

云计算环境下基于改进 GSO 的目标主机选择算法

陈伟,程家超,张超

(宿州职业技术学院计算机系,安徽 宿州 234101)

摘要:目标主机的选择是虚拟机动态迁移过程中的重要阶段,是实现负载均衡的关键。针对基本萤火虫算法存在的精度不高、收敛较慢的问题,提出了一种改进的萤火虫优化算法,用于解决虚拟机迁移时虚拟机和目标物理主机的映射问题,实现多目标最优求解。该算法通过引入步长调整因子,能够动态调整移动步长,克服了步长过大或过小导致的精度不高、后期收敛较慢的缺点。全面考虑物理主机负载指标,建立负载均衡模型,将萤火虫算法中个体与节点资源相对应,利用萤火虫发光机制寻优求解,以实现目标主机的优化选择。仿真实验表明,该算法能够快速完成目标主机的选择,有效平衡系统资源,实现数据中心负载均衡。

关键词:云计算;萤火虫群优化算法;动态步长;负载均衡;目标主机选择;虚拟机

中图分类号:TP301.6

文献标志码:A

引言

云计算^[1]是集分布式计算、虚拟化技术、网格计算和 Web 服务等技术发展起来的一种综合技术^[2],它为用户提供了基础设施、平台以及软件的服务,并且通过互联网将服务按需的提供给用户^[3]。虚拟化技术^[4]是云计算中的关键技术,可以把硬件资源如服务器、网络、内存及存储等进行虚拟化,在物理机上批量创建虚拟机来扩展基础设施层的云资源池,同时可以通过虚拟机的动态迁移技术来实现云数据中心的负载均衡。虚拟机的动态迁移,就是把虚拟机从高负载物理节点迁移至低负载物理节点,实现负载均衡,以提高资源利用率且节能降耗^[5]。虚拟机迁移过程中,如何选择目标主机来接

收迁出的虚拟机是本文的研究所在。目标主机选择是一个 NP 问题,目前大都采用一些启发式的算法进行求解。文献[6]中使用最佳适应算法通过向量比较方式寻找迁移后不会超出负载阈值上限的目标主机,实现虚拟机的调度;文献[7]通过蚁群算法进行目标求解,资源利用率相对提高,但算法收敛速度过慢;文献[8]采用遗传算法进行虚拟机的放置,实现物理机负载均衡;文献[9]引入粒子群优化算法并进行改进,将不满足迁移的物理机加入到规避列表,以避免迁移后资源占用率过高;文献[10]提出了一种多目标蚁群优化的虚拟机放置算法,主要考虑数据中心网络流量的优化和物理机资源浪费;文献[11]提出了数据中心负载不均衡度以及物理机与虚拟机之间不匹配度的概念,利用蚁群算法实现较低负

收稿日期:2017-08-04

基金项目:安徽省高校自然科学研究重点项目(KJ2016A778;KJ2016A781);安徽省高校优秀青年人才支持计划重点项目(gxyqZD2016586);安徽省质量工程项目(2016jyxm1039)

作者简介:陈伟(1982-),男,安徽阜阳人,讲师,硕士,主要从事计算机网络方面的研究,(E-mail)chenwei9737@163.com

载不均衡度;文献[12]采用权值适应度粒子群优化(WFPSO)算法,能更好地降低应用的执行时间,但未考虑负载均衡的情况。

本文针对虚拟机迁移中目标主机选择问题,引入人工萤火虫群优化(Glowworm Swarm Optimization, GSO)算法,并进行改进,从负载均衡角度出发,对目标主机寻优问题进行求解,并通过实验对该算法有效性进行验证。

1 云计算负载均衡模型

云计算环境下,当某物理主机上负载过高时可以通过迁出虚拟机的方式减轻负载,本文的算法是让迁出的虚拟机分配给负载值小的目标主机,避免加重高载主机的负担,从而提高系统运行效率和资源利用率。

