混合蚁群算法在网格计算任务调度中的应用①

Hybrid Ant Colony Algorithm for Task Scheduling in Grid Computation

王 亮 张险全 陈未如 （沈阳化工学院 计算机科学与技术学院 辽宁沈阳 110142）

摘 要： 网格环境下的资源分配与任务调度问题已经被证明了是一个 NP 难题，而传统的任务调度算法很难对大量的异构的、动态的网格任务进行有效的调度。本文提出了一种任务调度模型，并且在该模型中采用混合蚁群算法，该算法以信息素为启发，引导蚂蚁选择最优资源。蚂蚁选择资源之后不仅进行信息素的整体更新，还要求预分配网格资源时进行信息素的局部更新。模拟实验表明该算法是一种快速，有效，负载更均衡的算法。

关键词： 网格 任务调度 信息素 混合蚁群算法

1 引言

网格计算是并行和分布处理技术的一个发展方向。在网格计算中，任务管理、任务调度和资源管理是网格必须具备的三个基本功能，其中任务调度问题仍是必须解决的关键性问题之一。有效的任务调度策略可以充分利用网格系统的各种闲杂资源，尤其是计算能力，可以提高网格系统的性能，以便更好提供网格服务。然而，一般网格任务调度问题已经被证明是一个 NP 完全问题[1]。因此，它引起了众多学者的关注，成为目前网格计算研究领域中的一个焦点[2]。

目前学术界与工业界已经提出并实现了诸多任务调度算法[3]，例如最小对最小启发式算法（Min – Min）、最大对最小启发式算法（Max – Min）和容忍启发式算法等。传统的任务调度算法在网格环境下存在着一定的缺陷，一是不能很好的平衡资源节点的负载，二是不能更好的满足用户服务质量需求，还会出现低 QoS 需求占用高 QoS 资源的现象。本文利用蚁群算法的特性，结合 MAX – MIN 算法的优点，提出了混合蚁群算法进行网格环境的任务调度，资源节点通过信息素浓度来反映自己的负载平衡情况和服务质量情况，从而指导蚂蚁对资源的选择。本文将从以下几方面介绍，包括任务调度

模型、混合蚁群算法、信息素的建立、信息素的更新方式、路径选择和仿真实验。

① 基金项目：辽宁省教育厅科学技术研究项目（20060675）

2 网格任务调度描述

本文所讨论的网格任务调度是在满足用户服务质量的情况下，使得总的任务完成花费代价最小，并且使负载尽量的均衡。我们所研究的网格系统是多资源和多用户实体的，并且任务的调度是基于市场经济条件的。由于每个任务在各个资源上的运行时间受多种因素的影响，而且是动态变化的，所以如何有效的解决这个问题是任务调度的关键，这也是网格计算中的任务调度与传统的多机任务调度的主要区别。一般地，网格任务的执行要受用户预算的约束，用户预算是确定的，资源处理任务是不能超过其最大预算值。

网格任务调度的形式化描述[4]如下：假设任务总数为 n，网格系统中资源总数为 m。其中 $J=\{J_1,J_2,\cdots J_n\}$ 表示 n 个独立的用户任务，$R=\{R_1,R_2,\cdots R_n\}$ 代表 m 个计算资源的集合，t_{ij} 表示资源 j 处理任务 i 所需代价。将 m 个资源看作一个大的分布式系统，任务调度的策略就是将 n 个独立的任务分配到 m 个资源上，使得整个系统处理 n 个任务所需代价最少。

任务调度模型用一个四元组表示如下：G = (J, R, E, Q)，其中 J 是任务集合，表示用户队列中的所有任务，R 是表示网格系统中的资源，E 是表示笛卡儿积 J ×

