DOI：10.7652/xjtuxb201502001

一种短作业环境下的延迟调度算法

刘强[1]，董小社[1]，朱正东[1]，王寅峰[2]
（1.西安交通大学电子与信息工程学院，710049，西安；2.深圳职业技术学院，518172，广东深圳）

摘要：针对短作业场景下 YARN 平台中延迟调度算法基于静态时间等待阈值，不能进行合理等待的问题，提出了一种云计算环境中基于本地性资源预测的延迟调度算法（locality resource forecast delay scheduling，LRFD）。该算法综合考虑短作业和资源可用性动态变化的特点进行任务调度，根据节点上任务的完成进度和作业未处理数据在集群中的分布状况预估作业的本地性资源信息，从而判断是否需要进行等待以提高系统性能，实现了对本地性资源的合理等待。实验结果表明：在短作业场景下，LRFD 算法的性能和稳定性均优于已有的延迟算法，作业性能平均提升约 10%，最大加速比可达 3 倍以上。

关键词：云计算；延迟调度算法；短作业
中图分类号：TP391 **文献标志码**：A **文章编号**：0253-987X(2015)02-0001-05

A Delay Scheduling Algorithm for Short Jobs

LIU Qiang[1]，DONG Xiaoshe[1]，ZHU Zhengdong[1]，WANG Yinfeng[2]
（1. School of Electronics and Information Engineering，Xi'an Jiaotong University，Xi'an 710049，China；
2. Shenzhen Institute of Information Technology，Shenzhen Guangdong，518172，China）

Abstract：A delay scheduling algorithm based on locality resource forecasting（LRFD）is proposed to address the unreasonable waiting problem generalized by the static time-wait threshold in delay scheduling algorithm of YARN platform for short jobs. The algorithm takes both the characteristics of short jobs and dynamic resource availability into consideration to assign tasks. It estimates local resources to make reasonable waiting according to both the task progress on the nodes and the unhandled splits distribution in the cluster of the job. Experimental results and comparison with the traditional delay scheduling algorithm show that LRFD gets a better stability and improves the performance about 10% for short jobs on average and achieves a maximum speedup up to three times.

Keywords：cloud computing；delay scheduling algorithm；short jobs

MapReduce[1]是 Google 于 2004 年提出的一种并行计算模型，其开源版本 Hadoop 被 Yahoo 和 Facebook 等公司所使用，并广泛应用于 Web 服务、数据挖掘和文本处理等领域。最近的研究表明，在 Yahoo 的 Hadoop 集群中，超过 80%的作业时间小于 10 min[2]，Facebook 和 Microsoft 也存在同样的情况[3]。国内例如淘宝，约 80%的作业的响应时间小于 4 min，任务执行时间小于 20 s[4]。因此，短作业环境下的作业调度策略对云平台的吞吐率和服务质量有着非常重要的意义。

收稿日期：2014-05-21。 作者简介：刘强（1985—），男，教授，博士生；董小社（通信作者），男，教授，博士生导师。 基金项目：国家自然科学基金资助项目（61173039）；国家自然科学基金青年基金资助项目（61202041）；国家高技术研究发展计划资助项目（2012AA01A306）；深圳市科技计划资助项目（JCYJ20120615101127404）。
网络出版时间：2014-12-11 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20141211.0849.003.html

目前,作为第2代Hadoop平台YARN(yet another resource negotiator)的资源调度器主要采用了延迟等待策略以提高数据本地性,没有考虑作业的特点以及云计算环境下资源可用性动态变化的特点。对于普通作业而言,由于其输入数据较大,传输时间长,为获取更好的本地性而进行短时间等待的策略能够提高作业的本地性和响应时间。但是对于短作业而言,由于其数据量小、任务执行时间短的特点,不合理的等待反而会影响短作业的性能,造成系统吞吐率下降。同时,静态的等待阈值并不能很好地适应系统中资源可用性的动态变化,进而影响系统性能。

