集群下 Cholesky 分解的核外预取算法

刘 凤，刘青昆

(辽宁师范大学 计算机与信息技术学院，辽宁 大连 116081)

摘 要：核外计算中，由于 I/O 操作速度比较慢，所以对文件的访问时间占的比例较大。如果使文件操作和计算重叠则可以大幅度地提高运行效率。软件数据预取是一种有效的隐藏存储延迟的技术，通过预取使数据在实际使用之前从硬盘读到缓存中，提高了缓存(cache)的命中率，降低了读取数据的时间。通过设置两个缓冲区来轮流存放本次和下一次读入的数据块，实现访存完全命中 cache 的效果，使 Cholesky 分解并行程序执行核外计算的效率得到了大幅度的提高。同时，I/O 操作的时间与 CPU 的执行时间的比例也是影响效率的主要因素。

关键词：预取；核外；并行；集群；Cholesky 分解

中图分类号：TP311 **文献标识码**：A **文章编号**：1674-7720(2011)04-0014-04

Out-of-core prefetching algorithm of Cholesky decomposition on clusters

Liu Feng, Liu Qingkun

(College of Computer and Information Technology, Liaoning Normal University, Dalian 116081, China)

Abstract: The time of accessing files is larger proportion because the I/O operation is slow in out-of-core computation. If the file operation and the calculation is overlap, the operational efficiency can be improved greatly. Data prefetching of software is an effective technique to hide memory latency, because it can improve the cache hit rate and reduce the time of reading data that make the data read from the hard drive to the cache before using the date actually. We set the two buffers to store the time and the next block of reading data in turn, to achieve the effect of accessing memory hitting cache completely and the execution efficiency of the parallel program of Cholesky decomposition has been improved greatly in out-of-core computation. At the same time, the proportion between I/O operation time and CPU execution time is also an important factors effecting efficiency.

Key words: prefetching; out-of-core; parallel; clusters; Cholesky decomposition

随着科学技术的迅猛发展，人们需要处理的数据量迅速增长。大规模并行应用涉及的数据量是非常惊人的，内存容量常常不能满足涉及到大数据量计算问题的存储需求。因待处理数据无法全部读入内存，所以只能保存数据到外部磁盘系统。当程序运行时，某个时刻只能将部分数据调入内存并参加计算，并且在适当的时候写回外存，通过某种策略实现内外存数据的交换。由于运算过程中数据并没有全部存放在内核存储器中，所以称之为核外计算。核外计算程序中包含大量的文件操作，因访问磁盘数据的速度比较慢，所以在处理大数据量问题时，I/O 性能显然要超过 CPU 性能而成为重要的限制因素。因此，如何对核外数组进行合理调度与高效访问是解决核外计算应用问题的关键。

利用三角分解式求解对称正定方程组是一种有效方法。单行卷帘存储 Cholesky 分解[1]使每个节点所分配的数据最多相差一行，但是这种划分方式的通信开销比较大。多行卷帘存储 Cholesky 分解[2]可以减少通信开销，充分利用节点的缓存，但是没有充分考虑缓存的效率问题，因为缓存可以对程序性能带来巨大的改变。递归算法[3]通过将矩阵分块使各个子矩阵的运算能够在高速缓存中进行，以提高运算效率，同时递归算法良好的数据局部性和矩阵递归的分块，使其非常适合分层多级存储的计算机结构。但是在递归调度算法中，非对角块的运算不能充分利用计算机的分级存储结构。分块 Cholesky 分解[4-5]通过精心的挑选块的大小，使每个块都能充分地利用一级缓存，并且合理地划分块的大小还能使每个块常驻内存。参考文献文[4]充分利用了系统硬件的并行机制以使通信与计算重叠，减少了节点的等待时间。同

时通过矩阵元素的重排使每个块都被连续存储，以充分利用计算机的多层次存储结构，减少数据拷贝和内部的变化。但是这些算法只在数据量较小的情况下适用；当数据量较大时，数据将无法一次性读入内存，因此引进了核外计算的概念。

