

基于改进蚁群算法的服务器集群资源调度研究

刘万军, 王晓宇, 曲海成, 孟 煜

(辽宁工程技术大学 软件学院, 辽宁 葫芦岛 125105)

摘 要: 提出一种基于蚁群算法的服务器集群资源调度算法. 该算法引入等待因子的概念, 采用等待因子动态选取服务器资源, 提高了集群系统的整体性能. 仿真实验结果表明, 改进的蚁群算法优于其他负载均衡算法, 使服务器集群系统负载更加均衡, 整个集群系统的资源得到了充分的利用.

关键词: 服务器集群; 蚁群算法; 负载均衡; 资源调度

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2016)03-0098-04

Research on Server Cluster Resource Scheduling Based on Improved Ant Colony Algorithm

LIU Wan-jun, WANG Xiao-yu, QU Hai-cheng, MENG Yu

(Software Engineering Institute, Liaoning Technical University, Huludao 125105, China)

Abstract: In a server cluster, put forward a kind of ant colony algorithm of server cluster resource scheduling algorithm based on. The algorithm introduced waiting factor concept, awaiting the adoption of dynamic factor selected server resources and improve the overall performance of the cluster system. The simulation results show that the improved ant colony algorithm is superior to other load balancing algorithms to enable the server cluster system load more balanced, the resources of the cluster system has been fully utilized.

Key words: server cluster; ant colony algorithm; load balancing; resource scheduling

1 引言

随着互联网技术迅速发展, Web 服务器产生了大量的信息, 单个服务器的能力难以满足更高的数据量和访问量. 服务器集群^[1]是把多个服务器组合起来, 用来完成之前单个服务器不能完成的任务, 这样就形成一个高性能的服务器. 而且, 服务器集群成本低、可扩展性好, 可以很好地解决这类问题. 在服务器集群系统中, 各服务器负载能力差异较大, 选择好的负载均衡算法^[2]对提高集群系统资源利用率有着重要的意义.

目前负载均衡算法分为静态负载均衡算法^[3]和动态负载均衡算法两类. 静态负载均衡算法包括轮转算法、加权轮转算法等. 这类算法, 由于没有考

虑集群中节点负载信息的动态变化, 所以会导致服务器负载不均衡, 效率低, 请求任务不能合理分配等问题. 动态负载均衡算法^[4]包括最小连接算法、加权最少连接算法等. 这类算法由于不能全面地反映出当前服务器节点的状态, 所以对规模较小的集群系统较为适合, 而对规模较大的集群系统的负载均衡效果则不太明显. 蚁群算法^[5]是一种自适应算法, 它的动态更新性能可以适用于大规模的服务器集群系统.

本文针对服务器集群系统的特点, 为了提高集群系统的资源利用率, 构建出集群的资源调度^[6]策略, 并且提出一种改进的蚁群算法.

2 服务器集群的资源调度策略及算法分析

2.1 集群的资源调度策略

服务器集群的资源调度策略实质是解决任务与服务器的匹配问题. 集群系统中使用一种合理的分配算法把客户的请求任务均衡地分配到每个服务器上, 缩短客户的等待时间, 有效地提高集群系统的整体处理能力. 服务器集群系统的调度结构如图 1 所示.

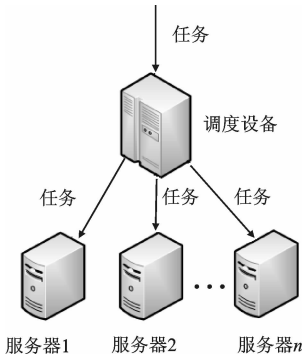


图 1 集群的调度结构

2.2 基于 ACO 的集群资源调度算法分析

蚁群算法(ACO)是一种启发式算法, 由意大利学者 M. Dorigo 等人提出. 自然界中的蚂蚁在觅食的过程中, 在所经过的道路上释放一种叫做信息素的化学物质, 并且蚂蚁会以较大的概率选择信息素浓度较高的路径. 蚁群算法以其协同工作机制、全局优化、并行性、鲁棒性等特点, 可以很好地用来解决 NP-hard 问题.

