2020 年 2 月 第 2 期
Vol. 41 No. 2 2020

云环境下基于强化学习的多目标任务调度算法

童 钊,邓小妹,陈洪剑,梅 晶,叶 锋

(湖南师范大学 信息科学与工程学院,长沙 410012)
(高性能计算与随机信息处理省部共建教育部重点实验室(湖南师范大学),长沙 410012)
E-mail: tongzhao@ hunnu. edu. cn

摘 要: 针对云计算环境下的多目标任务调度问题,提出一种新的基于 Q 学习的多目标优化任务调度算法(Multi-objective Task Scheduling Algorithm based on Q-learning,QMTS). 该算法的主要思想是: 首先,在任务排序阶段利用 Q-learning 算法中的自学习过程得到更加合理的任务序列; 然后,在虚拟机分配阶段使用线性加权法综合考虑任务最早完成时间和计算节点的计算成本,达到同时优化多目标问题的目的; 最后,以产生更小的 makespan 和总成本为目标函数对任务进行调度,得到任务完成后的实验结果. 实验结果表明,QMTS 算法在使用 Q-learning 对任务进行排序后可以得到比 HEFT 算法更小的 makespan; 并且根据优化多目标调度策略在任务执行过程中减少了 makespan 和总成本,是一种有效的多目标优化任务调度算法.

关 键 词: 云计算; 机器学习; makespan; 任务调度; 总成本
中图分类号: TP301 **文献标识码:** A **文 章 编 号:** 1000-1220(2020)02-0285-06

Multi-objective Task Scheduling Algorithm Based on Reinforcement Learning in Cloud Environments

TONG Zhao,DENG Xiao-mei,CHEN Hong-jian,MEI Jing,YE Feng
(College of Information Science and Engineering,Hunan Normal University,Changsha 410012,China)
(Key Laboratory of High Performance Computing and Stochastic Information Processing,Ministry of Education of China (Hunan Normal University),Changsha 410012,China)

Abstract: In order to solve the multi-objective task scheduling problem in the cloud environment,a novel multi-objective optimization task scheduling algorithm based on Q-learning (Multi-objective Task Scheduling Algorithm based on Q-learning,QMTS) is proposed. The main idea of the QMTS algorithm is: firstly,in the task ordering phase,uses the self-learning process in the Q-learning algorithm to obtain a more reasonable task order; secondly,in the virtual machine allocation phase,uses the linear weighting method to consider the task's earliest finish time and computation cost of computing nodes to optimize multi-objective task scheduling problems; finally,the task is scheduled with the objective function of generating smaller makespan and total cost,and the experimental results are obtained after tasks are completed. The experimental results show that QMTS algorithm can obtain a smaller makespan after sorting tasks by using Q-learning; and it reduces the makespan and total cost in the task execution process according to the optimized multi-objective scheduling strategy. QMTS algorithm is an effective multi-objective optimization task scheduling algorithm.

Key words: cloud computing; machine learning; makespan; task scheduling; total cost

1 引 言

随着计算机科学和互联网技术的快速发展,云计算作为一种新的并行与分布式计算技术的发展方向被提出来[1]. 云计算平台上的所有数据计算和存储都在云端进行,可以通过使用最少的代价来充分利用互联网上的资源,以实现资源利用率最大化的目的[2]. 通常,根据用户所需要的服务模式的不同,云计算可以分为三种类型: 基础设施即服务(IaaS)、平台即服务(PaaS)和软件即服务(SaaS)[3]. 在云计算平台上,如何将用户提交的请求合理的分配给云计算节点进行处理,是影响云用户的服务体验和云服务提供商的利润率的重要因素,也是任务调度要解决的核心问题. 云计算中的任务调度是一个 NP 难问题,根据所要解决的云用户需求的不同,任务调度算法往往侧重于不同的优化目标,主要包括: 调度长度(makespan)、负载均衡(load balance)、服务质量(Quality of Service,QoS)和经济原则(economical principle)等[4].

目前,大多数对于任务调度的研究只侧重了对单目标问题的优化,对于多目标调度问题由于其复杂性,研究相对较少. 但在现实问题中,只做到对任务调度问题进行单目标的优化通常是不够的,多数情况下需要同时考量云用户的用户体验和云服务提供商的消耗成本等多方面的因素进行决策. 因此,本文考虑同时对任务的完成时间和虚拟机的执行成本这

收稿日期: 2019-04-10 收修改稿日期: 2019-05-17 基金项目: 国家自然科学基金青年项目(61502165,61602170)资助; 湖南省教育厅一般项目(17C0959)资助. 作者简介: 童 钊,男,1985 年生,博士,副教授,CCF 会员,研究方向为云计算; 邓小妹,女,1995 年生,硕士研究生,研究方向为云计算、机器学习; 陈洪剑,男,1993 年生,硕士研究生,研究方向为并行算法、人工智能; 梅 晶,女,1988 年生,博士,讲师,CCF 会员,研究方向为分布式计算、组合优化; 叶 锋,男,1996 年生,硕士研究生,研究方向为云计算、目标优化.

