

NVIDIA Tegra K1 异构计算平台访存优化研究

梁 军^{1a}, 李 威^{1b}, 肖 琳², 徐歆恺^{1a}

(1. 北京联合大学 a. 电子信息实验实训基地; b. 自动化学院, 北京 100101;
2. 北京联合大学 应用科技学院, 北京 102200)

摘 要: 在异构计算平台的移植和优化过程中, 数字图像处理算法的访存性能已成为制约系统性能的主要因素。为此, 结合 NVIDIA Tegra K1 硬件架构特征和具体算法特性, 从合并与向量化访存优化、全局访存 bank 和 channel 冲突消除等方面, 对矩阵转置算法和拉普拉斯滤波算法在 NVIDIA Tegra K1 异构计算平台上的实现和访存性能优化进行研究。实验结果表明, 采用优化方法后的矩阵转置算法和拉普拉斯滤波算法在 NVIDIA Tegra K1 异构计算平台上取得了较大的访存性能提升, 并且具有较好的实时性。

关键词: GPU 优化; 访存带宽; 数据本地化; 向量化; 合并访问; 拉普拉斯滤波算法

中文引用格式: 梁 军, 李 威, 肖 琳, 等. NVIDIA Tegra K1 异构计算平台访存优化研究[J]. 计算机工程, 2016, 42(12): 44-49.

英文引用格式: Liang Jun, Li Wei, Xiao Lin, et al. Research on Memory Access Optimization of NVIDIA Tegra K1 Heterogeneous Computing Platform[J]. Computer Engineering, 2016, 42(12): 44-49.

Research on Memory Access Optimization of NVIDIA Tegra K1 Heterogeneous Computing Platform

LIANG Jun^{1a}, LI Wei^{1b}, XIAO Lin², XU Xinkai^{1a}

(1a. Training Center of Electronic Information; 1b. College of Automation, Beijing Union University, Beijing 100101, China;
2. College of Applied Science and Technology, Beijing Union University, Beijing 102200, China)

【Abstract】 During the transplantation and optimization of the heterogeneous computing platform, memory access performance of digital image data algorithm becomes the main factor. In order to solve the problem, this paper combines with the NVIDIA Tegra K1 hardware architecture's characteristics and the specific algorithm's characteristics, researches the implementation and memory access performance optimization of matrix transpose and Laplace filtering algorithms on the NVIDIA Tegra K1 heterogeneous computing platform from memory access optimization of consolidation and vectorization, eliminating global memory access's bank and channel conflict etc. Experimental result shows that the performance of matrix transpose and Laplace filtering algorithms on the NVIDIA Tegra K1 heterogeneous computing platform has an obvious improvement, and has good real-time performance.

【Key words】 GPU optimization; memory access bandwidth; data localization; vectorization; coalesced access; Laplace filtering algorithm

DOI: 10.3969/j.issn.1000-3428.2016.12.009

0 概述

由于受到功耗和硬件等方面的限制, 因此单纯依靠提升芯片频率来提高处理器性能的方法已经变得越来越困难^[1]。在这种情况下, 多核和众核成为计算体系结构发展的趋势。随着可编程性的增强和处理器性能的提高, GPU 受到越来越多开发人员的青睐。

各芯片厂商都推出了自己的 GPU 架构, 如 NVIDIA 的 kepler^[2]、Maxwell 架构、AMD 的 Cypress、GCN^[3] 架构等。其中, NVIDIA 发布的嵌入式异构计算平台 Jetson Tegra K1, 围绕革命性的 NVIDIA Tegra K1 SOC 打造, 其 GPU 采用 NVIDIA Kepler 计算核心, 具有强大的计算能力, 能够快速开发和部署用于计算机视觉、机器人以及医学等领域的计算密集型算

基金项目: 国家自然科学基金重大研究计划项目(91420202); 北京市教育委员会科技计划面上项目(SQKM201411417010, KM201511417003)。

