一个并行程序辅助开发工具*

徐 鸿　陈德来　张德富

（南京大学计算机软件新技术国家重点实验室　南京大学计算机系　南京　210093）

摘要　NUCAPPT是一个基于任务分配和通信规划启发式算法的并行程序辅助开发工具.它由任务分配器、通信规划器、通信表和通信语句生成器4部分组成.它不仅能保证处理器间数据通信的正确性,减小处理器的通信等待时间,而且能简化并行程序的编制过程,提高所编并行程序的质量.

关键词　并行程序辅助开发工具,任务分配,通信规划.

并行程序设计与并行处理结构模型密切相关,同一问题采用不同的结构设计出来的程序可能大不相同.故一般来说,并行程序较难具有可移植性.同时,由于目前还没有一套规范的工程化方法指导并行程序开发,编制并行程序带有很大的技巧性.这些不仅增加了并行程序设计的难度,而且使开发出来的并行程序的正确性得不到有效的保证.

在并行程序设计的几个步骤中(图1所示)任务分配是基础.任务分配结果直接影响算法能否充分利用系统资源,拓展问题对象内部并行性.数据通信处理也是并行程序设计的一个重要问题.任务划分和任务分配都以减小系统数据通信量为目标之一.程序编码、调试、测试的难点之一也是如何确定通信的先后次序及此次序是否正确.通信次序处理不当不仅会使系统效率低,而且可能会使整个系统发生死锁.在大规模并行处理系统(MPP系统)中,这个问题尤为突出.

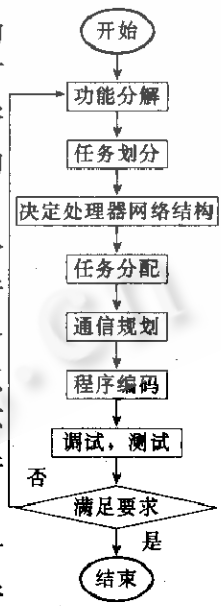

图1　并行程序设计流程

本文针对并行程序设计的2个关键问题即任务分配和数据通信,讨论了任务分配和通信规划的模型及其启发式算法,并据此设计了一个并行程序辅助开发工具— NUCAPPT.它首先将与应用程序的各个子功能相对应的每个任务分配到并行计算机的有关处理器,任务分配后系统负载比较平衡.在此基础上对每一处理器生成一通信表,通信表决定了各个任务的通信次序.它可以直接作为程序设计阶段的数据通信协议.只要程序员根据通信表编制程序就能保

* 本文研究得到国家863高科技项目基金资助.作者徐鸿,1972年生,硕士生,主要研究领域为并行处理.陈德来,1965年生,博士生,主要研究领域为并行处理,仿真技术.张德富,1937年生,教授,博士导师,主要研究领域为并行处理,分布式计算.

本文通讯联系人:张德富,南京210093,南京大学计算机系

本文1996－01－08收到修改稿

保证系统通信的正确性和有效性.

1 任务分配和通信规划模型

1.1 任务分配模型

影响任务分配的因素很多,主要有数据通信量、处理器的负载、任务执行的优先权、并行处理的结构模型等. 任务分配时,要综合考虑这些因素,选取恰当的目标函数.

定义 1. 分配矩阵 $X=(x_{ij})_{M\times N}$, $x_{ij}=1$ 表示任务 i 分配到处理器 j 上,否则 $x_{ij}=0$.

定义 2. 执行开销矩阵 $Q=(q_{ij})_{M\times N}$, q_{ij} 表示任务 i 在处理器 j 上的执行时间.

定义 3. 内部通信开销矩阵 $C=(c_{ij})_{M\times N}$, $c_{ij}(i\neq j)$ 表示任务 i 发送数据给任务 j 或任务 j 从任务 i 接收数据的通信开销.

定义 4. 任务 k 的外部通信开销 ec_k,表示任务 k 从外设接收或发送数据的开销.

定义 5. 处理器 k 的内部通信代价 $IPC(k,X)=\Sigma_{i|x_{ik}=1}\Sigma_{j|x_{jk}=0}(c_{ij}+c_{ji})$ $k=1,2,\ldots,N$.

定义 6. 处理器 k 的有效执行时间 $AET(k,X)=\Sigma_{i=1}^{M}x_{ik}q_{ik}$ $k=1,2,\ldots,N$.

