一种基于OPENACC的GPU加速实现高斯模糊算法

曾文权[1],胡玉贵[1],何拥军[1],林 敏[2]
(1.广东科学技术职业学院 计算机工程技术学院,广东 珠海 519090;
2.珠海司迈科技有限公司,广东 珠海 519090)

摘 要:针对使用底层API进行GPU加速时存在的编码复杂以及效率低下等缺陷,文中试图利用基于中间层的OPENACC加速技术对传统的串行代码进行改写,从而达到改善开发效率,简化代码之目的。文中以传统的串行高斯模糊算法为处理对象,在其中添加OPENACC指令,提出基于OPENACC指令的GPU加速算法,并对算法流程进行了分析和说明。通过与原生CUDA和串行高斯的结果对比之后,发现随着处理像素数量的增加,串行高斯性能呈指数变化,而CUDA和OPENAC则呈线性变化。结果表明,该算法能在不改变原有非并行代码结构的基础上,通过增加高效的OPENACC指令即可获得与CUDA近似的图像处理质量和处理性能,且较CUDA具有更高的代码开发效率。

关键词:OPENACC;图形处理器;统一计算架构;高斯模糊
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2013)07-0147-04
doi:10.3969/j.issn.1673-629X.2013.07.038

Implementation of a Gaussian Blur Algorithm Based on GPU Accelerated by OPENACC

ZENG Wen-quan[1], HU Yu-gui[1], HE Yong-jun[1], LIN Min[2]
(1. College of Computer Engineering and Technology, Guangdong Institute of Science and Technology, Zhuhai 519090, China;
2. Zhuhai Simaike Co., Ltd, Zhuhai 519090, China)

Abstract: For GPU acceleration using the underlying API exists coding complexity and inefficiency defects, attempt to make use of the traditional serial code to rewrite based on the intermediate layer OPENACC acceleration technology to achieve the purpose of improved development efficiency and simplifying the code. In this paper, the traditional serial Gaussian blur algorithm for handling objects, in which add OPENACC instruction, based on OPENACC instruction GPU accelerated algorithm is proposed and the algorithm flow is analyzed. In contrast with the results of native CUDA and serial Gaussian, found that with the increase in the number of processing pixel, serial Gaussian exponential performance changes, while the CUDA and OPENAC changes linearly. The results show that the algorithm can not change on the basis of the original non-parallel code structure by increasing efficient OPENACC instruction can obtain the image processing quality and processing performance approximate to CUDA, and compared with CUDA has higher code development efficiency.

Key words: OPENACC; GPU; CUDA; Gaussian blur

0 引 言

随着计算硬件加速设备价格的日益下降,作为图像处理和绘制的主要设备,GPU因为拥有专门为图像处理而设计的并行体系结构和特有处理元件,已经在图形图像处理的相关应用领域得到了普及[1]。

然而,要在GPU硬件上实现加速需要通过对底层的API进行编程来实现,程序编写复杂、难度大,且容易形成高度依赖特定设备的代码,即针对不同硬件厂商的各种加速设备,需要重新设计程序或调整代码[2]。为此,针对GPU加速出现了许多新的编程模式,目前最新的GPU计算编程模型主要有CUDA和OPENCL[3,4]。然而,CUDA只能应用于NVIDIA的

收稿日期:2012-11-26 **修回日期**:2013-03-01 **网络出版时间**:2013-04-08
基金项目:广东省自然科学基金(S2011010002537);广东省科技计划项目(2012A030400029)
作者简介:曾文权(1978-),男,副教授,硕士,CCF会员,研究方向为计算机应用技术、图像处理和分析。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20130408.1543.001.html

GPU 设备,而 OPENCL 在不同硬件上实现时需要做重复的移植,该过程容易产生错误,且它们都需要使用底层 API,容易导致开发过程的效率低下。因此,一种基于指令制导计算的编程模型 OPENACC 应运而生,它既支持跨系统、跨主机的 CPU 和 GPU 加速设备的协同工作,还能为程序的并行执行进行调优配置[5]。在基于 OPENACC 指令的执行模型里[6],主程序代码由 CPU 运行,而计算密集区域则分派给 GPU 加速设备,其执行过程和数据管理则由程序员使用指令来进行管理[7,8]。鉴于 OPENACC 指令在 GPU 加速方面的优势,文中以高斯模糊为加速处理对象,设计一种基于 OPENACC 的 GPU 加速算法,以实现利用更精简的指令来获得更高的加速比。

