基于蚁群模拟退火的云任务调度算法改进

秦 军[1],董倩倩[2],郝天曙[2]
(1. 南京邮电大学 教育科学与技术学院,江苏 南京 210003;
2. 南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:随着云计算的快速发展,如何高效地进行云任务调度逐渐成为云计算研究的重点。任务调度问题属于 NP 优化问题,许多超启发式算法被应用到任务调度问题。针对蚁群算法在任务调度中存在收敛速度慢、局部搜索能力差和易于陷入局部最优的问题,将蚁群算法和模拟退火算法相结合,提出了蚁群模拟退火算法,拟解决云计算中的任务调度问题。在该算法中,以减少任务的完成时间和保证资源负载均衡为目标,根据蚁群算法构造局部最优解,利用模拟退火算法较强的局部搜索能力,将局部最优解作为模拟退火算法的初始解进行局部搜索并以一定的概率接受当前搜索结果,从而避免算法陷入局部最优。仿真结果表明,蚁群模拟退火算法的性能优于先来先服务(First Come First Served,FCFS)和标准蚁群优化(Ant Colony Optimization,ACO)算法。

关键词:任务调度;云计算;蚁群算法;模拟退火算法
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2017)03-0117-05
doi:10.3969/j.issn.1673-629X.2017.03.024

Improvement of Algorithm for Cloud Task Scheduling Based on Ant Colony Optimization and Simulated Annealing

QIN Jun[1],DONG Qian-qian[2],HAO Tian-shu[2]
(1. College of Education Science & Technology,Nanjing University of Posts and Telecommunications,
Nanjing 210003,China;
2. College of Computer,Nanjing University of Posts and Telecommunications,
Nanjing 210003,China)

Abstract:With the rapid development of cloud computing,how to carry on task scheduling effectively is crucial in the research of cloud computing. Cloud task scheduling belongs to a NP-hard optimization problem,and many meta-heuristic algorithms have been proposed to solve it. ACO algorithm in task scheduling still has many shortcomings such as slow convergence speed,poor ability of local search and falling into local optimum easily. Therefore,a new algorithm-ACOSA is presented to solve task scheduling problem. In this algorithm,reducing task completion time and ensuring resource's load balance as the goal,according to the local ant colony algorithm the optimal solution is constructed,and the strong local search capability of simulated annealing algorithm is applied to make the local optimal solutions as the initial solutions of simulated annealing algorithm and accept the results of current search to a certain probability in order to avoid falling into the local optimal. Simulation results show that ACOSA is superior to First Come First Served (FCFS) and Ant Colony Optimization (ACO) by reducing make span and achieving load balance.

Key words:task scheduling;cloud computing;ACO;Simulated Annealing

1 概 述

云计算[1]是分布式计算、并行计算和网格计算的商业发展。云计算通过互联网实现计算设施、存储设备、应用程序等资源的共享,为不同的用户提供计算、存储等各种服务。在云计算环境中,用户通过付费的方式使用云提供商提供的服务或基础设施。根据用户提交的作业请求,云计算调度中心为其分配资源。然而,如何进行高效的任务调度仍然是云计算系统所面

收稿日期:2016-04-27 **修回日期**:2016-08-10 **网络出版时间**:2017-02-17
基金项目:江苏省自然科学基金项目(BK20130882)
作者简介:秦 军(1955-),女,教授,研究方向为计算机网络技术、多媒体技术、数据库技术;董倩倩(1989-),女,硕士研究生,研究方向为分布式计算机技术与应用。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20170217.1632.068.html

临的一个重大挑战。

云计算采用 Goolge 公司提出的 MapReduce[2] 框架结构。每一个单元是由一个独立的 Master 节点和多个 Worker 节点组成[3]。Master 节点负责调度所有的任务、监控任务的执行、重新运行失败的任务以及处理异常。Worker 节点只负责执行 Master 节点分配的任务[4-5]。一旦接收到 Master 节点分配的任务,Worker 节点首先要检测自身剩余的计算资源,如果 Worker 节点的资源可以满足用户需求,那么将优先分配该节点的资源。如果资源已经无法满足用户需求,Master 节点将会寻找其他合适的云计算资源。由此可见,任务调度策略的好坏直接影响任务执行效率和资源利用率,云计算的任务调度模型如图 1 所示。用户将任务提交到用户任务池中,任务分配节点根据 Master 节点分配的任务从任务池中取得相应的任务,通过任务调度器将不同的任务分配到不同的虚拟机。

