软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(8):2056−2073 [doi: 10.13328/j.cnki.jos.004807]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

MapReduce 集群环境下的数据放置策略*

荀亚玲[1], 张继福[1], 秦 啸[2]

[1](太原科技大学 计算机科学与技术学院,山西 太原 030024)

[2](Department of Computer Science and Software Engineering, Auburn University, USA)

通讯作者: 荀亚玲, E-mail: xunyl55@126.com, http://www.tyust.edu.cn

摘 要: MapReduce 是一种适用于大规模数据密集型应用的有效编程模型,具有编程简单、易于扩展、容错性好等特点,已在并行和分布式计算领域得到了广泛且成功的应用.由于 MapReduce 将计算扩展到大规模的机器集群上,处理数据的合理放置成为影响 MapReduce 集群系统性能(包括能耗、资源利用率、通信和 I/O 代价、响应时间、系统的可靠性和吞吐率等)的关键因素之一.首先,对 MapReduce 编程模型的典型实现——Hadoop 缺省的数据放置策略进行分析,并进一步讨论了 MapReduce 框架下,设计数据放置策略时需考虑的关键问题和衡量数据放置策略的标准;其次,对目前 MapReduce 集群环境下的数据放置策略优化方法的研究与进展进行了综述和分析;最后,分析和归纳了 MapReduce 集群环境下数据放置策略的下一步研究工作.

关键词: 数据放置;MapReduce;编程模型;能耗;负载均衡

中图法分类号: TP316

中文引用格式: 荀亚玲,张继福,秦啸.MapReduce 集群环境下的数据放置策略.软件学报,2015,26(8):2056−2073. http://www.jos.org.cn/1000-9825/4807.htm

英文引用格式: Xun YL, Zhang JF, Qin X. Data placement strategy for MapReduce cluster environment. Ruan Jian Xue Bao/ Journal of Software, 2015,26(8):2056−2073 (in Chinese). http://www.jos.org.cn/1000-9825/4807.htm

Data Placement Strategy for MapReduce Cluster Environment

XUN Ya-Ling[1], ZHANG Ji-Fu[1], QIN Xiao[2]

[1](School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China)

[2](Department of Computer Science and Software Engineering, Auburn University, USA)

Abstract: As an effective programming model for large-scale data-intensive applications, MapReduce has been widely and successfully applied in the field of parallel and distributed computing, and has the characteristics of good fault-tolerance and easy to implement and extend. Because MapReduce extends computing to the nodes of large-scale cluster system, reasonable placement of processing data has become one of the key factors affecting the performance of MapReduce cluster system, including energy efficiency, resource utilization, communications and I/O throughput, response time, and reliability. This study first analyzes characteristics of the default data placement strategy of Hadoop, which is a typical implementation of MapReduce programming model. Next, it investigates popular data placement strategies for MapReduce cluster computing environments. Finally, it presents future research directions in the area of data placement strategies for MapReduce-based cluster computing systems.

Key words: data placement; MapReduce; programming model; energy-efficient; load balancing

随着社交网络、云计算和物联网等技术的发展以及博客、社交网络等新型信息发布方式的不断涌现,产生了海量数据.大数据处理的需求正在迅速增加,典型的数据密集型应用,如数据挖掘和 Web 索引等在工业界和学

* 基金项目: 国家自然科学基金(61272263); NSF CAREER Award(CCF-0845257)
收稿时间: 2014-04-08; 修改时间: 2014-05-22, 2014-12-09; 定稿时间: 2014-12-21; jos 在线出版时间: 2015-02-02
CNKI 网络优先出版: 2015-02-02 15:17, http://www.cnki.net/kcms/detail/11.2560.TP.20150202.1517.003.html

术界正变得越来越重要,大数据研究成果直接影响了经济和社会的发展[1].因此,寻求一种有效的适合于大数据分析和数据密集型应用的数据处理模型,显得尤为重要.

MapReduce[2,3]是一个很有前途的支持数据密集型应用和科学分析的简单可行和可扩展的并行计算编程模型.该模型将简单的业务处理逻辑从复杂的实现细节中提取出来,使用户和程序员不需要关心如何将输入的数据分块、分配和调度,以及处理集群内节点失败及节点间通信等,其后台复杂的并行执行和任务调度对用户和编程人员是透明的,使得即使没有任何并行和分布式系统经验的程序员也能比较容易地使用一个大的分布式系统提供的计算资源,同时,还可使由普通 PC 组成的巨大集群达到极高的计算性能.MapReduce 程序的执行需要分布式文件系统的支持,因此,MapReduce 框架需要提供一套机制,将计算扩展到大规模的机器集群上进行,分布式文件系统正为适应这种扩展奠定了基础.通常,MapReduce 框架和分布式文件系统(在 Hadoop 中实现为 HDFS)是运行在一组相同的节点上的,也就是说,计算节点和存储节点通常在一起.这种配置允许框架在那些已经存好数据的节点上高效地调度任务,从而使整个集群的网络带宽被非常高效地利用.目前,分布式文件系统大多都采用基于复制的容错技术[4,5].基于复制的容错技术对一个数据对象创建多个相同的数据副本,并把多个副本散布到不同的存储节点上.当若干数据对象失效后,可以通过访问其他有效的副本获取数据.基于复制的容错技术主要关注的问题是:

(1) 数据组织结构,即,研究大量数据对象及其副本的管理方式;

(2) 数据复制策略,即,主要研究副本的创建时机、副本的数量、副本的放置等问题[6].

数据放置策略最基本的目的在于提高数据的容错性,使得用户在部分副本失效以后仍然能够通过其他副本获得数据[6].但创建的副本传输到放置节点上,需要占用大量带宽,消耗很长时间.因此,良好的放置策略不但要考虑容错性,更要考虑复制效率和网络带宽等.这种额外的副本存储为达到某个预定目标而探索不同的数据放置策略,提供了较强的灵活性.换句话说,对于一个给定的系统度量标准,可以通过采用不同的数据放置策略来优化系统性能.更广义的数据放置,应该涉及 MapReduce 整个执行过程中的数据放置问题,不仅包括原始数据如何被划分、复制和放置,即,在 map 端数据如何放置的问题,而且还应包括 map 的输出数据如何在 reduce 端进行放置的问题.

面对日益复杂的数据密集型应用和大数据处理需求,数据放置策略遇到了新的挑战[7],主要表现在如何保持数据间的依赖性以减少跨数据中心的数据传输、如何在提高效率的同时兼顾全局的负载均衡,以及如何在提高系统的可靠性和吞吐率的同时节约能耗等.本文首先对 MapReduce 框架下的数据数放置策略进行深入的分析,并探讨了在设计数据放置策略时所需考虑的关键问题和衡量数据放置策略的标准,然后归纳总结了目前一些主要的数据放置策略优化方法,阐述了 MapReduce 集群下数据放置策略的一系列开放性问题和挑战,并在此基础上给出了其进一步深入研究的方向.

1 MapReduce 编程模型与数据放置策略

1.1 MapReduce编程模型

MapReduce 通过 map 和 reduce 这两个步骤来并行处理大规模的数据集,为用户和编程人员提供了一个简单而又强大的接口.通过该接口,可以把大尺度的计算自动地并发和分布执行.其设计思想源自于函数式程序设计语言,将一个大型的分布式计算表达成一系列 key/value 对的并行操作序列.用户首先创建一个 map 函数,处理一个基于 key/value 对的输入数据的每个逻辑块,并产生一组中间 key/value 对;然后,reduce 对该列表中的元素根据 key 值进行适当的合并操作,最终输出结果.其实现过程涉及到很多复杂的细节问题,图 1 描述了 MapReduce 下的并行计算过程.

从图 1 可以看到:一个 MapReduce 作业(job)通常会把输入的数据集切分为若干个独立的数据块,并由 map 任务(map tasks)以完全并行的方式处理;随后,map 的输出被分区排序后复制给相应的 reduce 任务(reduce tasks)并执行.通常,作业的输入和输出都会被存储在文件系统中.整个框架负责任务的调度和监控及失败任务的重新调度.MapReduce 框架由一个单独的 master jobtracker 和每个集群节点的一个 slave tasktracker 组成.jobtracker

是一个在 Hadoop 中用来协调作业执行的应用程序,tasktracker 是运行作业划分后的任务,jobtracker 和 tasktracker 之间通过 tasktracker 运行简单循环定期产生的“心跳”进行通信.

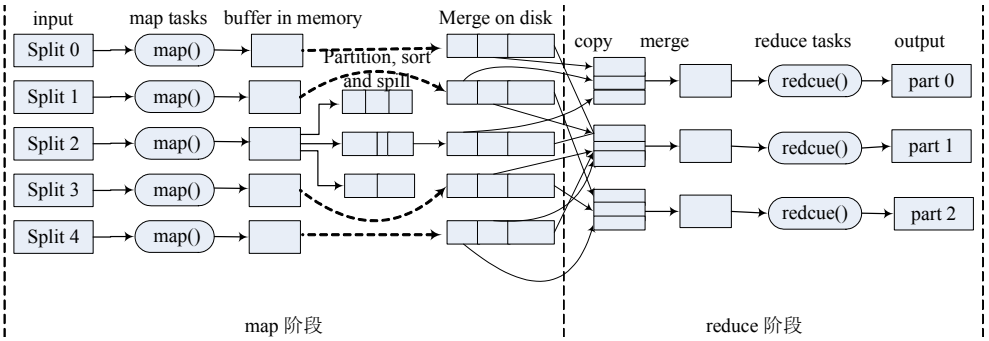

Fig.1 Parallel computing process of MapReduce

图 1 MapReduce 并行计算过程

在上述 MapReduce 并行计算过程中,采用分布式文件系统对其中数据进行管理.大多数分布式文件系统采用元数据集中管理、数据块分散存储的模式,通过数据的备份实现高度容错[6].分布式文件系统在数据建立阶段,首先需要将输入文件划分成数据片段,并按照某种数据放置策略进行复制和存放;在运行阶段,需要对新产生的数据进行管理.研究表明:有效的数据放置是影响 MapReduce 程序性能的一个关键因素,对减少跨节点和数据中心的数据传输量、提高程序流执行效率和资源利用率和降低能耗等方面具有重要意义.

Hadoop 是典型的 MapReduce 框架开源实现,实现了 MapReduce 分布式编程模型和分布式文件系统 HDFS[8].Hadoop 作为新的分布式存储与计算架构,由于能够部署在通用平台上,并且具有可扩展性、低成本、高效性与可靠性等优点,使其在分布式计算领域得到了广泛运用,并且已逐渐成为工业界与学术界事实上的海量数据并行处理标准,已经被许多的机构和公司采用.在下节中,将以 Hadoop 为例介绍其数据放置策略.

