# 基于GPU的反卷积算法并行优化

隽鹏辉[1],贾婷婷[2],窦爱萍[1],刘金学[1]

(1. 中航工业西安航空计算技术研究所,陕西 西安 710068;2. 西安科技大学 高新学院,陕西 西安 710109)

**摘 要**:反卷积是图像去模糊的基本算法,针对传统反卷积算法在图像去模糊处理中实时性较弱问题,提出基于众核GPU的Iterative Deconvolve 3d反卷积算法的并行优化实现。所提算法将原算法中的核心运算放在GPU上并行实现,利用CPU和GPU协同工作模式,CPU负责串行任务GPU负责并行任务。实验表明:与传统的算法相比,在不影响图片处理效果的前提下,计算速度比CPU上的实现速度提高了近11倍,并具有良好的可扩展性。

**关键词**:并行;反卷积;GPU;CUDA

**中图分类号**:TP332;TP399.4 **文献标识码**:A **文章编号**:1671-654X(2016)06-0076-04

## Optimization of Image Deconvolution on GPU

JUAN Peng-hui[1], JIA Ting-ting[2], DOU Ai-ping[1], LIU Jin-xue[1]

(1. *Xi'an Aeronautics Computing Technique Research Institute, AVIC, Xi'an* 710068, *China*
2. *Gaoxin College, Xi'an University of Science and Technology, Xi'an* 710109, *China*)

**Abstract**: Deconvolution is the basic algorithm of Image Deblurring. In order to deal with the problem of weak real time, the implementation of Iterative Deconvolve 3d based on the GPU is proposed that has implemented the core operation on GPU. In use of the collaborative work mode of GPU and CPU, CPU is responsible for the serial work and GPU is responsible for the parallel work. Compared with the traditional algorithm, the experiment shows that: under the premise of not affecting the image processing results, the calculation speed of this algorithm on GPU is nearly 11 times faster than that of CPU, and has a good expansibility.

**Key words**: parallel; deconvolution; GPU; CUDA

## 引言

随着计算技术和集成电路技术的发展,图形硬件的更新速度越来越快。1999年NVIDIA发布GeForce256绘图处理芯片时,首先提出GPU概念,其多流水结构、向量处理特性以及32位IEEE标准浮点精度的实现,使得它对于计算密集型的科学应用有非常大的吸引力,越来越成为通用计算的一个有效的并行平台。近几年来很多科学计算都已经被移植到GPU这个计算平台,如精确天气预报、卫星图像处理和核爆炸的模拟等。

由于光的衍射作用,传统的荧光显微镜由于焦平面外信息的干扰造成拍摄图像模糊。为了提高图像的分辨率,一方面改善荧光显微镜的成像技术,典型的代表是激光共聚焦以及CCD技术的使用;另一方面,通过计算机进行数字图像处理和分析。图像反卷积算法是图像去模糊的基本算法,使得光学显微镜发挥了很大的作用。

Iterative Deconvolve 3d算法是非负的迭代型图像处理算法,由Bob Dougherty开发作为图像处理软件ImageJ的插件[1-2],能够对2D和3D图像进行反卷积处理,由于该算法基于JAVA实现,对小数据速度尚可;随着显微镜成像技术的快速发展,需要处理的图像数据越来越巨大,对日益增长的图像数据进行反卷积处理已不现实,为了应对这种需求,本文在GPU众核处理器上实现了IterativeDeconvolve 3d并行版本。

## 1 并行优化的实现

### 1.1 CPU多核的优化实现

为了对该算法进行并行优化,首先使用VisualVM工具进行热点函数分析,图1是分析得到的程序调用

**收稿日期**:2016-10-17 **修订日期**:2016-11-15

**基金项目**:国家自然科学基金项目资助(31327901)

**作者简介**:隽鹏辉(1989-),男,陕西宝鸡人,助理工程师,硕士研究生,主要研究方向为计算机系统结构。

关系及热点函数占用总体执行时间的比重。

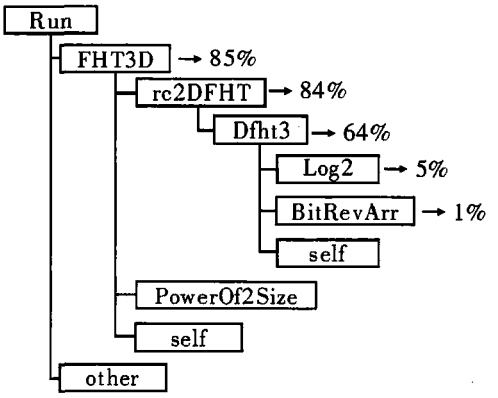


**图 1　函数调用关系及函数执行时间比重**

对 FHT3D 代码分析知,对一个三维图像,rc2DFHT 函数对每一帧图像的处理互不相关,因此,可以对 Iterative Deconvolve 3d 进行多核的并行优化。在 CPU 上进行程序的多核优化有多种方法可以使用,一般分为两类:线程级并行(OpenMP)[3-4]、进程级并行(MPI)。本文使用 OpenMP 进行线程级的优化。

