# Lab 3: 2D Stencil

编号:4 学号：201522060717 姓名：杨新雨

## 1. Objective

Read the Origin 2D Stencil code and find what can be improved.

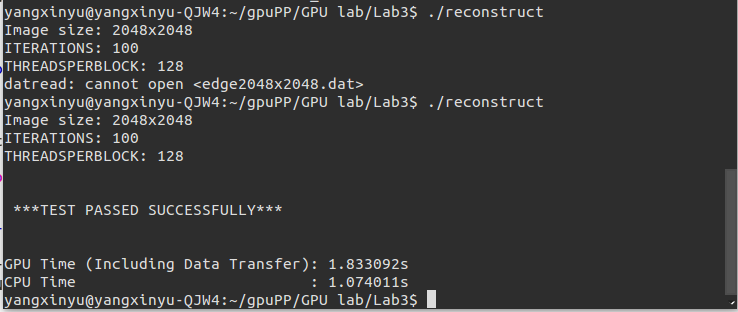
## 2. Analysis

1.在原文件reconstruct.cu中，在主函数中的迭代中存在将d\_output传输到gpu\_output，在由gpu\_output传输回d\_input中，由于gpu\_output是定义在cpu中的内存空间，而d\_output和d\_input是定义在gpu上的内存空间，在每一次迭代过程中都会存在两次cpu和gpu之间的数据交互，cpu和gpu之间的数据传输时间相对而言是很慢的，所以可以直接将d\_output传输到d\_input中，它们都是在gpu上的数据，数据传输速度更快，且数据传输次数减少，在迭代结束之后再将最后一次的d\_output数据传输到gpu\_output。

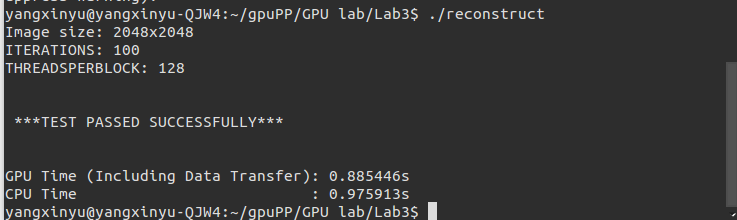
2. Memory coalescing:由于多数系统和编程语言都是采用row-major，所以要使并行线程在同一时刻访问的数据相对在一段连续接近的区域，就需要让它们各自负责读某一列而不是某一行。

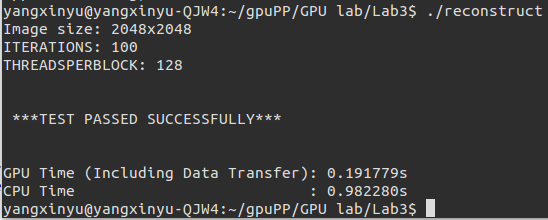
3. 可以改变blockSize来提升效率

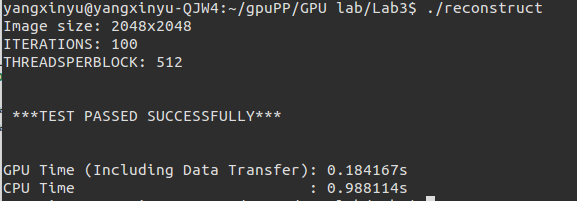
**3. 运行截图**

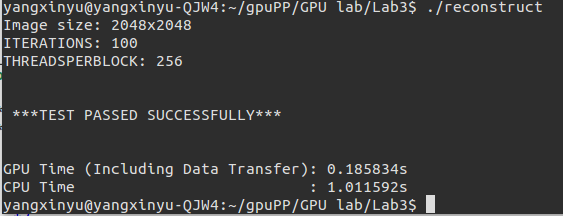
1. 原始代码未修改运行截图:

2. 修改删除loop中多余的数据传输后：



3. 每个线程改为列读取:

4. 改变THREADPERBLOCK:



**4. 总结**

由上面的运行截图可以看出每一种方式都能提升程序性能，其中改变线程读取数据方式为列处理后提升性能最大，在THREADPERBLOCK为512时GPU时间最短。而CPU时间远远超过GPU执行时间，故程序性能瓶颈在cpu方面。

**5. 关键代码修改**

**1. Reconstruct.cu:**删除d\_output与gpu\_output间多余的传输。

**---------------------------------------------**

/\* run on GPU \*/

for (i = 0; i < ITERATIONS; i++) {

/\* run the kernel \*/

dim3 blocksPerGrid(ceil(N/THREADSPERBLOCK),1,1);

dim3 threadsPerBlock(THREADSPERBLOCK,1,1);

inverseEdgeDetect<<< blocksPerGrid, threadsPerBlock >>>(d\_output, d\_input, d\_edge);

cudaThreadSynchronize();

//use this to reduce data transfer between cpu and gpu

cudaMemcpy(d\_input,d\_output,memSize,cudaMemcpyDeviceToDevice);

}

//reduce data transfer between cpu and gpu,copy the final result to cpu

cudaMemcpy(gpu\_output, d\_output, memSize, cudaMemcpyDeviceToHost);

**2. reconstruct\_kernels.cu:**改为每个线程处理一列

col = blockIdx.x\*blockDim.x + threadIdx.x + 1;

for(row = 1; row <= N; row++){

idx = row \* numcols + col;

idx\_south = (row - 1) \* numcols + col;

idx\_north = (row + 1) \* numcols + col;

idx\_west = row \* numcols + (col - 1);

idx\_east = row \* numcols + (col + 1);

/\* perform stencil operation \*/

d\_output[idx] = (d\_input[idx\_south] + d\_input[idx\_west] \

+ d\_input[idx\_north] + d\_input[idx\_east] - \

d\_edge[idx]) \* 0.25;

}