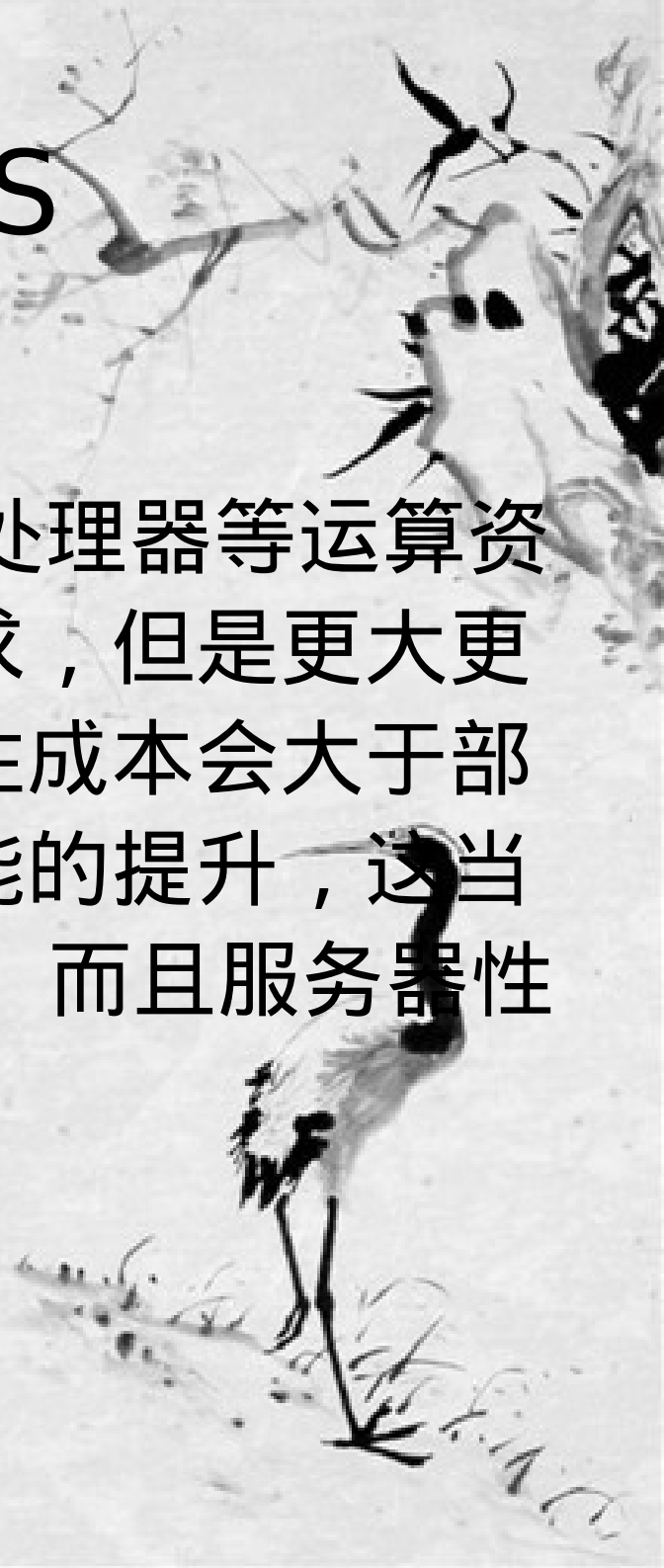
从字面意思来看， Scale Out 是使用靠增加处理器来提升运算能力和增加独立服务器来增加运算能力。就是指企业可以根据需求增加不同的服务器和存储应用，依靠多部服务器、存储协同运算，借负载平衡及容错等功能来提高运算能力及可靠度。



GNU/Linux-GFS

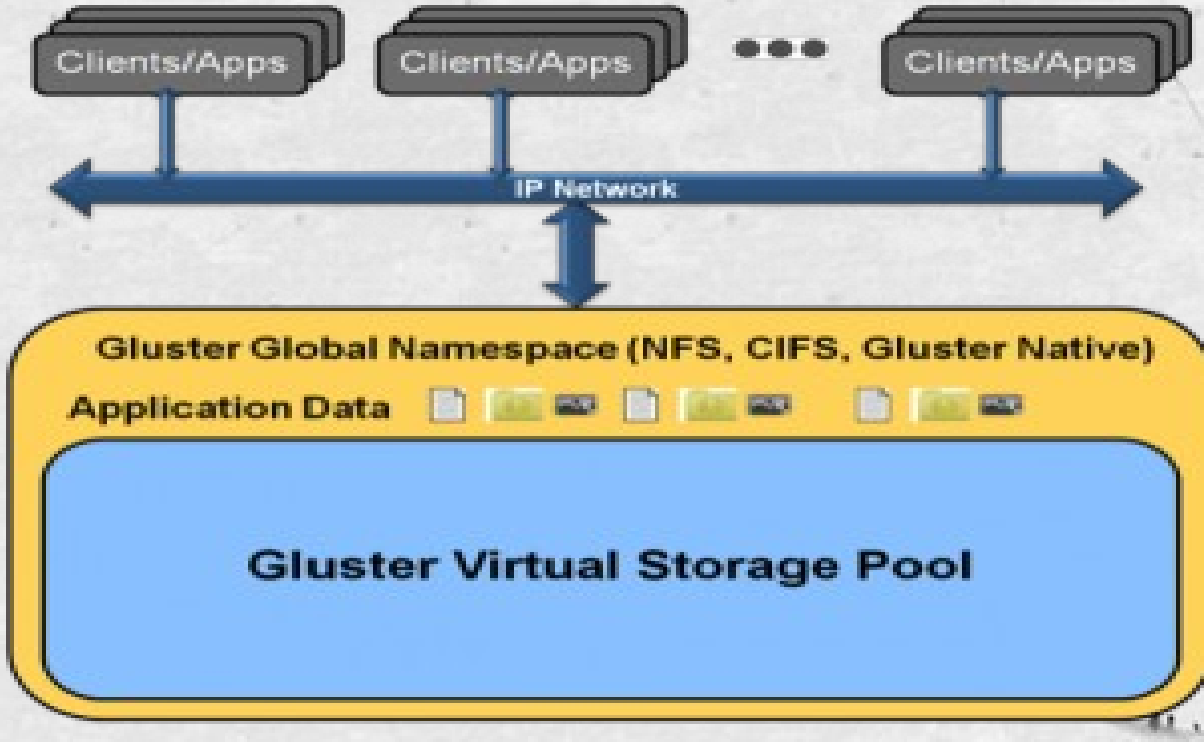
Scale Up(纵向扩展) :

指企业后端大型服务器以增加处理器等运算资源进行升级以获得对应用性能的要求，但是更大更强的服务器同时也是更昂贵的，往往成本会大于部署大量相对便宜的服务器来实现性能的提升，这当中的代表当属 IBM zSeries 大型机。而且服务器性能所能提高的程度也有一定的上限。



GNU/Linux-GFS

GFS



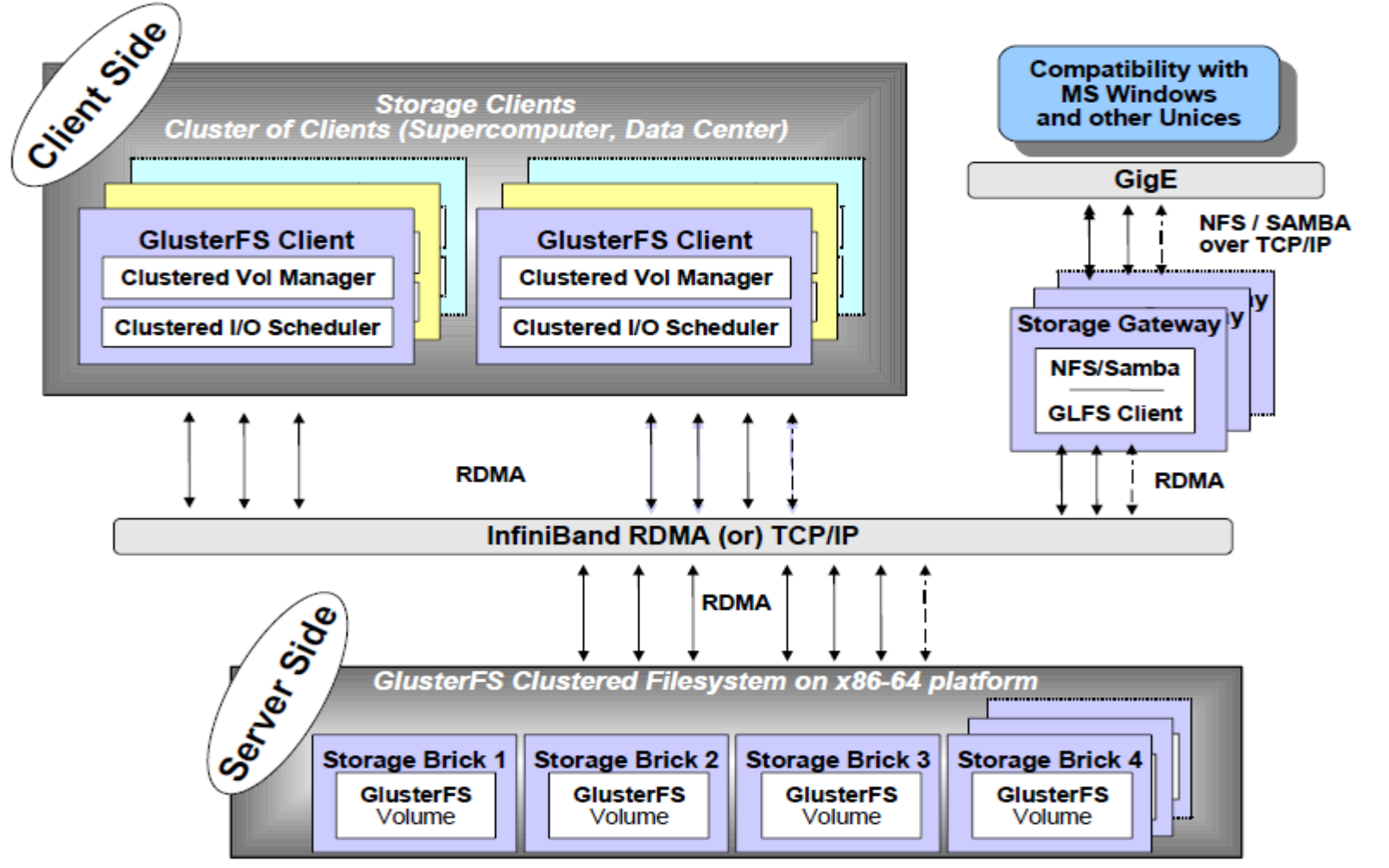
GNU/Linux-GFS

GFS

支持运行在任何标准 IP 网络上标准应用程序的标准客户端，如图 2 所示，用户可以在全局统一的命名空间中使用 NFS/CIFS 等标准协议来访问应用数据。 GlusterFS 使得用户可摆脱原有的独立、高成本的封闭存储系统，能够利用普通廉价的存储设备来部署可集中管理、横向扩展、虚拟化的存储池，存储容量可扩展至 TB/PB 级。

GNU/Linux-GFS

GFS 架构与组成



GNU/Linux-GFS

GFS 特点

1. 扩展性和高性能

GlusterFS 利用双重特性来提供几 TB 至数 PB 的高扩展存储解决方案。 Scale-Out 架构允许通过简单地增加资源来提高存储容量和性能，磁盘、计算和 I/O 资源都可以独立增加，支持 10GbE 和 InfiniBand 等高速网络互联。 Gluster 弹性哈希（ Elastic Hash ）解除了 GlusterFS 对元数据服务器的需求，消除了单点故障和性能瓶颈，真正实现了并行化数据访问。

GNU/Linux-GFS

GFS 特点

2. 高可用性

GlusterFS 可以对文件进行自动复制，如镜像或多次复制，从而确保数据总是可以访问，甚至是在硬件故障的情况下也能正常访问。自我修复功能能够把数据恢复到正确的状态，而且修复是以增量的方式在后台执行，几乎不会产生性能负载。

