

面向用户任务与云资源匹配度的调度算法研究

聂清彬^{1,2}, 霍敏霞¹, 曹耀钦¹

(1 重庆邮电大学 移通学院, 重庆 合川 401520; 2. 四川大学 锦江学院, 四川 眉山 620860)

摘 要: 为了提高任务在云环境中的执行效率, 提出一种基于任务资源匹配度函数和的成本函数的改进蚁群算法 (an advanced Matching function of Task Resource and Cost in Ant Colony Optimization, MTRCACO), 在综合参考各种最新蚁群算法的基础上, 创新地通过任务与资源匹配度函数来改进信息素中的启发信息, 并通过成本函数降低云计算中心的负载不均衡度, 使虚拟机通过多次算法迭代以后能够处于一种负载均衡的状态, 利用 CloudSim 工具进行仿真测试, 实验结果表明 MTRCACO 算法在任务的执行成本以及系统负载均衡方面均优于 IPSO 算法和 BACO 算法, 提高了资源的利用率。

关键词: 云计算; 匹配函数; 蚁群算法; 任务调度; 负载均衡

中图分类号: TP393

文献标识码: A

文章编号: 1000—7180(2016)07—0093—05

The Research of Scheduling Algorithm Faced to the Matching Degree of User's Task and Cloud Resource

NIE Qing-bin^{1,2}, HUO Min-xia¹, CAO Yao-qin¹

(1 College of Mobile Communication, Chongqing University of Posts and Telecom, Chongqing 401520, China;

2 Jinjiang College, Sichuan University, Meishan 620860, China)

Abstract: To improve the execution efficiency of task in cloud environment, the Matching function of Task Resource and Cost Advanced Ant Colony Optimization (MTRCACO) is proposed. The MTRCACO improves inspiration information from information elements by making a comprehensive reference of the latest ant colony algorithms and adopting the matching function of task and resource. It also decreases the load unbalancing degree in cloud computing center by cost function, and simultaneously keeps the load balance of virtual machines by executing the tasks for many times. Some experiments are carried out on the CloudSim platform. The comparison shows this MTRCACO algorithm is more efficient than the algorithms IPSO and BACO in reducing task execution cost and in keeping system load balance, thus optimizes resource utilization.

Key words: cloud computing; matching function; ant colony algorithm; task allocation; load balancing

1 引言

云计算是通过整合、管理、调配分布在网络各处的计算资源, 并以统一的界面同时向大量用户提供服务。由于云计算中的任务量巨大, 时刻都有海量的任务提交给云计算中心, 因此对于云计算系统而言, 如何采用有效的资源调度算法来合理地分配资源, 让整个云计算资源得到充分利用, 提高执行效率是

研究的重点。

很多学者提出了用改进的蚁群算法来更好地解决云环境下的任务调度^[1-3]。文献[1]提出了一种能改善任务并行性和兼顾任务串行关系的改进蚁群算法 DSFACO, 把提交的任务划分成能互相制约关系的子任务, 根据执行先后次序放到不同优先级的调度队列中, 缩短了任务的延迟时间, 实现了云环境下任务的最优分配。文献[2]提出了改进的粒子群算

法 IP SO. 这些算法都不同程度地优化了云计算中资源调度的过程,但对资源与任务的匹配度鲜有考虑,本文在已有的蚁群算法基础之上针对在云资源调度过程中资源与任务的匹配性差,导致资源分配不合理等缺点,提出基于任务资源匹配度与成本函数的改进蚁群算法(an advanced Matching function of Task Resource and Cost in Ant Colony Optimization MTRCACO)来实现对云计算系统中资源的有效分配.

2 成本函数模型

目前,云计算环境下中任务调度很多都采集用 Google 提出的 Map/Reduce 的分布式编程模式,分成 Map(映射)和 Reduce(化简)两个阶段.在 Map 阶段,将用户提交的任务都划分成一些小的子任务,然后分配到若干个虚拟机节点上并发地执行,执行结束并输出结果.在 Reduce 阶段,将 Map 阶段输出的结果进行汇总处理,并把最终的处理结果交付给用户,在 Map/Reduce 计算编程模式下,如何对分割的子任务进行高效低成本的调度是算法的关键^[4-8].