1 OPENACC 总述

用于 C/C++和 Fortran 的 OPENACC API 和指令负责把底层的 GPU 任务交给编译的同时,又提供跨系统、跨主机 CPU 和加速设备的支持,到目前为止,一些公司的 OPENACC 的实现仅支持 NVIDIA GPU 设备,在这里对 OPENACC 指令协议提供一个简要概述[9]。

OPENACC 指令协议假设了一个主机执行模型,在这个模型里,主程序代码运行在主机 CPU 里,而计算紧密区域则分派给附加的 GPU 加速设备,内存模型分别为主机内存和设备内存,但这二种内存并不会自动同步,GPU 设备实现的是一个弱内存模型,防止不同计算单元的相冲突,但是可以通过显式的同步调用则可以让同一计算单元实现相关[10]。

执行和数据管理是由程序员使用指令来进行,一些基本的指令架构如下:

```
//Initialization: | x[i] < 1,i = 0,…,size - 1
#pragma acc data copy(x[0:size])//Data movement to/from device
{
while(error > eps)
{
error = 0.0;
#pragma acc parallel present(x[0:size])//kernel execution
#pragma acc loop gang vector reduction( +:error)//loop schedule
for(int i = 0;i < size; ++i)
{
x[i] * = x[i];
error + = fabs(x[i]);
}
}
}
```

最重要的指令是 parallel 和 kernel 关键字,描述了同步或异步的代码区域,这二个关键字映射到一个 OPENCL 或 CUDA 的核函数,这个核函数可以在 N 维的工作单元内执行。为了提高性能,也可以规定 gang,worker,vector_length 等关键字[11],关键字 gang 和 vector 可以对应于 OPENCL 里面的工作组和工作项,而关键字 worker 定义了工作项的组合,或是 CUDA 架构的 warp 概念,在 parallel 区域里,loop 关键字表明对使用内核函数来对循环加速,程序员可以加入额外的条件在 parallel,kernels 或 loop 关键字里来优化或纠正默认的数据管理。例如,可以使用显式的 data 关键字来指示在主机和加速硬件之间的数据传输,也可以使用 create 关键字在硬件上创建私有数据,或通过 present 关键字来告诉编译器数据已经在设备上,用户也可以通过 update 关键字来同步主机和设备之间的数据。

除指令之外,OPENACCR 指令集还提供了运行时库调用和环境变量,例,库调用可获取关于加速设备信息,进行初始化或收集在设备上的数据[12,13]。

2 基于 OPENACC 的高斯模糊的算法

高斯模糊作为一种图像模糊算法,主要是通过改变模糊半径的二维高斯滤波卷积核来实现[6]。一种更理想的方式把该过程分成两个单独的步骤:首先使用一个一维的核在水平或垂直的方向上来卷积核,然后再使用该一维核在剩余方向上进行卷积,其效果与一个二维卷积是等同的。最初的高斯模糊算法都是通过 CPU 计算实现,GPU 出现之后,虽然可以把该算法移植到 GPU 上去并行执行,但编程难度大且易出错。因此,文中基于 OPENACC 指令设计一种基于 GPU 加速的高斯模糊算法,以实现通过简化的编程算法和简洁的 C 语言代码完成对图像的高斯模糊处理过程。

