多处理机系统循环间数据重用的cache优化*

丁永华 原庆能 臧斌宇 朱传琪

(复旦大学并行处理研究所 上海 200433)

摘要 cache的使用缓解了CPU和主存储器之间速度差距太大的矛盾,同时,也使cache的命中率成为影响多处理机系统性能发挥的重要因素.人们对如何加强数据的局部性,提高cache命中率,使多处理机系统的性能得到更好的发挥进行了积极的探索.但过去的工作主要集中于如何加强并行循环内的数据局部性,减少甚至消除并行循环内真假共享cache行所引起的cache抖动,对多处理机系统中循环间数据重用的开发和利用却少有论述.该文对如何开发和利用这些循环间数据重用进行了分析和讨论,并提出了一些切实可行、易于实现的方法.这些方法的应用能够有效地提高cache的命中率,从而提高多处理机系统的性能.

关键词 数据局部性,cache抖动,cache命中率,循环间数据重用,循环变换,同步.

中图法分类号 TP311

在单处理机系统中,通过将频繁使用的数据和代码保存在cache中,可以降低主存的平均访问延迟,增加有效带宽,因此,使用cache后,单处理机系统的性能获得了可喜的提高.然而,在共享主存的多处理机系统中,每个处理机拥有自己的局部cache,程序并行执行时,处理机的调度策略有时难以做到把对同一存储单元内容的访问操作集中在同一处理机执行,当同一存储单元内容的访问操作被分配给不同的处理机执行时,cache的命中率就会降低.[1]近来的研究表明:多机系统往往比单机系统的cache命中率低.[2]另一方面,在多处理机系统中,cache命中率低可能使程序并行执行的开销大大增加.这是因为:当两个或两个以上处理机的局部cache都保存了同一存储块的一个副本,而某一处理机执行的操作写引用了该存储块中某个存储单元的内容时,保证数据一致性的操作会带来额外的开销.[3]因此,提高cache命中率是改善多处理机系统性能的有效途径.以往工作集中于单处理机系统的cache优化和循环内数据局部性的加强,以提高cache命中率.如果一个程序的主要循环有多个,那么,对多处理机系统来说,除了要加强循环内的数据局部性以外,还要考虑这些主要循环之间存在的数据重用,这部分数据重用的开发和利用将有助于提高cache的命中率,从而提高多处理机系统的性能.本文主要讨论缺省调度下,并行循环与并行循环之间、并行循环与串行循环之间、不同过程的循环之间数据重用的开发和利用,以提高cache的命中率.

当程序并行执行时,缺省调度将可并行执行的循环按循环变量连续等分到相当于处理机个数的几个进程中,其中一个为主进程,其余的为副进程,每个进程交给一个处理机完成,各进程访问到的数据保存在执行该进程的处理机的局部cache中.而对于串行执行的循环,缺省调度将它划分到主进程中,串行循环访问到的数据保存在执行主进程的处理机的局部cache中.例如,一个将要运行在由4个处理机组成的并行机上的循环200次的DO循环,若它是可以并行执行的,缺省调度将1~50次循环划分到主进程,由一个处理机完成,将51~100,101~150,151~200次循环分别划分到3个副进程,每个副进程由一个处理机完成.若该DO循环不能并行执行,则缺省调度将200次循环都划分到主进程.

在缺省调度下,并行循环与并行循环之间的数据重用有时可以利用得很好,有时却得不到利用,我们通过循环变换来改善这一状况.并行循环与串行循环之间的数据重用,通常只有并行循环划分到主进程的部分与串行循环之间的数据重用能得到利用,并行循环划分到副进程的部分与串行循环之间的数据重用却无法利用,因此,我们提出在串行循环中引入同步机制,以开发和利用并行循环划分到副进程的部分与串行循环之间的数据重用.我们的并行化编译系

* 本文研究得到国家自然科学基金、国家863高科技项目基金、国家攀登计划基金和上海市重点学科基金资助.作者丁永华,1971年生,助教,主要研究领域为并行处理与并行化编译.原庆能,女,1966年生,硕士,主要研究领域为并行化编译.臧斌宇,1965年生,副教授,主要研究领域为并行处理与并行化编译.朱传琪,1943年生,教授,博士导师,主要研究领域为并行处理.

