Cholesky 分解综述

摘要　矩阵运算是高性能计算中核心问题之一,矩阵分解是提高矩阵运算并行性的重要途径。Cholesky 分解是一种分解矩阵的方法, 在线形代数中有重要的应用。本文分析了 Cholesky 分解的基本形式和算法实现，同时对基于 GPU 的稀疏矩阵 Cholesky 分解做了一定研究。

关键词　Cholesky 分解; 矩阵

1 介绍

矩阵运算是高性能计算中核心问题之一,矩阵分解是提高矩阵运算并行性的重要途径。Cholesky 分解是一种分解矩阵的方法, 在线形代数中有重要的应用。Cholesky 是生于 19 世纪末的法国数学家，曾就读于巴黎综合理工学院。Cholesky 分解是他在学术界最重要的贡献。后来，Cholesky 参加了法国军队，不久在一战初始阵亡。Cholesky 分解把矩阵分解为一个下三角矩阵以及它的共轭转置矩阵的乘积（那实数界来类比的话，此分解就好像求平方根）。Cholesky 分解法，又名平方根法，是求解对称正定线性方程组最常用的方法之一。与一般的矩阵分解求解方程的方法比较，Cholesky 分解效率很高。

稀疏矩阵 Cholesky 分解是求解大规模稀疏线性方程组的核心算法，也是求解过程中最耗时的部分. 近年来，一系列并行算法通过图形处理器（GPU）获得了显著的加速比，然而，由于访存的不规则性以及任务间的大量数据依赖关系，稀疏矩阵 Cholesky 分解算法在 GPU 上的计算效率很低。

1.1 Cholesky 分解的条件

- Hermitianmatrix：矩阵中的元素共轭对称（复数域的定义，类比于实数对称矩阵）。Hermitiank 意味着对于任意向量 x 和 y，x^*Ay 共轭相等。
- Positive-definite：正定（矩阵域，类比于正实数的一种定义）。正定矩阵 A 意味着，对于任何向量 x，x^TAx 总是大于零(复数域是 $x^*Ax>0$)。

1.2 Cholesky 分解的形式

可记作 $A = L L^*$。其中 L 是下三角矩阵。L^*是 L 的共轭转置矩阵。可以证明，只要 A 满足以上两个条件，L 是唯一确定的，而且 L 的对角元素肯定是正数。反过来也对，即存在 L 把 A 分解的话，A 满足以上两个条件。如果 A 是半正定的（semi-definite），也可以分解，不过这时候 L 就不唯一了。特别的，如果 A 是实数对称矩阵，那么 L 的元素肯定也是实数。另外，满足以上两个条件意味着 A 矩阵的特征值都为正实数，因为 Ax = lamda * x,x^*Ax = lamda * x^*x > 0, lamda > 0。

1.3 Cholesky 分解的方式

可以使用高斯消元法分解矩阵。过程如下：

设 $A = \begin{vmatrix} a_{11} & w^* \\ \end{vmatrix} = (R_1^*) * \begin{vmatrix} 1 & \\ 0 & \end{vmatrix} * R_1$

$$\begin{vmatrix} w & K \end{vmatrix} \qquad \begin{vmatrix} 0 & K - w(w^*)/a_{11} \end{vmatrix}$$

其中，$R_1^* = \begin{vmatrix} 1 & 0 \\ w/\mathrm{sqrt}(a_{11}) & 1 \end{vmatrix}$

$R_1 = \begin{vmatrix} 1 & w/\mathrm{sqrt}(a_{11}) \\ 0 & 1 \end{vmatrix}$

如果 $K - w(w^*)/a_{11}$ 大于零，那么就可以一直分解下去。因为它是一个正定矩阵的子矩阵，所以肯定可以满足。

最后：$R^* = (R_1^*)(R_2^*)\ldots(R_m^*)$，　$R = (R_m)\ldots(R_2)(R_1)$

因为矩阵的一半元素很相似，所以算法只需要实现一半就可以了。数据也可以只用一半，这样就节约了很多时间。同样，输出只需要一个上三角矩阵就可以了。

1.4 Cholesky 分解的算法实现

```
R = A
for (k = 0, k < m; k++){
    for(j = k; j < m; j++){
        R_{j,j:m} = R_{j,j:m} – R_{k,j:m} R_{kj} / R_{kk}        (1)
    }
    R_{k,k:m} = R_{k,k:m}/ sqrt(R_{kk})
}
```

