FastDFS 去中心化

一 、项目背景：

1. 增强对 C 语言和操作系统（尤其是分布式文件系统）的了解，增强编程能力，培养合作精神，学会自主查找文献和应用所学知识。

2.分布式文件系统的概况：

相对于本机端的文件系统而言，分布式文件系统是一种允许文件通过网络在多台主机上分享的文件系统，可让多机器上的多用户分享文件和存储空间。在这样的文件系统中，客户端并非直接访问底层的数据存储区块，而是通过网络，以特定的通信协议和服务器沟通。借由通信协议的设计，可以让客户端和服务器端都能根据访问控制清单或是授权，来限制对于文件系统的访问。相对地，在一个分享的磁盘文件系统中，所有节点对数据存储区块都有相同的访问权，在这样的系统中，访问权限就必须由客户端程序来控制。

3.分布式文件系统的性能评测：

一个衡量分布式文件系统性能的方式是：它需要用多少时间来完成服务请求。在传统的系统中，完成请求所需要的时间包括了实际的硬盘访问时间，和一小部分的中央处理器处理时间。但在一个网络文件系统中，由于分布式架构的关系，远程访问动作会产生额外的经常性负担，包括：把请求从客户端送到服务器端的时间、把回应从服务器端传回客户端的时间、以及这两个传输过程中用来运行网络传输协议的中央处理处时间。

4.FastDFS：

（1）.FastDFS 是基于互联网应用的分布式文件系统，主要解决了大容量的小文件存储和负载均衡的问题。FastDFS 由纯 C 实现，支持 Linux、FreeBSD 等 UNIX 类 Google FS，提供了 C、Java 和 PHP API 为互联网应用量身定做。其具有轻量级、高性能的优点，特别对于小文件存储和传输具有较高效率。

（2）.FastDFS 服务端由：跟踪服务器(tracker server)、存储服务器(storage server)、客户端(client)组成。

tracker server：跟踪服务器，主要做调度工作，起负载均衡的作用。在内存中记录集群中所有存储组和存储服务器的状态信息，是客户端和数据服务器交互的枢纽。Tracker 是 FastDFS 的协调者，负责管理所有的 storage server 和 group，每个 storage 在启动后会连接 Tracker，告知自己所属的 group 等信息，并保持周期性的心跳，tracker 根据 storage 的信息，建立 group==>[storage server list]的映射表。

Tracker 需要管理的元信息很少，会全部存储在内存中；另外 tracker 上的元信息都是由 storage 汇报的信息生成的，本身不需要持久化任何数据，这样使得 tracker 非常容易扩展，直接增加 tracker 机器即可扩展为 tracker cluster 来服务，cluster 里每个 tracker 之间是完全对等的，所有的 tracker 都接受 stroage 的信息，生成元数据信息来提供读写服务。

