基于LATE的Hadoop数据局部性改进调度算法

李丽英 唐 卓 李仁发
(湖南大学计算与通信学院 长沙 410082)

摘 要 调度问题是目前云计算研究中的热点问题,其目的是如何协同云计算资源,使其得到充分合理的利用。数据局部性是特定云平台Hadoop的主要特性之一。针对该特性,在Hadoop原有调度算法LATE的基础上提出了一种基于数据局部性的改进算法,以解决数据局部性带来的慢任务备份执行时读取数据要占用大部分时间而影响其处理速率的问题。最后,对该算法进行了实验及性能分析,并验证了算法在提高任务的响应时间和整个系统吞吐率方面有很大改进。

关键词 Hadoop,MapReduce,LATE,数据局部性
中图法分类号 TP393.03 **文献标识码** A

New Improvement of the Hadoop Relevant Data Locality Scheduling Algorithm Based on LATE

LI Li-ying TANG Zhuo LI Ren-fa
(School of Computer Science and Communication, Hunan University, Changsha 410082, China)

Abstract In the present, scheduling problem is a hot cloud computation research issues, and the purpose is to coordinate the cloud computation resources to be fully rational use. Data locality is one of the main properties in the particular cloud platform for Hadoop. The paper discussed the property, proposed a new improvement of the Hadoop relevant data locality scheduling algorithm based on LATE. The algorithm mainly soves the bakeup of slow task performance problem which arises during the implementation of data read, taking most of the time and envently influencing its processing speed. Finally, carried on experiment to the algorithm and analyzed the funcation, verified the algorithm to improve the response time and the whole system throughput.

Keywords Hadoop, MapReduce, LATE, Date locality

1 引言

Hadoop是最近几年发展比较成熟的云计算[1]平台之一,其技术已经在互联网领域得到广泛的应用,同时得到研究界的普遍关注。Hadoop主要由HDFS(Hadoop Distributed File System)和MapReduce引擎两个部分构成[2]。MapReduce主要用于并行任务处理,大规模的并行处理使得MapReduce的任务调度变得尤为重要。Hadoop是在假定了很多条件的基础上解决任务调度问题,例如假设所有的节点具有相同的性能、任务执行是以不变的速度进行的、自带的备份任务执行策略并不消耗任何资源。这些假设在异构且资源共享的云环境下很多是不成立的,其必然会影响到平台的调度性能,尤其已有MapReuce调度大多数都没有根据Hadoop平台的特殊性做具体问题的分析以及提出解决问题的办法。

Hadoop作为一个处理海量数据的云计算平台,会有大量的数据输入及输出,网络带宽是该系统中最稀缺的资源。此外,在分布式系统中任务分配与任务所需数据的存储很可能不在同一个节点[3],处理任务的节点应该尽可能包含需要的数据,因此数据局部性[4]是该平台最重要的特性之一。数据局部性优化问题最开始是从Active_disks[5,6]等技术研究中产生的,把计算尽量放在靠近本地硬盘中,以减少数据传输带来的I/O和网络消耗。网格系统调度器中[7],由于它主要是处理CPU-密集型作业,数据局部性问题也只是基于地理位置层次上的探讨。据统计,一个工作的输入数据存储节点在集群整个节点数中所占的比例将决定相应任务会在本地数据节点中进行处理的比例。Hadoop面临的数据局部性问题异常突出,产生这些问题的主要原因是Hadoop集群中的对等网路带宽相比一个节点上硬盘的总带宽要小得多。Hadoop集群资源又是共享的,这样节点之间传输数据时是双向通信,大量数据的传输会占用大部分网络资源,从而影响整个系统的吞吐率,最终影响任务的响应时间。

本文主要研究MapReduce的任务调度问题,分析现有的MapReduce已采用的任务算法,最后针对LATE算法[8]的不足进行改进。MapReduce最初采用的调度策略是简单的FIFO[9]、随后相应的改进有局部优化策略[4]、按存储需求调度算法CPAC(Compute Phase Admission Control)算法、FAIR[10]算法、LATE算法(Longest Approximate Time to end)以及异构环境下自适应的Map-Reduce调度算法[11]。

到稿日期:2010-12-14 返修日期:2011-02-26 本文受国家工信部核高基项目(2009ZX01038-001)资助。
李丽英(1984—),女,硕士,主要研究方向为云计算,E-mail:xxxturry@yahoo.com.cn;唐 卓(1981—),男,博士,讲师,主要研究方向为云计算和信息安全;李仁发(1957—),男,教授,博士生导师,主要研究方向为嵌入式和信息网络。

