基于MapReduce模型的推测执行优化算法

黄中平 白光伟 沈 航 承 骁 华志翔
(南京工业大学计算机科学与技术系 南京 210009)

摘 要 作为数据中心大规模处理框架,MapReduce集群包含成百上千个节点,多采用推测执行的方法来有效解决并行计算中的掉队任务。针对集群中实时性需求较高并且任务量较小的目标作业,提出基于MapReduce模型的推测执行优化算法,其目的是在满足实时性需求的基础上尽量减少目标作业的完成时间。首先通过分析任务模型和时间模型,引入数学0-1规划模型,求得整体作业的完成时间最小;然后设计可以在多项式复杂度内完成的启发式算法,目的是在可用资源允许的范围内尽量逼近最优值;最后通过大量实验模拟验证算法的执行效果。

关键词 MapReduce,并行计算,推测执行,实时性
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.042

Speculative Execution Optimization Algorithm with MapReduce

HUANG Zhong-ping BAI Guang-wei SHEN Hang CHENG Xiao HUA Zhi-xiang
(Department of Computer Science and Technology, Nanjing Tech University, Nanjing 210009, China)

Abstract In the framework of data center for large-scale data processing, MapReduce contains thousands of nodes. Speculative execution is a way to improve the efficiency of parallel computing, which can deal with the straggling task in parallel computing effectively. In this paper, we proposed a speculative execution optimization algorithm with MapReduce, focusing on the target jobs with higher demand of real time and less amount of calculation. The purpose is to minimize execution time while meeting real time demand. To this end, we established a task model and a time model. By the analysis of the task model and time model, we employed a 0-1 integer linear program to minimize the total finishing time. In addition, a heuristic algorithm was put forword to meet the optimal value, which can be done with the polynomial complexity. Finally, the simulation experiment results show that the proposed algorithm can gain remarkable effect.

Keywords MapReduce, Parallel computing, Speculative execution, Real time

1 引言

随着大数据时代和云计算的蓬勃发展,并行计算框架在数据挖掘、日志分析等领域起着越来越重要的作用。MapReduce[1]是Google于2004年提出的并行计算框架,已被广泛用于大规模数据集的分析以及互联网应用。然而,并行计算框架都存在一定的缺陷,现今集群可以由成百上千个节点构成,很难保证并行作业中的所有任务都能在正常运作的节点上运行,若有被分配任务的节点因程序缺陷、负载不均衡、资源或者节点本身性能差而造成该任务运行缓慢,则会影响整个作业的完成质量并且拖延并行作业的完成时间,称这种任务为掉队任务(Straggling Task)。

目前,推测执行(Speculative Execution)是有效解决掉队任务的最广泛的方法,主要的推测执行方法有两种:克隆方法(the Cloning Approach)[2]和基于异常检测的方法(Straggler-Detection-Based One)[3-6]。克隆方法是指在并行作业到达时,给作业的每一个任务都启用多个相同的备份任务(Attempt Tasks);在Hadoop[8-9]平台中,启用推测执行后,备份任务会被创建,当任意一个备份任务执行完成后,将杀死所有其他相同的备份任务,从而提高每个作业的平均执行效率。基于异常检测的方法则是在作业执行时实时监测任务的执行速率,一旦任务的执行速率缓慢或出现异常,则对可能延迟作业完成的掉队任务进行备份,增加掉队任务提早完成的机率。

在集群中存在多类作业,在没有时间要求或作业负载较大的情况下,通过克隆算法进行预测执行是不明智的,创建作业的多个副本会导致系统资源严重短缺,因此这些作业适用

到稿日期:2016-03-18 返修日期:2016-05-15 本文受国家自然科学基金项目(61502230,61073197),江苏省自然科学基金项目(BK20150960),江苏省科技支撑计划(工业)项目(BE2011186),江苏省未来网络前瞻性研究项目(BY2013095-4-09),江苏省普通高校自然科学研究项目(15KJB520015),江苏省六大高峰人才基金资助项目(第八批),江苏省2015年度普通高校研究生科研创新计划(KYLX15_0804)资助。

