计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201000023

一种基于三支决策的云任务调度优化算法

王 政 姜春茂
哈尔滨师范大学计算机科学与信息工程学院 哈尔滨 150025
(hsdrose@126.com)

摘 要 云任务调度作为云计算体系的一个重要组成部分,其调度策略的效果直接影响到云平台资源利用率及用户服务质量。为解决当前云调度策略中 Min-Min 算法和 Ma-Min 算法容易因云任务分布导致负载不均衡、资源综合使用率低和任务总体完成时间较大等问题,提出一种基于三支决策的云任务调度优化算法(Cloud Task Scheduling Algorithm based on three-Way Decision,CTSA-3WD)。根据云任务的执行时间和计算资源的实际情况来标定任务集合中的轻负载任务和重负载任务。借鉴三支决策基本思想,根据两种任务在其任务集合中所占比例进行三支划分,有针对性地对划分后的 3 个任务集合设计合适的调度策略:针对轻负载任务占比高的任务集合,使用 Max-Min 算法;针对重负载任务占比高的任务集合,使用 Min-Min 算法;针对轻重负载任务接近的任务集合,采用基于 Min-Min 和 Max-Min 的改进任务调度算法。对分配完毕的节点中的关键资源进行重新调度,在满足总体完成时间减少的前提下选择最匹配的任务分配给轻负载资源。CloudSim 仿真平台的实验结果表明,所提出的云任务调度优化算法(CTSA-3WD)相比 Min-Min,Max-Min 及选择调度算法可以有效提高整体资源利用率,提升了用户的服务质量,同时也使得整个系统中的资源达到更好的负载均衡水平。

关键词:三支决策;云计算;任务调度;多粒度;负载均衡
中图法分类号 TP301

Cloud Task Scheduling Algorithm Based on Three-way Decisions

WANG Zheng and JIANG Chun-mao
School of Computer Science and Information Engineering,Harbin Normal University,Harbin 150025,China

Abstract As an essential component of the cloud computing system,task scheduling directly impacts resource utilization and service quality. To solve the problems existing in Min-Min and Max-Min algorithms in the current cloud platform,such as load imbalance,low comprehensive resource utilization,and sizeable overall task completion time due to task distribution,a task scheduling optimization algorithm based on the three-way decision (CTSA-3WD) is proposed. First,the algorithm divides tasks into light-load and heavy-load tasks according to their execution time and computational resource requirements. Secondly,the algorithm divides the tasks into three categories according to the proportion of the task set's two types of tasks. It develops scheduling strategies for these three task sets. Specifically,the strategy uses the Max-Min algorithm for tasks with a high percentage of light load tasks and uses the Min-Min algorithm for a high proportion of heavily loaded tasks. An improved task scheduling algorithm based on Min-Min and Max-Min is used for the set,which has close numbers between light and heavy-duty tasks. Third,the critical resources in the allocated nodes are rescheduled. The algorithm selects the best matching tasks to be allocated to the light-load resources,subject to the overall completion time reduction. The experimental based on the CloudSim reveals that the CTSA-3WD algorithm can effectively improve the overall resource utilization and quality of service to users compared to Min-Min,Max-Min,selective scheduling algorithms. Moreover,it also makes the resources in the whole system reach a better load-balancing level.

Keywords Three-way decisions,Cloud computing,Task scheduling,Multi-granularity,Load balancing

1 引言

云计算作为一种按需服务、按量计费的计算范式,为用户提供各类资源服务,如 CPU、内存、网络带宽等[1-2]。目前,复杂繁多的工程研究、科学计算以及商业应用都需要大量的计算资源进行数据或业务的处理,而云计算自动把大型的计算、业务程序分解成更小的子程序,通过其网络编程将任务提交给多个服务器或虚拟机进行处理,最终返回给用户[3-4]。这其中的云任务调度是云计算中的主要挑战,即如何将任务合理地调配到不同的计算资源上。合理的任务调度算法可以显著提高用户服务质量(QoS),缩短总任务完成时间(MakeSpan),提高资源利用率,实现负载均衡,使之拥有良好的性能。满足资源约束的任务调度是一个 NP-hard 问题[5],迄今为止,尚没有一个任务调度算法能够在可接受的时间内获得大规模云任务调度问题的最优解。因此,云任务调度算法的优化成为了诸多学者研究的主要方向。

基金项目:黑龙江省自然科学基金(JJ2020LH0473)
This work was supported by the Natural Science Foundation of Heilongjiang Province,China(JJ2020LH0473).
通信作者:姜春茂(hsdrose@126.com)

实际云平台的运行数据表明云任务具有诸多属性[6]。调度的基本单位是Job,每个Job由多个Task组成,任务根据其特定属性可以划分为轻负载和重负载任务,与此同时,Job也可以根据其特定属性划分为轻负载Job与重负载Job等。本文依据用户提交作业的长度以及计算资源的MIPS属性计算出的期望完成时间将作业划分为轻负载Job与重负载Job。然后根据某类作业中轻重负载任务的占比情况,借鉴三支决策[7-11]的基本思想,将作业划分为3类。即,重负载任务多于轻负载任务的作业、轻负载任务多于重负载任务的作业、轻重负载任务接近的作业。根据作业的不同特征分别采取不同的调度策略。实际采用的调度策略是基于Max-min和Min-min调度算法改进而成的。Min-Min算法优先调度小任务,Max-Min算法优先调度大任务,因此,引入三支调度策略增加了对轻重负载任务接近的作业集合的调度策略,实现了整体情况更细粒度的策略选择。

