计算机科学 2002Vol. 29№. 2

环形互连并行系统上的任务分配

Distributing Divisible Task in a Ring of Processors

李国东　杨海荣　王东奎　张德富

(南京大学计算机软件新技术国家重点实验室　南京 210093)

Abstract This paper is about distributed processing of a divisible task in a ring of communicating processors, all works are based on ring topology. Our distribution scheme aims to guarantee minimal finish time of the whole task, which is divided into parts (subtasks) and scheduled to the individual processors. In our model the communication startup time and delay time are both taken into consideration; we also discuss the cases of returning results or not as well as sending data in parallel or not, and returning results through the ring are our focus for discussion. The volume of subtask assigned to each processor is gotten after creating and solving the linear equations. At last the empirical results are presented.

Keywords Divisible task, Distributed processing, Ring network, Scheduling

1. 介绍

近年来关于并行计算的研究十分活跃,并行任务的分配和调度策略是研究的热点之一。最近可分解任务(divisible task)的概念[1]被提出来,对它在并行系统上的分配策略亦进行了许多研究。可分解任务是这样的任务:它可被分解成多个任意大小的子任务,每个子任务在不同处理机上运行,由于这些子任务来自于同一个任务,它们在处理机上的处理类型是相同的。许多重要的应用都可归为可分解任务,例如向量处理,在数据库中搜寻记录,处理线性数据文件,求解线性代数中的某些问题等等。

几十年来针对并行和分布系统,研究者们提出了大量的计算和通信模型,其中在分布式存储器模型中,处理机之间用系统互连网络(System Interconnection Network, 简称 SIN)相连,消息和数据通过 SIN 进行传送。如在 Lesilie Valiant 提出的 BSP 模型中,系统由多个处理器/存储器对(结点)组成,它们之间借助通信网络进行互连。又如在 Culler 等人提出的 LogP 通信模型[2]中,用 4 个参数来刻画一个并行计算机系统,其中参数 o 代表通信开销,指结点从事发送和接收操作所需的时间;参数 L 代表网络硬件延时,指从发送结点到接收结点,消息和数据在 SIN 中传送所需的时间。通过对通信参数的考虑,LogP 模型能有效反映大量的现实并行计算机系统。同样地,本文的计算和通信模型既考虑了通信启动时间(类似于 LogP 中的参数 o),又考虑了通信延迟时间(类似于 LogP 中的参数 L),故具有广泛的代表性。

SIN 的拓扑结构有很多种,如总线形、环形和树形等。已有许多文献对于在 SIN 为特定拓扑结构的并行计算机系统中可分解任务的分配和调度进行研究,如文[3]针对线性结构,文[4]针对总线结构,文[5]针对树形结构,文[6]针对网格结构(meshes),文[7,8]针对超立方体结构(hybercube)。但它们普遍地缺乏对结果回送、数据处理和数据发送并行或串行等情况进行详细讨论;而文[5,6]没有同时把通信启动时间和通信延迟时间都考虑进去。目前尚没有专门针对环形系统中这些情况进行讨论的文献,本文研究了可分解任务在环形 SIN 互连的并行计算机系统上的分配和调度策略,提出了具体的分配方案,重点在于讨论处理结果通过环形 SIN 传回源结点(Originator)的情况。

2. 模型和概念

2.1 环形 SIN 的并行系统

处理器用环形 SIN 相连,消息和数据沿环单向传送。对于每个结点,它在环上的前一个结点称为它的直接前驱结点,环上的后一个结点称为它的直接后继结点。每个结点接收其直接前驱结点送来的数据,将数据的一部分分解出来在本地处理,并将余下的数据发给其直接后继结点。假定系统有 n 个处理器结点,开始时,整个可分解任务放在某一个结点中,这个结点被称为源结点(Originator),用 P_1 表示。P_1 的直接后继结点为 P_2,P_2 的直接后继结点为 P_3,…,P_n 的直接后继结点为 P_1,故 P_i 的直接后继结点为 P_{i+1}(对于结点 P_n 则 $P_{n+1}=P_1$)。这样 $P_1,P_2,\cdots,P_n$ 连成了一个环,任务分配和结果回送都在环上进行。