黄中平(1993—),男,硕士生,主要研究方向为移动云计算、分布式视频转码,E-mail:1031239402@qq.com;白光伟(1961—),男,博士,教授,博士生导师,CCF高级会员,主要研究方向为移动互联网、云计算、网络体系结构和协议、网络系统性能分析和评价、多媒体网络服务质量等;沈 航(1984—),男,博士,讲师,主要研究方向为云计算、移动互联网、无线多媒体通信协议等;承 骁(1990—),男,硕士生,主要研究方向为移动云计算、虚拟化架构;华志翔(1991—),男,硕士生,主要研究方向为移动云计算、大数据分析和处理。

于异常检测的推测执行方法，比如 Hadoop，Google 等集群系统。文献[11]分析了各种负载情况下不同推测执行方法的关系。对于时间要求较高、任务量少的作业，若采用异常检测的方法，可能导致一个作业中的任务在短时间内完成后系统还未检测出异常，或者检测出异常后再分配内存、计算等资源进行备份处理，这样大大增加了完成时间，本文针对此类目标作业(下文中作业即为该目标作业)，在 MapReduce 的框架的基础上优化克隆推测执行。

针对克隆方法不加区分备份和作业实时性的问题，本文在 MapReduce 框架下进行推测执行的优化，使得每个作业进行合适的副本创建并使完成时间最小。首先，在相关工作的基础上提出一个推测执行优化算法，通过分析任务模型和时间模型，引入数学 0-1 规划模型，使得整体作业的完成时间最小，并在此模型下提出优化的执行算法，即优化的智能克隆算法 ESCA(Enhanced Smart Cloning Algorithm)；然后，通过仿真实验的分析，观察此算法的特点。

本文第 2 节为相关工作和问题描述；第 3 节提出推测执行优化算法；第 4 节为仿真实验和性能分析；最后总结全文并展望未来工作。

2 相关工作

较多研究者提出相应的解决方案以优化在 MapReduce 框架下出现掉队任务的问题。文献[12]提出一种 LATE (Longest Approximate Time to End)方法，该方法计算每个任务的执行速率并估算任务的剩余完成时间，对剩余时间最长的任务进行备份，并选择计算能力较强的节点执行备份任务。这种方法可以有效地改进并行计算的掉队任务，避免备份任务成为新的掉队任务。文献[7]提出一种 MCP(Maximize Cost Performance)策略，该策略是在 LATE 的基础上进行改进的，通过分阶段的方式利用 EWMA(Exponentially Weighted Moving Average)预测算法预测每个阶段的执行速度，并通过此算法计算任务的剩余完成时间，从而准确地选择掉队任务，并构建代价收益模型，以最小的代价提高系统性能。

文献[11]提出 SCA(Smart Cloning Algorithm)算法，该算法是在集群负载较低的情况下实现系统资源最大化的一种策略。当集群中作业量小于一定的阈值时，通过考虑集群中的资源消耗、节点数量等情况推算出每个任务备份的有效值，使得整体的效用函数最大化，该方法适用于没有截止期限的作业。在一个集群中，作业被提交后，需要考虑作业的时效性，此时需要减少作业的完成时间，使用异常检测方法对作业中的每个任务进行监测，当发现作业中有掉队任务时，对其进行备份，由于分配内存、计算等资源执行备份时需要消耗较多时间，因此 SCA 方法的思想是在作业进入队列之前综合考虑资源以对每个任务进行备份，在最大化效用函数的同时减少作业的完成时间，代价是多消耗一定的资源。这种方法是针对一般的并行计算框架，没有结合 MapReduce 这种有两个阶段的情况，没有考虑 Map 和 Reduce 阶段之间的相关性和作业的实时性。文献[13]只分析了在 MapReduce 集群下单个作业的推测执行方法，此算法适用于多个作业的系统。

本文在 SCA[11]算法的基础上，在数学 0-1 规划模型上建立优化的副本方案，提出启发式优化算法。

3 推测执行优化算法

本节介绍提出的推测执行优化算法 ESCA。首先对系统中的任务副本创建模型和时间模型进行详细介绍，然后结合 0-1 规划给出完成时间的最优解，最后通过分析最优解的复杂度，提出 ESCA 算法，并给出具体实现。

