第 38 卷 第 18 期
Vol.38 No.18

计 算 机 工 程
Computer Engineering

2012 年 9 月
September 2012

·开发研究与设计技术· 文章编号：1000—3428(2012)18—0262—03 文献标识码：A 中图分类号：TP301.6

一种 Cholesky 分解重叠算法

张德好，刘青昆

(辽宁师范大学计算机与信息技术学院，辽宁 大连 116086)

摘 要：在图形处理单元(GPU)平台的计算中，GPU 设备存储器和内存容量相差较大，待处理数据通常无法一次性从内存拷贝至显存中进行运算。为此，提出一种 Cholesky 分解重叠算法。采用预存取技术，拷贝数据和计算重叠，降低设备的等待时间，将设备存储器划分为 2 个缓冲区，轮流存放本次运算数据和下次待运算数据，在设备运算过程中完成设备存储器和内存之间的数据交换。实验结果表明，该算法可以有效提高运算效率。

关键词：图形处理单元；预存取；重叠算法；通用计算；Cholesky 分解；集群系统

A Cholesky Decomposition Overlapped Algorithm

ZHANG De-hao, LIU Qing-kun

(School of Computer and Information Technique, Liaoning Normal University, Dalian 116086, China)

【Abstract】In the computation of Graphics Processing Unit(GPU) platform, GPU equipment storage and memory capacity is different. Processed data usually cannot finish operation from memory copy to the video memory in one-time. In order to solve this problem, this paper proposes a Cholesky decomposition overlapped algorithm. By dividing the device storage into two buffers, current data and next data for calculation are stored in turn, data swap between device storage and memory takes place in the process of computation. Experimental results show that the algorithm can increase the system efficiency.

【Key words】Graphics Processing Unit(GPU); prefetching; overlapped algorithm; general purpose computation; Cholesky decomposition; cluster system

DOI: 10.3969/j.issn.1000-3428.2012.18.071

1 概述

正定矩阵的 Cholesky 分解是线性代数计算领域重要的基础运算，是求解系数为对称正定矩阵的大型线性方程组的主要方法。超大规模正定矩阵 Cholesky 分解，在各种工程计算和科学研究计算中有广泛的应用。

随着图形处理单元(Graphics Processing Unit, GPU)在性能上的飞速发展，可编程性不断增强，使 GPU 可以更多更便捷地应用于通用计算领域，以提高算法和系统的效率[1]。

统一计算设备架构(Compute Unified Device Architecture, CUDA)是 NV 公司推出的基于其 GPU 处理器的通用计算架构。能利用现代 GPU 高度并行、多线程，高速带宽的优势，完成各种并行程度高、计算量大的通用计算模型，可应用到图像处理、分子动力学、市场分析、天体物理学、机器学习等各领域。

使用 GPU 和多点接口(Multi Point Interface, MPI)的混合平台，可以以较低的成本，构建出性能出色的并行计算平台，并且其更易于更新换代。我国在很多领域运算能力不足，将合适的算法在 GPU 平台上实现、优化具有现实和长远的意义。

由于 PC 机的主内存容量和其 GPU 存储器容量存在巨大的差距，现代 PC 基本支持 64 GB 以上的主内存，GPU 的存储器容量则基本处于 2 GB 以下，待处理数据通常无法一次性从内存拷贝至显存中进行运算。

通常程序先将一部分数据拷贝至设备显存，设备运算后，再将结果拷贝回内存，进行下一部分的计算、拷贝和运算交替进行，在拷贝时设备处于等待状态。通过预存取技术，可以将数据的存取和设备运算同时进行，减少设备等待的时间，提高运算效率。因此，在 CUDA 平台上进行重叠算法的研究是非常有必要的。本文提出一种 Cholesky 分解重叠算法。在 GPU 设备存储器划分出 2 个相对独立的工作区，使原本需要交替进行的数据存取和数据计算工作可以在 2 个工作区之间并行重叠进行。

2 算法实现

2.1 Cholesky 分解

N 阶正定矩阵 $\boldsymbol{A}$，可以分解成 $\boldsymbol{A}=\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ 的形式，$\boldsymbol{L}$ 是一个下三角矩阵，为避免对矩阵 $\boldsymbol{A}$ 对角线元素的开方运算，通过对 $\boldsymbol{L}$ 和 $\boldsymbol{L}^{\mathrm{T}}$ 矩阵的对角线归一化处理，可以得到正定

