常见 DFS 数据分布分析

作者：黄崇远

时间：2014/01/06

类型	详细
备注	该文档是几个典型分布式文件系统的总结，重点在于各个分布式文件系统的数据分布方式分析，顺带介绍系统架构及其优点缺点。
相关描述	✧ 其他相关文档请参考新浪博客 http://blog.sina.com.cn/huangchongyuan ✧ 有任何其他想法，可以邮件 874450476@qq.com ✧ 文档及相关资料下载请**个人 360 云盘** http://yunpan.cn/QGf2GDaRFpcDt 及百度文库、新浪爱问搜索。 ✧ 部分文档涉及到源码，有需要的博客留言，关注我的博客。 ✧ 欢迎加入 storm-分布式-IT 技术交流群（**191321336，群中有详细的资料，总体来说研究 storm 的人多点**），一起讨论技术，一起分享代码，一起分享设计。

目录

1 文档说明

研究分布式文件系统时间也不短了，接触过的文件系统也不少，趁着这 2014 到来之际，花点时间用来总结总结。

接触过的文件系统有 glusterfs、moosefs、lustre 及 hdfs 等，其架构简单顺带解说一点，总体来说分为元数据中心式及去中心式。其实除了 glusterfs，其他的都是元数据中心式的分布式文件系统。

对于文件系统的架构只进行简单的解说，现在主要对以上各种文件系统的数据分布方式进行总结分析，顺便从其数据分布方式中分析其性能，可用性，适用性等等。

2 典型 DFS 及其数据分布

2.1 Lustre

2.1.1 系统架构

Lustre 是个人最早接触的一个分布式文件系统，其给我留下最深刻的映象就是“无与伦比”的快，其存储速度是我见过用过的分布式文件系统中最快的。

先来张 Lustre 文件系统的架构图：

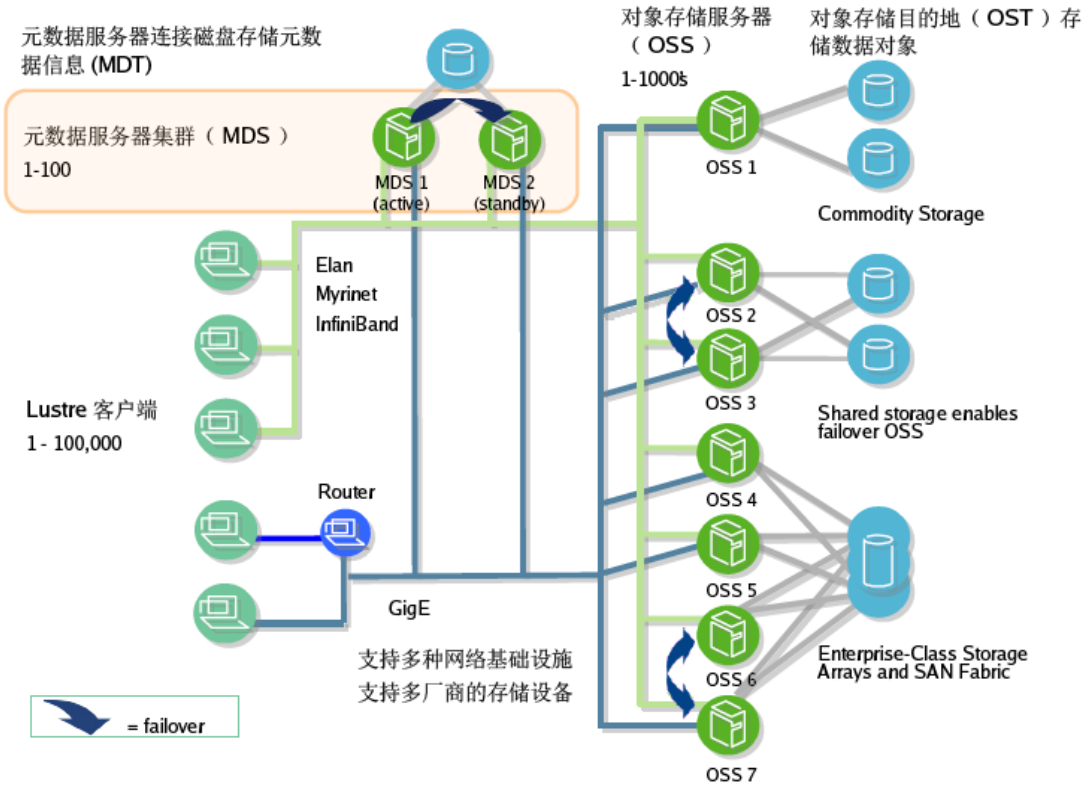

图 2.1.1 Lustre 架构图

从架构图中，我们可以看出 Lustre 是典型的元数据中心式的分布式文件系统，其元数据服务器为 MDS，用于存储文件的元数据信息，OSS 服务器用于存储实际的文件数据。Lustre 支持常规以太网，也支持快速的 IB 网络。