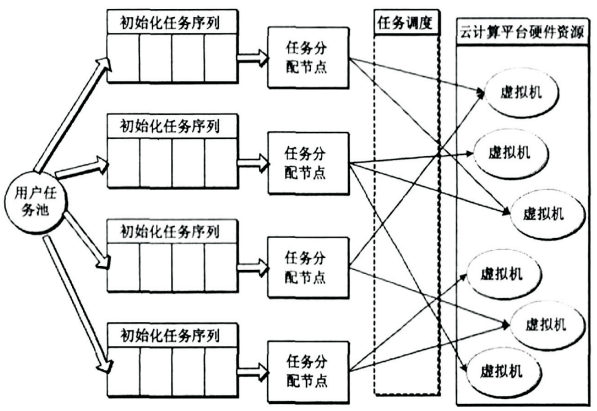

图 1 云计算任务调度模型

现有的启发式任务调度算法中,蚁群最优化算法[6]具有鲁棒性、正反馈、分布式计算和易于结合其他算法等特点[7]。因此,蚁群最优化算法的出现为各个领域解决复杂的组合问题提供了强大的工具。近年来,蚁群最优化算法在组合优化问题方面得到了持续发展,但是该算法依然存在易于停滞、早熟和易于陷入局部最优的缺点[8]。鉴于此,文中提出将蚁群算法和模拟退火算法相结合的蚁群模拟退火算法(ACOSA)。该算法以减少任务完成的时间为目标,同时考虑虚拟机资源的负载均衡[9-10]。

2 形式化描述

假设用户所提交的任务是相互独立、不可分割的;任务的执行顺序没有先后之分;任务一旦开始执行,除非出现虚拟机故障,在执行过程中任务不能中断。

定义 1:任务集 $T = \{T_1, T_2, \cdots, T_n\}$,表示当前队列有 n 个相互独立的任务。任务 T_i 由四元组{TID, InputFileSize, TLength, OutputFileSize}表示。其中,TID 表示任务编号;InputFileSize 表示任务执行前的长度;TLength 表示提交到虚拟机的任务长度;OutputFileSize 表示任务执行完成后的长度。

定义 2:虚拟机集 $V = \{VM_1, VM_2, \cdots, VM_m\}$,表示云计算数据中心当前可用的 m 个虚拟机资源。虚拟机资源 V_i 由{VMID, PesNum, MIPS, Band, Size}五元组表示。其中,VMID 表示虚拟机编号;PesNum 表示虚拟机的 CPU 数量,文中规定每一虚拟机只有一个 CPU;MIPS 表示虚拟机 CPU 指令执行速度;Band 表示虚拟机所允许的最大带宽;Size 表示虚拟机的存储大小。

3 基于模拟退火算法的蚁群算法

蚁群算法的基本原理是模拟自然界蚂蚁觅食过程中在路径上释放信息素,后面的蚁群根据路径上遗留信息素的浓度选择下一路径,遗留信息素的浓度越高,蚁群选择该路径的概率就越大,从而逐渐收敛于全局最优解的过程[11-12]。单纯的蚁群算法具有并行性、协同性和正反馈性等优点,但是在求解任务调度这种复杂的 NP 问题时,该算法存在局部搜索能力差、易于陷入局部最优解和收敛速度慢等问题[13]。

针对蚁群算法的缺点,文中提出 ACOSA。该算法首先利用蚁群算法在当前温度下构造一个局部最优解,其次根据当前局部最优解,利用模拟退火算法较强的局部搜索能力[14-15],构造局部最优解的一个邻域,在此邻域内通过置换规则构造一个新解,对此新解将按照一定的概率接受或拒绝,从而避免了蚁群算法陷入局部最优解;最后利用信息素更新原则更新全局信息素和退火降温,在新的温度下构造新的局部最优解。

