101 – 1
GlusterFS系统原理剖析

讲师介绍

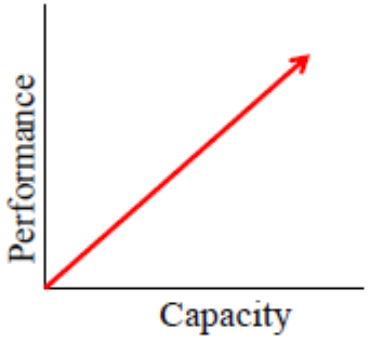
- 刘爱贵，博士
- 毕业于中科院高能物理研究所
- 主要研究方向：分布式存储、高性能计算、数据挖掘
- 专注于存储技术的研究与开发，GlusterFS等分布式文件系统资深理论研究与实践者
- 博客：http://blog.csdn.net/liuaigui
- 微博：http://weibo.com/liuag
- Email：aigui.liu@gmail.com
- QQ：9187434

培训提纲

- GlusterFS架构特点
- GlusterFS核心工作原理
- GlusterFS典型功能剖析

GlusterFS是什么？

GlusterFS架构设计目标



- **Elasticity**
  - Flexibility adapt to growth/reduction
  - Add, delete volumes & users
  - Without disruption
- **Scale linearly**
  - Multiple dimensions
    - Performance
    - Capacity
  - Aggregated resources
- **Eliminate metadata**
  - Improve file access speed
- **Simplicity**
  - Ease of management
  - No complex Kernel patches
  - Run in user space

