国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第14卷第3期 1992年9月 Vol. 14 No. 3

基于SBM的操作级并行处理算法研究

文涤屏 杨学军 陈立杰
（电子计算机系）

摘 要 SBM是支持操作级并行的一种有效的同步机制。文中基于SBM对结点调度和barrier插入算法进行了深入的研究，提出了一套有效的开发操作级并行的方案。用一有向图G(N,A)表示指令之间的相关关系，并以结点的临界路径为关键字将结点从小到大进行排序。按照排序后的结点顺序，描述了一种分配算法，将结点分配给各处理机。同时，描述了相关结点之间的barrier插入算法。

关键词 多处理机系统，同步，用户程序，指令系统

分类号 TP311.5

并行处理是提高计算机速度和性能最有效的方法。多处理机系统是并行处理发展的必然结果。为了充分开发多处理机系统的潜力，必须识别出用户问题的并行因素，把该问题的求解过程分为多个任务并用并行算法描述，然后分配到不同的处理机上去执行。

用户程序的并行性主要体现在五个层次：作业级、作业步级、子程序级、循环级和操作级（指令级）。根据美国1990年ICPP会议最新资料统计，以操作级并行的并行度最高；但因并行粒度小，所以必须有高效硬件同步原语的支持，才能使程序运行获得很高的加速比。

Barrier同步原语是实现各种层次上的并行处理的常用的方法之一。为支持操作级的并行处理，可用软件程序、硬件机制等各种方法设计barrier原语。文献1提出了一种新的、高效的同步机制SBM(Static Barrier MIMD)，通过软件设置barrier标志、硬件核对标志的方法实现进程的子集的barrier同步。基于这种同步机制，本文研究并实现了结点（指令）分配及barrier插入算法。

1 背景及概念

SBM实现进程同步的方法是在各进程的每一同步点设置一条barrier指令，只有当所有需在这一点同步的进程都到达同一barrier点时，这条barrier指令才能流出，然后执行下一条指令。

如图1所示，只有当5个进程都到达b_0时，b_0才能执行；b_1须等p_0、p_1到达时才能执行。

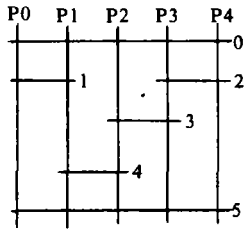

图1

* 1991年5月5日收稿

可以通过一个二元关系的集合$(B,\langle_b)$表示 barrier 的执行顺序。设 $b_1 \langle_b b_2$，则表示 b_2 要在 b_1后面执行。如图1，就有 $b_0 \langle_b b_1$，$b_0 \langle_b b_2$，$b_2 \langle_b b_3$. 显然，这种关系是非自反及传递的。

所有满足 $b_i \langle_b b_j$ 的关系集合 S 称线性关系。一个线性关系对应一个单一同步流。如果对所有 b_i，$b_j \in A$ 都有 $b_i \sim b_j$，则 A 是一个反链集。这里，$b_i \sim b_j$ 表示 $\neg(b_i \langle_b b_j)$及 $\neg(b_j \langle_b b_i)$. 显然，A 中的 barrier 是可以并行执行的。

分析指令级并行必须首先知道指令之间的相关信息。可以用一个有向图来表示这种指令的相关性。

定义一有向图 G(N,A)，其中 $N=\{s_1,s_2,\cdots,s_n\}$代表指令的集合，$A=\{e_{ij}=(s_i,s_j) \mid s_i,s_j \in N$，且 s_i 与 s_j 相关$\}$代表有向边集合，则称 G 为相关关系图。若$(s_i,s_j) \in A$，则称 s_j 是 s_i 的后继结点(consumer)，s_i 是 s_j 的前趋结点(procedure).