从计算资源(如VM等)的角度而言,可以划分为轻负载的计算资源、中等负载的计算资源、重负载的计算资源。过重负载的计算资源额外增加了系统的开销,如能耗、完成时间加长等。因此,在设计调度策略的时候,本文针对此问题提出的策略是将重负载计算资源中的任务在保证总体执行时间不变或减少的前提下分配给轻负载计算资源,从而保证计算资源的负载维持在一个合理的区间,提高负载均衡。

本文第2节对相关的调度算法进行了回顾;第3节阐述三支优化调度策略模型,即CTSA-3WD;第4节给出了实验设计并分析实验结果;最后总结全文并展望未来。

2　相关工作

作业的调度算法是通过某种算法,将用户提交的作业与数据中心的计算资源联系起来,按照算法设定的规则建立从任务集合到计算资源集合的映射关系。在这个过程中,作业的属性,包括静态属性和动态属性,对于调度过程具有重要的影响,如作业的性质,是批处理还是实时任务;任务的紧急程度、访问频率、存储要求、时间分布、用户满足度、任务优先级、任务调度级别、任务所需资源的更新、数据、任务的状态、任务之间的依赖关系、等待时间的有效组织和融合;用户请求模型等。

对于科学计算这类批处理作业的调度意味着等待某个时间间隔或某个条件成立后,将所有用户提交但没有执行的作业调度到相应的计算资源上。经典的批处理作业调度算法有Sufferage[12],Min-Min[13],Max-Min[14]等。Sufferage算法给某一计算资源分配某一任务时需满足该任务在期望完成时间上的损失是所有任务损失中最大的。Min-Min算法则考量作业的执行时间,计算作业集合中每个任务的任务执行时间,选取每个作业执行时间最短的计算资源与相应作业形成配对,调用其中最短执行时间的资源作业对提交给数据中心,直至作业队列中没有任务。该算法的局限性在于优先选择轻负载任务,并放置在资源最优的机器上,导致系统性能优异的机器负载过重,而性能较差的机器处于空闲状态,从而作业的总体完成时间较长。Max-Min算法的基本想法类似于Min-Min算法,但是其在形成资源任务对后,调用其中最长执行时间的资源任务对提交给数据中心。Max-Min算法是一种针对Min-Min算法的改进,但在优先调度重负载作业时会导致系统资源负载不均衡。

针对Max-min和Min-min调度算法,一些学者给出了相应的优化方案。Zhang等[15]提出基于QoS约束向量选择匹配资源与任务对的Min-Min任务调度算法。Li等[16]提出一种基于Max-Min的弹性云负载均衡调度算法,使用一张任务状态表实时评估虚拟机实时负载和任务期望完成时间。Beghdad等[17]提出一种提高负载均衡的改进Min-Min算法,缩短了任务的最大完成时间。Etminani等[13]提出基于Min-Min和Max-Min的选择调度算法,根据不同的情况选择相应的调度算法。

在对Max-Min和Min-Min算法分析的过程中,一些基于粒度的思考方式给予我们一些有益的思考和尝试。其中,三支决策思想提出以3个粒度进行思考和处理认知问题。在以往研究中,Wu等[18]提出基于三支聚类的负载敏感任务调度算法,从云任务和资源的多重属性进行聚类。Wang等[19]根据任务的容忍时间属性提出将任务队列划分为三支队列进行调度,从而高效地利用计算资源。而本文通过三支决策的思想将作业划分为3个类型,一类作业是其包含的重负载任务远多于轻负载任务,一类作业是其包含的重负载任务接近于轻负载任务,一类作业是其包含的重负载作业远少于轻负载任务,进而设计具有针对性的调度策略。

3　基于三支决策云任务调度算法

3.1　问题定义

云任务的调度过程本质就是在异构或同构的云计算环境中将用户提交的任务调度至相应的计算资源上。为了便于理解和分析,本文考虑的云计算环境基于以下假设条件:

1)云作业执行时采用非抢占式分配,即一个计算资源在任务执行期间内只考虑执行当前作业,不存在作业之间的抢占关系。

2)作业集合中的实例都属于元任务[20],即作业中的任务本身不可再分,任务与任务之间相互独立且一个任务一次只能在一台机器上运行。

3)作业属性中不包含截止时间或任务之间的关联性。

4)作业与计算资源的分配采用批处理模式。

5)云计算环境中参与作业调度的计算资源,其数量和各项属性已知。

6)用户提交给数据中心的元任务数量已知且每个元任务的各项属性已知。

7)数据中心执行各项元任务时,计算资源的等待就绪时间已知。

在上述假设条件皆满足的前提下,云计算环境中的作业调度问题可以形式化描述为一个四元组$\{T,R,ETC,ALLC\}$。为了方便说明,下文中的"任务"表示作业,"元任务"为作业中的任务。具体的参数解释如下。

$T=\{T_i \mid i=1,2,3,\cdots,m\}$表示用户提交给数据中心的$m$个云任务的集合。

$N(T)$表示云任务集合T中元任务的个数。

$type(T_i)$表示云任务T_i的任务类型,其值为0或1。0表示任务T_i为轻负载任务,1表示任务T_i为重负载任务。

$R=\{R_j \mid j=1,2,3,\cdots,n\}$表示数据中心的$n$个计算资源,本文的计算资源设置为物理主机上运行的虚拟机。

ETC(ExpectedTimeto Compute)是一个 $m\times n$ 的云任务期望完成时间矩阵。矩阵的行表示一个任务在各个虚拟机上的期望完成时间,矩阵的列表示一个虚拟机执行各个任务的期望完成时间,$ETC(i,j)$表示任务集合中任务T_i在虚拟机R_j上的期望完成时间,即式(1)。