本文对 Cholesky 分解的并行算法进行了深入的研究，给出的核外算法通过预取[6-7]的方法使得数据在实际使用之前从硬盘移到缓存中，从而进一步提高了缓存的命中率，减少了文件读取的时间。

1 基本概念

1.1 Cholesky 分解

Cholesky 分解用于求解系数矩阵对称正定的线性方程组。

$$AX=b \tag{1}$$

式(1)中，A 为实对称正定矩阵，且 A 的所有顺序主子式均不为零，X 和 b 为矩形矩阵或向量。为了避免平方根法 Cholesky 分解 $(A=LL^{T})$ 的开方运算且扩大使用范围，将矩阵对称正定矩阵 A 分解成 $A=LDL^{T}$。当 A 是正定对称矩阵时，A 的分解是唯一的。其中 D 是对角线元素全为正的对角矩阵，L 是单位下三角矩阵。其求解公式如下：

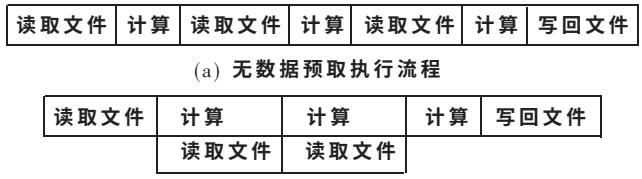
$$\begin{cases} d_k=a_{kk}-\sum_{m=1}^{k-1} l_{km}^2\ d_m,\ k=1,2,\cdots,n \\ l_{jk}=\left(a_{jk}-\sum_{m=1}^{k-1} l_{jm}l_{km}d_m\right)/d_k,\ j=k+1,k+2,\cdots,n \end{cases} \tag{2}$$

其中，$L=\begin{bmatrix} 1 & & & \\ l_{2,1} & 1 & & 0 \\ \vdots & & \ddots & \\ l_{n,1} & l_{n,2} & \cdots & 1 \end{bmatrix}$，$D=\begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & d_n \end{bmatrix}$，由(1)、(2)两式可变换得：$LDL^{T}X=b$。

1.2 预取方法

在核外计算中数据存储在磁盘上，只有 CPU 对数据进行处理时才将其读入内存缓冲区，处理完成后写回文件。这样，核外计算过程就可以描述成：读入核外数组文件的一块数据到内存缓冲区，进行计算等处理，处理完成后将其写回数组文件；读取下一块数据，进行数据处理，处理完成后写回文件。如此反复执行，直到整个核外数组处理完成后结束。显而易见，这是一个顺序流水线，执行 I/O 操作与 CPU 进行计算是顺序执行的；由于二者在执行前需彼此等待，所以程序的执行效率非常低。于是考虑到，当 I/O 操作的数据与 CPU 执行计算的数据无相关性时，可以把对下一个数据块的 I/O 操作与对当前数据块的计算重叠起来，这样就可以隐藏 I/O 的延迟，达到缩短整个程序运行时间的目的。图 1 所示为数据预取前后的对比。

本算法采用双缓冲区的方式，即为一个核外数组分配两个内存缓冲区 buffer[2]，轮流存放本次和下一次读入的数据块。用预取的方法对第 s 个子文件更新，其伪码如下：



(a) 无数据预取执行流程

(b) 有数据预取执行流程

图 1 数据预取前后的对比

(1)主节点将第一个子文件读入内存数组 buffer[0]中，并将其广播出去；

(2)依次用已分解的子文件更新第 s 个子文件，即对 $ka=1,2,\cdots\cdots,s-1$ 循环：

若该节点是主节点，做以下工作：①读入第 ka 个文件到 buffer[ka%2]中；②发送第 ka 个文件；

若该节点是从节点，则做以下工作：①用第 $ka-1$ 个文件更新第 s 个文件；②接收第 ka 个文件。

(3)若该节点是从节点，则用第 $s-1$ 个文件更新第 s 个文件。

2 用预取算法实现 Cholesky 分解

2.1 Cholesky 分解并行算法的思想

将大规模矩阵 A 连续拆分成 q 个子文件，并将其储存到主节点中。主节点一次调入内存一个子文件，依次对各子文件进行 Cholesky 分解，直至最后一个子文件分解完毕。所以只要拆分的子文件大小不超过空闲内存的范围，算法就可以运行。其算法流程见图 2，具体步骤如下：

