火龙果•整理
uml.org.cn

文章编号: 1006- 4729(2011)01- 0070- 05

基于 Hadoop架构的分布式计算和存储技术及其应用

田秀霞, 周耀君, 毕忠勤, 彭 源
(上海电力学院 计算机与信息工程学院, 上海 200090)

摘 要: 介绍了 Hadoop架构的主要构成,通过一个实例详细阐述了 Hadoop架构的 MapReduce实现机制;开发了一个基于 Hadoop架构职工工资统计应用实例,并根据该实例分析了其在单节点模式、伪分布模式和完全分布模式应用中的运行效率.

关键词: Hadoop架构; MapReduce机制; 分布式文件系统
中图分类号: TP333; TP316.4 文献标识码: A

The Technology and Application of Distributed Computing and Storage Based on Hadoop Architecture

TIAN Xia-xia, ZHOU Yao-jun, BI Zhong-qin, PENG Yuan
(School of Computer and Information Engineering, Shanghai University of Electric Power, Shanghai 200090, China)

Abstract The key components of Hadoop are introduced first; then the MapReduce implementation mechanism is analyzed. What is more important, an application for statistics of employee salary is developed and the efficiency comparison is given in the three different applications, namely, the single node model, pseudo-distribution model and the full distribution model.

Key words Hadoop architecture; MapReduce mechanism; distributed file system

在硬盘存储容量快速增加的同时,访问速度,即数据从硬盘读取的速度未能快速提高. 1990年,一个普通的硬盘驱动器可以存储 1 370 MB的数据并拥有 4.4 MB/s的传输速度,只需 5 min就可以读取整个磁盘的数据. 20年后的今天,海量数据的出现使得使用 1 TB存储容量的磁盘驱动器已很正常,由于数据传输速度在 100 MB/s左右,需要花 2.5 h以上才能读取整个驱动器的数据[1,2]. 如果可以一次从多个磁盘上读取数据,那么可以大大提高数据访问效率. 若拥有 100个磁盘,每个磁盘存储 1% 的数据,让它们并行运行,那么不到 2 min就可以读完存储的所有数据. Hadoop架构的引入使建立大型商业集群、解决超大数据量处理的瓶颈难题成为可能,改善了传统海量数据访问带来的访问效率低下的状况. 本文基于 Hadoop架构设计了职工工资统计实例,并对该实例在单节点模式、伪分布模式和完全分布模式应用中的运行效率进行了分析和比较.

收稿日期: 2010- 07- 12
通讯作者简介: 田秀霞 (1976-),女,在读博士,副教授,河南汤阴人. 主要研究方向为信息安全,数据库安全,隐私保护. E-mail: tianxxsmile@yahoo.com.cn

© 1994-2011 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

火龙果•整理 uml.org.cn

1 Hadoop的工作原理

Hadoop是 Apache软件基金会所研发的分布式基础架构[3,4], 于 2005年推出, 它使用分布式文件系统 (Hadoop Distributed File System, HDFS) 作为低层存储支持. HDFS有着高容错性的特点, 并将其设计部署在低廉的硬件设备上, 以提供高传输率来访问应用程序的数据, 适合那些有着超大数据集的应用程序. 目前国内外著名的公司如 Yahoo, 阿里巴巴, 百度, Facebook等都建立了基于 Hadoop的应用. 下面分别从 Hadoop的 MapReduce实现机制和 HDFS低层存储来说明如何构建基于 Hadoop的分布式应用.

1.1 MapReduce实现机制

Hadoop是 MapReduce的实现[5,6], 而 MapReduce的工作过程一般分为两个阶段: map阶段和 reduce阶段. 每个阶段都有一批关键值对 < key, value> 作为输入, 而另一批关键值对 < key, value> 作为输出. 关键字的类型可以由程序员选择设定. 程序员可以根据实际应用具体设计两个函数的实现体, 在 map阶段输入的是原始数据, 可以选择文本文件作为输入.