服务器集群系统在执行任务前, 要初始化以下相关参数: 服务器节点 j , 节点数 n , CPU 的利用率 p_{cpu} , CPU 的处理速度 v_{cpu} , 通信带宽 B . 在 t 时刻, 对服务器集群系统资源的信息素和启发式因子初始化为

$$\tau_j(t) = \sum_{j=1}^n (1 - p_{\text{cpu}}(t)) v_{\text{cpu}} + B \quad (1)$$

$$\eta_j(t) = \tau_j(0) \quad (2)$$

在服务器集群系统中, 一只蚂蚁作为一个任务, 蚂蚁根据自身对计算能力的需求和任务的大小依照相应的准则去选取服务器节点. 任务 i 选择服务器资源节点 j 的概率定义为

$$P_{jk}^i = \begin{cases} \frac{\tau_j^\alpha(t) \eta_j^\beta}{\sum_k \tau_k^\alpha(t) \eta_k^\beta}, j, k \in S \\ 0, j, k \notin S \end{cases} \quad (3)$$

式中, $\tau_j(t)$ 为 t 时刻服务器资源节点 j 的信息素浓度, η_j 为服务器资源节点 j 的启发因子, S 为服务器资源的集合, α 和 β 分别为信息素浓度和启发因子的

重要程度.

已知蚂蚁释放在路径上的信息素浓度会随着时间的推移自然挥发或变浓. 因此, 蚁群算法中信息素会进行相应的更新, 更新的准则定义为

$$\begin{cases} \tau_j(t+1) = (1 - \rho) \tau_j(t) + \Delta \tau_j \\ \Delta \tau_j = \sum_{k=1}^n \Delta \tau_j^k \end{cases} \quad (4)$$

$$\Delta \tau_j^k = \begin{cases} Q, & \text{第 } k \text{ 只蚂蚁经过节点 } j \\ 0, & \text{其他} \end{cases} \quad (5)$$

式中, ρ ($0 < \rho < 1$) 表示信息素的挥发程度, $\Delta \tau_j^k$ 表示第 k 只蚂蚁在本次循环中在节点 j 上释放的信息素浓度, $\Delta \tau_j$ 表示在本次任务调度中节点 j 上信息素浓度的增量, Q 表示蚂蚁循环一周后释放在所有经过节点上的信息素总量.

3 改进的蚁群算法

蚁群算法自身具有很强的鲁棒性和并行性. 除此之外, 蚁群算法具有正反馈的特点, 且有很强的自适应性, 这样可以加快服务器集群调度的求解速度, 由于算法对系统的初始条件要求不太高, 使得算法可以很好地适用于求解服务器集群的资源调度问题.

改进的蚁群算法中, 引入一个等待因子 λ , 它表示一个服务器节点要进入闲置状态所用的等待时间. λ 可以动态地掌握当前服务器资源的负载情况, 有利于选取最合适的服务器资源.

(1) 信息素初始化

在 t 时刻, 服务器节点的信息素初始化定义为

$$\tau_j(t) = 1/\lambda_j \quad (6)$$

式中, λ_j 是服务器节点 j 的等待因子, 其初始值 $\lambda_j = a$, 其中 a 是 $0 < a < 1$ 的常量, n 是服务器节点数.

启发因子定义为

$$\eta_j(t) = \tau_j(0) \quad (7)$$

(2) 选择服务器节点

与传统的蚁群算法类似, 蚂蚁通过计算概率来动态地选取当前集群系统中最优的服务器节点. t 时刻, 任务 i 选择服务器节点 j 的概率用 P_{ij}^t 来表示.

$$P_{ij}^t = \begin{cases} \frac{\tau_j^\alpha(t) \eta_j^\beta}{\sum_k \tau_k^\alpha(t) \eta_k^\beta}, j, k \in S \\ 0, j, k \notin S \end{cases} \quad (8)$$

式中, $\tau_j(t)$ 为 t 时刻服务器资源节点 j 的信息素浓度, η_j 为服务器资源节点 j 的启发因子, S 为服务器资源的集合, α 和 β 分别为信息素浓度和启发因子

