doi:10.3969/j.issn.1000-436x.2014.07.015

基于节点识别的慢任务调度算法

崔云飞[1,2]，李新明[2]，李艺[2]，刘东[2]
（1. 北京航天飞行控制中心，北京 100094；2. 装备学院 复杂电子系统仿真重点实验室，北京 101416）

摘 要：为了降低大数据处理集群在执行任务过程中的慢任务对作业执行效率的影响，提出了一种识别慢任务、备份慢任务、减少慢任务相结合的调度算法——TQST 算法。首先，通过判断节点能力和任务执行时间，建立慢节点、非常慢节点和慢任务队列；其次，根据预判备份执行价值确定如何启动慢任务的备份任务，提高了备份执行的作用；然后，在节点识别的基础上，规避为非常慢节点分配任务，从根本上减少慢任务的产生，提高作业执行效率。实验结果表明，TQST 算法在作业响应时间等方面优于已有的慢任务调度算法。

关键词：大数据；慢任务；备份任务；Map-Reduce
中图分类号：TP 302.7 **文献标识码**：A **文章编号**：1000-436X(2014)07-0122-07

Slow task scheduling algorithm based on node identification

CUI Yun-fei[1,2], LI Xin-ming[2], LI Yi[2], LIU Dong[2]
(1. Beijing Aerospace Control Center, Beijing 100094, China;
2. National Key Laboratory of Complex Electronic System Simulation, Academy of Equipment, Beijing 101416, China)

Abstract: In order to reduce the influence of the slow task, produced in big data processing, a scheduling algorithm (TQST) combining recognition, speculation and seduction of slow task was proposed. First of all, through the judgment of node ability and task execution time, slow node queue, very slow node queue and slow task queue were established. Secondly, according to the anticipation speculative execution value to decide how to start speculative task. Then, in the basis of node identification, avoid distributing tasks to very slow node, radically reduce slow task production, improve job execution efficiency. The experimental results show that TQST algorithm previous existing slow task scheduling algorithm in term of the job response time.

Key words: big data; slow task; speculative task; Map-Reduce

1 引言

随着科学技术的快速发展，科学研究、互联网服务、电子商务等多个领域均出现数据量激增的趋势，如何对大数据进行高效处理成为亟需解决的问题。为了应对大数据处理的挑战，Google 提出了 Map-Reduce[1]，Apache Hadoop 对 Map-Reduce 实现了开源，并成为目前最流行的大数据处理工具。Map-Reduce 目前已有 2 个版本：使用最广泛的 MR1 和正在测试的 Yarn，它们主要的框架均是设定系统由主控节点（Master）和数据节点（Slave）组成。Master 负责接收用户请求 job 和管理整个集群的 Slave，首先将 job 拆分成多个 task 并根据 Slave 的负载情况分配 task；然后对 Slave 节点信息和运行 task 进行状态监控，并根据设定的容错调度策略进行动态调度调整；最后汇总 Slave 完成的 task 信息，给出 job 处理结果。

每个程序员在编程时都会问自己 2 个问题“如何完成这个任务”，以及“如何能让程序运行的更快”。Map-Reduce 计算模型的使用和多次优化也是为了更好地解答这 2 个问题[2]。其中一个比较重要的是针对慢任务进行优化。在分布式集群环境下，

收稿日期：2013-03-01；**修回日期**：2013-06-15
基金项目：国家自然科学基金资助项目（60904082）；国家重大科技专项基金项目(2012ZX01045003-001)
Foundation Items: The National Natural Science Foundation of China(60904082); The National Science and Technology Major Project (2012ZX01045003-001)

因为程序 bug，负载不均衡或者资源分布不均，造成同一个 job 的多个 task 运行速度不一致，有的 task 运行速度明显慢于其他 task（比如：一个 job 的某个 task 进度只有 10%，而其他所有 task 已经运行完毕），则这些 task 拖慢了作业的整体执行进度，这种进度缓慢从而影响整个 job 执行速度的 task 称为慢任务。

