矩阵相乘算法优化的研究

钱晓捷，杨镇江，杜志刚，李秀芳
（郑州大学 信息工程学院，河南 郑州 450001）

摘要：本文对经典矩阵相乘 A*B 算法提出多种优化方法：根据局部性原理，提出对矩阵 B 进行转置；根据计算机缓存的大小与矩阵 A 与矩阵 B 的规模进行嵌套循环分块，通过对分块大小的调整比较获得最优的分块;利用循环展开技术以提高程序的并行性。实验结果表明，优化后的算法缩短了运行时间，获得了较优的运行效率。

关键词：矩阵相乘算法；矩阵转置；循环分块；循环展开
中图分类号：TP393.09　文献标识码：A

Research of Matrix Multiplication Algorithm Optimization

(QIAN Xiaojie, YANG Zhenjiang, DU Zhigang, LI Xiufang)
(School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract: Many kinds of optimization methods to the classical matrix multiplication A*B algorithm are proposed in this paper: transpose the matrix B according to the principle of locality; circulate and block the program according to the size of computer cache and the scale of matrix A and matrix B, compare and adjust the size of block to get the better blocking; enhance the program parallelism using loop unrolling. The result of the experiment shows that the optimized algorithm can not only reduce the time of operation, but also gain better operation efficiency.

Key words: matrix multiplication algorithm; matrix transposition; loop blocking; loop unrolling

1. 引言

矩阵乘法是数值计算中最重要的操作之一[1]，它在信号处理、石油勘探、理论物理、固态物理、编码理论、密码学、数字图像处理、线性预测、自回归滤波器设计和计算机时序分析等领域有着广泛的应用，它的性能对数值计算中大部分操作的性能都有影响，尤其是当矩阵阶数较高时，通常的计算过程需要占用较多的工作单元和较大的计算机内存，计算效率受到影响。从某种意义上说，它是数值计算操作的一个共同基础，因此，提高矩阵乘法的执行效率，无论在理论上还是在实际应用上对整个数值计算领域有着重大而深远的意义。

2. 矩阵相乘的经典算法

现假设矩阵 A，B，C=A*B，其中 A 是 m*l 矩阵，B 是 l*n 矩阵，而 C 是 m*n 矩阵，矩阵 A、B、C 的数据元素分别用 A[i][j]，B[i][j]，C[i][j]表示，其中 i，j 的值大于等于 0，小于矩阵的行列值。

矩阵相乘的经典算法如下：

i-j-k 形式

```
for(i=0,i<m, i++)
  for(j=0,j<n, j++)
    for(k=0,k<l, k++)
      {C[i][j]=0;
       C[i][j]= C[i][j]+ A[i][k]* B[k][j];
      }
```

j-k-i 形式[2]

```
 for(j=0, j<n, j++)
    for(k=0,k<l, k++)
       for(i=0,i<m, i++)
        {C[i][j]=0;
         C[i][j]= C[i][j]+ A[i][k]* B[k][j];
        }
```

3. 矩阵相乘优化策略

3.1 矩阵转置

矩阵相乘是矩阵 A 的一行数据元素与矩阵 B 的一列的数据元素对应相乘，然后再对其积求和，由于 C 与 C＋＋的二维数组是以行为主序存储的，因此矩阵 A 的行数据元素是连

续存储的，而矩阵 B 的列数据元素是不连续存储的（N*1 的矩阵除外），为了在矩阵相乘时对矩阵 B 也连续读取数据，根据局部性原理对矩阵 B 进行转置，矩阵转置是数据元素 B[i][j]与 B[j][i]进行交换，矩阵转置后，矩阵相乘是矩阵 A 的一行数据元素与转置后的矩阵 B 的一行的数据元素对应相乘，然后再对其积求和，这个和就是矩阵 C 的一个数据元素。

```
void con_arr( double B[NUM][NUM]) //本论文讨论的矩阵是方阵，并且是用二维数组进
{                                 //行存储的，采用一个中间变量实现对矩阵 B 的转
  int i,j;                        //置，可以节省存储空间，NUM 为二维数组的阶数。
  double temp;
  for(i=0;i<NUM;i++)
    for(j=0;j<i;j++)
      {  temp=B[i][j];
         B[i][j]=B[j][i];
         B[j][i]=temp;
      }
 }
```

经过对矩阵 B 进行转置后，经典矩阵相乘算法变为：

i-j-k 形式

```
for(i=0,i<m, i++)
  for(j=0,j<n, j++)
    for(k=0,k<l, k++)
      {C[i][j]=0;
       C[i][j]= C[i][j]+ A[i][k]* B[j][k];
      }
```

j-k-i 形式

```
for(j=0, j<n, j++)
  for(k=0,k<l, k++)
    for(i=0,i<m, i++)
      {C[i][j]=0;
       C[i][j]= C[i][j]+ A[i][k]* B[j][k];
      }
```

