一、综合当时所处的 时代特征、应用的要求 等因素

结合MapReduce、GoogleFile System两篇经典论文，试阐述:

GoogleFile System基本设计原则和工作原理

提示，可从如下方面思考:

1. 应用对系统提出新的要求等;

2. 从 纵向扩展 到 横向扩展 的变化及原因(基于廉价硬件)等;

3. 数据与存储的分离、数据与存储的融合等;

4. GFS(GoogleFileSystem)为何与传统文件系统、传统分布式文件系统存在差异，这种差异对系统用户的影响等

二、分布式系统课程学习体会，对授课内容、方式的意见与建议

# GoogleFile System基本设计原则和工作原理

## 1. 应用对系统提出新的要求

GFS是面向大规模数据密集型应用的、可伸缩的分布式文件系统。运行在廉价的普遍硬件设备上，但依然具备容灾能力，为大量客户机提供高聚合性能服务。

除了传统分布式文件系统设计的目标，如性能、扩展性、高可靠和高可用，GFS还面临如下需求：

一、大型系统中，组件失效是常态事件。

二、应用要处理的文件很大，数量空前的多。

三、应用中很多对文件的写操作属于追加写（Append）。写完之后，通常情况只进行读操作，而且是顺序读。

四、应用程序和文件系统 API 协同设计，系统灵活性高（双方配合）。对一致性要求不需要完全符合传统文件系统要求。

## 2. 从 纵向扩展 到 横向扩展 的变化及原因(基于廉价硬件)

纵向扩展意味着将资源添加到系统中的单个节点。就像在 PC 上添加硬盘一样，在过去的几十年里，这一直是公司升级存储的常见方式。然而，随着时间的推移，这种方法暴露出越来越多的局限性。

对于存储系统而言，垂直扩展只是在现有架构上增加硬盘来增加存储容量，而不会增加 CPU 和内存来帮助整个系统处理更多信息。这意味着当存储容量增加时，存储性能往往会下降。

横向扩展在扩展容量的同时，性能随容量线性增长。扩容的每个节点都有独立的CPU、独立的内存等。基于廉价硬件的横向拓展降低复杂的基础设施成本并快速受益于更新的CPU架构和磁盘驱动器密度，而无需进行昂贵的叉车升级；更好的硬件简化了系统管理，促进了冗余并提高了正常运行时间；使系统在未来实际扩大规模变得相对容易。传统垂直扩展架构的复杂性可能会带来升级过程中业务中断的风险，而水平扩展则相对容易。

此外，在横向扩展流行之前，企业往往会购买容量远超需求的存储系统（例如RAID方案需要额外购买磁盘用于奇偶校验和热备），以确保有足够的磁盘空间可供使用和未来扩展。但是如果这种扩容没有发生，或者需求没有达到预期的水平，那么原本购买的大量磁盘空间就会被浪费掉。

采用横向扩展架构，避免了初期投资成本过高的问题。如果存储需求增长超出预期水平，可以根据需要添加新阵列，基本没有限制。

一个GFS集群包含一个单独的Master节点（主控节点）、多台Chunk（数据块）服务器。GFS同时被多个客户端访问。所有的这些机器可由普通的、廉价的Linux服务器组成，运行用户级别(user-level)的服务进程。

通常来说，构建分布式系统的目的是为了获取人们常常提到的可扩展的加速。（Scalability）。这里说的可扩展或者可扩展性指的是，如果我用一台计算机解决了一些问题，当我买了第二台计算机，我只需要一半的时间就可以解决这些问题，或者说每分钟可以解决两倍数量的问题。这是一个很强大的特性。如果你构建了一个系统，并且只要增加计算机的数量，系统就能相应提高性能或者吞吐量，这将会是一个巨大的成果，因为计算机只需要花钱就可以买到。如果不增加计算机，就需要花钱雇程序员来重构这些系统，进而使这些系统有更高的性能，更高的运行效率，或者应用一个更好的算法之类的。花钱请程序员来修补这些代码，使它们运行的更快，通常会是一个昂贵的方法。我们还是希望能够通过从十台计算机提升到一千台计算机，就能扛住一百倍的流量。这是相对廉价的方案。

## 3. 数据与存储的分离、数据与存储的融合

GFS中一个文件被顺序分割为多个Chunk，在Chunk Server集群分散保存Chunk的多个副本，来实现存储高可用。这是典型的“数据分散集群”架构，即数据存储在多个分片上，有一个角色来负责执行数据分配算法，在GFS中这个工作是由Master节点完成的。Master节点负责分配Chunk存储在哪些Chunk Server。分配算法根据各个Chunk Server的机架分布、硬盘使用率、繁忙程度等因素来执行，以达到最大化数据可靠性和可用性，最大化网络带宽利用率的目标。

Chunk Server负责存储数据，维护数据可用性，并向Master节点同步本地Chunk元数据，以方便Master节点建立文件到Chunk映射的完整视图。另外，为了避免 Master 成为系统瓶颈，GFS设计了租约机制。租约由Master负责授权，持有Chunk租约的Chunk Server成为主节点，而其他Chunk Server成为跟随节点。在租约有效期内，对该Chunk的写操作由主节点来控制。

