一种用于网格的启发性智能调度策略

傅 明[1],刘凯雄[1],肖 静[2]
(1.长沙理工大学 计算机与通信工程学院,湖南 长沙 410076;
2.华中师范大学 计算机科学系,湖北 武汉 430079)

摘 要:任务调度是计算网格系统中极其关键的一部分,一种好的调度方法可以极大地提高整个系统的性能。针对蚂蚁算法在网格调度中早期信息素匮乏和蚂蚁分工单一的缺陷,提出了一种新的启发性智能调度方法。在调度过程前期,采用遗传算法为各网格节点生成丰富的信息素,作为调度中心进行任务调度的依据,然后在多群蚂蚁算法中,各种群的蚂蚁根据分工的不同在属于自己的空间中寻找最优解,从而缩小了搜索规模,加快了收敛速度,优化了调度性能。

关键词:任务调度;信息素;并行遗传算法;多群蚂蚁算法;负载平衡
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2006)11-0119-03

A Strategy of Heuristic Intelligent Scheduling Applied in Grid

FU Ming[1], LIU Kai-xiong[1], XIAO Jing[2]
(1. College of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410076, China;
2. Department of Computer Science, Central China Normal University, Wuhan 430079, China)

Abstract: Job scheduling is a key part in the computing grid system, a good scheduling method could enhance the performance of entire system. In this paper, a new heuristic intelligent scheduling method is proposed, which could solve the drawback of ant algorithm's pheromone lack when applied in job scheduling in the early stage and all ants carry out the same task. At the beginning of job scheduling, create abundant pheromones for each network-node by using genetic algorithm. And the scheduling center performs job scheduling depending on these pheromones. Because of dividing the work, each colony's ant has different task, and they search for the most optimum solution from the multi colony ant algorithm. As a result, it reduces the range during do the searching job and accelerates the convergent speed. In other words, it optimizes the capability of job scheduling.

Key words: job scheduling; pheromone; parallel genetic algorithm; multi colony ant algorithm; load balancing

0 引 言

随着计算机网络技术和分布式计算技术的发展,网格计算已经成为高性能计算领域的一个新的研究热点,也是并行和分布处理技术的一个发展方向。通过对网格的研究,取得了许多比较成功的研究成果,但仍然有很多问题没有得到很好解决,资源管理中的调度问题就是其中之一。作为网格系统中一个极其重要的组成部分——任务调度系统,它需要根据不同任务信息采用适当的策略把不同的任务分配到相应的资源节点上去运行。由于网格系统特有的特性,不同的调度策略将会对不同任务的执行时间、花费代价以及整个网格系统的吞吐量产生很大的影响,因此,必须探讨出具有良好性能的调度方法。

许多相关学者已经对资源调度算法做了大量研究,Buyya 提出了一种基于应用经济模型的优化调度模型[1],通过在资源的拥有者和使用者之间建立一种"交易",让资源使用者花费最低的代价获得完成计算任务所需要的资源;Vincenzo 介绍了一种基于遗传算法的资源调度算法[2,3],其目的是为了尽可能地提高资源的使用率和系统吞吐量。另外 Abraham 等人介绍了模拟退火等进化算法在网格资源调度中的应用[4],Xu 等人介绍了蚂蚁算法[5]在网格调度中的应用。

传统的蚂蚁算法是通过信息素的累计和更新收敛于最优路径上,具有分布式并行全局搜索能力。但初期信息素匮乏,各蚂蚁分工单一,求解速度慢。文中提出一种启发性调度策略,采用遗传算法生成丰富的信息素分布,利用多群蚂蚁算法求精确解,优化调度性能。

1 网格计算任务调度的特性

在网格计算中,任务调度的实质就是将 n 个相互独立的任务根据某种特定的原则分配到 m 个异构可用资源上,使得总任务的完成时间最短以及资源利用率最高。定义 T_i 为任务 i 的最后完成时间,定义跨度为 $T_{max}=$

收稿日期:2006-01-18
作者简介:傅 明(1961-),男,湖南长沙人,教授,博士后,硕士研究生导师,从事网格计算、基于网络的应用和数据挖掘等研究。

$\max\{T_i, i = 1,2,\cdots,n\}$，则调度任务就是在 2^m 个可能的资源子集空间中寻找最优集合使得跨度 T_{max} 和 $\sum_{i=1}^{n} T_i$ 最小，可见，网格计算的任务调度是一个 NP 完全问题[6]。