3.1 初始化参数

根据任务规模的大小设计合理的初始温度 T_0,初始温度要足够高,否则算法将会收敛过快。初始化蚁群算法的相关参数,根据式(1)初始化蚁群算法中的虚拟机 V_j 的信息素。

$$\tau_j(0) = \sqrt{\mathrm{MIPS}_j + \mathrm{Band}_j} \tag{1}$$

其中,MIPS_j 表示虚拟机 V_j 的 CPU 指令执行速度;Band_j 表示虚拟机 V_j 允许的最大带宽。

3.2 状态迁移规则

为了虚拟机资源的负载均衡,ACOSA 算法将虚拟机的负载均衡情况作为启发函数。根据式(2)为下一任务 T_i 计算选择虚拟机 V_j 的概率。

$$P_j^k(t) = \begin{cases} \dfrac{[\tau_j(t)]^{\alpha}[\eta_j(t)]^{\beta}}{\sum_{s \in \mathrm{allowed}_k}[\tau_s(t)]^{\alpha} \times [\eta_s(t)]^{\beta}}, j \notin \mathrm{tabu}_k \\ 0, \text{其他} \end{cases} \tag{2}$$

其中,$\tau_j(t)$ 表示虚拟机 V_j 在 t 时刻的信息素浓度函数;$\eta_j(t)$ 表示 V_j 在 t 时刻的启发函数;allowed_k 表示虚拟机集合 $\{V_1, V_2, \cdots, V_m\} - \text{tabu}_k$;参数 α,β 分别表示信息素浓度和负载均衡情况的重要程度。

启发函数的初始值 $\eta_j(0)$ 为常数 C,启发函数根据式(3)计算:

$$\eta_j(t) = \left(1 - \frac{\text{VMExeTime}_j - \text{VMAverage}}{\text{VMExeTime}_j + \text{VMAverage}}\right)^2 \quad (3)$$

其中,VMExeTime_j 表示截止到 t 时刻在虚拟机 V_j 上执行任务的总时间;VMAverage 表示根据上次迭代结束时,在当前最优解情况下每台虚拟机平均运行的时间。

根据式(4)计算虚拟机 V_j 上任务运行的总时间。

$$\text{VMExeTime}_j = \frac{T\_\text{TotalLength}}{\text{MIPS}_j} + \frac{\text{InputFileSize}}{\text{Band}_j} \quad (4)$$

其中,T_TotalLength 表示在虚拟机 j 上运行的任务长度之和。

由于云计算资源池中的资源存在异构性和动态性,有些虚拟机的性能优于其他虚拟机。一旦某些虚拟机被分配大量的任务,将会影响任务集的完成时间。因此,在 ACOSA 算法中将虚拟机的负载平衡情况作为启发函数来提高负载均衡能力。虚拟机 j 的启发函数 $\eta_j(t)$ 越大,则虚拟机 j 的综合实力越强,被选择的概率就越大。

3.3　信息素更新规则

根据抽样稳定原则,若满足该原则,则根据式(5)进行信息素的更新。

$$\tau_j(t+1) = (1-\rho) \times \tau_j(t) + \Delta\tau_j(t) \quad (5)$$

其中,ρ 表示信息素挥发系数,ρ 值越大,遗留信息素对当前路径选择的影响越小;$\Delta\tau_j(t)$ 的定义如下:

一个蚁群分配完所有任务后,更新所有访问过的虚拟机上的局部信息素,根据式(6)计算 $\Delta\tau_j(t)$。

$$\Delta\tau_j(t) = \frac{1}{\text{TVM}_j} \quad (6)$$

其中,TVM_j 表示第 i 次迭代时虚拟机 j 上的任务完成时间。

当所有蚁群都完成任务分配时,根据式(7)进行全局信息素更新。

$$\Delta\tau_j(t) = \frac{1}{\text{TVMBest}_j} \quad (7)$$

其中,TVMBest_j 表示在取得全局最优解后,虚拟机 j 运行完成任务的时间。

3.4　Metropolis 准则

通过蚁群算法获得局部最优解,利用模拟退火算法的局部搜索能力,通过置换规则扰动当前局部最优解即随机选择两个任务,如果两个任务对应的虚拟机不同,则进行虚拟机交换。如果交换后任务的完成时间减少,那么就接受新解,否则根据模拟退火的 Metropolis 准则判断是否接受新解。根据式(8)和式(9)得出接受新解的概率 p,若 p 的值小于在当前温度 T 下产生的随机值,则不接受新解,否则接受。

$$\Delta\text{TC} = \text{TC}_k - \text{TC}_{\text{local\_best}} \quad (8)$$

$$p = \begin{cases} \exp\left(-\dfrac{\Delta\text{TC}}{T}\right), \text{如果 } \Delta\text{TC} > 0 \\ 1, \text{其他} \end{cases} \quad (9)$$