云计算中单个节点的负载指标^[13]主要包括CPU利用率 C_i 、内存利用率 M_i 、磁盘利用率 D_i 和带宽利用率 B_i 。设 C_i 、 M_i 、 D_i 、 B_i 的权重分别为 α_1 、 α_2 、 α_3 、 α_4 ,则节点 i 的综合负载为:

$$L(i) = \alpha_1 C_i + \alpha_2 M_i + \alpha_3 D_i + \alpha_4 B_i \quad (1)$$

其中: $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$, C_i 、 M_i 、 D_i 、 B_i 的值分别为:

$$C_i = \frac{\sum_{j=1}^n V_j^{cpu}}{H_i^{cpu}} \quad (2)$$

$$M_i = \frac{\sum_{j=1}^n V_j^{mem}}{H_i^{mem}} \quad (3)$$

$$D_i = \frac{\sum_{j=1}^n V_j^{disk}}{H_i^{disk}} \quad (4)$$

$$B_i = \frac{\sum_{j=1}^n V_j^{bw}}{H_i^{bw}} \quad (5)$$

其中: j 表示物理节点 i 上第 j 个虚拟机; $\sum_{j=1}^n V_j^{cpu}$ 、 $\sum_{j=1}^n V_j^{mem}$ 、 $\sum_{j=1}^n V_j^{disk}$ 、 $\sum_{j=1}^n V_j^{bw}$ 分别表示物理节点 i 上虚拟机使用的CPU、内存、磁盘、带宽之和; H_i^{cpu} 、 H_i^{mem} 、 H_i^{disk} 、 H_i^{bw} 分别表示节点 i 上CPU、内存、磁盘、带宽总量。

云数据中心中物理机集群的平均负载为各物理主机负载的平均值,表示为:

$$\bar{L} = \frac{\sum_{i=1}^n L(i) \times q_i}{n} \quad (6)$$

由此可以计算出物理机负载信息值的标准差,衡量整个云数据中心的总体负载均衡情况,并将此标准差作为适应度函数。

$$S = \sqrt{\frac{1}{n} \sum_{i=1}^n (L(i) - \bar{L})^2} \quad (7)$$

由 S 的表达式可以看出,标准差越小表明数据中心中各个物理机负载差异较小,整个云计算环境负载效果较好^[14],反之,则越差。虚拟机迁移中目标主机的选择就是保证迁入后能让整体负载达到最优, S 的值达到最小。

2 GSO 算法

GSO算法是印度学者K. N. Krishnanand和D. Ghose于2005年提出的一种新型群智能优化算法^[15]。算法的思想源于模拟萤火虫在晚上群聚活动的自然现象而提出的,萤火虫通过散发荧光素与同伴进行信息传递告知食物源,荧光素越高越亮的萤火虫其对同伴的吸引力和号召力就越强,最终大量的萤火虫聚集在较亮的萤火虫周围,也即是食物丰富的地方。

在GSO算法中,每只萤火虫都被看作是解空间的一个解,每个萤火虫都被随机分布在目标函数的定义空间中,萤火虫各自拥有自己的荧光素和视线范围。每个萤火虫荧光素的亮度和自己所在位置对应的目标函数的适应度值有关。荧光素越亮的萤火虫所在位置视为越好,对应的目标函数值越优。GSO算法中萤火虫寻优的移动方式:萤火虫在自己的视野范围内寻找邻域内荧光较亮的萤火虫并向其移动靠拢,萤火虫的视野即决策域半径会随邻域中萤火虫密度发生变化,当密度较小时会加大决策半径以便可以找到更多的萤火虫,反之会减小决策半径。最终,通过不断地位置更新,使大部分萤火虫聚集在较优的位置从而实现目标的优化。

