ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.17, No.11, November 2006, pp.2352−2361
DOI: 10.1360/jos172352

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 树型网格计算环境下的独立任务调度*

林伟伟+, 齐德昱, 李拥军, 王振宇, 张志立

(华南理工大学 计算机科学与工程学院,广东 广州 510640)

## Independent Tasks Scheduling on Tree-Based Grid Computing Platforms

LIN Wei-Wei+, QI De-Yu, LI Yong-Jun, WANG Zhen-Yu, ZHANG Zhi-Li

(School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China)

+ Corresponding author: Phn: +86-20-38257397, E-mail: linweiwei2004@yahoo.com.cn, http://www.scut.edu.cn

**Lin WW, Qi DY, Li YJ, Wang ZY, Zhang ZL. Independent tasks scheduling on tree-based grid computing platforms. *Journal of Software*, 2006,17(11):2352−2361.** http://www.jos.org.cn/1000-9825/17/2352.htm

**Abstract**: Task scheduling is a fundamental issue in achieving high performance in grid computing systems. However, it is a big challenge for efficient scheduling algorithm design and implementation. In this paper, the problem of scheduling independent tasks on tree-based grid computing platforms, where resources have different speeds of computation and communication, is discussed. In contrast to minimizing the total execution time, which is NP-hard in most formulations, an integer linear programming model for this problem is presented. Using the model, the optimal scheduling scheme that determines the optimal number of tasks assigned to each computing node is obtained. With the optimal scheduling scheme, two demand-driven and dynamic heuristic algorithms for task allocation are proposed: OPCHATA (optimization-based priority-computation heuristic algorithm for task allocation) and OPBHATA(optimization-based priority-bandwidth heuristic algorithm for task allocation). The experimental results show that the proposed algorithms for the scheduling problem obtain better performance than other algorithms.

**Key words**: task scheduling; grid computing; integer linear programming; optimal scheduling scheme; heuristic algorithm

摘 要: 任务调度是实现高性能网格计算的一个基本问题,然而,设计和实现高效的调度算法是非常具有挑战性的.讨论了在网格资源计算能力和网络通信速度异构的树型计算网格环境下,独立任务的调度问题.与实现最小化任务总的执行时间不同(该问题已被证明是 NP 难题),为该任务调度问题建立了整数线性规划模型,并从该线性规划模型中得到最优任务分配方案——各计算节点最优任务分配数.然后,基于最优任务分配方案,构造了两种动态的需求驱动的任务分配启发式算法:OPCHATA(optimization-based priority-computation heuristic algorithm for task allocation)和 OPBHATA(optimization-based priority-bandwidth heuristic algorithm for task allocation).实验结果表明:在异构的树型计算网格环境下实现大量独立任务调度时,该算法的性能明显优于其他算法.

* Supported by the Natural Science Foundation of Guangdong Province of China under Grant No.05300200 (广东省自然基金); the Guangdong-Hong Kong Technology Cooperation Funding Scheme of China under Grant No.2005A10307007 (粤港关键领域重点突破项目)
Received 2006-06-06; Accepted 2006-08-25

关键词: 任务调度;网格计算;整数线性规划;最优任务分配方案;启发式算法
中图法分类号: TP393 文献标识码: A

随着互联网的飞速发展,利用互联网上大量计算资源的网格计算将成为解决规模庞大、复杂的问题的必由之路.要实现高效的网格计算需要处理许多复杂的问题,其中,任务调度问题是网格研究中所必须解决的一个关键问题,也是网格应用的基础.高效的任务调度策略和算法可以充分利用网格系统的处理能力,从而提高网格应用程序的性能,以便更好地利用网格资源.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].

在任务调度方面,已有人做了大量研究:最有影响的网格计算项目 SETI@home[3]是采用主-从模式任务调度;文献[1,4,5]中指出,在异构网格和分布式并行计算环境下的大部分任务调度问题都是 NP 难题;讨论树型结构计算环境下的任务调度问题也有一些[6−9],它们都是针对特定问题的各种类型任务的调度,如文献[7,8]讨论了在分布式多层树结构下考虑任务通信延迟的可分任务调度问题;与本文的研究最相近的文献[4]讨论了在异构树型多处理器计算平台下,独立相同任务调度问题的复杂性,并证明了该问题是 NP 难题,因此,不存在多项式时间复杂性的算法以找到全局最优解.为了获得近优解,存在许多启发式算法[10],如 Min-Min[10],Max-Min[10]、遗传算法[11,12]等.然而,本文利用线性规划建模任务调度问题,以获得最优任务分配方案来构造任务分配启发式算法.

我们针对在网格资源计算能力和网络通信速度异构的树型计算网格环境下独立相同任务的调度问题进行深入研究和分析,提出将该调度问题转化为线性规划问题,并根据线性规划模型获得网格中各计算节点的最优任务分配方案,然后以最优任务分配方案为基础,构造两种动态的需求驱动的任务分配启发式算法.通过实验与其他任务分配算法的比较,所提出的任务分配算法性能优于其他算法.

