文章编号:1001 - 9081(2008)06 - 1598 - 03

基于蚁群优化算法的网格任务映射策略

谭一鸣,张 苗,张德贤
(河南工业大学 信息科学与工程学院, 郑州 450001)
(god_brother@ sohu. com)

摘 要:针对网格环境下实现任务最优映射的问题,提出一种基于蚁群优化算法的网格任务映射策略(ACO-GTM)。该算法通过人工蚂蚁在构建图上行走构建初始解,利用最优改进 2-选择局部搜索方法对初始解进行局部优化,并采用全局信息素更新与局部信息素更新相结合的信息素更新策略。最后通过实验与其他算法进行比较,表明所提出的映射算法在最优跨度和负载平衡方面具有明显的优越性。

关键词:网格计算;任务映射;蚁群优化算法;局部搜索
中图分类号: TP393 **文献标志码**:A

Tasks mapping in grid computing environment based on ACO algorithm

TAN Yi-ming, ZHANG Miao, ZHANG De-xian
(*College of Information Science and Technology, Henan University of Technology, Zhengzhou Henan* 450001, *China*)

Abstract: In order to optimize the tasks mapping in grid, a grid tasks mapping algorithm based on Ant Colony Optimization (named ACO-GTM) was proposed. The algorithm generated initial solutions through these artificial ants traversed on the construction graph and optimized these initial solutions by using the Best-improvement 2-opt local search algorithm. It combined the global and local pheromone updates. The experiments show that the proposed algorithm for the mapping problem has better performance than other algorithms on optimum makespan and load-balancing.

Key words: grid computing; tasks mapping; ACO; local search

0 引言

网格计算中,任务映射是批处理调度系统和静态调度系统的关键技术,是任务调度的基础。静态启发式任务映射算法具有充足的时间利用启发式信息,从而能够提高算法的调度性能,并提高计算资源的利用率和程序的执行效率。网格多机环境下的任务映射是一个典型的组合优化问题,以最小化任务的执行跨度为目标,同时兼顾负载平衡。

目前,国内外已经做了大量网格任务映射算法的研究。Min-Min 算法[1]是研究网格任务映射算法的基础。但是 Min-Min 算法总是优先调度短任务,容易造成系统负载的严重不平衡。遗传算法(Genetic Algorithms, GA)[2]具有快速随机的全局搜索能力,但是不能很好地利用反馈信息,求精确解效率低。

蚁群优化(Ant Colony Optimization, ACO)算法是一种针对难解的离散优化问题的元启发式算法,它利用一群人工蚂蚁的协作来寻找好的解。ACO 在学术研究和实际应用等领域都非常成功地解决了多种组合优化问题。本文设计了一种基于 ACO 的任务映射算法(Ant Colony Optimization-Grid Task Mapping, ACO-GTM)。该算法维持了对解开发和探索的平衡,并采用了局部搜索算法对构建解进行优化。仿真结果证明其性能优于其他算法。

1 问题描述

网格任务映射算法可描述为 n 个任务到 m 个计算节点的满射(即 $m < n$)。目标是实现所有节点的执行跨度最小[3],即 $\min\{\max\{rt_1,\cdots,rt_n\}\}$。其中 rt_j 表示节点 r_j 上所有任务的执行时间。

1.1 构建图

应用 ACO 元启发式算法解决问题,首要的步骤就是把问题映射到构建图 $G_C = (C,L)$ 上。其中 C 是成分的集合,包含了所有的任务和计算节点;而 L 是完全连接图的边的集合。构建过程中,蚂蚁在构建图上行走,行走的每一步都对应着把任务映射到节点上的动作。一个可行解就是一个由 n 个任务与节点对(a_i,r_j) 组成的集合。

1.2 约束

规定此算法遵守下列约束:1) 每个任务只映射一个节点;2) 所有任务必须全部映射;3) 每个任务在各个不同节点上的期望执行时间已知;4) 每个任务的生命周期的最后期限已知。

1.3 信息素和启发式信息的定义

信息素和启发式信息的定义至关重要,在算法设计时,如果选择了低劣的方案,将导致算法出现低劣的性能。在此我们定义信息素 τ_{ij} 指的是映射任务 a_i 映射到节点 r_j 上的期望度,启发式信息 η_{ij} 代表将任务 a_i 映射到节点 r_j 上的启发式期望度。如式(1) 所示,令:

$$令\ \eta_{ij} = w_1 \cdot \eta_{ij}^1 + w_2 \cdot \eta_{ij}^2 \tag{1}$$

其中 η_{ij}^1 是基于任务期望执行时间的启发式信息,令 $\eta_{ij}^1 = 1/ar_{ij}$,其中 ar_{ij} 表示任务 a_i 在节点 r_j 上的期望执行时间,w_1 是启发式信息 η_{ij}^1 的权重系数。η_{ij}^2 是基于最迟开始时间启发式信息(nLST)[4] 的规格化形式:$\eta_{ij}^2 = \max\limits_{l\in N} LS_l - LS_i + 1$。其中 LS_i 是任务 a_i 在节点 r_j 上的最迟开始时间,可以在映射之前用任