如何确定真正的慢 task，并在合适的节点上为慢 task 启动备份 task 成为减少作业响应时间的关键。集群资源紧缺时，合理控制备份 task 的数量和启动节点，对确保在少用资源情况下，减少大作业响应时间有至关重要的作用。由于大数据处理系统的异构性，集群必然存在任务执行效率不同，任务执行时间不同的情况。应该对当前运行的 task 进行分析，确定对大作业响应时间影响最大的 task，即慢 task，采取以空间换时间的思路，为慢 task 启动备份 task，让备份 task 与原始 task 同时运行，哪个先运行完，则使用哪个结果，从而减少大作业的整体响应时间。慢 task 完成的时间是整个作业运行时间的关键，只有减小慢 task 完成的时间，才能减小大作业完成的总时间。如何判定慢 task，如何选择合适的节点启动备份任务，如何减少慢任务的产生，是 Map-Reduce 调度方式在异构环境中能够高效运行所必须解决的问题[3~6]。

针对慢任务问题，经典的解决方案[7~10]有 Google MapReduce、Hadoop Speculative task、Berkerley 的 LATE（Longest Approximate Time to End）和 Hadoop Yarn Speculative Execution。

Google MapReduce 采用以空间换时间的方式为慢任务启动多个备份任务，一定程度上解决了慢任务的影响。但存在以下不足：Google 是基于同构环境研究的，不能动态识别异构环境中节点性能，不能够选择最优节点启动任务拷贝；同时启动多个任务拷贝，对资源造成浪费。

Hadoop Speculative task 较 Google MapReduce 更精准地定位慢任务，但仍然没有解决异构的问题，没有考虑节点性能，容易造成调度抖动。

Berkerley 的 LATE 建立了节点队列和任务队列来解决慢任务识别和节点识别的问题，选择性能优异的节点启动备份任务。

Hadoop Yarn Speculative Execution 提出了备份价值[11~13]的概念，选择执行备份任务带来最大价值的节点，比原有算法更精准地定位哪个节点来执行备份任务。

上述 4 种算法共同的思路均是以空间换时间，在执行能力强、负载较轻的节点上对慢任务启动备份任务，4 种算法均在处理已经存在的慢任务时存在一定的缺陷，更重要的是都没有从根本上解决慢任务生成的问题，不能够有效地减少慢任务的生成。

本文分析上述几种慢任务调度算法存在的问题，提出异构环境中基于节点识别的慢任务调度算法。该算法通过实时调整运行任务中的慢任务队列和集群节点中归一化的慢节点队列，精确识别慢任务，在合适的节点上为慢任务启动合适的备份任务，并对后续任务进行动态调度，从根本上减少慢任务的生成。

2　研究背景

2.1　Hadoop 作业调度流程及执行描述

Apache Hadoop 的 MapReduce 框架是基于 Google MapReduce 原理实现的开源软件，目前是最流行的大数据处理工具。

Hadoop 的 MapReduce 框架执行作业时，单个作业 job 被拆分成多个任务 task 执行。由 JobInProgress 监控 job 的执行进度，TaskInProgress 监控单个 task 的执行，task 的执行采用 task attempt 机制。正常情况下，每个 task 启动一个 task attempt；当检测到任务执行失败后，控制中心会为该任务启动一个相同的 task attempt；当 task attempt 被判定为慢任务后，控制中心会选择一个合适的节点为对应的慢任务再启动一个 task attempt，称为备份任务，这 2 个 task attempt 同时运行，哪个先执行完，就采用哪个的结果，并 kill 掉另一个 task attempt。

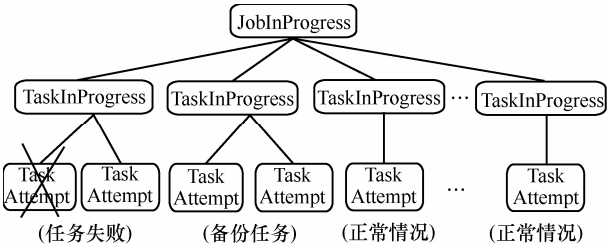

图 1　Hadoop MapReduce 作业描述方式

为了降低慢任务对作业整体执行效率的影响，Google、Apache Hadoop 以及一些研究机构对此进行了一定研究，目前主要的解决思路是：根据各 Slave 节点的负载情况，将慢任务调度到执行能力