2.1 荧光素更新

每只萤火虫都有自己的荧光素,通过荧光素的亮光来吸引同伴,荧光素的更新公式为:

$$l_i(t+1) = (1-\rho)l_i(t) + \gamma J(x_i(t+1)) \quad (8)$$

其中: $l_i(t)$ 为第 t 代第 i 只萤火虫的荧光素值; $x_i(t+1)$

表示第 $t+1$ 代萤火虫的位置; $J(x_i(t+1))$ 表示萤火虫所在位置对应的目标函数的适应度值; ρ 和 γ 分别表示萤火虫的消失速率和更新速率。

2.2 决策域半径更新

决策域半径表示萤火虫的视野范围,表示算法的搜索空间,在视野范围内搜索其他萤火虫。决策域半径是动态变化的,会随着搜索空间内萤火虫的数目多少而加大或减小,以确保视野范围内的萤火虫数量,决策域半径得更新公式为:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_i - |N_i(t)|)\}\} \quad (9)$$

其中: $r_d^i(t+1)$ 为 $t+1$ 代第 i 只萤火虫的决策域,即萤火虫的视野范围; r_s 为萤火虫个体的感知域范围; β 为决策域更新率; n_i 为邻域个体数的阈值; $N_i(t)$ 为邻域集合,由荧光素值大于自己的萤火虫个体构成。

$$N_i(t) = \{j: x_j(t) - x_i(t) < r_d^i(t); l_j(t) < l_i(t)\} \quad (10)$$

2.3 确定萤火虫移动方向

萤火虫 i 的移动方向受邻域内荧光素值大的萤火虫影响,荧光素值越大吸引力就越大,相应地,萤火虫选择向其移动的概率就越大。假设萤火虫向概率最大的萤火虫 j 移动,则有:

$$j = \max(p_i) \quad (11)$$

其中: $p_i = (p_{i1}, p_{i2}, \dots, p_{iN_i(t)})$; 萤火虫 i 向 j 移动的选择概率为:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)} \quad (12)$$

2.4 萤火虫位置更新

萤火虫向选择概率大的萤火虫移动后,新的位置表示为:

$$x_i(t+1) = x_i(t) + s_t \frac{x_j(t) - x_i(t)}{x_j(t) - x_i(t)} \quad (13)$$

3 改进 GSO 算法在目标主机选择中应用

3.1 算法应用分析

云计算环境下,在确保云数据中心负载均衡前提下,选择目标主机进行虚拟机迁移,解决虚拟机和物理机的映射问题,把虚拟机分配到对应的物理主机上,此

选择过程即是寻优的过程,可以借鉴 GSO 算法解空间寻优方式求寻找最适合的目标主机,即求算法的最优解。把虚拟机与萤火虫个体进行对应,把虚拟机迁移过程中需要迁移的虚拟机看作是新加入觅食的萤火虫,把物理机集群看作是萤火虫的聚集群,新加入的萤火虫可以不断地移动变化位置,直至寻找到最优位置。寻优过程需满足云数据中心负载标准差越小, $J(x_i(t+1))$ 的值越大,由此得出,萤火虫荧光素越亮,位置越好,负载标准差也越小,萤火虫的位置即为需要选择的目标主机。

3.2 改进 GSO 算法

3.2.1 GSO 算法存在问题

GSO 算法具有较强的搜索能力,操作方便,实现起来比较简单,但是后期收敛速度慢和求解精度不高等缺陷依然存在。GSO 算法采用固定移动步长,搜索初期,如果步长较小,则易陷入局部极值,降低收敛速度;搜索后期,萤火虫个体和峰值的距离也越来越近,此时如果步长较大,超过这个距离则会跳过全局最优解,造成峰值附近的震荡,导致精度降低。由此可见,在 GSO 算法采用固定步长对算法的精度和收敛速度都会有一定的影响,不适合实际需求,因此,要把算法进行改进,使得算法在不同的搜索时期,能够自动调整步长的大小,以满足求解的需要。