其中,TC_k 表示在当前温度 T 下,交换虚拟机后所有任务运行的时间之和;$\text{TC}_{\text{local\_best}}$ 表示在当前温度 T 下,蚁群算法取得虚拟机运行完成所有任务花费的最少时间;ΔTC 表示在当前温度 T 下,交换任务后,虚拟机的运行时间减少的值;p 表示当 ΔTC 大于 0 时,接受新值的概率。

3.5　终止准则

抽样稳定准则对应模拟退火过程中的抽样过程,即在同一温度下,经过连续 M 次干扰局部最优解均保持不变,则认为满足抽样稳定准则;算法终止准则对应模拟退火过程中的退火过程,即当前温度 T 小于 $T_{\min}$,则认为满足算法终止准则,终止算法。

3.6　蚁群模拟退火算法的基本流程

步骤 1:初始化参数。初始化 T_0,α,β,ρ,Q,M 和 $T_{\min}$;根据式(1)初始化信息素浓度。

步骤 2:将蚁群随机分布在虚拟机上,根据式(2)构造候选解。

步骤 3:按照所有任务完成时间最短的原则计算出局部最优解,从而构造局部最优解的邻域,根据式(5)和式(6)进行局部信息素更新。

步骤 4:根据模拟退火算法的置换规则在邻域内构造新解,由式(8)和式(9)判断是否接受新解。

步骤 5:判断当前温度 T 下局部最优值是否满足抽样稳定准则,满足则转步骤 6,否则返回步骤 4。

步骤 6:根据式(5)和式(7)更新信息素。

步骤 7:$T(t+1) = cT(t)$,c 为一常数。判断当前温度 $T(t+1)$ 是否小于 $T_{\min}$,若满足算法终止准则,则输出最优解,算法结束;否则返回步骤 2。

4　仿真实验与性能分析

文中将选择 CloudSim3.0 进行仿真实验。通过云计算仿真平台对调度策略 FCFS、标准蚁群算法(ACO)和 ACOSA 算法进行性能分析。

4.1　仿真参数设置

在 CloudSim 中设置 5 个数据中心,50 个虚拟机资源和 100 ~ 1 000 个任务进行仿真实验。提交到虚拟

机上的任务长度为 1 000 – 20 000MI(Million Instructions)。云仿真实验中的其他参数设置见表 1。标准蚁群算法和 ACOSA 算法的参数设置见表 2。

表 1 CloudSim 参数设置

实体类型	参数	值
任务(Cloudlet)	任务长度	1 000 ~ 2 000
	任务数	100 ~ 1 000
虚拟机	虚拟机总数	50
	MIPS	50 ~ 2 000
	RAM	256 ~ 2 048
	BandWidth	500 ~ 1 000
	Cloudlet Scheduler	Space_Shared
DataCenter	DataCenter 数量	2
	主机数量	10 ~ 20
	VmScheduler	Space_Shared

表 2 ACO 和 ACOSA 算法参数设置

算法	参数及取值
ACO	$\alpha = 0.3$
	$\beta = 1$
	$\rho = 0.4$
	$Q = 100$
	$\mathrm{Iter}_{max} = 100$
ACOSA	$T_0 = 1\ 000$
	$T_{min} = 10$
	$\alpha = 0.3$
	$\beta = 1$
	$\rho = 0.4$
	$Q = 100$
	$M = 50$
	$c = 0.995$

