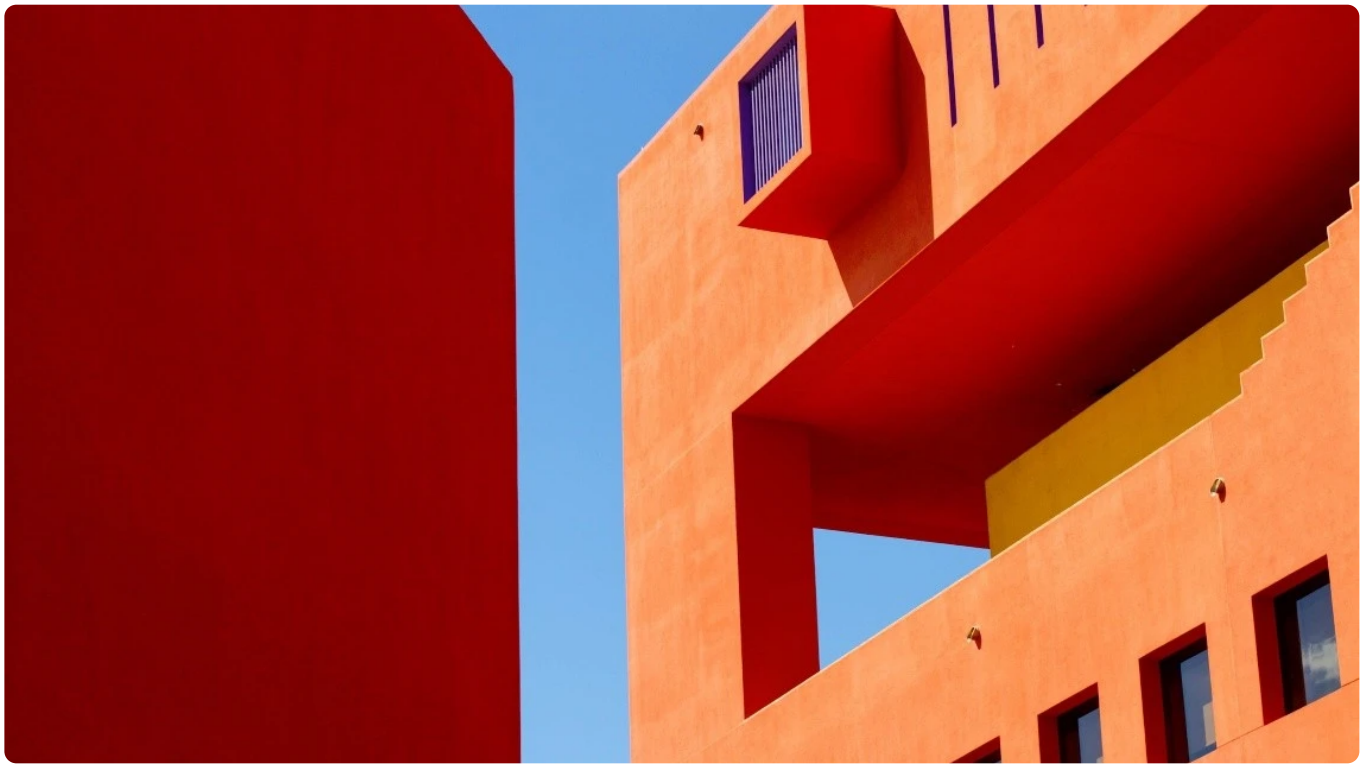
复习课 （一）| The Google File System

2021-10-27 黄金

《大数据经典论文解读》 课程介绍 >



讲述：王惠

时长 12:13 大小 11.20M

你好，我是《大数据经典论文解读》的课程编辑王惠，欢迎来到第一期复习课。

正如徐文浩老师在课程中说的那样：**学习论文并不是背诵，重要的是得总结和思考**。在看课程留言的时候，我就发现不少同学的思考和总结都很精彩，不仅认真回答了课后题，并且其中也不乏有自己独特的经验和观点输出。但是，在课程更新的过程中，也能发现有很多同学其实都掉队了，学习进度仍然停留在前几节课上。