GlusterFS架构特点

软件定义

无中心架构

全局命名空间

高性能

用户空间实现

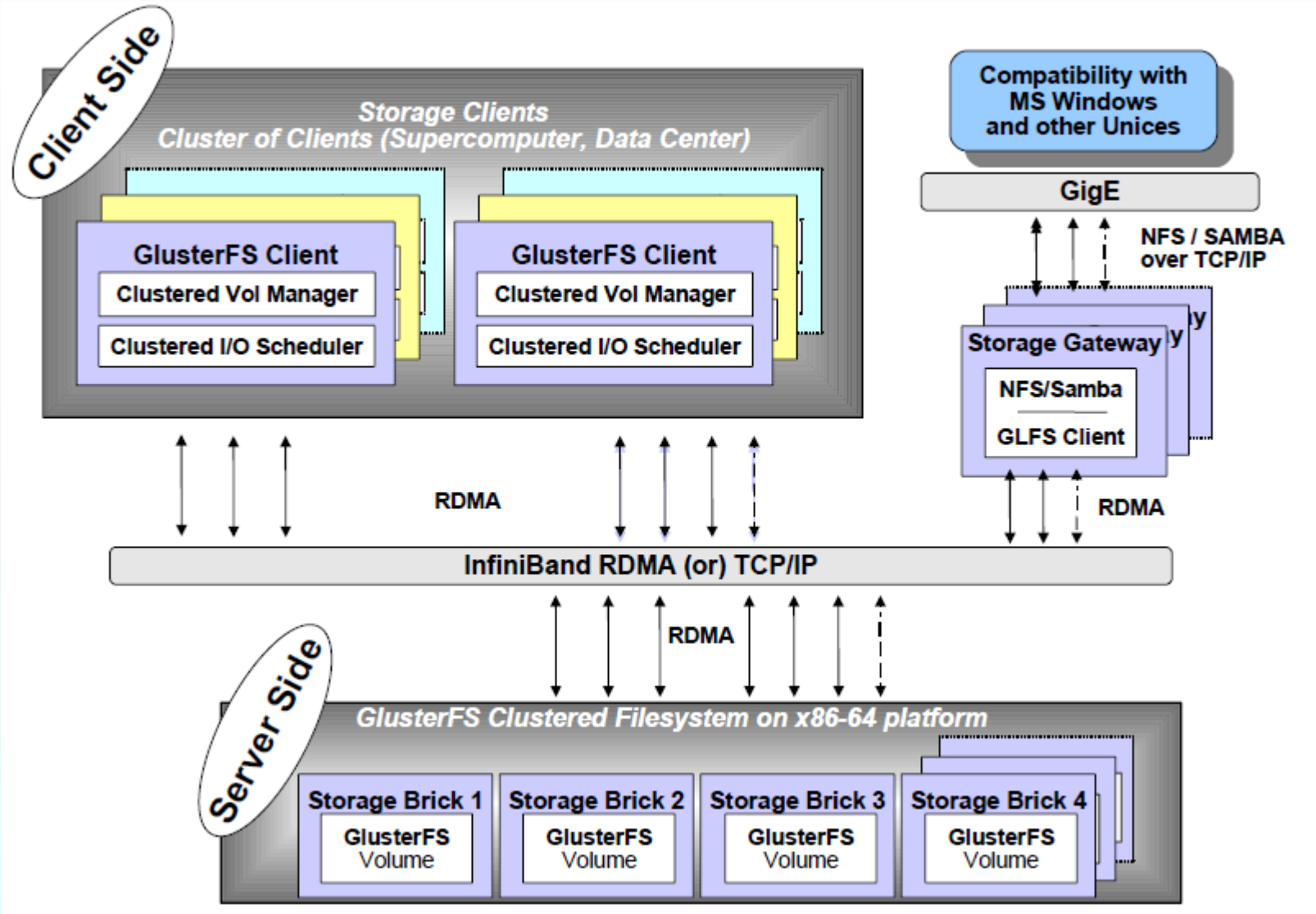
堆栈式设计

弹性横向扩展

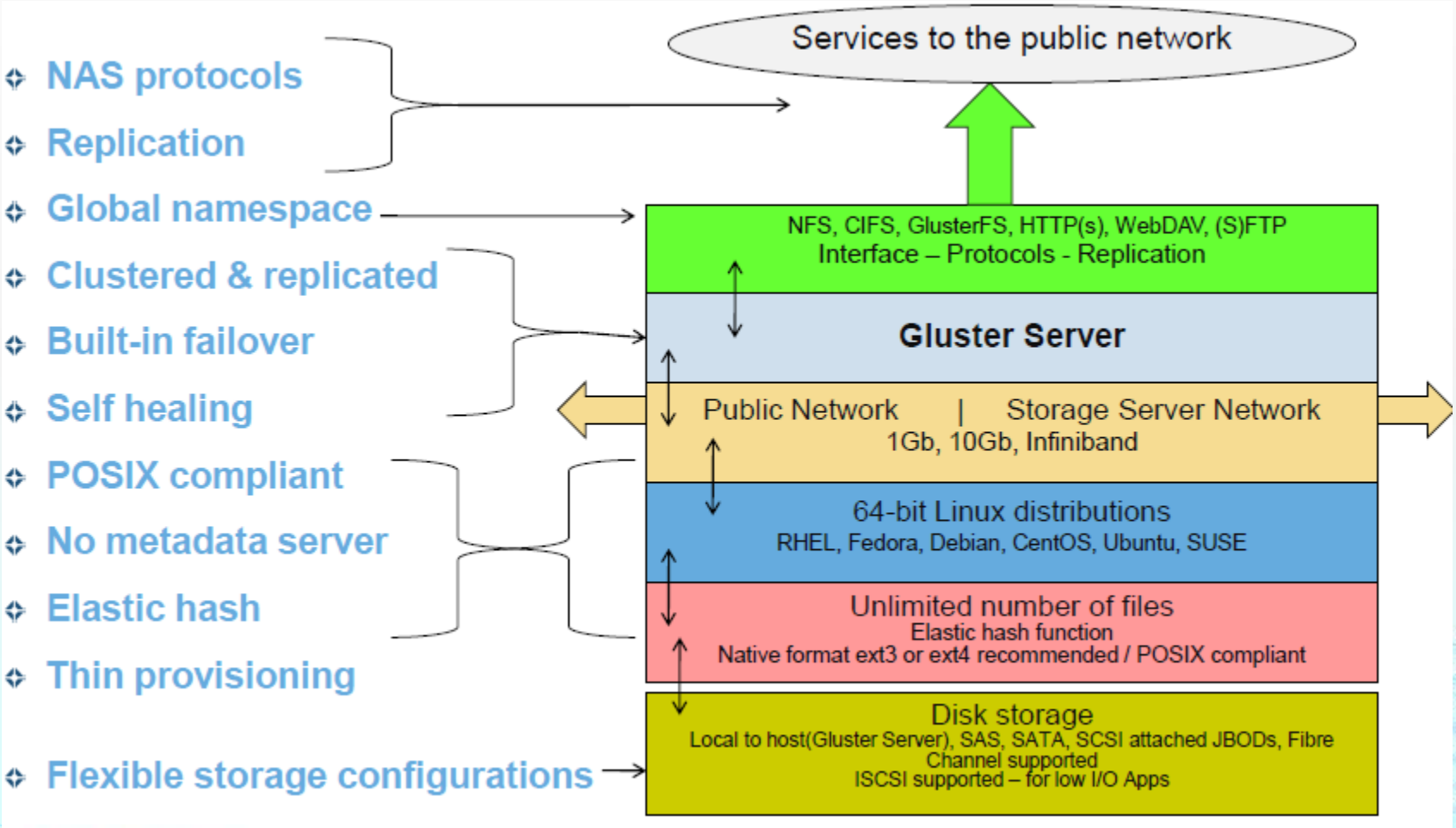
高速网络通信

数据自动修复

GlusterFS总体架构



模块化/堆栈式存储OS架构



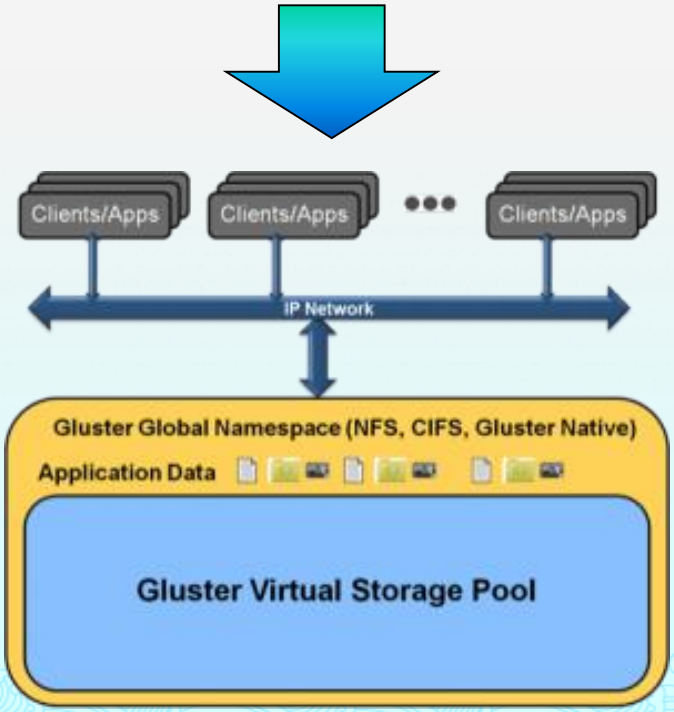
全局统一命名空间

通过分布式文件系统将物理分散的存储资源虚拟化成统一的存储池

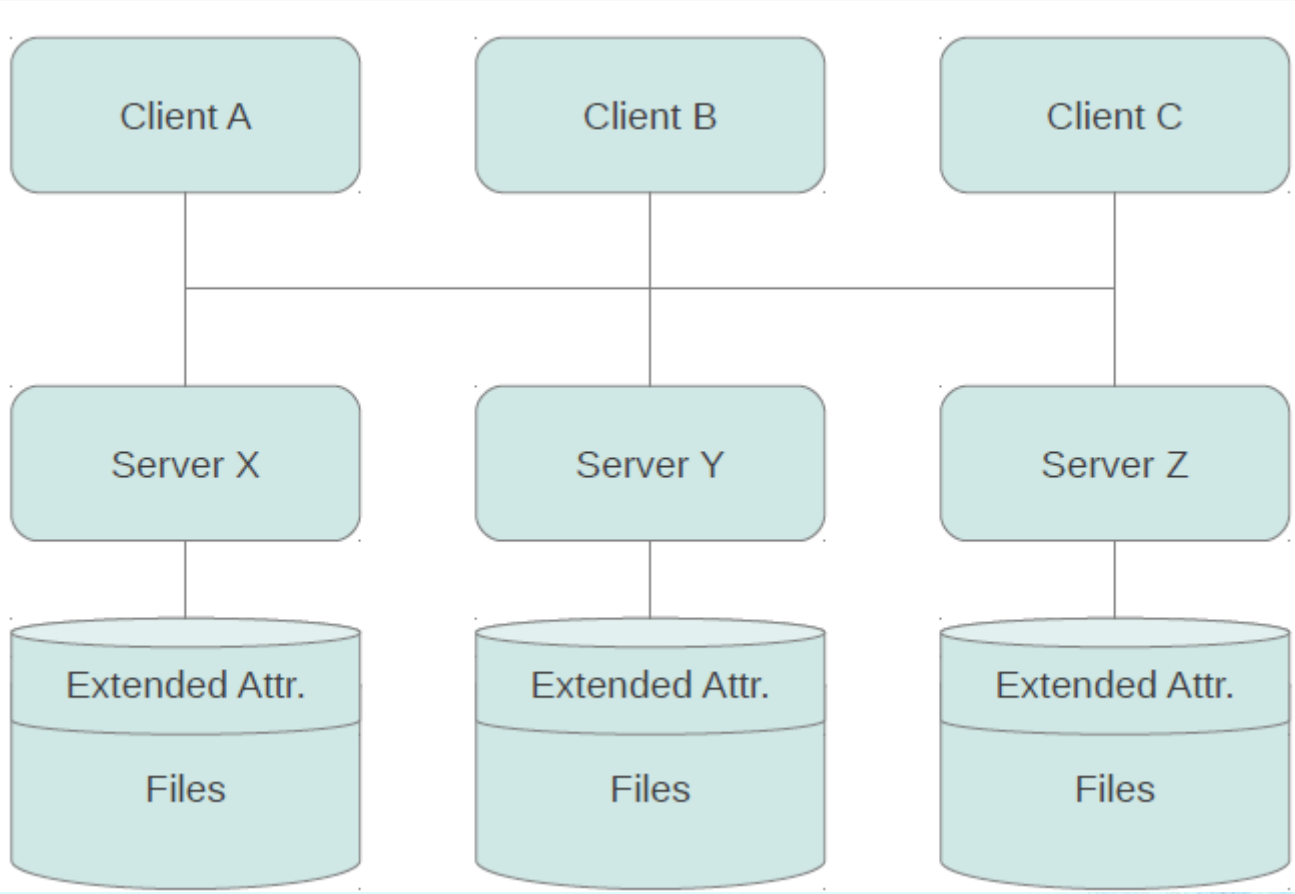
文件系统

GlusterFS

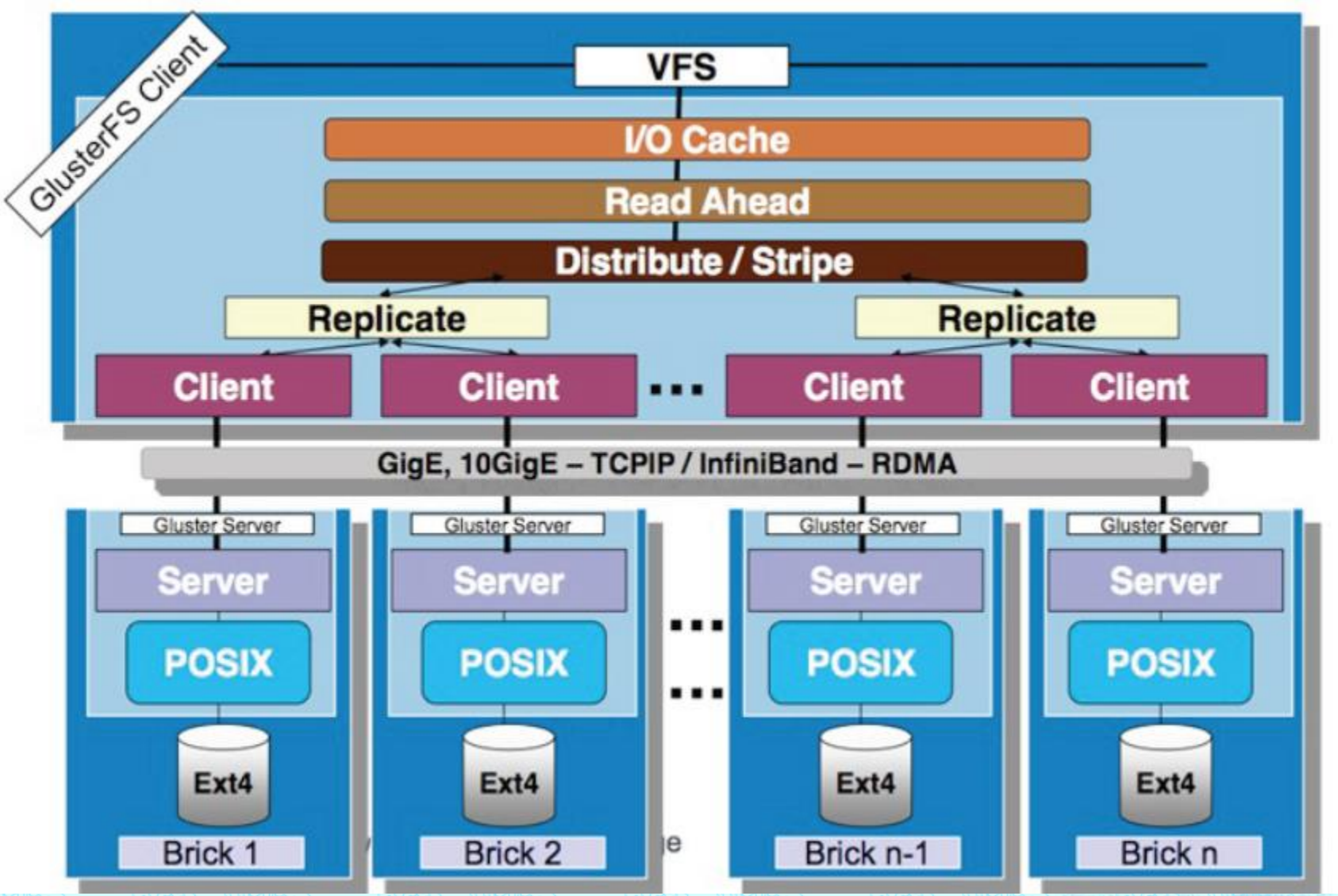
RAID



无集中元数据服务


Client A
Client B
Client C
Server X
Server Y
Server Z
Extended Attr.
Files
Extended Attr.
Files
Extended Attr.
Files

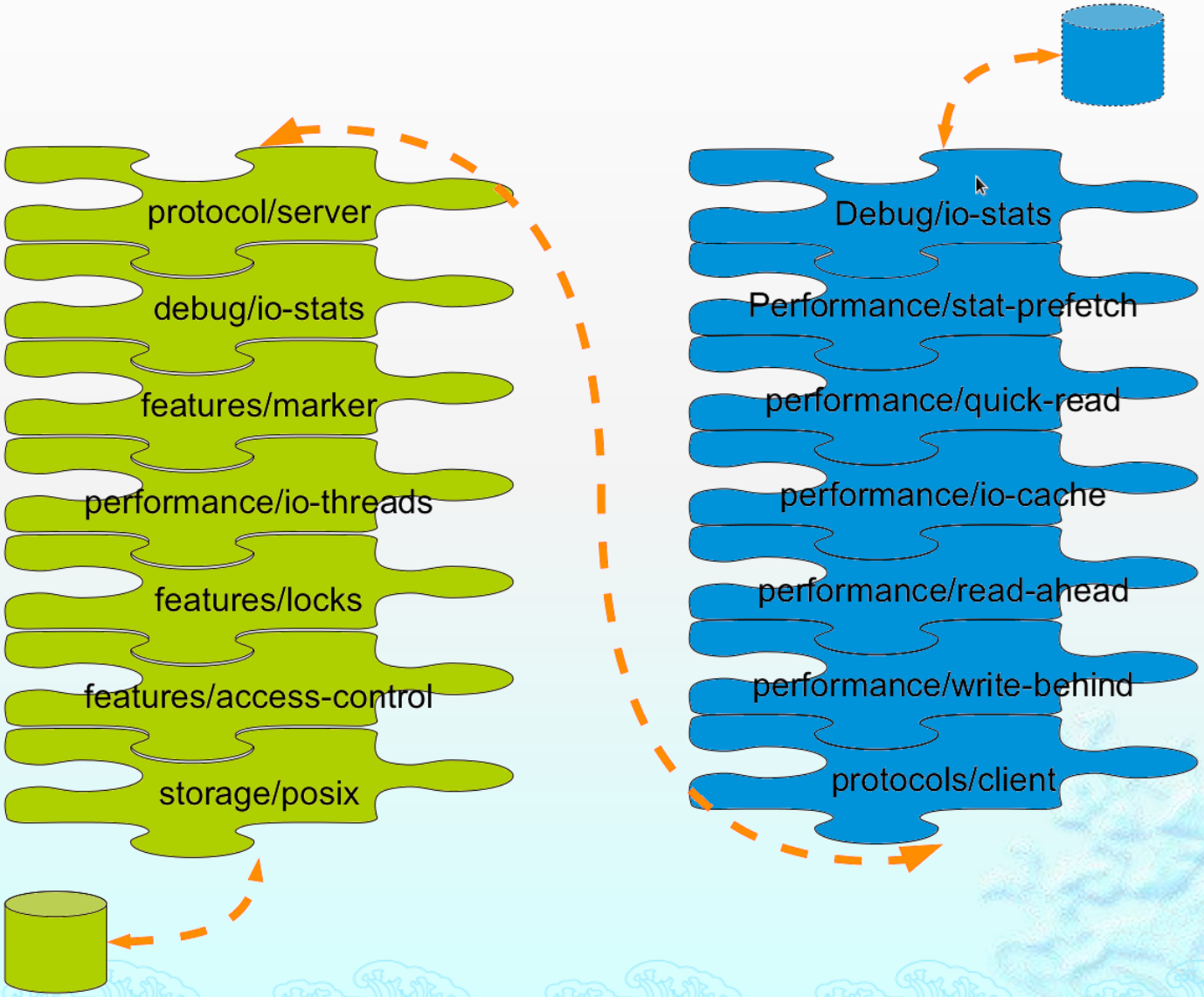
GlusterFS堆栈式软件架构



GlusterFS基本概念

- Brick
  - A filesystem mountpoint
  - A unit of storage used as a GlusterFS building block
- Translator
  - Logic between the bits and the Global Namespace
  - Layered to provide GlusterFS functionality
- Volume
  - Bricks combined and passed through translators
- Node / Peer
  - Server running the gluster daemon and sharing volumes

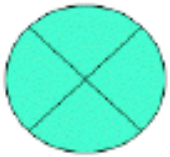