作者简介: 梁 军(1962—), 女, 副教授、硕士, 主研方向为异构并行计算、数字图像处理; 李 威, 学士; 肖 琳, 副教授; 徐歆恺, 讲师。

收稿日期: 2016-04-29 **修回日期:** 2016-06-30 **E-mail:** liangjun2236@sina.com

法和应用^[1]。

随着 NVIDIA Tegra K1 异构计算平台在智能驾驶和无人机等领域日益广泛的应用,数字图像处理向更高、更深层次发展,如何提高数字图像处理算法的性能已成为研究热点。图像处理算法在 NVIDIA Tegra K1 计算平台的移植和优化中,访存性能往往会成为系统性能提升的主要瓶颈。对此,本文从减少对访存带宽的依赖方面入手,研究针对该计算平台的访存性能优化方法。在此基础上,实现矩阵转置算法和拉普拉斯图像增强算法在 NVIDIA Tegra K1 计算平台上的性能优化。矩阵转置算法是数值计算中重要的操作之一,它在数字图像处理、信号处理等领域都具有广泛应用。该算法的数据之间相互独立,不存在数据依赖,算法瓶颈主要为不连续访存导致访存效率低下,通过克服该算法对访存带宽的依赖,实现性能提升^[4]。拉普拉斯滤波算法在数字图像处理领域中有着广泛应用。该算法没有数据依赖,但访存密集,所以,提升算法性能的关键在于提高访存效率,减少访存延迟。通过向量化可克服数据访存密集的性能瓶颈,实现性能提升。

1 相关工作

国内外已经对 GPU 访存优化做了大量的工作。文献[5]提出一个简单的允许程序员优化内存映射的 API 来提高访存在异构计算平台上的效率。文献[6]介绍了一系列编译器指令,通过转换全局数据布局实现更好的向量化。文献[7]提出利用动态编译器来编译的方案,并介绍了数据并行处理内核 SIMD 的功能。文献[8]针对 SIMD 代码的向量化访存,为减少访存数量和尽可能对齐访存,将数据重组,证明向量化访存的可行性和有效性。文献[9]基于 OpenCL 平台,验证了向量化在 NVIDIA 和 AMD 2 个 GPU 平台上都取得了良好的加速效果。

由于 GPU 程序的优化本质是实现从算法特性到底层硬件架构特征的有效映射,要求在优化过程中,不仅需要充分了解底层硬件架构,而且还要充分理解算法特性^[10],因此相同算法在不同硬件架构上的优化性能不同。研究在 Tegra K1 上某些算法的 GPU 优化,对开发异构计算平台的图像程序以及科学计算程序都有着重要的意义。

