Google三驾马车读后感

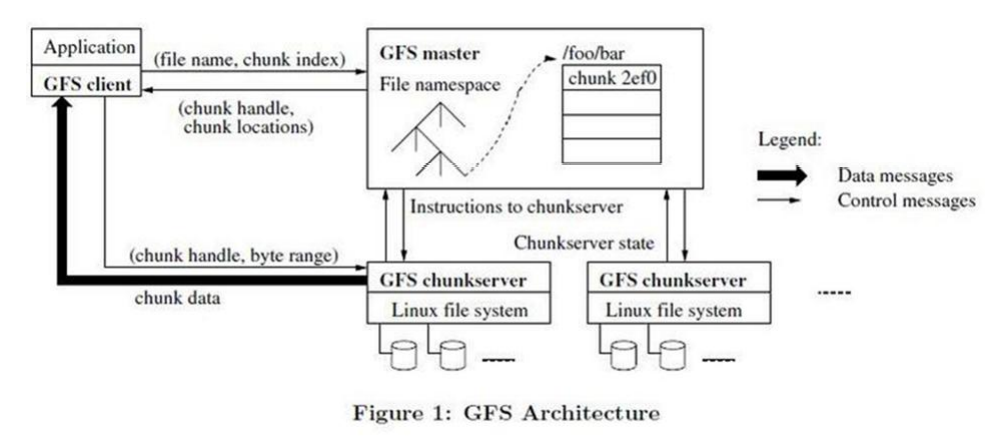
19301060黄博城

《The Google File System》主要叙述了Google设计并实现的一个面向大规模数据密集型应用的、可伸缩的分布式文件系统—GFS。该文件系统可在廉价的硬件上运行，并具有可靠的容错能力，该文件系统可为用户提供极高的计算性能，而同时具备最小的硬件投资和运营成本。

GFS的设计思路与原则：组件失效被认为是常态事件，而不是意外事件；文件非常巨大；绝大部分的文件修改时采用在文件尾部追加数据；应用系统与文件系统的API的协同设计提高了整个系统的灵活性，比如：放松了一致性模型的要求、引入原子性的记录追加操作等。

GFS的特点：高度扩展性，单个GFS集群文件系统可扩展至几千个节点，数PB的容量；高可靠性，GFS运行在普通x86服务器上，但GFS通过多副本等机制保证可靠性；满足POSIX语义，应用程序无需任何修改即可使用GFS。

GFS架构：原文中给出了GFS的架构如下图所示：



从图中可以看出一个 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。所有的这些机器通常都是普通的 Linux 机器，运行着用户级别(user-level)的服务进程。具体来说：GFS 存储的文件都被分割成固定大小的 Chunk，一般是64M，这与我们平常接触到的例如操作系统里面介绍的block size要大得多，文中介绍了使用较大的Chunk尺寸的几个重要的优点：一是减少了客户端和 Master 节点通讯的需求。其次，采用较大的 Chunk 尺寸，客户端能够对一个块进行多次操作，这样就可以通过与 Chunk 服务器保持较长时间的 TCP 连接来减少网络负载。第三，选用较大的 Chunk 尺寸减少了 Master节点需要保存的元数据的数量。这就允许我们把元数据全部放在内存中，这可以带来额外的好处。每个Chunk被赋予了唯一的 64 位的 Chunk 标识。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。无论是客户端还是 Chunk 服务器都不需要缓存文件数据。客户端缓存数据几乎没有什么用处，因为大部分程序要么以流的方式读取一个巨大文件，要么工作集太大根本无法被缓存。