基于 CPU 的程序使用 OpenMP 进行并行优化的性能受多方面影响,如线程数的设置、线程的调度模式等[5]。线程数的多少影响程序的并行度,通过 num_threads()或环境变量 OMP_NUM_THREADS 进行线程数的调整;线程的调度模式则通过 schedule 子句调整,在并行实现中尝试了 static,dynamic,guided 3 种程序调度模式。

### 1.2　GPU 众核的优化实现

#### 1.2.1　基本的 GPU 优化

由于 GPU 集成了更高的计算单元,具有更高的浮点计算峰值,Iterative Deconvolve 3d 是计算密集型算法,适合于在 GPU 众核进行并行优化。图 2 是 GPU 优化实现的主要流程图。

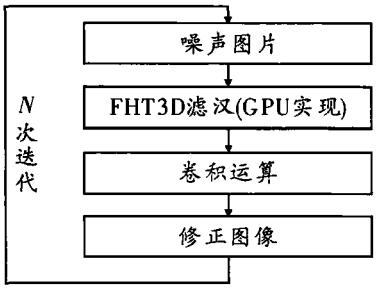


**图 2　GPU 优化实现的主要流程图**

由于整个反卷积算法包含多个分支,根据中间数据的状态和迭代次数来决定是否终止迭代,不适合将全部的迭代运算放在 GPU 上实现。FHT3D 是 Iterative Deconvolve 3d 热点函数,因此将其放在 GPU 上实现。对于 FHT3D,dfht3 函数是程序的主要计算,它每次对数组 $x[w]$ 或者 $x[h]$ 进行更新:

更新的数组为:$x[w]$ 或者 $x[h]$,更新次数为:$\log2(w)-1$ 次

更新规则是:

分别将数组 $x$ 划分为以 4,8,16,…,$2^{(\log2(w)-1)}$ 为小组的单元。

每次更新分别以划分好的小组为单位进行数值计算。

图 3 是将数组划分为 8 个元素为一个小组进行运算的示意图。

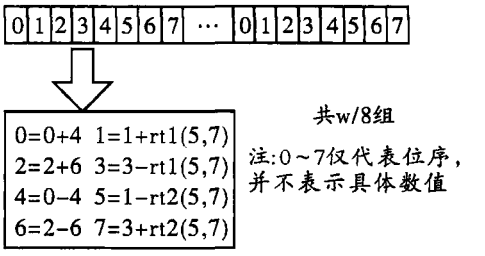


**图 3　dfht3 函数分析图**

在 FHT3D 函数,首先对一个三维的图像沿 $Z$ 轴方向的每一帧图像执行 rc2DFHT;在 rc2DFHT 函数中,对一个二维图像的每一行调用 dfht3 函数,对转置后的图像进行 dfht3 函数变换。本文将 rc2DFHT 改写为在 GPU 上执行的 kernel 函数。

中 dfht3_kernel 是对 dfht3 函数的 GPU 优化,在该 kernel 函数实现中,每个线程块对一帧图片进行处理,使用 *tpb* 个线程对 dfht3 进行并行处理。实现了 block 块的并行和 block 块内的线程并行两级并行,由于 dfht3 函数对数组 $x[w]$ 做 $\log W-1$ 次更新,不同的更新数据相互有依赖,因此每次数组更新都需要对数组进行同步。transpose_kernel 函数是矩阵的转置在 GPU 上的实现,对于矩阵的转置本文使用图 4 中算法实现。

#### 1.2.2　GPU 存储管理优化

CUDA 编程模型将 GPU 的内存划分为:全局存储器、纹理存储器、共享存储器和寄存器。它们的大小和速度各不一致,为了让程序有很好的性能,合理使用各个存储器对程序的性能至关重要[6]。

在 rc2DFHT kernel 函数中,图像数据需要被 Grid 网络中的所有线程访问,将其放置在全局内存中;共享存储器则用来存放图像数据中的每一行,每个线程块计算其中的一行;由于三维图片的每一帧图片都具有相同的高斯核,并且高斯核的数据能够放置在常量内存中,所以,将高斯核存放在常量存储器上加速程序的快速访问;将 dfht3 kernel 函数中的中间变量存放在寄存器中进行优化。