从结构上看，MDS 支持 active-standy 支持主备切换，OSS 也支持 failover，但实际上 MDS 的主备配置及 OSS 的故障恢复远没有想象中的好用。其主备配置也好，failover 也好只是支持服务器级别的切换，其底层真正用于存储对象结构 MDT 及 OST 是不支持主备及故障恢复的。一般底层对象存储会采用共享存储的方式来支持 MDS 的 active-standy、OSS 的 failover。但是这种配置不但会影响性能，配置起来复杂，安全性也没有想象中高，只要共享存储层出现故障，整个系统将无法正常使用。

2.1.2 数据分布方式

Lustre 支持两种数据分布方式，文件单个整体存储及文件分片(stripe)存储两种方式。

首先是文件整体存储，文件以单个文件的形式存储在 OST 中，不进行任何的数据分片、纠删码设置及 checksum(校验)设置等操作，这是一种比较常见的数据分布方式。

Lustre 还支持文件分片存储，也就是常说的 stripe 方式，很多 DFS 会提供这么一种数据分布方式。

Lustre 支持目录级的 Stripe 存储，即可以通过设置指定某个子目录的 Stripe 相关设置，包括 Stripe_count、Stripe_size、Stripe_ost。Stripe_size 指定子目录下的文件分片大小；Stripe_count 指定选择 OST 个数，即分片分布在多少个 OST 上；Stripe_ost 指定起始存储 OST 位置，系统默认为-1,即由 MDS 根据剩余容量及负载选择初始 OST，否则从指定的 OST 开始分片存储。

具体设置如下：

```
lfs setstripe -s Stripe_sizeM -c Stripe_count /mnt/SubDir
//Stripe_size 必须为 page_size 的整数据倍 X*64KB
//Stripe_count 小于等于 OST 数
```

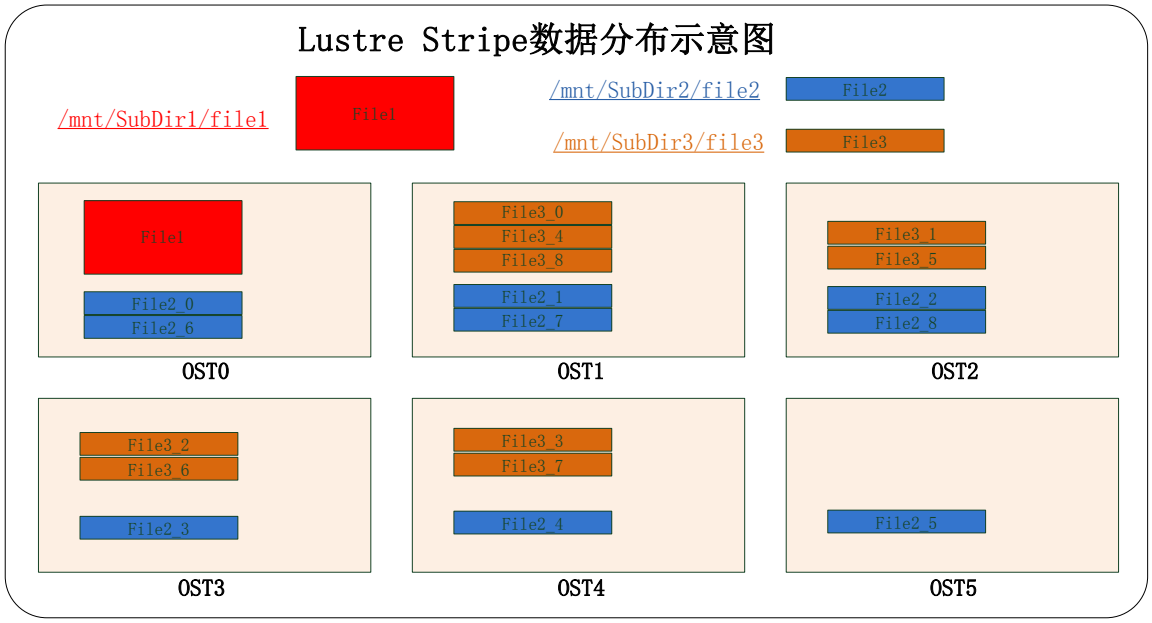

图 2.1.2 Lustre Stripe 数据分布示意图

如图示例：Lustre 存储由 6 个 OST 提供对象存储空间。/mnt/SubDir1 目录未进行 Stripe 处理，该目录下 file1 以文件的形式存储在单个 OST 中，例如存储在 OST0 中；/mnt/SubDir2 目录设置 Stripe_count=6，则如图所示，Stripe_size 大小的数据分别在 6 个 OST 中分布，且按照数据顺序轮询合并；/mnt/SubDir3 与 SubDir2 情形相似，Stripe_count=5。

2.1.3 系统优缺点

Lustre 可以设置具体子目录的 Stripe 参数，这种方式比较灵活，可以根据文件大小进行合适的 Stripe 目录设置。并且 Stripe 的数据分布方式比文件整体存储高效，无论是文件读还是写。