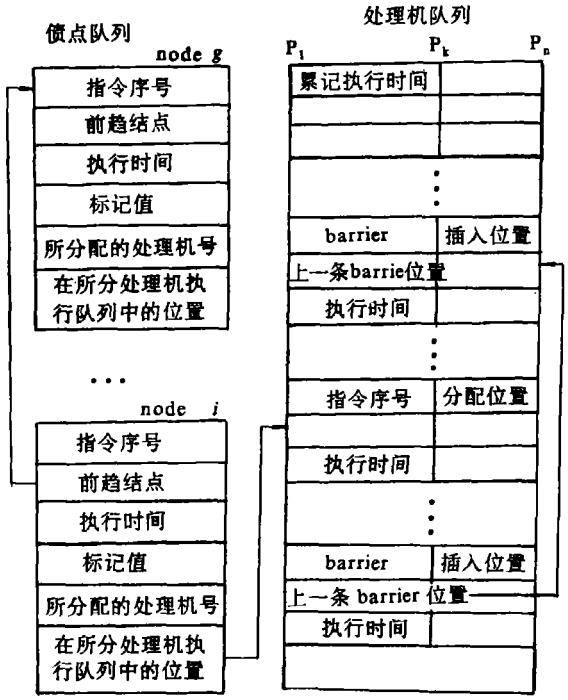

图2

每个结点即每条指令的执行时间，表示为 t(i). 结点 i 的标记值定义为从入口结点到此结点的最长路径。

$$\text{label}(i) = \max \sum_{j \in \pi_k} t(j)$$

π_k 表示从入口结点到此结点的第 k 条路径。

对每个结点作完标记后，可以以结点标记值为关键字，将结点从小到大进行排序。按照结点分配算法，分给排序后的结点以处理机，在需要插 barrier 的地方插一条 barrier 指令。

2 算法

依次简单或详细地描述标记结点、结点排序、结点分配及 barrier 插入的算法实现。

主要数据结构如图2所示。

首先计算从入口结点到各结点的最长路径，并由此给出各结点的标记值。假设有 m 个结点，结点标记值的初值取结点的执行时间。对每一个结点 i，取 i 的每一个前趋结点 g，将 g 结点的标记值加上 i 结点的执行时间，并与 i 结点的标记值作比较，取较大者为 i 结点的新标记值。对所有结点做完此类计算后则计算出各结点的标记值。

计算好各结点的标记值后，以标记值为关键字，将结点用通常的快速排序法进行重排序，排序后的结点仍放在 Vnode[1…m]中。

依照排序后的结点顺序，开始进行结点分配。对每个结点，首先判其前趋结点是否为所分处理机的最后一条指令。如果是，则将此结点分配给该处理机。当有多个满足此条件的前趋结点时，取分配给它们的处理机中累计执行时间最长的处理机进行分配。这样便于少插入 barrier 指令；，没有满足此条件的前趋结点，则将此结点分配给当前累计执行时间最少的处理机。

假设每个结点最多有 l 个前趋结点，结点分配算法详细描述如下（依照图2数据结构）：

```
procedure nodeassignment;
begin
    for i←1 to m do
    begin
      maxtime←o;
      for j←1 to l do
      begin
        g←前趋结点序号;
        cnum1←g 所分处理机号
        if  g 是 cnum1所分配的最后一条指令 then
        begin
            if cnum1累计执行时间>maxtime then
              cnum←cnum1;
              maxtime←cnum1累计执行时间;
        end;{if}
      end;{for  j}
      if cunm=0 then
      begin
      mintime←第一个处理机累计执行时间
      cnum←1
      for cnum2←2  to n do
        begin
            if cnum2累计执行时间<mintime then
              cnum←cnum2
              mintime←cnum2累计执行时间
            end{for cnum2}
        end {if}
        cnum 处理机指令计数←原指针+1;
        将 i 结点分配给 cnum 处理机;
        填写指令执行时间;
        回填结点 i 所分处理机号 cnum;
        回填结点 i 的分配位置;
```

```
            cnum 处理机累计执行时间←原执行时间+指令执行时间;
            barrier insertion;
          end;{for i}
    end;
```

此算法复杂性为 O(ml)+O(mn).

每当分配完一个结点后，都要判其前趋结点(procedure g)与该结点(consumer i)之间是否需插入同步等待，即 barrier 指令。假设在所有处理机都没有运行时，有一条公共 barrier 指令。若 g 和 i 分在同一处理机上，则无需 barrier 指令。若 g 与 i 不分在同一处理机上，则若 g 与 i 的最近公共 barrier 到 g 结点的最大执行时间大于此公共 barrier 到 i 结点的最小执行时间，需在 g 结点之后、i 结点之间插入一条 barrier 及通讯指令。

