**Lập trình song song trên GPU**

**Hoàng Minh Thanh (18424062)**



**TH2 : Reduction**

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

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](#_Toc46394330)

[1) Hoàn thành phần tính “gridSize” và hàm kernel 1 trong file “bt2.cu” 3](#_Toc46394331)

[2) Chạy chương trình ở câu 1 với các kích thước block khác nhau: 1024, 512, 256, 128. 5](#_Toc46394332)

[3) Hoàn thành hàm kernel 2 và 3 trong file “bt2.cu”. Trong file bài làm, bạn ghi nhận lại kết quả chạy 6](#_Toc46394333)

[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](#_Toc46394334)

[1. Kernel1 8](#_Toc46394335)

[2. Kernel2 8](#_Toc46394336)

[3. Kernel3 9](#_Toc46394337)

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>

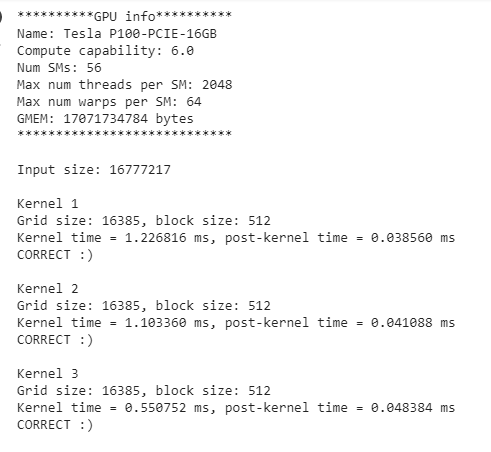
(Thầy có thể vào link Online để xem luồng chạy dễ hơn báo cáo )

# Quá trình cài đặt

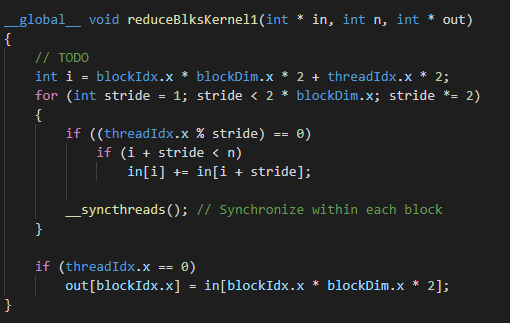
Chi tiết cài đặt trong file .cu và file Google Colab

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

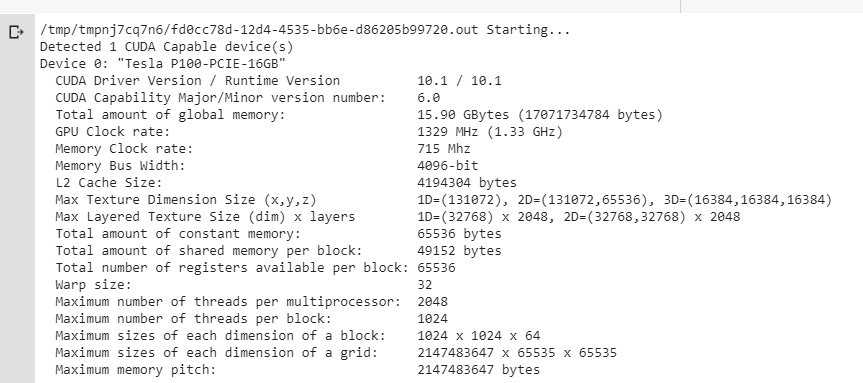
## Hoàn thành phần tính “gridSize” và hàm kernel 1 trong file “bt2.cu”