为了节约数据空间，其中，A 仅初始化为上三角矩阵。Cholesky 分解的效率分析，由于(1) 式占用了 O（m）的时间，所以总计 O（$m^3 / 3$）。

2 相关工作

George 等人[1]提出了一种在共享内存多处理器上进行 Cholesky 分解的并行算法。该算法基于任务池的自调度概念。George 等人考虑了六种基本消除算法的变化，这些变化对应于这三个循环的所有可能的安排。通过分析任务之间的优先关系、任务工作配置文件和由此产生的处理器利用率特征，George 等人可以确定最有希望的并行实现变体。所选择的算法，George 等人称之为 column-Cholesky，是在 Denelcor HEP 多处理器上实现的。对于同样的问题，它的性能超过了以前的算法，似乎已经接近机器能力的极限。

Rothberg 等人[2]研究了并行稀疏 Cholesky 分解的子块分解策略，该策略将稀疏矩阵分解为矩形块。与传统的面向列和面向板的分解方法相比，这种方法具有巨大的理论可扩展性。然而。在生产一种实用的子块方法方面进展甚微。Rothberg 等人提出并评估了一种易于实现的方法，该方法在小型并行计算机上提供的性能略高于列(和面板)并行算法，并且具有在大型并行计算机上提供更高性能的潜力。

Loehlin[3]注意到了对双变量 Cholesky 分析的一个常见的误解，好像它是一个常见的和具体的因素分析。初步的 Cholesky 行为遗传分析应经常转换成不同的解释形式。给出了二元情况下四种变换的公式。

Cheng 等人[4]给出一个对称的，不一定是正定的矩阵 A, 改进的 Cholesky 算法计算一个 Cholesky 因子分解 $P(A + E)P^T = R^TR$，其中 P 是一个置换矩阵，E 是一个使 A + E 为正定的微扰。目标包括产生一个小赋范的 E 和使 A + E 合理地适应条件。改进的 Cholesky 分解在优化中得到了广泛的应用。Cheng 等人提出了一种新的改进的 Cholesky 算法，该算法基于一种对称的不确定因子分解，计算使用新的旋转策略的 Ashcraft, Grimes，和刘易斯。Cheng 等人从理论和实践两方面分析了该算法的有效性，表明该算法与现有的 Gill、Murray、Wright、Schnabel 和 Eskow 算法是有竞争力的。新算法的吸引人的特性包括易于解释的不等式，这些不等式解释了它满足设计目标的程度，以及它可以用现有软件实现的事实。

论文[5]介绍了 ScaLAPACK 核外扩展中包含的| LU、QR 和 Cholesky |三个核心分解例程的设计与实现。这些例程允许对一个太大而不能完全存储在物理内存中的稠密系统进行因式分解和求解。整个矩阵的图像保存在磁盘上，因式分解例程将子矩阵传输到内存中。The `left-looking' 用于分解算法的变体实现减少磁盘 I/O trac.这些例程使用可移植的 I/O 接口实现，并利用高性能的 ScaLAPACK 分解例程作为核心计算内核。D'Azevedo 等人介绍了实现的细节为外核的 ScaLAPACK 因式分解例程，以及在英特尔 Paragon 上性能和可伸缩性的结果。

Davis[6]提出的 LDL 软件包，是一组用于分解对称正定稀疏矩阵的简短的例程，对对称不定矩阵有一定的适用性。它的主要目的是用尽可能简洁的代码来说明稀疏矩阵算法的许多基本理论，包括一种优雅的新的稀疏对称分解方法，它逐行计算因数分解，但逐列存储它。整个符号和数字分解总共只有 53 行代码。该软件包是用 C 语言编写的，并包含一个 MATLAB 接口。

袁晖坪[7]提出行(列) 转置矩阵与行(列) 对称矩阵的概念 ,研究它们的性质,获得一些新的结果. 给出行(列)对称矩阵的 LDU 分解、Cholesky 分解和三对角分解公式 ,可极大地减少行(列) 对称矩阵的 LDU 分解、Cholesky 分解和三对角分解的计算量与存储量 ,而且不会丧失数值精度。

Chen 等人[8]提出，CHOLMOD 是一组例程都稀疏对称正定矩阵的形式或在更新/修正稀疏的柯列斯基分解,解决线性系统,更新/修正解决三角系统 Lx = b,和许多其他对称和非对称矩阵的稀疏矩阵函数。它的超节点 Cholesky 分解依赖于 LAPACK 和 3 级 BLAS，并获得了 BLAS 峰值性能的很大一部分。同时支持实矩阵和复矩阵。CHOLMOD 是用 ANSI/ISO 写的 C，同时具有 C 和 MATLABTM 接口。当 A 是稀疏对称正定时，它出现在 MATLAB 7.2 中 x=A\b，以及其他几个稀疏矩阵函数中。