强、负载较轻的节点运行。关键的技术点是如何在大量 task 运行环境中动态判定慢任务，以及如何选择合适的节点启动备份任务。

2.2 问题的提出

在目前备份任务的机制下，由于慢节点的原因，某种情况下会出现多节点执行作业反而慢于较少节点执行同样作业。下面以一个简单的例子进行说明。

假设集群中有 slave1、slave2、slave3、slave4 共 4 个节点，其中 slave4 工作效率低。

一共有 12 个任务需要去做，slave1、slave2 和 slave3 执行一个任务需要 1 min，slave4 执行一个任务需要 3 min。假设每个 slave 完成自身的任务才去执行备份任务。

如果让 slave1、slave2 和 slave3 去做，则需要 4 min，并行执行总时间就是单个 slave 的执行时间 4 min；然而 4 台同时去做需要 5 min，分析如下。

这里 slave1、slave2 和 slave3 都完成了自身的 3 个任务，slave4 完成了 1 个任务，还有 2 个任务没开始执行，已经花费了 3 min，剩下最后 2 个任务中的一个考虑到数据的本地性分给了 slave4，另一个分给 slave1。1 min 后，slave1 上的任务执行完毕，slave4 上的任务仍在执行，基于目前的备份任务机制，jobtracker 会觉得 slave4 正在执行的任务为慢任务，假设在 slave2 上执行其备份任务，再经过 1 min，slave2 上的备份任务执行完毕，slave4 上的任务 kill 掉，最后执行时间是 5 min。

上述例子说明使用目前解决慢任务的备份机制，执行相同的作业，使用较多的节点可能会比使用较少的节点所需时间更长。为了避免此类情况的出现，有必要使用节点识别技术，通过资源的动态调度，从根本上减少慢任务的生成，减少作业的响应时间。

3 基于节点识别的慢任务调度

本文提出的基于节点识别的慢任务调度算法，解决 2 个问题：根据任务执行信息，判断已经产生的慢任务，并为其选择合适的节点启动备份任务；识别集群中的慢节点，动态调整集群的任务调度，降低慢任务生成的概率，从根本上解决慢任务问题。

基于节点识别的慢任务调度算法的基本思想是，首先，根据任务的执行进度，建立任务队列，并以此来判断可能的慢任务；其次，根据归一化的节点执行能力，建立节点队列，并以此来区分慢节点和快节点；然后，当一个节点空闲时，根据节点队列信息、任务队列信息和备份任务执行信息，确定是否为该节点分配任务，是否为该节点分配备份任务。

3.1 任务队列和节点队列

为了判断慢任务，设计了任务队列排序算法。使用 TaskQueue 记录任务快慢信息，根据任务近似结束时间升序排列 task；使用 NodeQueue 记录集群中各节点的快慢信息，根据节点计算能力进行降序排列 slave 节点。

算法 1 任务队列和节点队列建立算法

输入：slave 节点的心跳信息（任务执行进度、执行时间）

输出：任务队列和节点队列

Begin

1) When a heartbeat of slave node arrives:

2) 计算该 slave node 上正在运行的 tasks 的运行速率；

3) 根据任务运行进度 progress 和运行速率推测任务近似结束时间 AproximateEndTime；

4) sort TaskQueue by AproximateEndTime in descending order;

5) sort NodeQueue by average speed of tasks running on slave nodes in ascending order;

6) define first 25% of TaskQueue as SlowTaskSet;

7) define first 25% of NodeQueue as SlowNodeSet;

8) define first 10% of NodeQueue(and its speed<averagespeed*0.5)as VerySlowNodeSet;

End

3.2 减少慢任务生成及其处理算法

在算法 1 确定慢任务队列、慢节点队列和非常慢节点队列的基础上，提出减少慢任务生成及其处理算法。

算法 2 减少慢任务生成及其处理算法

输入：当前到达的空闲节点 n

输出：是否向节点 n 下发任务，是否向其下发备份任务

Begin

1) When anidle slave node n arrives:

2) **if** $n \in VerySlowNodeSet$ **then**

3) $as\sin gnTesttask(n)$;

