基于多 Agent 网格资源调度的负载均衡研究*

李红婵[1]，蔡乐才[2]，朱颢东[3,4]
(1.四川理工学院 自动化与电子信息学院，四川 自贡 643000；
2.四川理工学院计算机科学系，四川 自贡 643000；
3.中国科学院成都计算机应用研究所，四川 成都 610041；
4.中国科学院研究生院，北京 100039)

摘 要：介绍了一种基于多 Agent 的网格资源调度方法，并提出了一种负载均衡算法设计思想，以改善网格环境中资源分配不均的问题。

关键词：网格；Agent；资源调度；负载均衡
中图分类号:TP393.02 文献标识码：A

Load balancing technology based on multi-Agent grid resource scheduling

LI Hong Chan[1], CAI Le Cai[2], ZHU Hao Dong[3,4]
(1.Institute of Automation and Electronic Information, Sichuan University of Science & Engineering, Zigong 643000, China ;
2.Dept.of Computer Science, Sichuan University of Science & Engineering, Zigong 643000, China ;
3.Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdu 610041,China;
4.The Graduate School of the Chinese Academy of Sciences, Beijing 100039, China)

Abstract: This paper introduces an agent-based grid resource scheduling method. Then the arithmetic of load balancing is presented in order to improve this problem in the grid environment.

Key words : Grid; Agent; resource; load balancing

网格是把分布在不同地理位置上的计算资源(包括高端服务器、集群系统、MPP 系统大型存储设备、数据库等)通过因特网整合成一台巨大的超级计算机，实现各种资源的全面共享。网格节点是这些网格计算资源的提供者。网格的根本特征不是它的规模，而是资源共享，消除资源孤岛。因此网格的关键技术是资源调度，如何有效地调度网格资源将成为网格系统是否可用的关键问题。

资源调度和负载均衡是分布式系统设计中的关键问题。传统的主从节构无法避免单点故障、性能瓶颈等问题，而对等网络[1]P2P 是一种完全分布式的计算模型，不存在这些问题。如果一个系统能保持良好的负载均衡状态，那么该系统可以获得较高的吞吐量。本文首先介绍了一种基于多 Agent 的网格资源调度方法，并在此方法的基础上提出了一种负载均衡算法的设计思想。

1 基于多 Agent 的网格资源调度模型

在网格应用开发过程中，采用多 Agent 技术能够提高网格系统的智能性、灵活性和健壮性。Agent 具有自主性，无需进行集中的控制就可以自主决定自己的行为，也具有交互性，相互间可以进行协作，协同完成任务。此外，可以将较复杂的任务封装在 Agent 中，由多个 Agent 相互协作，可以完成单个 Agent 或其他软件不能完成的任务。鉴于网格的动态性，需要动态、有自适应能力的调度算法来对网格资源进行有效调度。

1.1 基于多 Agent 的网格系统节构图

基于多 Agent 的网格系统在逻辑上可以分为 4 个层次，如图 1 所示。系统中，客户端的任务主要是确

* 基金项目：四川省计划项目(2008GZ0003)；四川省科技厅科技攻关项目(07GG006-014)；四川省教育厅科研项目(2007ZL048)

定用户的操作请求，资源端负责处理用户请求，网格中间件致力于整合各种网格资源。网格资源请求Agent作为用户代理，负责协调用户与网格资源Agent间的交互。其目的是使用户无需与网格资源直接联系，即可更加方便地使用所需的资源。

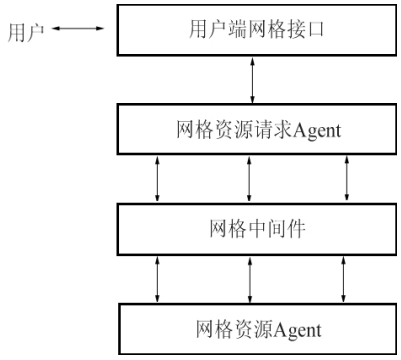

图1 整合多Agent技术的网格系统

用户通过客户端的应用程序提出服务请求，激活网格资源请求Agent。网格资源请求Agent依据用户提出的要求，在众多的资源提供者中进行筛选。满足用户要求的资源可能会有多个，网格资源请求Agent可以代替用户做出最终决策，确定一个最佳资源提供者。满足要求的资源也可能不存在，此时，网格资源请求Agent可以把若干个资源进行组合，以此得到满足用户要求的资源。任务完成后，再由网格资源请求Agent将最终节果传给客户端。

