一种分布式动态负载平衡算法

须成忠 张德富 孙钟秀

(南京大学,南京 210008)

A DYNAMIC AND DISTRIBUTED ALGORITHM FOR LOAD BALANCING ON MULTIPROCESSORS

Xu Chengzhong, Zhang Defu and Sun Zhongxiu

(*Nanjing University, Nanjing* 210008)

Abstract On a multiprocessor, load balancing is an essential technique to improve the performance of parallel processing by efficiently utilizing the processing power of the system. In this paper, we propose a dynamic and distributed algorithm for load balancing on TRANSCUBE, a multiprocessor system without shared memory. The algorithm adopts receiver-initiated asychronous strategy by which an idle processor first initiates a "hadshake" procedure to locate a heavily loaded processor, then absorbs some work from it. This algorithm is further demonstrated in the context of parallel solving of a maze problem. Simulated results show that it is effective and is more applicable to large size problems.

摘要 在多处理机系统中,负载平衡是提高并行处理效率的一条重要途径.基于分布存贮的TRANSCUBE多处理机环境,本文提出一种分布式动态负载平衡算法.算法采用接收者开始的异步调度策略,通过"握手"协议在空载和重载处理机间建立联系,并自动实现任务(或进程)从重载处理机到空载处理机的迁移.该算法适于并行解具有动态特性的应用问题,而且在问题规模较大和处理机负载变化较慢时,性能较好.

§1. 引 言

在多处理机系统中,任务分配和调度的一个主要目标是平衡各处理机间的负载,并以此提高并行处理的效率[1].静态任务分配是指任务在计算前就被指派给确定的计算机.这种分配方式虽然简单,但是它很难适用于那些具有动态特性的应用问题.在那些问题中,任务的计算量较难预先估计,而且任务间的相互作用和依赖关系又是动态可变的.为了提高这类问题的并行

本文1990年6月18日收到,1991年2月2日定稿.本课题得到国家863计划的资助.作者**须成忠**,1989年硕士毕业于南京大学,现为博士生,主要研究领域为并行处理、计算机软件.**张德富**,教授,主要研究领域为并行处理、分布式计算.**孙钟秀**,学部委员,教授,主要研究领域为计算机软件、分布式计算、并行处理.

© 中国科学院软件研究所 http://www.jos.org.cn

求解效率，系统需要有任务动态调度的能力，允许任务（或进程）在计算过程中在处理机间迁移.动态负载平衡就是一种典型的调度策略.它是指在问题并行求解过程中，系统根据处理机中的任务执行情况进行动态调度，通过进程迁移，重新平衡处理机间的计算负载.

过去，动态负载平衡算法在分布处理环境中得到较多的研究[2][3][4]，但是其中大多数的讨论都是基于广播通信网络结构，而且针对一组没有数据和控制依赖关系的任务.本文基于分布存贮的多处理机系统，提出一种分布式动态负载平衡算法（简称 DDLB）算法.

TRANSCUBE 是我们构筑的一个实验型的多处理机系统．它的核心是一个由多个 Transputer[5] 组成的超立方体（Hypercube）[6]结构的多处理机阵列.每个处理机结点包含一个 32 位的 T414 单片处理机，1M 字节内存及 4 个双向 I/O 链路端口，处理机结点按 BRGC 技术编码，处理机间的通信通过信件传递实现.

§2. 算　法

现先对算法中涉及的几个基本概念作如下解释：

(1)负载：处理机上运行的用户进程未完成的工作量.处理机负载的计算与应用有关，而且由于用户进程的许多动态特性，负载计算只能按一定规则估值.

(2)负载阀值：为了度量处理机负载轻重而预先设定的一个界限值.设负载阀值为 J，处理机 P_i 的负载为 Q.如果 $Q>J$，P_i 定义为重载；如果 $0<Q\leqslant J$，则 P_i 为轻载；如果 $Q=0$，则 P_i 为空载.当 P_i 为空载时，P_i 处于 Idle 状态，否则 P_i 处于 Busy 状态.

