基于 Hadoop 的海量共现矩阵生成方法*

杨代庆[1,2] 张智雄[1]
[1](中国科学院国家科学图书馆 北京 100190)
[2](中国科学技术信息研究所 北京 100038)

【摘要】海量数据的处理分析是当前信息处理技术的热点之一,介绍开源并行系统 Hadoop 的体系结构以及基于 Hadoop 的 MapReduce 编程框架,并在 Hadoop 基础上提出一种通过多重 MapReduce 操作,实现海量共现矩阵的生成方法。

【关键词】Hadoop MapReduce 共现矩阵 开源软件
【分类号】G350

A Method for Generating Co-occurrence Matrix of Mass Data Based on Hadoop

Yang Daiqing[1,2] Zhang Zhixiong[1]
[1](National Science Library, Chinese Academy of Sciences, Beijing 100190, China)
[2](Institute of Scientific and Technical Information of China, Beijing 100038, China)

【Abstract】Mass data processing is a focal point of information techniques. This paper introduces architecture of open source parallel system - Hadoop, analyzes the MapReduce programming framework based on Hadoop, and proposes a method for generating co-occurrence matrix of mass data through multiple MapReduce operations.

【Keywords】Hadoop MapReduce Co-occurrence matrix Open-source-software

1 介 绍

信息化的今天,如何从海量数据中发现特定知识,如何高效地处理海量数据几乎是任何一个信息分析机构要面对的问题。一般来说,并行计算能够加快处理速度,而并行计算也有多种选择,例如,在 Web Service 基础上的网格计算、基于 P2P 的点对点计算、基于服务器集群的"云计算"等。虽然方法众多,但却需要根据实际情况进行选择,以 Google 为代表的"云计算"以其应用简单、高效得到了广泛认可。它通过在分布式文件系统 GFS 基础之上的 MapReduce 编程模型以及廉价集群的建立,解决了许多大规模数据的计算问题。

由雅虎资助的开源项目 Hadoop,是一个类似于 Google "云计算"的技术平台,专注于海量数据存储、处理的分布式系统,同时提供了基于 Java 的 MapReduce 框架,能够将分布式应用部署到大型廉价集群上。Amazon 公司基于 Hadoop 推出了 Amazon S3 (Amazon Simple Storage Service),以提供可靠、快速的网络存储服务,而在 IBM 公司的云计算项目——"蓝云计划"中,Hadoop 也是其中重要的基础软件。与此同时,Hadoop 在图书馆也得到了应用,例如,Nutch 中的分布式搜索、索引,以及分布式 Lucene 都有使用。

收稿日期:2009-03-28
收修改稿日期:2009-04-02
*本文系国家"十一五"科技支撑计划子课题"网络科技信息监测与评价"(项目编号:2006BAH03B05)的研究成果之一。

本文的背景是在“网络信息科技监测与评价”课题研究中需对文本进行共现分析,而在实际过程中,使用单一服务器,在特定时间限制的情况下,完成大数据量的处理存在困难,因而需要寻求一种简便、可行、有效的并行计算方法。根据调研分析,Hadoop 最终成为解决该问题所采用的工具。

2 Hadoop 架构及配置

2.1 Hadoop 架构

Hadoop 能够实现高效计算,存储的核心在于其可运行于大规模集群上的分布式文件系统 HDFS(Hadoop Distributed File System)以及 MapReduce 分布式并行编程框架。

HDFS 采用 Master/Slave 架构,HDFS 架构如图 1 所示。一个 HDFS 集群由一个 Namenode 和一定数目的 Datanode 组成。Namenode 是中心服务器,管理文件系统的 Namespace 和客户端对文件的访问。Datanode 在集群中一般是一个节点一个,负责管理节点上附带的存储。在 Datanode 内部,一个文件被分成多个 Block。Datanode 在 Namenode 的指挥下进行 Block 的创建、删除和复制。典型的部署方式是一台机器运行一个单独的 Namenode 节点,而其他机器各运行一个 Datanode 实例。

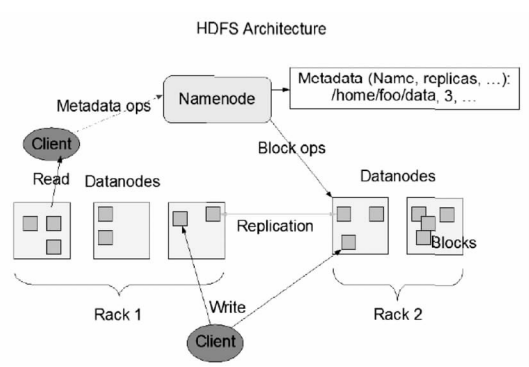

图 1 HDFS 架构[1]