GlusterFS 没有设计自己的私有数据文件格式，而是采用操作系统中主流标准的磁盘文件系统（如 EXT3、 ZFS ）来存储文件，因此数据可以使用各种标准工具进行复制和访问。

GNU/Linux-GFS

GFS 特点

3. 全局统一命名空间

全局统一命名空间将磁盘和内存资源聚集成一个单一的虚拟存储池，对上层用户和应用屏蔽了底层的物理硬件。存储资源可以根据需要在虚拟存储池中进行弹性扩展，比如扩容或收缩。当存储虚拟机映像时，存储的虚拟映像文件没有数量限制，成千虚拟机均通过单一挂载点进行数据共享。虚拟机 I/O 可在命名空间内的所有服务器上自动进行负载均衡，消除了 SAN 环境中经常发生的访问热点和性能瓶颈问题。

GNU/Linux-GFS

GFS 特点

4. 弹性哈希算法

GlusterFS 采用弹性哈希算法在存储池中定位数据，而不是采用集中式或分布式元数据服务器索引。在其他的 Scale-Out 存储系统中，元数据服务器通常会导致 I/O 性能瓶颈和单点故障问题。

GlusterFS 中，所有在 Scale-Out 存储配置中的存储系统都可以智能地定位任意数据分片，不需要查看索引或者向其他服务器查询。这种设计机制完全并行化了数据访问，实现了真正的线性性能扩展。

GNU/Linux-GFS

GFS 特点

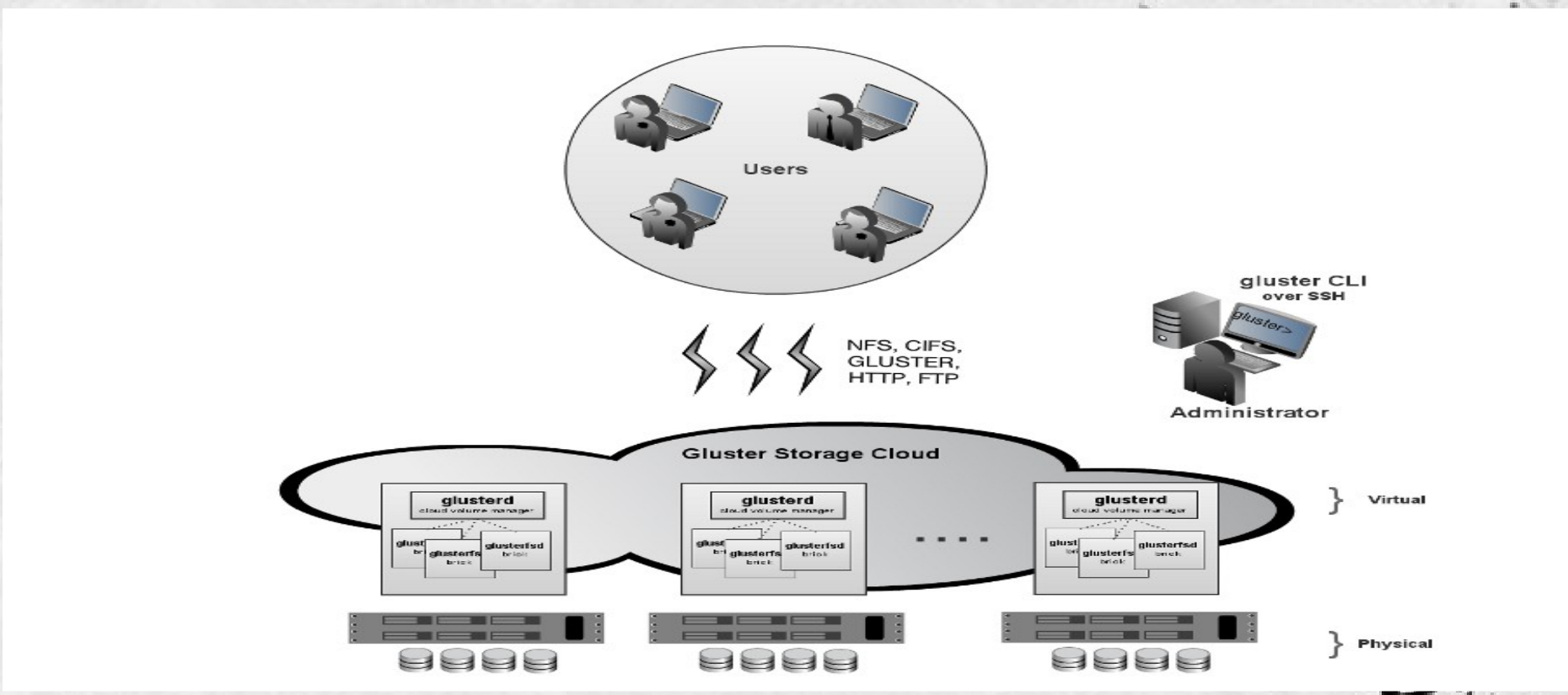
5. 弹性卷管理

数据储存在逻辑卷中，逻辑卷可以从虚拟化的物理存储池进行独立逻辑划分而得到。存储服务器可以在线进行增加和移除，不会导致应用中断。逻辑卷可以在所有配置服务器中增长和缩减，可以在不同服务器迁移进行容量均衡，或者增加和移除系统，这些操作都可在线进行。文件系统配置更改也可以实时在线进行并应用，从而可以适应工作负载条件变化或在线性能调优。

GNU/Linux-GFS

GFS 特点

5. 弹性卷管理



GNU/Linux-GFS

GFS 特点

6. 基于标准协议

Gluster 存储服务支持 NFS, CIFS, HTTP, FTP 以及 Gluster 原生协议，完全与 POSIX 标准兼容。现有应用程序不需要作任何修改或使用专用 API ，就可以对 Gluster 中的数据进行访问。这在公有云环境中部署 Gluster 时非常有用， Gluster 对云服务提供商专用 API 进行抽象，然后提供标准 POSIX 接口。

GNU/Linux-GFS

GFS 技术特点

1. 完全软件实现

2. 完整的存储操作系统栈

GlusterFS 不仅提供了一个分布式文件系统，而且还提供了许多其他重要的分布式功能，比如分布式内存管理、 I/O 调度、软 RAID 和自我修复等。GlusterFS 汲取了微内核架构的经验教训，借鉴了 GNU/Hurd 操作系统的设计思想，在用户空间实现了完整的存储操作系统栈。

GNU/Linux-GFS

GFS 技术特点

3. 用户空间实现 (User Space)

GlusterFS 在用户空间实现，这使得其安装和升级特别简便。另外，这也极大降低了普通用户基于源码修改 GlusterFS 的门槛，仅仅需要通用的 C 程序设计技能，而不需要特别的内核编程经验。

GNU/Linux-GFS

GFS 技术特点

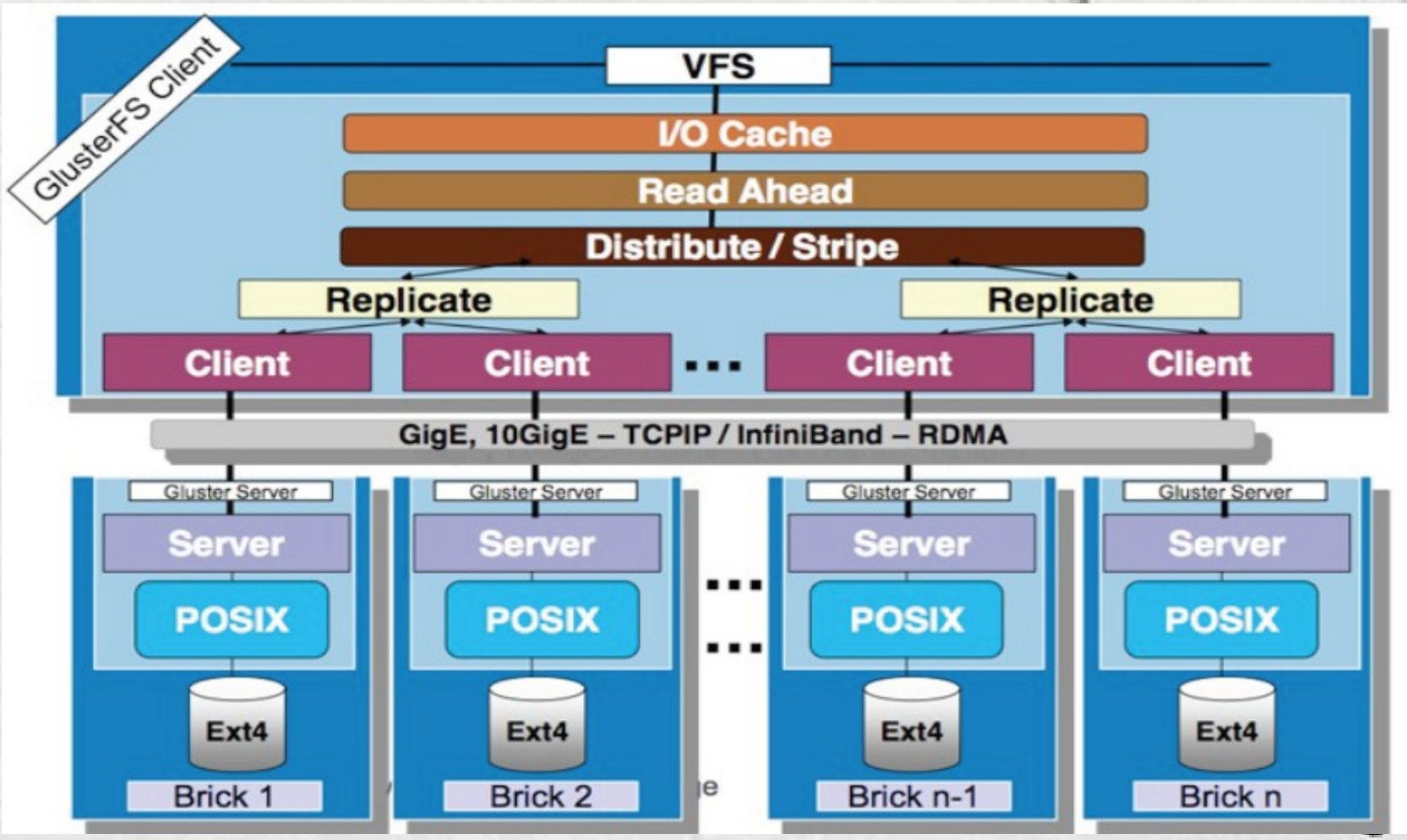
4. 模块化堆栈式架构

GlusterFS 采用模块化、堆栈式的架构，可通过灵活的配置支持高度定制化的应用环境，比如大文件存储、海量小文件存储、云存储、多传输协议应用等。每个功能以模块形式实现，然后以积木方式进行简单的组合，即可实现复杂的功能。比如：

Replicate 模块可实现 RAID1 ，Stripe 模块可实现 RAID0 ，通过两者的组合可实现 RAID10 和 RAID01 ，同时获得高性能和高可靠性。

GNU/Linux-GFS

GFS 模块化堆栈式设计



GNU/Linux-GFS

GFS 技术特点

5. 原始数据格式存储

GlusterFS 以原始数据格式（如 EXT3、 EXT4、 XFS、 ZFS ）储存数据，并实现多种数据自动修复机制。因此，系统极具弹性，即使离线情形下文件也可以通过其他标准工具进行访问。如果用户需要从 GlusterFS 中迁移数据，不需要作任何修改仍然可以完全使用这些数据。

GNU/Linux-GFS

GFS 技术特点

6. 无元数据服务设计

对 Scale-Out 存储系统而言，最大的挑战之一就是记录数据逻辑与物理位置的映像关系，即数据元数据，可能还包括诸如属性和访问权限等信息。传统分布式存储系统使用集中式或分布式元数据服务来维护元数据，集中式元数据服务会导致单点故障和性能瓶颈问题，而分布式元数据服务存在性能负载和元数据同步一致性问题。特别是对于海量小文件的应用，元数据问题是个非常大的挑战。