本文第 1 节给出本文讨论的问题的描述.第 2 节分析并给出单层树型结构网格计算平台下任务调度的线性规划模型.第 3 节重点给出多层树型结构网格计算平台下任务调度的线性规划模型,并分析任务调度实例.第 4 节给出两个基于最优任务分配方案的任务分配启发式算法.第 5 节对所提出算法进行实验和比较分析.最后是总结以及对未来工作的展望.

## 1 问题描述

本文研究在网络速度和网格计算节点处理能力不同的网格计算平台下的独立任务调度问题,而且该网格计算平台采用层次化树型覆盖网络模型.以树型作为网格计算环境的通信模型,可以简化网格计算的实现,降低节点通信的复杂度,因为各节点只需要负责与父亲和儿子通信,根节点到所有后代节点的路由是唯一的.而且,树型适用于许多编程范型,如主/从、RPC(remote procesdure call)、分而治之等.在树型网格计算平台中,每个网格计算节点有一个或多个儿子节点,但仅有一个父亲节点.我们考虑主/从模型(master-slave)网格任务调度问题,即任务只在根节点处理器产生,并由根节点负责传输任务给它的各个儿子节点.儿子节点接收到任务后,在开始处理任务的同时,继续转发部分任务给它的儿子节点,它接收父亲节点传输任务的同时可以转发任务给儿子,但只能给其中一个儿子节点传输任务,即服从单口模式(single-port model)[12].

本文讨论任务调度问题的前提是:1) 树型异构的网格计算平台;2) 考虑任务迁移代价,即任务传输是需要时间的;3) 调度的任务是相同大小的独立任务;4) 一个任务只能由一个节点完成计算,一个计算节点只能同时执行一个任务;5) 采用单口主/从模式的任务调度;6) 所有需要调度的任务都在根节点上输入.

图 1 给出一个由 5 个计算能力不同的计算节点组成树型网格计算平台下的简单任务调度示例.在该示例中:树的节点权表示节点计算能力,其值为节点计算单位任务所需的单位时间,即计算节点速度的倒数;树的边权表示节点间通信能力,其值为在节点间传输单位任务所需的单位时间,即两节点间网络速度的倒数.在这个调度示例中,在树型网格计算平台执行了 5 个大小相同的任务,所有这些任务都由担任 Master 的根节点 $n_0$ 来分配,同时,根节点也可以计算任务.在如图 1 所示右边的调度图中,垂直虚线表示单位时间,水平虚线表示计算节点,带箭头的水平线表示任务的执行,带箭头的斜线表示任务的传输.该任务调度示例服从单口模式,根节点 $n_0$ 在没

有完成一个任务传输之前不会启动另一个任务的传输,而且节点 $n_2$ 在接收根节点 $n_0$ 传送任务的同时,可以传输任务给一个儿子节点 $n_3$ 或 $n_4$,根节点 $n_0$ 和节点 $n_2$ 在传输任务的同时可以计算一个任务,但不能同时计算两个任务.在该任务调度示例中,调度 5 个相同大小的独立任务所需的总的执行时间为 15.当然,该调度不是最优调度,在本文的后续部分中我们会对其进行更加详细的分析.

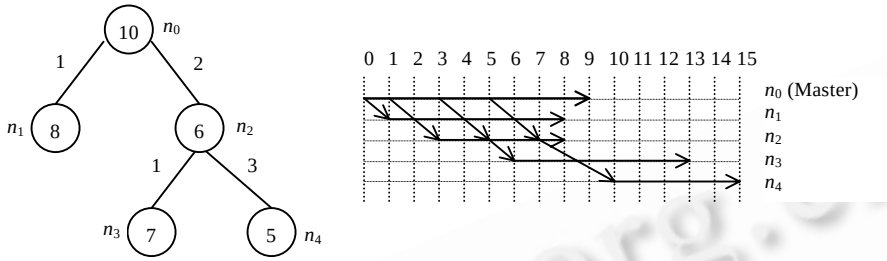



Fig.1 A tree-based grid computing platform and a task scheduling example

图 1 树型网格计算平台和任务调度示例

## 2 单层树型结构下任务调度模型

### 2.1 基本模型

图 2 给出了一般的单层树结构网格计算平台,该网格计算环境由 $k$ 个计算节点组成:$n_0,n_1,\ldots,n_{k-1}$,它们计算单位任务所需的时间分别为 $w_0,w_1,\ldots,w_{k-1}$,根节点 $n_0$ 作为 Master,负责传输任务给各儿子节点,但一次只能与一个儿子节点通信,它传输单位任务给各儿子节点所需时间分别为 $c_0,c_1,\ldots,c_{k-1}$.