Volkov 等人[9]使用 8 系列 NVIDIA gpu 给出稠密线性代数的性能结果。Volkov 等人的矩阵-矩阵乘法程序(GEMM)运行速度比供应商在 CUBLAS 1.1 上的实现快 60%，已经接近硬件能力的顶峰。LU、QR 和 Cholesky 分解可以达到峰值 GEMM 率的 80-90%。并行逻辑单元在两个 gpu 上运行，可以达到大约 300 Gflop/s。这些结果是通过挑战 GPU 架构和编程指南的公认视图来实现的。Volkov 等人认为现代的 gpu 应该被看作是多线程的多核向量单位，利用类似于矢量计算机的阻塞和系统的异构性，在 GPU 和 CPU 上进行计算。这项研究包括详细的 GPU 内存系统的基准测试，以揭示缓存和 TLB 的大小和延迟。Volkov 等人提出了一些算法优化，旨在增加并行性和规律性的问题，提供略高的性能。

Rothman 等人[10]针对协方差反协方差的 Cholesky 因子的回归解释，提出了一种新的协方差矩阵的 Cholesky 因子的回归解释，从而得到了一类适用于高维问题的正则化协方差估计量。通过这种回归解释正则化协方差的 Cholesky 因子，总是得到正的估计。特别是，可以用与 Bickel & Levina (2008b)的流行的带状估计量相同的计算成本获得协方差矩阵的正定带状估计量，但不能保证它一定是正定的。建立了带带协方差矩阵的切勒斯基因子与带带约束的最大似然反估计之间的理论联系，比较了几种方法在仿真和声纳数据实例中的数值性能。

邹丹等人[11]实现了一种新的基于 GPU 的稀疏矩阵 Cholesky 分解算法．在数据组织方面，改进了稀疏矩阵超节点数据结构，通过超节点合并和分块控制计算粒度；在计算调度方面，将稀疏矩阵 Cholesky 分解过程映射为一系列的数据块任务，并设计了相应的任务生成与调度算法，在满足数据依赖性的前提下提高任务的并行性．实验结果表明，该算法能够显著提高稀疏矩阵 Cholesky 分解算法在 GPU 上的实现效率，在单个 GPU 上获得了相对 4 核 GPU 平台 2.69～3.88 倍的加速比．

刘书勇[12]基于子矩阵更新同一化算法,针对 Cholesky 分解的求解特征,提出以求解矩阵 L 的转置阵 L^T 的 L^T-SC 算法,并在分析 LTGSC 线性代数特征的基础上,给出了 L^T-SC 的并行结构实现方案.实验结果表明:与通用处理器的软件实现相比,该文实现的 Cholesky 分解的 FPGA 并行结果在核心计算性能上可以取得 10 倍以上的加速比,该算法针对矩阵三角化计算过程具有更高的数据和流水并行性。

3 稀疏矩阵的 Cholesky 分解算法

稀疏矩阵 Cholesky 分解算法将对称正定稀疏矩阵 A 分解为 $L L^T$，其中 L 是对角元素为正的下三角矩阵．由于 A 为对称矩阵，只需要存储和处理矩阵 A 的下三角部分的非零元素，L 的数值直接覆盖矩阵 A 的下三角部分．稀疏矩阵 Cholesky 分解算法按照从左向右的顺序逐列更新 L．为了便于描述计算过程，我们定义两类计算任务：列分解任务 cdiv 和列更新任务 cmod．其中，cdiv(k)对第 k 列进行分 解，cmod(k,j)使用第 j 列更新第 k 列．稀疏矩 阵 Cholesky 分解过程可以表示为一系列计算任务：按照由左至右的顺序，首先生成 cdiv(j)任务，对当前列 j 进行分解；然后生成 cmod(j+1:n,j)任务，使用当前列 j 更新右侧所有相关列．在分解过程中，矩阵 A 中 的一部分零元素会转变为非零元素，这种元素称为填充元．稀疏矩阵 Cholesky 算法描述如算法 1 所示．

算法 1. 稀疏矩阵 Cholesky 分解算法．

输入：分解前的稀疏下三角矩阵 **L**（$l_{i,j}$），矩阵规模 n

输出：分解后的稀疏下三角矩阵 **L**（$l_{i,j}$）

```
FOR j = 0 to n-1 DO
    FOR i = j to n-1 DO
        l_{i,j} = l_{i,j}/sqrt(l_{j,j})          // 列分解任务 cdiv
    FOR k = j+1 to n-1 DO
        IF l_{k,j} != 0 THEN
```