GNU/Linux-GFS

GFS 技术特点

6. 无元数据服务设计

GlusterFS 独特地采用无元数据服务的设计，取而代之使用算法来定位文件，元数据和数据没有分离而是一起存储。集群中的所有存储系统服务器都可以智能地对文件数据分片进行定位，仅仅根据文件名和路径并运用算法即可，而不需要查询索引或者其他服务器。这使得数据访问完全并行化，从而实现真正的线性性能扩展。无元数据服务器极大提高了 GlusterFS 的性能、可靠性和稳定性。

GNU/Linux-GFS

GFS 技术特点

7. Translators

Translators 是 GlusterFS 提供的一种强大文件系统功能扩展机制，这一设计思想借鉴于 GNU/Hurd 微内核操作系统。 GlusterFS 中所有的功能都通过 Translator 机制实现，运行时以动态库方式进行加载，服务端和客户端相互兼容。 GlusterFS 3.1.X 中，主要包括以下几类 Translator ：

GNU/Linux-GFS

GFS 技术特点

1. Clustet: 存储集群分布，目前有 AFR, DHT, Stripe 三种方式
2. Debug: 跟踪 GlusterFS 内部函数和系统调用
3. Encryption: 简单的数据加密实现
4. Features: 访问控制、锁、 Mac 兼容、静默、配额、只读、回收站等
5. Mgmt: 弹性卷管理
6. Mount:FUSE 接口实现
7. Nfs: 内部 NFS 服务器

GNU/Linux-GFS

GFS 技术特点

8. Performance ： io-cache, io-threads, quick-read, read-ahead, stat-prefetch, sysmlink-cache, write-behind 等性能优化

9. Protocol: 服务器和客户端协议实现

10.Storage: 底层文件系统 POSIX 接口实现

GNU/Linux-GFS

GFS Cluster Translators:

属于 GFS 集群存储核心，其包括：

DHT(Distributed Hash Table)

AFR(Automatic File Replication)

Stripe

GNU/Linux-GFS

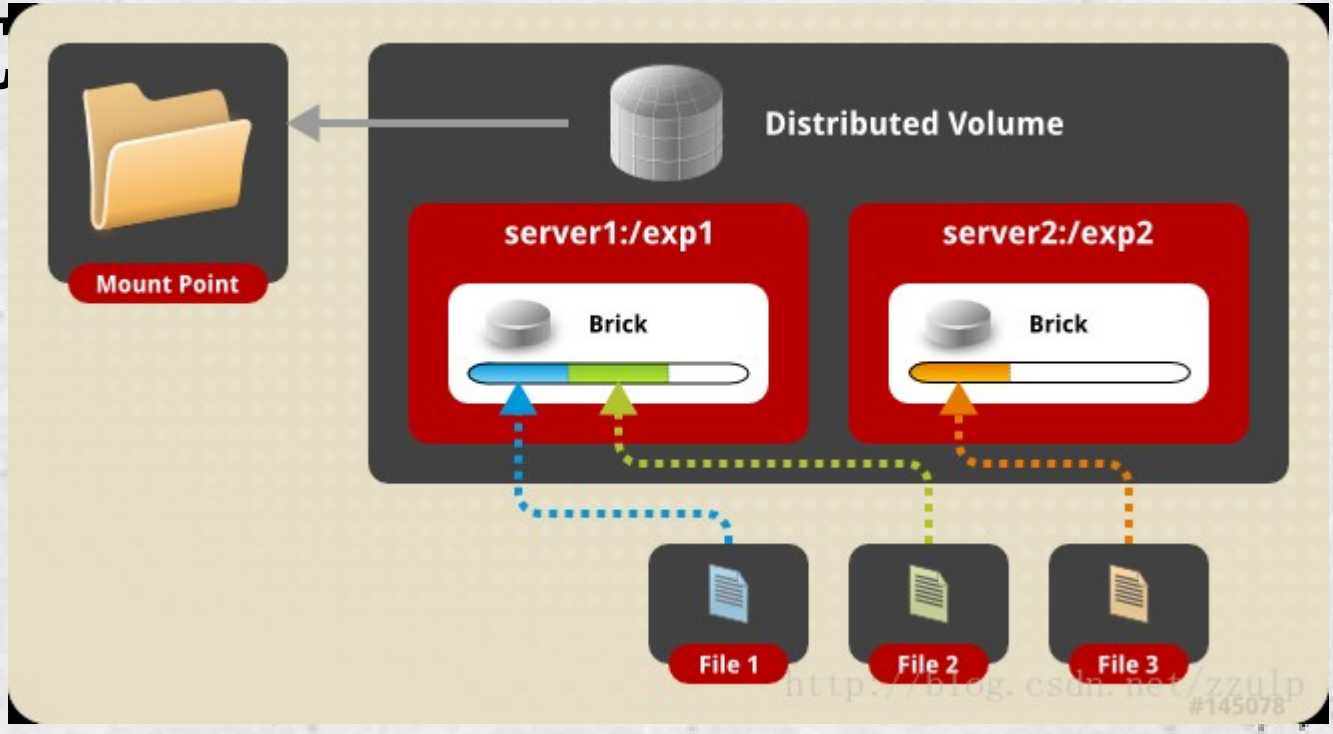
GFS Cluster Translators:

DHT 即上面所介绍的弹性哈希算法，它采用 hash 方式进行数据分布，名字空间分布在所有节点上。查找文件时，通过弹性哈希算法进行，不依赖名字空间。但遍历文件目录时，则实现较为复杂和低效，需要搜索所有的存储节点。单一文件只会调度到唯一的存储节点，一旦文件被定位后，读写模式相对简单。 DHT 不具备容错能力，需要借助 AFR 实现高可用性。

GNU/Linux-GFS

GFS Cluster Translators:

分布卷可以将某个文件随机的存储在卷内的一个brick(GFS 存储单元）内，通常用于扩展存储能力，不支持数据的冗余。除非底层的 brick 使用 RAID 等外部的冗



GNU/Linux-GFS

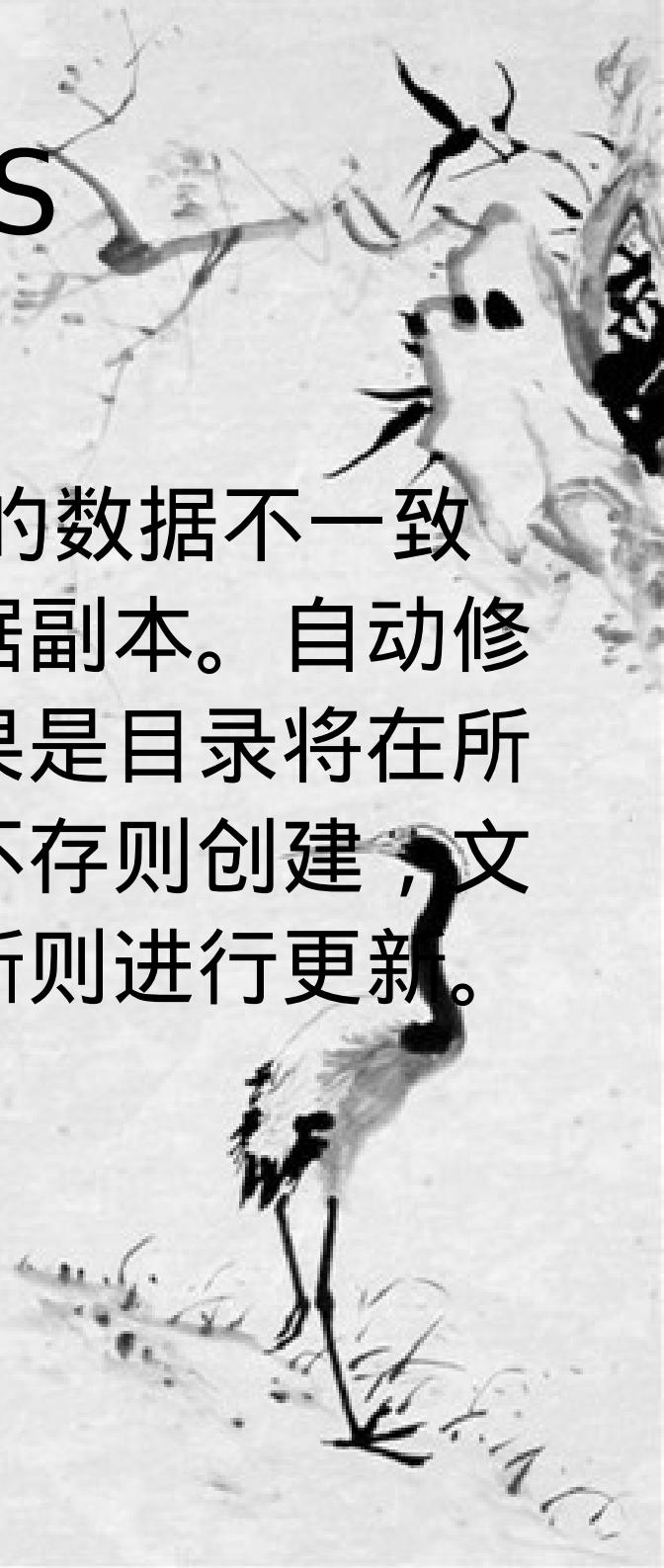
GFS Cluster Translators:

AFR 相当于 RAID1 ，同一文件在多个存储节点上保留多份，主要用于实现高可用性以及数据自动修复。 AFR 所有子卷上具有相同的名字空间，查找文件时从第一个节点开始，直到搜索成功或最后节点搜索完毕。读数据时， AFR 会把所有请求调度到所有存储节点，进行负载均衡以提高系统性能。写数据时，首先需要在所有锁服务器上对文件加锁，默认第一个节点为锁服务器，可以指定多个。然后， AFR 以日志事件方式对所有服务器进行写数据操作，成功后删除日志并解锁。

GNU/Linux-GFS

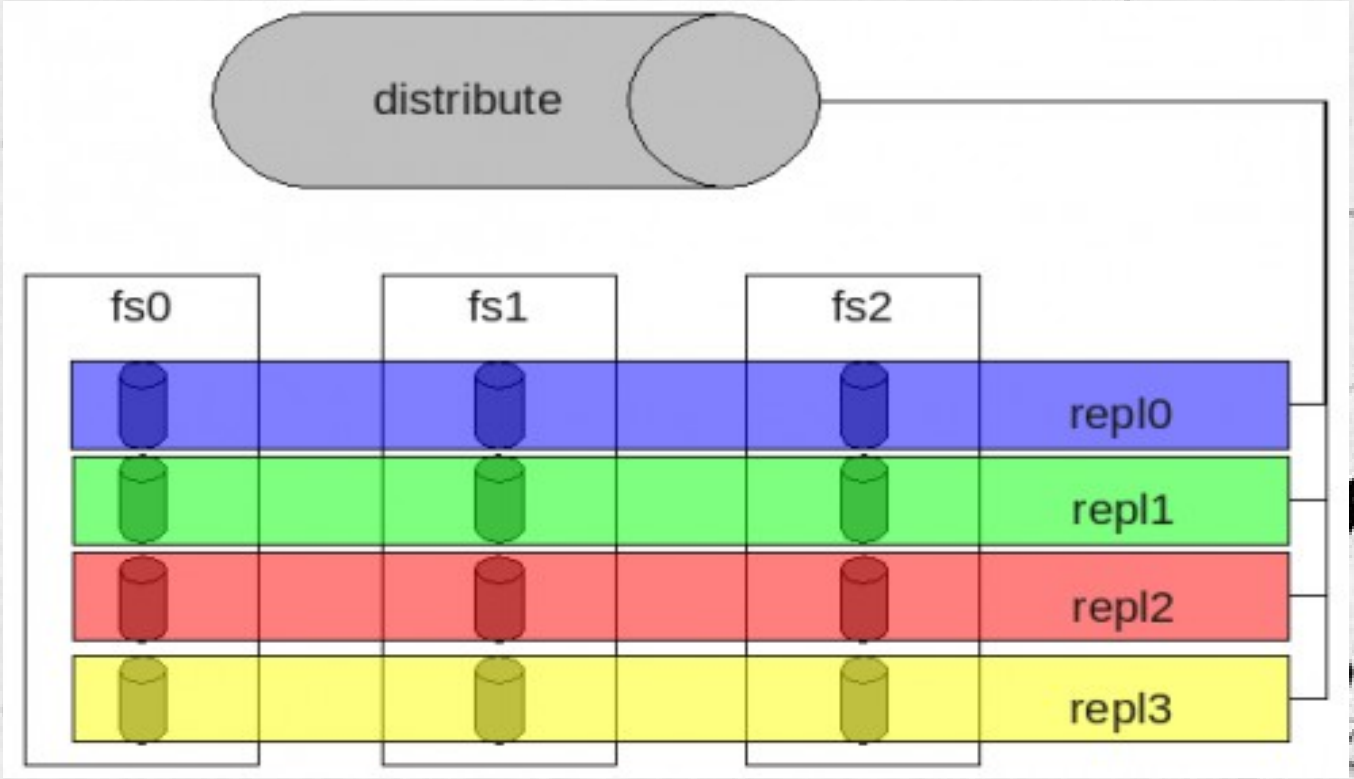
GFS Cluster Translators:

AFR 会自动检测并修复同一文件的数据不一致性，它使用更改日志来确定好的数据副本。自动修复在文件目录首次访问时触发，如果是目录将在所有子卷上复制正确数据，如果文件不存则创建，文件信息不匹配则修复，日志指示更新则进行更新。



GNU/Linux-GFS

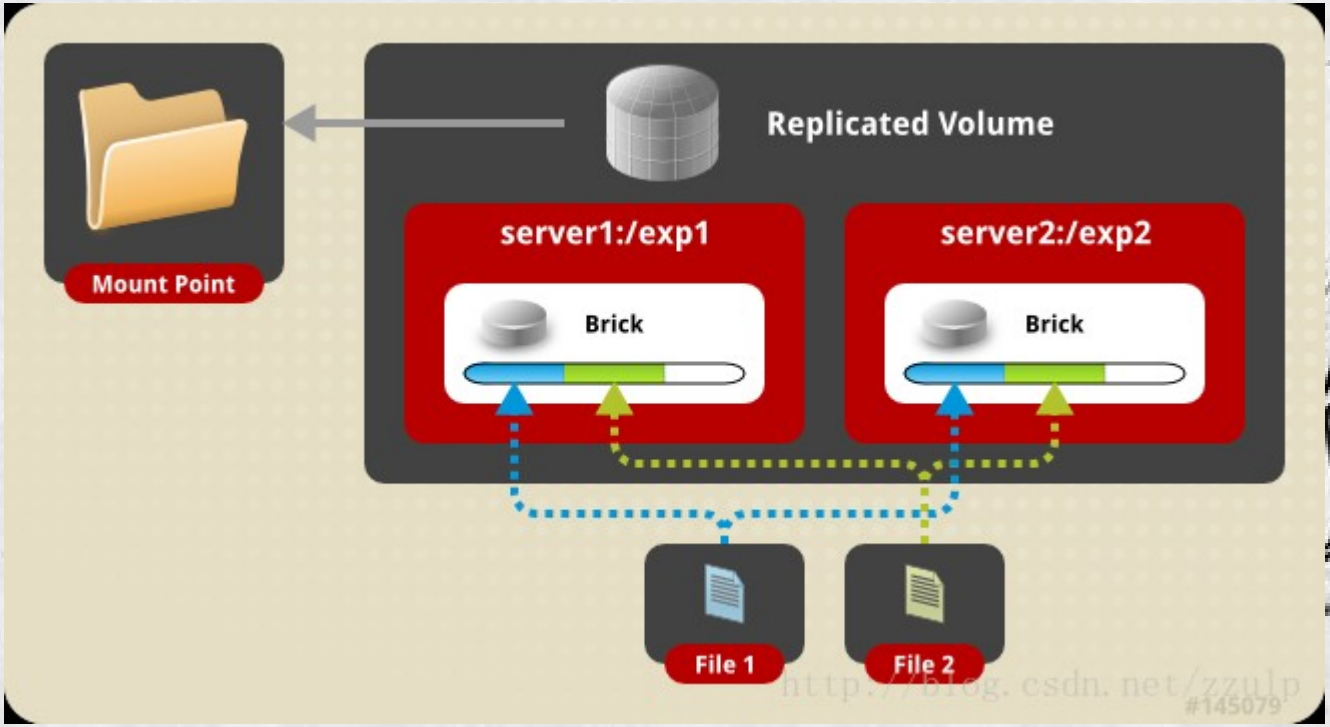
GFS Cluster Translators:
DHT+AFR



GNU/Linux-GFS

GFS Cluster Translators:

复本卷在创建时可指定复本的数量，复本在存储时会在卷的不同 brick 上，因此有几个复本就必须提供至少多个 brick。

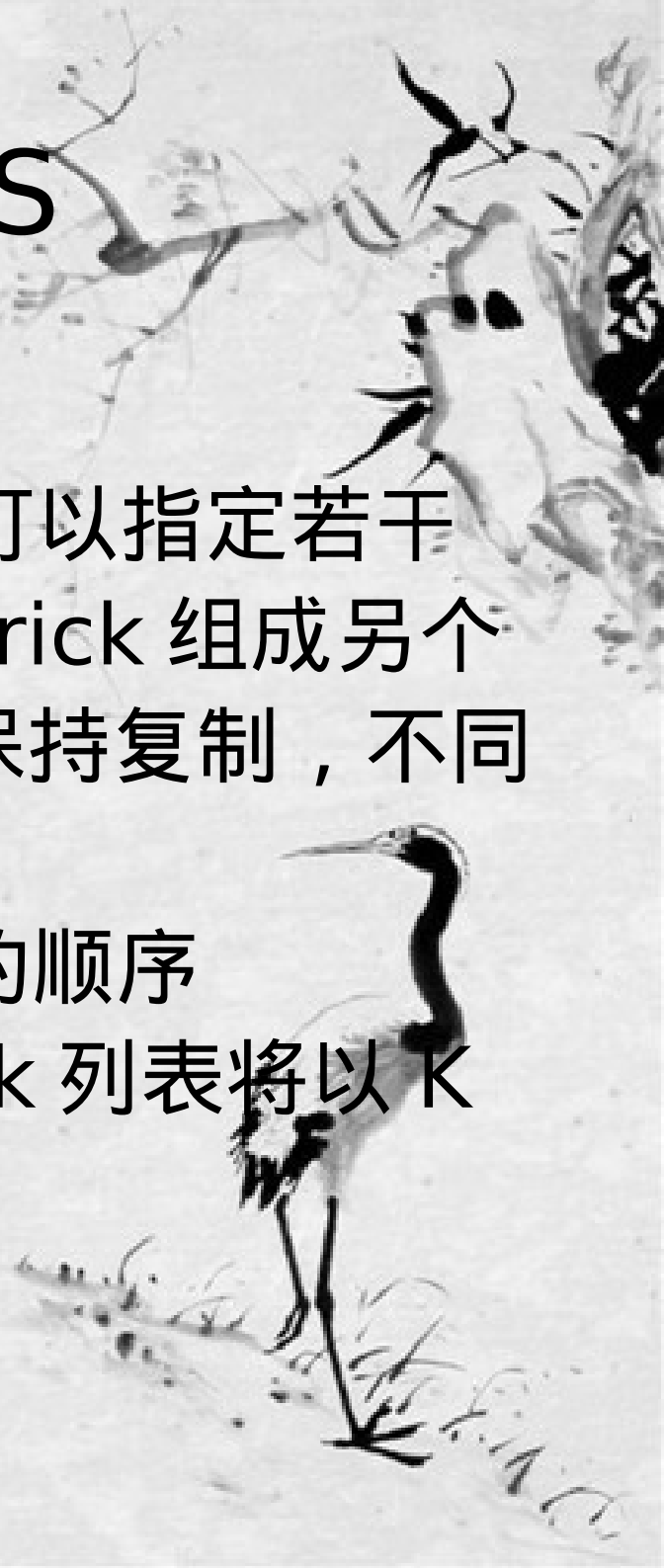


GNU/Linux-GFS

GFS Cluster Translators:

　　此类型卷是基本复本卷的扩展，可以指定若干 brick 组成一个复本卷，另外若干 brick 组成另个复本卷。单个文件在复本卷内数据保持复制，不同文件在不同复本卷之间进行分布。

　　复本卷的组成依赖于指定 brick 的顺序 brick 必须为复本数 K 的 N 倍 ,brick 列表将以 K 个为一组，形成 N 个复本卷



GNU/Linux-GFS

GFS Cluster Translators:

注意：在创建复本卷时， brick 数量与复本个数必须相等；否则将会报错。

另外如果同一个节点提供了多个 brick ，也可以在同一个结点上创建复本卷，但这并不安全，因为一台设备挂掉，其上面的所有 brick 将无法访问。

GNU/Linux-GFS

GFS Cluster Translators:

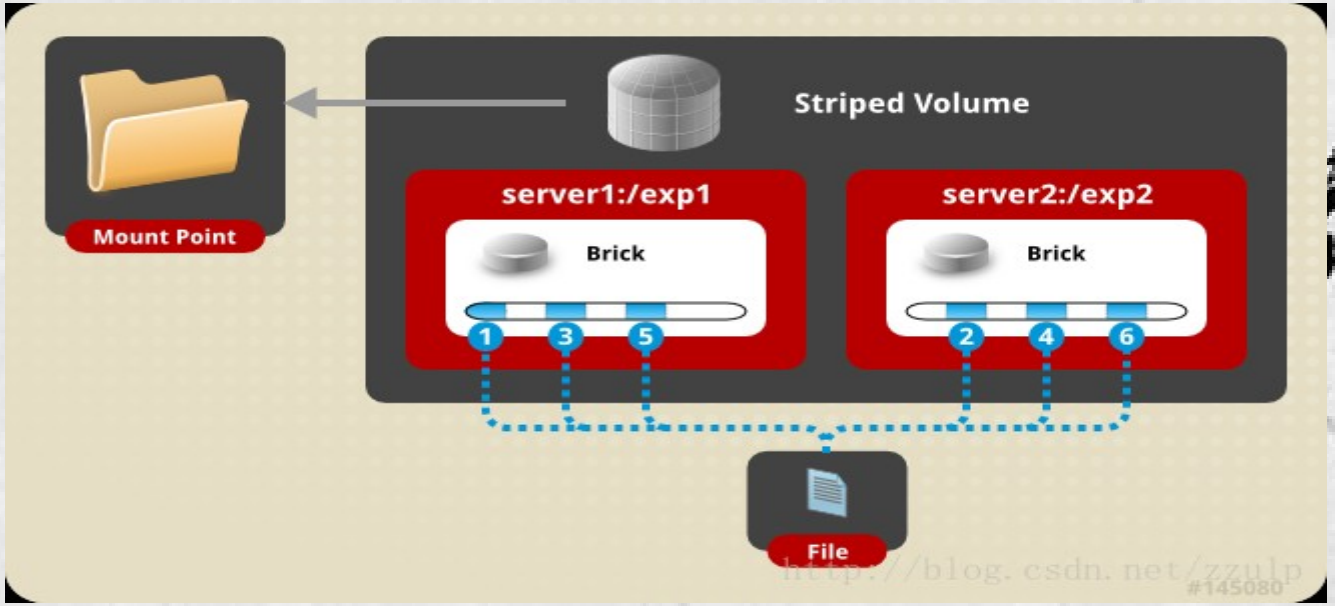
Stripe 相当于 RAID0 ，即分片存储，文件被划分成固定长度的数据分片以 Round-Robin 轮转方式存储在所有存储节点。 Stripe 所有存储节点组成完整的名字空间，查找文件时需要询问所有节点，这点非常低效。读写数据时， Stripe 涉及全部分片存储节点，操作可以在多个节点之间并发执行，性能非常高。 Stripe 通常与 AFR 组合使用，构成 RAID10/RAID01 ，同时获得高性能和高可用性，当然存储利用率会低于 50%。

GNU/Linux-GFS

GFS Cluster Translators:

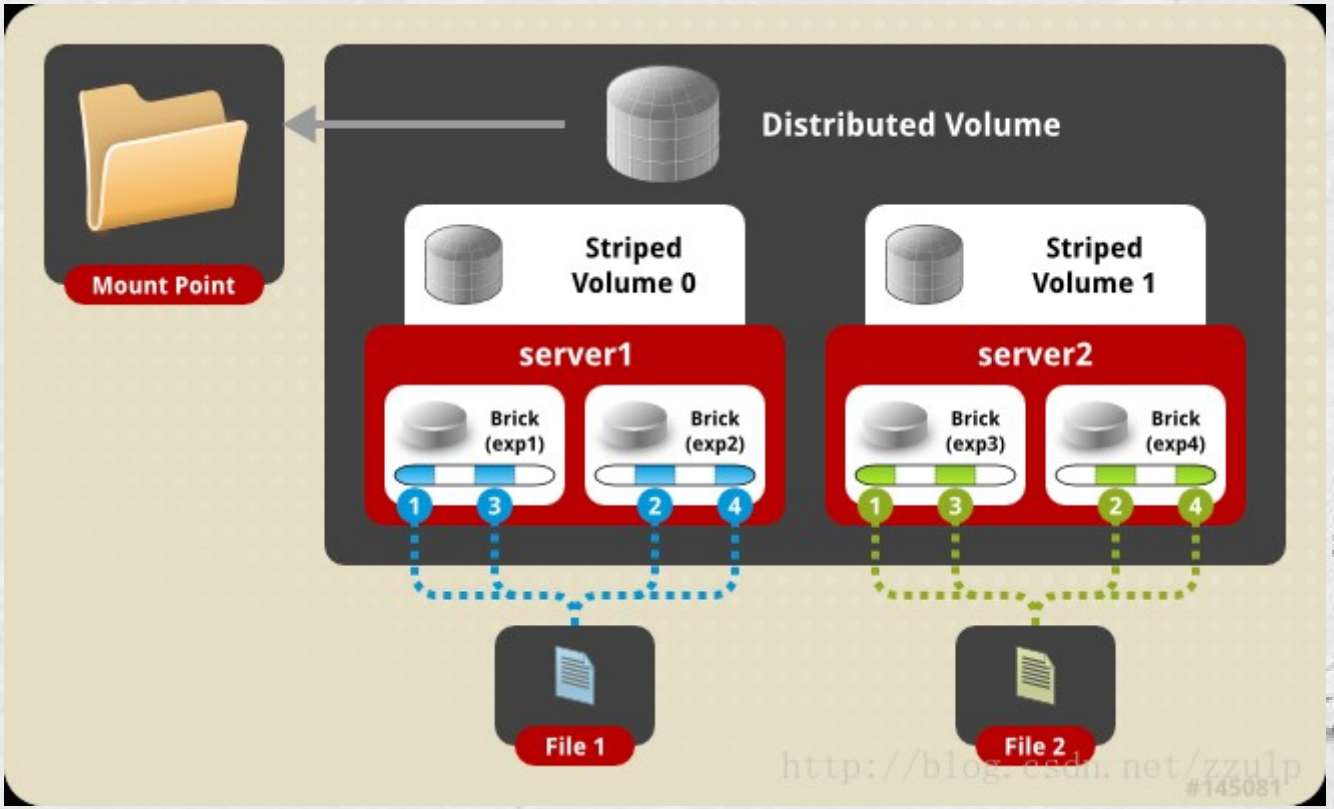
分片卷将单个文件分成小块（块大小支持配置，默认为 128K），然后将小块存储在不同的 brick 上，以提升文件的访问性能。

注意： brick 的个数必须等于分布位置的个数



GNU/Linux-GFS

GFS Cluster Translators:
Distribute+Striped



GNU/Linux-GFS

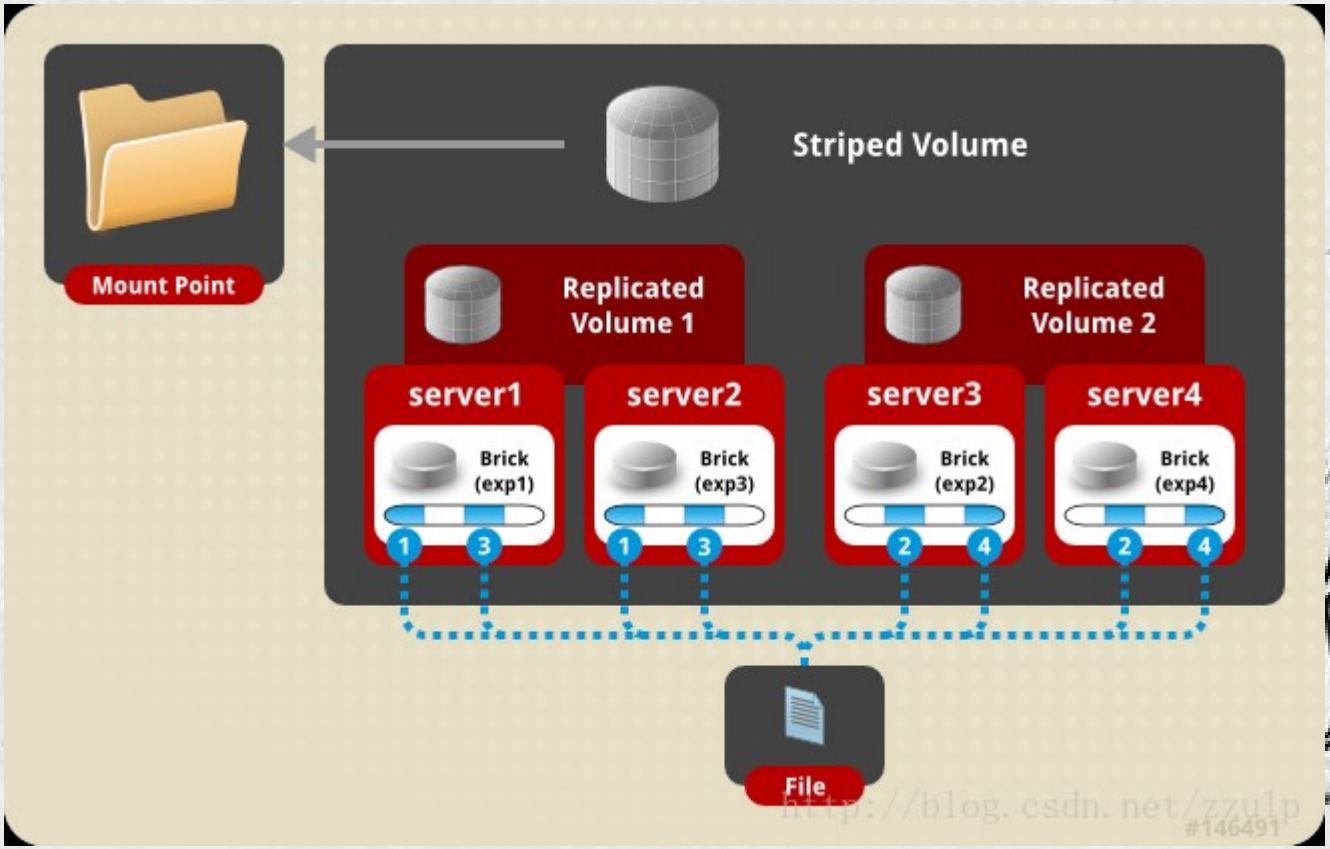
GFS Cluster Translators: Distribute+Striped

类似于分布式复本卷，若创建的卷的节点提供的 bricks 个数为 stripe 个数 N 倍时，将创建此类型的卷。

注意：切片卷的组成依赖于指定 brick 的顺序，brick 必须为复本数 K 的 N 倍 ,brick 列表将以 K 个为一组，形成 N 个切片卷

GNU/Linux-GFS

GFS Cluster Translators:
Striped+Replicated



GNU/Linux-GFS

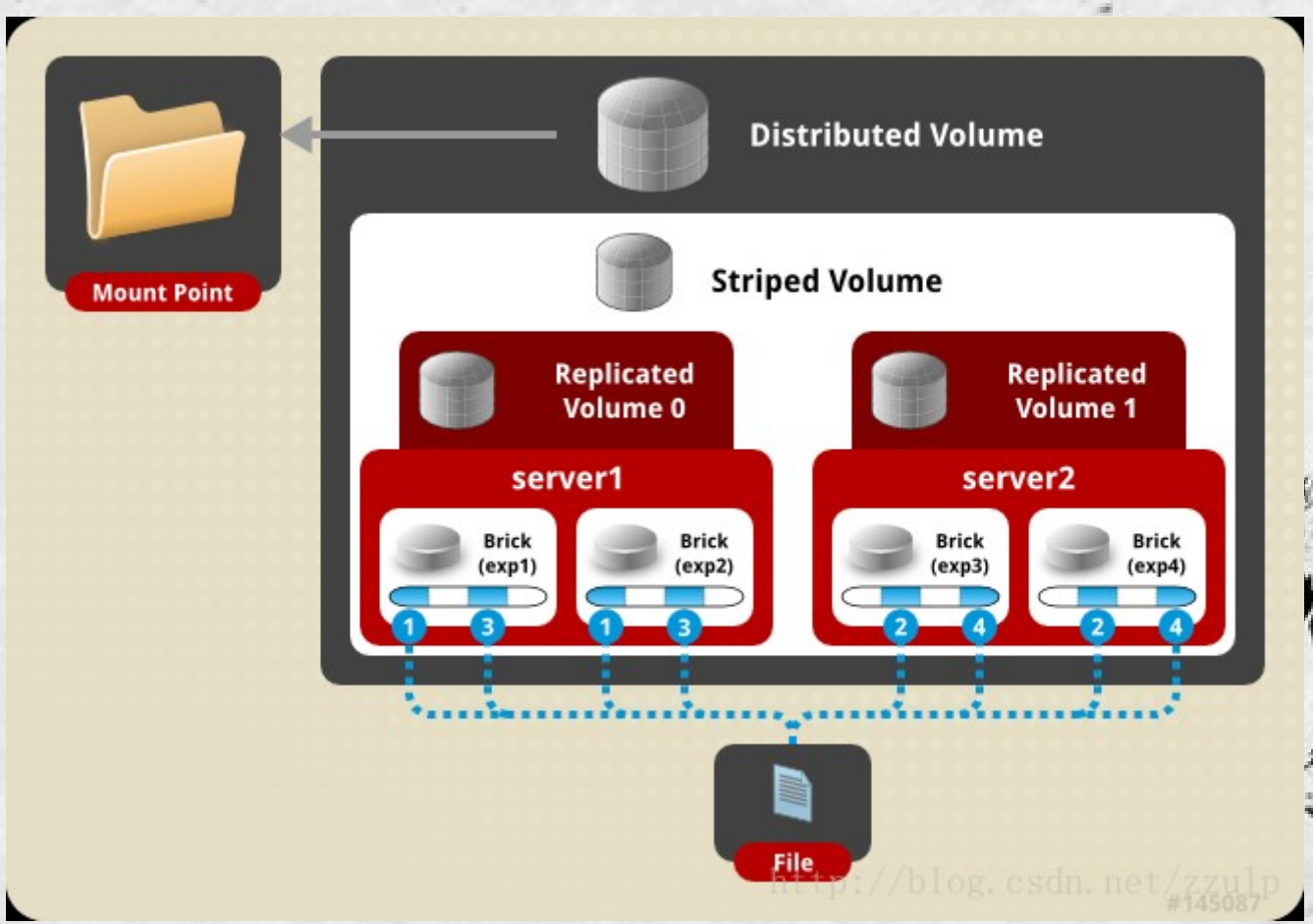
GFS Cluster Translators: Distribute+Striped

数据将进行切片，切片在复本卷内进行复制，在不同卷间进行分布。 exp1 和 exp2 组成复本卷，exp3 和 exp4 组成复本卷，两个复本卷组成分片卷。

注意： brick 数量必须和 stripe 个数 N 和 repl 参数 M 的积 N*M 相等。即对于 brick 列表，将以 M 为一组，形成 N 个切片卷。数据切片分布在 N 个切片卷上，在每个切片卷内部，切片数据复本 M 份。