HOD[12]算法是给用户请求一个相当于私有集群的节点进行任务处理，该节点是固定的。但是，Hadoop是采用分布式存储文件，数据存储在系统的各个节点上，此算法会因为数据的局部性问题导致大量的数据跨机架传输，从而严重影响任务的处理时间、系统的吞吐率和资源利用率；LATE算法是备份任务执行的调度策略，MapReduce中的备份任务执行策略的思想起源于配置管理[13]以及信息收集[14]等技术，LATE就是考虑在尽可能不影响任务响应时间的情况下怎样备份执行一些进度比较慢的Map任务和Reduce任务。异构环境下自适应的Map-Reduce调度是在LATE算法的基础上提出的一种改进算法，核心思想是基于历史信息动态调整Map和Reduce任务各阶段的时间比例，找出真正需要启动备份任务执行的任务。

本文第2节分描述MapReduce任务调度LATE算法；第3节分析LATE算法存在的主要问题，并根据LATE算法存在的具体问题提出改进思路；第4节对改进的LATE算法进行性能实验评估；最后总结全文。

2 MapReduce任务调度LATE算法

Hadoop中一个工作被划分成若干个任务并行处理，在其它的任务已经完成时，部分任务由于处理节点故障或异构环境下节点性能的不同等原因而进展缓慢，这时系统就会启用备份任务机制，重新执行那些进展缓慢的任务。备份任务机制的主要目的是最小化任务响应时间及提高系统的吞吐率。LATE算法结合其它分布式系统备份执行策略[15]，不让执行过多的没必要的备份任务，以达到提高整个系统吞吐率及任务响应时间的目的。

LATE算法跟Hadoop一样采用分片策略来表示任务的进度，如图1所示。map任务主要是以输入数据的读取作为任务进度的标识，reduce任务的进度分片包括3个阶段：copy阶段、sort阶段以及reduce执行阶段，3阶段各占1/3。

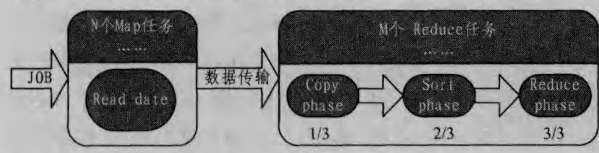


图1 MapReduce中任务进度表示图

LATE开始备份任务的原则是：根据任务的进度情况，推算出任务完成还需要多少时间，LATE优先备份执行那些即将最快完成的任务。LATE算法设定了以下参数：

(1) SpeculationCap：同一时刻备份任务执行数；

(2) SlowTaskThreshold：任务的进程速率与该参数进行比较，用于判断任务是否需要备份执行；

(3) SlowNodeThreshold：判断节点处理任务快慢的阈值，备份任务放置在快节点中进行处理。

$$\text{Progressrate}=\text{ProgresScore}/t \quad (1)$$

$$T=(1-\text{ProgressScore})/\text{Progressrate} \quad (2)$$

式(1)中t是任务已经执行了的时间，ProgresScore是任务按照分片策略得到的进程分，Progressrate是进程速率。式(2)中T是最终要得到的任务完成还需要的时间。

LATE算法工作流程是：如果有一个节点请求新的任务，并且整个Hadoop集群系统中备份任务执行数量小于SpeculationCap，则执行以下步骤：

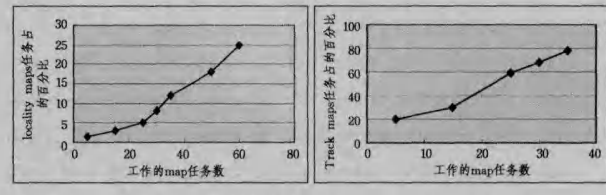
①判断节点的快慢，如果低于SlowNodeThreshold，则忽略这个请求，结束这个算法，否则继续执行；

②对正在执行的任务(不包括已经在备份执行的任务)进行排序，按照式(1)以及式(2)计算任务完成还需要多少时间，从低到高对任务进行排序；

③重新备份执行低于SlowTaskThreshold阈值并且在上面排序排在第二步排序排在首位的任务。

3 基于数据局部性对LATE算法的改进

LATE算法最主要的缺陷是在特定的云计算平台Hadoop中没有考虑数据局部性问题。它假定了大部分的map任务都是在本地数据节点中执行，同时完全没考虑reduce任务是否本地读取数据执行。根据Facebook使用Hadoop的情况统计分析，拥有1至25个map任务的小工作只能达到5%的任务在所需存储数据的本地节点(数据局部性)上执行，如图2(a)所示；59%的任务在所需存储数据的本地机架上执行，如图2(b)所示。因此，需要对LATE算法在这数据局部性上进行优化，进一步提高算法的性能，提高整个集群资源的利用率。