4.2 实验结果与分析

文中提出虚拟机负载不均衡值 DI(Degree of Imbalance)对虚拟机负载均衡情况进行测量,计算如下:

$$T_j = \frac{\mathrm{TotalLength}_j}{\mathrm{MIPS}_j} \tag{10}$$

$$\mathrm{DI} = \frac{T_{max} - T_{min}}{T_{avg}} \tag{11}$$

其中, $\mathrm{TotalLength}_j$ 表示提交到虚拟机 j 上的所有任务长度之和; MIPS_j 表示虚拟机 j 处理指令的能力; T_j 表示虚拟机 j 完成分配任务的运行时间; T_{max} 、T_{min} 和 T_{avg} 分别表示虚拟机 j 运行时间的最大值、最小值和平均值。

ACOSA 算法设计的目标之一是改善负载不均的情况。文中以 DI 作为负载不均的参考值。

实验中将 ACOSA 算法同先来先服务(First Come First Served,FCFS)和标准 ACO 算法进行对比分析。FCFS 算法的目的是为每一任务找到最小完成时间。标准 ACO 算法以减少完成任务的时间为目标。ACOSA 算法根据任务大小和资源分配情况为任务选择合适的资源,该算法不仅减少了任务的完成时间,同时改善了负载不均衡的情况。

为了验证 ACOSA 算法的可行性和有效性,将收敛速度、任务完成时间和资源负载均衡作为评价各个算法性能的标准。

实验 1 中,如图 2 所示,设置 500 个任务比较算法 ACOSA 和 ACO 的收敛速度。

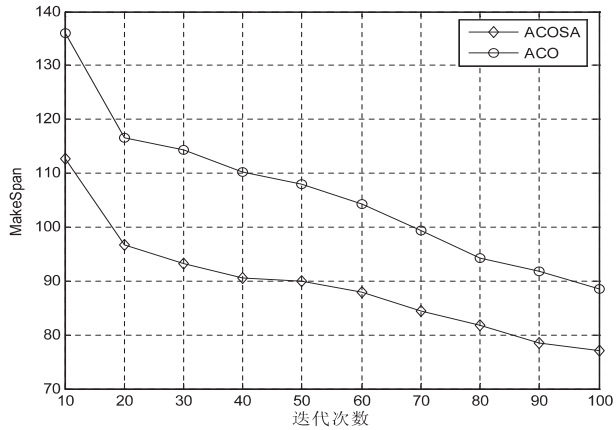

图 2 500 个任务不同迭代次数的完成时间

图 2 表明,随着迭代次数的增加,算法 ACO 和 ACOSA 的任务完成时间逐渐减少,但是对算法 ACO 和 ACOSA 而言,迭代次数大于 60 后任务完成时间减少速度开始变慢,因此,文中将使用迭代次数 60 作为其他实验的参考值。