2.1 算法及描述

基于 OPENACC 指令的 GPU 加速实现高斯模糊的基本思想是:将高斯模糊过程分为两个独立的阶段,通过在基于串行方式执行分离卷积的代码中添加 OPENACC 指令来实现,其实现代码为:

```
int main()
{
//加载源图像,创建目标图像
…
//根据用户输入创建卷积核
…
//使用 data 指令将源图像数据,目标图像数据,高斯核数据复制进显存
(1) #pragma acc data copyin(srcData[0:width * height]) copyin(kernel[0:4 * ksize]) copyin(dstData[0:width * height])
//开始第一阶段计算
//使用 kernels 指令定义核心
```

```
(2) #pragma acc kernels present_or_copyin(srcData[0:width
* height]) present_or_copyin(dstData[0:width * height]) present
_or_copyin(kernel[0:4 * ksize])
{
//使用 loop 指令定义外层循环
(3)#pragma acc loop independent
for(….)
{
//使用 loop 指令定义内层循环
(4)#pragma acc loop independent
for(…)
{
//高斯模糊计算,同时把结果放入目标内存
}
}
}
//开始第二阶段计算
(5) #pragma acc kernels present_or_copy
(srcData[0:width * height]) present_or_copyin
(dstData[0:width * height]) present_or_copyin
(kernel[0:4 * ksize])
{
//使用 loop 指令定义外层循环
(6)#pragma acc loop independent
for(…)
{
//使用 loop 指令定义内层循环
(7)# pragma acc loop independent
for(…)
{
//高斯模糊计算,同时把结果放入目标内存
}
}
}
```

2.2　算法说明

(1)在计算高斯模糊的第一阶段,当图像数据加载完成后,采用 data 指令将相应的源图像数据、目标图像数据,以及高斯核数据复制进显卡内存。在 GPU 完成第一阶段计算后,目标图像数据无需复制出显卡内存,所以采用 copyin 属性。

(2)采用 kernels 指令定义在 GPU 上运行的代码核心,由于目标数据已存于显卡内存,则采用 present_or_copyin 指令防止编译器向显卡重复复制目标数据。

(3)代码的(3)、(4)、(6)、(7)处采用 loop 指令来优化代码,即让编译器把 GPU 的核心内循环展开。此外,最底层的代码作为 GPU 内核函数在 GPU 上运行,还可以通过 gang,worker 等关键字来指定对应 CUDA 下的 block 和 thread 数目。

(4)第二阶段计算的代码与第一阶段类似,由于第一阶段的计算结果要进行垂直或水平卷积,计算结果要放入 srcData 且输出,所以在指定 srcData 的输入输出格式时使用 present_or_copy 属性。

相应的逻辑流程图如图 1 所示:

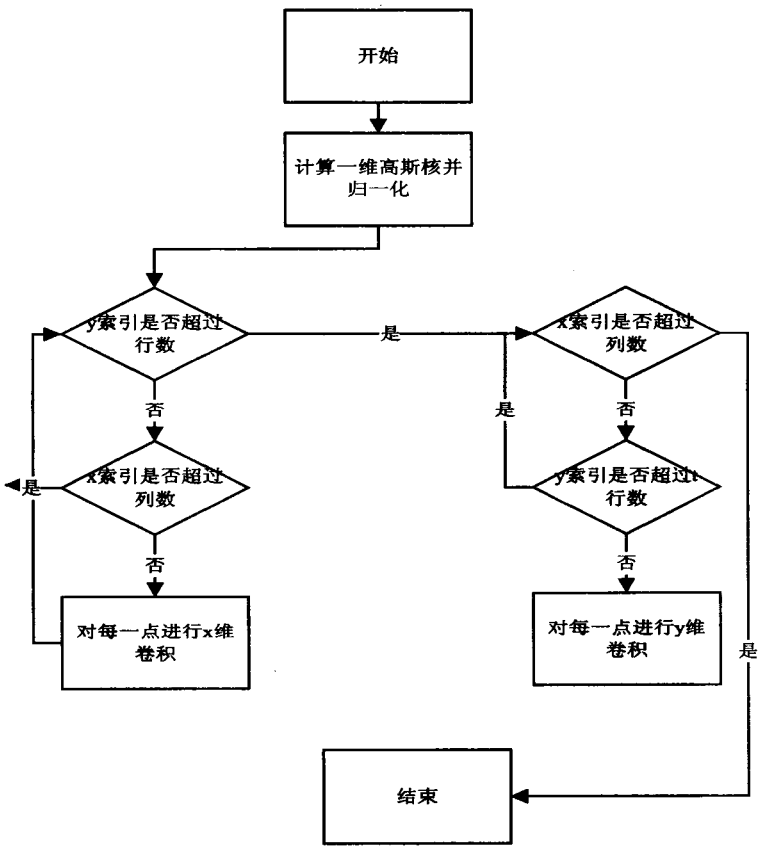


图 1　算法逻辑流程图

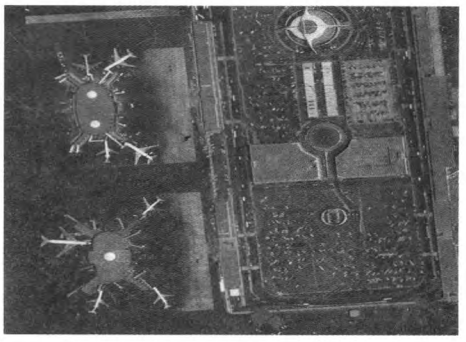
3　实验及结果

为验证文中算法的可行性,实验的软硬件平台为:redhat fedora 64 位 linux 系统,I7-2600 型号的 CPU(主频为 3.4G),16G 内存,NIVIDA TELSA C2070 的显卡,支持 OPENACC 部分指令的编译软件 HMPP 3.2。采用 CPU 与文中算法对图像进行高斯模糊处理后的效果如图 2 所示。

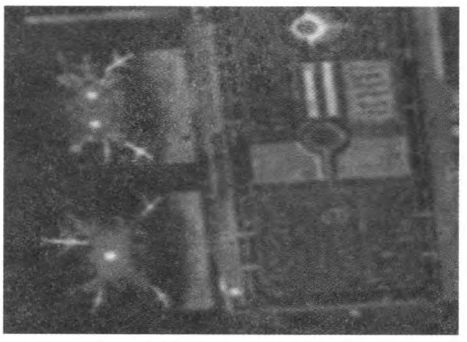
图(a)是像素为 1024×1024 原图,图(b)与(c)分别在 CPU 直接处理和 OPENACC 指令加速处理下,经过半径为 5 高斯模糊处理后的图像。从图像处理结果看,图(b)与(c)两图像模糊处理后结果基本相同。

为了验证文中算法的高效性,在上述软硬件平台下,分别采用图像数据为 256、512 和 1024 的三种图像来比较 CPU、OPENACC 和 CUDA 三种处理下高斯模糊方法的速度性能,实验结果如图 3 所示。

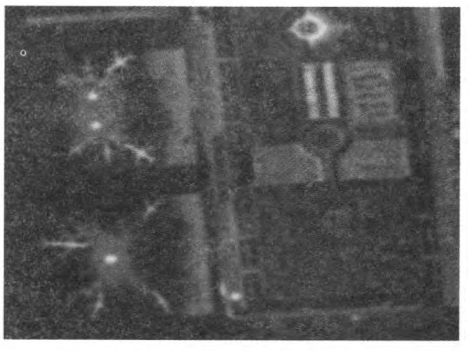
实验结果表明,随着图像像素的增大,CPU 直接处理所花费的计算时间呈指数增长趋势;在 GPU 代码中添加 OPENACC 指令后,计算机所花费的计算时间近似于线性增长;而 CUDA 代码因为经过块内内存访问