由于 C++编译器对多维数组以行优先的形式存放，因此算法 i-j-k 形式的空间局部性比 j-k-i 形式略好，后续论文只讨论 i-j-k 形式的经典矩阵相乘算法。

3.2 嵌套循环分块

循环分块是一种广泛使用的程序性能优化技术，它将循环分割成小的块，使一个块的工作集能够容纳在 cache 中，并将对相同数据的多次重用转移到时间上靠得更近地方，使这些重用能够在 cache 中实现，从而改善了数据访问的空间局部性。为了简便起见，本文只讨论数据稠密、分块形状为正方的分块。

分块变换结合了条状化变换，条状化是将单个循环转换为一个两层的嵌套循环。我们把条状化后的外层循环称为控制循环，内层循环称为基本循环，分块的大小我们用 BLOCK_SIZE 表示。

对多层循环的分块，在对各层循环进行条状化处理之后，还要把各层的分块控制移到嵌套的外层，各层的基本循环移到嵌套的内层，这种变换按照循环交换的规则进行。

嵌套循环分块是单个能够获得更多好处的变换技术[3]：（1）当实施寄存器级别时，它能够改善程序的 ILP（指令级并行）；（2）通过多维分块，它能够利用多维迭代空间上的数据重用；（3）通过多级分块，它同时改善多个存储层次的数据局部性。

研究表明，程序使用不同分块大小时的性能差别非常大，最优的分块大小不仅与机器结构相关，也与程序特征相关。所以，分块大小选择是分块变换的核心问题。循环分块分为 cache 级分块与寄存器级分块。Cache 级分块值太大或太小均会增加一级 cache 失效次数并导致 cache 利用率的降低。寄存器级的分块太小则不能充分利用寄存器，太大将会导致寄存器溢出。最内层循环展开可以降低循环控制开销，但是展开次数太多可能导致指令 cache 溢出，

反而降低程序性能。

根据陆平静等研究成果[4]，分块的大小 BLOCK_SIZE 满足的不等式方程

$(BLOCK\_SIZE/B)^2+3BLOCK\_SIZE+1 \leqslant C/B$ (1)

其中 B 是 cache 每行存储的元素个数，C 为 cache 的容量，在这个范围内寻找使运行效率最高的 BLOCK_SIZE 值。

3.3 循环展开

循环展开是开发循环并行性的一种优化技术。它将原循环体复制多份，并修改相关数据索引变量。循环体展开增大了循环体大小，有利于发现指令间存在的并行性，并减少循环转移开销。在有些情况下，循环展开还减少了访存次数。这些都对提高程序性能有利。

循环展开分为两种情况，一种是内层循环展开，一种是外层循环展开。循环展开前的循环次数必须能整除循环展开的次数。

对任意循环的循环展开变换总是合法的，并且其实施也比较简单。但是，循环展开也会对程序性能带来一些副作用，如[5]：（1）增加寄存器压力：当循环展开后循环体的工作集超过可用的寄存器文件大小时，导致寄存器溢出，影响性能；（2）代码膨胀：由于多次复制循环体，程序代码增长，可能导致指令 cache 不命中数增加；（3）边界处理：当循环展开因子不能整除循环迭代次数时，需要额外的循环来处理，这也会增加代码大小与控制开销。

所以针对具体嵌套循环与机器结构，需要选择合适的展开因子，以使循环展开在性能上的收益最大。

由于本论文中矩阵是方阵，根据陆平静等研究成果[4]，循环展开的次数 U 满足不等式：

$U^2+U+1 \leqslant NR$ (2)

其中 NR 是浮点寄存器的数量，U^2 个寄存器用于矩阵 C 的寄存器分块，U 个寄存器用矩阵 B 的寄存器分块，1 个寄存器用于矩阵 A 的寄存器分块。

4. 实验及分析

4.1 实验环境

平台：Pentium(R) 4 CPU，时钟频率是 2.94GHz，内存是 736MB，测试工具是 Microsoft Visual Studio 2005。做实验时关闭了所有与本实验无关的应用程序与软件。

4.2 实验结果

下面的测试结果是以时钟为单位的矩阵相乘的运行时间，直接使用了处理器指令 RDSTC[6]获得，这样能够测得较高的计时精度。嵌入的汇编代码如下：

```
__int64 StartTime,EndTime;
_asm
    {
     RDSTC
     mov DWORD PTR StartTime,eax
     mov DWORD PTR StartTime,edx
    }
〈…test code…〉
_asm
    {
     RDSTC
     mov DWORD PTR EndTime,eax
     mov DWORD PTR EndTime,edx
    }
```

