Hadoop MapReduce 体系结构分析

MapReduce 是本来是在函数式编程中的一种编程范式(模型)，最早是在 Lisp 中设计并使用。其含义就是在主要使用 Map 和 Reduce 这两种操作来实现对于数据的操作。之后 Google 在 OSDI '04 中《MapReduce: Simplified Data Processing on Large Clusters》提出了一种基于 MapReduce 范数的分布式计算框架，但是这个框架是 Google 内部使用的，论文中只提到框架的设计，没有提供对于的源代码。之后 Hadoop 平台根据 MapReduce 这篇论文的思想设计实现了 Hadoop MapReduce 这个分布式计算系统，这就是这篇报告的主角。

本篇报告将从以下几个方面进行汇报。

系统面向问题

一个系统提出并设计实现总是有其需要解决的问题。MapReduce 在 2004 年 Google 在论文中提出的时候，Google 主要面对的问题是爬虫从互联网中爬取的海量数据，这些数据的如果使用当时传统的单机系统去处理是不现实。原因是高性能的计算机是昂贵的，数据量太大单机系统处理的速度太慢，并且单机系统的扩展性差。为了解决这个问题，Google 想利用数量众多普通的商用计算机，搭建一个分布式计算的环境。如果之前了解过分布式环境下的一些算法(比如 Paxos，Raft)的话，我们会知道在分布式复杂的环境下进行编程是困难的，各个单机可能会出现宕机，通信过程时序信息不确认，已经网络环境的问题。这些问题都是分布式环境下编程需要考虑的，这极大地增加了分布式环境下编程的难道。

面对这些问题，Google 希望提出一个能够使得编程能够在分布式环境自动

并行执行，且易于扩展的系统。这样来简化分布式计算任务程序的编写难度。

系统设计

在现实许多编程任务中，作者分析，大部分都可以通过 MapReduce 的编程模型进行描述。这说明 MapReduce 的编程模型的表达能力比较强。并且 MapReduce 的编程模型是非常适合用于并行化。在并行计算中，基于数据分割的并行化计算是相对简单的，如果将数据分割为一个个依赖性低的小块，彼此之前可以独立的进行计算，这样同步控制开销小，逻辑也简单。而 MapReduce 非常适用于这种数据分割的并行，Map 本来就是针对可迭代对象中元素进行映射操作的算子，是基于数据分割的操作。Reduce 操作是将可迭代对象的元素进行聚合，这种操作可能于数据分割没有太多关系。但是可以将可迭代对象进行分段，然后聚合这样就能够进行 Reduce 进行基于数据分割的并行化。

上面将 MapReduce 进行并行化就是系统设计的主要思路。在具体设计时，需要将这种思路具体化。

首先是系统中待处理的数据应该以何种形式存储和传输。首先对系统针对的待处理是网页文档和日志处理的，处理的任务有计算文档的 IDF，统计频率，PageRank 算法。对于这些问题，最兼容的方式是使用 Key-Value 的方式进行存储和管理，Key 可以表示有实际含义的值(比如词频中的词)，也可以只是单独表示一个下标。这种 Key-Value 在接下来的操作中也能带来许多好处。

然后系统总体的架构是 Master-Slaver 架构，由一个 Master 节点管理多个 Slaver 节点，让这些 Slaver 节点来执行 Map 操作和 Reduce 操作。这种架构比较明显的问题是 Master 容易成为瓶颈。如果一旦 Master 节点宕机，容易使得整个

系统崩溃，为了解决这个问题，系统配置者可以提前配置一个影子服务器，来增加 Master 节点的鲁棒性。

Map 算子原本是对于每个元素的操作，而在系统中 Map 节点是对于一个数据分片(例如大小为 64M)进行的，这样的相比对于单个元素显著提升了效率，避免频繁的读写操作。那如何产生这样的数据分片呢？这就是 MapReduce 的第一个组成部分——Shuffle。Shuffle 是将输入的文件(数据)进行分片，分发给不同的 Map 节点。这种分发需要考虑两个细节 1）数据分片大小的选取，2）数据分发节点的选择。其中分片大小的一般来说不要太小也不太大，一般为 64MB。分发节点，由于在系统实际运行的环境下，计算节点和存储节点在物理上可能是重合的，所以一般采取就近原则以减少通信的代价。