两个有重要意义并且相互冲突的目标进行优化，拟解决在满足云用户对减少 makespan 的需求的基础上，同时保证对云服务提供商的成本消耗的控制.

针对云环境下的任务调度算法的研究，Topcuouglu 等[5]提出了优化 makespan 的异构最早完成时间(Heterogeneous Earliest Finish Time，HEFT)算法. HEFT 算法在小规模的有向无环图(Directed Acyclic Graph，DAG)上可以得到较好的性能，但是随着任务规模的增大，DAG 任务的复杂性增加，导致 HEFT 算法的性能随之下降；并且该算法只考虑了对单目标 makespan 的优化，没有考虑其他影响任务调度性能的因素，使得该算法在实际应用当中具有一定的局限性. 李君等[6]提出一种综合时间能耗成本的任务调度算法(Time and Energy Consumption Cost Scheduling，TECCS)，通过引入通信因子和计算因子综合时间与能耗成本来决定任务的调度顺序，在期望完成时间条件下节省更多任务执行成本；但是该算法考虑的是在期望完成时间的条件下实现对能耗成本的优化，没有实现对双目标的同时优化. 可以得知，传统的任务调度算法存在着对不同的任务类型适应度低、搜索时间长以及容易陷入局部最优解等不足.

目前，机器学习已经开始被用于有效的解决任务调度这类组合优化问题[7-9]. 机器学习的研究最早从 Samuel 发明的下棋程序开始，发展至今已经被广泛用于解决诸多领域的研究问题，是指机器通过学习数据的内在信息，获得自身的经验信息来指导其完成智能行为[10]. 强化学习(Reinforcement Learning，RL)是建立在马尔科夫决策过程(Markov Decision Process，MDP)上的一种重要的机器学习算法，它是指在无外界监督指导的情况下通过与不确定的外部环境进行交互，从而获得最优解的一种学习过程[11,12]. 经典的 RL 算法包括 Q-learning、SARSA 和策略梯度(Policy Gradients)，其中，Q-learning 算法在解决任务调度问题上具有较好的效果. Q-learning 算法是指通过引入期望延迟反馈来解决无完整信息的 MDP 问题的一种方法，它不需要提前获取环境模型，主体(Agent)是 Q-learning 中的一个重要组成元素，它可以根据历史经验来学习并且选择一个最优的动作，最终被选择的一系列动作组成一个最优策略[13].

Siar 等[14]提出了一种将 RL 和智能技术结合的方法用于解决并行系统中的任务调度问题，Agent 通过 Q-learning 方法来学习如何将效率最大化，并且使用一种新的协作 Q-learning 算法来促进 Agent 之间的交互，从而提高整个系统的效率. Cui 等[15]提出一种新的基于 Q-learning 的任务调度算法，在一定数量的虚拟机资源下最小化 makespan 和平均等待时间. Wei 等[16]提出一种基于 Q-learning 和共享值函数的任务调度算法，以解决现有合作学习算法中信息频繁交换的问题，该算法设计了任务模型和 Q-learning 模型，在约束条件下降低了协作信息的切换频率，具有良好的学习效果.

本文针对 HEFT 算法只适合解决小规模 DAG 任务的调度问题以及只考虑了对单目标 makespan 的优化，提出一种基于 Q-learning 的多目标任务调度算法(Multi-objective Task Scheduling Algorithm based on Q-learning，QMTS). 在 QMTS 算法中，首先利用 Q-learning 算法指导任务进行优先级排序，其中使用 HEFT 算法的 Upward Rank 值(式(9))作为 Q-learning 更新过程中的立即奖励值，可以得到比 HEFT 算法更小的 makespan；其次针对 HEFT 算法在处理机分配阶段只考虑将任务的最早完成时间作为影响分配的因素，从而导致云服务提供商的消耗成本没有得到有效的控制. 本文使用线性加权法将最早完成时间和消耗成本综合考虑作为虚拟机的分配依据. 实验结果表明，本文所提出的 QMTS 算法得到了比 HEFT 及其他经典的调度算法更好的多目标优化结果.

