文章编号:1671-4598(2014)06-1846-04　　中图分类号:TP393　　文献标识码:A

基于云计算 Hadoop 异构集群的并行作业调度算法

郭其标[1]，吕春峰[2]
(1. 嘉应学院 计算机学院，广东 梅州 514015；
2. 郑州铁路职业技术学院 软件学院，郑州 450052)

摘要：针对 Hadoop 异构集群中计算和数据资源的不一致分布所导致的调度性能较低的缺点，设计了一种基于 Hadoop 集群和改进 Late 算法的并行作业调度算法；首先，介绍了基于 Hadoop 框架和 Map－Reduce 模型的调度原理，然后，在经典的 Late 调度算法的基础上，对 Map 任务和 Reduce 任务的各阶段执行时间进度比例进行存储和更新，为了进一步地提高调度效率，将慢任务迁移到本地化节点或离数据资源较近的物理节点上，并给了基于改进 Late 算法的作业调度流程；为了验证文中方法，在 Hadoop 集群系统上测试，设定 1 个为 Jobtracker 主控节点和 7 个为 TaskTracker 节点，实验结果表明文中方法能实现异构集群的作业调度，且与其它方法比较，具有较低的预测误差和较高的调度效率。

关键词：云计算；作业调度；集群资源；慢任务

Algorithm for Concurrent Job Scheduling Based on Cloud Computing Hadoop Heterogeneous Computer Cluster

Guo Qibiao[1], Lu Chunfeng[2]
(1. School of Computer, Jia Ying University, Meizhou 514015, China;
2. Software College, Zhengzhou Railway Vocational and Technical College , Zhengzhou 450052, China)

Abstract: Aiming at the Hadoop heterogeneous computer clusters that have the defects of low scheduling efficiency for in－conformal computing and data resource, a job scheduling method based on Late algorithm based on Hadoop computer cluster was proposed. Firstly, the principle of scheduling based on Hadoop and Map－Reduce model was introduced, then on the basis of the classic scheduling algorithm, the executing time ratio for stages of Map task and Reduce task was stored and renewed, In order to improve the scheduling efficiency further, the slow task is transferred to the local node or the neighbor nodes near the data resource, and the algorism flow for the improved Late algorism was given. In order to verify the method in this paper, the experiment was simulated in the Hadoop computer cluster, the number for Jobtracker nodes and Tasktracker number is 1 and 7, respectively, and the result shows the method in this paper can realize job scheduling for heterogeneous computer clusters, and compared with the other method, it has the low predicating error and high scheduling efficiency.

Key words: cloud computing; job scheduling; computer cluster; slow task

0 引言[1]

云计算[1-2]（Cloud computing）是一个虚拟化的资源池，是在分布式计算、网格计算的基础上发展而来，具有透明性、可扩展性、可伸缩性、冗余性和可用性等特点。

Hadoop[3-5]是 Map－Reduce 调度方式和 HDFS（Hadoop Distributed File System）数据存储方式的开源实现。Hadoop 采用主从式结构来实现 MapReduce 编程模型，将整个集群划分为 1 个 Jobtracker 主控节点和若干个 TaskTracker 从节点。

Hadoop 平台下中计算资源和数据资源通常位于不同的位置，当计算资源和数据位于不同的节点时需要对数据进行迁移，由于在云计算多用户共享环境中，各物理资源之间的共享使得其包含所处理任务的数据资源的概率降低，因此，任务的执行经常需要从远程拷贝数据，从而浪费了网络带宽和降低了任务执行的效率[6-7]。文献［8］设计了一种适用于 Hadoop 平台的延迟调度 Delay 算法，其通过将在作业执行期间设定等待阀值，等待含有其所需数据资源的节点到达或等待时间超过阀值。文献［9］采用时间和费用预算双目标对用户作业进行优化调度，并通过设计权值计算、预算评价和权值更新模型实现对集群资源分配，并通过动态调节作业的权值和资源槽数来实现集群物理资源分配。文献［10］设计了一种 Hadoop 平台作业调度算法，并通过建立数据模型和控制变量实现网络带宽的动态分配，以实现数据的负载均衡。文献［11］设计了一种延迟－容量调度算法，将部分任务将非本地化的节点作为目标节点执行以提高作业的任务的并行程度和减少作业响应时间。