网格计算任务调度具有以下特性：

(1)网格任务调度是分布式的。网格中的资源在地理上分布，在规模最大的全球性网格中，资源分布在全世界，要实现一种全局的统一集中式的调度是不可能的，必须采取一种并行分布式的策略来进行任务的管理与调度。

(2)网格任务调度策略必须具有自治性。网格中的资源可以被网格作业和资源所属节点的内部任务所利用，网格调度系统不能干涉节点内部的调度策略。

(3)网格任务调度是面向异构平台的。由于网格系统包括的资源是连接在 Internet 上的所有节点上的资源，包括各类主机、工作站甚至 PC 机，可能使用不同硬件和软件，它们都是异构的，因此，任务调度必须面向异构平台。

(4)网格任务调度具有动态可扩展性。网格中的资源不仅是异构的而且是动态改变的。随时都有新的资源加入网格，也有资源随着节点的离开或因资源故障而消失于网格系统。调度系统必须从不断变化的可利用资源中选择出最佳资源以满足用户的需要。随着网格资源的不断增加和应用任务的增长，任务调度必须具有可扩展性才不至于降低这个网格系统的性能。

2 遗传算法(Genetic Algorithm)

文中采用并行遗传算法中的粗粒度模型，称之为岛屿群体模型(Island model PGA)。该模型在每个处理机上子群体所含个体数大于 1，子种群形同一岛屿，大部分时间孤立地在其各自处理机上并行独立地运行简单遗传算法(与网格上资源的分布式存储相符合)，且以随机的时间间隔，在随机选择的处理机之间交换个体信息。如图 1 所示。

不同子群通过赋予其不同的控制来实现不同的搜索目的。各子群间通过迁移系数交流信息完成子群的协同进化，利用人工选择系数保存子群中的最优个体。在完成一代进化后，将源子群中的最优个体替代目标子群中的最差个体。在整个进化过程完成后，对每一代进化中各个子群中的最优个体进行排序，最终得到最优解。

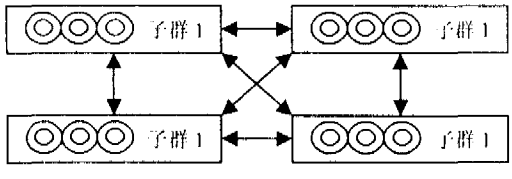

图 1 粗粒度并行遗传算法

3 蚂蚁算法(Ant Algorithm)

蚂蚁算法[7]最开始由意大利人 Dorigo 等人提出，它和模拟退火算法等都属于启发式方法。蚂蚁算法固有的并发性和可扩展性，使得它非常适合用于网格计算中的任务调度。网格上异构且动态的资源时刻不停地在变化，而可扩展性就是在原有一个规模 n 的问题上求出最优解后，再增加 m 个节点，可在原有解的基础上快速找到该问题在规模为 $m + n$ 上的最优解[8]。正是由于蚂蚁算法具有并发性和可扩展性，所以在某一具体时间，网格上所有可利用的资源仅用信息素来描述，这样调度程序就可以根据当前的信息素快捷地获得最优解。同时，网格任务调度也属于 NP 完全问题，大量实验证明，蚂蚁算法在解决 NP 完全问题上也有着极其优越的性能。

但是，现有算法在执行过程中，蚂蚁在选择下一节点的过程中以转移概率为依据，在所有剩下的节点中选择下一个，搜索空间规模过大，影响求解速度。文中提出了一种多群蚂蚁算法：根据实际生活中分工的不同，将蚂蚁分为不同的种群，不同种群的蚂蚁在觅食过程中所承担的任务不一样，每只蚂蚁只在属于自己的空间中寻找最优解，从而大大缩小了搜索空间规模，加快了收敛速度。

4 启发性智能调度模型

网格调度系统中的调度器将用户提交的需求清单中的内容解析出来，通过网格信息服务收集网络上现时可用的资源，经过一系列的筛选，使得只有一部分合适的资源被选中用来被服务所使用，用并行遗传算法生成被选中资源集合中的初始信息素分布，再采用蚂蚁算法搜索出精确解。图 2 为启发性智能调度框架。