3.1 任务机制

本文工作是在包含多个同构计算资源节点的集群系统中进行的。在集群中，每个时间段都有作业到来和处理结束，假设在 L 时刻，集群中等待被处理的作业有 N 个，记为 $J=\{J_1, J_2, J_3, \cdots, J_N\}$，作业是并行处理的，并且每个作业中含有多个任务，各任务在节点上相对独立运行，不考虑作业之间的关联性。另外，对作业中的所有任务平均分配作业量，若作业量分配不均匀则会出现倾斜任务(Shew Task)[11]，限制作业的并行实现，也会影响系统资源的分配。假设在 L 时刻集群中可用的节点资源有 M 个，每个节点在同一时间只能处理一个任务，处理完后即释放资源。在没有使用推测执行的系统中，作业中的任务完成时间都符合帕累托分布，即 J_i 中任意一个 x_j 符合：

$$x_j \approx F_j(t)=\begin{cases}1-(\frac{t_m}{t})^{\alpha}, & t\geqslant t_m\\ 0, & \text{others}\end{cases} \tag{1}$$

其中，t_m 表示任务中最短的完成时间，为方便模拟 J_i 的完成时间，将每个作业的完成时间定义为作业中任务的最晚完成时间，不考虑任务完成后的合并处理以及分布式存储等问题。

文献[2]指出基于克隆的推测执行方法分为两种：作业级的克隆(Job-level Cloning)和任务级的克隆(Task-level Cloning)。作业级的克隆是基于作业来实现的，创建关于整个作业的副本数；任务级的克隆指在作业中为任务创建副本。假设每个任务出现掉队任务的概率为 p，作业中含有 n 个任务，克隆副本数为 c，那么作业级克隆出现掉队任务的概率为 $(1-(1-p)^n)^c$，而任务级克隆出现掉队任务的概率为 $1-(1-p^c)^n$，由此发现在创建相同的副本数时，任务级克隆方法明显优于作业级克隆方法，并且出现掉队任务的概率下降的速度较为明显，所以本文采用任务级的克隆方法，即基于每个作业的任务创建副本数。

3.2 时间模型

在 L 时刻，集群中到来 N 个作业等待被调度，每个作业包含 m_i 个任务，记为 $J_i=\{x_1, x_2, x_3, \cdots, x_{mi}\}$，在节点上处理的每个任务有等可能概率遇到掉队节点，从而形成掉队任务而假设每个任务遇到掉队节点的概率为 p，那么在未使用推测执行之前，第 i 个作业未出现掉队任务而被节点正常运行处理的概率为：

$$P_i(m_i)=(1-p)^{m_i}, 1\leqslant i\leqslant N \tag{2}$$

由式(2)可知，作业中含有的任务数越多，整个作业完成的时间会因为掉队任务而大幅增加。

在使用推测执行后，如果每个任务创建 c_i 个副本任务，那么第 i 个作业能正常运行的概率为：

$$P_i(m_i, c_i)=(1-p^{c_i})^{m_i}, 1\leqslant i\leqslant N \tag{3}$$

比较式(2)和式(3)可以发现，在一定范围内，随着 c_i 的增长，作业正常运行的可能性越来越高，随之出现掉队任务的概率会降低，作业完成时间也会随之减少。

每个作业的完成时间记为 t_i，所有任务在同一时间被调度，以防止晚调度的任务会延长整个作业的完成时间。定义第 i 个作业中第 k 个任务的完成时间为 t_k^i，不同的作业可以创建不同的副本数，但每个作业内必须保持相同的副本数。另外，本文定义 $t_{k,j}^i$ 为第 i 个作业中第 k 个任务所创建的副本中第 j 个副本的完成时间，定义 t_i 由组成 m_i 个任务中最大的 t_k^i 生成，即为第 i 个作业的完成时间，表示如下：

$$t_k^i=\min\{t_{k,1}^i,t_{k,2}^i,t_{k,3}^i,\cdots,t_{k,c_i}^i\},1\leqslant k\leqslant m_i \tag{4}$$