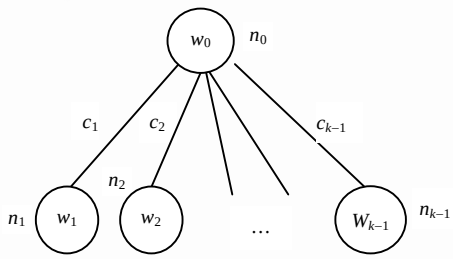



Fig.2 A single level tree grid computing platform

图 2 单层树结构网格计算平台

在该单层树型结构网格计算平台下调度 $M$ 个大小相等的独立任务,为了使完成任务的总时间最小,根节点 $n_0$ 需要决定自己计算多少个任务,以及需要传输多少个任务给各儿子节点,以实现一个最优的任务分配.我们可以把该任务调度问题转化为线性规划问题:a) 我们假定每个节点 $n_i$ 所执行的任务数为 $x_i$,所有节点执行的任务之和应该等于 $M$,由此约束条件可以得到下面的等式(1);b) 对于任意节点,计算任务个数都小于等于总的任务个数 $M$,由该约束条件可以得到不等式(2);c) 根节点 $n_0$ 计算任务的时间应小于等于总的任务完成时间,即有不等式(3); d) 除根节点以外的每个节点 $n_i$ 计算任务的总时间必然小于等于总的任务完成时间($T$)减去该节点最早启动任务计算时间(最早启动任务计算时间:单个任务从根节点到该任务计算节点的传输时间),可以得到下面的不等式(4);e) 因为根节点不能同时给几个儿子节点传输任务,所以它与各儿子节点传输任务的总时间必然小于等于总任务完成时间,即可得到下面的不等式(5);f) 由于我们这里所讨论问题的性质,有式(6).由上面的分析我们可以得到在单层树型网格计算平台下任务调度数学模型如下:

Minimize $T$ 满足:

$$\begin{cases} (1)\ \sum_{i=0}^{k-1} x_i = M \\ (2)\ 0 \le x_i \le M \quad \text{for } 0 \le i \le k-1 \\ (3)\ w_0 \cdot x_0 \le T \\ (4)\ w_i \cdot x_i \le T - c_i \quad \text{for } 1 \le i \le k-1 \\ (5)\ \sum_{j=1}^{i} c_j \cdot x_j \le T \\ (6)\ M, T, c_i, w_i, k, x_i \text{均为正整数} \end{cases},$$

其中:$M,c_i,w_i,k$ 均为已知量;$x_i$ 为变量;$T$ 表示任务的完成时间;Minimize $T$ 为目标函数.

通过上面的分析,我们将单层树任务调度问题转化为上述线性规划问题.容易发现,该模型属于整数线性规划问题.该问题可以由 Karmarkar 算法[13]在多项式时间内获得近似最优解,求解结果即为单层树任务调度的最优任务分配方案,即获得每个节点的最优分配任务数.

### 2.2 任务调度实例

考虑一个只有 4 个计算节点的小型单层树型网格计算环境,各计算节点的计算能力及网络通信速度都不同,如图 3 所示.如果在该环境下完成 8 个单位任务,用上述的线性规划模型建模该任务调度问题,结果如下:在使用 Linguo 或者 Matlab 工具获得该问题的最优解为$\{x_0,x_1,x_2,x_3\}=\{1,2,2,3\}$时,$T$ 的最优值为 17.

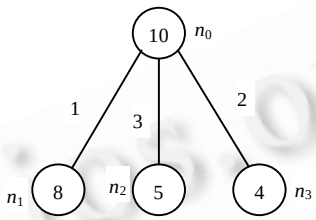



Fig.3 A single-level tree grid computing platform

图 3 一个单层树型网格计算环境

Minimize $T$ 满足:

$$\begin{cases}(1)\ x_0+x_1+x_2+x_3=8\\(2)\ 10x_0\le T\\(3)\ 8x_1\le T-1\\(4)\ 5x_2\le T-3\\(5)\ 4x_3\le T-2\\(6)\ x_1+3x_2+2x_3\le T\\(7)\ T,x_0,x_1,x_2,x_3\text{均为正整数}\end{cases}.$$

由于该问题涉及的任务数和计算节点个数都比较小,故容易得到一个最优任务调度,如图 4 所示.图中每个计算节点都有两条时间轴(任务计算和通信):带箭头的粗线表示任务的执行;不带箭头的粗线表示任务通信.由图 4 可以看出,该最优调度方案为$\{x_0,x_1,x_2,x_3\}=\{1,2,2,3\}$,任务的总完成时间为 18,它与线性规划模型得到最优值 17 最接近.但是,如果我们给每个计算节点平均分配任务,那么该任务调度总的完成时间至少为 20.