1.2 Hadoop默认数据放置策略

Hadoop 的核心设计思想就是“移动计算能力的代价要小于移动数据的代价”.Hadoop 在每个计算节点(datanode)的本地存储数据上进行计算,再将计算的结果进行归并,因此,Hadoop 与之前大多数的分布式系统最大的不同就是“不共享数据”.整个过程就是一个 MapReduce 任务,其中的 map 代表着分解,reduce 代表着归并.在其底层,Hadoop 参考 Google 的分布式文件系统 GFS(Google file system)[4],实现了 Hadoop 分布式文件系统 HDFS[5]来管理数据.HDFS 采用了主从模式,即,一个 namenode(管理者)和多个 datanode(工作者):namenode 负责管理文件的命名空间,维护文件系统内所有的文件和目录;datanode 是文件系统的工作节点,存储其本地文件系统,根据客户端或者是 namenode 的调度存储和检索数据,并且定期向 namenode 发送他们所存储的块列表.HDFS 提供了一个冗余的存储方案,即:HDFS 对输入文件进行分块存储,并在多个节点上存储每个数据块的多个副本,以保持数据的可靠性和容错性.

如图 2 所示,在当前 Hadoop 的 HDFS 系统中,每个文件被分成默认大小为 64MB 的数据块,系统默认为每一个数据块存放 3 个副本,按照部署在 namenode 上的默认机架感知策略存放数据块副本[9].namenode 如何选择在哪个 datanode 存储副本,需要在可靠性、写入带宽和读取带宽之间进行权衡.其基本存储策略是:在运行客户端的节点上放第 1 个副本(如果客户端运行在集群之外,就随机选择一个节点,但是,系统会避免挑选那些存储太满或太忙的节点);第 2 个副本放在与第 1 个不同且随机的另外一个机架上的节点上;第 3 个副本放在与第 2 个副本相同的机架上,且随机选择一个节点.如果还有更多副本,则在整个集群中随机选取节点存放.总的来说,该副本放置策略通过将数据块存储在两个机架上,为集群系统提供了很好的稳定性,同时实现了很好的负载均衡,包括写入带宽(仅需要遍历一个交换机)、读取性能(可以在两个机架中进行选择读取)和集群中块的均匀分布

(只在本地机架写入一个块).但是,当整个本地节点都失效时,HDFS 将自动地通过远端机架上的数据副本,使数据副本的数量恢复到标准数量,将导致系统在保证数据可用性的同时,需要占用大量带宽,消耗很长时间.

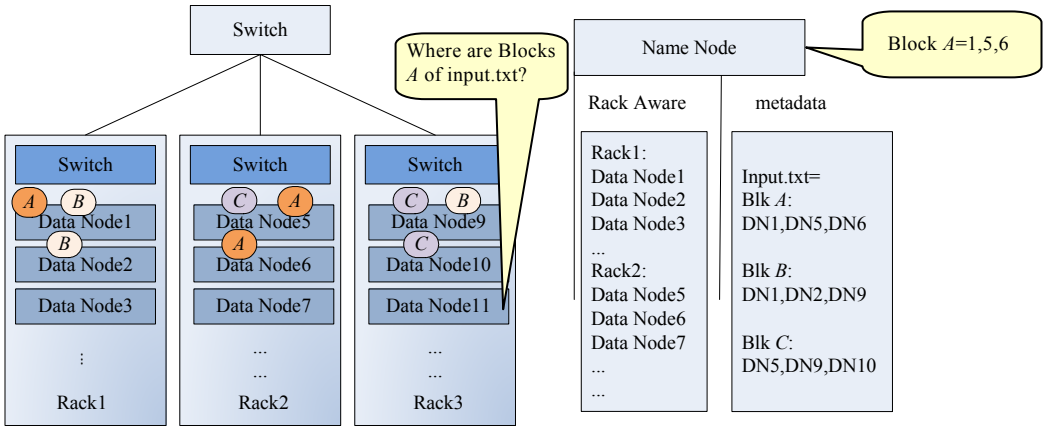

Fig.2 Data placement strategy of Hadoop

图 2 Hadoop 数据放置策略

HDFS 数据管理方式为 MapReduce 模型将作业分解成任务执行奠定了基础.对于一个 map 任务,jobtracker 会考虑 tasktracker 的数据本地性,在最理想的情况下,任务是数据本地化的,也就是任务运行在输入分片所在的节点上.对于 reduce 任务,jobtracker 只是简单地从待运行的 reduce 任务列表中选取一个来执行,而没有考虑数据的本地化.MapReduce 模型还提供了“推测执行策略”,以解决少数“拖后腿”的任务延缓作业的整体执行时间的问题,在作业执行过程中,Hadoop 不会诊断或修复慢的任务,相反,它会启动一个相同的任务作为备份.在该过程中,需要考虑“推测执行”任务所对应数据的复制.

1.3 数据放置的关键问题与度量标准

在集群环境下的传统并行处理模式(例如 MPI)中,数据存储节点和数据处理节点往往不是同一节点.在每次执行计算任务时,首先需要从数据存储节点读取并进行相应的数据划分后,被传输到相应的计算节点上,然后进行数据处理,即,数据存储和数据处理是分离的.对于海量的大数据分析,这种“数据向计算迁移”的数据处理方式将导致大量的数据在节点间进行交换,网络带宽和通信时间将成为影响系统性能的重要因素,系统性能会大幅度降低,且数据的划分、节点失效、数据可用性、集群的可扩展性等问题的处理都需要用户的参与,也极大地增加了用户编程的复杂性.因此,人们开始研究适合大数据处理的文件系统及其数据放置策略.

分布式文件系统将文件划分为块大小的多个分块(chunk),作为独立的存储单元.其主要原因在于:

(1) 一个文件的大小可以大于网络中任意一个磁盘的容量,即,一个文件可以利用集群上的任意一个磁盘进行存储;

(2) 使用块而非整个文件作为存储单元,极大地简化了存储子系统的设计;

(3) 块非常适合于数据备份进而提供数据容错能力和可用性.

而且每个块被复制,在多个节点上存储每个数据块的多个副本,确保即使在节点或者甚至是完整的机架连接失败的情况下数据的可用性.因此,考虑数据如何在各个节点上的放置问题时主要涉及到以下几个方面:

(1) 文件该如何分片;

(2) 每个分片什么时间创建副本以及该生成多少副本;

(3) 所有分片的副本如何分发到各个存储节点上;

(4) 完成数据的有效放置需要维护哪些信息;

(5) 根据运行时不同状态如何调整和维护分片.

Hadoop 采用了静态复制策略,即,在数据进入时就创建指定数目的副本.但静态复制策略却不能依据环境的变化做出动态的调整,容易造成资源浪费.因此,研究人员提出了动态复制策略,可以依据网络状况、存储空间和用户需求等动态地创建或者删除副本:在存储空间紧张时,删除部分副本以节省存储空间;当存储资源丰富时,为频繁访问的数据增加副本以提高效率,均衡节点负载.动态放置策略在动态创建或者迁移副本时需要执行大量额外的操作,特别是频繁的数据传输会带来巨大的网络开销.另外,Hadoop 默认的放置策略为了提高数据的容错性,使得部分副本失效后仍然能够通过其他的副本获得数据,而副本传输到放置节点上,需要占用大量带宽,消耗很长时间.因此,良好的放置策略不但要考虑容错性,更要考虑复制效率和传输代价等.

目前,衡量数据放置策略好坏的标准一般有以下几个方面:

(1) 数据本地化程度:无论是在 map 阶段还是在 Reduce 阶段,好的数据本地化可以减少数据传输造成的时间浪费,也极大地减少了 I/O 和通信开销.

(2) 可靠性的恢复开销:当系统中的某些节点出现故障时,通常需要执行一些操作来调整处于正常工作状态的节点上的数据分布,使得系统重新达到故障前的可靠性.一个好的数据放置策略可以尽可能地减少在恢复过程中的数据迁移量、网络带宽的开销以及节点原有数据分布的变化.

(3) 对节点故障的敏感程度,即容错能力:由于文件或者说文件的片段在系统中存在多个副本,只要系统中至少存在其中一份副本,都可以说这个数据是完整的.数据放置策略的目标之一应该是保证系统在发生较多的节点故障时仍能确保数据的完整,不发生丢失.

(4) 保持集群的均衡:当文件块在集群中均匀分布时,系统中各节点能够达到最佳工作状态,从而提高执行效率,增加系统吞吐量等.

(5) 能耗:Hadoop 本身没有降低能耗的模块,而能耗问题已成为数据处理研究的一个敏感话题.研究表明,好的数据放置策略可以降低能耗[10−32].

在实际应用中,数据放置策略很难同时兼顾上述几个方面,例如,较强的容错能力通常是通过使系统中数据副本具有较高冗余度来换取的,但是这也会增大对副本进行维护的开销,导致数据在节点间迁移的操作更加复杂[9].所以在现有数据放置策略的设计中,都是根据具体的应用环境来寻求这几个因素的折中.

2 数据放置策略的优化

根据上述 MapReduce 环境下设计数据分配策略时所需考虑的关键问题和衡量数据放置策略的标准的分析,从用户的不同既定目标出发,例如能耗、负载均衡、I/O 性能和通信负载以及系统可靠性、运行效率等,归纳整理了当前 MapReduce 集群环境下的数据放置策略优化方法.

2.1 MapReduce集群系统能耗与数据放置

能耗已经日益成为数据中心和网络公司运营代价的的一个重要支出,也是影响云计算可扩展性和可靠性的重要因素.美国环境保护局报道:2000 年~2006 年之间,数据中心的能耗翻了一番;而且预计从 2007 年到 2011 年,如果集群服务器不采取任何节能措施,其能耗将翻两番.与 IT 设备的成本相比,在其生命周期期间,其运行能耗代价甚至超过了其本身成本[13].因此,无论从经济角度还是从环保角度,有效降低集群系统的运营开销以及对外界环境的影响,都已成为目前后台服务商着重考虑的问题.