的重要程度。

(3) 更新信息素

当任务和服务节点匹配结束之后,服务器资源的信息素浓度会更新,更新公式为

$$\lambda_j^{\text{new}} = \lambda_j^{\text{old}} + T_{ij} \quad (9)$$

$$\tau_j(t) = (1 - \rho) / \lambda_j^{\text{new}} \quad (10)$$

式中, ρ 是信息素的挥发程度, T_{ij} 是任务 i 在服务器节点 j 上的完成时间。

在服务器集群的资源调度中,每个任务作为一只蚂蚁,改进蚁群算法的流程图如图2所示。

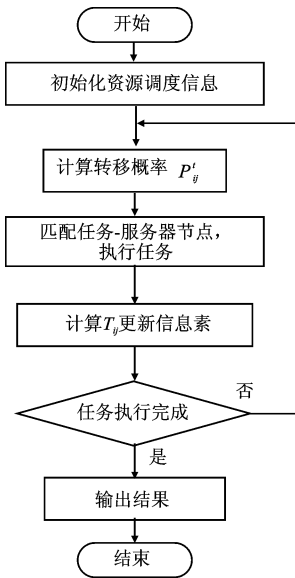


图2 蚁群算法的流程

4 实验及结果分析

仿真测试用 Gridsim 来模拟一个大规模的服务器集群研究系统。在 Gridsim 平台上,采用任务请求的响应时间、CPU 的利用率和系统吞吐率作为算法性能评估指标,分别进行了改进的蚁群算法、蚁群算法、Min-min 算法、Max-min 算法、Random 算法的对比试验。经大量实验表明在蚁群算法中,参数 ρ 取 0.2, α 和 β 都取 0.5,较为适合。

实验1:任务请求的响应时间。

实验中选取服务器节点数为 50,用户请求的任务数分别为 1 500,2 000,2 500,3 000,3 500,4 000,4 500 进行实验。

响应时间的实验结果如图3所示。

由实验结果可知,随着用户请求的任务数增加,集群的响应时间随之增加,而且后期增加的速度较快。服务器由于负载等任务的增加,响应时间也会增加,当服务器满载运行时,有一部分任务处于等待状

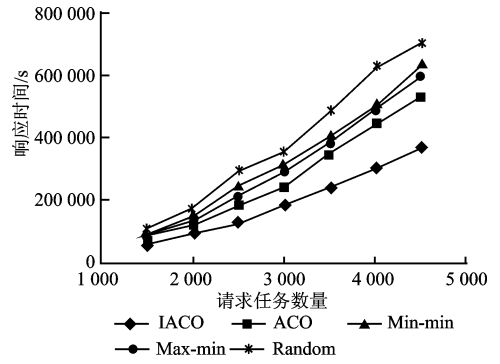


图3 响应时间

态,响应时间就会急剧增加。由于改进的蚁群算法更加均衡的利用了集群资源,所以响应时间较低,效果明显优于其他算法。

实验2:CPU 的利用率。

选取服务器节点数为 30,任务数分别为 2 000,3 000,4 000,5 000,6 000 做 5 组对比实验。

CPU 的利用率定义为

$$\text{per} = \frac{\sum_{i=1}^n T_j}{\max(T_1, T_2, \dots, T_n) \times n} \times 100\% \quad (11)$$

式中, T_j 为任务在服务器节点 j 上执行完成的时间, $\max(T_1, T_2, \dots, T_n)$ 为所有服务器节点中完成任务所花费的最长时间。

集群中 CPU 利用率的实验结果如图4所示。

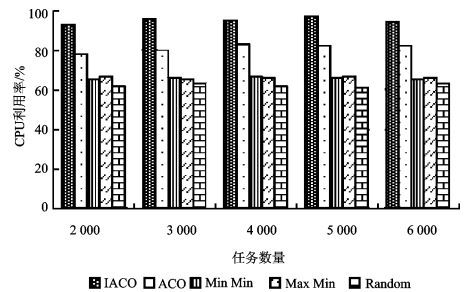


图4 CPU 的利用率

由实验结果可知,相比其他算法,采用改进的蚁群算法 CPU 的利用率最高。主要是因为考虑到每个服务器处理任务的能力存在差异,通过等待因子能更好的了解各服务器当前的状态,可以提高 CPU 的利用率。