Translators



弹性hash算法

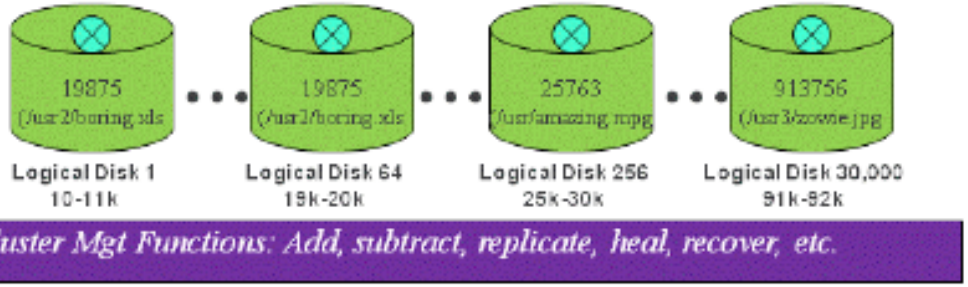
Elastic Hashing Algorithm

Goal: Systematically locate files based solely on their name



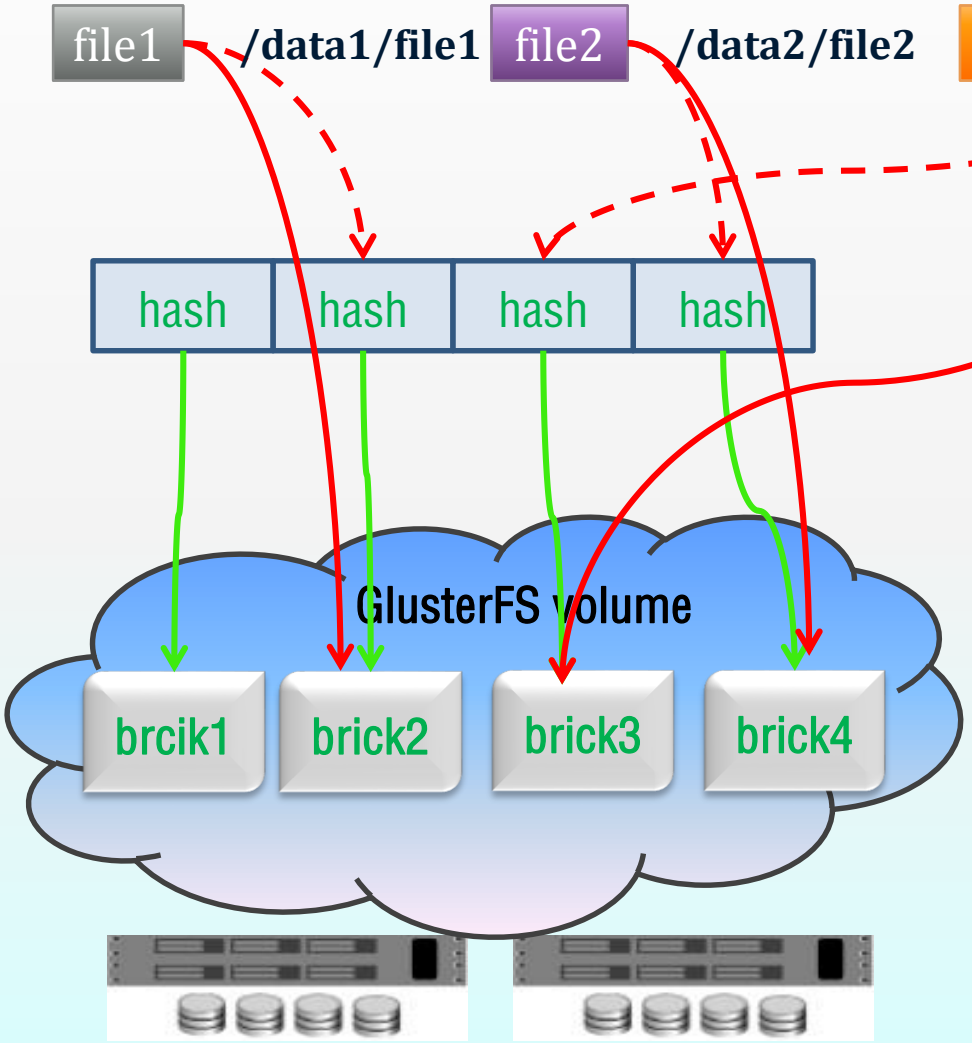
Algorithm

"1) Run path/filename through hash
2) Assign to logical disk based on numerical result
3) Separate logical storage from physical storage



- Provision huge # of virtual disks
- Use hash algorithm to assign systematically & in distributed fashion to *virtual* storage locations
- Virtualization lets you deal flexibly with physical disks
  - …add, subtract, deal with different disk performance or capacity parameters, etc.
- **WE HAVE MET OUR GOAL!**

弹性Hash算法流程



- 1、使用Davies-Meyer算法计算32位hash值，输入参数为文件名；
- 2、根据hash值在集群中选择子卷（存储服务器），进行文件定位；
- 3、对所选择的子卷进行数据访问。

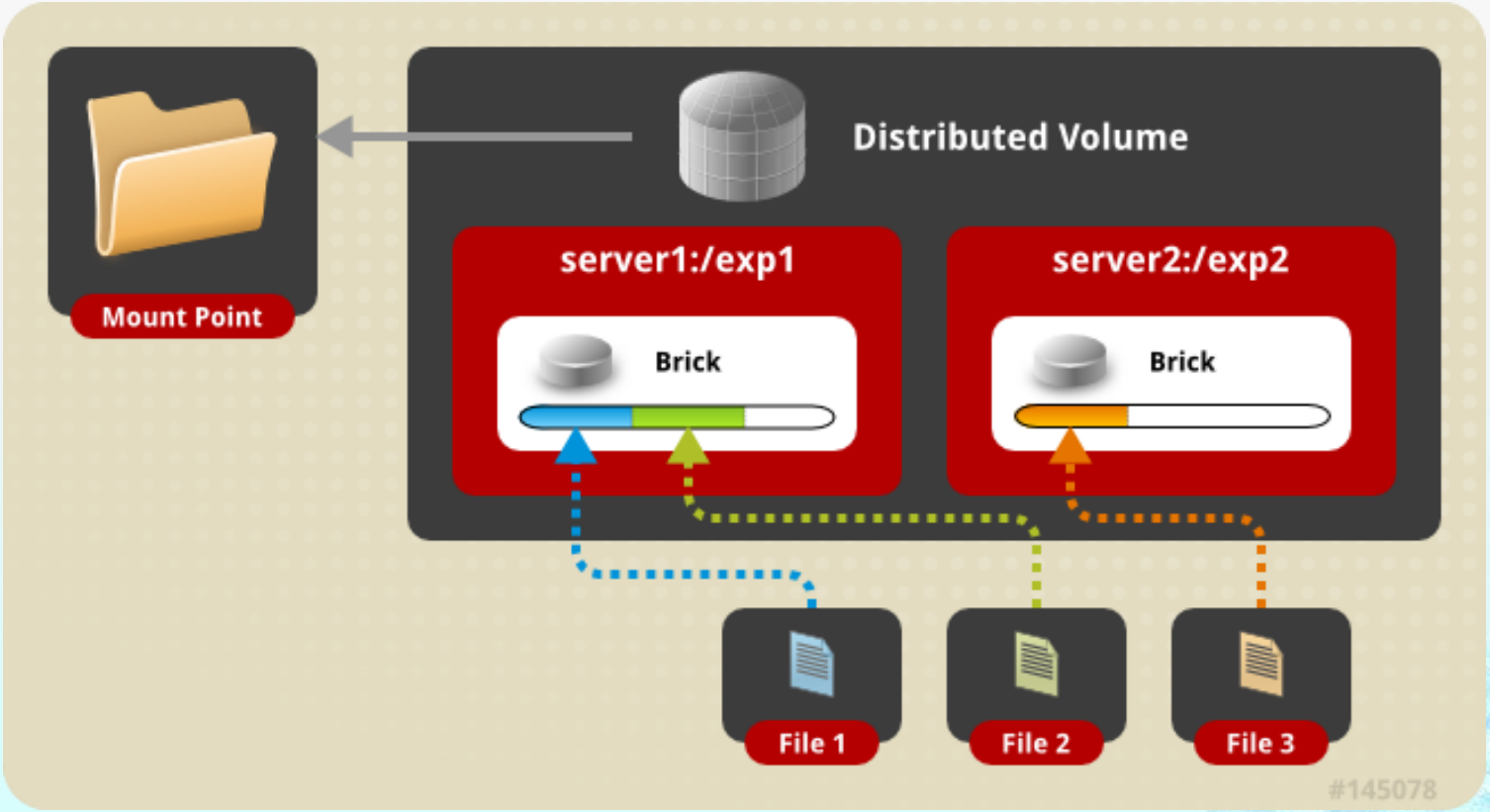
Brick	Hash range
Brick1	00000000 ~ 3FFFFFFF
Brick2	4FFFFFFF ~ 7FFFFFFF
Brick3	8FFFFFFF ~ BFFFFFFF
Brick4	CFFFFFFF ~ FFFFFFFF