针对短作业场景下原延迟算法的上述问题,本文提出了一种云计算环境中基于本地性资源预测的延迟调度算法。该方法综合考虑了短作业和资源可用性动态变化的特点进行作业调度,通过对子节点增加任务运行时状态监控,同时在主节点对本地性资源进行等待代价估算,实现了短作业情况下本地性资源的合理等待,可有效提升系统性能,同时算法性能具备良好的稳定性。

1 相关工作

随着集群规模和作业负载的增加,Hadoop1.X版本在可扩展性、可靠性以及性能等方面暴露出了严重不足。同时,随着新的数据密集型应用的计算框架不断涌现,例如Spark[5]、Imapla等,传统的作业管理系统例如PBS和Condor[6-7]不能满足可扩展性、可靠性、多种数据密集型计算框架并存等方面的需求,由此产生了新的资源管理平台,其典型代表有UCBerkely的Mesos[8]、Yahoo的YARN[9]等。

目前,在YARN中提供使用的调度器主要包括公平调度器和容量调度器[10]。公平调度器满足了多用户共享集群环境下的公平性需求,但是其存在的队列头和槽黏滞问题影响了数据本地性。基于此,加州大学伯克利分校的Zaharia等提出了Delay-Scheduling算法[11]。该算法采用延迟等待策略,在基本保证作业公平分享数据中心资源的同时,提高了数据本地性,其基本思想是,当一个空闲节点请求任务时,按照公平性原则调度队首作业,若该节点不包含该作业所需要的数据,则先调度其他作业而让队首作业等待,如果队首作业的等待时间超过设置的阈值,则立即调度队首作业而不再等待。与Delay相似的还有Quincy算法[3],它支持DAG模型定义作业,采用最小代价流法进行作业调度。此外,Khaled根据Yahoo集群中短作业具有的执行时间短、作业重执行的代价低的特点,提出了Piranha系统[12],将短作业的map任务结果从内存直接通过网络传递给reduce任务,避免了原有的心跳通信和磁盘读写代价。

延迟算法改进了本地性,但在资源可用性动态变化的情况下,其对所有作业统一等待时间,难以适应短作业数据量小的特点。由于短作业具有数据块小、传输时间短的特点,为了等待更好本地性所付出的时间代价可能高于数据传输的时间代价,进而影响短作业的响应时间。随着任务的执行,空闲计算节点所含本地任务的数量不断减少,而延迟调度算法每次都会等待一个固定的时间,这样不但影响效率,也会影响后期任务的执行。

本文提出的基于本地性资源预测的延迟调度(locality resource forecast delay scheduling, LRFD)算法,通过监控任务执行状态以获取本地性资源的变化情况,进而估算等待代价,避免了原延迟算法中粗糙地针对所有节点的请求进行固定时长等待的缺陷,从而合理等待调度。

2 问题分析

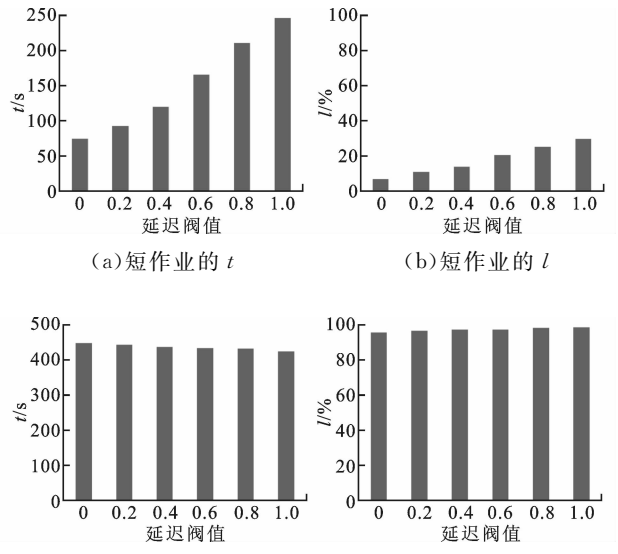
为了测试短作业和普通作业在原算法下的响应时间t(s)和本地性任务比率l(%),参考短作业的特点[2,11-12],本文使用YARN自带的短作业测试用例MrBench和普通作业的用例WordCount,分别测试短作业和普通作业的响应时间随延迟阈值变化的趋势。其中:MrBench的map数量为1 000,输入行数量为80 000(map的平均任务时间为2 s);WordCount的数据块大小设为256 MB(map的平均任务执行时间为38 s),数据量为100 GB;集群规模均为25个节点。

