Lập trình song song trên GPU

Hoàng Minh Thanh (18424062)

TH2 : Reduction

Bộ môn Công nghệ phần mềm

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

Contents

I. Quá trình cài đặt 3

1) Hoàn thành phần tính "gridSize" và hàm kernel 1 trong file "bt2.cu" 3

2) Chạy chương trình ở câu 1 với các kích thước block khác nhau: 1024, 512, 256, 128. 5

3) Hoàn thành hàm kernel 2 và 3 trong file "bt2.cu". Trong file bài làm, bạn ghi nhận lại kết quả chạy 6

4) Giả sử block có kích thước là 128. Với mỗi hàm kernel: với mỗi giá trị "stride", cho biết trong mỗi block có những warp nào bị phân kỳ (không xét block cuối). 8

1. Kernel1 8

2. Kernel2 8

3. Kernel3 9

Vì máy tính cả nhân của em không có GPU nên bắt buộc em phải sử dụng Google Colab :

https://colab.research.google.com/drive/1piW2XzDPXJOg3WBliJAvCGBbQqDXeT8N

(Thầy có thể vào link Online để xem luồng chạy dễ hơn báo cáo)

I. Quá trình cài đặt

Chi tiết cài đặt trong file .cu và file Google Colab

Vì các file thực thi và câu lệnh em để trên github private repos nên phải cấu hình clone bằng ssh trên Google Colab

1) Hoàn thành phần tính "gridSize" và hàm kernel 1 trong file "bt2.cu"

```
**********GPU info**********
Name: Tesla P100-PCIE-16GB
Compute capability: 6.0
Num SMs: 56
Max num threads per SM: 2048
Max num warps per SM: 64
GMEM: 17071734784 bytes
****************************

Input size: 16777217

Kernel 1
Grid size: 16385, block size: 512
Kernel time = 1.226816 ms, post-kernel time = 0.038560 ms
CORRECT :)

Kernel 2
Grid size: 16385, block size: 512
Kernel time = 1.103360 ms, post-kernel time = 0.041088 ms
CORRECT :)

Kernel 3
Grid size: 16385, block size: 512
Kernel time = 0.550752 ms, post-kernel time = 0.048384 ms
CORRECT :)
```

Kết quả cài đặt

```cpp
__global__ void reduceBlksKernel1(int * in, int n, int * out)
{
    // TODO
    int i = blockIdx.x * blockDim.x * 2 + threadIdx.x * 2;
    for (int stride = 1; stride < 2 * blockDim.x; stride *= 2)
    {
        if ((threadIdx.x % stride) == 0)
            if (i + stride < n)
                in[i] += in[i + stride];

        __syncthreads(); // Synchronize within each block
    }

    if (threadIdx.x == 0)
        out[blockIdx.x] = in[blockIdx.x * blockDim.x * 2];
}
```

Ta xâu dựng hàm lấy thông tin info phần cứng

```
/tmp/tmpnj7cq7n6/fd0cc78d-12d4-4535-bb6e-d86205b99720.out Starting...
Detected 1 CUDA Capable device(s)
Device 0: "Tesla P100-PCIE-16GB"
  CUDA Driver Version / Runtime Version          10.1 / 10.1
  CUDA Capability Major/Minor version number:    6.0
  Total amount of global memory:                 15.90 GBytes (17071734784 bytes)
  GPU Clock rate:                                1329 MHz (1.33 GHz)
  Memory Clock rate:                             715 Mhz
  Memory Bus Width:                              4096-bit
  L2 Cache Size:                                 4194304 bytes
  Max Texture Dimension Size (x,y,z)             1D=(131072), 2D=(131072,65536), 3D=(16384,16384,16384)
  Max Layered Texture Size (dim) x layers        1D=(32768) x 2048, 2D=(32768,32768) x 2048
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Maximum sizes of each dimension of a block:    1024 x 1024 x 64
  Maximum sizes of each dimension of a grid:     2147483647 x 65535 x 65535
  Maximum memory pitch:                          2147483647 bytes
```

2) Chạy chương trình ở câu 1 với các kích thước block khác nhau: 1024, 512, 256, 128.

Theo cú phép hiển thị Registers Per Thread

```
[32]  1 !nvcc bt2.cu -o btdemo2 --ptxas-options=-v

ptxas info    : 0 bytes gmem
ptxas info    : Compiling entry function '_Z17reduceBlksKernel3PiiS_' for 'sm_30'
ptxas info    : Function properties for _Z17reduceBlksKernel3PiiS_
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 12 registers, 344 bytes cmem[0]
ptxas info    : Compiling entry function '_Z17reduceBlksKernel2PiiS_' for 'sm_30'
ptxas info    : Function properties for _Z17reduceBlksKernel2PiiS_
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 13 registers, 344 bytes cmem[0]
ptxas info    : Compiling entry function '_Z17reduceBlksKernel1PiiS_' for 'sm_30'
ptxas info    : Function properties for _Z17reduceBlksKernel1PiiS_
    0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 14 registers, 344 bytes cmem[0]
```

```
1
```