2.2 通信和计算模型

整个可分解任务用 T 来表示,共有容量为 W 的数据。从源结点 P_1 开始,结点将任务的一部分分解出来在本地处理,并将余下的任务发给直接后继结点。用 T_1 代表 P_1 处理的子任务,其数据容量为 W_1;对于结点 P_i,其处理的子任务用 T_i 来表示,其数据容量为 W_i。系统在开始时将 T 放在 P_1 中,P_1 在本地处理容量为 W_1 的子任务 T_1 后,将剩下容量为 $W-W_1$ 的数据发往直接后继结点 P_2;同样地 P_2 在本地处理容量为 W_2 的子任务 T_2 后,将剩下容量为 $W-W_1-W_2$ 的数据发往直接后继结点 P_3;如此直到 P_n,P_n 处理容量为 $W-(W_1+W_2+\cdots+W_{n-1})$的数据。并行处理系统可用(P,A,S,C,n)来表示,其中 n 是处理器的个数;P_i 为第 i 个处理器;A_i 是处理机 P_i 的数据处理速率,此速率以每单位数据所需处理时间来衡量,通常单位为秒/字节;S_i 是 P_i 往 P_{i+1} 发送数据所需的通信启动时间;C_i 是 P_i 和 P_{i+1} 间网络通路的通信速率,是网络传

李国东　硕士生,主要研究领域为并行与分布处理,网络通信。王东奎　硕士生,主要研究领域为并行与分布处理,面向对象技术。杨海荣　硕士生,主要研究领域为并行与分布处理,网络计算。张德富　教授,博士生导师,主要研究领域为并行处理技术和分布式系统。

输速率的倒数，通常以秒/字节为单位，若 P_i 往 P_{i+1} 发送容量为 x 的数据，则通信延迟时间为 xC_i，这样从 P_i 到 P_{i+1} 传送容量为 x 的数据共需 S_i+xC_i 的时间[1,4]。

对于回送结果的情况，每个处理器 P_i（除 P_1 外）处理完所分配的数据后，把结果回送给源结点 P_1（经 P_{i+1}，P_{i+2}，…，P_n），最后在 P_1 中进行汇总。本文重点讨论了这种回送结果的情况。另外，P_i 在处理本地数据时，可同时发送余下数据给 P_{i+1}，这是并行发送的情况；否则 P_i 只能在发送完数据后才开始本地数据处理，这是串行发送的情况，本文对这两种发送方法都进行了讨论。

3. 可分解任务的分配和调度

3.1 不回送结果的情况

各处理器完成本地的数据处理后，不需要回送结果到源结点 P_1。为了使整个任务的完成时间最短，所有处理器的子任务结束时间应该相等。根据处理器是否可以并行发送分两种情况：串行发送和并行发送。串行发送指 P_i 在开始处理子任务 T_i 之前，首先将余下数据发往 P_{i+1}，这种情况适用于处理机中 I/O 处理与 CPU 处理不能同时进行的系统；而并行发送指 P_i 在开始处理 T_i 的同时，将余下数据送往 P_{i+1}，这种方法适用于处理机中设有专门 I/O 设备的系统。

由于不用回送结果，环形并行系统上的可分解任务的分配和调度与线性并行系统上[3]的情况相似。对于采用并行发送的情况，可建立 n−1 个等式：

$$W_iA_i=S_i+(W_{i+1}+\cdots+W_n)C_i+W_{i+1}A_{i+1},\quad i=1,2,\cdots,n-1 \tag{1}$$