作者简介：张德好(1981－)，男，硕士，主研方向：分布式系统，并行计算；刘青昆，副教授

收稿日期：2011-11-14 **修回日期：**2011-12-23 **E-mail：**haode_1981@msn.com

矩阵 $\boldsymbol{A}=\boldsymbol{LDL}^{\mathrm{T}}$，被称为改进后的 Cholesky 分解，公式如下：

$$\begin{cases} d_{00}=a_{00} \\ d_{ii}=a_{ii}-\sum_{k=0}^{i-1} l_{ik}^2 d_{kk} & i=1,2,\cdots,n-1 \\ l_{ij}=(a_{ij}-\sum_{k=0}^{j-1} l_{ik}l_{jk}d_{kk})/d_{jj} & j<i \end{cases}$$

其中，$\boldsymbol{L}=\begin{vmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{vmatrix}$；$\boldsymbol{D}=\begin{vmatrix} d_1 & & & \\ 0 & d_2 & & \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & d_n \end{vmatrix}$。

Cholesky 分解串行算法主要有 3 种，分别是：submatrix-cholesky 分解算法，column-cholesky 分解算法和 row 分解算法，主要区别是 k、j、i 分别处于循环的最外层[2]。

2.2　MPI 与 CUDA 混合构架

CUDA 是 NVIDIA 的通用计算图形处理器(General Purpose GPU, GPGPU)模型。现代 GPU 已经具有了高度的可程序化能力，可用于通用计算工作，而不是单纯的绘制[3]。GPU 上搭载了大量的结构相对简单的流处理器，能够同时并发大量线程，非常适用于处理单指令流多数据流(Single Instruction Multiple Data, SIMD)。

在并行性高的计算处理场合，其性能要比相同晶体管规模的 CPU 高很多。但是 GPU 本意是面向图形演算开发的，通常只能使用 GPU 特有的变量类型，直接使用 GPU 的图形 API 开发 GPU 通用计算程序的难度很高。CUDA 的编程模型语法使用的是应用更为广泛的 C 语言，它可以看作是 C 语言的一个极小扩展[4]，具有很好的适用性，更易于学习和实际应用。

多点接口是消息传递并行程序设计的标准之一，用以连接多个节点机实现程序并行运算。通过多点接口与统一计算设备架构相结合，CPU 主要负责任务调度、节点间的消息传递，GPU 负责数据并行计算，利用此种方式实现高性能的运算。

2.3　通信与计算重叠

GPU 与 CPU 组成的异构并行系统，运算流程如下：数据由主节点机分配到各节点机的内存，CPU 将内存中的数据拷贝至 GPU 存储器，然后 CPU 调用 GPU 进程进行计算。在一次任务过程中，可能需要进行多次上述操作。由于 I/O 操作与 GPU 运算是顺序执行的，当进行 I/O 操作时，GPU 处于空闲状态，当 GPU 进行数据计算时，传输总线处于空闲状态，两者均没有得到充分利用，因此造成资源浪费。

减少这种浪费的方法是将数据传输与计算重叠[5-7]。数据划分成合适的大小的数据块，在 GPU 存储器上建立 2 个缓冲区，轮流存放本次和下次待处理数据，GPU 对前一个数据进行处理的同时，传输下一个数据块至另一个缓冲区。这样就重叠了传输和计算时间，有效提高了资源利用效率，非重叠算法与重叠算法对比如图 1 所示。

(a)通信计算未重叠状态

(b)通信计算重叠状态

图 1　非重叠算法与重叠算法对比

2.4　Cholesky 的重叠算法

本文通过 CUDA 提供的异步并发执行功能实现设备计算与传输重叠。在 CUDA 模型中，CPU 调用 GPU 内核程序，GPU 处于计算状态后，控制权会重新返回到 CPU 线程，CPU 可以继续处理其他与 GPU 运算结果无关的任务。带有 Async 后缀的 CUDA API 存储器拷贝函数也具有上述特性，在 CPU 调用其发出拷贝命令后，系统总线通过 DMA 传输方式完成拷贝任务，无需 CPU 继续参与或等待，此特性需要主机内存上分配的分页锁定主机存储器来支持。CUDA 的 GPU 设备也支持数据传输和内核执行的并发(需要计算能力 1.1 以上的设备)，即 GPU 在处于计算状态的同时，也可以进行与正在进行的计算不相关的数据在设备存储器和主机存储器之间的传输，此功能也需要通过分页锁定主机存储器。

