云计算环境下基于蚁群优化的任务负载均衡调度算法

赵 梦，李蜀瑜
（陕西师范大学 计算机科学学院，陕西 西安 710119）

摘要：随着云计算的蓬勃发展，针对云计算中虚拟机负载不均衡及任务集完成时间较长的问题，提出了一种基于蚁群优化的任务负载均衡调度算法（WLB-ACO）。首先基于当前虚拟机的资源利用情况判断虚拟机的负载状态，其次，根据虚拟机的负载因子定义信息素的挥发因子（w），改进信息素更新规则，并利用 WLB-ACO 合理的分配任务，使整个系统达到负载均衡状态的同时任务集的完成时间最短。最后，采用 Cloudsim 工具设计仿真实验，实验结果表明，提出的基于蚁群优化的任务调度算法在性能、最短完成时间及算法的稳定收敛性上有了一定的提高。

关键词：云计算；任务调度；负载均衡；最短完成时间；蚁群优化
中图分类号：TN915 文献标识码：A 文章编号：1674-6236（2016）08-0030-04

Load balancing of task scheduling based on ant colony optimization in cloud computing environment

ZHAO Meng，LI Shu-yu
（*School of Computer Science，Shaanxi Normal University，Xi'an* 710119，*China*）

Abstract: With the rapid development of cloud computing，cloud computing for the virtual machine load imbalance and take a long time to complete tasks set ，it is proposed based on ant colony optimization task scheduling algorithm load balancing（WLB-ACO）. Firstly，the current virtual machine based on the resource utilization in order to determine a load state of the virtual machine ，and secondly，the pheromone volatile factors （w）is defined by virtual machine 's load factor. To improve the pheromone update rules ，and use WLB-ACO rational allocation of tasks ，completion of the entire system to achieve load balancing state while the minimum set of tasks . Finally ，the use of simulation tools to design Cloudsim experimental results show that the task scheduling based on ant colony optimization algorithm in terms of performance，the shortest time to complete a certain increase stability and convergence of the algorithm.

Key words: cloud computing；task scheduling；load balancing；the shortest completion time；ant colony optimization
DOI:10.14022/j.cnki.dzsjgc.2016.08.009

随着计算机技术的蓬勃发展，传统的计算模式在现阶段已不能完全满足用户的需求。云计算应运而生，它通过计算机网络向用户提供满足用户需求的并且灵活、可伸缩的计算和存储资源。云计算有着非常广泛的用户群体，几乎时刻都在处理海量的任务。因此，考虑如何合理地分配和利用云环境中的资源、有效地调度用户提交的海量任务且保证整个系统的负载均衡但也面临着一些问题，它会导致任务的完成时间增加、资源的利用率及任务的完成效率达不到期望的要求等等。为了解决这个问题，文献[4]中将 MapReduce 模型与 ACO 算法的动态性、并行性融合到了 Hadoop 架构中，不仅提高了对系统资源的利用率，而且使任务的完成时间最短；文献[5]中将整个系统的负载状态与蚁群算法结合起来，利用蚁群算法完成系统的负载均衡，并且使得资源的利用率有了大大的提升；文献[6]中利用蜂群算法来改善任务分配的问题，该算法在多目标约束条件下不仅达到负载均衡的状态并且使得任务集完成时间最短。文献[7]针对该问题提出了新的蚁群算法中信息素的更新规则，可降低任务执行时间，还可有效保持云中虚拟机的负载平衡。然而，上述文献中对于负载均衡和任务集的最短完成时间的折中考虑不足。因此针对上述问题，本文提出了在整个系统达到负载均衡状态时且任务集的完成时间最短。

1 云计算任务调度模型

1.1 负载均衡

假设一个虚拟机的工作负载的详细状态可由一个计算资源利用向量[3]来表示，本文利用 CPU 利用率，内存利用率和带宽利用率这 3 个向量来表示资源的利用向量$\vec{\lambda}$=(cpu，memory，bandwidth)，$\vec{\lambda}$称之为状态向量。

云计算任务调度问题描述为将 n 个任务随机分配到 m 个虚拟机上，在其分配任务的过程中考虑虚拟机的负载均衡

收稿日期：2015-11-13 稿件编号：201511124
基金项目：国家自然科学基金面上项目（41271387）
作者简介：赵 梦（1990—），女，陕西西安人，硕士研究生。研究方向：云计算。