Kết quả cài đặt

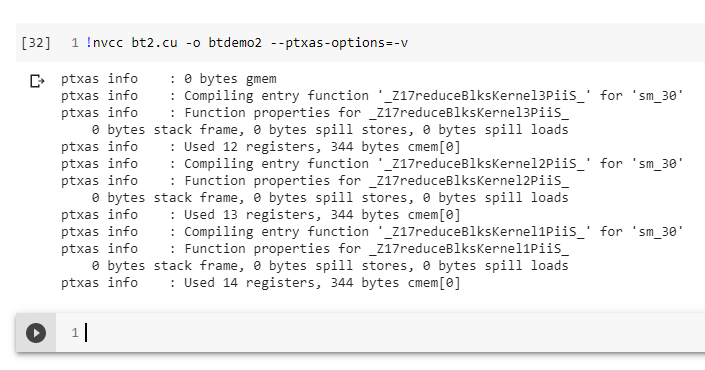


**Ta xâu dựng hàm lấy thông tin info phần cứng**

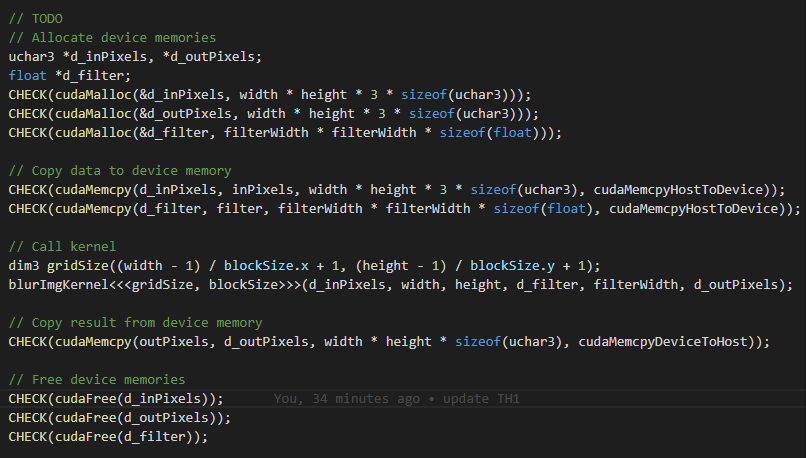
****

## 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



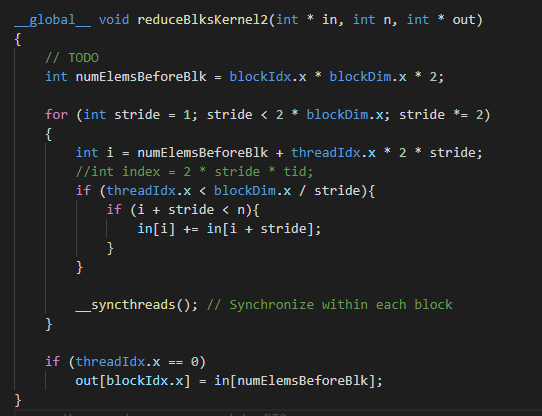
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Block  Size | Grid  Size | Occupancy | Num blocks / SM | Kernel time | Post-kernel time | Total time |
| 1024 | 8193 | 100% | 2 | 1.440032 | 0.020352 | 1.460384 |
| 512 | 16385 | 100% | 4 | 1.225440 | 0.041568 | 1.267008 |
| 256 | 32769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

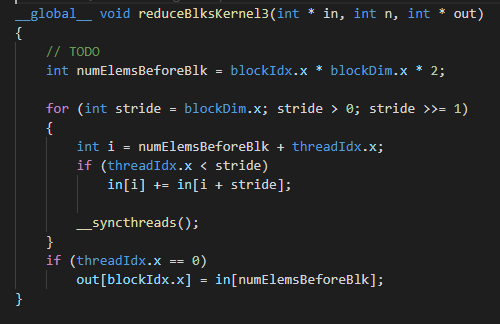


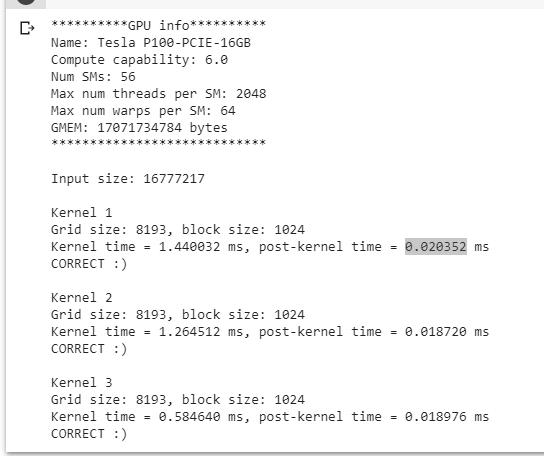
Khi thay đổi blockSize thì thứ tự index bị chồng chéo nhau và dẫn đến thay đổi tốc độ truy cập bộ nhớ

## 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







## 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