网格计算中任务调度研究综述*

罗 红[1,2]，慕德俊[1]，邓智群[1]，王晓东[1,2]

(1. 西北工业大学 自动化学院，陕西 西安 710072；2. 空军工程大学 电讯工程学院，陕西 西安 710077)

摘 要：阐述了网格计算领域任务调度的特点和目标;综述了现有的任务调度技术和算法,包括启发性智能任务调度,基于 Agent 的任务调度,基于 Petri 网的任务调度,基于成本的任务调度等算法,以及任务调度的负载均衡问题,最后给出任务调度的研究展望。

关键词：网格计算；任务调度；算法；NP 完全

中图法分类号：TP393.1 **文献标识码**：A **文章编号**：1001-3695(2005)05-0016-04

A Review of Job Scheduling for Grid Computing

LUO Hong[1,2], MU De-jun[1], DENG Zhi-qun[1], WANG Xiao-dong[1,2]

(1. College of Automation, Northwestern Polytechnical University, Xi'an Shanxi 710072, China; 2. College of Telecommunication Engineering, Air Force Engineering University, Xi an Shanxi 710077, China)

Abstract: This paper addresses the characteristic and performance metric of job scheduling for grid computing, reviews existing job scheduling technologies and algorithms including heuristic job scheduling, Agent-based job scheduling, modeling and analysis using Petri net, cost-based job scheduling and load balancing, and finally makes an expectation about its future research directions.

Key words: Grid Computing; Job Scheduling; Algorithms; NP-complete

网格计算(Grid Computing)是当前互联网研究中的一个热点,也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。用户通过任务管理系统向网格提交任务、为任务指定所需资源、删除任务并监测任务的运行状态。用户提交的任务由任务调度系统按照任务的类型、所需资源、可用资源等情况安排运行日程和策略。而资源管理系统监测网格资源状况,收集任务运行时的资源占用数据等[1]。由于网格计算任务调度面临的是一个 NP 完全问题,它引起了众多学者的关注,成为目前网格计算研究领域的一个焦点[2~4]。

1 网格计算任务调度的特点和主要目标

在网格系统中,任务调度系统是其重要的组成部分,它要根据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。由于网格系统的异构性和动态性,以及运行于网格系统之中的应用程序对于资源的不同需求,使得任务调度变得极其复杂,不好的任务分配策略,将会增加任务的执行时间、降低整个网格系统的吞吐量。

1.1 网格计算任务调度的特点

网格计算任务调度系统具有以下几个特点:

(1) 任务调度是面向异构平台的。由于网格系统是由分布在 Internet 上的各类资源组成的,包括各类主机、工作站甚至 PC 机,它们是异构的,可运行在 UNIX, Windows NT 等各种操作系统下,也可以是上述机型的机群系统、大型存储设备、数据库或其他设备。因此网格系统中的任务调度必须面向异构平台,并在这些平台上实现网格任务的调度。

(2) 任务调度是大规模的、非集中式的。由于网格系统是一个大到整个 Internet 的分布式巨系统。要实现一种全局的统一集中的任务调度管理是根本不可能的。因此,网格的任务调度必须以分布、并行方式进行任务的管理与调度。

(3) 任务调度不干涉网格节点内部的调度策略。在网格系统中,各网格节点的内部调度策略是自治的,网格任务调度系统干预其内部的调度策略是没有必要的,也是不可能的。

(4) 任务调度必须具有可扩展性。网格系统初期的计算规模较小,随着超级计算机系统的不断加入,系统的计算规模也必将随之扩大。因此,在网格资源规模不断扩大、应用不断增长的情况下,网格系统的任务调度必须具有可扩展性,不致降低网格系统的性能。

(5) 任务调度能够动态自适应。网格中的资源不但是异构的而且网格的结构总是不停地改变:有的资源出现了故障,有的新资源要加入到网格中,有些资源重新开始工作等。总之网格的动态性是明显的,所以任务调度系统必须适应网格的这种动态性,从可利用的资源中选取最佳资源为用户提供应用服务。

1.2 网格计算任务调度的主要目标

简单地说,网格计算任务调度的目标就是要对用户提交的任务实现最优调度,并设法提高网格系统的总体吞吐率。具体的目标包括:最优跨度(Optimal Makespan)、服务质量 QoS(Quality of Service)、负载均衡(Load Balancing)、经济原则(E-