1.1.1 气象数据集

分布在全球各地的气象传感器每隔 1 h便收集当地的气象数据, 从而累积了大量的日志数据, 这些数据是可以用 MapReduce来分析的最佳数据.

1.1.2 数据存放格式

假设数据是以面向行的 ASCII格式存储, 每一行是一条记录. 该格式支持许多气象元素, 其中许多数据的长度可选或可变. 为简化起见, 我们将选择始终都有固定宽度的数据 (如温度)进行讨论. 数据文件按照日期和气象站进行组织, 从 1901~ 2010年, 每一年都有一个目录, 每一个目录都包含一个打包文件, 文件中的每一个气象站都带有当年的数据. 实际生活中有上万个气象台, 所以整个数据集由大量较小的文件组成. 通常情况下, 我们更容易、更有效地处理数量较少的大型文件. 因此, 数据会被预先处理, 并将每年记录的读数连接到一个单独的文件中.

1.1.3 MapReduce过程

使用 map函数找出年份和气温, 在这种情况下, map函数处理过程仅是数据的一个准备阶段, map函数输出的是能让 reduce函数在其中工作的数据. map函数也可以很好地去除损坏的记录, 即在 map函数中过滤掉丢失、不可靠或错误的气象数据.

以下面的几行示例作为输入数据 (一些未使用的列已经去除, 为了符合页面宽度, 用省略号表示):

(0, 006701199099999919500515070004 … 9999999N9+ 00001+ 99999999999…)

(106, 004301199099999919500515120004 … 9999999N9+ 00221+ 99999999999…)

(212, 004301199099999919500515180004 … 9999999N9- 00111+ 99999999999…)

(318, 004301265099999919490324120004 … 0500001N9+ 01111+ 99999999999…)

(424, 004301265099999919490324180004 … 0500001N9+ 00781+ 99999999999…)

关键值是文件中的行偏移, 而这往往是 map函数中所忽视的. map函数的功能仅仅提取年份和空气温度 (用黑体表示), 并作为它的输出发送出去. 温度值已被转换成整数:

(1950, 0), (1950, 22), (1950, -11), (1949, 111), (1949, 78)

map函数的输出先由 MapReduce中的 shuffle来处理, 然后再被发送到 reduce函数. 这种数据是一个一个键值对. 因此, reduce函数有以下输入, 每年的年份后面都有一系列温度的数据. 所有的 reduce函数现在必须重复这个列表并从中找出最大的读数:

(1949, (111, 78)), (1950, (0, 22, -11)).

最后输出的全球气温中, 每年的最高温度为:

(1949, 111), (1950, 22)

整个数据流向如图 1所示. 在图 1的底部是 Unix的管道, 以模拟整个 MapReduce流程.

1.2 HDFS低层存储

HDFS分为 3个部分[3,4]: 客户端 (client); 名称节点 (NameNode); 数据节点 (DataNode). Client是基于 HDFS的应用程序; NameNode是分布式文件系统的管理者, 主要负责文件系统的命名空间, 集群的配置信息和数据块的复制信息等, 并将文件系统的元数据存储在内存中; DataNode是文件

© 1994-2011 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

火龙果•整理 uml.org.cn

实际存储的位置, 它将块 (Block)的元数据信息, 存储在本地文件系统中, 周期性地将所有的 Block 信息发给 NameNode.

HDFS的构成如图 2所示.