所以接下来，为了帮你真正吃透学习过的内容，我们找到了一位优秀的同学 @在路上 作为咱们课程的学习委员，每当解读完一篇经典的大数据论文之后，学习委员都会和你一起复习这篇论文中老师提到的知识要点。希望在这个回顾复习、思考总结和知识沉淀的过程中，你可以做到查漏补缺，让自己的大数据知识体系更加完善，并能够获得跳跃式的进步和成长。

那么下面，先让我们的学委来做个自我介绍吧。

你好，我是黄金。我来自国内一家游戏公司，拥有 12 年的服务端开发经验，其中一半时间担任过游戏服务端主程的职位，参与过多款游戏研发与维护的全生命周期，也有月流水过亿的产品，是 IT 行业的一名老兵。最近三个月，我离开了游戏项目组，加入公司的大数据团队，开始接触大数据平台的开发工作。

对我而言，大数据是全新的领域，涉及非常多的理论与技术，徐老师的课程高屋建瓴又深入浅出、清晰易懂，能够帮助我快速建立大数据领域的知识体系，找到学习的重点。我非常开心能够有这样的机会，和你一起总结、思考学习过程中的知识要点。也希望接下来，我们可以在不断重复、刻意练习和不断思考的过程中，逐渐积累并沉淀大数据知识，厚积薄发，实现自己的学习目标。

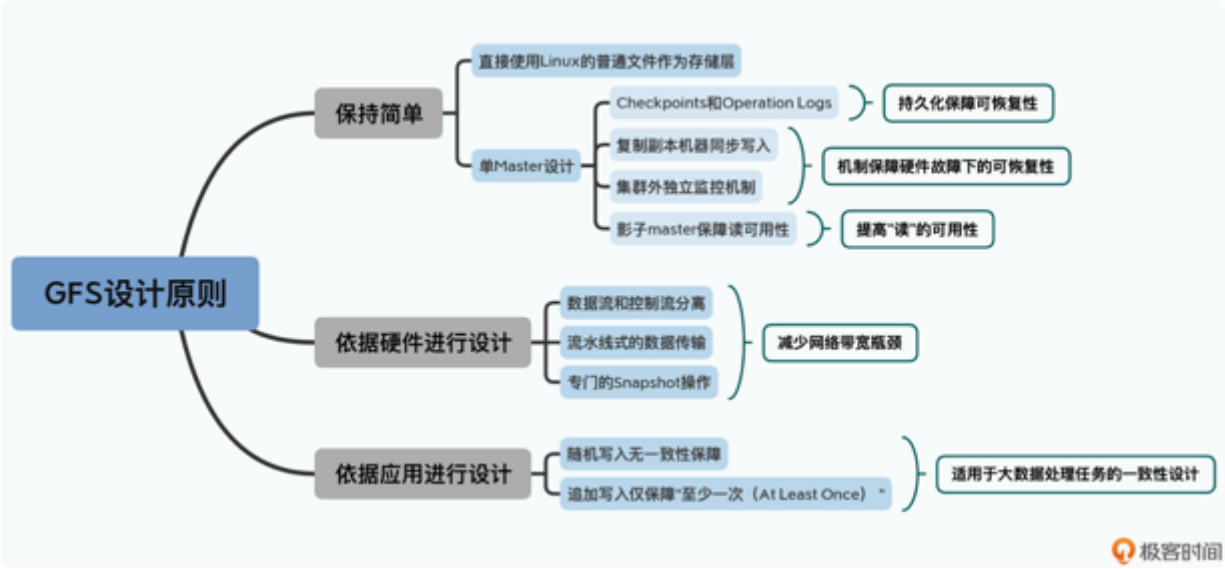
好，那么今天这期复习课，我们就从 GFS 这篇经典的大数据论文开始。

知识内容复盘

GFS 介绍

The Google File System 论文发表于 2003 年，当时工业界的分布式系统最多也就是几十台服务器的 MPI 集群。而这篇论文一下子拿出了一个运作在 1000 台服务器以上的分布式文件系统。GFS 的目标是借助大规模的集群，实现高并发、高吞吐的文件读写功能，集群规模越大，并发和吞吐能力越强。

徐老师在 03 和 04 讲分别介绍了 GFS 读取和写入的操作流程，它们是理解 GFS 的基础。



GFS 在设计上有**四个特点**，一是单 Master 架构，二是一个文件由多个 64MB 的大 Chunk 组成，三是使用内存存储 Master METADATA，四是放宽的一致性模型。