Lustre 是基于内核级的分布式文件系统。其文件的读写性能在分布式文件系统是比较快的(目前个人使用过，测试过的 DFS 中是最快的，其他分布式文件系统拍马难及)。鉴于其性能及其强悍的扩展性，多数用于高性能计算中。高性能，这正是 Lustre 最大的亮点。

在某种角度上说，Lustre 基本不提供数据的保护，无论是数据冗余保护，还是数据的纠删保护，还是数据校验保护等等；当然，同样，其元数据也存在单点问题。这是 Lustre 的一大弱点。

2.2 HDFS

2.2.1 系统架构

HDFS，或者说 Hadoop 大家会了解多些。其实对于 hadoop，个人了解的并不是很多，只是接触过一些相关的培训及进行过相关的了解使用，并不是特别的熟悉(正在一步一步的研究他呢)。所以，要是有不对的地方，欢迎大家指正。

先给大家上个架构图：

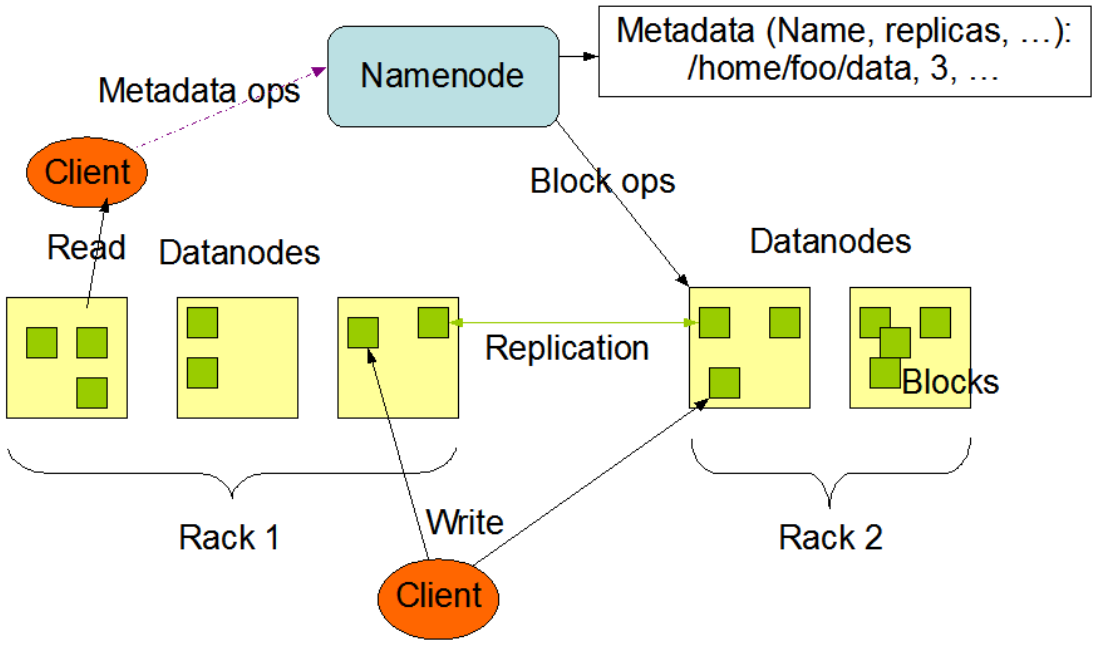

图 2.2.1 HDFS 架构图

相信很多人会很熟悉这张 hdfs 的架构图。如图所示，HDFS 由两部分组成，一部分是负责元数据存储的 Namenode，一部分是负责实际存储数据的 Datanodes。Namenode 是一个中心服务器，负责管理文件系统的名字空间、负责文件数据块的组织及处理客户端对数据的访问；而 Datanode 则是一般是一个节点一个，负责管理其所在节点的数据存储。

最新版本的 HDFS 会支持一个备用的 Namenode，负责定时的备份 Namenode 上的持久化的元数据信息，但实际上 HDFS 依然存在单点问题(熟悉 MFS 架构的朋友会发现，这种架构方式与 MFS 是如此的相似，又是如此的没用)。

2.2.2 数据分布方式

HDFS 只支持数据分块的方式存储，默认的数据块大小为 64MB。其与一般的 Stripe 存储方式不同(参考 Lustre Stripe 存储)，其会把文件分成一个个 64MB 大小的数据块，在 Datanodes 存储着一个个的 64MB 大小的数据块而不是数据块的集合。

HDFS 支持文件存储时创建副本，用于保证数据的安全性。并且系统保证文件的数据块的副本块不会出现在同一个 Datanode 上，避免某个 Datanode 失效时，文件无法访问。并且 HDFS 可以支持多个副本。

