# GFS：Google File System

## 1. 简介

* 组件失效被认为是常态事件

* 文件非常巨大。数GB的非常普遍

* 绝大部分文件的修改是采用文件尾部追加数据，而不是覆盖原油数据

* 应用程序和文件系统API的协同设计提高了整个系统的灵活性

## 2. 设计概述

### 2.1 设计预期

* 组件失效是常态，要求系统可以迅速地侦测、冗余并恢复失效的组件

* 系统支持大文件，但也必须同时支持小文件

* 系统的工作负载主要由两种读操作组成：大规模的流式读取（数百KB或1MB）和小规模的随机读取（几KB）

* 支持大规模顺序的数据追加，认为数据被写入一般不会改变；但也支持小规模随机写入（效率不佳）

* 系统高效且行为定义明确

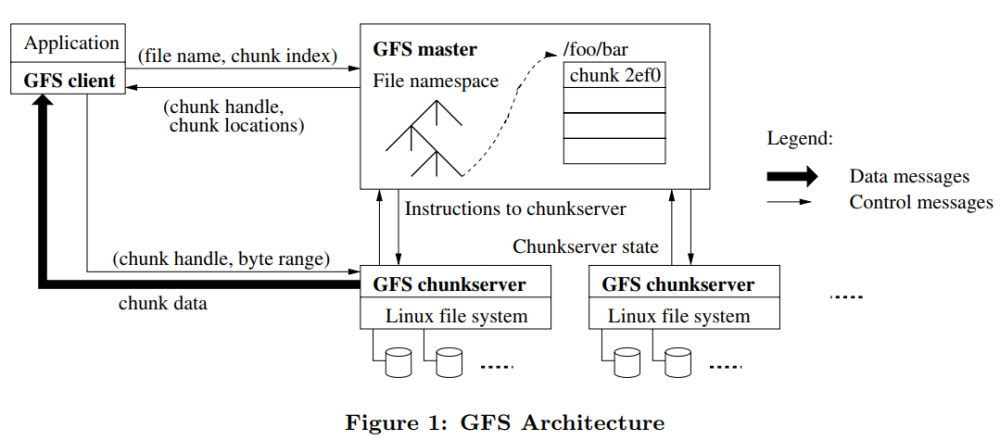
* 高速率的、大批量的处理数据，极少有程序对单一的读写操作要求响应时间

### 2.2 接口：增删改查、打开关闭，快照和记录追加

### 2.3 架构

* 只存在单一master节点（逻辑上，其实可以实现master节点复制），管理所有文件的元数据（名字空间、访问控制信息、文件和Chunk的映射信息以及当前Chunk的位置信息）；master只是作为管理者，基本不会参与数据的读写；图一，实际应用中Client会一次询问多个Chunk

* 多chunk服务器，可以被多个客户端访问。GFS文件被分割为固定大小的Chunk，每个Chunk在创建的时候有一个唯一的64位标识，为了安全，每个Chunk被复制到多个服务器上：Chunk尺寸为64MB（减少客户端和Master节点通讯的需求、可以对一个Chunk进行多次的读写操作、减少元数据的大小，可以将元数据放入内存）



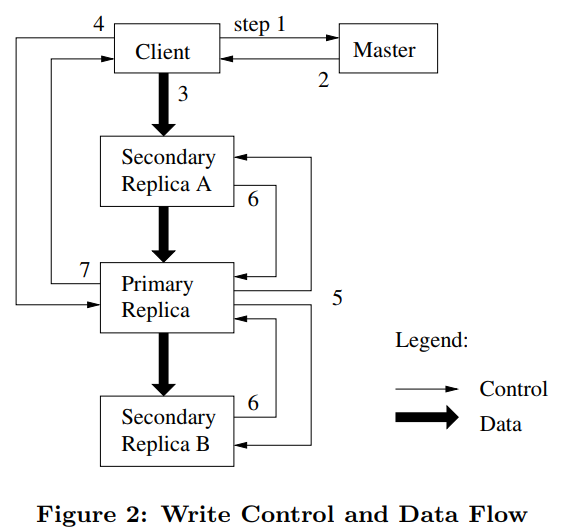
master并不是持久地保存元数据，而是在启动的时候轮询Chunk服务器，之后每隔一段时间轮询一次

### 2.4 一致性模型

文件命名空间的修改是原子性的，仅由master节点控制。文章2.7节

## 3 系统交互

租约机制用来保持多个副本间变更顺序的一致性。



* 客户机向Master节点询问哪一个Chunk服务器持有当前的租约，以及其它副本的位置。如果没有一个Chunk持有租约，Master节点就选择其中一个副本建立一个租约（这个步骤在图上没有显示）。

* Master节点将主Chunk的标识符以及其它副本（又称为secondary副本、二级副本）的位置返回给客户机。客户机缓存这些数据以便后续的操作。只有在主Chunk不可用，或者主Chunk回复信息表明它已不再持有租约的时候，客户机才需要重新跟Master节点联系。

* 客户机把数据推送到所有的副本上。客户机可以以任意的顺序推送数据。Chunk服务器接收到数据并保存在它的内部LRU缓存中，一直到数据被使用或者过期交换出去。由于数据流的网络传输负载非常高，通过分离数据流和控制流，可以基于网络拓扑情况对数据流进行规划，提高系统性能，而不用去理会哪个Chunk服务器保存了主Chunk。

* 当所有的副本都确认接收到了数据，客户机发送写请求到主Chunk服务器。这个请求标识了早前推送到所有副本的数据。主Chunk为接收到的所有操作分配连续的序列号，这些操作可能来自不同的客户机，序列号保证了操作顺序执行。它以序列号的顺序把操作应用到它自己的本地状态中 。

* 主Chunk把写请求传递到所有的二级副本。每个二级副本依照主Chunk分配的序列号以相同的顺序执行这些操作。

* 所有的二级副本回复主Chunk，它们已经完成了操作。

* 主Chunk服务器回复客户机。任何副本产生的任何错误都会返回给客户机。在出现错误的情况下，写入操作可能在主Chunk和一些二级副本执行成功。（如果操作在主Chunk上失败了，操作就不会被分配序列号，也不会被传递。）客户端的请求被确认为失败，被修改的region处于不一致的状态。我们的客户机代码通过重复执行失败的操作来处理这样的错误。在从头开始重复执行之前，客户机会先从步骤（3）到步骤（7）做几次尝试

### 3.1 快照

快照操作几乎可以瞬间完成对一个文件或者目录树做一个拷贝，并且几乎不会对正在进行的其他操作造成任何干扰。

当master收到快照请求，会首先取消作快照的文件的所有Chunk的租约。

## 4 Master节点的操作

Master节点执行所有的名称空间操作。此外，它还管理着整个系统里所有Chunk的副本：它决定Chunk的存储位置，创建新Chunk和它的副本，协调各种各样的系统活动以保证Chunk被完全复制，在所有的Chunk服务器之间的进行负载均衡，回收不再使用的存储空间

* **名称空间管理和锁**

* **副本的位置**

* **创建、重新复制、重新负载均衡**

* **垃圾回收**

## 5 容错和诊断

* **高可用性：**快速恢复（kill掉进程，关闭服务器，重新连接后重启）、Chunk复制、Master的复制

* **数据的完整性：**每个Chunk对应一个32位的Checksun，把存在内存和硬盘中（与其他用户数据分开）

* **诊断工具：**Log、RPC日志与记录

## 实验：不写了，看原文去吧