在云环境中,将 n 个分别独立的子任务分配到 m 个虚拟机上并行执行($m < n$),虚拟机表示为 $vm = \{vm_1, vm_2 \dots vm_m\}$,子任务表示为 $t = \{t_1, t_2 \dots t_n\}$,每个子任务只能在一个虚拟机上运行.可以用矩阵 etc 来表示子任务被分配到虚拟机上的执行时间,那么一个任务就描述为 $(t_i, vm_j, etc(t_i, vm_j))$,用 rcu_j 表示虚拟机 vm_j 单位时间内的运行成本,任务 t_i 在虚拟机 vm_j 上执行成本 $cost_{ij}$ 表示为

$$cost_{ij} = rcu_j \times etc(t_i, vm_j) \quad (1)$$

虚拟机 vm_j 上执行的任务成本之和为

$$cost_j = \sum_{i=1}^n cost_{ij} \quad (2)$$

执行完所有任务云计算中心所需的总成本为

$$cost_{total} = \sum_{j=1}^m cost_j \quad (3)$$

云计算中心的平均成本为

$$cost_{avg} = \frac{cost_{total}}{m} \quad (4)$$

3 任务资源匹配度函数模型

每个任务 t_i 可以用一个六元组来进行表示: $t_i = \{tid_i, tc_i, tb_i, tr_i, tf_i, tl_i\}$,其中 tid_i 表示任务编号, tc_i 表示任务期望资源 cpu 的处理能力, tb_i 表示任务期望带宽, tr_i 表示任务期望的内存, tf_i 表示任务期

望资源的故障率, tl_i 表示任务期望的执行任务延迟率.同理,每个资源节点六元组表示如下:

$$vm_j = \{vid_j, vc_j, vb_j, vr_j, vf_j, vl_j\},$$

式中, vid_j 表示资源节点编号; vc_j 表示资源节点 cpu 处理能力; vb_j 资源节点网络带宽; vr_j 表示资源节点内存大小; vf_j 表示资源节点故障率; vl_j 表示资源节点执行任务延迟率.

通过计算每个任务期望和资源的属性的差值来表示为

$$dc_{ij} = \sqrt{(vc_j - tc_i)^2} \quad (5)$$

$$db_{ij} = \sqrt{(vb_j - tb_i)^2} \quad (6)$$

$$dr_{ij} = \sqrt{(vr_j - tr_i)^2} \quad (7)$$

$$df_{ij} = \sqrt{(vf_j - tf_i)^2} \quad (8)$$

$$dl_{ij} = \sqrt{(vl_j - tl_i)^2} \quad (9)$$

dc_{ij} 表示任务期望 cpu 处理能力值和资源节点的实际 cpu 处理能力值之差.

db_{ij} 表示任务期望网络带宽值和资源节点的实际网络带宽之差.

dr_{ij} 表示任务期望内存值和资源节点的实际内存值之差.

df_{ij} 表示任务期望故障率值和资源节点的实际故障率值之差.

dl_{ij} 表示任务期望资源延迟率和资源节点的实际延迟率之差.

通过以上的差值,引入匹配函数 d_{ij} ,确定它们之间的匹配程度,则任务和资源节点的匹配函数 d_{ij} 表示为

$$d_{ij} = w_1 dc_{ij} + w_2 db_{ij} + w_3 dr_{ij} + w_4 df_{ij} + w_5 dl_{ij} \quad (10)$$

$$w_1 + w_2 + w_3 + w_4 + w_5 = 1.$$

式中, w_1, w_2, w_3, w_4, w_5 分别表示各差值的权重.

匹配函数 d_{ij} 越小,则表示任务选择该节点时,任务所期望的资源节点值和被选择的资源属性值差别越小,匹配度越高,任务在该虚拟机上执行的效率越高,所获的服务质量越高.

4 基于成本改进的蚁群调度算法

本文提出通过成本函数构造出信息素改进因子 PIF 来改进信息素的更新规则,降低系统负载不均衡率,使得在执行同等任务的情况下,任务的执行成本降低,虚拟机的利用率提高.

4.1 虚拟机的选择

结合蚁群算法公式(11) 计算出任务 t_i 被分配

到虚拟机 vm_j 的概率值:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^\alpha [\eta_{ik}(t)]^\beta}, j \in \text{allowed} \\ 0, j \notin \text{allowed} \end{cases} \quad (11)$$

式中, $\tau_{ij}(t)$ 表示在 t 时刻路径 (i, j) 上的信息素浓度; $\eta_{ij}(t)$ 表示在 t 时刻路径 (i, j) 上的启发信息; α 为期望启发因子, 表示蚂蚁在任务调度过程中的启发信息对任务选择的相对重要程度; β 为信息素启发因子, 表示路径上的残留信息素浓度对任务选择的影响程度; $\text{tabu}_k (k = 1, 2, \dots, n)$, 记录蚂蚁已经走过的城市, 被访问过的城市会加入禁忌表; allowed_k 表示第 k 只蚂蚁允许访问的下一个城市的节点, 是禁忌表之外蚂蚁还没走过的城市, 设置蚂蚁的迭代次数为 nc , 最大迭代次数为 $nc_{\max}$.