收稿日期:2013-11-21； **修回日期**:2014-02-17。
基金项目:广东省高校优秀青年创新人才培养计划基金资助项目(LYM10121)。
作者简介:郭其标(1982-)，男，广东梅县人，硕士，讲师，主要从事数据安全、数据挖掘及计算机网络方向的研究。
吕春峰(1971-)，男，河南郑州人，硕士，讲师，主要从事计算机应用与网络和图像处理等方向的研究。

文献［12］在 Delay 算法的基础上设计了一种基于资源预测的 Delay 调度算法，通过对资源可用性进行预测以对作业进行合理调度。文献［13］设计了一种改进的 Map－Reduce 模型，在 Map 任务处理完后，通过 Balance 任务对数据进行均衡操作，再将其分配给 Reduce 任务节点。

上述工作均研究了 Hadoop 平台的作业调度，具有重要意义，但主要针对同构集群进行处理，通常对 Reduce 的各节点赋予相同的执行时间，具有一定的片面性，本文在上述工作的基础上，设计了一种基于数据局部性和慢任务迁移执行的算法，能克服上述方法的不足。

1　Hadoop 和 Map－Reduce 模型

1.1　Hadoop 框架模型

Hadoop 集群是由 1 个 Jobtracker 主控节点和若干个 TaskTracker 从节点组成，如图 1 所示。

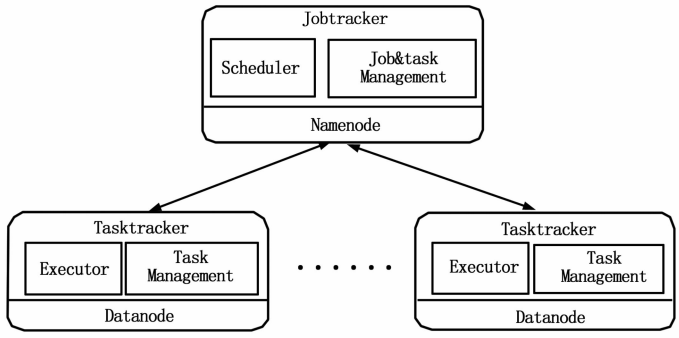

图 1　Hadoop 框架模型

在图 1 中，Jobtracker 主控节点负责管理集群任务和资源管理，例如作业提交、作业调度和任务跟踪、监控作业状态、创建并行的 Map 任务和 Reduce 任务等，TaskTracker 从节点主要负责执行由 Jobtracker 发送的任务以及向其汇报任务执行结果，在执行过程中向 Jobtracker 发送心跳信息，以将任务的执行情况和节点当前的运行状态发送给 Jobtracker 主控节点。

1.2　Map－Reduce 编程/调度模型

Hadoop 采用主从式结构实现 Map－Reduce 编程/调度模型，Jobtracker 主控节点在接收用户任务时，首先将任务划分为若干个 Map 任务独立运行，并将 Map 任务执行的中间结果分类作为 Reduce 任务的输入，Map－Reduce 作业执行的过程可以分为 6 个阶段，如图 2 所示。

图 2 表示了用户作业执行的一次完整过程，其各阶段如下所示：

（1）用户作业划分。文件分割器根据数据分块的个数将作业分为与之数目相同的任务数。

（2）Jobtracker 主控节点在接收到 TaskTracker 从节点的任务请求信息后，选取任务并将其移动到 TaskTracker 对应的从节点上，并启动任务监督模块对其运行状态进行监督。

