第33卷 第24期 计 算 机 工 程 2007年12月
Vol.33 No.24 Computer Engineering December 2007
·网络与通信· 文章编号：1000—3428(2007)24—0088—03 文献标识码：A 中图分类号：TP393.03

基于群智能的P2P计算网格负载均衡算法

吴湘宁[1]，汪 渊[2]
（1. 中国地质大学计算机学院，武汉 430074；2. 国防科技大学计算机学院，长沙 410073）

摘 要：对等网络(P2P)计算网格是采用非集中控制的动态网络环境，在P2P网络环境的各个对等节点间均匀分配任务是网格计算的重要研究内容。传统C/S模式的负载均衡算法无法适用于分布式且动态变化的P2P网络。文章提出了一种基于群智能和多代理技术的P2P网络负载均衡算法，设计并实现了基于蚁群优化算法的分布式作业调度策略。仿真结果表明该算法是合理而有效的。

关键词：对等网络；网格计算；群智能；多代理系统；蚁群优化算法

Load-balancing Algorithm in Peer-to-Peer Computing Grid Based on Swarm Intelligence

WU Xiang-ning[1], WANG Yuan [2]
（1. Computer Department, China University of Geosciences, Wuhan 430074;
2. Computer Institute, National University of Defence Technology, Changsha 410073）

【Abstract】 Peer-to-Peer(P2P) computing grid is dynamic network environment characterized by decentralized control. How to disperse tasks uniformly over peer nodes of P2P network environment becomes primary concerns of grid computing. Traditional Client/Server based load-balancing algorithms can not be applied to distributed and dynamic environment of P2P networks. This paper presents a P2P load-balancing algorithm based on swarm intelligence and multi-agent technique, designs and realizes a distributed task scheduling method based on ant colony optimization. Simulation results show that the algorithm is valid and effective.

【Key words】 Peer-to-Peer (P2P); grid computing; swarm intelligence; multi-agent system; ant colony optimization

对等网络(Peer-to-Peer, P2P)是无中心节点的动态分布式网络，由许多地位均等、可提供共享资源的对等节点(peer node)组成[1]。采用P2P体系结构的计算网格所研究的一个核心问题是负载均衡，即如何使负载较重的节点将一部分任务转移到负载较轻的节点中，从而使整个系统负载均衡分布的策略和算法。传统的C/S模式下的均衡算法大都只考虑了负载分配的单一参数制约因素，而且多是面向非实时环境，无法适用于具有很强动态伸缩性、开放性和自组织性的P2P网络。本文介绍一种基于蚁群智能计算和多代理技术的P2P网络负载均衡算法[2]。

1 系统的结构与实现

1.1 系统模型

P2P网络的每个对等节点都拥有一定的计算资源，可对外提供服务，每个对等节点可看作一个蚁巢(ant nest)，蚁巢负责接收本地用户以作业方式提交的计算请求及相关数据，并向网络上发送搜索蚂蚁以寻找能完成作业的计算资源。

以下是几个基本概念：

(1)源节点：接收用户作业并向外发出计算请求的节点。

(2)候选节点：一条搜索路径上所找到的计算能力最强的节点。

(3)目标节点：最终被选中并执行用户作业的候选节点。

(4)中间节点：在搜索候选节点的路径上所经过的节点。

(5)固有计算能力δ：表示某个节点的运算能力，$\delta=MIPS\times P$。其中，$MIPS$是节点的MI/s值；P是节点的CPU数目。

(6)下一作业预期启动时间$T_{start_next_job}$：表示节点中新到达任务可以开始执行的时间，等于节点当前作业队列中现有作业全部处理时间的总和。

系统的总体结构如图1所示。

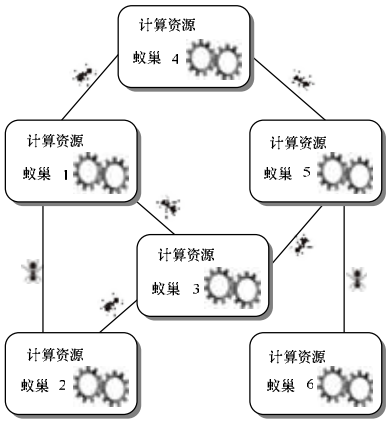

图1 P2P计算网格中蚁巢网络结构

源节点通过搜索蚂蚁在网络上找到一批候选节点，从候

基金项目：湖北省自然科学基金资助项目(12003ABA043)；中国地质大学优秀青年教师基金资助项目(CUGQNL0617)
作者简介：吴湘宁(1972－)，男，副教授、硕士，主研方向：分布式计算，高性能网络；汪 渊，高级工程师、硕士
收稿日期：2007-01-10 **E-mail：**000sun@sohu.com

