

基于负载均衡的云计算资源分配算法

张继荣, 陈 琛

(西安邮电大学 通信与信息工程学院, 陕西 西安 710061)

摘 要: 针对云计算资源分配过程中负载不均问题, 提出了一种基于能耗成本和用户成本的改进蚁群算法(CLB-ACO). 首先, 在云环境下建立用户成本及运营商能耗成本的函数模型; 其次, 在信息素更新机制中引入动态调节因子 DRF, 使算法经过多次迭代后达到负载均衡状态; 最后, 在 Cloudsim 仿真平台上进行仿真测试, 测试结果表明, 该算法能较好地降低任务执行成本, 满足资源负载均衡.

关键词: 云计算; 蚁群优化算法; 资源分配; 成本函数模型; 负载均衡

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2017)09-0043-05

Resource Allocation Algorithm for Cloud Computing Based on Load Balancing

ZHANG Ji-rong, CHEN Chen

(School of communication and Information Engineering, Xi'an University of Posts and
Telecommunications, Xi'an 710061, China)

Abstract: In this paper, an improved ant colony algorithm based on the cost of energy consumption and user is proposed to solve the problem of load imbalance in the process of cloud resource allocation. First, the energy consumption cost function model of users and operators is established in the cloud environment. Secondly, we have introduced the dynamic adjustment factor DRF in pheromone update mechanism, which makes the algorithm achieves load balancing state after several iterations. Finally, the simulation test is carried out on the cloud simulation platform named Cloudsim. The experimental results show that the algorithm can reduce the task execution cost, and meet the resource load balancing well.

Key words: cloud computing; ant colony optimization; resource allocation; the cost function model; load balancing

1 引言

云计算的目的是为用户提供快速可靠的服务, 这要求云平台对资源进行合理的分配管理, 在快速响应用户请求的同时也要充分利用资源, 保证系统的负载均衡, 但云计算资源分配的三个特点, 即云计算服务的动态性, 云计算资源分布的广泛性, 用户任务的多样性, 决定了资源分配的过程中必然会存在着巨大挑战, 而这些挑战使得资源分配成为了云计算领域中的研究热点之一.

已有研究证明资源分配问题是一个 NP 完全问

题^[1], 因此有很多学者采用了蚁群算法或改进的蚁群算法进行云计算资源分配. 如文献[2]提出云计算负载均衡的双向蚁群优化算法(BACO), 该算法能提高寻优能力, 实现负载均衡. 文献[3]提出基于资源状态蚁群算法, 文献[4]提出了一种云计算环境下基于用户体验的成本最优存储策略; 文献[5]提出云计算中运营商效益最优的资源分配机制, 该算法能大大节约运营商成本, 并提高资源利用率.

然而, 上述文献中对于负载均衡和成本问题考虑不足. 针对此问题, 本文在现有蚁群算法的基础上进行改进, 提出一种基于能耗成本的资源分配负

载均衡算法 (CLB-ACO), 并进行 Cloudsim 仿真实验.

2 云计算任务调度模型

2.1 问题描述

目前, 云计算环境基本采用 Google 提出的 Map/Reduce 并行编程模式^[6]. MapReduce 模型一般分为 Map(映射)阶段和 Reduce(规约)阶段. Map 是将提交的用户作业统一分割映射成若干个子任务. Reduce 是将 Map 所产生的结果集中整合分类. 根据该模型可知, 云环境下的任务调度问题可概括为:

(1) 系统将总任务切分为若干个子任务; (2) 系统根据子任务的相应信息采取分配算法将其都分配到合适的虚拟机资源上运行; (3) 当所有子任务运行结束, 系统将各个子任务的执行结果汇总成最终结果传给用户. 本文只考虑将各个相互独立的子任务按照分配算法分配到虚拟机资源的过程.