4.2 初始化信息素和启发信息

在蚁群算法中, 信息素 $\tau_{ij}(t)$ 和启发函数 $\eta_{ij}(t)$ 是最重要的参数, 由于云计算的动态性和异构性, 本文通过虚拟机的计算能力, 网络带宽和执行成本对启发函数和信息素进行初始化:

$$\eta_{ij}(t) = \frac{1}{d_{ij}} \quad (12)$$

$$\tau_j(t) = v_num_j \times v_mips_j + vb_j + vr_j \quad (13)$$

式中, v_num_j 表示虚拟机 vm_j 的处理器数量; v_mips_j 表示虚拟机 vm_j 处理器运算速度; vb_j 表示虚拟机 vm_j 的网络带宽; vr_j 表示虚拟机内存.

则下一跳公式(11) 根据匹配函数 d_{ij} 改进为

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\frac{1}{d_{ij}}(t)]^\beta}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^\alpha [\frac{1}{d_{ik}}(t)]^\beta}, j \in \text{allowed} \\ 0, j \notin \text{allowed} \end{cases} \quad (14)$$

在进行任务分配选择虚拟机过程中, 可以通过轮盘赌算法来决定某一个任务分配到哪个虚拟机上去执行, 首先通过概率转移公式(14) 计算得到任务 t_i 选择到虚拟机 vm_j 上执行的概率, 最后用轮盘赌注法来了确定某一个任务在哪个虚拟机上执行. 每一个任务到虚拟机上的执行概率就等同于赌盘上的每一个扇区, 扇区面积越大, 该扇区被指针选中的概率也就越大, 则任务分配到该虚拟机的转移概率值越大, 更有可能被选中来执行下一个任务.

4.3 信息素的更新

当蚂蚁将任务分配到虚拟机上以后, 随着迭代

次数的增加, 原有的信息素就会随之增加, 如果不对路径上残留的信息素进行更新, 算法将会陷入局部收敛的状态, 因此在每只蚂蚁完成任务分配时, 就利用式(15) 进行信息素更新

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

式中, ρ 是信息挥发因子, 表示信息素在单位时间内的挥发程度, $1-\rho$ 表示信息素浓度残留程度, $\rho \in (0, 1]$, 如果 ρ 越大, 说明信息素挥发越快, 则过去搜索过的路径对现在的选择影响也就越小, $\Delta\tau_{ij}$ 表示信息素浓度的增量, 信息素增量 $\Delta\tau_{ij}$ 用成本函数 cost_{ij} 定义. 改进的信息素更新由局部更新和全局更新组成.

(1) 当一只蚂蚁通过路径搜索, 也就是任务被分配到某一个虚拟机上, 找到了相应的分配方案以后, 路径上所有的虚拟机都将进行一次局部信息素更新, 此时 $\Delta\tau_{ij}$ 定义如下:

$$\Delta\tau_{ij} = \frac{D_1}{\text{cost}_{ij}} \quad (16)$$

式中, D_1 表示常量, cost_{ij} 表示第 i 只蚂蚁在 vm_j 上的执行成本.

(2) 当所有蚂蚁完成路径搜索时, 找到本次迭代中最优的路径, 然后对路径上的所有虚拟机进行全局信息素更新, 此时 $\Delta\tau_{ij}$ 定义如下:

$$\Delta\tau_{ij} = \frac{D_2}{\text{cost}_{\text{total-best}}} \quad (17)$$

式中, $\text{cost}_{\text{total-best}}$ 表示在第本次最优分配方案中为任务搜索到的最低执行成本, D_2 为常量.

4.4 负载均衡度的定义

虚拟机 vm_j 负载不均衡度定义 L_j 如下:

$$L_j = \frac{\text{cost}_j - \text{cost}_{\text{avg}}}{m} \quad (18)$$

式中, cost_j 表示虚拟机 vm_j 上的执行所有任务所需成本, cost_{avg} 表示所有虚拟机执行任务的平均成本.

