基于树型结构的 MapReduce 并行模型

唐 兵[1] 贺海武[2]
(湖南科技大学计算机科学与工程学院 湘潭 411201)[1] (中国科学院计算机网络信息中心 北京 100190)[2]

摘 要 MapReduce是Google提出的一种分布式计算模型,已在海量数据处理领域得到了广泛的应用。提出一种基于树型结构的新型MapReduce并行模型。该模型适合于利用Internet或Intranet环境下不可靠的桌面PC资源进行海量科学数据分析。该模型以P2P的形式将计算节点进行组织,模型的底层采用了P2P-MPI框架,采用基于消息传递的模式来实现MapReduce应用层。在MapReduce应用层的实现中,在Map阶段采用广播的形式来分发数据块,在Reduce阶段建立反向二叉树来实现有效的结果合并和化简。将提出的MapReduce模型与现有主流MapReduce模型进行了比较,结果表明,基于树型结构的MapReduce并行模型在容错性能方面具有较优的性能,且系统简单,易于应用开发。

关键词 MapReduce,树型结构,二叉树,消息传递接口
中图法分类号 TP302 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.11.013

MapReduce Parallel Model Based on Tree Structure

TANG Bing[1] HE Hai-wu[2]
(School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China)[1]
(Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract MapReduce is a distributed computing model introduced by Google, which has been widely used in the field of massive data processing. A novel MapReduce parallel model was presented in this paper. The model is suitable for massive scientific data analysis, using unreliable desktop PC resources in the Internet or Intranet environment. Computing nodes are organized in the form of P2P, and the P2P-MPI framework is utilized in the lower layer, while message passing interface model is utilized to achieve the MapReduce application layer. In the implementation of MapReduce application layer, the way of broadcast is used to distribute data chunks in the Map stage, and an inverse binary tree is constructed to realize effective intermediate results reduction in the Reduce stage. The proposed MapReduce mode was compared with existing popular MapReduce modes. The results show that the proposed tree structure-based MapReduce parallel model has a good performance in terms of fault-tolerance and it is simple and easy for application development.

Keywords MapReduce, Tree structure, Binary tree, Message passing interface(MPI)

1 引言

作为云计算的核心技术,近年来MapReduce也受到了人们的广泛关注。MapReduce是Google在2004年提出来的一种处理大规模数据集的并行编程模型,用来简化分布式系统的编程[1]。应用程序编写人员只需将精力放在应用程序本身,而关于集群的处理问题,包括可靠性、可扩展性、任务并行化、数据分布存储、负载均衡、节点通信等复杂过程被屏蔽,用户不需要关心如何将输入的数据分块、分配和调度。这使得那些没有多少并行计算经验的程序员也可以开发并行应用来处理和分析海量数据。

随着志愿计算项目SETI@home的流行,人们逐渐关注利用桌面PC这种几乎免费的资源来进行科学计算或数据分析[2-4]。这种方式为解决基础科学运算规模较大、计算资源需求较多的难题提供了一种行之有效的途径。利用大规模的桌面PC构建高性能计算系统已成为可能。但桌面PC容易产生故障,给计算系统带来了挑战。近年来,在桌面PC这种不可靠资源上运行MapReduce应用成为了一个研究热点,人们纷纷提出了一些系统和实现,如MOON[5]、P2P-MapReduce[6]、VMR[7]、BitDew-MapReduce[8,9]等。面对桌面PC的动态易失效性,需要解决容错问题,输入数据、中间结果集、最终结果的存储问题,以及Map/Reduce任务调度问题。这些系统普遍采用了副本容错、调度算法改进、通信方式改进等策略来克服桌面PC故障失效所带来的影响,能够可靠地运行MapReduce应用。

在传统的MapReduce模型中,少数计算节点固定充当Reducer节点,具有相同key的中间结果集会聚集到这些Reducer上。在大规模桌面PC的环境下,计算节点数目在数以