(1) 能耗模型与度量

针对如何减少 MapReduce 框架下的能源消耗问题,已报道了许多研究成果.一些研究成果重在能效模型和度量方法的研究,旨在给出一种通用的能效度量标准,为基于 MapReduce 框架的能耗研究提供理论依据.早在 2008 年,加利福尼亚大学的研究人员[10]就对 MapReduce 的能耗问题展开研究,对负载在 MapReduce 执行的各个阶段(分布式文件的读写,中间数据的写、排序、合并等)的能耗进行分析,指出,良好的系统参数配置和好的负载设计可以极大地改善系统能耗.随后给出了一整套基于能耗分析的框架[11,12],并在不同的配置参数(如工作节点的数量、数据块大小、备份参数等)下测试了多种工作负载(读操作、洗牌操作和写操作)的能量有效性,并分析了能量效率和参数配置的关系[11];此外,他们还设计了基于 MapReduce 的能量代价模型,但是该模型只能用

于分析 MapReduce 中能量消耗情况,尚不能用于优化过程.宋杰等人在文献[13]中描述了一种云计算环境下的能效模型和度量方法,详细阐述了能效模型的数学表达、测量方法、计算方法,并从数学上推理了能效最大值的发生条件以及云计算环境中 CPU 密集型运算、I/O 密集型运算和交互型运算的能效特点,最后给出了进一步优化能效的思路.廖彬等人在文献[14]中也对节能问题建模,通过研究集群结构、节点与数据块状态与数据块存储机制,建立了 datanode 节点矩阵、节点状态矩阵、文件分块矩阵、数据块存储矩阵与数据块状态矩阵,为研究数据块可用性、节能算法等提供了基础模型.文献[15]提出了一种云环境下的绿色计算模型.该模型从程序设计语言、绿色数据中心框架构建、绿色分布式文件系统策略的角度详细做了研究,将为有效的绿色节能带来新的设计思路和方法.

(2) 能量节省的数据放置

针对数据放置,一些研究成果则给出了其优化系统能耗的实现策略和算法.文献[16]分别从集群级、应用程序级以及节点级探讨了目前 Hadoop 平台下的一些主要的节能优化技术.下面从这几个角度就数据放置策略对集群系统能耗的影响进行讨论.

在集群级的能源管理方面,几乎都是通过对系统资源进行有效的调度以达到降低分布式存储能耗的目的.Standford 的研究小组在文献[17]中指出,Hadoop 能耗效率较低的主要原因是其中有大量节点长时间运转在空闲状态,同时提出了 Covering Set(CS)方法,CS 从底层改变了 DFS 中数据放置方式,其主要思想是:将数据块与其副本中的至少 1 个数据块放在一部分特殊的节点(称为 covering set 节点)上,且这些节点不会被关闭.在保证所有数据块的可访问性的前提下,通过关闭与该数据块子集无交集的 datanode 节点以达到节能的目的.但该文仅提出了 CS 工作架构,并没有给出任何决定决策如何关闭节点(关闭哪些节点、关闭多少节点)的策略,因此并不能直接使用,除非使用者自己设计有效 CS 的实现方法.文献[18]就基于 CS 方法提出了通过使用称为镜像数据块副本(镜像)的数据放置方法(mirrored data block replication)降低数据中心的能耗.该方法将其他节点上的副本放置遵循 CS 节点集上的副本放置策略.在 CS 节点集不足以启动一个新任务的时候,使用新的数据块副本镜像,而不需要搜索并启动更多节点,从而降低能耗,但该方法仅考虑了 CPU 的能耗,并假设集群系统是同构的,具有一定的局限性.威斯康星大学麦迪逊分校的研究人员在文献[19]中也是采用将那些使用率低的节点关闭来提高能效,并提出一种新的节能技术 All-In Strategy(AIS).在某一时间窗 v 中机群收到任务,能量管理模块根据能量管理策略和工作负载特征选择性地关闭或开启若干节点;当机群结束计算时,如果 v 中还有剩余时间,则集群中的部分节点被关闭,机群保持 v 开始时在线节点的数量;其中,AIS 的一种极端情况就是 CS[17]方法.文中将 AIS 和 CS 做了比较,结果表明,CS 只在工作负荷的计算复杂度较低且状态转换时间占用整个任务执行的大部分时间时优于 AIS,在其他情况下,AIS 比 CS 更为有效.其克服了机群状态转换带来的额外开销代价,获取更好的能量有效性,而且不需要修改底层 DFS(分布式文件系统)的代码.Vasić 等人在文献[20]中对底层 MapReduce 和 HDFS 实现做了修改,使其具有能量感知能力,利用机器的睡眠状态节省能耗.文献[21]提出一个新的 Hadoop 集群系统的能源管理框架——VOVO.VOVO 同样也采用修改 Hadoop 平台的方法引入一个能耗管理模块以关闭部分空闲节点,从而节省能耗.但 VOVO 将一些节点挂起后,如果存储在该节点上的数据是唯一副本,则会影响系统的可访问性.文献[22]在不影响原始副本因子的前提下实现了集群中电源控制器和 HDFS 之间的连接通道,以动态地扩展集群中的节点数.当节点负载达到设定阈值时,通过自动地打开与关闭某些 DataNode 节点达到节能的目的,使之适应当前集群服务需求,并给出具体的动态配置算法(scaling up 和 scaling down).文献[23]提出了一种启发式的数据放置算法和两种节点调度策略,其思想是:数据块利用数据放置算法找到包含所有数据块的最小节点集合后被合理放置,然后,其余节点通过节点调度策略关闭以节省能源.其思想与 CS[17]方法的思路相似,仅是 CS 方法是事先确定存储数据的最小节点集合,然后进行放置.文献[24]对分布式存储系统下的节能问题进行定义,所提出的节能算法将 Rack 划分为 Active-Zone 与 Sleep-Zone 两个存储区域,通过改变数据块存储矩阵将数据块放入不同区域,并通过适时地关闭 Sleep-Zone 中的服务器达到节能的目的.其将系统存储结构的改变控制在 Rack 内部,一方面适应了现有的存储策略(如机架感知的存储策略),另一方面减小了存储结构配置时的资源开销.

在应用程序级,研究人员充分考虑了数据的性质和访问模式,努力开发节能、高效的数据处理应用程序.Harnik 等人在文献[25]中针对实际系统的研究发现数据访问具有很强的周期性,通过关闭空闲时段大量节点以达到节能的目的.Kaushik 在文献[26,27]中提出了一种基于数据划分的数据放置策略,通过对 Yahoo 公司 HDFS 集群内部数据块访问规律的研究,提出一种基于数据划分的 HDFS——Green HDFS.在 Green HDFS 中,集群被分成 Cold 区与 Hot 区,在 Hot 区的节点被频繁访问,因为它们驻留着集群的常用数据且具有高功率、高性能的 CPU,因此在集群负载较轻的情况下,可以将处于 cold 区的节点的 CPU,DRAM 和 Disks 转化成 inactive power 模式,从而节省能耗.Le 等人在文献[28,29]中采用 power-proportional 的概念,考虑了更新数据块的放置问题,通过在元数据管理引入索引技术,将 namenode 和 DataNode 有效耦合来有效地放置更新数据块.

从节点级来看,一般是通过在硬件层面提出很多节能的方法,如电压设置、处理器速度调整、扩大内存和使用低功率固态硬盘等[1].典型的工作有:文献[30]量化了 CPU 温度和能耗之间的关系,指出对于单个工作节点 CPU 温度是其功率消耗的可靠指标,并设计和实现了利用 CPU 温度动态地放置数据和调度任务的 MapReduce 框架,它是对 MARLA 框架[31]的改进,其底层采用了共享文件系统.由于这些方法可以完全脱离集群研究,这里不再赘述.

2.2 负载均衡的数据放置策略

负载均衡就是将工作任务进行平衡,分摊到多个操作单元上进行执行,也就是将重负载的运算分担到多台节点设备上做并行处理.每个节点设备处理结束后,将结果汇总,返回给用户,从而有效地扩展了网络设备和服务器的带宽,增加了吞吐量,加强了网络数据处理能力,提高了网络的灵活性和可用性,使系统处理能力得到大幅度提高.

Hadoop 自身的均衡策略包括:

(1) 在分布式文件系统开始运行时,系统采用静态放置策略尽量合理分配数据块的放置,优先照顾写入者的速度,让多份备份分配到不同的机架去.

(2) 当系统初始化完成后,Hadoop 采用基于负载迁移的动态负载均衡策略(运行 Balancer 程序),其使用单一的负载评价指标即磁盘空间使用率(默认是 80%)与单一的固定阈值(默认是 10%)来动态地实现负载均衡:当某服务器的磁盘空间使用率低于该规定阈值时,系统将其标记为“轻载节点”,待接收迁移来的负载;当某服务器的磁盘空间使用率高于该规定阈值时,系统将其标记为“超载节点”,待迁移过量的负载.

近年来,随着越来越多的企业和数据中心采用 MapReduce 作为其编程模型,利用 MapReduce 集群环境下负载均衡的数据放置策略来进一步提高集群系统的性能,也越来越受到关注.Yahoo Bangalore 网格计算集团工程经理 Devaraj Das 总结:不平衡的输入分片、不平衡的计算、不平衡的分区大小以及不平衡的异构硬件,这 4 个因素导致了负载不均衡问题[32].同样,文献[34,35]也分析了产生 map-偏斜和 reduce-偏斜的各种原因和表现,基本与 Devaraj Das 的一致.

下面从导致负载不均衡的上述 4 个方面对数据放置策略的优化方法进行深入的分析和总结:

- 不平衡的输入分片

对于传统的 MapReduce 实现,在执行一个 map 任务时就考虑了数据的本地性问题,但节点性能及运行状态、集群系统的负载均衡问题却被忽略.目前,相关方面的研究工作有:文献[36]指出,分块大小和文件访问模式有关.在并行文件系统中,依据访问模式对文件进行动态选择分块大小并合理布局,可以提高 I/O 性能及均衡负载.Vernica 等人在文献[37]中提出了一种状态感知的 mappers 的自适应 MapReduce 系统,可以通过持续监控所有 mappers 的执行情况,自适应地重新分割 map 的输入数据.林伟伟在文献[38]中,针对传统 Hadoop 数据放置策略(机架感知策略)存在的问题,即:(1) 存放数据副本的节点随机选取,可能导致在数据恢复时,数据的传输距离太远与传输延迟太长;(2) 由于随机选取节点没有考虑到各存储节点的负载平衡问题,提出基于评价值选取策略来改进数据块副本放置,该评价值同时考虑了节点距离与负载两个因素,提高了系统的存储性能并平衡了节点负载.Ye 等人在文献[39]中提出了一种新的副本放置策略.该策略在一开始在选择放置副本的节点时就考虑

了各节点的磁盘利用率和各节点的实时状态,这样既平衡了负载,又避免了系统运行过程中运行 balancer 产生的大量数据传输开销.

上述方法在一开始进行副本放置时就综合考虑了各节点性能和数据访问模式,而性能和数据访问模式信息的获取是问题的关键.一般来讲,该信息的获取有两种方式:一种是通过任务执行的历史信息获取,即,边执行边获取;另一种是事先通过某种手段(如实验)获取.