GlusterFS卷类型

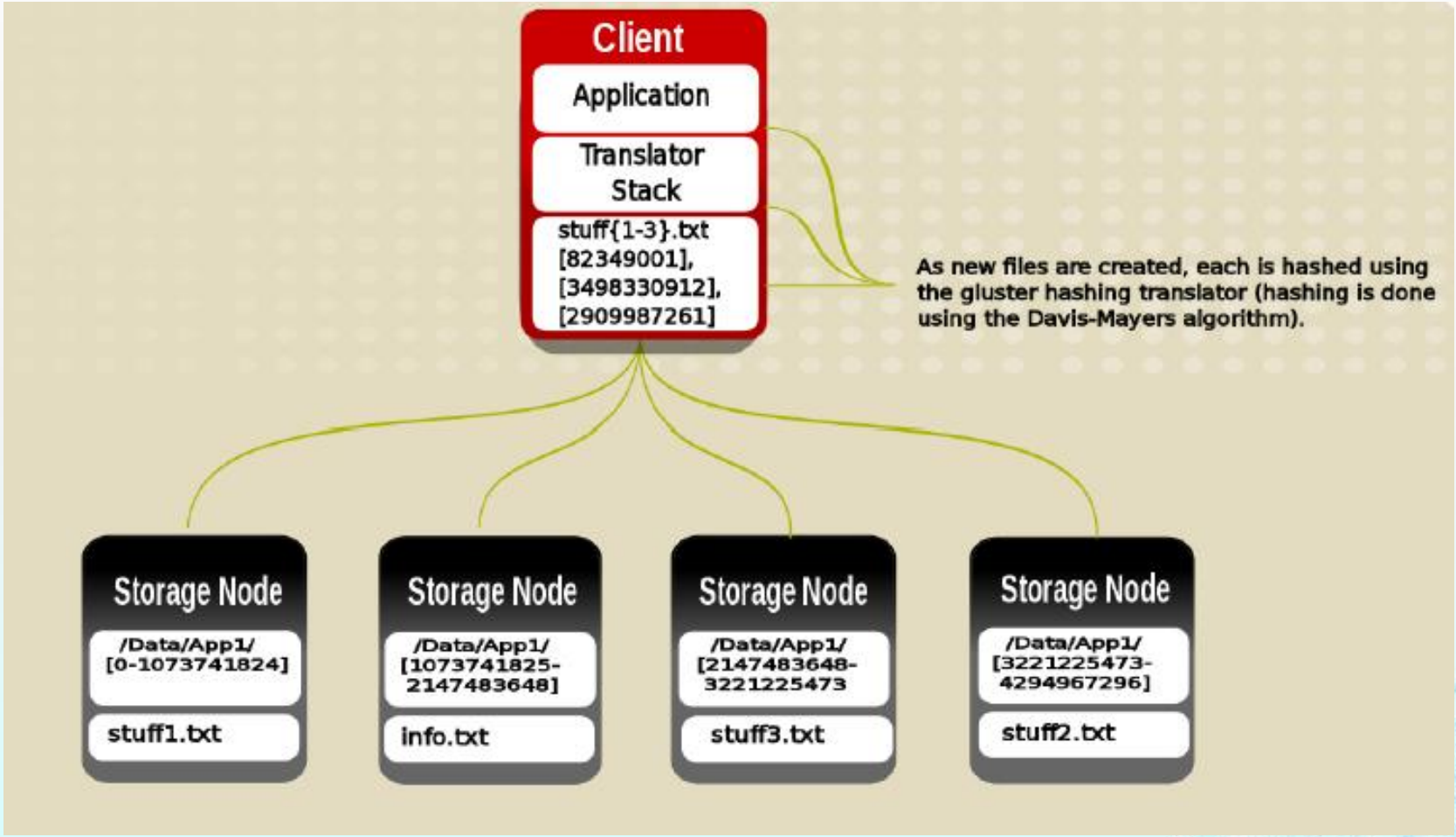
- 基本卷
  - 哈希卷（Distributed Volume）
  - 复制卷（Replicated Volume）
  - 条带卷（Striped Volumes）
- 复合卷
  - 哈希复制卷(Distributed Replicated Volume)
  - 哈希条带卷（Distributed Striped Volume）
  - 复制条带卷（Replicated Striped Volume）
  - 哈希复制条带卷（Distributed Replicated Striped Volume）

哈希卷（Distributed Volume）

- 文件通过hash算法在所有brick上分布
- 文件级RAID 0，不具有容错能力

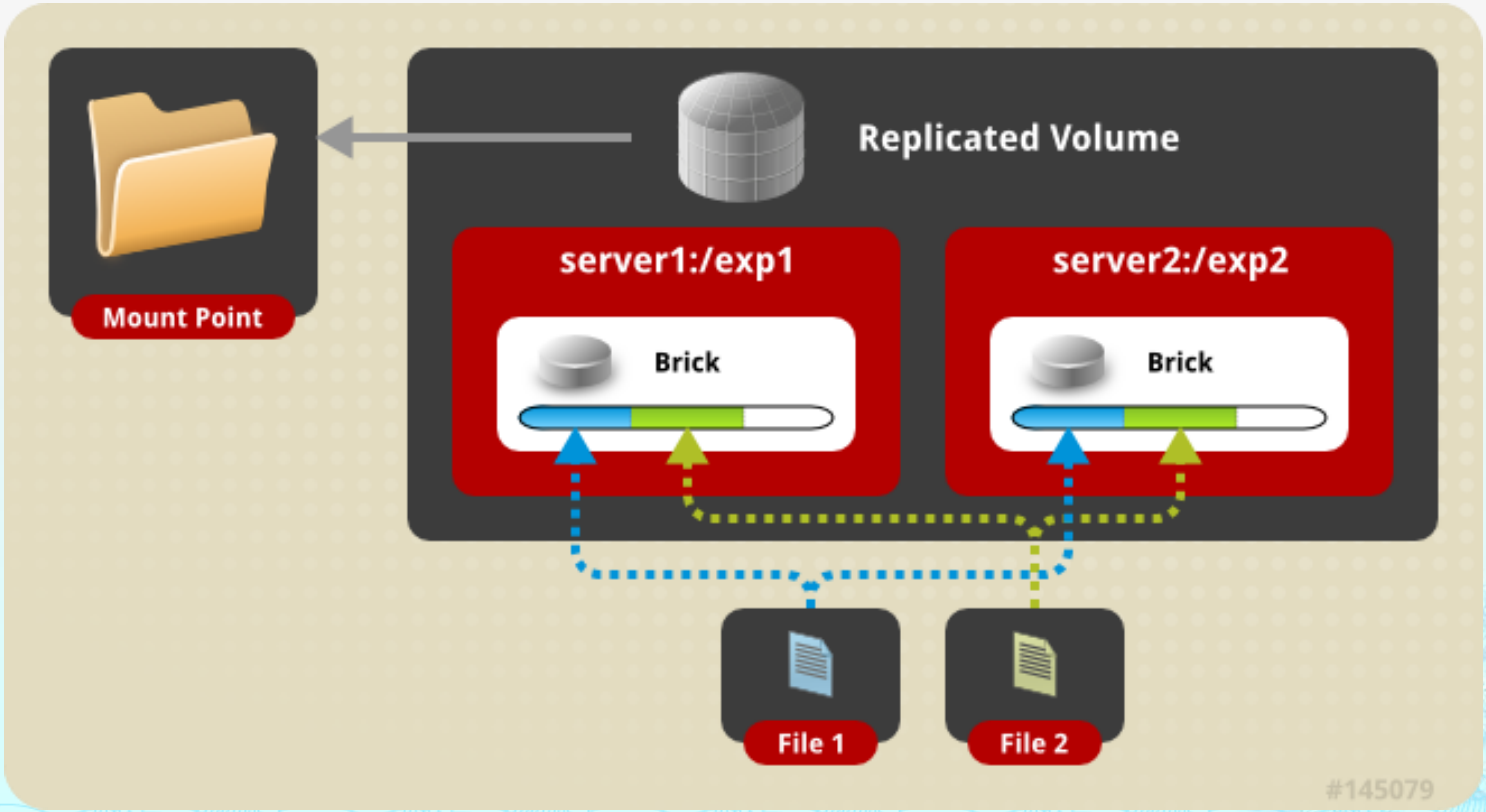


哈希卷工作原理

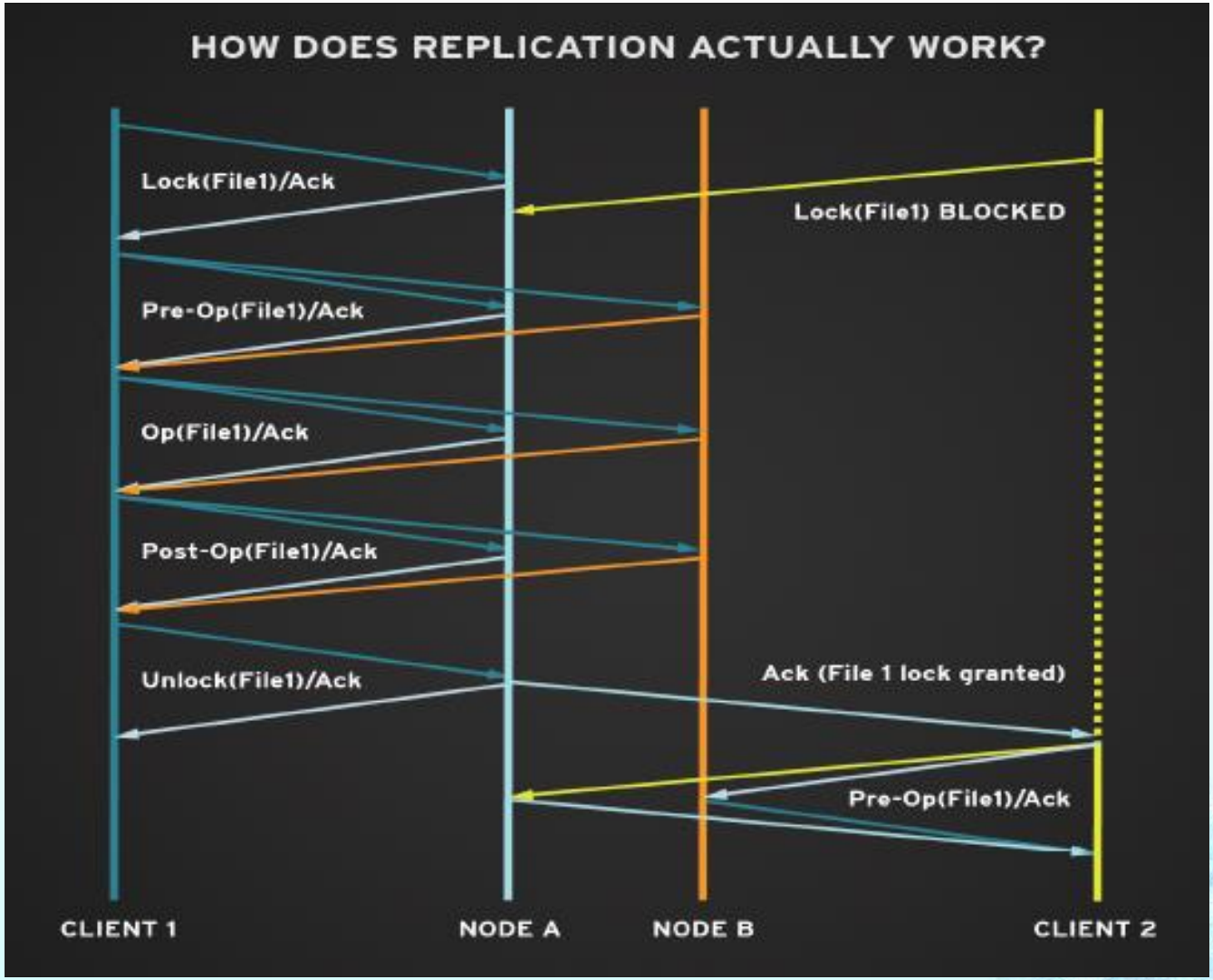

Client
Application
Translator Stack
stuff{1-3}.txt
[82349001],
[3498330912],
[2909987261]
As new files are created, each is hashed using the gluster hashing translator (hashing is done using the Davis-Mayers algorithm).
Storage Node
/Data/App1/
[0-1073741824]
stuff1.txt
Storage Node
/Data/App1/
[1073741825-
2147483648]
info.txt
Storage Node
/Data/App1/
[2147483648-
3221225473
stuff3.txt
Storage Node
/Data/App1/
[3221225473-
4294967296]
stuff2.txt

复制卷（Replicated Volume）

- 文件同步复制到多个brick上
- 文件级RAID 1，具有容错能力
- 写性能下降，读性能提升

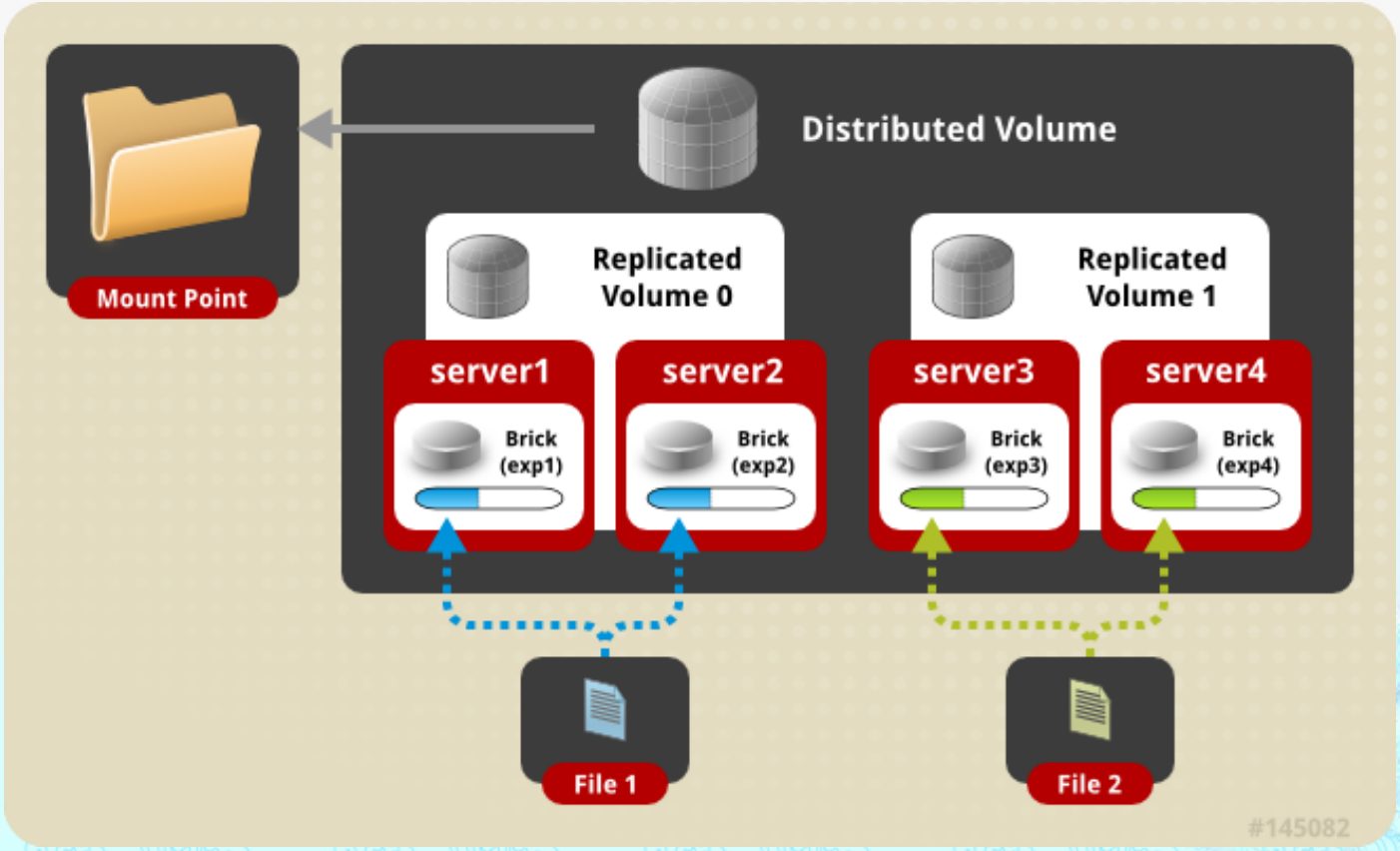


复制卷工作原理


HOW DOES REPLICATION ACTUALLY WORK?
Lock(File1)/Ack
Lock(File1) BLOCKED
Pre-Op(File1)/Ack
Op(File1)/Ack
Post-Op(File1)/Ack
Unlock(File1)/Ack
Ack (File 1 lock granted)
Pre-Op(File1)/Ack
CLIENT 1
NODE A
NODE B
CLIENT 2