实验 2 中,比较了不同任务数量的完成时间。图 3 为算法 FCFS、ACO 和 ACOSA 的任务完成时间。

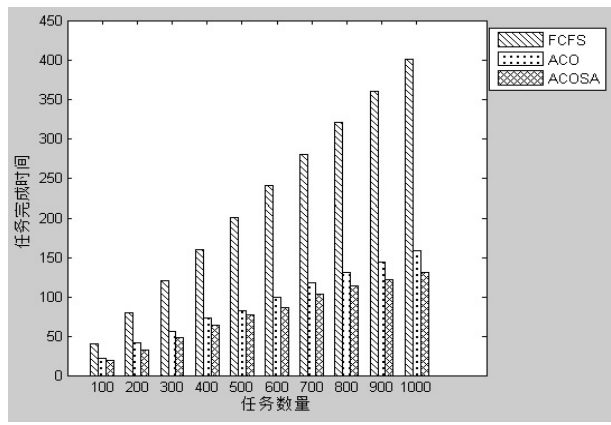

图 3 各算法的不同任务集的完成时间

根据实验结果可知,随着任务数量的增加,ACOSA 算法在任务调度中任务完成时间小于算法 FCFS 和 ACO。

根据实验 2 的结果,计算 DI,结果见图 4。

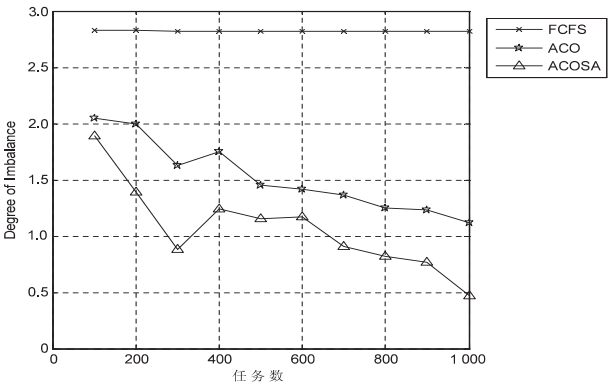

图 4　各算法的 DI 值

从图 3 和图 4 可以看出,ACOSA 算法的任务完成时间比算法 FCFS 和 ACO 分别减少了 50% ~60% 和 15% ~20%,同时虚拟机的负载不均值也明显降低。利用 ACOSA 算法进行云任务调度,可以有效降低任务的完成时间,同时实现虚拟机资源的负载均衡。通过以上对云计算调度任务算法的分析,ACOSA 算法在收敛速度、任务完成时间和负载均衡方面具有更好的优势。图 4 中任务数量较少时,ACOSA 算法的 DI 值较高,这是因为性能好的虚拟机可能会执行 3 ~5 个任务,性能差的虚拟机执行 0 ~2 个任务,造成资源负载的不均衡,此处可以作为下一步改进的方向。

5　结束语

针对云计算中的任务调度问题,提出了一种基于蚁群优化算法的蚁群模拟退火算法。该算法通过引入模拟退火算法提高蚁群算法的收敛速度、加强局部搜索能力和避免算法陷入局部最优解,不仅将减少任务的完成时间作为目标,而且综合考虑了虚拟机的负载均衡情况并将降低虚拟机负载不均衡值作为算法的目标。仿真结果表明,该算法在减少任务的完成时间和计算资源负载均衡等方面明显优于算法 FCFS 和 ACO。

参考文献:

[1] Jadeja Y, Modi K. Cloud computing-concepts, architecture and challenges[C]//International conference on computing, electronics and electrical technologies. [s. l.]: IEEE, 2012: 877-880.

[2] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[3] 李建江,崔　健,王　聃,等. MapReduce 并行编程模型研究综述[J]. 电子学报,2011,39(11):2635-2642.

[4] Stonebraker M, Abadi D, Dewitt D J, et al. MapReduce and parallel DBMSs: friends or foes? [J]. Communications of the ACM, 2010, 53(1): 64-71.

[5] 徐　洁,朱健琛,鲁　珂. 基于双适应度遗传退火的云任务调度算法[J]. 电子科技大学学报,2013,42(6):900-904.

[6] Dorigo M, Birattari M, Stutzle T. Ant colony optimization[J]. IEEE Computational Intelligence Magazine, 2006, 1(4): 28-39.

[7] Liu C Y, Zou C M, Wu P. A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing[C]//13th international symposium on distributed computing and applications to business, engineering and science. [s. l.]: IEEE, 2014: 68-72.

[8] 黄　翰,郝志峰,吴春国,等. 蚁群算法的收敛速度分析[J]. 计算机学报,2007,30(8):1344-1353.

[9] Zhu J, Rui T, Fang H, et al. Simulated annealing ant colony algorithm for QAP[C]//Eighth international conference on natural computation. [s. l.]: IEEE, 2012: 789-793.

[10] 吴　斌,史忠植. 一种基于蚁群算法的 TSP 问题分段求解算法[J]. 计算机学报,2001,24(12):1328-1333.

[11] 陈华根,吴健生,王家林,等. 模拟退火算法机理研究[J]. 同济大学学报:自然科学版,2004,32(6):802-805.

[12] 高　尚. 蚁群算法理论、应用及其与其它算法的混合[D]. 南京:南京理工大学,2005.

[13] 华夏渝,郑　骏,胡文心. 基于云计算环境的蚁群优化计算资源分配算法[J]. 华东师范大学学报:自然科学版,2010(1):127-134.

[14] 高　鹰,谢胜利. 基于模拟退火的粒子群优化算法[J]. 计算机工程与应用,2004,40(1):47-50.

[15] 张　晖,吴　斌,余张国. 引入模拟退火机制的新型遗传算法[J]. 电子科技大学学报,2003,32(1):39-42.