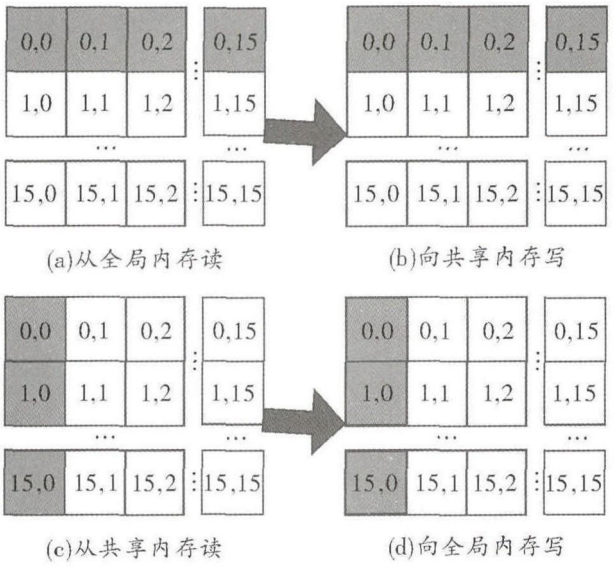


**图4　矩阵转置在GPU上的实现**

#### 1.2.3　数据传输优化

本节将实现CPU和GPU的异步传输,重叠计算与数据传输[7]。在Iterative Deconvolve 3d算法的实现中,由于该算法包含多个分支判断,不适合于将整个迭代计算放在GPU上实现,因此,在每次计算FHT3D时,计算前将数据从CPU传输到GPU,计算完将结果数据从GPU拷贝回CPU,在每次数据传输中,CPU和GPU的计算资源处于闲置状态,为了充分利用计算资源,重叠图片数据的传输和rc2DFHT计算。

## 2　实验结果

### 2.1　实验环境和测试数据

本文实验平台采用曙光星云高性能计算机部分计算节点。曙光高性能服务器计算节点搭配了2个Intel Xeon E5-2620六核的CPU和4个Tesla K40 GPU。软件环境为CUDA 6.5,GCC4.47。

测试数据来源于MRI Stack和北京大学分子医学研究所陈良怡老师提供的Light-sheet拍摄到的小鼠胚胎发育数据截取成指定大小的时间序列数据。

### 2.2　并行优化正确性验证

首先选择的测试数据为MRI Stack,将该数据使用ImageJ插件Convolve 3D进行卷积模糊处理,再分别使用原JAVA、CPU多核和GPU众核版本的Iterative Deconvolve 3d分别进行去模糊处理,最后比较它们最终反卷积后的图片,从而验证算法的正确性。

对模糊后的图片分别使用JAVA、CPU多核、GPU众核的程序进行反卷积去模糊处理结果图片见图5,从肉眼看经过CPU多核和GPU众核的优化算法与原JAVA的处理效果一致;通过比对3幅图片划线处的像素值也完全一致。经过对多个数据的测试,验证了基于GPU并行优化的正确性。

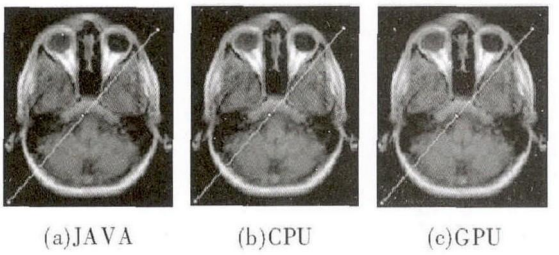


**图5　Iterative Deconvolve 3d优化结果图片验证**

### 2.3　性能测试

基准程序是改写成C++的串行版本,对其进行GPU众核的并行,并实现了数据传输和核函数的计算重叠与非重叠两个版本,下边使用它对三维图像进行反卷积,最大迭代次数为20次,测试结果见下表。

**Iterative Deconvolve 3d GPU众核优化结果表**

| 图片规模 | 收敛次数 | JAVA/s | C++/s | | CUDA/s | | 加速比 |
|---|---|---|---|---|---|---|---|
| | | | 1线程 | 6线程 | sync | async | 1线程/async |
| 256×256×32 | 10 | 80.6 | 78.5 | 30.0 | 9.5 | 10.0 | 7.9 |
| 256×256×64 | 8 | 141.4 | 130.4 | 52.1 | 14.8 | 15.2 | 8.6 |
| 512×512×32 | 11 | 473.9 | 382.2 | 147.7 | 38.6 | 38.1 | 10.0 |
| 512×512×64 | 12 | 1 060.6 | 843.2 | 340.2 | 82.1 | 80.2 | 10.5 |
| 1024×1024×16 | 4 | 679.9 | 353.3 | 139.7 | 33.9 | 32.8 | 10.8 |