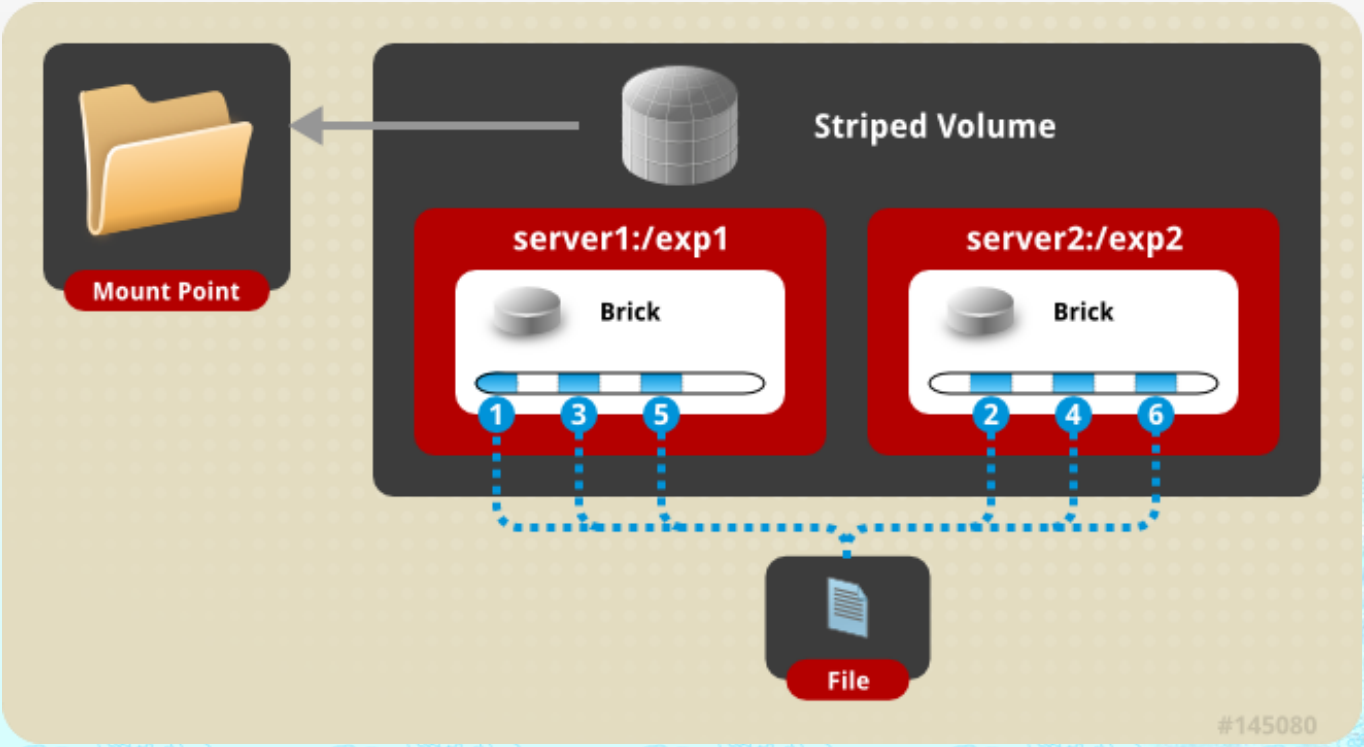
复合卷：哈希 + 复制

- 哈希卷和复制卷的复合方式
- 同时具有哈希卷和复制卷的特点



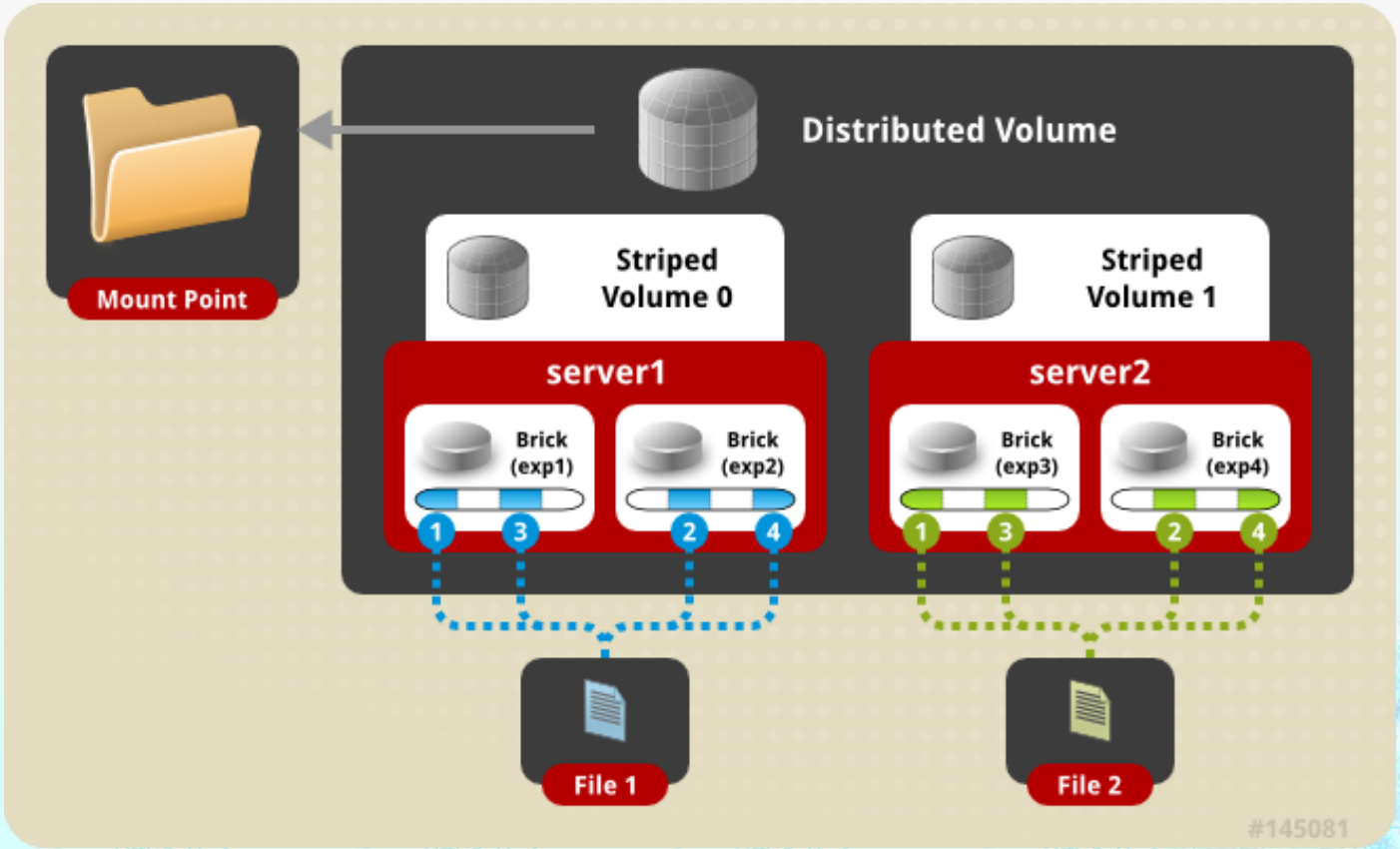
条带卷（Striped Volumes）

- 单个文件分布到多个brick上，支持超大文件
- 类似RAID 0，以Round-Robin方式
- 通常用于HPC中的超大文件高并发访问



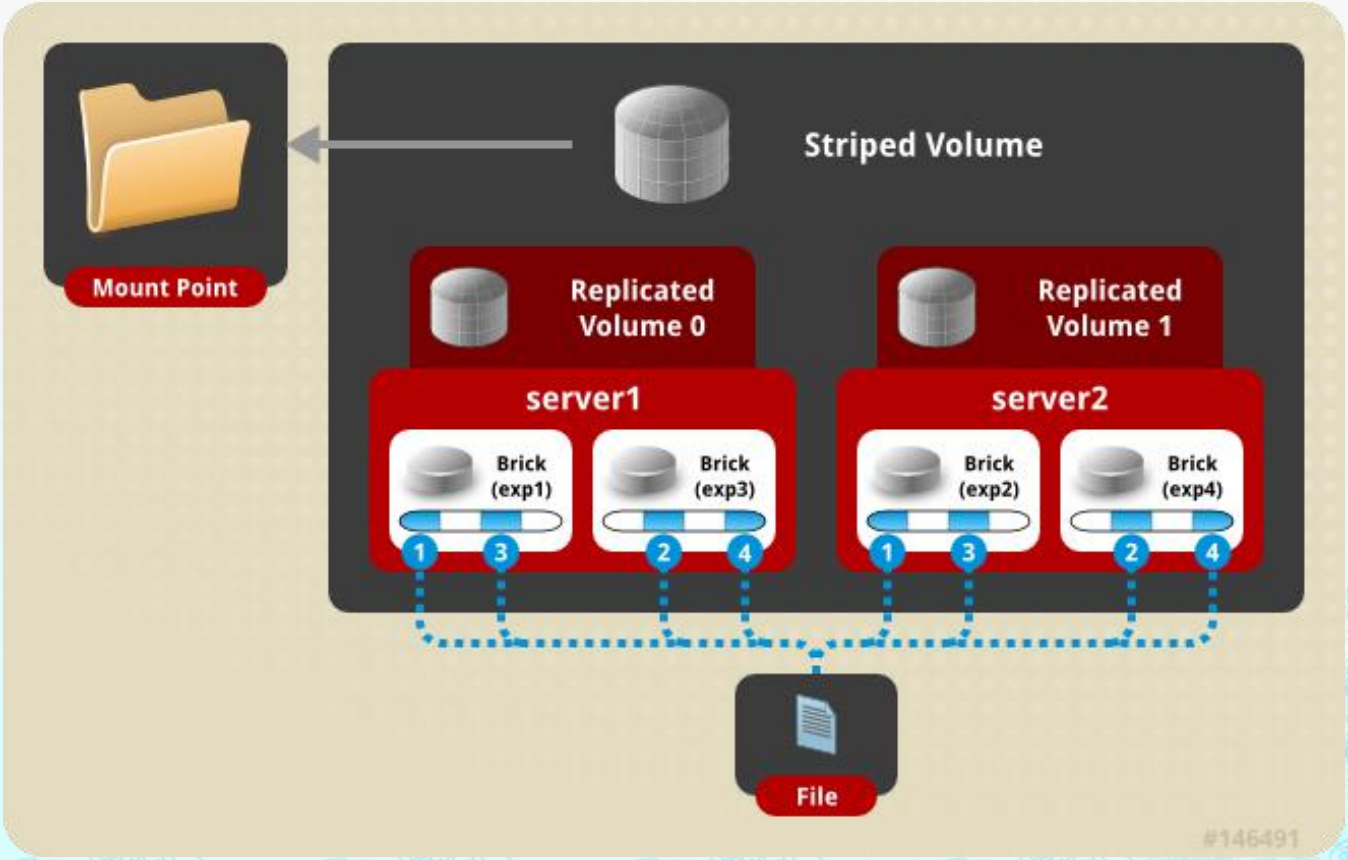
复合卷：哈希 + 条带

- 哈希卷和条带卷的复合方式
- 同时具有哈希卷和条带卷的特点



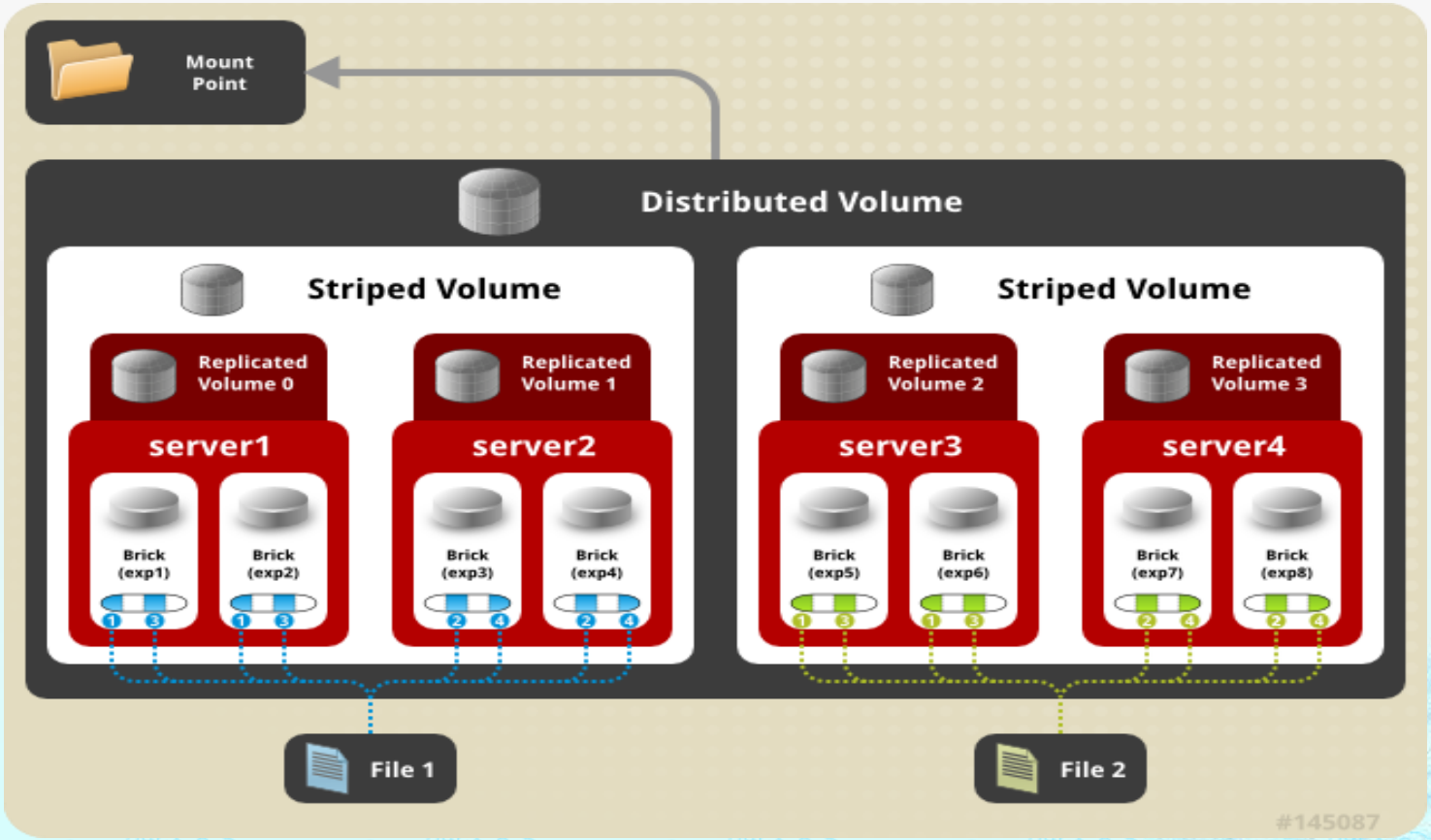
复合卷：条带 + 复制

- 类似RAID 10
- 同时具有条带卷和复制卷的特点

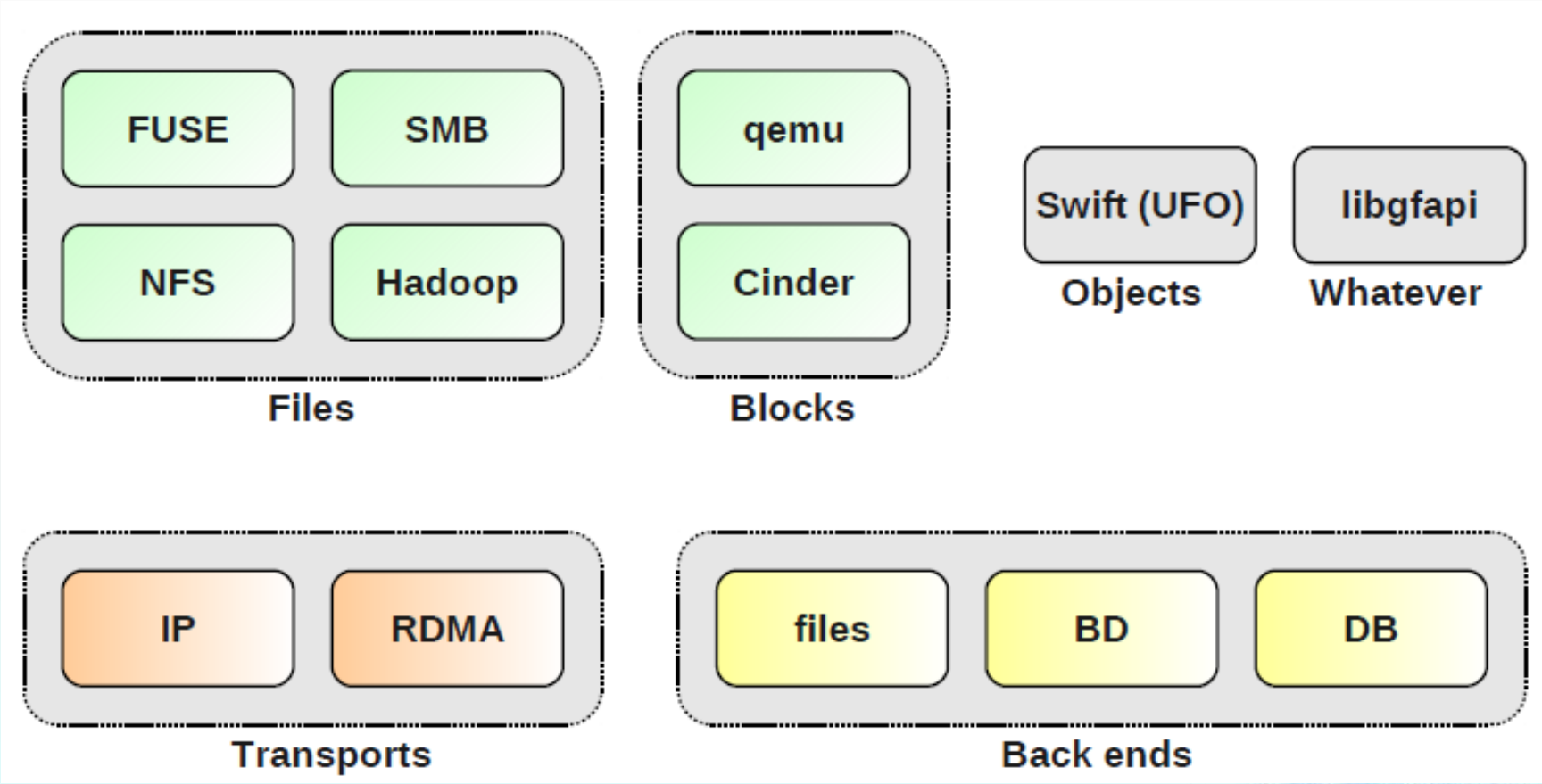


复合卷：哈希 + 条带 + 复制

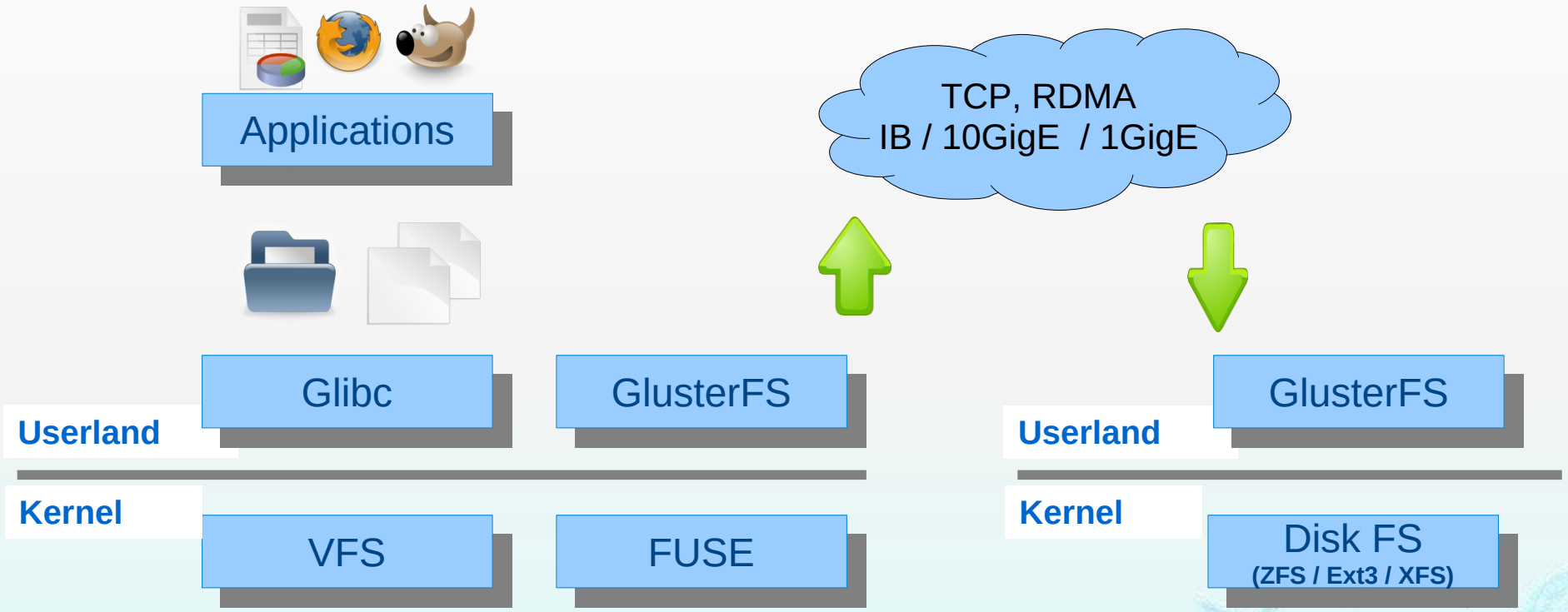
- 三种基本卷的复合卷
- 通常用于类Map Reduce应用



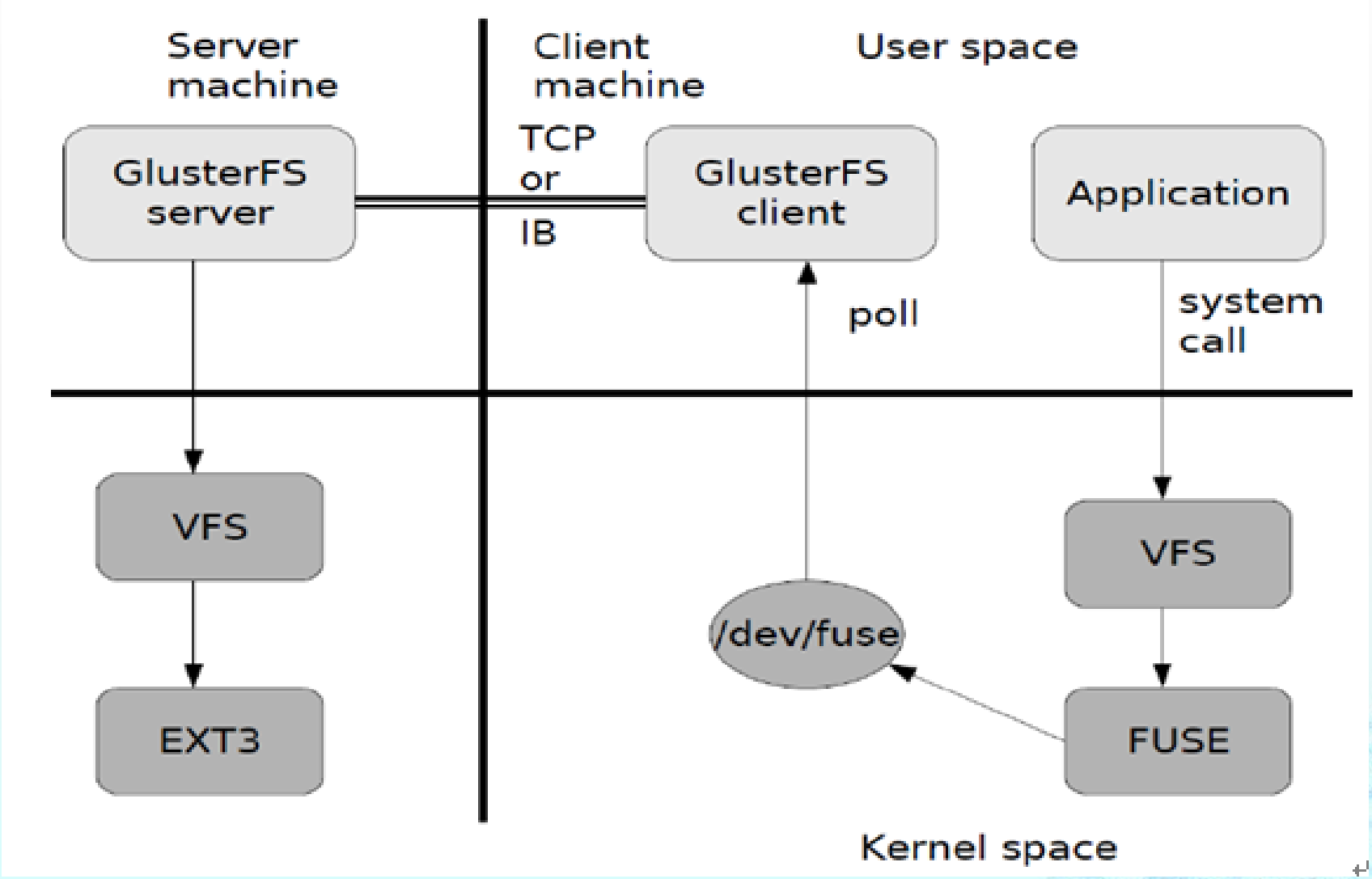
GlusterFS访问接口



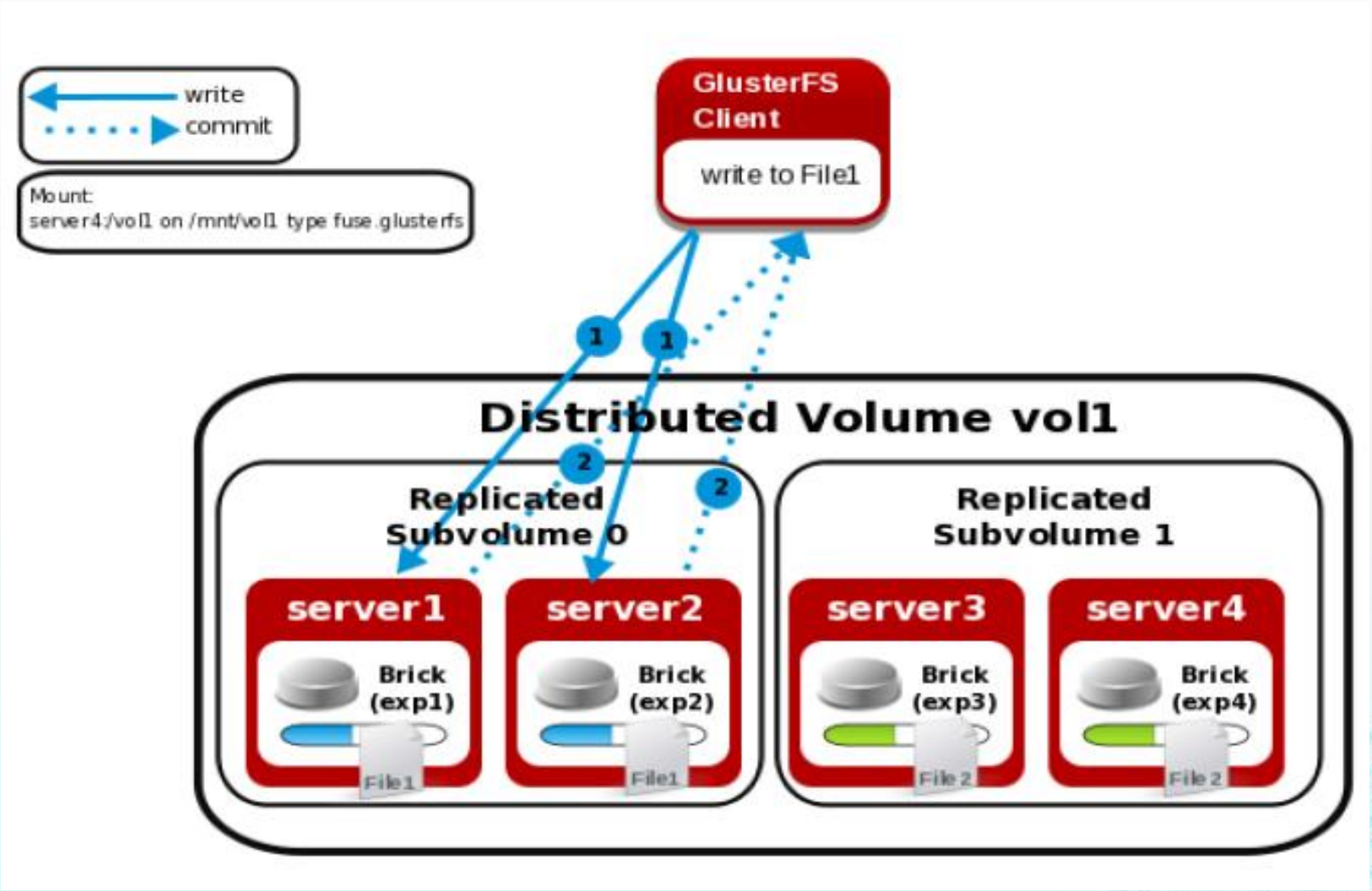
GlusterFS – FUSE Architecture



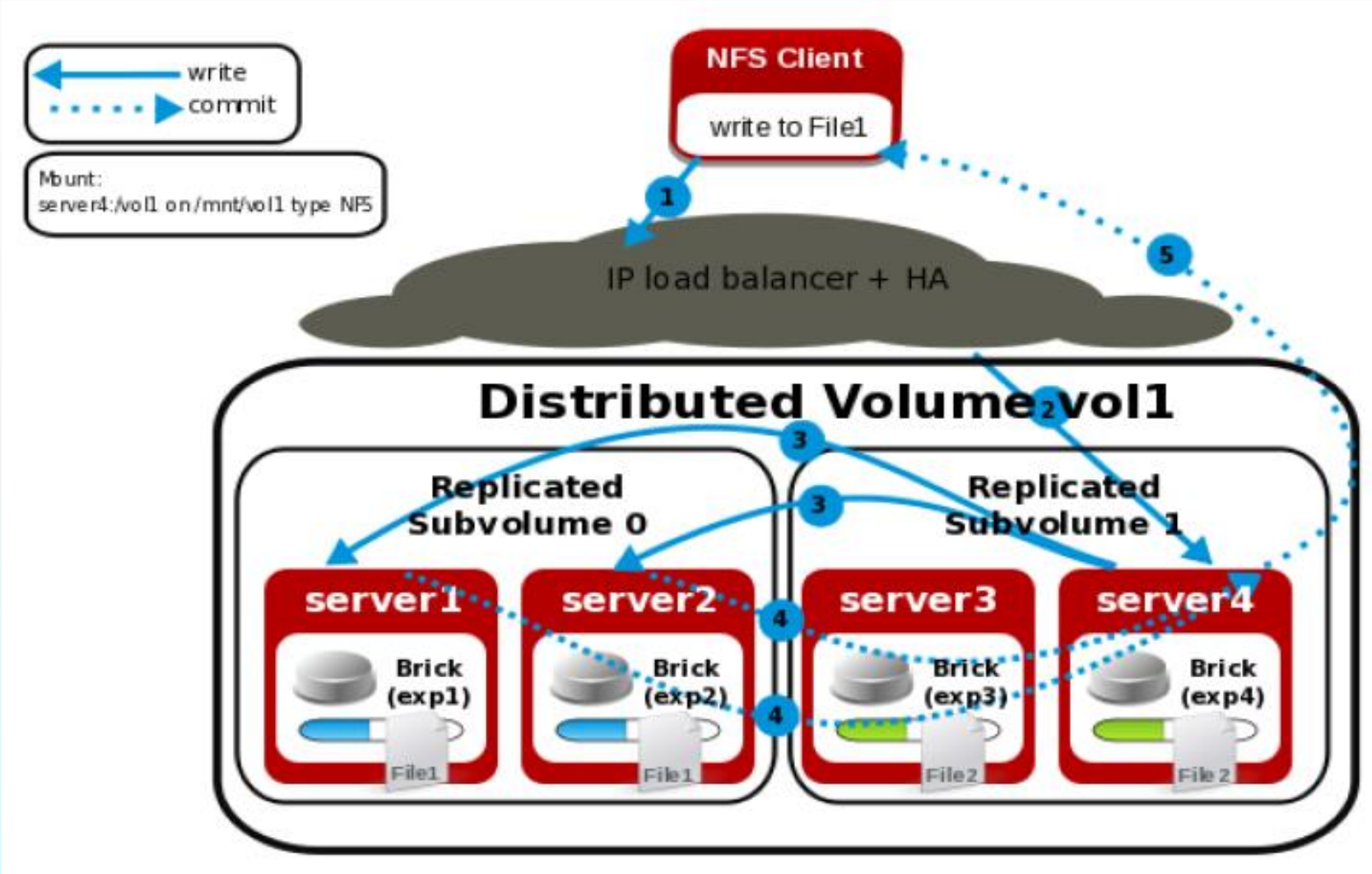
