GlusterFS/Lustre/MooseFS 实践总结

作者：黄崇远

时间：2013/10/16

类型	详细
备注	该文档是根据个人对一些开源分布式系统的一些实践总结。
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**），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

该文档为个人对一些开源分布式文件系统实践之后的总结，主要包括系统安装分析、可用性、可靠性、扩展性等方面。

由于个人工作及个人好爱原因，最近对几个开源分布式文件系统进行了实践，包括环境的搭建、性能的测试及功能的测试等等，总觉得要把这些实践过后的经验记录下来，希望能给其他人一些帮助。

根据这些总结，希望能对一些分布式文件系统爱好者或者开发者在进行原型开发时能有所帮助，主要是一些性能指标的分析。

2 系统实践总结

2.1 GlusterFS

2.1.1 系统概况

GlusterFS 是一款比较成熟稳定的开源分布式文件系统，其闻名于业界主要是其无元数据架构。其架构舍弃了常规的分布式文件系统元数据部分，文件及目录通过算法计算进行定位代替了常规的元数据服务器。

其组成只有存储服务器及客户端两部分，没有元数据。

其他系统概述就不多说了，网上比较多对系统的架构啊，及一些其他的概况说明的文档。

2.1.2 系统搭建

在 GlusterFS 实践的环节中，系统的搭建部署是第一步。我在进行 GlusterFS 系统的搭建是使用源码进行安装，版本是 3.3.1，官方网站提供的指导文档是 3.3.0 版本，最新的版本是 3.4.0。

在做分布式文件系统技术选型时，系统的搭建复杂性是一个比较重要的参考指标，因为涉及到集群的搭建，这是一个比较大的工程。

GlusterFS 系统的搭建，比较简单。系统管理也相对来说比较简单，系统提供比较多的命令行工具。GlusterFS 无元数据服务器，配置时不需要进行配置文件创建修改等等操作，只需使用命令行工具将存储节点添加到虚拟存储池中，然后使用命令行工具将存储节点磁盘 mount 出来的目录进行系统卷的创建即可，最后使用客户端进行 mount 挂载，就可以使用其统一命名空间了。具体过程可以参考网上一些博客及其他资料，有时间我也会整理出来。

总体来说，GlusterFS 的搭建部署还是比较简单的。

2.1.3 系统可用性

2.1.3.1 系统可靠性

在所有用于生产环境的分布式文件系统中，对系统的可靠性或多或少的都有所要求。最低要求保证数据的安全性，能够应对一般的突然事件（机器断电，系统宕机等），这就是系统可靠性的最直接体现。

首先 GlusterFS 是无元数据结构，所以很多分布式文件系统所谓的元数据安全问题可以忽略不计，例如元数据是否存在单点故障，元数据故障恢复等等。

在数据安全性方面，GlusterFS 在创建系统卷的时候提供数据副本机制，并且可以随意指定副本数（不过子卷的个数与副本个数有耦合），数据副本保证了数据安全性，在部分存

储节点出故障时，副本能够进行数据恢复。

在系统的稳定性方面，GlusterFS 已经开源多年，有较多的实验及生产实例，目前系统版本已经比较稳定了，越来越多的公司或者机构将其用于生产环境。

GlusterFS 使用 AFR 方式（replica）创建卷，当一个节点出现故障之后，客户端在短暂的时间内无法连接系统，一段时间后，系统自动恢复正常（启用了数据副本）。故障节点恢复之后，副本数据会自动更新到重新上线的节点机，也可以使用 self-heal 工具手动恢复。

2.1.3.1 系统功能扩展

首先 GlusterFS 是用户态的分布式文件系统，其提供了标准的 POSIX 接口，其存储的文件，使用 ext3/4 文件系统能够正常访问。

GlusterFS 提供 ARF、DHT 及 Stripe 三种系统卷的创建方式，DHT 将文件 hash 分布存储，ARF 提供数据副本，Stripe 类似条带化存储，满足了多种的用户需求。

GlusterFS 支持在线扩容，并且在添加新的子卷之后进行负载均衡（命令行工具），其容量扩充由于是无元数据结构，所以无性能损耗（据说，这个目前没有测试）。

2.1.4 系统性能

所有基于用户态开发的分布式文件系统相对于基于内核开发的分布式文件系统性能都会较差。基于用户态的分布式文件系统大部分使用 fuse 模块进行数据处理，这会导致数据多次与内核交互，产生性能损耗。

GlusterFS 的性能，个人没有严格的进行测试过，不过从一些技术群友中（测试过 GlusterFS）了解到，GlusterFS 在没有优化的情况下，其性能并不是很好。