本文通讯联系人:丁永华,上海200433,复旦大学并行处理研究所

本文1997-05-12收到原稿,1997-07-18收到修改稿

统AFT实现了上述两种数据重用的自动变换,实验结果显示,我们的方法是有效的.不同过程之间也存在循环间数据重用问题,开发和利用这一部分数据重用将能进一步提高cache的命中率.

本文第1节回顾相关工作,第2~4节分析和讨论并行循环与并行循环之间的数据重用、并行循环与串行循环之间的数据重用和不同过程的循环间数据重用,第5节是总结.

1　相关工作

当并行执行某一循环时,不同处理机对同一存储单元内容或相邻存储单元内容的引用操作有可能引起真共享或假共享cache行的抖动.一旦cache抖动现象发生,程序并行执行与串行执行的加速比不仅达不到理想值,严重时,程序并行执行的时间甚至会超过程序串行执行的时间.由于cache抖动严重影响了并行处理系统性能的发挥,多年来,并行处理领域的研究人员一直在努力寻求加强数据局部性,减少以至消除cache抖动的各种技术,如循环的交换、块化、错位、加边等方法,并取得了一定的成效.[4,5]但这些工作局限于加强循环内的数据局部性.实际上,不仅并行循环内数据重用的开发和利用能够有效地提高并行处理系统的性能,同一过程和不同过程的循环之间数据重用的开发和利用也将有助于提高cache的命中率,进一步提高并行处理系统的性能.

Cache的容量较主存的容量小得多,如果cache中的数据在被重用之前就已经被置换出cache,那么,cache就得不到很好的利用.文献[6]提出了跨循环重用信息的收集算法,通过循环合并(Loop Fusion)、循环分块(Loop Strip-Mining)、循环颠倒(Loop Reversal)等技术加强数据的局部性,使cache中的数据在被置换出cache前尽可能得到重用,从而提高单处理机系统的性能.在共享主存的多处理机系统中,各处理机有自己的局部cache,因此,要提高cache的命中率,不仅要考虑各处理机局部cache中的数据在被置换出cache前尽可能得到重用,还要考虑在处理机的调度策略下如何使得对同一存储单元内容或相邻存储单元内容的访问操作尽量集中在同一处理机执行.

2　并行循环与并行循环之间的数据重用

在缺省调度下,两个并行循环之间的数据重用有时可以得到很好的利用,如图1中的例1和例2.在例1中,两个并行执行的循环需要引用的A数组区间完全重合,且按相同方向的次序访问,它们之间对A数组区间$A(1:N)$的重用得到了完全的利用.在例2中,如果cache的行长不小于2,则两个并行循环之间存在对A数组区间$A(1:2*N)$中下标为偶数的数组元素的重用,这部分重用在缺省调度下也能得到完全的利用.

```
DOALL  I=1,N                DOALL  I=1,N
  A(I)=...                    ...=A(2*I-1)...
ENDDO                       ENDDO
DOALL  J=1,N                DOALL  J=1,N
  ...=A(J)...                 A(2*J)=...
ENDDO                       ENDDO
```

例1　　例2

图1

```
DOALL  I=1,N                DOALL  I=1,N
  ...=A(I)...                 A(I)=...
ENDDO                       ENDDO
...                         ...
DOALL  J=1,N                DOALL  J=1,N
  A(J+N/4)=...                ...=A(N-J+1)=...
  ...                         ...
ENDDO                       ENDDO
```

例3　　例4

图2

但有时存在于两个并行循环之间的数据重用却得不到很好的利用,如图2中的例3和例4.假设多处理机系统由cpu0,cpu1,cpu2,cpu3这4个处理机组成,则缺省调度下,例3和例4的两个并行循环在各相应进程中引用的A数组元素分别如图3和图4所示.