实验3:系统吞吐率。

集群系统的吞吐率 (Mb/s) 定义为集群单位时间内的处理数据的速率。吞吐率和集群的处理器类型、缓冲区大小、磁盘速度、网络状况等因素有关。由于系统的吞吐率受外界环境的限制,所以仿真实验

在处理数据时,将同时并发请求数目作为对比维度,对比不同算法在相同并发请求数目下,集群吞吐率的大小。

同样选取服务器节点数为 50,用户请求的任务数分别为 1 500,2 000,2 500,3 000,3 500,4 000,4 500 进行实验。

集群系统吞吐率的实验结果如图 5 所示。

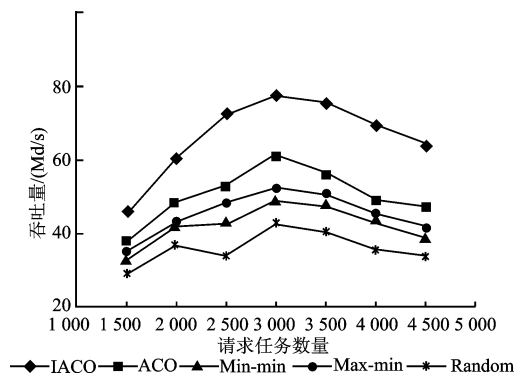


图5 集群的吞吐率

由实验结果可知,随着用户请求的任务数增加,集群的吞吐率也增加,但是当达到一个最大值时,吞吐率不增反降。这是由于请求的任务的数量已经达到了集群所能处理的最大值,服务器满载运行,增加的任务会消耗更多的资源导致吞吐率的降低。在集群资源调度中,改进的蚁群算法可以提供更高的吞吐率,说明该算法的性能优于其他算法的性能。

5 结束语

本文提出了一种改进的蚁群算法,通过引入等待因子,可以更好地了解集群系统中资源的利用情况,调度效果更加明显。从仿真实验可以看出,采用改进的蚁群算法,随着用户请求数量的增加,用户任务的响应时间更短,CPU 的利用率和系统吞吐率均得以提高。由此可见改进的蚁群算法是一种有效的

集群负载均衡算法,可以很好地适用于服务器集群环境。

参考文献:

- [1] Hai-jun L. Summarize about the server cluster technology [J]. Computer Knowledge & Technology, 2013,14(3): 38-45.
- [2] Wang T, Liu Z, Chen Y, et al. Load balancing task scheduling based on genetic algorithm in cloud computing [C]//IEEE 12th International Conference on Dependable IEEE Computer Society. Dalian:IEEE, 2014:146-152.
- [3] Penmatsa S, Chronopoulos A T. Game-theoretic static load balancing for distributed systems. [J]. Journal of Parallel & Distributed Computing, 2011, 71 (4): 537-555.
- [4] Long Q, Lin J, Sun Z. Agent scheduling model for adaptive dynamic load balancing in agent-based distributed simulations[J]. Simulation Modelling Practice & Theory, 2011, 19(4):1021-1034.
- [5] Mavrovouniotis M, Yang S. A memetic ant colony optimization algorithm for the dynamic travelling salesman problem[J]. Soft Computing, 2011, 15(7):1405-1425.
- [6] Chan W, Chua D K H, Kannan G. Construction resource scheduling with genetic algorithms[J]. American Society of Civil Engineers, 2014, 122(2):125-132.

作者简介:

刘万军 男,(1959-),教授,博士生导师.研究方向为计算机网络通信、数字图像处理。

王晓宇(通讯作者) 女,(1991-),硕士研究生.研究方向为计算机网络通信、遥感影像快速处理。

E-mail:410867038@qq.com.

曲海成 男,(1981-),博士研究生.研究方向为智能信息处理。

孟 煜 男,(1990-),硕士研究生.研究方向为大数据计算。