ALLC 是一个 $n\times m$ 的虚拟机任务分配表。矩阵的每一行表示一台虚拟机上当前分配的任务,$ALLC(am)$表示虚拟机集合中虚拟机R_i的第 j 个任务的任务编号,即式(2)。

$$\boldsymbol{ETC}_{ij}=\begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_m \end{array}\begin{array}{c} \begin{array}{cccc} R_1 & R_2 & \cdots & R_n \end{array} \\ \begin{pmatrix} ect_{11} & ect_{12} & \cdots & ect_{1n} \\ ect_{21} & ect_{22} & \cdots & ect_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ ect_{m1} & ect_{m2} & \cdots & ect_{mn} \end{pmatrix} \end{array} \tag{1}$$

$$\boldsymbol{ALLC}_{ij}=\begin{array}{c} \\ R_1 \\ R_2 \\ \vdots \\ R_n \end{array}\begin{array}{c} \begin{array}{cccc} T_1 & T_2 & \cdots & T_m \end{array} \\ \begin{pmatrix} ALLC_{11} & ALLC_{12} & \cdots & ALLC_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ ALLC_{21} & ALLC_{22} & \cdots & ALLC_{2m} \\ ALLC_{n1} & ALLC_{n2} & \cdots & ALLC_{mn} \end{pmatrix} \end{array} \tag{2}$$

3.2 算法思想

在所有云任务和计算资源的属性中,任务长度和虚拟机的处理能力是影响云计算环境中任务执行时间的主要因素,其直接影响任务的分配和调度,是任务调度算法的主要考量。依据任务长度,可以将任务划分为长任务和短任务。依据计算资源的处理能力,可以将虚拟机划分为高性能虚拟机和低性能虚拟机。本文将这两个至关重要的属性结合,提出重负载任务和轻负载任务,并将整个系统中最后执行完任务的虚拟机定义为关键虚拟机,因其直接决定任务集合的任务总体完成时间。

用户提交到数据中心的任务可以依据任务负载的轻重划分为 3 种情况,对应三支决策的 3 个决策域,分别是轻负载任务明显多于重负载任务、重负载任务明显多于轻负载任务和轻重负载任务数量相接近。现有的文献大多将这 3 种情况视为 1 种,部分学者考虑了前两种情况,缺乏对于边界域的判断,导致在轻重负载任务数量相接近的情况下陷入算法的最差情况,进而使得计算资源的负载均衡性下降且总体完成时间变长。本文结合三支决策的思想,提出基于任务负载情况的三支表现形式。

令$T_p=\{T_i\mid type(T_i)=0,0\leqslant i\leqslant N(T)\}$表示云任务集合 T 中的轻负载任务集合。同理,令$T_n=\{T_i\mid type(T_i)=1,0\leqslant i\leqslant N(T)\}$表示云任务集合 T 中的重负载任务集合。

依据云任务集合 T 中轻负载和重负载的任务情况,将 T 可能属于的 3 种情况划分成 3 个区域。

$$\begin{aligned} T_P&=\left\{T\in RF\mid 0\leqslant\frac{N(T_p)}{N(T)}\leqslant\alpha\right\}\\ T_B&=\left\{T\in RF\mid \alpha<\frac{N(T_p)}{N(T)}<\beta\right\}\\ T_N&=\left\{T\in RF\mid \beta\leqslant\frac{N(T_p)}{N(T)}\leqslant 1\right\} \end{aligned} \tag{3}$$

式(3)相应地也可以用$\frac{N(T_n)}{N(T)}$进行表示。其中 α 和 β 是区域划分的阈值,且 $0\leqslant\alpha<0.5<\beta\leqslant 1$。

由此,一个三支区域划分为 $RF=T_P\cup T_B\cup T_N$,且

$$\begin{aligned} &T_P\cap T_B=\varnothing\\ &T_B\cap T_N=\varnothing\\ &T_P\cap T_N=\varnothing \end{aligned} \tag{4}$$

其中,T_P,T_B和T_N分别对应上文中的 3 种情况,即轻负载任务较多重负载任务较少、轻负载任务和重负载任务相接近和轻负载任务较少重负载任务较多。三者的并集就是所有任务负载情况。RF 表示所有云任务可能情况的全集。针对每个域,根据其特点选择最适合的任务调度算法。图 1 给出了本文提出的 CTSA-3WD 云任务调度算法框架,其借鉴了三支决策的基本思想。