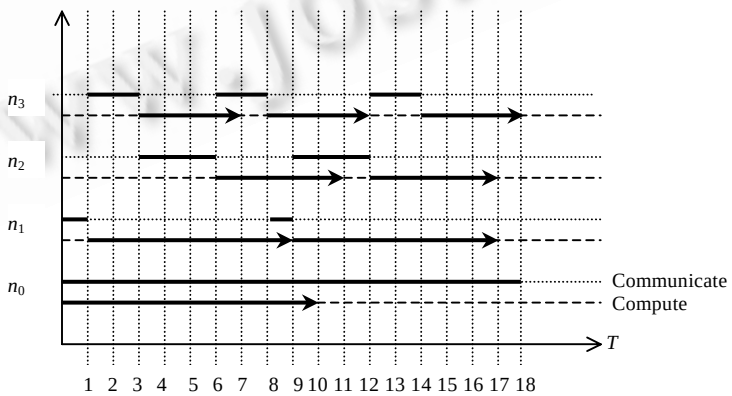



Fig.4 An optimal task scheduling

图 4 最优任务调度

## 3 多层树型结构下的任务调度模型

### 3.1 相关符号说明

图 5 为一个多层树型网格计算平台,每个网格计算节点只与父亲节点及其儿子节点通信,各计算节点的计算速度(或计算能力)和各节点之间网络通信速度都异构.下面给出在该平台下进行任务调度所用到概念的符号说明.(1) $N=\{n_0,n_1,\ldots,n_{k-1}\}$表示树型网格计算平台的网格计算节点(或称为树节点)集,其中,$k$ 为树的节点总个数;(2) $x_i$ 表示给树节点 $n_i$ 安排的任务数;(3) $w_i$ 表示树节点 $n_i$ 计算单位任务所需要的时间,即网格计算节点处理任务的速度的倒数;(4) $c_i$ 表示节点 $n_i$ 的入边传输单位任务所需要的时间,即节点 $n_i$ 的入边的任务传输速度的倒数;(5) $M$ 表示需要调度任务的总个数;(6) $N'$表示树中的非叶子节点集,则有 $N'\subset N$; (7) $s_i$ 表示节点 $n_i$ 的最早启动任务计算时间;(8) $Y_i$ 表示节点 $n_i$ 的后代节点的序号集,其中 $n_i\in N$; (9) $Z_i$ 表示节点 $n_i$ 的儿子节点的序号集,其中 $n_i\in N$.

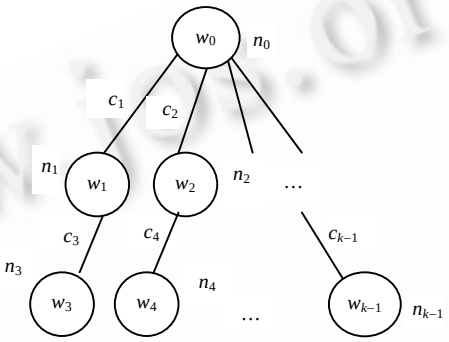



Fig.5 A multi-level tree grid computing platform

图5 多层树型网格计算平台

### 3.2 多层树型结构下的任务调度模型

前面的单层树型结构计算平台下的任务调度相对比较简单,然而,它的一些结论可以用于多层树中.当然,两种环境下的任务调度也有不同之处:a) 首先,在多层树结构任务调度中,除了根节点外,其他非叶子节点不但要计算任务,而且同时可以传输任务给它的儿子节点;b) 对于多层树中的每个中间节点,需要考虑其任务到达的速度,即在一定的时间范围内,它所获得的任务应该等于其己身计算的任务数与传输给儿子节点的任务数之和;c) 在单层树结构中,我们只需要考虑从根节点到各儿子节点的带宽分配情况,而在多层树结构中,需要考虑每一个子树中带宽分配问题,即任意一个非叶子节点传输任务给它的儿子的总时间是一定的;d) 每个节点的最早启动任务计算时间($S_i$)应该是父亲节点的最早启动任务计算时间加上它与父亲节点传输单位任务的时间,因此,每个节点计算任务的时间的约束条件式必须修改为下面的线性规划模型中的不等式(4).由 a),b)和 c)可知,在多层树型结构的任务调度中,每个子树都有一个带宽分配的约束条件,因此有不等式(5).下面以树根节点所在的子树为例进行分析,树根节点分配给所有儿子节点的传输任务的时间之和应小于等于总的任务完成时间 $T$,其中:树根节点分配给每个儿子的传输任务的时间为任务传输速度的倒数与传输任务个数的乘积,可以表示为 $\sum_{j\in Z_0}\left(c_j\cdot\left(x_j+\sum_{p\in Y_j}x_p\right)\right)$,其中:$Z_0$ 为根节点 $n_0$ 的所有儿子节点的序号组成的集合;$Y_j$ 为节点 $n_j$ 所有后代节点的序号组成的集合,$\left(x_j+\sum_{p\in Y_j}x_p\right)$ 为根节点 $n_0$ 传输给儿子节点 $n_j$ 的任务数,$x_j$ 为儿子节点 $n_j$ 计算的任务数,$\sum_{p\in Y_j}x_p$ 为儿子节点 $n_j$ 的后代节点计算的任务数总和.