2 问题描述

2.1 云任务调度问题

虚拟化是云计算中的核心，云计算平台上的每个计算节点代表了具有不同处理能力的虚拟机，定义如下:

$$V=\{V_1,V_2,\cdots,V_m\} \tag{1}$$

在本文中，拟实现在云计算环境下任务之间有依赖关系的多目标静态任务调度算法，有依赖关系的任务类型通常使用 DAG 任务来表示，定义如下:

$$G=(T,E,C,W) \tag{2}$$

其中: $T=\{T_1,T_2,\cdots,T_n\}$ 表示 DAG 任务上 n 个节点(即任务)的集合; $E=\{edge_{T_i,T_j}\mid T_i,T_j\in T,i<j\}$ 表示任务之间所有相连边的集合，T_i 是 T_j 的直接前驱任务节点(也称为 T_j 的父节点); C 是 DAG 任务中有边相连的任务之间的通信时间集合，例如任务 T_i 和任务 T_j 之间的通信时间可以表示为 $c_{i,j}$，计算公式为: $c_{i,j}=l+d_{i,j}/b$，l 是虚拟机的通信启动时间，$d_{i,j}$ 是从任务 T_i 传输到任务 T_j 的数据量，b 是虚拟机之间的通信速率(在本文中，假设在不同虚拟机上的 l 都等于 0 以及 b 都相同); W 是所有任务的平均计算时间的集合，任务 T_i 在虚拟机 V_k 上的计算时间表示为 $w_{i,k}$，计算公式为: $w_{i,k}=d_i/s_k$，d_i 是任务 T_i 的计算量，s_k 是虚拟机 V_k 的计算速度；任务的平均计算时间表示为: $\overline{w_i}=\sum_{k=1}^{m}w_{i,k}/m$，$m$ 为虚拟机总个数.

一个简单的 DAG 任务模型如图 1 所示，图中共有 8 个任务节点，每个任务节点由两个部分组成: 任务节点编号和该任务在所有虚拟机上的平均计算时间. 任务图中有向边上的数值代表两个任务之间的通信时间.

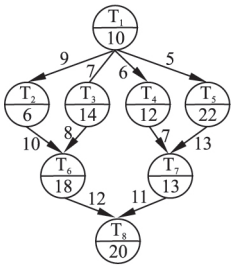


图 1 一个 8 个任务的简单 DAG 模型

Fig. 1 A simple DAG model with 8 tasks

2.2 优化目标模型

在本文中，拟实现同时减少任务的 makespan 和虚拟机消耗总成本的多目标任务调度算法(QMTS).

2.2.1 makespan 模型

在云环境下任务调度问题的研究中，makespan 一直是一个重要的评价指标. 对于给定的任务集合，最后一个任务的完

成时间即为整个任务调度的 makespan 结果. 任务 T_i 在虚拟机 V_k 上的最早开始执行时间定义为:

$$EST(T_i,V_k)=\max\left(\max_{T_q\in pred(T_i)}(FT(T_q)+c_{q,i}),ST(V_k)\right) \tag{3}$$

其中: $pred(T_i)$ 是任务 T_i 的所有直接前驱任务节点的集合, $FT(T_q)$ 是任务 T_q 的完成时间; $ST(V_k)$ 是虚拟机 V_k 最早可开始执行任务 T_i 的时间. 根据定义, DAG 任务中入口任务的最早开始执行时间为 0.

任务 T_i 在虚拟机 V_k 上的最早完成时间定义为:

$$EFT(T_i,V_k)=w_{i,k}+EST(T_i,V_k) \tag{4}$$

可以得知,最后一个任务的最早完成时间即为整个 DAG 任务调度的 makespan,表示为:

$$makespan=\max_{i=1,2,\cdots,n}(EFT(T_i)) \tag{5}$$

2.2.2　总成本模型

总成本(total cost) 是指完成所有任务之后,在所有虚拟机上所耗费成本的总和. 定义如下:

$$total\ cost=\sum_{k=1}^{m}price_k\times time_k \tag{6}$$

其中: $price_k$ 是虚拟机 V_k 在计算单位时长的任务上的执行费用. 通常,计算速度越快的虚拟机所需的单位执行费用越高; $time_k$ 是分配到虚拟机 V_k 上的任务的总执行时长. 对于 $price_k$ 的定义有多种方式,如线性定价方式和指数定价方式[17],本文给出一种定价方式如下所示:

$$price_k=price_h\times\left(\frac{s_k}{s_h}\right)\times\left(1+\ln\left(\frac{s_k}{s_h}\right)\right) \tag{7}$$

式(7) 的推导思路为: 首先给出不同计算速度的虚拟机之间的单位执行费用的线性关系为 $price_k=price_h\times(s_k/s_h)$, $k=h+1\leqslant m$,这里将所有虚拟机按照计算速度从小到大进行排序, $price_h$ 是虚拟机 V_h 的单位执行费用, s_k 和 s_h 分别是虚拟机 V_k 和 V_h 的计算速度; 其次由于在线性关系中不同虚拟机处理相同的任务实际上消耗的成本是相同的,因此引入相关的函数来解决这个问题; 最后考虑到在指数定价方式中虚拟机的计算速度加快时,其单位执行费用的增长过快. 于是引入对数函数来避开现有的两种定价方式的不足,由于虚拟机 V_k 的计算速度大于 V_h,因此 $(s_k/s_h)>1$,得到 $ln(s_k/s_h)>0$,即 $1+ln(s_k/s_h)>1$,使得虚拟机价格增加速度与对数函数相似,更符合实际生活中的定价模式. 由此可知,单个任务 T_i 在虚拟机 V_k 上的执行成本表示为: $cost_{T_i}=w_{i,k}\times price_k$.