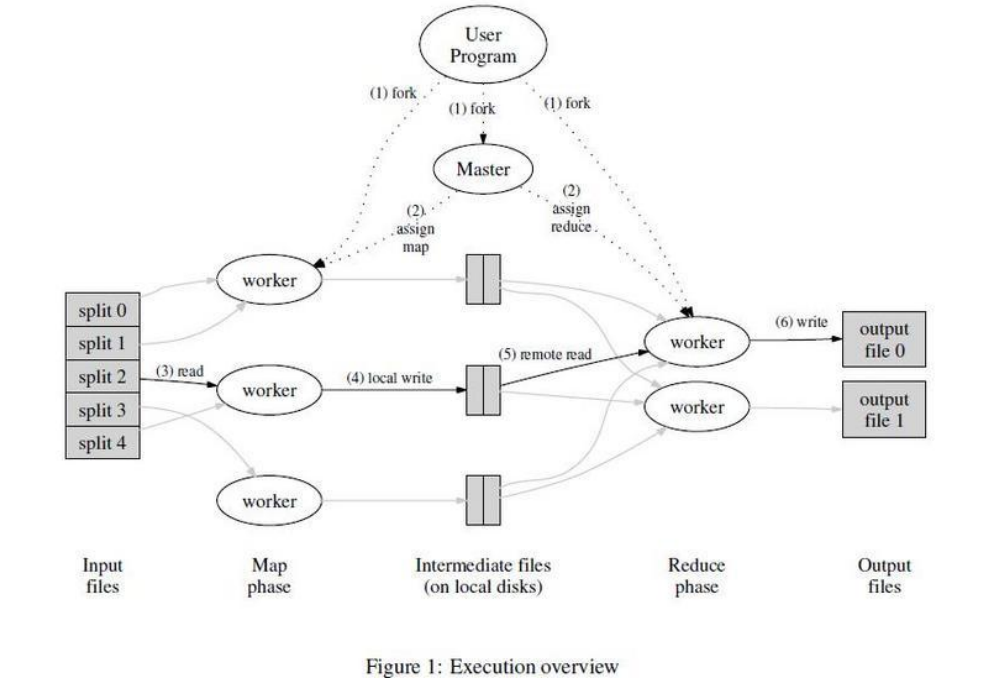
总的来说，GFS展示了一个使用普通硬件支持大规模数据处理的系统的特质。读这篇文章最大的感受在于GFS文件系统对硬件的低要求特性。它只要求普通的硬件支持就可以完成大规模的数据处理。此外该系统采用了持续监控，复制关键数据，快速和自动恢复提供灾难冗余的策略极大的提高了系统的健壮性。例如对Chunk的复制大大提高了服务器失效的容错率。同时因为磁盘数量很大的时候，损坏率是相当高的，GFS采用了Checksum在磁盘和IDE子系统级别检测数据损坏。GFS初衷是满足Google迅速增长的数据处理需求，和早期的分布式文件系统的设想都有明显的不同，但是也有许多相同的设计目标，例如在性能、可伸缩性、可靠性以及可用性等方面都有很大的参考价值。

《MapReduce: Simplified Data Processing on Large Clusters》提出了一种编程模型—MapReduce，用于大规模数据集（大于1TB）的并行运算。它的主要思想依赖于Map（映射）和Reduce（规约）。具体而言，用户首先创建一个 Map 函数处理一个基于 key/value pair 的数据集合，输出中间的基于 key/value pair 的数据集合；然后再创建一个 Reduce 函数用来合并所有的具有相同中间 key 值的中间 value 值。

个人感觉MapReduce最大的特点在于使用这个架构的程序能够在大量的普通配置的计算机上实现并行化处理，通过在大量计算机组成的集群上的调度，集群中计算机的错误处理，管理集群中计算机之间必要的通信。采用 MapReduce 架构可以使那些没有并行计算和分布式处理系统开发经验的程序员有效利用分布式系统的丰富资源。一个典型的 MapReduce计算往往由几千台机器组成、处理以 TB 计算的数据。

例如面对海量的数据要进行一些查询操作，很简单的可能会想到简单的遍历，但是这样需要的时间太长，完全无法接受。为了解决这样一个复杂的问题。MapReduce设计一个新的抽象模型，使用这个抽象模型，只要表述想要执行的简单运算即可，而不必关心并行计算、容错、数据分布、负载均衡等复杂的细节，这些问题都被封装在了一个库里面。即在输入数据的“逻辑”记录上应用 Map 操作得出一个中间 key/value pair 集合，然后在所有具有相同 key 值的 value 值上应用 Reduce 操作，从而达到合并中间的数据，得到一个想要的结果的目的。使用 MapReduce 模型，再结合用户实现的 Map 和 Reduce 函数，我们就可以非常容易的实现大规模并行化计算。

原文中给出了MapReduce的一个执行过程概述如图所示：



1. 用户程序首先调用的MapReduce 库将输入文件分成 M 个数据片度，然后用户程序在机群中创建大量的程序副本。

2. 这些程序副本中的有一个特殊的程序–master。副本中其它的程序都是 worker 程序，由 master 分配任务。有 M 个 Map 任务和 R 个 Reduce 任务将被分配，master 将一个 Map 任务或 Reduce 任务分配给一个空闲的 worker。

3. 被分配了 map 任务的 worker 程序读取相关的输入数据片段，从输入的数据片段中解析出 key/value pair，然后把 key/value pair 传递给用户自定义的 Map 函数，由 Map 函数生成并输出的中间 key/value pair，并缓存在内存中。

4. 缓存中的 key/value pair 通过分区函数分成 R 个区域，之后周期性的写入到本地磁盘上。缓存的key/value pair 在本地磁盘上的存储位置将被回传给 master，由 master 负责把这些存储位置再传送给Reduce worker。

