第36卷 第2期 河北工程大学学报(自然科学版) Vol.36 No.2
2019年6月 Journal of Hebei University of Engineering (Natural Science Edition) Jun.2019

文章编号：1673-9469（2019）02-0101-04 doi:10.3969/j.issn.1673-9469.2019.02.023

MATLAB 与 OpenACC 结合实现中值滤波算法并行优化

黄伟建，王月兴，黄 远

（河北工程大学 信息与电气工程学院，河北 邯郸 056000）

摘要：针对当前基于 CUDA 的中值滤波并行加速算法存在的编程复杂，图像数据提取和显示环节实现繁琐等问题提出了利用 MATLAB 和 OpenACC 结合的并行算法。该算法采用 MATLAB 实现图像的提取并转换成数组文件，之后利用 OpenACC 实现串行代码到并行代码的转换并处理 MATLAB 转换的数组文件，最后通过 MATLAB 将滤波后的数据显示成图像。通过仿真实验进行中值滤波算法的加速实验，结果表明：提出的算法在大量减少代码转换工作量的情况下加速效果较串行程序提升明显且与 CUDA 加速效果相当。

关键词：OpenACC；MATLAB；中值滤波；GPU 并行优化

中图分类号：TP31 **文献标识码**：A

Parallel Optimization of Median Filtering Algorithm Based on the Combination of MATLAB and OpenACC

HUANG Weijian，WANG Yuexing，HUANG Yuan

(School of Information & Electrical Engineering，Hebei University of Engineering，Handan 056000，China)

Abstract：In this paper，a parallel algorithm combining MATLAB and OpenACC is proposed for the complex programming problems of current CUDA-based median filtering parallel acceleration algorithm and the cumbersome implementation of image data extraction and display. The algorithm uses MATLAB to extract the image and converts it into an array file. Then it uses the OpenACC language to convert the serial code to the parallel code and processes the array file converted by MATLAB. Finally，the filtered data is displayed as an image through MATLAB. The acceleration experiment of the median filtering algorithm is carried out by simulation experiments. The experimental results show that the proposed algorithm accelerates the algorithm faster than the serial program in the case of greatly reducing the amount of code conversion workload. And it is equivalent to the effect of CUDA acceleration.

Key words：OpenACC；MATLAB；Median filter；GPU parallel algorithm

GPU 处理器在浮点运算方面的能力逐渐超越了 CPU，在图像处理方面，很多算法并非基于 GPU 的而是基于 CPU 的，并且一个好的图像算法可能是几个算法的综合运用，这样一来，如果想把这些算法移植到 GPU 上如果是利用 CUDA 或者 OpenCL 的话，其工作量是非常繁重的。文献 [1] 提出了利用 CUDA 实现遥感图像的并行加速，该算法加速效果明显，但是从串行代码到并行的代码转换工作量量非常巨大，不利于复杂算法的并行优化实现。文献 [2] 提出了基于 CUDA 的并行中值滤波算法，该算法实现了中值滤波的并行加速，但是同样的存在编程过程复杂的问题，并且图像数据的提取和显示过程较为复杂。OpenACC 提供了一种基于编译制导语句的编程方式，基于这种方式，无需大量改动原程序，而只需在原程序上添加一些控制语句即可实现代码移植。这就为广大科研人员提供了一种方便快速的并行优化方法。因此出现了大量基于 OpenACC 的并行加速研究实例如文献 [3-8] 所示。另一方面由

收稿日期：2019-02-26
基金项目：河北省自然科学基金资助项目 (F2015402077)；河北高等学校科学技术研究项目 (QN2018073)
作者简介：黄伟建 (1964-)，男，河北邯郸人，博士，教授，从事云计算与大数据，并行计算方面的研究。