1.2 基于多Agent的网格资源调度节构图

网格的实现可以依赖不同的拓扑节构，将Agent引入网格，并采用树型节构作为网格模型[2-6]。其优点是跨域搜索资源时，可以在多项式时间内完成匹配。其中，底层Agent直接管理其辖区内的资源节点，记录资源信息，采用一定的资源策略来实现任务和资源的匹配。上层Agent主要负载任务的跨域调度。它可以将其子Agent辖区内没有得到匹配的任务，提交到另一个子Agent进行匹配。

基于Agent网格模型中，Agent节点包括5个功能模块和2种数据节构。每个节点都有任务请求客户端和任务应答客户端,图2是基于Agent网格模型下的资源调度模型，它描述了底层Agent节点和资源节点的各个模块和模块的工作流程。

在Agent接受任务到建立任务连接过程中，如果任务池中为非空，第一步从任务池中提取第i项任务；第二步分析第i项任务的资源需求；第三步由动态平衡负载模块(全局资源剩余率，全局任务空闲率，全局负载变化率)生成网格资源调度的上限；第四步调用此模块返回匹配节果；第五步如果有匹配节果，就创建资源节点和任务请求节点间的工作流，成功通讯后断开本连接；最后在第十步检测到第i项任务完成后，在Agent中清楚所有任务。

任务处理模块将收到的任务放入任务等待队列，如果任务等待队列为非空，在第八步任务管理器从任务队列中选取第i项任务运行并更新任务状态表汇总的状态，第九步中若任务完成，调用任务释放模块与任务请求节点通讯，第十一步中若任务已经成功归还任务

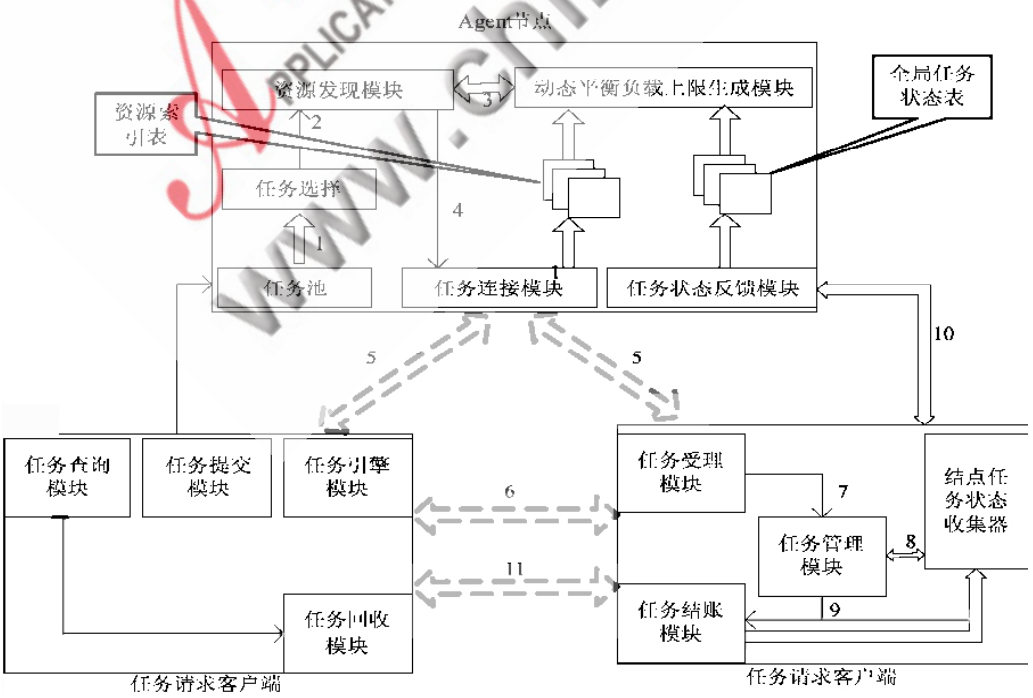

图2 资源高度模型

请求节点，则更新任务状态，收回所占资源。

1.3 基于多 Agent 的网格资源管理模型下的的紧邻算法

代理对计算资源和 Web 用户是透明的，它只与自己相连的节点(代理)打交道，接受这些代理或者计算资源的注册和撤销[7]。如图 3 所示，计算资源可以选择某个代理进行注册，每个代理最终管理一组资源，响应用户的任务请求。由于可用带宽以及主机性能等原因，各个代理的资源利用率、负载等不尽相同，可能会出现某些代理服务器负载过重、有些代理服务器处于轻负载运行状态的情况。一个代理接收到一个新的计算任务以后，如果自身负载过重，或者自身的可用资源难以满足新的计算任务的要求，就会根据负载平衡算法，通过 ACP(Agent Communication Protocol)把计算任务交给其他代理完成。