普通作业的测试结果如图1c和图1d所示,随着延迟阈值的增大,其性能和本地性均表现出上升趋势,但是对于短作业(图1a和图1b),随着延迟阈值的增大,在本地性任务比率增大的同时,其作业的响应时间也明显上升,可见延迟算法并不一定能带来性能上的改进,不合理的等待同样会造成性能下降。因此,需要一种新的延迟调度方法,根据作业和资源的可用性动态变化特点进行合理等待。

3 基于资源预测的延迟调度算法

3.1 LRFD算法思想及框架

非本地化计算数据和等待过久以获取更好的本



(a)短作业的 t　(b)短作业的 l

(c)普通作业的 t　(d)普通作业的 l

图 1　YARN 平台下短作业和普通作业的响应时间以及本地性任务比率随延迟阈值的变化趋势

地性都会浪费计算资源。如图 2 所示，当节点 1 的节点请求到达时，该节点并没有作业 1 所需的数据，而延迟算法在等待次数未达到阈值前，需要一直跳过新的任务请求，直到具有本地性的节点出现才进行任务分配。如果能够提前对系统中该作业的未处理数据块和空闲计算资源情况做出预测，判断出在未来某一时间段内是否会出现符合该作业要求的本地性资源以进行合理调度，则可避免集群资源浪费。

基于此，本文提出 LRFD 算法。如图 2 所示，当节点 1 请求到来时，LRFD 根据当前作业 1 的本地性资源列表进行判断，如果未来某一时间段内作业 1 有本地性资源到来，则调度作业 2 的任务给节点 1，否则应立即调度作业 1 任务给节点 1，避免其等待时间过长。

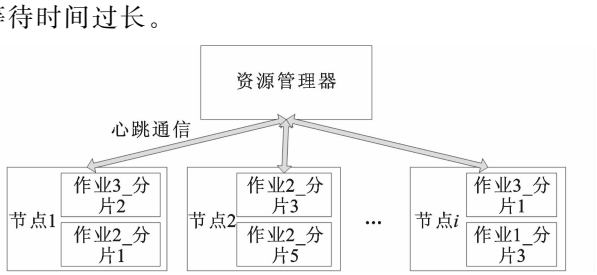


图 2　LRFD 调度示意图

LRFD 总体架构如图 3 所示，主要包括任务状态监控模块和本地性资源预测模块。任务监控模块负责监控节点上任务的进度和本地性资源信息，包含了一个远程过程调用(remote process call protocol，RPC)服务器，用于接收周期性的任务进度汇报。本地性资源预测模块用于对当前待调度作业的本地性资源进行预测判断，决定是否延迟当前任务。

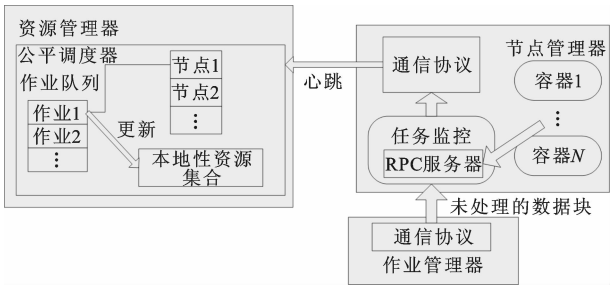


图 3　LRFD 总体架构

3.2　任务执行时间监控

为了描述需要，本文对系统进行如下定义：对于一个节点 i 上作业 J 正在运行的 map 任务，其已完成的进度为 P_m，则剩余进度为 $1-P_m$，设该任务已运行的时间为 T_m，该 map 任务剩余时间 T_l 可由式(1)[13]进行计算