在例3中,第1个并行循环需访问的A数组区间$A(1:N)$的后3/4部分与第2个并行循环需访问的A数组区间$A((N/4)+1:N+N/4)$的前3/4部分重叠,两个并行循环对A数组区间$A((N/4)+1:N)$的重用在缺省调度下完全得不到利用.但是,如果让第1个循环在循环变量从1变化到$3*N/4$时访问它被第2个循环重用的A数组区间$A((N/4)+1:N)$,在循环变量从$(3*N/4)+1$变化到N时访问不被第2个循环重用的A数组区间$A(1:N/4)$,或者让第2个循环在循环变量从1变化到$N/4$时访问A数组区间$A(N+1:N+N/4)$,在循环变量从$(N/4)+1$变化到N时访问A数组区间$A((N/4)+1:N)$,则两个并行循环对A数组区间$A((N/4)+1:N)$的重用就能得到完全的利用.在例4中,两个并行循环按相反方向的次序访问A数组区间$A(1:N)$,它们对该数组区间的重用在缺省调度下也完全得不到利用.如果让其中一个循环改变访问A数组区间的方向,使它们按相同方向的次序访问A数组区间,则它们对A数组区间$A(1:N)$的重用就能得到完全的利用.

cpu0	cpu1	cpu2	cpu3
I⟶ A(1) ... A(N/4) J⟶ A((N/4)+1)...A(N/2)	A((N/4)+1)... A(N/2) A((N/2)+1)... A(3*N/4)	A((N/2)+1)... A(3*N/4) A((3*N/4)+1)... A(N)	A((3*N/4)+1)... A(N) A(N+1)... A(N+N/4)

图3

cpu0	cpu1	cpu2	cpu3
I⟶ A(1) ... A(N/4) J⟶ A(N)...A((3*N/4)+1)	A((N/4)+1)... A(N/2) A(3*N/4)... A((N/2)+1)	A((N/2)+1)... A(3*N/4) A(N/2) ... A((N/4)+1)	A((3*N/4)+1)... A(N) A(N/4) ... A(1)

图4

对于两个存在循环间数据重用的并行循环，如果它们重用的数组是一维数组，且其元素的下标是循环变量的线性函数，我们可以通过循环变换使它们对该数组的重用全部得到利用.

```
DOALL   I=1 ,N
  ... A( a1*I+b1)...
ENDDO
...
DOALL   J=1 ,M
  ... A( a2*J+b2)...
ENDDO
```

图5

对任一个循环语句，都可以将其一般化为循环的下界和步长均为1. 我们用图5所示的一般模式来说明我们的循环变换方法. 开发图5所示的两个并行循环之间对 A 数组区间重用的循环变换分以下3步进行，因为循环是可并行执行的，所以，下面的各种变换不会影响程序的语义.

(1) 如果 $a_1*a_2<0$，两个并行循环按相反方向的次序访问 A 数组，这时，我们对其中一个循环实施循环颠倒(Loop Reversal)变换，使它们按相同方向的次序访问 A 数组. 若 $a_1<0$，我们选择对第1个循环实施 loop reversal 变换，即在第1个循环的循环体中，用 $N-I+1$ 替换循环变量 I，设替换后 A 数组元素的下标为 $a'_1*I+b'_1$，则

$$\begin{cases} b'_1=b_1+a_1*(N+1) \\ a'_1=-1*a_1 \end{cases}$$

若 $a_2<0$，我们选择对第2个循环实施 loop reversal 变换，即在第2个循环的循环体中，用 $M-J+1$ 替换循环变量 J. 设替换后 A 数组元素的下标为 $a'_2*J+b'_2$，则

$$\begin{cases} b'_2=b_2+a_2*(M+1) \\ a'_2=-1*a_2 \end{cases}$$

(2) 当 $a_1*a_2>0$ 时，两个并行循环按相同方向的次序访问 A 数组. 设第1个循环在循环变量 $I=K_1$ 时开始访问的 A 数组区间被第2个循环重用，设第2个循环在循环变量 $J=K_2$ 时开始访问的 A 数组区间重用第1个循环引用的 A 数组区间. 两个并行循环对 A 数组区间的重用可归纳为如图6所示的3种情况.