2.1.5 系统适性

GlusterFS 底层存储是以文件形式存在的，即使是 Stripe 方式创建系统卷，文件在子卷中的表现方式是 ext3/4 能够访问的文件，不过只是其中相应文件的部分而已。这种存储方式相对于部分文件系统私有数据格式，在数据出现故障时，我们可以进行底层备份查看，但由于其是有 ext3/4 组织的，安全性较低。

总体而言，利用 GlusterFS 做大文件存储的较多，如果是做小文件存储，一般不会选择其作为系统原型。虽然 GlusterFS 号称对小文件支持良好（无元数据结构）。

GlusterFS 有原生态的 Linux/Unix 客户端，没有 windows 客户端，想要在 windows 下进行访问，只能标准协议进行访问，如 ISCSI 协议。在 windows 下通过 SAMBA 进行统一目录导出。

2.1.6 二次开发

GlusterFS 开源度较高，其代码量大约在 5-6W 行左右，且使用 C 进行开发，其代码高度模块化，所以其代码的阅读难度相对比较小。

在原有代码基础上进行部分模块修改及部分代码优化会比较容易。

开源分布式系统需要用于生产环境，必然面对系统优化，系统定制，及相应功能修改等问题，GlusterFS 在这方面是比较有优势的。

2.1.7 实际应用

个人目前对市场方面了解的也不多，但从相关技术及部分技术论坛的反馈来看，目前研究 GlusterFS 的人相当多。许多公司已经将 GlusterFS 用于生产环境。

2.2 Lustre

2.2.1 系统概况

Lustre 是个人最先研究的一款开源分布式文件系统，其最主要的特点就是基于对线存储并且高性能。其在当前世界级高性能计算中占有很大的比重。

其架构主要分为 MDS，OSS 及 Client 三个部分。至于其他系统相关的概述就不多说了，网上比较多的资料，并且官网有详细的操作手册。

2.2.2 系统搭建

Lustre 的系统搭建是一大特色，当然是指其搭建难度无系统能及啊。Lustre 系统搭建相当具有难度。本人进行系统实践时，选择的是最新的系统版本 2.4.0。

Lustre 安装时首先需要对内核进行升级，其与操作系统内核高度耦合。在安装时有较多的系统依赖，并且有较多的系统组件需要进行安装，并且需要对操作部分服务组件进行升级。

总而言之，Lustre 的安装是比较复杂的，这是它的一个弱点，

2.2.3 系统可用性

2.2.3.1 系统可靠性

Lustre 目前最新版本仍然存在一个比较致命的问题，就是元数据单点问题。虽然根据最新官方资料，MDS 的 MDT 可以扩充，但是经过实践，发现扩充的 MDT 根本没有生效（也可能是个人配置有问题）。

Lustre 提供服务器级别 Failover，但底层 MDT 无法进行故障恢复，通常底层使用共享存储提供 MDT，主备 MGS 共用 MDT，从而实现服务器级别的 Failover，OSS 服务器级别的 Failover 实现也类似。这种级别的 Failover 实际作用不大，底层共享存储一经损坏则系统就无法使用。

Lustre 不提供数据副本，所以对数据安全性保障不了，这也是 Lustre 历经多个版本仍然未解决的问题。

对于 Lustre 的安全性，可以提供如下一种架构：Lustre+keepalived+drbd，使用第三方类似 drbd 进行数据网络备份，通过心跳检测软件 keepalived 或者是 heartbeat 检测主服务器的健康状态，当主服务器出现故障时，心跳检测软件执行自动切换工作。这种架构虽然能够解决数据安全性的问题，但是由于过多使用了第三方软件，整体架构稳定性下降，并且后期的维护难度提高。

2.2.3.2 系统功能扩展

Lustre 提供标准 POSIX 接口，ext3/4 系统能够像访问正常文件那样访问 Lustre 中存储的文件数据，OST 上的数据格式为 Lustre 私有格式。

Lustre 提供了强悍的扩容能力，根据官方文档 OST 能够扩充至 8150，客户端能够支持 131072 个。这些数据目前在当前条件下无法进行验证。

Lustre 支持两种存储方式，一种是普通存储方式，文件以整个文件的形式存储在某个 OST 上，还有一种是 Stripe 分块方式，文件以规定数据块大小条带化到指定个数的 OST 上，数据块大小可以设置，最小 64K。

2.2.4 系统性能

Lustre 使用 Stripe 存储方式，文件分块，当读写文件时，OST 能同步读写，经过测试，其大文件性能相当好，在额定网络带宽情况下增加客户端，能够迅速的达到网络瓶颈。

