ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.6,No.3,January 2010, pp.693-695

E-mail: eduf@cccc.net.cn
http://www.dnzs.net.cn
Tel:+86-551-5690963 5690964

基于并行存储优化的矩阵乘法运算

吴猛,刘振

(中国矿业大学 计算机科学与技术学院,江苏 徐州 221008)

摘要:该文就数值运算中常见的矩阵乘法运算的实现算法展开讨论,从时间和空间不同角度分析矩阵乘法运算中影响性能的主要因素,改良了原有算法,提出了基于存储优先的数据访问方式,并结合当今比较热门的并行运算机制,提高了矩阵乘积运算的速度。

关键词:矩阵;复杂度;Cache miss;并行运算;存储;shared memory

中图分类号:TP312 **文献标识码**:A **文章编号**:1009-3044(2010)03-693-03

An Improved Parallel Matrix Multiplication Algorithm Based on Memory

WU Meng, LIU Zhen

(College of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221008, China)

Abstract: In this paper, we talk about the realization of the matrix multiplication algorithm, which is common in numerical calculation, from a different perspective of time and space matrix, and discuss its main factors affecting the performance, Improved the original algorithm, based on the priority of the data storage access methods, and compared with today's popular parallel computing mechanism, enhances the speed of the matrix multiplication operation.

Key words: Matrix; complexity; Cache miss; parallel computing; storage; shared memory

1 概述

在数值计算中,矩阵乘法是最基本和经常使用的运算之一,它的性能对数值计算的操作性能产生直接的影响。我们知道,典型n×n稠密矩阵乘法运算的时间复杂度为$O(n^3)$,它的平凡下界是$O(n^2)$。可以用$O(n^{w+e})$来表示标准矩阵的乘积运算复杂度,w表示n*n矩阵必须的复杂度$2(e>0)$[1]。后来Strassen引入分治思想将w+e从3将为2.81(lg7),目前已知的最好计算时间上界是Coppersmith和Shmuel Winograd提出的$O(n^{2.376})$。

目前,对于大型矩阵乘积运算的处理普遍采用分治思想,将运算分布在多个结点上。每个结点单独完成部分运算,然后将结果汇总。基于Coppersmith和Shmuel Winograd的算法(甚至是Strassen的算法)实现复杂,在结点运算中,如果采用,不仅在算法实现难以实现,而且会导致大量的冗杂数据,CPU运算次数少了,但是大量数据的频繁交换还是会使存储体的读取速度远不及CPU的速度,这样的交换在追求效益的社会上是不划算的,因而实际运算中采用的还是经过优化了的普通矩阵乘法。

本文从影响运算性能的实际因素中找到关键点,并由此提出优化算法,改善运行环境,从而解决矩阵乘法运算的效率问题。

2 基本思想

2.1 Cache miss以及分块思想

矩阵乘法在算法上很容易实现,即三重循环:

```
matrix multiplication (a, b: matrices)
for i:=0 to n-1
for j:=0 to n-1
begin
cij :=0
for q:=1 to n-1
cij := cij+aiq xbqj
end
```

而由于计算机存取数据的时间已经对运算时间产生了影响,Cache的存在一定程度上解决了存储器传输速度和CPU处理速度的瓶颈。但对于大型的数据处理时,数据的读写仍制约着运算的效率。在实际运算中,数据读写所花的时间已经远超过CPU有效处理时间。比如,上式中n很大,嵌套循环中的数据频繁访问便导致cache hit次数的急剧下降,嵌套循环中的数据访问顺序也同样导致较多的Cache miss,如此繁琐无序的Cache读取使得CPU不得不从内存中单独读取所需的数据,而CPU对主存的访问时间是对Cache访问时间的10倍左右,使得实际运算效率远远低于理论效率。如图1,当矩阵规模增加时,实际运算时间几乎成指数方式增长。

由此可见,Cache的有效载入对运算性能的提升起到了举足轻重的作用。考虑到影响Cache hit次数的影响因素:1)空间局部性;2)时间局部性,我们从这两方面入手。

2.1.1 空间局部性

由于Cache读取数据是以块(block)为单位的,每操作一次内存,便读取相邻的一块数据。就数组形式存储的矩阵来说,一般情况

收稿日期:2009-12-09
作者简介:吴猛(1988-),男,江苏徐州人;刘振(1988-),男,江苏睢宁县人。

下,其在内存中的存储方式是按行存储的。[2]即:

Ad.a[s1,s2] = Ad.a[1,1] + {(s1-1)*n + (s2-1)}*k　　(k 为元素大小)