GNU/Linux-GFS

GFS Cluster Translators:
Distribute+Striped+Replicated



GNU/Linux-GFS

GFS Cluster Translators: Distribute+Striped+Replicated

注意： bricks 数量为 stripe 个数 N ，和 repl 个数 M 的积 N*M 的整数倍。

exp1 exp2 exp3 exp4 组成一个分布卷， exp1 和 exp2 组成一个 stripe 卷， exp3 和 exp4 组成另一个 stripe 卷， 1 和 2 ， 3 和 4 互为复本卷。

GNU/Linux-GFS

GFS 术语

Brick:GFS 中的存储单元，通过是一个受信存储池中的服务器的一个导出目录。可以通过主机名和目录名来标识，如 'SERVER:EXPORT'

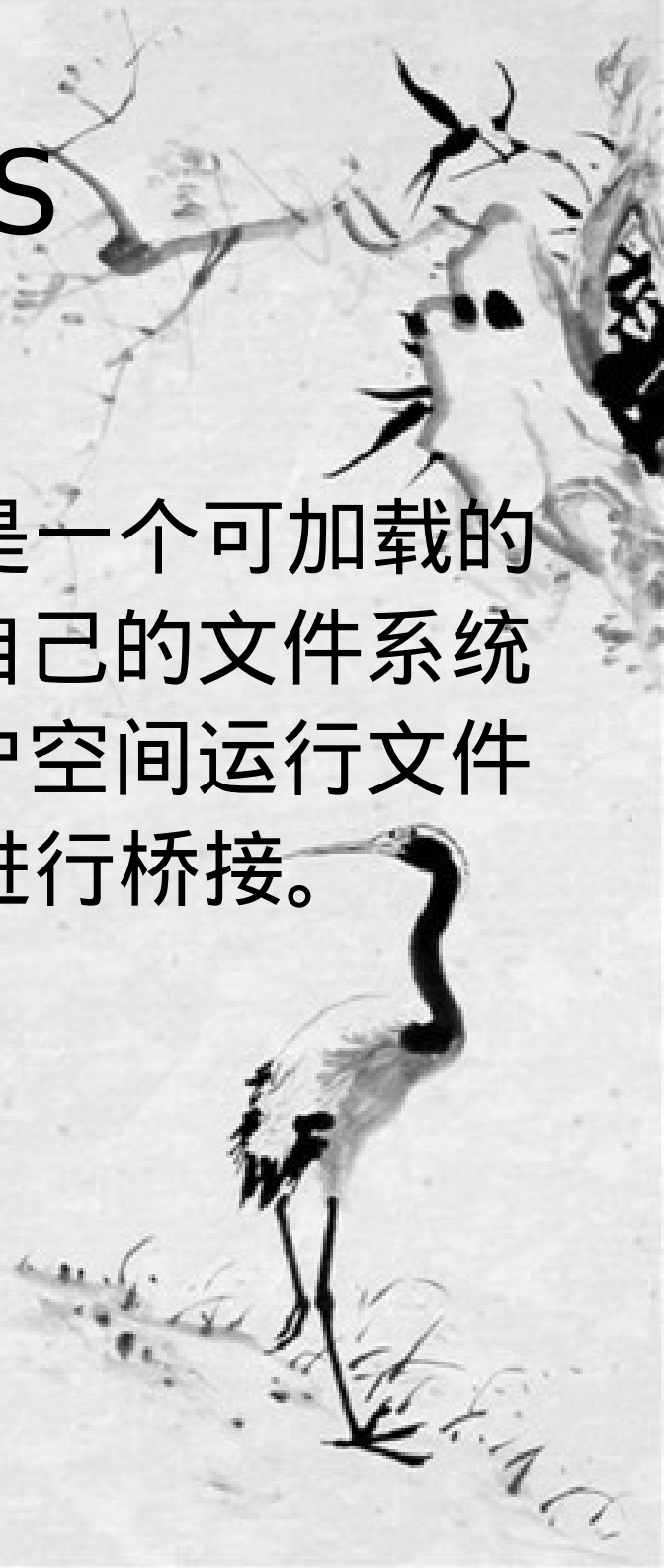
Client: 挂载了 GFS 卷的设备

Extended Attributes:xattr 是一个文件系统的特性，其支持用户或程序关联文件 / 目录和元数据。

GNU/Linux-GFS

GFS 术语

FUSE:Filesystem Userspace 是一个可加载的内核模块，其支持非特权用户创建自己的文件系统而不需要修改内核代码。通过在用户空间运行文件系统的代码通过 FUSE 代码与内核进行桥接。



GNU/Linux-GFS

GFS 术语

Geo-Replication

GFID:GFS 卷中的每个文件或目录都有一个唯一的 128 位的数据相关联，其用于模拟 inode

Namespace: 每个 Gluster 卷都导出单个 ns 作为 POSIX 的挂载点

Node: 一个拥有若干 brick 的设备

GNU/Linux-GFS

GFS 术语

RDMA: 远程直接内存访问，支持不通过双方的 OS 进行直接内存访问。

RRDNS:round robin DNS 是一种通过 DNS 轮转返回不同的设备以进行负载均衡的方法

Self-heal: 用于后台运行检测复本卷中文件和目录的不一致性并解决这些不一致。

GNU/Linux-GFS

GFS 术语

Translator:

Volfile:glusterfs 进程的配置文件，通常位于 /var/lib/glusterd/vols/volname

Volume: 一组 bricks 的逻辑集合

GNU/Linux-GFS

所有节点环境建立操作

1. 启用 GFS 源

#curl http://download.gluster.org/pub/gluster/glusterfs/LATEST/EPEL.repo/glusterfs-epel.repo -o /etc/yum.repos.d/glusterfs-epel.repo

EPEL: 企业版 Linux 附件软件包

GNU/Linux-GFS

所有节点环境建立操作

2. 安装 GFS SERVER

#yum install glusterfs-server -y

3. 开启 GFS 服务

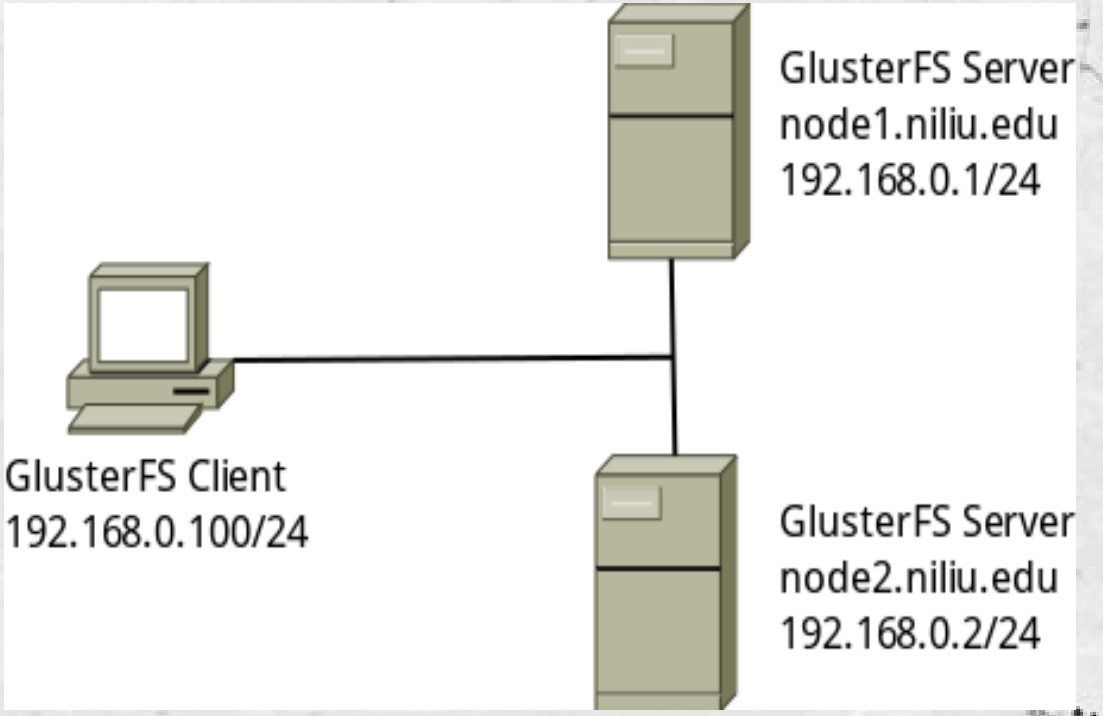
#systemctl start glusterd

#systemctl enable glusterd

4. 确认 DNS 能够解析各 node 或在 /etc/hosts 配置各 node 的 FQDN、IP、别名并通畅。

GNU/Linux-GFS

实现 GFS 之 Distributed



GNU/Linux-GFS

实现 GFS 之 Distributed

1. 建立 GlusterFS 卷所需要的目录

   #mkdir -pv /gfs/dist

2. 将 node2 加入至 gfs 集群组中

   #gluster peer probe node2.niliu.edu

3. 确认集群组状态

   #gluster peer status

GNU/Linux-GFS

实现 GFS 之 Distributed

4. 创建 Distributed Volume

```
#gluster volume create vol_dist transport tcp \
node1:/gfs/dist \
node2:/gfs/dist force
```

5. 启动卷

```
#gluster volume start vol_dist
```

6. 查看卷信息

```
#gluster volume info
```

GNU/Linux-GFS

实现 GFS 之 Distributed

7. 客户端配置

#wget \
http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-epel.repo -O /etc/yum.repos.d/glusterfs-epel.repo

#yum install glusterfs glusterfs-fuse -y

GNU/Linux-GFS

实现 GFS 之 Distributed

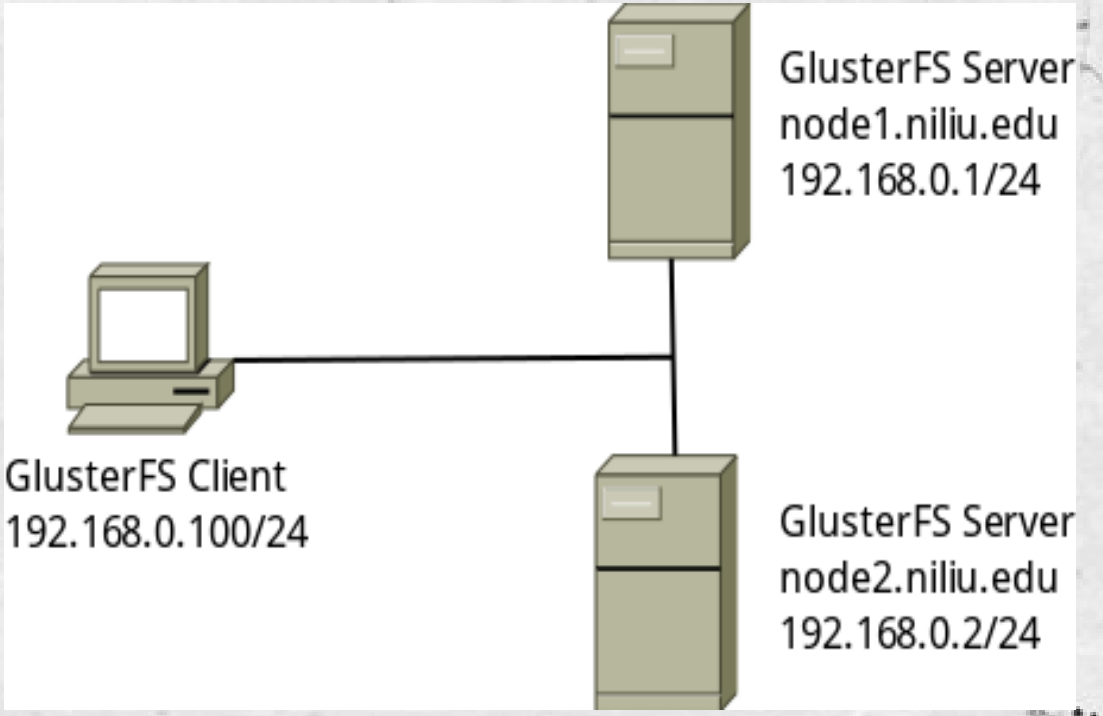
7. 客户端配置

```
#mkdir -pv /mnt/gfs/dist
#mount -t glusterfs \
node1.niliu.edu:/vol_dist /mnt/gfs/dist
#df -hT
```