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

R 上的二元关系，Q 是表示 J 中任务的服务质量需求。对于任务队列中的任意任务 i，在满足 Q 中服务质量需求的情况下，所分配的网格资源 R_i 满足：$R_i \subseteq R,\ 1 \le i \le |J|$。如果将以上四元组用图来表示，则可以得到一个满足服务质量需求的任务调度图（如图 1 所示）。假设 $|J|=n, |R|=m$，图 1 中的边表示任务 i 在资源 j 上处理完毕所需要的代价，$1 \le i \le n, 1 \le j \le m$。因此，任务调度问题则可转化为在图中找 n 条边，每条边代表一个任务的分配，所以与每条边相连接的任务节点都是不同，则形式化描述如下：

$$f = Min\sum_{j \in R} e.j \qquad (1)$$

上式中 表示以资源 j 为终点的边，即是代表着分配到资源 j 上的任务。当网格系统中有新的节点加入时，图 1 中资源节点就增加，当网格系统中的节点离开时，该资源节点就返回失败信息给任务调度器，然后再从资源集合中删除。

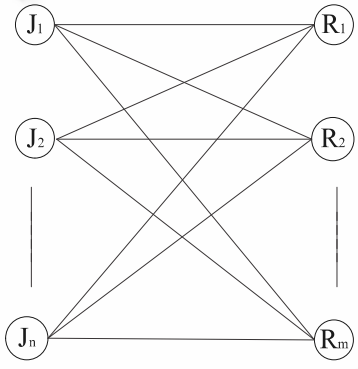

图 1　任务调度图

3　网格任务调度中的混合蚁群算法描述

3.1　混合蚁群算法原理

蚁群算法（AC）由意大利人 M. Dorigo 等人首先提出的[5]，是一种基于蚂蚁行为的启发式算法。该算法按照启发式思想通过信息素的诱导作用，逐步收敛到问题的全局最优解。网格计算环境下的资源分配是众所周知的 NP 问题，大量的实验表明，蚁群算法是一种求解 NP 类问题的有效算法，具有较强的鲁棒性和内在的分布并行性，且易于和其他方法相结合。

本文提出了混合蚁群算法（Hybrid Ant Colony Algorithm，HAC）在网格环境下进行任务调度，其基本思想是：采用极大极小算法与蚁群算法相结合的方式，也即是在信息素的指引下让未分配的任务按任务量从大到小依次按概率选择资源节点，当然最早完成任务的资源节点被选择的概率较大。该任务调度模型采用四元组来表示，模型集合 J 中的任务被映射为蚂蚁，每次蚂蚁选择资源时根据集合 Q 中的资源 QoS 需求关系来选择满足 QoS 需求的资源。蚂蚁选择路径的过程看成是在图 1 中选择以任务节点为起始点以资源节点为终点的边，并且以任务集合中的节点为出度的边有且仅有一条。蚂蚁选择资源节点也就是选择图 1 中边的终点。选择边的终点的方法是根据资源节点的信息素浓度按照概率转移公式（4）进行计算。用边的权值表示任务在对应资源上处理花费的代价，则使得（1）成立的边即是最优解。

3.2　网格资源信息素的建立

我们把任务抽象成人工蚂蚁，蚂蚁搜索路径是依据概率选择的，而概率选择是根据信息素浓度来确定，真实的蚂蚁通常是选择信息素浓度大的路径，而人工蚂蚁是依概率选择路径的，信息素浓度越大的路径被选择的概率就越大。

假设第 i 个资源的处理能力是 P_i，并且任务调度节点与资源节点 i 之间的通信能力为 K_i，资源负载平衡能力为 L_i，则每个资源的初始信息素可以表示为：

$$\tau_i(0) = \rho_0 P_i + c_0 K_i + d_0 L_i \qquad (2)$$

其中参数 ρ_0，c_0，d_0 分别表示资源的处理能力初始化因子、资源节点间通信能力初始化因子和负载平衡能力初始化因子。参数 ρ_0，c_0，d_0 的确定是根据对资源处理能力、资源间的通讯能力以及负载均衡能力的关心程度。上式能够实时的反映初始资源节点的计算能力、通讯能力和负载平衡能力。