2 NVIDIA Tegra K1 简介及优化方法

2.1 NVIDIA Tegra K1 介绍

目前对图像处理的实时性要求不断提高,如何提高图像处理速度已经成为该领域技术发展的一个瓶颈。并行图像处理技术应运而生,为图像处理效率和性能的

提升提供了借鉴^[11]。并行图像处理技术的实现原理与传统的串行处理不同,依赖于不同的体系结构或硬件环境,不同的并行图像处理方法以及不同的应用环境。

由 NVIDIA 公司新推出的结合 NVIDIA Kepler 架构和 Cortex A15 架构的四核 ARM CPU 的移动超级计算平台——Jetson Tegra K1,不仅拥有强大的数据处理能力,而且大幅降低了功耗。Tegra K1 上搭载的 192 个 CUDA 核心的嵌入式 GPU,更加有利于并行运算,同时支持 CUDA 编程高效地完成 GPU 并行程序设计。这些线程可以通过 CUDA 编程划分到一个 3 层网络结构中,即网格(grid)、线程块(block)和线程(thread)^[12]。块内所有线程必须存在于同一个处理器核心并且共享核心有限的存储器资源,所以,一个块内的线程数目有限。一个内核可被多个同样大小的线程块执行,总的线程等于每个块内的线程数乘以线程块数。线程块也可以被组织成一维、二维或三维的线程网格。每个执行内核的线程拥有一个独一无二的线程 ID,可以通过内置的 ThreadIdx 变量在内核中进行访问(块内唯一)。线程块必须独立执行,而且能够以任意顺序(串行或并行)执行。这种独立性要求线程块能以任意顺序在任意数目的核心上进行调度^[13]。每个线程作为最小的处理单位拥有私有存储寄存器和局部存储器。每个线程块都可以共同访问共享存储器(shared memory),同一个线程块可以访问共享存储器与私有寄存器,且都远快于访问全局存储器(global memory)。因此,在程序设计方面要针对不同应用在线程访问存储器上的性能进行优化。本文采用多种不同的 CUDA 并行程序设计的优化方法来充分利用嵌入式 GPU 的计算资源,达到良好的性能优化效果^[14]。

2.2 访存优化方法

2.2.1 全局存储器合并访问

合并访问是指连续的线程访问连续的数据。实际算法在访问数据时,往往会出现连续的线程访问不连续的数据地址。由于线程寻址而导致不必要的时间损耗,因此为了提高全局存储器的访问效率,通常通过 shared memory 解决这类问题。

2.2.2 共享存储器与 bank 冲突

shared memory 是 GPU 的片内缓存存储器。它是一个可以被同一 block(由多个线程组成的线程块)中的所有线程访问的可读存储器。共享内存自身不存在合并访问的问题,通常被用于处理非合并访问的问题。

bank 冲突是指一个线程束(warp)内的多个线程同时访问同一个存储体(bank)时,将导致线程访问 shared memory 的串行化,即当一个线程访问 shared

memory 时, warp 中的其他线程将会排队等候,直到 bank 请求被满足。因此,应该尽量避免 bank 冲突。

2.2.3 解决 channel 冲突的方法

channel 冲突与 bank 冲突类似,是由 warp 发起的访存请求所造成的全局内存通道冲突,即 grip 内开启的线程没有均衡的分配到全局内存的通道,导致分配的通道数量不多,致使其他的通道处于空闲状态。线程同时对少量的通道进行访问会造成在单个通道中串行化的问题。

2.2.4 向量化

向量化主要指一条向量指令操作向量寄存器中的多个元素。向量化访存的主要作用有:1)充分利用访存带宽利用率;2)可以进行向量计算,开发指令级并行(Instruction Level Parallelism, ILP)。

3 矩阵转置与 Laplacian 变换的并行优化

3.1 矩阵转置算法并行性分析

矩阵转置过程中数据与数据之间不存在依赖关系,通过开启大量的线程处理不同的数据,实现数据的并行处理。因此,在最大线程范围内构建线程和原始矩阵数据的映射关系,使数据都得到相应的线程处理,即实现了目标矩阵转置的并行操作。

3.1.1 矩阵转置算法初步实现

通过矩阵转置串行算法可以确定矩阵转置并行算法的基本映射关系,即相同偏移量的线程处理相同偏移量的数据,映射关系如图 1 所示。

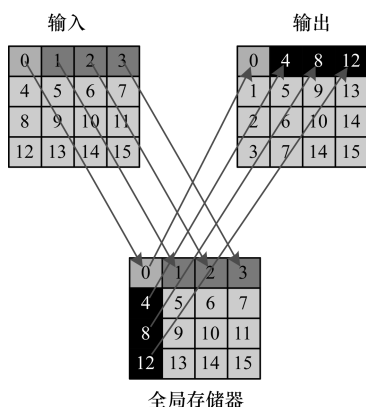


图 1 矩阵转置并行算法初步映射关系

3.1.2 全局存储器合并访问

合并访问是 GPU 算法优化的一种基本方法,其要求连续的线程访问连续的数据。在矩阵算法中,数据读入操作是连续的,相邻的线程读取相邻的数据。然而,对于数据的读出操作,相邻线程访问数据的内存跨度为矩阵一行所占的字节数,如此大的线程跨度一方面会增加每次访存请求所产生的访存事务,增加无效数据的传输^[1],从而降低访存带宽的利用率。