利用上述特性可以完成 Cholesky 的重叠算法。在设备存储器中开辟 2 块同大小的缓存区。将节点机中的待分解数据划分为合适大小的数据块，将一个数据块传输到 1 号设备存储器缓存区中，在传输完毕后，CPU 调用 GPU 对 1 号缓存中的数据进行分解，此时，GPU 处于忙状态，控制权返回到 CPU 线程，CPU 调用 Async 后缀的拷贝函数将下一个待分解数据块发送至 2 号缓存，系统总线 PCI-E 进入忙状态，此时数据传输和内核执行重叠。然后，CPU 开始等待，当数据传输与 GPU 执行的完毕后，CPU 调用 GPU 对 2 号缓存中的数据进行分解，并将 1 号缓存中计算完毕的数据拷贝至节点机内存中，然后将下一个待分解数据块发送 1 号缓存中，此时再次进入重叠状态。重复上面的流程，直到所有数据分解完毕，节点机内分解算法流程如图 2 所示。

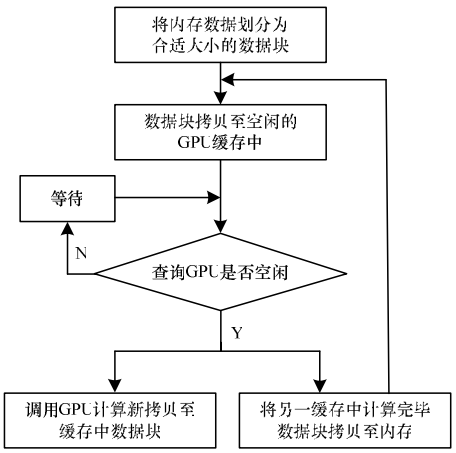

图 2　节点机内分解算法流程

节点机内分解算法描述如下：主节点将矩阵 A 以卷帘行存储[8]，分发给各个节点，行块大小为 m，节点数为 np。通过 CUDA API 函数在各节点机显存内分配 3 块空间，分别为 d_a、d_b1、d_b2，并在主机内存中分配一块页锁定内存 h_b，d_b1、d_b2、h_b 大小相同。

对于矩阵 A 的第 i 行，其存储在节点机 P_X 上，则：

(1)节点 P_X 进行如下运算：

1)将 i 行数据拷贝至 d_a。

2)调用内核函数对 d_a 内数据进行 Cholesky 分解。

3)将分解完数据拷贝回该节点机，覆盖内存原数据。

4)将 i 行数据发送到其他节点。

5)将 i 行所在的连续行块拷贝至 d_b1。

6)调用 GPU 对 d_b1 进行 Cholesky 分解，同时将其后的行块拷贝至 d_b2。

7)将 d_b1 分解完成数据拷贝回节点内存，并将下一个行块拷贝至 d_b1，同时调用 GPU 对 d_b2 内数据进行 Cholesky 分解。

8)重复步骤 6)、步骤 7)。

(2)其他节点机进行下面的运算：1)接收 P_X 发送的数据，并将其拷贝至 d_a。2)大于 i 的连续行块拷贝至 d_b1。3)调用 GPU 对 d_b1 进行 Cholesky 分解，同时将其后的行块拷贝至 d_b2。4)将 d_b1 分解完成数据拷贝回节点内存，并将下一个行块拷贝至 d_b1，同时调用 GPU 对 d_b2 内数据进行 Cholesky 分解。5)重复步骤 3)、步骤 4)。

3 实验结果与分析

本文实验环境采用由 5 台通用 PC 构建的集群系统。其配置为主频 2.4 GHz 的 Intel Xeon CPU，4 GB DDR3 内存，以及 GeForce GT 260 显卡，主机与显卡直接通过 PCI-E2.0X16 总线连接。集群采用 Linux RH9 系统，安装 CUDA Toolkit2.1 与 MPI 并行编译环境。

