**读后感**

# MapReduce

MapReduce是一个编程模型，也是一个处理和生成超大数据集的算法模型的相关实现，可以有效的解决很多程序员都会遇到的问题，即：使用大量的、专用的计算方法处理海量的原始数据；计算处理各种类型的衍生数据，由于输入的数据量巨大，只有将这些计算分布在成百上千的主机上才可以在能够接受的时间内完成运算。

而MapReduce模型可以很好的处理并行计算、分发数据并处理错误，可以将原本需要大量的代码处理而使得运算变得难以处理的困难有效避开。MapReduce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，同时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，实现运行在规模可以灵活调整的由普通机器组成的集群上。

这篇论文重点强调了并行式计算还有容错性，并行式计算最初是为了满足需要大量计算的领域，比如天气预报，石油勘探以及人工智能研究等，现在它的研究内容不断拓宽，同时也重视新的计算模式比如神经计算，量子计算等。就如同数学之美告诉我们的那样，有时候毫无关联的两件事情可以用一个算法解决，但是怎么实现用什么方法实现，这就必须要依靠我们自己，有些大师学者终其一生都在寻找最简便精巧的算法，不得不说能够直接学到他们的思想和研究成果是一件很幸运的事情。

MapReduce模型系统在运行时只关心：如何分割输入数据，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。其编程模型的原理具体是：

1.利用一个输入key/value pair集合来产生一个输出的key/value pair集合。MapReduce库的用户用两个函数表达这个计算：Map和Reduce。

2.用户自定义的Map函数接受一个输入的key/value pair值，然后产生一个中间 key/value pair值的集合。MapReduce库把所有具有相同中间key值I的中间value值集合在一起后传递给reduce函数。

3.用户自定义的Reduce函数接受一个中间key的值I和相关的一个value值的集合。Reduce函数合并这些value值，形成一个较小的value值的集合。一般的，每次Reduce函数调用只产生0或1个输出value值。通常我们通过一个迭代器把中间value值提供给Reduce函数，这样我们就可以处理无法全部放入内存中的大量的value值的集合。

MapReduce模型根据具体环境的不同进行选择，可以有多种不同的实现方式。例如，一种实现方式适用于小型的共享内存方式的机器，另外一种实现方式则适用于大型NUMA架构的多处理器的主机，而有的实现方式更适合大型的网络连接集群。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，使得MapReduce库易于使用。即便对于完全没有并行或者分布式系统开发经验的程序员而言；其次，大量不同类型的问题都可以通过MapReduce简单的解决。

对于现在的我来说，这篇论文还有些晦涩难懂，我还需要储备更多的基础知识，继续努力学习专业知识。有人说“学大数据首先要掌握的就是MapReduce算法”，通过这篇论文我对这句话有了更深刻的理解。看完这篇论文后，我觉得MapReduce模型作为一种高效的分布式计算模型，面对现阶段互联网日益增加的大规模数据的计算任务，MapReduce模型将会发挥更重要的作用，今后若需要用到这些知识的时候我还会回头继续着重阅读这篇论文。

# Bigtable

Bigtable是一个分布式的结构化数据存储系统，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的很多项目都使用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。

通过简要阅读这篇论文，我了解到Bigtable是一个稀疏的、分布式的、持久化存储的多维排序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中的行关键字可以是任意的字符串，并且每行的读写操作都是原子的；Bigtable中的行关键字是按照字典顺序排序存储的，表中的行都可以进行动态分区，每个分区叫tablet，tablet是数据分布和负载均衡的最小单位。由于行键是按照字典序存储的，所以查询时以行关键字作为条件查询速度毫秒级。

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内部存储数据的问文件时Google SSTable格式的。SSTable是一个持久化的、排序的、不可更改的Map<key, value>结构，其值都是任意的byte串，因此使用key查询速度很快。Big Table还依赖一个高可用的、序列化的分布式锁服务组件——Chubby。

Bigtable包括了三个主要的组件：链接到客户程序中的库、一个Master服务器和多个Tablet服务器。针对系统工作负载的变化情况，Bigtable可以动态的向集群中添加（或者删除）Tablet服务器。每个Tablet服务器都管理一个Tablet的集合，每个Tablet的服务器负责处理它所加载的Tablet的读写操作，以及在Tablets过大时，对其进行分割。客户端读取的数据都不经过Master服务器；客户程序直接和Tablet服务器通信进行读写操作。在任何一个时刻，一个Tablet只能分配给一个Tablet服务器。Master服务器记录了当前有那些活跃的Tablet服务器、那些Tablet分配给了那些Tablet服务器、那些Tablet还没有被分配。

整个Bigtable设计符合大部分大数据程序的需求，打破了关系型数据库的结构化存储，能够部署在成千上万台服务器上，可以存储PB级数据，对整个互联网行业的快速发展提供了坚实的理论基础与成功案例。

# File System

为了满足Google迅速增长的数据处理需求，Google文件系统Google File System – GFS应运而生, GFS最开始的设计思路是：

1.组件失效被认为是常态事件，而不是意外事件；

2.以通常的标准衡量，我们的文件非常巨大；

3.绝大部分文件的修改是采用在文件尾部追加数据，而不是覆盖原有数据的方式；

4.应用程序和文件系统 API 的协同设计提高了整个系统的灵活性。

GFS与传统的分布式文件系统有着很多相同的设计目标，但是，基于应用的负载情况和技术环境的观察的影响，GFS和早期文件系统的假设都有明显的不同。GFS重新审视了传统文件系统在设计上的折衷选择，衍生出了完全不同的设计思路。

与传统文件相比，它认为组件失效是很平常的事件，因为GFS包括几千台的廉价设备组装的存储机器，被很多的客服机访问，GFS的组件的质量不一样、数量有多，所有的机器在任何时间都有可能出现故障，但是GFS拥有持续监测、错误监测、灾难冗余以及可以自动恢复的机制。传统文件是覆盖原有数据方式来修改，而GFS大多文件修改是在文件尾部追加。

通过阅读这篇论文，我了解到Google文件系统构建在廉价服务器之上。它将服务器故障视为正常现象，通过软件的方式自动容错，在保证系统可靠性和可用性的同时，大大降低系统的成本。GFS 成功的实现了我们对存储的需求，无论是作为研究和开发的存储平台，还是作为生产系统的数据处理平台，都得到了广泛的应用。而GFS是为Google应用程序本身而设计的，Google已经部署了许多GFS集群，有的集群拥有超过1000个存储节点，超过300T的硬盘空间，被不同机器上的数百个客户端连续不断地频繁访问着。