基于 Hadoop 框架的 MapReduce 计算模式的优化设计

孙彦超 王兴芬
(北京信息科技大学 北京 100192)

摘 要 对某高校教学资源平台的海量日志进行了分析,将传统单机分析处理模式,转变为 Hadoop 框架下的 MapReduce 分布式处理模式。MapReduce 采用分而治之的思想,很好地解决了单机对海量数据处理产生的瓶颈问题。通过分析 Hadoop 源码的使用,认真研究 MapReduce 对海量数据处理作业流程分析,提出了 MapReduce 分布式作业计算的优化策略,从而更好地提高了海量数据的处理效率。

关键词 Hadoop,海量数据,MapReduce,分布式计算
中图法分类号 TP301 **文献标识码** A

MapReduce Designed to Optimize Computing Model Based on Hadoop Framework

SUN Yan-chao WANG Xing-fen
(Beijing Information Science & Technology University, Beingjing 100192, China)

Abstract Aiming at a university teaching resource platform for massive log analysis, analysis and processing are transformed from the traditional stand-alone mode to using Hadoop MapReduce framework under the distributed processing. MapReduce uses the idea of dividing and rule, which is good solution to the bottleneck problem alone generated massive data processing. Through the use of Hadoop source code analysis and careful study of massive data processing using MapReduce job flow analysis, this paper presented optimization strategy MapReduce distributed computing operations to better improve the processing efficiency of massive data.

Keywords Hadoop, Massive data, MapReduce, Distributed computing

大数据时代的到来对于高校管理者而言,既带来了挑战,同时也带来了机遇。一方面,以往依赖的数据统计分析以及决策管理工具已经趋于淘汰,迫使管理者构建新的平台。另一方面,大数据的分析研究得到了空前的重视,研究者已经从不同角度,采用各种方法、各种技术手段从中挖掘出隐含的有价值的信息。管理者对海量数据统计分析时,通常要求系统尽可能在最短时间内返回最终结果。虽然可以通过基于关系数据库的内存计算平台,或者并行处理集群,以及采用 HDD 的架构来实现,但均需要在购置软硬件上花费巨额成本。由于传统的 ETL(Extract-Transform-Load)工具面对海量信息处理时开销过大,无法达到统计分析的性能需求,因此分布式计算架构 Hadoop 应运而生。

Hadoop 是 Apache 软件基金会提供的一个开源分布式计算平台。它是 Google 的 GFS(分布式文件系统)和 MapReduce(分布式计算)的 java 语言的实现,利用该平台可以轻松地对海量数据进行分布式处理. 由于该平台是一个开源代码的平台,因此任何机构及个人均可在网上免费获取其源代码,根据自身需求对其修改,同时该平台可以运行在廉价的服务器机群上,再加上在海量数据计算和存储方面具有高效性、高容错性和高扩展性,因而它在日志存储分析、金融领域、电子商务、在线搜索、客户分析等领域得到了广泛应用。

1 Hadoop 平台及 MapReduce 计算模式介绍

Hadoop 是一个分布式并行计算平台,具有高可靠性、高可扩展性、高效性及高容错性等优点。用户可以完全不了解平台底层实现细节,利用平台提供的接口,根据自身需要编写分布式计算程序,对海量数据进行分布式存储和计算。

Hadoop 平台主要由 HDFS(Hadoop Distributed File System)、MapReduce(分布式计算模型)、HBase(列式数据库)、Hive(SQL 解析引擎)和 ZooKeeper(分布式应用程序协调系统)等组成,如图 1 所示。其中,HDFS(分布式文件系统)、MapReduce(分布式计算模型)是 Hadoop 平台中最核心最基础的重要组成部分。HDFS 是一个分布式文件管理系统,负责集群中各节点(DataNode)数据的存取;MapReduce 是建立在 HDFS 技术上进行分布式计算的模式。

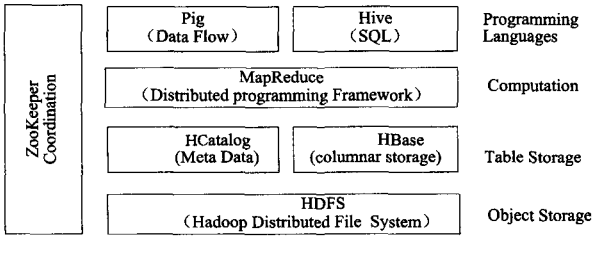

图 1 Hadoop 平台

1.1 HDFS 文件系统

HDFS 是一个分布式文件系统,采用冗余的方式存储文件,具有较高的可靠性和容错性。该系统以数据流的访问方式读取文件系统中的数据,极大地提高了系统存取数据的吞

孙彦超(1978—),男,硕士生,主要研究方向为数据库与信息系统、数据挖掘等;王兴芬(1968—),女,教授,博士,主要研究方向为高等教育管理。

吐量，因此，HDFS特别适合搭建基于海量数据的处理平台。HDFS架构采用Master/Slave(主从)架构。一般HDFS系统由一个主节点(NameNode)和多个从节点(DataNode)构成，如图2所示。主节点(NameNode)负责维护集群中的元数据，维护整个分布式文件系统的目录，通过响应的心跳控制DataNode。DataNode负责管理节点上数据存储和和响应读写请求，并定期通过心跳机制响应主节点回报节点状态。在DataNode内部，每个文件实际是被分成一个或多个块(block)来存储。

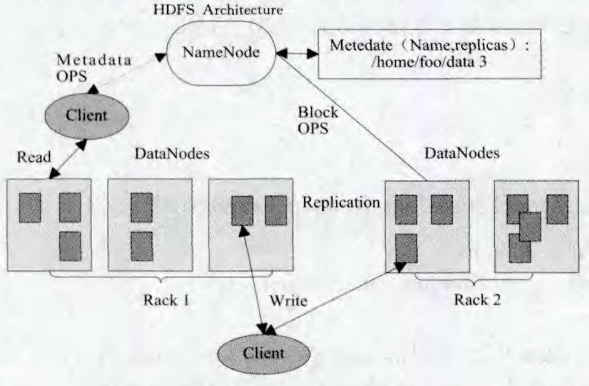

图2 HDFS文件系统架构

1.2 MapReduce分布式计算模型

MapReduce是一个分布式计算模型，是Google最早提出用来进行创建和更新索引的一种分布计算模式，该模式提供了一个简单分布式计算框架，来降低分布式计算编成的难度，从而很好地解决了一般商用机及服务器面对海量数据计算响应过慢甚至不能完成作业的问题。目前MapReduce主要用于海量数据的分布式计算，它主要起源于函数编程思想中的Map函数和Reduce函数操作。Map函数的操作原理是把一组数据映射为一个键值，Reduce函数的工作原理是对Map函数输出结果进行合并处理。比如，输入信息为The latest news and headlines from Yahoo! News. Get breaking news stories and in-depth coverage with videos and photos 在此信息上运行Map操作输出的结果为：

(The,1) (latest,1) (news,1) (and,1)
(headlines,1)(from,1)(Yahoo,1)(news,1)
(Get,1)(breaking,1)(news,1)(stories,1)
(and,1)(in-depth,1)(coverage,1)(with,1)
(videos,1)(and,1)(photos,1)

在此结果上进行Reduce运算，结果如下：

(The,1)(latest,1)(news,3)(and,3)
(headlines,1)(from,1)(Yahoo,1)
(Get,1)(breaking,1)(stories,1)
(in-depth,1)(coverage,1)(with,1)
(videos,1)(photos,1)

在Hadoop框架中，MapReduce以组件的模式工作，主要包括JobTracker和TaskTrackers两个组成部分。JobTracker是一个master服务，负责JobTracker接收及分配作业，并调度作业对应的子任务运行在TaskTracker上，MapReduce框架图如图3所示。

MapReduce的工作流程如下：

1. 用户提交一个作业，该作业被发送到JobTracker服务器上，JobTracker是MapReduce的核心，它通过心跳机制管理所有的作业。

2. TaskTracker为MapReduce集群众的一个工作单元，主要完成JobTracker分配的任务。

3. TaskTracker监控主机任务运行情况，通过心跳机制向JobTracker反馈自己工作状态。

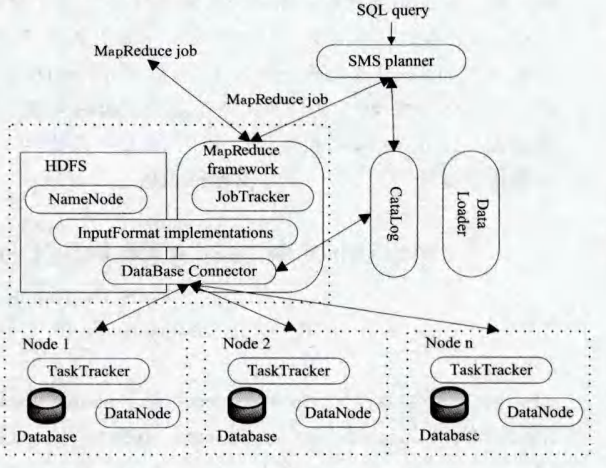

图3 MapReduce框架图

MapReduce架构简单，可以对海量数据进行高效的分布式运算。然而随着数据量的巨增，该架构会出现以下问题：

JobTracker和NameNode一样，存在着单点故障。如果节点过多，JobTracker需要完成的任务数量过大，会造成资源消耗过多，影响整个集群运行效率。如果I/O属性设置不当，会增加读写磁盘次数，从而影响系统性能。

MapReduce属性值设置不合理，也会造成系统资源浪费。

2 MapReduce计算模式优化

本文通过对Hadoop框架深入研究，使用HDFS文件系统和MapReduce计算框架来处理某高校教学管理平台海量日志，每天管理平台从各服务器将日志导入到HDFS系统时，由于HDFS适合管理大文件，因此首先将日志文件合并存储。在HDFS中存入海量数据，从而完成日志处理需求：

1. 分析平台各功能模块某一时间的访问量；

2. 分析平台中图像文件及视频文件的上传情况；

3. 分析某一IP或IP段地址的访问情况；

4. 分析各个网页的访问情况。

通常每台服务器每天产生的日志大小约在3G～5G之间。所以可以方便地使用MapReduce来进行分析。在对平台日志分析处理时，主要的工作是对平台参数进行设置，MapReduce根据设置的参数执行作业，其中需要设置的关键参数如下：

1. 日志文件上传时是否合并，上传文件存放的目录；

2. 执行作业(Job)时间间隔，数据分析时按天还是按小时；

3. 作业执行完的输出格式，及输出路径等。

日志分析时，具体处理流程图如图4所示。

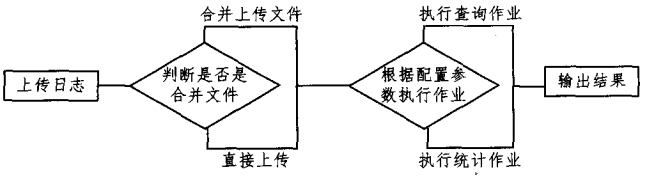

图 4　日志分析流程

本文对 MapReduce 优化的思路是：1）由于 I/O 读写磁盘次数会降低系统的性能，因此尽可能地利用内存区域来存储数据，同时考虑分布式并行运算，从而提高整个集群性能。2）根据集群自身的特性，针对集群特性反复进行实验，从而提出合理的优化策略。

2.1　I/O 参数优化

I/O 参数主要指 Shuffle（combine，partition，combine 的组合）过程中涉及的 I/O 操作属性，在对 MapReduce 模型深入分析及进行了大量实验验证每个重要属性参数后，对以下重要属性优化：

1. 参数：io. sort. mb，该参数是指在进行 Map 运算时，运行结果暂时存放的缓冲区（Buffer），该缓冲区的大小默认为 100MB，若该参数的大小设置不当，将影响并行运算的整体性能，因此当 Map 运算结果输出信息量较大时，需要调整该参数的值，从而减少系统对磁盘读写的次数，以便减少 I/O 开销提高运行效率。

2. 参数：io. sort. spill. percent，该参数是 io. sort. mb 的阈值，一般默认值为 0. 8，当缓冲区（Buffer）中存放的信息达到该阈值后，系统会对缓冲区中的数据进行排序，同时写入磁盘，同时 Map 运算结果继续向剩余的百分之二十缓冲区写入，如果剩余缓冲区写满，但排序还没结束，此时 Map 运算任务需要等待。如果 Map 运算结果基本有序，该参数可以调高，从而提升系统性能。

3. 参数：io. sort. factor，该参数指进行 Map 运算和 Reduce 运算时需要将 spill 文件进行合并（merge sort），每次同时打开 spill 文件个数就是由该参数设定的。其默认值是 10，根据集群的作业量，可相应调整从而提高并行计算的能力，在实验中增加到 50～100 时，性能有显著提高。

4. 参数：io. file. buffer. size，该参数指 MapReduce 进行 I/O 操作的缓冲区大小，默认为 4kB，可以增大该值来减少访问 I/O 次数，以最终提高系统处理性能。

5. 参数：io. sort. record. percent，该参数指 Map 运算数据在 io. sort. mb 中占内存的比例，默认值为 0. 05，需要根据集群特点调整，以便 io. sort. mb 的内存得到充分利用。

2.2　MapReduce 对象参数优化

MapReduce 对象参数包括在并行计算过程中涉及的相关参数，主要对以下参数进行性能分析并优化。

1. 参数：mapred. job. shuffle. merge. percent，指 Map 输出缓冲区使用比例阈值，当达到此阈值，缓冲区中的数据将会被归并然后 spill 到磁盘。默认值为 0. 66。增加到 0. 8 就可以减少磁盘读写次数，提高运算性能。

2. 参数：mapred. job. reduce. input. buffer. percent 指在执行 Reduce 任务时，保存 Map 输出占内存的比例。默认值为 0，在执行 Reduce 任务前，Map 任务的所有输出全合并到磁盘上，从而为 Reduce 任务提供最多内存。但是，在很多情况下 Reduce 任务需要的内存相对较少，因此可以增加该值以便减少访问磁盘的次数，从而提高 Reduce 运算性能。

3. 参数：mapred. reduce. parallel. copies，指 Reduce 任务从 Map 任务服务器复制文件的线程数量，其默认值为 5，可以根据需要增加至 50，从而修改执行 Reduce 任务的并行数量，提高系统运算性能。

4. 参数：mapred. job. shuffle. input. buffer. percent，指 Reduce 任务在 Shuffle 阶段分配给 Map 运算输出数据缓存区占内存的比例，默认值为 0. 7，增大该值可以减少访问磁盘次数，从而提高 Map 运算速度。

5. 参数：tasktracker. http. threads，指 Map 运算结果输出到 Reducer 的线程数量，默认为 40，可以增加到 50，增加并行运算线程数，从而提高传输效率。

6. 参数：mapred. inmem. merge. threshold，指 Map 运算结果输出到缓冲区中文件数，默认值为 1000，可以将该值设置为 0，从而取消文件数量限制，进而减少磁盘读写次数，提高集群运行性能。

3　优化后 MapReduce 计算模型实验及性能分析

本文使用 MapReduce 框架对某高校教学资源平台的海量日志进行分析，对 I/O 参数及 MapReduce 参数进行优化。在整集群中，总共包括 6 台服务器，其中 1 台主服务器为 NameNode 主服务器，其余为 5 台 DataNode 服务器，其中每台服务器的具体配置如表 1 所列。

表 1　服务器配置表

CPU	Intel 四核 Xeon E5530 2. 4GHz
Memory	32G
Hard Disk	1T/10k 6Gbps SAS 2. 5″
NetWork	NetLink BCM5784M Gigabit Ethernet
Hadoop	0. 20. 2
linux	Redhat as 5. 5
Java Version	1. 6. 0_02

进行分析的数据为服务器日志，大小为 10G，通过对系统优化 I/O 参数及 Reduce 参数优化，系统优化前后系统访问时段完成时间统计对比图如图 5 所示。

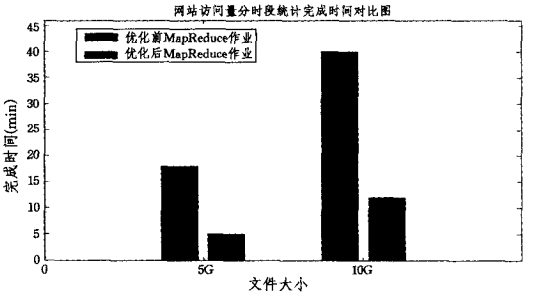

图 5　网站访问两份时段统计完成时间对比图

由图 5 可以看出，完成同样的日志分析工作量，优化后 MapReduce 作业的完成时间比优化前显著减少，这主要是系统通过增大缓冲区，减少了磁盘访问次数，从而提高了工作效率。同时对运行 MapReduce 任务的服务器 CPU 利用率、内存使用情况、集群间数据通信流量，系统负载等指标进行分析，可以看出，优化后的集群在整体性能方面比优化前提高了 30%左右。

结束语　针对单机对海量数据处理瓶颈问题，本文通过

利用 Hadoop 框架的 MapReduce 分布式处理模式对高校教学资源平台产生的海量日志进行分析，通过在该框架下反复的实验以及减少 I/O 访问次数和调整 MapReduce 参数等方法，来提高整个集群并行处理速度，从而更好地使用 MapReduce 分布式作业计算策略进行海量数据分析处理。

参 考 文 献

[1] 汤姆. Hadoop 权威指南[M]. 北京：清华大学出版社，2010：63-65

[2] HDFShttp://hadoop. apache. org

[3] MapReducehttp://hadoop. apache. org

[4] 徐子沛. 大数据：正在到来的数据革命[M]. 桂林：广西师范大学出版社，2012：23-30

[5] 李小庆. 银行面向大数据分析决策系统的构建[J]. 金融科技时代，2013：1-2

[6] 刘欢，张瑾. 数据挖掘改善校园网体验 [J]. 中国教育网络，2012(1)：27-30

[7] 范范. 大数据前景展望[N]. 网络世界，2012，(5)

[8] 李开复. 云计算[J]. 中国教育网络，2008(6)：34

[9] NfcKinsey Global Institute. Big data：The next frontier for innovation ompetition and productivity [R]. 2011(1)

[10] 白云川. 迎接大数据的时代[J]. 中国制造业信息化，2011(2)

[11] 蒋杰. Big Data 技术综述[J]. 程序员，2011：2-3

[12] 董彩云，等. 数据挖掘及其在高校教学系统中的应用[J]. 济南大学学报：自然科学版，2004：1-2

[13] 大数据时代下. 企业信息管理的新革命[J]. 网络与信息，2012(4)：7

(上接第 326 页)

数据集的查询响应时间如图 7 所示。

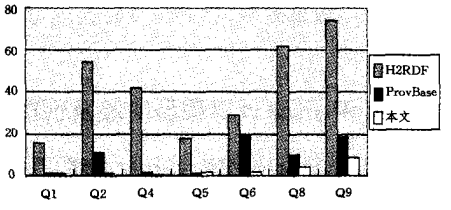

图 7 LUBM 100 个大学数据集的查询响应时间(单位：秒)

由图 7 可以看出，和 ProvBase 相比：1. HMSST 在 Q1、Q4、Q5 3 条简单的查询语句中和 ProvBase 的查询性能相当，ProvBase 利用 HBase Java API 查询时有效地利用了 HBase 提供的 row-key 索引，因此对于简单的查询语句具有很快的查询效率；2. 在 Q6 中，HMSST 的查询性能要比 PorvBase 好，因为 HMSST 是按 Type-P 划分存储的，对于 Type 已知、直接查询 S 的查询语句，具有很高的查询效率，而 ProvBase 的每张表都过于庞大，因此对于此类查询性能会略低。3. 在 Q2、Q8、Q9 这 3 条具有多个元组模式且连接操作较多的查询语句中，SST 提供的选择连接策略就发挥了很大的查询优化效果，使 HMSST 算法的查询性能在这 3 条上优于 ProvBase。而对于 H2RDF，因为 H2RDF 采用了 MapReduce 处理 BGP 连接，而一个 MapReduce 作业的启动时间就要 3 秒，对于选择度很高的查询来说，启动时间就占了整体查询时间的大部分，所以其查询效率很低。