因素。假设 i 任务的运行都需要满足 3 个资源需求 q，定义一个负载均衡因子 LB_{ij} 来衡量 i 任务分配到虚拟机 j 上后该虚拟机的负载情况，假定任务 i 分配到虚拟机 j 上，根据下式计算虚拟机 j 的负载均衡因子：

$$LB_{ij}=\sum_{q-1}^{3}\omega_{ij}^{q}(1-u_{ij}^{q}) \tag{1}$$

上式中：

当 q=1 表示 CPU 资源；当 q=2 表示内存资源；当 q=3 表示带宽资源。

ω_{ij}^{q} 表示任务 i 分配到虚拟机 j 后，ω_{ij}^{1} 表示 CPU 资源在该虚拟机的 3 种资源中所占的权重；ω_{ij}^{2} 表示内存资源在该虚拟机的 3 种资源中所占的权重；ω_{ij}^{3} 表示带宽资源在该虚拟机的 3 种资源中所占的权重；

u_{ij}^{q} 表示任务 i 分配到虚拟机 j 上后虚拟机 j 上可用的 q 资源的利用率：

$$u_{ij}^{q}=\frac{\text{request\_Resource}_{i}^{q}}{\text{available\_Resource}_{j}^{q}} \tag{2}$$

$\text{request\_Resource}_{i}^{q}$ 表示任务 i 请求 q 资源的数量；

$\text{available\_Resource}_{j}^{q}$ 表示虚拟机 j 上可用的 q 资源的数量；

而 $\text{available\_Resource}_{j}^{q}$ 的计算如下：

$$\text{available\_Resource}_{j}^{q}=\text{total\_Resource}_{j}^{q}-\sum_{k-1}^{r}\text{request\_Resource}_{i}^{q} \tag{3}$$

其中 r 表示已经分配到虚拟机 j 上任务的个数，$\text{tatal\_Resource}_{j}^{q}$ 表示 j 虚拟机上 q 资源的总量。

通过上式计算可得负载均衡因子 LB_{ij} 和资源的利用率 u_{ij}^{q}，我们可以定义当前 VM_j 节点的负载状态 Load_statue_j，分别是：超载状态（Overloaded）、满载状态（Fullloaded）、正常负载状态（Normalloaded），其定义如下：

$$\text{Load\_statue}_j=\begin{cases}\text{Overloaded} & \exists q \quad u_{ij}^{q}>1\\ \text{Fullloaded} & \exists q \quad u_{ij}^{q}=1\\ \text{Normalloaded} & \exists q \quad 0<u_{ij}^{q}<1\end{cases} \tag{4}$$

如果，$u_{ij}^{q}\geqslant 1$ 表示任务 i 请求 q 资源的数量大于或等于虚拟机 j 上可用 q 资源的数量，因此虚拟机 j 上可用的资源不能满足任务 i 的资源请求，如果强制将任务 i 分配到虚拟机 j 上，则 j 虚拟机就会处于超载或满载状态，从而导致该虚拟机负载不均衡或虚拟机的利用率降低。因此，该任务不会分配到该虚拟机上，会转移到下一个可以接收该任务的虚拟机上。反之，如果 $0<u_{ij}^{q}<1$，表示该虚拟机处于可负载状态，且资源利用率不高，可以给该虚拟机分配任务。当虚拟机负载过重即 $u_{ij}^{q}\geqslant 1$ 时，它会对云计算服务质量与任务集的完成时间产生影响。

1.2　最短完成时间

最短完成时间是指使任务集的完成时间最短[6]，负载均衡技术对于减少任务集的完成时间是非常有效的。从用户的角度来看，是总任务完成时间最小化，因此，每个任务的完成时间也不能被忽略。

第 i 个任务在第 j 个虚拟机上的处理时间：

$$t_{ij}=\frac{\text{tasklength}_i}{\text{processV}_j} \tag{5}$$

当前任务 i 在某个虚拟机上的处理时间 CT_{ik}：$CT_{ik}=\{t_{i1},t_{i2},\cdots,t_{ij}\cdots,t_{im}\}$

tasklength_i 表示当前任务 i 和已分配到 j 虚拟机上的所有任务的长度之和，单位为 MI（Million Instructions）；processV_j 指的是 VM_j 的执行速度，单位是 mips，即每秒处理百万条的指令数。