在图 2 中,相邻的线程访问相邻的地址空间。然而,线程读取数据传到 out_data 中并不符合合并访问的要求。由于共享内存不存在因访存不连续而导致访存性能下降的问题,因此对矩阵转置操作进行分块处理,将每个块所处理的数据加载到共享内存,再在共享内存中完成转置操作,然后写入全局内存中。

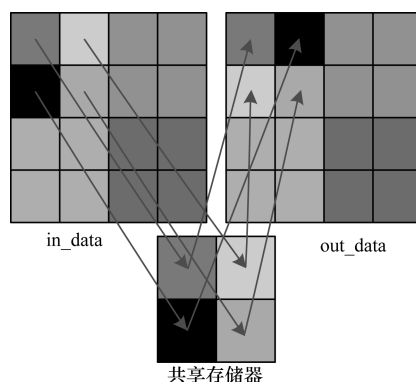


图 2 合并访问的映射关系

3.1.3 矩阵转置算法共享存储器与 bank 冲突

通过 3.1.2 节的合并访问,相较于基本实现的算法,优化算法性能得到提升,然而性能提升并不明显,本节将通过共享内存的访存方式对算法进一步优化分析,以提高性能。使用共享内存往往会遇到 bank 冲突,即指如果有 2 个或 2 个以上的线程访问的数据在同一个 bank 中,则不能在一次 bank 访问中满足访存单元的所有要求。

如图 3 所示,开启线程数目为 4×4 ,共享内存为 4 个 bank,相邻线程读取共享内存数据的内存跨度为 4 个 bank,这样就会产生 4 路 bank 冲突,限制了共享内存的访存性能。

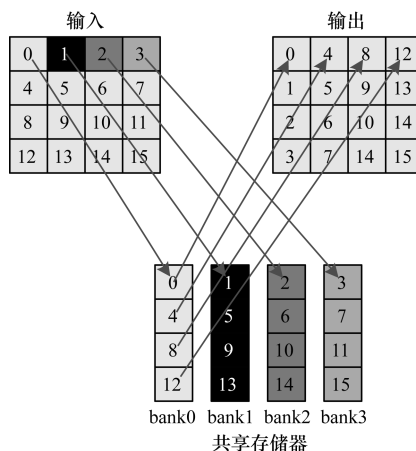


图 3 bank 冲突

由于 bank 数量受硬件限制,因此通常无法直接通过增加 bank 的数量来解决冲突问题。bank 冲突问题的一般解决方法是增加 padding(逻辑上增加一个 bank 空间,实际上是加入一列空白数据来消除 bank 冲突),即申请共享内存为 5×4 ,在每行后加一个 padding,如图 4 所示。

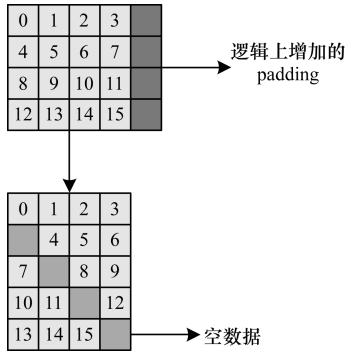


图 4 通过增加 padding 解决 bank 冲突的方法

3.1.4 channel 冲突

使用共享内存只能解决 block 内部的非连续访存问题,但是仍不能解决并行执行的 warp 发起的访存请求所造成的全局内存通道冲突。理论上,可以采用解决 bank 冲突的方法原理,通过对线程索引的映射转换来解决此问题,然而该方法需要大量的空间写入,增加了算法的复杂性。因此,通过使用 Diagonal 坐标转换,对每个 grid 处理的数据块进行重新排序,使相邻的数据块在计算时不相邻,如图 5 所示。