云计算资源分配问题可描述为: 在云环境中, 将 N 个相互独立的待执行子任务合理有效的分配给 M 个异构虚拟资源节点上执行 ($M < N$), 其中, 任务集合表示为: $\text{SubTask} = \{st_1, st_2, \dots, st_n\}$, $st_i (i = 1, 2, \dots, n)$ 表示第 i 个子任务; 虚拟资源集合表示为: $V = \{v_1, v_2, \dots, v_m\}$, $v_j (j = 1, 2, \dots, m)$ 表示第 j 个虚拟资源. 每个任务只能在一个虚拟资源节点上运行, 可以使用矩阵 R 来表示任务与虚拟资源间的分配关系,

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{nm} \end{bmatrix} \quad (1)$$

式(1)中, r_{ij} 表示子任务 st_i 与虚拟资源 v_j 的分配关系, $\sum_{j=1}^m r_{ij} = 1, i \in \{1, 2, \dots, n\}, j \in \{1, 2, \dots, m\}$, 如果子任务 st_i 在虚拟资源节点 v_j 上执行, 则 $r_{ij} = 1$, 否则 $r_{ij} = 0$.

2.2 成本函数模型

我们假设 m 个虚拟机资源 $\{v_1, v_2, \dots, v_m\}$ 运算速度分别为 $\{s_1, s_2, \dots, s_m\}$, 并且满足 $s_1 < s_2 < \dots < s_m$; m 个虚拟机资源对应的能耗表示为 $\{p_1, p_2, \dots, p_m\}$, p_j 表示虚拟资源 v_j 的单位时间能耗, 并满足 p_1

$< p_2 < \dots < p_m$ 且 $\lambda p_1 > p_m, \lambda \geq 2, \lambda$ 为整数; N 个子任务 $\{st_1, st_2, \dots, st_n\}$ 的任务长度为 $\{l_1, l_2, \dots, l_n\}$. 此外, 设单位时间单位能耗成本为 cost_p , 单位时间任务执行成本为 cost_q . 子任务 st_i 在虚拟资源节点 v_j 的执行时间为:

$$t_{ij} = \frac{\text{tasklength}_i}{\text{processV}_j} = \frac{l_i}{s_j} \quad (2)$$

公式(2)中, tasklength_i 表示子任务 st_i 的长度, processV_j 表示虚拟资源节点 v_j 的运算速度. N 个子任务在 M 个虚拟资源上的执行时间使用一个 $N \times M$ 的矩阵 T 来表示, 即

$$T = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & & \vdots \\ t_{m1} & t_{m2} & \cdots & t_{nm} \end{bmatrix};$$

则虚拟资源节点 v_j 能耗成本可表示为

$$f_{ij}^{\text{provider}} = \text{cost}_p \times p_j \times t_{ij} \times r_{ij} \quad (3)$$

子任务 st_i 分配到虚拟资源节点 v_j 上用户需支出的成本可表示为

$$f_{ij}^{\text{user}} = \text{cost}_q \times t_{ij} \times r_{ij} \quad (4)$$

子任务 st_i 在虚拟资源节点 v_j 总成本为

$$f_{ij} = a \times f_{ij}^{\text{provider}} + b \times f_{ij}^{\text{user}} \quad (5)$$

式(5)中, a, b 分别表示云资源提供商能耗成本和用户成本占成本的权重.

则云资源分配的总成本为

$$f = \sum_{j=0}^m \sum_{i=0}^n f_{ij} \quad (6)$$

该算法的优化目标是找到合适的资源分配关系矩阵, 使得 f 的值达到最小.

云资源分配的平均成本为

$$f_{\text{avg}} = \frac{f}{m} \quad (7)$$

2.3 任务资源匹配度函数

定义 子任务 $st_i (i = 1, 2, \dots, n)$ 与虚拟资源 $v_j (j = 1, 2, \dots, m)$ 的匹配度可定义为子任务请求资源量与虚拟机可用的资源量的欧氏距离

$$d_{ij}(t) = \sqrt{\sum_{\rho} \omega_{ij}^{\rho} (\text{AR}_{ij}^{\rho} - \text{RR}_{ij}^{\rho})^2} \quad (8)$$

公式(8)中, ρ 表示虚拟资源种类, 分别代表内存容量、CPU 和带宽. AR_{ij}^{ρ} 表示虚拟机 v_j 上可用的 ρ 资源量, RR_{ij}^{ρ} 表示子任务 st_i 请求的 ρ 资源量. ω_{ij}^{ρ} 表示内存容量、CPU 和带宽占总资源的权重.

