基于Hadoop平台的高阶矩阵相乘MapReduce算法研究

向林泓 陈芋文 张昱琳
(重庆绿色智能技术研究院 重庆 404100)

摘 要 目前,针对基于单一节点的中高阶矩阵相乘存在着计算瓶颈,甚至因内存溢出导致计算机崩溃等问题,结合利用云计算分布式处理和虚拟化技术的优势,提出一种基于Hadoop平台的高阶矩阵相乘的MapReduce算法。实验结果表明:该算法能够有效地解决高阶矩阵相乘中存在的计算瓶颈问题,同时提高了计算效率。

关键词 MapReduce,高阶矩阵相乘,云计算,Hadoop平台
中图法分类号 TP311.1 **文献标识码** A

High Order Matrix Multiplication by MapReduce Algorithm Based on Hadoop Platform

XIANG Lin-hong CHEN Yu-wen ZHANG Yu-lin
(Chongqing Institute of Green and Intelligent Technology, Chongqing 404100, China)

Abstract Currently, high-order matrix multiplication using single-node computer causes computational bottlenecks and even leads the computer to crash for memory overflow. Taking advantage of distributed processing and virtualization technology from cloud computing, this paper presented a new method of high-order matrix multiplication by MapReduce based on Hadoop platform. The experimental results show that this algorithm can solve the computational bottlenecks with respect to high order matrix multiplication matrix and improve computational efficiency accordingly.

Keywords MapReduce, High-order matrix multiplication, Cloud computing, Hadoop platform

1 引言

Hadoop[1]起源于Nutch[2]项目,现在是Apache基金会的一个开源项目。它是Google公司的分布式文件系统GFS(Google File System)[3]和MapReduce计算模型[4,5]的开源实现。目前,全世界大约有150家各种规模公司和企业在使用Hadoop,包括:Facebook、Amazon、百度、淘宝等知名品牌公司。另外卡内基美隆、康奈尔大学、马里兰大学和帕洛阿尔托研究中心也在使用Hadoop来做地震模拟、自然语言处理以及互联网信息等研究工作[6-8]。

Hadoop的核心是MapReduce计算框架,它是一种处理海量数据的并行编程模型和计算框架,用于并行计算大规模数据集。它最早由Google公司提出,并运行在Google的GFS上,为服务于全球亿万用户的搜索引擎提供后台的网页索引处理,同时也适用于Google内部数以千计的应用程序和数据处理[9]。

MapReduce计算框架的核心思想是把对大规模数据集的操作,分发到一个主节点管理下的各分节点共同完成,然后通过整合各个分节点产生的中间结果,得到最终结果。图1显示了MapReduce计算框架的计算流程。从图1可以看出整个处理过程被MapReduce计算框架高度地抽象为两个函数:Map和Reduce。Map函数主要负责将任务分解成为多个并行执行的任务,Reduce函数主要负责将这些并行执行的任务所产生的结果汇总处理。至于在并行编程中会遇到的其他种种复杂问题,如分布式存储、工作调度、负载均衡、网络通信等都由MapReduce计算框架负责处理,使编写和运行用于处理海量数据的应用程序更加容易。

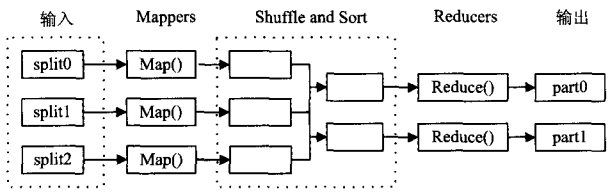

图1 MapReduce计算框架执行流程

矩阵相乘是数值计算中最重要的操作之一,它在信号处理、编码理论、密码学、数字图像处理、线性预测和计算机时序分析等领域都有着广泛的应用。目前,用于矩阵相乘的算法主要有经典算法[10]、斯特拉森算法、循环分块算法、Strassen算法等。本文提出基于Hadoop平台的高阶矩阵相乘的MapReduce算法,它以传统的经典算法为基础,被改进为MapReduce并行处理算法。

2 高阶矩阵的结构化存储

在单一计算机节点中进行矩阵相乘运算时,生成的矩阵以及所有中间结果均以二维数组的形式保存在内存中。然而对于一般的存储空间为3G~4G的计算机来说,在生成和处

本文受农情气象专家决策系统技术研发(cstc2012gg-yyjs0606),重庆市科技攻关重大项目基于国产CPU平台的"办公云"关键技术研究与应用示范(cstc2011ggC40010)资助。