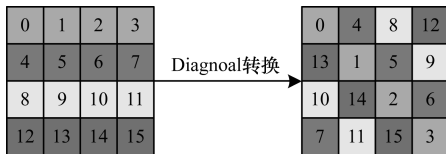


图 5 Diagonal 坐标转换

3.2 Laplace 算法

3.2.1 Laplace 算法介绍

Laplace 算法是基本图像处理算法,初始图片通过该算法可以增强图像中灰度突变区域的对比度,从而产生一副把图像中的浅灰色边线和突变点叠加到暗背景中的图像。基于 Laplace 的图像增强已成为图像锐化处理过程中的一个基本功能。Laplace 算法的关键在于 Laplace 算子。Laplace 算子是 n 维欧几里德空间中的一个二阶微分算子^[2]。Laplace 算子的定义为:

$$Laplace(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

如图 6 所示,该方程的实际运算是原始 3×3 的矩阵与 3×3 的模板矩阵的点乘,得出目标矩阵。

0	-1	0
-1	4	-1
0	-1	0

图 6 Laplace 算子模板矩阵

3.2.2 并行性分析

从 Laplace 算法运算过程来看,该算法主要通过

图像的像素矩阵与模板矩阵进行 3×3 的点乘运算,目标矩阵的每个点之间不存在数据依赖,完全可以并行执行,而且该算法属于访存密集型算法。对此,本文通过提高访存效率并合理使用 GPU 硬件资源,实现算法的优化。

3.2.3 Laplace 算法初步实现

在 Laplace 算法中,每个线程负责计算一个目标矩阵的元素,通过中心元素来确定坐标位置并读取 8 个边界元素。该算法容易实现,但是由于 global memory 访存效率不高以及相邻 2 个点的计算需要重复读取 6 个元素,导致访存片外的内存开销增大。为了提高算法的性能,还需通过了解算法特性和底层硬件资源特征,实现两者的有效映射。如图 7 所示,图 7(a)中的黑色部分为边界元素,白色为中心元素,图 7(b)对相邻 2 个中心元素中重复的 6 个数据进行计算。

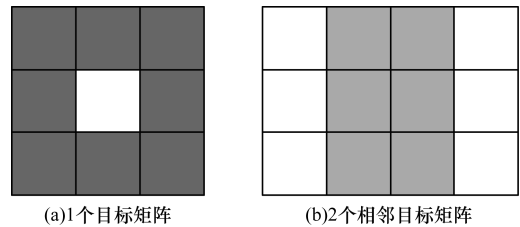


图 7 Laplace 算法分析

3.2.4 共享内存使用

将 Laplace 滤波的模板矩阵存储在常量内存中,通过将图像划分为多个图像块,使一个线程块加载一个图像块到共享内存中,线程块中的一个线程将共享内存相应位置处的图像像素与模板矩阵进行点乘处理^[15],即滤波处理,计算得出一个输出像素。由于输入共享内存中的图像块与模板矩阵进行计算时,会出现依赖不存在共享内存的图像像素的情况,因此需要在共享内存中多存入一圈图像边缘。本文通过将原始像素矩阵的 3 行像素读入共享内存中,再利用模板矩阵对读入数据进行滤波处理,通过向量化 1 个线程实现 4 个数据的处理。

数据本地化伪代码具体如下:

```

输入  原数据 src, 图像的宽 Width, 图像的长 Height
输出  计算处理后的数据 dst
x ← get_local_id(0)
y ← get_block_id(1)
offset ← x * 4 + (y + 1) * width
// 数据传入寄存器
If threadIdx.x < 127 then
    tid[0][threadIdx.x * 4 + (0 ~ 3)] = src[offset - width + (0 ~ 3)]
    tid[1][threadIdx.x * 4 + (0 ~ 3)] = src[offset + (0 ~ 3)]
    tid[2][threadIdx.x * 4 + (0 ~ 3)] = src[offset + width + (0 ~ 3)]
// 计算数据