$$t_i=\max\{t_1^i,t_2^i,t_3^i,\cdots,t_{m_i}^i\} \tag{5}$$

在集群中，作业在截止期之前完成才能满足作业的有效性，所以作业的完成时间须满足 $t_i<\lambda_i$，其中 λ_i 是第 i 个作业的最晚完成时间。

创建副本时需考虑系统有限的可用资源，不能超过资源阈值，有：

$$\sum_i^N m_i * c_i\leqslant M_l \tag{6}$$

在满足 N 个作业符合截止期的条件下，考虑系统资源的分配情况，得出最优化时间模型：

$$\begin{aligned}&\min_{t_i}\sum_i^N t_i\\&\text{s. t.}\sum_i^N m_i * c_i\leqslant M_l\\&\quad t_k^i=\min\{t_{k,1}^i,t_{k,2}^i,t^i{}_{k,3},\cdots,t_{k,c_i}^i\},1\leqslant k\leqslant m_i\\&\quad t_i=\max\{t_1^i,t_2^i,t_3^i,\cdots,t_{m_i}^i\},t_i<\lambda_i,1\leqslant i\leqslant N\end{aligned} \tag{7}$$

本文的目的在于在集群可用资源已知的情况下，使用推测执行的方法，使得每个作业在符合实时性的条件下可以取得最优化的完成时间。解决这类问题的关键在于为每个作业创建最佳的副本数，此类问题属于 NP 难问题，目标是将该问题简化为 0-1 整数规划问题，结合任务的复杂度提出多项式时间内求解的启发式算法。

3.3　最小化完成时间模型

在可用资源不足的情况下，MapReduce 框架不可能无限创建副本来达到最优的完成时间，因此本节在上文的基础上将时间最小化问题建模成一个 0-1 整数规划问题。

问题定义：本文用一个二维数组 $x[N][C]$ 来定义作业集 J 中的工作状态，当 x_{ij} 的取值为 1 时表示 job 作业集中的作业 J_i 采用创建副本数为 c_j 的策略在集群中进行处理，并且 $1\leqslant i\leqslant N,0\leqslant j\leqslant C$，其中 C 表示可以创建副本数的上限。文献[2]通过模拟式(3)的数据图发现，当副本数取值为 4 时作业产生掉队任务的概率相当小，相对作业的完成时间也达到稳定，在此将创建副本数阈值设定为 4。

优化目标：使得 N 个 job 在满足实时性条件下所完成的时间最小化，得到最优化目标函数：

$$\min_{t_i}\sum_j^N\sum_i^C x_{ij} * t_{ij} \tag{8}$$

$$\text{s. t.}\sum_i^N\sum_j^C x_{ij} * m_i * c_j\leqslant M_l \tag{9}$$

$$t_k^i=\min\{t_{k,1}^i,t_{k,2}^i,t_{k,3}^i,\cdots,t_{k,c_i}^i\},1\leqslant k\leqslant m_i \tag{10}$$

$$t_i=\max\{t_1^i,t_2^i,t_3^i,\cdots,t_{m_i}^i\} \tag{11}$$

$$\sum_j^C x_{ij} * t_{ij}\leqslant\lambda_i,1\leqslant i\leqslant N \tag{12}$$

$$\forall i,\sum_j^C x_{ij}=1,x_{ij}\in\{0,1\} \tag{13}$$

其中，式(8)是优化目标函数，表示第 i 个作业采用 c_j 副本策略所完成的时间集合；式(9)的约束为所有 job 处理消耗的系统计算资源之和必须小于 L 时刻可用资源节点数 M；为了满足系统的实时性要求，式(10)－式(12)需要在截止期之前完成系统的各个作业；式(13)表明任何作业在处理时只能有一种副本创建机制，不能出现作业使用两种或两种以上的副本机制的情况，x_{ij} 的取值为布尔型变量 0 或 1。

在这里，通过式(9)－式(13)的共同约束使得目标函数式(8)取得最优值。根据上述公式，本文构建了一个变量数为 $N*C$ 的 0-1 整数线性规划问题。

由以上讨论可知，该推测执行最优化模型问题不能在一个确定的多项式时间内解决。众所周知，此类 0-1 规划问题的求解是属于 NP 难问题。为了在多项式复杂度的前提下求解此类 0-1 规则问题，提出优化的智能克隆算法 ESCA(Enhanced Smart Cloning Algorithm)，通过控制可以有效地调整各个作业的副本创建数，使得系统能在满足截止期的条件下实现整个作业集群完成时间最小。