(a)节点locality与工作大小关系图 (b)机架locality与工作大小关系图

图2

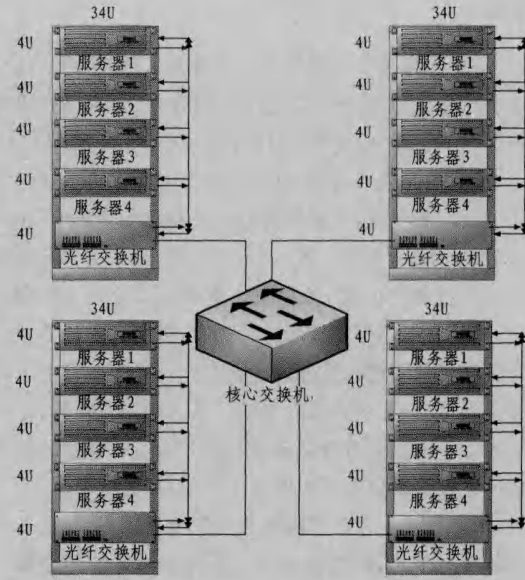


图3 Hadoop集群部署图

改进LATE算法的主要目的是通过考虑数据局部性问题来提高系统的吞吐率及任务的响应时间。Hadoop中节点按机架分配。Hadoop简单集群部署图如图3所示，每个机架典型地部署有40个节点。机架之间以一个交换机相连接，形成一个Hadoop系统。由于跨地域或者多重网络的影响，机

架内部数据的传输速率远远高于机架之间的数据传输速率。map 任务的进程速率主要由数据的输入速率决定，reduce 任务的输入数据是 map 任务的输出中间结果。但 map 任务的输出数据都存储在执行节点的本地磁盘上，因此 reduce 任务大部分时间都花在 copy 数据的阶段。如果 map 任务和 reduce 任务的输入数据都在执行节点或者执行机架上，则必定会在很大程度上提高任务的进程速率。

本文在改进策略中继续沿用 LATE 算法设定的参数，首先通过计算任务完成还需要多少时间的方式来找到最需要备份执行的任务。在找到备份任务执行过程中并不只是按照整个集群中执行的总任务排序，而是首先考虑本机机架上是否有需要备份执行的任务。如果没有，则对其它机架上的任务进行排序，最终找到合适的、需要的备份执行的任务。具体算法的流程如图 4 所示。备份算法的执行过程描述如下：

如果有一个节点请求新的任务，并且整个 Hadoop 集群系统中备份任务执行数量小于 SpeculationCap，则往下执行：

①判断节点的快慢，如果低于 SlowNodeThreshold，则忽略这个请求，结束这个算法，否则继续执行；

②根据请求任务节点所在的机架，对该机架上正在执行的任务（除了已经备份执行的任务）在计算完成还需要多少时间后，从低到高进行排序，如果没有低于 SlowTaskThreshold，则执行⑤，否则执行③；

③对其它机架上正在执行的任务计算完成还需要多少时间后，从低到高进行排序，选择排在最前并且低于 SlowTaskThreshold 的任务备份执行；

④在该队列中寻找是否有数据存储在请求节点并且低于 SlowTaskThreshold 的任务，如果有则执行，否则从该队列中找到排在首位并且低于 SlowTaskThreshold 的任务执行；

⑤结束。

从整个算法流程图（见图 4）可以看出，该算法首先考虑的是数据相关性的慢任务，在没有数据相关的任务时，则按 LATE 算法找到即将最快完成的任务。此外，算法对集群中的所有正在执行的任务进行两次排序。对本地机架上的任务进行计算及排序后，如果在本地机架上找到数据相关的任务，则不需要对整个集群中所有任务排序。这样做在一定程度上减少了该算法的计算任务，提高了算法的效率。