(1)文件拆分过程

由主节点机进行矩阵拆分存储的操作，打开存储于硬盘中的 A.txt 文件，依内存容量的需要进行拆分，读取相应规模的数据，生成子文件，并将这些子文件 j.txt(j=1,2,3)存储回硬盘。

(2)A 阵的 Cholesky 分解过程

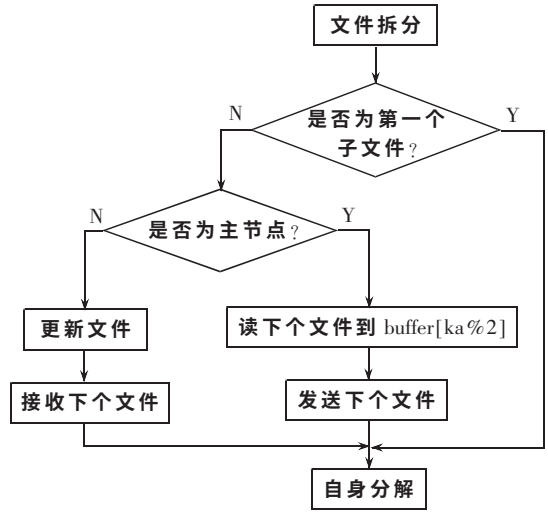


图 2 Cholesky 分解流程图

考虑到 Cholesky 分解过程中的依赖关系，对其由上至下依次进行分解。由于在三角分解中，既需要分解后的上三角，又需要分解后的下三角，所以可以将分解后的上三角复制到下三角。这样既可以减少运算量又可以减少存储量。在分解过程中利用其并行性，主节点打开第一个子文件 1.txt 后，赋值给 $a_{6\times15}$，再将 a 广播出到各个从节点。每个从节点经卷帘计算出分解因子后，广播出去，如此地计算 $kb=2$ 次，该子文件的 Cholesky 分解完毕。当 1.txt 计算结束后，主节点打开下一个子文件 2.txt，赋值给 $a_{6\times15}$，再将 a 广播到各个从节点。同时主节点还要打开分解完毕的子文件 1.txt，赋值给 $aa_{6\times15}$（即 buffer[0]），再将 aa 广播到各个从节点。各个从节点用 $aa_{6\times15}$ 卷帘去更新 $a_{6\times15}$，更新完毕后 $a_{6\times15}$ 再像上面的操作，每个从节点卷帘的计算出分解因子后，广播出去，如此的计算 $kb=2$ 次后，主节点打开下一个子文件 3.txt，进行如文件 2.tx 的操作。

2.2 Cholesky 分解并行算法的描述

将 A 连续拆分存储到 q 个子文件中，依次对各子文件进行 Cholesky 分解，直至对最后一个子文件分解后，原大规模系数矩阵已被拆分到几个小文件中存储。将节点机分别标记为 $P_0,P_1\cdots P_{np}$，np 代表从节点机个数，P_0 代表主节点，kb 表示在一个子文件中单个节点机计算的行数，s 是子文件依次分解的循环控制变量，ka 表示读取已分解子文件的循环控制变量，在拆分后 $a\in R_{[kb\times np]\times n}$，$aa\in R_{[kb\times np]\times n}$。具体步骤如下：

(1)文件拆分

主节点机打开存于硬盘中的 A.txt 文件，按内存容量的需要进行拆分，读取相应规模的数据，生成 q 个子文件。

其中，$q=\begin{cases}n/(np\times kb),q\times np\times kb=n\\ n/(np\times kb)+1,\text{其他}\end{cases}$

