文章编号：1674-8085(2016)01-0062-07

基于不同分配策略的云计算任务调度性能比较与分析

孙凌宇，*冷 明

（井冈山大学流域生态与地理环境监测国家测绘地理信息局重点实验室, 江西，吉安 343009）

摘 要：形式化描述了云计算环境下的负载均衡任务调度问题，借助动态规划方法形式化推导了最早完成时间的启发式优先分配策略，给出了基于先易后难优先分配策略、先难后易优先分配策略的启发式云计算任务调度算法。阐述了基于顺序调度策略、先易后难优先分配策略、先难后易优先分配策略等启发式任务调度算法和基于禁忌搜索策略、元胞演化策略等智能任务调度算法。针对不同分配策略的云计算任务调度进行性能比较与分析，提出了完成时间可改进百分比和资源负载平衡因子的调度性能评价指标，实验数据对比充分表明：与启发式调度算法相比，智能调度算法能减少任务执行时间，优化资源负载均衡性能。

关键词：云计算；任务调度；分配策略；负载均衡；性能分析

中图分类号：TP391 **文献标识码**：A DOI:10.3969/j.issn.1674-8085.2016.01.013

THE COMPARISON AND ANALYSIS OF TASK SCHEDULING PERFORMANCE IN CLOUD COMPUTING BASED ON DIFFERENT ALLOCATION STRATEGIES

SUN Ling-yu，*LENG Ming

（Key laboratory of watershed ecology and geographical environment monitoring of NASG, Jinggangshan University, Ji'an,Jiangxi 343009, China）

Abstract: The formal description of load balancing task scheduling problem in cloud computing is presented. We make its formal derivation based on dynamic programming method and built the heuristic scheduling strategy of the earliest finish time (EFT) for task scheduling. Furthermore, we propose the priority-to-easy scheduling strategy and priority-to-difficult scheduling strategy for task scheduling in cloud computing based on the EFT strategy. We present the heuristic scheduling algorithm based on the sequential scheduling strategy, the priority-to-easy scheduling strategy and priority-to- difficult scheduling strategy. We also present the intelligence task scheduling algorithm based on the tabu search scheduling strategy and the cellular automata scheduling strategy. Then, we propose two evaluation factors of scheduling performance analysis, which are the improvement percent of the latest time and the load balancing factor. Finally, we carry out the comparative experiments of scheduling performance under the CloudSim simulation platform of cloud computing based on five allocation strategies, which are the sequential scheduling strategy, the priority-to-easy scheduling strategy, priority-to-difficult scheduling strategy, the tabu search scheduling strategy and the cellular automata scheduling strategy. The experiment and analysis show that intelligence scheduling strategy has better performance in comparison with

收稿日期：2015-10-28；修改日期：2015-12-03
基金项目：国家自然科学基金项目（61363014,61163062）；流域生态与地理环境监测国家测绘地理信息局重点实验室招标课题（WE2015012）；江西省青年科学家培养对象计划（20153BCB23003）；江西省科技厅支撑项目（20132BBE50048）；江西省自然科学基金项目(20132BAB201035)；江西省教育厅科技计划项目(GJJ150779)
作者简介：孙凌宇(1976-)，女，江西永丰人，教授，硕士，主要从事云计算、任务调度等研究(E-mail: Lzylmsly@gmail.com)；
*冷 明(1975-)，男，江西高安人，教授，博士，主要从事算法分析与设计、组合分析与优化等研究(E-mail:Lengming@idsu.edm.cn).

the heuristic scheduling strategy in terms of the decreasing the task completing time and the improvement of resource load balancing.

Key words: cloud computing; task scheduling; allocation strategy; load balancing; performance analysis

0 引言

云计算的核心思想是利用分布在各地闲散异构的大规模廉价物理资源，整合形成巨大的虚拟资源池，再通过网络将用户提交的计算和存储任务调度到不同的虚拟机上，使得人们能够以极低的成本投入来提升计算能力及存储容量，获得较高的服务质量[1]。

任务调度作为云计算平台的重要组成部分，是将用户提交的任务进行合理高效地调度和分配，其实质就是将 n 个相互独立的任务分配到 m 个闲散异构的物理资源上，使得总任务完成时间最小并且可用资源得到充分利用[2]。任务调度的效率直接影响到云计算平台的整体性能和服务质量。目前，云计算的任务调度机制还未形成统一的标准和规范[3]，但由于该问题的重要性，国内外研究者提出了大量的云计算任务调度算法来计算任务调度的近似最优解，既有传统网格计算中的顺序调度[2]，Min-Min[4]、Max-Min[5]、动态规划[6]等启发式调度算法，也有基于遗传算法[7]、粒子群算法[8]、蚁群算法[3]、免疫算法[9]、差分进化算法[10]和禁忌搜索算法[11]等智能调度算法。