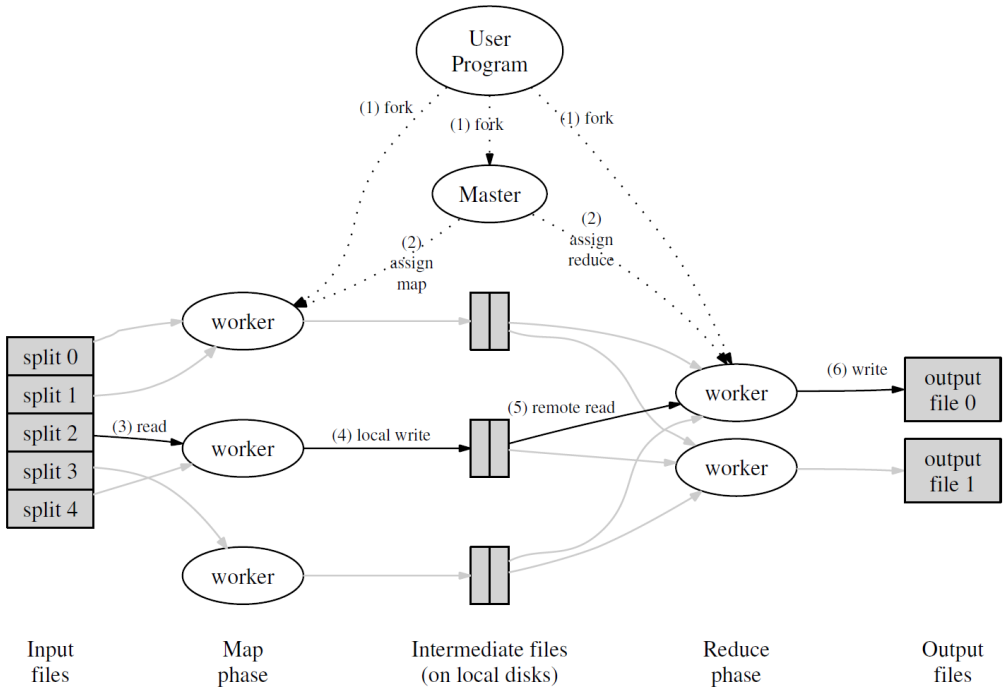
数据经过 Shuffle 阶段以后，就是对数据进行处理的 Map 阶段。Map 阶段是每个数据分片中的元素的执行预先定义好的程序。值得注意的时，该阶段部分产生的数据形式也是 Key-Value 的格式。如果在这阶段发生 Map 节点宕机怎么办。系统设计者采用了一个简单的来恢复结果——选择新的空闲节点重新执行该任务，这种方式简化了系统的设计。

Map 节点处理完成以后将结果发送给数据 Reduce，在发送之前可以选择进行一个优化步骤——Combiner，即在 Map 节点提前进行 Reduce 操作。这样可以显著减少传输的通信代价。但是执行 Combiner 有一个前提是 Reduce 操作可交换性和可结合性(例如累加操作)，否则会影响结果的正确性。

在 Map 节点计算结果的发送，怎么决定结果发送到到那个 reduce 节点呢。系统引入 Partition 阶段，来根据 Key 来决定发送给的 Reduce 的节点。这个 Partition 默认使用的是 Hash 算法，Hash(key) mod N 的方式。这个算法是可以

由用户进行定义，需要考虑的到数据分配 Reduce 节点的数据均衡和任务结果的正确性。

经过 Partition 阶段，数据已经被分发给 Reduce 节点，Reduce 任务是对同一个 Key 进行 Reduce 操作。而 Reduce 节点收到的 Key-Value 键值对是无序的，所以设计者引入了 Sort 阶段，将接受的 Key-Value 按照 Key 进行排序，排序方式采用的是外部排序的方式。在排序之后便是执行预先定义的 Reduce 程序。下面是整个 MapReduce 的结构流程图。



优点与不足

整个系统是一个设计简洁优雅的系统，在实际的表现中有良好的横向扩展性，满足了当时的分布式计算任务需求。但是相对于之后提出的 Spark 系统，MapReduce 这种编程模式的表现能力还是相对有限。而且在 MapReduce 运行的时候需要频繁的 IO 操作，这减缓了系统的运行速度。