优化前使用 release 版本运行时间如下（单位：时钟）

	NUM=128	NUM=256	NUM=512	NUM=1024
第 1 次实验	13704152	305627663	36514333790	289407752766
第 2 次实验	13698773	327034785	36412358191	289482595732
第 3 次实验	13699158	265769944	36519242650	289486428926
第 4 次实验	13702975	296282052	36411396186	289714621647
第 5 次实验	13737790	311720266	36361762910	289423908796
第 6 次实验	13700478	312090779	36501929805	289366855085
第 7 次实验	14068109	237262894	36518671926	289425403388
第 8 次实验	13687124	291020158	36502552944	289587022032
第 9 次实验	13691909	305612571	36462381956	289417199181
第 10 次实验	13713117	319425975	36382396644	289425433132

优化后使用 release 版本运行时间如下（单位：时钟）

	128|64|4	256|64|4	512|64|4	1024|64|4
第 1 次实验	9125611	74041385	749353924	5877229413
第 2 次实验	9125831	77483648	725082226	5876553969
第 3 次实验	9123037	74228341	745932836	5917390204
第 4 次实验	9137590	74152034	734283561	5831320187
第 5 次实验	9128493	74032079	725245257	5833471633
第 6 次实验	9130891	74320345	764198710	5880502859
第 7 次实验	9138459	74041539	722679342	5878561337
第 8 次实验	9131045	74244049	723165619	5828282889
第 9 次实验	9146698	79021734	721541788	5883396728
第 10 次实验	9125754	74049129	721403606	5864720048

注：标题行 3 个数据的含义依次是 NUM、BLOCK_SIZE 和循环展开次数。

4.3 实验数据处理与分析

	128	256	512	1024
优化前	13740359	297184709	36458702700	289473722069
优化后	7131341	74961428	733288687	5867142927
加速比	1.826756	2.964502	48.719440	48.338106

由于测试数据受机器操作系统等影响，所以每次执行程序测试数据不尽相同。为了测试准确，采用测试多次求平均值的办法。另外，优化后程序所采用的 BLOCK_SIZE 和循环展开次数是根据算法分析得到的最佳值。这个最佳值也是经过程序运行结果验证的。

通过实验数据分析结果可以看到，使用矩阵 B 转置、嵌套循环分块与循环展开后矩阵相乘程序运行所使用的时钟周期数比使用前减少了很多，加速比最多将近五十倍，这是因为未优化的经典矩阵相乘算法没有考虑到局部性原理与并行计算技术对程序总的执行时间的影响。从实验加速比上看随着数据规模的扩大加速比越来越大，但是，当 NUM 值为 512 时达到最大值，可见使用本论文的优化技术优化矩阵相乘算法效果是显著的。

5. 结束语

本文首先根据局部性原理，对矩阵 B 进行转置，通过 VTune™ 性能分析器分析矩阵乘法程序的热点——循环，运用嵌套循环分块、循环展开技术消除了矩阵乘法程序中的热点，并做了实验验证运用上述方法取得了较好效果。

本文作者的创新点：运用矩阵转置后，在矩阵相乘核心程序段运用嵌套循环分块与循环展开提高程序的并行度。

参考文献

[1] 张新菊，刘羽，韩枭. 行划分的矩阵相乘并行改进及其 DSP 实现. [J]微计算机信息 2008（24）7－2,216

[2] 多核系列教材编写组. 多核程序设计[M]. 北京：清华大学出版社,2007.9,61

[3] Jose M. Llaberia Marta Jimenez, Agustin Frernandez. Register tiling in nonrectangular iteration space. ACM Transactions on Programming Languages and Systems, 2002, 24(4): 409-453

[4] 陆平静，王正华. 结合经验搜索和分析模型的代码优化方法研究. 国防科学技术大学硕士论文, 2006, 29,36

[5] http://www.sc-conference.org/sc2004/.

[6] [美]Richard Gerber, Kevin B.Simth, [荷]Aart J.C.Bik, [加]Xinmin Tian. 王涛, 单久龙, 孙广中等译. 软件优化技术－－IA-32 平台的高性能手册（第 2 版）[M]北京：电子工业出版社 2007

作者简介：钱晓捷（1963－）男，副教授，研究方向：微处理器结构，微型机应用，计算机辅助教学

Biography: QIAN Xiaojie (1963-), Male, Associate Professor, Major in: Microprocessor Architecture, Microcomputer Application, Computer Aided Instruction