本文提出了云计算任务分配方案的完成时间可改进百分比和资源负载平衡因子的性能评价指标，进行了基于不同分配策略的云计算任务调度性能分析。首先，给出了云计算环境下的负载均衡任务调度问题的形式化描述；接着，通过动态规划方法的形式化推导得到最早完成时间的启发式优先分配策略，并给出了基于先易后难优先分配策略、先难后易优先分配策略的云计算任务调度算法；进而，阐述了基于顺序调度策略、先易后难优先分配策略、先难后易优先分配策略等启发式任务调度算法和基于禁忌搜索策略、元胞演化策略等智能任务调度算法，提出了完成时间可改进百分比和资源负载平衡因子的调度性能评价指标；最后，在 CloudSim 仿真实验平台下进行了基于以上各调度策略的云计算任务调度算法的性能对比实验。实验数据对比充分表明，与启发式调度算法相比，智能调度算法能减少任务执行时间，优化资源负载均衡性能。

1 云计算负载均衡任务调度问题的形式化描述及优先分配策略推导

1.1 形式化描述

定义 1：假设云计算环境下，用户提交作业分解成 n 个任务的集合，且任务之间相互独立，其调度不需要考虑任务间的数据关联与优先约束关系，定义任务集合 $T=\{t_i\}$，其中 t_i 为分解成的第 i 个任务 $(i=1,2,\cdots,n)$，n 为分解后的任务数量，且 t_i 的总指令长度为 MI_i。

定义 2：假设云计算环境下，有 m 个虚拟资源的集合参加任务调度，且虚拟资源为云计算集群中的虚拟机。定义虚拟机集合 $VM=\{vm_j\}$，其中 vm_j 为第 j 个虚拟机资源 $(j=1,2,\cdots,m)$，m 为虚拟机数量，且 vm_j 的指令执行速度(每秒执行指令条数)为 $MIPS_j$。

定义 3：假设分解后的任务数量 n 不小于虚拟机资源数量 m $(n\geq m)$，每个任务只能分配给一个虚拟机执行，且在某一时间段一个虚拟机只能执行一个任务。定义 n 个不同的任务调度到 m 个不同的虚拟机上的预期执行时间 C 是一个 $n\times m$ 的矩阵，其中 c_{ij} 表示第 i 个任务 t_i 在第 j 个虚拟机 vm_j 上执行的时间，且 $c_{ij}=MI_i/MIPS_j$，即预期执行时间 c_{ij} 等于任务 t_i 的总指令长度 MI_i 除以虚拟机 vm_j 的每秒执行指令条数 $MIPS_j$。

$$C=\begin{pmatrix} c_{11} & \cdots & c_{1m} \\ \cdots & c_{ij} & \cdots \\ c_{n1} & \cdots & c_{nm} \end{pmatrix}$$

定义 4：定义 n 个不同任务 $T=\{t_i\}$ 调度到 m 个不同虚拟机 $VM=\{vm_j\}$ 上所有可能的任务分配方案集为 $\aleph$。定义 X 代表任务分配方案集 $\aleph$ 中的一种分配方案，即一个 $n\times m$ 的矩阵。其中，x_{ij} 表示任务 t_i 与虚拟机 vm_j 的分配关系，且 $x_{ij}\in\{0,1\}$，$\sum_{j=1}^{m}x_{ij}=1$，$i\in\{1,2,\cdots,n\}$，$j\in\{1,2,\cdots,m\}$。即如果任务 t_i 分配在虚拟机 vm_j 上执行，则 $x_{ij}=1$，否则 $x_{ij}=0$。

$$X=\begin{pmatrix} x_1 & \cdots & x_{1m} \\ \cdots & x_{ij} & \cdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$$

定义 5：对于某任务分配方案 X，定义虚拟机的当前负载 $vt_{(k-1)j}$ 为当前状态下(前 k-1 个任务分配完毕的状态)，分配给第 j 个虚拟机 vm_j 的所有任务所需执行时间，即 $vt_{(k-1)j}=\sum_{i=1}^{k-1}\left(x_{ij}\cdot c_{ij}\right)$。定义第 k 个任务 t_k 分配在第 j 个虚拟机 vm_j 上的时间跨度 $makespan_{kj}$ 为任务 t_k 在 vm_j 上执行的最早完成时间，即 $makespan_{kj}=vt_{(k-1)j}+c_{kj}$。

定义 6：对于某任务分配方案 X，定义虚拟机的负载 VT_j 为分配给第 j 个虚拟机 vm_j 所有任务的预期完成时间，即 $VT_j=\sum_{i=1}^{n}\left(x_{ij}\times c_{ij}\right)$。