2.2.3　优化目标函数

本文提出的算法试图对 makespan 和总成本(total cost) 同时进行优化,可将目标函数定义为:

$$\begin{aligned}&\min:\{makespan,total\ cost\}\\&s.t.\ makespan\leqslant makespan_{\min}\\&total\ cost\leqslant total\ cost_{\min}\end{aligned} \tag{8}$$

其中, $makespan_{\min}$ 和 $total\ cost_{\min}$ 分别是在本文其他 3 种对比算法中产生的最小的 makespan 结果和最小的总成本结果.

3　多目标调度算法设计

本章结合 HEFT 算法与 Q-learning 算法提出了同时实现最小化 makespan 和总成本的多目标任务调度算法 QMTS,达到用户体验最优和云服务提供商总成本最小的目的.

3.1　HEFT 算法

HEFT 是静态 DAG 任务调度中的经典算法之一,对于减少整个任务执行过程的 makespan 有着较好的表现. 算法在任务排序阶段,使用 Upward Rank 值(式(9)) 作为任务优先级值,充分考虑到了任务的计算时间、任务间的通信时间以及子任务的优先级对父任务优先级的影响; 在分配处理机阶段, HEFT 算法使用最早完成时间策略(式(4)),即将每个任务都分配到能使其最早完成的处理机上执行.

Upward Rank 值的定义如下所示:

$$rank_u(T_i)=\overline{w_i}+\max_{T_p\in succ(T_i)}(\overline{c_{i,p}}+rank_u(T_p)) \tag{9}$$

其中: T_p 是任务 T_i 的直接后继任务节点,并且根据式(9), DAG 任务中的出口任务 T_{exit} 的 Upward Rank 值为 $\overline{w_{exit}}$.

HEFT 算法的执行过程如图 2 所示.

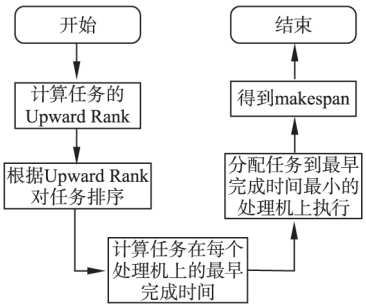


图 2　HEFT 算法流程图

Fig. 2　Flow chart of the HEFT algorithm

HEFT 算法虽然在任务排序和处理机分配阶段都有着较充分的考虑,但是由于其只依据 Upward Rank 值进行了一次性的排序,导致在任务节点多且结构较复杂的 DAG 任务上往往得不到合理的任务执行顺序,以至于在产生最优 makespan 上的性能下降; 并且在处理机分配阶段只考虑了优化单目标 makespan,没有受到任务完成后处理机所消耗成本的约束,而在现实问题中,成本也是很重要的考虑因素.

3.2　Q-learning 算法

Q-learning 是强化学习中经典的无监督、自学习方法,在学习过程中不需要外界环境对它进行指导,依据与环境"试错"交互得到的历史经验来指导学习方向. Q-learning 有五个重要组成部分: 主体(Agent) 、环境/状态(s) 、动作(a) 、奖励值(r) 以及一个用于存储 Q 值的 Q 表(Q-table) . 在进行一次学习之前,首先将 Q 表初始化(一般初始化为 0),表的横向和纵向分别代表动作和状态,表中元素即为 Q 值,表示为 $Q(s,a)$. 在每一步的学习更新过程中, Agent 通过随机选择一个可行动作 a 从当前状态 s 转移到下一个状态 s',并且新的环境会反馈一个正的或者负的奖励值 r 给 Agent 用来更新 Q 值. Q-learning 算法简化模型如图 3 所示.

Q-learning 算法的更新公式如下:

$$\begin{aligned}Q_{t+1}(s,a)&=Q_t(s,a)+\\&\alpha(r+\gamma\max_{a'}Q_t(s',a')-Q_t(s,a))\end{aligned} \tag{10}$$

其中: $Q_t(s,a)$ 表示更新前的 Q 值, $Q_{t+1}(s,a)$ 表示更新后的 Q 值; $\alpha(0<\alpha\leqslant 1)$ 是学习率(learning rate),表示 Agent 在学习过程中对未来收益的重视程度. α 值越小,表示 Agent

越重视当前已经获得的收益;反之,表示越重视远期获得的收益;$\gamma(0 < \gamma \leqslant 1)$ 是折扣因子(discount rate),表示对未来奖励值的重视程度,γ值越小表示越重视立即奖励;反之,表示越重视未来奖励.