由上面的分析我们可以得到多层树型结构网格计算平台下任务调度的数学模型如下:

Minimize $T$ 满足:

$$\begin{cases}(1)\sum_{i=0}^{k-1}x_i=M\\(2)\,0\le x_i\le M\ \text{ for } 0\le i\le k-1\\(3)\,w_0\cdot x_0\le T\\(4)\,w_i\cdot x_i\le T-s_i\ \text{ for } 0\le i\le k-1\\(5)\sum_{j=1}^{i}\left(c_j\cdot\left(x_j+\sum_{p\in Y_j}x_p\right)\right)\le T-s_i\ \text{ for } \forall n_i\in N'\\(6)\,M,T,c_i,w_i,k,x_i\text{均为正整数}\end{cases},$$

其中:$M$,$w_i$,$c_i$,$S_i$,$Y_i$,$Z_i$ 均为已知量;$x_i$ 为变量;$T$ 表示任务完成时间;Minimize $T$ 为目标函数.

上面的整数线性规划问题的求解与单层树任务调度模型相同,它可以在多项式次时间内得到近似最优解,所获得的近似最优解即为多层树任务调度的最优任务分配方案.

### 3.3 任务调度实例

考虑如图 1 所示的由 5 个计算节点组成的树型网格计算平台下的任务调度问题.用上面的线性规划模型求解该任务调度问题,我们可以得到下面的整数线性规划问题.如果在该环境下调度任务数 $M$ 为 8 和 50,则可以很容易地获得两个任务调度问题的最优解,分别为:$\{x_0,x_1,x_2,x_3,x_4\}=\{1,1,2,2,2\}$,最优值 $T$ 为 17;$\{x_0,x_1,x_2,x_3,x_4\}=\{7,9,12,8,14\}$,最优值 $T$ 为 77.

Minimize $T$ 满足:

$$\begin{cases}(1)\,x_0+x_1+x_2+x_3=M\\(2)\,10x_0\le T\\(3)\,8x_1\le T-1\\(4)\,6x_2\le T-2\\(5)\,7x_3\le T-3\\(6)\,5x_4\le T-5\\(7)\,x_1+2(x_2+x_3+x_4)\le T\\(8)\,x_3+3x_4\le T-3\\(9)\,T,M,x_0,x_1,x_2,x_3,x_4\text{均为正整数}\end{cases}.$$

## 4 基于最优任务分配方案的任务分配启发式算法

经过前面的讨论,我们容易建立树型网格计算平台下任务调度的线性规划模型,且求解线性规划模型可以获得最优任务分配方案,即获得各节点的最优任务分配数.然而,各父亲节点具体以什么顺序或优先级给多个儿子节点分配任务仍然无法确定,寻找该问题的最优任务分配算法是 NP 难题.但是,可以构造基于最优任务分配方案的任务分配启发式算法,即以最优任务分配数为启发信息来实现任务分配算法.下面给出基于最优任务分配方案的两个启发式算法:计算速度优先启发式算法(optimization-based priority-computation heuristic algorithm for task allocation,简称 OPCHATA)和带宽优先启发式算法(optimization-based priority-bandwidth heuristic algorithm for task allocation,简称 OPBHATA).

在给出 OPCHATA 和 OPBHATA 算法之前,需要实现两种算法:一种是模型预处理算法 PREModel,用来获得模型中需要的节点计算能力和节点间的通信能力.考虑到网格节点的动态性和异构性,在每批任务调度之前采用相对量化的方式来获得当前网格环境所有节点的计算能力和节点间的通信能力.PREModel 算法的具体实现是:在调度每批任务之前,将单位任务在树型网格环境的各节点上执行,以获得当前各节点的相对计算能力;将单位任务在树型网格环境的所有边上传输,以获得当前各节点间的相对通信能力;另一种是模型求解算法 SoveModel,用来求解任务分配的线形规划模型,它可以参照 Karmarkar 算法来实现.

### 4.1 计算速度优先启发式算法

该启发式算法首先使用 PREModel 和 SoveModel 算法求得每个节点的最优任务分配数,然后按照最优任务分配数和节点计算能力来优先为节点分配任务,即对于每个非叶子节点:当有儿子节点的任务请求时,首先检查该儿子节点是否还有未分配的任务,如果有,则分配一个任务给它;当有多个儿子节点的任务请求时,首先检查这些儿子节点是否还有未分配的任务,然后按照计算节点的计算速度以从大到小顺序来分配任务,即按照 $w_i$ 从小到大的顺序来确定任务分配优先级.该算法的伪码描述如下:

```
Procedure task_assign_OPCHATA()   //节点 n_i 为根的子树任务分配算法
  PREModel()   //模型预处理算法,只需在总根节点 n_0 部署
  SoveModel()   //模型求解算法,只需在总根节点 n_0 部署
   GetNodeTaskNum()   //从树根节点获得各节点任务分配数
   InitNodeTaskNum(array)   //初始化子树各节点任务分配数
   While (Receive_task())   //当从父亲节点收到任务时
      Add_task(Queue)   //将新任务放到队列中
   End While
   While (Receive_REQ())   //当收到儿子节点任务请求时
      Add_req(Queue2(n_x))   //将任务请求放到队列中
   End While
   If (Exist_task(Queue1))   //任务队列是否存在任务
      If (array(n_i)>0)   //节点 n_i 是否有未分配的任务
         Execute_task(Queue)   //执行一个任务
      End If
      //给请求任务的各儿子节点分配任务
      Select_compute(Queue2(n_x))   //从任务请求队列中选择计算能力最大的节点
      If (array(n_x)>0)   //该节点是否有未分配的任务
         TransTask(n_x)   //传输任务给节点 n_x
         Del_req(Queue2(n_x))   //删除任务请求
      End If
   Else   //发送任务请求给父亲节点,树根节点则不需要
      Send_req(n_i)
   End If
End
```