4) //为非常慢的节点分配一个测试任务，测试该节点的性能，直到该节点不属于非常慢的节点。

5) **return;**

6) **else if** $n \in SlowNodeSet$ **then**

7) $as\sin gnNewtask(n)$;

8) //根据系统事先部署的FIFO或Capacity等调度算法下发一个新任务，避免在慢节点上启动备份任务。

9) **return;**

10) **else if** 符合启动备份任务的条件 **then**

11) **for** $task_i$ in SlowTaskSet **do**

12) compute speculativeValue of $task_i$ if it runs on slave node n;

13) //计算慢任务队列中所有任务在 node n 上备份执行的价值。

14) **end for**

15) **return;**

16) 选择 speculativeValue 最大的 $task_j$;

17) $as\sin gntask_j(n)$;

18) //在 node n 上为 $task_j$ 启动备份任务。

19) **return;**

20) **else**

21) $as\sin gnNewtask(n)$;

22) **return;**

End

下面重点对算法中测试任务、启动备份任务的条件和 speculativeValue 的计算方法说明如下。

测试任务：在算法 2 中，被认定为特别慢的节点 VerySlowNode，在其空闲时将不再被分配正常的任务，而怎样对其能力进行实时监测以及何时将其重新纳入正常节点的范畴成为必须解决的问题。本文使用测试任务对 VerySlowNode 进行测试，测试任务是一个随机的正常任务的副本执行，其执行过程及执行结果均与正常任务无关（规避测试任务对正常任务的影响）。使用测试任务监测 VerySlowNode 归一化的处理能力，一旦监测到该节点的处理能力达到集群使用的标准（该节点的实时能力大于 VerySlowNode 的判定值），将该节点从 VerySlowNodeSet 中释放。

VerySlowNodeSet 中某个节点只要满足以下 2 个条件中的任意一个，那么就将节点重新纳入正常节点范畴，并让其正常执行任务。

1) 该节点归一化的执行能力大于所有节点队列 NodeQueue 中最慢的 10%的节点的执行能力。

2) 该节点归一化的执行能力大于所有节点平均执行能力的 50%。

说明：第一个条件是确认节点执行性能不属于最差范畴；第二个条件避免把性能还不错的节点划入 VerySlowNodeSet 节点范畴，避免造成资源使用的浪费。

VerySlowNode 节点只运行测试任务原因如下。

1) 目前，以 Hadoop 为代表的大数据处理体系，采取了一种粗放的方式处理海量的数据，机器学习的原理很多时候也是依靠大量的样本而不是精确的逻辑。想要用好大数据，需要通过技术手段快速高效地分析整理海量的样本，需要尽量用简单的方式去处理大量的数据，避免复杂的处理方式带来不必要的开销。因此，本文在对慢任务调度进行优化的过程中，尽量避免复杂化大数据处理主流程。VerySlowNode 节点变慢的原因可能会有很多种（如磁盘故障、内存溢出、程序 bug、负载不均衡等），在处理过程中分析节点变慢的原因并进行修复，会影响大数据处理主流程的效率。本文采用简单的方式处理非常慢的节点（不再分发任务），最大可能减少慢任务的产生，减少处理方式本身对大数据处理主流程的影响；在 VerySlowNode 节点上运行测试任务，当检测到该节点归一化后的执行能力达到阈值时，将其纳入正常节点范畴，并让其正常执行任务。

2) VerySlowNode 节点只运行测试任务会造成资源使用的浪费，但能够减少慢任务的产生。避免资源的浪费和减少慢任务的产生是一对矛盾体。在 2.2 节（问题的提出部分）对资源使用个数和作业响应时间之间的可能关系进行了说明。为了避免资源浪费，而在非常慢的节点上正常执行任务，产生慢任务的可能性会很大，反而会降低作业的整体执行效率。因此，本文不向“真正的慢节点”分发正常任务，减少慢任务的产生；同时，使用慢节点判定条件 2)减少误判慢节点的概率，尽量避免资源浪费。

