The Google File System 读后感

1. 概念

谷歌文件系统也称为GFS，与传统的分布式文件系统有着很多相同的设计目标，比如，性能、可伸缩性、可靠性以及可用性。同时，针对谷歌的其他需求，设计出了GFS，需要应对的需求比如：常态化的组件失效，巨大的文件，大部分在文件尾部追加数据的操作等等。

1. 设计概述

针对这些需求，我们设计出的系统统必须持续监控自身的状态，它 必须将组件失效作为一种常态，能够迅速地侦测、冗余并恢复失效的组件。系统存储一定数量的大文件。系统的工作负载主要由两种读操作组成：大规模的流式读取和小规模的随机读取。系统的工作负载还包括许多大规模的、顺序的、数据追加方式的写操作。

因此，GFS的架构如下：一个 GFS 集群包含一个单独的 Master 节点，多台 Chunk 服务器，并且同时被多个客户端访问。GFS 存储的文件都被分割成固定大小的 Chunk。Master 节点管理所有的文件系统元数据，GFS客户端代码以库的形式被链接到客户程序里。

1. GFS的交互
   1. lease和变更顺序

设计lease机制的目的是为了最小化 Master 节点的管理负担。lease的初始超时设置为60秒。 不过，只要 Chunk 被修改了，主 Chunk 就可以申请更长的租期，通常会得到 Master 节点的 确认并收到租约延长的时间。这些租约延长请求和批准的信息通常都是附加在 Master 节点 和 Chunk 服务器之间的心跳消息中来传递。

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。

* 1. 数据流

为了提高网络效率，我们采取了把数据流和控制流分开的措施。在控制流从客户机到 主 Chunk、然后再到所有二级副本的同时，数据以管道的方式，顺序的沿着一个精心选择的 Chunk 服务器链推送。我们的目标是充分利用每台机器的带宽，避免网络瓶颈和高延时的连 接，最小化推送所有数据的延时。

* 1. 原子的记录追加

GFS 提供了一种原子的数据追加操作–记录追加。传统方式的写入操作，客户程序会指定数据写入的偏移量。记录追加在我们的分布应用中非常频繁的使用。

1. 高可用性

当给定的时间里有些服务器不可用时，使用两条简单但有效的策略保证高可用性：快速恢复和复制。

不管 Master 服务器和 Chunk 服务器是如何关闭的，它们都被设计为可以在数秒钟内恢复它 们的状态并重新启动。每个 Chunk 都被复制到不同机架上的不同的 Chunk 服务器上。用户可以为文件命名空间的不同部分设定不同的复制级别，default为3。

1. 性能评估

在小规模测试中，当读取的客户机增加时，多个客户机同时读取一个Chunk服务器的几率也增加了，导致整体的读取效率下降。写入的速度要比理论值慢，读取速率要比写入速率高得多。

实际的应用中，由于集群存储的文件较大，它的Chunk数量也较多，产生的“死文件”也更多。。某些情况下，消费者读取的速度超过了生产者写入的速度，这就会导致没有读到任何数据的情况。集群 X 通常用于短暂的数据分析任务，而不是长时间运行的分布式应用，因此，集群 X 很少出现这种情况。写操作按数据量大小也同样呈现为双峰分布。大的写操作（超过256KB）通常是由于 Writer 使用了缓存机制导致的。

1. 经验

最大的问题是磁盘以及和 Linux 相关的问题。很多磁盘都声称它们支持某个范围内的 Linux IDE 硬盘驱动程序，但是实际应用中反映出来的情况却不是这样，它们只支持最新的 驱动。因为协议版本很接近，所以大部分磁盘都可以用，但是偶尔也会有由于协议不匹配，导致驱动和内核对于驱动器的状态判断失误。这会导致数据因为内核中的问题意外的被破坏 了。这个问题促使我们使用 Checksum 来校验数据，同时我们也修改内核来处理这些因为协议不匹配带来的问题。

另一个和 Linux 相关的问题是单个读写锁的问题，也就是说，在某一个地址空间的任意一个 线程都必须在从磁盘 page in（读锁）的时候先 hold 住，或者在 mmap()调用（写锁）的时候改写地址空间。我们发现即使我们的系统负载很轻的情况下也会有偶尔的超时，我们花费了很多的精力去查找资源的瓶颈或者硬件的问题。最后我们终于发现这个单个锁在磁盘线程交换以前映射的数据到磁盘的时候，锁住了当前的网络线程，阻止它把新数据映射到内存。