2.2 数据复制分发

HDFS 被设计成可在集群中跨机器存储海量文件的文件系统。它将每个文件存储成 Block 序列,除最后一个 Block 外,所有的 Block 大小相同,并且所有 Block 都会建立副本。Namenode 周期性地从集群中的每个 Datanode 接收心跳包(Heart - beat)和一个 Blockreport。心跳包用于监测该 Datanode 节点是否工作正常,而 Blockreport 内则包括了该 Datanode 上所有 Block 组成的列表。默认情况下,HDFS 保存三个副本,Datanode 的死亡可能引起一些 Block 的副本数量低于预定值,Namenode 会不断跟踪需要复制的 Block,在需要时启动复制,如图 2 所示:

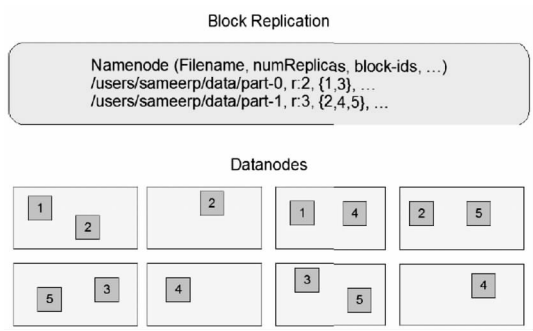

图 2 HDFS 副本存储示例[1]

2.3 Hadoop 配置

Hadoop 有三种运行方式:单机模式、伪模式和完全分布式模式。

单机模式、伪模式配置简单,这里仅讨论完全分布模式的配置。在实际使用中,笔者使用三台机器来搭建集群,其中一台作为 Namenode,另外一台作为 SecondNamenode,第三台作为 Datanode,并最好保证每台机器的路径及文件位置完全相同。

(1)配置 SSH

在每台机器上利用 SSH 生成密钥对,并且将彼此公钥追加到 authorized_keys 文件中,分发到各台机器,以保证机器之间能够不输入密码就能直接登录,同时每台机器自身也要求不输入密码就能登录。

(2)配置文件

①hadoop - env. sh 文件中修改 Java 运行环境路径。

②slaves 及 masters 文件。

在每台机器的 slaves 以及 masters 中分别添加 Datanode 及 Namenode 的 IP 地址,每行一个。

③hadoop - site. xml。一般情况下需设置如表 1 所示的参数。

(3)启动 Hadoop

格式化:在 Namenode 机器上执行:hadoop namenode - format

运行:start - all. sh

停止:stop - all. sh

表 1　参数配置表[2]

参数名称	参数说明
fs. default. name	缺省文件系统 URI,可设置为 Namenode 地址
mapred. job. tracker	作为 jobtracker 的机器地址及端口
mapred. job. tracker	副本数量
hadoop. tmp. dir	Hadoop 临时文件路径
dfs. name. dir	持久化存放 NameNode 的元数据和事务日志的目录,是本地文件系统路径。支持多个目录,用逗号分隔,每个目录都会存放元数据的一个拷贝。用来保证 NameNode 出故障时不会丢数据
dfs. data. dir	用逗号分隔的多个本地文件系统目录,用来存放 DataNode 的数据块
mapred. system. dir	HDFS 路径,用来存放 Map Reduce 产生的系统文件
mapred. local. dir	逗号分隔的多个本地文件系统路径,用于存放 Map Reduce 的临时数据。多个目录可以增加 I/O 吞吐
mapred. tasktracker. map. tasks. maximum	可以同时运行的 Map Reduce 的任务数。缺省是 2 (2 maps and 2 reduces),根据机器硬件环境调整
mapred. tasktracker. reduce. tasks. maximum	

3　MapReduce 程序框架

3.1　原　理

MapReduce 是用于并行处理大数据集的软件框架。它由可能包含多个 Map 和 Reduce 的操作组成。计算模型的核心是 Map 和 Reduce 两个函数,这两个函数由用户实现,功能是按一定的映射规则将输入的 <key, value> 对转换成另一个或一批 <key, value> 对输出。