启动备份任务的条件如下。

1) 还没有为慢任务 $task_j$ 启动备份任务。

2) 整个作业 job 的备份任务数目小于其上限，

该数目是以下 3 个数值的最大值：

① MINIMUM_ALLOWED_SPECULATIVE_TASKS（常量 10）

② PROPORTION_TOTAL_TASKS_SPECULATABLE（常量 0.01）×totalTaskNumber

③ PROPORTION_RUNNING_TASKS_SPECULATABLE（常量 0.1）×numberRunningTasks

3)在目前的空闲节点上为慢任务 $task_j$ 启动备份任务的价值 speculativeValue 比其他 task 启动备份任务的价值大。

speculativeValue 的计算方法：借鉴 hadoop- 0.23 系列中 speculationValue 的计算方法。

speculationValue=estimatedEndTime_estimatedReplacementEndTime

其中，estimatedEndTime 是通过预测算法推测的该任务的最终完成时刻，计算方法为

estimatedEndTime=estimatedRunTime_task AttemptStartTime

其中，taskAttemptStartTime 为该任务的启动时间，而 estimatedRunTime 为推测出来的任务运行时间，计算方法如下

estimatedEndTime=(timestamp_start)/Math.max(0.0001, progress)

其中，timestamp 为当前时刻，而 start 为任务开始运行时间，timestamp_start 表示已经运行时间，progress 为任务运行进度（0~1.0）。

estimatedReplacementEndTime 含义为：如果此刻启动该任务，（可推测出来的）任务最终可能的完成时刻。

4 实验与结果分析

为了分析文中提出的基于节点识别的慢任务调度算法（TQST）的性能，下面将 TQST 算法和 Berkeley LATE 算法、Hadoop Yarn Speculatve Execution 算法进行比较。基于 Hadoop 开发了 Adaptive Capacity Scheduler 模块。通过在异构集群的实验，分析算法的性能。

4.1 实验环境

本节主要描述实验的环境，以及环境的各个参数。使用实验室的 10 台 PC 机进行实验集群的搭建，各 PC 机采用 1 000 Mbit/s 的局域网互联。这 10 台 PC 机是异构的，如表 1 所示。

表 1 实验用集群环境配置

机器	机器数	操作系统	磁盘/GB	CPU	内存
Lenovo	4	Centos 6.3	500	2	2 GB
Lenovo	1	Centos 6.3	80	1	512 MB
Lenovo	2	Centos 6.3	500	2	1 GB
Dell	3	Centos 6.3	500	2	2 GB

原型系统基于 Hadoop-0.23.5 开发，Master 模块部署于管理节点，Slave 模块部署于 9 个计算节点，1 000 Mbit/s 以太网作为数据传输网络。输入文件由 Hadoop 分布式文件系统管理，文件块存储于计算节点的本地硬盘，每个文件块的大小为 100 KB 至 64 MB（用于测试处理不同大小数据块时的效率），并且有 2 个副本。每一个 MapReduce job 作为一个作业，而一个作业中的 Map Task 作为任务。实验利用 Hadoop 自带的 Capacity Scheduler 模块实现 Hadoop Yarn Speculatve Execution 算法，利用 Adaptive Capacity Scheduler 模块实现 Berkeley LATE 算法和 TQST 算法。

4.2 实验设置

为了比较调度算法对不同规模作业的影响，实验依照单个 task 处理的数据量分成 5 组，分别为 100 kB、1 MB、10 MB、32 MB 和 64 MB。每组测试的任务数均取 20 个任务、100 个任务和 500 个任务。具体设置如表 2 所示。

表 2 作业参数设置

组号	单任务处理的数据量	任务数	数据量
1	100 kB	20	2 MB
		100	10 MB
		500	50 MB
2	1 MB	20	20 MB
		100	10 0MB
		500	500 MB
3	10 MB	20	200 MB
		100	1 GB
		500	5 GB
4	32 MB	20	640 MB
		100	3.2 GB
		500	16 GB
5	64 MB	20	1.2 GB
		100	6.4 GB
		500	32 GB

通过提交不同大小的作业，主要考察 2 个指标：备份任务执行数量，主要反映慢任务生成个数；算法完成作业的响应时间。

4.3　慢任务生成个数分析

在使用不同算法的实验中，采用相同的慢任务判断标准：任务执行效率为最慢的 20%的任务，并且小于作业中所有任务平均执行效率的 50%。TQST 算法和 Berkeley LATE 算法、Hadoop Yarn Speculatve Execution 算法执行完作业过程中共启动的备份任务数量对比如图 2~图 4 所示。

