Lập trình song song trên GPU

Hoàng Minh Thanh (18424062)

BT3 : Tính tổng tích luỹ

Bộ môn Công nghệ phần mềm

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

Contents

1 Cài đặt chương trình

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/1-5Rznm3w2rAGpjHAzhTOqPP9ohnH-9S4#scrollTo=jQaNUnaMTQEJ

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

a) Chương trình CUDA tính tổng tích lũy :

Cài đặt ở file **sumVector.cu**

Đây là kết quả tính tổng vector

```
%%cu
#include <stdio.h>
#include <stdlib.h>
#include <cuda_runtime.h>
#include <sys/time.h>

using namespace std;

#define CHECK(call)                                                          \
{                                                                            \
    const cudaError_t error = call;                                          \
    if (error != cudaSuccess)                                                \
    {                                                                        \
        printf("Error: %s:%d, ", __FILE__, __LINE__);                        \
        printf("code:%d, reason: %s\n", error, cudaGetErrorString(error));   \
        exit(1);                                                             \
    }                                                                        \
}

double seconds(){
    struct timeval tp;
    gettimeofday(&tp, NULL);
    return ((double)tp.tv_sec + (double)tp.tv_usec * 1.e-6);
}
```

```
void initialData(float *data, int size)
{
    srand(0);
    for (int i = 0; i < size; i++)
    {
        data[i] = (float)(rand()) / RAND_MAX;
    }
}

void printFirst10(float *data, int size){
    for (int i = 0; i < size && i < 10; i++){
        printf("%f\n", data[i]);
    }
}

float recursiveReduce(float *data, int const size)
{
    if (size == 1)
        return data[0];

    if (size % 2 == 1) {
        data[0] += data[size];
    }

    int const stride = size / 2;
    for (int i = 0; i < stride; i++){
        data[i] += data[i + stride];
    }

    return recursiveReduce(data, stride);
}

__global__ void reduceNeighbored(float *g_idata, float *g_odata, unsigned int n
)
{
    // set thread ID
    unsigned int tid = threadIdx.x;
    unsigned int idx = threadIdx.x + blockIdx.x * blockDim.x;

    // convert global data pointer to the local pointer of this block
    float *idata = g_idata + blockIdx.x * blockDim.x;

    // boundary check
    if (idx >= n) return;
```

```
    // in-place reduction in gloabl memory
    for (int stride = 1; stride < blockDim.x; stride *= 2){
        if ((tid % (2 * stride)) == 0){
            idata[tid] += idata[tid + stride];
        }

        // synchronize within threadblock
        __syncthreads();
    }

    // write result for this block to global mem
    if (tid == 0)
        g_odata[blockIdx.x] = idata[0];
}

__global__ void reduceNeighboredLess(float *g_idata, float *g_odata, unsigned i
nt n){
    // set thread ID
    unsigned int tid = threadIdx.x;
    unsigned int idx = blockIdx.x * blockDim.x + threadIdx.x;

    // convert global data pointer to the local pointer of this block
    float *idata = g_idata + blockIdx.x * blockDim.x;

    // boundary check
    if (idx >= n) return;

    // in-place reduction in gloabl memory
    for (int stride = 1; stride < blockDim.x; stride *= 2){
        // convert tid into local array index
        int index = 2 * stride * tid;
        if (index < blockDim.x){
            idata[index] += idata[index + stride];
        }

        // synchronize within threadblock
        __syncthreads();
    }

    // write result for this block to global mem
    if (tid == 0)
        g_odata[blockIdx.x] = idata[0];
```

```
}

int main()
{
    // set up device
    int dev = 0;
    cudaSetDevice(dev);
    int const BLOCK_SIZE = 32;

    // set up vector
    int size = pow(2, 2);

    float *data, host_total, device_total = 0; // vector

    int nBytes = size * sizeof(float);
    data = (float *)malloc(nBytes);

    // initialize data at host side
    initialData(data, size);
    printFirst10(data, size);

    // malloc device global memory
    float *g_idata, *g_odata;
    CHECK(cudaMalloc((float **)&g_idata, nBytes));
    CHECK(cudaMalloc((float **)&g_odata, nBytes));

    CHECK(cudaMemcpy(g_idata, data, nBytes, cudaMemcpyHostToDevice));

    double host_start = seconds();
    host_total = recursiveReduce(data, size);
    double host_elaps = seconds() - host_start;
    printf("Total host : %f, Time host : %f sec\n", host_total, host_elaps);

    // Launch add() kernel on GPU
    dim3 blockSize(BLOCK_SIZE, 1, 1);
    unsigned int gridWith = (size + blockSize.x - 1) / blockSize.x;
    printf("gridWith %d\n", gridWith);
    dim3 gridSize(gridWith, 1, 1);

    // output each block in gpu
    float *odata;
    odata = (float *)malloc(gridWith * sizeof(float));

    double device_start = seconds();
```

```
    reduceNeighboredLess<<<gridSize, blockSize>>>(g_idata, g_odata, size);
    CHECK(cudaDeviceSynchronize());

    // Copy result back to host
    CHECK(cudaMemcpy(odata, g_odata, gridWith * sizeof(float), cudaMemcpyDevice
ToHost));
    printFirst10(odata, gridWith);
    for (int i = 0; i < gridWith; i++){
        device_total += odata[i];
    }
    double device_elaps = seconds() - device_start;
    printf("Total device %f, Time device : %f sec\n", device_total, device_elap
s);

    // Cleanup
    cudaFree(g_idata);
    cudaFree(g_odata);

    free(data);
    free(odata);

    return 0;
}
```

Kết quả :

```
176 }

0.840188
0.394383
0.783099
0.798440
Total host : 2.816110, Time host : 0.000000 sec
gridWith 1
2.816110
Total device 2.816110, Time device : 0.000040 sec
```

Với BLOCK SIZE = 64

```
0.840188
0.394383
0.783099
0.798440
0.911647
0.197551
0.335223
0.768230
0.277775
0.553970
Total host : 16330.581055, Time host : 0.000092 sec
gridWith 512
33.225426
35.780441
30.649490
34.197861
34.362057
31.503387
28.800552
32.531731
34.956345
36.892906
Total device 16330.581055, Time device : 0.000061 sec
```

Với BLOCK SIZE = 1024

```
0.840188
0.394383
0.783099
0.798440
0.911647
0.197551
0.335223
0.768230
0.277775
0.553970
Total host : 33559940.000000, Time host : 0.200697 sec
gridWith 65553
518.912537
507.985718
507.931274
514.882690
506.994385
515.319702
498.363617
518.969482
499.768463
496.252075
Total device 33563996.000000, Time device : 0.004759 sec
```

2 Báo cáo và rút ra nhận xét, kết luận

* Vì xử lý trên số thực nên khi số lượng tính toán càng lớn có thể sẽ dẫn đến sai số

* Có thể thấy với khối lượng tính toán càng lớn thì GPU xử lý nhanh hơn rất nhiều so với ho st (CPU)

* Khi tăng số block size, và thay đổi tương ứng với grid size thì tốc độ tính toán của GPU tă ng nhanh hơn