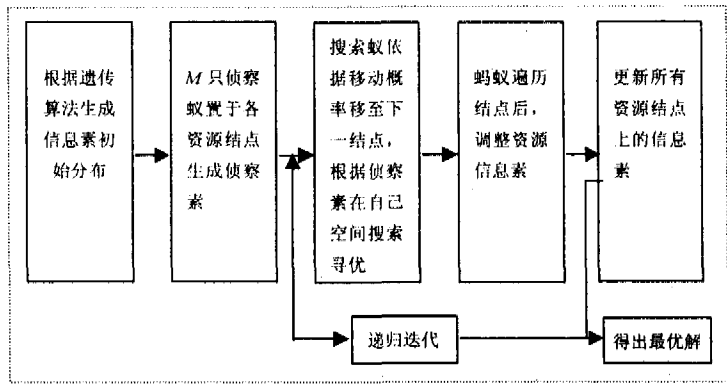

图 2 启发性智能调度框架

4.1 初始化信息素分布

网格调度中资源分配建模如下：

$$\min \sum_{i=1}^{n} f_i(x_i) \tag{1}$$

$$\text{s.t.} \sum g_i(x_i) \leqslant r_0 \tag{2}$$

$$h_i(x_i) \leqslant r_i \quad i = 1,2,\cdots,n \tag{3}$$

在上式中，r_0 表示网格系统中的公共资源，是为各个子系统占有的资源。(1) 式为求解的目标函数，(2) 式表示耦合约束，(3) 式表示每个子问题约束[9]。

通过资源预选，将选出 2^{n-1} 个可用的资源，每一个资源表示一个个体。由于网格环境本身的复杂性和网格资源

的分布性和异构性,资源可能由于某种故障或其他原因造成当前不可以被利用,对此设置一个信息标识位 flag 表示其当前可用状态。采用 n 位二进制数对资源进行编码,可以表示 2^{n-1} 个资源的完整信息。设适应度函数和目标函数一致,即:

$$F_i = \min\sum_{i=1}^{n} f_i(x_i)$$

对每个个体计算出它的适应度值,选择生存能力强的个体进行交配产生新的子染色体。在交配的过程中采用单点交叉,即随机产生一个整数(大于 1 而小于 n)作为交叉点,将该点及该点以后的部分交叉,得到两个新的染色体取代双亲。变异操作用来产生新的个体,确保发现最佳方案的概率永不为零且可以抑制早熟,可以取固定变异概率。变异过程中检测信息标识位,将不可用的资源清除掉,提高效率。

4.2　求全局最优解

文献[10]中分别设计了一种多态蚂蚁算法。文中的算法设计如下:在网格调度系统中将蚂蚁分为 3 个种群,即侦察蚁、搜索蚁和工蚁。侦察蚁负责局部侦察,当搜索蚁搜索到该资源节点时,侦察蚁对其提供侦察结果侦察素作为搜索蚁转移到下一个节点的一部分依据;搜索蚁则根据信息素和侦察素实行全局搜索,寻找最优解;工蚁和蚂蚁在实际觅食过程中一样,只负责搬运食物回蚁穴,在搜索全局最优解时不承担任何工作。

侦察蚁和搜索蚁的具体工作如下:

为每个资源节点创建一只侦察蚂蚁,以所在资源为中心侦察周围($m-1$)个资源,将侦察结果与可选资源节点 Usable(如同文献[11]中的 MAXPC)生成侦察素 $s[i][j]$,表示从资源 i 到资源 j 的信息:

$$s[i][j] = \begin{cases} \dfrac{d_{\min_i}}{d_{ij}}, & \text{节点 } j \text{ 属于节点 } i \text{ 的 Usable 集} \\ 0, & \text{其他} \end{cases} \tag{4}$$

(4) 式中 d_{ij} 表示以资源 i 节点为中心的第 j 个节点完成任务所需要付出的代价,$d_{\min_i}$ 表示以资源 i 节点为中心的第 j 个节点完成任务所需要付出的最小代价。生成的侦察素可以为搜索蚁的全局搜索提供辅助信息。

根据当前资源的信息素浓度和目标函数,得出任务分配给其他在线资源的概率[5]:

$$P_j^k(t) = \begin{cases} \dfrac{[\tau_j(t)]^\alpha \times [\eta_j]^\beta}{\sum_u [\tau_u(t)]^\alpha \times [\eta_u]^\beta} & j,u \in \text{在线资源} \\ 0, & \text{其它} \end{cases} \tag{5}$$