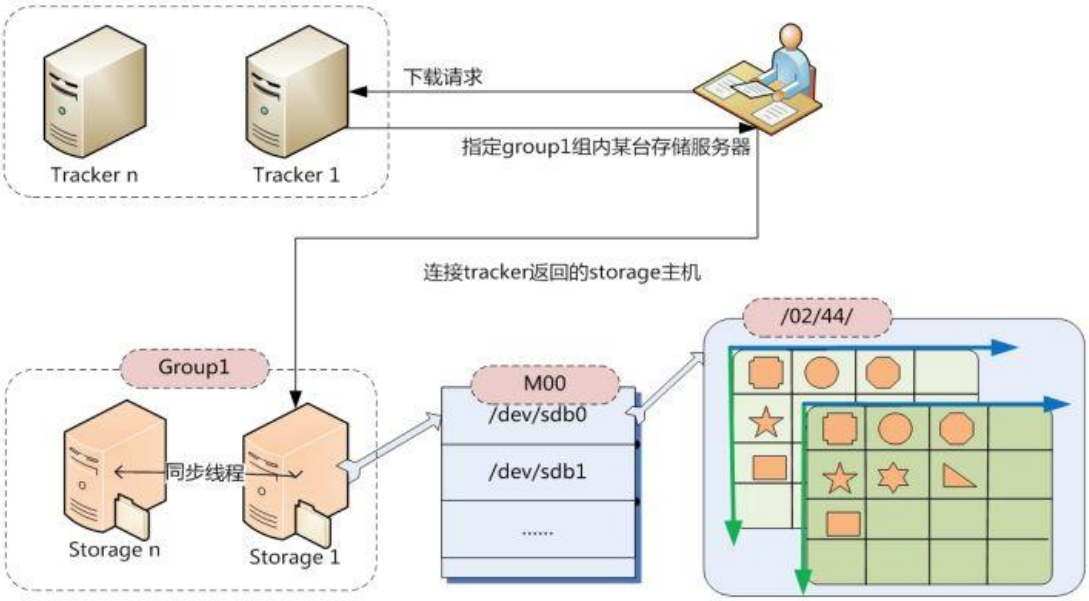
storage server：存储服务器，文件和文件属性(meta data)都保存在存储服务器上。Storage server 直接利用 OS 的文件系统调用管理文件。它以组(group)为单位组织，一个 group 包含多台 storage 机器，数据互为备份，存储空间以 group 内容量最小的 storage 为准。以 group 为单位组织存储能方便的进行应用隔离、负载均衡、副本数定制（group 内 storage server 数量即为该 group 的副本数），比如将不同应用数据存到不同的 group 就能隔离应用数据，同时还可根据应用的访问特性来将应用分配到不同的 group 来做负载均衡；缺点是 group 的容量受单机存储容量的限制，同时当 group 内有机器坏掉时，数据恢复只能依赖 group 内地其他机器，使得恢复时间会很长。

client：客户端，作为业务请求的发起方，通过专有接口，使用 TCP/IP 协议与跟踪器服务器或存储节点进行数据交互。FastDFS 向使用者提供基本文件访问接口，比如 upload、download、append、delete 等，以客户端库的方式提供给用户使用。

(3).快速定位文件：

1.通过组名 tracker 能够很快地定位到客户端需要访问的存储服务器组，并将合适的存储服务器提供客户端访问；

2.存储服务器根据“文件存储虚拟磁盘路径”和“数据文件两级目录”可以很快定位到文件所在目录，并根据文件名找到客户端需要访问的文件。



5.去中心化：

“去中心化”出现在拥有众多用户或众多节点的系统中，每个用户可连接并影响其它节点。对于分布式文件系统，去中心化的存储可以使网络变得稳健且有弹性。将数据集中或者把文件系统的核心集中在某一部位会使得系统安全性降低。对于 FastDFS 来说，如果核心结构 tracker 被攻击，则文件系统失效。

实现去中心化算法：DHT(distributed hash table)

用来将一个关键值（key）的集合分散到所有在分布式系统中的节点，并且可以有效地将消息转送到唯一一个拥有查询者提供的关键值的节点（Peers）。这里的节点类似散列表中的存储位置。分布式散列表通常是为了拥有极大节点数量的系统，而且在系统的节点常常会加入或离开（例如网络断线）而设计的。在一个结构性的延展网络（overlay network）中，参加的节点需要与系统中一小部分的节点沟通，这也需要使用分布式散列表。分布式散列表可以用以创建更复杂的服务，例如分布式文件系统、点对点技术文件分享系统、合作的网页缓存、多播、任播、域名系统以及即时通信等。

分布式散列表本质上强调以下特性：

- 离散性：构成系统的节点并没有任何中央式的协调机制。
- 伸缩性：即使有成千上万个节点，系统仍然应该十分有效率。
- 容错性：即使节点不断地加入、离开或是停止工作，系统仍然必须达到一定的可靠度。

分布式散列表的结构可以分成几个主要的组件[2][3]。其基础是一个抽象的**关键值空间**（keyspace），例如说所有 160 位长的字符串集合。**关键值空间分区**（keyspace partitioning）将关键值空间分区成数个，并指定到在此系统的节点中。而**延展网络**则连接这些节点，并让他们能够借由在关键值空间内的任一值找到拥有该值的节点。