Lustre 的小文件读写效率也没有想象中的差。

对几个关键参数进行调优，例如根据 OST 数设置合理的 Stripe 数，设置 Stripe 的块大小，起始位置等等，效果并不明显。

2.2.5 系统适性

Lustre 总体而言是比较适合大文件读写的，而且用于高性能计算的居多。将 Lustre 应用于实际生产环境的话，其系统安全性需要好好进行设计。

2.2.6 二次开发

Lustre 整体工程使用 C 语言进行开发设计，核心代码接近 50W 行，且涉及到内核编程。虽然其整体代码比较完善，但二次开发难度比较大，需要较高的编程水平。

2.2.7 实际应用

07 年到 09 年国内研究 Lustre 的人挺多，到现在研究 Lustre 系统的人比较少，各大技术群及技术论坛 Lustre 活跃人数较少。

目前普通机构或者是公司或者是个人将 Lustre 用于生产环境的较少。

就个人所知，曙光及高能计算所还在使用优化过的 Lustre，其他机构或者是公司未有耳闻。

Lustre 对于分布式爱好者来说，还是一个不错的研究对象，毕竟其在性能上有着无可匹敌的优势，我们可以好好的研究了解下它的架构及内部实现机制等等。

2.3 MooseFS

2.3.1 系统概况

MooseFS（简称 MFS）其最大的系统特点就是元数据服务器在系统运行时将元数据放入内存中，这也就意味着，当存储节点有数据写入或者读取是，客户端或者文件元数据信息是直接从内存中获取的，不用通过磁盘的读写，这个速度是非常快的。这也是 MFS 的一大特色。

就其架构来说，MFS 有 Master、Metalogger、Chunkserver 及 Client（Mfsmount）等组成部分。其中日志服务器（Metalogger）负责备份 Master 的元数据信息及日志变化文件。

至于其他 MFS 的基本情况就不一一列举了。

2.3.2 系统搭建

MFS 的搭建是比较简单的，个人搭建的 MFS 是最新的版本 1.6.27，总体搭建过程比较简单。

安装的过程中需要注意的是需要为其创建专门的系统用户，并在编译的时候指明。

MFS 系统的配置（master 配置，chunkserver 配置等等）需要修改相关配置文件，其有较多的配置文件，存储节点提供的目录也是通过配置提供，而不是像 GluserFS 那样通过命令行方式配置。

2.3.3 系统可用性

2.3.3.1 系统可靠性

在 1.5.X 版本以前，MFS 无 metalogger 部分，master 挂掉之后，系统就挂了。在 1.6.X 版本之后添加了 metalogger（日志服务器），号称解决了元数据单点问题。这也是个人在实

践之后比较无语的地方。

MFS 的故障恢复比较麻烦，其 metalogger 只是相当于一个 metadata 备份服务器的作用，并没有想象中强大。Master 故障之后（无法启动，机子宕机），如果需要使用 metalogger 中的备份元数据恢复，则需要重建一个 master，然后通过 metalogger 中的备份来恢复，或者直接把 metalogger 转换为 master。这是一个比较繁琐的过程。

可以考虑使用类似 keepalived 心跳检测工具，配合脚本，在 master 宕机之后实现自动接管和自动恢复。

MFS 在挂载目录下支持数据副本，使用工具在对挂载目录下的子目录指定副本数，保证了数据的安全。

2.3.3.2 系统功能扩展

MFS 扩容平滑，但其扩容规模受 master 内存限制，因为 MFS 元数据是放入内存中的，也就意味着扩容导致大量文件增加（100W files->300M RAM），进而增加 master 的内存压力，而 master 的内存扩充能力有限，这也就限制了其扩容规模。

MFS 提供回收站功能，使用回收站功能，必须先进行 meta 挂载。使用 mfssettrasdtime 工具对目录或者是文件进行回收站彻底删除时间间隔。根据个人实践，其回收站定时删除机制并不是很准确（时间）。

2.3.4 系统性能

MFS 使用 chunk 分块存储，每个块大小为 64M，不足 64M 文件，只存储在一个 chunk 中，超过 64M 文件分别存储在不同的 chunk 中。这种分块的存储方式对大文件存储是有利的，能够提高一个大文件的并行读写效率。

对于小文件，小文件存储的瓶颈一般在于元数据，而 MFS 相对于一般的分布式文件系统的不同之处在于，系统运行时，元数据放入内存中，这样大大的提高了元数据的处理能力，所以相对于一般的分布式文件系统（非小文件文件系统），其对小文件的性能也有一定的优势。

