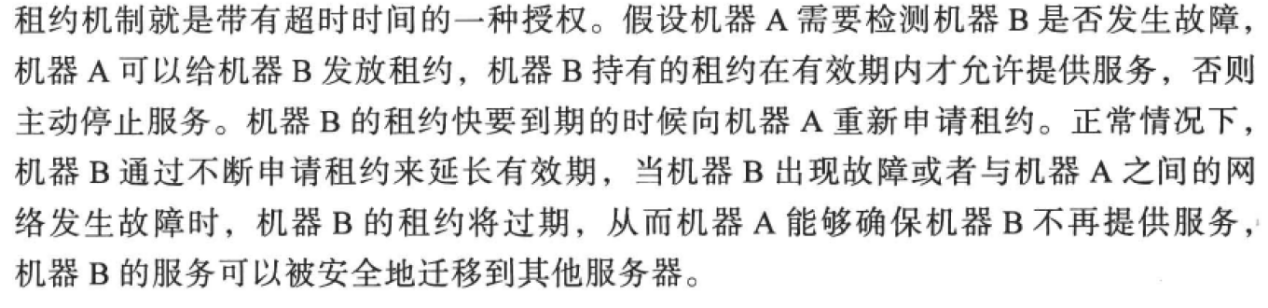
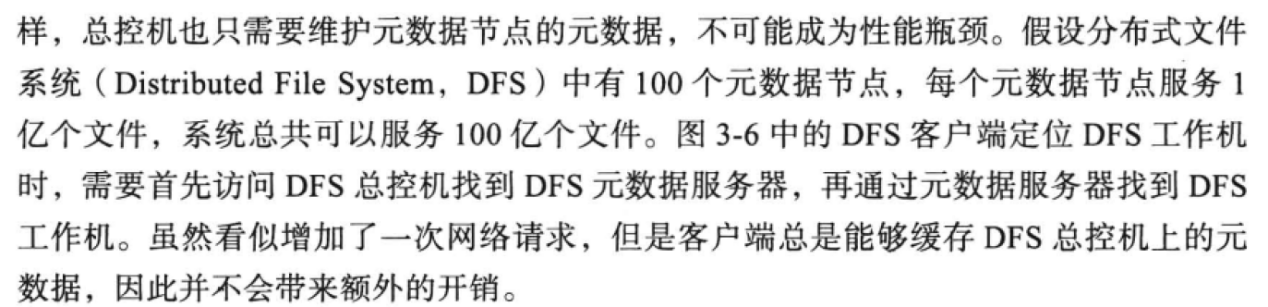
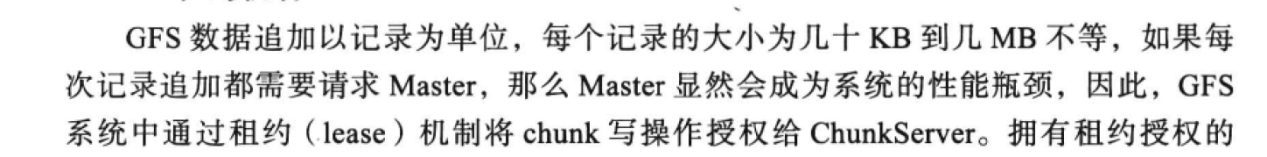
第一章 概述

1. 传统存储系统的纵向扩展，即先买小型机，不够再中型机，再不然就大型机，即增加单机的容量。
2. 分布式存储系统分为分布式文件系统，分布式键值系统，分布式表格系统，分布式数据库系统，划分的依据是存储数据的结构化程度不同，分别对应为非结构化数据，半结构化数据，更为复杂些的半结构化数据，结构化数据
3. 为了解决传统关系型数据库可扩展性的问题（很难高效地扩展到多个存储节点上），随之产生了nosql数据库
4. 单机存储系统
5. sql和nosql的区别：nosql存储的数据结构相对更弱些，sql是结构化数据，所有的数据都严格定义在几张表内，且表内的属性都是确定的。而nosql更灵活些，利用键值来存储系统，对应的值没有任何的格式要求（通过value的长度字段来标识值得信息），所以更适合海量数据和高并发的情况。由于sql为了避免数据冗余，所以每张表内存的属性不能有重复（除了主键），所以当查找某个属性时需要查找几张表，而在nosql中，通常采取存储冗余数据来达到高效的查找。
6. 列式存储，一个OLAP类型的查询需要访问几百万甚至几亿个数据行，且该查询往往只关心数个数据列，所以其他的数据列对于查询来说是没有意义的，这个可以提高大数据查询的效率（减少了数据加载到内存），同时由于同一列的数据重复比较多，所以适合压缩数据。
7. 分布式文件系统
8. 高可用性和一致性有冲突
9. 高吞吐和低延迟有冲突
10. 通过租约机制进行故障检测。租约机制就是带有超时时间的一种授权
11. 为了进行选主或者维护系统中重要的全局信息，可以维护一套通过Paxos协议实现的分布式锁服务，比如Google Chubby或者它的开源实现Apache Zookeeper
12. 若总控节点成为瓶颈，可以在总控节点和工作节点之间加一层元数据节点，每个元数据节点只维护一部分而不是整个分布式文件系统的元数据



1. 分布式文件系统
2. 租约的另外一个用途，在GFS中避免master成为性能瓶颈



1. TFS，解决小文件的存储问题，设计思路：多个逻辑图片文件共享一个物理文件。TFS内部没有维护文件目录树，使用一个64位的编号来标识一个小文件，同时由于TFS面对是一个读多写少的场景，所以放弃使用租约来减少nameserver的压力，因为每次写操作经过nameserver都不会有问题，简化了系统的设计。
2. 追加数据时的差别：GFS的数据推送采用的是流水线，减少延迟，但是设计更困难些；而TFS的数据推送没有用，旨在用最简单的方式解决用户面临的问题。
3. 分布式键值系统
4. 分布式表格系统
5. BigTable实际系统中就是存储一张表，然后分割成一张张子表，组织形式类似于操作系统内存管理中的页表的管理，BigTable分为三级，根表，元数据表，用户表，即对应为一级页表，二级页表和三级页表。
6. leveldb可以理解成单机版的BigTable