Lập trình song song trên GPU
BT04: Song song hóa Radix Sort

Cập nhật 24/12/2020

Level 4!

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.

Trong môn học, để thống nhất, tất cả các bạn (cho dù máy bạn có GPU) đều phải dùng Google Colab để biên dịch và chạy code (khi chấm Thầy cũng sẽ dùng Colab để chấm). Với mỗi bài tập, bạn thường sẽ phải nộp:

1) **File code** (file .cu)
2) **File báo cáo** là file notebook (file .ipynb) của Colab (nếu bạn nào biết Jupyter Notebook thì bạn thấy Jupter Notebook và Colab khá tương tự nhau, nhưng 2 cái này hiện chưa tương thích 100% với nhau: file .ipynb viết bằng Jupyter Notebook có thể sẽ bị mất một số cell khi mở bằng Colab và ngược lại). File này sẽ chứa các kết quả chạy. Ngoài ra, một số bài tập có phần viết (ví dụ, yêu cầu bạn nhận xét về kết quả chạy), và bạn sẽ viết trong file notebook của Colab luôn. Colab có 2 loại cell: code cell và text cell. Ở code cell, bạn có thể chạy các câu lệnh giống như trên terminal của Linux bằng cách thêm dấu ! ở đầu. Ở text cell, bạn có thể soạn thảo văn bản theo cú pháp của Markdown (rất dễ học, bạn có thể xem ở đây); như vậy, bạn sẽ dùng text cell để làm phần viết trong các bài tập. Bạn có thể xem về cách thêm code/text cell và các thao tác cơ bản ở đây, mục "Cells" (đừng đi qua mục "Working with Python"). Một phím tắt ưa thích của mình khi làm với Colab là ctrl+shift+p để có thể search các câu lệnh của Colab (nếu câu lệnh có phím tắt thì bên cạnh kết quả search sẽ có phím tắt). File notebook trên Colab sẽ được lưu vào Google Drive của bạn; bạn cũng có thể download trực tiếp xuống bằng cách ấn ctrl+shift+p, rồi gõ "download .ipynb".

Đề bài

Song song hóa Radix Sort, trong đó bước scan viết trong một hàm kernel duy nhất (xem file slide "13-Sort.pdf").

Code (8đ)

Mình đã viết sẵn khung chương trình trong file bt04.cu đính kèm. Bạn sẽ cần phải viết code ở những chỗ mình để // TODO.

Một số gợi ý:

- Bạn nên bắt đầu với file "12-Scan.cu" trước, và đảm bảo là đã cài đặt đúng phần scan trong một hàm kernel duy nhất (exclusive scan nha)

- Khi cài đặt song song Radix Sort, bạn nên bắt đầu với cài đặt tuần tự và lần lượt song song hóa từng bước (rút trích bit, scan, …); bằng cách này, nếu xảy ra lỗi thì bạn sẽ biết ngay lỗi ở bước nào
- Ở mỗi vòng lặp của Radix Sort, khi song song hóa bước scan, bạn nhớ reset lại cho 2 biến bCount và bCount1 (bạn có thể dùng hàm cudaMemcpyToSymbol)
- Ý nhỏ: khi gọi hàm kernel mà gridSize và blockSize chỉ có một chiều thì bạn có thể đơn giản là truyền vào 2 con số; ví dụ, myKenel<<<10, 1024>>>(…)

Hướng dẫn về các câu lệnh:

(Các câu lệnh dưới đây là chạy trên terminal của Linux, khi chạy ở code cell của Colab thì bạn thêm dấu ! ở đầu)

- Biên dịch file bt04.cu: nvcc bt04.cu -o bt04
- Chạy file bt04: ./bt04
  Mặc định thì chương trình sẽ dùng block có kích thước 512; nếu bạn muốn chỉ định kích thước block thì truyền thêm vào câu lệnh một con số ứng với kích thước block (ví dụ, ./bt04 256).

Báo cáo (2đ)

Trong file "bt04.ipynb" mà mình đính kèm:

- Bạn biên dịch và chạy file "bt04.cu" với các kích thước block là 256, 512, 1024.
- Giải thích tại sao khi thay đổi kích thước block thì kết quả lại thay đổi như vậy. Chỗ nào mà bạn không biết tại sao thì cứ nói là không biết tại sao.

Nộp bài

Bạn tổ chức thư mục bài nộp như sau:

Thư mục <MSSV> (vd, nếu bạn có MSSV là 1234567 thì bạn đặt tên thư mục là 1234567)
- File code "bt04.cu"
- File báo cáo "bt04.ipynb"

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