收稿日期:2007 - 12 - 19;**修回日期**:2008 - 02 - 18。 **基金项目**:河南省科技攻关项目(0524220042)。

作者简介:谭一鸣(1982 -),男,河南偃师市人,硕士研究生,主要研究方向:网格计算; 张苗 (1979 -),女,河南尉氏县人,讲师,硕士研究生,主要研究方向:模式识别、文本分类; 张德贤 (1961 -),男,河南新密人,教授,博士,主要研究方向:计算机智能技术。

务的最后期限与任务期望执行时间计算得出。而 N 是给定一个部分映射后符合条件的任务集合。nLST 使用规格化的原因在于,如果在生成任务队列的最后阶段,只有很少任务符合条件,那么任务的开始时间之间的相对差异会变得很小,而通过使用决定开始时间与最大最迟开始时间的差值来规格化绝对开始时间的值的方法,可以减少这个问题造成的影响。

2　基于 ACO 的任务映射算法设计

2.1　解的构建

在每一步的构建中,蚂蚁 k 通过利用伪随机比例规则把一个任务 a_i 映射到计算节点 r_j 上。规则如式(2):

$$j=\begin{cases}\arg\max\limits_{l\in N_i^k}\{\tau_{il}[\eta_{il}]^\beta\}, & q\leqslant q_0\\ J, & \text{其他}\end{cases}\tag{2}$$

其中 N_i^k 代表了任务 a_i 可选择的节点集合,q 是均匀分布在区间[0,1]中的一个随机变量,$q_0(0\leqslant q_0\leqslant 1)$ 是一个参数,J 是根据式(3)给出的概率分布产生出来的一个随机变量(其中 $\alpha=1$)。如式(3)所示:

$$p_{ij}^k=\frac{[\tau_{ij}][\eta_{ij}]^\beta}{\sum\limits_{l\in N_i^k}[\tau_{il}][\eta_{il}]^\beta};\ j\in N_i^k\tag{3}$$

节点集合 N_i^k 引入了静态候选列表[5]机制,通过限制 N_i^k 中节点的数目来实现,N_i^k 中存放的是根据任务在不同节点的期望执行时间为启发式信息评定出的 f 个节点,其数目为 $f(f\leqslant m)$。

2.2　局部搜索

在众多 NP-难的组合优化问题的应用中,当 ACO 算法与局部搜索相结合时,表现出了优秀的性能。局部搜索实际上是 ACO 元启发式算法的一种特殊的后台动作。

此算法中,当蚂蚁构建出完整路径后,通过使用最优改进 2-选择局部搜索方法[6]交换 r_i 和 r_j 上的任务以求得更好的解。这些经局部优化的解将在信息素更新步骤中使用。

2.3　信息素更新

当采用局部搜索算法对初始解进行局部优化后,此算法采用了全局信息素更新与局部信息素更新相结合的策略。

1)全局信息素更新

在每一次迭代后,至今最优蚂蚁构建的解(即具有最优跨度的方案)被允许释放信息素。更新规则如式(4):

$$\tau_{ij}\leftarrow(1-\rho)\tau_{ij}+\rho\Delta\tau_{ij}^{bs};\ \forall(i,j)\in T^{bs}\tag{4}$$

其中 $\Delta\tau_{ij}^{bs}=1/C^{bs}$,$C^{bs}$ 代表最优方案的跨度值,参数 ρ 代表信息素蒸发速率。

2) 局部信息素更新

在路径构建过程中,蚂蚁每经过一条边 (a_i,r_j),都将立刻调用这条规则更新该边上的信息素,更新规则如式(5):

$$r_{ij}=(1-\xi)\tau_{ij}+\xi\cdot\tau_0\tag{5}$$

其中 ξ 和 τ_0 是两个参数,ξ 满足 $0<\xi<1$,τ_0 是信息素的初始值。

3　算法实现

3.1　数据结构

为了能有效实现算法,一些额外的数据结构是必须的。这些数据可能是冗余的,但能提高算法的执行速度。

1)期望执行时间链表。不同任务在不同节点上有不同的计算时间,如果在构建解的过程中再计算,显然会降低算法的性能。因此,为每个任务设置一个五维链表,链表节点结构如图 1 所示。其中 j 是计算节点编号,ar_{ij} 是任务 a_i 在节点 r_j 上的执行时间,$1/ar_{ij}$ 是为了方便计算启发式信息 η_{ij}^1 而设置的,可以提高算法的效率。链表按照 ar_{ij} 值的非降序排列。通过此链表可以实现静态候选列表。