定义 7. 处理器 k 的工作负载 $Load(k,X)=AET(k,X)+IPC(k,X)+\Sigma_{i=1}^{M}ec_ix_{ik}$,它由 3 部分组成:有效执行时间、内部通信开销、外部通信开销.

定义 8. 完成时间 $ET(X)=\max_{1\leqslant i\leqslant N}\{Load(i,X)\}$,即负载最大的处理器的工作时间.

处理器速度限制:执行开销矩阵 Q 体现了这一限制.

处理器网络拓扑结构限制:它表明并行处理结构模型是否固定.

实时限制:$\Sigma_{k=1}^{M}x_{ik}q_{ik}\leqslant T_k$ $k=1,2,\ldots,N$, T_k 为处理器 k 总的执行时间限制.

负载平衡限制:负载是否平衡对系统性能的影响是很大的. 系统负载平衡能增加系统稳定性,提高系统吞吐量,减小系统响应时间. 它是一个十分重要的约束条件.

代价函数:代价函数的选择很大程度上依赖于应用的性质和并行处理的结构模型. 任务分配各种工作都是围绕如何刻划各种约束条件和选取适当的代价函数这个核心.[1,2]常用的面向性能优化的代价函数有:

系统通信代价 $IPC(X)=\Sigma_{k=1}^{N}IPC(k,X)$. 在网络中,它是不可忽略的.

系统总运行和通信代价 $EC(X)=\Sigma_{k=1}^{N}\{IPC(k,X)+AET(k,X)+\Sigma_{i=1}^{M}ec_ix_{ik}\}=\Sigma_{k=1}^{N}Load(k,X)$,在以提高系统的吞吐率和资源利用率为目标的非实时应用中,常被采用.

工作完成时间 $ET(X)$,实时应用要求完成时间尽可能短.

任务分配是在满足以上各种限制条件的基础上,寻找使代价函数最小的一个从任务图到系统图的最优映射 π.

1.2 通信规划模型

定义 9. 处理器 k 的干扰开销 $Interf(X,k)$ 它是由于分配到处理器 k 的各个任务竞争处理器资源(*CPU*,内存和 *I/O* 等)而产生的,将导致系统性能下降,其突出表现为由于任务竞争 *I/O* 而使处理器处于空闲等待状态.

定义 10. 系统干扰开销 $Interf(X)=\Sigma_{k=1}^{N}Interf(X,k)$.

定义 11. 任务 k 的内部通信集 $I(K)=\{c_{k1},c_{k2},\ldots,c_{kt_k}\}$,且 $t_i\neq k(i=1,2,\ldots,k)$, $c_{kt_i}\neq 0$.

定义 12. 系统内部通信集 $I=\bigcup_{k=1}^{M}I(k)$.

通信规划可描述为寻找一个使系统干扰开销最小的映射 γ,使 $I\rightarrow\{1,2,3,\ldots,\Sigma_{k=1}^{M}t_k\}$,且若 $\gamma(e_i)<\gamma(e_j)$ $e_i,e_j\in I$ 则 e_i 在 e_j 前先执行. 即 $\min_\gamma\{Interf(X)\}$.

2　启发式算法

任务分配和通信规划属于组合优化问题,具有指数级复杂度. 以下讨论如何用启发式算法来逼近其最优解. 假定:网络中各个处理器是同构的;任务间没有优先权限制;基于分布存储多处理系统讨论,同一处理器上的任务间通信开销为零;选择完成时间 $ET(X)$作为代价函数. 由于系统负载平衡能缩短响应时间,且实时限制条件很难精确刻划,故用负载平衡条件代替实时限制条件. 本文讨论的启发式算法就是在满足系统负载平衡的约束条件下,使工作完成时间 $ET(X)$最小. 算法的目标是寻找一个使这 N 个处理器中工作负载最大的处理器之中工作负载最小的一个映射 X,即 $OF_{\text{minmax}}=\min_x\{\max_{1\leqslant i\leqslant N}\{Load(i,X)\}\}$.

这样简化任务分配模型能减小处理器间的通信开销以及在整个处理器网络内实现各处理器的工作负载平衡,满足实时限制. 为表达方便用 c_{ii}表示任务 i 的执行时间且定义任务分配集 $P_i=\{k\mid x_{ki}=1,k=1,2,\ldots,M\}$,用 $Load_i$ 表示 $Load(i,X)$. 这样,目标函数可以表达为:　　$OF_{\text{minmax}}=\min_x\{\max_{1\leqslant j\leqslant N}\{\Sigma_{k\in P_j}c_{kk}+\Sigma_{k\in P_j}ec_k+\Sigma_{k\in P_j,l\notin P_j}(c_{kl}+c_{lk})\}\}$

2.1　任务分配

任务分配包括任务初始分配和通信链路切割 2 步. 任务初始分配在简化的模型基础上,将任务分配到处理器网络上. 但它是在每个处理器可以和其余的处理器互连的假定条件下进行的,实际情况是每个处理器的通信链路数目是有限的,故还要进行通信链路切割.

