Advanced GPU Computing
- CUDA
- CUBLAS

Presentation

Generations of CUDA cards

Some performance considerations

Plan of the course

References:

CUDA C Programming Guide - v3.2; NVIDIA.

CUDA C Best Practices Guide - v3.2; NVIDIA.

CUDA C Reference Manual - v3.2; NVIDIA.

CUBLAS Library - v3.2; NVIDIA.

Presentation

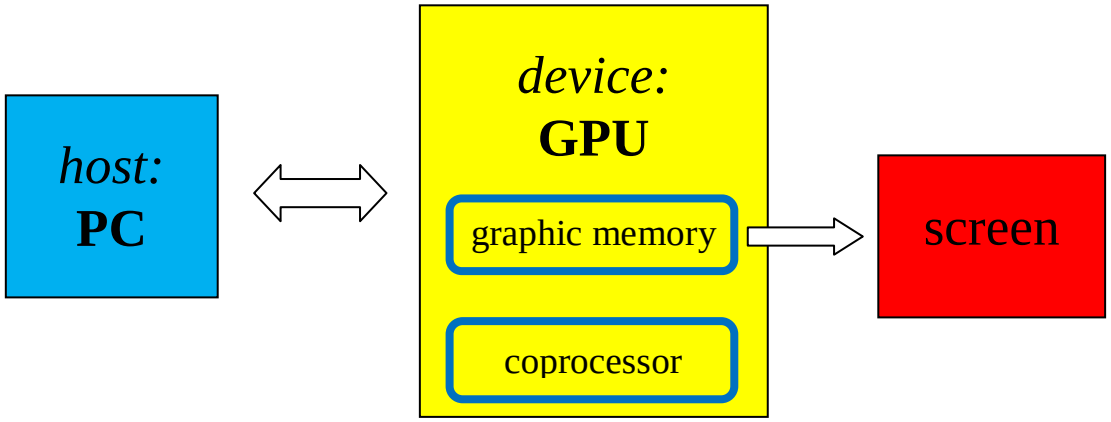
- **CUDA** = *Compute Unified Device Architecture*

- **GPU** = *Graphic Processing Unit* = graphic card = used traditionally for graphics

  → graphic memory to refresh the screen

  → parallel coprocessor for fast graphic calculations: hundreds of cores sharing a global memory

  idea: each core processes one pixel



- latest trend: **GPGPU** = *General Purpose Graphic Processing Unit*

  → for general parallel calculations, not only for graphics

- parallel system designed for *data-based parallelism*:

  → grid of pixels

  → matrix of values

  → grid of space elements for physical calculations

  →swarm of particles

  → . . . . . .

Generations of CUDA cards

history:

since 2006, several generations of CUDA cards

old generations are very primitive but can be found inside most PCs

only the latest generation is really suitable for general parallel calculations

→ the primitive graphic card has been gradually transformed into a supercomputer

labelling of a CUDA generation: *compute capability* = 1.x or 2.x

- *major revision number*: type of architecture, 1 or 2 (*Fermi architecture*, latest)
- *minor revision number*: improvements of a given type of architecture

most important restrictions for older generations:

- *printf* inside parallel code only for Fermi (2.x) → debugging...
- *clock()* inside parallel code only for Fermi (2.x) → performance measurement
- *cache memory* (L1, L2) only for Fermi (2.x)
- *double precision floating-point arithmetic* only for 1.3 and 2.x
- recursion of parallel code only for 2.x

Tesla C2050 and C2070:

- compute capability = 2.0 → Fermi architecture
- 448 cores
- size of global memory = 3GB for C2050, 6GB for C2070
- building block of the latest generation of supercomputers

Some performance considerations

PC with Core i7 hexacore 980x

number of cores	6
frequency	3.33 GHz
memory transfer	25.6 GB/s
TDP	130 W

→ *performance for matrix multiplication (basic C code for jacobian product):*

- sequential on just one core with other 5 cores idle: 1.4 Gflops
- sequential on just one core with other 5 cores busy: 0.38 Gflops
- parallel on 6 cores: ≈ 2.3 Gflops

→ speed up of at most 2.3 / 1.4 ≈ 1.7 on 6 cores...

= most likely due to *RAM bottleneck*

Note: possible to obtain better performance with *BLAS* library

Tesla C2050:

number of cores	448
frequency	1.15 GHz
memory transfer	144 GB/s
TDP	238 W
peak double precision floating point arithmetic	515 Gflops

→ *performance for matrix multiplication (jacobian product):*

- parallel on all 448 cores: 40 Gflops
- 15 times faster than Intel hexacore with about twice as much energy consumption

NVIDIA announces a peak performance of 515 Gflops. . .

→ use *CUBLAS* library to obtain about 400 Gflops ! ! !

Plan of the course

Introduction

CUDA architecture and programming model

CUDA programming interface - CUDA C

Optimization of a CUDA code

CUBLAS linear algebra library

Application: linear system resolution with Gauss method

Using OpenGL with CUDA

Application: all-pairs n-body problem