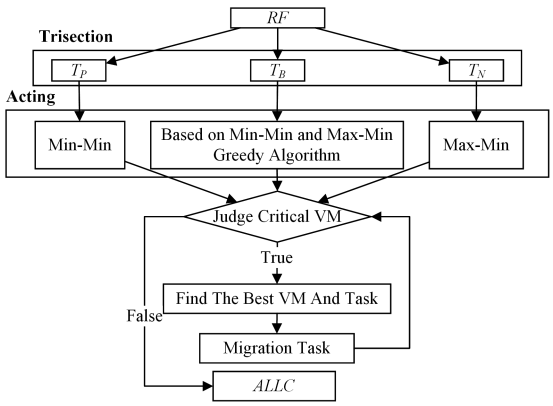


图 1 CTSA-3WD 云任务调度算法框架图

Fig. 1 Scheduling framework of CTSA-3WD

Etminani 等[13]指出,当轻负载任务多于重负载任务时 Min-Min 算法的执行效果不如 Max-Min 算法,但在重负载任务多于轻负载任务时,Min-Min 算法的执行效果优于 Max-Min 算法。据此,经过大量实验探究,Min-Min 算法因其将轻负载任务分配至高性能虚拟机上并优先调度轻负载任务的特点,可以在重负载任务明显多于轻负载任务的情况下,取得较好的性能表现。但是在轻负载任务明显多于重负载任务的情况下,算法的效果则明显差于将重负载任务分配至高性能虚拟机上并优先调度重负载任务的 Max-Min 算法。而针对边界域的调度问题,单纯地使用 Max-Min 算法或者 Min-Min 算法都会因优先调度轻负载任务或是重负载任务使得情况相互转化进而无法避免陷入算法的最差情况。本文采用贪心的思想,同时调度当前任务队列中最小任务完成时间的最大和最小任务,以避免轻重负载任务在调度过程中累计。同时采用二阶段的思想,在算法第二阶段通过关键虚拟机任务重调度,改善第一阶段可能出现的负载不均衡情况,进一步提高系统的负载均衡程度,缩短整个任务集合的执行时间。

3.3 算法的实现

本文提出基于三支决策的云任务调度算法(CTSA-3WD),该算法分为两个阶段。第一阶段进行预分配,考虑到用户提交的云任务很大程度上可以理解为随机变量,通过计算任务集合中所有任务的执行时间,获得其整体的均方差值来判断整个任务集合轻重负载的离散程度并通过其离散程度判断任务集合属于 3 种决策情况中的哪一种,调用相应情况下的算法。第二阶段则是进一步检查整个任务调度中的关键资源是否可以优化负载情况,降低关键资源的任务负载,使得整个任务调度过程中的总体任务完成时间得以缩短,实现负载均衡。

3.3.1　第一阶段：基于均方差的三支划分调度

首先，对于当前任务集合中的所有任务计算出每个任务的期望完成时间。任务的期望完成时间由该任务的等待时间与预期执行时间组成，可由式(5)得出。然后，将任务集合中每个任务的最小期望完成时间进行升序排列，并记录相应任务与虚拟机对形成任务队列 TQ。

$$ect_{ij}=wt_j+ext_{ij} \tag{5}$$

其中，wt_j 表示虚拟机资源 R_j 执行完之前分配的任务所需时间之和，即计算资源的就绪时间，可由 $wt_j=\sum_{k=1}^{TN_j}ext_{kj}$ 计算，TN_j 表示当前该资源已经分配的任务数量。相较于任务的执行时间，任务之间的切换时间在此忽略不计。ext_{ij} 表示任务 T_i 在虚拟机 R_j 上的预期执行时间，如式(6)所示。

$$ext_{ij}=\frac{Length_i}{Mips_j} \tag{6}$$

其中，$Length_i$ 表示任务 T_i 的任务长度，$Mips_j$ 表示虚拟机 R_j 的 $Mips$ 值。

为了更好地实现负载均衡，在任务与虚拟机匹配的过程中，如果出现多个虚拟机对于同一任务的期望完成时间相同，则选取其中最小资源利用率的虚拟机作为该任务匹配的计算资源，并记录进队列 TQ 中。虚拟机资源利用率的计算公式如下：

$$RU_j=\frac{\sum_{k=1}^{TN_j}te_k-ts_k}{T} \tag{7}$$

式(7)表示当前分配给虚拟机 R_j 的所有任务执行时间之和在任务集合中已分配的任务运行时间中所占的比例。其中，te_k 表示任务 T_k 在虚拟机 R_j 上的运行完成时间，ts_k 表示任务 T_k 在虚拟机 R_j 上的运行开始时间。T 表示当前所有已经分配至相应虚拟机的任务运行完毕花费的时间，其形式化如下：

$$T=\max(te_m)-\min(ts_m) \tag{8}$$

其中，$\max(te_m)$ 表示当前所有已分配虚拟机的任务运行完成时间的最大值，$\min(ts_m)$ 表示当前所有已分配虚拟机的任务运行完成时间的最小值。

为了更好地区分任务负载的轻重程度，引入任务集合中每个任务的平均最小期望完成时间和最小期望完成时间的均方差。平均最小期望完成时间和最小期望完成时间的均方差形式化如下：

$$aveect=\frac{\sum_{i=1}^{m}ect_{ij}}{m} \tag{9}$$

$$MSD=\sqrt{\frac{\sum_{i=1}^{m}(ect_{ij}-aveect)^2}{m}}$$

$$\Rightarrow MSD=\sqrt{E(ect_{ij}^2)-E(ect_{ij})^2} \tag{10}$$

其中，$E(x)$ 表示计算 x 的均值。

通过式(10)计算出任务集合的最小期望完成时间的均方差，寻找在任务队列 TQ 中划分轻重负载任务的分割点。分割点需满足式(11)的条件。

$$\begin{aligned}&ect_{ij},ect_{i+1\,j}\in TQ\\&sub_{ect}=|ect_{i+1\,j}-ect_{ij}|\\&sub_{ect}\geqslant MSD\end{aligned} \tag{11}$$

其中，sub_{ect} 表示任务队列 TQ 中两个相邻任务期望完成时间的绝对差。分割点为大于或等于 MSD 的最小 sub_{ect} 对应的任务 T_i 在 TQ 中的索引位置。