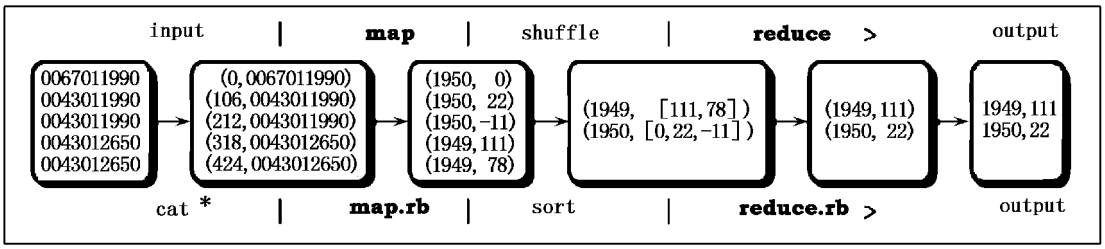

图 1 MapReduce逻辑数据流

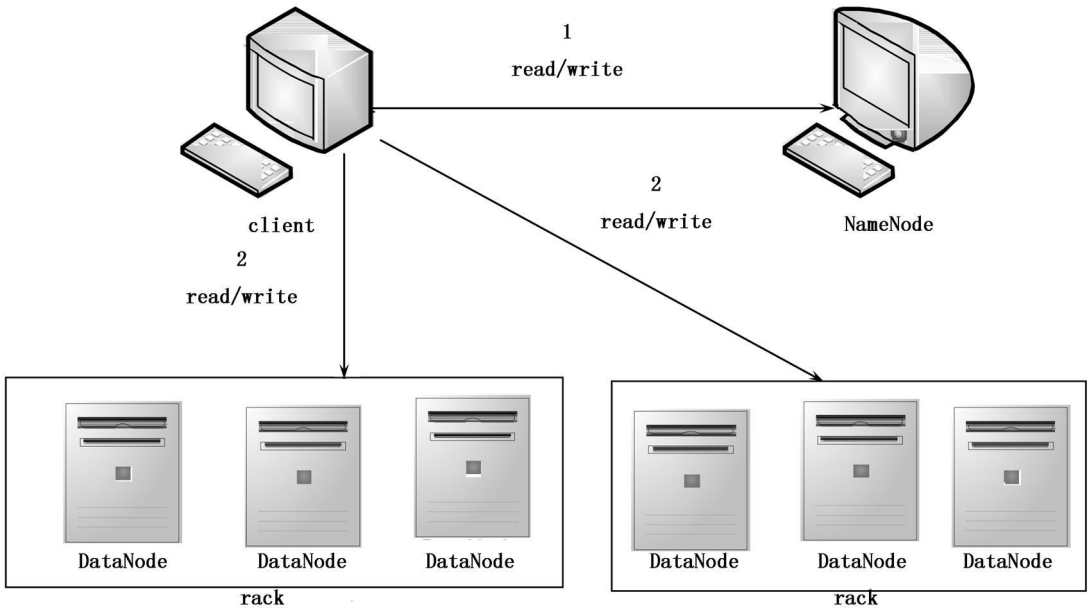

图 2 HDFS的构成

NameNode上的文件名称空间信息存储在 FsImage文件中, NameNode上还有一个事务日志文件 EditLog 这两个文件要有备份, 以防文件损坏或者 NameNode 宕机导致系统不可恢复. NameNode的作用主要有以下 3个: 一是文件映射, 把一个文件映射到一批数据 block, 把文件数据映射到 DataNode上; 二是集群的配置管理, 数据块的管理和复制; 三是日志处理.

2 职工工资统计的实现

根据前面的 Hadoop架构及其 MapReduce实现机制和 HDFS低层存储, 我们将一个公司所有员工的工资按照员工姓名、月工资存放于不同的文本文件中, 并通过编写统计工资的 map 和 reduce函数, 将计算输出的员工姓名、员工年工资存放在结果文件中, 如图 3所示.

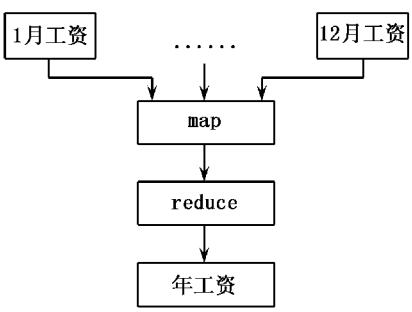

图 3 工资统计的 MapReduce过程

2.1 程序代码及分析

由 TokenizeMapper类和 IntSumReducer类分析工资统计功能的 MapReduce实现.