5. 当 Reduce worker 程序接收到 master 程序发来的数据存储位置信息后，使用 RPC 从 Map worker 所在主机的磁盘上读取这些缓存数据。当 Reduce worker 读取了所有的中间数据后，通过对 key 进行排序后使得具有相同 key 值的数据聚合在一起。由于许多不同的 key 值会映射到相同的 Reduce 任务上，因此必须进行排序。如果中间数据太大无法在内存中完成排序，那么就要在外部进行排序。

6. Reduce worker 程序遍历排序后的中间数据，对于每一个唯一的中间 key 值，Reduce worker 程序将这个 key 值和它相关的中间 value 值的集合传递给用户自定义的 Reduce 函数。Reduce 函数的输出被追加到所属分区的输出文件。

7. 当所有的 Map 和 Reduce 任务都完成之后，master 唤醒用户程序。在这个时候，在用户程序里的对MapReduce 调用才返回。

总的来说，MapReduce 封装了并行处理、容错处理、数据本地化优化、负载均衡等等技术难点的细节，这使得 MapReduce库易于使用。例如当下比较流行的Hadoop框架，正是一个实现了MapReduce模式的开源的分布式并行编程框架。MapReduce的实现使得有效利用丰富的计算资源变得非常简单，也适合用来解决其他很多需要大量计算的问题。

《Bigtable: A Distributed Storage System for Structured Data》提出了一个分布式的结构化数据存储系统—Bigtable。它被设计用来处理海量数据，通常是分布在数千台普通服务器上的 PB 级的数据。Google 的很多项目使用 Bigtable 存储数据，包括 Web 索引、Google Earth、Google Finance。这些应用对Bigtable 提出的要求差异非常大，无论是在数据量上（从 URL 到网页到卫星图像）还是在响应速度上（从后端的批量处理到实时数据服务）。尽管应用需求差异很大，但是，针对 Google 的这些产品，Bigtable 还是成功的提供了一个灵活的、高性能的解决方案。由此可见Bigtable之健壮和强大和它的高可扩展性和高性能。

在很多方面，Bigtable 和数据库很类似，它使用了很多数据库的实现策略。并行数据库和内存数据库已经具备可扩展性和高性能，但是 Bigtable 提供了一个和这些系统完全不同的接口。Bigtable 不支持完整的关系数据模型；与之相反，Bigtable 为客户提供了简单的数据模型，利用这个模型，客户可以动态控制数据的分布和格式，用户也可以自己推测底层存储数据的位置相关性。数据的下标是行和列的名字，名字可以是任意的字符串。Bigtable 将存储的数据都视为字符串，但是 Bigtable 本身不去解析这些字符串，客户程序通常会在把各种结构化或者半结构化的数据串行化到这些字符串里。通过仔细选择数据的模式，客户可

以控制数据的位置相关性。最后，可以通过 BigTable 的模式参数来控制数据是存放在内存中、还是硬盘上。

Bigtable的数据模型：Bigtable是一个稀疏的、分布式的、持久化存储的多维度排序 Map。Map 的索引是行关键字、列关键字以及时间戳；Map 中的每个value都是一个未经解析的 byte 数组。Bigtable 通过行关键字的字典顺序来组织数据。表中的行关键字可以是任意的字符串列关键字组成的集合叫做“列族“，列族是访问控制的基本单位。在 Bigtable 中，表的每一个数据项都可以包含同一份数据的不同版本；不同版本的数据通过时间戳来索引。Bigtable 时间戳的类型是 64 位整型。 BigTable 构件Bigtable 是建立在其它的几个 Google 基础构件上的。BigTable 使用 Google 的分布式文件系统(GFS)存储日志文件和数据文件。BigTable 集群通常运行在一个共享的机器池中，池中的机器还会运行其它的各种各样的分布式应用程序，BigTable 的进程经常要和其它应用的进程共享机器。BigTable 依赖集群管理系统来调度任务、管理共享的机器上的资源、处理机器的故障、以及监视机器的状态。BigTable 内部存储数据的文件是 Google SSTable 格式的。BigTable 还依赖一个高可用的、序列化的分布式锁服务组件，叫做 Chubby。 总的来说，Bigtable是Google 的一个分布式的结构化数据存储系统。与传统的数据库很类似，并且使用到其它的Google的基础构件，对于Google来说，出现问题时，例如系统出现瓶颈或效率低下的情况时，能够快速的解决这些问题。

GFS、MapReduce和BigTable，即所谓的Google三驾马车。这三篇重量级的核心论文和相关的开源技术极大地普及了云计算中非常核心的分布式技术。极大地推动了大数据的发展。