```

```

for i = 1 to 3 i + + do
dst[ offset + i ] ← compute ( )
end for
end if

```

3.2.5 Laplace 算法向量化

GPU 的每个流多处理器 (SM) 有 32 768 个寄存器。一个 SM 可以看作是一个多线程的 CPU 核,在 Tegra K1 上只有一个 SM。虽然开普勒架构上的 SM 数量比早期的一些设备要少,但是开普勒架构设备上 SM 具有更多的流处理器 (SP),对于 Tegra K1 来说其拥有 192 个 CUDA core。寄存器是 NVIDIA GPU 存储器空间中访存性能最高的存储器,但是 GPU 上寄存器的数量有限。所以,向量化时应避免寄存器的过度使用,本文采用 1 个线程处理 4 个数据的方式。

数据向量化伪代码具体如下:

```

输入  原数据 src,图像的宽 Width,图像的长 Height
输出  计算处理后的数据 dst
x ← get_local_id(0)
y ← get_block_id(1)
//数据传入寄存器
temp_s0[6] = *src - width
temp_r[6] = *src
temp_c2[6] = *src + width
//计算数据
for i = 0 to 3 i + + do
temp_dst[i] ← compute ( )
end for
* (dst + 1) ← * temp_dst)
End if
Else
temp_dst[0 ~ 3] ← 0
* (dst) ← * (temp_dst)
End else

```

对于大多数程序,影响性能的因素主要来自设备获取输入数据和写回输出数据的 I/O 速度。片外访存带宽是 CPU kernel,特别是访存密集型算法的主要影响因素。通过最大限度地计算数据从片外转移到片上的寄存器中,实现数据共享,从而减少对访存带宽的依赖,是缓解这种限制因素的主要方法^[1]。由于 Laplace 算法是连续访存且是数据访存密集型算法,能提高访存效率,隐藏访存延迟,并且对提高程序性能有很大帮助,因此本文采用此算法来实现数据向量化,从而提升程序的性能。

4 性能评估

4.1 实验平台

NVIDIA Tegra K1 硬件平台主要性能参数如表 1 所示。

表 1 主要性能参数

参数	参数值
GPU 型号	NVIDIA TK1
GPU 时钟频率/MHz	852
CUDA core 个数	192
warp 规模	32

4.2 矩阵转置算法性能

经过 GPU 优化后,矩阵转置算法的性能得到了明显提升,除了矩阵数据规模为 512×512 以外,在其他矩阵数据规模上,消除 bank 冲突对算法性能提升影响较大。当数据规模为 2048×2048 时,会出现优化后算法效率反而降低的情况,其原因是 shared memory 中共有 32 个 bank,block 的线程规模为 16×16 ,所以,并没有很好地消除 bank 冲突。当 block 为 16×16 时,对于 global memory 的合并访问的优化比较明显,但是对 shared memory 的 bank 冲突的优化并不理想,甚至性能还有所降低。文献[4]在针对 K20X (Kepler) shared memory 的 bank 冲突的优化时,同样出现性能有所降低的情况(当数据规模为 2048×2048 时,block 为 32×32)。性能下降有很大部分的原因在于 NVIDIA 公司 Tegra K1 的 shared memory 的 cache 访存效率已经比较高,以至于 bank 冲突的延迟并不明显,而 bank 冲突的优化加入增加了一部分的计算量,最终导致性能略微下降。相对而言,block 为 32×32 时 shared memory 的 bank 冲突的优化比较明显。优化后的最高带宽为 7.11 GB/s。为比较不同情况下矩阵转置并行算法的优化效果,用平均数据计算出相对于 CPU 的加速比,如图 8 ~ 图 10 所示。