因而,在数据操作时尽可能的对连续数据进行集中处理。我们进行了如下处理:先将 matrix b 转置,得到 b^T,再与 a 相乘,于是在内层循环中可以对读取的连续数据块进行集中处理,而不是标准算法中对 matrix b 的跨空间读取,因而在数据较大时(Cache 读取的一块存不了一次操作所需的所有数据),有效地降低了 Cache miss 次数。由图 1 可见,当 N>500 时,改良后的算法比普通算法有近一倍的性能提升。这里,我们姑且将此改良算法称为 T-Matrix,即基于存储访问方式优化的算法。

进一步深入 Cache 的空间因素,当矩阵足够大时,Cache 无法载入一行或一列,或是频繁的换行换列读取,直接导致了较多的 Cache miss。采用分块思想可有效解决上述数据过大的问题。通过分块思想,集中访问取入 Cache 的块状矩阵,避免了全行全列的读写,增强了空间和时间的局部性,分块的算法如下:

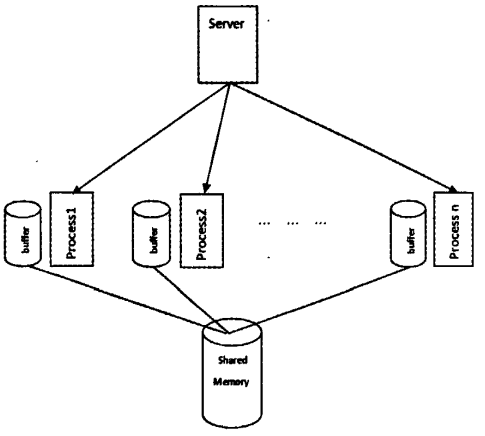

图 1　并行运算实现机制

```
Matrix multiplication (a, b: matrices  nb:block  factor)
for x:=0 to N;  y:=0 to N
for i:=0 to N;  j:=0 to nb
r = 0;
for k:=0 to nb
r = r + aik*bkj;
cij =cij + r;
```

分块思想的难点和重点在于如何定分块的大小(即 nb 的大小),过大过小都可能影响运算性能[3]。根据一般层次,按行或按列分块的选择,可以有 2*2*2 种选择(考虑到前一个矩阵的列数要和后一个矩阵的行数相同,即 $a_{ik}*b_{kj}$),上述分块算法在内层循环(也就是上述分块形式中最小块)重复使用 $a_{ik}*b_{kj}$,而这些数据一直保存在 Cache 中,提高了 Cache hit 的效率。

2.1.2 时间局部性

在上例中,将要访问的一小块数据在统一的时间内集中处理,避免了不同时间重复读取相同数据的时间浪费,增强数据的可重复利用性,并将计算所得分批次的顺序存储在 matrix c 中,这种实现机制从时间利用的角度出发,充分利用数据的重复使用特性,减少了数据读取的频繁程度,获得了较多的 Cache hit 次数。

```
...
Begin
for k:=0 to nb
r = r + aik*bkj;
cij =cij + r;
end
```

2.2 并行机制

讨论关于空间时间局部性的算法实现都是基于串行化原理处理的,并未引入并行化或是分布式并行处理的思想。我们下面将从此处着手,提出更加优化的算法。

3 并行处理

并行处理的基本思路就是利用多个部件完成同一个任务。它的好处就在于可以很好的缩小解题规模和缩短解题时间,并且它对硬件的要求不高,因而可以有效地降低成本。

基于分布式存储,将矩阵乘积运算划分成相对对立的几个模块,每个模块对应整个数据场的一小部分,而每个 CPU 则负责这数据处理这一小部分的数据,从而达到分而治之的目的。[4]

基于上述原理,我们提出了并行处理下的优化算法,即 PT-M。图 1 总体说明了算法的实现机制。Server 是分发任务并将结果汇总,是整个系统的控制中心。

一个 Process 对应着一块缓存区,将 ai 对应 Process i 以及所在的缓冲区。Server 将 ai 复制到相应的缓冲区内,与共享内存中的 matrix b 完成最终的乘法运算,并将结果同样保存在 shared memory 中。图 2 显示了运算过程中数据在不同存储设备中的流向。

算法实现:

```
Begin:
将 ai 按行分组至 process i 相应的缓冲区内
求的 b^T;存储在 shared memory 中;
for i:= 0 to n
process i: ai*b^T;结果存储在 shared memory 中;
直接从 shared memory 读取最终的结果
end
```