根据位置的不同，匹配 3 种相适应的任务调度算法。下面给出 3 种可能的情况及相应的调度算法。

1)情况 1：该分割点在整个任务集合前部至 2/5 时，即任务集合中重负载任务明显多于轻负载任务。此时，相较于 Max-Min 算法，Min-Min 算法能取得更好的系统负载均衡程度，缩短任务总体完成时间。因此，这种情况下采用 Min-Min 算法进行任务预调度。具体步骤为找出具有最小期望完成时间的最小任务，即式(5)中的 min。通过式(7)找出资源利用率最低的虚拟机执行任务。更新式(1)、式(2)中的 ***ETC*** 和 ***ALLC*** 矩阵，重复执行直至任务集合中没有未分配任务。

2)情况 2：该分割点在整个任务集合 3/5 至末尾时，即任务集合中轻负载任务明显多于重负载任务。此时，相较于 Min-Min 算法，Max-Min 算法能取得更好的系统负载均衡程度，缩短任务总体完成时间。因此，这种情况下采用 Max-Min 算法进行任务预调度。具体步骤为找出具有最小期望完成时间的最大任务，即式(5)中的 max。通过式(7)找出资源利用率最低的虚拟机执行任务。更新式(1)、式(2)中的 ***ETC*** 和 ***ALLC*** 矩阵，重复执行直至任务集合中没有未分配任务。

3)情况 3：该分割点在整个任务集合中部位置或不存在，即任务集合中轻负载任务和重负载任务数量相近或期望完成时间相近。实际云平台上的运行数据显示，在这种情况下 Min-Min 算法和 Max-Min 算法因任务之间复杂度相近或波动较小导致效果并不稳定，使用其中任意一种算法都可能陷入算法的最差情况，无法达到相对较好的负载均衡性。对此，采用时间贪心的云任务调度算法，利用最大和最小任务组合的方式调度任务，能够有效提高负载均衡水平。具体步骤为找到未分配任务中最小期望完成时间的最小和最大的任务，即式(5)中的 min 和 max。通过式(7)找出资源利用率最低的虚拟机并运行上一步的两个任务。更新式(1)、式(2)中的 ***ETC*** 和 ***ALLC*** 矩阵，重复执行直至任务集合中没有未分配任务。

3.3.2　第二阶段：关键虚拟机重调度

根据不同的 3 种情况进行预调度后，预调度的分配信息记录在 ***ALLC*** 矩阵中。依据 ***ALLC*** 矩阵中的分配信息，计算每台虚拟机运行任务花费的总时间，标记其中的最大值与最小值对应的虚拟机。若上述过程中，存在不止一个虚拟机的运行总时间等于最小值或最大值，则选取其中资源利用率最低或最高的虚拟机，以提高系统整体的负载均衡程度。计算最大值对应虚拟机上所分配的任务在最小值对应的虚拟机上运行所花费的时间，寻找最接近两台虚拟机之间运行时间差值的任务，并将其部署至最小值对应的虚拟机上，约束条件形式化如下：

$$\begin{aligned}&ect_{T_p}+\min(ect_{ij})\leqslant\max(ect_{ij})\\&ect_{T_q}\geqslant ect_{T_p}\\&T_q\in T_p,T_p\in\max(ect_{ij})\end{aligned} \tag{12}$$

其中，任务 T_q 属于任务集合 T_p，任务集合 T_p 属于运行总时间最大值的虚拟机任务列表，任务 T_p 调度至最小值的虚拟机后，整体任务运行的总时间小于原本调度的整体任务运行的总时间。任务 T_q 的 ect_{T_q} 是任务集合 T_p 中的最大值。

基于上述表达，本文给出基于粒度思想的云任务调度算法(CTSA-3WD)的具体实现，如算法 1 所示。

算法 1　CTSA-3WD 云任务调度算法

输入：云任务 $T_i(i=1,2,\cdots,m)$，虚拟机资源列表 $R_j(j=1,2,\cdots,n)$

输出：云任务集合T_i(i=1,2,…,m)的任务调度方案 **ALLC**

BEGING

1. 创建任务资源对的 **ETC** 矩阵
2. FOR $T_i \in TQ$ DO
3. TQ⟨task,vm⟩←T_i,{R_j|min(ect_{ij})&&min (RU_j)}
4. 按照式(10)计算云任务队列 TQ 的 MSD 值
5. 按照式(11)找出分割点
6. IF 分割点不存在或分割点在 TQ 的中部位置 THEN
7. WHILE 剩余任务数量≠0 DO
8. 获得未分配任务的 min(ect_{ij})的 min 和 max，部署给相应 min(RU_j)的虚拟机资源R_j
9. IF 分割点在整个任务队列的前部 THEN
10. WHILE 剩余任务数量≠0 DO
11. 获得未分配任务的 min(ect_{ij})的 min，部署给相应 min(RU_j)的虚拟机资源R_j
12. IF 分割点在整个任务队列的后部 THEN
13. WHILE 剩余任务数量≠0 DO
14. 获得未分配任务的 min(ect_{ij})的 max，部署给相应 min(RU_j)的虚拟机资源R_j
15. WHILE max(ect_{ij})−min(ect_{ij})>0 DO
16. FOREACH $\exists$max(T_i)$\in R_j$ of min(ect_{ij})，WHERE ect(T_i)≤max(ect_{ij})−min(ect_{ij})
17. 任务 max(T_i)从 max(ect_{ij})虚拟机迁移至 min(ect_{ij})虚拟机
18. RETURN **ALLC**