2.1.1 TokenizeMapper类工资统计功能的实现

定义 TokenizeMapper类, 继承 Hadoop提供

© 1994-2011 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

火龙果•整理 uml.org.cn

的 Mapper类来实现个性化的 map函数. map函数中, 由于输入的文本内容格式固定, 故将文本文件中一行数据 (value) 提取出 name和 monthSalary后, 写入 context. 其中用到的 IntWritable类是 Hadoop为了统一格式所自带的封装类, 与 Integer无异. 此类 map方法将分析后的结果写入 context, 作为 IntSumReducer类 reduce方法的输入.

TokenizeMapper类中 map函数的基本算法思想为:

(1) 读取文本文件中的一行数据 StringTokenizer itr= new StringTokenizer(value. toString());

(2) 获取姓名 String name= itr. nextToken();

(3) 获取月薪 IntWritable monthSalary= new IntWritable(Integer. parseInt(itr. nextToken()));

(4) 计算结果输出 将结果写入 context, 作为 reduce方法的输入 context. write(word, monthSalary).

2.1.2 IntSumReducer类工资统计功能的实现

定义 IntSumReducer类, 继承 Hadoop提供的 Reducer类来实现个性化的 reduce函数. reduce函数中, key为 map提取出的姓名作为输出参数, values为 map之后通过 combiner归并该名员工工资的集合, 对每一名员工, 假设其姓名是唯一的, 那么可以作为主键区别. 计算其对应的月工资的总和 (年工资), context输出年工资.

IntSumReducer类中 reduce函数的基本算法思想:

(1) 计算年工资总和 根据 key和 values计算每个员工的年工资总和

```
for( IntWritable val: values)
  {
    sum + = val. get();
  };
```

(2) 计算结果输出 将结果写入 context, 并显示输出 context. write(key, result).

在完成了用于 map的 TokenizeMapper类和用于 reduce的 IntSumReducer类之后, 通过 Hadoop所提供的应用类去设置运行时的一些参数, 最主要的是 Job类, 其中有许多设置, 包括 Jar类、Mapper类、Combiner类、Reducer类, 以及输入输出路径等, 通过 waitForCompletion方法执行任务, 直至完成.

2.2 运行结果

运行完成后输出的文本文件的内容格式为员工姓名、年工资, 符合预期的运行结果. 程序的 TokenizeMapper类实现了 Mapper接口的 map方法, 将输入的文本文件逐行读取, 并以 "空格" 来划分, 将员工姓名作为 key, 其月工资作为 value. map的结果集再由 combiner归并后输出到 reducer, 对相同 key的结果, 求其 value的和, 并将其输出到对应目录下.

3 职工年工资统计效率测试

分布式计算法适合于大数据量的计算, 大集群的效率无可置疑, 但由于实验条件的限制, 只能测试双机集群的效率.

3.1 主要配置参数

两台主机的主要软硬件配置见表 1.

表 1 主要配置参数

名 称	CPU主频/MHz	高速缓存/kB	内 存/MB	操作系统
computerA	800×2	512	1 000	Ubuntu
computerB	1 826.106	256	512	Ubuntu

3.2 测试方案

单节点模式: 不涉及分布式文件系统, 在 Eclipse中运行 (需配置 Hadoop插件和运行参数).

伪分布模式: 运行于 A机.

分布式模式 (A - B): A作为 master, B作为 slave, 下传速度平均为 2MB左右.

分布式模式 (B - A): B作为 master, A作为 slave, 下传速度平均为 3.6MB左右.