本文采用的 3 组不同规模的正定矩阵 A，测试其在不同算法下的性能。表 1 是矩阵 A 的阶数 N 分别为 1 793、2 478、9 054 时，在单纯 MPI 环境、MPI+CUDA 混合并行环境、MPI+CUDA 混合并行环境+重叠算法 3 种不同情况下的 Cholesky 分解执行时间对比。

表 1 不同情况下 Cholesky 分解执行时间 s

情况	N=1 793			N=2 478			N=9 054		
	1 个节点	2 个节点	4 个节点	1 个节点	2 个节点	4 个节点	1 个节点	2 个节点	4 个节点
MPI	12.414 0	6.293 0	3.202 0	24.478	12.858	6.914	1 132.22	568.13	290.47
MPI+CUDA	9.128 5	4.614 6	2.366 7	17.867	9.041	4.679	862.95	434.15	221.36
MPI+CUDA 重叠	8.124 3	4.113 3	2.108 2	15.866	8.050	4.146	759.36	380.78	195.07

由表 1 可知，GPU 对算法有明显的加速效果。同样在 CUDA+MPI 混合平台下，重叠 Cholesky 分解重叠算法的性能比顺序执行算法有较大提升，重叠算法在执行时间相比于顺序算法减少了 10%以上。

2 种算法的分解执行时间对比如图 3 所示。

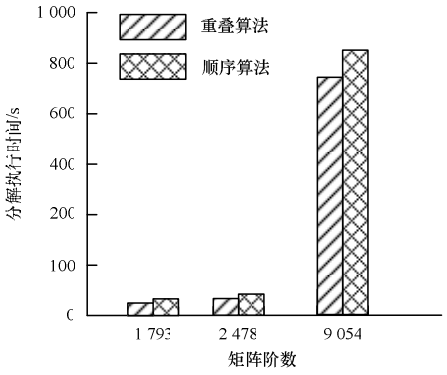

图 3 2 种算法的分解执行时间对比

由图 3 可知，由于重叠算法实现了与 GPU 的通信以及与 GPU 运算的重叠，因此减小了因为通信和计算相互等待而造成的额外开销，避免了 GPU 由于等待通信而造成的计算资源浪费，有效地提高了系统的利用率。

4 结束语

本文提出一种 Cholesky 分解重叠算法。实验结果表明，该算法使通信与计算时间重叠，提高了程序的整体性能。因为 GPU 设备存储器和 PC 主内存之间的容量不匹配性，在进行大规模数据计算时，通常将数据划分为的数据块分批进行处理，采用重叠算法能更好地提高计算效率。本文采用的是固定大小的数据划分，今后可以按照实际情况划分数据，进一步缩小系统空闲等待时间。

参考文献

[1] 肖 江, 胡柯良, 邓元勇. 基于 CUDA 的矩阵乘法和 FFT 性能测试[J]. 计算机工程, 2009, 35(10): 7-10.

[2] 邬贵明, 窦 勇, 王 淼. Cholesky 分解细粒度并行算法[J]. 计算机工程与科学, 2010, 32(9): 102-106.

[3] 吴恩华, 柳有权. 基于图形处理器(GPU)的通用计算[J]. 计算机辅助设计与图形学学报, 2004, 16(5): 601-612.

[4] 田 力. CUDA 在高性能计算中的应用[D]. 杭州: 浙江大学, 2008.

[5] 张 保, 曹海军, 董小社, 等. 面向图形处理器重叠通信与计算的数据划分办法[J]. 西安交通大学学报, 2011, 45(4): 1-6.

[6] Yang Yang, Raartkv C. Multi-round Algorithms for Scheduling Divisible Loads[J]. IEEE Transactions on Parallel and Distributed Systems, 2005, 16(11): 1092-1102.

[7] Shet G, Sadayappanp A, Bernholdt E D, et al. A Framework for Characterizing Overlap of Communication and Computation in Parallel Applications[J]. Cluster Computing, 2008, 11(1): 75-90.

[8] 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.

编辑 刘 冰