由定义可以看出:子任务与虚拟机间的匹配度越小,虚拟机越能达到任务的需求,任务的执行效率越高。

3 基于能耗成本的蚁群优化负载均衡算法 (CLB-ACO)

蚁群算法 (Ant Colony Optimization, ACO) 由 Marco Dorigo 首次提出的一种仿生优化算法^[7], 由于其并行正反馈, 鲁棒性, 简单易实现的优点, 比较适合来解决云资源分配问题。但是 ACO 算法同时也存在初期求解速度慢, 易陷于局部最优的缺点, 为提高收敛速度, 实现全局最优, 本文在信息素初始化、虚拟机选择及信息素更新等方面均进行了改进。

3.1 初始化

初始时刻将所有蚂蚁随机放置于虚拟资源节点上, 并初始化所有虚拟资源节点的信息素, 受文献^[8]的启发, 可以使用虚拟机的硬件资源因子来进行信息素的初始化, 因此, 不同的虚拟资源节点上所对应的信息素大小均不相同, 则虚拟资源节点 v_j 的初始信息素为

$$\tau_j(0) = \omega_1 s_j + \omega_2 \text{ram}_j + \omega_3 \text{cpu}_j + \omega_4 \text{bw}_j \quad (9)$$

公式(9)中, s_j 、 ram_j 、 cpu_j 和 bw_j 分别表示虚拟机 v_j 的处理速度、内存容量、CPU 和带宽。 ω_1 、 ω_2 、 ω_3 和 ω_4 分别表示处理速度、内存容量、CPU、带宽各占总资源的权重, 且 $\omega_1 + \omega_2 + \omega_3 + \omega_4 = 1$ 。

初始化虚拟资源节点的禁忌表 tabu 、可搜索任务列表 ATatbu 和硬件资源状态表 HRtabu 。禁忌表 tabu 里存放的是已经访问过的虚拟资源节点; 可搜索任务列表 ATatbu 存放的是还未访问过的虚拟资源节点; 硬件资源状态表 HRtabu 存放的是虚拟机的硬件资源信息, 在进行搜索前, 需要对虚拟资源节点的处理速度, 内存容量, CPU 和带宽等进行信息的采集, 并将采集到的信息存入 HRtabu 中。在算法运行过程中, 集合 tabu 、 ATatbu 、 HRtabu 随着算法迭代过程在不断的进行动态调整。

3.2 选择虚拟机

将 n 个子任务随机放置在 m 个虚拟资源节点上, 并以概率 $P_{ij}(t)$ 和轮盘函数的方式进行任务与虚拟资源配对。 $P_{ij}(t)$ 表示在 t 时刻将子任务 st_i 分配给虚拟资源 v_j 的概率。 概率函数如下:

$$P_{ij}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{n \notin \text{tabu}} [\tau_{in}(t)]^\alpha [\eta_{in}(t)]^\beta}, & \text{if } j \notin \text{tabu} \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

式(10)中, $\tau_{ij}(t)$ 表示 t 时刻虚拟资源节点上的信息素浓度, $\eta_{ij}(t)$ 为启发信息, 本文选取匹配度的倒数作为启发信息, $\eta_{ij}(t) = \frac{1}{d_{ij}(t)}$, $d_{ij}(t)$ 的值越小, $\eta_{ij}(t)$ 的值越大, $P_{ij}(t)$ 的值就越大。 α 和 β 均是调节因子, 分别表示信息素浓度和启发信息的重要程度。

3.3 信息素更新

在分配过程中, 蚂蚁走过的路径上积累的信息素会随着迭代次数的增加而增多, 为防止路径中信息素积累过多而导致启发信息失效的问题, 蚁群算法中引入了信息素更新机制, 其包括局部更新策略, 全部更新策略。在每只蚂蚁完成所有任务分配后, 可利用公式(11)进行信息素的更新:

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

式(11)中, ρ 为信息素挥发因子, 表示单位时间信息素的挥发程度, 取值范围为 $\rho \in [0, 1)$; $\Delta\tau_{ij}$ 表示信息素的浓度增量, 使用总成本函数 f 定义。