向林泓(1986—),男,硕士,助理研究员,主要研究方向为云计算、自然语言处理,E-mail:xlh@cigit.ac.cn。

理上千阶矩阵相乘的运算时，往往会造成内存的溢出、计算机崩溃等情况，导致根本不能进行高阶矩阵相乘的运算。

在 Hadoop 平台的 MapReduce 计算框架下，大多数数据都是以文件的形式保存在 Hadoop 的分布式文件系统 HDFS 中。为了提高基于 MapReduce 的数据处理的可扩展性，Hadoop 提供了一组更高层次的容器将结构化对象转化为字节流，以便在网络上传输或写入磁盘永久存储。于是利用 Hadoop 提供的 SequenceFile 类可以非常方便地进行二进制 Key/Value 键值对的永久存储，并且只要存储在 SequenceFile 中的 Key/Value 键值对实现了序列化或者 Writable 接口，就可以进行二进制结构化存储[11]。

那么对于一个 M * N 数组来说，数组中的每一个元素都有其所在的数组中的行列值以及数据值。数组元素所在矩阵中的位置值是一个二元变量，可以使用自定义的 IntegerPair 类来进行存储。IntegerPair 类是一个实现了 Writable 接口的自定义类，用于存储两个整型变量，对应于 SequenceFile 中的 Key 键。数组元素的值使用 Hadoop 自带的 IntWritable 类，对应于 SequenceFile 中的 Value 键。于是一个二维数组被重新定义为结构化存储的顺序文件。在转换过程中，我们可以按照行或者列遍历进行结构化存储。图 2 显示了 M * N 的二维数组的结构化存储过程。

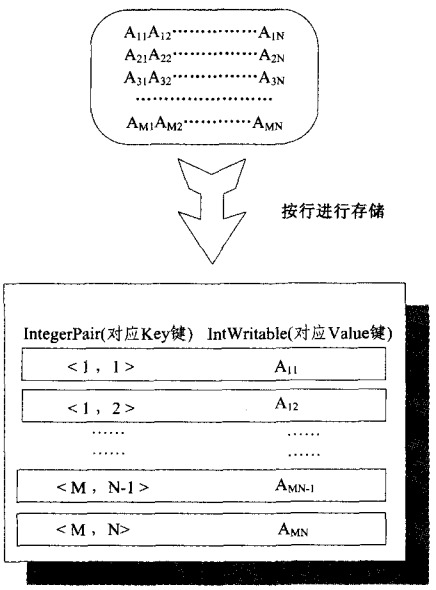

图 2 矩阵按行遍历结构化存储

3 高阶矩阵相乘 MapReduce 算法的分析和设计

在 MapReduce 计算框架中，大致可以分为两个阶段：Map 阶段和 Reduce 阶段。

在 Map 阶段，将任务的输入数据分割成为固定大小的片段 Splits，由 InputFormat 负责分节工作。MapReduce 计算框架会为每一个 Split 创建一个 Map 任务用于执行用户自定义的 map 函数，并将每个 Split 进一步分解成一批 Key/Value 键值对作为 map 函数的输入参数。接着 MapReduce 计算框架会将 map 函数产生的中间结果根据 Key 值进行排序，并将 Key 值相同的 Value 值放在一起形成一个新列表{Key，List(Value)}传递给 Reduce 任务。

在 Reduce 阶段，MapReduce 计算框架会把不同 Mapper 接收到的数据整合在一起进行排序，然后调用用户自定义的 reduce 函数进行处理[12]。

在矩阵相乘中，假设：$C_{MK}=A_{MN} * B_{NK}$

那么，$C_{ij}=A_{i1} * B_{1j}+A_{i2} * B_{2j}+A_{i3} * B_{3j}+\cdots+A_{in} * B_{nj}$，其中 $1\leqslant i\leqslant M, 1\leqslant j\leqslant K$。

从计算矩阵元素 C_{ij} 的过程可以看出，矩阵 C_{MK} 中的每一个矩阵元素是相互独立的，元素之间并没有依耐性，并且在计算元素 C_{ij} 的过程中，各个元素乘积之间也是相互独立的，只需要将各个乘积的结果相加即可。

于是在构建 MapReduce 算法时，将乘积运算放在 Map 阶段执行，而和运算放在 Reduce 阶段执行。

算法的步骤如下：

步骤 1 矩阵的结构化存储。针对 Hadoop 平台 MapReduce 计算框架，将矩阵 A 按行、矩阵 B 按列结构化存储到 Hadoop 分布式文件系统 HDFS。

步骤 2 结构化提取数据。同时遍历矩阵 A 和矩阵 B，提取矩阵元素并构造成为 Map 函数所需要的 Key/Value 键值对传递给 Map 函数。

步骤 3 在 Map 函数中，对得到的元素进行乘积操作。

步骤 4 对 Map 产生的中间结果进行排序合并操作，并将 Key 值相同的 Value 放在一起形成一个新列表〈Key，List(Value)〉传递给 Reduce 函数。