2.1.1　任务初始分配

将 M 个任务分成 N 组,每组分配到一个处理器上而不考虑并行机结构模型的限制,如每个处理器的最大通信链路数目限制. 它属于最优搜索算法,但能否找到最优解,取决于初始的 N 个任务如何选择,即如何产生初始分配.

算法 1. 任务初始分配

(1) / * :将所有任务按执行时间从大到小的顺序排列,将前 N 个任务一一对应地分配到处理器上 * /

$P_i=\{i\},R=\{N+1,N+2,\ldots,M\},(i=1,2,\ldots,N)$

(2) 循环直到 R 为空　/ * 找到 k、l,使任务 k 加到处理器 l 上时,处理器 l 的工作负载增加最少 * /

$D_i=ec_k-c_{kk}+\Sigma_{j\in P_i}(c_{kj}+c_{jk})-\Sigma_{j\in P_i}(c_{kj}+c_{jk})(i=1,2,\ldots,N,k\in R)$

$D_l=\min_{1\leqslant i\leqslant N}\{D_i\},\Sigma_{j\in P_j}(c_{kj}+c_{jk})\neq 0$

/ * 将任务 K 分配到处理器 l 上去,同时改变处理器 l 的工作负载 * /

$P_l=P_l+\{k\},R=R-\{k\},Load_l=Load_l+D_l$

(3) / * 将所有处理器按工作负载从小到大的顺序排序 * /

$P_1,P_2,\ldots,P_N,Load_1\leqslant Load_2\ldots\leqslant Load_N$

(4) / * 将任务 S 从处理器 N 中移出,分配到处理器 v 上去 * /

$Load'_k=Load_k+[ec_i-c_{ii}+\Sigma_{j\notin P_k}(c_{ij}+c_{ji})-\Sigma_{j\in P_K}(c_{ij}+c_{ji})](i\in P_N)$

$Load'_N=Load_N-[ec_i+3c_{ii}+\Sigma_{j\notin P_N}(c_{ij}+c_{ji})-\Sigma_{j\in P_N}(c_{ij}+c_{ji})](i\in P_N)$

$\max(Load'_v,Load'_N)=\min_{i\in P_N,1\leqslant k\leqslant N-1}\{\max(Load'_k,Load'_N)\},\Sigma_{j\in P_V}(c_{sj}+c_{js})\neq 0$

$Load'_N$ 为将任务 i 从处理器 N 去掉后处理器 N 的工作负载,

$Load_k$ 为将任务 i 加到处理器 k 后处理器 k 的工作负载

(5) /* 判断本阶段是否结束 */

若 $max(Load_V, Load_N) > Load_N$，则转通信链路切割阶段

否则[$P_v = P_V + \{s\}$，$P_N = P_N - \{s\}$，$Load_V = Load_V$，$Load_N = Load_N^s$ 转(3)]

2.1.2 通信链路切割

它根据处理器的链路数目限制，去掉多余的处理器间的链路连接，对每条被切割链路，将链路两端的 2 个处理器的任务之间的通信转移到连接此 2 处理器的一条最短路径上. 它构造了一个“真实”的满足处理器的链路数目限制的处理器网络.

2.2 通信规划启发式算法

它对已分配在各处理器上的任务之间的通信进行规划，确定任务间的数据通信次序. 基本思想是每一次规划时，对当前逝去时间(通信规划算法执行过程中，某一时刻某一处理器已经占用的时间，当算法结束时，它表示该处理器的结束时间)最小的 2 个处理器上的 2 个通信开销最大的任务进行规划. 如 2 处理器的逝去时间不等，则这 2 个任务需要进行同步，此时一处理器将处于空闲等待状态. 算法尽量使每一次规划时各个处理器的逝去时间相差不大，因而就尽可能的减少了处理器的等待时间. 同时由于开销大的通信都在前面规划，遵循这个次序进行规划可以保证任务间的通信不会产生死锁.