//* 复制或创建文件进行测试 *//

GNU/Linux-GFS

实现 GFS 之 Replication



GNU/Linux-GFS

实现 GFS 之 Replication

1. 建立 GlusterFS 卷所需要的目录

   #mkdir -pv /gfs/replica

2. 将 node2 加入至 gfs 集群组中

   #gluster peer probe node2.niliu.edu

3. 确认集群组状态

   #gluster peer status

GNU/Linux-GFS

实现 GFS 之 Replication

4. 创建 Replication Volume

```
#gluster volume create vol_replica replica 2 transport tcp node1:/gfs/replica  \
node2:/gfs/replica force
```

5. 启动并查看卷信息

```
#gluster volume start vol_replica
#gluster volume info
```

GNU/Linux-GFS

实现 GFS 之 Replication

7. 客户端配置

#wget \
http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-epel.repo -O /etc/yum.repos.d/glusterfs-epel.repo

#yum install glusterfs glusterfs-fuse -y

GNU/Linux-GFS

实现 GFS 之 Replication

7. 客户端配置

#mkdir -pv /mnt/gfs/replica

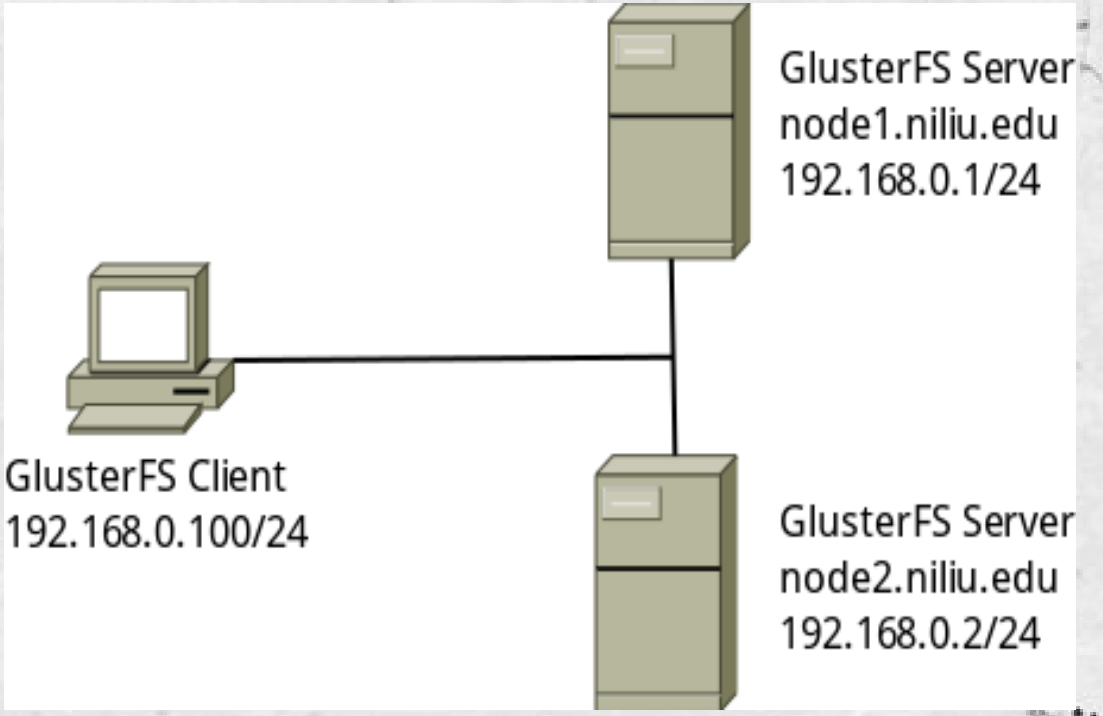
#mount -t glusterfs \
node1.niliu.edu:/vol_replica /mnt/gfs/replica

#df -hT

//* 复制或创建文件进行测试 *//

GNU/Linux-GFS

实现 GFS 之 Striping



GNU/Linux-GFS

实现 GFS 之 Striping

1. 建立 GlusterFS 卷所需要的目录

   #mkdir -pv /gfs/spriped

2. 将 node2 加入至 gfs 集群组中

   #gluster peer probe node2.niliu.edu

3. 确认集群组状态

   #gluster peer status

GNU/Linux-GFS

实现 GFS 之 Striping

4. 创建 Striping Volume

```
#gluster volume create vol_spriped stripe 2 transport tcp node1:/gfs/spriped  \
node2:/gfs/spriped force
```

5. 启动并查看卷信息

```
#gluster volume start vol_spriped
#gluster volume info
```

GNU/Linux-GFS

实现 GFS 之 Striping

7. 客户端配置

#wget \
http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-epel.repo -O /etc/yum.repos.d/glusterfs-epel.repo

#yum install glusterfs glusterfs-fuse -y

GNU/Linux-GFS

实现 GFS 之 Striping

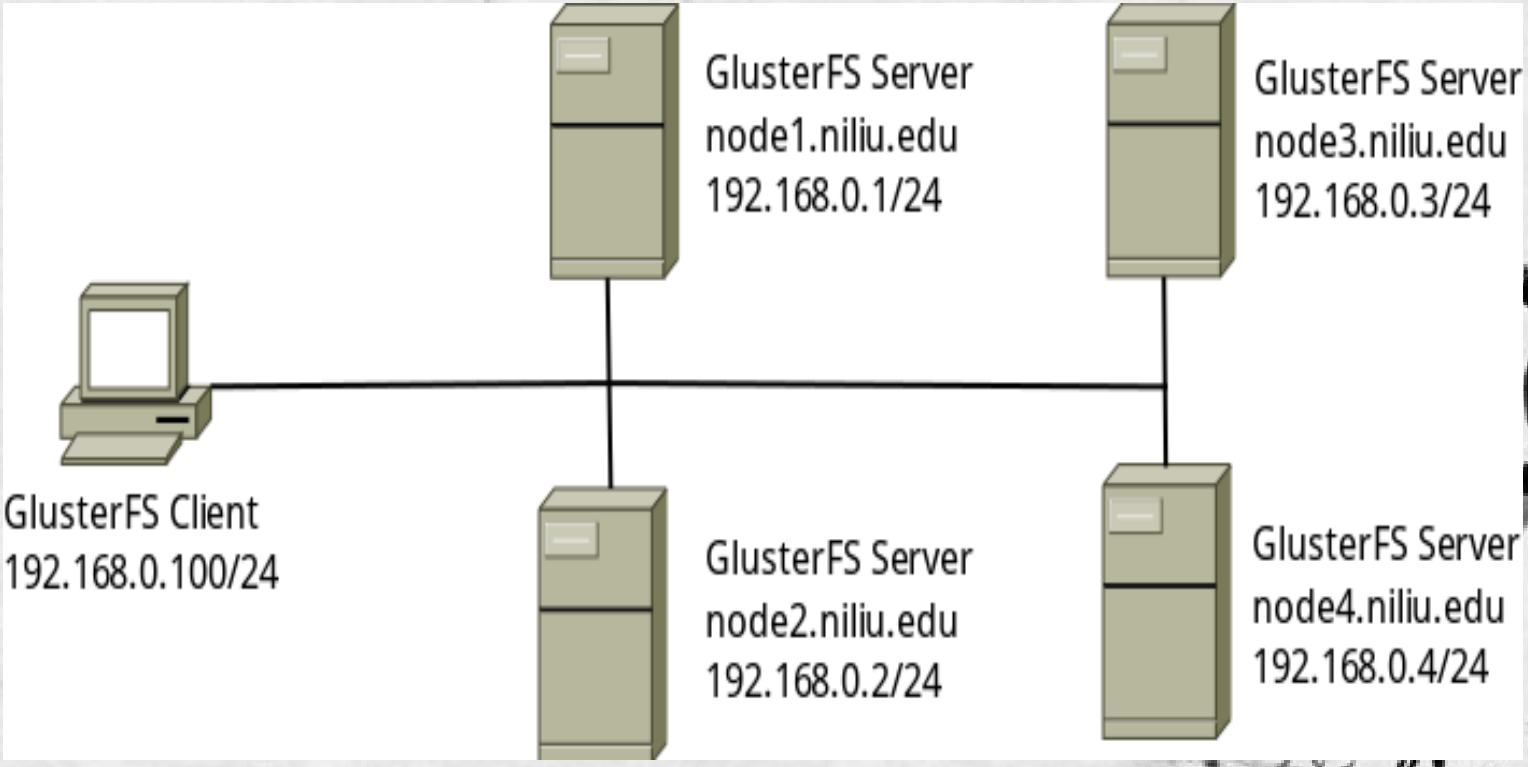
7. 客户端配置

```
#mkdir -pv /mnt/gfs/replica
#mount -t glusterfs \
node1.niliu.edu:/vol_spriped
/mnt/gfs/replica
#df -hT
```

//* 复制或创建文件进行测试 *//

GNU/Linux-GFS

实现 GFS 之 Dist+Replica



GNU/Linux-GFS

实现 GFS 之 Dist+Replica

1. 建立 GlusterFS 卷所需要的目录

```
#mkdir -pv /gfs/dr
```

2. 将 node2 加入至 gfs 集群组中

```
#gluster peer probe node2.niliu.edu
#gluster peer probe node3.niliu.edu
#gluster peer probe node4.niliu.edu
```

3. 确认集群组状态

```
#gluster peer status
```

GNU/Linux-GFS

实现 GFS 之 Dist+Replica

4. 创建 Dist+Replica Volume

```
#gluster volume create vol_dr replica 2 transport tcp node1:/gfs/dr \
node2:/gfs/dr node3:/gfs/dr node4:/gfs/dr force
```

5. 启动并查看卷信息

```
#gluster volume start vol_dr
#gluster volume info
```

GNU/Linux-GFS

实现 GFS 之 Dist+Replica

7. 客户端配置

#wget \
http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-epel.repo -O /etc/yum.repos.d/glusterfs-epel.repo