收稿日期：2004-04-02；修返日期：2004-06-21

基金项目：西北工业大学研究生创业种子基金(Z20030051)

conomic Principles) 等。

(1) 最优跨度。跨度是一个最主要、最常见的目标,指的是调度的长度,也就是从第一个任务开始运行到最后一个任务运行完毕所经历的时间。跨度越短说明调度策略越好。当用户向网格系统提交任务后,最大的愿望是网格系统尽快完成自己的任务。可见,实现最优跨度是用户和网格系统的共同目标。

(2) 服务质量 QoS。网格系统要为用户提供计算和存储服务时,用户对资源需求情况是通过 QoS 形式反映出来的。任务管理与调度系统在进行分配调度任务时,保障网格应用的 QoS 是完全应当的。

(3) 负载均衡。在开发并行和分布计算应用时,负载平衡是一个关键问题。网格系统更进一步扩展了这个问题。网格任务调度是涉及交叉域和大规模应用的调度。解决好系统的负载均衡是一个非常重要的问题。

(4) 经济原则。网格环境中的资源在地理上是广泛分布的,而且每个资源都归属于不同的组织,都有各自的资源管理机制和政策。根据现实生活中的市场经济原则,不同资源的使用费用也应是不相同的。市场经济驱动的资源管理与任务调度必须使消费双方(资源使用者和资源提供者)互惠互利,才能使网格系统长久地发展下去。

2　网格计算的任务调度研究

在网格计算中,任务调度的实质就是将 n 个相互独立的任务分配到 m 个异构可用资源上,使得总任务的完成时间最小以及资源得到充分利用。定义 F_i 为任务 i 的最后完成时间,定义跨度为 $F_{\max} = \max\{F_i, i = 1, \cdots, n\}$,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 $F_{\max}$ 和 $\sum F_i$ 最小,可见,网格计算的任务调度是一个 NP 完全问题[12]。

国际数学界研究 NP 问题的历史开始于 20 世纪 70 年代。1971 年,Cook 找到了第一个 NP 完全问题,寻求标准布尔方程解的问题(S 问题),开创了 NP 完全问题研究的历史。到目前为止,大约有几千个问题已被证明是 NP 完全问题。由于大量的 NP 完全问题至今没有找到有效算法,而指数型算法对解决规模较大的问题又十分困难,以求近似最优解为目的的启发性算法自然就受到了极大的重视[5]。近年来,人们提出了很多启发性智能化算法,诸如神经网络、模拟退火、禁忌搜索、遗传算法、蚂蚁算法等,它们毫无争议地成为解决 NP 问题及 TSP 问题的锐利工具[6]。

2.1　启发性智能任务调度

2.1.1　基于遗传算法 GA(Genetic Algorithms)的网格任务调度

遗传算法是将问题的求解表示成染色体(Chromosome),从而构成一群染色体。将它们置于问题的环境中,根据适者生存的原则,从中选择出适应环境的染色体进行复制,通过交叉、变异产生出新一代更适应环境的染色体群,这样一代一代地不断进化,最后收敛到一个最适合环境的个体上,求得问题的最优解[7]。一个标准的遗传算法有三个基本成分:①种群。也称为个体群。一个个体代表问题解空间的一个元素,遗传算法就是要在给定的时间里搜索到或进化出一个可以接受的解。一般个体用二进制数表示,称作染色体,其长度根据问题而定。②适应度函数。用适应度函数来评价问题解的好坏,它不受连续可微的约束,且定义域可以是任何函数。标准的遗传算法只用适应度函数作为依据,适应度值为大于 0 的实数,适应度值越大表示个体的生存能力越强。③遗传操作。包括选择、交叉和变异[8~10]。

染色体的设计是制作遗传算法的关键。好的编码计划会使适应度函数容易计算,并使遗传操作容易实现。文献[8]中,把染色体定义为一个两元组{*PE*, *Priority*}。PE(Processing Elements)中每个元素代表相应任务分配到的处理单元,*Priority* 是任务的优先权值(该文取值从 1 ~512),用来构成一个满足先后次序条件的优先权列表。此外,任务以它权重 *pri* 的升序存放。优先权列表由下列公式构成:

$$\begin{cases} pri(v_j) = priority(v_j) & \text{如果 } IPred(v_j) = \Phi \\ pri(v_j) = \max_i\{pri(v_i) \mid v_i \in IPred(v_j)\} + priority(v_j) & \text{其他} \end{cases}$$