步骤 5 在 Reduce 函数中，遍历〈Key，List(Value)〉得到的所有 Value 元素并进行和运算得到最终结果。

4 高阶矩阵相乘 MapReduce 算法的实现

基于以上对 MapReduce 算法的描述，下面介绍该算法在 Hadoop 平台的 MapReduce 计算框架的具体执行流程：

(1)将测试样本矩阵 A_{MN} 和矩阵 B_{NK} 按照{〈i,j〉，$value$}的格式结构化存储在 Hadoop 分布式文件系统 HDFS，由 SequenceFile 类进行存储工作。其中 i 代表矩阵元素在该矩阵中的行值，j 代表矩阵元素在该矩阵中的列值。

(2)将结构化存储文件矩阵 A_{MN} 按行划分成 M 份，矩阵 B_{NK} 按列划分成 K 份，然后将矩阵 A_{MN} 的每行与矩阵 B_{NK} 的每列组成一个 InputSplit，该工作由自定义的 MatrixSequenceInputFormat 类负责。每一个 InputSplit 对应着生成结果矩阵 C_{MK} 所需要的矩阵元素值，然后将每一个 InputSplit 按照格式{〈i,j〉，〈A_{i1}，B_{1j}〉}、{〈i,j〉，〈A_{in}，B_{nj}〉}发送给 Map 函数。其中 i 代表最终在结果矩阵 C_{MK} 中的行值，j 代表在结果矩阵 C_{MK} 中的列值。

(3)Map 函数将得到一系列的数据，如{〈i,j〉，〈A_{i1}，B_{1j}〉}、{〈i,j〉，〈A_{in}，B_{nj}〉}等，Map 函数将提取出的元素 A_{i1} 和元素 B_{1j} 进行乘积操作：$A_{i1} * B_{1j}$，这一步对应着经典算法中的乘积操作。接着 Map 函数将生成{〈i,j〉，〈$A_{i1} * B_{1j}$〉}，{〈i,j〉，〈$A_{in} * B_{nj}$〉}，{〈$i+1,j$〉，〈$A_{i+11} * B_{1j}$〉}等一些中间结果数据。

(4)MapReduce 计算框架中将 Map 生成的中间结果进行排序，并将 Key 值相同的 Value 放在一起形成列表发送到同一个 Reduce 函数。于是 MapReduce 计算框架就会将 Key 值为〈i,j〉的放在一起形成列表发送给一个 Reduce 函数，Key 值为〈$i+1,j$〉的放在一起形成列表发送给另一个 Reduce 函数。

(5)在 Reduce 阶段，只需要遍历列表里面的值并进行相加操作即可，即 $A_{i1} * B_{1j}+A_{i2} * B_{2j}+A_{i3} * B_{3j}+\cdots+A_{in} * B_{nj}$。

图 3 展示了整个 MapReduce 的计算执行过程：

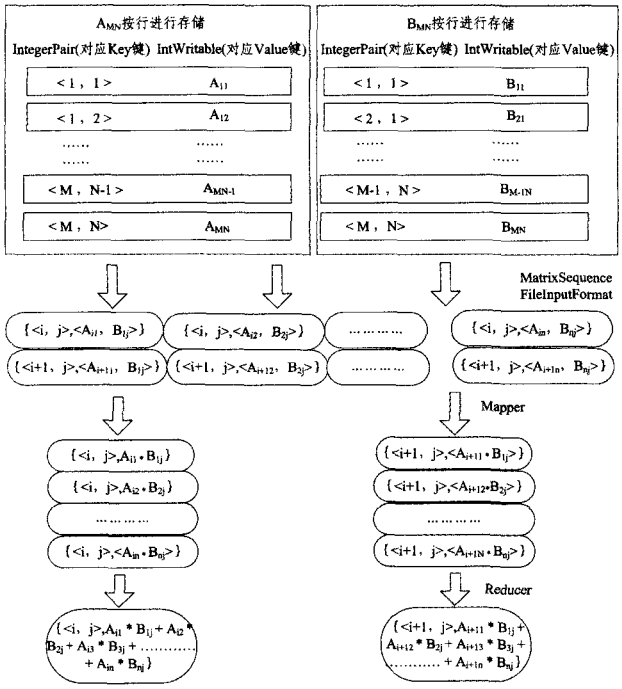

图 3 MapReduce 计算执行过程

5 实验结果及分析

5.1 实验环境及参数配置

实验运行在 10 台计算机组成的集群上。其中，一台作为主节点 Jobtracker，另外 9 台作为计算节点和分布式数据存储节点 Tasktracker。每 Tasktracker 能够同时运行的 Map 数量为 8，所以集群中能够同时运行的 Map 数量为 72。10 台服务器均安装有 Hadoop 和 JDK，实验代码由基于 Eclipse 的 Hadoop 应用开发环境开发产生。表 1 显示 MapReduce 计算框架配置。