3.2.2 步长动态调整的 GSO 算法

针对 GSO 算法在收敛速度和求解精度上存在的问题,对 GSO 算法进行了改进,使其步长能够随着迭代次数进行动态调整。迭代初期,个体与最优解距离较远,设置较大的步长值,有利于全局搜索且易跳出局部极值,提高收敛速度;迭代后期,个体与最优解越来越靠近,逐步减少步长值对局部搜索有利,防止跳过最优解,提高算法精度^[16]。为此,引入式(14)和(15)对步长进行动态调整。

$$s(t) = c \times s_0 \quad (14)$$

$$c = \frac{t_{\max} - iter}{t_{\max}} \left(\ln \frac{\|x_i(t) - x_j(t)\| + s_{\min}}{r_d^i(t) + s_{\max}} \right) \quad (15)$$

其中: s_0 为步长初始值; c 为步长调整因子,其值随着迭代次数的增加而适当减小,随着实际位置的变化进行动态调整,初始阶段个体移动寻优时离高亮度萤火虫实际

距离越大,则步长越大; t_{max} 为最大迭代次数; s_{max} 和 s_{min} 分别表示移动步长的最大值和最小值。由式(13)~式(15)可得出改进后的萤火虫位置更新公式:

$$x_i(t+1) = x_i(t) + \frac{t_{max} - iter}{t_{max}} \left(\ln \frac{\|x_i(t) - x_j(t)\| + s_{min}}{r_d^i(t) + s_{max}} \right) \cdot s_0 \frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|} \quad (16)$$

改进后的算法步长会随着迭代次数和实际位置的变化而自适应调整,不会因为步长太大导致跳过最优解,也不会因为寻优初期阶段步长太小而影响收敛速度,因此,改进后算法在搜索精度和收敛速度能够得到一定的改善。

3.3 目标主机选择算法实现流程

算法实现步骤描述如下:

(1) 初始化。初始化参数 $n, \rho, \gamma, \beta, n_i, s_0, r_i, l_0, s_{max}, s_{min}, t_{max}$, 并随机产生 n 只萤火虫, 定义其初始位置。

(2) 计算荧光素。根据式(8)更新各萤火虫的荧光素值, 将其作为目标函数, 并需满足云数据中心负载标准差即式(7)的值越小, $J(x_i(t+1))$ 的值越大。

(3) 寻找邻域 $N_i(t)$ 。在决策域内寻找荧光值高于自己的个体构成邻域 $N_i(t)$ 。

(4) 确定移动方向。按照式(12)计算选择概率, 确定萤火虫移动方向。

(5) 更新萤火虫位置。根据式(14)和(15)对移动步长进行调整, 然后根据式(16)进行萤火虫位置的更新。

(6) 更新决策域。根据式(9)更新决策域半径。

(7) 判断迭代次数是否超过最大值, 未达到则转至步骤(2), 否则转步骤(8)。

(8) 输出最优解。

由于萤火虫荧光素越高吸引力越大, 使得萤火虫在移动过程中趋向于荧光素亮的萤火虫, 因此, 把负载较小的物理主机看作是荧光素值较高的萤火虫, 从而吸引其他萤火虫向其移动, 实现虚拟机往低负载目标主机迁移。另外把式(7)作为适应度函数, 保证了虚拟机在迁移到目标主机后云数据中心整体的负载标准差较小, 实现集群负载均衡。

4 仿真实验

本文采用仿真实验工具 CloudSim^[17] 模拟云计算环境, 采用 Java 语言编程实现本文算法进行仿真实验, 观察虚拟机迁移后云中心负载均衡及资源使用率情况。

4.1 负载均衡测试

在云数据中心模拟 60 台物理主机, 每台物理机上随机部署虚拟机, 并通过随机方式改变虚拟机的资源使用量, 以此模拟负载的变化。

图 1 展示了利用本文算法前后云数据中心物理机资源使用率对比情况。h1 ~ h60 表示物理机节点, 未调整之前 CPU、内存、带宽使用率差异较大, 数据中心负载严重失衡, 存在很多高负载状态的物理机。利用本文算法调整之后, 物理机的 CPU、内存、带宽使用率都趋于正常, 维持在 60% ~ 70% 左右, 物理机负载比较均衡。