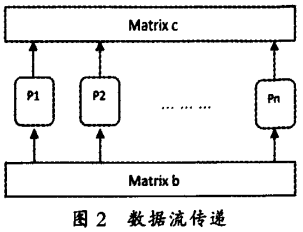

图 2　数据流传递

4 测试结果及分析

图 3 为 T-M 算法与其他算法的比较。

图 4 显示了基于 T-M 算法优化后的并行运算在 n = 4 下和普通算法在运算时间上的对比。性能提升上基本在 4 倍以上,但在规模较小时(N<500),PT-M 算法并没有多大优势。另外此种算法在稳定上还有待提高。

5 结束语

本文从实际角度出发,就影响矩阵运算性能的因素逐步展开讨论,提出了更加优化的算法,并同其它算法进行了比较,验证了算法的有效性。并行运算在科学运算中经常遇到,单它的实现较为复杂,实现过程中增加了不少外部因素,使得运行效率可能远不如原有算法。看来,关于并行运算的高效实现还有待进一步的研究。

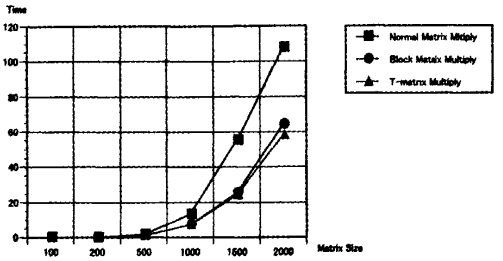

图 3 T-M 算法与其他算法的比较

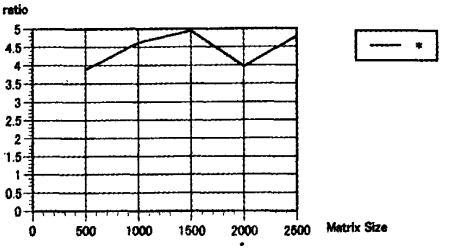

图 4 PT-M 算法与普通算法在不同规模下的对比

参考文献:

[1] Cohn H,Kleinberg R,Szegedy B,et al.Group-theoretic Algorithms for Matrix Multiplication[C].Proceedings of the 2006 international symposium on Symbolic and algebraic computation,2006.

[2] Dhamfhere D M.系统编程与操作系统[M].北京:电子工业出版社,2001.

[3] 蒋孟奇,张云泉,宋刚,李玉成.GOTOBLAS 一般矩阵乘法高效实现机制的研究[J].计算机工程,2008,34(7):84-86.

[4] 陈国良.并行算法的设计与分析[M].北京:高等教育出版社,2002.

[5] Md Islam N, Md Islam S,Kashem M A,et al.An Empirical Distributed Matrix Multiplication Algorithm to Reduce Time Complexity[C].IMECS 2009,2009.

(上接第 690 页)

4 总结

Matlab 编程代码接近数学推导公式,简捷直观,与科技人员的思维和书写习惯相适应,操作简单,人机交互性能好,并且能方便迅速的用三维图形、图像、声音、动画等表达计算结果。本文用 Matlab 语言实现了基本的水力学仿真计算,对明渠流、有压管流和堰流等水力计算进行了深入研究,得到了相应的结果并进行了验证。

参考文献:

[1] 严煦世,范瑾初.给水工程[M].3 版.北京:中国建筑工业出版社,1995.

[2] 彭永臻,崔福义.给水排水工程计算机程序设计[M].北京:中国建筑工业出版社,1994.

[3] 郑阿奇.ASP.NET 实用教程[M].北京:电子工业出版社,2004.

[4] 汤代禄.ASP 案例开发集锦[M].北京:电子工业出版社,2005.

[5] 甄镭..NET 与设计模式[M].北京:电子工业出版社,2005.

图 7

基于并行存储优化的矩阵乘法运算

作者： 吴猛， 刘振， WU Meng， LIU Zhen
作者单位： 中国矿业大学,计算机科学与技术学院,江苏,徐州,221008
刊名： 电脑知识与技术
英文刊名： COMPUTER KNOWLEDGE AND TECHNOLOGY
年，卷(期)： 2010,6(3)

参考文献(5条)

1. Md Islam N;Md Islam S;Kashem M A An Empirical Distributed Matrix Mtdtiplication Algorithm to Reduce Time Complexity 2009

2. 陈国良 并行算法的设计与分析 2002

3. 蒋孟奇;张云泉;宋刚;李玉成 GOTOBLAS一般矩阵乘法高效实现机制的研究[期刊论文]-计算机工程 2008(07)

4. Dhamfhere D M 系统编程与操作系统 2001

5. Cohn H;Kleinberg R;Szegedy B Group-theoretic Algorithms for Matrix Multiplication 2006