IF3230 Sistem Paralel dan Terdistribusi
Praktikum 4 Pengenalan dan Penggunaan CUDA - Quicksort

A. Persiapan Praktikum

Pada praktikum kali ini anda ditugaskan untuk menggunakan CUDA, perkakas untuk melakukan pemrograman secara paralel yang berbasiskan GPGPU (General-Purpose GPU). Server yang dapat dipakai untuk menggunakan CUDA adalah 167.205.35.25 - 167.205.35.33 (Kecuali 35.27). Anda dapat mengakses server menggunakan username NIM dan password 'guest'. Adapun untuk pembagian tiap-tiap server bagi praktikan pada praktikum kali ini adalah sebagai berikut.

1. 167.205.35.25 : NIM **13514001 – 13514015**
2. 167.205.35.26 : NIM **13514016 – 13514030**
3. 167.205.35.28 : NIM **13514031 – 13513045**
4. 167.205.35.29 : NIM **13514046 – 13514060**
5. 167.205.35.30 : NIM **13514061 – 13514075**
6. 167.205.35.31 : NIM **13514076 – 13514090**
7. 167.205.35.32 : NIM **13514091 – 13514105**
8. 167.205.35.33 : NIM > **13514106, 13513xxx, 135156xx, dll.**

Sebelum dapat memakai CUDA pada praktikum kali ini, terdapat beberapa tahapan yang harus dilakukan oleh peserta terlebih dahulu. Berikut adalah tahapan setup CUDA.

1. Set PATH
   `export PATH=/opt/cuda-5.5/bin:$PATH`
2. Set LD_LIBRARY_PATH
   `export LD_LIBRARY_PATH=/opt/cuda-5.5/lib64:$LD_LIBRARY_PATH`

Anda dapat juga membuat file .bashrc yang berisi kedua perintah di atas. Namun file tersebut tidak akan dijalankan secara otomatis setiap kali anda login. Sehingga anda harus menjalankannya secara manual dengan menggunakan perintah "`. .bashrc`" tanpa tanda kutip.

B. Pengenalan CUDA

CUDA adalah standar yang dibuat oleh Nvidia untuk melakukan pemrograman GPU pada Graphic Card Nvidia. Penjelasan CUDA yang lebih lengkap dapat dilihat di slide yang diunggah di gitlab. Sebelum membahas lebih lanjut mengenai pemrograman CUDA, berikut ini adalah istilah-istilah dalam pemrograman CUDA:

1. Host: CPU
2. Device: GPU
3. Kernel: Kode yang berjalan di atas GPU. Satu GPU hanya dapat menjalankan satu kernal pada satu saat.
4. Thread: Satuan eksekusi pada GPU. Terdapat banyak thread yang menjalankan kernel secara bersamaan.
5. Block: Kumpulan Thread. Block merupakan satuan sinkronisasi eksekusi. Satu blok tidak dapat berkoordinasi dengan blok lainnya.
6. Grid: Kumpulan Block.

Eksekusi Kernel

Sebuah kernel akan dijalankan oleh sejumlah thread. Setiap thread akan menjalankan kernel yang sama. Setiap thread akan mendapatkan ID unik yang dapat dipakai untuk menentukan alur eksekusi thread. Thread terkelompok menjadi satuan yang disebut *block*. Thread yang berada pada *satu* block yang sama dapat melakukan koordinasi yang lebih lanjut seperti melakukan *sharing memory* dan sinkronisasi.

Arsitektur Memory dan Hardware

Setiap thread memiliki akses kepada *register*, *shared memory*, dan *device memory*. Setiap thread processor memiliki register, register tersebut *hanya dapat* diakses oleh thread processor tersebut. Setiap block memiliki shared memory yang dapat diakses oleh setiap thread pada blok tersebut. Device memory dapat diakses oleh *semua* thread pada blok manapun. Secara hardware thread berjalan di *thread processor*. Block berjalan di atas *stream multiprocessor*. Satu stream multiprocessor (SM) dapat menjalankan banyak block sekaligus. Hal ini dapat memengaruhi ukuran shared memory yang dapat digunakan oleh setiap block.