① 局部信息素更新

当一只蚂蚁完成所有的任务分配后, 对所有的虚拟资源节点进行一次局部信息素更新, 此时信息素增量 $\Delta\tau_{ij}$ 可表示为:

$$\Delta\tau_{ij} = \frac{Q_1}{f} \quad (12)$$

式(12)中, f 表示蚂蚁在进行任务分配时的总成本, Q_1 为常量。

② 全局信息素更新

当所有蚂蚁完成所有的任务分配后, 可以得到资源分配的最优分配方案, 然后对所有的虚拟资源节点进行全局信息素更新, 此时信息素增量 $\Delta\tau_{ij}$ 可表示为:

$$\Delta\tau_{ij} = \frac{Q_2}{f_{\min}} \quad (13)$$

式(13)中, $f_{\min}$ 表示最优分配方案对应的总成本, Q_2 为常量。

3.4 引入动态调节因子

本文通过引入动态调节因子 DRF 来改进信息素更新规则, 改进后的信息素更新机制可以缓解虚拟机负载过重或较轻问题, 达到系统整体的负载均衡, 动态调节因子 DRF 是通过成本函数来定义的,

$$DRF = e^{-(\frac{f_{ij}-f_{avg}}{f})}$$
 (14)

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij} * DRF$$
 (15)

在某次迭代过程中,如果虚拟机 v_j 的负载过重,则任务 s_i 在此虚拟机的完成时间会更长,所花费的能耗成本会比其他虚拟机完成任务 s_i 的更大,则此时的动态调节因子 DRF 与其他虚拟机资源的相比,DRF 变小了, $\tau_{ij}(t+1)$ 则变小了,所以在下一次迭代过程中任务被分配到虚拟机 v_j 的可能性会降低,经过多次迭代之后,改进的蚁群算法 CLB-ACO 最终会实现系统整体的负载均衡。

4 算法流程及参数设置

基于能耗成本与用户成本的蚁群优化负载均衡算法(CLB-ACO)流程,如图 1 所示。

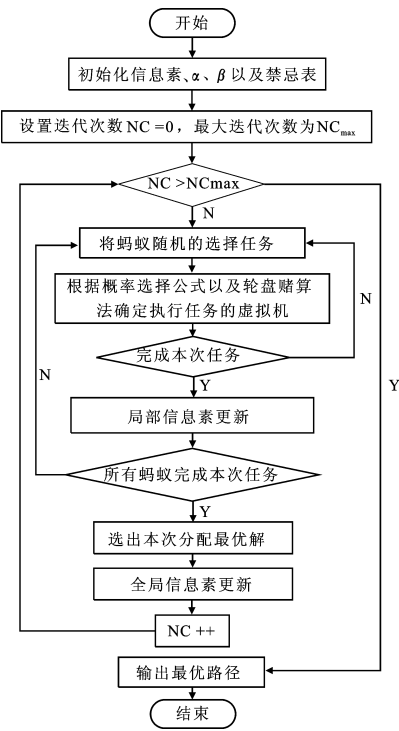


图 1 基于能耗成本的蚁群优化负载均衡算法

其中,根据已有文献的研究情况及算法基本框架,对算法进行参数设置,并且在进行设置时,应遵循对比试验中的三种算法共同涉及到的参数设置必须相同的原则,算法中参数设置如表 1 所示。

表 1 CLB-ACO 算法参数

参数项目	取值	
CLB-ACO	α	1
	β	2
	ρ	0.01
	NC_{max}	50

5 实验与分析

本文选用云计算仿真平台 Cloudsim3.0 来验证 CLB-ACO 算法在云计算资源分配上的有效性及可行性,利用 Cloudsim 来模拟云环境和资源,通过重写 DataCenterBroker,Cloudlet 类来构造 CLB-ACO 算法,并且在基于同等配置和参数设置的条件下与标准的蚁群算法以及文献[9]中的 MTRCACO 算法进行对比。

在本实验中,对 Cloudsim 模拟器参数进行设置,设置数据中心为 20 个,每个数据中心设置 10 个虚拟机,虚拟机性能参数进行随机设置:计算能力为 [500,2 000]MIP,内存为 [512,2 048]MB,带宽为 [5 000,10 000]b/s,用户任务数量为 200~1 000 个,任务长度随机设置为 [300,3 600]MI,最大迭代次数为 50 次。