### 4.2 带宽优先启发式算法

与第 1 种启发式算法不同的是,当有多个儿子节点的任务请求时,首先检查这些儿子节点是否还有未分配的任务,然后按照计算节点与父亲节点的网络通信速度以从大到小的顺序来分配任务,即按照 $c_i$ 从小到大的顺序来确定任务分配优先级.该算法的伪码描述如下:

```
Procedure task_assign_OPBHATA()   //节点 n_i 为根的子树任务分配算法
   …//与 OPCHATA 算法内容相同
      //给请求任务的各儿子节点分配任务
      Select_bandwidth(Queue2(n_x))   //从任务请求队列中选择与根节点 n_i 网络通信速度最大的节点
   …//与 OPCHATA 算法内容相同
End
```

## 5 实验与结果

### 5.1 相关算法

为了验证所提出的基于最优任务分配方案的任务分配启发式算法(OPCHATA 和 OPBHATA)的性能,需要与其他任务分配算法进行比较.在同类算法中,Min-Min 算法具有较好的性能,常用作调度算法的评测基准[5,10],因此,本文另外设计了两种任务调度算法:FCFS 算法和 Min-Min 算法.FCFS 算法使用所有网格计算节点,其思想是:按照任务请求的到达顺序给各儿子节点分配任务,即先请求任务的先获得任务.该算法可能使各节点获得的任务数与最优任务分配方案求解的结果不一致.Min-Min 算法使用所有网格计算节点,其思想是:尽量把更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.

### 5.2 实验方法与结果

本文采用 SimGrid[14]来模拟和实现以上 4 种任务分配启发式算法.SimGrid 提供了一系列核心函数,用以建立和模拟异构分布计算环境,并能满足特定应用需求和实现多种算法,特别适合于网格任务调度的模拟和研究.我们分别模拟了 3 个树型网格计算平台:(1) 图 2 描述的由 4 个节点组成的单层树型网格计算平台;(2) 图 1 描述的由 5 个节点组成的 2 层树型网格计算平台;(3) 图 6 由 8 个节点组成的 4 层树型网格计算平台.然后,在这 3 个平台上分别用 FCFS,Min-Min,OPCHATA,OPBHATA 这 4 种算法实现 10 个、50 个、250 个、1250 个相同大小独立任务的调度,任务执行完成的时间分别如图 7、图 8 和图 9 所示.

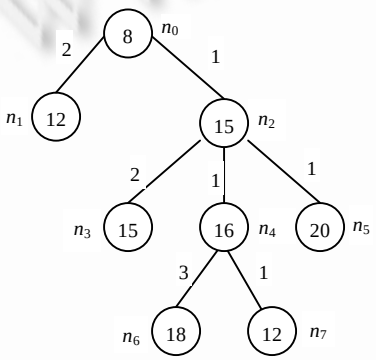
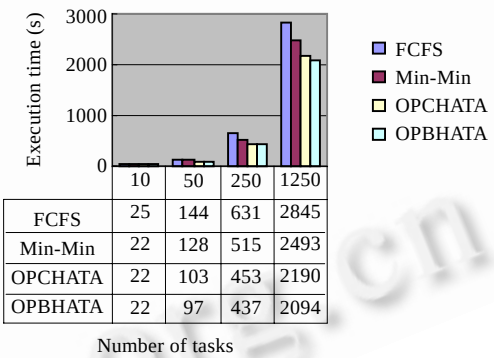



Fig.6 A multi-level grid computing platform

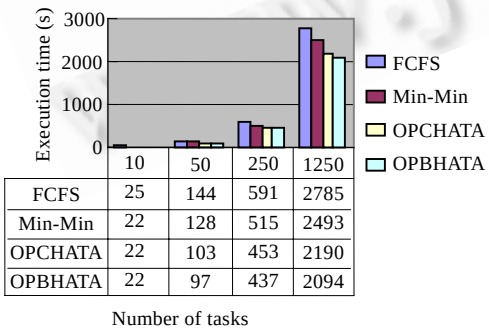
图 6 3 层树结构网格计算环境




| | 10 | 50 | 250 | 1250 |
|---|---|---|---|---|
| FCFS | 25 | 144 | 631 | 2845 |
| Min-Min | 22 | 128 | 515 | 2493 |
| OPCHATA | 22 | 103 | 453 | 2190 |
| OPBHATA | 22 | 97 | 437 | 2094 |