文献[40,41]均针对大规模云计算环境下,节点结构经常发生变化时数据的再平衡问题进行了深入研究.文献[40]提出了一种分布式的负载均衡方法,该方法利用分布式哈希表(DHT)来实现负载均衡,不需要依赖中心节点,各节点之间也不需要同步和获取全局信息,同时极大地减少了数据移动代价.但它只关心文件块的分布均衡问题,没有考虑节点性能.文献[41]实现了新的负载 rebalancer 算法,避免了中心节点过载.

- 不平衡的计算

不平衡的计算主要由于相同的任务针对不同的数据对象,其产生的计算量和资源需求不同而导致的,例如循环操作.文献[42]提出了一种自动 skew 迁移方法——SkewTune,其基本思想是:当集群中的一个节点变为空闲时,SkewTune 标识出最大的预期剩余处理时间的任务,并将其还未处理的数据重新划分,以充分利用集群中各节点,同时保留了输入数据的顺序,使得原来的输出可以被重构.SkewTune 重复该过程,直到完成所有作业.文献[32]提出了一种负载自适应的 MapReduce 实现——MARLA,其底层使用了共享文件系统,导致每个节点不需要负责存储数据.这种从节点分离的 I/O 结构可以快速地进行系统的重新配置和加速应用程序的周转时间,同时, MARLA 的设计采用了不同于目前大多数的 MapReduce 实现(如 Hadoop)的任务调度机制,即,一种动态任务调度机制,使得各节点可以按照各自的处理能力请求任务,从而使其能够很好地适应异构集群环境.

- 不平衡的分区大小

map 阶段产生的中间结果,需要被 partition 到不同 reduce 任务,partitioner 函数的有效性决定了分区的平衡性.因此,针对不同的应用,可能需要设计不同的 partitioner 函数,以保证分区的平衡,从而合理而有效地利用系统资源.Ibrahim 等人在文献[43]中采用了异步的 map 和 reduce 模式,通过在 map 阶段之后增加一个明确的规划阶段,以便跟踪各数据节点上中间键值的变化和分布情况,从而在 partition 时,可以按照本地性和公平性的原则进行数据分配,既减少了网络带宽,又避免了 reduce 阶段的计算倾斜.同理,Kolb 等人在文献[44]中将实体分配给 reduce 任务前,先利用一个预处理 MapReduce 作业分析了数据的分布情况.Slagter 等人在文献[45]中,专门针对排序算法提出一种改进的使用数据抽样的 partition 机制,并将 mapper 的输出负载尽量均衡地分配给每个 reducer,以均衡 reducer 端的负载.上述文献都试图先获取中间键值的分布情况,然后再进行 partition 操作,解决不平衡分区大小问题.Fan 等人在文献[46]中提出了一种根据中间键值对的分布情况进行 MapReduce 操作调度的新算法,以解决 MapReduce 操作的负载均衡问题.该方法在 partition 后,根据数据的分布情况进行任务调度.此外,Gufler 等人在文献[47]中针对传统的 MapReduce 将 mapper 的输出结果传递给 reducer 时只考虑关键字值而未考虑实际的负载量,提出了一种改进的负载均衡方法,通过创建比 reducers 数更多的分区,并基于代价进行分区分配,有效地提高了负载分配的灵活性.该方法通过创建更多的分区提高 partition 的灵活性,从而平衡负载.Fan 等人在文献[48]中提出了虚拟分区技术以改善 reduce 端的负载均衡,在每个 map 任务完成后,输出关键字均根据 Hash 函数划分到不同的虚拟分区,并通过 LBVP(load balance algorithm based on continuous virtual partition)算法将虚拟分区合并为和 reduce 任务数相同数量的输入数据,以确保每个 Reduce 任务有平衡的输入数据.

- 不平衡的异构硬件

大规模数据中心集群日益趋向于异构模式.文献[49]在进行数据放置时考虑了各节点的处理能力,即,在应用程序运行前评估集群中各节点,然后按照节点处理能力分配数据块,更快的节点获得更多的处理数据,以均衡负载.Liu 等人在文献[50]中也为平衡异构节点的负载提出了一种新的负载均衡算法.Fan 等人在文献[51]中针对输入数据被分割成预定义块大小的数据块,依据异构环境下机器的性能差异,提出了性能感知的数据分布和执行中的动态数据迁移方法.文献[49,50]并没有改变数据分块的大小,而是根据节点处理能力,通过调整分配给

各个节点的分块个数来负载均衡;文献[51]则直接根据节点性能调整分块大小.

Ahmad 等人在文献[52]中分析了异构环境下导致 MapReduce 性能差的两个关键因素:其一是在 map 端,由于高性能节点可能去抢夺分配给低性能节点的任务,而数据还在低性能节点上,从而造成额外的网络通信开销,而且更糟糕的是,这部分通信开销要与 shuffle 竞争通信资源,使得在 map 的后期,会产生突发性的网络流量;其二是异构性放大了 reduce 计算的不平衡性.由此,Ahmad 等人开发了一套 Tarazu 组件,用来改进 MapReduce 性能,该组件由包括:(1) 通信感知负载均衡的 map 计算(CALB),CALB 依据集群资源情况和 shuffle 负载情况判定集群是处于 No-steal 模式还是 Task-steal 模式,在保证通信量不太集中的同时,尽量使任务能够本地执行;(2) 通信感知的 map 计算调度算法 CAS,CAS 采用边执行边调度的执行方式,避免突发性的网络流量;(3) 负载均衡预测的 reduce 计算(PLB),PLB 依据对 map 阶段各节点的执行能力估计,来给各节点分配任务.Gandhi 等人在文献[53]中研究了 Tarazu 的局限性[52],即,Tarazu 在 reduce 任务执行阶段过早地预测负载情况,导致再平衡远离了最优目标,并提出自己的解决方案——PIKACHU 任务调度器.该方案在预测的精确度和延迟负载调整代价之间做了一个微妙的平衡.这两种方法不同于上述的方法,它们不仅考虑了 map 端的数据分配,而且考虑了 reduce 的数据分配,且主要通过通信负载感知的任务调度实现系统的负载平衡.

除上述的研究成果以外,文献[54–56]等从另一种角度对 MapReduce 集群环境下的负载均衡的放置策略进行了研究.Fischer 等人在文献[54]中通过研究指出:增加副本的数量总是有助于负载均衡,这种传统观点是不正确的.Groot 等人在文献[55]中针对 MapReduce 在负载均衡方面的局限性,借鉴了 MapReduce 在扩展性、容错性和易用性方面优势,提出了一种新的分布式数据处理平台 Jumbo.Jumbo 为了更好地解决负载均衡问题,设计了自己的 Jumbo DFS 和 Jumbo Jet:Jumbo DFS 采用了共享式文件系统(Google file system);Jumbo Jet 提供了数据处理环境,其最大的优点就是可以轻松地将无论多复杂的算法表示成一个单一的作业.Wang 等人在文献[56]中提出了一种负载感知的数据放置策略,根据各服务器的负载,在存储服务器之间分发数据,并自动地将数据从负载很重的服务器向负载较轻的服务器迁移,以均衡负载.

2.3 改善I/O性能与通信负载的数据放置策略

在分布式环境下,节点的 I/O 能力及网络带宽是影响整个集群性能的两个关键因素.为了充分利用集群的计算资源,系统可能将任务部署到集群中的所有节点,从而导致一个任务很可能需要处理不存储在本地的数据.考虑到数据传输的开销,传统的 MapReduce 集群系统在作业执行时,对于一个 map 任务,系统会尽量将任务分配给处理数据所在的节点上,避免移动数据带来的 I/O 与通信开销;但在任务执行的过程中,如果出现“拖后腿”的任务,那么系统会重新启动一个备份任务,将会造成数据的大量迁移;特别是在异构环境下,处理能力快的节点可能“抢夺”分配给低性能节点的任务,也会造成额外的网络通信开销;更具典型意义的就是 map 的中间结果如何传给 reduce 节点的整个过程,该过程涉及到复杂的 shuffle 阶段,对于这一过程,Hadoop 没有任何优化策略.近年来,针对以上几个方面对 MapReduce 集群系统性能的影响,研究表明,良好的数据本地性是改善 I/O 性能和通信负载的最有效的途径[57,58].

- 基于数据相关性的数据本地性

基于数据相关性的数据本地性是考虑到数据本身的特点(如访问频率、大小、位置等)及数据之间的相关性,尽量将相关度高的数据放在一起,减少数据移动的代价.郑湃等人在文献[59]中对数据集之间的数据依赖关系进行建模,将数据集间的数据依赖关系表示成聚类矩阵,然后采用聚类矩阵对所有数据集组成的集合进行划分,并分别为每个划分分配存放位置.Yuan 等人在文献[60]中不仅考虑了数据之间的相关性,而且还考虑了数据与计算节点的相关度,提出了同样采用聚类矩阵表示相关性的数据放置策略.该策略分为两步:在建立阶段,先将数据聚类成 K 个集合;在执行阶段,再计算数据与节点的相关度,并将其放置到相关度最大的节点上.实验结果表明,该方法可以有效减少数据传输次数.但是由于它没有考虑网络带宽和数据大小,可能会导致数据传输次数减少;但传输数据较大或者传输带宽较小,反而会延长网络传输时间.文献[61]根据实际数据的一些特点,例如共享数据、数据间的相互依赖、实际应用和用户位置的变化等,提出了一种动态数据放置策略.Palanisamy 等人在文献[62]中提出了一种智能云计算环境下的 MapReduce 资源分配系统——Purlieus.Purlieus 利用特别的数据和

VM 放置技术,尽量减少 map 任务和其处理数据之间的距离以及 reduce 任务和 map 产生的中间数据的距离,以降低传输开销.刘少伟等人在文献[63]中提出了基于相关度的两阶段高效数据放置策略,通过量化计算数据集之间的关系,将关系紧密的数据集放置到同一个数据中心,将关系松散的数据集放置在不同数据中心,保证了不同数据中心数据集之间的低耦合性和同一数据中心数据集之间的高内聚性;任务调度策略在运行阶段将任务调度到数据依赖最大的数据中心执行,并将新产生数据集放置到相关度最高的数据中心.Wu 等人在文献[64]中通过分析用户上传文件请求,从中挖掘出这些文件的相关性.同时,为了提高数据访问效率,在初始放置时,尝试把相关的文件放置在用户或程序经常使用的节点上.