$$T_l = (1-P_m)T_m/P_m \tag{1}$$

节点管理器(node manger，NM)对节点 i 上作业 J 正在执行的任务进行监控，主要包括：①作业管理器(application master，AM)在通过 NM 启动任务时，将该作业在此 NM 上未处理任务(一个任务对应一个未处理分片)集合记为 S_i；②当每次 NM 向资源管理器(resource manager，RM)发送心跳前，NM 需要统计下个周期内节点 i 将要结束的任务集合，记为 D_i，心跳周期记为 T_h，map 任务集合记为 Q，每个任务的剩余时间记为 T_l，则下个心跳周期内，节点 i 能够产生的作业 J 的本地性资源集合为

$$R_i = S_i \cap D_i\{D_i \in Q \mid T_l < T_h\} \tag{2}$$

NM 获取其上运行的作业的本地性资源集合后，通过心跳反馈给 RM 用以进行本地性资源预测。

3.3　本地性资源预测

LRFD 的调度流程如图 4 所示，RM 收到心跳

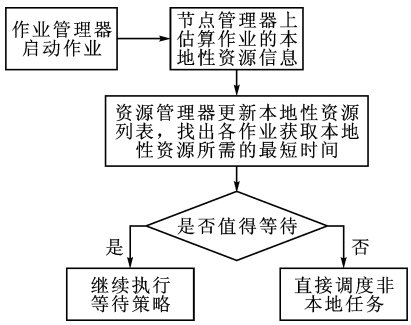


图 4　LRFD 调度流程

后,将对应的本地性资源信息经过中央的异步事件处理器传递给公平调度器,公平调度器更新其所有作业的本地性资源集合,各作业将其本地性资源按照剩余时间从大到小排序,找出最快结束的容器时间,设为 $T_{\min}$,即 $\min(R_i)$,见式(2),同时由于集群中网络带宽会随着系统负载的变化而变化,这里可根据作业的历史信息获取带宽,设 T_b 为一个数据块所耗费的传输时间,B_l 为一个数据块的数据量,B_h 为带宽,则

$$T_b = B_l / B_h \tag{3}$$

如果 $T_{\min} < T_b$,则说明延迟等待值得,反之则说明一个数据块传输时间内没有本地性资源到达,应立即调度非本地性任务。

LRFD 算法伪码如算法 1 所示。

算法 1 LRFD Scheduling

```
1  Node i:calculate_LocalResources();
2  when a heartbeat is received from Node i to RM
3  sort(apps);
4  for (app:appLeafQueue)
5    update(DelayContainersList);
6   for (priority p:app.priorities)
7   if (p.LocalityLevel=allowedLocalityLevel)
8    assign localTasks;//直接分派本地性任务
9    else minWaitTime ←
10    getMinWaitTime(DelayContainersList);
11    if (blockTransferTime < minWaitTime)
12    Then settleDownAllowedLocality Level(p);
13   assign unlocalTasks;//分派非本地性任务
14    else continue wait;
15    endif
16   endif
17   end for
18  end for
19  return
```

由算法 1 可知,若系统中正在运行的作业数为 n,任务分配函数的时间复杂度为 T,节点请求的优先级常数为 p,每个子节点上所能运行的最大任务数为 c,则 LRFD 算法的时间复杂度为 $O(\log n + np(c\log c + T)) \approx O(nT)$。

由前述公式可知,当系统中存在正在运行的作业 J 且其数据可分时,子节点上总会存在作业的未处理任务和正在执行的任务,而数据分片大小已知,因此总存在本地性资源集合 R,使得式(1)、式(2)、式(3)成立。可见本算法具备合理性,兼顾了作业和环境特点。