（3）Map－Worer 节点在接收了 Map 节点任务后，从文件中解析出键值并传递给 Map 函数。

（4）Map－Reduce 周期性地将 Map 函数输出的中间结果写入文件系统中，将将其存储的磁盘位置发送给 Master 节点。

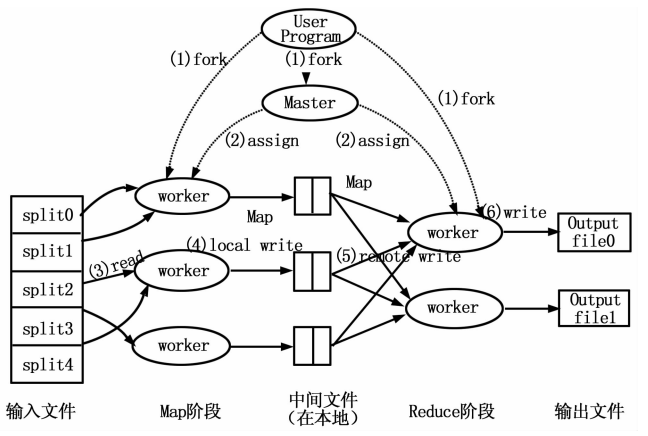

图 2　Map－Reduce 编程/调度模型

（5）Reduce－Worker 将 Master 收到的发送的位置信息后，根据 Master 提供的位置信息，从磁盘中读取数据，并对其进行排序。

（6）将排序后的结果映射给 Reduce 函数进行处理。

2　基于数据局部化和预测的作业调度

2.1　基于 Hadoop 的 LATE 调度算法

Hadoop 集群中用户提交的作业被划分为若干任务并行处理，由于任务的计算量大小不同，因此，其分配到各节点时进度不一，因此，此时将其启动备份任务执行机制，即将进展缓慢的任务迁移到其它节点处运行。LATE 算法采用分片策略表示任务进度。Map 任务划分为执行 Map 函数和中间结果排序两个阶段，两个阶段所需要的执行时间占 Map 任务执行总时间的比例表示分别为 100％和 0％，Reduce 任务分为 3 个阶段，即 Copy 阶段、Sort 阶段和 Reduce 阶段，每个阶段的时间进度均为 1/3。

Late 算法的工作流程可以描述为：

（1）TaskTracker 从节点向 Jobtracker 发送心跳信息请求新任务，Jobtracker 主控节点首先判断整个 Hadoop 集群中备份任务的数量是否小于数量阀值 $SpeculationCap$ ：

如果大于，则算法结束，不能进行慢任务迁移；

否则进入步骤（2）。

（2）对发送请求的 TaskTracker 从节点进行判断，如果其节点处理进度低于阀值 $SlowNodeThreshold$ ，则忽略该节点请求，否则转入（3）；

（4）对当前所有 TaskTracker 从节点正在执行的非备份执行任务进行排序。

假设采用 $ProgresScore$ 表示任务按分片策略得到的进程分，则进程速率 $Progressrate$ 可以表示为：

$$Progressrate = ProgresScore/t \tag{1}$$

其中：t 为任务已花费的执行时间，则任务完成还需要的剩余时间 t_{rem} 可以表示为：

$$t_{rem} = (1 - ProgresScore)/Progressrate \tag{2}$$

对所有任务根据剩余时间 t_{rem} 从大到小进行排序得到队列 S_{rem} 。

（5）对 S_{rem} 中低于慢任务阈值 $SlowTaskThreshold$ 的首个任务迁移到发送请求任务执行的 TaskTracker 从节点上执行。

2.2 LATE 调度算法不足

LATE 调度算法充分考虑了不同任务和节点的计算和性能差异，但也存在下述问题：

(1) LATE 算法，将 Map 任务的两个阶段的时间片占 Map 总时间的比例分别表示为 100%和 0%，而将 Reduce 任务的 3 个阶段的时间片均等地分为 1/3，而实际上对于不同性能的异构节点，Reduce 的 3 个阶段所需要的时间各不相同，影响了作业调度的效率；

(2) 在考虑剩余时间较多的慢任务时，仅考虑其剩余时间作为是否迁移的条件，没有考虑到其所需要的数据是否在本地，如果一个慢任务迁移到不含其所需数据的非本地化节点上，仍然需要较多的执行时间。

2.3 预测机制

由于 Hadoop 中采用任务的当前进度来估计任务的剩余完成时间，因此，只有准确地估计任务的当前进度才能较为准确地估计任务的剩余执行时间，而 Late 算法中将 Map 任务的 Map 函数和中间结果排序两个阶段花费时间片赋值分别为总时间进度的 100%和 0%，而 Reduce 任务的 Copy 阶段、Sort 阶段和 Reduce 阶段的花费时间均赋值为占总时间进度的 1/3，而事实上各节点由于异构对应的各阶段的时间进度应该各不相同，同时不同类型的任务所对应的 3 个阶段所需要的时间是不同的，因此，文中对阶段所需的时间进行预测，如图 3 所示。