上述方法均没有改变底层文件系统的数据放置策略,而 Abad 等人在文献[65]中提出了一种分布式的自适应数据副本分配和放置方法——DARE,该方法将访问频率高的数据分配更多的副本,以提高对数据并发访问的本地性;同时,该方法还将总是被同时访问的数据块放置在不同的节点上,以减少在特定节点上的资源争用.文献[66]提出了 CoHadoop,通过引入一个新的文件属性来标识数据文件之间的相关性,修改了底层 HDFS 的数据放置策略,允许应用程序来控制数据的存储位置,尽量将相关数据放置在同一组节点上.然而,CoHadoop 把工作量转移到了用户和管理员身上,由他们来确定哪些数据应该放在一起,以及放在什么位置.文献[67]将该过程自动化,其将数据放置问题转化为标准图划分,以有效地计算近似最优解;同时,还提出了两种启发式副本放置方案.

文献[68,69]与前面不同,提出了面向列的存储技术.文献[68]提出了一种用于处理连接的列数据放置策略,面向列的文件存储可选特定列数据的索引信息,列数据只有在真正被处理的时候才被物化,极大地减少了传输开销.文献[69]也是一种面向列的存储技术,通过修改 Hadoop 的数据放置策略,将列存储在一个单独的文件,但每列定位相关联的列在同一个节点上,且采用了面向列的数据压缩技术以节省 I/O.

- 基于数据预取技术的数据本地性

数据预取机制是通过提前将数据预取到相应的计算节点,在保证良好的数据本地性的同时,有效地降低作业的执行时间.这是由于数据预取和数据处理是同步进行的,数据传输和数据处理在时间维度是重叠的.Sangwon 等人在文献[70]中,针对多用户共享 MapReduce 环境提出了 pre-fetching 和 pre-shuffling 两种优化技术及实现——HPMR.HPMR 改变了原来的 hash partitioner 方法,不论在 map 端还是 reduce 端,pre-fetching 通过提前计算,尽量将数据预取到或靠近其所对应的任务节点上;而 pre-shuffling 是在 map 任务执行前增加一个模块,以保证其对应的输入数据所产生的输出能够被划分到离自己最近的 reducer 上,保证了良好的数据本地性,减少了 I/O 与通信开销.Hammoud 等人在文献[71]中提出了一种本地感知的 reduce 任务调度策略,考虑分区的位置和大小,其 partitioner 函数仍然采用默认的 hash partitioner,使 reduce 任务尽量本地化,以减少 shuffle 数据量,提高 MapReduce 的性能.文献[72]也针对异构和共享环境下差的数据本地性对系统性能的影响,提出了数据预取机制.在作业执行过程中,当任意节点接收到任务时,如果检测到其需要的输入数据不在本地,就立即触发预取线程将数据预取到预取缓存,但该方法只考虑了 map 任务.

此外,Jin 等人在文献[73]中提出了一种可用性感知的数据放置策略(ADAPT),在数据放置前检测节点性能,并根据各节点的可用性来分配数据,减少网络流量,提高数据本地性和优化了系统性能.Xie 等人在文献[74]中针对 shuffle 处理策略的不足,采取管道策略,将 map 生成的数据通过管道直接传输到 reduce,降低了 I/O 代价,提高了效率.

2.4 考虑其他因素的数据放置策略

数据放置策略还涉及到其他资源(CPU、存储系统等)利用情况、可用性、容错性、系统运行效率等方面.文献[75]考虑到集群中不同负载可能对资源的需求不同,有的可能是 CPU 密集的,有的可能是 I/O 密集的,根据不同负载对资源的实际需求以及各节点的资源情况进行任务的动态分配和数据放置,提高了资源利用率.文献[76]设计了采用文件的“流行度”进行副本放置的系统,有效地缓解了热点,并且加快了程序执行效率.IBM 公司 T. J. Watson 研究中心的研究人员在文献[77,78]中为 Hadoop 设计了一个资源感知的调度器.该调度器通过耦合 map 和 reduce 的处理过程,并联合优化了这两个阶段的数据放置策略,从而缓解了“job starvation”问题,并分别在

map阶段和reduce阶段采用不同的调度算法提高了两个阶段的的数据本地性.文献[79,80]均通过建立模型分析了可用性和副本数量之间的关系,并提出了各自的动态副本策略——CDRM 和 D2RS.CDRM 是在保证数据可用性的同时,根据负载变化和节点能力调节副本数量和位置[78];在 D2RS 中,越是被频繁访问的数据越被赋予更大的权值,当某个文件的访问率超过规定阈值时,动态副本放置算法被触发,以负载平衡的方式放置副本[80].为了提高集群中各节点存储空间利用率,文献[81]提出了一种大数据放置策略及其在 Hadoop 中的实现——RCFile (record columnar file),采用了面向列的文件格式,并为了保证一条记录的所有列都被存储到一个节点上.RCFile 首先对记录进行分组,然后以类似于 PAX 存储模式存储[82].

2.5 数据放置策略的分析与归纳

以上从不同的度量角度对目前的数据放置策略研究进行了分析,然而在实际应用中,用户可能并不只关心一种度量标准,而不同的度量标准间往往又是相互矛盾的.例如,

- 为了降低能耗,往往通过节点调度方法,将一些空闲或利用率低的节点挂起或关闭,导致副本个数的减少,不仅降低了系统可靠性,而且也会出现“热点”问题,使大部分任务被集中在个别节点上执行,影响了系统负载更灵活的分配.
- 较强的容错能力通常是以系统中数据副本具有较高冗余度为代价的,但这也会增大副本对系统资源的占用及副本维护的开销,导致数据在节点间迁移的操作更加复杂.
- 为了充分利用系统资源、提高系统吞吐量和系统运行效率,往往希望将任务平均分配到各计算节点,然而这可能是以能耗损失为代价的;另外,为了平衡负载需要迁移大量的数据,增大数据传输可能性,从而增加总传输时间,加重带宽负荷,延长任务等待数据时间.
- 为了提高数据本地性,系统往往会将任务和数据聚集在少数几个节点,降低数据传输的可能性,但会使任务分配不均衡,增加任务排队等待时间,降低公平性和机器使用率.
- 基于 MapReduce 的集群系统通过数据副本的动态平衡来提高系统的可靠性和可扩展性,数据的动态迁移既增加了副本管理的难度,又造成大量的 I/O 和网络开销.

除此之外,用户的实际需求可能远比我们想象的更加复杂,在设计数据放置策略时,应根据具体的应用环境及用户需求来寻求上述因素折中的合作数据放置策略.为了度量各种不同放置策略的优劣,如何有效地设计方案就显得尤为重要.一般来讲,实验性能评估指标往往会涉及到以下几点:

(1) 任务独立响应时间与任务总响应时间:任务独立响应时间是指单个任务从提交到返回最终处理结果的时间,反映了 MapReduce 模型和用户的交互能力;任务总响应时间是指所有机器完成所有任务的时间,反映了整个系统的计算能力和吞吐量.

(2) 平均响应时间:该指标用来评价大量任务并行处理的性能,是并行编程模型好坏的重要衡量标准.

(3) 加速比:该指标是指对于特定的数据和任务量,当增加计算节点数量时,对并行计算能力提升的比率,主要反映了 MapReduce 模型在可扩展方面的性能.

(4) 公平性:公平性是指在多个任务同时执行的情况下,系统任务的调度策略是否满足某些公平性原则.

(5) 容错性:集群中节点失效是正常的,在节点出现故障后的任务恢复能力和系的稳定性,成为 MapReduce 的一个重要的测试指标.

另外,不同的度量标准又各有侧重,例如,

- 面向节能的数据放置策略需要考虑能耗的度量方式,一般采用能效即单位能量完成的有效任务作为评价指标;
- 面向负载均衡的数据放置策略一般采用在用户平衡因子的约束下数据分布情况作为评价指标;
- 对于改善 I/O 和通信负载的数据放置策略,一般将数据的本地性作为一个重要的衡量指标等等.

为了更直观地表达对上述数据放置策略的分析,表 1 按数据处理阶段对数据放置策略的典型研究方法及特点进行了归纳总结:

(1) map 端数据放置.

默认的 Hadoop 数据放置策略采用了静态放置,不适合异构环境或者节点结构经常发生变化的情况.为了解决该问题,一般有两种研究思路:一种是先静态放置再动态调整,即,在初始化时先按预定义好的数据块大小、副本数量及放置方法放置数据,随后在执行过程中如果检测到不平衡,再动态调整副本的放置;另一种思路是直接采用动态放置策略,即,数据块的大小、副本数量、放置位置从一开始就考虑系统各节点性能和系统运行情况,并在运行时,可以依据网络状况、存储空间和用户需求等动态地创建或者删除副本.虽然都可以均衡节点负载,但在动态调整时需要执行大量额外的操作,特别是频繁的数据传输会带来巨大的网络开销.另外,一些研究成果还在上述两种数据放置策略的基础上,通过节点调度等技术将数据放置在一些特殊节点上,在保证数据可用性的同时,可以根据系统资源和负载情况将空闲节点关闭或转换到低能耗状态,以达到节能的目的.

(2) Reduce 端数据放置涉及到 map 的输出传递到 reduce 端的整个复杂的过程,即 shuffle 阶段.

为了改善 reduce 端的数据放置性能,研究人员主要针对以下方面进行优化:(1) 提高 reduce 端数据放置的本地性,即,尽量将 map 的输出放置到本地或离自己最近的节点上执行,特别是在异构环境下,对减少 shuffle 的数据传输开销、均衡负载及提高资源利用率均有重要的意义;(2) 尽量简化 shuffle 过程,例如,文献[73]采用管道技术直接将 map 结果输出到 reduce 端.

此外,上述这两个方面可能还涉及到更加细节的问题,例如在数据放置时,在考虑数据本地性的同时,如何减少在特定节点上的资源争用问题:根据不同负载对各类资源的实际需求不同,是否可以根据各节点的资源情况进行任务的动态分配和数据放置,以充分提高各类资源的利用率等.

