国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第19卷第5期 1997年10月 Vol. 19 No. 5

# 基于多层循环并行化的负载平衡优化*

郭克榕 唐新春

(国防科技大学计算机系 长沙 410073)

**摘 要** 负载平衡是并行处理中的一个重要概念。参与一个程序执行的各处理机所承担的工作量是否均衡直接影响该程序的并行性能。本文对面向 MPP 系统程序循环级并行化中负载平衡的优化进行了探讨，提出了优化策略及其实施算法。

**关键词** 程序并行化，负载平衡，数据分布，循环迭代划分，程序并行性能

**分类号** TP301

# Optimization of Load Balance Based on Nested Loop Parallelization

Guo Kerong Tang Xinchun

(Department of Computer, NUDT, Changsha, 410073)

**Abstract** Load balance is an important concept in the parallel processing. Whether the work load carried by each processor participating in a program execution is balanced or not directly influences the parallel performance of the program. This paper discusses optimization of the load balance in the loop order parallelization and presentes an optimization strategy as well as its practical algorithm.

**Key words** program parallelization, load level, data distribution, loop iteration distribution, program parallel performance

负载平衡是并行处理中的一个重要概念。将并行机系统中一组处理机用于执行同一个应用程序，以求完成单个程序运行时间大大减少，这不仅需要充分挖掘应用程序中的并行性，同时还需要在参与程序执行的所有处理机间尽可能均衡划分程序中可并行的成份。由于负载平衡涉及到执行一个并行程序的所有处理机的完成时间，假定所有处理机同

* 国防预研基金资助项目
1996年10月30日收稿

时开始程序的执行，最后完成执行的处理机决定了程序并行执行的时间。负载愈平衡，程序的并行性能就愈好[1]。所以，负载平衡是程序并行化所追求的重要目标之一。

对于具有极强并行计算能力和超大容量分布主存的大规模并行处理(MPP)系统，要使系统的高性能有效发挥，增强程序并行执行时的数据局部性，以减少处理机间的通讯开销固然十分重要，但尽可能地挖掘程序中可并行成份，使丰富的处理机资源得以充分、均衡地利用也不容忽视[2]。本文对面向 MPP 系统程序循环级并行化中负载平衡的优化进行了探讨，提出了优化策略及其实施算法。

## 1 负载平衡优化策略

MPP 系统与传统共享主存多机系统[3]相比，并行计算潜能大大增强。处理机数目由几十到几百，甚至上千。正因为有丰富的处理机资源，系统对处理机的分配和调度常采取静态完成的方式：由用户程序申请所需要的处理机数，当系统可满足用户程序对处理机的需求时，才启动程序运行。程序中数据的分布和并行任务的划分也在编译时得以确定，参与执行该用户程序的所有处理机保留于整个程序的运行期间，直至运行结束。可见，并行调度动态开销对程序并行性能的影响大大减小。因此，并行化时对任务粒度的追求可相应减弱，以便获得好的负载平衡性，从而最终提高程序的并行性能。所以，面向 MPP 系统进行程序并行化时，对多层紧嵌套的可并行化循环，仅最外层并行化的传统方法已并非总是最佳选择。当最外层循环迭代次数少于程序所获得的处理机数目时，采取多层并行的方法，可望能提高程序的并行性能。

设有 n 层均可并行化的紧嵌套循环 LP 如图1所示(经循环规范化后可获得)。其中 $M_i$ ($i=1, 2, \cdots, n$) 为大于1的整数值，$M_i \geqslant M_j$ ($i<j$)(可通过循环交换获得)。循环体为含有对数组访问的一组语句，至少包含对数组的写访问。数组各维下标表达式为：$a_iI_i+b_i$，$I_i$ 为循环控制变量，最多出现在一维下标表达式中，$a_i$ 和 $b_i$ 取整数值。

```
DO  I1=1, M1
    DO  I2=1, M2
        ...
        DO In=1, Mn
        循环体
        ENDDO
        ...
    ENDDO
ENDDO
```

图1 n 层可并行化循环 LP

为支持循环的并行执行，循环中的写访问数组被定义为共享数组，分布在参与循环执行的 P 个处理机上（P 可静态确定）。各维分布方式由自动数据分布机制对数组访问的下标形式进行分析后，确定取下三种方法之一[4]：