4 实验验证

本文面向 YARN 平台,设计了基于本地性资源预测的 LRFD 调度算法,并使用了 MrBench、Pi 计算、WordCount 和 Grep 等 YARN 的自带测试用例分别对 Delay[11] 和 LRFD 进行对比测试。实验集群由 25 台刀片节点构成。刀片节点的 CPU 为 2× Intel Xeon E5-2670,主频为 2.6 GHz,8 核处理器,内存为 32 GB,硬盘为 300 GB,采用1 000 M以太网互连,实验软件为 YARN 2.2.0 版本。

4.1 测试用例

本实验使用的测试用例包括 MrBench、Pi 计算、Grep 和 WordCount 等,延迟阈值分别设为 0、0.2、0.4、0.6、0.8、1,每组作业测试 3 次。测试中重点关注各测试用例中作业的平均完成时间和作业的本地化任务比率。测试中需要调整任务向监控器的汇报频度,否则可能出现由于短作业的任务执行时间较短而无法及时获取任务信息的情况。

MrBench 为 YARN 自带的短作业测试用例,用于模拟交互式作业。测试时输入数据行数设为 80 000,map 任务数量为 1 000。

Pi 计算使用蒙特卡洛方法估算 π 值,主要接受两个参数:map 任务的数量和每个 map 任务产生的样本点数量。测试时 map 任务数量为 1 500,样本点数量为 20 000。

WordCount 和 Grep 均为经典的 Hadoop 测试用例,WordCount 用于词频统计,Grep 用于在文本文件中进行字符串检索并统计其次数。输入数据由 YARN 自带的随机文本写入工具生成。以上 3 个测试用例输入数据量均为 2 GB,maps 任务数均为 1 052,数据块大小均为 2 MB(map 的平均任务时间为 2 s,作业执行时间均小于 2 min)。

同时为了测试普通作业场景下 LRFD 的性能,将 WordCount 和 Grep 的数据分块大小均设为 256 MB,输入数据由 YARN 自带的随机文本写入工具生成,大小均为 100 GB(map 的平均任务时间均大于30 s,作业执行时间均大于 5 min)。

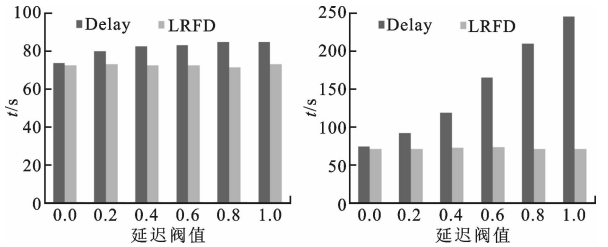
4.2 测试结果

测试结果如图 5 所示,在短作业场景下,随着延迟阈值的增大,延迟算法的等待时间代价逐渐增大,其性能呈下降趋势。在性能和稳定性方面,LRFD 均优于延迟算法,平均性能提升约 10%;在算法稳定性方面,LRFD 明显优于延迟算法,随着延迟阈值

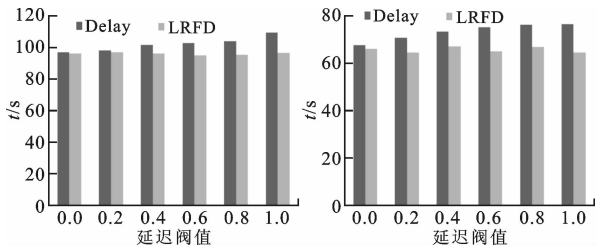
的增大表现出较为稳定的性能。

其中，MrBench 改进最大，如图 5b 所示，其加速比可达 3 倍以上，这是因为其数据分布节点较少，本地性比率很低（如图 6b 所示），因而由原延迟调度算法导致的等待损失较大，而 LRFD 算法能合理等待，因此性能提升较大。

本地性任务比率如图 6 所示，相比延迟算法，LRFD 在本地性方面略有下降（约占 5%），原因是 LRFD 算法减少了延迟算法中所需延迟，虽然本地性略有损失，但由于其数据量小，传输数据的时间代价小于等待的时间代价，因此性能反提升了约 10%，从另一方面验证了本算法的有效性。