3.3　人工蚂蚁的路径选择

人工蚂蚁是依据概率选择资源节点的，人工蚂蚁选择执行效率高的资源概率大，反之则小。通常的情况下是大概率事件发生，但是有时小概率事件也会发生。小概率事件在蚁群算法中有特别的意义，因为小概率事件的发生可以逃离局部最优，使得趋向整体最优，这就是蚁群算法的多样性。

假设在 t 时刻，第 i 个人工蚂蚁选择资源 j 可以由

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

下面概率公式[6,7]给出：

$$P_{ij}(t)=\begin{cases}\dfrac{[\tau_j(t)]^{\alpha}[\eta_j(t)]^{\beta}}{\sum[\tau_u(t)]^{\alpha}[\eta_u(t)]^{\beta}} & j\ u\in R\\ 0 & \text{else}\end{cases} \quad (3)$$

其中 R_i 是在网格任务调度描述中所讲到的第 i 个任务在满足服务质量需求下的可用资源。$\tau_{ji}(t)$ 是 t 时刻资源 j 上的信息素的浓度；$\eta_j(t)$ 是能见度因素，一般是资源初始计算能力，即是 $\eta_j(t)=\tau_{ji}(0)$；参数 α 是信息素的重要程度；参数 β 是资源原始属性的重要程度。因此，由公式(3)可知，信息素浓度高的资源被选择的概率大，而信息素浓度是由信息素更新公式所决定的。

当高 QoS 资源与中等 QoS 资源相差不大时，由转移概率公式(3)可知，它们被选中的概率相差不大，如果中等 QoS 资源数量较多，那么可能导致很多情况下高 QoS 资源实际未被选中。为了解决这个问题，提高算法收敛的速度，我们采用伪随机的方法如下：

$$S_{ij}=\begin{cases}\underset{j\in R}{Max}[\tau_j(t)]^{\alpha}[\eta_j(t)]^{\beta}, & \mu\geq\mu_0\\ P_{ij}, & \mu<\mu_0\end{cases} \quad (4)$$

μ_0 是常数，事先设置的阈值；μ 是随机数；P_{ij} 是公式(3)。当人工蚂蚁要选择资源时，产生一个随机数 μ，然后根据公式(4)选择资源，如果 $\mu<\mu_0$，则按照公式(3)选择资源。

3.4 信息素更新方式

信息素更新分为两个阶段：一是局部信息素更新，二是全局信息素更新。局部信息素更新是在所有任务未被分配完之前，当蚂蚁按照概率选择了资源之后，则该资源按一定的规则更新信息素。假设第 i 个蚂蚁选择了资源 j，则资源 j 按以下公式(5)更新信息：

$$\tau_j(t+1)=\tau_j(t)+(1-\rho)\ \tau_j(t)+bC_j \quad (5)$$

其中参数 ρ 是局部挥发系数；$\tau_{ji}(t)$ 是在 t 时刻资源 j 的信息素浓度；$\triangle\tau_{ji}(t)=-K$，K 是按第 i 个任务的大小进行评估而得到的评估值；b 是负载平衡因子，$C_j=-K0$，K0 是按资源 j 的负载情况进行评估得到的评估值，代表着本地负载情况。资源节点的负载受到两个方面的因素影响，一是资源节点本地任务处理情况，二是其他节点请求的任务执行情况。

全局信息素更新是指当所有的蚂蚁选择了资源之后，则各资源节点根据已经分配到该资源节点上的任务进行信息素更新，从而指导下一次的蚂蚁进行资源选择。当所有蚂蚁完成了资源选择之后要进行下一次资源选择之前进行全局信息素更新按照公式(6)进行：

$$\tau_j(t+1)=(1-\rho)\ \tau_j(t)+\triangle\ \tau_j(t)+bC_j \quad (6)$$

信息素更新时，由于迭代时间比较长，各资源节点上的任务分配情况差异比较大，可能出现两种情况，一是信息素浓度过低，二是信息素浓度过高。信息素浓度过低和过高都可能导致陷入局部最优和负载不平衡。为了减少陷入局部最优和负载不平衡情况的发生，我们给信息素定义一个取值区间[L,H]，其中 L,H 是常数，分别是信息素的取值下限和上限。假设按以上信息素更新公式(5)和(6)计算出来的值为 r，则按公式(7)进行调整信息素浓度。

$$\tau_j(t+1)=\begin{cases}L, r\leq L\\ r, L<r\leq H\\ H, H<r\end{cases} \quad (7)$$