4.5 信息素改进因子的建立

在实际云环境中, 由于云计算资源存在动态性和异构性, 每个虚拟机的网络带宽、计算能力、内存大小等差异较大, 单位时间内执行任务的成本有较大差别, 有的虚拟机性能比较差, 而有的虚拟机性能比较好. 这样就容易造成大量任务聚集到一部分性能高的虚拟机上执行, 使等待的任务增多, 而那些性能较差的虚拟机资源却处于闲置的状态, 使得整个云计算中的资源利用率很低, 为了解决这个问题, 提出通过改进信息素更新规则, 通过成本函数提高虚拟机的负载均衡度, 改进以后信息素更新规则可

以让闲置的虚拟机分配更多的任务,为分配过多任务的虚拟机减轻负载,改进信息素因子 PIF 定义为

$$PIF = \frac{cost_{ij} - cost_{avg}}{cost_{total}} \tag{19}$$

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \frac{\Delta\tau_{ij}(t)}{PIF} \tag{20}$$

如果虚拟机 vm_j 在之前迭代过程中负载过重,执行同样的任务,执行时间会更长,执行任务的成本 $cost_{ij}$ 就会比其他虚拟机完成该任务的成本更大,则虚拟机 vm_j 的信息素调整因子 PIF 和其他虚拟机因子相比,PIF 更大, $\tau_{ij}(t+1)$ 却变小,根据式(14) 计算任务分配时,虚拟机 vm_j 被分配到任务的概率变小,经过算法多次迭代以后,改进后的算法最终能实现云计算中心虚拟机的负载平衡。

5 算法流程

步骤 1:初始化云计算中所有虚拟机的信息素的值,信息素启发因子 α 和挥发因子 ρ ,期望启发因子 β ,蚂蚁个数 m ,设置迭代次数 $nc = 0$,以及最大迭代次数 nc_{max} 。

步骤 2:将蚂蚁随机地选择任务。

步骤 3:根据式(14)计算每个虚拟机被选中的概率,通过轮盘赌算法确定被最终执行任务的虚拟机。

步骤 4:如果完成本次任务分配,根据式(16)进行局部信息素的更新,否则跳转到步骤 2。

步骤 5:如果所有蚂蚁完成了本次任务的分配,根据分配情况找出本次任务分配的最优的分配方案,根据式(17),通过本次的最优解进行全局信息素更新。

步骤 6:判定当前迭代次数是否达到最大迭代次数,如果是,结束算法流程,否则,跳转到步骤 2 继续执行。

6 仿真实验结果与分析

利用云计算仿真平台 CloudSim3.0 验证算法 RRLB-ACO 的有效性与可行性,利用 CloudSim 里面的 Cloudlet,DataCenter,VM 等类来模拟云系统中真实环境和资源,创建虚拟机以及任务,并重写 Cloudlet,DataCenterBroker 类来构造 MTRCACO 算法,模拟云计算的真实环境,并在基于同样配置和参数设置的条件下与标准的蚁群算法以及文献[3]中的云计算改进任务调度算法 IPSO 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,虚拟机性能参数设置为 500~2 000 MIPS,内存为 512~2 048 MB,带宽为 5 000~10 000 b/s,任务指令长度为 300~3 600 MI(Million Instruction),任务数量为 200~1 000 个任务,虚拟机单位时间运行所消耗成本 $rcu(vm_j)$ 值为 20 \$,最大迭代次数 nc_{max} 为 50 次,在资源中心执行时间共享和空间共享的策略,调度算法中的参数设置如下:

$$\alpha=1, \beta=2, \rho=0.01.$$

CloudSim 仿真步骤如下所示。

步骤 1:初始化 CloudSim 包。

步骤 2:创建数据中心。

步骤 3:创建数据代理中心。

步骤 4:创建虚拟机。

步骤 5:创建云任务。

步骤 6:调用任务调度策略,分配任务到虚拟机。

步骤 7:启动仿真。

步骤 8:仿真结束后分析结果。

试验中在同等条件下将标准的蚁群算法 ACO,文献[3]中的 IPSO 算法和本文的 MTRCACO 算法进行试验仿真,对比任务的执行成本以及虚拟机的负载率状况,具体实验数据如图 1 所示,图中任务执行成本单位为美元(\$),执行时间单位为秒(second)。

从图 1 可以看出,当任务量较少时,IPSO 和 MTRCACO 算法执行成本差别不大,随着任务数量的增加,执行同样多数量的任务,MTRCACO 算法的执行成本最低,说明通过本算法成本函数 $cost_{ij}$ 改进信息素的局部更新和全局更新,可以达到让任务执行成本降低的目的。