(1)通过模拟实验对比相同条件下 CLB-ACO 算法与标准的蚁群算法以及文献[9]中的 LBACO 算法的任务执行成本,其结果如图 2 所示。

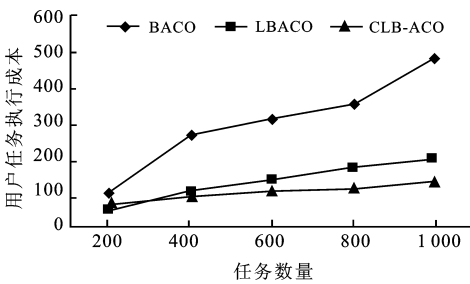


图 2 任务成本比较

从图 2 可看出,当任务量较少时,这三种算法的任务执行成本差别不大,随着任务量的增加,执行同样多数量的任务,CLB-ACO 算法的执行成本相对较低,说明 CLB-ACO 算法可以达到降低成本的作用。

(2)通过模拟实验对比 CLB-ACO 算法与标准的蚁群算法以及文献[9]中的 MTRCACO 算法的负载不均衡值,图 3 为负载不均值对比图。本文利用文献[9]中提出的虚拟机负载不均公式对完成资源分配后的虚拟机负载不均均衡度(L_j)进行测量,负载不均公式为:

$$L_j = \frac{f_j - f_{avg}}{m}$$
 (16)

式中, f_j 表示在虚拟资源节点 v_j 上的总成本。

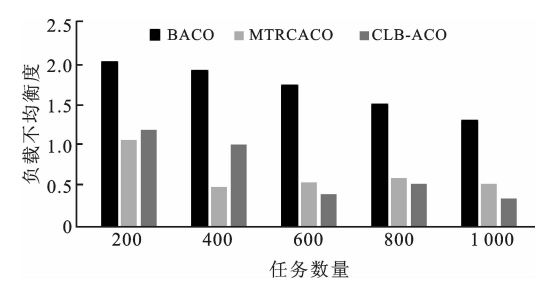


图 3 任务负载均衡比较

从图 3 中可看出,标准蚁群算法 BACO 的负载不均衡度在三种算法中最高,说明该算法不能达到负载均衡;当任务量较少时,MTRCACO 算法的负载不均衡度是最低的,但随着任务量的增加,CLB-ACO 算法的负载不均衡度低于另两种算法,说明 CLB-ACO 算法能在有效降低成本的同时,也达到负载均衡.

6 结束语

负载均衡是云计算资源分配过程中需要考虑的一个关键因素,针对此问题,引入了信息素动态调节因子,通过云计算资源分配能耗成本函数来调节因子的大小,实现系统的负载均衡,最后通过 Cloudsim 仿真平台进行了仿真,仿真结果表明 CLB-ACO 算法在云计算资源分配过程中,能有效降低成本并实现系统的负载均衡. 在下一步工作中,需要考虑的是在负载均衡条件下资源分配工作还需满足用户的多 QoS 需求.

参考文献:

[1] Mahesh A. A new common subexpression elimination algorithm for realizing low — complexity higher order

digital filters[J]. IEEE Trtans. Compt. -Aided Design Integr. Circuits Syst,2008,27(2):217-229.

[2] 王常芳,徐文忠. 一种用于云计算资源调度的双向蚁群优化算法[J]. 计算机测量与控制,2015,23(8):2861-2863.

[3] 黄俊,王庆凤,刘志勤,等. 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.

[4] 王宁,杨扬,孟坤. 云计算环境下基于用户体验的成本最优存储策略研究[J]. 电子学报,2014,42(1):20-27.

[5] 尹红军,李京,宋浒,等. 云计算中运营商效益最优的资源分配机制[J]. 华中科技大学学报(自然科学版),2011,39(z1):51-55.

[6] 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,2004:137-152.

[7] Dorigo M,Gambardella L M. Ant colony system;a co-operative learning approach to the traveling salesman problem[J]. IEEE Transaction on Evolutionary Computation,1997,1(1):53-66.

[8] 张千,梁鸿,李振. 基于改进蚂蚁算法的网格资源管理的研究[J]. 微电子学与计算机,2009,26(9):71-74.

[9] 聂清彬,霍敏霞,曹耀钦. 面向用户任务与云资源匹配度的调度算法研究[J]. 微电子学与计算机,2016,33(7):93-97.

作者简介:

张继荣 女,(1963-),博士,教授. 研究方向为交换机、接入设备和通信网络等. E-mail:comnet@xupt.edu.cn.

陈 琛 女,(1991-),硕士研究生. 研究方向为信号与信息处理.