GlusterFS数据流



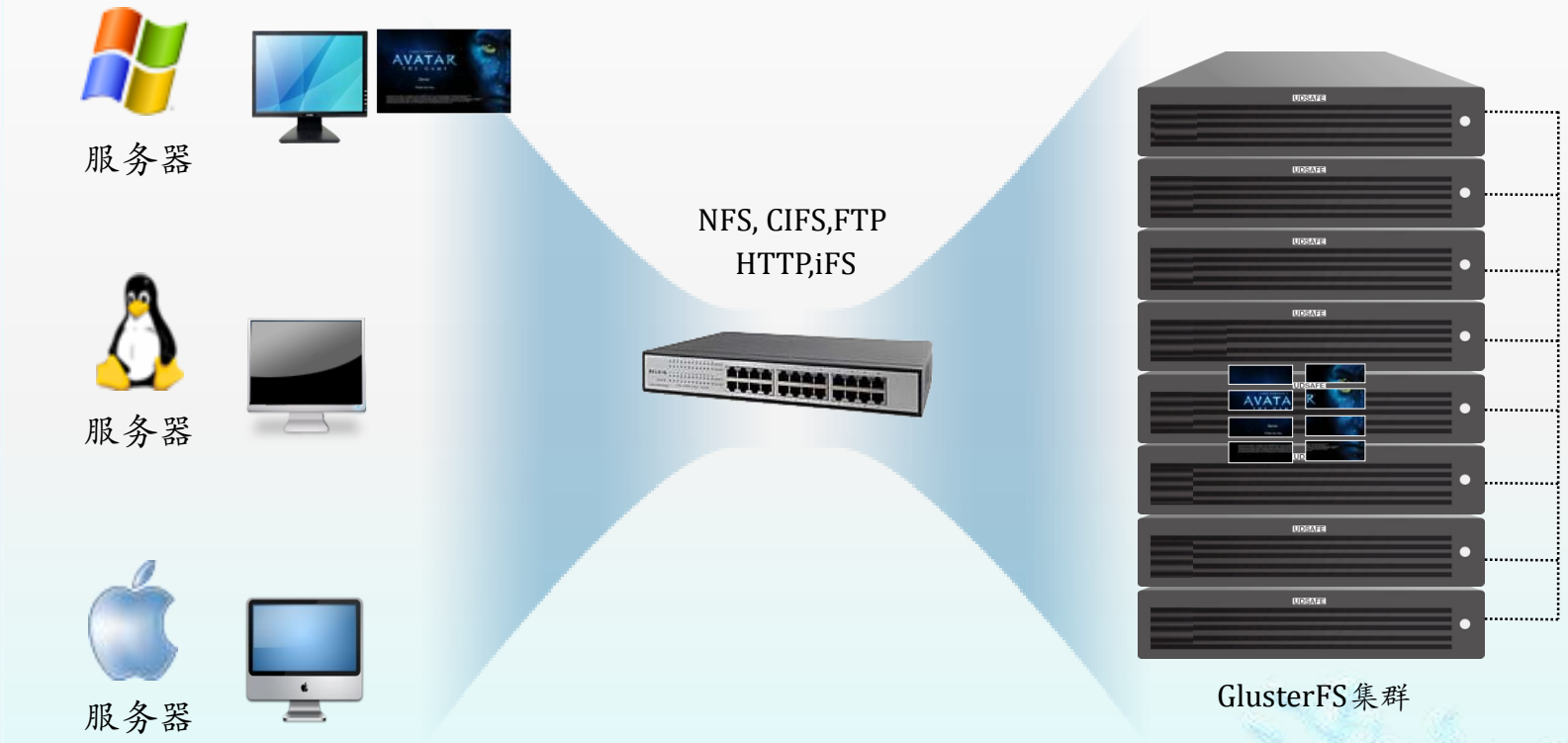
FUSE 访问


write
commit
Mount:
server4:/vol1 on /mnt/vol1 type fuse.glusterfs
GlusterFS Client
write to File1
1
1
2
2
Distributed Volume vol1
Replicated Subvolume 0
Replicated Subvolume 1
server1
server2
server3
server4
Brick (exp1)
Brick (exp2)
Brick (exp3)
Brick (exp4)
File1
File1
File2
File2

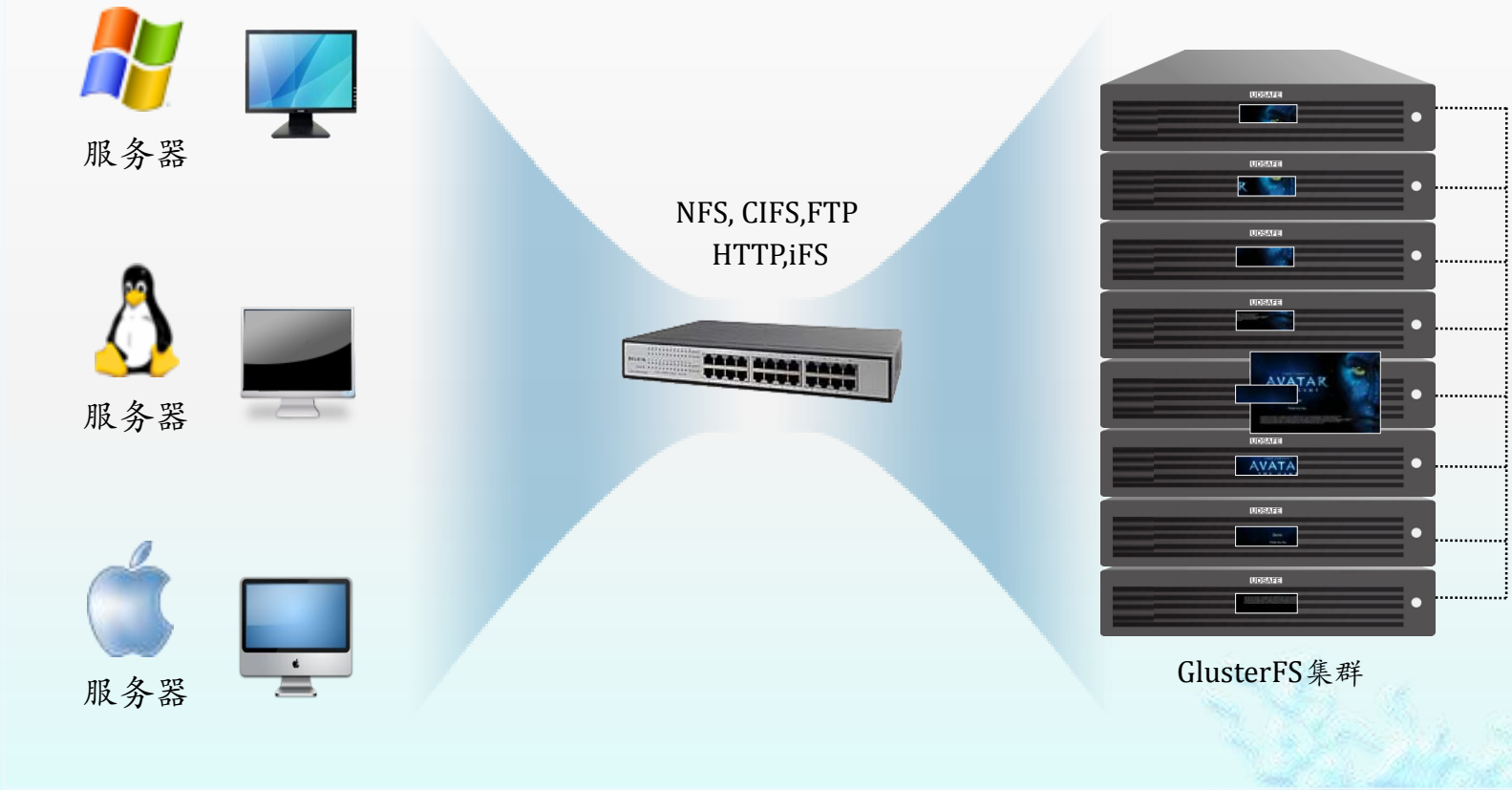
NFS/CIFS 访问


write
commit
Mount:
server4:/vol1 on /mnt/vol1 type NFS
NFS Client
write to File1
1
IP load balancer + HA
5
2
Distributed Volume vol1
3
3
Replicated
Subvolume 0
Replicated
Subvolume 1
server1
server2
server3
server4
4
4
Brick
(exp1)
Brick
(exp2)
Brick
(exp3)
Brick
(exp4)
File1
File1
File2
File2