定义 7：定义 n 个不同任务调度到 m 个不同虚拟机上的平均负载，等于 n 个任务的总指令长度除以 m 个虚拟机指令执行速度累加和，即总任务最优完成时间 $\overline{VT}=\sum_{i=1}^{n}MI_i\Big/\sum_{j=1}^{m}MIPS_j$。

定义 8：对于某任务分配方案 X，定义虚拟机的负载均衡度 $LB_X=\sqrt{\frac{1}{m}\times\sum_{j=1}^{m}\left(VT_j-\overline{VT}\right)^2}$。负载均衡度 LB_X 数值越小，表明云计算系统中各虚拟机之间的任务负载越均衡。

定义 9：对于 n 个不同任务 $T=\{t_i\}$ 调度到 m 个不同虚拟机 $VM=\{vm_j\}$ 的任务调度问题是寻找分配方案 X，使得该分配方案中虚拟机的任务最迟完成时间最早 $TS(n,m)=\underset{X\in\aleph}{Min}\left(\underset{j=1}{\overset{m}{Max}}\left(VT_j\right)\right)$，或者说各虚拟机的最长处理时间最短 $TS(n,m)=\underset{X\in\aleph}{Min}\left(\underset{j=1}{\overset{m}{Max}}\left(\sum_{i=1}^{n}x_{ij}\times c_{ij}\right)\right)$，且负载均衡度 LB_X 最小。

1.2 最早完成时间的启发式优先分配策略推导

根据**定义 9**，对于 n 个不同任务分配到 m 个不同虚拟机的任务调度问题是寻找分配方案，使得虚拟机的最长处理时间 $TS(n,m)$ 最短且负载均衡度 LB_X 最小。当只有一个任务的调度问题时，$TS(1,m)=\underset{j=1}{\overset{m}{Min}}\left(c_{1j}\right)$。当有 k-1 个任务的调度问题时，$TS(k-1,m)=Min\left(\underset{j=1}{\overset{m}{Max}}\left(\sum_{i=1}^{k-1}x_{ij}\times c_{ij}\right)\right)$

$=Min\left(\underset{j=1}{\overset{m}{Max}}\left(vt_{(k-1)j}\right)\right)$。

定理 1：对于 k 个任务的调度问题，假设第 k 个任务 t_k 分配给第 z 个虚拟机 vm_z，即第 z 个虚拟机 vm_z 的时间跨度为 $makespan_{kz}=vt_{(k-1)z}+c_{kz}$，且 $\begin{cases} vt_{kj}=vt_{(k-1)j} & if\quad j\neq z \\ vt_{kj}=vt_{(k-1)j}+c_{kj} & if\quad j=z \end{cases}$，则满足递推关系

$TS(k,m)=\underset{z=1}{\overset{m}{Min}}\left(TS(k-1,m),makespan_{kz}\right)$。

证明：由**定义 9** 给出的 $TS(k,m)$ 定义可知，

$$TS(k,m)=Min\left(\underset{j=1}{\overset{m}{Max}}\left(\sum_{i=1}^{k}x_{ij}\times c_{ij}\right)\right)=$$

$$\underset{z=1}{\overset{m}{Min}}\left(Max\left\{vt_{(k-1)1},L,vt_{(k-1)m},vt_{(k-1)z}+c_{kz}\right\}\right)=$$

$$\underset{z=1}{\overset{m}{Min}}\left(\underset{j=1}{\overset{m}{Max}}\left(vt_{(k-1)j}\right),vt_{(k-1)z}+c_{kz}\right)=$$

$$\underset{z=1}{\overset{m}{Min}}\left(\underset{j=1}{\overset{m}{Max}}\left(vt_{(k-1)j}\right),makespan_{kz}\right)=$$

$$\underset{z=1}{\overset{m}{Min}}\left(\underset{j=1}{\overset{m}{Max}}\left(\sum_{i=1}^{k-1}x_{ij}\times c_{ij}\right),makespan_{kz}\right)=$$

$$\underset{z=1}{\overset{m}{Min}}\left(TS(k-1,m),makespan_{kz}\right) \qquad (1)$$

由式(1)可得定理 1 成立。因此，由定理 1 可得最早完成时间的启发式优先分配策略：第 k 个任务 t_k 将分配给具有最早完成时间的虚拟机 vm_z。

1.3 基于最早完成时间的启发式优先分配策略的启发式任务调度算法

本文提出了基于最早完成时间的优先分配策略求得任务调度解的启发式调度算法，其中包括基于先易后难优先分配策略的启发式任务调度算法和基于先难后易优先分配策略的启发式任务调度算法。该启发式任务调度算法的基本步骤如下：