如下图所示：

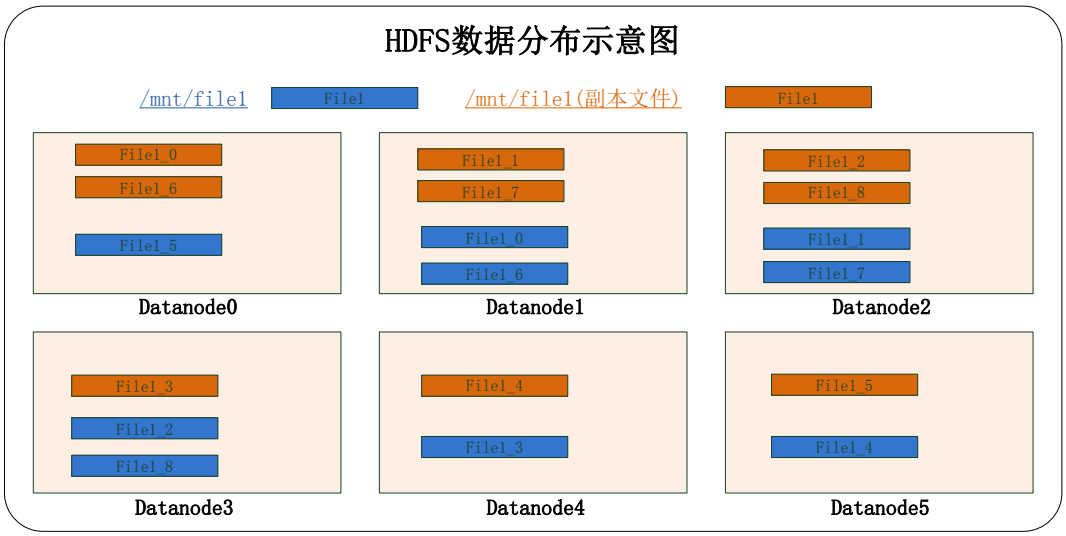

图 2.2.2 HDFS 数据分布图

副本数据块与原数据块不在同一个 datanode 中，这种方式保证了任何一个 datanode 失效，都有其他 datanode 上的副本数据块进行替换，从而保证了数据的安全性。

在 HDFS 的文件实际存储节点 Datanodes 中，数据块是以单独的文件块存在，而不是与 Lustre 那样将数据块轮询合并(对比 Lustre 数据分布图)。

HDFS 在数据分布中，最特别的是其对数据的校验。在存储文件时，HDFS 会为每一个 64MB 的数据块创建一个对应的 **checksum**，用于文件访问时对数据块的校验。这在分布式文件系统中是很不常见的。

文件在 HDFS 中是以私有格式存储的，只能通过系统的 API 进行访问。

2.2.3 系统优缺点

HDFS 作为存储本身来说，没有多少存储优势。例如其系统本身并不支持标准的 POSIX 接口，文件需要以专门的数据操作 API 进行文件数据操作，这在通用存储中是很不方便的方式；在性能上，其存储并没有显著的优势，其为每一个数据块创建一个 checksum，虽然在数据安全性上提高了，在某种程度上说会降低其存储效率；其元数据处理方式，即将元数据加载在内存中，这种方式可以说是优点，即提高了客户端与元数据及存储节点与元数据的交互效率，但是由于内存的扩充限制，这会导致其规模扩充受阻(这点与 MFS 又是极其相似)。

所以就目前来说，很少人会将 HDFS 单纯用于存储的。

然而，就目前研究热度来说，hadooop 绝对是首屈一指的。HDFS 其优势在于以其为基础的一系列衍生架构，就是大家所说的 hadoop 生态环境，包括 Nosql 数据库系统 HBase，Sql 操作化的 Hive，及数据仓库 Pig 等等。Hadoop 在批处理上有着无与伦比的优势，所以，配合其衍生架构，大多数人将其用于数据挖掘数据分析等。

2.3 MooseFS

2.3.1 系统架构

MFS 主要由四部分组成，元数据服务器 Master、元数据日志服务器 Metalogger、数据存储服务器 chunkservers、及挂载客户端 client。

