CUDA Competition 算法说明

董林森 自动化工程学院 lukedong123@gmail.com

简介：

CCL 为图像识别和处理中一个重要的算法，本文提供的算法基于 cuda 平台，在经典的并行 Union-Find 2 Pass 算法（并查集）的基础上，做了一定的改进和性能提升。以下包括三个部分，算法说明，即算法的原理及基本步骤；性能测试，对该算法和其他已有算法进行了性能的对比；最后进行总结和仍存在问题的讨论。

算法说明：

1.算法主要文件说明:

ccl_gpu.cu：包括 CCL 算法的主程序。

get_label_count.cu：包括统计不同标记的个数。

以及其对应的.cuh 头文件:

ccl_gpu.cuh，get_label_count.cuh

以及 gpu1.bmp, gpu2.bmp, gpu2.bmp, gpu3.bmp 为该算法下的对应给定四个输入图像的输出结果。

2.在该 CCL 算法中基本步骤如下：

A. init_label：初始化 d_label 和 d_fa。
B. ccl_find：对某个像素点 p 检查它的 8 个相邻点，并使用其中最小的 d_label 值更新到 d_fa[p]中。
C. ccl_merge：首先对所有像素点 p 的 d_fa 进行路径压缩。
D. ccl_update_label：对某个像素点 p，找到 d_fa 中 p 的根节点更新 d_label[p]。
E. 重复步骤 B，直到步骤 B 中没有 d_label 被更新为止，最终 d_label 的值即为 imgOut 的结果。

3.计算 label 的数量的算法：

主要利用了 CUDA 平台的内存读取与线程等特性进行了优化。在未优化时计算 label 的数量要使用大量的原子操作 atomicAdd()，这是由于对全局内存的资源竞争造成的，从而使得部分操作变成了串行代码。进行优化后原子操作大量减少，从而提升了计算速度。主要思想为使用每个线程对内存进行扫描，将结果累加到寄存器中，当线程完成自己需要计算的部分后，最后一次性将结果写回全局内存。

性能测试：

1.测试平台：

CPU Intel Core i7-6700HQ @ 2.60GHz 四核 2.60 GHz (100 MHz x 26.0)
Nvidia GeForce GTX 960M

2.与 CPU CCL 算法和经典 Union-Find 算法运行时间对比：

	1.bmp	2.bmp	3.bmp	4.bmp
CPU	1.527ms	3.867ms	1.579ms	0.971ms
Union-Find	33.379ms	70.241ms	82.458ms	102.659ms
ccl_gpu	5.665ms	6.906ms	2.888ms	4.135ms
Speed Up（with Union-Find)	5.892	10.179	28.631	24.667

Union-Find 实现代码详见附件中 union_find.cu 文件，该算法基于[1]的 edge set 版本。

可见与 Union-Find 相比有较大的加速比，但有趣的是对于不同类型的输入图形，加速比变化较大。基本可以得出该算法对于较稀疏的图像有较好的性能提升。

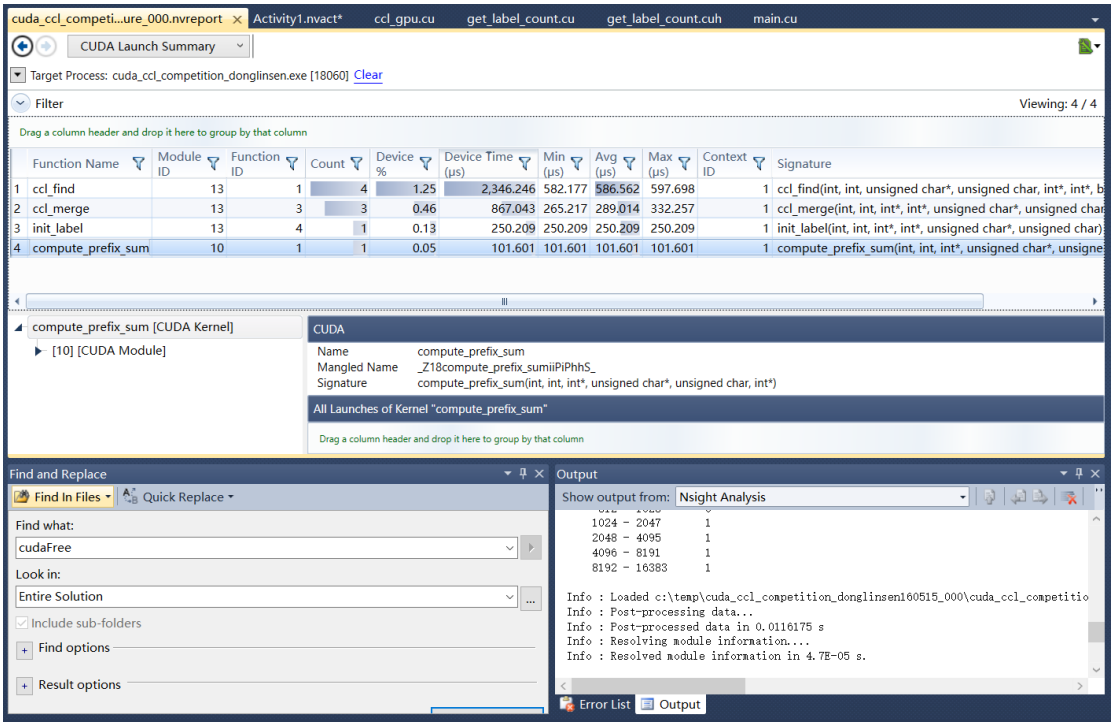
3.get_label_count 与未优化版运行比较：

	1.bmp	2.bmp	3.bmp	4.bmp
原版本	293.604us	724.838us	280.002us	162.015us
get_label_count	101.601us	200.641us	92.863us	57.120us
Speed Up	2.900	3.620	3.017	2.842

由此可以看出加速比在 3 左右，而且可以显然得出结论：随着输入数据的规模增大，加

速比会随之增加。

测试数据由 cuda 平台的调试工具 Nisght 统计得，具体如下图：



图中被选中的 compute_prefix_sum 即为计算 Label 的函数

未优化版本代码：

```
__global__ void get_label_num(int w, int h, unsigned char *img, int *d_label, unsigned
    char byF, int *d_lable_num)
{
    int loc = get_loc(w,h);
    if(loc>=w*h) return;
    if(img[loc] == byF && d_label[loc] == loc) { //判断是否为根节点
        atomicAdd(d_lable_num, 1); //此处原子操作导致其变为串行代码
    }
}
```

总结和改进方向：

该算法在一定程度上改进了 Union-Find 算法，同时对于计算 Label 的优化也使得其性能有一定的提升。但由于未进行大量的随机测试等，仍无法对其性能有很好的定论。其效率仍然低于源代码提供的 CPU 算法，经过分析，基本得出的结论为：由于在 ccl_find 和 ccl_merge 大量多次的对全局内存进行访问，从而导致了效率的下降。

因此个人认为改进的方向应为优化内存访问机制，例如应充分利用线程块内共享内存等，但还未得出一个比较好的方案。

另外一个比较有趣的现象为，该算法在较稀疏的图上的性能要好过密集的图，这应该是由于其迭代次数大量减少的原因。此现象由通过 Nsight 观察内核的调用次数可得。

参考资料：

[1] K.A. Hawick, A. Leist , D.P. Playne Parallel graph component labelling with GPUs and CUDA Parallel Computing Volume 36, Issue 12, December 2010, Pages 655–678
[2] Shane Cook 《CUDA 并行程序设计》 机械工业出版社