2.3.5 系统适性

MFS 自身的分块存储特性决定了其对大文件读写的优势，而其元数据的特殊结构又对小文件读写也有一定优势。

总体来说，MFS 还是比较适用于大文件的存储，进行一定的优化之后，也可以用于小文件的存储。

2.3.6 二次开发

MFS 整体代码使用 C 语言编写，代码量适中，其代码高度模块化。二次开发选择 MFS 进行优化修改，还是比较方便的。

2.3.7 实际应用

目前在国内使用 MFS 的人不少，不少公司已经将 MFS 用于实际生产。据个人了解到，有部分公司将 MFS 做为原型，用于存储图片等。

国内研究 MFS 的技术群及相关论坛还是比较活跃的，并且也较多的博客发表相关的博文。

3 其他 DFS 概览

GoogleFS 分布式文件系统是 Google 为自身业务量身定制的分布式文件系统，适合大型的、海量的、分布式的数据访问。无论是元数据还是存储数据都有数据副本，支持服务器主备切换，不提供 POSIX 标准接口，只提供数据访问的专用 API。目前 GoogleFS 不提供开源，官网只提供一份系统的设计论文。

HDFS（hadoop）是近年来最火的 DFS 之一，其在批处理领域有着无可匹敌的优势。同样其局限性也比较强，只适合批处理，并且其不追求数据响应时间。HDFS 元数据存在单点问题，提供数据自动均衡。目前国内相当多的公司在使用 HDFS，大部分用于日志分析，数据挖掘等。HDFS 开源比较早，并且有相当多的开源子项目，已经形成一个生态群，其使用 JAVA 进行系统开发，在系统效率上，可能会低点。

FastDFS 是国人庆余开发的，目前已经开源，其支持冗余备份，支持负载均衡，不提供标准的 POSIX 接口，数据以私有数据格式存储，其适合中小规模的集群存储应用，特别适合互联网小文件存储应用。FastDFS 使用 C 编写，大约有 5W 行核心代码。目前已经有一个相当具有规模的 FastDFS 论坛。大家可以去看看。

MogileFS 是一款专门为小文件存储设计的分布式文件系统，支持负载均衡，无单点故障，支持数据副本。但是其有一个很大的缺点，就是不支持对文件的随机读写，也就意味着它只适合静态存储，例如 html 方式静态化处理的文件。目前国内也是较多公司在使用，不过大部分是互联网公司，例如 yupoo，digg，土豆，豆瓣，1 号店，大众评，搜狗等等。MogileFS 使用 Perl 实现。

Ceph 是一款比较年轻的开源分布式文件系统，使用 C 开发，无单点故障，支持 POSIX 标准接口，具有复制和容错功能。唯一的缺点就是太年轻，不够稳定，目前研究居多，就个人了解，还没有用于生产环境的。

TFS 是淘宝为自身海量小文件存储量身定制的分布式文件系统，使用 C 开发，目前已经开源，具有数据副本，支持主备切换。就实际应用的话，只了解到淘宝内部在使用，其他情况不详。

4 总结

只有清楚了解各个分布式文件系统的各自特点，才能根据自身定制指标进行原型选择，选择一个合适的分布式文件系统作为原型，往往能事半功倍，站在巨人的肩膀上，我们才能看的更远。

在调研分布式文件系统时，可以从以下几点进行考察，环境搭建、系统可用性（视对安全性要求高低）、扩展性（规模）、二次开发（优化度）及系统适用性。

还有好多其他的一些 DFS，基于文章篇幅，就不一一列举，这是一个很大的圈子，使劲游我们也达到不了边界的。

5 啰嗦几句

写博客写文章的老规矩了，感觉这篇文章，这篇博客对你有用吗？如果觉得有用的话，来关注我的博客吧，会时常更新哦，也期待你加我的技术群 191321336，这里现在很多研究 storm，研究分布式的朋友哦。

顺便展望下下，大数据时代已经来临，大数据的处理（hadoop 啊、storm 啊等等），大数据的存储（DFS、集群 NAS 等等）必将是一个难点，也是一个热点。来吧，我们一起来搞它，搞 storm，搞分布式！

6 参考文献

(1) Gluster_File_System-3.3.0-Administration_Guide-en-US.pdf
(2) http://blog.csdn.net/liuaigui/article/details/6284551 贵哥的 gluster 相关专栏
(3) gluster 组件安装及群集部署_MapleZhou_新浪博客
(4) Lustre Manual 2x.pdf
(5) MooseFS 分布式文件系统安装向导.pdf -version 田逸大神版
(6) MooseFS 权威指南.pdf 网友整理的实用资料