再加上一个等式 $W=W_1+W_2+\cdots+W_n$

对于这 n 个等式，若合理解存在（$W_i\geqslant 0$），则可在 O(n) 时间内求解，否则可在 O(nlogn) 时间内找到最大可用集[3]。系统的线性加速比为

$$S_n=WA_1/W_1A_1=W/W_1$$

对于串行发送的情况，等式(1)改为

$$W_iA_i=S_{i+1}+(W_{i+1}+\cdots+W_n)C_{i+1}+W_{i+1}A_{i+1},\quad i=1,2,\cdots,n-2$$

$$W_{n-1}A_{n-1}=W_nA_n$$

这种情况下的线性加速比为

$$S_n=W/(W_1A_1+(W-W_1)C_1+S_1)$$

3.2 回送结果的情况

3.2.1 *结果回送机制* $P_i(i>=2)$ 完成对 W_i 的处理之后，生成数据量为 R_i 的结果，该结果经过环形 SIN 的结点 P_{i+1}，P_{i+2}，…，P_n 传回 P_1，传回时间为 $\sum_{j=1}^{n}S_j+R_i\sum_{j=i}^{n}C_j$。

若每个结果独立在环中传回，则所有结果的传回时间为 $\sum_{i=2}^{n}(\sum_{j=1}^{n}S_j+R_i\sum_{j=i}^{n}C_j)$。

我们采用一种合并回送机制：在 P_i 回送本地产生的结果之前，先将 $P_j(2\leqslant j\leqslant i)$ 的结果全部汇合起来，再和 P_i 产生的结果一起发送给 P_{i+1}。这样所有结果传回源结点的时间为 $\sum_{j=2}^{n}S_j+\sum_{i=2}^{n}(R_i\sum_{j=i}^{n}C_j)$，从而节省了通信启动时间。

3.2.2 *并行发送* P_i 在开始处理 T_i 的同时，将余下数据送往 P_{i+1}，数据处理和数据发送同时进行；由于 P_i 必须完成 T_i 的处理后才能回送结果，故结果的发送与数据的处理只能串行进行，而且，根据我们的结果合并回送机制，P_i 在发送本地结果前先搜集前面结点传送到来的结果，然后和本地结果一起发送，把这 i−1 个结果（分别由 P_2，P_3，…，P_i 产生）传给 P_{i+1} 共需时间 $S_i+C_i(R_2+\cdots+R_i)$。为了使整个任务的完成时间最短，所有结果到达 P_i 的时间应该相等。图 1 是各子任务计算和通信的时空图，从中建立方程组：

$$W_1A_1=S_1+(W_2+W_3+\cdots+W_n)C_1+W_2A_2+\sum_{j=2}^{n}\sum_{k=2}^{j}R_kC_j+\sum_{j=2}^{n}S_j$$

$$W_2A_2+S_2+R_2C_2=W_3A_3+S_2+(W_3+W_4+\cdots+W_n)C_2$$

$$W_3A_3+S_3+(R_2+R_3)C_3=W_4A_4+S_3+(W_4+W_5+\cdots+W_n)C_3$$

…

$$W_iA_i+S_i+(R_2+R_3+\cdots+R_i)C_i=W_{i+1}A_{i+1}+S_i+(W_{i+1}+W_{i+2}+\cdots+W_n)C_i$$

…

$$W_{n-1}A_{n-1}+S_{n-1}+(R_2+R_3+\cdots+R_{n-1})C_{n-1}=W_nA_n+S_{n-1}+W_nC_{n-1}$$