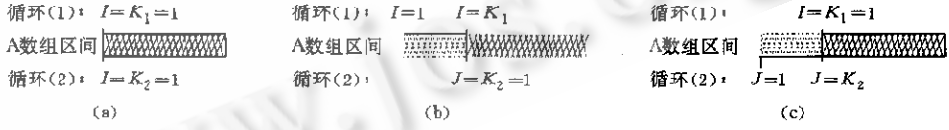

图6

由 $a_1+b_1=a_2J'+b_2$ 可得：$J'=[(a_1+b_1-b_2)/a_2]$，由 $a_2+b_2=a_1I'+b_1$ 可得：$I'=[(a_2+b_2-b_1)/a_1]$

则　　$K_1=\max(I',1)$　　$K=\max(J',1)$

如果 $K_1>N$ 或 $K_2>M$，两个并行循环之间不存在对 A 数组的循环间数据重用.

当 $K_1<N$ 且 $K_2<M$ 时，第1个循环在循环变量 I 从 K 开始访问的 A 数组区间与第2个循环在循环变量 J 从 K 开始访问的 A 数组区间存在数据重用.

定义函数 $sdmod(x,y)$ 为　　$sdmod(x,y)=\begin{cases} x & x\leqslant y \\ x-y & x>y \end{cases}$

若 $K_1=1$ 且 $K_2=1$，两个并行循环都先访问重用的 A 数组区间，不再需要做这一步循环变换.

若 $K_1>1$，则 $K_2=1$，第2个循环在循环变量 J 从1变化到 M 时，先访问重用的 A 数组区间. 我们选择对第1个循环进行循环变换，即在第1个循环的循环体中用 $sdmod(I+K_1-1,N)$ 替换循环变量 I，使第1个循环在循环变量 I 从1变化到 N 时，也先访问重用的 A 数组区间，后访问不被重用的 A 数组区间.

若 $K_2>1$，则 $K_2=1$，第1个循环在循环变量 I 从1变化到 N 时，先访问重用的 A 数组区间. 我们选择对第2个循

环进行循环变换，即在第 2 个循环的循环体中用 $sdmod(J+K_1-1,M)$ 替换循环变量 J，使第 2 个循环在循环变量 J 从 1 变化到 M 时，也先访问重用的 A 数组区间，后访问不被重用的 A 数组区间.

如果 $|a_1|*N=|a_2|*M$，两个并行循环访问同样长度的 A 数组区间，经过上述的循环变换后，存在于两个并行循环之间的对 A 数组区间的重用可以得到完全的利用.

(3) 如果 $|a_1|*N\neq|a_2|*M$，两个并行循环访问不同长度的 A 数组区间，经过上述的循环变换后，存在于两个并行循环之间的对 A 数组区间的重用在缺省调度下只有一部分能得到利用. 要使它们对 A 数组区间的重用全部得到利用，必须把访问较长 A 数组区间的那个循环分割成两个循环，变换后的结果如图 7 所示.

```
DOALL I=int(a2*M/a1)+1,N          DOALL I=1,N
...                               ...
ENDDO                             ENDDO
DOALL I=1,int(a2*M/a1)            ...
...                               DOALL J=1,int(a1*N/a2)
ENDDO                             ...
...                               ENDDO
DOALL J=1,M                       DOALL J=int(a1*N/a2)+1,M
...                               ...
ENDDO                             ENDDO
(a) |a1|*N>|a2|*M                 (b) |a1|*N<|a2|*M
```

图 7

调度循环并行执行会带来一定的开销，是否将访问较长 A 数组区间的循环进行分割，需权衡分割后增加的开销和可获得的 cache 命中率的提高.