END

4 实验及分析

本文通过基于 Etminani 等提出的测试环境进行实验及数据设置，观察性能方面的表现，来验证提出的 CTSA-3WD 云任务调度算法相较于 Min-Min 算法、Max-Min 算法和选择调度算法的有效性与性能优势。

4.1 实验环境

实验的机器采用笔记本电脑一台，使用的操作系统为 ArchLinux，内核版本号为 4.19.133-1，处理器为 Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70GHz，8.00GB 内存。本文的云计算环境采用墨尔本大学研发的开源仿真模拟工具 CloudSim4.0 进行实验，以便于在一台主机上对大规模的云计算环境集群机器进行仿真。

4.2 评估指标

为了评估本文提出的云计算任务调度算法综合性能，实验采用任务总体完成时间、负载均衡水平这两个评估指标。所选取的评估指标的定义及其选取理由如下。

1)任务总体完成时间(*Makespan*)

该指标描述用户提交的所有云任务在数据中心计算集群中执行所花费的总时间，即最后一个执行完毕任务的完成时间。任务调度算法产生的 *MakeSpan* 越小则该调度算法的执行效率越高。设定 $M(R_j)$为虚拟机R_j执行所分配的所有任务总时间，则：

$$Makespan=\max(M(R_j)) \tag{13}$$

$$M(R_j)=\sum_{i=1}^{l} ETC(i,j)\times F(i,j) \tag{14}$$

$$F(i,j)=\begin{cases}1, & \text{任务}T_i\text{分配给虚拟机}R_j\\ 0, & \text{其他}\end{cases},1\leqslant j\leqslant n \tag{15}$$

其中，n 为虚拟机数量，l 为相应虚拟机分配到的任务数量。

2)负载均衡水平(LoadBalancing Level)

该指标定义为系统资源利用率 RU_j 的均方差 sd 相较于平均资源利用率 U 的相对偏差 β，反映任务调度算法产生的分配情况对于系统负载均衡的影响，较为直观地体现整个数据中心的负载情况。单个虚拟机的资源利用率可以利用式(7)计算。通过式(16)计算数据中心的平均资源利用率。该值的取值范围为 0～1，该值越高表明系统的负载均衡程度越高。完整的计算公式如下：

$$U=\frac{1}{n}\sum_{j=1}^{n} RU_j \tag{16}$$

$$sd=\sqrt{(RU_j-U)^2} \tag{17}$$

$$\gamma=\left(1-\frac{sd}{U}\right)\times 100\% \tag{18}$$

4.3 实验参数与仿真步骤

4.3.1 实验参数

实验中所使用的相关参数初始化如表 1 所列，未展示的参数采用 CloudSim 中的默认值。

表 1 实验初始化参数

Table 1 Experimental initialization parameters

种类	参数	值
数据中心	数据中心数量	1
	物理主机数	4
	每台物理主机的内核数	2
	每台物理主机的内核性能/MIPS	5000
	内存与存储容量	4096&100000
	带宽	1000
虚拟机	虚拟机总数	8
	每台虚拟机的内核数	1
	每台虚拟机的性能	100～500 MIPS
	内存与存储容量	2048&100000
云任务	共享方式	空间共享
	每个云任务的内核数	1
	每个云任务的任务长度	1000～30000

本文实验建立一个数据中心及其代理，其中包含 4 台物理主机，每台物理主机拥有 2 个内核，每个内核的运算性能为 5000 MIPS。创建 8 台虚拟机，每台虚拟机单核，每个内核的运算性能在 100～500 之间随机生成。云任务在虚拟机上的执行策略采用空间共享策略，采用均匀分布生成任务长度在 1000～30000 之间的任务，轻负载任务和重负载任务的数量在不同情况下随机获取。每个任务使用的核心数为 1。云任务数量分别选取 250,500,750,1000 个进行实验。

4.3.2 仿真步骤

实验在 CloudSim4.0 中执行，以下是实验的主要仿真步骤。

步骤 1 设置数据中心中主机与虚拟机的相应参数，使用随机的方法在指定范围内生成虚拟机与任务的各项属性。

步骤 2 初始化 CloudSim，利用步骤 1 中的数据创建任务列表、虚拟机列表和主机列表并统一提交至数据中心代理。

步骤 3 重写 CloudSim 中 DatacenterBroker 类的任务分配策略，实现本文提出的任务调度算法 bingCloudletsToVms-ByThreeWay()。在该方法内部直接按照算法生成的 ***ALLC*** 矩阵调度。

步骤 4 实现 Min-Min，Max-Min，选择算法，按照实验参数配置实验环境。

步骤 5 启动 CloudSim 模拟并输出实验结果，计算相应

评估指标并比较实验结果。

4.4　实验结果分析

为了更直观地展现提出的算法的性能，实验基于以下 3 个云任务集合的分布情况进行。

4.4.1　轻负载任务多于重负载任务

在此种情况下，云任务集合中轻负载任务较多，重负载任务较少。实验结果如图 2 所示。

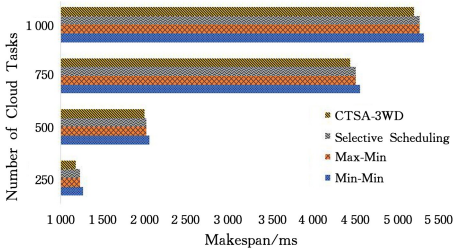


图 2　轻负载任务多于重负载任务时任务总体完成时间对比

Fig. 2　Overall task completion time comparison when light-load tasks are more than heavy-load tasks

