Fast DFS调研报告

一、项目背景

1.文件系统

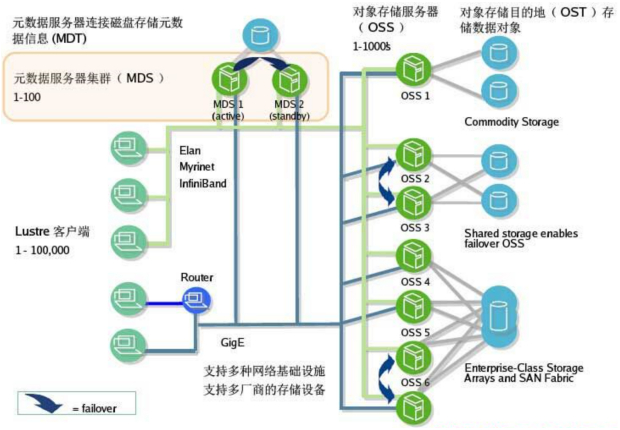
- **在计算机中，文件系统是一个管理数据和他们名称的结构和逻辑规则。当前有很多不同种类的文件系统，他们拥有不同的逻辑结构、性能、安全性、规模等等。文件系统一般可以按类型的不同分为磁盘文件系统(Disk File System)、分布式文件系统(Distributed File System)、专用文件系统(Special File System)等。**

2.分布式文件系统

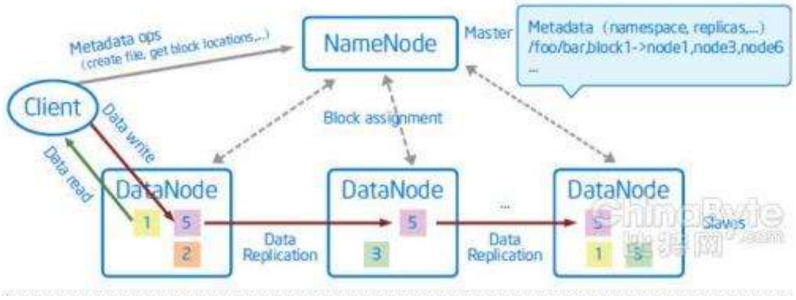
- **分布式文件系统不像传统的磁盘文件系统那样在同一个文件存储块上管理数据，而是通过网络协议来管理数据。**
- **分布式文件系统在设计时是为了把大型任务划分成若干小任务。然后分而治之，从而达到每个结点处理少量的数据任务，节点之间互不妨碍的效果。**
- **同时分布式文件系统通过保存数据的副本保证了即使发生单点故障，也不会造成严重的数据丢失。**
- **另外，分布式存储系统的工作就是将数据分散存储在多台独立设备设备之上；相比于集中式存储方式在系统结构上可以扩展，利用多台存储服务器来共享内存负载。**

3.典型分布式文件系统介绍

- **Lustre：**
  - **Lustre分布式文件系统是1999年卡内基梅隆大学的Peter J.Braam所发起的研究项目，Lustre是典型的元数据中心式的分布式文件系统，MDS（元数据服务器集群）用于存储文件的元数据信息，OSS（对象存储服务器）用于存储实际的文件数据。**
  - **Luster支持文件整体存储和文件分片两种存储方式。整体存储就是将文件整体存储在OST（对象存储目的地）中；而分片存储就是将文件分块存储在子目录Stripe下，Stripe有size（文件分片大小），ost（起始存储位置），和count（分片数量）三个参数。**
  - **Lustre的优点是可以设置具体子目录Stripe的参数，从而灵活的存储不同大小的文件，相比于单一的整体存储方式更为高效。但是Lustre不提供数据保护，也存在着元数据存储上单点问题，这两点决定了Lustre在安全性和容错性上的短板。**



- **HDFS（Hadoop Distributed File System）**
  - **HDFS是为Hadoop架构所设计的分布可扩展轻量级分布式文件系统，其由负责元数据存储的Namenode和负责存储实际数据的Datanodes组成。Namenode是一个中心服务器，负责管理文件系统的名称空间、文件数据快的组织以及客户端对数据的访问；而Datanodes就是一个个存储实际数据的节点。**
  - **HDFS只支持文件的分块存储，其将每个文件默认分为若干个64MB的数据块，集中存储在一个Datanode上。在存储某个文件的时，HDFS会为其创建另一个Datanode以存储其副本，从而在一定程度上保证了数据的安全性。**
  - **HDFS本身在性能等方面没有很大的优势，其优势在于以其为基础的一系列衍生架构，配合其衍生架构，大多数人将其用于数据挖掘数据分析等领域。**



