Lập trình song song trên GPU

Hoàng Minh Thanh (18424062)

Bài tập 6: Bộ nhớ trong CUDA

Bộ môn Công nghệ phần mềm

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

Contents

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/1ldRlVbebOm2qs6xkw-cPTCYjnAO7-7ja

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

Quá trình cài đặt

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

1) Hàm tích chập trên GMEM

Ta khai báo trước một filter ở GMEM

```
__device__ float d_gflt[NF];
```

Hàm cài đặt

```
// ################ Device(GPU) ################
__global__ void convUseGMEM(float *d_in, float *d_out){
    int i = blockIdx.x * blockDim.x + threadIdx.x;

    if (i < NO){
        d_out[i] = 0; // use all global memory
        for (int j = 0; j < NF; j++){
            d_out[i] += d_gflt[j] * d_in[i + j]; // d_gflt for GMEM
        }
    }
}
```

Chạy

```
// ################################################
// convUseGMEM
iStart = seconds();
convUseGMEM<<<gridSize, blockSize>>>(d_in, d_out);
iElaps = seconds() - iStart;
printf("convUseGMEM : %f sec\n", iElaps);
// Copy results from device memory to host memory
cudaMemcpy(deviceRes, d_out, NO * sizeof(float), cudaMemcpyDeviceToHost);
checkResult(hostRes, deviceRes, NO);
```

2) Hàm tính tích chập ở CMEM:

Ta khai bao báo biến filter ở CMEM

```
#define NF 100
#define NI (1<<24)
#define NO (NI - NF + 1)
__constant__ float d_flt[NF];
```

```
__global__ void convUseCMEM(float *d_in, float *d_out){
    int i = blockIdx.x * blockDim.x + threadIdx.x;

    if (i < NO){
        d_out[i] = 0;
        for (int j = 0; j < NF; j++){
            d_out[i] += d_flt[j] * d_in[i + j]; // d_flt for RMEM
        }
    }
}
```

Chạy

```
// ###############################################
// convUseCMEM
iStart = seconds();
convUseCMEM<<<gridSize, blockSize>>>(d_in, d_out);
iElaps = seconds() - iStart;
printf("convUseCMEM : %f sec\n", iElaps);
cudaMemcpy(deviceRes, d_out, NO * sizeof(float), cudaMemcpyDeviceToHost);
checkResult(hostRes, deviceRes, NO);
```

3) Hàm tính tích chập sử dụng RMEM và GMEM

Đã khai báo ở trên nên ta chỉ việc dùng lại

Hàm

```cpp
__global__ void convUseRMEMAndGMEM(float *d_in, float *d_out){
    int i = blockIdx.x * blockDim.x + threadIdx.x;

    if (i < NO){
        float s = 0; // s for RMEM
        for (int j = 0; j < NF; j++){
            s += d_gflt[j] * d_in[i + j]; // d_gflt for GMEM
        }
        d_out[i] = s;
    }
}
```

Chạy

```cpp
    // ##############################################
    // convUseRMEMAndGMEM
    iStart = seconds();
    convUseRMEMAndGMEM<<<gridSize, blockSize>>>(d_in, d_out);
    iElaps = seconds() - iStart;
    printf("convUseRMEMAndGMEM : %f sec\n", iElaps);
    cudaMemcpy(deviceRes, d_out, NO * sizeof(float), cudaMemcpyDeviceToHost);
    checkResult(hostRes, deviceRes, NO);
```

Toàn bộ kết quả đo thời gian hàm tích chập

convOnHost : 5.683722 sec

convUseGMEM : 0.000045 sec

convUseCMEM : 0.000055 sec

convUseRMEMAndGMEM : 0.000042 sec

4) Hàm Reduction sử dụng SMEM:

Cài đặt

```cpp
__global__ void reduceUseSMEM(int *in, int *out, int n){
    // Each block loads data from GMEM to SMEM
    __shared__ int blkData[2 * 256];
    int numElemsBeforeBlk = blockIdx.x * blockDim.x * 2;
    blkData[threadIdx.x] = in[numElemsBeforeBlk + threadIdx.x];
    blkData[blockDim.x + threadIdx.x] = in[numElemsBeforeBlk + blockDim.x + threadIdx.x];
    __syncthreads();

    // Each block does reduction with data on SMEM
    for (int stride = blockDim.x; stride > 0; stride /= 2){
        if (threadIdx.x < stride){
            blkData[threadIdx.x] += blkData[threadIdx.x + stride];
        }
        __syncthreads(); // Synchronize within threadblock
    }

    // Each block writes result from SMEM to GMEM
    if (threadIdx.x == 0)
        out[blockIdx.x] = blkData[0];
}
```

Chạy và kiểm tra

```cpp
// ########################################
iStart = seconds();
reduceUseSMEM<<<gridSize, blockSize>>>(d_in, d_out, N);
CHECK(cudaMemcpy(out, d_out, gridSize.x * sizeof(int), cudaMemcpyDeviceToHost));

for (int i = 0; i < gridSize.x; i++){
    deviceRes += out[i];
}
iElaps = seconds() - iStart;
printf("reduceUseSMEM : %f sec\n", iElaps);
if (hostRes != deviceRes){
    printf("%d != %d", hostRes, deviceRes);
}else{
    printf("%d == %d", hostRes, deviceRes);
}
```

Kết quả chạy

```
reduceOnHost : 0.052101 sec
reduceUseSMEM : 0.000647 sec
1808906555 == 1808906555
```