Table 1 Analysis and comparison of data placement strategy

表 1 数据放置策略的分析与比较

数据处理阶段	解决的关键问题	研究方法	特点	典型算法
map	异构环境下节点性能差异、节点结构变化导致的负载不均衡及资源竞争	先静态放置再动态调整	主要解决处理节点结构变化时数据的动态再平衡;边执行边探测节点性能,而后动态调整,但动态调整会导致较大的传输和网络开销	文献[40−42,79,80]
		动态放置	需要提前预测和分析节点性能、网络状况和用户需求等,充分考虑了系统资源,同样动态创建或者迁移会增加巨大的网络开销	文献[37−39,49,51,52]
	集群系统的I/O性能与通信负载	基于数据特征及相关性的动态放置	需要对数据间的数据依赖关系及访问特征进行分析甚至建模	文献[59,60,62,63,65−67]
	集群系统能耗	采用原始底层数据放置方法	未改变原始数据放置策略,但根据各节点负载情况关闭或休眠空闲节点	文献[19,22,25,28,29]
		改变底层数据放置方法	数据被放置在一些特殊节点上,在保证数据可用性的同时,可以根据需求将空闲节点关闭或转换到低能耗状态	文献[17,18,20,21,23,26,27]
reduce	reduce端数据放置的本地性,以均衡负载、减少数据传输开销、提高资源利用率	优化partition	根据中间键值的变化、分布情况及节点负载,在partition时,可以按照本地性和公平性的原则进行数据分配	文献[43−47]
		基于数据的相关性	需要对数据间的数据依赖关系及访问特征进行分析甚至建模	文献[60,62]
		数据预取技术	由于数据预取和数据处理往往是同步进行的.既保证了良好的数据本地性又有效地降低了作业的执行时间.	文献[70,71]
	简化shuffle阶段	管道技术	map和reduce实现直接互连	文献[74]

3 总结与展望

MapReduce 为日益膨胀的海量数据处理提供了一个易于实现且可靠的并行编程平台,并已经有很多成功的应用.在 MapReduce 执行过程中,如何可靠而合理地放置数据,是影响其系统性能的一个关键因素.综合上述分析,将目前 MapReduce 集群环境下的数据放置研究的热点问题归纳总结如下:

(1) MapReduce 模型运行在众多廉价的普通硬件上,硬件错误和节点失效被视为正常情况.

虽然在设计上,MapReduce 已经具备了一定的容错性——推测执行,以保证任务的能够被正常执行,然而对用户而言,仅仅能够保证任务的正常执行是远远不够的,例如在推测执行过程中,如果系统面临负载过量,则会导致作业响应时间的降低,而他们同时又会相互竞争资源,不能快速完成的任务会被标记为掉队者.随着掉队者数量的增加,需要执行的任务数也在不断增加,从而进一步加剧了系统负载,使系统形成了恶性循环,严重影响了系统性能.另外,在异构环境下,由于节点性能差异会导致不正确的、过度的推测执行任务的执行,会导致系统性能的急剧下降,系统在检测到存在掉队任务时(称为后备任务),会选择另外一个工作节点执行该后备任务,如果所需数据不在该节点上,而且往往执行后备任务所需的输入数据不在该执行节点上,将会导致大量的数据迁移,从而影响系统性能.因此,在考虑系统容错性的同时,还必须考虑系统的稳定性和用户需求.

(2) 随着网络和云计算的发展,研究人员发现:Hadoop 在处理海量小文件时,存在很大的缺陷.

当海量小文件出现在 HDFS 中时,系统需要为每个小文件保存元数据信息,存在名称节点占用率高和访问效率低的问题,大量的元数据将服务器的内存耗费殆尽;当文件数量大且存在多个副本时,系统需要为其分配多个存储节点,将导致大部分时间都花费在数据传输上,且占用了大量的系统开销(I/O、网络资源等),以至于出现小文件的存储效率下降以及大量小文件等待 map 任务处理等任务管理问题.因此,针对小文件的存储、检索以及对应的任务调度研究,是 MapReduce 技术领域的一个重要研究课题.

(3) MapReduce 对中间数据的处理,往往成为影响系统性能的主要瓶颈.

主要体现在:1) map 任务的输出和 Reduce 任务的输入都被预先存放在节点内存,如果达到溢写阈值,系统需要启动线程完成溢写及合并数据操作,该操作也将占用大量内存,从而导致磁盘阻塞;2) reduce 端数据的非本地性会导致网络流量的突增,耗费大量网络资源甚至造成网络拥塞;3) map 的中间输出数据被保存在本地磁盘,当数据被复制到运行 reduce 任务的节点时,也被写入本地磁盘且之前的数据不能被删除,直到整个作业执行完毕.如果 map 产生的中间数据量大,将极大地增加磁盘负荷.特别是当多个任务同时运行时,很容易造成磁盘空间紧张甚至不足而导致任务运行失败.合理的 partition 方案及 shuffle 阶段的有效管理,都是提高中间数据处理性能(磁盘空间利用率、I/O 性能、网络带宽及作业响应时间等)的非常重要和迫切需要解决的问题.

(4) 数据放置与任务调度关系密切,二者之间相互影响,并对系统性能产生较大的影响.

在实现 MapReduce 框架时,为了提高作业的执行效率和系统资源利用率,MapReduce 任务调度尽量采用移动计算而不移动数据的调度方式——任务本地化计算,即,尽量将任务分配给包含要处理数据的计算节点上,否则会导致大量的数据传输浪费系统资源.而 MapReduce 的典型实现仅考虑了 map 端的数据本地性,未考虑 reduce 端的数据本地性.由于 reduce 任务不同于 map 任务,其处理的数据是 map 任务处理后的数据,是动态产生的,只有在 map 任务完成后才能确定数据的分布和大小.一般情况下,数据经过 map 任务处理后,部分中间数据就被复制到某个 reduce 端去执行了,不能再更改且 reduce 任务的个数也不能修改,从而导致了 reduce 端的数据倾斜.此外,如第 2.2 节所分析的,不平衡的异构硬件和不平衡的计算都可能导致 reduce 端的数据倾斜.因此,应该将任务调度转换为节点位置公平性求解,通过综合考虑数据位置和节点负载,做到均衡配置以达到最佳任务分配.

(5) 目前,针对大数据计算的能耗问题备受关注,正如第 2.1 节所分析的那样,出现了许多研究成果.

随着计算机技术的发展,以处理器频率不断增长来维系的摩尔定律受到了能耗的挑战.研究表明,处理器频率每提高 40HZ,就会同时带来 60%的能耗增加.因此在大规模集群下,一般采用软件节能技术来降低能耗,主要包括节点管理技术和数据管技术两个方面的研究[6],其不需要改变现有的硬件设施,可推广性强、灵活性好,因此得到了更加广泛的关注.

(6) 随着计算机技术的迅猛发展,多核技术为数据并行和分布式应用提供了更为广阔的舞台.

由于其数据存储是共享存储和分布式存储共存的方式,基于普通的集群平台设计的并行编程模型直接移植到多核平台上,会导致无法获得应有的性能和可伸缩性.随着计算节点核数目的增加与计算能力的不断提升,编程模型的不匹配所造成的性能损失将更为显著.因此,适应多核环境的编程模型及数据存储管理,将成为未来研究的一个重点.

综上所述,数据放置策略是一个极其复杂的研究课题,采用合理的数据放置,可以有效地提高集群系统性

能.以下结合我们的研究工作,将其进一步的研究工作和研究内容归纳如下:

- 副本放置与任务调度关系密切,二者之间相互影响,并对系统性能产生较大的影响.为了提高数据本地性,系统往往会将任务和数据聚集在少数几个节点,降低数据传输的可能性,但是却使任务分配不均衡,反而增加了任务排队等待时间,降低了公平性和机器使用率;而为了提高机器使用率,将任务分配到各个节点,均衡系统负载,往往又会降低数据本地性,增大数据传输可能性,增加总传输时间,加重带宽负荷,延长任务等待数据时间.因此,如何平衡副本放置与任务调度,提高系统吞吐量,对于提高云计算系统性能也是一个非常重要和迫切需要解决的问题.
- 动态副本放置策略.当集群中出现节点失效、增加新节点或大多数副本的访问集中在少数节点时,如何调整数据放置以保证数据的可用性、减少数据传输开销及节点的 I/O 性能,是值得研究的课题.
- 研究新的 partition 方案,将 map 输出尽量分区到合适的 reduce 节点,提高 reduce 端的数据本地性.可以利用数据样本的调试运行,了解数据的性质和 reduce 函数,并相应地划分后续运行数据和任务.
- shuffle 阶段的管理.使其可感知各节点的计算性能和网络环境,既可以充分利用集群资源,又能平衡网络负载和 I/O,做好 map 和 reduce 之间的衔接.更深入的研究还包括将 shuffle 从 reduce 中分离出来,使其可以方便地进行统一管理和优化,有效地提高任务执行的效率.
- 研究与数据放置策略相关的节能机制及与该机制相匹配的节点失效处理、元数据管理、任务调度与负载均衡算法.

4 结束语

随着社会信息化程度的提高,数据呈爆炸式增长,MapReduce 编程模型凭借其强大的数据处理能力,已经成为支持大规模数据密集型应用的有效的编程模型.在其运行过程中如何有效、合理地放置数据,是影响其性能的一个关键因素,也是目前研究的一个热点课题.本文首先对 MapReduce 编程模型的典型实现——Hadoop 缺省的数据放置策略进行分析,并进一步讨论了 MapReduce 框架下设计数据放置策略时所需考虑的关键问题和衡量数据放置策略的标准;其次,对目前 MapReduce 集群环境下的数据放置策略优化方法进行分析和探讨;最后,探索了 MapReduce 集群环境下数据放置策略研究的几个问题和下一步的研究方向.

References:

[1] Meng XF, Ci X. Big data management: Concepts, techniques and challenges. Journal of Computer Research and Development, 2013,50(1):146−169 (in Chinese with English abstract).

[2] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008,51(1): 107−113. [doi: 10.1145/1327452.1327492]

[3] Lee KH, Lee YJ, Choi H, Chung YD, Moon B. Parallel data processing with MapReduce: A survey. ACM SIGMOD Record, 2012, 40(4):11−20. [doi: 10.1145/2094114.2094118]

[4] Ghemawat S, Gobioff H, Leung ST. The Google file system. ACM SIGOPS Operating Systems Review, 2003,37(5):29−43. [doi: 10.1145/1165389.945450]

[5] Hadoop distributed file system. 2012. http://hadoop.apache.org/hdfs

[6] Wang YJ, Sun WD, Zhou S, Pei XQ, Li XY. Key technologies of distributed storage for cloud computing. Ruan Jian Xue Bao/ Journal of Software, 2012,23(4):962−986 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4175.htm [doi: 10.3724/SP.J.1001.2012.04175]

[7] Zicari RV. Big Data: Challenges and Opportunities. Big Data computing, 2014:103−128.

[8] Apache software foundation, “Hadoop”. http://hadoop.apache.org/core

[9] White T. Hadoop: The Definitive Guide. Sebastopol: O’Reilly Media, Inc., 2012.

[10] Chen Y, Wang T, Hellserstein JM. Energy Efficiency of Map Reduce. Berkeley: University of California at Berkeley, 2008.

[11] Chen Y, Keys L, Katz RH. Towards energy efficient mapreduce. Technical Report, UCB/EECS-2009-109, Berkeley: EECS Department, University of California, 2009.