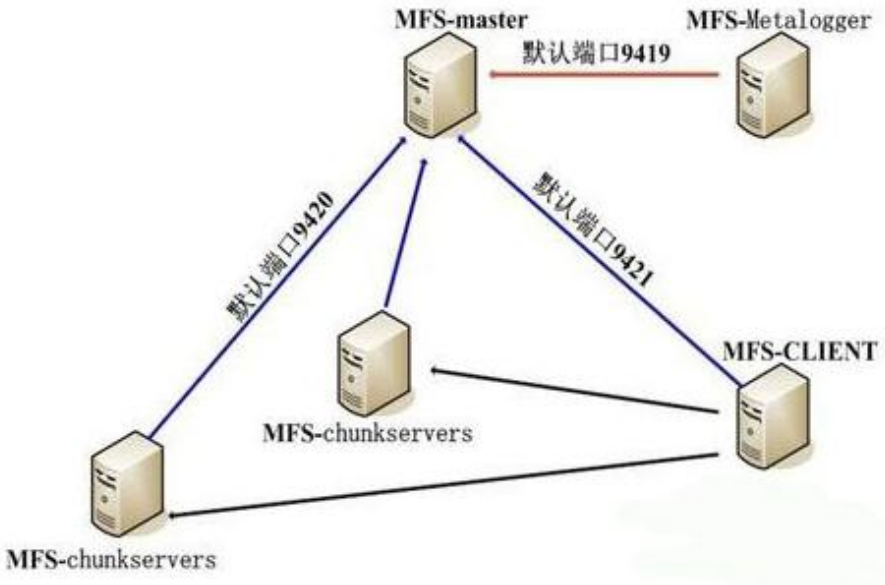

图 2.3.1 MFS 架构图

如图所示，其架构与一般的元数据中心化的分布式文件系统架构相似，多出来不同之处在于 Metalogger 即元数据日志服务器的增加(这是 1.5.X 版本之后添加的组件)。该部分组件用于定时备份 Master 上的元数据文件及 Master 使用期间产生的 changelog 文件。在元数据服务器当机的情况下，可以通过 Metalogger 上的元数据文件及 changelog 文件来恢复元数据，并且可以通过修复元数据及修改 Metalogger 的 IP 方式来替换当机的 Master。

2.3.2 数据分布方式

MFS 支持两种数据分布方式，一种是常规的以文件为单位的存储；另一种就是分块存储。

MFS 的分块存储与 HDFS 的分块相似，以 64MB 的数据块分别存储在不同的 chunkserver 中，并且不进行数据块再次合并存储。其实 MFS 与 HDFS 有很多相似的地方，之前所说的文件分块方式，其元数据在系统启动时置于内存的方式及添加日志服务器来备份元数据的方式等。但 HDFS 的数据是以私有格式存储，不支持 POSIX 标准协议的访问，这点是不同的，其次 MFS 的数据块不进行创建校验值(checksum)，这样做，降低了一定的数据安全性，但是提高了副本方式的存储效率。

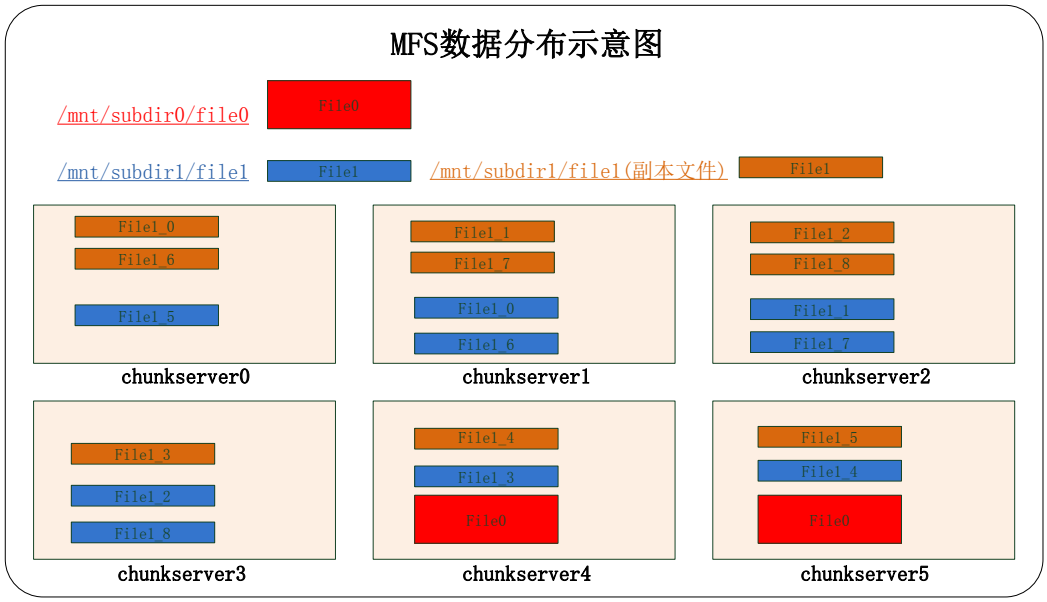

图 2.3.2 MFS 数据分布图

如图所示，可以看出 MFS 的两种数据存储方式，此外，MFS 还支持多数据副本，其副本的设置是是根据目录进行设置的，即设置目录内的文件为副本方式存储。以目录为单位，设置比较灵活。单文件存储方式的副本存储在另一个 chunkserver 中，而以数据块存储的方式，在保证副本块不在同一个 chunkserver 中，如此，则能保证 chunkserver 不会出现单点问题。

2.3.3 系统优缺点

由于 MFS 机制中，在系统启动时会将元数据缓存在内存中，这种方式在某种程度上提高了 chunkserver 与 master 的交互效率，但是，同样，由于内存扩充的局限性，这会导致 MFS 的扩展容易出现限制。根据官方说法，8G 的内存能够存储 2500KW 的文件元数据信息，这样的话，存储海量的小文件就很容易达到文件个数的限制，而不是 chunkserver 的容量限制。其实 HDFS 也会面临同样的问题，只是 HDFS 在元数据结构进行了优化，减少了单个文件的元数据 SIZE。

此外，就目前版本的 MFS，依然存在单点问题，虽然 1.6 版本之后添加了 Metalogger 组件，但是并不能很好的解决元数据的单点问题。实际上，系统至今仍然无法达到故障切换的目的，添加了日志服务器之后，只是在某些情况下出现故障后能够根据日志服务器进行元数据恢复。