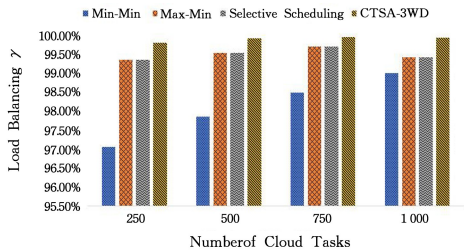


图 3　轻负载任务多于重负载任务时负载均衡水平对比

Fig. 3　Load balance level comparison when light-load tasks are more than heavy-load tasks

根据图 2 和图 3 中可以得出结论：在任务总体完成时间和负载均衡水平方面，本文提出的算法最好，Max-Min 算法和选择算法次之，Min-Min 算法最差。分析其原因在于云任务集合中轻负载任务较多且重负载任务较少时，Max-Min 算法偏重于将重负载任务分配给高性能虚拟机上，更适合此种情况下的任务调度。选择算法在此种情况下选择执行 Max-Min 算法的调度方法，因此性能同 Max-Min 算法的效果。而 Min-Min 算法使得高性能虚拟机一直处于运行状态，低性能虚拟机处于空闲状态，使得任务总体完成时间和负载均衡水平略差。本文的任务调度算法在此基础之上，进行二阶段重调度，使得虚拟机中关键虚拟机的任务总体完成时间得以缩短并改善了系统的负载水平。

4.4.2　轻负载任务和重负载任务数量接近

在此种情况下，云任务集合中轻负载任务和重负载任务数量相接近。实验结果如图 4 和图 5 所示。

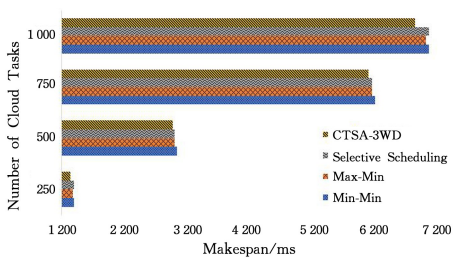


图 4　轻重负载任务数量相近时任务总体完成时间对比

Fig. 4　Overall task completion time comparison when number of light-load tasks and heavy-load tasks are similar

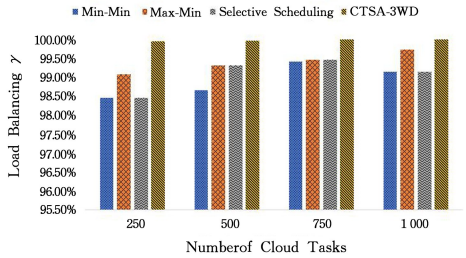


图 5　轻重负载任务数量相近时负载均衡水平对比

Fig. 5　Load balance level comparison when number of light-load tasks and weight-load tasks are similar

根据图 4 和图 5 中可以得出结论：在任务总体完成时间和负载均衡水平方面，本文提出的算法最好，Max-Min 算法和选择算法次之，Min-Min 算法最差。分析其原因在于云任务集合中轻负载任务和重负载任务数量相接近时，Max-Min 算法大多优于 Min-Min 算法，但偶尔 Min-Min 算法优于 Max-Min 算法。任意选择其中一种算法都无法保证其选择的是最优解。选择算法根据其计算指标在 Min-Min 和 Max-Min 算法中切换，效果不稳定，存在 Max-Min 算法优于 Min-Min 算法时却调用 Min-Min 算法，或 Min-Min 算法优于 Max-Min 算法时却调用 Max-Min 算法的情况。本文提出的算法采用贪心的思想，一个调度周期内分别调度轻负载任务和重负载任务给高性能虚拟机，这样改善了整体的负载均衡。再结合二阶段重调度，检查关键虚拟机是否可以进行再次优化，最终在提高负载均衡水平的同时，缩短了任务总体完成时间，且优化效果相较于 Max-Min 算法和 Min-Min 算法较为明显。

4.4.3　轻负载任务少于重负载任务

在此种情况下，云任务集合中重负载任务较多，轻负载任务较少。实验结果如图 6 和图 7 所示。

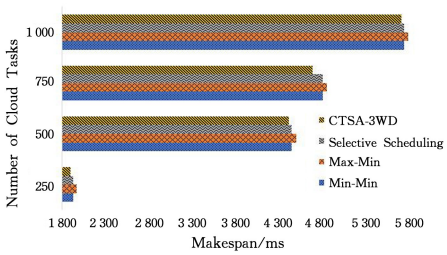


图 6　轻负载任务少于重负载任务时任务总体完成时间对比

Fig. 6　Overall task completion time comparison when light-load tasks are less than heavy-load tasks

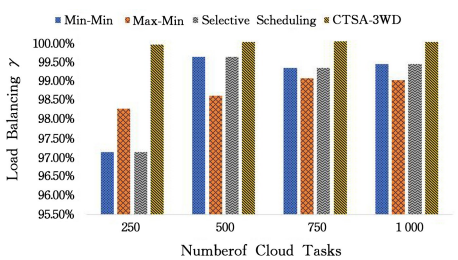


图 7　轻负载任务少于重负载任务时负载均衡水平对比

Fig. 7　Load balance level comparison when light-load tasks are less than heavy-load tasks

