链滴

FastDFS 简介

作者：ZhangVincent
原文链接：https://ld246.com/article/1577938644531
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

<h2 id="什么是FastDFS">什么是 FastDFS</h2>
FastDFS 是一个开源的轻量级分布式文件系统。它解决了大数据量存储和负载均衡等问题。特别合以中小文件（建议范围：4KB < file_size <500MB）为载体的在线服务，如相册网站、视频网等等。
<h2 id="FastDFS相关概念">FastDFS 相关概念</h2>
<h3 id="三种角色">三种角色</h3>
一个 FastDFS 集群中存在三种角色：Storage、Tracker 和 Client
<h4 id="Storage">Storage</h4>
Storage，存储节点，负责数据的物理存储。一个集群中有多个 Storage，Storage 启动时可以定分组名称，相同分组名的 Storage 组成一个组，同组内 Storage 存储的数据应该是一致的，从而到数据冗余备份的目的。
Storage 的存储依赖于本地文件系统。如果一台机器挂载了 5 个磁盘，则这 5 个磁盘在本地文系统中对应 5 个数据存储目录。Storage 会在每个数据存储目录下创建 256 个二级子目录，在每个级子目录下再创建 256 个三级子目录（共 256*256=65536 个三级子目录）。用户上传的一个文件最终被 Storage 存储为某个三级子目录下的一个本地文件。
另外，Storage 还会定期向 Tracker 汇报自身信息，主要包含自身所属的分组、向同组内其他 Strage 同步数据的进度等。
<h4 id="Tracker">Tracker</h4>
Tracker，跟踪者节点。一个集群中可以有多个 Tracker，他们的地位是同等的。Tracker 通过接 Storage 的汇报信息，维护整个集群的状态，并对来自于 Client 的请求进行负载均衡。
由于 Storage 会定期向所有的 Tracker 汇报信息，因此每个 Tracker 都会维护一个从分组名到 Storage 列表的映射关系。Client 向集群写入数据或者读取数据时，都需要先与任意一个 Tracker 连接
（1）当写入数据时，Tracker 根据一定的策略，从映射关系中选择某个分组的某个 Storage 返回给 Client，然后 Client 连接该 Storage 写入数据。
（2）当读取数据时，结合文件的 ID【ID 中包含分组名等信息】，Tracker 根据一定的策略，从映射系中选择某个分组的某个 Storage 返回给 Client，然后 Client 连接该 Storage 读取数据。
<h4 id="Client">Client</h4>
即请求发起方。Client 通过 API 向 FastDFS 写入、读取、删除数据等。
<h3 id="其他概念">其他概念</h3>
<h4 id="组">组</h4>
一个 FastDFS 集群可以有多个组，每个组包含多个 Storage。同一组内部，Storage 的地位是等的，文件的上传、删除等操作可以在同组内任意 Storage 上进行，这些操作最终会同步给同组内其 Storage。
<h4 id="Metadata">Metadata</h4>
元数据，是与文件相关的其他属性，有 Client 在上传文件时指定。文件的元数据和文件一起，保存在 storage 上。元数据表现为键值对形式，如 width:1024,height:768
<h4 id="文件id">文件 id</h4>
Client 端上传一个文件后，最终文件被 Storage 保存为一个本地文件。同时 Storage 向 Client 返回文件 id，后续 Client 端可以通过该文件 id 读取或者删除文件。
一个文件 id 的样式如下：
<pre><code class="highlight-chroma">/组名/一级目录/二级目录/三级目录/文件名.文件后缀名</code></pre>
对应的一个样例：
<pre><code class="highlight-chroma">/group1/M00/00/00/wKgz6lnduTeAMdrcAAEoRmXZPp870.jpeg</code></pre>
其中，

组名：即存储该文件的 Storage 所属的组名
目录：文件在 Storage 中存储的路径。一级目录对应着某个挂载的磁盘，二级和三级目录则是 Strage 在该磁盘下建立的子目录。
文件名：一个文件名由 storage ip+ 文件创建时间 + 文件大小 + 文件校验码 + 一个随机数拼接

，将其二进制串进行 Base64 编码而成
文件后缀名：文件名后缀由 Client 端上传文件时指定

<h2 id="FastDFS中文件操作">FastDFS 中文件操作</h2>
FastDFS 中文件操作，主要涉及到 Client 端向集群上传文件、从集群读取文件，以及集群内部
件的同步。
<h3 id="文件上传过程">文件上传过程</h3>
总体过程：

Client 选择一个 Tracker
Client 端请求 Tracker 获取到一个 Storage 服务器的 ip 和端口
根据 ip 和端口，向 Storage 服务上传文件
Storage 接收到文件后，选择一个三级目录存储该文件，并生成文件 id，返回给 Client（此时文
还未同步到该组内其他 Storage 上）

涉及细节：

Client 选择一个 Tracker
一个集群中可以有多个 Tracker，但是每个 Tracker 的地位都是对等的，因此 Client 可以与任何一个
racker 进行通信从而获取到一个 Stroage