到稿日期:2014-11-19 返修日期:2015-01-21 本文受法国国家科研署科研项目(ANR-10-SEGI-001-01),中科院百人计划(1101002001),湖南省自然科学基金(2015JJ3071),湖南省教育厅一般项目(12C0121)资助。

唐 兵(1982—),男,博士,讲师,主要研究领域为并行处理、云计算、大数据;贺海武(1977—),男,博士,研究员,主要研究领域为并行处理、云计算、大数据。

万计的情况下，如果能够有更多的节点参与 Reduce，将会提高 Reduce 阶段的效率。针对此问题，本文提出一种基于树型结构的 MapReduce 并行模型。在 Reduce 阶段，以反向二叉树的形式进行结果的合并和化简，有更多的节点可参与 Reduce，节点间两两合并，最终汇总至 Master。该模型基于 P2P-MPI 框架，更适合于大规模桌面 PC 环境。

2 相关背景

2.1 MapReduce

MapReduce 通过 Map(映射)和 Reduce(化简)这样两个简单的概念来构建运算基本单元。用户只需编写 Map 函数和 Reduce 函数即可实现对大规模海量数据集的并行处理。在 Map 函数中指定对各分块数据的处理过程，在 Reduce 函数中指定如何对分块数据处理的中间结果进行化简。映射-化简过程如下。

映射(Map)过程：$Map(key1, value1) \rightarrow list(key2, value2)$

化简(Reduce)过程：$Reduce(key2, list(value2)) \rightarrow list(value3)$

此外，在 Reduce 过程之前，一般还包括排序(Sort)过程和合并(Merge)过程。

当前流行的 Hadoop 是一种基于 Java 的 MapReduce 实现，利用 Hadoop 进行海量数据处理程序设计，用户不需要关心如何将输入的数据分块、分配和调度，同时系统还将处理机群内节点失败以及节点间通信的管理等。在现有技术中，MapReduce 系统通常和分布式文件系统相耦合，如 Hadoop 实现了一个分布式文件系统 HDFS (Hadoop Distributed File System)和 MapReduce 任务调度框架，大数据经过分片存储在由工作节点所组成的分布式文件系统中。MapReduce 节点间的通信如图 1 所示。

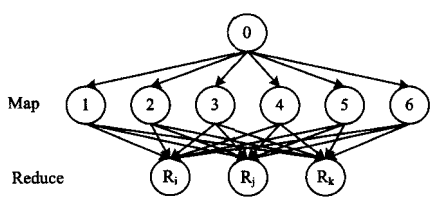

图 1 MapReduce 节点间通信

2.2 P2P-MPI

当面向非共享存储系统开发并行程序时，程序的各部分之间通过来回传递消息的方式通信。MPI 是一种被广泛采用的消息传递标准。目前主流的 MPI 实现主要采用了 C、C++、Fortran 编程语言，如 MPICH、OpenMPI、LAM-MPI 等。P2P-MPI 是法国斯特拉斯堡大学的 Genaud 等实现的一种基于 P2P 的框架，在该框架中能够可靠地运行 MPI 程序[10,11]。与 mpiJava 类似，P2P-MPI 也是采用 Java 语言的一种 MPI 实现[12]，它的最主要的特点是采用了基于 JXTA 的 P2P 通信协议，实现了节点的容错。用户可以自定义副本数来实现容错。P2P-MPI 框架的分层结构如图 2 所示，核心的部分是：错误检测服务(FD)、文件传输服务(FT)、消息传递服务(MDP)。P2P-MPI 提供了 MPI 标准的 API，可执行并行的 Java 程序，它为用户屏蔽了后台细节。用户如果开发过 C/C++的 MPI 程序，那么可以很容易地编写适应于 P2P-MPI 的 MPI 程序。