- **GFS（Google File System）**
  - **GFS是Google公司为了存储海量搜索数据而设计的专用文件系统，其最为鲜明的特点是独特的容错机制以及自动负载均衡技术。GFS系统中包括一个独立的GFS Master控制服务器，多台GFS Chunkserver数据块服务器以及多个GFS Client客户端。**
  - **和HDFS一样，上传到GFS中的文件都被分割成固定大小的Chunk，每一个Chunk在不同的Chunkserver中保存多份。Chunk数据块用以存储Chunk数据，并根据指定的Chunk标识符和字节范围来读写块数据。**
  - **GFS Master存储Chunk的元数据，接受来自各个Chunkserver的周期性信息，并向各个Chunkserver发送指令**
  - **在GFS中，GFS服务器生成诊断日志用于记录许多关键事件（如Chunkserver的启动和关闭）以及所有的远程调用的请求和相应。在空间允许的情况下，GFS会尽量保存这些日志。在系统发生故障后，通过对远程调用日志的分析，可以重建交互历史从而诊断出错误。**

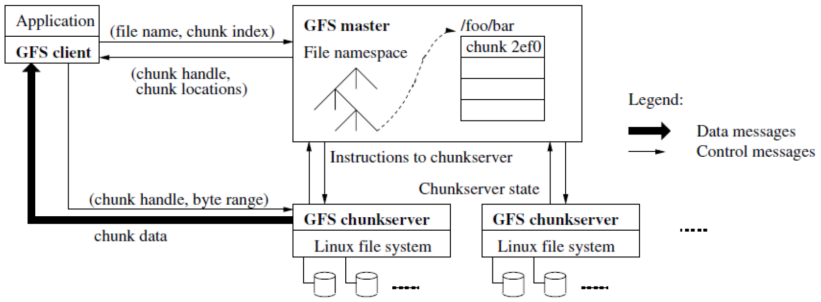
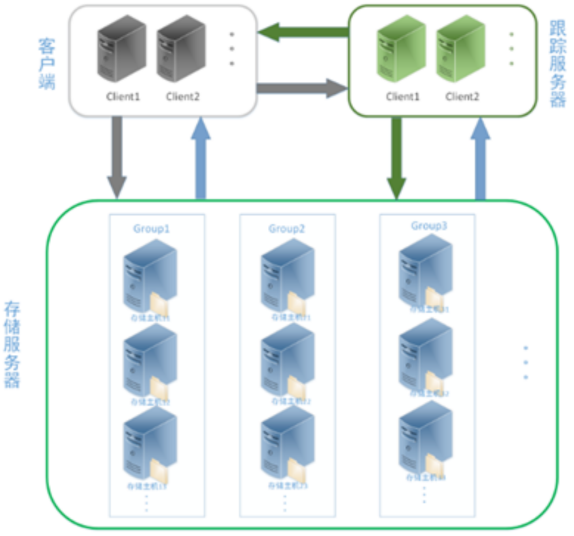


Figure 1: GFS Architecture

- **FastDFS**
  - **Fast DFS是一款类GFS的开源分布式系统，通过纯C语言实现，支持Linux、AIX等UNIX系统。Fast DFS是为互联网应用量身定做的分布式文件系统，与现有的类GFS系统相比，其主要有轻量化、分组存储和对等结构三个特点**
  - **轻量化**
    - **tracker server在内存中记录分组的storage server的状态等信息，不记录文件索引信息，占用的内存量很少。**
    - **不对文件分块存储，更加简洁高效**
    - **文件ID由storage server生成，因此不需要存储文件索引信息**
    - **代码量较小**
  - **分组方式：存储集群由多个组构成，一个组由多台存储服务器（**storage server**）构成，同组内的**storage server**互相备份，存储的文件完全一致。**
  - **对等结构：**Fast DFS**集群中的**tracker server**也是多台，各台之间是对等关系且相互独立，因此不存在单点问题。**Storage server**有一个单独的线程来完成对某一台**tracker server**的连接和定时报告。**



二、立项依据

1.**应用场景**

- **FastDFS是为互联网应用量身定做的一套分布式文件存储系统，非常适合用来存储用户图片、视频、文档等文件。对于互联网应用，和其他分布式文件系统相比，优势非常明显。**

2.**用户量**

- **截止2012年底至少有25家公司在使用FastDFS，其中有好几家是做网盘的公司。其中存储量最大的一家，集群中存储group数有400个，存储服务器超过800台，存储容量达到6PB，文件数超过1亿。以下是部分使用Fast DFS的用户列表：**