3　并行循环与串行循环之间的数据重用

在程序并行执行时，有些循环可以并行执行，有些循环由于存在跨循环的数据相关，只能串行执行，并行循环与串行循环之间的数据重用在由 4 个处理机组成的多处理机系统中一般只有 1/4 能得到利用，如图 8 所示的程序段.

```
DO 100 L=L3,L4
QMLT(L)=QMULT
100   IF (X(L).LT.0.) QMLT(L)=0.
      DO 130 L=L3,L4
         ......
         IJ(L)=I(L)+NX2*(J-1)
         W1(L)=FXC*FYC
         W2(L)=FX(L)*FYC
         W3(L)=FXC*FY(L)
         W4(L)=FX(L)*FY(L)
130   CONTINUE
      DO 140 L=L3,L4
         Q(IJ(L))=Q(IJ(L))+W1(L)*QMLT(L)
         Q(IJ(L)+1)=Q(IJ(L)+1)+W2(L)*QMLT(L)
         Q(IJ(L)+NX2)=Q(IJ(L)+NX2)+W3(L)*QMLT(L)
         Q(IJ(L)+NX2+1)=Q(IJ(L)+NX2+1)+W4(L)*QMLT(L)
140   CONTINUE
```

图 8

这是测试程序包 spec95 中 wave5 的一段程序. 在这段程序中，循环 100 和循环 130 经过相关性测试后，都能并行执行，而循环 140 由于存在跨循环的数据相关，只能串行执行(该循环由于存在稀疏矩阵，因此归约也无效). 我们对循环 140 的执行时间进行测试后发现，循环 140 在程序并行执行时的执行时间要比在程序串行执行时的执行时间长. 经过分析，我们认为造成这种现象的原因是：我们所使用的多处理机系统由 4 个处理机组成，程序串行执行时，整个程序划分为一个进程，循环 100、循环 130、循环 140 都由同一处理机执行，循环 100、循环 130 对数组 $QMLT$、IJ、$W1$、$W2$、$W3$、$W4$ 的定义都保留在该处理机的局部 cache 中，循环 140 对这几个数组的引用都能 cache 命中；而程序并行执行时，循环 100 和循环 130 在缺省调度下按循环变量 L 连续被划分为长度相等的 4 段，分别交给 4 个进程，每个进程由一个处理机来完成，这 4 段是

$$L3+((L4-L3+1)/4)*i\sim L3-((L4-L3+1)/4)*(i+1)-1 \quad (i=0,1,2,3).$$

设$i=0$这一段划分到主进程,由处理机cpu0完成,其余3段划分到副进程,分别由处理机cpu1,cpu2,cpu3完成,则循环100和循环130执行完后,数组$QMLT$,IJ,$W1$,$W2$,$W3$,$W4$驻留在处理机cpu$i$局部cache中的数组区间为

$$L3+((L4-L3+1)/4)*i:L3+((L4-L3+1)/4)*(i+1)-1 \quad (i=0,1,2,3)$$

当执行循环140时,由于该循环只能串行执行,整个循环被划分到主进程,由处理机cpu0完成,因此,循环140对数组$QMLT$,IJ,$W1$,$W2$,$W3$,$W4$的引用只有下标从$L3$到$L3+(L4-L3+1)/4-1$的元素能cache命中,它们驻留在处理机cpu0的局部cache中,占需要引用的数组元素的1/4,而对其余3/4的数组元素的引用都cache失效,它们分别驻留在其它3个处理机的局部cache中.cache失效增加了循环140的执行时间,因而循环140在程序并行执行时的执行时间要比在程序串行执行时的执行时间长.