可能的性能瓶颈

随着 GFS 集群规模的不断扩大，可能会产生的性能瓶颈在哪里呢？我们很容易能想到它的 Master 节点。单 Master 的架构符合简单的设计原则，不过它的内存、硬盘、CPU、网络会不会成为集群的性能瓶颈，是我们需要考虑的。**那么，GFS 对此有没有做什么优化呢？**

通过 GFS 论文第 6 节“Measurements”，我们可以来认识下 GFS 的实际性能。在一个由 227 个 chunkserver 构成的生产集群里，存储了 73.7 万个文件，总大小为 155TB，Master METADATA 大小为 60MB。60MB 的内存非常小，所以 Master 节点的内存不会率先成为性能瓶颈，当然如果我们在 GFS 里存储了非常多的小文件的话，就另当别论了，不过这显然违背了 GFS 的设计初衷。

在 GFS 中，一个文件大概需要 100 Bytes 记录 METADATA 信息，一台 64GB 内存的 Master 服务器可以存储 6 亿多个文件，假设每个文件是 256MB，单台 Master 可以管理 15PB 的数据，这就足以满足很多的应用场景。并且由于 METADATA 信息全部存储在内存当中，所以硬盘也不会成为性能瓶颈。

我们再来看看 CPU，GFS 论文第 6.2.4 节“Master Load”提到，Master 每秒大概需要处理 200~500 个请求，这对 Master 来说毫无压力。

正如徐老师所言，**GFS 真正的性能瓶颈在于网络**。04 讲中提到，GFS 采用了控制流和数据流分离、流水线式传输文件数据和巧妙的 Snapshot 操作三种方法，减少了 Master 节点对所有操作的参与，从而减少了网络流量的传输。

结合业务的设计

我们从 05 讲的推荐文章《CAP 理论十二年回顾：“规则”变了》中可以看出，在现实的业务场景中，分区（P）并不是常态，即便在分区存在的情况下，一致性（C）和可用性（A）并非不可兼得，但要同时支持 C、A、P，就意味着增加了系统的复杂性。**如果想让系统保持简单，就需要在一致性（C）和可用性（A）中取舍**。GFS 的选择，是保证可用性（A）的同时，放宽了对一致性（C）的要求。

追加写入是 GFS 中主要的写操作，我们可以通过这个操作，来观察它的放宽的一致性模型。

追加写入只保证至少一次的语义，每一次写入都需要等待所有副本节点写入成功，如果在任意一个节点上失败，客户端都会得到写入失败的通知，然后发起重试。GFS 并不会对之前在部分节点上写入的脏数据做处理，而是直接暴露给了应用程序，让应用程序完成去重和排序的工作。

这样一来，虽然说增加了应用程序的一点复杂性，但是却让 GFS 保持了简洁，减少了 Master 和 chunkserver、chunkserver 和 chunkserver 之间的通信和协同。

高可用设计

GFS 是通过 Master、Backup Master 和 Shadow Master 实现了 Master 节点的高可用，以及通过多副本的 chunk 机制，保证了 chunkserver 节点的高可用。

在论文第 6.2.2 节“METADATA”中提到，通过 Checkpoint 和 Operation Log 恢复一个 60MB 大小的 METADATA，只需要几秒钟，加上等待 chunkserver 上报 chunk 信息的时间，大概花费 30~60 秒，Backup Master 就能够对外提供服务了。

而在此期间，GFS 是由 Shadow Master 提供读取服务的。虽然 Shadow Master 是异步复制 Master 中的数据，但是一般也只比 Master 慢了几分之一秒。

另外，论文第 6.2.5 节“Recovery Time”还提到，多个 chunkserver 挂掉后，GFS 会优先恢复存活副本数较少的 chunk。在一个实验中，测试人员关闭了 2 个 chunkserver，每个 chunkserver 包含了大概 16000 个 chunk 和 660GB 的数据，这个操作导致 266 个 chunk 只拥有一个副本，不过 GFS 在 2 分钟内就把它们恢复到了至少 2 个副本。当时的机器带宽是 100Mbps，而现在的机器普遍达到了 1000Mbps，甚至是 10000Mbps，恢复速度会更快。

