Lập trình song song trên GPU
Đồ án cuối kỳ: tối ưu hóa Radix Sort trên GPU

Đề bài

Tối ưu hóa Radix Sort trên GPU, dựa trên ý tưởng trong bài báo này của NVIDIA (xem ý tưởng trong slide có thể sẽ dễ hiểu hơn). Nhóm bạn có thể bắt đầu từ file code "11-sort-sol.cu" mà mình đã upload trên moodle (bạn lấy bản mới nhất, ở đầu file có dòng "Update 2020/09/10").

Yêu cầu của đồ án:

Làm việc nhóm. Tất cả các thành viên trong nhóm đều phải hiểu về đồ án mà nhóm mình làm (tất nhiên là phải hiểu cả code). Ví dụ, nếu trong nhóm có một bạn hiểu – được 10 điểm – và có một bạn không hiểu – được 0 điểm – thì điểm chung cho cả hai sẽ vào khoảng ở giữa là 5 điểm. Nhóm bạn sẽ cần nộp lại file làm việc nhóm.

Chạy đúng và tổng quát hóa. Chương trình của nhóm bạn phải chạy đúng với các block size khác nhau; nhóm bạn được phép giả định: (i) block size = 2^k với k là số nguyên dương, và (ii) block size không quá nhỏ để làm cho số lượng block vượt qua mức tối đa. Chương trình của bạn cũng phải chạy đúng với k = 1, 2, 4, 8 (16 sẽ làm số lượng bin lớn và SMEM sẽ không chứa nổi nên thôi).

Thời gian chạy. Thời gian chạy phải tương đương với thời gian chạy của thuật toán sort song song trong thư viện Thrust thì mới có thể được 10 điểm.

Slide báo cáo. Trong slide báo cáo, nhóm bạn sẽ trình bày những nội dung sau:

- Quá trình tối ưu hóa (đi từ tuần tự đi lên, đi từ đơn giản đi lên):
    - 4 phiên bản baseline:
        - Cài đặt tuần tự thuật toán Radix Sort tuần tự (đã có trong file "11-sort-sol.cu").
        - Cài đặt song song 2 bước histogram và scan của thuật toán Radix Sort tuần tự.
        - Cài đặt song song thuật toán Radix Sort với k = 1 bit (trang 19-21 trong file slide "11-SapXep.pdf").
        - Cài đặt tuần tự ý tưởng của thuật toán Radix Sort song song trong bài báo của NVIDIA (đây là phiên bản mà lúc sau ta sẽ tập trung song song hóa và tối ưu hóa).

      Nói rõ code của các phiên bản cài đặt này là file nào; chụp ảnh kết quả chạy để cho thấy là đã chạy đúng; dùng bảng biểu để tổng hợp lại thời gian chạy của các phiên bản (thời gian chạy có thể thay đổi một ít từ lần chạy này qua lần chạy khác → để chính xác hơn, bạn nên chạy nhiều lần rồi lấy trung bình lại; ngoài ra, nếu bạn chạy trên Colab thì mỗi một lần kết nối vào có thể sẽ được cấp phát một GPU có cấu hình khác → để có thể so sánh các kết quả một cách công bằng, ở một lần kết nối mới và chạy thử một phiên bản mới, bạn cũng cần chạy lại các phiên bản trước). Lúc này, bảng biểu có các dòng sau:
        - Dòng header

        - Các dòng cho biết thời gian chạy của các phiên bản baseline - là điểm bắt đầu. Với phiên bản được highlight ở trên, ngoài thời gian tổng, bạn cũng cần đo thời gian chạy của các đoạn code thành phần vì đây là phiên bản mà bạn sẽ tập trung tối ưu hóa (xem "bước 1 - phân tích" ở dưới).
        - Dòng cho biết thời gian chạy của thuật toán sort song song trong Thrust - là đích hướng đến.
    - Lặp (mỗi vòng lặp ứng với việc tạo ra một phiên bản mới từ các phiên bản trước đó; cái ta muốn làm ở đây là song song hóa và tối ưu hóa phiên bản được highlight ở trên):
        - Bước 1 - Phân tích. Đây là bước quan trọng vì bước này trả lời cho câu hỏi phiên bản mới được ra đời như thế nào từ các phiên bản trước. Ở bước này, đầu tiên, xem thử nên ưu tiên tối ưu hóa các đoạn code nào trong phiên bản trước (bằng cách đo thời gian chạy của các đoạn code để xác định các đoạn code ngốn nhiều thời gian nhất)? Sau đó, xem thử có thể tối ưu hóa các đoạn code này như thế nào (ở mức ý tưởng)?
        - Bước 2 - Thiết kế. Triển khai cụ thể hơn ý tưởng tối ưu hóa ở bước 1.
        - Bước 3 - Cài đặt. Cài đặt thiết kế ở bước 2. Nói rõ code của phiên bản cài đặt này là file nào; chụp ảnh kết quả chạy để cho thấy là đã chạy đúng; chèn thêm vào bảng biểu trước đó dòng ghi nhận thời gian chạy của từng đoạn code và thời gian tổng (chèn trước dòng của Thrust); nhận xét về kết quả chạy so với phân tích ban đầu.
- Cuối cùng, với phiên bản tốt nhất, nhóm bạn cũng cần chụp lại kết quả chạy với các block size khác nhau và với các k khác nhau để cho thấy là đã chạy đúng và đã tổng quát hóa.

Các file code. Cần lưu lại tất cả các phiên bản (tương ứng với các phiên bản được trình bày trong file slide).

Vấn đáp và nộp bài trên moodle

Vấn đáp

- Thời gian vấn đáp: vào cuối thời gian thi của trường, cụ thể thì sẽ thông báo sau.
- Online hay offline? Sẽ thông báo sau.
- Khi vấn đáp, mỗi nhóm chuẩn bị: laptop, file làm việc nhóm, slide, code.

Nộp bài trên moodle

Nhóm bạn sẽ nộp bài (gồm file làm việc nhóm + slide + code) trên moodle sau buổi vấn đáp.