根据图 6 和图 7 中可以得出结论：在任务总体完成时间和负载均衡水平方面，本文提出的算法最好，Min-Min 算法和

选择算法次之，Max-Min 算法最差。分析其原因在于云任务集合中重负载任务较多且轻负载任务较少时，Min-Min 算法优先将轻负载任务调度到高性能虚拟机上，其效果优于 Max-Min 算法。选择算法在此种情况下选择执行 Min-Min 算法的调度方法，因此性能同 Min-Min 算法的效果。Max-Min 算法将重负载任务调度到高性能虚拟机上导致负载均衡稍差。本文提出的算法在处理这种情况时采用 Min-Min 算法进行预调度，再结合二阶段思想，尽量缩短关键虚拟机的任务完成时间，最终提高系统的负载均衡水平。

结束语 为了缩短任务总体的完成时间，提高系统的负载均衡水平，提出了基于粒度思想云任务调度算法。本算法分为两个阶段。第一阶段，依据任务集合与虚拟机计算出的均方差判断轻重负载任务的分布情况，结合三支划分的任务集合种类，选择最为合适的调度算法。在保证预调度算法优越性的同时，算法的第二阶段检查影响任务总体完成时间的虚拟机是否可以继续进行优化，降低预调度阶段算法造成的缺陷。实验结果表明，本文提出的算法能够有效缩短任务总体的完成时间，并提高系统的负载均衡水平。未来的研究将会从更细粒度的角度进行思考，加入云计算环境中的其他因素，比如服务质量、任务之间的关联程度等，并在此基础之上进一步优化算法。

参 考 文 献

[1] PANDA S K, JANA P K. An efficient task scheduling algorithm for heterogeneous multi-cloud environment[J]. The Journal of Supercomputing, 2015, 71(4): 1505-1533.

[2] GAVVALA S K, JATOTH C, GANGADHARAN G R, et al. QoS-aware cloud service composition using eagle strategy[J]. Future Generation Computer Systems. 2019, 90: 273-290.

[3] KAUR P, MEHTA S. Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm[J]. Journal of Parallel and Distributed Computing, 2017, 101: 41-50.

[4] CHEN X, CHENG L, LIU C, et al. A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems[J]. IEEE Systems Journal, 2020, 14(3): 3117-3128.

[5] GE D, DING Z, JI H. A task scheduling strategy based on weighted round robin for distributed crawler[J]. Concurrency & Computation Practice & Experience, 2016, 28(11): 3202-3212.

[6] VERMA A, PEDROSA L, KORUPOLU M, et al. Large-scale cluster management at Google with Borg[C]// Proceedings of the Tenth European Conference on Computer Systems. 2015: 1-17.

[7] YAO Y. Three-way decision and granular computing[J]. International Journal of Approximate Reasoning, 2018, 103: 107-123.

[8] YAO Y. Set-theoretic models of three-way decision[J]. Granular Computing, 2021, 6(1): 133-148.

[9] JIANG C, GUO D, DUAN Y, et al. Strategy selection under entropy measures in movement-based three-way decision[J]. International Journal of Approximate Reasoning, 2020, 119: 280-291.

[10] YAO Y. Tri-level thinking: models of three-way decision[J]. International Journal of Machine Learning & Cybernetics, 2019: 1-13.

[11] GUO D D, JIANG C M. Multi-stage Regional Transformation Strategy in Move-based Three-way Decisions Model [J] Compute Science, 2019, 46(10): 279-285.

[12] BRAUN T D, SIEGEL H J, BECK N, et al. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems[J]. Journal of Parallel & Distributed Computing, 2001, 61(6): 810-837.

[13] ETMINANI K, NAGHIBZADEH M. A Min-Min Max-Min selective algorithm for grid task scheduling[C]// 2007 3rd IEEE/IFIP International Conference in Central Asia on Internet. IEEE, 2007: 1-7.

[14] REHMAN A, JAVED K, BABRI H A, et al. Selection of the most relevant terms based on a max-min ratio metric for text classification[J]. Expert Systems with Applications, 2018, 114: 78-96.

[15] ZHANG Y, XU B. Task Scheduling Algorithm based-on QoS Constrains in Cloud Computing[J]. International Journal of Grid and Distributed Computing, 2015, 8(6): 269-280..

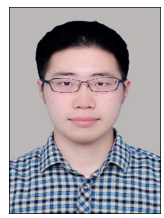
[16] LI Z, SHEN H, MILES C. PageRankVM: A PageRank Based Algorithm with Anti-Collocation Constraints for Virtual Machine Placement in Cloud Datacenters[C]// 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2018.

[17] BEY K B, BENHAMMADI F, BENAISSA R. Balancing heuristic for independent task scheduling in cloud compu[C]// International Symposium on Programming & Systems. IEEE, 2015.

[18] WU J W, JIANG C M. Load-aware score scheduling of three-way clustering for cloud task[J]. CAAI transactions on intelligentsystems, 2019, 14(2): 316-322.

[19] JIANG C M, WANG K X. Real-time Cloud Tasks Schedule Algorithm for Saving Energy Based on Tri-queue System, 2019, 51(2): 66-71.

[20] INGBO J, JU W, DALI W, et al. The research on meta-job scheduling heuristics in heterogeneous environments[J]. Journal of Intelligent & Fuzzy Systems, 2018, 34(2): 1141-1151.



WANG Zheng, born in 1996, postgraduate. His main research interests include cloud computing and granularity computing.

JIANG Chun-mao, born in 1972, Ph.D, professor, is a member of China Computer Federation. His main research interests include cloud computing and big data, intelligent data decision making.