一个任务的 *pri* 值越小,表示它的优先级越高。如果两个任务的 *pri* 值相同,一般假设早算的任务比后算的优先级高。

在初始化过程中要产生初始染色体。首先对于每一个任务,*PE* 值在[1.. $2^{|P|} - 1$]之间产生,其中 |P|为任务图的宽度(即任务所需处理单元 *PE* 个数的上界),并在[1..512]之间随机选择一个值作为其优先级,这样就获得了一个染色体。重复以上过程产生指定个数的染色体。

为了选择出好的染色体,定义适应度函数为

$$f(i) = \left(\frac{\sum_{i=1}^{pop\_size} (makespan(i) + AR(i))}{pop\_size \cdot (makespan(i) + AR(i))} \right)^2$$

其中染色体 i 代表一个调度。假设染色体的跨度 *Makespan* 是一个正整数。$AR(i)$ 是一个比 1 小但比 0 大的值。

为了计算染色体的跨度,定义如下的公式:

$$\begin{cases} FT(v_j, p) = \dfrac{C(v_j)}{comp(p)}, IPred(v_j) = \Phi \\ FT(v_j, p) = \max_i\{FT(v_i, q) + \dfrac{C(v_i)}{comp(p)} + \dfrac{M(v_i, v_j)}{bandwidth(p, q)}, FT(v_k, q) + \dfrac{C(v_k)}{comp(q)}\} \\ v_i, v_j, v_k \in V, v_i \in IPred(v_j), P \neq q, pri(v_k) < pri(v_i) \\ FT(v_j, p) = \max_i\{FT(v_i, p) + \dfrac{C(v_i)}{comp(p)}, FT(v_k, p) + \dfrac{C(v_k)}{comp(p)}\} \\ v_i, v_j, v_k \in V, v_i \in IPred(v_j), pri(v_k) < pri(v_i) \end{cases}$$

其中,$C(v_i)$ 表示节点 v_i 的计算成本;$IPred(v)$ 表示节点 v 的直接前驱;$comp(p)$ 表示处理单元 p 的运算能力;$bandwidth(p, q)$ 表示处理单元 p 与处理单元 q 之间的通信能力;$M(v_i, v_j)$ 表示任务 v_i 向任务 v_j 发送数据的通信成本。

在把以上公式用于调度以前,应该从所有已产生的调度中消除掉无效的任务复制。如果一个任务复制的直接后继能从其他副本中得到数据,那么这个任务复制就是无效的,意味着这些后继的开始时间会较早。为了消除无效任务复制,要做如下工作:

(1) 为所有任务和 *PE*,计算 $FT(v, p)$。

(2) 对每一个任务(以反序):如果 $0 < FT(v, q) \leq FT(v, p)$, $FT(w, p) = 0$, $< v, w > \in E$,那么删除 q 上的任务 v。

一旦计算出所有染色体的适应度值,就能通过赌轮法选择出好的染色体。具有较高适应度值的染色体有更多的机会被选中。

对于交叉操作,该文采用了最简单的交叉形式,叫做单点交叉。具体步骤是:首先随机选择两个染色体,然后在 1 和

|V |之间随机产生一个整数,作为交叉点,第三,交换从交叉点开始的后半部分(包括交叉点),于是两个新的染色体就替代了它们的双亲。

变异是一个确保发现最佳方案的概率永不为零的基本操作。变异操作会改变一个染色体的基因。具体步骤是:对每一个染色体,先随机选择一个有变异可能的任务,并产生一个新的 *PE* 值;再随机选择一个有变异可能的任务,并产生一个新的优先级值 priority。这样,一个新的染色体就替代了原来的染色体。