·循环分布（CYCLIC)。当循环中共享数组的下标形式基本一致、可对准分布时，采取此分布方式负载平衡性最好。

·块循环分布（BLOCK（$m_i$））。其中 $m_i=\lfloor (a_i(M_i-1)+1)/P_i \rfloor$，$P_i$ 为分配给数组第 i 维的处理机数。$m_i$ 的取法是为了将该维下标子集（$a_i+b_i : a_iM_i+b_i$）在 $P_i$ 个处理机上均匀分布（当 $a_i(M_i-1)+1 \leqslant P_i$ 时，取循环分布方式）。若循环中共享数组下标的对应维有一定偏移（一般小于 $m_i$），采取此分布方式可获得较好的数据局部性。

·退化分布（*）。当循环中共享数组的某维对应的循环层不宜并行化时，采取此分布方式也是为求得较好的数据局部性。

面向分布式系统并行循环的迭代划分通常采取写访问数组“拥有者计算”原则[5]。即编译以某个写访问数组作为循环迭代空间划分的目标数组，按照该共享数组的分布方式，将循环迭代空间划分到参与循环执行的所有处理机上，使得循环执行时，每个处理机对该共享数组的访问总是局部的。

不难看出，下标表达式中 $a_i$ 的取值除了对程序固有的并行性有影响外（$a_i \neq 1$时，数组第 i 维的下标仅有1/$a_i$ 被访问），对迭代空间划分均匀性的影响已由上述共享数组逐维分布方式中 $m_i$ 的取值所考虑。$b_i$ 的取值仅对迭代空间划分时边界上部分子空间的大小有影响，其余子空间总是相等的。下面我们对循环 LP 最外层并行化与多层并行化的性能进行分析，并提出负载平衡的优化算法。为便于分析，不妨设 $a_i=1$，$b_i=0$，迭代划分目标数组为：A（$I_1$，$I_2$，…，$I_n$），并且采取最均匀的分布方式——循环分布。$M_i$ 为大于1的整常数，t 表示循环体执行时间，$T_1$表示最外层并行化时循环 LP 的并行执行时间，$T_k$ 表示 k 层并行化时循环 LP 的并行执行时间（$2 \leqslant k \leqslant n$）。分析针对两种情况进行：

（1）$P > M_1$

1）最外层并行化：数组 A 第一维在 P 个处理机上循环分布，其余维退化分布。可并行执行的循环迭代次数为 $M_1$，每次迭代执行时间为 $M_2 * M_3 * \cdots * M_n * t = L * t$，显然，P 个处理机中仅有 $M_1$个处理机各分到一次迭代，$P-M_1$个处理机不能参加循环 LP 的执行。循环 LP 的并行执行时间为 $T_1=L*t$。

2）n 层并行化：数组 A 的 n 维在 P 个处理机上循环分布。可并行执行的循环迭代次数为 $M_1 * M_2 * \cdots * M_n = M_1 * t$，每次迭代的执行时间为 t。若 $M_1 * L \leqslant P$，即可并行的迭代数仍不超过处理机数 P，可取分配给每维的处理机数 $P_i$ 为 $M_i$，则有 $P_1 * P_2 * \cdots * P_n$（$=M_1 * L \leqslant P$）个处理机对 n 层循环迭代空间进行了最均匀的划分，$T_n=t$。为 $T_1$的 L 分之一。若 $M_1 * L > P$，即可并行的迭代数已超过处理机数 P，显然应尽可能利用这 P 个处理机，充分开发循环 LP 的并行性。因此，分配给每维的处理机数 $P_i$ 应如此选择 $P_i \leqslant M_i$，并且 $P_1 * P_2 * \cdots * P_n \leqslant P$。具体算法如下：

①初始化：$i=1$，$PT_0=P$。

②从（i，i+1，…，n）中选取某个 j，使得 mod（$PT_{i-1}$，$M_j$）最小，判 j=i 吗?成立，转④。

③将第 j 层循环提到第 i 层循环之前（1到i−1层不变），产生新的一组迭代次数：$M_1$，$M_2$，…，$M_n$。

④取 $P_i=M_i$，$PT_i=\lfloor PT_{i-1}/P_i \rfloor$，判 $PT_i=1$吗?成立，记录 k=i，转⑦。

⑤判 $PT_i > M_{i+1}$吗?成立，置 i=i+1，转②。

⑥从（i+1，…，n）中选取某个j，使得$\lceil M_j/PT_i \rceil * M_{i+1} * M_{i+2} \cdots * M_{j-1} * M_{j+1} * \cdots * M_n$ 最小，将第j层提为第i+1层，取$P_{i+1}=PT_i$，记录$k=i+1$。