Pre	j	ar_{ij}	$1/ar_{ij}$	next

图 1　执行时间链表

2)信息素值链表。同时为每个任务 a_i 设置一个信息素链表,存储该任务映射到不同节点 r_j 的信息素值 τ_{ij}。节点结构如图 2 所示。链表按照信息值的非递增顺序排列。

Pre	j	τ_{ij}	next

图 2　信息素值链表

3)启发式信息值链表。在路径构建中,η_{ij}^β 的值在整个算法中都没有改变,所以保存它可以避免多次迭代计算。因此为每个任务设置一个链表,存放该任务映射到 j 节点的启发式信息值的 β 次方,来提高算法的执行效率。链表结构如图3所示。

Pre	j	η_{ij}^β	next

图 3　启发式信息值链表

3.2　映射算法

当到达任务调度器的任务达到一定数量时,就进行一次映射。已经达到、并准备进行映射的所有任务称为一个调度集合(Scheduling Set,SS)。任务映射算法的实现主要包括解的构建、信息素管理以及一些附加技术(例如局部搜索等)。此外,程序需要初始化一系列的数据结构和参数,并维护算法执行期间的统计信息。该映射算法的伪码描述如下:

```
Procedure ACO-GTM          //基于 ACO 的网格任务映射(GTM)
  Procedure InitializeData
    InitializeData();
  /*初始化每个任务的执行时间链表、信息素值链表、启发式信息值链表,算法参数等.*/
  End_Procedure InitializeData
  while(not terminate) do
                    //算法的终止条件是迭代次数达到最大值
    Procedure Constructions
      ant[k].visited[i]←false;
                     //把构建图中所有节点设置为"未访问"
      step←1;
      ant[k].tour[step]←r;    //把蚂蚁随机设置一个开始节点
      ant[k].visited[r]←true;    //把开始节点设置为"已访问"
      if q < q0
        selection-BestNext();
                    //选择具有最大 值的节点作为映射对象
      else
        selection-candidatelist();
                     //从候选列表选择下一个访问的节点
      ant[k].makespan←ComputeMakespan;
                     //计算每只蚂蚁构建解的跨度
  End_Procedure Constructions
  Procedure LocalSearch
    s'←LocalSearch(s); //应用最优改进 2-选择局部搜索算法
    If f(s') ≤f(s)          //如果修改的解比当前解好,保存
      修改解
      s^best←s';
  End_Procedure LocalSearch
  Procedure LocalPheromone Update
    LocalUpdate(i,j);
          //每当一只蚂蚁移动到下一个节点就调用此程序
```

```
    End_Procedure LocalPheromone Update
    Procedure GlobalPheromone Update
      GlobalUpdate();              //更新至今最优跨度解的信息素
    End_Procedure GlobalPheromone Update
End_Procedure ACO-GTM
```

4 实验结果与分析

为评估基于ACO-GTM算法的性能,与Min-Min算法、GA算法进行了比较。在同类算法中,Min-Min算法具有较好的性能,常用作映射算法的评测基准。其思想是:尽量把多的任务分配到执行它的速度最快并能最早完成它的机器上,当任务请求队列有多个任务请求时,首先给计算速度最快的节点分配任务。GA算法是一种通过模拟自然进化过程解决最优化问题的计算模型。并行性和全局解空间搜索是GA的两个显著特点,非常适合求解NP问题,在网格任务映射方面也有很好的性能。

本文采用GridSim[7]模拟和实现了以上三种算法。GridSim为网格有效资源分配技术的研究提供一个模拟环境,基础模拟环境是网格计算平台。它能够模拟异构且分布的资源、简单的网络、资源的查找、任务的虚拟处理等。GridSim采用面向对象技术构建,整个框架比较清晰,且开发应用也较简单。我们模拟了一个由5个异构节点构成的实验环境,任务数为30个,蚂蚁数30只,迭代50次。最后将得到的最优解与Min-Min算法,GA算法的解进行比较。ACO-GTM实验参数的设置如下:$\alpha = 1$; $\beta = 2$; $\rho = 0.1$; $\tau_0 = 1/C^{nm}$, $C^{nm} = \sum_{i=1}^{n}\sum_{i=1}^{m} ar_{ij}/m$; $q = 0.1$; $q_0 = 0.98$。实验结果如图4、5所示。