本文设计的算法目标是减少调度的跨度和所需 *PE* 的个数,并且消除无效的任务复制。

2.1.2 基于蚂蚁算法(Ant Algorithms, AA)的网格任务调度

蚂蚁算法是一个新的启发算法,它是基于蚂蚁的行为。当蚂蚁在寻找食物时,每个走动的蚂蚁都会在其经过的路上释放一些信息素,那么在较短路径上的信息素会很快地增加,每条路径上信息素的数量,会反映出其他蚂蚁选择该路径的概率。最终,所有的蚂蚁将选择最短的路径。蚂蚁算法固有的并发性和可扩充性,使它非常适合用于网格计算的任务调度。在同一时间内,所有影响资源状态的因素都能由一个事情,即信息素描述。那么调度程序就能非常简单、快速地获得预测结果。

文献[11]设计了一个网格计算的蚂蚁任务调度算法:当资源 j 要加入到网格时,它应提交其性能参数、PE 序号、每个 PE 的 MIPS(Million Instructions Per Second)等。资源监控器为了确认这些参数应对其进行测试,并对连接的信息素进行初始化:$\tau_j(0) = m \cdot p + c/s_j$

其中,$\tau_j(0)$ 为资源 j 的初始信息素,m 是 *PE* 的序号,p 是一个 *PE* 的 MIPS,c 是参数的长度,s_j 是从资源 j 到资源监控器的传输时间。

每当出现一个新资源加入到网格中、一个资源出现故障、一个任务被分配,或是有一些任务返回等情况时,从调度中心到相应资源路径上的信息素将会作如下变化:

$$\tau_j^{new} = \rho \cdot \tau_j^{old} + \Delta\tau_j$$

其中,$\Delta\tau_j$ 是从资源监控器到资源 j 路径上信息素的变化值;ρ 是持久的信息素($0 \leq \rho \leq 1$);$1-\rho$ 是挥发掉的信息素。

当任务被分配到资源 j 上以后,$\Delta\tau_j = -K$,K 是在该任务的计算和传输质量。

当任务成功地从资源 j 返回时,$\Delta\tau_j = Ce \cdot K$,Ce 是鼓励因子。

当任务从资源 j 失败返回时,$\Delta\tau_j = Cp \cdot K$,Cp 是惩罚因子。

在 t 时刻,任务被分配到每个任务上去的概率可表示为

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

其中,$\tau_j(t)$ 是在 t 时刻从调度中心到资源 j 路径上的信息素。η 是资源 j 的先天的性能,也即 $\tau_j(0)$。

$\tau_u(t)$ 是在 t 时刻从调度中心到资源 u 路径上的信息素。η_u 是资源 u 的先天的性能,也即 $\tau_u(0)$。

α 是信息素的权重。β 是资源先天属性的权重。

本文验证了蚂蚁算法的可扩展性,提出了一个简单的资源管理和任务调度的网格仿真结构。

2.1.3 遗传算法与蚂蚁算法的对比分析

文献[6]指出,遗传算法具有快速随机的全局搜索能力,但对于系统中的反馈信息利用却无能为力,当求解到一定范围时往往做大量无为的冗余迭代,求精确解效率低。蚂蚁算法是通过信息素的累积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,求解速度慢。

通过分析两个算法的优缺点,在实际设计网格任务调度算法时,可以将遗传算法与蚂蚁算法融合,即采用遗传算法生成信息素分布,利用蚂蚁算法求精确解,把两个算法的优势互补。

2.2 基于 Agent 的任务调度

Agent 技术源自于人工智能,其概念在 60 年代就已提出来,真正的发展在 90 年代,现在正向计算机领域的各方面渗透[22]。面向 Agent 技术是面向对象技术的发展和飞跃,面向 Agent 的软件开发方法是为了更确切地描述复杂的并发系统的行为而采用的一种抽象描述形式,与面向对象技术一样,它是观察客观世界及解决问题的一种方法。面向 Agent 技术为复杂、开放、分布系统的开发和实现提供了新途径。面向 Agent 技术的智能性、代理性、学习性、合作性、持续性使它非常适合网格计算及其任务调度的实现。基于 Agent 的任务调度其基本设计思想是:将每个网格资源节点均封装成为一个 Agent,把网格系统看成是一种多层次 Agent 系统的集合。这样,所有分布于网格系统中不同地方的网格资源节点(如一个机群)就构成了底层的 Agent 系统,它们为基于网格的应用程序提供高性能计算能力,于是,调度问题被简化成如何在各 Agent 之间分配计算任务并随时根据 Agent 的变化情况进行调整,以及在各 Agent 内如何进行子任务的继续分配。这样的层次结构具有高度的可扩展性,便于在网格这样的庞大异构环境中运用[12]。