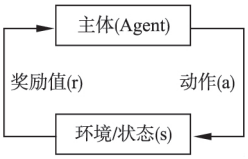


图3 Q-learning 算法简化模型

Fig. 3 Simplified model of Q-learning algorithm

3.3 QMTS 算法设计

针对所要解决的任务调度中的 makespan 和总成本多目标优化问题,本文将算法设计分为任务排序和虚拟机分配两个主要阶段,QMTS 算法的伪代码如算法 1 所示.

算法 1. QMTS 算法

输入: DAG 任务中的所有任务.

输出: makespan 和总成本的结果.

初始化 Q 表的所有元素为 0

初始化参数

计算立即奖励值 r

repeat for each episode

  随机选择一个入口任务作为当前任务 $T_{current}$

  repeat for each step of episode

    随机选择一个合法任务(不包括已经被选择过的任务)作为下一个任务 T_{next}

    根据式(10)更新 $Q(T_{current}, T_{next})$

    $T_{current} \leftarrow T_{next}$

  until T_{next} *is terminal*

  根据更新后的 Q 表获得任务排序

  根据式(11)给每个任务分配一个虚拟机

  得到 makespan 和总成本结果

until *convergence* (*makespan* 和总成本的结果不再改变)

3.3.1 任务排序阶段

在本文中,使用 Q-learning 算法对任务进行排序,将任务视为 Q-learning 中的状态和动作,从当前任务到下一个任务的排序过程看作 Agent 在当前状态选择一个可行动作后转移到下一个状态的过程. 在 Q 表的更新过程中,将每个任务的 Upward Rank 值作为该任务对应状态的立即奖励值,经过一定次数的迭代,Q 表最终会收敛到固定不变. 根据收敛的 Q 表,使用最大 Q 值优先原则对任务进行排序,即每次都选择符合 DAG 任务的依赖关系中的 Q 值最大的任务作为下一个执行任务,直至所有任务都完成排序过程. 由于 Q-learning 算法在寻优问题上具有很好的学习效果,上述排序方法可以得到一个比 HEFT 算法更加合理的任务执行顺序,从而产生更小的 makespan.

3.3.2 虚拟机分配阶段

在得到任务执行顺序之后,使用分配策略将每个任务映射到虚拟机上执行. 本文将任务在每个虚拟机上的最早完成时间和虚拟机的消耗成本通过线性加权的方式进行综合考虑,每次将任务分配到加权和最小的虚拟机上执行,使得在产生更小的 makespan 的同时做到总成本的下降,具体公式定义如下所示:

$$sum_{T_i} = \min_{k=1}^{m}(\omega_1 \times EFT(T_i, V_k) + \omega_2 \times cost_{T_i, V_k}) \quad (11)$$

其中,ω_1 和 ω_2 分别代表任务 T_i 在虚拟机 V_k 上的最早完成时间和消耗成本的权重系数,并且 $\omega_1 + \omega_2 = 1$.

4 实验结果与分析

4.1 实验概述及参数设置

本文中的所有实验都在云计算仿真平台 WorkflowSim 上进行[18]. 首先通过实验证明了在任务优先级排序阶段结合 Q-learning 算法对产生更小的 makespan 结果是有显著效果的,这是由于在 Q-learning 算法中 Agent 的自我学习调节过程产生了一个更好的任务执行顺序;其次,根据式(11)中权重系数的变化,分析了其对 makespan 和总成本两个目标的影响;再次,分别测试了在不同虚拟机数量下 4 种算法的性能,得到 QMTS 算法性能最好;最后,对不同节点数量的 DAG 任务进行实验,测试了 QMTS 算法在小任务图和大任务图上都有较好的表现,尤其是在处理规模较大的任务集时有着突出的性能优势.

本文实验参数设置如表 1 所示. 首先学习率α和折扣因子γ依据由大量实验所得的经验确定一个较好的取值范围,其次在这个取值范围内将这两个参数进行两两组合进行实验,最后得到学习率α和折扣因子γ分别为 1.0 和 0.8 时,本文的实验效果最佳. 为了验证实验结果的鲁棒性,进行了在虚拟机数量和任务数量上的扩展实验,这两个参数中的各个取值遵循数值差异化的原则. Inspiral 测试集是在其他相关研究中使用较多的测试集.