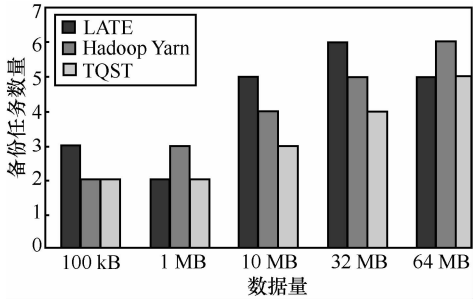

图 2　备份任务数量对比（20 任务）

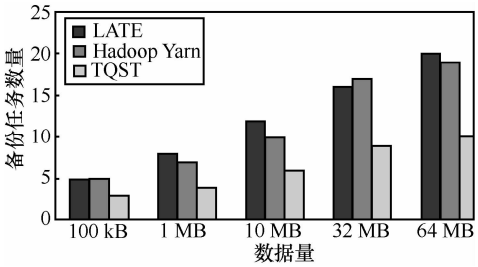

图 3　备份任务数量对比（100 任务）

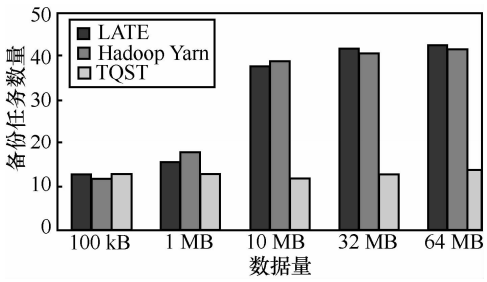

图 4　备份任务数量对比（500 任务）

从上述比较中可以直观看出，原来的慢任务调度算法 Berkeley LATE 算法和 Hadoop Yarn Speculatve Execution 算法没有采取异构环境中减少慢任务产生的机制，会产生较多的慢任务，同时会启动较多的备份任务；而本文提出的 TQST 算法，采取基于节点识别的调度算法，避免向非常慢的节点调度新任务，从而减少慢任务的产生，大幅度降低了慢任务的产生。同时，从上述几个图中可以看出，随着单个任务处理数据量的增加，备份任务执行的数量变多，原因是单个任务的执行时间增大，更容易达到识别慢任务的时间限制。