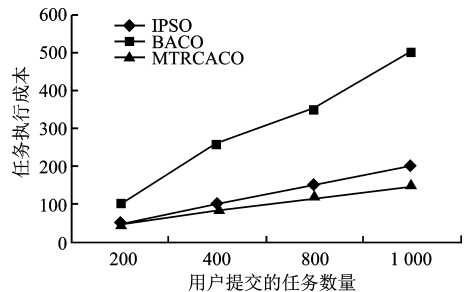


图 1 任务执行成本对比

图 2 反映了虚拟机的负载不均衡度的对比,从图中可以看出,标准的蚁群算法 ACO 负载不均衡度是最高的,说明负载不平衡,IPSO 算法负载有所改进,但是算法改进不够明显,MTRCACO 算法相

比前面两种算法将负载不均衡度降低了 0.5~1.2 左右,可有效降低成本的同时,同时也很好地实现了负载均衡.

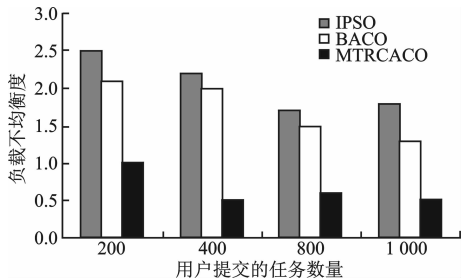


图 2 负载均衡率比较

7 结束语

本文提出了基于任务资源函数匹配度函数与成本函数的改进蚁群优化算法 MTRCACO,算法使得任务合理地分配到云计算系统资源,同时通过信息素因子改进系统负载不均的情况,实验结果表明,MTRCACO 算法性能明显优于 IPSO 和 ACO,有效降低了任务执行成本,提高云计算系统的负载均衡度.

参考文献:

[1] 魏赞,陈元元. 基于改进蚁群算法的云计算任务调度模型[J]. 计算机工程, 2015,41(1):12-16.

[2] 蔡琪,单东红,赵伟艇. 改进粒子群算法的云计算环境资源优化调度[J]. 辽宁工程技术大学学报(自然科学版),2016,35(1):93-96.

[3] 张焕青,张学平,王洲清,等. 基于负载均衡蚁群优化算法的云计算任务调度[J]. 微电子学与计算机,2015,32

(5):31-40.

[4] Ramesh D, Krishnan A. Optimal parameter identification in ant colony optimization for load balancing in grid computing [J]. European Journal of Scientific Research,2012,75 (3):370-376.

[5] Zheng Z N, Wang R. An approach for cloud resource scheduling based on parallel genetic algorithm [C] // In Proc Of 3 rd International Conference on Computer Research and Development. Shanghai: IEEE, 2011: 444-447.

[6] Brototi Mondal, KOUSIK Dasgupta, Paramartha Dutta, Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft computing Approach [C] // International Conference on Computer Communication, Control and Information Tedsnology. [s. l.]: IEEE, 2012:783-789.

[7] Cristian Mateos, Elina Pacini, Carlos Garcia garion. An ACO-inspired algorithm for minimizing weighted flow time in cloud-based parameter sweep experiments [J]. Advances in Engineering Software , 2013, 56 (11):38-50

[8] Zhan S B, Huo H Y. Improved PSO-based task scheduling algorithm in cloud computing [J]. Information & Computational Science, 2013,13 (9): 3821-3829.

作者简介:

聂清彬 男,(1982-),硕士,讲师. 研究方向为云计算与物联网. E-mail:270104318@qq. com.

霍敏霞 女,(1983-),硕士,讲师. 研究方向为软件测试.

曹耀钦 男,(1962-),博士,教授,博士生导师. 研究方向为计算机网络.

(上接第 92 页)

[3] Savage S,Burrows Michael,Nelson Greg,et al. Eraser: a dynamic data race detector for multithreadel programs[J]. Acm Sigops Operation Systems Review, 1997,15(4):391-394.

[4] Prodromos Gerakios, Papaspyrou N,Sogonas K, et al. Race-free and Memory-safe Multithreading:Design and Implementation in Cyclone[R]. TLDP Int. 2010.

[5] Amdahl GM. Validity of the single-processor approach to achieving large scale computing capabilities[C] // Proceedings of the American Federationof Information Processing Societies. New York, N Y, USA: ACM,

1967;483-485.

作者简介:

沙 晗 女,(1991-),硕士研究生. 研究方向为并行程序、动态编译. E-mail:18813148088@163. com.

应 欢 女,(1988-),博士研究生. 研究方向为并行程序、动态编译.

唐博文 男,(1991-),硕士研究生. 研究方向为动态编译、程序分析、并行程序.

乔 扬 男,(1993-),硕士研究生. 研究方向为软件安全.