算法 2. 通信规划算法

(1) /* 初始化，各个处理器先完成自己的计算任务 */

$ET_j = \Sigma_{i \in P_j} c_{ii}$，$Id_i = 0 (i \leqslant M)$，$Id_i = 1 (i > M)$，$PM_1 = \{1, 2, \ldots, N\}$，$IT_j = 0$

ET_j，IT_j 分别为处理器的逝去时间和空闲时间，Id_i 为任务 i 的优先权指示器，通信链路切割阶段增加的中继任务的值为 1，表示要等其前的任务完成后本任务才能进行调度

(2) /* 求逝去时间最小的处理器集 $M1$ */

$M_1 = \{i \mid ET_i = \min_{k \in PM_1}\{ET_k\}, NL_i \neq 0\}$，$PM_2 = PM_2 - M_1$；

若 M_1 为空 /* 表明各个任务间的通信任务已完成，现进行外部通信 */

$ET_j = ET_j + \Sigma_{i \in P_j} ec_i (j = 1, 2, \ldots, N)$，结束

否则 若 M_1 只有一个元素 转(4)

(3) /* 求 M_1 中，位于同处理器上且通信开销最大的任务 s 和 r */

$c_{sr} = \max_{i, j \in M_1, i \neq j}\{\max_{k \in P_i, l \in P_j}(c_{kl})\}$ 且 $Id_s = 0 (s \in P_u, r \in P_v$，且 $u, v \in M_1)$

若 c_{sr} 不为零，则转(6)

(4) /* 求逝去时间第二小的处理器集 M_2 */

$M_2 = \{i \mid ET_i = \min_{k \in PM_2}(ET_k), NL_i \neq 0\}$

若 M_2 为空 [$PM_1 = PM_1 - M_1$，转(2)]

(5) /* 求 M_1 和 M_2 间，通信开销最大的任务 s 和 r，任务 s 和 r 间要进行同步数据传输 */

$c_{sr} = \max_{i \in M_1, j \in M_2}\{\max_{k \in P_i, l \in P_j}(c_{kl})\}$ 且 $Id_i = 0 (s \in P_u, r \in P_v$，且 $u, v \in M_1$ 或 $M_2)$

若 c_{sr} 为零 [$PM_2 = PM_2 - M_2$，转(4)]

否则/* 任务 s 和任务 r 间要等待，有一处理器空闲 */

若 $v \in M_1, u \in M_2$ [$IT_v = IT_v + (ET_u - ET_v)$，$ET_v = ET_u$]

否则 [$IT_u = IT_u + (ET_v - ET_u)$，$ET_u = ET_v$]

(6) /* 计算逝去时间 */

$ET_u = ET_u + c_{sr}$，$ET_v = ET_v + c_{sr}$，$c_{sr} = 0$，$Id_r = 0$

若 $\Sigma_{i \in P_u, j \in P_v} c_{ij} = 0$ [$NL_u = NL_u - 1$，$NL_v = NL_v - 1$，转(2)

否则 [$PM_1 = \{1, 2, \ldots, N\}$，转(2)]

3　NUCAPPT及使用

图2中虚框内的部分即为NUCAPPT,它由4部分组成:任务分配器、通信规划器、通信表和通信语句生成器.其核心为任务分配和通信规划启发式算法.任务分配器能保证任务分配后系统负载比较平衡,响应时间比较短,决定了并行程序的基本性能.通信规划器对每一处理器生成一通信表.通信表代表一个动作序列.它记录了处理器的任务在哪一时刻和哪个处理器上的哪个任务进行数据通信,以及它们的通信时间等.程序员根据通信规划器生成的通信表手工编制程序,或者由通信语句生成器产生仅含这些通信语句的程序框架,在编码阶段在程序框架中插入计算代码,从而组成程序.产生的程序经过调试测试后如还不满足系统要求.根据这些反馈信息可以由程序员手工或任务分配器再进行任务分配和通信规划.从而改进所编并行程序的质量.

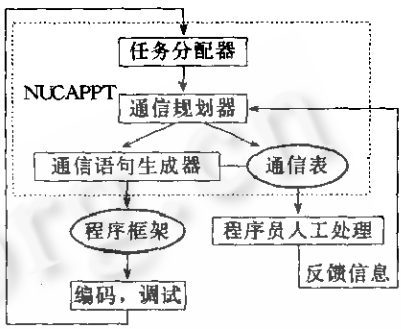

图2　NUCAPPT构成及其使用

例中,有15个任务要分配到6个处理器上,代价矩阵C已知,每个任务的外部通信费用均为零.