Fig.7 The results of task scheduling on the single-level tree grid computing platform

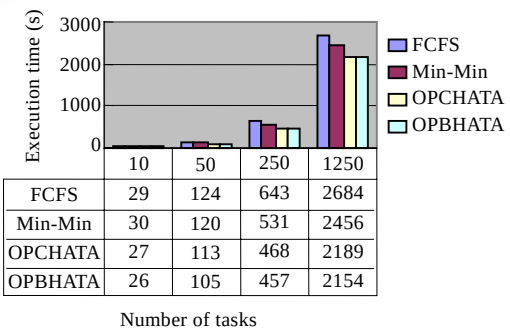
图 7 单层树型网格计算平台下的任务调度结果




| | 10 | 50 | 250 | 1250 |
|---|---|---|---|---|
| FCFS | 25 | 144 | 591 | 2785 |
| Min-Min | 22 | 128 | 515 | 2493 |
| OPCHATA | 22 | 103 | 453 | 2190 |
| OPBHATA | 22 | 97 | 437 | 2094 |

Fig.8 The results of task scheduling on the two-level tree grid computing platform

图 8 2 层树型网格计算平台下任务调度结果




| | 10 | 50 | 250 | 1250 |
|---|---|---|---|---|
| FCFS | 29 | 124 | 643 | 2684 |
| Min-Min | 30 | 120 | 531 | 2456 |
| OPCHATA | 27 | 113 | 468 | 2189 |
| OPBHATA | 26 | 105 | 457 | 2154 |

Fig.9 The results of task scheduling on the three-level tree grid computing platform

图 9 3 层树型网格计算平台下任务调度结果

为了验证所提出的模型和算法的普遍性,随机生成具有 50 个不同拓扑结构的树型网格计算平台,其中:树节点数在[2,20]范围内;节点权的值在[10,50]范围内;边权的值在[1,10]范围内.然后,在这些平台上分别用 FCFS, Min-Min,OPCHATA,OPBHATA 这 4 种算法实现 1 000 个单位任务的调度,任务执行完成时间如图 10 所示.

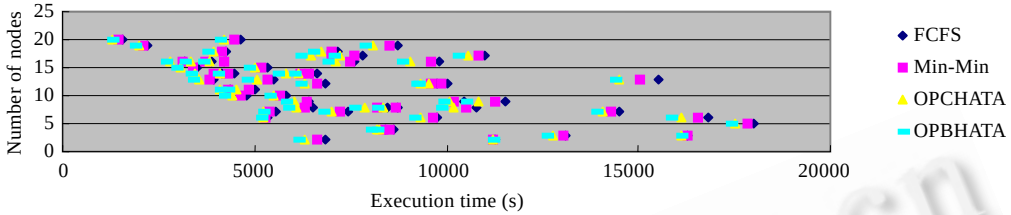



Fig.10 The experimental results of task scheduling on random tree-based grid computing platforms

图 10 多个随机树型网格计算平台下任务调度的实验结果

### 5.3 实验结果的分析

由实验测试结果(如图 7~图 10 所示)可以得出下面的结论:1) 在 3 种模拟网格计算平台下完成独立任务调度,OPCHATA 和 OPBHATA 算法的总完成时间小于 FCFS 和 Min-Min 算法,特别是当调度的任务数量比较大时,OPCHATA 和 OPBHATA 算法性能明显优于 FCFS 和 Min-Min 算法.也就是说,本文提出的基于最优任务分配方案的任务分配算法有明显优势.2) 在如图 6 所示的 8 个节点组成的 3 层树型网格计算平台下进行任务调度时,4 种算法的总处理时间比较接近,这是因为带宽对调度结果的影响.也就是说,当网络通信速度相对于任务计算速度比较快时,4 种算法的性能较为接近;反之,当带宽非常有限时,由于 OPCHATA 和 OPBHATA 算法只使用通信速度快的计算节点,结果实现更好的调度,因此性能更加优于 FCFS 和 Min-Min 算法(如图 7 和图 8 调度结果).3) 从 3 个平台下的实验结果还可以看出,OPBHATA 性能略优于 OPCHATA 算法,说明网络通信速度对任务调度结果的影响更大.因为 OPBHATA 算法尽量给网络通信速度快的节点优先分配任务,所以性能更好.4) 图 10 的结果显示,在随机产生的多个不同结构的树型网格计算平台下进行任务调度时,本文所提出算法的性能普遍优于 FCFS 和 Min-Min 算法,说明了本文给出的任务调度模型和算法的普遍性.