(3)负载平衡：当系统中所有处理机都处于 Busy 状态，或者当某个处理机处于 Idle 状态，系统不存在重载处理机时，该系统状态称为负载平衡.

DDLB 算法采用接收者开始的异步策略，即处于 Idle 状态的处理机向外申请负载.为了便于说明，将算法分为两个过程，首先是“握手”过程，空载处理机经过“握手”协议找到某个重载结点，从而两者建立联系；然后是“迁移”过程，重载处理机派生用户子进程，将该子进程迁往已建立联系的空载处理机上计算.

2.1 “握手”过程

“握手”过程的主要任务是在处理机间建立高效、可靠的通信协议（“握手”协议）.设在某一时刻 P_k 为空载处理机，P_k 向外广播一封 DRAFTING 信，信体包括项 Source，Pid 和 Ld，它们分别代表信源，应征处理机号及应征处理机负载.DRAFTING 信按 BRGC 逻辑环顺序在处理机间逐个传递.设有两个重载处理机 P_i 和 P_j 先后收到该 DRAFTING 信，P_i 首先应征，在 DRAFTING 信的 Pid 和 Ld 中分别填入自己的处理机号和自己的负载估值.如果 P_j 的负载大于 P_i，则 P_j 发 REJECT 信给 P_i，告诉 P_i 应征失败，同时 P_j 将 DRAFTING 中 Pid 和 Ld 改为自己的值.当 DRAFTING 信走完全程回到 P_k 时，P_k 向信中 Pid 标识的应征者发 ADMISSION 信.

为了方便表示“握手”协议，除了 Busy 和 Idle 状态外，我们还引入下面四个处理中间状态.

- Wait For Ack：处理机等自己的 DRAFTING 回来.
- Wait For Admission：处理机应征后等候 ADMISSION 信.
- Ready For Accept：处理机准备接收进程.

© 中国科学院软件研究所 http://www.jos.org.cn

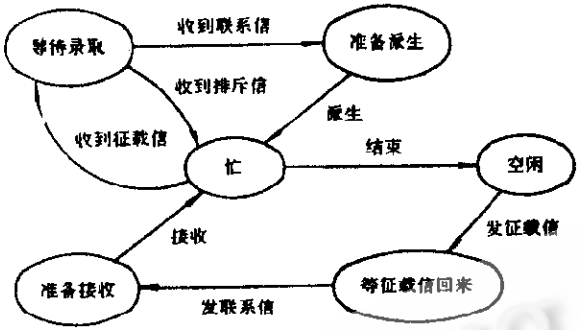

图1 状态转换图

• Ready For Split:处理机准备派生子进程并迁出.

"握手"协议可直观表示为处理机状态转换图如图1所示,当两个分别处于Ready For Split和Ready For Accept状态的处理机"握手"后,再分别迁出/入进程,随后都进入Busy状态."握手"协议算法表示为一组处理机的状态转换规则,其算法如下:

```
    Algorithm for processor Pi
Rule1:  if state=Free
        begin DRAFTING.source:=i;
              DRAFTING.pid:=i;
              DRAFTING.ld:=0;
              send (DRAFTING) to next;
              state:=WaitForAck;
        end

Rule2:  if state=Busy and receive (DRAFTING)
        begin Q:=loadvaluate; /* load calculation */
              if (Q=<DRAFTING.ld)
                  send (DRAFTING) to next
              else if (Q=<J) /* J is threshold */
                  send (DRAFTING) to next
              else if (DRAFTING.pid=0)
                  begin DRAFTING.pid:=i;
                  DRAFTING.ld:=Q;
                  send (DRAFTING) to next;
                  state:=WaitForAdmission;
                  end
              else begin
                  send (REJECT) to DRAFTING.pid;
                  DRAFTING.ld:=Q;
                  DRAFTING.pid:=i;
                  send (DRAFTING) to next;
                  state:=WaitForAdmission;
              end
        end
```