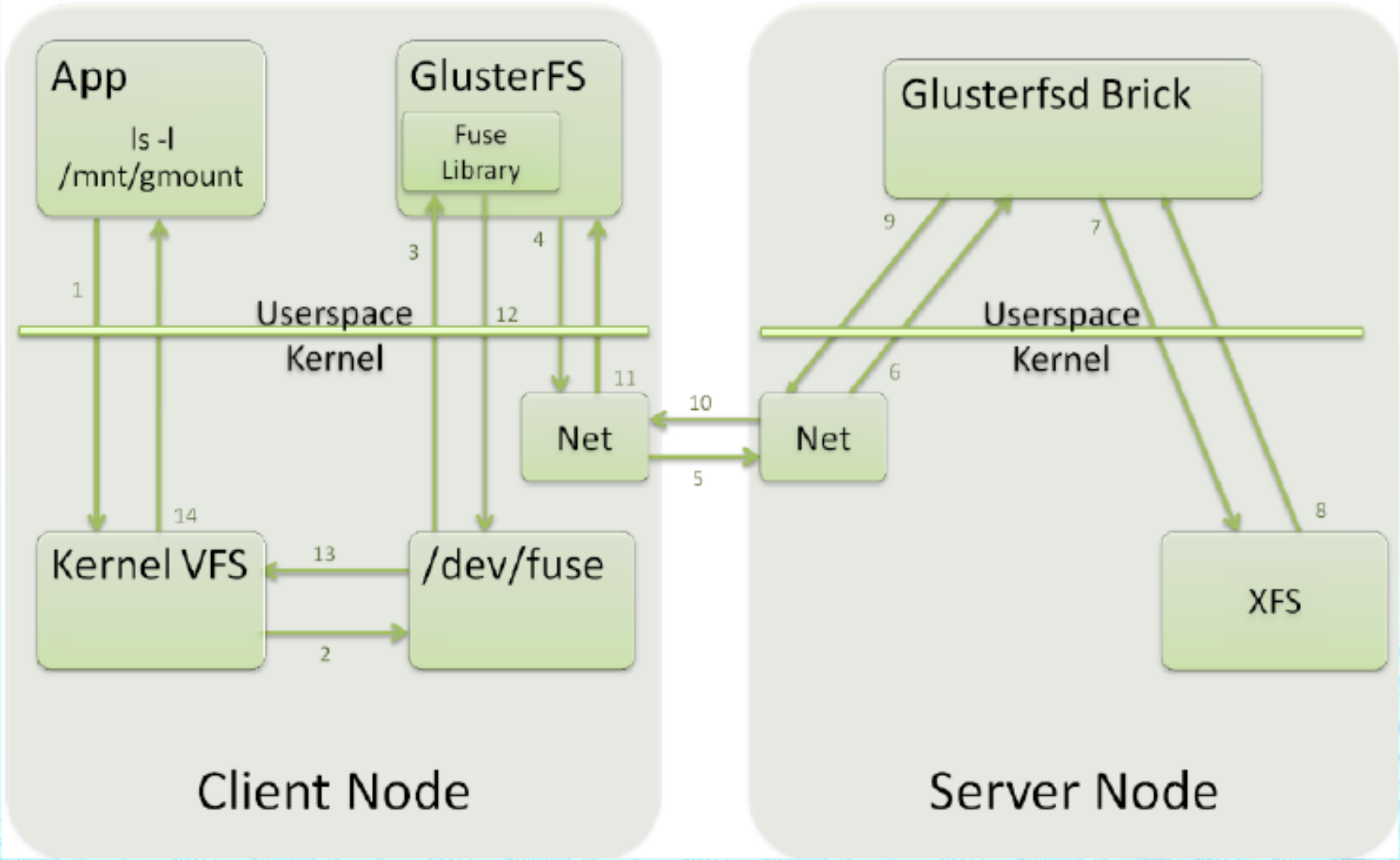
写入一个文件



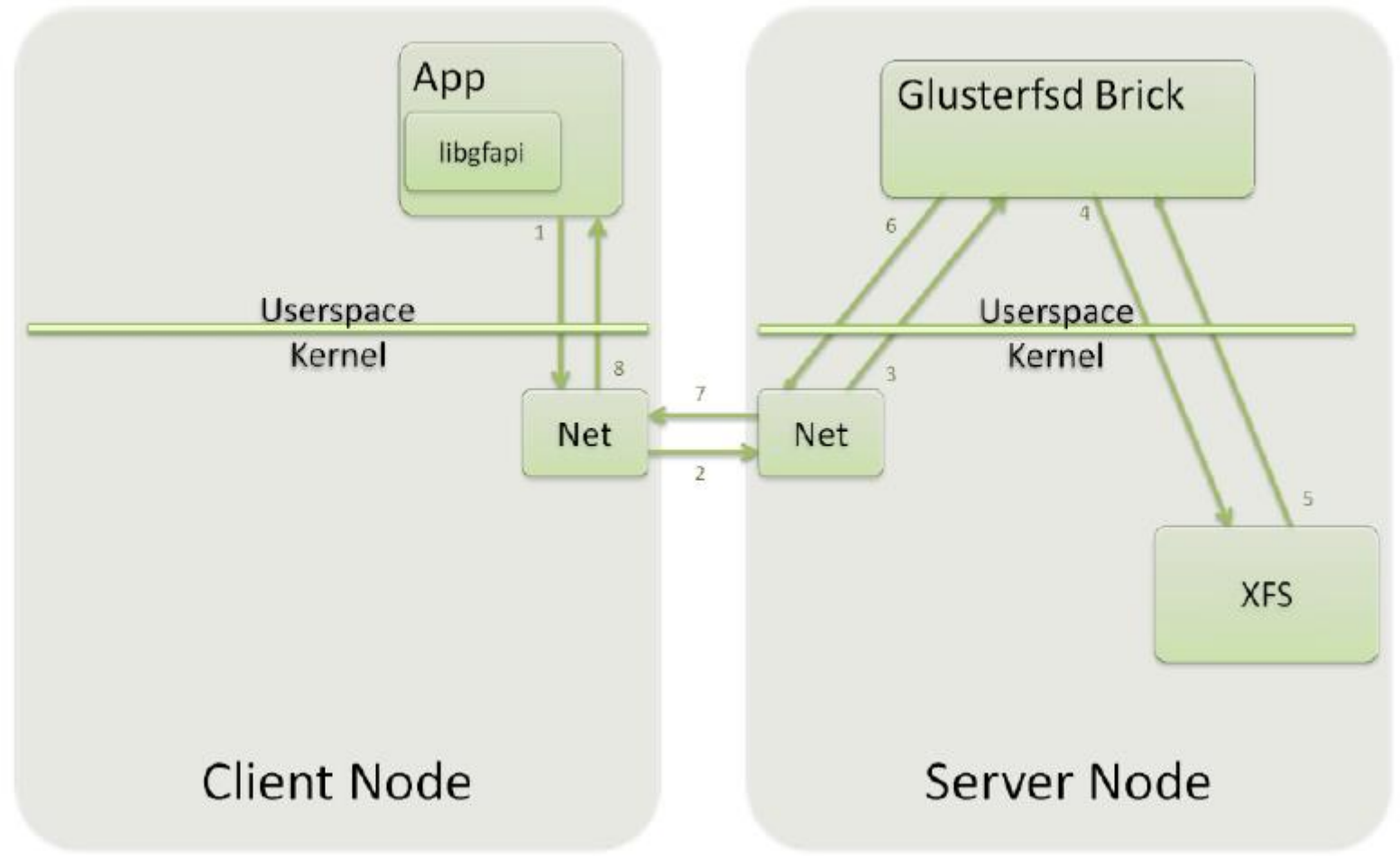
读取一个文件



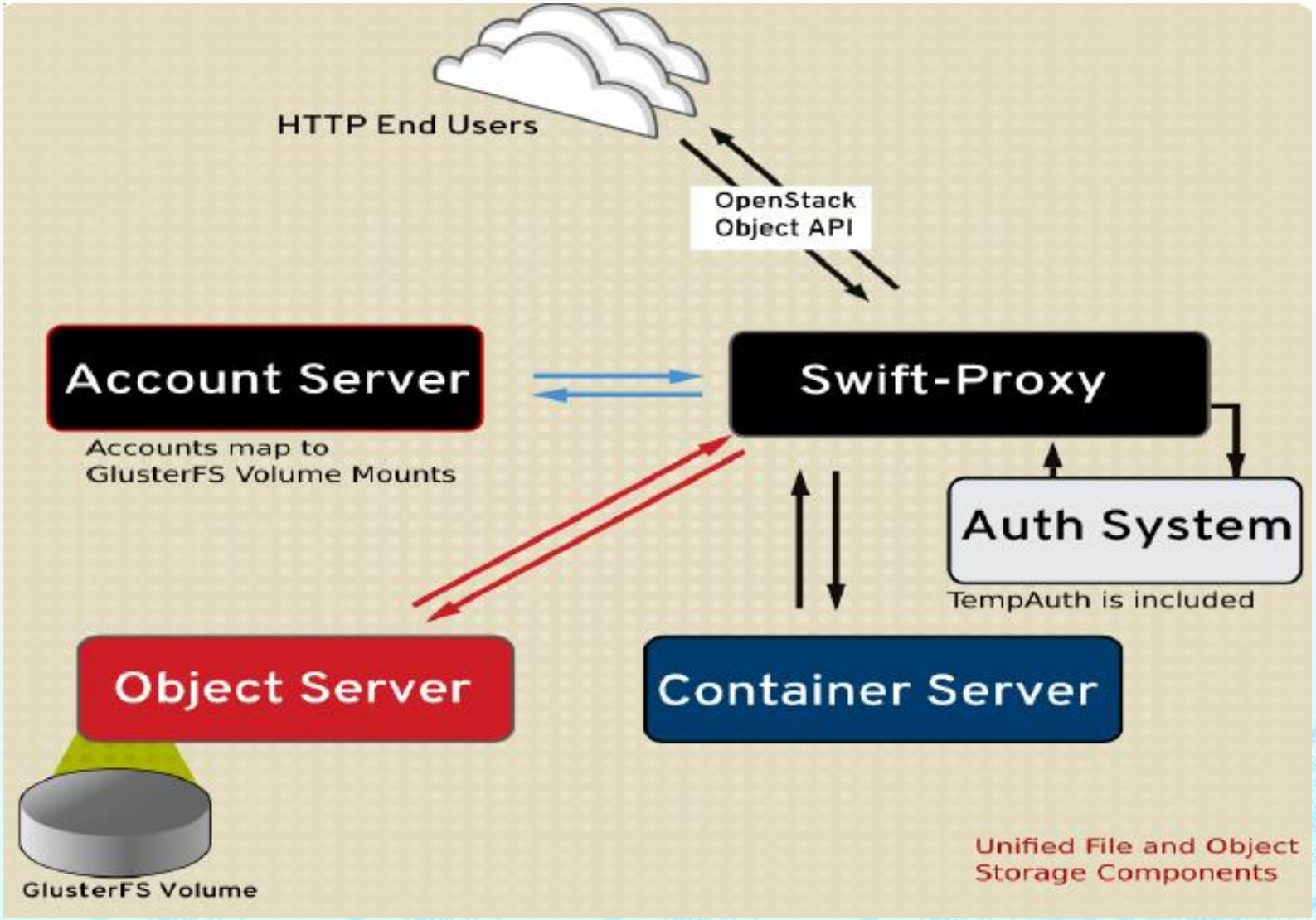
FUSE w/ Libgfapi 访问



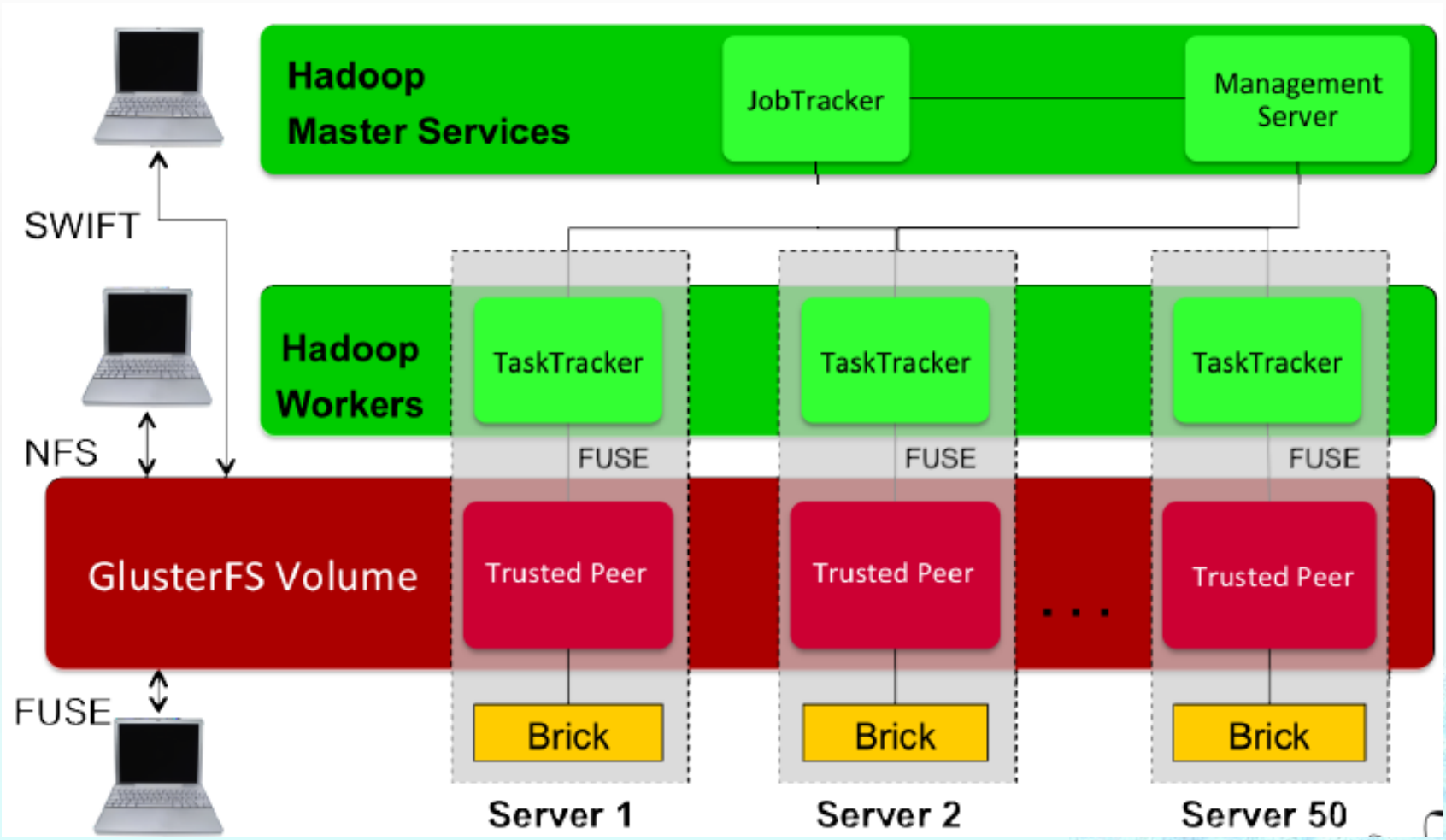
libgfapi访问


App
libgfapi
Glusterfsd Brick
1
2
3
4
5
6
7
8
Userspace
Kernel
Userspace
Kernel
Net
Net
XFS
Client Node
Server Node

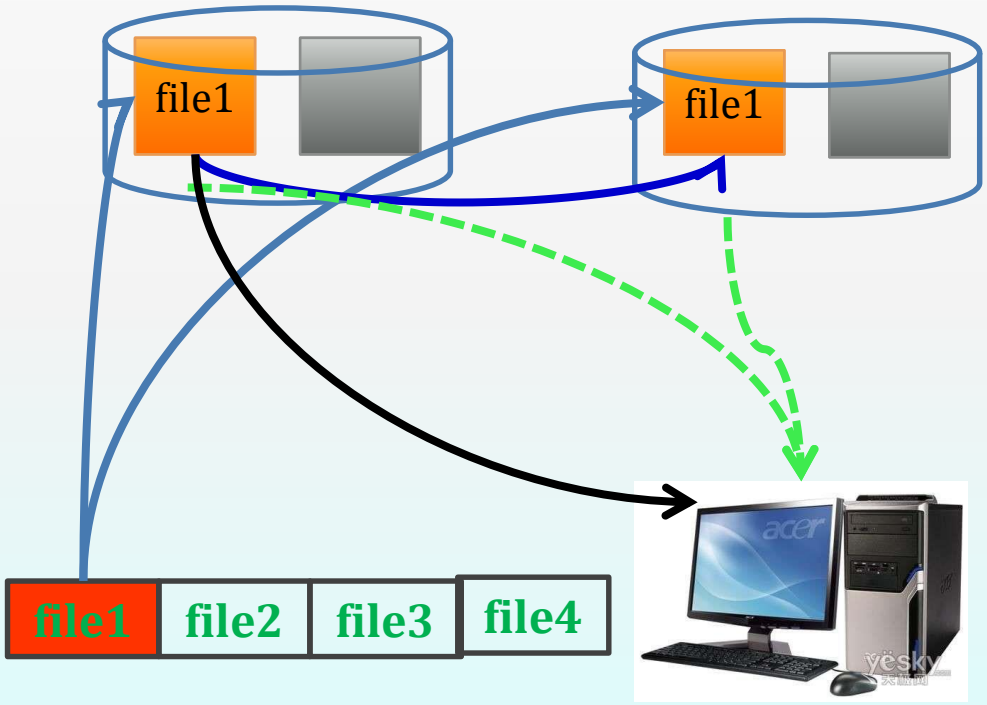
REST API 访问


HTTP End Users
OpenStack
Object API
Account Server
Accounts map to
GlusterFS Volume Mounts
Swift-Proxy
Auth System
TempAuth is included
Object Server
Container Server
GlusterFS Volume
Unified File and Object
Storage Components

Hadoop 访问


Hadoop
Master Services
JobTracker
Management
Server
SWIFT
Hadoop
Workers
TaskTracker
TaskTracker
TaskTracker
NFS
FUSE
FUSE
FUSE
GlusterFS Volume
Trusted Peer
Trusted Peer
Trusted Peer
FUSE
Brick
Brick
Brick
Server 1
Server 2
Server 50