我在读 GFS 论文的时候注意到一个细节，就是**当系统在执行 Snapshot 操作时，会把当前还需要写入数据的 chunk 拷贝一份，让后续的写请求将数据写入拷贝出来的 chunk 当中**。这样的话，就可以减少 Snapshot 操作对写操作的影响了，不用等到 Snapshot 操作结束后，才允许写入数据。

系统交互设计

最后，GFS 在设计系统时，其实尽可能地减少了 Master 在所有操作中的参与程度。这里在回顾和总结课程内容之余，我想再来补充下**数据变更顺序**（Data Mutation Order）和**原子性的追加写入**（Atomic Record Append）的一些细节，以及我的思考。

在追加写入时，客户端会请求 Master，获悉数据要写入哪一个 chunk，以及 chunk 所在的 primary（主数据）chunkserver。

Master 会给 primary chunkserver 分配一个 lease（租期），这个 lease 默认持续 1 分钟，在这 1 分钟时间内，会由这个 chunkserver 决定多个客户端写请求的处理优先级。在此期间，如果 Master 和这个 chunkserver 失联了，Master 不会立即分配新的 primary chunkserver，因为此时客户端可能还可以和这个 chunkserver 通信，完成写入操作。只有等到 lease 到期，Master 才会考虑重新分配 primary chunkserver。

老师在 05 讲中有提到，如果当前的 chunk 剩余的空间，不能写下要追加的记录，那么 GFS 会先把它填满空数据，然后在新的 chunk 中写入要追加的记录。我想，正是因为 Master 对写入操作参与程度低，所以 GFS 不能把一次追加写入的操作拆成两份，先把旧的 chunk 写满，再把剩下的数据写入新的 chunk，因为 GFS 并不保证在这两次写入之间，不会有其他操作插进来。

读论文给我带来的启发

好了，以上就是根据徐老师课程中，针对 GFS 论文解读和学习的总结与回顾。

实际上，在接触徐老师这门课以前，我没有读过一篇论文。我不是不知道阅读论文的重要性，但就是没有勇气迈出第一步。原因有两个，自己英文差，感觉论文难。不过学完了课前导读的 3 讲，我深深地被徐老师的知识和语言所折服，碰到这么好的老师都不去学习，还要等到什么时候呢？

我读论文也并没有什么好方法，就是**结硬寨，打呆仗**，一字一句地读，1 个小时读 1~2 页，十几页的论文要读好几天。因为我认为，学习方法固然重要，但是都比不上经年累月、持之以恒的学习，只有坚持的力量才能带来厚积薄发的效应。这就像我们从小接受义务教育，认真听讲按时完成作业，长大以后就能成为被社会所需要的人才，大多数人走的都是这条普通的道路，而不是采用最先进的方法，接受最优质的教育。

而且，在阅读论文的过程中，我发现很多论文不仅通俗易懂，也提供了一般书籍中没有的知识。之前我把自己读过的技术书籍分成了两类，一类是通俗易懂的操作指南，理论上比较浅尝辄止，另一类是艰难晦涩的大部头书，理论比较全面丰富，但功力不够就难以体会其中的奥妙。

可是经典论文则不同，它不仅详细介绍了实现细节，还会告诉你为什么这样设计、在这样设计之后系统会有什么瓶颈、它在生产环境中的实际表现如何等等。通过论文，我不仅能知其然，更能知其所以然。

这些都是我通过阅读论文所感受到的，希望我的一些思考和尝试，也能对你有所帮助。好了，如果你对 GFS 这篇论文也有不同的思考，期待你在留言区分享出来，咱们一起交流探讨，一起进步！

上一篇 18 | Spark：别忘了内存比磁盘快多少

下一篇 复习课（二）| MapReduce

精选留言 (3)

写留言

尚春
2021-10-28

我也是从前端逐渐接触数据方向，准备开始啃论文，共勉！

1

吴小智
2021-10-27

"三是内存存储在 Master METADATA 里" 这个看起来怎么这么别扭呢？应该是：METADATA 存储在 Master 内存里？？

展开

1

Gerry
2021-10-27

很赞的分享形式，多一些互动可以帮助同学们相互进步，可以对老师的讲解有更多元的理解，支持更多不同的思想交流

展开