选节点中挑出一个目标节点，将作业下载到目标节点，目标节点将该作业放入自身作业队列中，并在处理完这个作业后将计算结果返回给源节点，源节点再将计算结果返回给用户。

每一个蚁巢的标识符(ID)对应一种类型的信息素，搜索蚂蚁根据信息素的指引搜索目标节点，并留下新的信息素，一段时间后，通往强计算能力节点的信息素轨迹就变得明显，而通往弱计算能力节点的信息素轨迹则较淡。因此，强计算能力节点比弱计算能力节点有更多的机会获得新的作业，从而实现负载均衡。

1.2 蚁巢的结构

蚁巢负责本地计算资源的管理、信息素的维护、蚂蚁的创建与调度，并接收用户请求。它由计算资源管理器(Computing Resource Manager, CRM)，蚂蚁调度(Ant Scheduler, AS)和拓扑管理器(Topology Manager, TM)3 个逻辑模块组成。

(1)计算资源管理器(CRM)：负责本地计算资源的管理，向外提供网格计算服务，接收分配给当前蚁巢的作业，对作业队列进行维护，并向作业的源节点返回计算结果。

(2)蚂蚁调度模块(AS)：负责创建和销毁蚂蚁，为到访的蚂蚁提供访问当前蚁巢的接口，可根据蚂蚁的状态修改蚁巢中的信息素。AS维护着一张信息素表$\boldsymbol{\tau}$，$\boldsymbol{\tau}$是一个$m\times n$的矩阵，m是当前节点N_C的邻居节点的数目，n代表所有节点的数目，$\boldsymbol{\tau}$的元素τ_{ij}是信息素的强度，表示当前节点N_C通过前往第i个邻居节点的路径找到并使用第j个节点计算资源的可能性。τ_{ij}的初始值为 0.002。

(3)拓扑管理器(TM)负责探测邻居节点，以得到邻居节点的可达性、通信开销，还可接受新节点加入P2P网络的请求，更新邻居节点信息，从而实时维护网络拓扑结构。TM维护一张邻居表η，其长度为m，第k行元素记录当前节点N_C到其第k个邻居节点N_K的距离。

蚁巢的内部结构如图 2 所示。

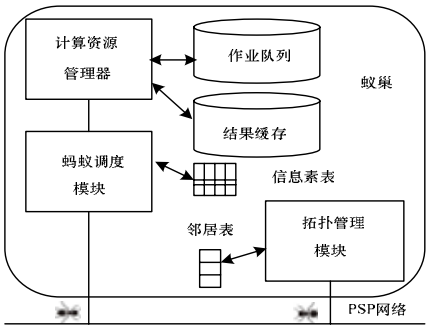

图 2 蚁巢的内部结构

1.3 蚂蚁种群的结构

根据蚂蚁所完成任务的不同，对其分类如下：

(1)搜索蚂蚁(search ant)：由源节点生成，可在蚁巢间移动寻找计算资源。每个蚂蚁有记录已访问过的蚁巢的列表，称为禁忌表(tabu list)，蚂蚁不能重复访问禁忌表中列出的已走过的蚁巢。每走一步，其生存时间TTL加 1，当TTL超过最大生存时间TTL_{MAX}时，搜索蚂蚁按原路返回源节点，并一路留下信息素。

(2)克隆蚂蚁(clone ant)：当出现多个搜索方向时，由中间节点创建，是搜索蚂蚁的克隆，也可在蚁巢间移动，与搜索蚂蚁分别探索不同分支的资源路径。克隆蚂蚁也需原路返回源节点并一路留下信息素。

(3)奖励和惩罚蚂蚁(reward and punish ant)：由源节点创建，沿到达目标节点的搜索路径前进。若任务成功完成，派出奖励蚂蚁，将路径上有关目标节点的信息素值提高到原值的 1.1 倍，从而鼓励其他蚂蚁选择这个目标节点。若超出一定时间，作业仍未完成，说明目标节点发生故障，则派出惩罚蚂蚁，将路径上有关目标节点的信息素值减少到原值的 0.85 倍，从而减少其他蚂蚁选择这个目标节点的概率。

1.4 计算资源的查找以及信息素的维护

基于蚁群算法的 P2P 网络负载均衡算法与普通负载均衡算法主要的不同之处在于：前者无固定的负载调度器，负载调度是由各个代理共同协作完成。而且可利用正反馈机制，通过信息素将负载调度和执行情况的历史信息用于指导将来的负载分配。算法描述如下：