于 OpenACC 目前仅支持类似 C 语言或 Fortran 语言这样的底层语言，因此图像数据的提取和显示过程较为复杂，而 MATLAB 在图像的提取和显示方面较为便捷。因此本文以图像滤波过程中的中值滤波算法为例，探索 OpenACC 和 MATLAB 结合在图像处理程序中并行优化的可行性和高效性。

1 OpenACC 概述

OpenACC 是一种新的加速器编程接口，以隐式和可移植的方式为加速器编程提供了一组类似 OpenMP 的编译指令。通过在原程序上添加一些简单的制导语句将数据和计算转移到加速器，从而可以显著简化类似 CUDA 编程方式的并行程序的移植过程。OpenACC 主要关注两个方面的内容，计算管理和数据管理。OpenACC 的大量制导语句就是从这两个方面进行设计的。OpenACC 将计算分为 CPU 上的计算和 GPU 上的计算两种，又将内存分为 CPU 内存和 GPU 内存两部分。

在计算管理方面，OpenACC 设计了基于三个并行层次的计算执行模型。这三个层次就是 gang、work、vector。这三个层次既对应着硬件上的流多处理器和计算核心等概念同时也对应着 CUDA 上的网格，线程块，和线程束等概念。

一些基本的制导语句语法如下：

```
#pragma acc 导语名字 子语列表
{
   程序段；
}
```

OpenACC 有两个计算构件，parallel 和 kernels，也就是制导语句

```
#pragma acc parallel
#pragma acc kernels
```

编译器把 kernels 区域编译成一系列在加速器设备上执行的 CUDA kernel，一方面 OpenACC 支持 for 循环的自动并行展开，这种支持既包括代码块内含有多个 for 循环，也支持嵌套循环，因此用 for 循环实现像素的遍历有助于并行化。另一方面 OpenACC 提供了 routine 导语，该语句可以自动将串行的调用函数转换为可被设备调用的设备函数。这有利于将每个密码散列的过程快速的放置于设备上。

示例代码如下：

```
#pragma acc routine( 被调函数 1)
#pragma acc routine( 被调函数 2)
#pragma acc routine( 被调函数 3)
#pragma acc parallel loop
for(int i=0，i<M；i++)
   {
   函数调用 1；
   函数调用 2；
   函数调用 3；    }
   }
```

2 OpenACC与MATLAB结合实现的中值滤波算法

OpenACC 与 MATLAB 结合实现并行优化的过程如图 1 所示：

图 1 OpenACC 与 MATLAB 结合原理示意图
Fig.1 The schematic diagram of the combination principle of MATLAB and OpenACC

2.1 中值滤波概述

中值滤波是一种依据统计学理论提出的非线性信号处理技术，中值滤波用相邻各像素点的中值代替该点的像素值，从而使其更接近真实值，以此来去掉单独的噪点。首先建立一个滑动模板，然后对滑板内部的像素值进行排序并获取中间值。可以根据不同的需要定义模板的大小和形状。当模板内的像素个数为奇数时，中值算法如公式 1。

$$G(x,y)=P(a)\left[\frac{i-1}{2}\right] \tag{1}$$

当模板内的像素个数为偶数时，中值算法如公式2。

$$G(x,y)=P(a)\left[\frac{i}{2}\right] \tag{2}$$

其中，a 是由模板内像素组成的一个长度为 m 的数组，P 函数用来对数组进行排序，(i-1)/2 与 i/2 是数组的下标。增加图像的像素或者增滑动模板的规模会使得计算量增高。为了实现在 GPU 上的并行化，必须充分利用像素运算之间独立性的特点，为每个 GPU 线程分配一个像素的计算任务，这里必须保证该像素的计算所涉及的变量是线程私有的。对中值滤波算法，每个线程需要从总的图像数组中读取其自己所涉及的像素及其周围的像素值。本文结合图像处理和 OpenACC 的特点，将程序的设计

分为 3 个部分，第一部分图像数据的提取和转化，主要利用 MATLAB 来进行，利用 MATLAB 读取图像并转化为单通道的灰度数据存放于 data.txt 文件。第二部分运用基于 OpenACC 的中值滤波算法对数据文件进行处理。第三部分，处理后的数据传回 MATLAB 显示图像处理效果。