设 barrier 为一递增整数值，barrier 插入算法详细描述如下：

```
procedure barrier insertion;
begin
  barrier←0;
  各处理机←公共 barrier 0;
  记公共 barrier 位置;
  for  j←1  to  l  do
  begin
  g←前趋结点序号;
  if  g 所分处理机号≠i 所分处理机号 then
    begin
        cpug←g 所分处理机号;
        lastbarg←cpug 上 g 结点前的最后一条 barrier 指令;
        cpui←i 所分处理机号;
        lastbari←cpui 上 i 结点前的最后一条 barrier 指令;
        nextbarg←g 结点后的第一条 barrier;
        if cpui 上的 lastbari 之前有等同于 nextbarg 的 barrier 指令 then
        begin
            在 cpug 上 g 结点之后插一条通讯指令;
            在 cpui 上 i 结点之前插一条通讯指令;
            cpug 累计执行时间←原执行时间+通讯指令执行时间;
            cpui 累计执行时间←原执行时间+通讯指令执行时间;
            cpug 上指令分配计数指针←原指针+1;
            cpui 上指令分配计数指针←原指针+1;
        end{if}
  else  begin
        combarrier←由 barrier 指针查得 cpug 上 lastbarg 之前与 cpui 上 lastbari 之前的公共
barrier;
            timesumg←从 combarrier 到 g 结点的执行时间之和;
            timesumi←从 combarrier 到 i 结点前一条指令的执行时间之和;
            if timesumi≥timesumg then
```

```
                    begin
                      在 cpug 上 g 之后插一条通讯指令;
                      在 cpui 上 i 之前插一条通讯指令;
                      cpug 累计执行时间←原执行时间+通讯指令执行时间;
                      cpui 累计执行时间←原执行时间+通讯指令执行时间;
                      cpug 上指令分配计数指针←原指针+1;
                      cpui 上指令分配计数指针←原指针+1;
                    end;{if}
            else begin
                      在 cpug 上 g 之后插一条通讯指令;
                      barrier←barrier+1
                      插一条 barrier 指令;
                      填 cpug 上一条 barrier 位置;
                      cpug 上累计执行时间←原执行时间+通讯指令执行时间+barrier 执行时
间;
                      cpug 指令分配计数指针←原指针+2;
                      在 i 结点之前插 barrier;
                      插通讯指令;
                      填 cpui 上一条 barrier 位置;
                      cpui 累计执行时间←原执行时间+通讯指令执行时间+barrier 执行时间;
                      cpui 指令分配计数指针←原指针+2;
                    end;{else}.
               end;{else}
               end;{if}.
            end;{for j}.
    end;
```

此算法复杂性为 O(ml).

3 结束语

以上算法均在 VAX－11/780上模拟实现，并证明是行之有效的。目前所做的工作是在用户程序的基本块中进行的。我们准备将这个工作扩展到程序的循环一级，同时对 barrier 指令的插入算法做进一步的优化工作。

参 考 文 献

1 Mattew T. O'Keefe, Henry G, Dietz. Hardware Barrier Synchronization: Static Barrier MIMD(SBM). International Conference on Parallel Processing, 1990

2 Mattew T. O'Keefe, Henry G. Dietz. Hardware Barrier Synchronization: Dynamic Barrier MIMD(DBM). International Conference on Parallel Processing, 1990

3 Aderrazek Zaafrani, Henry G. Dietz, Mattew T. O'eefe. Static Scheduling for Barrier MIMD Architectures

A Study of Algorithms Supporting for Exploiting Operation Level Parallism Basad on SBM

Wen Diping Yang Xuejun Chen Lijie
(Department of Computer Science)

Abstract

SBM is a highly efficient synchronization mechanism supporting parallelism on operation level parallel. Based on SBM, we deeply studied the algorithms of node scheduling and barrier placement, and propose an efficient plan for exploiting parallism on operation level. We use a direct graph G(N,A) to describe the dependence among instructions, sort nodes with node's critical path as key words, describe an assignment algorithm that assign the nodes to each processor, Meanwhile, we describe a barrier placement algorithm that inserts the barrier instruction between two dependent instructions if necessary.

Key words multiprocessor system, synchronization, application program, instruction systems

---

世界环境的状况信号

全球环境的恶化，这是人类必须意识到的状况，应积极开展全球性的自救运动。

全球环境的巨大压力之一是人类自身数量的可怕膨胀。1991年全球人口已过54亿，预计2070年将突破100亿。人口数量的增加给自然资源和生态系统带来沉重的负担。

世界经济令人担扰。80年代世界经济增长极不稳定，区域间差别很大，总的情况低于70年代。80年代后期，全球粮食生产停滞不前。

大气污染使全世界城市人口的一半，生活在二氧化硫超标的大气环境中，使10亿城市人口生活在颗粒物超标的环境中。

世界范围内，大量生活污水和工业废物排入河流、水道。发展中国家31%的人口无法获取安全清洁的水。

严重的土壤侵蚀危害农业，缩短水利设施的寿命，造成运河和港湾的淤积，并损害生产业湿地。据估计每年从世界耕地上流失的表层土壤约250亿吨。

每年全球热带森林的采伐可能在2千万亩左右。这将比剧地球上生物物种的灭绝。

全球环境恶化是人类自身行为造成的，人类毫无选择地必须承担起责任。