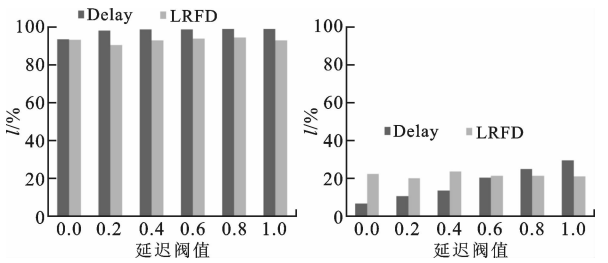


(a) Wordcount　　(b) MrBench

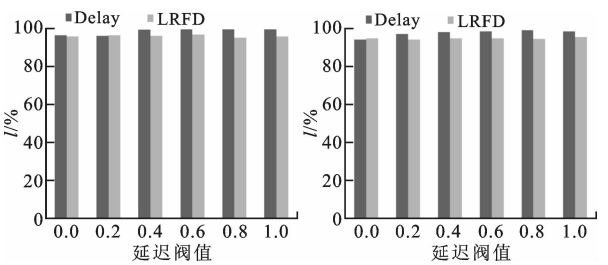


(c) Pi　　(d) Grep

图 5　WC、MrBench、Pi 和 Grep 的性能对比



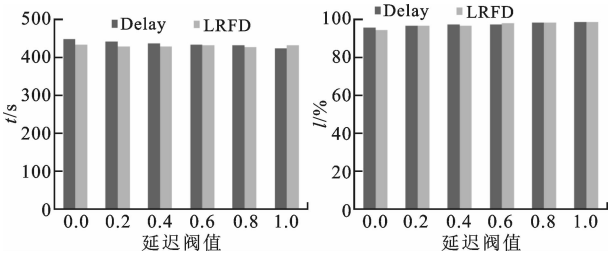
(a) Wordcount　　(b) MrBench



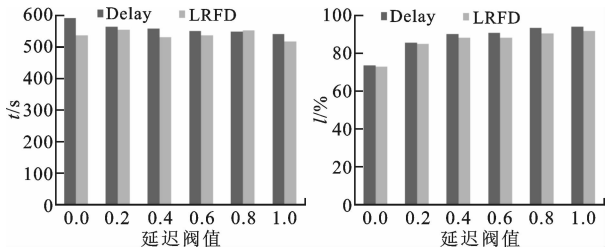
(c) Pi　　(d) Grep

图 6　WC、MrBench、Pi 和 Grep 的本地性比率对比

普通作业在 LRFD 算法下的性能和本地性表现如图 7 所示。同延迟算法相比，LRFD 的性能略有上升（如图 7a 和图 7c 所示，约上升 4.5%），本地性则与延迟算法基本相当，这是因为普通作业任务时间较长（平均任务时间均大于 30 s，作业时间大于 5 min），因而 LRFD 改进效果有限。



(a) Wordcount 的 t　　(b) Wordcount 的 l



(c) Terasort 的 t　　(d) Terasort 的 l

图 7　普通作业的性能和本地性比率对比

5　结　论

本文提出了一种基于本地性资源预测的延迟调度算法，该算法根据节点上任务的完成进度和作业未处理数据在集群中的分布状况预估作业的本地性资源信息，从而判断是否需要进行等待以提高系统性能。搭建了实验平台并进行了实验测试，实验结果表明，LRFD 调度算法能合理地进行等待，在作业响应时间和稳定性等方面均优于已有的延迟调度算法。

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] CHEN Y, ALSPAUGH S, KATZ R. Interactive analytical processing in big data systems: a cross-industry study of MapReduce workloads [J]. Proceedings of the Very Large Data Base Endowmat, 2012, 5(12): 1802-1813.

[3] ISARD M, PRABHAKARAN V, CURREY J, et al. Quincy: fair scheduling for distributed computing clusters [C] // Proceedings of the ACM SIGOPS 22nd

（下转第 54 页）