表 1 实验参数设置

Table 1 Experimental parameter settings

参数	值
学习率α	1.0
折扣因子γ	0.8
虚拟机数量	4,8,16,31
任务数量	30,50,100,150,200,300
测试集	Inspiral

4.2 基于 Q-learning 算法的任务排序实验

在云计算环境下的任务调度问题中,如何以更加合理有效的顺序将任务提交给处理机执行是调度器首先要解决的一个问题,对虚拟机最终完成任务的 makespan 大小起着至关重要的影响. 本文使用 Q-learning 算法对任务进行优先级排序,并且本实验在分配任务给虚拟机时使用与 HEFT 算法一致的分配方式,测试了在 16 和 32 个虚拟机数量下的 makespan 结果,并与文献[5]中的基于 Upward Rank 的异构最早完成时间(Heterogeneous Earliest Finish Time based on Upward Rank, HEFT_U)算法、基于 Downward Rank 的异构最早完成时间(Heterogeneous Earliest Finish Time based on Downward Rank, HEFT_D)算法和处理器上的关键路径(Critical Path on a Processor, CPOP)算法进行比较. HEFT_U、HEFT_D 和 CPOP 算法是经典并且性能较好的静态 DAG 任务调度算法,在优化

makespan 上有着较好的表现,在很多静态任务调度的研究中都被作为对比算法.

从图 4 的实验结果可以得知,任务集在使用了 Q-learning 算法对其进行执行顺序的排序后,得到了比其他 3 种算法更好的 makespan 结果,表明本文所提出的 QMTS 算法使用 Q-learning 算法对任务进行优先级排序对于减少 makespan 有着显著的效果.

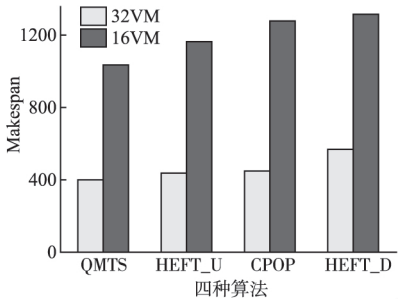


图 4　在 16 和 32 个虚拟机数量下的 makespan 结果

Fig. 4　Makespan with 16 and 32 virtual machines

4.3　权重系数实验

在对多目标优化问题进行线性加权求解时,目标函数中每个目标对象所占的权重大小情况直接影响了目标优化结果的好坏. 通常情况下,当某个目标所占的权重值越大,则表示该目标越受重视,相应的目标函数会更偏向于对它的优化,并可能会直接导致其他目标产生不理想的结果. 在本实验中,首先通过大量的实验,选取了在不同权重系数情况下两个目标的实验结果,权重变化对 makespan 和总成本的影响趋势如图 5 所示.

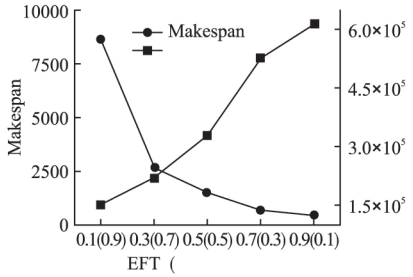


图 5　权重变化对 makespan 和总成本的影响

Fig. 5　Impact of weight changes on makespan and total cost

从图 5 中可以得知,随着 EFT 的权重从小到大(成本权重由大到小)的变化,makespan 产生从大到小的变化趋势,而总成本产生从小到大的变化趋势. 可知,在对 makespan 和总成本同时进行优化时,如果对其中一个目标赋予相对过大的权重,则另一个目标会因此产生劣解,表明 makespan 和总成本是一对相冲突解. 因此,在实验中需要选择合适的权重组合对 makespan 和总成本进行同时优化. 在本文中,首先根据 makespan 和总成本结果随权重变化的大致趋势将不可能达到双目标优化目的,即会产生过差的单个目标解的权重组合剔除掉,剩下的权重组合组成一个区间,然后对这个区间内的权重组合进行大量实验,最后找到相对最优的多目标实验结果.

4.4　Inspiral_100 测试集实验

本实验使用 Inspiral_100 测试集分别在 4 种不同虚拟机数量下测试了 QMTS 算法对 makespan 和总成本同时进行优化的性能,并与其它 3 种经典的任务调度算法进行比较.

图 6(a) 至图 6(d) 分别是在 4VM、8VM、16VM、32VM 情况下四种算法优化 makespan 和总成本的结果. 图中横坐标表示 4 种任务调度算法,左纵坐标表示 makespan 结果,右纵坐标表示总成本结果. 从图 6 可以得知,除了在 32VM 下,CPOP 算法产生的总成本结果有略微的优势之外. 在余下情况中,本文提出的 QMTS 算法都得到了比其他 3 种算法更小的 makespan 和总成本结果. 这是因为: 首先,QMTS 算法在任务排序阶段使用 Q-learning 算法找到了一个比其他 3 种算法更加合理的任务执行顺序,从而产生了比其他算法更小的 makespan 结果; 其次,又在虚拟机分配阶段综合考虑了任务最早完成时间和虚拟机执行成本两个影响因素,避免了在虚拟机分配过程中只重视对 makespan 的减少而忽略了对总成本消耗的控制.