结束语 本文提出的 HMSST 算法有如下的特点与创新：1. RDF 存储系统通常都需要面对数据划分策略的选择，而本文采取的 Type-P 划分存储策略可以有效地缩小查询范围，且划分简单，速度快；2. 本文采取的两级缓存策略弥补了哈希表在高效存储查询时需要耗费较大内存的缺点；3. 我们提出了与哈希映射高效结合的 SST 选择策略，实现了复杂连接操作语句的快速查询。实验表明，我们的存储策略相比现有的查询存储方案，具有更小的存储代价、更快的查询速度，以及更高的查询能力，在大数据集下可以高效地工作，并且该优化方案在查询的元组模式个数较多和语义较复杂时效果更加明显。

本文的不足在于：因为是集中式查询，HMSST 算法局限于有限的内存下，数据的内外存交换耗费了大量的时间，90% 的查询时间要消耗在内外存交换中。因此本文以后的方向在于研究分布式缓存技术，将 HMSST 算法应用到分布式文件系统中，从而让其在更大的数据量下发挥出更有效的查询效率。

参 考 文 献

[1] 朱敏，程佳，柏文阳. 一种基于HBase的RDF数据存储模型[C]// Proceedings of the NDBC. 2013

[2] 王琰，田翠华，朱顺痣，等. 基于 SPARQL 查询小枝关联的 RDF 数据索引方案[J]. 厦门大学学报：自然科学版，2014，53(3)：322-329

[3] Schmidt M，Meier M，Lausen G. Foundations of SPARQL query optimization [DB/OL]. [2009-01-26]. The Computing Research. Repository，http://arxiv. org/abs/0812. 3788

[4] Abadi D J，Marcus A，Madden S R，et al. Scalable semantic web data management using vertical partitioning [C]// Chan C Y. Proceedings of the 33rd International Conference on Very Large Data Bases. New York：ACM，2007：411-422

[5] Weiss C，Karras P，Bemstein A. Hexastore：sextuple indexing for semantic web data anagementl [C]// Hass L. Proceedings of the 34rd International Conference on Very Large Data Bases. New York：ACM，2008：1008-1019

[6] 叶育鑫. 混合语义约简和选择估值优化 SPARQ[J]. 电子学报，2010，38(5)：1205-1210

[7] Hatting O. Querying trust in RDF Data with tSPARQL[C]// Proceedings of 6th European Semantic Web Conference (ESWC'09). Berlin：Springer，2009：5-20

[8] Stocker M，Seaborne A，et al. SPARQL basic graph pattern Optimization using selectivity estimation[C]// Liu Y H. www' 08. New York：ACM，2008：595-604

[9] Guo Y，Pan Z，Heflin J. LUBM：A benchmark for OWL knowledge base systems[J]. Web Semantics：Science，Services and Agents on the World Wide Web，2005，3(2)：158-182

[10] Huang H，Liu C. Selectivity estimation for SPARQL graph pattern[C]// Proceedings of the 19th International Conference on World Wide Web. ACM，2010：1115-1116

[11] Papailiou N，Konstantinou I，Tsoumakos D，et al. H2RDF：adaptive query processing on RDF data in the cloud[C]// Proceedings of the 21st International Conference Companion on World Wide Web. ACM，2012：397-400

[12] Abraham J，Brazier P，Chebotko A，et al. Distributed storage and querying techniques for a Semantic Web of scientific workflow provenance[C]// Services Computing (SCC)，2010 IEEE International Conference on. IEEE，2010：178-185

[13] McBride B. Jena：A Semantic Web Toolkit [J]. IEEE Internet Computing，2002，6(6)：55-9