在小规模存储上 MFS 还是比较有优势的，目前已经有不少公司在使用他进行相关的存储业务，或者在此基础上进行相关的二次开发。

2.4 GlusterFS

2.4.1 系统架构

GlusterFS 是个人研究时间最长一个分布式文件系统，总体来说对其还是比较熟悉，无论是从架构还是从他的一些系统机制来说。

GlusterFS 架构相对于其他分布式文件系统是最简单的架构，如图所示，除去网络层、

外挂 Samba 及 NFS 相关东西，就只剩下 client 及 Storage Brick，其实真正的存储核心就是 Storage Brick。

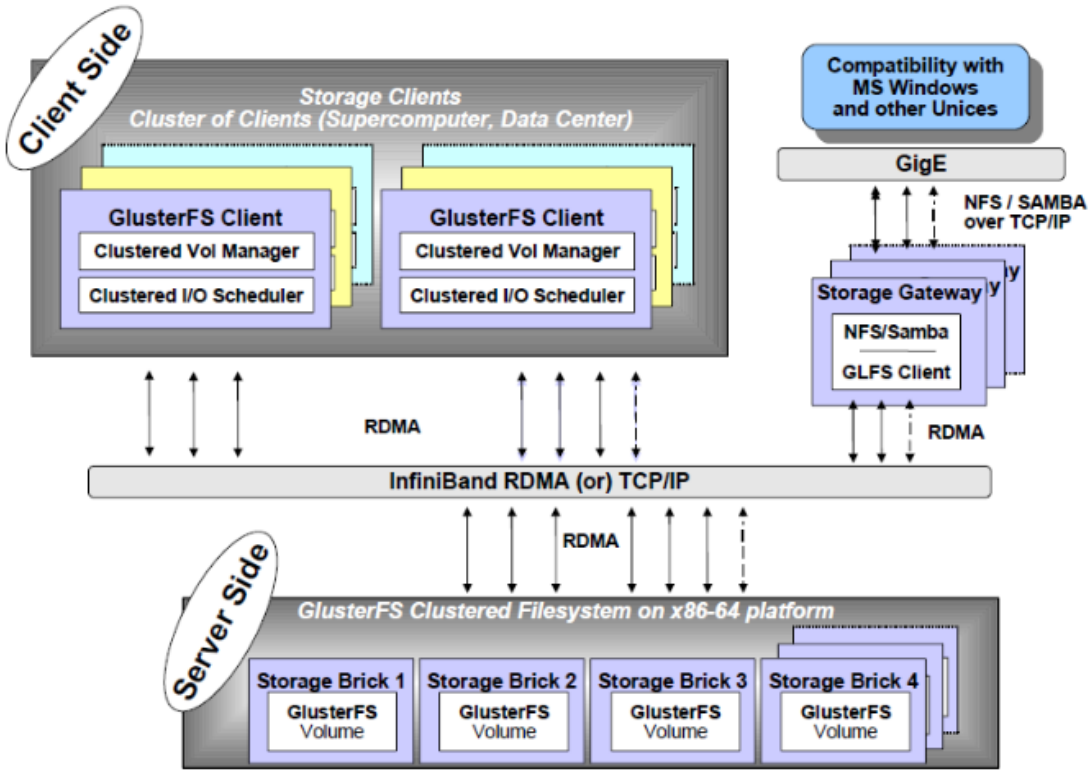

图 2.4.1 GlusterFS 架构图

GlusterFS 的架构是去中心式架构，即没有元数据中心结构，也就意味着其没有存储元数据的结构，这正是其异于 Lustre、HDFS 及 MooseFS 的地方，这也是其架构的最大特色。

那么其究竟是如何完成文件数据读写定位的呢？这就是去中心式的分布式文件系统的特色，GlusterFS 通过内部的 hash 算法实现文件的定位，通过这种方法代替元数据组件产生的作用。就目前来说，去中心式的分布式文件系统就个人所知，只有 GlusterFS。

2.4.2 数据分布方式

GlusterFS 是一个比较全面的分布式文件系统，包括其数据的分布方式。GlusterFS 支持三种数据分布，即其可以创建三种卷：分布式卷(Distributed)、条带卷(Stripe)及副本卷(Replicated)。

分布式卷(Distributed)，系统在存储文件时，根据文件路径及 brick 进行 hash 计算，根据计算结果决定其将存储于哪个 brick 之上，并且在 brick 中是以单个文件的形式而存在的。这种 hash 计算会做到尽可能的负载均衡。同样，在读取文件时，也会根据 hash 计算定位文件的位置。

条带卷(Stripe)，这种数据分布方式是大部分常见的分布式文件系统所支持的。GlusterFS 的条带化数据分布与 Lustre 的条带化数据分布很相似。系统根据 Stripe count 进行轮询分块，并且在单个 brick 中再进行数据块合并。并且其 Stripe size 大小默认为 128KB。