3.5 混合蚁群算法伪代码描述与时间复杂度分析

假设网格系统中的任务是独立的，RsQoS 表示资源的 QoS，Q 表示所有任务 QoS 需求集合，则算法伪代码如下：

```
设置参数并初始化信息素浓度;
将集合 J 中的任务从大到小排序;
While(循环次数)
For 蚁群中的蚂蚁
    For 所有的资源
        If(RsQoS 满足 Q 中需求)
            蚂蚁根据信息素浓度以及启发式信息的引导,选择图 1 中的边;
局部信息素更新;
构造一步问题的解空间;
        EndIf
    EndFor
EndFor
根据当前已获得解的情况进行全局信息
素更新;
按照公式(1)计算当前最优值;
If(当前最优值小于总体最优值)
```


©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

更新总体最优值和当前最优解空间;
EndWhile

时间复杂度分析:假设循环次数为 T,则该算法的时间复杂读为 O(Tmn),当 m = n 时,该算法的时间复杂度为 O(Tn2)。一般情况下 m < n,所以算法的时间复杂度取决于 T,如果问题规模很大的情况下,我们可以选择适当的 T 使得这个复杂度在计算时间上是可以接受的。

4 实验及结果分析

实验 1:本实验随机生成 7 个作业,5 个资源节点,假设各个资源节点的带宽相等并且各个资源节点的任务队列为空,利用 Max – Min 算法、Min – Min 算法和改进的蚁群算法进行模拟实验,主要参数取值 $\rho_0 = 0.6$, $\rho = 0.8$, $\mu_0 = 0.9$, $b = 0.8$, $\alpha = 3$, $\beta = 4$,其它实验数据如下表 1 所示。

表 1　资源处理任务所需时间表

	R1	R2	R3	R4	R5
J1	4	8	5.3	10.6	9
J2	4.5	9	6	12	7
J3	1.5	3	2	4	6
J4	2.5	5	3.3	6	8
J5	5	10	6.7	13	7
J6	2.25	4.5	3	6	9
J7	9	3	6	8	5

表 2 任务分配表

	R1	R2	R3	R4	R5	最优值
Max – Min	J5, J6	J1, J7	J2	J4	J3	11.0
Min – Min	J1, J3, J6	J7	J4, J5		J2	10.0
HAC	J2, J6	J3, J7	J1	J4	J5	7.0

以表 1 中的数据为实验数据,分别用 Max – Min 算法、Min – Min 算法和本文 HAC 算法进行任务分配,最后得到的分配结果及最优值如表 2 所示。

从表 2 可知,混合蚁群算法分配结果为:R1 = {J2, J6},R2 = {J3, J7},R3 = {J1},R4 = {J4},R5 = {J5},最优值为 7.0,分配结果表明混合蚁群算法能够得到比 Max – Min 算法和 Min – Min 算法较优的结果。

实验 2:本实验随机生成 8 个资源,100 个任务。一般情况下是任务多而资源少,因此我们将随机生成的资源固定不变,依次增加任务量模仿真实网格环境。本实验中主要参数取值 $\rho_0 = 0.6$, $\rho = 0.8$, $\mu_0 = 0.9$, $b = 0.8$, $\alpha = 3$, $\beta = 4$。实验中的负载平衡度计算公式:

$$\eta = 1 - \frac{MaxT - MinT}{TotalT}$$

MaxT 表示分配到资源节点上的最大任务量;MinT 表示分配到资源节点上的最小任务量;TotalT 表示所有已经分配到资源节点上的总任务量。实验结果如图 2 资源负载平衡图。