(1)用户U向源节点N_S提交作业Q。N_S检查邻居表η，若未发现邻居节点，则将Q放到本地的作业队列中，否则执行下一步。

(2)N_S生成搜索蚂蚁ANT_{SEARCH}，置其TTL为 0，将Q的基本信息，如估计的作业运算量存入ANT_{SEARCH}，同时将N_S加入到ANT_{SEARCH}的禁忌表$S(ANT_{\mathrm{SEARCH}})$中，并记下N_S的固有计算能力δ_S，以及N_S的下一作业预期启动时间$T_{\mathrm{S,start_next_job}}$。

(3)ANT_{SEARCH}对每个未去过的邻居节点u计算转移参数P，选择P最大的邻居节点继续前进。算法如式(1)所示。

$$P=\arg\max_{u\in U_{\mathrm{allowedneighbor}}}([\tau_{u,j}]^{\alpha}.[\delta_j]^{\beta}) \tag{1}$$

其中，$U_{\mathrm{allowedneignbor}}$是$ANT_{\mathrm{SEARCH}}$还未去过的当前蚁巢邻居节点的集合；$\tau_{u,j}$是信息素矩阵中的元素，代表通往邻居节点$u$的链路上关于节点$j$的信息素浓度；$\delta_j$是节点$j$的固有计算能力。$\alpha,\beta$是常量，分别代表历史信息素以及资源固有计算能力的重要性。

还可将式(1)改进成为式(2)和式(3)，计算所有未去过的邻居节点u的转移参数P在总的转移参数中所占比例，称为转移概率P_u，若P_u超过了阈值q，则下一步前往u。改进后搜索蚂蚁不但可利用信息素历史信息，而且可按一定概率探索新的路径，避免陷入局部最优解。

$$P_u=\frac{[\tau_{u,j}]^{\alpha}.[\delta_j]^{\beta}}{\sum u\in U_{\mathrm{allowedneighbor}}([\tau_{u,j}]^{\alpha}.[\delta_j]^{\beta})} \tag{2}$$

$$I_{if\_select\_u}=\begin{cases}1 & \text{if}(P_u>q)\wedge(u\in U_{\mathrm{allowedneighbor}})\\ 0 & \text{otherwise}\end{cases} \tag{3}$$

若转移概率超过阈值q的邻居节点数n多于一个(n>1)，则将ANT_{SEARCH}克隆n-1 份，并分别将搜索蚂蚁和克隆蚂蚁发往n个被选中的邻居节点。若所有邻居节点的转移概率都小于q，则将搜索蚂蚁发往转移概率最大的节点。

(4)当搜索蚂蚁或克隆蚂蚁ANT_J到达一个新蚁巢N_{NEW}时，若N_{NEW}未被ANT_J的克隆兄弟访问过，则转到步骤(5)。否则转到步骤(6)。

(5)ANT_J将N_{NEW}加入到禁忌表$S(ANT_J)$中，并记录N_{NEW}的固有计算能力以及下一作业预期启动时间。修改N_{NEW}的信息素表，即修改上一步刚走过的节点N_{LAST}关于已走过节点N_{HISTROY}(有多个)的信息素$\tau_{\mathrm{LAST,\ HISTORY}}$，若该值等于初始值 0.002，则用$N_{\mathrm{HISTROY}}$的固有计算能力$\delta_{\mathrm{HISTORY}}$来代替，否则保持不变。$ANT_J$的$TTL$值加 1，若$TTL$小于$TTL_{\mathrm{MAX}}$，则将$N_{\mathrm{NEW}}$加入到$ANT_J$的禁忌表$S(ANT_J)$中，并记录$N_{\mathrm{NEW}}$的固有计算能力及下一作业预期启动时间，转到步骤(3)，否则转到步骤(6)。

(6)ANT_j在搜索路径上找出候选节点，其选择标准是候选参数$P_{\text{candidate}}$，它是节点的固有计算能力δ_j以及下一作业预期启动时间$T_{\text{j,start_next_job}}$倒数的加权平均，如式(4)所示。

$$p_{\text{candidate}} = \arg\max_{j \in N_{\text{history}}} (\text{w}.\delta_j + (1-\text{w}).\frac{1}{T_{\text{j,start\_next\_job}}}) \quad (4)$$

式中，w是常量，表示固有计算能力所占的权重。ANT_j找出候选参数最大的节点作为候选节点，再将候选节点的URL(Uniform Resource Locator)、候选参数以及到达候选节点所走的路径传送给源节点。

(7)ANT_j按原路返回N_S，沿途修改各个中间节点N_{MIDDLE}上的信息素表，若信息素表中N_{MIDDLE}对应的信息素等于初始值0.002，则可用N_{MIDDLE}的固有计算能力δ_{MIDDLE}来代替。否则保持不变。