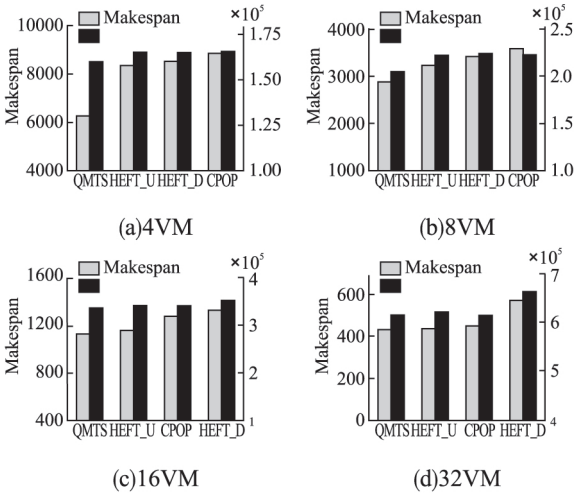


图 6　不同虚拟机数量下 4 种算法的 makespan 和总成本结果

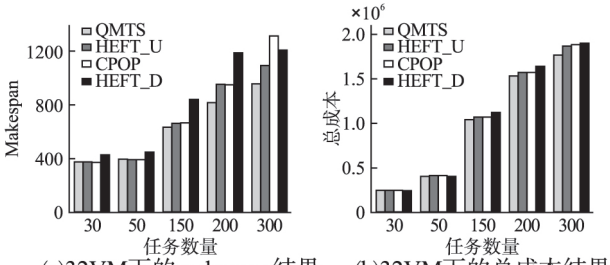
Fig. 6　Makespan and total cost of four algorithms under different numbers of virtual machines

4.5　Inspiral_100 测试集下不同任务数量实验

最后,本文还对其他任务数量不同的 Inspiral 测试集在 32 个虚拟机环境下进行实验,表明 QMTS 算法在不同规模的 DAG 任务上的可扩展性. 不同任务数量下 4 种算法优化 makespan 和总成本的结果如图 7 所示.

图 7(a) 和图 7(b) 分别展示了在 30、50、150、200 以及 300 个任务数量下 4 种算法的 makespan 和总成本优化结果,可以得知随着任务数量的增多,相应的 makespan 和总成本结果也随之增大. 在 30 和 50 个任务数量下,HEFT_D 算法的总成本结果相对最好,其次是 QMTS 算法总成本结果优于其他 2 种算法; 但是 QMTS 算法的 makespan 结果远小于 HEFT_D 算法,并且和其他 2 种算法差不多,因此 QMTS 算法在小规模任务上对两个目标的优化结果是最好的. 在 150 ~ 300 个任务的规模较大的 DAG 任务上,随着任务数量的增多,QMTS 算法产生的 makespan 结果和总成本结果明显小于其他算法,说明 QMTS 算法随着任务数量的增多,优化性能越好. 因此,本

文提出的 QMTS 算法在不同规模的 DAG 任务上都能得到最好的 makespan 和总成本的多目标优化结果. 但是由于 Q-learning 算法中的 Q 表在本文中是一个二维数组,随着任务数量和虚拟机数量的增多,相应的 Q-learning 中的状态 s 和动作 a 的数量也会增多,导致 Q 表空间增长快速,内存开销大; 并且遍历整张 Q 表需要的时间较长,CPU 计算速度随之变慢.



(a)32VM下的makespan结果 (b)32VM下的总成本结果

图 7 不同任务数量下 4 种算法的 makespan 和总成本结果

Fig. 7 Makespan and total cost of four algorithms under different task numbers

5 结 语

本文针对在云计算环境下任务调度问题中的两个重要评价指标: makespan 和总成本,提出同时优化这两个目标的基于 Q-learning 算法的静态任务调度算法 QMTS. 该算法首先使用 Q-learning 算法对任务进行优先级排序,并将 HEFT 算法的 Upward Rank 值作为 Q-learning 更新过程中的立即奖励值,得到一个更加合理有效的任务执行顺序; 待所有任务完成排序之后,任务依次进入虚拟机分配阶段,计算任务在每个虚拟机上的最早完成时间和执行成本的线性加权和,并将任务分配给加权和最小的虚拟机执行. 实验结果表明,QMTS 算法在任务调度的多目标优化问题上性能有着突出的表现,说明该算法是有效的. 在以后的工作中,我们将考虑使用深度神经网络拟合 Q 值的方法解决在 Q-learning 算法更新过程中 Q 表过大导致收敛过慢的问题; 并且在优化 makespan 和总成本两个目标的基础上考虑对其他有重要价值意义的目标进行优化.

References:

[1] La Fratta P A, Kogge P M. Heterogeneity in parallel and distributed computing [J]. Journal of Parallel & Distributed Computing, 2013, 73(12): 1523-1524.