从上表可以看出,GPU众核优化相对于C++串行程序的加速比可以达到11倍,性能明显提升。随着数据量的增大,加速比增大,具有良好的可扩展性。

在GPU上的众核优化中,数据传输和计算的重叠优化前后执行时间几乎没有变化,这是由于数据传输的时间和核函数的执行时间差异大而导致收益小。

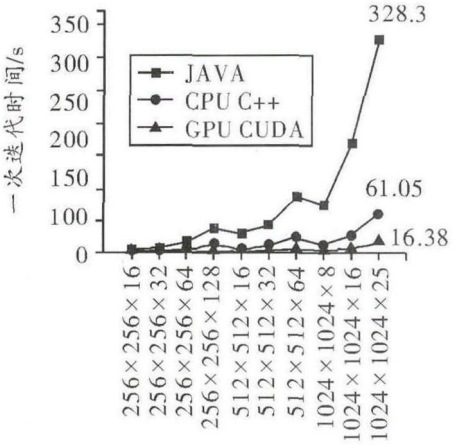


**图6　Iterative Deconvolve 3d在不同平台及编程环境下的最佳性能对比**

采用原 JAVA 程序、CPU 多核优化和 GPU 众核优化的程序对不同大小的图片进行反卷积,指定最大的迭代次数为 20 次,由于收敛次数的不一致,图 6 将对比不同平台下的一次迭代时间。

从图 6 中可以看到,随着数据量的增加,原 JAVA、CPU 多核和 GPU 众核程序的执行时间都变长,GPU 众核与 CPU 多核和 JAVA 程序的加速对比越来越明显,使用 GPU 进行 Iterative Deconvolve 3d 优化有明显的优势。

## 3　结束语

反卷积在生物图像的去模糊、重构中起到了非常重要的作用,但是由于算法的计算密集型特点,导致图像处理的时间过长,限制了其使用。本文对传统的经典反卷积算法 Iterative Deconvolve 3d 进行了 CPU 多核和 GPU 众核并行优化实现。我们的并行优化实现提高了生物专家对图像的分析效率,在实际工程中已被广泛使用;同时本文所使用的并行优化技巧,给基于众核的其他应用开发提供参考。

**参考文献:**

[1] NIH. ImageJ User Guide[EB/OL]. [2013 - 07 - 11]. http://imagej. nih. gov/ij/docs/guide/146. html. American: National Institutes of Health,2013.

[2] Bob Dougherty. Iterative Deconvolution 3D[EB/OL]. [2005 - 05 - 01]. http://imagej. net/Iterative_Deconvolve_3D. American: ImageJ.

[3] BarBara Chapman, Gabriele Jost, Ruud Van Der Pas. Using OpenMP[M]. MIT: The MIT Press,2007.

[4] Blaise Barney. OpenMP[EB/OL]. http://[2013 - 09 - 08]. computing. llnl. gov/tutorials/openMP/. American: Lawrence Livermore National Laboratory.

[5] Ryoo S, Rodrigues C I, Stone S S. Program optimization Space Pruning for a Multithreaded Gpu in CGO[C]//American: Proc. of the Int'l Symp. on Code Generation & Optimization, 2008.

[6] Nvidia. Cuda C Best Practices Guide[M]. American: Nvidia, 2012.

[7] Jason Sanders, Edward Kandrot . GPU 高性能编程 CUDA 实战[M]. 北京:机械工业出版社,2011.

(上接第 75 页)

供参考,与此同时,标准仅从过程层面对软件适航审定要求的目标进行了考虑,并未对具体的技术实施提供明确的指南(DO - 178C 的附件 DO - 330, DO331, DO - 332, DO - 333 对工具鉴定、模型开发、面向对象及形式化技术的考虑仅作为技术应用的推荐考虑)。因此研制单位在研究分析理解标准的要义后,结合工程经验及现有标准体系(如 GJB5000A)进行融合应用实施将是重要的研究方向。

**参考文献:**

[1] RTCA. DO - 178B/ED - 12B, Software Considerations in Airborne Systems and Equipment Certification[S]. Washington: RTCA,1992.

[2] RTCA. DO - 178C, Software Considerations in Airborne SystemsEquipment Certification[S]. Washington: RTCA,2012.

[3] Society of Automation Engineering. SAE ARP 4754A, Guidelinesfor Development of Civil Aircraft and Systems[S]. USA: Society of Automation Engineering S - 18 Committee,2010: 37 - 45.

[4] 郑军. 机载软件适航认证标准的进展及展望[J]. 计算机工程与设计,2012,33(1):204 - 208.

[5] 宋婷婷,刘志刚,柳俊成. DO - 178B 与 GJB5000A 对比分析研究[J]. 科技创新导报,2014,16(4):180 - 190.