副本卷(Replicated)，文件在 brick 中会存储文件副本，并且副本数可以自由设置，但是会根据 brick 数进行相关的限制。系统还针对副本卷设计了文件自动修复机制，以保持副本文件的正确性。

GlusterFS 是数据分布最灵活的分布式文件系统，以上三种数据分布方式可以任意的搭

配使用。

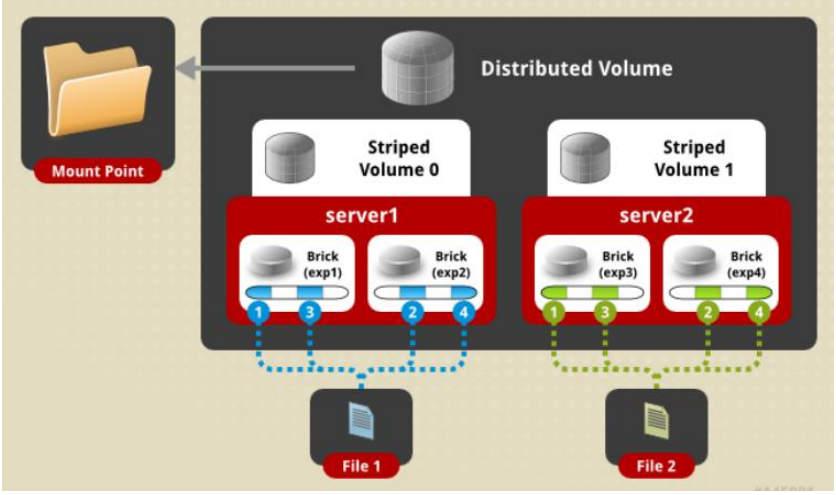

图 2.4.2-1 GlusterFS Distributed-Striped Volume

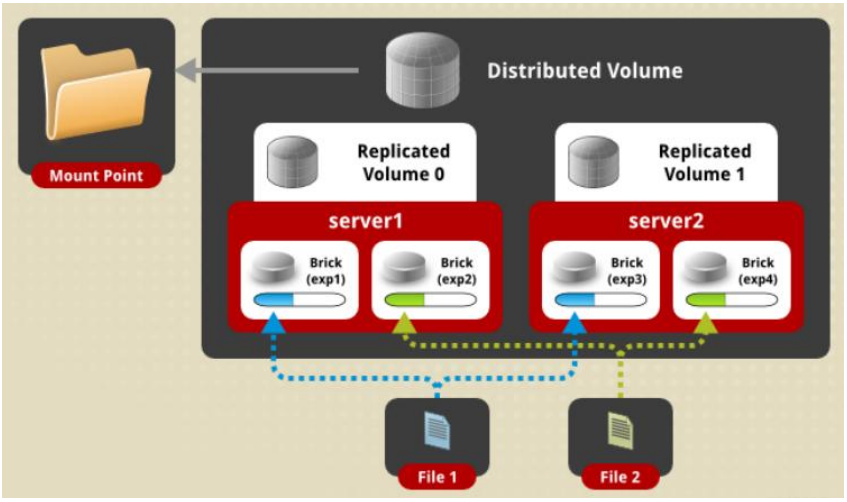

图 2.4.2-2 GlusterFS Distributed-Replicated Volume

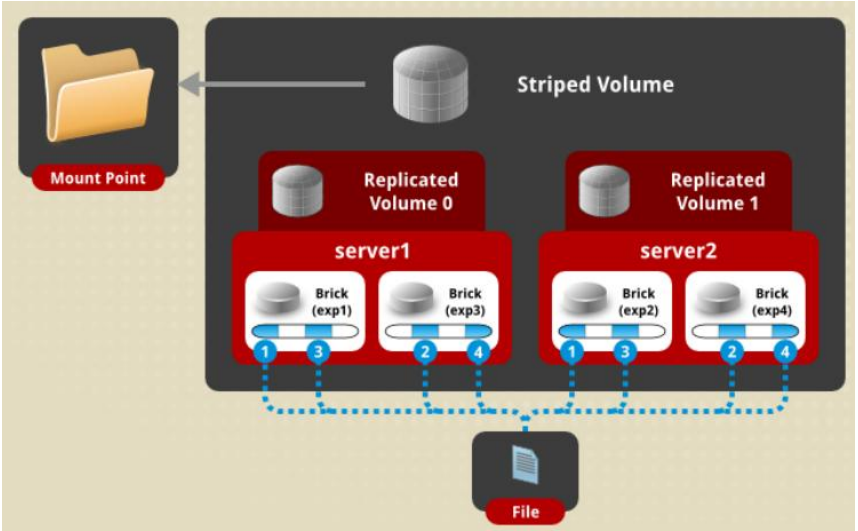

图 2.4.2-3 GlusterFS Replicated-Striped Volume

如图 2.4.2-1 所示，底层一共两个 server，每个 server 上做两个 brick，供四个 brick 做成 2x2 的 Distributed-Striped 卷。File1/2 是以分布式方式，即 hash 方式存储在不同的 brick 上；而从单个文件的角度看，如 File1，其内部是通过 Stripe 的方式进行条带化存储的，其 Stripe

count 为 2，通过文件分块编号，可以看出其数据是 Stripe 之后再组合成两个文件存储在两个 brick 上。

如图 2.4.2-2 所示，同样是 4 个 brick，做成 2x2 的 Distributed-Replicated 卷。File1/2 分别以单个文件为单位 hash 存储在不同的 brick 上，并且保证每个文件都会一个副本文件在其 brick 之外，从而保证了数据的安全。