2.2 MATALAB 提取数据图像

由于现在 OpenACC 只支持 C/C++，Fortran，而利用 C 语言或者 Fortran 来提取图像数组，是非常不方便的，因此我们在图像提取阶段选择了 MATLAB，利用 MATLAB 我们可以非常方便的获取接下来所需要的数据。为了方便数据的传输，将二维的像素灰度值按照行优先放入一维数组，然后将该数组放入 .txt 文件。代码如下：

```
S=imread('ff.bmp')； % 读取灰度图像
ss=S(：，：，1)；% 提取单通道数据
fid=fopen('t4.txt'，'wb')； % 打开要写入的文件
p2=ss(：)； % 数据向量化处理
p3=p2'；
fwrite(fid，p3，'uchar')； % 图像数据写入文件
```

2.3 OpenACC 进行中值滤波

利用 OpenACC 实现中值滤波的并行化需要从两个方面入手，一方面是数据的转移，另一方面是计算的合理规划，即找出计算量巨大的关键环节，并且合理的调整线程配置，只有这样才能达到理想的效果。首先不难发现图像数组是需要从 CPU 内存放入 GPU 内存的。而且要避免该数组被反复的读取，因此采用 OpenACC 的 data copy 子语，该子语确保接下来的计算过程中只从 GPU 显存读取而不会跳转到 CPU 内存。同时考虑到变量的私有化问题，由于本文是将每一个像素交于一个线程，所以中间涉及到的变量，必须为该线程私有，这就要用到 private 子语，如果未这样声明将导致计算结果错误。该错误的出现是由于线程处理了相同的数据，导致所有线程计算结果相同。

对于计算环节，由于 OpenACC 是良好支持循环结构自动并行化的语言，所以串行程序中以循环的方式遍历所有像素的计算过程即是要添加 OpenACC 的地方，因为该过程是整个程序的计算核心。需要注意的是并非所有循环都要并行化，并行化也是有代价和设备开销的。对于一些小规模的循环，强制并行化是不能提升速度的。另一面线程的布局是十分关键的，不同的线程配置会导致计算速度出现数倍的差距。这里非常重要的有两点：一方面是并行方式，即 vector，gang，worker 三种不同并行方式的选择，另一方面是线程块内的布局，即 vector 的数量和 worker 的数量，他们两者之积对于确定的 GPU 硬件而言是存在固定上限的。超过上限会编译报错。本文所用到的硬件其 vector 和 worker 之积上限为 1024。经反复调试，本实验中在 vector 并行模式下，且 vector 数量大于 256 时取得了最好效果，这个结果和中值滤波算法中各像素之间较高的数据和计算独立性相适应。vector 并行模式以此见长。伪代码如下：

```
#pragma acc data copy(aValue，in1，out1)
  // 将数据从 CPU 内存转移至 GPU 内存
  #pragma acc parallel  num_gangs(65535) vector_length(512) num_workers(2)
 // 定义 GPU 的并行参数
#pragma acc loop private(aValue) vector// 声明线程私有变量。
   for(int i = iFilterMY；i < KUAN - iFilterH + iFilterMY + 1；i++)
  {
     for(int j = iFilterMX；j < GAO - iFilterW + iFilterMX + 1；j++)
   // 基于滤波器的大小实现二维像素的遍历处理
 {
   读取单点像素灰度值；
   根据滤波器大小读取该像素周围数据并保存至数组 x；
   对数组 x 内的数据进行排序并取得中间值，并赋予新的图形向量。
      }
}
新向量写到新的图像数据存储文件。
```