#yum install glusterfs glusterfs-fuse -y

GNU/Linux-GFS

实现 GFS 之 Dist+Replica

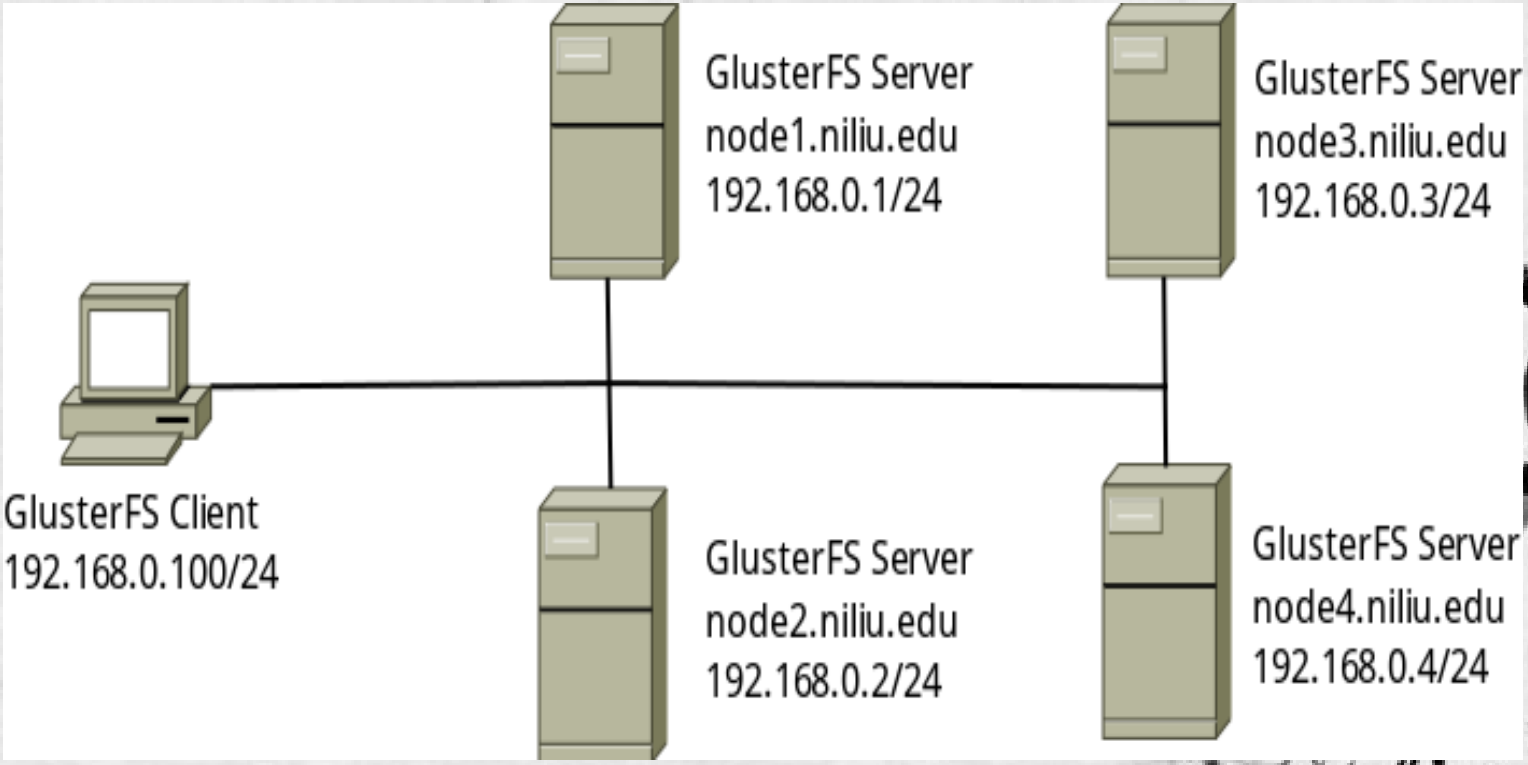
7. 客户端配置

```
#mkdir -pv /mnt/gfs/dr
#mount -t glusterfs \
node1.niliu.edu:/vol_dr /mnt/gfs/dr
#df -hT
```

//* 复制或创建文件进行测试 *//

GNU/Linux-GFS

实现 GFS 之 Stripe+Replica



GNU/Linux-GFS

实现 GFS 之 Stripe+Replica

1. 建立 GlusterFS 卷所需要的目录

```
#mkdir -pv /gfs/sr
```

2. 将 node2 加入至 gfs 集群组中

```
#gluster peer probe node2.niliu.edu
#gluster peer probe node3.niliu.edu
#gluster peer probe node4.niliu.edu
```

3. 确认集群组状态

```
#gluster peer status
```

GNU/Linux-GFS

实现 GFS 之 Stripe+Replica

4. 创建 Stripe+Replica Volume

#gluster volume create vol_sr stripe 2 replica 2 transport tcp node1:/gfs/dr \ node2:/gfs/dr node3:/gfs/dr node4:/gfs/dr force

5. 启动并查看卷信息

#gluster volume start vol_dr

#gluster volume info

GNU/Linux-GFS

实现 GFS 之 Stripe+Replica

7. 客户端配置

#wget \
http://download.gluster.org/pub/gluster/glusterfs/LATEST/CentOS/glusterfs-epel.repo -O /etc/yum.repos.d/glusterfs-epel.repo

#yum install glusterfs glusterfs-fuse -y

GNU/Linux-GFS

实现 GFS 之 Stripe+Replica

7. 客户端配置

#mkdir -pv /mnt/gfs/sr

#mount -t glusterfs \
node1.niliu.edu:/vol_sr /mnt/gfs/sr

#df -hT

//* 复制或创建文件进行测试 *//

GNU/Linux-GFS

GFS 其他操作

1. 查看集群组（存储池）状态

#gluster peer status

2. 将指定设备从集群组（存储池）删除

#gluster peer detach hostname|ip

3. 停止卷

#gluster volume stop 卷名称

GNU/Linux-GFS

GFS 其他操作

4. 删除卷

#gluster volume delete 卷名称

//* 卷状态查看

#gluster volume status [all|volname] [detail|clients|mem|fd|inode|callpoll]

GNU/Linux-GFS

GFS 其他操作

4. 删除卷

#gluster volume delete 卷名称

5. 收缩扩展卷

#gluster volume add-brick 卷名称 [strip|replica <count>] brick1...

#gluster volume remove-brick 卷名称 [strip|replica <count>] brick1...

//* 扩展或收缩卷时，也要按照卷的类型，加入或减少的 brick 个数必须满足相应的要求。 *//

GNU/Linux-GFS

GFS 其他操作

6. 收缩或扩展后，对卷的数据进行重新均衡

#gluster volume rebalane niliu start|stop|status

GNU/Linux-GFS

GFS 其他操作

//* 扩展 gfs

1) 本实验基于 Dist 类型完成 (s1,s3 为 niliu 卷)

2) 确认客户端当前挂载分区大小

GNU/Linux-GFS

GFS 其他操作

//* 扩展 gfs

3) 配置 s1

#glusterfs peer probe s4

#glusterfs peer status

#glusterfs volume add-brick niliu s4:/dfs/dist force

#glusterfs vol niliu status

#glusterfs volume rebalance niliu start

4) 确认客户端容量增加

GNU/Linux-GFS

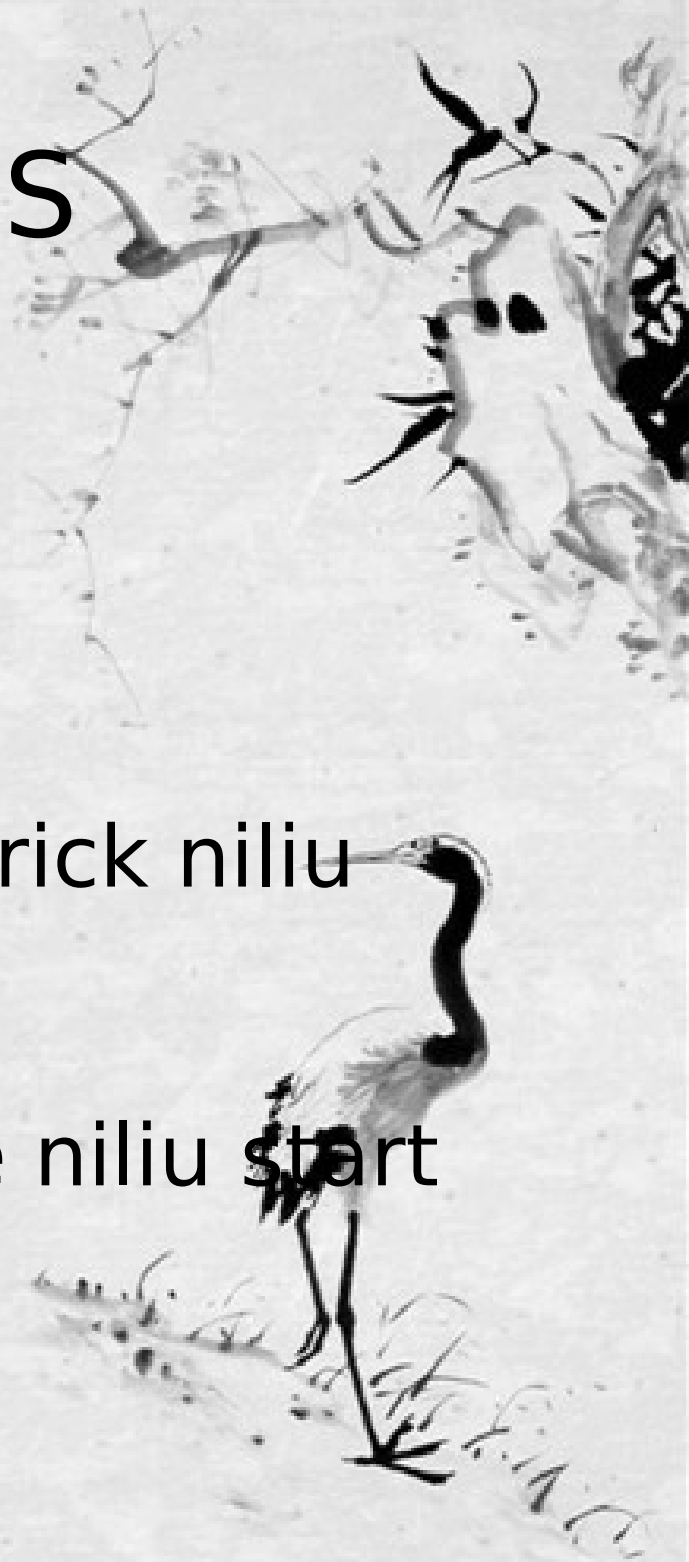
GFS 其他操作

//* 缩减 gfs

1) 本实验基于 Dist 类型完成 (s1,s3,s4 为 niliu 卷)

2) 确认客户端当前挂载分区大小

GNU/Linux-GFS



GFS 其他操作

//* 缩减 gfs

3) 配置 s1

#glusterfs peer status

#glusterfs volume remove-brick niliu s4:/dfs/dist force

#glusterfs vol niliu status

#glusterfs volume rebalance niliu start

#glusterfs peer detach s4

4) 确认客户端容量缩减

GNU/Linux-GFS

GFS 其他操作

7. 触发副本自愈

1) 只修复有问题的文件

#gluster volume heal 卷名称

2) 修复所有文件

#gluster volume heal 卷名称 full

GNU/Linux-GFS

GFS 其他操作

3) 查看自愈信息

#gluster volume heal 卷名称 info

4) 查看信息的其他方式

#gluster volume heal 卷名称 info healed|heal-failed|split-brain

GNU/Linux-GFS

GFS 其他操作

8. 迁移卷

主要作用：主要完成数据在卷之间的在线迁移

1. 语法格式

#gluster volume replace-brick niliu old-brick new-brick [start|pause|abort|status|commit]

GNU/Linux-GFS

GFS 其他操作

8. 迁移卷 - 示例

如将 niliu 卷中的 node3 替换为 node5

1) 启动迁移过程

#gluster peer probe node5

#gluster volume replace-brick niliu node3:/exp3 node5:/exp5 start

2) 暂停迁移过程

#gluster volume replace-brick niliu node3:/exp3 node5:/exp5 pause

GNU/Linux-GFS

GFS 其他操作

8. 迁移卷 - 示例

3) 中止迁移过程

#gluster volume replace-brick niliu node3:/exp3 node5:/exp5 abort

4) 查看迁移状态

#gluster volume replace-brick niliu node3:/exp3 node5:/exp5 status

GNU/Linux-GFS

GFS 其他操作

8. 迁移卷 - 示例

5) 迁移完成后提交完成

#gluster volume replace-brick niliu node3:/exp3 node5:/exp5 commit

GNU/Linux-GFS

GFS 其他操作

9. 客户端自动加载

#vim /etc/fstab

node1:/dp /mnt/dp glusterfs defaults,_netdev 0 0

GNU/Linux-GFS

GFS 其他操作

10. 性能监控

#gluster volume profile niliu start

#gluster volume profile info

#gluster volume profile niliu stop

11. 实时 top

1) 显示当前某个 brick 或 NFS 文件打开 / 读 / 写 / 打开目录 / 读目录的计数

#gluster volume top niliu {open|read|write|opendir|readdir} brick node1:/exp1 list-cnt 1

GNU/Linux-GFS

GFS 其他操作

11. 实时 top

2) 显示当前某个 brick 路径读文件或写文件数据的性能

#gluster volume top niliu read-perf|write-perf bs 256 count 10 brick node1:/exp1 list-cnt 1

GNU/Linux-GFS

GFS 其他操作

12. 内部计数导出

#gluster volume statedump niliu

13. 设置导出路径

#gluster volume set server .statedump-path /var/log/

14. 查看导出数据

#gluster volume info dumpfile