⑦$P_{k+1}$，…，$P_n$ 均置1，结束。

将最终形成的n层循环的迭代次数记为$N_1$，$N_2$，…，$N_n$，可估算循环的执行时间为：

$$T_k=\lceil N_k/P_k \rceil * N_{k+1} * \cdots * N_n * t$$

其中$1<P_k \leqslant N_k$；$N_k$，$N_{k+1}$，…，$N_n \in (M_1, M_2, \cdots, M_n)$。

若$k=1$，设算法第②步选出的并行层为j，即$c=\mathrm{mod}(P, M_j)$最小，由算法第①步可知$P_1=M_j$，$PT_1=1$，由$P=P_1 * PT_1+c$，即$P=M_j+c$可知$j=1$，即为最外层并行的情况，$T_k=T_1$。

若$k \geqslant 2$，可分两种情况分析：

原第一层现处在k层之前，则$M_1 \notin (N_k, N_{k+1}, \cdots, N_n)$，显然有$T_k<T_1=L * t$。

原第一层现处在k层或k层之后，由算法第④步和第⑥步可知

$$P_k=PT_{k-1}=\lfloor PT_{k-2}/N_{k-1} \rfloor \quad 即 \quad PT_{k-2}=P_k * N_{k-1}+c_1 \tag{1}$$

由算法第⑤步可知 $PT_{k-2}>M_1$ 即$PT_{k-2}=a * M_1+c$ (2)

其中 $a=\lfloor PT_{k-2}/M_1 \rfloor$，$c=\mathrm{mod}(PT_{k-2}, M_1)$

由（1）、（2）式可得 $P_k=(a * M_1+(c-c_1))/N_{k-1}$ (3)

由算法第⑥步可知$\lceil N_k/P_k \rceil * M_1 \leqslant \lceil M_1/P_k \rceil * N_k$

即$\lceil N_k/P_k \rceil \leqslant \lceil M_1/P_k \rceil * N_k/M_1$

将（3）式代入上式右部并化简得

$$\lceil N_k/P_k \rceil \leqslant \lceil N_{k-1}/(a+(c-c_1)/M_1) \rceil * N_k/M_1$$

由算法第②步可知 $c-c_1>0$，而$a \geqslant 1$，

所以 $\lceil N_k/P_k \rceil \leqslant N_{k-1} * N_k/M_1$

则 $T_k=\lceil N_k/P_k \rceil * N_{k+1} * \cdots * N_n * t$

$$\leqslant (N_{k-1} * N_k/M_1) * ((M_1 * \cdots * M_n)/(N_1 * \cdots * N_k)) \tag{4}$$

当$k=2$，由（4）式有 $T_k \leqslant L * t=T_1$

当$k>2$，由（4）式有

$$T_k \leqslant (L * t)/(N_1 * \cdots * N_{k-2}) < T_1$$

可见，当$P>M_1$：最外层并行化，处理机资源得不到充分利用，循环LP蕴含的并行性也得不到充分开发，而多层并行化，则通过并行性的进一步开发和负载平衡的改善，明显提高了循环LP的并行性能。

（2）$P \leqslant M_1$

1）最外层并行化：A数组第一维在P个处理机上循环分布，其余维退化分布。设

$$M_1=J_1 * P+k_1$$

其中 $J_1=\lfloor M_1/P \rfloor$，$k_1=\mathrm{mod}(M_1, P)$

当$k_1=0$，P个处理机各分得最外层循环的$J_1$个迭代，负载最平衡，循环LP的并行性能

达到最优。

$$T_1=J_1*L*t$$

当$k_1\neq 0$，$k_1$个处理机各完成$J_1+1$个迭代，$P-k_1$个处理机各完成$J_1$个迭代。

$$T_1=(J_1+1)*L*t$$

2）n 层并行化：数组 A 的 n 维在 P 个处理机上循环分布。设分配给每维的处理机数为 $P_i$，则 $P_i$ 所完成的迭代数最多为$\lceil M_1/P_1\rceil*\cdots*\lceil M_n/P_n\rceil$，有 $T_n=\lceil M_1/P_1\rceil*\cdots*\lceil M_n/P_n\rceil*t$. 显然，很难找到一组 $P_i$，使 $P_1*P_2*\cdots*P_n\leqslant P$ 时，总有下式成立。