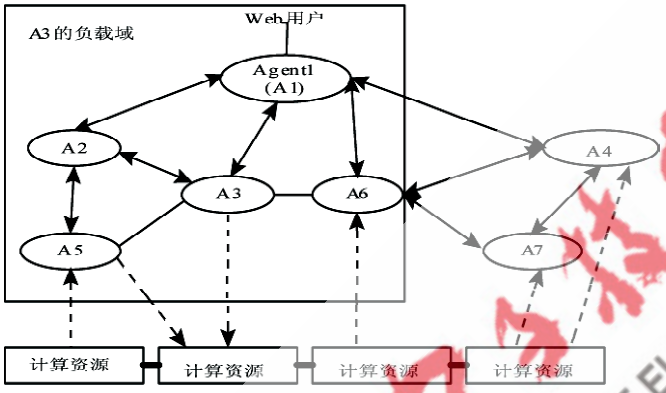

图 3 风格资源管理模型节构图

假定并行处理系统是由多个节点按照一定的节构相互连接构成的并行处理网络，每个节点只能与其直接连接的节点通信，计算负载也只能通过这些连接移动。在负载平衡过程中，每个节点只能与其周围的节点进行负载交换，通过多次循环迭代实现系统的负载平衡。

2 基于多 Agent 的网格资源调度层次模型的负载均衡算法

2.1 负载平衡算法的思想

本文设计的负载平衡算法属于动态平衡算法，决策取决于系统当前的状态。也就是说，系统可以根据当前的负载分布情况，对各个节点上的负载进行动态调整，使已经分配给重载节点上的任务，通过通信设备，迁移到轻载的节点上去，从而提高系统的资源利用率，减小任务的平均响应时间。

算法思想是紧邻上限机制(Neareast Neighbor-Upper Bound)，它假定并行处理系统是由多个节点按照一定的节构相互连接构成的并行处理网络，每个节点只能与其直接连接的节点通信，计算负载也只能通过这些连接移动。每个节点对于任务提交的资源要求进行分析，得出其任务可以完成的底限；再在底限的基础上适当上浮一定比例，得到一个资源要求的上限；这样，就有一个资源要求区间，用这个区间值在资源中进行匹配，在所得的资源节点中，使用一定的资源搜索算法，进行具体的资源定位。在负载平衡过程中，通过节点上限机制和紧邻的节点多次循环迭代实现系统的负载平衡。

2.2 负载平衡算法的设计

当接受某项任务后，网格的负载和吞吐量必然发生变化；不同任务引起的变化不同，对上限的影响也是不同的。任务对整个网格负载影响，是以接受节点资源能力的变化表现出来的。先给出 2 个公式：

$$NLR=NUR_i / \sum_{i=1}^{n} NUR_i \tag{1}$$

$$NRRR=(NAR_i-NUR_i)/\sum_{i=1}^{n}(NAR_i-NUR_i) \tag{2}$$

其中，节点负载率(NLR)描述节点的当前已耗费的资源能力；节点剩余资源率(NRRR)描述节点的未来资源能力。为了定义上面 2 个概念，再引入 2 个概念：单节点已使用资源(NUR)和单节点全部资源(NAR)。NLR 与单节点的上限(SUB)成正向关系。

单个任务对网格吞吐量的影响很小，可近似看作不变。此时，网格中部分资源的负载增大，剩余资源减少。为使负载不继续变大，这部分资源通过紧邻算法将更多能力较类似的节点(能力较强的)考虑进来确保上限度量 UB<1。

相同负载的强节点和弱节点的剩余资源能力是不一样的，剩余资源能力高的节点对 SUB 调高的愿望较低；剩余资源能力低的节点则相反。所以，NRRR 与 SUB 成逆向关系。于是，有下面的表达式：

$$SUB_i=C_i(NLR_i / NRRR_i) \tag{3}$$

其中，SUB 由节点的资源能力在整个网格中的地位决定。因此，它不仅与节点自己的能力有关，还与网格资源的节构有关。

$$\sum_{i=1}^{n} SUB_i = \sum_{i=1}^{n} C_i \sum_{i=1}^{n}(NLR_i / NRRR_i)$$

$$= SUBS\sum_{i=1}^{n}(NLR_i / NRRR_i) \tag{4}$$

其中，$SUBS=\sum_{i=1}^{n} C_i$即网格所有节点的上限之和，被称为网格动态节点。它应当只与整个网表初始权值设置格的资源能力节构有关。而网格的资源能力节构是动态变化的，它取决于很多因素:全局任务空闲率(TTFR)反映了网格当前的吞吐量，全局负载变化率(TLCR) 反映了全局任务资源要求的波动速率，全局剩余资源率(TRRR) 反映了网格今后的能力。