a)原图



b)CPU 处理高斯模糊



c)OPENACC 处理高斯模糊

图 2 图像处理效果图

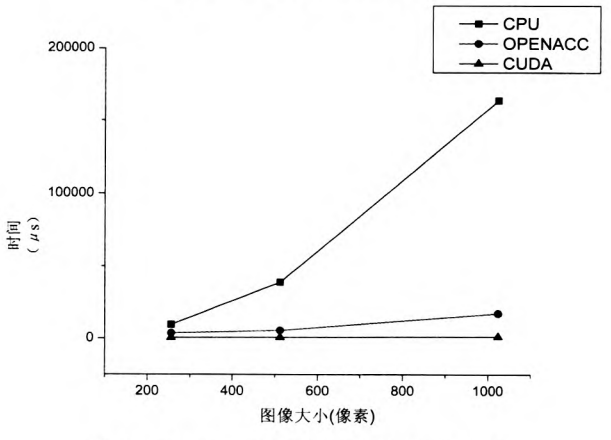


图 3 三种高斯模糊方法的性能曲线

优化和全局内存访问优化,其加速比较 OPENACC 多出 6 倍左右。但是,从代码的开发的效率上看,OPENACC 只需要在运行于 CPU 上的串行代码中增加十几行指令就能在 GPU 上取得 10 倍左右的加速比。该方式既保留了代码的通用性,又可以方便地将非并行代码迅速地移植到 GPU 处理器上进行并行处理。更为重要的是,随着编译器的进一步优化和硬件技术的发展,OPENACC 与 CUDA 等在底层技术实现上的差距将会越来越小,而且支持 OPENACC 指令转换将不仅仅针对 CUDA 设备,还包括其他更多厂商的硬件加速设备,从而能极大地提高 OPENACC 指令的普适性。

4 结束语

文中以高斯模糊算法为加速对象,通过采用为数不多的几条 OPENACC 指令,在原本非并行代码的基础上,就能得到较高的加速比,且不改变原来的代码结构,大大增强了代码的可扩展性、可维护性和可读性,该方法对其他的图像处理算法有一定的参考价值。当然,对于高斯模糊算法的实现来说,使用 OPENACC 指令进行优化还有很大的提升空间,下一步将继续深入研究 OPENACC 指令,以进一步缩小与 CUDA 和 OPENCL 之间的差距。

参考文献:

[1] 吴恩华. 图形处理器用于通用计算的技术、现状及其挑战[J]. 软件学报,2004,15(10):1493-1504.

[2] 方旭东,唐玉华,王桂彬,等. 模板操作在 GPU 上的实现与优化[J]. 计算机工程与科学,2011,33(3):41-45.

[3] 王 涛. 基于 GPU 的程序分析与并行化研究[D]. 郑州:解放军信息工程大学,2010.

[4] 姚 平. CUDA 平台上的 CPU/GPU 异步计算模式[D]. 合肥:中国科学技术大学,2010.

[5] 董 荦,葛万成,陈康力. CUDA 并行计算的应用研究[J]. 信息技术,2010(4):11-15.

[6] 钱 悦. 图形处理器 CUDA 编程模型的应用研究[J]. 计算机与数字工程,2008,35(12):177-180.

[7] 张 健,陈 瑞. 图形处理器在通用计算中的应用[J]. 计算机工程与设计,2009,30(14):3359-3361.

[8] 徐 品,蓝善祯,刘兰兰. 利用 GPU 进行通用数值计算的研究[J]. 中国传媒大学学报(自然科学版),2009,16(2):73-78.

[9] NVIDIA CUDA Programming Guide Version 2.3[EB/OL]. 2009. http://www.nvidia.com/content/cudazone/download/OpenCL/NVIDIA_OpenCL_ProgrammingGuide.pdf.

[10] ATI Stream Computing PenCL Programming Guide[EB/OL]. 2010. http://www.ljll.math.upmc.fr/groupes/gpgpu/tutorial/ATI_Stream_SDK_OpenCL_Programming_Guide.pdf.

[11] The OpenACC Application Programming Interface[EB/OL]. 2011. http://www.openacc.org/sites/default/files/OpenACC.1.0_0.pdf.

[12] NVIDIA. CUDA C Programming Guide v4.0[EB/OL]. 2011. http://www.shodor.org/media/content//petascale/materials/UPModules/matrixMultiplication/cudaCguide.pdf.

[13] Alvarez L, Lions P L, Morel J M. Image selective smoothing and edge detection by nonlinear diffusion[J]. SIAM J. on Numerical Analysis, 1992, 29(3):845-866.

一种基于OPENACC的GPU加速实现高斯模糊算法

作者： 曾文权， 胡玉贵， 何拥军， 林敏， ZENG Wen-quan， HU Yu-gui， HE Yong-jun， LIN Min

作者单位： 曾文权,胡玉贵,何拥军,ZENG Wen-quan,HU Yu-gui,HE Yong-jun(广东科学技术职业学院计算机工程技术学院,广东珠海,519090)， 林敏,LIN Min(珠海司迈科技有限公司,广东珠海,519090)

刊名： 计算机技术与发展 ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2013,23(7)