文献[13]提出了一种基于 Agent 的计算网格(Agent-based Computational Grid, ACG)任务管理方案,给出了网格资源的描述,并设计了实现网格资源管理的三个网格协议原型:服务登记协议,服务发现协议和服务访问协议。描述了它们的构成要件及功能作用。该 ACG 资源管理方案可为网格的计算资源和服务提供一个统一的高层管理框架。

2.3 基于 Petri 网的任务调度模型

Petri 网理论是由德国的 Carl Adam Petri 博士提出的,主要研究分布式系统中并发、冲突现象的一种理论,它是描述和分析离散事件动态系统的一种模型工具。它综合了数据流、控制流和状态转移,能自然地描述并发、同步、资源争用等系统特性,而且本身自含执行控制机制,集规范表示与执行于同一模型。同时,Petri 网是严格定义的数学对象,借助数学开发的 Petri 网分析方法和技术,既可用于静态的结构分析,又可用于动态的行为分析[14]。

文献[15]提出了一个可用于网格计算任务调度的高级时间 Petri 网(Extended High-Level Timed Petri Net, EHLTPN)模型。在该模型中,分配给转移的点火时间是输入库所 Tokens 的函数。本文利用 EHLTPN 构造了一个网格计算的任务调度的简单模型,并定义了一个 EHLTPN 的可达调度图(Reachable Scheduling Graph, RSG)用来分析任务调度的时间特性。

2.4 基于成本的任务调度

文献[16]在经济原则的基础上为调度问题引入了成本

(Cost)概念。其核心思想是把诸如 CPU、存储器、带宽等不同类型资源的使用情况都转变成单一的成本,并按照总成本最小化的目标,对网格系统中的任务进行调度。为此设计了网络通信成本函数、CPU 成本函数、存储器成本函数。当一个新任务到达后,分别计算相应的成本函数为该任务选择一个成本最低的主机进行调度。

2.5　任务调度中的负载均衡

文献[17]中提出了一个基于 Agent 的网格管理基础设施,它连接着一个性能驱动的任务调度器,开发该调度器是为了平衡本地网格负载。每个网格调度器都使用可预测的应用性能数据和迭代启发算法,通过多处理节点达到本地负载平衡。在更高层次上,利用同种 Agent 的层次结构代表多网格资源。同时,通过使用服务广告和发现机制,Agent 之间相互协作,从而平衡全局网格的负载。文中还给出了研究例子和相关的实验结果,表明通过本地调度器和其他的 Agent 共同协作,可对全局的网格负载进行平衡,也显著地提高了网格执行性能和资源利用率。

2.6　其他任务调度算法

除了上述调度算法之外,人们还提出了很多异构系统的启发调度算法,有不少论文试图把这些算法用于网格环境。文献[18]提出了一个扩展的 Sufferage,叫 Xsufferage,可在网格环境中调度参数扫描应用。文献[19]提出了一个期限调度策略,可适用于多客户多服务器(Multi-Client Multi-Server)方式,并增加了能够改进算法性能的负载纠错(Load Correction)和退却(Fallback)机制。文献[20]提出了利用不同站点多同时请求的分布调度算法。文献[21]提出了主人 - 工人(Master-Worker)应用的调度策略,该策略能动态地检测任务的执行时间,并利用这个信息去动态地调整工人的数量,从而取得一个理想的工效。

3　研究展望

任务调度是网格计算的基本功能,它面临的问题是一个 NP 完全问题。到目前为止,人们提出了很多网格系统的调度技术和算法。但是,不少调度算法还没有完整的理论依据、一些调度技术的结论还只是来自仿真结果,至今还没有形成网格计算的任务调度理论。因此,该领域的研究还有待进一步发展和完善。具体地说,今后还要做的工作是:

(1)任务调度算法研究的粒度需要进一步细化。目前不少算法研究资源和任务是粗粒度的,一些调度思想还处于初步的原型阶段,离具体实现还有较大距离。

(2)启发性智能化方法将是网格计算任务调度的一个重要研究领域。目前启发性任务调度算法的研究还处于模拟仿真阶段,还需要在理论和实践中不断完善。

(3)面向 Agent 技术将是网格计算及其任务调度的一个重要的软件开发方法,在该领域同样还有很多工作要做。

(4)网格系统中的信息安全离不开任务调度的支持。也就是说,网格任务调度系统必须具有一定的安全机制,以便为网格提供安全可靠的任务调度服务。目前,安全任务调度技术有待进一步研究。

参考文献:

[1] 刘忠中. 网格计算及其技术需求分析[J]. 江西通信科技, 2003, (2): 36-40.