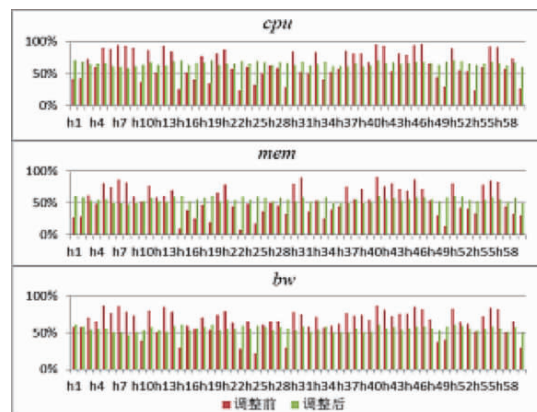


图 1 使用本文算法调整前后资源使用率对比

采用式(7)物理主机负载标准差作为负载均衡度来衡量数据中心的负载均衡情况。负载标准差越小, 整体负载均衡效果越好。图 2 展示了几种基于虚拟机迁移算法的负载均衡情况对比。最优负载调度策略 (Optimality Based Schedule, OBS) 基于最优负载, 负载较轻的目标主机会瞬间急剧升高, 导致产生群聚效应, 负载均衡度效果不佳。文献[5]的算法采用负载预测进行迁移触发, 使用加权概率进行目标主机选择, 对负载均衡有一定的改善。本文在文献[5]的基础上, 对目标节点选择采用改进 GSO 算法进行解决, 使个体在移动时保持负载标准值最小, 从而有效避免了群聚效应, 负载均衡度维持较低值, 负载均衡表现非常出色。

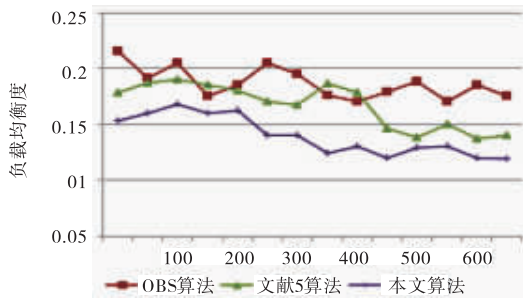


图 2 负载均衡对比

4.2 目标主机选择

选取不同时刻对云数据中心发出虚拟机迁移请求,对本文改进的 GSO 算法和基本 GSO 算法进行性能比较。不同时刻提交的虚拟机迁移请求个数见表 1。

表 1 不同时刻迁移数

时刻	迁移数
t1	18
t2	13
t3	15
t4	20
t5	17

不同时刻两种算法在发出迁移请求到选择目标主机的运行时间如图 3 所示。可见,在处理相同虚拟机迁移请求个数时,对目标主机的寻优过程中,本文改进的 GSO 算法所用时间明显低于基本的 GSO 算法,能够快速定位目标主机,使整体负载值标准差最小,数据中心负载均衡情况良好。

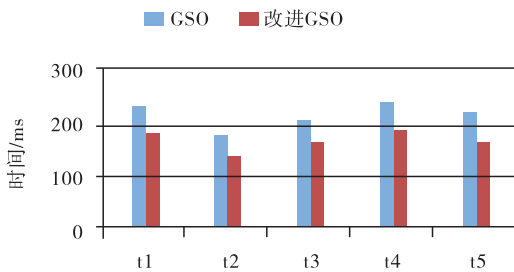


图 3 不同时刻迁移时间对比

5 结束语

负载均衡问题是云计算中的关键问题。通过虚拟机迁移方式解决云数据中心负载均衡问题是本文的研究基础。本文针对虚拟机迁移过程中目标主机选择问

题,引入了萤火虫优化算法,并对其进行改进,通过动态调整步长方式,完善和克服了精度低、收敛速度慢等问题,有效实现了目标主机的最优求解。仿真实验表明,本文算法能够快速选择目标主机,平衡各物理机负载差异,实现云计算环境下负载均衡。

参考文献:

[1] VAQUERO L M,RODERO-MERINO L,CACERES J, et al.A break in the clouds:towards a cloud definition [J].Acm Sigcomm Computer Communication Review, 2008,39(1):50-55.