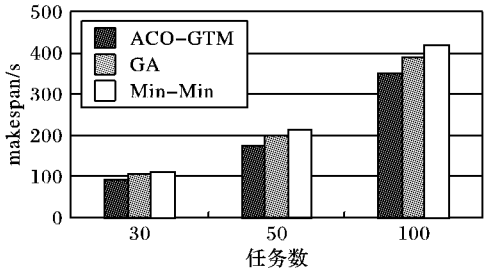

图4 任务量和任务完成时间

由实验测试结果可以得出下面的结论:1)如图4所示,此模拟网格计算平台下完成的3组映射实验中,ACO-GTM算法的执行跨度均小于GA和Min-Min算法,特别是当任务量较大时,ACO-GTM算法性能明显优于GA和Min-Min算法;2)如图5所示,ACO-GTM算法有较好的负载平衡效果,跨度最小的节点4与跨度最大的节点3相差只有9 s;GA和Min-Min算法的负载波动较大,GA算法跨度最小的节点4与跨度最大的节点2相差13 s;Min-Min算法跨度最小的节点5与跨度最大的节点2相差48 s。

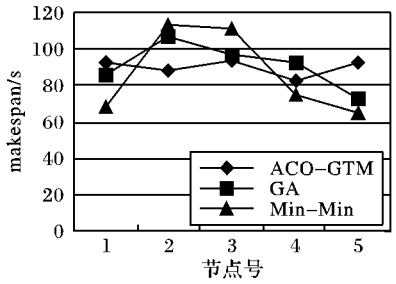

图5 节点负载

实验结果分析:由于ACO-GTM结合了局部搜索算法,对初始解进行了局部优化,大大提高了解的质量,不仅使各节点跨度降到最优或近优,而且使各节点的负载趋于平衡。

5 结语

本文提出了一种基于蚁群优化算法的静态网格任务映射算法,讨论了使用蚁群优化算法解决网格环境中多个子任务在异构资源之间的映射问题,并通过三组实验证明了该算法优于GA算法和Min-Min算法。

为了集中研究任务映射算法,本文忽略了任务传输时间,节点失效等若干因素。因此下一步将在考虑其他因素的情况下进一步对任务映射进行研究。另外,本文并没有涉及在任务映射之后的任务调度算法的内容,所以在此映射算法基础上设计怎样的任务调度算法才能使整个任务调度性能最佳,将成为我们下一步的主要研究内容。

参考文献:

[1] 魏天宇,曾文华,黄宝边. 基于Min-Min改进后的网格调度算法[J]. 计算机应用, 2005,25(5):1190-1192.

[2] 林剑柠,吴慧中. 基于遗传算法的网格资源调度算法[J]. 计算机研究与发展, 2004,41(12):2195-2199.

[3] 亓旭光,梁正友. 基于蚁群算法的网格资源分配与调度研究[J]. 广西民族学院学报: 自然科学版,2006,12(2):83-86.

[4] 张军,胡晓敏,罗旭耀. 蚁群优化[M]. 北京: 清华大学出版社, 2007:171-173.

[5] 金华征,程浩忠,奚珣,等. 贪婪随机自适应搜索法在电网规划中的应用[J]. 上海交通大学学报, 2006,40(4):563-567.

[6] 邹鹏,周智,江贺,等. 求解旅行商问题的循环局部搜索算法的运行时间和性能分布分析[J]. 计算机学报, 2006,29(01):92-99.

[7] BUYYA R, MURSHED M. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing [J]. Concurrency and Computation: Practice and Experience, 2002, 14(13-15): 1175-1220.

(上接第1529页)

参考文献:

[1] VINCENT L, SOILLE P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991, 13(6): 583-598.

[2] HARIS K, EFSTRATIADIS S. Hybrid image segmentation using watersheds and fast region merging [J]. IEEE Transaction on Image Processing, 1998, 7(12):1684-1699.

[3] GAUCH J M. Image segmentation and analysis via multi-scale gradient watershed hierarchies [J]. IEEE Transaction on Image Processing, 1999, 8(1): 69-79.

[4] KIM J B, KIM H J. Multiresolution-based watersheds for efficient image segmentation [J]. Pattern Recognition Letters, 2003,24(1): 473-488.

[5] HIEU T, MARCEL M, REIN V. Watersnakes: Energy-driven watershed segmentation [J]. IEEE Transaction on Pattern Analysis and Machine Intelligence, 2003, 25(3): 330-342.

[6] PERONA P, MALIK J. Scale-space and edge detection using anisotropic diffusion [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990,12(7):629-639.

[7] 王蜀,李永宁,陈楷民,等. 基于数学形态学的医学图像分割[J]. 计算机应用, 2005,25(10): 2381-2382.

[8] 卢官明. 一种计算图像形态梯度的多尺度算法[J]. 中国图象图形学报, 2001, 6A(3):214-218.