[2] Ullman J. NP-Complete Scheduling Problems[J]. Journal of Computer and Syst. Sciences, 1975, 10: 384-393.

[3] Andronikos T, Koziris N. Optimal Scheduling for UET-UCT Grids Into Fixed Number of Processors[C]. Parallel and Distributed Processing 2000 Proceedings, 8th Euromicro Workshop on, 2000. 237-243.

[4] 查礼, 徐志伟, 林国璋, 等. 基于 Simgrid 的网格任务调度模拟[J]. 计算机工程与应用, 2003, (14): 90-92.

[5] 刘家壮, 徐源. 网络最优化[M]. 北京: 高等教育出版社, 1991.

[6] 丁建立, 陈增强, 袁著祉. 遗传算法与蚂蚁算法的融合[J]. 计算机研究与发展, 2003, 40(9): 1351-1356.

[7] 陈国良. 遗传算法及其应用[M]. 北京: 人民邮电出版社, 1996.

[8] Wensheng Yao, *et al*. Genetic Scheduling on Minimal Processing Elements in the Grid[M]. Springer-Verlag Heidelberg, 2002.

[9] Di Martino V, *et al*. Scheduling in A Grid Computing Environment Using Genetic Algorithms[C]. Parallel and Distributed Processing Symposium, Proceedings International IPDPS, 2002. 235-239.

[10] Di Martino V. Sub Optimal Scheduling in A Grid Using Genetic Algorithms[C]. Parallel and Distributed Processing Symposium, 2003. 148-154.

[11] Zhihong Xu, Xiangdan Hou, Jizhou Sun. Ant Algorithm-based Task Scheduling in Grid Computing [C]. IEEE CCECE, 2003.

[12] 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.

[13] 李春林, 卢正鼎, 李腊元. 基于 Agent 的计算网格资源管理[J]. 武汉理工大学学报(交通科学与工程版), 2003, 27(1): 7-10.

[14] 袁崇义. Petri 网原理[M]. 北京: 电子工业出版社, 1998.

[15] Yaojun Han, *et al*. Resource Scheduling Algorithms for Grid Computing and Its Modeling and Analysis Using Petri Net[C]. Shanghai: The 2nd International Workshop on Grid and Cooperative Computing, 2003.

[16] Chuliang Weng, Xinda Lu. A Cost-based On-line Scheduling Algorithm for Job Assignment on Computational Grids[M]. Springer-Verlag Heidelberg, 2003.

[17] Junwei Cao, Daniel P Spooner, *et al*. Agent-based Grid Load Balancing Using Performance-driven Task Scheduling[C]. International Parallel and Distributed Processing Symposium, 2003. 49-58.

[18] Casanova H Legrand, A Zagorodnov, D Berman. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments[C]. Proceedings of the 9th Heterogeneous Computing Workshop, IEEE, Los Alamitos, 2000. 349-363.

[19] Takefusa, A Casanova, *et al*. A Study of Deadline Scheduling for Client-Server Systems on the Computational Grid[C]. Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, IEEE, Los Alamitos, 2001. 406-415.

[20] Subrammani V, *et al*. Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests[C]. Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, IEEE, Los Alamitos, 2002. 359-367.

[21] Heymann E Senar, M A Luque, *et al*. Adaptive Scheduling for Master-worker Applications on the Computational Grid[M]. Springer-Verlag Heidelberg, 2000.

[22] 崔燕妮. 面向 Agent 的理论与应用[J]. 云南师范大学学报, 2002, 22(3): 11-14.

作者简介:

罗红(1963-),男,副教授,博士生,主要研究方向为分布式系统、网格计算等;慕德俊(1963-),男,教授,博士生导师,研究方向为并行处理、网络安全等;邓智群(1976-),男,博士生,主要研究方向为信息安全、网格计算等;王晓东(1974-),男,博士生,主要研究方向为分布式处理技术、网络管理等。