[12] Chen Y, Ganapathi AS, Fox A, Katz RH, Patterson D. Statistical workloads for energy efficient mapreduce. Technical Report, UCB/EECS-2010-6, Berkeley: University of California at Berkeley, 2010.

[13] Song J, Li TT, Yan ZX, Na J, Zhu ZL. Energy-Efficiency model and measuring approach for cloud computing. Ruan Jian Xue Bao/ Journal of Software, 2012,23(2):200−214 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4144.htm [doi: 10.3724/SP.J.1001.2012.04144]

[14] Liao B, Yu J, Zhang T, Yang XY. Energy-Efficient algorithms for distributed file system HDFS. Chinese Journal of Computers, 2013,36(5):1047−1064 (in Chinese with English abstract).

[15] Zhang GG, Li C, Xing CX. A green computing model based on cloud environment. Journal of Chinese Computer Systems, 2013, 34(5):1016−1020 (in Chinese with English abstract).

[16] Feng BL, Lu JH, Zhou YL, Yang N. Energy efficiency for MapReduce workloads: An in-depth study. In: Proc. of the 23rd Australasian Database Conf. Vol.124. Australian Computer Society, Inc., 2012. 61−70.

[17] Leverich J, Kozyrakis C. On the energy (in) efficiency of Hadoop clusters. ACM SIGOPS Operating Systems Review, 2010,44(1): 61−65. [doi: 10.1145/1740390.1740405]

[18] Yazd SA, Venkatesan S, Mittal N. Boosting energy efficiency with mirrored data block replication policy and energy scheduler. ACM SIGOPS Operating Systems Review, 2013,47(2):33−40. [doi: 10.1145/2506164.2506171]

[19] Lang W, Patel JM. Energy management for mapreduce clusters. Proc. of the VLDB Endowment, 2010,3(1-2):129−139. [doi: 10.14778/1920841.1920862]

[20] Vasić N, Barisits M, Salzgeber V, Kostić D. Making cluster applications energy-aware. In: Proc. of the 1st Workshop on Automated Control for Datacenters and Clouds. ACM Press, 2009. 37−42. [doi: 10.1145/1555271.1555281]

[21] Oppenheim BM. Reducing cluster power consumption by dynamically suspending idle nodes [MS. Thesis]. San Luis Obispo: Facalty of California Polytechnic State University, 2010.

[22] Maheshwari N, Nanduri R, Varma V. Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework. Future Generation Computer Systems, 2012,28(1):119−127. [doi: 10.1016/j.future.2011.07.001]

[23] Xiao YW, Wang JB, Li YP, Gao H. An energy-efficient data placement algorithm and node scheduling strategies in cloud computing systems. In: Proc. of the 2nd Int'l Conf. on Advances in Computer Science and Engineering (CSE 2013). Paris: Atlantis Press, 2013. 59−63.

[24] Liao B, Yu J, Sun H, Nian M. Energy-Efficient algorithms for distributed storage system based on data storage structure reconfiguration. Journal of Computer Research and Development, 2013,50(1):3−18 (in Chinese with English abstract).

[25] Harnik D, Naor D, Segall I. Low power mode in cloud storage systems. In: Proc. of the IEEE Int'l Symp. on Parallel & Distributed Processing (IEEE IPDPS 2009). Piscataway: IEEE Computer Society, 2009. 1−8. [doi: 10.1109/IPDPS.2009.5161231]

[26] Kaushik RT, Bhandarkar M. GreenHDFS: Towards an energy-conserving storage-efficient, hybrid Hadoop compute cluster. In: Proc. of the USENIX Annual Technical Conf. Berkeley: USENIX Association, 2010. 1−9.

[27] Kaushik RT, Bhandarkar M, Nahrstedt K. Evaluation and analysis of green HDFS: A self-adaptive, energy conserving variant of the hadoop distributed file system. In: Proc. of the 2010 IEEE 2nd Int'l Conf. on Cloud Computing Technology and Science (CloudCom 2010). Indianapolis: IEEE, 2010. 274−287. [doi: 10.1109/CloudCom.2010.109]

[28] Le HH, Hikida S, Yokota H. Efficient gear-shifting for a power-proportional distributed data-placement method. In: Proc. of the 2013 IEEE Int'l Conf. on Big Data. Washington: IEEE Computer Society, 2013. 76−84. [doi: 10.1109/BigData.2013.6691557]

[29] Le HH, Hikida S, Yokota H. NDCouplingHDFS: A coupling architecture for a power-proportional Hadoop distributed file system. IEICE Trans. on Information and Systems, 2014,97(2):213−222.

[30] Hartog J, Dede E, Govindaraju M. MapReduce framework energy adaptation via temperature awareness. Cluster Computing, 2014, 17(1):111−127. [doi: 10.1007/s10586-013-0270-y]

[31] Fadika Z, Dede E, Hartog J, Govindaraju M. MARLA: MapReduce for heterogeneous clusters. In: Proc. of the 12th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing (CCGrid 2012). IEEE Computer Society, 2012. 49−56. [doi: 10.1109/CCGrid.2012.135]

[32] Das D. How to Hadoop. 2010. http://trac.nchc.org.tw/cloud/raw-attachment/Fwiki/HadoopWorkshop/hadoop-assembled.pdf

[33] Brown RE, Brown R, Masanet E, Tschudi B, Shehabi A, Stanley J, Koomey J, Sartor D, Chan P. Report to congress on server and data center energy efficiency: Public law. Berkeley: Ernest Orlando Lawrence Berkeley National Laboratory, 2007. 109−431.

[34] Kwon Y, Balazinska M, Howe B, Rolia J. A study of skew in mapreduce applications. Open Cirrus Summit, 2011.

[35] Kwon YC, Ren K, Balazinska M, Howe B. Managing skew in Hadoop. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2013,36(1):24−33.

[36] Song H, Yin Y, Sun XH, Thakur R, Lang S. A segment-level adaptive data layout scheme for improved load balance in parallel file systems. In: Proc. of the 11th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing (CCGrid). Piscataway: IEEE Computer Society, 2011. 414−423. [doi: 10.1109/CCGrid.2011.26]

[37] Vernica R, Balmin A, Beyer KS, Ercegovac V. Adaptive MapReduce using situation-aware mappers. In: Proc. of the 15th Int'l Conf. on Extending Database Technology. New York: ACM Press, 2012. 420−431. [doi: 10.1145/2247596.2247646]

[38] Lin WW. An improved data placement strategy for Hadoop. Journal of South China University of Technology (Natural Science Edition), 2012,40(1):152−158 (in Chinese with English abstract).

[39] Ye X, Huang M, Zhu D, Xu P. A novel blocks placement strategy for Hadoop. In: Proc. of the ACIS-ICIS. Piscataway: IEEE Computer Society, 2012. 3−7. [doi: 10.1109/ICIS.2012.11]

[40] Hsiao HC, Chung HY, Shen H, Chao YC. Load rebalancing for distributed file systems in clouds. IEEE Trans. on Parallel and Distributed Systems, 2013,24(5):951−962. [doi: 10.1109/TPDS.2012.196]

[41] Chiwande VN, Tayal AR. An approach to balance the load with security for distributed file system in cloud. In: Proc. of the 2014 Int'l Conf. on Electronic Systems, Signal Processing and Computing Technologies (ICESC). Washington: IEEE Computer Society, 2014. 266−270. [doi: 10.1109/ICESC.2014.51]

[42] Kwon YC, Balazinska M, Howe B, Rolia J. Skewtune: Mitigating skew in mapreduce applications. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2012. 25−36. [doi: 10.1145/2213836.2213840]

[43] Ibrahim S, Jin H, Lu L, Wu S, He BS, Qi L. Leen: Locality/Fairness-Aware key partitioning for mapreduce in the cloud. In: Proc. of 2010 IEEE the 2nd Int'l Conf. on Cloud Computing Technology and Science (CloudCom). Piscataway: IEEE Computer Society, 2010. 17−24. [doi: 10.1109/CloudCom.2010.25]

[44] Kolb L, Thor A, Rahm E. Load balancing for mapreduce-based entity resolution. In: Proc. of the 2012 IEEE 28th Int'l Conf. on Data Engineering (ICDE). Piscataway: IEEE Computer Society, 2012. 618−629. [doi: 10.1109/ICDE.2012.22]

[45] Slagter K, Hsu CH, Chung YC, Zhang DQ. An improved partitioning mechanism for optimizing massive data analysis using MapReduce. The Journal of Supercomputing, 2013,66(1):539−555. [doi: 10.1007/s11227-013-0924-9]

[46] Fan LY, Gao B, Sun X, Zhang F, Liu ZY. Improving the load balance of MapReduce operations based on the key distribution of pairs. arXiv preprint 1401.0355, 2014.

[47] Gufler B, Augsten N, Reiser A, Kemper A. Handing data skew in MapReduce. In: Proc. of the 1st Int'l Conf. on Cloud Computing and Services Science, Vol.146. 2011. 574−583.

[48] Fan YQ, Wu WG, Cao HJ, Zhu H, Wei W, Zheng PF. LBVP: A load balance algorithm based on virtual partition in Hadoop cluster. In: Proc. of the 2012 IEEE Asia Pacific on Cloud Computing Congress (APCloudCC). Washington: IEEE Computer Society, 2012. 37−41. [doi: 10.1109/APCloud CC.2012.6486508]

[49] Xie J, Yin S, Ruan XJ, Ding ZY, Tian Y. Improving mapreduce performance through data placement in heterogeneous Hadoop clusters. In: Proc. of the 2010 IEEE Int'l Symp. on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW). Piscataway: IEEE Computer Society, 2010. 1−9. [doi: 10.1109/IPDPSW.2010.5470880]

[50] Liu Y, Li MZ, Alham NK, Hammoud S, Ponraj M. Load balancing in MapReduce environments for data intensive applications. In: Proc. of the 2011 8th Int'l Conf. on Fuzzy Systems and Knowledge Discovery (FSKD), Vol.4. Piscataway: IEEE Computer Society, 2011. 2675−2678. [doi: 10.1109/FSKD.2011.6020071]

[51] Fan YQ, Wu WG, Cao HJ, Zhu H, Zhao X, Wei W. A heterogeneity-aware data distribution and rebalance method in Hadoop cluster. In: Proc. of the 2012 7th ChinaGrid Annual Conf. (ChinaGrid). Washington: IEEE Computer Society, 2012. 176−181. [doi: 10.1109/ChinaGrid.2012.22]

[52] Ahmad F, Chakradhar ST, Raghunathan A, Vijaykumar TN. Tarazu: Optimizing mapreduce on heterogeneous clusters. ACM SIGARCH Computer Architecture News, 2012,40(1):61−74. [doi: 10.1145/2189750.2150984]

[53] Gandhi R, Xie D, Hu YC. PIKACHU: How to rebalance load in optimizing mapreduce on heterogeneous clusters. In: Proc. of the 2013 USENIX Annual Technical Conf. (USENIX ATX 2013). Berkeley: USENIX Association, 2013. 61−66.

