

Lập trình song song trên GPU

BT2: Cách thực thi song song trong CUDA



Nên nhớ mục tiêu chính ở đây là **học, học một cách chân thật**. Bạn có thể thảo luận ý tưởng với bạn khác, nhưng **bài làm phải là của bạn, dựa trên sự hiểu thật sự của bạn**. **Nếu vi phạm thì sẽ bị 0 điểm cho toàn bộ môn học**.

Đề bài

Trong bài này, bạn sẽ áp dụng những hiểu biết về cách thực thi song song trong CUDA để viết chương trình giải quyết bài toán “reduction” (cụ thể là, ta sẽ tính tổng của một mảng số nguyên). Mình đã viết sẵn cho bạn khung chương trình trong file “bt2.cu” đính kèm; bạn chỉ viết code ở những chỗ có từ “// TODO”:

- Tính “gridSize” (từ “blockSize” và “n”).
- Hàm kernel 1, 2, và 3 (bạn xem nội dung cụ thể của các hàm kernel này trong slide “05_06-CachThucThiTrongCUDA_P2.pdf”; ở đây, để đơn giản, ta giả định $2 * \text{kích-thước-block} = 2^k$ với k là một số nguyên dương nào đó). Hàm kernel sẽ tính giá trị tổng cục bộ trong từng block; sau đó, host sẽ chép các giá trị tổng cục bộ này về bộ nhớ của mình và cộng hết lại để ra giá trị tổng toàn cục.

Với mỗi hàm kernel, chương trình sẽ in ra:

- Kích thước grid và kích thước block.
- Thời gian chạy của hàm kernel (“kernel time”) và thời gian host thực hiện cộng các giá trị tổng cục bộ của các block (“post-kernel time”).
- “CORRECT” nếu kết quả tính được giống với kết quả đúng, “INCORRECT” nếu ngược lại.

Hướng dẫn về các câu lệnh:

- Biên dịch file “bt2.cu”: `nvcc bt2.cu -o bt2`
- Chạy file “bt2”: `./bt2`
Mặc định thì sẽ dùng block có kích thước 512; nếu bạn muốn dùng block có kích thước khác, chẳng hạn 256, thì bạn truyền thêm tham số dòng lệnh: `./bt2 256`

Cụ thể, các yêu cầu của bài tập này như sau:

1. Hoàn thành phần tính “gridSize” và hàm kernel 1 trong file “bt3.cu”. Trong file bài làm, bạn ghi nhận lại kết quả chạy; ví dụ:

```

ttkien@fitcuda1:~/[HC]BT2$ ./bt2
*****GPU info*****
Name: TITAN Xp
Compute capability: 6.1
Num SMs: 30
Max num threads per SM: 2048
Max num warps per SM: 64
GMEM: 12788498432 bytes
*****


Input size: 16777217

Kernel 1
Grid size: 16385, block size: 512
Kernel time = 1.102656 ms, post-kernel time = 0.079968 ms
CORRECT :)

Kernel 2
Grid size: 16385, block size: 512
Kernel time = 0.294944 ms, post-kernel time = 0.079680 ms
INCORRECT :(

Kernel 3
Grid size: 16385, block size: 512
Kernel time = 0.116224 ms, post-kernel time = 0.145952 ms
INCORRECT :(

```

(Để chụp lại một phần màn hình, trong Windows 10, bạn có thể ấn +Shift+S, rồi ấn giữ chuột trái và kéo chọn vùng cần chụp, rồi Ctrl+V vào file word là xong.)

- Chạy chương trình ở câu 1 (ta đang chỉ tập trung vào hàm kernel 1) với các kích thước block khác nhau: 1024, 512, 256, 128. Với mỗi kích thước block, ngoài việc ghi nhận lại thời gian chạy, bạn cũng cần tính độ đo occupancy, số block / SM. Cụ thể, bạn điền các kết quả theo mẫu bảng biểu bên dưới (trong đó, Total time = Kernel time + Post-kernel time); với occupancy và số block / SM, bạn cần giải thích thêm là tại sao lại tính ra được các giá trị như vậy. Khi tính occupancy, tạm thời ta chỉ xét 2 ràng buộc của SM là số block tối đa và số thread tối đa; bạn có thể tra cứu 2 ràng buộc này ở [document của CUDA](#), mục “Programming Guide” (“Guide” **không có s**), mục “H. Compute Capabilities”, bảng “Table 14. Technical Specifications per Compute Capability”), 2 ràng buộc đó là “Maximum number of resident blocks per multiprocessor” và “Maximum number of resident threads per multiprocessor”.

| Block size | Grid size | Occupancy (%) | Num blocks / SM | Kernel time (ms) | Post-kernel time (ms) | Total time (ms) |
|------------|-----------|---------------|-----------------|------------------|-----------------------|-----------------|
| 1024 | 8193 | | | | | |
| 512 | 16385 | | | | | |
| 256 | 32769 | | | | | |
| 128 | 65537 | | | | | |

Giải thích tại sao khi thay đổi block size thì “kernel time” và “post-kernel time” lại thay đổi như vậy?

3. Hoàn thành hàm kernel 2 và 3 trong file “bt2.cu”. Trong file bài làm, bạn ghi nhận lại kết quả chạy (tương tự câu 1; để kích thước block là 512).
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).

Nộp bài

Trong thư mục <MSSV> (vd, nếu bạn có MSSV là 1234567 thì bạn đặt tên thư mục là 1234567), bạn để:

- File “bt2.cu” ứng với code của câu 1 và 3.
- File bài làm “bt2.pdf” (ở header của file bạn ghi họ tên và MSSV).

Sau đó, bạn nén thư mục này lại và nộp ở link trên moodle.