Master节点用来管理文件和Chunk的信息，而Chunk服务器用来存储实际的数据。这是GFS设计中比较好的一面，它将这两类数据的管理问题几乎完全隔离开了，这样这两个问题可以使用独立设计来解决。Master节点知道每一个文件对应的所有的Chunk的ID，这些Chunk每个是64MB大小，它们共同构成了一个文件。如果我有一个1GB的文件，那么Master节点就知道文件的第一个Chunk存储在哪，第二个Chunk存储在哪，等等。当我想读取这个文件中的任意一个部分时，我需要向Master节点查询对应的Chunk在哪个服务器上，之后我可以直接从Chunk服务器读取对应的Chunk数据。

## 4. GFS(GoogleFileSystem)为何与传统文件系统、传统分布式文件系统存在差异，这种差异对系统用户的影响

人们设计大型分布式系统或大型存储系统出发点通常是，他们想获取巨大的性能加成，进而利用数百台计算机的资源来同时完成大量工作。因此，性能问题就成为了最初的诉求。 之后，很自然的想法就是将数据分割放到大量的服务器上，这样就可以并行的从多台服务器读取数据。我们将这种方式称之为分片（Sharding）。如果你在成百上千台服务器进行分片，你将会看见常态的故障。如果你有数千台服务器，那么总是会有一台服务器宕机，每天甚至每个小时都可能会发生错误。所以，我们需要自动化的方法而不是人工介入来修复错误。我们需要一个自动的容错系统，这就引出了容错这个话题（fault tolerance）。如果有复制，那就有了两份数据的副本。可以确定的是，如果你不小心，它们就会不一致。所以，你本来设想的是，有了两个数据副本，你可以任意使用其中一个副本来容错。但是如果你不够小心，两个数据的副本就不是完全一致，严格来说，它们就不再互为副本了。而你获取到的数据内容也将取决于你向哪个副本请求数据。这对于应用程序来说就有些麻烦了。所以，如果我们有了复制，我们就有不一致的问题（inconsistency）。通过聪明的设计，你可以避免不一致的问题，并且让数据看起来也表现的符合预期。但是为了达到这样的效果，你总是需要额外的工作，需要不同服务器之间通过网络额外的交互，而这样的交互会降低性能。所以如果你想要一致性，你的代价就是低性能。但这明显不是我们最开始所希望的。当然，这里并不是绝对的。你可以构建性能很高的系统，但是不可避免的，都会陷入到这里的循环来。现实中，如果你想要好的一致性，你就要付出相应的代价。如果你不想付出代价，那就要忍受一些不确定的行为。我们之后会在很多系统中看到这里介绍的循环。通常，人们很少会乐意为好的一致性付出相应的性能代价。

Google的目标是构建一个大型的，快速的文件系统。并且这个文件系统是全局有效的，这样各种不同的应用程序都可以从中读取数据。

GFS论文提出了一个当时非常异类的观点：存储系统具有弱一致性也是可以的。当时，学术界的观念认为，存储系统就应该有良好的行为，如果构建了一个会返回错误数据的系统，那还有什么意义？为什么不直接构建一个能返回正确数据的系统？GFS并不保证返回正确的数据，借助于这一点，GFS的目标是提供更好的性能。

如果GFS返回错误的数据，会不会影响应用程序？

讽刺的是。有谁关心网上的投票数量是否正确呢，如果你通过搜索引擎做搜索，20000个搜索结果中丢失了一条或者搜索结果排序是错误的，没有人会注意到这些。这类系统对于错误的接受能力好过类似于银行这样的系统。当然并不意味着所有的网站数据都可以是错误的。如果你通过广告向别人收费，你最好还是保证相应的数字是对的。

尽管GFS可能会返回错误的数据，但是可以在应用程序中做一些补偿。例如论文中提到，应用程序应当对数据做校验，并明确标记数据的边界，这样应用程序在GFS返回不正确数据时可以恢复。

在一些学术论文中，你或许可以看到一些容错的，多副本，自动修复的多个Master节点共同分担工作，但是GFS却宣称使用单个Master节点并能够很好的工作。

## 5. 分布式系统课程学习体会，对授课内容、方式的意见与建议

难。

缺少很多前置知识。

毕竟我连git的分布式是什么都不是很清楚。

无法联系现实，联系自己能接触到的应用需求细节。

很抽象。

通过学习mapreduce和gfs这两篇论文，我大致了解了设计分布式系统时设计者面对的需求，了解了分布式系统的一些基础功能算法。

我希望授课能够不借助视频进行。视频、论文等阅读材料都应当作为课前作业在课间完成。并且需要有明确的阅读目的，比如看完材料思考问题，实践代码等等。授课教材是必要的。虽然授课时可以不用，但是我们需要一种目录式的知识索引（可以是一系列经过整理的博客链接，可以是技术词汇目录，PPT），方便了解这方面的专业词汇，补足在理解复杂知识时欠缺的前置知识，减少阅读压力。

如果默认学生已经完成了这门课的知识索引的建立，并且能够利用浏览器补足基础了，那么更进一步，我们需要代码实战，对于算法的讲解很无聊很抽象，哪些算法是我们实际学习工作生活中能够接触的到的，哪些问题是怎么解决的，在工程具体实现上如何体现这种问题的存在，以及提倡学生自行用简单的代码尝试解决。也就是说，需要小型实验。但是这种小型实验必然是需要大量的准备的。那么，我也希望能够更加联系实际，讲述国内分布式系统使用中出现的问题。连使用都没使用过，又何谈什么设计呢？

除了从最简单的历史上最早一批的分布式系统，我更想、也想了解现代的分布式系统的实现与使用。