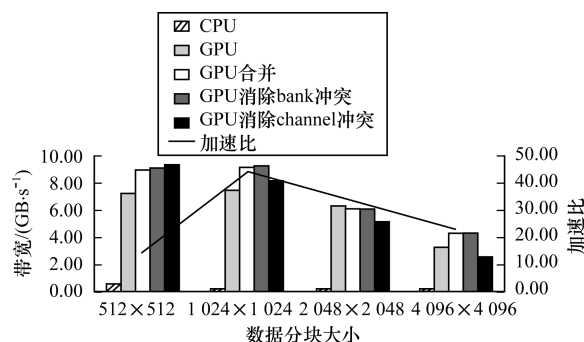


图 8 block 尺寸为 16×16 时的算法性能比较

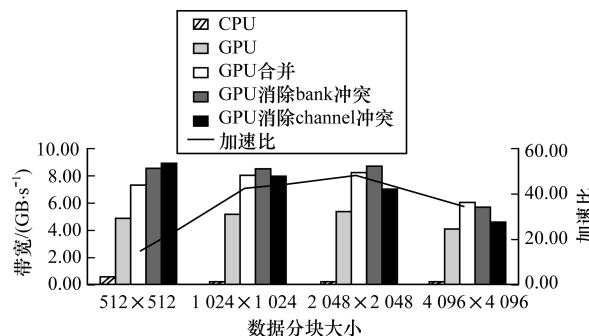
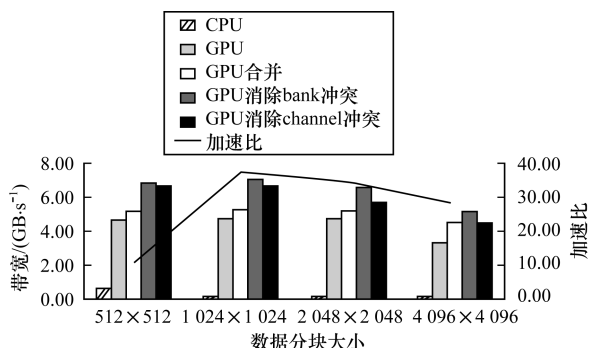


图 9 block 尺寸为 32×8 时的算法性能比较

图10 block 尺寸为 32×32 时的算法性能比较

可以看出,block 大小为 16×16 的最优加速效果优于 block 大小为 32×32 的加速效果,最高加速比达到 44 倍。这时的数据规模是 1024×1024 , block 大小为 16×16 。总体来讲,在数据规模为 1024×1024 , block 大小为 16×16 时,在该平台上的加速性能最优。也可以发现,解决全局内存通道冲突的问题并没有对算法的提升带来明显的优化,甚至会由于索引坐标的计算而导致计算数据量增大从而降低算法效率。

4.3 Laplacian 算法性能

Laplace 算法在 Tegra K1 异构计算平台下的性能优化是通过将数据存入寄存器中,缓解算法对访存的依赖。通过向量化开启少于数据量的线程,达到一个线程处理多个数据增加线程的运算时间,从而隐藏访存延迟。如图 11 所示,相继通过不同存储类型的向量化处理后,算法性能最终提高了 5.27 倍,由此可以得到以下结论:由于 Laplace 算法的自身特点,需要对相同的某一内存进行多次读写,而且算法的访存方式为不连续访存,因此通过数据本地化或者将数据存入寄存器中,有利于减少数据的重用以及对片外访存的依赖。

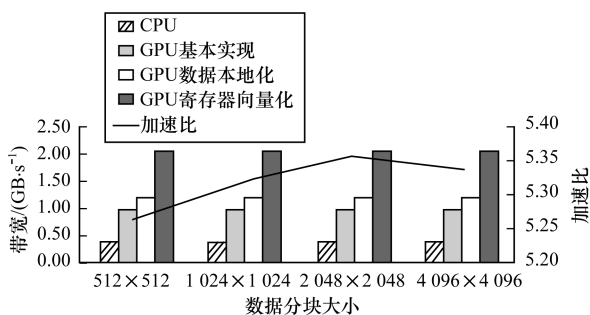


图11 对应不同像素图像的 Laplace 算法性能

5 结束语

访存密集型算法的性能瓶颈在于对访存带宽的依赖。本文给出图像数据处理应用程序中常用的矩阵转置算法和拉普拉斯算法在 NVIDIA Tegra K1 异构计算平台上的性能优化方法,主要从合并访存,向