转移时间：如果任务 i 分配给 VM_j 后，Load_statue_j 为 Fullloaded 或者 Overloaded 状态，则表明该 VM_j 超载，那么需要将该任务 i 从 VM_a 转移到 VM_b，即任务 i 从 VM_a 转移到 VM_b 所花费的时间；假如该虚拟机可以处理该任务而不导致虚拟机处于超载或满载状态，则该任务的转移时间为 0。

$$\text{transferTime}_{a,b}^{i}=\max\{\text{tasklength}_i/\text{sendV}_a,\text{tasklength}_i/\text{receiveV}_b\} \tag{6}$$

sendV_a 表示 VM_a 向 VM_b 发送任务的速度，单位为 mips；

receiveV_b 表示 VM_a 从 VM_a 接收任务的速度，单位为 mips。

任务的转移时间为：

$$CT_i=t_{ib}+\text{transferTime}_{a,b}^{i} \tag{7}$$

我们用 $\text{CT}_{\max}$ 表示最短完成时间，理想情况下：

$$\text{CT}_{\max}=\max\{t_{ib}+\text{transferTime}_{a,b}^{i}\} \tag{8}$$

2　基于负载均衡的任务调度算法(WLB-ACO)

1）基于蚁群算法中经典的旅行商问题进行设计，设有 n 只蚂蚁，用蚂蚁的一次旅行表示任务集的分配过程，在蚂蚁的一次旅行中蚂蚁需要走 m 步，每走一步表示分配一项任务，这样就形成了一个 $n\times m$ 的矩阵，蚂蚁所走的步数为 S；当所有蚂蚁完成一次旅行，视为算法循环一次；用 N_c 表示算法循环的次数。

初始时刻，各个虚拟机上的信息素量相等，设 $\tau_{ij}(0)=\frac{1}{m}$（m 为虚拟机的个数）。

所以，蚂蚁 k 选择虚拟机 j 完成任务 i 的概率为：

$$p_{ij}^{k}=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[LB_{ij}(t)]^{\beta}}{\sum_{s\in \text{allowedk}}[\tau_{is}(t)]^{\alpha}[LB_{is}(t)]^{\beta}} & \text{if } j\in \text{allowed}_k\\ 0 & \text{otherwise}\end{cases} \tag{9}$$

其中，allowed_k 表示蚂蚁还未选择的虚拟机的集合；$\tau_{ij}(t)$ 是虚拟机 VM_j 在 t 时刻信息素浓度值，$\text{LB}_{ij}(t)$ 是表示 VM_j 的负载均衡因子，用于维持虚拟机的负载均衡。虚拟机的负载均衡因子 LB_{ij} 越小，则选择该虚拟机执行下一个任务的概率就越高，也就是 VM_j 的综合能力比较强，期望值也就高。α，β 这两个系数表示的是：控制信息素和虚拟机的负载程度以及虚拟机的效率值所占的权重指数且 $\alpha+\beta=1$。

2）虚拟机信息素更新策略：

为了使负载更加均衡，在信息素更新规则上增加挥发因子[8]，该挥发因子主要用来描述信息素的挥发变化情况，而不再是常数。

$$\omega=1-\left|\frac{\sum \text{totaltasklength}_j-\text{tasklength}_i}{\sum \text{totaltasklength}_j+\text{tasklength}_i}\right| \quad (10)$$

信息素的更新:

$\tau_{ij}(t)$是虚拟机 VM_j 在 t 时刻的信息素浓度,信息素浓度的更新公式如下所示:

$$\tau_{ij}(t+1)=(1-\omega*\rho)*\tau_{ij}(t)+\Delta\tau_{ij}(t,t+1) \quad (11)$$

上式中:$\rho\in(0,1)$是信息素的衰减系数,如果 p 值越大就表明路径的信息素浓度受以前的影响程度就越小,$\Delta\tau_{ij}(t,t+1)$表示在本次循环中在 VM_j 上的信息素增量,其值更新如下:

当一只蚂蚁完成了它的遍历之后, 在它所爬行过的虚拟机节点上应用局部信息素更新,这时 $\Delta\tau_{ij}(t,t+1)$值的定义如下所示:

$$\Delta\tau_{ij}(t,t+1)=D_1/T_{ij} \quad (12)$$

T_{ij} 表示任务 i 在 VM_j 上处理花费的时间,D_1 是一个常数。

当一只蚂蚁完成了它的遍历,然后找出在当前时刻任务的最短完成时间, 那么就应该给这次路径旅行一个增强值,对它所爬行过的虚拟机节点进行全局信息素更新,这时,$\Delta\tau_{ij}(t,t+1)$值的定义:

$$\Delta\tau_{ij}(t,t+1)=D_2/\text{bestclock}_{ij} \quad (13)$$

$$\text{bestclock}_{ij}=\min\{CT_{ik}\}\quad k\in 1,2,\cdots,m \quad (14)$$

D_2 是激励系数[9],bestclock_{ij} 表示此次最优分配方案中任务 i 在虚拟机 j 上的完成时间。

3)WLB-ACO 的算法步骤

为了搜索基于 WLB-ACO 的云任务调度策略的最小任务完成时间步骤如下:

①初始化云平台虚拟机的信息素浓度;

②所有蚂蚁随机的选择任务;

③每只蚂蚁根据公式(1)和(9)选择完成任务的虚拟机;

④当一只蚂蚁完成了他的任务分配后,依据公式(12)进行信息素的局部更新策略。

⑤当所有的蚂蚁完成一次迭代之后,即所有蚂蚁都完成了路径搜索时,跳转到步骤⑥,同时找到本次迭代中的最优路径,然后对路径上的虚拟机按照公式(13)进行全局信息素更新;否则跳转到步骤②;

⑥计算每只蚂蚁的的任务完工时间,并保留当前时刻的最小的任务完工时间;

⑦判断是否满足迭代结束条件;如果满足,则结束迭代过程并输出最优任务分配方案,否则,转到步骤 ②直到满足迭代条件为止。

3 实验结果及分析

本文采用的实验平台是 CloudSim[12]云计算仿真软件,实验的任务调度算法包括基本蚁群算法、FCFS(First Come First Service)算法和 WLB-ACO。实验中创建了 100~500 个任务,任务的计算工作量在 1 000 MI 到 10 000 MI 之间,虚拟机的处理器速度在 200 MIPS 到 2 500 MIPS 之间, 虚拟机的内存为 512~2 096 MB,带宽为 512~1 048 bit,每个虚拟机的处理器数目为 2~6 个。

在 CloudSim 平台下实现 WLB-ACO 与其他算法在任务调度方面评价比较,WLB-ACO 的参数设置为: 蚂蚁数量 20 只,迭代次数为 0~500 次,α=0.3,β=0.7,ρ=0.02。

在实验中,通过设置不同迭代次数,比较 FCFS 算法、基本蚁群算法(ACO)和 WLB-ACO,3 种算法最短完成时间的比较如图 1 所示,3 种算法收敛速度的比较如图 2 所示;3 种算法的负载均衡值的表现如图 3 所示。

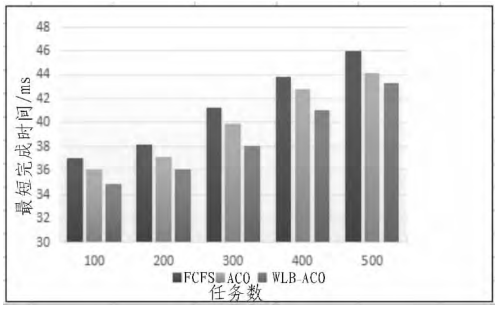

图 1 最短完成时间的比较

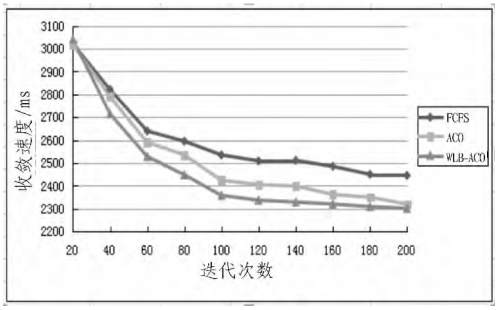

图 2 收敛速度比较

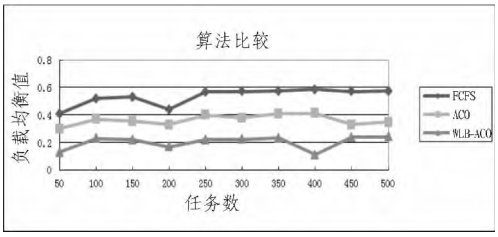

图 3 负载均衡值比较

从上图分析可知, 相比于基本蚁群算法及先来先服务(FCFS)算法,WLB-ACO 在任务集的最短完成时间、收敛速度有了 1%~4%的提升,且在任务数为定值的情况下,负载均衡值相较于其他两种算法也有所降低。因此,WLB-ACO 在最短完成时间、收敛速度和负载均衡 3 个方面都优于其他两种算法。从而,WLB-ACO 具有可行性。