表1为任务分配后,各个处理器的工作负载及它所分配的任务集.可以看出任务分配后系统负载比较平衡.表2为通信规划后处理器1的通信表(表中带负号的任务为接收数据的任务.不带负号的为发送数据的任务.表中每列后3行元素为0表示此任务处于等待状态,最后2行元素为0表示此任务处于执行状态.内部任务表示该处理器上的任务.外部任务为和该处理器上的任务通信的外部处理器上的任务).表2说明了处理器1上的每个任务何时从哪个处理器上的哪个任务发送或接收数据,在并行程序设计时就可以根据此通信表安排处理器1上的任务的通信次序.

表1　任务分配结果

处理器	工作负载	任务集	处理器	工作负载	任务集
p1	150	11,2	p4	165	6,1,3
p2	155	8,5	p5	170	4,12
p3	165	13,15,14	p6	165	7,10,9

表2　处理器1的通信表

动作序列	1	2	3	4	5	6	7	8	9
动作时间	30	10	10	20	5	30	20	20	10
内部任务	11	2	2	2	0	2	11	−11	−2
外部任务	0	0	−5	−10	0	−4	−12	1	1
外部处理器	0	0	2	6	0	5	5	4	4

图3为整个系统的通信规划图,图的右边列出了各个处理器的结束时间和空闲时间.它说明整个任务分配和通信规划是十分有效的,系统负载比较平衡,处理器的空闲时间(图中的阴影部分)很小,处理器利用率较高(在本例中处理器利用率为92.8%).因此根据由各个处理器的通信表确定的任务间的通信先后次序设计出来的并行程序质量是比较高的.

设通信原语为:Receive(外部处理器,外部任务,...);Send(外部处理器,外部任务,...);则对处理器1的2个任务2和11,根据表1就可产生如下程序框架:

任务2:send(2,5,...); send(6,10,...);send(5,4,...); receive(4,1,...);

任务11: send(5,12,...); receive(4,1,...);

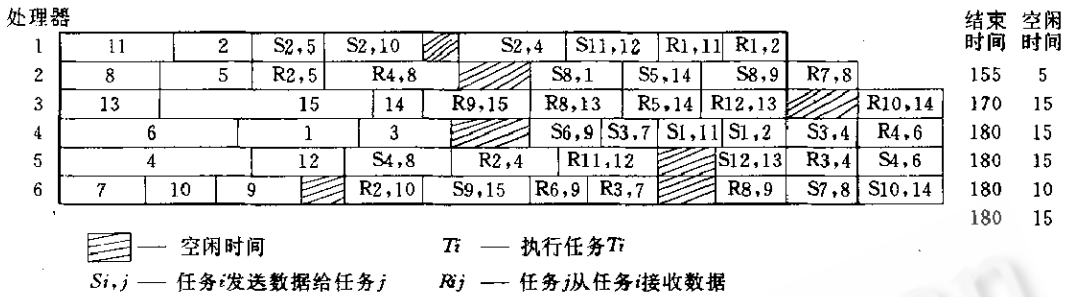

图3 通信规划结果图

4 结束语

本文讨论的并行程序辅助开发工具 NUCAPPT 由任务分配器、通信规划器、通信表和通信语句生成器 4 个部分组成，其核心为任务分配和通信规划启发式算法. 利用它程序员不需了解任务如何分配以及如何安排并行程序的数据通信等繁琐细节. 它能保证根据通信语句生成器产生的程序框架编程或者根据通信表手工编程，所编制的并行程序质量是较高的.

参考文献

1 Wu S S, Sweeting D. Heuristic algorithms for task assignment and scheduling in a processor network. Parallel Computing, 1994, (20): 1～14.

2 Tao L, Zhao Y C. Efficient heuristics for task assignment in distributed systems. Proceeding of 1992 International Conference on Parallel and Distributed Systems, 1992. 16～18.

A SOFTWARE TOOL FOR PARALLEL PROGRAMMING

Xu Hong Chen Delai Zhang Defu

(State Key Laboratory for Novel Software Technology at Nanjing University
Department of Computer Science and Technology Nanjing University Nanjing 210093)

Abstract This paper discusses a parallel programming tool based on the task assignment and communication planning heuristic algorithms. It consists of four components: task allocator, communication planner, communication schedule, and communication statement generator. A program framework composed of communication statements can be generated and used at the programming phase. By using this tool, developing parallel programs will become easier and it is certain that high efficient programs can be produced.

Key words Parallel programming tool, task assignment, communication planning.