量化访存,消除全局访存 channel 冲突,使用共享内存减少对访存带宽的依赖 4 个方面对算法性能进行优化。实验结果表明,优化后的算法不仅在整体性能上得到了明显提升,而且具有更好的实时性,从而提高其工程应用价值。下一步工作将结合硬件架构特征和算法特性对 CPU + GPU 协同工作进行深入优化,提高数字图像处理算法性能。

参考文献

- [1] 贾海鹏. 面向 GPU 计算平台的若干并行优化关键技术研究[D]. 青岛:中国海洋大学,2012.
- [2] AMD Corporation. Accelerated Parallel Processing Open CLTM[EB/OL]. (2011-01-15). <http://vdisk.weibo.com/s/tScaKBfIKP-A? sudaref = www.baidu.com>.
- [3] NVIDIA Corporation. Whitepaper NVIDIA GeForce GTX 680 [EB/OL]. (2012-05-24). http://www.drivsky.com/nvidia/GeForce_GTX_680.htm.
- [4] Ruetsch G, Micikevicius P. Optimizing Matrix Transpose in CUDA [EB/OL]. (2011-04-28). https://www.researchgate.net/publication/228376840_Optimizing_matrix_transpose_in_CUDA.
- [5] Che S, Sheaffer J W, Skadron K. Dymaxion: Optimizing Memory Access Patterns for Heterogeneous Systems[C]// Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. New York, USA: ACM Press, 2011: 1-11.
- [6] Xu S, Gregg D. Semi-automatic Composition of Data Layout Transformations for Loop Vectorization [M]// Hsu Ching-hsien, Shi Xuanhua, Salapura V. Network and Parallel Computing. Berlin, Germany: Springer, 2014: 485-496.
- [7] Kerr A, Diamos G, Yalamanchili S. Dynamic Compilation of Data-parallel Kernels for Vector Processors[C]// Proceedings of the 10th International Symposium on Code Generation and Optimization. New York, USA: ACM Press, 2012.
- [8] 徐金龙, 赵荣彩, 徐晓燕. SIMD 代码中的向量访存优化研究[J]. 计算机科学, 2015, 42(12): 18-22.
- [9] 贾海鹏, 张云泉, 龙国平, 等. 基于 OpenCL 的拉普拉斯图像增强算法优化研究[J]. 计算机科学, 2012, 39(5): 271-277.
- [10] 徐金龙, 赵荣彩, 刘鹏, 等. 程序向量化中非规则访存问题研究[J]. 计算机工程, 2015, 41(12): 86-90.
- [11] 张晶, 郭春学, 陈凯明. 基于负载均衡的 ZigBee 动态路由优化算法[J]. 计算机工程, 2016, 42(3): 138-142.
- [12] 杜晓刚, 党建武, 王阳萍. 基于 CUDA 的改进互信息并行计算方法[J]. 计算机工程, 2015, 41(12): 288-292, 298.
- [13] 刘景松. H. 265 自适应插值滤波器的研究与 CUDA 优化[D]. 北京: 北京邮电大学, 2011.
- [14] 吴鑫, 张建奇, 杨琛. Jetson TK1 平台实现快速红外图像背景预测算法[J]. 红外与激光工程, 2015, 44(9): 2615-2621.
- [15] 刘进锋. 几种 CUDA 加速高斯滤波算法的比较[J]. 计算机工程与应用, 2013, 49(23): 14-18.