步骤 1　计算任务集合 T 中 n 个任务在虚拟机集合 VM 的 m 个虚拟机上的预期执行时间，得到预期执行时间矩阵 C；

步骤 2　初始化 m 个虚拟机的当前负载数组 $vt[1..m]$为零；

步骤 3　从没有分配任何虚拟机的任务集合 T 开始，顺序访问每个任务 t_k，执行下述步骤 3.1、3.2 和 3.3，分配任务 t_k 到相应的虚拟机上；

步骤 3.1 依据 m 个虚拟机的当前负载数组 vt 和预期执行时间矩阵 C，计算出任务 t_k 分配至各个虚拟机相应的时间跨度 $makespan$；

步骤 3.2 基于最早完成时间的启发式优先分配策略，找出时间跨度 $makespan$ 最小(先易后难优先分配策略)或最大(先难后易优先分配策略)的虚拟机 vm_x；

步骤 3.3 分配任务 t_k 至虚拟机 vm_x，更新 vm_x 虚拟机负载 $vt[x]$为 $vt[x]+c_{kx}$；

2 基于不同分配策略的启发式调度算法和智能调度算法

任务调度问题已经被证明是一个 NP 完全问题，具体为：在 m^n 个任务调度的解空间寻找近似最优解，使得总任务的执行时间和负载均衡度最小，其中执行时间最小是满足用户的服务质量，负载均衡度最小是保证云环境的稳定性。

2.1 基于顺序调度策略、先易后难优先分配策略、先难后易优先分配策略等启发式任务调度算法

启发式调度算法以最早完成时间为目标进行调度，有着较好的负载均衡性能，但总任务的实际执行时间并非最少。

基于顺序调度策略的启发式任务调度算法把一组任务顺序分配给一组虚拟机，尽量保证每个虚拟机运行相同数量的任务以平衡负载，但没有考虑任务的需求和虚拟机之间的差别[1]。

基于先易后难优先分配策略的启发式任务调度算法采用先易后难的策略，先执行完成时间短的任务，然后执行完成时间长的任务，并采取贪心策略把每个任务优先指派给执行它最早完成的计算资源。

基于先难后易优先分配策略的启发式任务调度算法则恰恰相反，采用先难后易和贪心策略，每次选取完成时间最长的任务，优先指派给执行它最早完成的计算资源。

2.2 基于禁忌搜索策略、元胞演化策略等智能任务调度算法

智能任务调度算法通过对任务调度方案的编码，并依据禁忌搜索策略、元胞演化策略和差分进化算法等智能算法思想，在 m^n 大小的解空间多样性搜索和集中性搜索之间建立平衡，最终有效降低任务的执行时间。然而，智能调度算法在进行海量任务调度过程中，易陷入局部最优解，在收敛速度和负载均衡方面的效果有待提高[10]。

基于禁忌搜索策略的智能任务调度算法通过指导性的邻域搜索优化策略——禁忌搜索来产生候选交换任务对，采用贪心原则选择收益值大的任务对进行交换，并依据相应的禁忌准则避免迂回搜索，在将交换收益值作为启发信息搜索局部最优解的同时，更大自由地对解空间进行搜索[11]。

基于元胞演化策略的智能任务调度算法借助元胞自动机模型处理复杂系统问题的特点——根据微观个体的简单局域自组织相互作用机制来描述宏观系统整体复杂行为，通过构建任务调度问题对应的元胞自动机模型，循环遍历每个元胞产生候选交换元胞对，并采用贪心原则选择收益值大的元胞对进行交换。

3 云计算任务调度性能对比实验结果与分析

3.1 云计算任务调度性能对比实验模型

本文选用墨尔本大学网格实验室的 Cloudsim

云计算仿真平台[13]，具体的仿真对比实验步骤如下：

步骤 1　初始化 CloudSim 包：对 CloudSim 的用户数量、日期和跟踪标志等参数进行初始化；

步骤 2　创建数据中心：创建主机列表且设置每个主机的 ID、内存等配置参数，创建 PE 列表且设置每个 PE 的 ID、MIPS 等配置参数，创建数据中心对象等；

步骤 3　创建数据中心代理：通过对 DataCenterBroker 代理类扩展，在其内部分别实现基于顺序调度策略、先易后难优先分配策略、先难后易优先分配策略等启发式调度算法和基于禁忌搜索策略、元胞演化策略等智能调度算法；