(2)A 矩阵的 Cholesky 分解过程

①1.txt($s=0$)的 Cholesky 分解：(a)主节点打开 1.txt，读取文件内容，赋值到 $a[np*kb][n]$ 数组中，并将数组 a 广播出去。(b)从节点 myid(myid=1,2,……,np)将数组 a 中的第 i 行(i%np=myid−1)进行 Cholesky 分解。(c)从节点将已分解的对角元素赋给数组 $d[n]$。(d)从节点用数组 a 已分解的元素重写其下三角。(e)从节点将分解后的数组 a 写回主节点。

②文件 j.txt($s=1,2,\cdots\cdots,q-1,j=s+1$)的 Cholesky 分解：(a)主节点打开并读取 j.txt 内容，赋值到 a 且将其广播出去。(b)用预取的方法对第 j 个子文件更新。即主节点依次打开已分解文件 ka.txt(ka=0,1,⋯,$s-1$)，赋值给 buffer[ka%2]，并将 buffer[ka%2]广播出去。在各个从节点根据 buffer[ka%2]的值更新数组 a 值的同时，主节点读取下一个已分解的文件。(c)用数组 buffer[ka%2]的元素重写数组 a 的下三角。(d) 将数组 a 进行 Cholesky 分解。(e)将数组 a 本次分解的对角线元素赋值给数组 $d[n]$。(f)用数组 a 本次分解的元素重写其下三角。(g)将分解后的数组 a 写回主节点。

2.3 算法的复杂度分析

对于点对点的通信，测量开销使用乒乓法：节点 0 发送 m 个字节给节点 1；节点 1 从节点 0 接收 m 个字节后，立即将消息发回节点 0。总的时间除以 2，即可得到点到点通信时间，也就是执行单一发送或接收操作的时间。通信开销的解析表达式是消息长度 m(字节)的线性函数：$t_{comm}(m)=T_s+T_b\times m$；其中 T_s 表示通信的启动时间，T_b 表示发送每个字节所需时间（它是带宽的倒数）。由于 MPI_Bcast 函数采用树算法，所以一次 MPI_Bcast 的通信开销为 $t_{comm}(s)\times\log P$。Cholesky 分解的执行时间可分为子文件的更新时间和自身分解时间。因此，Cholesky 分解时间：$T_c=T_g+T_f$，其中 T_g 是子文件更新时间，T_f 是自身的分解时间。在文件的更新时间中，又细化为读文件时间 T_d、计算时间 T_j 和通信时间 T_t。为了减少分解的执行时间，本文通过预取的方法使更新过程中的读文件与计算重叠，使得 Cholesky 分解的时间开销：$T_c<T_f+T_d+T_j+T_t$。当文件的读取与计算完全重叠时，Cholesky 分解的时间可表示为 $T_c=T_f+T_t+\max(T_d,T_j)$。

子文件的大小是通过参数 kb($1\leqslant kb\leqslant n/np$)而设定的，当 $kb=n/np$ 时相当于无文件划分并行求解；当 $kb=1$ 时，在每个子文件中每个节点机只计算一行，此时的通信量最大(按照这种分配方法划分数据后，每个处理器上须存储的内容为 $kb\times np\times n$ 规模的矩阵 A)。因此算法的并行执行时间：$T_n=T_1+T_c+T_2$，其中，T_1 为划分子文件消耗的时间，T_c 是 Cholesky 分解时间，T_2 为冗余的控制和管理开销。

3 实验测试与结果分析

实验环境采用由 5 台主频为 2.8 GHz 的 Intel Xeon CPU，内存为 ECC DDR-2 SDRAM 2 GB 的 Dell PowerEdge 2850 构建的集群。该集群运行 Linux RH9 操作系统，并且建立了 MPI 并行编程环境。本文测试的两个程序分别为：带状循环划分的核外程序 out 和带状循环划分的核外预取程序 pre。

表 1 表示当 A 的阶数 n=2 478，核外 Cholesky 分解各个部分的执行时间对比。表 2 表示了在节点数 np 分别为 2,3,4,5 时，Cholesky 分解中的子文件更新各个部分的时间对比($kb=10,n=2\,479$)。图 3 为核外预取程序相对于核外程序更新时间的效率提高百分比。