综上所述，节点动态上限机制最终可以下面的形式来描述：$C=\{f(TTFR,TLCR,TRRR,UC)\}$，UC 为其他因素。其中 TTFR，GSRR 与 UB 成正向关系，TLCR 与 UB 成加速关系.则有：

$$UB=k\sum_{i=1}^{n} SUB_i = k\sum_{i=1}^{n} C_i \sum_{i=1}^{n}(NLR_i / NRRR_i)$$

$$= kSUBS\sum_{i=1}^{n}(NLR_i / NRRR_i)$$

其中 k 为调整常数，将求和项调整为网格可以感知的程度，并确保 UB<1。

网格环境中存在着多个 Agent，如图 4 所示。

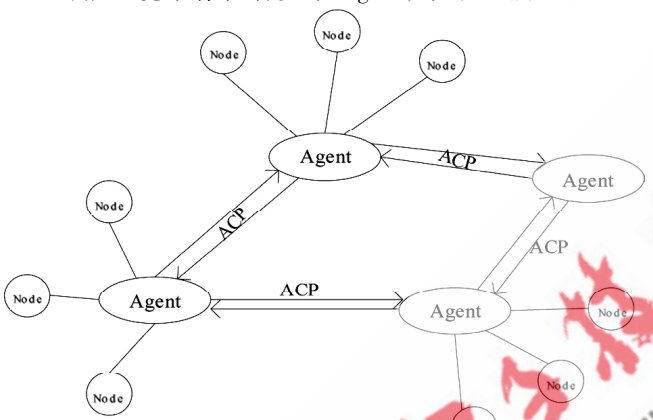

图 4 多 Agent 的分布协调

每个 Agent 周围都分布了多个节点。在单个 Agent 周围的节点和节点之间通过紧邻算法，实现系统的协调，解决节点和节点之间的负载平衡问题。在 Agent 和 Agent 之间也通过紧邻算法和 ACP，达到网格资源调度的协调。在多 Agent 网格资源环境中，利用紧邻算法实现了节点与节点的协调、单个 Agent 与单个 Agent 的协调，使整个网格环境实现分布式的协同。再利用上限度量的设定，最终实现系统的网络负载平衡。

3 基于多 Agent 的负载均衡算法的优点

本负载均衡算法最大的特点在于：根据网格实时负载和吞吐量等信息进行动态调度，节点与节点之间运用紧邻算法，设定上限量度，恰如其分地选择资源；能很好地解决小任务堆积强节点的问题；能够实现资源预留功能；可扩展性强，它利用了 1 种两级调度：一级调度采用动态上限机制，二级调度可以根据不同的网格系统来灵活、透明、有效率地选择。

本文将 Agent 技术引入网格资源管理调度，提出了 1 个网格环境下基于 Agent 的上限机制负载均衡的设计思想，充分发挥了 Agent 的智能性、自主性，并提高了网格资源的利用效率。本算法较大地发挥了系统的负载平衡能力，提高了网格计算能力和资源利用率。

参考文献

[1] STOICA I, MORRIS R, KARGER D, et al. A scalable peer-to-peer lookup service for internet applications[A].Processing of ACM SIGCOMM[C]. New York: ACM Press, 2001. 149-160.

[2] LI Chun Lin, LI La Yuan. Agent framework to support computational grid[J]. The Journal of Systems and Software, 2004, 70(1/2): 177-187.

[3] LI Chun Lin, LI La Yuan. Apply agent to build grid service management[J]. Computer Applications 2003(26): 323-240.

[4] CAO Jun Wei, SPOONER D P. Grid load balancing using intelligent agents[J]. Future Generation Compute Systems, 2005(21): 135-149.

[5] FREY J, TANNENHAUM T, FOSTER I, et a1. A computation management agent for multi—institution all grids[J]. Cluster Computer, 2002, 5(3): 237-246.

[6] CAO JunWei, JARVIS S A, SAINI S, et a1. An agent—based resource management system for grid computing[J]. Scientific Programming(Special Issue on Grid Computing), 2002, 10(2): 135-148.

[7] 赵姗,胡根,周兴社.基于多Agent网格资源管理模型的负载均衡研究[J].微电子学与计算机, 2005, 22(10): 51-53.

[8] 邸烁,郑纬民,王鼎兴,等. 并行WWW服务器集群请求分配算法的研究[J]. 软件学报,1999,10(7):713-718.

(收稿日期：2009-05-27)