### 5.4 算法评价

PREModel 算法需要在 $n$ 个节点上循环执行单位任务,并在这些节点间传输单位任务,因此,它的时间复杂度为 $O(n)$.SoveModel 算法是参照经典线性规划求解算法 Karmarkar 算法[13]实现的,所以其时间复杂度为 $O(n^{3.5}L)$,其中 $n$ 为计算节点个数,$L$ 为线性方程组输入的规模.对于 OPCHATA 和 OPBHATA 算法,由于每个非叶子节点的最大分配任务数为 $m$,从请求任务的节点队列中选择一个节点任务最多需要 $n$ 次,因此,OPCHATA 和 OPBHATA 算法的时间复杂度为 $O(n^{3.5}L+mn)$,其中 $n$ 为主机数量,$m$ 为需要执行的任务数量.与 Min-Min 算法(调度 $m$ 个大小相同任务,其时间复杂度为 $O(mn)$)相比,OPCHATA 和 OPBHATA 算法的时间复杂度要高一些,但当 $m>n^{2.5}L$(任务数量比较大)时,它们的时间复杂度比较接近.而且,在网格环境下,任务粒度一般比较大,与任务的执行时间相比,任务调度算法的执行时间经常可以忽略.

另一方面,OPCHATA 和 OPBHATA 算法考虑带宽限制对任务传输时间的影响,通过线形规划模型获得最优的任务分配方案;而且在网格环境下,任务的传输时间往往对任务的最终完成时间有很大影响.因此,它们与 Min-Min,Max-min,FCFS 算法相比,能够获得更好的调度性能.

## 6 结束语

为了提高网格计算的性能,高效的任务调度和分配算法是其中一个主要途径.本文讨论了在异构树型网格计算环境下独立任务的调度问题,通过对该调度问题的深入研究和分析,针对单层和多层树型网格计算平台下的最小总任务处理时间的任务调度问题分别建立了线性规划模型,并通过求解线性规划模型获得最优任务分配方案,即获得网格中各计算节点最佳的任务分配数,然后提出基于最优任务分配方案的两个启发式算法:

OPCHATA 和 OPBHATA.通过使用 SimGrid 模拟和实现了所提出的任务分配算法,并将它们与 FCFS 和 Min-Min 算法进行比较,结果显示,本文提出的算法在树型网格计算平台下独立任务调度优于 FCFS 和 Min-Min 算法.

当然,本文提出的任务调度模型并没有讨论到网格环境的多个节点输入任务的调度情况,这也是我们下一步的研究内容.另外,还要考虑在任务之间的数据依赖性条件下如何建模任务调度问题.

**References**:


[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. http://www.cs.queensu.ca/TechReports/Reports/2006-504.pdf

[3] SETI@home home page. http://setiweb.ssl.berkeley.edu/

[4] Dutot P. Complexity of master-slave tasking on heterogeneous trees. European Journal on Operational Research, 2005,164(3): 690−695.

[5] Ibarra OH, Kim CE. Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the ACM, 1977, 24(2):280−289.

[6] Cheng YC, Robertazzi TG. Distributed computation for a tree network with communication delays. IEEE Trans. on Aerospace and Electronic Systems, 1990,26(3):511−516.

[7] Veeravalli B, Yao J, Divisible load scheduling strategies on distributed multi-level tree networks with communication delays and buffer constraints. Computer Communications, 2004,27(1):93−110.

[8] Beaumont O, Casanova H, Legrand A, Robert Y, Yang Y. Scheduling divisible loads on star and tree networks: Results and open problems. IEEE Trans. on Parallel and Distributed Systems (TPDS), 2005,16(3):207−218.

[9] Banino C, Beaumont O, Carter L, Ferrante J, Legrand A, Robert Y. Scheduling strategies for master-slave tasking on heterogeneous processor platforms. IEEE Trans. on Parallel and Distributed Systems, 2004,15(4):319−330.

[10] Braun TD, Siegel HJ, Beck N. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing, 2001,61(6):810−837.

[11] Vincenzo DM, Marco M. Sub optimal scheduling in a grid using genetic algorithms. Parallel Computing, 2004,30(5/6):553−565.

[12] Lin JN, Wu HZ. Scheduling in grid computing environment based on genetic algorithm. Journal of Computer Research and Development, 2004,41(12):2195−2199 (in Chinese with English abstract).

[13] Karmarkar N. A new polynomial-time algorithm for linear programming. Combinatorica, 1984,4(4):373−395.

[14] Casanova H. Simgrid: A toolkit for the simulation of application scheduling. In: Craig AL, Paul P, eds. Proc. of the 1st IEEE/ACM Int'l Symp. on Cluster Computing and the Grid. Brisbane: IEEE Computer Society Press, 2001. 430−437.


附中文参考文献:


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



林伟伟(1980－),男,江西武宁人,博士生,主要研究领域为计算机体系结构,网格计算.

齐德昱(1959－),男,教授,博士生导师,主要研究领域为计算机体系结构,分布式系统,网络安全.

李拥军(1968－),男,博士,副教授,主要研究领域为计算机网络体系结构,主动多播技术.

王振宇(1967－),男,博士,副教授,主要研究领域为高性能计算.

张志立(1964－),男,博士生,教授,主要研究领域为网格计算,Web 加速.