© 中国科学院软件研究所 http://www.jos.org.cn

```
Rule3: if state=WaitForAck and receive (DRAFTING)
          if DRAFTING.source=i and DRAFTING.ld≠0
             begin
             send (ADMISSION) to DRAFTING.pid
             state:=ReadyForAccept;
             end

Rule4: if state=WaitForAdmission
          if receive (ADMISSION)
            state:=ReadyForSplit
          else if receive (REJECT)
             state:=Busy;
```

2.2 "迁移"过程

在进行迁移前，首先要对处于 Ready For Split 状态的处理机上的用户进程进行分裂. 进程分裂是将进程划分成若干可并行执行的子进程，它们协作完成父进程正在计算的任务. 进程迁移是改变进程的运行环境，即将进程从一台处理机迁移到另一台处理机上运行. 由于"迁移"过程与应用问题密切联系，这里只给出进程分裂和迁移的一般性方法. 在下一节，将以迷宫问题为例作详细说明.

在一般情况下，进程分裂和迁移要同时考虑程序代码，数据和工作区状态等三个方面的内容. 但是由于大多数问题的并行求解实际上只利用了问题本身蕴含的算法并行性和几何并行性的某一个方面. 或者是多个功能不同的任务在相同的数据基上计算，或者是多个功能相同的任务在不同的数据基上计算. 所以进程迁移时，在前一种情况下仅需传送程序代码和工作区状态；后一种情况只需传送代码作用的数据场和工作区状态. 例如，在状态空间搜索问题的并行解中，直观的方法是利用该问题的几何并行性. 各处理机运行相同的搜索程序，分别对不同的状态空间区进行搜索. 因而进程分裂和迁移只需考虑状态空间的划分和移动.

另外，由于 TRANSCUBE 是同构型的多处理机系统，所以在进程迁移时，除了数据传递外，只需在迁出和迁入的处理机上做些简单的薄记工作，不需要改变代码和变换数据格式.

§3. 并行解迷宫问题

本节我们以迷宫问题为例进一步说明 DDLB 算法的应用.

迷宫问题是一种典型的状态空间搜索问题. 迷宫问题求解是在迷宫中找出一条从入口到出口的通路. 设用方阵 $MAZE_{dim*dim}$ 表示迷宫，dim 为迷宫的阶. 设变量 X，Y 是两个方向上的变量.

$$MAZE[x,y]=\begin{cases}0:\text{表示该点}(x,y)\text{ 可通过}\\1:\text{表示点}(x,y)\text{ 不能通过}\\2:\text{表示点}(x,y)\text{ 是围墙}\end{cases}$$

MAZE[0,0]为入口，MAZE[dim-1,dim-1]为出口，迷宫的路径从 MAZE[0,0]开始，只要到达 MAZE[dim-1,dim-1]便算走出迷宫. 图 2 是一个阶为 16 的迷宫方阵.

© 中国科学院软件研究所 http://www.jos.org.cn

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	0	2
2	1	0	0	0	1	1	1	1	1	1	1	1	1	0	1	0	2
2	1	0	1	0	1	1	0	0	0	0	1	1	0	1	1	0	2
2	1	0	1	0	1	1	0	1	1	1	1	0	1	1	1	0	2
2	1	0	1	0	1	1	0	0	0	0	0	1	1	1	1	1	2
2	1	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	2
2	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	2
2	1	1	1	0	1	1	0	1	0	1	0	0	0	0	0	0	2
2	0	0	0	0	1	1	0	0	0	1	0	1	1	1	1	1	2
2	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	2
2	1	1	1	0	1	1	1	1	1	0	1	0	1	1	1	1	2
2	1	1	0	1	1	0	0	0	0	1	1	1	0	0	0	1	2
2	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	2
2	0	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	2
2	1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	1	2
2	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

图 2 迷宫 dim=16

迷宫图的通路是(0,0),(1,1),(1,2),(1,3),(2,3),(3,3),(4,3),(5,3),(6,3),(6,4),(6,5),(6,6),(5,6),(4,6),(4,7),(4,8),(4,9),(4,10),(5,10),(6,10),(7,10),(8,10),(9,10),(10,11),(11,12),(12,12),(13,12),(14,12),(14,13),(14,14),(15,15).