当这些组件都准备好后，一般使用分布式散列表来存储与读取的方式如下所述。假设关键值空间是一个 160 位长的字符串集合。为了在分布式散列表中存储一个文件，名称为 filename 且内容 data，我们计算 filename 的 SHA1 散列值——一个 160 位的关键值 } k——并将消息 put(k,data)送给分布式散列表中的任意参与节点。此消息在延展网络中被转送，直到抵达在关键值空间分区中被指定负责存储关键值 k 的节点。而 (k,data)即存储在该节点。其他的节点只需要重新计算 filename 的散列值 k，然后提交消息 get(k)给分布式散列表中的任意参与节点，以此来找与 k 相关的数据。此消息也会在延展网络中被转送到负责存储 k 的节点。而此节点则会负责传回存储的数据 data。

实例：FastDHT

6.动态负载均衡算法：

动态负载均衡算法包括: 最少连接数,最快响应速度，观察方法，预测法，动态性能分配，动态服务器补充，服务质量，服务类型，规则模式等。

最少的连接方式（Least Connection）：传递新的连接给那些进行最少连接处理的服务器。当其中某个服务器发生第二到第 7 层的故障, BIG-IP 就把其从服务器队列中拿出，不参加下一次的用户请求的分配，直到其恢复正常。

最快模式（Fastest）：传递连接给那些响应最快的服务器。当其中某个服务器发生第二到第 7 层的故障，BIG-IP 就把其从服务器队列中拿出，不参加下一次的用户请求的分配，直到其恢复正常。

观察模式（Observed）：连接数目和响应时间以这两项的最佳平衡为依据为新的请求选择服务器。当其中某个服务器发生第二到第 7 层的故障，BIG-IP 就把其从服务器队列中拿出，不参加下一次的用户请求的分配，直到其恢复正常。

预测模式（Predictive）：BIG-IP 利用收集到的服务器当前的性能指标，进行预测分析，选择一台服务器在下一个时间片内，其性能将达到最佳的服务器相应用户的请求。(被 BIG-IP 进行检测)

动态性能分配(Dynamic Ratio-APM):BIG-IP 收集到的应用程序和应用服务器的各项性能参数，动态调整流量分配。

动态服务器补充(Dynamic Server Act.):当主服务器群中因故障导致数量减少时，动态地将备份服务器补充至主服务器群。

服务质量(QoS）:按不同的优先级对数据流进行分配。

服务类型(ToS): 按不同的服务类型（在 Type of Field 中标识）负载均衡对数据流进行分配。

规则模式：针对不同的数据流设置导向规则，用户可自行。

二、立项依据：

1.去中心化：尽管 fastdfs 具有不少优点，但是对于中心结点依赖过重

2.动态负载均衡：动态负载均衡稳定性更好、系统效率更高。

三、前瞻性/重要性分析：

分布式文件系统具有高稳定性、高扩展性、高性能等优点。虽然部署起来较为复杂，可是种种优点使得分布式文件系统成为未来文件系统的方向之一。因此选择分布式文件系统来作为研究方向。目前互联网的发展越来越快，用户之间的交流不光光依靠文字，还需要图片、语音、视频等方式，这些方式的共同点在于属于中小文件，而 FastDFS 特别适合以中小文件(4KB 到 500MB)为载体的在线服务，因此我们选择 FastDFS 作为我们的研究课题。

虽然 FastDFS 具有种种优点，但是由于 FastDFS 的结构是中心化的，如果核心结构 tracker 出现问题，会使得整个文件系统崩溃，因此将中心化的 FastDFS 改为去中心化的文件系统具有实际意义。

FastDFS 原先采用静态负载均衡的算法。为了提高效率和稳定性，可以将之改进为动态均衡算法。

四、相关工作：

1．去中心化算法：DHT
2．去中心化系统：FastDHT
3．Koorde : A Simple Degree-Optimal Distributed Hash Table
4．动态均衡：
   1.《Fast DFS 分布式文件系统负载均衡算法的改进研究》 熊建波， 武汉理工大学
   2.《基于 FastDFS 的目录文件系统的研究与实现》 周子涵， 电子科技大学
   3.《基于 FastDFS 架构的小文件存储系统的设计与实现》 李长平 哈尔滨工业大学
   4.《FastDFS 负载均衡算法的改进及其在水土保持网站系统的应用》 曾致远 华中科技大学