[2] 李乔,郑啸.云计算研究现状综述[J].计算机科学, 2011,38(4):32-37.

[3] 韩德志,李楠楠,毕坤.云环境下的虚拟化技术探析 [J].华中科技大学学报:自然科学版,2012,40(S1): 262-265.

[4] 董运萌.一种云计算环境下负载均衡敏感的聚类部署方法研究[D].长春:吉林大学,2015.

[5] 陈伟,张超.基于负载趋势预测的虚拟机动态迁移策略研究[J].绥化学院学报,2016,36(12):145-149.

[6] 周文煜,陈华平,杨寿保,等.基于虚拟机迁移的虚拟机集群资源调度[J].华中科技大学学报:自然科学版,2011,39(S1):130-133.

[7] GAO Y,GUAN H,QI Z,et al.A multi-objective ant colony system algorithm for virtual machine placement in cloud computing [J].Journal of Computer & System Sciences,2013,79(8):1230-1242.

[8] HU J,GU J,SUN G,et al.A scheduling strategy on load balancing of virtual machine resources in cloud computing environment[C]//Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming, Dalian, China, December 18-20,2010:89-96.

[9] 武兴宇,孙磊,胡翠云,等.基于改进粒子群优化算法的虚拟机迁移选择策略研究[J].计算机科学,2015, 42(S1):20-23.

[10] 赵君,马中,刘驰,等.一种多目标蚁群优化的虚拟机

- 放置算法[J].西安电子科技大学学报:自然科学版,2015,42(3):173-178+185.
- [11] 孟凡超,张海洲,初佃辉.基于蚁群优化算法的云计算资源负载均衡研究[J].华中科技大学学报:自然科学版,2013,41(S2):57-62.
- [12] 何利文,袁野,王延松,等.基于WFPSO算法的云虚拟机放置策略[J].计算机应用研究,2017,34(2):591-594.
- [13] 顾桓瑜,石磊,郭俊廷.基于改进萤火虫算法的云计算负载均衡研究[J].大连交通大学学报,2015,36(6):107-110.
- [14] 宁彬,谷琼,吴钊,等.云计算环境下的混沌萤火虫的资源负载均衡算法[J].计算机应用研究,2014,31(11):3397-3400.
- [15] 赵光伟.人工萤火虫群优化算法改进与应用研究[D].南宁:广西民族大学,2012.
- [16] 顾忠伟,徐福缘.一种新颖的萤火虫算法求解PID控制器参数自整定问题[J].系统管理学报,2017,26(1):101-106.
- [17] CALHEIROS R N,RANJAN R,BELOGLAZOV A,et al.CloudSim:a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms[J].Software Practice & Experience,2011,41(1):23-50.

A Selection Algorithm of Destination Host Based on Improved Glowworm Swarm Optimization Algorithm in Cloud Computing Environment

CHEN Wei, CHENG Jiachao, ZHANG Chao

(Department of Computer Information, Suzhou Vocational and Technical College, Suzhou 234101, China)

Abstract: The selection of destination host is an important stage in the dynamic migration of virtual machines, and is the key to realize load balancing. In order to overcome the shortcomings of basic glowworm swarm optimization algorithm including low accuracy and slow convergence speed, an improved glowworm swarm optimization algorithm is proposed to solve the mapping problem between the virtual machine and the target physical host when the virtual machine is migrated, and the multi-objective optimal solution is realized. By introducing the step adjustment factor, the algorithm can dynamically adjust the moving step and overcome the shortcomings of low accuracy and slow convergence speed caused by too large step or too small step. Considering the physical load index, the load balancing model is established, and the individual and node resources in the improved glowworm swarm optimization algorithm are matched with each other, and the optimal selection of the destination host is realized by using the luminous mechanism of fireflies. The simulation experiment shows that the improved algorithm can select the destination host quickly, balance the system resource effectively and realize the load balancing of data center.

Key words: cloud computing; glowworm swarm optimization algorithm; dynamic step; load balancing; destination host selection; virtual machine