实验 1 和 2 表明混合蚁群算法不仅能够得到比 Max – Min 算法和 Min – Min 算法较优的结果,而且负载平衡度也较高,能够平衡各资源节点之间的负载,进一步说明了混合蚁群算法的信息素浓度能够实时的反映出资源节点的带宽、负载、服务质量和处理能力等因素,并指导蚂蚁选择资源。

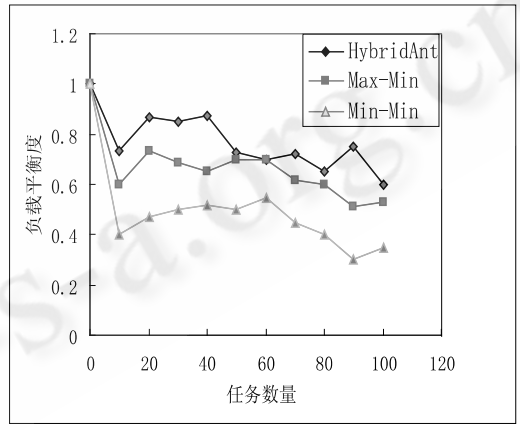

图 2　资源负载平衡图

Max – Min 算法和 Min – Min 算法存在着一定的缺点,在动态的环境下显得更加明显。Max – Min 算法的缺点是当任务队列中任务量大的任务非常少,而中等任务量的任务非常多的时候,就会导致任务量较大的任务占用高性能资源使得大量中等任务量的任务得不到合理的分配,从而出现了负载不平衡现象;而 Min – Min 算法同样会出现低 QoS 的任务占用了高 QoS 的资源,而且更严重的是任务量大的任务将排在任务量小的任务后面,使得并行执行任务的时间短,还可能导

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

致低 QoS 的资源一直处于空闲状态,这样使得高 QoS 资源的负载加重。本文提出的混合蚁群算法主要是通过信息素来判断资源节点的处理能力、带宽、和负载等综合情况,蚂蚁根据信息素的浓度按概率选择资源,综合能力强的资源节点选中的概率较大,大概率事件的发生使得资源节点的信息素浓度增强,为下一只蚂蚁选择资源节点提供帮助,这种正反馈的性质使得逐步的趋向最优解,而且小概率事件的发生能够保证蚂蚁的多样性,使得蚂蚁能够跳出局部最优的困境。因此,本文所提算法在大规模的任务调度中能够在允许的范围内得到最优解或近似最优解。

5 结束语

通过仿真实验及比较分析,证明了混合蚁群算法可以弥补其它一些经典任务调度算法的不足,能够在满足 QoS 的情况下,尽量的使得各资源节点负载平衡。该算法有效地解决网格资源分配问题,是一种快速而有效的启发式调度算法。但同时不足之处在于获得信息素调整时的动态信息也需要一定的通信开销。进一步的工作是将该算法投入实际的网格系统中,根据实际的资源请求进行调度来验证该算法。

参考文献

1 Abraham A, Buyya R, Nath B. Nature's heuristics for scheduling jobs on computational grids. In: Proc. of the 8th Int'l Conf. on Advanced Computing and Communications (ADCOM 2000). New Delhi: Tata McGraw-Hill Publishing, 2000. 45-52.

2 Dong F, G. Akl S. Scheduling algorithms for grid computing: state of the art and open problems. Technical Report, 2006.

3 吕桦. 网格环境下自适应任务调度方法的研究. 广西大学,2005. 6.

4 李季. 网格资源定位和任务调度的研究[D]. 重庆大学,2005. 12.

5 亓旭光. 基于蚁群算法的网格资源分配与调度研究. 广西大学,2006. 6.

6 Hui Yan, Xue-Qin Shen, Xing Li, Ming-Hui Wu. An improved ant algorithm for job scheduling in grid computing. 20005.

7 段海滨. 蚁群算法原理及详解. 科学出版社, 2005. 12.

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn