Course **IT00CG19-3002**

GPU Programming

Fall 2023, period 1, 5 ECTS

Agora XX

Mondays and Wednesdays 14.15-15.45

Slides available on moodle.abo.fi

Slide set #1: **Introduction**

Jan Westerholm

professor, high performance computing

office: Agora 346 I

Syllabus

- Course topic: GPU programming using CUDA for NVIDIA GPUs.
- Course objectives
- What is CUDA?
- Why Cuda? Why ***not*** Cuda?
- GPU parallel programming model
- GPU execution mechanism
- Course hardware and software environment
- How to pass the course: design a program, compile and run on an NVIDIA GPU and report your results!

Top500 Supercomputers in the World

- Twice a year top500.org updates a list of the “biggest” supercomputers in the world
- The latest list is from June 2023
  - https://top500.org/lists/top500/list/2023/06/
- The topmost machine, Frontier at Oak Ridge, Tennessee, has 606,208 AMD CPU cores and 37,888 AMD MI250X GPUs, achieves ≈ 1200 petaflops, roughly 10^{18} flop/s, ≈ 1 exaflops when running the *High Performance Linpack* benchmark, flop/s = floating point operations per second

Åbo Akademi University

Quick Reminder: Unit Prefixes

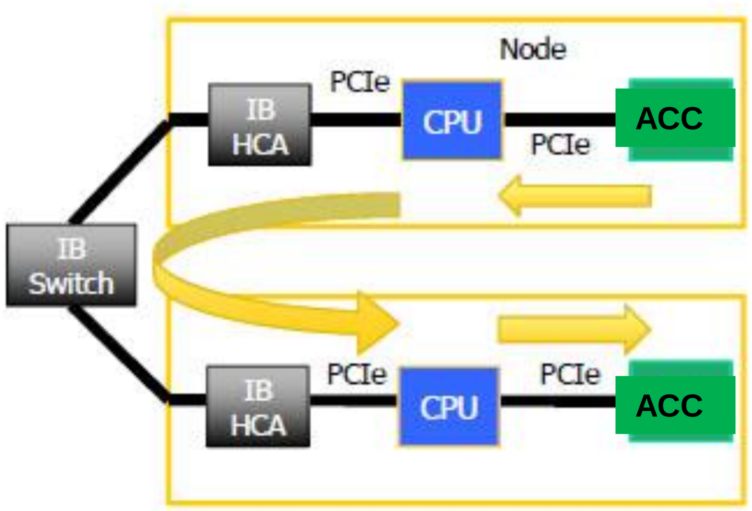
Prefix	Symbol	10^n	Decimal number	Name
yotta	Y	10^{24}	1 000 000 000 000 000 000 000 000	septillion
zetta	Z	10^{21}	1 000 000 000 000 000 000 000	sextillion
exa	E	10^{18}	1 000 000 000 000 000 000	quintillion
peta	P	10^{15}	1 000 000 000 000 000	quadrillion
tera	T	10^{12}	1 000 000 000 000	trillion
giga	G	10^{9}	1 000 000 000	billion
mega	M	10^{6}	1 000 000	million

Recent additions:

ronna	R	10^{27}
quetta	Q	10^{30}

Top500 Supercomputers in the World

- Top500 supercomputers are often based on **nodes** that contain **multicore** *host* CPUs running programs on *devices* / accelerators / **GPUs**
- For multi-node systems we coordinate tasks/data between nodes using e.g. Message Passing Interface MPI



Top500 Supercomputers in the World

- Supercomputer/cluster machine accelerators in use today have GPUs by NVIDIA or by AMD.
- The Finnish supercomputer center CSC is hosting a European 200 MEUR pre-exascale machine LUMI, performance is 0.4 exaflop/s with 2560 nodes, each node with one AMD Epyc 64c and 4 AMD MI250X GPUs.

Why CUDA?

- Major software challenge today: how do we program accelerators on supercomputers / clusters / laptops?
  - GPU programs: programs run on GPUs while CPUs handle internode communication
  - Hybrid execution: the same calculations on both CPUs and GPUs, and CPUs handle internode communication
- Accelerator architectures are based on executing thousands/millions of independent threads concurrently.
- Existing algorithms must be rewritten for extreme parallelism in order to run efficiently on accelerators.
- In this course we write programs using CUDA, a collection of APIs for NVIDIA GPUs.

Why Not Cuda?

- CUDA is not the only way to program GPUs: there are compiler directive programming concepts
  - OpenACC, http://www.openacc-standard.org/

    #pragma acc parallel *[clause [[,] clause]…]*

    { *structured block, eg. For-loop* }
  - Latest specification: OpenMP 5.2, http://openmp.org
- There are other GPUs than NVIDIA
  - AMD Radeon cards, previously programmed using OpenCL, nowadays with ROCm.
  - Good news: CUDA programs can be converted to run on AMD GPUs: the CUDA program is textually translated to ROCm APIs using ***hipify-perl***, and then compiled with ***hipcc***.

What Is Cuda? Quick Overview

- CUDA is a programming model and environment for general purpose computing on NVIDIA GPUs, GPGPU
- CUDA is an extension of C/C++ or Fortran
- A small set of C language extensions
  - to e.g. launch programs, *kernels*, on the GPU
- A large set of APIs. With these we can
  - check for available CUDA capable GPUs
  - (de)allocate memory on the CPU and GPU
  - copy data from host CPU to device GPU
  - copy data from GPU device to CPU host

What Is Cuda? Quick Overview

- GPU Programming model: small scale parallelism
  - millions or billions of threads
- Structure your program as a large collection of almost independent threads, each thread using a small set of variables
- NOTE: CUDA threads are executed in any order!
- A practical test of execution order independence in for-loops: compare the results from looping your for-loop in opposite directions. If the results are the same then for each i we can use one thread!

```
for ( int i = 1; i <= max; ++i )
for ( int i = max; i > 0;  --i )
```

Course Hardware/Software Environment

- You need an NVIDIA GPU on a laptop, desktop, cluster,…
- GPU with CUDA compute capability 3.0, …, 9.0
- The GPU cluster at ÅAU/UTU: dione.abo.fi
  - ÅAU and UTU are members of the Finnish Computing Competence infrastructure FCCI, promoting the nationwide use of clusters, some of which have GPUs
  - dione has 2 GPU nodes, in total 8 GPUs, NVIDIA V100
  - go to **https://p55cc.utu.fi** to request for a user account on dione.utu.fi

Course Hardware/Software Environment

- CUDA software development kit
  - Windows, Mac OSX and Linux versions available
  - http://docs.nvidia.com/cuda/index.html
  - Latest CUDA toolkit version is 12.2
    - Earlier versions are sufficient for this course
  - Contains compiler ***nvcc***, visual profiler ***nvvp***, command line profiler ***nvprof***, run time environment etc.
- Cuda is available on dione.abo.fi
  - module load cuda
  - module load GCC
  - nvcc -arch=sm_70 program.cu -lm
  - srun –p gpu –w di37 ––mem=1G –t 1:00:00 ./a.out

Course Objectives

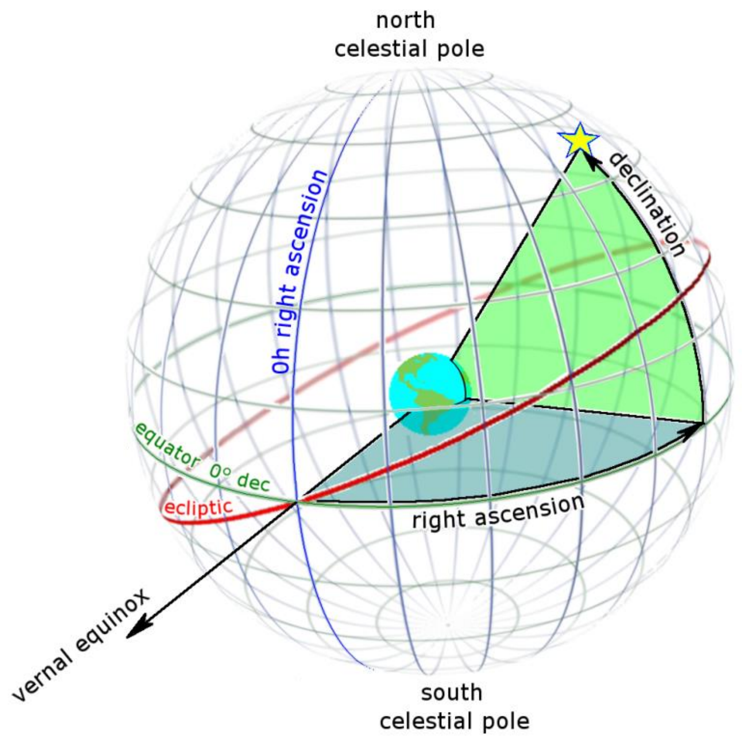
- **Programming**: For a quick start, take the sample CUDA code as a template and start programming!
- **Performance**: One of the course objectives is to understand how to utilize the GPU resources well.
- **Architecture**: We need an understanding of the architecture of present day GPUs
  - Memory layout, execution mechanism
  - Designing your program with individual threads and blocks of threads
  - Coordinating tasks with the CPU(s)
- In some cases we can use GPU libraries: cuBLAS, cuFFT, …

Course Requirements

- To pass this course, you are asked to **design**, **implement** and **run** a CUDA program that indirectly points to the existence of dark matter in the universe!
- Computer simulations indicate that galaxies do not have enough mass to attract each other or even hold together. Hence galaxies should be randomly distributed in the universe.
- You are given a list of galaxies at roughly the same distance from Earth (same red-shift) using celestial coordinates (right ascension, declination). Your task: show that the observed distribution of galaxies statistically differs from a random distribution of galaxies.

Celestial Coordinates

- Celestial coordinates are defined like longitudes and latitudes on earth.



Åbo Akademi University

Course Requirements

- We have two lists of galaxies, for each given RA and declination
  - A list of observed galaxies, list D
  - A random evenly distributed synthetic set of galaxies, list R
- Calculate three histograms DD, DR and RR for the two point angular distribution function
- Choose histogram bins $[0,180^o]$ with even spacing 0.25 degrees (equivalently $0.25*\pi/180$ radians). Hint: $[0,90^o]$ is enough!
- The estimator for evenness between R and D in each histogram bin i is calculated as

$$\omega_i(\theta) = (DD_i - 2*DR_i+RR_i)/RR_i$$

- if $|\omega_i(\theta)|$ is closer to 0 than 1 then the galaxies are evenly distributed
- You can also plot the histograms DD and RR, and compare visually!

Course Requirements

- Input data is available from moodle.abo.fi
- Reference results and further explanations available from

  D. Bard, M. Bellis, M.T.Allen, H. Yepremyan, J.M. Kratochvil, "*Cosmological calculations on the GPU*", **Astronomy and Computing**, Vol. 1 (2013) 17–22
- Present your results to me. Explain which memory layout you have chosen, how your threads are grouped into blocks, what an individual thread is calculating, and provide the total time for your program to calculate the estimators $\omega_i(\theta)$.