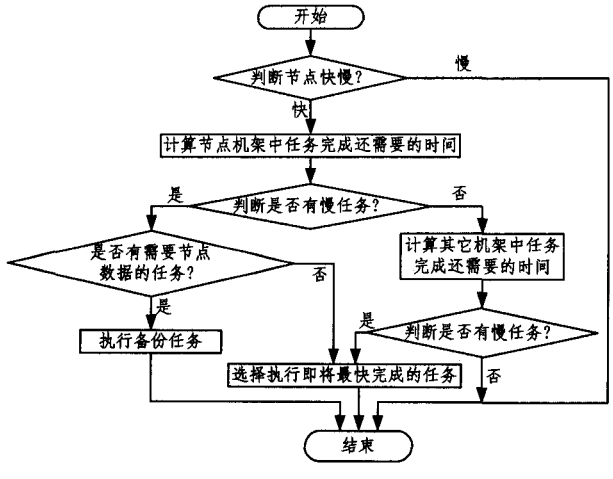


图 4　算法流程图

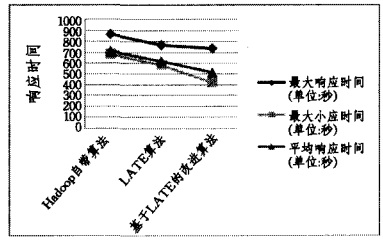
4　实验结果及性能分析

实验在本地搭建的 Hadoop 平台中完成，Hadoop 版本为 0.20.1，具体的实验平台配置如表 1 所列。为了测试算法的性能，使得跨机架读取数据的几率一样，设定了 3 个机架，每个机架上的总主机数跟总虚拟机数是一样的。为了体现 Hadoop 平台的异构性，在主机中设定了不同的虚拟机数。每台虚拟机中设定了两个 map 任务槽和两个 reducer 任务槽。

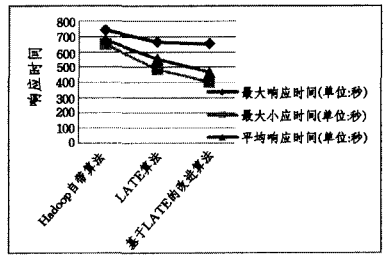
表 1　Hadoop 集群配置表

机架	虚拟机/主机(个)/(台)	主机数(台)	虚拟机数(个)
机架 1	2 个/台	1	2
	3 个/台	1	3
机架 2	2 个/台	1	2
	3 个/台	1	3
机架 3	2 个/台	1	2
	3 个/台	1	3

实验中首先选取的任务是 Sort（排序）任务和 Grep（字符串查找）任务，这两个任务是 Hadoop 平台自带的任务。选取这两个任务的原因是它们在 map 任务完成后会涉及大量数据的读取作为 reduce 任务的输入数据，跨机架读取数据的概率会比较高。因此，可以明显地看到在考虑与不考虑数据局部性的 LATE 调度器之间的性能差异；第二，实验中设定了 3 个已经在文献[8]中论证的最佳参数值，分别是 SpeculationCap 为 20%、SlowTaskThreshol 为 25%、SlowNodeThreshold 为 25%。第三，采用 Hadoop 自身带的推测执行算法、LATE 算法以及改进的 LATE 算法，这 3 者之间进行比较。第四，为了保证实验数据的有效分析，设定每个工作运行不少于 5min，每个工作重复 5 遍，取平均性能最好情况下以及最坏情况下进行分析。



(a) Sort 工作在 3 种调度算法下的响应时间对比



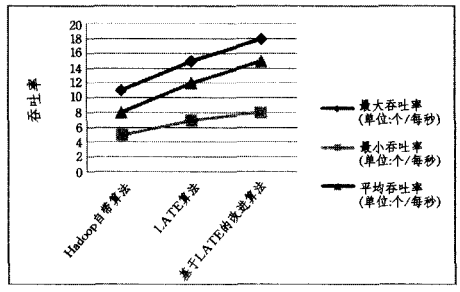
(b)Grep 工作在 3 种调度算法下的响应时间对比

图 5

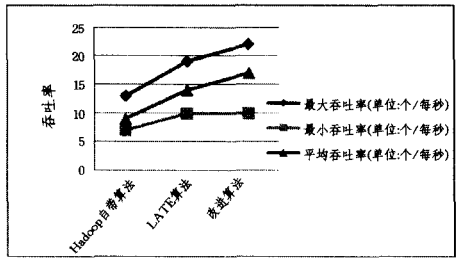
图 5 是单个 Sort 工作和单个 Grep 工作在 3 种算法下的工作响应时间的比较。在最好的情况下，基于数据局部性改进的 LATE 算法会远远高于其它两种算法，主要原因是需要备份执行任务的数据都在申请任务节点的本地机架上，不需要跨机架读取数据。而在最坏的情况下，改进的算法跟 LATE 算法相差不大，这是因为绝大部分备份任务的数据都

要跨机架读取，甚至在有的时候 LATE 算法还会略优于改进的算法。主要原因是改进的算法要进行两个排序工作，还要首先尽量在本地机架上找备份任务的执行，这会浪费些时间。但是，在平均性能方面，改进的算法还是比较明显的优势。