从上面的分析我们知道,要使程序并行执行时与并行循环存在数据重用的串行循环的执行时间缩短,必须提高串行循环引用重用数据的cache命中率.而并行循环定义的数组元素分段保存在各个处理机的局部cache中,为此,我们提出开发和利用并行循环与串行循环之间数据重用的优化方法,即根据循环间数据重用信息以及并行循环在缺省调度下的分段情况将串行循环划分为相当于处理机个数的几段,由于串行循环存在跨循环的数据相关,各段间是不能并行执行的,因此,需要在串行循环各段间引入同步变量,使串行循环各段在同步机制的控制下并行执行.将我们的优化方法用于图8的程序段,需把串行循环140按循环变量L连续划分为如图9所示的4段.

```
DO I=0,3
   DO L=L3+((L4-L3+1)/4)*I,L3+((L4-L3+1)/4)*(I+1)-1
   ......
   ENDDO
ENDDO
```

图9

然后在每段之间引入一个同步变量,用以控制一个段,只有在它的前一段执行完后才开始执行,本例引入$A(0)$,$A(1)$,$A(2)$,$A(3)$,$A(4)$共5个同步变量.最后将图9中外层循环标识为可并行执行.我们用图10来简单地表示图9的变形结果.

```
POST A(0)
C$DOACROSS I=0,3
   WAIT A(I)
   DO L=L3+((L4-L3+1)/4)*I,L3-((L4-L3+1)/4)*(I+1)-1
   ......
   ENDDO
   POST A(I+1)
ENDDO
```

图10

图10所示的程序段在缺省调度下将按外层循环的循环变量I把它划分到4个进程中去,分别交给4个处理机执行,因此,内层循环对数组$QMLT$,IJ,$W1$,$W2$,$W3$,$W4$的元素的引用都能在执行它的处理机的局部cache内找到,提高了cache的命中率.我们在SGI的ORIGIN2000(4个196MHz的R10000 CPU,1 024M内存,2M cache)上对SPEC95中的wave5作了测试.在我们的系统作cache优化前,其并行执行时间为154s,而cache优化后并行执行时间为110s,加速比分别为1.3和1.8,因此,我们提出的方法是实际有效的.

为了更好地开发和利用并行循环与串行循环之间的数据重用,有时需要把本节提出的优化方法和上节提出的循环变换方法结合起来考虑.例如,当并行循环访问到的数组区间与串行循环访问到的数组区间不是重叠得很好,或并行循环与串行循环按相反方向的次序访问重用的数组区间时,仅仅使用本节提出的优化方法是不够的,必须把上节提出的循环变换方法结合起来考虑,才能使它们之间的数据重用得到充分利用.

4 不同过程的循环间数据重用

在FORTRAN程序中,由于不同过程的变量通过虚实结合和公用区方式共享存储单元,使得不同过程的并行循环与并行循环之间、并行循环与串行循环之间也存在数据重用.开发和利用不同过程的循环间数据重用将能进一步提高cache的命中率.

不同过程的循环间数据重用包括被调用过程之间的循环间数据重用和调用过程与被调用过程之间的循环间数据重用,与过程内的数据重用相比,还要考虑跨过程信息的传递,用以计算循环间的重用信息.另外,由于同一过程在不同的调用点具有不同的上下文,因此,同一过程在不同的调用点与其他过程的循环间数据重用情况也不相同.为开发和利用一个过程在某一调用点与其他过程的循环间数据重用而对该过程作循环变换后,可能有利于开发和利用该过程在此调用点与其他过程的循环间数据重用,但却妨碍了该过程在别的调用点与其他过程的循环间数据重用的利用,因此,有时需根据需要生成该过程作循环变换的一个副本,在此调用点调用该过程副本,而在其他的调用点调用原来的过程.[7]

不同过程的循环间数据重用信息的收集要比同一过程内的循环间数据重用信息的收集难度更大,且当两个过程中共享一段连续存储单元的不是同维同大小的数组时,为开发和利用两个过程中访问这段存储单元内容的循环间数据重用,需要作较为复杂的循环变换,所采用的循环变换对提高整个并行处理系统性能的有效性也较难预测.