数据自修复 Self-heal

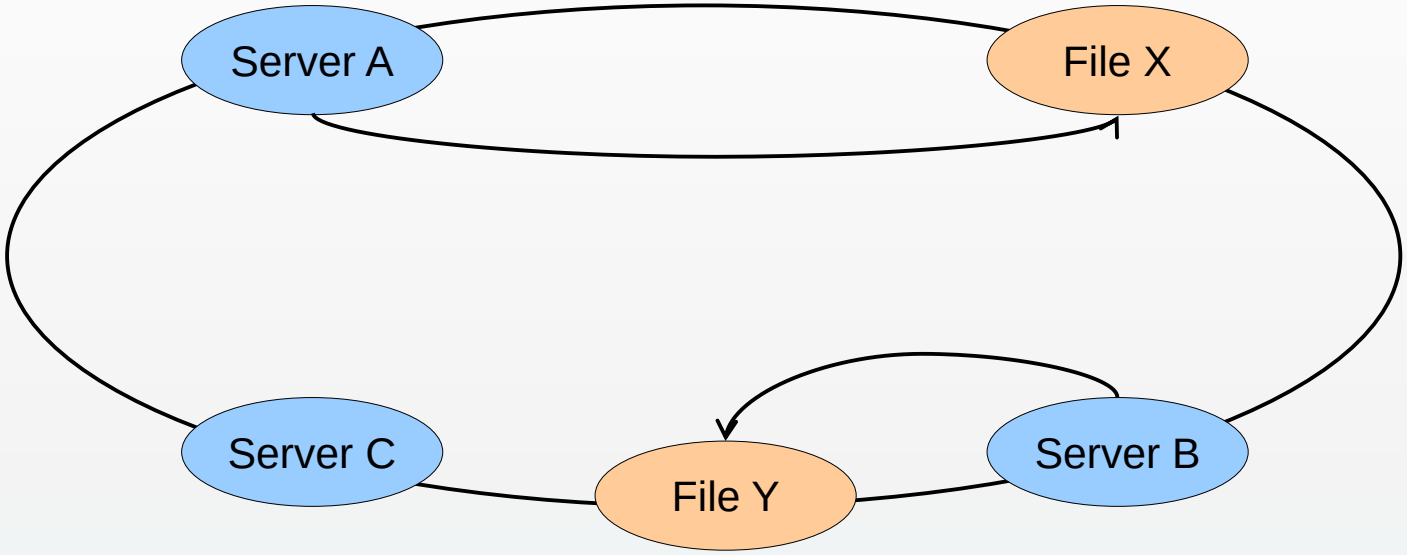


Self-heal发展

- 第一代：按需同步进行
- 第二代：完全人工扫描
- 第三代：并发自动修复(3.3)
- 第四代：基于日志

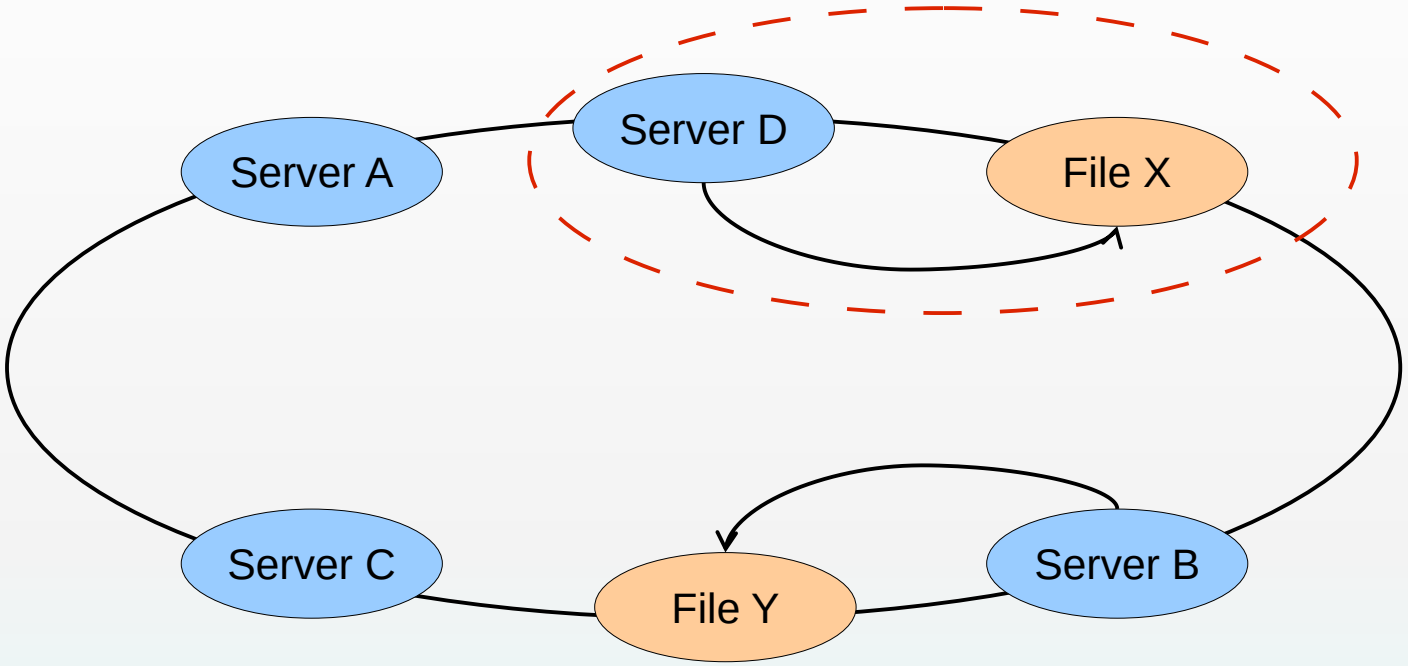
- 镜像卷文件副本保持一致性
- 触发时机：访问文件目录时
- 判断依据：扩展属性
- 脑裂问题：报错或按规则处理

Distributed Hash Table (DHT)



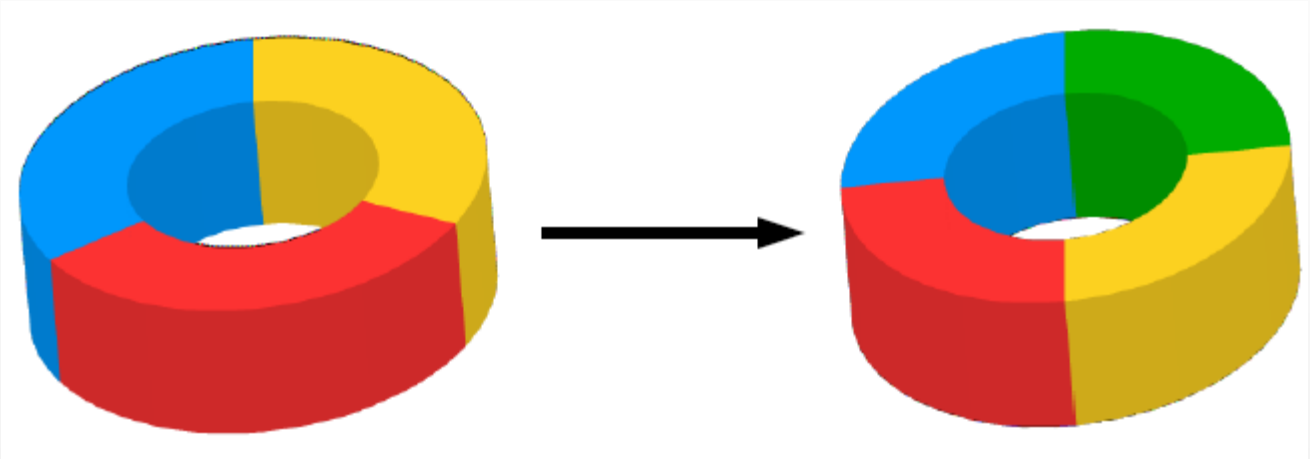
- GlusterFS弹性扩展的基础
- 确定目标hash和brick之间的映射关系

添加节点



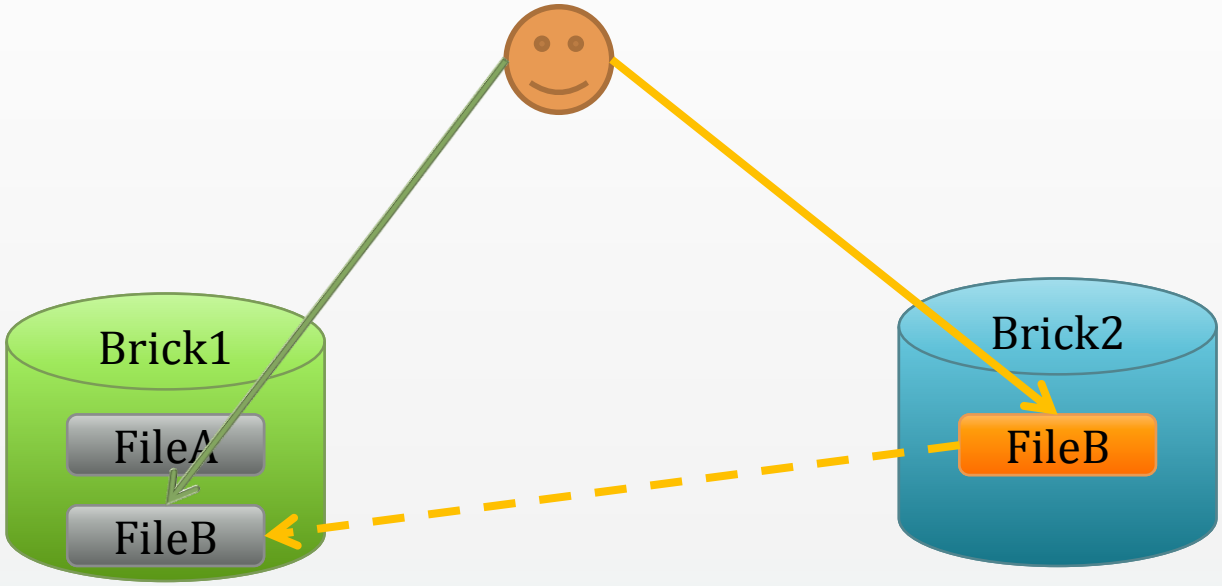
- 添加新节点，最小化数据重新分配
- 老数据分布模式不变，新数据分布到所有节点上
- 执行rebalance，数据重新分布

容量负载均衡



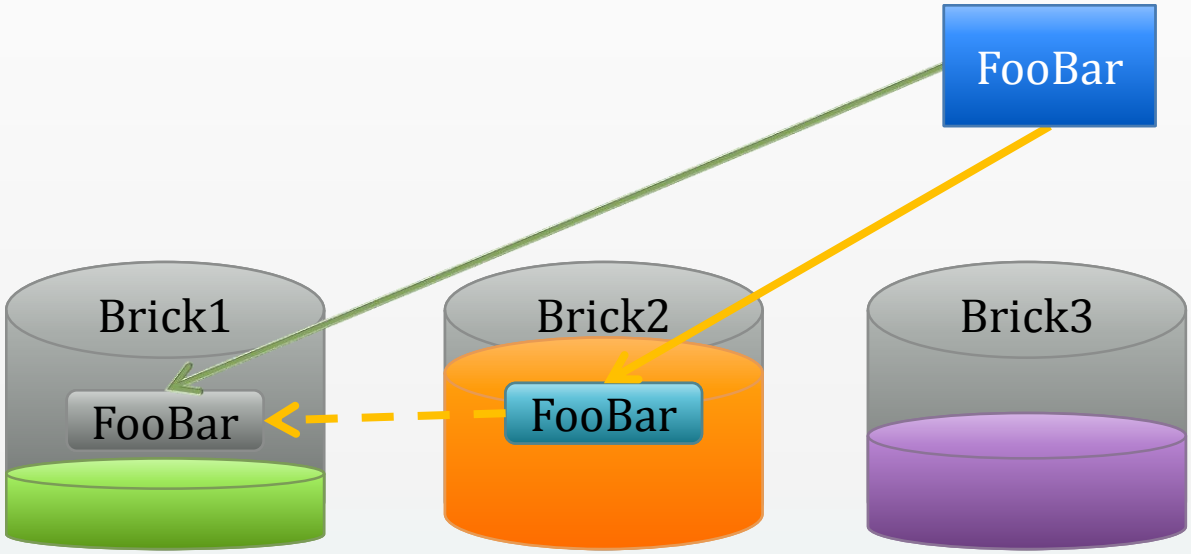
- Hash范围均衡分布，节点一变动全局
- 目标：优化数据分布，最小化数据迁移
- 数据迁移自动化、智能化、并行化

文件更名



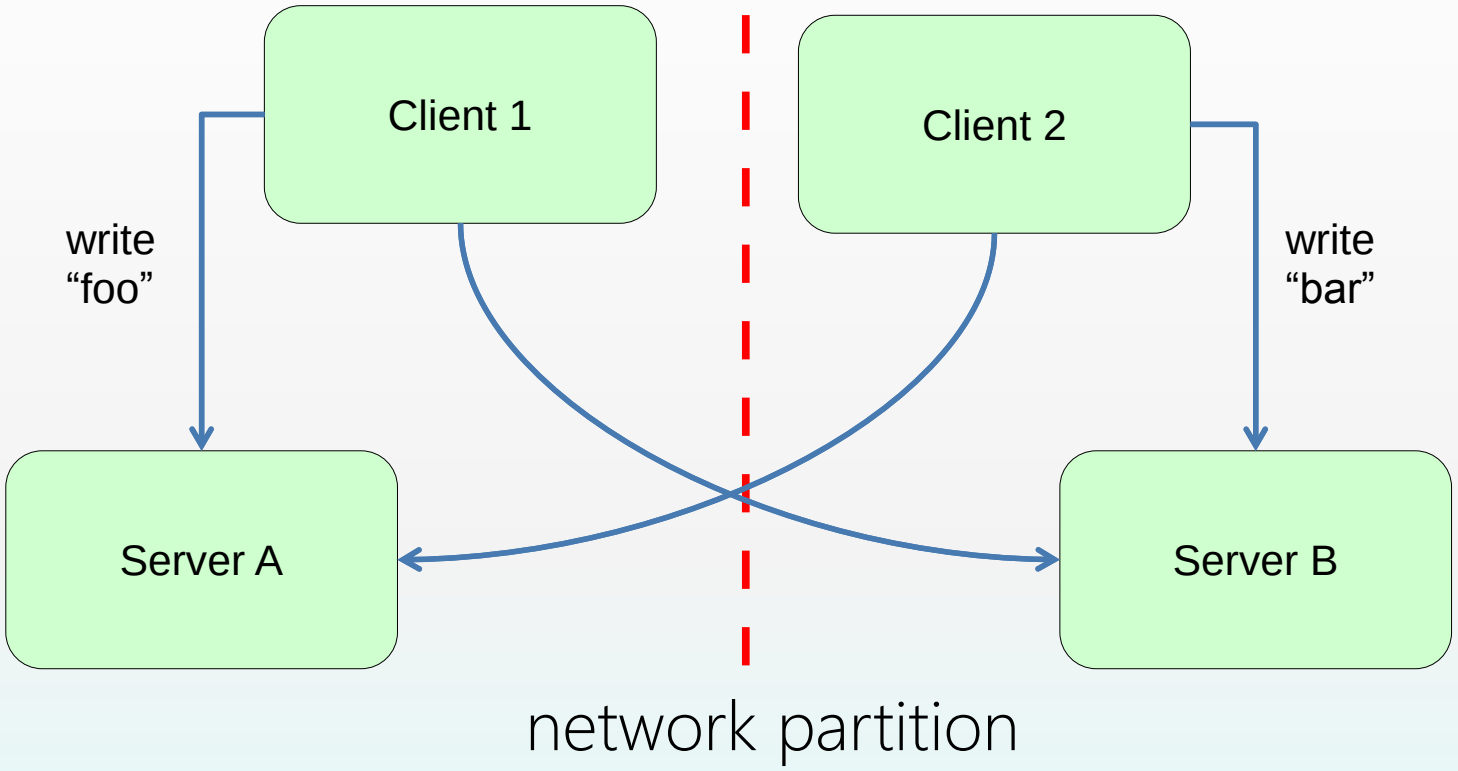
- 文件更名：FileA → FileB
- 原先的hash映射关系失效，大文件难以实时迁移
- 采用文件符号链接，访问时解析重定向

容量负载优先



- 设置容量阈值，优先选择可用容量充足brick
- Hash目标brick上创建文件符号链接
- 访问时解析重定向

Split Brain



- 裂脑如何产生的？
- 解决方法：1、报错处理；2、Quorum方法(N=2？)；3、仲裁机制

Q & A