步骤 4　创建虚拟机：对虚拟机的 ID、MIPS、CPU 数量等参数进行设置并提交任务代理；

步骤 5　创建云任务：创建指定参数的云任务，设定任务的用户 ID 并提交任务代理；

步骤 6　调用任务调度算法：借助自定义的任务调度策略，分配任务并绑定到虚拟机；

步骤 7　启动仿真；

步骤 8　收集仿真实验数据。

对比实验中的参数设置如下：为了保证实验结果的可重复性和公正性，固定选用的随机数种子为 1000；实验中设定虚拟机数量为 5，虚拟机指令执行速度在[100,300] MIPS 随机产生；五个对比实验分别设定任务数量为 500、1000、1500、2000、2500、3000、3500、4000、4500 和 5000，且每个任务指令长度在[1000,5000] MI 随机产生。通过分析仿真实验数据，对比不同分配策略的云计算任务调度性能。

3.2　云计算任务调度方案的完成时间可改进百分比及计算对比分析

定义 11：对于 n 个不同任务到 m 个不同虚拟机的某分配方案 X，各虚拟机的任务最迟完成时间(最长处理时间)为 $VT=\underset{j=1}{\overset{m}{Max}}\left(VT_j\right)$，定义该分配方案 X 的完成时间可改进百分比 δ 为 VT 与定义 7 的总任务最优完成时间 $\overline{VT}$ 百分比差，即 $\delta=\left(VT-\overline{VT}\right)\Big/\overline{VT}$。

对于某任务分配方案 X，其完成时间可改进百分比 δ 计算步骤如下。

步骤 1　计算出任务调度的平均负载，等于所有任务的总指令长度除以所有虚拟机指令执行速度累加和，即总任务最优完成时间 $\overline{VT}=\sum_{i=1}^{n}MI_i\Big/\sum_{j=1}^{m}MIPS_j$；

步骤 2　计算出分配给不同虚拟机 vm_j 所有任务的预期完成时间 $VT_j=\sum_{i=1}^{n}\left(x_{ij}\times c_{ij}\right)$；

步骤 3　计算出各虚拟机的任务最迟完成时间 $VT=\underset{j=1}{\overset{m}{Max}}\left(VT_j\right)$；

步骤 4　计算出完成时间可改进百分比 $\delta=\left(VT-\overline{VT}\right)\Big/\overline{VT}$。

表 1　基于不同分配策略的不同数量任务下算法的最迟完成时间对比表

Table1 The latest time comparison under different number of tasks and different allocation strategies

任务数	顺序调度	先易后难	先难后易	禁忌搜索	元胞演化	$\overline{VT}$
500	2855.32	1460.23	1450.72	1449.08	1448.8	1448.72
1000	5655.66	2884.57	2878.97	2877.47	2877.37	2877.07
1500	8472.68	4322.17	4314.76	4314.07	4313.61	4313.54
2000	11280.23	5757.05	5747.21	5746.02	5745.52	5745.51
2500	14071.73	7183.25	7171.57	7169.53	7169.43	7169.37
3000	17415.89	8635.8	8621.99	8619.01	8619.45	8618.99
3500	20153.25	10056.12	10050.02	10047.2	10047.79	10047.01
4000	23127.82	11513	11501.76	11500.28	11501.05	11500.19
4500	25750.98	12951.91	12938.75	12936.44	12937.26	12936.42
5000	28684.63	14407.41	14401.76	14399.06	14399.22	14399.03

表 1 为基于不同分配策略的不同数量任务下算法的最迟完成时间对比表，表明顺序调度策略是一种简单的轮询调度，算法简单，因此所耗费时间远长于其它分配策略，其完成时间可改进百分比接近 100%；元胞演化策略与先易后难优先分配策略、先难后易优先分配策略和禁忌搜索策略相比，所需完成时间最短并接近于任务最优完成时间 $\overline{VT}$。

图 1 的基于不同分配策略的不同数量任务下算法的完成时间可改进百分比对比表明：相比启发式调度算法，智能调度算法的完成时间最接近于任务最优完成时间 $\overline{VT}$，其中元胞演化策略的任务完成时间可改进百分最优。

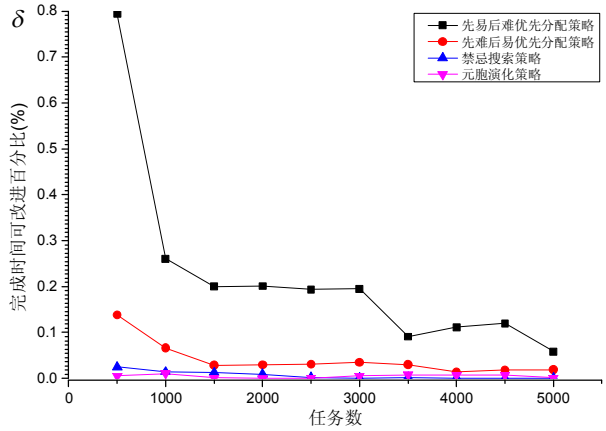

图 1　基于不同分配策略的不同数量任务下算法的完成时间可改进百分比对比图

Fig.1 The improvement percent of the latest time under different number of tasks and different allocation strategies