3.4　启发式算法

ESCA 算法的核心思想：首先，将所有作业的副本数初始值设为 0；然后，根据式(12)筛选出可能出现的超时作业，并通过式(9)的条件约束整体的可用资源，创建更多的副本，直到符合实时性要求的创建机制出现为止；其次，等所有作业符合实时性要求后，再根据系统资源的利用情况逐步适当增加创建的副本，使得最优化目标式(8)取得最小值，以确保整体完成时间最小。

ESCA 算法的具体实现如下：

```
1. Input:J[N],λ[N],M_L
      //J[N]表示集群中时刻 L 的作业集合
      //λ[N]表示集群 J[N]中每个作业 deadline 的集合
      // M_L 表示时刻 L 系统中的可用资源
2. Output:F[N] //F[N]表示 N 个 jobs 采用的最优副本创建机制
3. if Σ_i^N m_i ≤ M_L then
4.    return no feasible solution;/资源短缺,无法进行推测执行
5. end if
6. F[N]:={f_1(0),f_2(0),f_3(0),…,f_N(0)};
   //初始化所有作业的副本创建数;
7. while Σ_i^N m_i * f_i(j) ≤ M_L
   //满足创建副本后整体资源小于可用资源;
8.    for i:=1→N do
9.       if t_ij > λ_i then //估算完成时间,找出有拖延的作业;
10.         f_i(j)←f_i(j+1)
            //将拖延任务的副本创建数适当增加 1;
11.      end if
12.   end for
13.   if j>4 then//副本创建数阈值设定为 4;
14.   break;
15.   end if
16. end while
```

17. for j:=0→3 //利用剩余系统资源进行优化副本的创建

18. if $\sum_{i}^{N} m_i * f_i(j+1) \leqslant M_L$ then

19. $f_i(j) \leftarrow f_i(j+1)$

20. end if

21. end for

22. return F[N], $\sum_{i}^{N} t_i$

ESCA 算法的输入是在 L 时刻系统中可用的资源 M_L 和当前在队列中作业的集合 $J[N]$，$\lambda[N]$表示当前进入队列的每个作业的 Deadline。算法的第 3－5 行判断系统中是否有可用资源，如果系统中资源数量都不满足 job 中各个任务进行单独处理，那么此时系统将不可能进行推测执行，返回没有可行的解。算法的第 6－16 行是对作业分组进行副本的创建，创建开始时将所有任务都置为最低的副本数，然后估算每个作业的完成时间，基于系统中的可用剩余资源判断完成时间是否在截止时间之前，若否则增加创建的副本数并加入到 $F[N]$中，循环上述步骤，直至所有作业的估算完成时间满足截止期限为止。另外，算法的第 13－15 行判断副本创建数是否满足阈值上限。上述步骤实现各个作业满足截止期之前完成的副本创建，没有考虑剩余资源的利用。第 17－21 行进行副本创建的优化，充分利用系统的资源实现整个系统作业的完成时间最小。算法结束时，即可得到每个作业的最优化副本创建数 $F[N]$和总体完成时间。

由此可知，ESCA 算法在执行过程中的最多循环次数为 $O(CN)$，所以此时的时间复杂度为 $O(CN)$，从而提高了系统的运行效率，减少了完成时间。

4 仿真实验与性能分析

4.1 实验设计和参数设置

为验证 ESCA 算法的有效性，本节通过 Matlab 仿真推测执行环境，分析比较作业数和系统可用资源数量，同时探讨它们对完成时间的影响。部分重要参数定义如下。

1)作业数。在并行计算集群系统中，作业数的到达符合泊松分布，为了使实验计算和算法具有说服性，规定在 L 时刻作业数为 100 个左右，这样可以充分反应作业量和任务数对系统性能的影响。实验将分析作业数为 100 个时最优解和 ESCA 算法的差异性。

2)任务数。每个作业会包含多个任务，实验定义每个作业包含的任务数为 *random*(1～20)个，实验中总的任务数会不断地变化，通过大量反复实验，取实验中出现的任务总数次数最多的情况为标准进行分析。

3)作业的期望完成时间。根据文献[11]中的仿真实验可知，一般作业在不采用推测执行时的完成时间为 50s 左右，所以本文采用此标准定义每个作业的期望完成时间为 *random*(15～40)s。另外，可以根据作业的任务数适当调整该作业的期望完成时间。