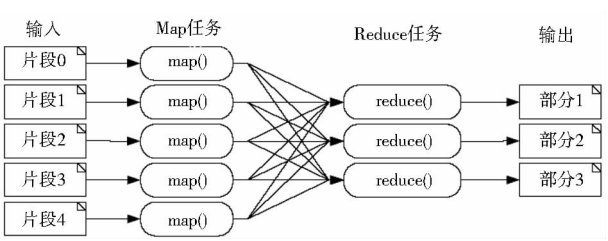

图 3　MapReduce 过程[3]

图 3 说明了用 MapReduce 来处理大数据集的过程,简而言之 MapReduce 的计算过程就是将大数据集分解为成百上千的小数据集,每个(或若干个)数据集分别由集群中的一个结点进行处理并生成中间结果,然后这些中间结果又由大量的结点进行合并,形成最终结果。

利用 MapReduce 编写分布式并行程序,主要是实现 Map 和 Reduce 函数,其它并行编程中的复杂问题,均由 Hadoop 处理。最简单的 MapReduce 应用程序只需包含三个部分:Map 函数、Reduce 函数和 Main 函数。Main 函数将作业控制和文件输入/输出结合起来。

3.2　处理流程

客户机在主系统上启动的 MapReduce 应用程序称为 JobTracker,类似于 NameNode,它是 Hadoop 集群中唯一控制 MapReduce 应用程序的系统。在应用程序提交时,需提供包含程序在 HDFS 中的输入、输出目录。JobTracker 使用文件块信息创建 TaskTracker(每个 TaskTracker 是 JobTracker 的子任务)。MapReduce 应用程序被复制到每个出现输入文件块的节点,并为特定节点上的每个文件块创建 TaskTracker。每个 TaskTracker 会将状态报告给 JobTracker。完成计算后,JobTracker 将结果存放到输出路径中。图 4 给出了一个集群中的作业执行分布状况图。

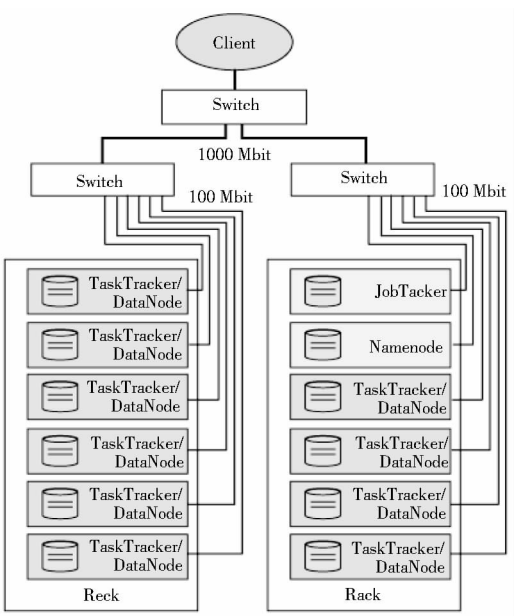

图 4　作业分布示例[4]

4　海量共献矩阵处理算法及实现

假设:有词串文档集合 $A = \{(X_i, Y_j) \mid X \in Y$ 中包含的词串,$Y \in$ 文档名,i,j 自然数$\}$ 求基于文档的词串共现矩阵 $B = \{b_{mn} \mid (X_m, Y_j) \times (Y_j, X_n)$ 中元素总数,$m > n$ $Y_j \in$ 文档名$\}$ 通常的处理方式是将数据导入数据库中,在数据库中对 A 集合自身做连接运算,得到具有共现关系的词串对,不妨设为(X_i, X_j)。对所有词串对根据 i,j 的不同值,分别统计词串对的个数,得到词串

X_i,X_j 的共现强度。

上述处理方式在面对海量数据处理时会遇到一些不容易解决的问题。主要问题在于,在数据库中做大数据量的连接运算非常消耗存储空间以及时间,因而在有限的时间内难以得到最后结果。一种解决办法是在增加机器数量,利用并行处理技术,实现时间节省。

提出的处理方法如下:

(1)对每个词串文档对(X_i,Y_j)进行预处理,生成 X_i,‖ Y_j 格式,并且用空格分隔词串文档对,将所有处理后的数据保存为一个或数个文本文件。

(2)Map(line) - > 将输入行进行空格拆分后,得到若干个 Tokens,每个 Tokens 形如:X_i,‖ Y_j。对每个 Tokens 按“‖”进行拆分,得到若干个子 Tokens。

(3)将(2)中的每个子 Tokens 转换为 < Y_j, X_i >,对每个 Y_j 建立变量 $List_j$,保存对每个 Y_j关联的词串联接,并以“‖”分隔。

If 存在 < Y_j, X_i > {$List_j$ + =“‖” + X_i}

Map 输出为:< Y_j, $List_j$ >

(4)Reduce(Y_j, $List_j$) - > 对每个 $List_j$ 作如下操作:

①按“‖”拆分,存放到数组 $TempList_j$

②For(int m = 0;m < $TempList_j$. count;m + +)

For(int n = 0,n < j;n + +)

③生成 < $TempList_j$[n] +“‖” + $TempList_j$[m],“” >

(5)在完成(4)后可在输出目录得到第一次 MapReduce 后的结果文件(见表 2),然后在此结果文件基础上进行第二次 MapReduce 操作。

(6)Map(line) 将输入进行空格拆分后,得到若干个 Tokens,每个 Tokens 形如 X_i,‖ X_j。生成 < X_i,‖ X_j,1 >,将所有生成的 < X_i,‖ X_j,1 > 装入集合 values 作为 Map 输出。

(7)Reduce(< X_i ‖ X_j,values >) - > 对以 X_i ‖ X_j 为键值的每个 values 值求和,然后存入 sum,作为(X_i,‖ X_j)共现强度。输出 < X_i,‖ X_j,sum >

(8)对最终输出文件,按照“‖”进行拆分,由于矩阵过于稀疏,为节省内存可利用三元组表来存储共现矩阵。

5 结 语

上述算法的各个步骤得到实验数据片段如表 2 所示。

表 2 实验结果

操作过程	数据示例
原始数据	Books‖ ftp://194.44.214.3/pub/e - books/ linux‖ ftp://194.44.214.3/pub/e - books/ game‖ ftp://194.44.214.3/pub/e - books/ free‖ ftp://194.44.214.3/pub/e - books/ network‖ ftp://194.44.214.3/pub/e - books/ libros‖ ftp://194.44.214.3/pub/e - books/ ebooks‖ ftp://194.44.214.3/pub/e - books/ ……
第一次 MapReduce 后	Linux‖ game linux‖free linux‖ network linux‖ libros linux‖ ebooks game‖ free game‖ network game‖ libros game‖ ebooks free‖ network free‖ libros free‖ ebooks network‖ libros network‖ ebooks libros‖ ebooks ……
第二次 Mapreduce 后	Linux‖ game 155 linux‖free 1072 linux‖ network 625 linux‖ libros 17 linux‖ ebooks 55 ……

实验中,在数据量较小的情况下,采用多节点的 Hadoop 计算速度明显不如采用 Hadoop 单节点的速度快,也比 Hadoop 并行技术的方式慢很多。因此,对于小规模的运算,不适宜采用 Hadoop。另外 Hadoop 在处理诸如含有需要修改数据操作的算法时,会遇到困难,这是由其 HDFS 的“只写一次,多次读取”特点决定的。同时,对于快照支持某个时间的数据拷贝,当 HDFS 数据损坏的时候,可以恢复到过去一个已知正确的时间点等功能尚不支持。尽管如此,基于 Hadoop 的一批开源海量数据相关处理软件如 Hbase[5]、Hive[6]、Pig[7]、CloudBase[8]等,正被越来越多的人所使用。随着 Hadoop 的不断完善,它将会得到更多的应用和关注。

参考文献:

[1] HDFS Architecture[EB/OL]. [2008 - 12 - 10]. http://hadoop.apache.org/core/docs/current/hdfs_design.html.

[2] Hadoop Cluster Setup[EB/OL]. [2008 - 12 - 15]. http://hadoop.apache.org/core/docs/current/cluster_setup.html.

[3] HadoopMapReduce[EB/OL]. [2008 - 12 - 16]. http://wiki.apache.org/hadoop/HadoopMapReduce.

[4] Distributed Computing with Linux and Hadoop. [EB/OL]. [2009 - 01 - 10]. http://www.ibm.com/developerworks/linux/library/l - hadoop/index.html.

[5] Hbase[EB/OL]. [2009 - 01 - 10]. http://hadoop.apache.org/hbase/.

[6] Hive[EB/OL]. [2009 - 01 - 15]. http://hadoop.apache.org/hive/.

[7] Pig[EB/OL]. [2009 - 01 - 15]. http://hadoop.apache.org/pig/.

[8] CloudBase[EB/OL]. [2009 - 01 - 16]. http://sourceforge.net/projects/cloudbase/.

(作者 E - mail:yangdq@mail.las.ac.cn)