4 结束语

本文基于云环境下任务调度模型[14-15],提出了负载均衡与最短完成时间约束的模型。结合 3 种模型, 本文提出的 WLB-ACO, 不仅在整个系统达到负载均衡的状态下任务集的完成时间最短,而且 WLB-ACO 的性能也优于基本的蚁群

算法和 FCFS 算法，资源的利用率有大大的提高。然而，本文中的任务集只考虑了各子任务相互独立的情况，但实际应用中并非如此。因此，下一步的研究目标应是考虑各个子任务存在关联的情况下，如何提高云计算当中任务调度的综合性能。

参考文献：

[1] 张浩荣. 云环境下基于蚁群算法的任务调度策略研究[D]. 广东:广东工业大学，2014.

[2] 吴皓. 云环境下任务调度算法研究[D]. 南京:南京邮电大学，2013.

[3] Zhenhua Wang. Workload Balancing and adaptive resource management for the swift storage system on cloud[J]. Future Generation Computer Systems，2014，38(4):32–37.

[4] 史恒亮. 云计算任务调度研究[D]. 南京:南京理工大学，2012.

[5] Li K，Xu G，Zhao G，et al. Cloud task scheduling based on load balancing ant colony optimization[C]//Chinagrid Conference(ChinaGrid)，2011 Sixth Annual. IEEE，2011:3–9.

[6] Krishna P V. Honey bee behavior inspired load balancing of tasks in cloud computing environments[J]. Applied Soft Computing，2013，13(5):2292–2303.

[7] 陈冬林，姚梦迪. 基于蚁群算法的云计算联盟资源调度[J]. 武汉理工大学学报，2014，36(3):337–341.

[8] 孙冬冬，柳青. 面向负载均衡的自主式虚拟机动态迁移框架[J]. 计算机科学，2014，41(4):80–86.

[9] 段卫军，付学良.云计算环境下融合遗传算法和蚁群算法QoS约束任务调度[J]，2014，34(S2):66–69.

[10]尔雅莉，王庆生. 云中基于蚁群算法改进的负载均衡策略[J]. 计算机工程与设计，2014，35(12):4095–4099.

[11]Yu Q，Chen L，Li B. Ant colony optimization applied to web service compositions in cloud computing[J]. Computers & Electrical Engineering，2015，41:18–27.

[12]Nishant K，Sharma P，Krishna V，et al. Load balancing of nodes in cloud using ant colony optimization[C]//Computer Modelling and Simulation (UKSim)，2012 UKSim 14th International Conference on. IEEE，2012:3–8.

[13]Zhang Z，Zhang X. A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation[C]//Industrial Mechatronics and Automation (ICIMA)，2010 2nd International Conference on. IEEE，2010，2: 240–243.

[14]De Falco I，Laskowski E，Olejnik R，et al. Extremal Optimization applied to load balancing in execution of distributed programs[J]. Applied Soft Computing，2015，30:501–513.

[15]黄俊，王庆凤. 基于资源状态蚁群算法的云计算任务分配[J]. 计算机工程与设计，2014，35(9):3305–3309.

(上接第 29 页)

[11]李海燕.基于Android移动电子商务平台的设计与实现[J].计算机安全，2014(7):36–39.

[12]肖磊.基于云计算的移动商务研究与实现[D].南昌：南昌航空大学,2013.

[13]武海斌. 推送平台:大数据时代中Android系统的冲锋舟[J]. 中国电子商务，2013(11):39.

[14]杨林,周岩. 基于Android平台的服药提醒系统[J].中国电子商务，2014(22):46.

[15]周勇文.基于Android平台的电子优惠券系统的研究与设计[J]. 信息与电脑：理论版，2014(10)：45.