在迷宫中搜索前进时,可在八个不同的方向上试探有无可行之路,这八个方向顺序编号如图 3. 每个方向上 X 和 Y 的增量分别是

$$dx=[0,-1,-1,-1,0,1,1,1]$$
$$dy=[1,1,0,-1,-1,-1,0,1]$$

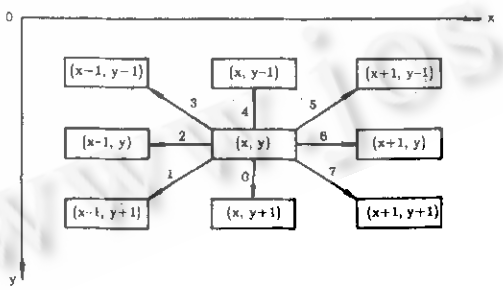

图 3 方位图

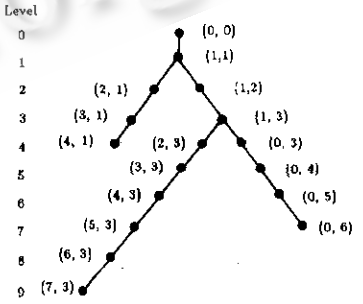

图 4 搜索树

按深度优先次序搜索迷宫,首先扩展新产生的结点,深度相同的结序按序排列. 该搜索过程可用搜索树表示(图 4 括号中的坐标值与图 2 是对应的).

现设有 N 个处理机对迷宫某位置 (X, Y) 下的若干分支路径进行并行搜索, P_k 为父进程结点, P_i 和 P_j 的搜索范围分别是 ($base_i$, $upper_i$) 和 ($base_j$, $upper_j$), 变量 base 和 upper 分别表示搜索方位的上下界. 若 P_i 首先发现在 ($base_i$, $upper_i$) 内无解, P_i 向 P_k 报告失败结果,

© 中国科学院软件研究所 http://www.jos.org.cn

同时再向其它处理机征载. 若 P_j 的负载最重，则 P_j 应征成功. P_j 将自己的搜索空间分一半给 P_i 搜索，这样，处理机间完成一次负载平衡.

在负载平衡过程中，处理机负载的估算是影响问题并行求解效率的又一次关键因素. 设变量 level 表示结点在搜索树中的深度，路径栈 Pathstack 记录可行路径. 栈中记录：(x,y,base,upper,n)，n 表示点(x,y)下在(base upper)间尚未尝试的路径数. n＝0 表示点(x,y)下在(base,upper)间的所有方位都尝试过，过程 Loadvaluate 给出一种以 Pathstack 的当前状态来度量处理机负载的函数.

```
Procedure loadvaluate (top)
begin
    level: =0;
    while pathstack [level]. n=0 and level=<top
          level: =level+1;
    if level>top return (0)
    else return (top—level + pathstack [level]. n/8);
end
```

在并行搜索过程中，设系统的负载阀值 J＝0.125，当搜索空间只剩下当前结点下一条未尝试路径时 (level＝top 且 n＝1)，处理机为轻载. 当搜索空间中已搜索过的结点下还有未尝试路径(level＜top)或者当前结点下还有多于一条未尝试路时，处理机为重载. 因为 Pathstack [levle]，n＜8，所以在表达式 (top—level＋pathstack [level]. n/8) 中，level 越小负载值越大. 在 DDLB 算法实现中，尽可能使调度算法独立于应用程序.

§4. 算法分析

下面以迷宫问题为例分析 DDLB 算法的开销和并行处理效率.

DDLB 算法开销包括“握手”协议和进程“迁移”两部分开销. 影响“握手”协议开销的关键是 DRAFTING 信的广播方式和负载估算函数. DRAFTING 信在 TRANSCUBE 中按逻辑环顺序传递. 设 N 是 TRANSCUBE 中处理机数，d 为网络直径且 d＝[log N]，t 是相邻两个处理机间一次信件传递的时间，w 是负载估算时间. 某重载处理机 P_j 在接到空载结点 P_k 的 DRAFTING 信后到两者建立联系的平均时间开销为

$$N/2*t+N/2*w+d/2*t$$

在迷宫问题中，W 的复杂性是 O(dim)，dim 是迷宫的阶.