[2] Lu Jia-wei, Li Jie, Zhang Yuan-ming, et al. Research on the load balancing scheduling policy of tasks in cloud computing environments based on improved Tabu Search [J]. Journal of Chinese Computer Systems, 2018, 39(10): 2254-2259.

[3] Li Chao, Dai Bing-rong, Kuang Zhi-guang, et al. Research on task scheduling with multiple constraints based on genetic algorithm in cloud computing environment [J]. Journal of Chinese Computer Systems, 2017, 38(9): 1945-1949.

[4] Tong Zhao, Xiao Zheng, Li Ken-li. A queueing model and probabilistic scheduling for multi-user network applications [J]. Acta Electronica Sinica, 2016, 44(7): 1679-1688.

[5] Topcuoglu H, Hariri S, Wu M. Performance-effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Transactions on Parallel & Distributed Systems, 2002, 13(3): 260-274.

[6] Li Jun, Yin Xiao-long, Wan Ming-xiang. Task scheduling based on time and energy consumption cost in heterogeneous cloud [J]. Computer Technology and Development, 2014, 24(8): 121-125.

[7] Tong Z, Xiao Z, Li K, et al. Proactive scheduling in distributed computing-a reinforcement learning approach [J]. Journal of Parallel & Distributed Computing, 2014, 74(7): 2662-2672.

[8] Xu C Z, Rao J, Bu X. URL: a unified reinforcement learning approach for autonomic cloud management [J]. Journal of Parallel & Distributed Computing, 2012, 72(2): 95-105.

[9] Xiao Z, Tong Z, Li K, et al. Learning non-cooperative game for load balancing under self-interested distributed environment [J]. Applied Soft Computing, 2017, 52(3): 376-386.

[10] Zhang Run, Wang Yong-bin. Research on machine learning with algorithm and development [J]. Journal of Communication University of China: Science and Technology, 2016, 23(2): 10-18.

[11] Arulkumaran K, Deisenroth M P, Brundage M, et al. Deep reinforcement learning: a brief survey [J]. IEEE Signal Processing Magazine, 2017, 34(6): 26-38.

[12] Tong Zhao, Xiao Zheng, Li Ken-li, et al. Scheduling algorithm in distributed systems based on non-cooperative game [J]. Journal of Hunan University (Natural Sciences), 2016, 43(10): 139-147.

[13] Watkins C J C H, Dayan P. Q-learning [J]. Machine Learning, 1992, 8(3-4): 279-292.

[14] Siar H, Nabavi S H, Shamshirband S. Static task scheduling in cooperative distributed systems based on soft computing techniques [J]. Australian Journal of Basic & Applied Sciences, 2010, 4(6): 1518-1526.

[15] Cui D, Peng Z, Xiong J, et al. A reinforcement learning-based mixed job scheduler scheme for grid or IaaS Cloud [J]. IEEE Transactions on Cloud Computing, 2017, 14(99): 1.

[16] Wei Z, Zhang Y, Xu X, et al. A task scheduling algorithm based on Q-learning and shared value function for WSNs [J]. Computer Networks, 2017, 126(10): 141-149.

[17] Zhu Li-ling, Yang Zhi-ying. A multi-objective scheduling algorithm of many tasks in cloud platforms based on method of VOO [J]. Computer Technology and Development, 2017, 27(1): 11-15.

[18] Chen W, Deelman E. WorkflowSim: a toolkit for simulating scientific workflows in distributed environments [C]//2012 IEEE 8th International Conference on E-Science (eScience 2012), IEEE, 2012: 1-8.

附中文参考文献:

[2] 陆佳炜,李 杰,张元鸣,等. 基于改进禁忌搜索的云任务负载均衡调度策略研究 [J]. 小型微型计算机系统, 2018, 39(10): 2254-2259.

[3] 李 超,戴炳荣,旷志光,等. 云计算环境下基于改进遗传算法的多维约束任务调度研究 [J]. 小型微型计算机系统, 2017, 38(9): 1945-1949.

[4] 童 钊,肖 正,李肯立. 分布式系统中多用户网络应用的概率型调度算法研究 [J]. 电子学报, 2016, 44(7): 1679-1688.

[6] 李 君,殷小龙,万明祥. 异构云中综合时间能耗成本的任务调度算法 [J]. 计算机技术与发展, 2014, 24(8): 121-125.

[10] 张 润,王永滨. 机器学习及其算法和发展研究 [J]. 中国传媒大学学报: 自然科学版, 2016, 23(2): 10-18.

[12] 童 钊,肖 正,李肯立,等. 分布式系统中基于非合作博弈的调度算法 [J]. 湖南大学学报(自然科学版), 2016, 43(10): 139-147.

[17] 朱丽玲,杨智应. 基于 VOO 方法的云计算平台多目标任务调度算法 [J]. 计算机技术与发展, 2017, 27(1): 11-15.