[54] Fischer MJ, Su XY, Yin YT. Assigning tasks for efficiency in Hadoop. In: Proc. of the 22nd ACM Symp. on Parallelism in Algorithms and Architectures. New York: ACM Press, 2010. 30−39. [doi: 10.1145/1810479.1810484]

[55] Groot S, Kitsuregawa M. Jumbo: Beyond MapReduce for workload balancing. In: Proc. of the 36th Int'l Conf. on Very Large Data Bases. Singapore: VLDB Endowment, 2010. 7−12.

[56] Wang Y, Xing J, Xiong J, Meng D. A load-aware data placement policy on cluster file system. In: Proc. of the Network and Parallel Computing. Berlin, Heidelberg: Springer-Verlag, 2011. 17−31. [doi: 10.1007/978-3-642-24403-2_2]

[57] Zaharia M, Konwinski A, Joseph AD, Katz R, Stoica I. Improving MapReduce performance in heterogeneous environments. In: Proc. of the 8th USENIX Conf. on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2008. 29−42.

[58] Guo ZH, Fox G, Zhou M. Investigation of data locality in mapreduce. In: Proc. of the 2012 12th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing (CCGrid 2012). Piscataway: IEEE Computer Society, 2012. 419−426. [doi: 10.1109/CCGrid.2012.42]

[59] Zheng P, Cui LZ, Wang HY, Xu M. A data placement strategy for data-intensive applications in cloud. Chinese Journal of Computers, 2010,33(8):1472−1480 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2010.01472]

[60] Yuan D, Yang Y, Liu X, Chen JJ. A data placement strategy in scientific cloud workflows. Future Generation Computer Systems, 2010, 26(8):1200−1214. [doi: 10.1016/j.future.2010.02.004]

[61] Agarwal S, Dunagan J, Jain N, Saroiu S, Wolman A. Volley: Automated data placement for geo-distributed cloud services. In: Proc. of the 7th USENIX Conf. on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2010. 17−32.

[62] Palanisamy B, Singh A, Liu L, Jain B. Purlieus: Locality-aware resource allocation for MapReduce in a cloud. In: Proc. of the 2011 Int'l Conf. for High Performance Computing, Networking, Storage and Analysis (SC 2011). New York: ACM Press, 2011. 1−11. [doi: 10.1145/2063384.2063462]

[63] Liu SW, Kong LM, Ren KJ, Song JQ, Deng KF, Leng HZ. A two-step data placement and task scheduling strategy for optimizing scientific workflow performance on cloud computing platform. Chinese Journal of Computers, 2011,34(11):2121−2130 (in Chinese with English abstract).

[64] Wu SC, Shuai X, Chen L, Ye L, Yuan BW. A replica pre-placement strategy based on correlation analysis in cloud environment. In: Proc. of the 1st Int'l Workshop on Cloud Computing and Information Security. Paris: Atlantis Press, 2013. 541−544.

[65] Abad CL, Lu Y, Campbell RH. DARE: Adaptive data replication for efficient cluster scheduling. In: Proc. of the 2011 IEEE Int'l Conf. on Cluster Computing (CLUSTER). New York: IEEE, 2011. 159−168. [doi: 10.1109/CLUSTER.2011.26]

[66] Eltabakh MY, Tian YY, Özcan F, Gemula R, Krettek A, McPherson J. CoHadoop: Flexible data placement and its exploitation in Hadoop. Proc. of the VLDB Endowment, 2011,4(9):575−585. [doi: 10.14778/2002938.2002943]

[67] Golab L, Hadjieleftheriou M, Karloff H, Saha B. Distributed data placement to minimize communication costs via graph partitioning. In: Proc. of the 26th Int'l Conf. on Scientific and Statistical Database Management. New York: ACM Press, 2014. 20. [doi: 10. 1145/2618243.2618258]

[68] Lin YT, Agrawal D, Chen C, Ooi BC, Wu S. Llama: Leveraging columnar storage for scalable join processing in the MapReduce framework. In: Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2011. 961−972. [doi: 10.1145/1989323.1989424]

[69] Floratou A, Patel JM, Shekita EJ, Tata S. Column-Oriented storage techniques for MapReduce. Proc. of the VLDB Endowment, 2011,4(7):419−429. [doi: 10.14778/1988776.1988778]

[70] Seo S, Jang I, Woo K, Kim I, Kim JS, Maeng S. HPMR: Prefetching and pre-shuffling in shared MapReduce computation environment. In: Proc. of the IEEE Int'l Conf. on Cluster Computing and Workshops (CLUSTER 2009). New York: IEEE, 2009. 1−8. [doi: 10.1109/CLUSTR.2009.5289171]

[71] Hammoud M, Sakr MF. Locality-Aware reduce task scheduling for MapReduce. In: Proc. of the 2011 IEEE 3rd Int'l Conf. on Cloud Computing Technology and Science (CloudCom). Piscataway: IEEE Computer Society: IEEE, 2011. 570−576. [doi: 10. 1109/CloudCom.2011.87]

[72] Gu T, Zuo C, Liao Q, Yang YL, Li T. Improving MapReduce performance by data prefetching in heterogeneous or shared environments. Int'l Journal of Grid and Distributed Computing, 2013,6(5):71−82. [doi: 10.14257/ijgdc.2013.6.5.07]

[73] Jin H, Yang X, Sun XH, Raicu I. Adapt: Availability-Aware mapreduce data placement for non-dedicated distributed computing. In: Proc. of the IEEE 32nd Int'l Conf. on Distributed Computing Systems (ICDCS). Piscataway: IEEE Computer Society, 2012. 516−525. [doi: 10.1109/ICDCS.2012.48]

[74] Xie J, Tian Y, Yin S, Zhang J, Ruan XJ, Qin X. Adaptive preshuffling in Hadoop clusters. Procedia Computer Science, 2013,6(2): 79−92. [doi: 10.1016/j.procs.2013.05.422]

[75] Yong M, Garegrat N, Mohan S. Towards a resource aware scheduler in Hadoop. In: Proc. of the 2009 IEEE Int'l Conf. on Web Services. Washington: IEEE Computer Society, 2009. 102−109.

[76] Ananthanarayanan G, Agarwal S, Kandula S, Greenberg A, Stoica I, Harlan D, Harris D. Scarlett: Coping with skewed content popularity in mapreduce clusters. In: Proc. of the sixth Conf. on Computer Systems. New York: ACM Press, 2011. 287−300. [doi: 10.1145/1966445.1966472]

[77] Tan J, Meng XQ, Zhang L. Coupling scheduler for mapreduce/Hadoop. In: Proc. of the 21st Int'l Symp. on High-Performance Parallel and Distributed Computing. New York: ACM Press, 2012. 129−130. [doi: 10.1145/2287076.2287097]

[78] Tan J, Meng XQ, Zhang L. Coupling task progress for mapreduce resource-aware scheduling. In: Proc. of the IEEE INFOCOM. Piscataway: IEEE, 2013. 1618−1626. [doi: 10.1109/INFCOM.2013.6566958]

[79] Wei QS, Veeravalli B, Gong BZ, Zeng LF, Feng D. CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster. In: Proc. of the 2010 IEEE Int'l Conf. on Cluster Computing (CLUSTER). New York: IEEE, 2010. 188−196. [doi: 10.1109/CLUSTER. 2010.24]

[80] Sun DW, Chang GR, Gao S, Jin LZ, Wang XW. Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. Journal of Computer Science and Technology, 2012,27(2):256−272. [doi: 10.1007/s11390-012-1221-4]

[81] He YQ, Lee RB, Huai Y, Shao Z, Jain N, Zhang XD, Xu ZW. RCFile: A fast and space-efficient data placement structure in MapReduce- based warehouse systems. In: Proc. of the 2011 IEEE 27th Int'l Conf. on Data Engineering (ICDE). Piscataway: IEEE Computer Society, 2011. 1199−1208. [doi: 10.1109/ICDE.2011.5767933]

[82] Ailamaki A, DeWitt DJ, Hill MD, Skounakis M. Weaving relations for cache performance. In: Proc. of the 27th Int'l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 2001. 169−180.

附中文参考文献:

[1] 孟小峰,慈祥.大数据管理:概念、技术与挑战.计算机研究与发展,2013,50(1):146−169.

[6] 王意洁,孙伟东,周松,裴晓强,李小勇.云计算环境下的分布存储关键技术.软件学报,2012,23(4):962−986. http://www.jos.org.cn/1000-9825/4175.htm [doi: 10.3724/SP.J.1001.2012.04175]

[13] 宋杰,李甜甜,闫振兴,那俊,朱志良.一种云计算环境下的能效模型和度量方法.软件学报,2012,23(2):200−214. http://www.jos.org.cn/1000-9825/4144.htm [doi: 10.3724/SP.J.1001.2012.04144]

[14] 廖彬,于炯,张陶,杨兴耀.基于分布式文件系统 HDFS 的节能算法.计算机学报,2013,36(5):1047−1064.

[15] 张桂刚,李超,邢春晓.一种云环境下的绿色计算模型.小型微型计算机系统,2013,34(5):1016−1020.

[24] 廖彬,于炯,孙华,年梅.基于存储结构重配置的分布式存储系统节能算法.计算机研究与发展,2013,50(1):3−18.

[38] 林伟伟.一种改进的 Hadoop 数据放置策略.华南理工大学学报(自然科学版),2012,40(1):152−158.

[59] 郑湃,崔立真,王海洋,徐猛.云计算环境下面向数据密集型应用的数据布局策略与方法.计算机学报,2010,33(8):1472−1480. [doi: 10.3724/SP.J.1016.2010.01472]

[63] 刘少伟,孔令梅,任开军,宋君强,邓科峰,冷洪泽.云环境下优化科学工作流执行性能的两阶段数据放置与任务调度策略.计算机学报,2011,34(11):2121−2130.

荀亚玲(1980－),女,山西临汾人,博士生,讲师,主要研究领域为数据挖掘,并行计算.

张继福(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据挖掘,并行与分布式计算,人工智能.

秦啸(1974－),男,博士,副教授,博士生导师,主要研究领域为并行与分布式系统,存储系统,容错和性能评估.