5 结束语

本文分析和讨论了多处理机系统中并行循环与并行循环之间、并行循环与串行循环之间、不同过程的循环之间数据重用的状况,并提出开发和利用两个并行循环之间数据重用的循环变换方法,提出在串行循环中引入同步机制以开发和利用并行循环与串行循环之间普遍得不到利用的循环间数据重用.这两种方法简单易行,能够有效地提高cache的命中率,从而提高整个并行处理系统的性能.不同过程的循环间数据重用虽然比同一过程内的循环间数据重用更难以开发和利用,但仍不失为提高cache命中率的一条重要途径,它也将是我们今后工作需要进一步解决的问题.

参考文献

1 Polychronopoulos C, Kuck D, Padua D. Execution of parallel loops on parallel processor systems. In: Hawang Kai ed. Proceedings of the International Conference'96 on Parallel Processing. Pennsylvania: Computer Society Press, Aug. 1986. 519~527

2 Eggers S. Simulation analysis of data sharing in shared memory multiprocessors [Ph. D. Thesis]. University of California, Apr. 1989

3 Mounes-Toussi Farnaz, Lilja David J, Li Zhi-yuan. An evaluation of a compiler optimization for improving the performance of a coherence directory. In: Jalby William ed. Proceedings of International Conference on Supercomputing. Manchester England: ACM Press, Jul. 1994. 75~83

4 Kennedy Ken, McKinley Kathryn S. Optimizing for parallelism and data locality. In: Kennedy Ken ed. Proceedings of International Conference on Supercomputing. Washington, D.C.: ACM Press, Jul. 1992. 323~334

5 朱传琪,臧斌宇,陈彤.程序自动并行化系统.软件学报,1996,7(3):180~186
(Zhu Chuan-qi, Zang Bin-yu, Chen Tong. An automatic parallelizer. Journal of Software, 1996,7(3):180~186)

6 Cooper Keith, Kennedy Ken, McIntosh Nathaniel. Cross-loop reuse analysis and its application to cache optimizations. In: Banerjee Utal ed. Proceedings of the 9th Workshop on Languages and Compilers for Parallel Computers. San Jose, California: Intel Corporation, Aug. 1996. 1~15

7 丁永华,陈彤,臧斌宇,朱传琪.过程繁衍及其实现方法.软件学报,1996,7(11):662~668
(Ding Yong-hua, Chen Tong, Zang Bin-yu *et al*. Cloning and its implementation. Journal of Software, 1996,7(11):662~668)

Cross-loop Reuse Techniques for Cache Optimizations on Multiprocessors

DING Yong-hua YUAN Qing-neng ZANG Bin-yu ZHU Chuan-qi

(*Parallel Processing Institute Fudan University Shanghai* 200433)

Abstract The use of the cache reduces the gap between the CPU speed and the memory latency, so the cache hit ratio becomes an important factor which affects the performance of multiprocessor system. Researchers have developed a number of optimizations to enhance data locality, increase the cache hit ratio and bring the multiprocessor system performance into better play. These techniques focus on how to enhance data locality within a parallel loop, reduce and even eliminate the cache line thrashing due to true or false sharing of the cache line. Exploitation and utilization of cross-loop reuse on multiprocessors are seldom discussed. How to exploit and utilize these cross-loop reuse, and put forward some feasible and easy ways for implementation are discussed in this paper. Application of these methods can effectively increase the cache hit ratio, thus improve the performance of multiprocessor system.

Key words Data locality, cache thrashing, cache hit ratic, cross-loop reuse, loop transformation, synchronism.