  支付宝（http://www.alipay.com/**）**
  京东商城（http://www.360buy.com/**）**
  淘淘搜（http://www.taotaosou.com/**）**
  飞信（http://feixin.1008**/**）**
  赶集网（http://www.ganji.com/**）**
  淘米网（http://www.61.com/**）**
  迅雷（http://www.xunlei.com/**）**
  蚂蜂窝（http://www.mafengwo.cn/**）**

3.**易于修改**

- **FastDFS由纯C语言实现，在阅读其源码时，在语言层面上不存在太大的障碍。相比于其他分布式文件系统，其5.2万行的代码量减轻了在阅读源码时的负担**

三、Fast DFS**未解决的问题**

1.**动态负载均衡**

- Fast DFS**现有的负载均衡算法属于静态负载算法，该算法为每个存储组定义了一个评价值**Vi，Vi**的计算方法是：**Vi = Vi0 + Ti / Tmax (Vi0**是上一层该组的**Vi**值，**Ti**为该组内各**storage server**的存储空间之和，**Tmax**是所有组中**Ti**的最大值**)。Tracker server**在选择存储组的时，总是选择**Vi**最高的两个存储组。而**Vi**显然与该存储组当前的空间利用率、性能、连接数无关，这就意味着总存储空间大但是当前负载很重，连接数很大的**storage server**可能被选中。也许可以通过改进**Vi**的算法和选择**storage server**的机制实现动态负载均衡**

2.**去中心化**

- Fast DFS**通过增加冗余的**tracker server**解决了单点故障的问题，但是多个**tracker server**就意味着**storage server**需要同时向每个**tracker server**报告自身的磁盘剩余空间、文件同步状况、文件上传下载次数等状态信息；而这些任务的完成就要求每个**storage server**都要在后台运行着一个单独的线程来完成对**tracker server**的连接和定时报告。这种机制必定意味着资源的浪费。**

四、可能的解决方案

1.**动态负载均衡的解决方案**

- **为每一组定义负载系数Li，**Li =∑Lij = ∑Rij * Nij**（其中**Nij**为第**i**组第**j**个服务器的连接数；Rij = Uij / Tij，是第**i**组中的第**j**个**storage server Sij**的存储空间利用率**Rij**）。**

- **为每一个storage server定义性能负载比，定义第i组第j个storage server的性能负载比为：PLij = Pj / Lij；其中Pj为第j组的性能。**

- **每一次先选择Li最小的两个存储组，再选择这两组内∑PLij最大的组来接受存储请求**

- **新算法中加入了空间利用率、性能和连接数三个因子，相比于现有的算法，在选择storage server时，综合考虑到storage server在存储上的压力（Rij）和传输上的压力(Nij)；在选出的两个组内，有根据性能负载比进行进一步筛选，这是出于对传输速率的考虑。因此新算法显然比原来的算法考虑更为全面，能在各种环境下保持较好的传输效果。**

2.**去中心化的解决方案**

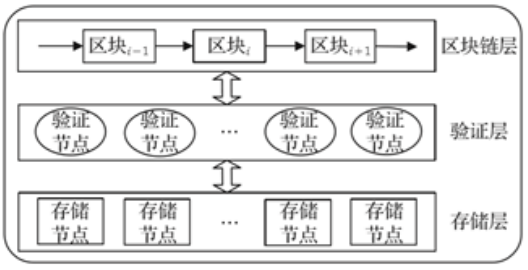


图2　DMB模型架构

- **存储层：与传统分布式存储系统的存储层结构相同，但是在存储过程中产生元数据，这些元数据包括数据副本位置信息、副本书等。**

- **验证层：每个验证层节点都将区块链副本保存在本地存储，同时维护本地区块链状态。在元数据存储阶段，验证节点功能包括收集和验证用户的签名信息、采集元数据信息、构造元数据区块、记录新区块。在元数据验证阶段，验证节点功能包括接受用户的验证请求、检查本地区块链副本状态、同步本地区块副本状态、检索给定签名的元数据并返回结果**

- **区块链层：区块的块体负责保存元数据，区块链层保存元数据区块链的全局状态，所有验证节点可以查询该层来同步本地区块链状态。当存储层中发生单点故障或者宕机等问题时，分布式存储系统可以通过读取该层的元数据区块链来恢复关键数据。元数据区块链是只读的，如果用户存储的数据位置信息发生了变化，那么只能重新生成新的元数据信息、构造新的元数据区块。**