表 1 MapReduce 计算框架配置

Cluster Summary(Heap Size is 35.25 MB/1 GB)		
Nodes	Map Task Capacity	Reduce Task Capacity
9	72	72

为了验证基于 Hadoop 的高阶矩阵相乘的 MapReduce 算法的有效性和高效性，做了以下两个实验。

5.2 实验结果及分析

实验 1 在传统的单节点计算机中利用 Matlab 进行高阶矩阵相乘。图 4 显示了实验结果。发现利用 Matlab 在进行计算的时候，运算时间会随着矩阵阶数的上升先行上升，但是运行到 5000 阶级别时，Matlab 就会因为内存溢出的原因无法正常进行计算。

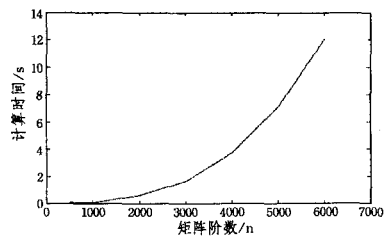

图 4 单节点实验结果

实验 2 在基于 Hadoop 平台的 MapReduce 计算框架下进行高阶矩阵相乘。时间的计算从执行第一个 Map 函数开始，而不是提交 MapReduce 任务开始。图 5 显示了实验结果，从中发现 MapReduce 计算 4000 阶以下的高阶矩阵相对 Matlab 并没有较大的优势，但是在计算 5000 阶以上的高阶矩阵时优势就会相当突出。并且因为 MapReduce 的分布式特性，我们可以看到计算时间并没有随着矩阵阶数的增长呈现线性增长，而是相对较平滑地缓慢增长。

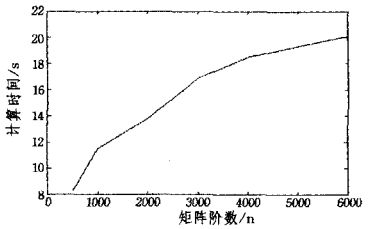

图 5 Hadoop 平台实验结果

实验结果表明，该方法有效地解决了高阶矩阵相乘的问题，并且通过云计算技术将原先由一个节点完成的计算工作合理地分配到了其他节点，使得网络资源得到了充分利用，提高了高阶矩阵相乘的效率。

结束语 文中提出的 MapReduce 算法在 Map 函数的数量控制以及时空消耗上都有很大的改进，但还存在一些不足：一方面，实验条件受限。由于个人所配置的 Hadoop 平台计算节点较少，不能拥有上千的节点，没有能够体现出云计算带来的强大魅力；另一方面，文中所用到的 MapReduce 算法是以经典算法为基础而改进的，没能用其他算法体现出矩阵相乘的优势。但该算法还是解决了单节点高阶矩阵相乘困难甚至不能相乘的问题，充分利用了网络计算资源，并且能够面向商业，提供更高的价值。

参 考 文 献

[1] Hadoop W-K [EB/OL]. http://en.wikipedia.org/wiki/Hadoop,2012-02-21

[2] Nutch W-K [EB/OL]. http://ca.wikipedia.org/wiki/Nutch, 2012-02-21

[3] Ghemawats, Gobioffh, Leungst. The google file system. [EB/OL]. Http://labs.google.com.hk/papers/gfs.html,2012-02-21

[4] Jean D, Ghemawats. Map/Reduce: simplified data processing on large clusters [EB/OL]. http://userpages.uni-koblenz.de/~laemmel/MapReduce/paper.pdf,2012-02-21

[5] Map/Reduce [EB/OL]. http://en.wikipedia.org/wiki/MapReduce,2012-02-21

[6] Applications powered by Hadoop [EB/OL]. Http://wiki.apache.org.hadoop/PoweredBy,2012-02-21

[7] Schlossers, Lin J. Hadoop Summit 2008[R/OL]. Http://developer.yahoo.com/events/hadoopsummmit2010/agenda.html, 2012-02-21

[8] Zaharia M, Konwinski A, Anthony D. Improving mapreduce performance in heterogeneous environments [C] // 8th USENIX Symposium On Operating Systems Design and Implementation. Washingtom, DC: IEEE, 2008, 1: 29-42

[9] 刘鹏，黄宜华，陈卫卫. 实战 Hadoop——开起通向云计算的捷径 [M]. 北京：电子工业出版社，2011

[10] 矩阵乘法 Wi-ki[EB/OL]. http://zh.wikipedia.org/wiki/矩阵乘法，2012-02-21

[11] White T, Cutting D. Hadoop 权威指南 [M]. 北京：清华大学出版社，2011

[12] 刘鹏. 云计算 [M]. 北京：电子工业出版社，2011