4)作业进行副本创建的完成时间。根据式(3)，结合作业出现掉队任务的概率情况，估算采用不同副本创建数的任务完成时间，其中出现掉队任务的概率与任务完成时间呈线性关系，影响因子取决于系统的性能。

5)可用资源节点。考虑集群的合理性，可用资源节点数 M 取值适量方能体现 ESCA 算法的有效性。M 偏小时所有作业都不能进行推测执行副本的创建，M 过大则会造成每个作业都以最高的副本创建机制进行执行，达不到预期的效果，所以规定作业数为 100 个，在每个作业包含 1～20 个任务的条件下对应的任务量为 *random*(20～2000)，实验中可用节点 M 的取值为 1500～3800。

为了减少实验的随机性和波动性，实验结果都是经过多次反复运行的平均值。

4.2 结果分析与评价

本次仿真实验主要探讨可用资源节点数 M 和所有作业的任务数对整体完成时间的影响。首先验证 ESCA 算法是否能够取得有效的副本创建机制逼近完成时间的最优值，然后分析作业量、任务数和可用资源的变换对 ESCA 算法的影响，尤其在不同资源的情况下比较 ESCA 算法的性能。

图 1 表明 ESCA 算法在相同作业数($jobs=100$)的情况下，整体作业的完成时间是可以逼近最优解的，证明了此启发式算法的有效性。从图中发现，随着可用资源节点数的增加，副本创建机制越来越完善，ESCA 算法越来越接近最优值，并且整体完成时间在采用推测执行后明显下降。

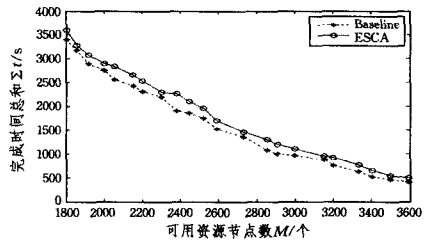

图 1 ESCA 算法与最优解的对比图

图 2 中作业数固定为 100，可用资源节点数 M 为 2000，可以看出任务数的变化对系统整体完成时间的影响较大，当作业的平均任务数达到 8 时，完成时间的增长较快，这反应了任务数的改变会大大影响系统的性能，包括推测执行的有效性。图 3 示出了 ESCA 算法与作业数、系统可用节点、完成时间之间的趋势变化关系。

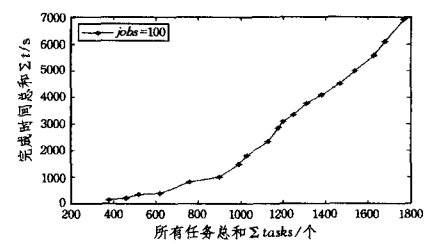

图 2 任务数与完成时间的关系图

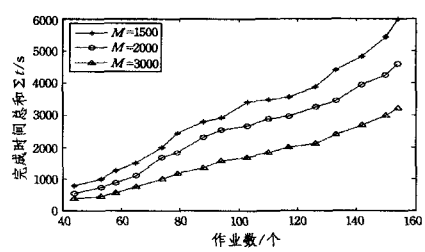

图 3 作业数与完成时间的关系图

(下转第 212 页)

[8] ISHII H, WORNELL G. OFDM blind parameter identification and transmission parameter estimation[C]// IEEE International Conference on Personal. 2005:242-247.

[9] PUNCHIHEWA A, VIJAY K B, CHARLES D. Blind estimation of OFDM parameters in cognitive radio networks[J]. IEEE Transactions on Wireless Communications, 2007, 10(3):48-54.

[10] COLERI S, ERGEN M, BAHAI A. Channel estimation techniques based on pilot arrangement in OFDM systems[J]. IEEE Transations Broadcast, 2002, 48(3):223-229.

[11] DOBRE O A, BARNESS Y, WEI S. A survey of automatic modulation classification techniques classical: approaches and new developments[J]. IET Communications, 2007, 1(2):137-156.

[12] PHAM T H, MCLOUGHLIN I V, FAHMYS A. Robust and efficient OFDM synchronization for FPGA-based radios[J]. Circuits System Signal Process, 2014, 33(8):2475-2493.