进程“迁移”开销包括进程分裂和数据传输两部分. 在迷宫问题中，进程分裂仅需进行一些簿记工作，不涉及链路通信，也无需变换数据格式. 因此进程“迁移”的开销主要反映在数据传输上. 数据量为 $O(dim^2)$.

设迷宫中每个位置的搜索时间相同，并以此为单位度量搜索过程的长度. 在解图 2 迷宫时，一个处理机按深度优先次序搜索的长度 $L_1＝67$.

现设有两台处理机 P_1 和 P_2 并行搜索迷宫，开始时 P_1 在位置(0,0)，P_2 空载并向 P_1 征载. 不难验证，按照 DDLB 算法调度，用两台处理机并行搜索的长度 $L_2＝47$. 其中包括 10 次负载平衡. 所以在理想情况下的并行搜索的加速比 $S_2＝67/47＝1.43$，并行处理效率 $E_2＝0.715$. 同

© 中国科学院软件研究所 http://www.jos.org.cn

样，用四台处理机搜索图2迷宫时，搜索长度 $L_4=33$，其中包括13次负载平衡.理想情况下的加速比 $S_4=67/33=2$，并行处理效率 $E_4=0.5$.

由于负载平衡开销的存在，实际效果要比理想情况差.但是当问题规模增大或每个计算步的计算量增大时，各处理机负载增大，且负载变化缓慢，负载平衡次数减少，并行处理效率将会提高.

结论：DDLB算法是我们基于超立方体结构的多处理机系统TRANSCUBE提出的一种分布式动态负载平衡算法.算法采用接收者开始的异步调度策略，通过"握手"协议在空载和重载处理机间建立联系，并自动实现任务(或进程)从重载处理机到空载处理机的"迁移".该算法是全分布的，不依赖于任何主控制器或共享变量的假设，因而也特别适合于不带公共存贮器的多处理机系统；它适用于具有动态特性的应用问题的并行解.特别是在问题规模较大，处理机负载变化较慢时，算法表现出较好的性能.

参考文献

[1]T. H. Casavant and J. G. Knhl, "A Taxonomy of Scheduling in General Purpose Distributed Computing Systems", IEEE Trans. on Software Engineering, Vol. 14, No. 2, Feb. 1988.

[2]D. L. Eager, E. D. Lazowska and J. Zahorjan, "A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing", Performance Evaluation, Vol. 6, No. 1, Mar. 1986.

[3]A. Hac and X. Jin, "Dynamic Load Balancing in a Distributed System Using a Sender Initiated Algorithm", Proc. 13th Conf. on Computer Networks, Oct. 1988.

[4]L. M. Ni, C. W. Xu and T. B. Gendreau, "Drafting Algorithm; A Dynamic Process Migration Protocol for Distributed Systems", Proc. 5th Int. Conf. Distributed Comput. Sys., May 1985.

[5]C. L. Seitz, "The Cosmic Cube", Communication of ACM, Dec. 1984.

[6]"The Transputer Family", INMOS Product Information, 1986.

"第四届全国计算机系统软件学术研讨会"纪要

受中国计算机学会软件专业委员会系统软件学组委托，由电子科技大学计算机系承办的"第四届全国计算机系统软件学术研讨会"于1992年10月8日—11日在四川省成都市新都县召开，出席会议的有全国23个单位的44名代表。

会议共收到论文73篇，内容涉及操作系统、编译系统及相关领域等多个方面，总结了近年来所取得的成果，不少文章有创新，有新意，大会共报告了13篇。

与会代表还就我国系统软件目前取得的进展、成果、存在问题及今后的发展进行了广泛深入的讨论，一致认为，我们的软件产品包括系统软件应尽快打入国际市场，同时可选择国际上流行的主流系统软件作为我国软件发展的基础，以此带动整个软件产业的发展。

与会代表对电子科技大学的热情接待，对新都县的大力支持表示衷心的感谢。

© 中国科学院软件研究所 http://www.jos.org.cn