图 6 描述了多个 sort 工作和多个 Grep 工作在 3 种算法下的系统吞吐率分析。受搭建平台规模的局限性，整个系统吞吐率的值比较小。从图 6 中可以看到数据局部性对系统吞吐率的影响。在绝大部分备份任务执行都需要跨机架读取数据时，基于数据局部性的改进比其它算法提高了系统吞吐率将近 150%，这充分说明了大规模跨机架读取数据会占有大部分的稀缺资源——网络带宽，造成机架之间的网络阻塞，无法给其它任务分配带宽，最终影响整个系统的吞吐率。



(a)Sort 工作在 3 种调度算法下系统吞吐率对比



(b)Grep 工作在 3 种调度算法下系统吞吐率对比

图 6

结束语 本文在 Hadoop 平台中的 MapReduce 调度算法之一 LATE 算法的基础上，充分考虑了 Hadoop 这个平台架构的特性，提出了基于 LATE 的有关数据局部性的改进调度算法，解决了大规模的跨机架读取数据会严重影响任务处理速度的问题。最后，还对改进的算法在本地搭建的 Hadoop 平台上进行了实验测试分析。实验数据表明，本算法在很大程度上减小了任务的响应时间，提高了系统的吞吐率，达到了提高 Hadoop 平台资源利用率的目的。

虽然本算法相比其它算法有其明显的优势，但也有局限性。一是在用改进的算法选择备份任务执行时，要对整个系统进行两次排序：一次是本机架内部任务，另一次是其它机架的任务。这样会消耗一些时间，并且有可能会因为跨机架的原因跳过最需要备份执行的任务，因此因考虑用排队论的原理对整个机架上所有的任务统一进行一次排序，而其它机架上的任务由于跨机架读取数据的原因，将其乘以一个权值后参与到整个排序当中，估计这样性能会更好。只是这个权值比较难以取舍，希望以后能从这方面得以改进。另外，由于实验条件的影响，只能在本地的搭建一个比较小的 Hadoop 集群中进行试验，希望能在更大的实验平台中或者是应用平台中进行进一步的测试性能分析。

参 考 文 献

[1] Vaquero L M, Rodero-Merino L, Caceres J, et al. A break in the cloud: Towards a Cloud Definition[J]. ACM SIGCOMM Computer Communication Review, 2009, 39(1): 50-55

[2] Vaquero L M, Rodero-Merino L, Caceres J, et al. A break in the cloud: Towards a Ckoud Definition[J]. ACM SIGCOMM Computer Communication Review, 2009, 39(1): 50-55

[3] Crovella M, Harchol-Balter M, Murta C D. Tasassignment in a distributed system: Improving performance by unbalancing load [M]. Measurement and Modeling of Computer Systems, 1998: 268-269

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

[5] Huston L, Sukthankar R, Wickremesinghe R, et al. Diamond: A storage architecture for early discard in interactive search[C]// Proceedings of the 2004 USENIX File and Storage Technologies FAST Conference. April 2004

[6] Riedel E, Faloutsos C, Gibson G A, et al. Active disks for large-scale data processing[C]// IEEE Computer. June 2001: 68-74

[7] Thain D, Tannenbaum T, Livny M. Distributed computing in practice: the Condor experience[J]. Concurrency and Computation-Practice and Experience, 2005, 17(2-4): 323-356

[8] Zahafia M, Konwinski A, Joseph A. Improving MapReduce Performance in Heterogeneous Environments [C]// Proc of the 8th Usenix Symp on Operating Systems Design and Implementation. 2008: 29-42

[9] HADOOP-3759: Provide ability to run memory intensive jobs without affecting other running tasks on the nodes[EB/OL]. https://issues.apache.org/jira/browse/HADOOP-3759

[10] Zaharia M, Borthakur D, Sarma J S, et al. Job Scheduling for Multi-user MapReduce Clusters[R]. EECS-2009-55. April 2009

[11] 陈全，邓倩妮. 异构环境下自适应的 Map—Reduce 调度[J]. 计算机工程与科学，2009，31(A1)

[12] Personal communication with Owen O' Malley and Arun C. Murphy of the Hadoop development team at Yahoo!

[13] Su Y, Attariyan M, Flinn J. AutoBash: improving conguration management with operating system causality analysis [C] // ACM SOSP. 2007

[14] Barish G. Speculative plan execution for information agents[D]. University of Southern California, Dec. 2003

[15] Nightingale E B, Chen P M, Flinn J. Speculative execution in a distributed file system[J]. ACM Trans. Comput. Syst., 2006, 24 (4): 361-392