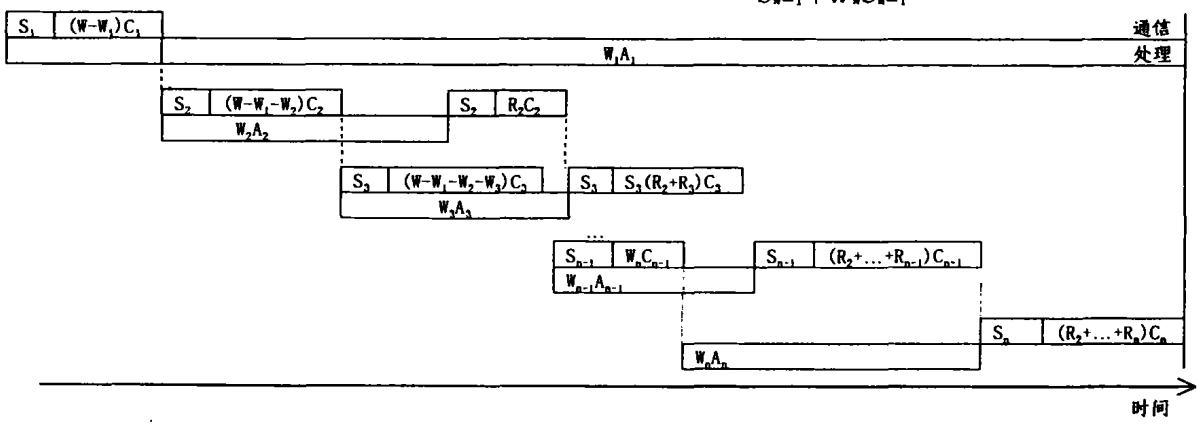

图 1 并行发送时各子任务计算和通信的时空图

其中第一个等式代表 P_1 的处理完成时间等于 P_2 的完成时间加上所有结果回送到 P_1 的时间，这个回送时间为 $\sum_{j=2}^{n}\sum_{k=2}^{j}R_kC_j+\sum_{j=2}^{n}S_j=\sum_{j=2}^{n}(R_j\sum_{k=1}^{n}C_k)+\sum_{j=2}^{n}S_j$；对于子任务 $T_i(2\leqslant i\leqslant n-1)$，其在 P_i 上的处理完成时刻加上结果传送到 P_{i+1} 的时间等于子任务 T_{i+1} 的处理完成时刻，这样可得到 n−2 个等式；另外，由于各子任务 T_i 由任务 T 分解而成，故得第 n 个等式 $W=W_1+W_2+\cdots+W_n$。即方程组(1)：

$$W_1A_1=S_1+(W_2+W_3+\cdots+W_n)C_1+W_2A_2+\sum_{j=2}^{n}$$

$$(R_i\sum_{k=i}^{n}C_k)+\sum_{j=2}^{n}S_j$$

$$W_iA_i=W_{i+1}A_{i+1}+(W_{i+1}+W_{i+2}+\cdots+W_n)C_i-(R_2+R_3+\cdots+R_i)C_i,\ i=2,3,\cdots,n-1 \quad (1)$$

$$W=W_1+W_2+\cdots+W_n$$

各处理器完成本地任务后产生回送结果，子任务 T_i 的结果的容量 R_i 为容量 W_i 的函数，即 $R_i=f(W_i)$。下面针对函数 f 的类型进行讨论。

1)f 为线性函数，则可设 $R_i=mW_i+b$，其中 m 是线性系数(斜率)。将此式代入方程组(1)中，i=2,3,…,n−1 时的等式变为

$$mC_i(W_2+W_3+\cdots+W_{i-1})+(A_i+mC_i)W_i-(A_{i+1}+C_i)W_{i+1}-C_i(W_{i+2}+W_{i+3}+\cdots+W_n)=-bC_i(i-1)$$

然后得到系数矩阵如下：

	1	2	3	4	…	n
1	A_1	$-(A_2+C_1+\sum_{k=2}^{n}C_k)$	$-(C_1+\sum_{k=3}^{n}C_k)$	$-(C_1+\sum_{k=4}^{n}C_k)$	…	$-(C_1+C_n)$
2		A_2+mC_2	$-(A_3+C_2)$	$-C_2$	…	$-C_2$
3		mC_3	A_3+mC_3	$-(A_4+C_3)$	…	$-C_3$
4		mC_4	mC_4	A_4+mC_4	…	$-C_4$
…	…	…	…	…	…	…
n−1		mC_{n-1}	mC_{n-1}	mC_{n-1}	…	MC_{n-1}
n	1	1	1	1	…	1