(8)N_S在时间超过TTL_{MAX}后收到所有候选节点，从中挑选出候选参数值最大的候选节点作为目标节点。N_S将作业Q下载到目标节点上执行。

(9)作业成功完成后，源节点派出奖励蚂蚁修改原来搜索路径上关于目标节点的信息素，使其提高到原来的ρ_{reward}倍，这里ρ_{reward}=1.1，称为奖励因子。若由于故障等原因作业超时仍未完成，则源节点派出惩罚蚂蚁，将搜索路径上关于目标节点的信息素值减少为原来的ρ_{punish}倍，这里ρ_{punish}=0.85，称为惩罚因子。

(10)蚁巢每隔一定的时间间隔T_{interval}，按式(5)进行信息素的挥发操作。

$$\tau_{u,j} = (1-\rho).\tau_{u,j} \quad (5)$$

其中，ρ 是常数，称为挥发系数，这里设为 0.05。若信息素值等于初始值 0.002，则不进行挥发。

2 仿真结果

结合本文给出的算法，利用P2P网格仿真工具Anthill[3]进行了仿真实验，生成了一个 30 个节点的P2P网络，每个节点上有 3 个用户，1~4 个CPU，MIPS都置为 400，每个用户生成 0~5 个作业请求，每个作业请求的运算负载从 400 M~ 8×10^6 M条指令不等。各个节点使用同样的时钟周期，实验所用参数如表 1 所示。仿真结果如图 3、图 4 所示。

表 1 数据集

名称	说明	值
α	信息素中历史信息素的重要性	0.6
β	信息素中固有计算能力所占权重	0.4
w	候选参数中固有计算能力所占权重	0.5
TTL_{MAX}	蚂蚁的最大生存时间	20
ρ	挥发系数	0.05
ρ_{reward}	奖励因子	1.1
ρ_{punish}	惩罚因子	0.85

从图 3 可看出，每个节点上的负载是随机变化的，用(指令条数/400 M)来衡量。图 4 显示 50 个时钟周期后，负载几乎被均匀地分配到所有节点上，整个系统的负载趋于平衡。

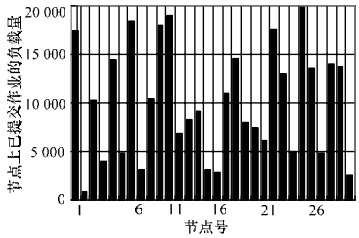

图 3 各个节点上提交的负载量

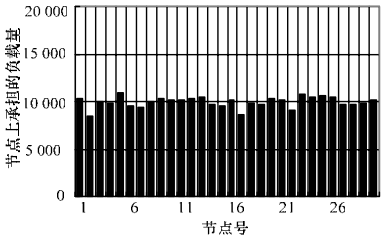

图 4 50 个时钟周期后各个节点承担的负载量

实验结果还表明，实验中所设置的参数，尤其是蚂蚁的最大生存时间TTL_{MAX}、信息素挥发系数ρ、奖励因子ρ_{reward}、惩罚因子ρ_{punish}对算法性能的影响较大，而且当系统规模较大时，会带来较大的系统通信开销并影响算法效率。

3 结论

本文设计的基于群智能的 P2P 网络负载均衡算法，利用蚁群算法的正反馈性、分布性，可有效解决动态变化的网格计算环境下作业的平均分配问题，是一种有效的启发式分布式负载均衡算法。仿真结果表明，算法在计算资源的搜索和分配方面取得较好效果。但是当系统规模扩大时，在参数设置、算法开销控制方面还需进一步研究。

参考文献

[1] Rowstron A, Druschel P. Pastry: Scalable, Distributed Object Location and Routing for Large Scale Peer to Peer Systems[C]//Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware). Heidelberg: [s. n.], 2001: 329-350.

[2] Hackwood S, Beni G. Self-organization of Sensors for Swarm Intelligence[C]//Proc. of the IEEE International Conference on Robotics and Automation. Piscataway, New Jersey: [s. n.], 1992: 819-829.

[3] Russo F. Design and Implementation of a Framework for Agent-based Peer-to-Peer Systems[D]. Bologna: University of Bologna, 2002-07.

(上接第 87 页)

Performance Forecasting Service for Metacomputing[J]. Journal of Future Generation Computing Systems, 1999, 15(5): 757-768.

[9] Gong L, Sun X H, Waston E. Performance Modeling and Prediction of Non-dedicated Network Computing[J]. IEEE Transactions on Computer, 2002, 51(9): 1041-1055.

[10] Casanova H. Simgrid: A Toolkit for the Simulation of Application Scheduling[C]//Proc. of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid. [S. l.]: IEEE Computer Society Press, 2001: 430-437.