(5) 式中 $\tau_j(t)$ 为在时刻 t 资源 j 的信息素值,η_j 表示资源 j 先天的计算能力,即 $\eta_j = \tau_j(0)$,η_u 表示资源 u 先天的计算能力,即 $\eta_u = \tau_u(0)$。$\alpha(\alpha > 0)$ 表示资源信息素的权值,$\beta(\beta > 0)$ 表示资源先天的权值。

由初始信息素分布可以得知资源本身的固有属性计算能力和通信能力,搜索蚁根据转移概率转移到下一个可利用的资源上。在到达一个新的资源节点时,搜索蚁根据侦察蚁提供的侦察素,在一个相对较小的搜索空间中寻找最优解,大大缩小了搜索规模,加快了收敛速度。

当完成一次搜索时,结合侦察素对个资源节点的信息素做一定的修改:

$$\tau_j^{new} = \begin{cases} \rho \times \tau_j^{old} + \Delta\tau_j, & \text{当 } s[i][j] \neq 0 \\ p \times \tau_j^{old}, & \text{其他} \end{cases} \tag{6}$$

(6) 式中 $\Delta\tau_j = \cdot k$,k 表示分配至该资源上任务的计算量和通信量,$\rho(0 < \rho < 1)$表示信息素的持久度。搜索蚁判断侦察素是否为零,仅在可能为最优解的节点资源上增加信息素。这样,调度系统就可以根据系统当前的运行状况客观、准确适时地调整资源信息素。各种群的蚂蚁根据自己的分工,互相协作,共同完成任务。

对每次任务的分配结果计算出目标函数值,选出此次最优解,重复迭代,最终得到全局最优解。为保证得到全局最优解,不让算法过早收敛,可以将信息素设置在一个区间 $[\tau_{max}, \tau_{min}]$ 上,当信息素值超出这个范围时,强行设为上限或者下限。

4.3　负载平衡

由于网络中的资源由其信息素来描述,信息素多的资源将受到更多调度任务的青睐。当网格中有许多任务都调用系统中信息素最多的资源时,该资源的负载就会异常的沉重,这就有可能使得分配到该资源的任务不能够按时完成。各任务能否成功完成又直接影响到资源的负载情况,从而更进一步影响到整个网格系统的任务调度。为此,文中根据分配到某资源的任务实际完成的情况来调整资源的负载,任务完成率高的资源的信息素得到增加,说明资源可以被更多其他的任务所调用,而任务完成率低的资源的信息素将减少,因为它当前的负载比较重,可以让负载相对较轻的资源满足更多任务的需要,从而优化了整个系统的负载平衡性能。

5　结束语

针对应用传统的蚂蚁算法解决任务调度问题出现的早期信息素匮乏和蚂蚁执行同一任务的缺陷,提出一种启发式智能调度方法,将遗传算法与蚂蚁算法相结合,丰富其早期信息素分布,不同种群的蚂蚁执行不同的任务且种群间可以互相协助工作,缩小了求解空间的规模,加快求解速度且兼顾了系统的负载平衡,适合于计算网格中的任务调度。启发性智能调度方法是网格计算中的一个重要研究领域。目前,启发性任务调度算法还处于模拟仿真阶段,还需要在理论和实践中不断完善。

参考文献:

[1]　Buyya, Abramson D, Giddy J. An economy driven resource management architecture for global computational power grids

(下转第 124 页)

```
public static void main(String[] args) {
  try {
Integer i = new Integer(4);
    Integer j = new Integer(5);
    String endpoint = "http://localhost/MyWebService/Service1.asmx";
    Service service = new Service();
    Call call = (Call) service.createCall();
    call.setTargetEndpointAddress(new java.net.URL(endpoint));
    //创建一个服务,传递 WSDL 位置和想要调用的服务名称作为参数。
    call.setOperationName(new QName("http://tempuri.org/SU","IntAdd"));
    //调用 Web 服务所提供的方法
    call.addParameter("a", org.apache.axis.encoding.XMLType.XSD_DATE,
    javax.xml.rpc.ParameterMode.IN);
    call.addParameter("b", org.apache.axis.encoding.XMLType.XSD_DATE,
    javax.xml.rpc.ParameterMode.IN);
    call.setReturnType(org.apache.axis.encoding.XMLType.XSD_INT);
    //传递参数,并设置参数的类型
    call.setUseSOAPAction(true);
    call.setSOAPActionURI("http://tempuri.org/Rpc");
    //设置 http 标头字段
    Integer k = (Integer) call.invoke(new Object[]{i, j});
    //通过 Call.Invoke()方法调用服务。
    System.out.println("result is " + k.toString() + ".");
    }
    catch (Exception e) {
    System.err.println(e.toString()); }}}
```

程序将两个参数传递到 XML Web Service 通过 Web 方法 IntAdd 得出两数之和,并将结果返回。

4 总 结

XML Web Service 技术的出现,使得异构平台之间的通信更加方便、易用。当然在使用 XML Web Service 技术进行异构平台之间的通信时有以下几点是需要注意的:

(1)在提供 Web Services 的时候,尽量使用 XML schema 中支持的变量类型做参数。如果使用.Net 中的 dataset 这种类型,对于 Java 来说解析起来将是一个灾难,当然,理论上是可以解析的。但是从效率角度来说,在 Web Services 与客户端交换信息的过程中,始终有一个序列化和反序列化的问题。如果使用 dataset 这种类型,系统还需要对它进行序列化操作,这将是一个很耗费资源的过程。而使用 string 类型将简单很多。

(2)如果使用了 soap header 等扩展功能,例如使用了微软提供的 WSE 技术,它们之间的相互通信需要作如下特殊处理:

String endpoint = "http://localhost/MyWebService/Service1.asmx? wsdl";

总之,XML Web Service 作为一个崭新的面向服务的分布式计算模型,其应用具有巨大的潜力,Web Service 已成为计算机领域研究的热点之一。

参考文献:

[1] Freeman A, Jones A.. NET XML Web 服务程序设计[M]. 北京:清华大学出版社,2003:3-15.

[2] Kao J. 构建基于 XML 的 Web 服务开发者指南[S/OL]. 2001. http://geeclub.sun.com.cn/staticcontent/html/j2ee/J2EEWebServicesDev.HTML.

[3] W3C Note. Web Services Description Language (WSDL) 1.1 [EB/OL]. 2001-03-15. http://www.w3.org/TR/wsdl.

[4] Ferrara A, MacDonald M.. NET Web 服务编程[M]. 北京:清华大学出版社,2003:32-66.

[5] 刘 洋,魏 飞. 精通 JBoss—— EJB 与 Web Services 开发精解[M]. 北京:电子工业出版社,2004:300-330.

(上接第 121 页)

[C]//Int.l Conf on Parallel and Distributed Processing Techniques and Applications. Las Vegas:[s.n.],2000.

[2] Di Martino V. Scheduling in a grid computing environment using genetic algorithms[C]//Mililotti M. the 16th Int.l Parallel and Distributed Processing Symp(IPDPS2002). Florida, USA:[s.n.],2002.

[3] DiMartino V, Mililotti M. Sub-optimal scheduling in a grid using genetic algorithms[J]. Parallel Computing, 2004, 30(5/6):553-565.

[4] Abtaham A, Buyya R. Nature's heuristics for scheduling jobs on computational grids[C]//The 8th Int'l Conf on Advanced Computing and Communications (ADCOM2000). Cochin, India:[s.n.],2000.

[5] Xu Zhihong, Hou Xiangdan, Sun Jizhou. An algorithm-based task scheduling in grid computing. CCECE[C]. IEEE CCECE, 2003.

[6] 张颖峰,李毓麟. 基于进化算法的网格计算资源管理调度系统[J]. 计算机工程,2003,29(15):110-175.

[7] Dorigo M, Gambardella L M. Ant Colonied for the Travelling Salesman Problem[J]. Biosystems, 1997, 43(2):73-81.

[8] 侯向丹. 蚂蚁算法扩展性及应用研究[D]. 天津:河北工业大学,2002:15-16.

[9] 刘树安,尹 新,郑秉霖,等. TS 与 GAs 混合算法在大规模资源分配问题中的应用[J]. 控制与决策,1998,13(4):40-44.

[10] 徐精明,曹先彬,王煦法. 多态蚁群算法[J]. 中国科学技术大学学报,2005,35(1):62-68.

[11] 全惠云,文高进. 求解 TSP 的子空间遗传算法[J]. 数学理论与应用,2002,22(1):36-39.