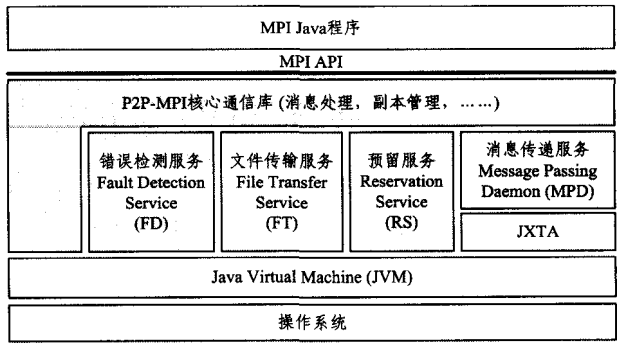

图 2 P2P-MPI 框架分层结构

在 P2P-MPI 框架中，错误检测器采用的是基于 Gossip 协议的方式[11]。错误检测器是分散至每一个计算节点的。每个错误检测器都维护着一个表格，表格中存储的每一项是一个二元组〈已知的错误检测器、心跳计数器〉。从表格中随机选择一个错误检测器，更新所对应的心跳计数器(值加 1)，然后将表格发送给所选择的错误检测器所在的计算节点。接收方收到后，与自身存储的表格的每一项进行融合，每个错误检测器所对应的心跳计数值取最大值。如果检测到在规定的时间内某心跳计数值没有增加，则对应的计算节点出现故障。本地的 MPI 程序从错误检测服务(FD)接收到了节点故障消息，将该节点标记为故障。

3 基于树型结构的 MapReduce 模型

与 Hadoop 的工作原理不同，在所设计的树型结构的 MapReduce 模型中，不存在分布式文件系统，采用 MPI 程序实现 MapReduce 应用，所有的 Slave 进程从 Master 进程接收数据。设计的总体目标是：1)尽量使节点不处于空闲状态而都执行一定的工作，实现负载均衡；2)程序尽可能简单。模型总体的思想是：以广播形式分发数据块，以反向二叉树形式进行结果化简。

首先，用户通过 Master 提交作业。Master 进行任务调度，在 Map 阶段采用广播的形式，Master 向 Slave 分发数据块。Slave 收到数据块后，执行 Map 操作，并进行 Sort 操作和 Merge 操作，然后进行 Reduce 操作。按照一定的规律，会产生一个反向二叉树，反向二叉树的生成主要依据的是进程标识号 rank 的奇偶性，如图 3 所示，Reduce 阶段为一个反向二叉树。按照这个反向二叉树进一步进行结果集的化简，基本方法是：Slave 将结果数据发送至另一个 Slave，数据被接收后进行进一步的 Sort 操作、Merge 操作、Reduce 操作。最后，最终结果汇总至 Master。简单来说，即两个进程合并为一个进程，最终结果合并化简汇总至进程 0，即 Master。与图 3 相对应，图 4 给出了 6 个进程运行时的甘特图。

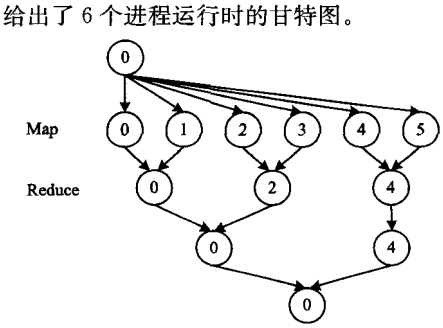

图 3 树型结构 MapReduce 模型中节点通信

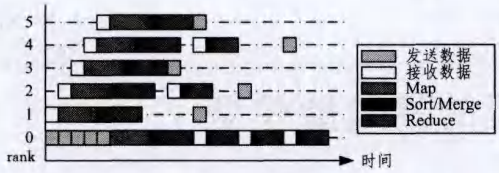

图 4　树型结构 MapReduce 模型的任务执行甘特图

在描述结果返回阶段的算法之前，先介绍以下定义。

1）rank：进程的标识号；

2）branch：循环运行次数的计算器；

3）processes：总的进程数；

4）data：Slave 存储的 key/value 对的集合；

5）receiveData()：接收数据函数；

6）merge(para1，para2)：将数据 para1 和数据 para2 进行合并的函数；

7）reduce(para1)：对数据 para1 进行化简的函数；

8）wait()：等待函数；

9）passto(para1，para2)：将数据 para1 传输至进程 para2 的函数。

算法　树型结构 MapReduce 中结果返回阶段算法

```
1. id←rank of process
2. branch←0
3. max ← processes−1
4. //如果是偶数，则进入循环接收数据；如果是奇数，则跳过循环
5. while id%2==0
6.     //左移后如果比 max 小，则接收并化简数据
7.     if id≪branch < max
8.         data←merge(data,receiveData( ))
9.         reduce(data)
10.        max←max−1
11.    else
12.        //左移后如果比 max 大，此 Slave 进程需要等待
13.        //其他进程则不受影响，继续各自发送数据和化简数据
14.        wait( )
15.    endif
16.    id ← id≫1
17.    branch ← branch+1
18. endwhile
19. //结果发送至另一个进程(id−1)≪branch
20. passto(data,(id−1)≪branch)
```

在图 3 中，进程 0 将数据进行广播，其他进程根据不同的文件偏移量和文件长度，获取不同的数据块，然后各进程启动 Map 操作，计算完毕后按照上述算法进行结果的合并和化简。下面结合图 3 和上述算法进行实例介绍。

以进程 3 为例，rank=3，processes=6，id=3，branch=0，不进入循环，(id−1)≪branch=2，直接发送数据至进程 2。

以进程 2 为例，rank=2，processes=6，id=2，max=5，在第 1 轮循环中，接收来自于进程 3 的数据并对数据进行合并和化简之后，max=4，id=1，branch=1，此时退出循环，并且有(id−1)≪branch=0，发送数据至进程 0。

以进程 4 为例，rank=4，processes=6，id=4，max=5，在第 1 轮循环中，接收来自于进程 5 的数据并对数据进行合并和化简之后，max=4，id=2，branch=1。在第 2 轮循环中，由于 id≪branch=4，此时不满足“比 max 小”这个条件，不接收数据，进程 4 必须进行等待。进程 4 等到进程 0 空闲时，id=1，branch=2，不再进入第 3 轮循环，(id−1)≪branch=0，因此进程 4 发送数据至进程 0。

从图 4 中可以看出，如果每个计算节点启动一个进程，则每个节点（包括 rank=0 所在的 Master）都参与了 Map、Sort、Merge、Reduce 4 种计算。在每个节点上，Map 之后还进行了 Sort、Merge、Reduce，然后才把数据发到下一节点。下一节点经过 Sort、Merge、Reduce 之后，再继续传递给下一节点，以此类推。与传统的基于 Hadoop 的 MapReduce 模型相比，本文提出的这种模型能够使得更多的节点参与到 Reduce 中来，提高了结果合并化简的效率。

4　性能测试及分析

为了测试所提出的基于树型结构的 MapReduce 模型，验证反二叉树方法，同时为了清晰地了解节点间的通信及数据传送的过程，将采用 Java 编程语言所实现的原型系统（称为 Tree-MapReduce）与现有的两种 MapReduce 模型进行了性能比较：1）开源的 Hadoop；2）基于数据驱动模式的 BitDew-Map-Reduce[8,9]。BitDew-MapReduce 是我们前期在 BitDew 中间件的基础上开发的一种专门适合于桌面网格环境的 Map-Reduce系统[13]，它充分利用了{*lifetime*，*affinity*，*replica*，*fault-tolerance*，*transfer-protocol*，*distrib*}六元组的数据属性值来灵活地控制 MapReduce 应用中所涉及的各种数据。在数据分发方面，BitDew-MapReduce 采用的是基于 Pull 模式的数据块调度，Tree-MapReduce 采用的是 MPI 并行读取模式，而 Hadoop 中的 DataNode 采用了 pipeline 复制，数据最终存储在 HDFS 中。

对上述 3 种模型即 Tree-MapReduce、Hadoop、BitDew-MapReduce进行性能测试的实验环境配置如下：利用学生计算机室中的桌面 PC，构建一个桌面网格计算环境，每台的硬件条件为 Intel Core 2 Duo 1.86GHz，内存为 1GB，网络连接为 100Mbps。测试的应用为 WordCount（词频统计程序），文本文档大小为 5GB。实验中记录下任务完成时间，比较了随着工作节点数目增大的可扩展性能以及容错性能。其中 1 台桌面 PC 充当 Master，其余为工作节点。

首先是运行环境的比较，Tree-MapReduce 和 BitDew-MapReduce 均可运行在 Internet 环境和 Windows 平台，而 Hadoop 却不能运行于 Windows 平台。在后面的性能比较中，桌面 PC 的操作系统均为 Ubuntu Linux 12.04。Hadoop 采用的是 hadoop-0.20.0 版本。在 3 种模型中，输入数据的副本数均设置为 3，数据分块大小均设置为 64MB。

然后，比较工作节点数目由 8 变化至 40 的作业完成时间开销，结果如图 5 所示，随着节点数目增多，时间开销减小。在 Hadoop 和 BitDew-MapReduce 中，Reduce 节点压力过大，而 Tree-MapReduce 将结果合并的压力分散到更多的节点来执行，效率比 BitDew-MapReduce 高，与 Hadoop 接近。

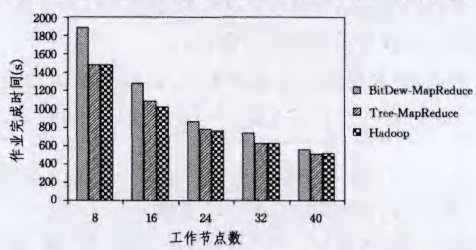

图 5　随着工作节点数增加的作业完成时间开销

（下转第 89 页）

toring[C]//Proceedings of the 2013 International Conference on Software Engineering. San Francisco, USA, 2013:1287-1290

[12] Schafer M, Sridharan M, Dolby J, et al. Refactoring java programs for flexible locking[C]//Proceedings of 33rd International Conference on Software Engineering (ICSE). Hawaii, USA, 2011:71-80

[13] Brown C, Hammond K, Danelutto M, et al. Paraphrasing: Generating Parallel Programs Using Refactoring[J]. Formal Methods for Components and Objects, 2013, 10(8):237-256

[14] Zhang C. FlexSync: An aspect-oriented approach to Java synchronization[C]//Proceedings of the 31st International Conference on Software Engineering. Vancouver, Canada, 2009: 375-385

[15] McCloskey B, Zhou F, Gay D, et al. Autolocker: synchronization inference for atomic sections[J]. ACM SIGPLAN Notices, 2006, 23(2):346-358

[16] 赵思奇. 面向并发的程序重构技术研究[D]. 南京:东南大学, 2010
Zhao Si-qi. A novel approach of concurrency-oriented software refactoring[D]. Nanjing: Southeast University, 2010

[17] 吕建,杨大军,廖宇,等. 一种并发面向对象同步模型研究[J]. 软件学报, 2002, 13(1):71-80
Lv Jian, Yang Da-jun, Liao Yu, et al. Research on a Concurrent Object-Oriented Synchronization Model[J]. Journal of Software, 2002, 13(1):71-80

[18] 陶彬贤,张磊,钱巨. Java 程序自动锁分解重构[J]. 计算机科学与探索, 2013, 7(5):451-459
Tao Xian-bin, Zhang Lei, Qian Ju. Automated Split Lock Refactoring for Java Programs[J]. Journal of Frontiers of Computer Science and Technology, 2013, 7(5):451-459

(上接第 67 页)

最后,手动关闭工作节点的进程来引入节点失效的方式,以此来测试容错性能。BitDew-MapReduce 和 Tree-MapReduce 中设置的节点故障超时时间均为 30s。最开始采用了 40 个工作节点,每隔 10s 产生一个节点失效,失效的节点数分别为 5、10、15、20。在节点失效的情况下,作业完成时间的比较结果如图 6 所示。从图 6 中可看出,在 20 个节点失效的情况下,采用 Tree-MapReduce 比采用 Hadoop 时间开销减少了约 17.9%。