要求解这个方程组，先按列消除左下三角的系数，这项操作的时间复杂度为 O(n)，再按行消回，时间复杂度也是 O(n)，这样总共的时间复杂度为 O(n)。线性加速比为 $S_n=WA_1/W_1A_1=W/W_1$，利用率为 $U_n=S_n/n=W/_nW_1$。

当 m=0 时，$R_1=R_2=R_3=\cdots=R_n=b$，所有结果的容量相同，系数矩阵中将第 n 行消去后矩阵变为上三角矩阵，可直接按行消除来求解方程组。

2)f 为非线性函数，如二次函数、立方根函数等，此时方程组(1)不再是线性方程组，不能用线性消元法求解。应根据具体的 f 函数来求解方程组。

3.2.3 串行发送 P_i 在开始处理子任务 T_i 之前，首先将余下数据发往 P_{i+1}，然后处理本地数据；结果的回送也是串行发送。图 2 是串行发送时各子任务计算和通信的时空图。

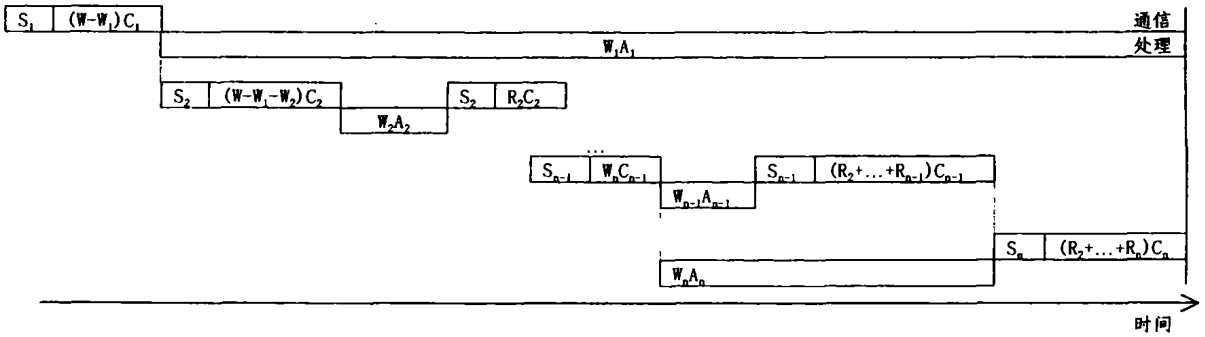

图 2 串行发送时各子任务计算和通信的时空图

这种情况下方程组(1)改为方程组(2)：

$$W_1A_1=W_2A_2+\sum_{j=2}^{n}\sum_{k=2}^{j}R_kC_j+\sum_{j=2}^{n}S_j$$

$$W_iA_i=W_{i+1}A_{i+1}+S_i-(R_2+R_3+\cdots+R_i)C_i,\ i=2,3,\cdots,n-1 \quad (2)$$

$$W=W_1+W_2+\cdots+W_n$$

时间复杂度与并行发送的情况一样，而线性加速比为 $S_n=W/(W_1A_1+(W-W_1)C_1+S_1)$，利用率为 $U_n=S_n/n=W/(n(W_1A_1+(W-W_1)C_1+S_1))$。

4 实验结果

实验中的对象(可分解任务)是在数据库中搜寻记录并返回匹配记录的条数。测试环境为用 IBM 令牌环网连接的 SGI indy 工作站(处理器)并行系统，每台处理器的数据处理速率 A_i(单位为秒/记录)通过对数据库记录查找速率进行重复测试得到；将数据沿环传送测得网络通信速率 C_i(单位为秒/记录)和通信启动时间 S_i(单位为秒)的值。