2.4 MATLAB 显示图像去噪效果

图像显示环节和图像数据的提取阶段原理相同，为了方便的查看数据处理的结果，首先要将经 OpenACC 处理的数组放入 .txt 文件，然后将该文件放入 MATLA 工作目录，并利用下面的程序将图像显示出来。代码如下：

```
fi=fopen('o99.txt'，'rb')；% 打开图像数据存储文件
data=fread(fi，3 648*2 736，'uchar')；% 读取数据
```

d=reshape(data，2 736，3 648)；% 向量数据还原为二维图像数据

g=uint8(d)；% 格式化图像数据

imshow(g)；% 显示经中值滤波处理后的图像

3 实验及结果

为了验证算法的可行性，实验的硬件平台为：

CPU：Intel Core i5-4210 M @2.60 GHz

GPU：NVIDIA GeForce GTX 960M

软件及编译环境为：ubuntu 16.04&PGI，Win10 &MATLAB

本文选取了一幅分辨率为 3 648*2 736 的图像，首先利用 MATLAB 为其加入了椒盐噪声，然后，经过 OpenACC 中值滤波程序的处理，椒盐噪声被去掉，实验中我们选取了 6 组大小不同的滤波器进行试验，分别对照在不同大小滤波器下串行 CPU 程序，CUDA 程序和本文加速程序的用时，来统计加速效果。单位是毫秒。图片处理效果如图 3 所示，图 2 为加入椒盐噪声的效果。程序运行的时间如表 1 所示。

从上面的实验我们可以看到本文提出的算法滤波效果非常清晰，而其所用时间比串行程序显著减少并且与 CUDA 程序加速效果相近。

图 2 加入椒盐噪声的图像

Fig.2 Image with salt&pepper noise

图 3 去除椒盐噪声后的图像

Fig.3 Image after removing salt & pepper noise

表 1 本文算法相较串行程序和 CUDA 的加速效果

Tab.1 Contrast of the acceleration effect of the program proposed in this paper with serial program and CUDA program

滤波器大小	串行程序用时 /ms	CUDA 程序用时 /ms	本文算法用时 /ms
3*3	3 046	60	59
4*4	9 002	139	133
5*5	20 324	276	272
6*6	39 936	518	522
7*7	70 592	933	943
8*8	116 781	1 520	1 485

4 结论

本文提出的并行中值滤波算法减少了传统并行方法中由串行代码到并行代码转换的工作量，图像的提取和显示环节实现起来较 C 语言简便。经本文并行算法滤波后的图像清晰无噪声，计算速度与串行程序相比加速效果明显，同时也不逊色于高效的 CUDA 算法。这就为图像处理的加速提供了一个快速的并行代码移植方法。

参考文献：

[1] 丁毅乐 . 基于 CUDA 架构的遥感图像滤波算法并行处理 [D]. 郑州：解放军信息工程大学，2017.

[2] 吕亚飞，贾堃阳 . 基于 CUDA 的快速中值滤波算法 [J]. 现代计算机：专业版，2011，2011(13)：3-6.

[3] 雷 芸，LEIYun. 基于中值预滤波的非局部平均去噪算法 [J]. 微电子学与计算机，2015(5)：138-142.

[4] 蔡明晧，范秋华 . 一种便于实时传输的中值滤波算法 [J]. 制造业自动化，2018(7).

[5]SHARMA S，SUBRAMANYAM A V，JAIN M，et al.Anti-forensic technique for median filtering using L1-L2 TV model[C]// IEEE International Workshop on Information Forensics and Security. IEEE，2017：1-6.

[6]ZHANG C R，QIU Y E，LEI T，et al.A fast vector median filtering algorithm based on minimal reference distance[J]. Electronic Design Engineering，2018.

[7]GREEN O.Efficient Scalable Median Filtering Using Histogram-Based Operations.[J]. IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society，2018，27(5)：2217.

[8]ZHANG S，LI X，ZHANG C.Modified Adaptive Median Filtering[C]//International Conference on Intelligent Transportation，Big Data & Smart City. IEEE，2018：262-265.

（责任编辑 李新）