Block Size	Grid Size	Occupancy	Num blocks / SM	Kernel time	Post-kernel time	Total time
1024	8193	100%	2	1.440032	0.020352	1.460384
512	16385	100%	4	1.225440	0.041568	1.267008
256	32769	100%	8	1.084576	0.083904	1.16848
128	65537	100%	16	0.977472	0.165504	1.142976

```
// TODO
// Allocate device memories
uchar3 *d_inPixels, *d_outPixels;
float *d_filter;
CHECK(cudaMalloc(&d_inPixels, width * height * 3 * sizeof(uchar3)));
CHECK(cudaMalloc(&d_outPixels, width * height * 3 * sizeof(uchar3)));
CHECK(cudaMalloc(&d_filter, filterWidth * filterWidth * sizeof(float)));

// Copy data to device memory
CHECK(cudaMemcpy(d_inPixels, inPixels, width * height * 3 * sizeof(uchar3), cudaMemcpyHostToDevice));
CHECK(cudaMemcpy(d_filter, filter, filterWidth * filterWidth * sizeof(float), cudaMemcpyHostToDevice));

// Call kernel
dim3 gridSize((width - 1) / blockSize.x + 1, (height - 1) / blockSize.y + 1);
blurImgKernel<<<gridSize, blockSize>>>(d_inPixels, width, height, d_filter, filterWidth, d_outPixels);

// Copy result from device memory
CHECK(cudaMemcpy(outPixels, d_outPixels, width * height * sizeof(uchar3), cudaMemcpyDeviceToHost));

// Free device memories
CHECK(cudaFree(d_inPixels));      You, 34 minutes ago • update TH1
CHECK(cudaFree(d_outPixels));
CHECK(cudaFree(d_filter));
```

Khi thay đổi blockSize thì thứ tự index bị chồng chéo nhau và dẫn đến thay đổi tốc độ truy cập bộ nhớ

3) Hoàn thành hàm kernel 2 và 3 trong file “bt2.cu”. Trong file bài làm, bạn ghi nhận lại kết quả chạy

Cài đặt

```cpp
__global__ void reduceBlksKernel2(int * in, int n, int * out)
{
    // TODO
    int numElemsBeforeBlk = blockIdx.x * blockDim.x * 2;

    for (int stride = 1; stride < 2 * blockDim.x; stride *= 2)
    {
        int i = numElemsBeforeBlk + threadIdx.x * 2 * stride;
        //int index = 2 * stride * tid;
        if (threadIdx.x < blockDim.x / stride){
            if (i + stride < n){
                in[i] += in[i + stride];
            }
        }

        __syncthreads(); // Synchronize within each block
    }

    if (threadIdx.x == 0)
        out[blockIdx.x] = in[numElemsBeforeBlk];
}
```

```cpp
__global__ void reduceBlksKernel3(int * in, int n, int * out)
{
    // TODO
    int numElemsBeforeBlk = blockIdx.x * blockDim.x * 2;

    for (int stride = blockDim.x; stride > 0; stride >>= 1)
    {
        int i = numElemsBeforeBlk + threadIdx.x;
        if (threadIdx.x < stride)
            in[i] += in[i + stride];

        __syncthreads();
    }
    if (threadIdx.x == 0)
        out[blockIdx.x] = in[numElemsBeforeBlk];
}
```

```
**********GPU info**********
Name: Tesla P100-PCIE-16GB
Compute capability: 6.0
Num SMs: 56
Max num threads per SM: 2048
Max num warps per SM: 64
GMEM: 17071734784 bytes
*****************************

Input size: 16777217

Kernel 1
Grid size: 8193, block size: 1024
Kernel time = 1.440032 ms, post-kernel time = 0.020352 ms
CORRECT :)

Kernel 2
Grid size: 8193, block size: 1024
Kernel time = 1.264512 ms, post-kernel time = 0.018720 ms
CORRECT :)

Kernel 3
Grid size: 8193, block size: 1024
Kernel time = 0.584640 ms, post-kernel time = 0.018976 ms
CORRECT :)
```

4) Giả sử block có kích thước là 128. Với mỗi hàm kernel: với mỗi giá trị "stride", cho biết trong mỗi block có những warp nào bị phân kỳ (không xét block cuối).

1. Kernel1

stride = {1,2,4,8,16,32,64, 128}

stride = 1 : không wrap nào bị phân kỳ

stride = 2, 4, 8, 16, 32 : có tất cả các wrap đều bị phân kỳ

stride = 64, 128 : Chỉ có một wrap bị phân kỳ

2. Kernel2

stride = {1,2,4,8,16,32,64, 128}

stride = 128, 64, 32 : không wrap nào bị phân kỳ

stride = 2, 4, 8, 16 : đều có 1 wrap bị phân kỳ

3. Kernel3

stride = {128, 64, 32, 16, 8, 4, 2, 1}

stride = 128, 64, 32 : không wrap nào bị phân kỳ

stride = 2, 4, 8, 16 : đều có 1 wrap bị phân kỳ