4.4　作业响应时间分析

作业响应时间如图 5~图 7 所示。

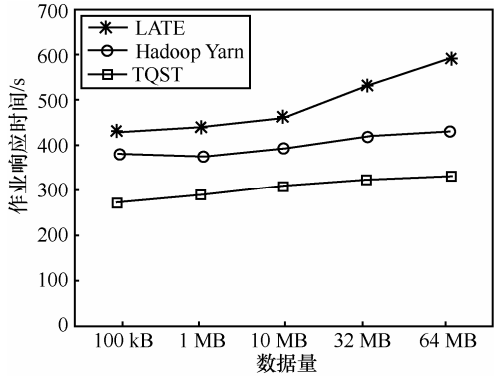

图 5　作业响应时间（20 任务）

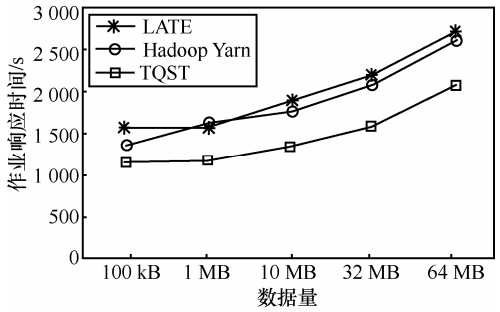

图 6　作业响应时间（100 任务）

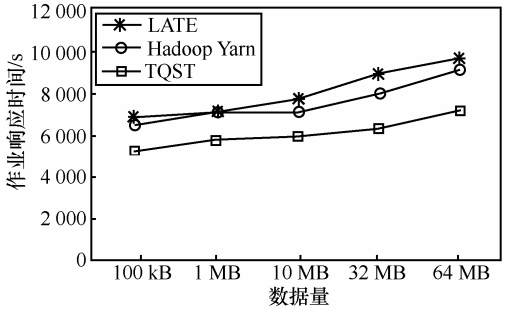

图 7　作业响应时间（500 任务）

从上述比较中可以直观地看出，Hadoop Yarn Speculatve Execution 算法由于采用备份价值最大的启动机制，能够更准确地为慢任务启动备份任务，从而比 Berkeley LATE 算法减少了作业响应时间。本文提出的 TQST 算法，采用基于节点识别的慢任务调度算法，不为非常慢的节点调度新任务，虽然部分慢节点不能正常参加集群工作，但是减少了慢任务的产生，从而最大可能降低了慢任务对作业响

应时间的影响，明显提高了作业效应效率。

5 结束语

本文提出了一种基于节点识别的慢任务备份执行和减少慢任务产生的调度算法。该算法与已有慢任务处理算法的不同在于非常慢节点不再执行新任务。根据备份任务启动价值，为慢任务启动价值最大的备份任务，解决已经产生的慢任务；在确保集群资源利用率的前提下，规避非常慢的节点，从根本上减少慢任务的产生。该算法能显著降低慢任务的数量，提高作业的响应效率。最后的实验结果证明了 TQST 算法的正确性和合理性。

参考文献：

[1] DEAN J, GHEMAWAT S. MapReduce: simplified data processingon large clu8ters [J].Communications of the ACM, 2008,51(1): 107-113.

[2] 陆嘉恒. Hadoop 实战[M]. 北京：机械工业出版社, 2012.
LU J H. Hadoop actual combat[M]. Beijing: China Machine Press, 2012.

[3] Adaptive scheduler[EB/OL]. https://issues.apache.org/jira/browse/MAPREDUCE-1380,2013.

[4] Improve speculative execution[EB/OL]. https://issues.apache.org/ jira/browse/MAPREDUCE-2039,2013.

[5] Speculative execution is too aggressive under certain conditions[EB/OL]. https://issues.apache.org/jira/browse/MAPREDUCE- 2062,2013.

[6] Speculative execution algorithm in 1.0 is too pessimistic in many cases[EB/OL]. https://issues.apache.org/jira/browse/MAPREDUCE-3895, 2013.

[7] FLORIN D T. S. Eugene ngunderstanding the effects and implications of compute node related failures in hadoopHPDC'12[A]. The Netherlands ACM[C]. 2012.187-197.

[8] 段翰聪, 李俊杰, 陈宬等. 异构环境下降低任务抖动的调度法——DPST[J]. 计算机应用, 2012,32(7): 1910 -1912, 1938
DUAN H C, LI J J, CHEN C, *et al*. DPST: a scheduling algorithm of preventing slow task trashing in heterogenous environment [J]. Journal of Computer Applications, 2012,32(7): 1910 -1912, 1938.

[9] LEE K H, LEE Y J, CHOI H, *et al*.Parallel data processing with MapReduce: a survey[J]. SIGMOD Record, 2011,40(4):11-20.

[10] MATEI Z, ANDY K, ANTHONY D. Improving MapReduce performance in heterogeneous environments[A]. 8th Usenix Symposium on Operating Systems Design and Implementation[C]. 2008.29-42.

[11] Resource manager rest[EB/OL].www.hadoop.apace.org/docs/r0.23.6, 2013

[12] Speculative execution for reads[EB/OL]. https://issues.apache.org/jira/browse/CASSANSRA-4705,2013.

[13] Looking for speculative tasks is very expensive[EB/OL]. https://issues.apache.org/ jira/browse/MAPREDUCE -4499,2013.

作者简介：

崔云飞（1986-），男，山东德州人，博士，北京航天飞行控制中心工程师，主要研究方向为大数据和云计算。

李新明（1965-），男，湖南益阳人，博士，装备学院教授，主要研究方向为大数据和云计算。

李艺（1964-），男，湖北宜都人，硕士，装备学院教授，主要研究方向为大数据和云计算。

刘东（1981-），男，四川巴中人，博士，装备学院讲师，主要研究方向为大数据和云计算。