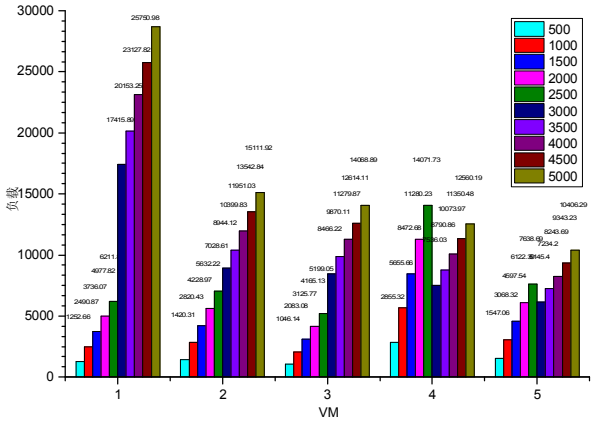

图 2　不同数量任务下顺序调度策略的资源负载平衡图

Fig.2 The load balancing of the sequential scheduling strategy under different number of tasks

3.3　云计算任务调度方案的资源负载平衡因子及计算对比分析

定义 12：对于 n 个不同任务到 m 个不同虚拟机的某分配方案 X，定义该分配方案 X 的资源负载平衡因子 LB_X 为不同虚拟机任务完成时间的方差值，即 $LB_X = \sqrt{\sum_{j=1}^{m}\left(VT_j - \overline{VT}\right)^2 \Big/ m}$。

对于某任务分配方案 X，其资源负载平衡因子 LB_X 计算步骤如下：

步骤 1.　计算出任务调度的平均负载，等于所有任务的总指令长度除以所有虚拟机指令执行速度累加和，即总任务最优完成时间 $\overline{VT} = \sum_{i=1}^{n} MI_i \Big/ \sum_{j=1}^{m} MIPS_j$；

步骤 2.　计算出分配给不同虚拟机 vm_j 所有任务的预期完成时间 $VT_j = \sum_{i=1}^{n}\left(x_{ij} \times c_{ij}\right)$；

步骤 3.　计算出不同虚拟机 vm_j 任务完成时间的方差值，即资源负载平衡因子 $LB_X = \sqrt{\sum_{j=1}^{m}\left(VT_j - \overline{VT}\right)^2 \Big/ m}$；

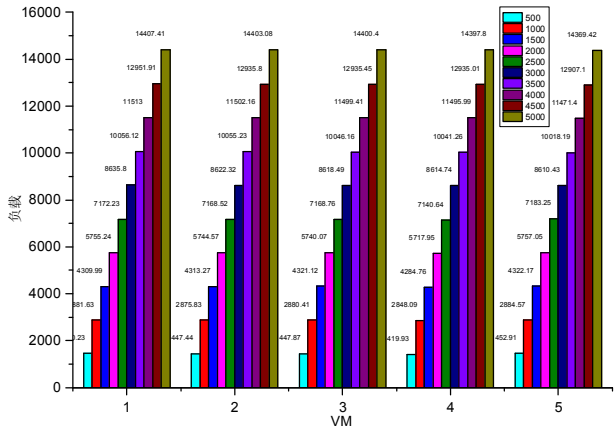

图 3　不同数量任务下先易后难优先分配策略资源负载平衡图

Fig.3 The load balancing of the priority-to-easy scheduling strategy under different number of tasks

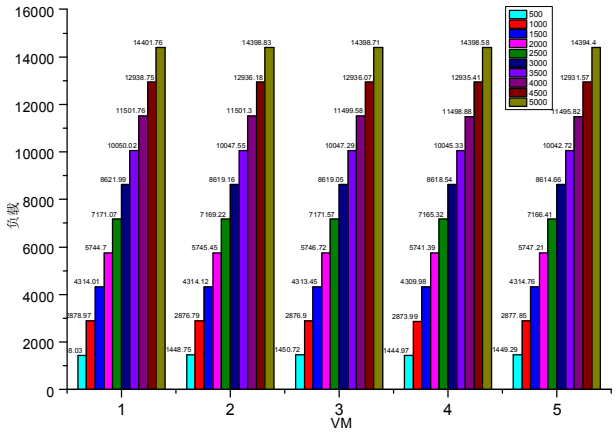

图 4　不同数量任务下先难后易优先分配策略资源负载平衡图

Fig.4 The load balancing of the priority-to-difficult scheduling strategy under different number of tasks