$$T_n<T_1=\lceil M_1/P\rceil*M_2\cdots*M_n*t$$

也就是说，当$P\leqslant M_1$时，欲通过 n 层并行化进一步提高程序并行性能已较为困难（算法很复杂）。

综上分析，可得出负载平衡优化的基本策略如下：对 n 层均可并行化的紧嵌套循环 LP，当最外层迭代次数小于参与循环执行的处理机数目时，对 n 层实施并行化。否则，并行化仅对最外层实施，在工程上我们采取的方法是：若存在某个 $M_i$（$2\leqslant i\leqslant n$），使$\lfloor M_i/P\rfloor\geqslant 1$，mod（$M_i$，P）$=0$，则将第 i 层循环交换到最外层，否则从（1，2，…，n）中选取某个 i，使得$\lceil M_i/P\rceil*M_1*\cdots*M_{i-1}*M_{i+1}*\cdots*M_n$ 最小，将第 i 层循环提为最外层，然后对最外层并行化，可获得较好的并行性能。

## 2 实 例

用数据并行语言 Cray MPP Fortran[6]，可描述根据上述负载平衡优化策略所产生的并行循环为：

```
CDIR $ DOSHARED ($I_1$, $I_2$, …$I_k$) ON 目标数组
DO $I_1=1$, $M_1$
  DO $I_2=1$, $M_2$
    …
    DO $I_k=1$, $M_k$
      …
      DO $I_n=1$, $M_n$
        循环体
      ENDDO
      …
    ENDDO
    …
  ENDDO
ENDDO
```

图2 并行循环 LP

其中 CDIR $ 打头的指导命令说明将紧接其后的 n 层紧嵌套循环的最外 k 层并行化，对应的 k 层循环迭代空间按照 ON 子句中目标数组的分布方式进行划分。

例1…

```
DO   I=1, 17
  DO J=10
  A (I, J) =~
  ENDDO
ENDDO
```

例2…

```
DO   I=1, 33
  DO J=1, 31
  A (I, J) =~
  ENDDO
ENDDO
```

设 P=32，例1中 $P>M_1$，故多层并行，由算法可将 J 层交换至外层，以及 k=2，$P_1=10$，$P_2=3$。显然 $T_k=6t$ 优于 $T_1=10t$（t 为循环体执行时间），并行化后的循环见例3。例2中 $P\leqslant M_1$，故一层并行，由算法选得 J 层优于 I 层作为外层：$T_i=62t$，$T_j=33t$，$T_i$、$T_j$ 分别为 I 层、J 层作为外层时整个循环的执行时间。并行化后的循环见例4。

例3 …

```
CDIR $  SHARED A (:, CYCLIC,: CYCLIC)
CDIR $  DOSHARED (J, I) ON A (I, J)
      DO   J=10
        DO I=1, 17
        A (I, J) =~
        ENDDO
      ENDDO
  …
```

例4…

```
CDIR $  SHARED A (:,: CYCLIC)
CDIR $  DOSHARED (J) ON A (I, J)
        DO   J=1, 31
          DO I=1, 33
          A (I, J) =~
          ENDDO
        ENDDO
    …
```

当循环迭代次数 $M_i$ 静态无法确定时（多数情况下，可通过常数传播，表达式向前替换等常规优化手段确定），可根据对 $M_i$ 取值的测试，进行动态负载平衡优化。

## 3 结束语

本文提出的是面向 MPP 系统，进行程序循环级并行化时负载平衡优化的一个基本策略。参与循环执行的各处理机的负载平衡性与循环中迭代划分目标数组的各维分布方式以及处理机在目标数组各维的分配方式均密切相关。因此，在制定数据分布、处理机分配和并行划分方案时，欲求得好的并行性能，则必须兼顾数据的局部性和负载的平衡性。

## 参 考 文 献


1 Constantine D P. Parallel Programming and Compilers. Kluwer Academic Publishers. 1988

2 Hans P. z, Barbara M C. Compiling for Distributed-Memory Systems. proc. of IEEE, 1993, 81 (2)

3 Baneriee U etal. Automatic Program Parallelization. proc. of IEEE, 1993, 81 (2)

4 High Performance Fortran Forum. High Performance Fortran Language Specification. Technical Report CRPC-TR92225, Center for Research on Parallel Computation, Rice University, Houston, TX., January 1993

5 Hiranandani S, Kennedy K, Tseng C. Compiling Fortran D for MIMD Distributed-Memory Machines. ACM comm., August 1992, 35 (8): 66～80

6 Douglas M P, Mcdonald T, Meltaer A. MPP Fortran Programming Model. Cray Research, Inc.. Eagan, Minnesota, February 1993


（责任编辑 张 静）