[13] HOUCKE S, CHEVREUIL A, LOUBATON P. Blind equalization: case of an unknown symbol period[J]. IEEE Transactions on Signal Processing, 2003, 51(3):781-793.

(上接第 196 页)

由仿真结果可知：

1)当作业数在一定条件下时，ESCA 算法的完成时间可以逼近推测执行的最优解，并且随着系统资源的增加，ESCA 算法在降低完成时间上的优势较为明显。在可用资源节点保证让每个作业可以有两个平均副本数的情况下，作业平均完成时间控制在 15～19s 左右。

2)每个作业所包含的任务数对系统的性能影响较大，所以作业的负载量能影响 ESCA 算法的有效性，其也是影响系统性能的因素之一。

3)系统可用资源的增加可以进一步体现 ESCA 算法的性能，使得整体的完成时间减少得较为明显。

结束语 本文在 MapReduce 的基础上，提出一种优化的推测执行 ESCA 算法。通过对系统可用资源和作业数的分析，能够对所有作业采取合适的副本创建机制，并且通过最优解与本文算法所得结果的对比可知，本文算法缩短了整体完成时间。实验表明，本文提出的推测执行算法 ESCA 是正确且有效的。进一步的研究工作可以从以下两个方面展开：

1)进一步考虑 Map 和 Reduce 两个阶段的副本优化创建方案，并且优化 MapReduce 的调度策略；

2)考虑更加完善的数据交换、存储等因素，增加节点的处理能力，考虑多个时间段的任务等待及调度过程的研究。

参 考 文 献

[1] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[C]// Proceedings of the 6th Symposium on Operating System Design and Implementation. New York: ACM Press, 2004:137-150.

[2] ANANTHANARANAN G, GHODSI A, Shenker S, et al. Effective straggler mitigation: Attack of the clones[C]// Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). 2013:185-198.

[3] SUN X, HE C, LU Y. Esamr: An enhanced self-adaptive mapreduce scheduling algorithm[C]// Proceedings of the 18th International Conference on Parallel and Distributed Systems (ICPADS). 2012:148-155.

[4] ANANTHANARANAN G, KANDULA S, GREENBERG A, et al. Reining in the outliers in mapreduce clusters using mantri [C]// Usenix Symposium on Operating Systems Design and Implementation(OSDI 2010). Canada, 2010:265-278.

[5] CHEN Q, LIU C, XIAO Z. Improving mapreduce performance using smart speculative execution strategy[J]. IEEE Transactions on Computers, 2014, 63(4):954-967.

[6] ISARD M, BUDIU M, YU Y, et al. Dryad: distributed data-parallel programs from sequential building blocks[C]// Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems. 2007:59-72.

[7] LIU C. Improving Speculative Execution and Skew Data Handing in MapReduce[D]. Beijing: Peking University, 2012. (in Chinese)
刘成. MapReduce 推测执行策略及倾斜数据处理优化[D]. 北京：北京大学，2012.

[8] BORTHAKUR D. The hadoop distributed file system: Architecture and design[J]. Hadoop Project Website, 2007, 11(11): 1-10.

[9] RAO B T, REDDY L S S. Survey on improved scheduling in Hadoop MapReduce in cloud environments[J]. International Journal of Computer Applications, 2012, 34(9):29-33

[10] WANG X Q. Optimization of High-performance MapReduce System [D]. Hefei: University of Science and Technology of China, USTC. 2010. (in Chinese)
王向前. 高性能 MapReduce 系统的优化[D]. 合肥：中国科学技术大学，2010.

[11] XU H, LAU W C. Optimization for Speculative Execution of Multiple Jobs in a MapReduce-like Cluster[C]// IEEE INFOCOM 2015-IEEE Conference on Computer Communications. IEEE, 2015:1017-1079.

[12] ZAHARIA M, KONWINSKI A, JOSEPH A D, et al. Improving MapReduce Performance in Heterogeneous Environments[C]// Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation(OSDI). San Diego, California, 2008:29-42

[13] XU H, LAU W C. Speculative Execution for a Single Job in a MapReduce-Like System[C]// Proceedings of 7th IEEE International Conference on Cloud Computing (CLOUD). 2014:586-593.

[14] KWON Y C, BALAZINSKA M, HOWE B, et al. Skewtune: mitigating skew in mapreduce applications[C]// Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012:25-36.