Tracker 选择一个 Storage
Tracker 根据接收到的文件上传请求，首先确认一个 Group。确认 Group 方式可以是如下之一：<br

（1）用户上传时指定 Group 名称
（2）Tracker 根据轮询的方式选择一个 Group
（3）Tracker 选择剩余存储空间多的 Group
Tracker 从 Group 中确定一个 Storage。从 Group 中确认一个 Storage 可以是如下方式：
（1）轮询 Group 内的 Storage
（2）根据 Ip 排序
（3）根据优先级排序（优先级在 Storage 上配置）

Storage 选择存储路径
Storage 接收到 Client 的文件上传请求后，根据如下规则确定一个存储目录（磁盘）：（1）轮询（
）剩余存储空间最多的优先

Storage 生成文件 id
Storage 生成文件 id，并保存文件。文件 id 的生成规则在前面已有介绍。

<h3 id="文件下载过程">文件下载过程</h3>
一个文件的下载过程总体如下：

Client 向 Tracker 发送文件 id
Tracker 根据文件 id 确认一个 Storage
Client 向 Storage 发送文件 id
Storage 向 Client 返回文件内容

文件上传过程中，FastDFS 的某台 Storage 接收到文件后，会立马向 Client 返回文件 id，此时
件还未同步到同组内的其他 Storage 机器。为了保证 Client 根据文件 id 能正确获取到文件内容，Tra

ker 在选择 Storage 时，遵循如下规则：
（1）从文件 id 中解析出文件上传时 Storage 对应的 group、ip、文件创建时间
（2）根据 ip，如果该 Storage 还存在的话，则肯定能从该 Storage 中读取到文件，因此向 Client 返回该 Storage
（3）对于同组内的其他 Storage，如果某个 Storage 的同步时间戳等于文件创建时间，且当前时间文件创建时间的时间差，大于文件同步最大时间（如 5 分钟），则可认为经过最大同步时间后，文件被同步到了该 Storage 上，因此可向 Client 端返回该 Storage
（4）对于同组内的其他 Storage，如果文件创建时间早于某个 Storage 的同步时间戳，则该文件已同步到了该 Storage 上，因此可向 Client 端返回该 Storage
（5）如果当前时间与文件创建时间的时间差，大于同步延迟阈值（如一天），则认为经过同步延迟值时间，文件肯定已经同步了。所以可返回同组内任意一 Storage
<h3 id="集群内部文件同步过程">集群内部文件同步过程</h3>
一个文件被上传后，源 Storage 向 Client 端返回文件 id。同时，源 Storage 内部会写一份 binlog 文件，该文件记录了文件名、文件创建时间等信息（不记录文件内容）。源 Storage 会对同组内的他每个 Storage，创建一个对应的同步线程。该同步线程会读取 binlog，根据 binlog 信息向对应的 Storage 进行文件同步。同步线程还会记录下同步进度，以便于系统重启后的继续同步。
同步进度会作为汇报信息的一部分，汇报给 Tracker。在 Client 端读取文件时，Tracker 会根据同步度来确定对应的 Storage。
比如一个 group 内有 A、B、C 三个 storage server，A 向 C 同步到进度为 T1 (T1 以前写的文件都经同步到 B 上了），B 向 C 同步到时间戳为 T2（T2 > T1)，tracker 接收到这些同步进度信息时就会进行整理，将最小的那个做为 C 的同步时间戳，本例中 T1 即为 C 的同步时间戳为 T1（即所有 T 以前写的数据都已经同步到 C 上了）；同理，根据上述规则，tracker 会为 A、B 生成一个同步时间。
<h2 id="FastDFS的优缺点">FastDFS 的优缺点</h2>
优点：
（1）不管是上传下载，同步恢复，对机器造成的负载非常低，因此 FastDFS 对机器的硬件配置要求。
（2）文件上传时只要写入了源 Storage 就返回，因此上传速度很快
（3）可将不同业务的数据也到不同的组，实现业务数据隔离
（4）大规模成功案例多
缺点：
（1）由于文件在写入源 Storage 就返回，因此存在数据丢失的风险。
（2）由于不会对小文件进行整合，导致小文件较多的情况下，故障恢复时间长。
（3）由于不会对大文件进行分片存储，当大文件为热点数据时，访问压力会集中到固定的分组机器上/p>
<h2 id="FastDFS最佳部署">FastDFS 最佳部署</h2>
生产环境中，往往会将 FastDFS 与 Nginx 结合，除了通过 FastDFS API 实现文件访问外，还可通过 Http 的方式访问文件。架构示意图如下：
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https://b3logfile.com/file/2020/01/image-ce9ce58d.png?imageView2/2/interlace/1/format/jpg"></>
<h2 id="参考资料">参考资料</h2>
分布式文件系统 FastDFS 详
fastdfs 和 ceph 对比br>
FastDFS 之 Binlog 同步</a
Fastdfs 与 nginx 结合提供 web 服务
<a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.com%2Fhappyfish100%

Ffastdfs" target="_blank" rel="nofollow ugc">FastDFS github 源码</p>