如图 2.4.2-3 所示，系统是 2x2 的 Replicated-Striped 卷。File 在两个 brick 上 Stripe 切片存储，且在另外两个 brick 上保存了所有切片的对应副本。

在最新的 GlusterFS3.3.X 上，系统还是支持 Distributed-Replicated-Stripe 卷，即三种数据分布方式组成的系统卷，其数据具体分布方式由上面三个图就很容易推断出来了。GlusterFS 其卷的数据分布设置与其 brick 数目有很大的关系。

就目前所知的情况，Stripe 卷一般较少的人使用，其性能有待于提高，而最常使用的数据分布组合方式就是 Distributed-Replicated。总体上说，GlusterFS 提供了多种数据分布方式，并提供了灵活多变的组合方式，让我们在使用方便了许多。

2.4.3 系统优缺点

GlusterFS 最大的特点在于其无元数据的整体架构，但这只能说是他的一大特色，算不上其优点。据其官网上所说的无中心结构使其存储扩展近似线性，这也只是理论上的线性扩展，实际上 GlusterFS 的扩展性能比能达到 70%-80%就很不错了。这可能会比其他分布式文件系统扩展性能损耗上好一点点(中心化的分布式文件系统随着存储节点的增加，并行量上升，元数据负载会越来越大，导致其性能会损耗很大)，但随之而来的问题是，文件遍历时的效率呈直线下降，特别是在存储了大批量文件时。由于其是无元数据结构，导致文件遍历需要遍历需要实际遍历整个系统卷，而不是如其他分布式文件系统那样直接从元数据服务器中获取。这是其一大缺点。

GlusterFS 提供了比较多的功能，其多种卷的组合是一个，他还提供了文件修复机制、远程地域备份、在线扩容缩容、存储节点在线替换等等。此外，个人认为其最大的优点在于其具有一个完善的命令行管理接口，在众多分布式文件系统中，GlusterFS 管理起来是最方便的，所有操作都可以通过命令行工具进行管理，而不是像很多文件系统那样通过修改配置文件进行操作等等。

3 总结

从数据基本分布方式来看，目前开源的总体就分为单个文件存储、副本或者是镜像文件存储、及数据分块存储。在数据分块存储中，有典型的条带化 Stripe 存储，如 lustre 及 gluster 的 stripe 存储方式，以及 HDFS、MFS 文件真正分块(chunk)存储的方式。

分块存储在大多数情况下并不能提升文件写效率，当然这也和系统机制有关，如 lustre 的 stripe 存储就比文件单独存储的方式效率要高，但其他的分布式文件系统没有如此明显的差异，glusterfs 的 stripe 存储效率还极其的低，很少人会使用它。但分块存储在读效率上会有明显的提升，特别是针对大文件读时。

副本存储或者镜像存储的数据存储方式是一种比较普遍的数据安全保证的数据分布。实现起来也比较容易。还有一种提供数据保护的数据分布方式就是纠删码。目前在开源分布式文件系统中好像没有看到有人将其实现，但在一些商业分布式文件系统中能看到它的身影，其实现难度也比副本存储的难度大。一些国内的分布式文件系统研发公司将其作为开源分布式文件系统二次开发目标。这种数据存储方式类似于 RAID5，不仅能够提供数据保护，还节

约了硬件成本。

在技术群中不少人都会问一个相类似的问题“大家认为哪个开源分布式文件系统最好？”其实这个问题是很难回答的。就目前这些开源的文件系统中，各有各的特色，但同样他们都存在一定缺陷或者说是不足之处。哪怕是商业分布式文件系统，同样也会存在这个问题。没有最好，只有最合适！

如果你追求的高效，并且在一定程度上可以容忍少量数据的丢失，那么 lustre 将会是你的不二选择，至于数据的安全性，你可以使用第三方的软件或者系统来实现，或者说直接就忽略之；如果你的业务需求是做数据挖掘数据分析，那么不用多想了，没有人不选择 HDFS 而去选择其他的；如果你想使用起来功能比较完善，并且管理起来比较方便，那么 GlusterFS 会是不错的选择；如果你想使用它来存储小文件，好吧，以上那些除了 MFS 适合一点点(不少公司使用 MFS 进行小规模存储小文件)，其他的也都是扯犊子的；当然，如果确实需要存储海量小文件，也是有专门为小文件设计的开源分布式文件系统，例如国人开发的 FastDFS、MogileFS 以及淘宝开源的 TFS，只是对于这些分布式文件系统个人就不是很熟悉了，也只是知道个大概而已。

PS：以上纯属个人观点，个人资质有限，如果有错，欢迎指正。