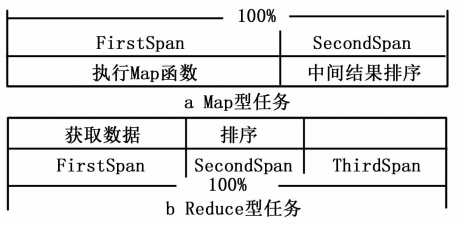

图 3 Map 任务和 Reduce 任务进度

预测的过程可以表示为：

(1) 首先在各 TaskTracker 从节点中存储各种类型任务对应的 Map 任务和 Reduce 任务的各阶段的时间进度；

(2) 当 TaskTracker 从节点在接收到新任务后，首先从本地读取是否存在着该类型节点的历史信息，如果存在，则直接读取，并设定其各阶段所占的进度比例；

(3) 当任务分解对应的 Map 任务和 Reduce 任务均执行完毕后，将本次任务执行的各阶段时间进度比例发送给 TaskTracker 从节点；

(4) TaskTracker 从节点在收到任务发送的各阶段时间进度比例后，对历史信息进行更新。

2.4 数据局部化

任务处理的数据在文件系统中的分布情况决定了作业是否为本地化调度，本地化任务能够减少任务执行时需要从远程节点上传输数据的开销，因此，在进行任务调度时应尽量进行数据局部化，即将任务分配给含有其数据的物理节点进行处理，同时在对备份任务进行迁移时尽量能迁移到含有其处理数据的物理节点或邻近节点上，以提高作业调度的效率，因此，数据的局部化可以从新任务的调度和慢任务的迁移两个方面对作业调度过程进行优化：

(1) 新任务分配。当 Jobtracker 主控节点上有新任务到达时，将所有发送心跳信息的 TaskTracker 从节点上看作可选节点集，判断其中是否有 TaskTracker 节点存储任务所需数据块：

如果 TaskTracker 从节点存储了其所需数据块，则将任务调度到该节点上执行。

如果 TaskTracker 从节点没有存储其所需数据块，则任务开始等待，直到存储了其所需数据块的 TaskTracker 从节点发送了心跳信息或者等待调度的次数超过预设阈值 T；

当等待调度的次数超过预设阈值时，选择可选节点集中的与数据块位置最近的节点进行调度。

(2) 慢任务的迁移。在传统的 Late 算法中，往往将最慢的任务迁移到具有最快的可选节点上，没有考虑数据局部化，因此，这里在 S_{rem} 中从头开始选择能进行本地化的任务，并将其迁移到本地化 TaskTracker 从节点上运行。

3 算法流程

文中算法在传统 Late 算法的基础上，加入了基于历史信息的预测机制和数据局部化处理，其流程如图 4 所示。

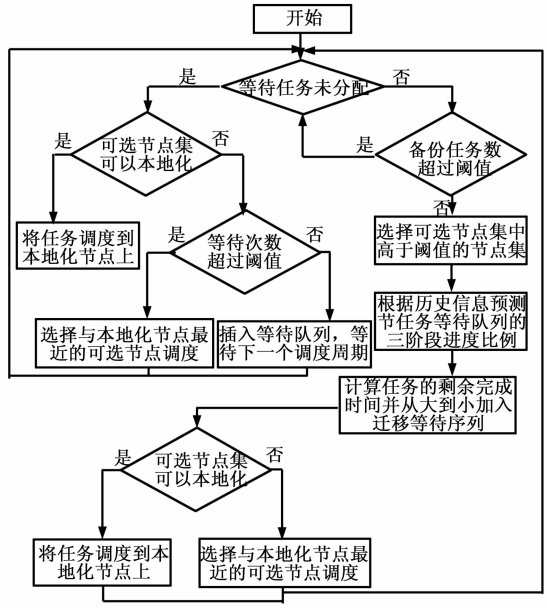

图 4 改进的作业调度算法

4 仿真实验

在本地搭建的 Hadoop 集群系统上测试 MapReduce 模型，网络速率为 100MB/s，集群中共有 8 个节点，其中 1 个为 Jobtracker 主控节点，其它 7 个为 TaskTracker 节点，每个 TaskTracker 节点的 CPU、内存和硬盘参数均不相同。采用文中的调度算法进行作业调度，并与文献 [12] 方法以及经典的 Late 算法进行比较，得到的各 Map 任务对应的剩余完成时间偏差，即与真实值的误差如图 5 所示。

从图 5 中可以看出，文中方法预测的各 Map 任务的剩余完成时间与实际的剩余完成时间的误差最小，除了任务 8、9 和 18 偏差相对较大约为 10 ms，其它任务偏差均小于 5 ms，