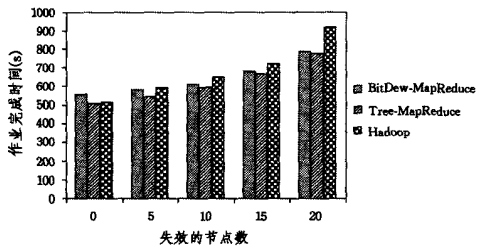

图 6 随着失效节点数增加的作业完成时间开销

结束语 本文提出一种基于树型结构的新型 MapReduce 并行模型。该模型以 P2P 的形式将大规模的桌面 PC 节点组织起来,模型的底层采用了 P2P-MPI 框架,采用基于消息传递的模式来实现 MapReduce 应用层。在 MapReduce 应用层的实现中,在 Map 阶段采用广播的形式来分发数据块,在 Reduce 阶段建立反向二叉树来实现有效的结果合并和化简。这种基于树型结构的 MapReduce 模型充分利用了 MPI 和 MapReduce 各自的优点,具有较好的可扩展性。在 P2P-MPI 框架的支撑上,由于克服了 MPI 的缺点,使得所提出的 MapReduce 模型具有较好的容错性能。性能比较的结果表明,基于树型结构的 MapReduce 并行模型在容错性能方面具有较优的性能,且系统简单,易于应用开发。该模型适合于利用 Internet 或 Intranet 环境下不可靠的桌面 PC 资源进行海量科学数据分析。

参 考 文 献

[1] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters[J]. Communications of the ACM, 2008, 51(1): 107-113

[2] Anderson D P. BOINC: A System for Public-Resource Computing and Storage[C]//Proc. of the 5th International Workshop on Grid Computing (GRID 2004). 2004:4-10

[3] Cappello F, Djilali S, Fedak G, et al. Computing on Large-scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid[J]. Future Generation Computer Systems, 2005, 21(3):417-437

[4] Litzkow M J, Livny M, Mutka M W. Condor-A Hunter of Idle Workstations[C]//Proc. of the 8th International Conference on Distributed Computing Systems (ICDCS 1988). 1988:104-111

[5] Lin H, Ma X, Feng W. Reliable MapReduce Computing on Opportunistic Resources[J]. Cluster Computing, 2012, 15(2):145-161

[6] Marozzo F, Talia D, Trunfio P. P2P-Mapreduce: parallel data processing in dynamic cloud environments[J]. Journal of Computer and System Sciences, 2012, 78(5):1382-1402

[7] Costa F, Silva J N, Veiga L, et al. Large-scale volunteer computing over the Internet[J]. Journal of Internet Services and Applications, 2012, 3(3):329- 346

[8] Tang B, Moca M, Chevalier S, et al. Towards mapreduce for desktop grid computing[C]//Proc. of the 5th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC 2010). 2010:193-200

[9] Lu L, Jin H, Shi X, et al. Assessing mapreduce for Internet computing: a comparison of Hadoop and BitDew-MapReduce[C]//Proc. of the 13th ACM/IEEE International Conference on Grid Computing (GRID 2012). 2012:76-84

[10] Genaud S, Rattanapoka C. P2P-MPI: A Peer-to-Peer Framework for Robust Execution of Message Passing Parallel Programs on Grids[J]. Journal of Grid Computing, 2009, 5(1):27-42

[11] Genaud S, Rattanapoka C. A Peer-to-Peer Framework for Message Passing Parallel Programs[M]//Xhafa F, eds. Parallel Programming, Models and Applications in Grid and P2P Systems, Advances in Parallel Computing. IOS Press, 2009:118-147

[12] Carpenter B, Getov V, Judd G, et al. MPJ: MPI-like message passing for Java[J]. Concurrency-Practice and Experience (CONCURRENCY), 2000, 12(11):1019-1038

[13] Fedak G, He H, Cappello F. BitDew: A Data Management and Distribution Service with Multi-protocol File Transfer and Metadata Abstraction[J]. Journal of Network and Computer Applications, 2009, 32(5):961-975