

Lập trình song song trên GPU

BT03: Các loại bộ nhớ trong CUDA (phần 1)



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

Viết chương trình làm mờ ảnh RGB (cách làm mờ giống như ở BT01).

Mình có đính kèm file ảnh đầu vào [in.pnm](#) (trong Windows, bạn có thể xem file *.pnm bằng chương trình [IrfanView](#)). Mình cũng đã viết sẵn khung chương trình trong file [bt03.cu](#) đính kèm. Bạn sẽ cần phải viết code ở những chỗ mình để `// TODO`:

- Định nghĩa hàm kernel [blurImgKernel1](#) – hàm kernel làm mờ ảnh ở BT01. Bạn nào đã làm BT01 rồi thì chỉ cần copy-paste (code lại một lần nữa cũng tốt); bạn nào chưa làm BT01 thì đầu tiên bạn phải hoàn thành hàm kernel này. Đây là hàm kernel cơ bản nhất; nếu bạn chưa cài đặt được hàm kernel này thì không thể qua hàm kernel 2 và 3 (sẽ được trình bày ở dưới) được. Nếu cần thì bạn có thể tham khảo code cài đặt tuần tự ở hàm [blurImg](#) (ở `if (useDevice == false)`).
- Gọi hàm [blurImgKernel1](#).
-
- Định nghĩa hàm kernel [blurImgKernel2](#) – hàm kernel làm mờ ảnh **có sử dụng SMEM**. Mỗi block sẽ đọc phần dữ liệu của mình từ [inPixels](#) ở GMEM vào SMEM, sau đó phần dữ liệu ở SMEM này sẽ được **dùng lại nhiều lần** cho các thread trong block. Bạn sẽ **cấp phát động** một mảng ở SMEM của mỗi block để cho phép kích thước của mảng này có thể thay đổi theo [filterWidth](#) và [blockSize](#):
 - Trong hàm kernel [blurImgKernel2](#), bạn khai báo mảng [s_inPixels](#) ở SMEM như sau:
`extern __shared__ uchar3 s_inPixels[];`
 - Khi gọi hàm kernel [blurImgKernel2](#), trong cặp `<<<...>>>`, ngoài tham số [gridSize](#) và [blockSize](#), bạn sẽ truyền vào **tham số thứ ba** cho biết kích thước (byte) của mảng [s_inPixels](#) trong SMEM của mỗi block (kích thước này được tính theo biến [filterWidth](#) và biến [blockSize](#), ở đây ta cũng lưu mảng 2 chiều dưới dạng mảng 1 chiều).
- Gọi hàm [blurImgKernel2](#).
-
- Hàm kernel [blurImgKernel3](#) (hàm kernel làm mờ ảnh có sử dụng SMEM cho [inPixels](#) và **sử dụng CMEM cho filter**) và các phần liên quan: tạm để lại tuần sau.

Hướng dẫn về các câu lệnh:

(Phần hướng dẫn dưới đây là cho Linux, nếu bạn nào dùng GPU cá nhân trên Windows thì cũng tương tự, chỉ khác là: file chạy sau khi biên dịch có đuôi exe, và khi chạy file này thì dùng `.\` thay vì `./`)

- Biên dịch file [bt03.cu](#): `nvcc bt03.cu -o bt03`

- Chạy file `bt03` với file ảnh đầu vào là `in.pnm`, file ảnh đầu ra là `out.pnm`:

`./bt03 in.pnm out.pnm`

Lúc này, chương trình sẽ: đọc file ảnh `in.pnm`; làm mờ ảnh bằng host (để có kết quả đúng làm chuẩn), và làm mờ ảnh bằng device với 3 phiên bản của hàm kernel: `blurImgKernel1`, `blurImgKernel2`, và `blurImgKernel3`; các kết quả sẽ được ghi xuống 4 file: `out_host.pnm`, `out_device1.pnm`, `out_device2.pnm`, và `out_device3.pnm`. Với mỗi hàm kernel, chương trình sẽ in ra màn hình thời gian chạy và sự sai biệt so với kết quả của host (mình chạy thì thấy kết quả của device có sự sai biệt **nhỏ** so với kết quả của host, khoảng 0.000x; đó là do GPU tính toán số thực có thể hơi khác so với CPU, chứ không phải là do cài đặt sai).

Mặc định thì chương trình sẽ dùng block có kích thước 32×32 ; nếu bạn muốn chỉ định kích thước block thì truyền thêm vào câu lệnh 2 con số lần lượt ứng với kích thước theo chiều x và theo chiều y của block (ví dụ, `./bt03 in.pnm out.pnm 32 16`).

Nộp bài

Ở buổi sau, sẽ có phần 2. Khi đó, bạn sẽ nộp một lần cả phần 1 lẫn phần 2. Nhưng bạn nên hoàn thành phần 1 trong tuần này.