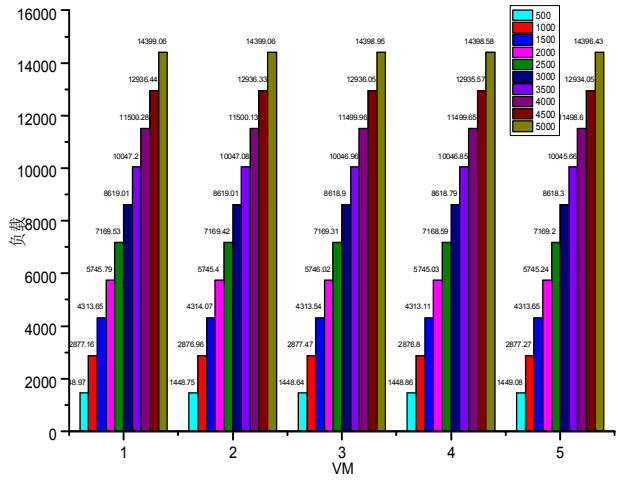

图 5 不同数量任务下禁忌搜索策略资源负载平衡图

Fig.5 The load balancing of the tabu search strategy under different number of tasks

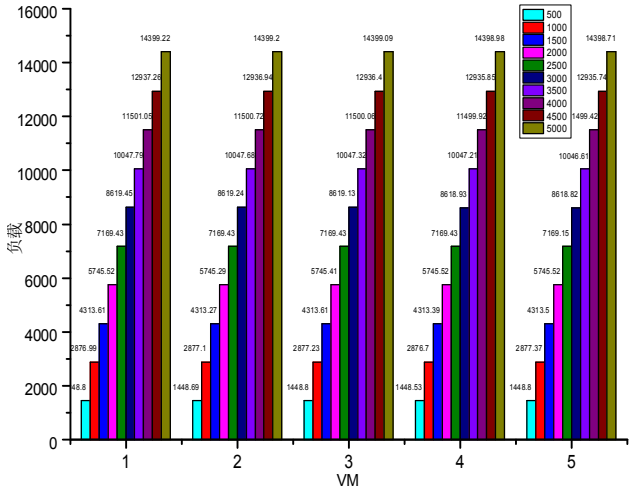

图 6 不同数量任务下元胞演化策略资源负载平衡图

Fig.6 The load balancing of the cellular automata strategy under different number of tasks

图 2 至图 6 为不同数量任务下各算法的资源负载平衡图，其中 number 代表一次调度的任务数量，分别为 500、1000、1500、2000 和 2500，资源编号为 5 个虚拟机的编号。图 2 至图 6 对比表明，图 2 的 Cloudsim 顺序调度策略由于是轮询调度，没有考虑计算资源的负载均衡，因此其资源负载平衡最差；图 4 的先难后易优先分配策略的资源负载平衡能力好于图 3 的先易后难优先分配策略；图 5 的禁忌搜索策略的资源负载平衡能力略好于图 4 的先难后易优先分配策略和图 3 的先易后难优先分配策略，但劣于图 6 的元胞演化策略的资源负载平衡能力。

表 2 不同数量任务的算法资源负载平衡因子对比表

Table 2 The load balancing factor comparison under different number of tasks

任务数	顺序调度	先易后难	先难后易	禁忌搜索	元胞演化
500	648.54	13.99	1.89	0.131	0.051
1000	1293.94	13.52	1.53	0.014	0.012
1500	1942.82	13.78	1.59	0.033	0.011
2000	2589.97	14.22	1.99	0.036	0.014
2500	3233.01	14.21	2.27	0.015	0.017
3000	4118.39	14.89	2.21	0.006	0.006
3500	4728.24	13.77	2.20	0.012	0.011
4000	5442.28	13.77	1.82	0.015	0.015
4500	6001.84	14.57	1.87	0.015	0.013
5000	6693.56	13.33	1.98	0.008	0.003

表 2 为不同数量任务的算法资源负载平衡因子对比表，表明顺序调度策略的资源负载平衡能力最差，元胞演化策略的资源负载平衡能力最好，与图 2 至图 6 的负载平衡对比结论一致：相比启发式调度算法，智能调度算法的资源负载均衡性能更优。

4 结束语

云计算环境下任务调度问题是在 m^n 个可能任务调度的解空间寻找近似最优解，使得总任务的执行时间最短且负载均衡度最小。本文在负载均衡任务调度问题形式化描述的基础上，通过形式化推导得到了最早完成时间的启发式优先分配策略，并给出了基于先易后难优先分配策略、先难后易优先分配策略的云计算任务调度算法。本文提出了完成时间可改进百分比和资源负载平衡因子的调度性能评价指标。最后，基于 CloudSim 云计算仿真实验平台下，进行了五种分配策略的云计算任务调度算法性能对比实验。实验数据对比充分表明：与启发式调度算法相比，智能调度算法能减少任务执行时间，优化资源负载均衡性能。

参考文献：

[1] 白延敏,薛进,马维华. 云环境下弹性负载均衡方法的研究[J]. 小型微型计算机系统, 2014, 35(4): 814-816.