Pemrograman CUDA

CUDA menambahkan beberapa *sintaks*, *built-in variables*, dan *runtime functions*:

1. *function type qualifier*: membedakan fungsi yang dieksekusi di host dan di device
   a. __device__: dieksekusi di device, dapat dipanggil melalui device. (fungsi internal yang digunakan oleh kernel).
   b. __global__: dieksekusi di device, hanya dapat dipanggil oleh host. (berfungsi sebagai titik awal eksekusi kernel di device).
   c. __host__: dieksekusi di host, hanya dapat dipanggil oleh host.
2. *variable type qualifier*: dipakai untuk mendeklarasikan sifat variable sehingga diletakkan di lokasi memory yang sesuai.
3. *built-in variables*: variable yang terdefinisi secara otomatis di saat runtime.
   a. gridDim: variable yang berisi dimensi dari grid.
   b. blockIdx: variable yang berisi index block dimana thread ini berada.
   c. blockDim: variable yang berisi dimensi dari block.
   d. threadIdx: variable yang berisi index thread di dalam block. (untuk membedakan thread yang berada di block yang berbeda, gunakan blockIdx).
4. parameter eksekusi kernel: kernel diseksekusi dengan memanggil fungsi __global__ dengan memberikan nilai `<<<grid, block>>>` tepat di belakang nama fungsi yang ingin dieksekusi (sebelum ‘(‘). `grid` dan `block` inilah yang akan menjadi gridDim dan blockDim di saat berjalannya kernel.
5. *runtime functions*: fungsi yang mengatur alur eksekusi fungsi. (contoh: __syncthreads()).

Eksekusi dan Contoh Program

Kompilasi program CUDA dapat dilakukan dengan menggunakan perintah ini

```
nvcc <file-name>.cu -o <executable-name>
```

Jalankan perintah di bawah ini untuk menjalankan program hasil kompilasi

```
./<executable-name>
```

Pada repository project telah diunggah beberapa contoh program yang dapat dijalankan, silahkan gunakan program tersebut sebagai acuan.

C. Persoalan, Deliverables, dan Pengumpulan

Anda diminta untuk mengimplementasikan permasalahan Quicksort paralel menggunakan CUDA. Adapun untuk implementasi sorting pada bagian lokal dapat mengacu pada kode yang telah diunggah di repository project. Tugas ini dikerjakan dalam kelompok 2 orang (boleh sama atau berbeda dengan tugas sebelumnya, tidak boleh lintas kelas).

Deliverables yang harus dikumpulkan pada program ini adalah source code program anda serta laporan singkat yang berisi:

1. Deskripsi solusi
2. Analisis solusi
3. Pengukuran performansi

Pengukuran performansi dilakukan pada bagian distribusi awal data ke tiap-tiap proses dan saat proses sorting dimulai hingga selesai. Adapun untuk kasus-kasus pengurutannya adalah sebagai berikut.

1. N elemen pada array dengan jumlah blok dan threads yang sesuai
    a. Nilai N: 50.000, 100.000, 200.000, 400.000

Mekanisme pengumpulan untuk tugas ini sama seperti praktikum-praktikum sebelumnya, yaitu sebagai berikut:

1. Lakukan fork pada repo di gitlab
2. Lakukan git clone (boleh di server)
3. Lakukan commit setelah selesai mengerjakan satu program
4. Lakukan merge request dari repo hasil fork anda di akhir praktikum untuk penilaian dengan judul **Praktikum4_<NIM1>_<NIM2>**
5. Batas waktu pengumpulan adalah **Senin, 6 Maret 2017, 20:00 WIB**
6. Segala bentuk kecurangan akan mendapatkan konsekuensi