我们对不返回结果和返回结果这两种情况都进行了试验。图 3 反映了试验实际结果和理论计算结果的相对误差情况(比较两者的任务完成时间)，实验用了 4 个处理器进行并行处理，处理器返回结果到源结点，随着可分解任务容量的增大，误差相对减少，证明了方案的有效性。另外，根据 Amdahl 定律，并行系统中的工作负载可分为串行部分和并行部分两部分，串行部分是任务的顺序瓶颈，其比例越大则加速比越小。在我们的方案中，P_{i+1}必须收到 P_i 传来的数据和结果后才能进行下一步操作，故任务在环中传送是串行的，通信延迟是系统的瓶颈。图 4 反映了并行系统的加速比情况，随着处理器数目的增多，加速比提高越来越慢，这主要由于环形 SIN 的通信路径过长，通信开销过大造成的，这对于串行发送的情况尤为明显，因为此时处理器不能重叠通信和计算。

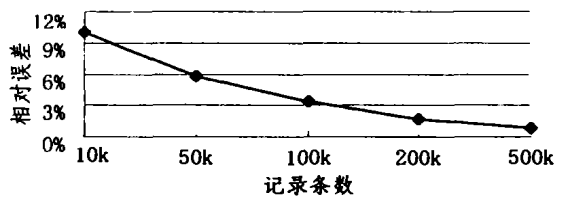

图 3 试验实际结果和理论计算结果的相对误差

(下转第 71 页)

权类型 a_2。

(5)零元谓词 error( )用来表示授权系统中的异常。

4.2 规则库的组成

1)推导规则。根据谓词 rup(X,Y),oup(X,Y),以及 aup(X,Y)本身的传递性以及规则3.1,3.2,3.3与推论3.1,3.2,3.3可有如下九条推导规则生成。

(1)rup(X,Y) & rup(Y,Z)→ rup(X,Z)

(2)oup(X,Y)& oup(Y,Z) → oup(X,Z)

(3)aup(X,Y) & aup(Y,Z) → aup(X,Z)

(4)rup(S_i,S_j) & abop(S_j,O,A) → abop(S_i,O,A)

(5)rup(S_i,S_j) & abop(S_i,O,−A) → abop(S_j,O,−A)

(6)aup(A_i,A_j) & abop(S,O,A_i) → abop(S,O,A_j)

(7)aup(A_i,A_j) & abop(S,O,−A_j) → abop(S,O,A_i)

(8)oup(O_i,O_j) & abop(S,O_i,A) → abop(S,O_j,A)

(9)oup(O_i,O_j) & abop(S,O_j,−A) → abop(S,O_i,−A)

2)一致性检查规则。主要用来保证授权在 S,O,A 三个域上传播的正确性和有效性。

(1)角色一致性规则

·角色至多只有一个上层角色

rup(S_1,S_2) & rup(S_2,S_1) & S_1<>S_2→ error ( )

·角色层次结构上无循环

rup(R_1,R_2)&…& rup(R_i,R_{i+1}) &…& rup(R_{j-1},R_j) &… & rup(R_{n-1},R_n) & R_i=R_j & i≥1 & i≤n & j≥1& j≤n → error( )

(2)授权对象一致性规则

·授权对象集合间的无循环性

oup(O_1,O_2)&…&oup(O_i,O_{i+1}) &…& oup(O_{j-1},O_j)&… &oup(O_{n-1},O_n)& O_i=O_j& i≥1& i≤n & j≥1& j≤n → error ( )

(3)授权类型一致性规则

·授权类型集合元素间的无循环性

aup(A_1,A_2) &…&aup(A_i,A_{i+1})&…& aup(A_{j-1},A_j)& …&aup(A_{n-1},An) & A_i=A_j & i≥1 & i≤n & j≥1& j≤n → error ( )

(4)授权事实库的一致性保证规则

abop(S,O,A) & abop(S,O,−A) →error ( )

(5)授权事实库的无冗余性保证规则

→(∃ (X,Y,Z)(ab(X,Y,Z) →ab(S,O,A))))

