引子

在阅读一篇论文时，我的习惯是从数据（包括其类型、格式、跨度、数据及其维度的流动变化），模型（如何实现），输入输出，文章的创新点，文章内所有的图（含义、目的），写论文的角度，这六个方面去入手。

以下是我对这三篇论文的读后感，若有不正确的地方，请老师斧正。

The Google File System 读后感

作者在论文的摘要就简明扼要的提出自己的与其他文件系统的不同点，他们的谷歌文件系统是基于对包括当前的和预期的应用程序工作负载和技术环境的观察所驱动的。而在摘要末尾，作者也简述了自己这篇文章大概内容，他们提供了设计用于支持分布式应用程序的文件系统接口扩展，讨论了设计的许多方面，并报告了来自微基准测试和现实世界使用的度量值。简明的提出自己论文的方向和创新点。

在introduction部分，作者进一步阐述论文设计的文件系统与早期文件系统的不同点以及GFS的创新点，从文件组件故障与恢复，文件传统的标准计算容量大小，文件对数据的写入和缓存，应用程序和文件系统的API这四个角度论述早期文件系统的缺陷以及论文中提出GFS的所改进后的优势。

在Design overview这一部分，作者先提出对GFS文件系统的整体设计要求，需要对不断检测自己从而在组件故障中迅速恢复，有效管理许多大型文件，更好的对数据读取方式，更有效的客户端并发运行，更高速舱处理数据，当提完这五大要求后，作者论述GFS模型的简要架构，由一个主服务器和若干数据块服务器为多个客户端服务，并描述了GFS的如何缓存数据，GFS客户端连接等设计。并详细描述GFS模型各部分的设计以及数据类型，如主服务器、内存的数据结构与设计。

在描述完GFS整体以及内部详细的架构设计后，作者从主服务器，块服务器和客户端的角度出发。详细描写了突变的这一更改元数据操作。用一张图生动的描述当客户端询问主服务器时，客户端，主服务器以及块服务器之间数据传输以及交互情况。在数据流传输中，为了避免网络瓶颈和高延迟，GFS将数据分块到最近的数据块服务器上，然后通过TCP连接上的数据实现延迟最小化。像这种许许多多细节设计无法在读后感中一一体现。

随后由分到总，讲GFS的主操作，详细描述了GFS文件系统的主要操作。当描述完GFS整体到部分的逻辑后，论文再次展示整体设计完后的GFS优点，它的高可用性，快速恢复能力，块复制等。

当描述完整体逻辑框架后，那便是对GFS模型的测试，作者分别从文件的读、写、记录、存储等方面去测试GFS。并用了大量折线图、表格去支撑测试的真实性以及可靠性。在测试的最后，就是对整体实验的复盘与讲解。

在第八部分接近尾声时，作者选择写论文的相关工作（可能我论文读的少或者类型不同，我之前看的论文related work基本都放在第二三部分）。描述在研发GFS中所参考的其他文件系统和论文，点出其他现有文件系统和其他论文提出的假设现有的不足，以及自身所作出的改良，从而再次突出自身设计GFS的创新点和改良点。

最后的结尾简明扼要的点出自身设计的GFS作用和优点，以及未来进一步改进的方向。

整篇论文逻辑清晰，语句明了，开头简明扼要点出当前文件系统的种种缺陷，进而提出自己设计的GFS文件系统的特色和改良点，接着从整体到部分各个框架逻辑的设计与实现，实现后对GFS的不同角度的测试，期间利用大量图表和数据去佐证，使其富有说服力，最后述写开发GFS期间所阅读的论文与其他文件系统的不足点，进一步突出GFS的优点，最后在简明的总结概括结尾中又提出对未来系统进一步改进的方向与期待。在这篇文章中，我可能在数据研读、细节逻辑设计上因专业水平有限没能真正读懂，但作者写次论文的逻辑顺序，叙述结构让我大有收获。

Google Big table读后感

作者在摘要处简要的介绍BigTable是一个分布式结构化数据存储系统，而现有的数据存储系统在面对数据处理的数据量还是在响应速度上都有着一定的欠缺，而针对这一问题，作者提出了Bigtable这个灵活、高性能的解决方法。

作者在introduction中介绍了Bigtable的设计目的以及在数据库中采用的实现策略，使其有一定适用性广泛,可扩展,高性能和高可用性。

第三部分作者介绍了bigtable数据模型，从行，列族，时间戳等角度论述数据模型。Bigtable 集群(cluster)通常在运行其他分布式应用程序的共享机器池(shared pool of machines)中，这是因为 Bigtable 本质是一个进程，Bigtable 进程通常与其他应用程序的进程共享同一台机器。Bigtable 依赖于集群管理系统来调度作业、管理共享机器上的资源、处理机器故障和监视机器状态之后作者由整到细讲三个主要的组件。链接到客户程序中的库，一个Master服务器和多个Tablet服务器。然后进行优化与压缩。

最后结论再次论述Big table的高性能和高可用性。并且对未来Bigtable进一步改进做出阐述。

整篇论文设计思路逻辑清晰，从分析问题出发，并且为了改善这一缺陷，自己提出一个新的解决方法，

然后从数学角度去解决问题，开头简明扼要点出当前计算方法对大型数据具有计算慢的缺陷，并且为了改善这一缺陷，自己提出一个新的解决方法，从整体框架逻辑设计，到其中内部设计的特点，如bigtable数据模型，从行，列族，时间戳等角度论述数据模型设计，在这篇文章中我学习到了许多。

MapReduce: Simplified Data Processing on Large Clusters 读后感

作者在摘要处概括性地介绍MapReduce这一模型的功能与用处，并且进一步提出他们的MapReduce在大型商品及其集群上所具有的高度可伸缩性这一创新点。

在introduction中，作者重点提出现有的计算方法对大型数据具有计算慢的缺陷，计算必须分布在数百台机器上才能在合理的时间得出结果，针对这一问题现状，作者提出的一个新的算法，可以隐藏并行化的混乱细节，并且对内部计算的key-value进行了优化，从而提供一个简单而强大的接口，使其自动化和分发大规模计算。

在第二部分，作者简述了他的模型，Map的过程主要包括初始化、Map操作执行和清理三个部分，而Reduce和Map过程基本类似。

接下来作者论述了MapReduce的实现，详细的描述了整个模型的操作流程，也描述了其Master的数据结构，而个人觉得整篇最大的创新点便是此处MapReduce的容错机制，作者分别从work故障、master失败和失效方面的处理机制进行论述。其中master失败就是指如果这个 master 任务失效了，可以从最后一个检查点（checkpoint）开始启动另一个master 进程。

在这个模型中，输入输出有不同的格式如key/value。在本文中也使用大量折线图、表格去支撑测试的真实性以及可靠性。

总的来说，MapReduce能将海量的数据浓缩成人们想要的数据。Hadoop的两大核心是HDFS和MapReduce，Hadoop的体系结构主要通过HDFS的分布式存储作为底层数据支持的。并且通过MapReduce来进行计算

整篇论文设计思路清晰，从问题出发，然后从数学角度去解决问题，开头简明扼要点出当前计算方法对大型数据具有计算慢的缺陷，并且为了改善这一缺陷，自己提出一个新的解决方法，从整体框架逻辑设计，到其中内部设计的特点，如由上百机器集群处理超大规模数据的容错设计，在实现模型的设计后，本文从集群配置、排序等方法对该模型性能进行测试，期间穿插许多图表去佐证，在这篇文章中我学习到了许多，数学的灵活应用在改善模型中起了不可置疑的作用，在未来的毕业论文中，或许我可以多从数学角度优化模型