3.3 测试结果

由 1MB数据量至 1GB数据量进行测试效率的比较, 运行时间结果记录见表 2

© 1994-2011 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

火龙果•整理 uml.org.cn

表 2 不同测试模式下不同文件的运行时间

文件大小 /MB	测试模式 /10^4 ms			
	单 机	伪分布	分布式 A-B	分布式 B-A
1	0.513 9	3.004 1	4.659 0	2.566 3
4	0.881 1	3.642 4	5.963 9	2.992 9
16	1.457 3	5.294 7	15.534 2	5.217 6
64	4.556 4	14.333 1	55.492 9	15.771 8
256	20.841 2	52.651 1	226.413 5	65.911 0
1 024	86.409 0	175.609 5	时间太长	279.291 5

4 结 语

基于 Hadoop架构计算模式的出现, 突破了传统的数据库系统对海量数据处理的速度限制, 通过对大量数据的并发访问, 有效缩短了数据的查询计算时间, 使用户的查询响应更快速. 本文通过比较不同大小文件中的工资数据在 4种不同分布下的运行效率, 得到分布式 A-B的效率最高, 单机的效率最低. Hadoop架构模式可以充分利用不同机器同时参与并发运算, 提高了运算效率. 目前 Hadoop架构已广泛应用于信息检索领域, 如 Google, Baidu等. 我们将继续对一些符合分布式运算的算法在 Hadoop架构下的实现方案进行集中研究, 并根据其特点应用于特定的场景.

参考文献:

[1] 易剑. Hadoop开发者第 1期 [R/OL]. [2009-07-11] http://ishare.iask.sina.com.cn/f/9389548.html?from=like.

[2] 易剑. Hadoop开发者第 2期 [R/OL]. [2010-01-30] http://ishare.iask.sina.com.cn/f/9389548.html?from=like.

[3] Tom White. Hadoop: The Definitive Guide[M]. America: O'Reilly Media, 2009: 18-25.

[4] 维基. Hadoop[OL]. http://en.wikipedia.org/wiki/Hadoop, 2010.

[5] 王耀聪, 陈威宇. Map-Reduce编程 [CP/OL]. [2010-01-20] http://trac.nchc.org.tw/cloud.

[6] 王鹏. 走近云计算 [M]. 北京: 中国人民邮电出版社, 2009: 100-120.

(上接第 69页)

security protocols for sensor networks[C]//In Proceeding of MobiCom 2001, 2001: 189-199.

[10] ZHU S, SETIA S, JAJODIA S. LEAP: efficient security mechanisms for large-scale distributed sensor networks[C]// Proc. of the 10th ACM Conf. on Computer and Communications Security, 2003: 62-72.

[11] ELTOWEISSY M, HEYDARI H, MORALES L, *et al*. Combinatorial optimization of key management in group communications[J]. J. Network and Systems Management, 2004, 12(1): 33-50.

[12] WEN M, ZHENG Y F, YEW Jun, *et al*. A key management protocol with robust continuity for sensor networks [J]. Journal of Computer Standards & Interfaces, 2009, 31(4): 642-647.

[13] WEN M J, LEI Jing-sheng, TANG Zhong, *et al*. A verified group key agreement protocol for resource-constrained sensor networks[C]//WISM, 2009, LNCS 5854, 2009: 413-425.

[14] DAVID Malan, THADDEUS R F, FULFORD-Jones, *et al*. CodeBlue: an ad hoc sensor network infrastructure for emergency medical care[R]. Emerging Technology and Best Practices Seminar, Boston Univesity, Boston, Massachusetts, [2004-05-01]. http://www.cs.harvard.edu/malan/publications/bsn-codeblue-short.pdf

[15] GOUVEAN C P L, LOPEZ J. Software implementation of pairing-based cryptography on sensor networks [C]// Indocrypt 2009, Springer, LNCS 5922, 2009: 248-262.

[16] TAN C C. Body sensor network security: an identity-based cryptography approach[C]//Proceedings of the First ACM Conference on Wireless Network Security, 2008: 148-153.

[17] BOGDANOV A, LEANDER G, PAAR C, *et al*. Hash functions and RFID tags: mind the gap[C]//CHES 2008, LNCS 5154, Springer, 2008: 283-299.

© 1994-2011 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net