由表 1 可见，在求解过程中 Cholesky 分解中的更新部分占了大部分时间，所以提高更新部分的执行率能明

表 1 矩阵规模 n=2 478，Cholesky 分解各部分执行时间

划分时间/s	文件更新时间/s	文件自身分解时间/s
11.2	644.7	6.1

表 2 out 和 pre 的 Cholesky 分解更新时间对比

节点数	通信时间/s		读取文件和计算时间/s		更新时间/s	
	out	pre	out	pre	out	pre
2	13	13	692	635	705	648
3	13	129	487	310	617	439
4	304	305	283	201	587	506
5	358	354	207	155	565	509

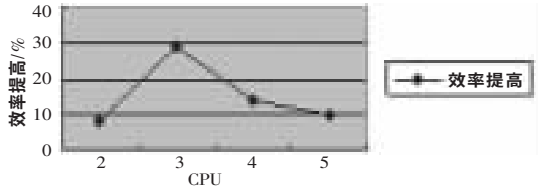


图 3 pre 相对于 out 的更新时间效率提高百分比

显缩短总的执行时间。

由表 2 可见，预取方法使得子文件的更新时间平均缩短了 15%左右。

图 3 表明，核外预取分解并行算法的效率明显好于核外分解并行算法，并且随着节点个数的增加，提高的效率先增大后减小。这是因为随着节点数目的增加，子文件容量增加，计算量增加，计算与文件读取的重叠时间越多。在节点数 $np=3$ 时，效率提高幅度最大。当节点数 $np>3$ 时，随着节点数的增加通信时间急剧增加，而计算时间逐渐减少，计算与文件读取的重叠时间减少，所以执行效率提高的幅度降低。

本文通过对 Cholesky 分解并行算法的研究，表明预取方法是计算核外稠密线性方程组的有效方法，非常有利于缩短 I/O 与 CPU 速度间的差距。数据预取的方法可以做到计算与 I/O 并行，即在计算一块数据的同时读入下一块要处理的数据。为了存放预取的数据，本文采用双缓冲区的方式，即为一个核外数组分配两个内存缓冲区，轮流存放本次和下一次读入的数据块。此外，预取方法同样适用于 QR 分解、LU 分解、高斯削去等其他线性代数问题。当然采用数据预取技术时，I/O 操作时间与 CPU 执行时间的比例是保证预取效果的关键。

事实上，Cholesky 分解也可以采用数据重用的方法。通过减少对子文件的读取次数，可以进一步提高 Cholesky 分解算法的效率。由于目前分布存储计算机的处理速度都很高，而其网络通信速度较慢，所以用增加计算和通信粒度的办法来降低通信成本。

参考文献

[1] 迟学斌. Transputer 上 Cholesky 分解的并行实现[J]. 计算数学，1993(3):289-294.

[2] 王顺绪，周树荃.卷帘行存储下的一种并行 Cholesky 分解及其在 PAR95 上的实现[J].南京航空航天大学学报，1999，31(4)：428-432.

[3] 陈建平. Jerzy Wasniewski，Cholesky 分解递归算法与改进[J].计算机研究与发展，2001，38(8):923-926.

[4] GUSTAVSON F G, KARLSSON L, KAGSTRO B M. Distributed SBP cholesky factorization algorithms with near-optimal scheduling[J]. ACM Transactions on Mathematical Software, 2009, 36(2):1-25.

[5] ANDERSEN B S, GUNNELS J A, GUSTAVSON F G, et al. A fully portable high performance minimal storage hybrid format cholesky algorithm[J]. ACM Transactions on Mathematical Software, 2005,31(2):201-227.

[6] 丁文魁，汪剑平，向华，等.p-HPF 并行编译系统核外计算的实现及优化策略[J].计算机学报，1999，22(10):1042-1049.

[7] 姚维.Linux 下一种磁盘节能的预取算法[J].计算机系统应用，2010,19(7):91-94.

(收稿日期：2010-11-16)

作者简介：

刘凤，女，1982 年生，硕士，主要研究方向：并行计算、集群计算机系统。

刘青昆，男，1971 年生，副教授，主要研究方向：并行计算与分布式处理、集群计算机系统、嵌入式系统。