FOR p = k to n-1 DO

IF $l_{p,j}$!=0 THEN

$l_{p,j} = l_{p,k} - l_{p,j} \times l_{k,j}$ // 列更新任务 cmod

4 GPU 架构

当前用于通用计算的主流 GPU 架构是 CUDA（Compute Unified Device Architecture）架构．硬件结构方面，GPU 由多个流多处理器（Streaming Multiprocessor，SM）组成，每个 SM 包含多个标量处理器（Scalar Processor，SP）．每个 SM 中的指令发射部件将相同的指令发射到各个 SP 上，各 SP 在不同的数据集上执行相同的指令．软件结构方面，GPU 的基本编程单元是核程序（kernel），基本运算单元是线程．每个线程执行同一个核程序的代码，根据线程编号处理不同的数据．多个线程构成线程块，线程块内的线程以单指令多线程（Single Instruction Multiple Threads，SIMT）方式执行．

CUDA 的流机制能够支持多个核程序 在 GPU 上同时执行．流是一系列顺序执行的操作，流之间并发执行各自的命令．每个流拥有唯一的编号．发射到同一个流的核程序按照发射顺序执行，发射到不同流的核程序在计算资源满足的前提下能够同时运行．CUDA 的事件机制提供了检测流内的任务执行进度的方式．事件插入流的位置称为事件记载点，只有当流中在时间记载点之前的所有任务全部完成后，事件才会被记载．

在 CPU－GPU 异构系统中，GPU 作为协处理器，由 CPU 控制将数据从主机的 DRAM（Dynamic Random Access Memory）传输到 GPU 的 DRAM，然后将计算任务对应的核程序发射到 CPU 上，由 GPU 完成对数据的处理．当 GPU 运算结束后，再由 CPU 控制将 数 据从 GPU 的 DRAM 传回主机的 DRAM．

5 结论

矩阵运算是高性能计算中核心问题之一,矩阵分解是提高矩阵运算并行性的重要途径。Cholesky 分解是一种分解矩阵的方法, 在线形代数中有重要的应用。稀疏矩阵 Cholesky 分解是科学工程计算领域的核心算法之一，广泛应用在有限元分析、系统仿真模拟等领域的大规模稀疏线性方程组求解过程。为了降低大规模稀疏矩阵的存储和计算量，需要采用压缩存储数据结构，只存储和处理非零元素，从而引入了大量不规则间接寻址操作，造成了访存和计算的不规则。相应的，稀疏矩阵 Cholesky 分解并行算法需要解决数据组织与计算调度两个基本问题，即如何提高访存和计算的效率。

6 致谢

感谢这篇论文所涉及到的各位学者。本文引用了数位学者的研究文献，如果没有各位学者的研究成果的帮助和启发，我将很难完成本篇论文的写作。由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师和学友批评和指正！

参考文献

[1] A. George , M. T. Heath , and J. Liu . "Parallel Cholesky factorization on a shared-memory multiprocessor." Linear Algebra and its Applications 77.none(1986):165-187.123

[2] Rothberg, Edward , and A. Gupta . "An Efficient Block-Oriented Approach to Parallel Sparse Cholesky Factorization." Supercomputing '93. Proceedings IEEE, 1993.2

[3] Loehlin, John C. . "The Cholesky Approach: A Cautionary Note." Behavior Genetics 26.1(1996):65-69.12

[4] Cheng, Sheung Hun , and N. J. Higham . "A Modified Cholesky Algorithm Based on a Symmetric Indefinite Factorization." SIAM Journal on Matrix Analysis and Applications 19.4(1998):1097-1110.

[5] D'Azevedo, Eduardo F. , and J. Dongarra . "Design and implementation of the parallel out-of-core ScaLAPACK LU, QR, and Cholesky factorization routines." Concurrency Practice and Experience 12.15(2000):1481-1493.4

[6] Davis, Timothy A. . "Algorithm 849: A concise sparse Cholesky factorization package." ACM Transactions on Mathematical Software 31.4(2005):587-591.

[7] 袁晖坪. "行（列）对称矩阵的 LDU 分解与 Cholesky 分解." 华侨大学学报（自然科学版） 28.1(2007):88-91.

[8] Chen, Yanqing , et al. "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate." ACM Transactions on Mathematical Software 35.3, article 22(2008):1-14.

[9] Volkov, Vasily, and James Demmel. "LU, QR and Cholesky factorizations using vector capabilities of GPUs." EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-49 49 (2008).

[10] Rothman, Adam J. , E. Levina , and J. Zhu . "A new approach to Cholesky-based covariance regularization in high dimensions." Biometrika 97.3(2009):539-550.

[11] 邹丹, 窦勇, and 郭松. "基于 GPU 的稀疏矩阵 Cholesky 分解." 计算机学报 37.7(2014):1445-1454.

[12] 刘书勇, et al. "基于矩阵三角化分解的 Cholesky 分解及 FPGA 并行结构设计." 清华大学学报(自然科学版) 9(2016):963-968.