论域为当前授权事实库。

结束语 我们提出的隐式授权推导模型适用于关系型 DBMS 环境下的多用户信息系统,在给定显式授权的情况下得到系统中所有角色的隐式授权。这种动态生成隐式授权的方法不但节约了存储空间,而且使信息系统中的授权管理变得容易和简单。但是,时空永远是一对矛盾,这种方法虽然节省了存储空间,却以增加运行时间为代价。

参 考 文 献

1 Rabbitti F,Bertino E,Kim W,Woek D. A model of authorization for next-generation database systems. ACM Transactions On Database Systems, 1991,16(1):88～131

2 Bertino E, et al. An Extended Authorization Model for Relation Databases. IEEE Transaction on Knowledge and Data Engineering,1997,9(1)

3 Bertino E,et al. Authorizations in relational database management systems. In:Proc. ACM Conf. on Computer and Communications Security,Fairfax, VA, Nov. 1993. 140～150

4 Jajodia S,Samarati P,Subrahmanian V S. A Logical Language for Expressing Authorization. IEEE Symposium on Security and Pricacy,1997

5 罗雪平. 一种扩展的基于角色的访问控制的设计与实现:[硕士学位论文]. 华东师范大学计算机系,2001

6 王景光. 数据库授权系统的研究与应用. 计算机应用,1997,5

(上接第16页)

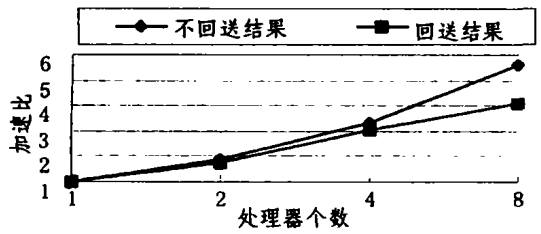

图4 数据库记录并行搜寻的加速比曲线

结论 本文针对环形 SIN 互连的并行计算机系统讨论了可分解任务的分配和调度,处理器间通过环来传递数据和消息。我们的模型同时把通信启动时间和通信延迟时间都考虑进去,能有效反映大量的现实并行计算机系统。本文重点考虑处理结果通过环形 SIN 传回源结点的情况,并为并行发送和串行发送两种情况分别建立了反映子任务处理和通信情况的线性方程组,通过对方程组进行求解可得出保证任务完成时间最短的分配和调度方案。文章给出了算法的时间复杂度和线性加速比,并进行了实验。设处理器的个数为 n,则当合理解存在时求解方程组的时间复杂度为 O(n),否则在 O(nlogn)的时间内可找到方程组的最大可用集,故本文提出的方案简单实用,能有效地在同构和异构并行系统上进行可分解任务的分配和调度。

参 考 文 献

1 Blazewicz J, Drozdowski M, Markiewicz M. Divisible Task Scheduling-Concept and Verification. Parallel Computing, 1999, 25:87～98

2 Culler D E,et al. LogP: a practical model of parallel computation. Communications of the ACM,1996, 39(11):78～85

3 Mani V, Ghose D. Distributed computation in linear networks: Closed-form solutions. IEEE Transactions on Aerospace and Electronic Systems,1994,30(2) :471～483

4 Blazewicz J,Drozdowski M. Scheduling divisible jobs with communication startup costs. Discrete Applied Mathematics ,1997,76 (1-3)

5 Bharadwaj V,Ghose D,Mani V. Optimal sequencing and arrangement in distributed single-level tree networks with communication delays. IEEE Transactions on Parallel and Distributed Systems, 1994,5(9):968～976

6 Blazewicz J,Drozdowski M. Performance limits of two-dimensional network of load-sharing processors. Foundations of Computing and Decision Sciences,1996,21(1): 3～15

7 Blazewicz J,Drozdowski M. Scheduling divisible jobs on hybercubes. Parallel Computing,1995,21:1945～1956

8 Drozdowski M,Glazek W. Scheduling divisible loads in a three-dimensional mesh of processors. Parallel Computing,1999,25:381～404