[2] 叶枫,王志坚,徐新坤,等. 一种基于 QoS 的云负载均衡机制的研究[J]. 小型微型计算机系统, 2012, 33(10): 2147-2152.

[3] 查英华,杨静丽. 改进蚁群算法在云计算任务调度中的应用[J]. 计算机工程与设计, 2013, 34(5): 1716-1816.

(参考文献[4]- [9]转第 74 页)

48(11): 4085-4098.

[9] Bioucas-Dias J M, Plaza A, Dobigeon N, et al. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches[J]. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing , 2012, 5(2): 354-379.

[10] Iordache M D, Bioucas-Dias J M, Plaza A. Sparse unmixing of hyperspectral data[J]. IEEE Transactions on Geoscience and Remote Sensing, 2011, 49(6): 2014-2039.

[11] Stein D W J, Beaven S G, Hoff L E, et al. Anomaly detection from hyperspectral imagery[J]. Signal Processing Magazine, IEEE, 2002, 19(1): 58-69.

[12] Zhang L, Zhang L, Tao D, et al. Sparse transfer manifold embedding for hyperspectral target detection[J]. IEEE Transactions on Geoscience and Remote Sensing, 2014, 52(2): 1030-1043.

[13] Buades A, Coll B, Morel J M. A non-local algorithm for image denoising[C]. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, 2005, 2: 60-65.

[14] 徐冬,孙蕾,罗建书. 结合NAPCA和复小波变换的高光谱遥感图像去噪[J]. 红外与激光工程, 2015(1):327-334.

[15] 阳雄耀,钟平,王润生. 基于RTF的高光谱图像去噪方法[J]. 现代电子技术,2015(21):1-5,10.

[16] 印佳,杜战战. 基于主成分分析的高光谱遥感图像非局部去噪[J]. 现代电子技术, 2015(11):70-72.

[17] Acito N, Diani M, Corsini G. Subspace-based striping noise reduction in hyperspectral images[J]. IEEE Transactions on Geoscience and Remote Sensing,2011, 49(4): 1325-1342.

[18] Rasti B, Sveinsson J R, Ulfarsson M O, et al. Hyperspectral image denoising using first order spectral roughness penalty in wavelet domain[J]. Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of, 2014, 7(6): 2458-2467.

[19] Zhang H, He W, Zhang L, et al. Hyperspectral image restoration using low-rank matrix recovery[J]. IEEE Transactions on Geoscience and Remote Sensing, 2014, 52(8): 4729-4743.

[20] Candès E J, Recht B. Exact matrix completion via convex optimization[J]. Foundations of Computational mathematics, 2009, 9(6): 717-772.

[21] Cai J F, Candès E J, Shen Z. A singular value thresholding algorithm for matrix completion[J]. SIAM Journal on Optimization, 2010, 20(4): 1956-1982.

[22] Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to structural similarity[J]. IEEE Transactions on Image Processing, 2004, 13(4): 600-612.

[23] Wang Z, Bovik A C. Mean squared error: love it or leave it? A new look at signal fidelity measures[J]. Signal Processing Magazine, IEEE, 2009, 26(1): 98-117.

(上接第 68 页)

[4] 史少锋,刘宴兵. 基于动态规划的云计算任务调度研究[J]. 重庆邮电大学学报:自然科学版,2012,24(6): 687-692.

[5] 李建锋,彭舰. 云计算环境下基于改进遗传算法的任务调度算法[J]. 计算机应用, 2011, 31(1): 184-187.

[6] 封良良,张陶,贾振红,等. 云计算环境下基于改进粒子群的任务调度算法[J]. 计算机工程, 2013, 39(5): 183-188.

[7] 叶菁,陈国龙,阮一文. 异构环境下相关任务调度免疫遗传算法的研究[J]. 小型微型计算机系统, 2011, 32(10): 2124-2129.

[8] 董丽丽,黄贲,介军. 云计算中基于差分进化算法的任务调度研究[J]. 计算机工程与应用,2014,50(5): 90-95.

[9] 孙凌宇,冷明,朱平,等. 云计算环境下基于禁忌搜索的负载均衡任务调度优化算法[J]. 小型微型计算机系统, 2015, 36 (9): 1948-1952.