An Introduction to GPUs and their Applicability to MC

Internal Seminar at 27th GEANT4 Collaboration Meeting

Jonas Hahnfeld (CERN, EP-SFT)

September 29, 2022

Disclaimer

- Not a classic textbook lecture on GPUs
  - Will skip over most aspects of programming

Disclaimer

- Not a classic textbook lecture on GPUs
  - Will skip over most aspects of programming
- Focus on strengths and peculiarities of GPUs
  - ... with applicability to Monte Carlo simulations in mind

Disclaimer

- Not a classic textbook lecture on GPUs
  - Will skip over most aspects of programming
- Focus on strengths and peculiarities of GPUs
  - ... with applicability to Monte Carlo simulations in mind
- ... condensed into 30 minutes
  - (will focus on GPUs by NVIDIA, but similar concepts apply to AMD)

Contents

Strengths of GPUs

Strengths of GPUs

What exactly **IS** a GPU?

- GPU = **Graphics** Processing Unit
  - CPU = **Central** Processing Unit

(humorous video)

Strengths of GPUs

What exactly **IS** a GPU?

- GPU = **Graphics** Processing Unit
  - CPU = **Central** Processing Unit
- Create images for output to display device
  - Today's hardware: programmable shaders
  - Highly-parallel architecture, efficient for its task

(humorous video)

Strengths of GPUs

What exactly **IS** a GPU?

- GPU = **Graphics** Processing Unit
    - CPU = **Central** Processing Unit

- Create images for output to display device
    - Today's hardware: programmable shaders
    - Highly-parallel architecture, efficient for its task

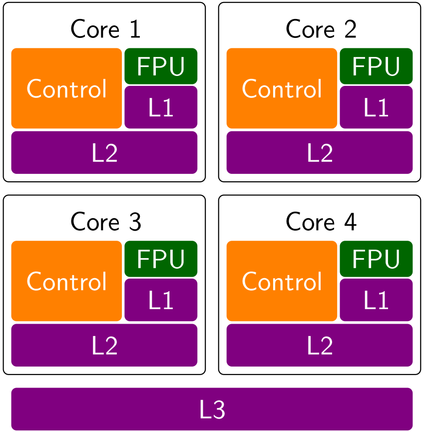
(humorous video)

- GPGPU = General-purpose GPU
    - or "General-purpose computing on GPUs" (according to Wikipedia)

Strengths of GPUs

High-level comparison with (modern) CPUs

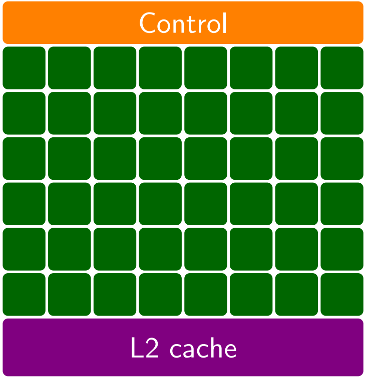
- Current CPUs have few cores
  - At least compared to current GPUs
- Instead optimized for sequential programs (or “mildly” threaded)
  - Hierarchy of cashes
  - Complex control logic, branch prediction
  - Out-of-order execution
  - …



Strengths of GPUs

High-level comparison with (modern) CPUs

- GPUs have many “dumb” cores
- No focus on a single thread
  - Optimize throughput of all cores
- Hide latencies via scheduling
  - Oversubscribe hardware, more threads than cores
  - Efficiently switch between threads
    (for example if waiting for memory)



Strengths of GPUs

Massive data parallelism

- GPUs have to be efficient at their task
  - For graphics, throughput is measured in “frames per second” (FPS)...

Strengths of GPUs

Massive data parallelism

- GPUs have to be efficient at their task
  - For graphics, throughput is measured in “frames per second” (FPS)...
- Need to process millions of triangles for millions of pixels
  - Computations are independent, can be parallelized
  - Simply not important how long a single triangle / pixel would take...

Strengths of GPUs

(Ab)using GPUs for science

- In the early 2000s, scientists noticed the available processing power.
  - They began writing graphics pipelines to “offload” parts of their applications.
  - Effectively abusing the programmable vertex and pixel shaders

Strengths of GPUs

(Ab)using GPUs for science

- In the early 2000s, scientists noticed the available processing power.
  - They began writing graphics pipelines to "offload" parts of their applications.
  - Effectively abusing the programmable vertex and pixel shaders

- In 2007, NVIDIA introduced the CUDA platform
  - Explicit APIs for compute kernels with less overhead
  - Proprietary interface, defined by NVIDIA

Strengths of GPUs

(Ab)using GPUs for science

- In the early 2000s, scientists noticed the available processing power.
  - They began writing graphics pipelines to “offload” parts of their applications.
  - Effectively abusing the programmable vertex and pixel shaders
- In 2007, NVIDIA introduced the CUDA platform
  - Explicit APIs for compute kernels with less overhead
  - Proprietary interface, defined by NVIDIA
- Since then: number of standards for using GPUs (OpenCL, OpenACC, OpenMP)

Strengths of GPUs

Favorable applications

- GPUs are good at highly data-parallel tasks

Strengths of GPUs

Favorable applications

- GPUs are good at highly data-parallel tasks
- Prime use case for science: linear algebra
  - Vectors similar to lists of vertices, matrices represented as arrays...
  - Anyway: transformations are core tasks of GPUs for graphics
- Traditionally important for simulations in HPC

Strengths of GPUs

Favorable applications

- GPUs are good at highly data-parallel tasks
- Prime use case for science: linear algebra
  - Vectors similar to lists of vertices, matrices represented as arrays...
  - Anyway: transformations are core tasks of GPUs for graphics
- Traditionally important for simulations in HPC
- One example: general matrix-matrix multiplication, `gemm`
  - Also at the core of neural networks!

Peculiarities of GPUs

Peculiarities of GPUs

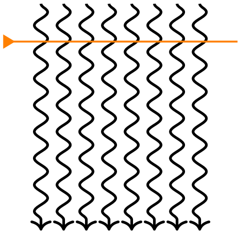
Thread divergence

- GPUs are less good at non-uniform workloads

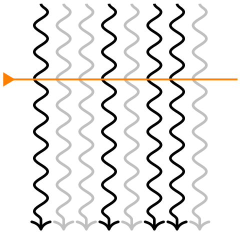
Peculiarities of GPUs

Thread divergence

- GPUs are less good at non-uniform workloads
- Hardware is optimized to work in “lockstep”
  - Initially program counter was per warp ($=32$ threads)!



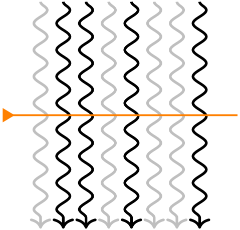
- GPUs are less good at non-uniform workloads
- Hardware is optimized to work in "lockstep"
  - Initially program counter was per warp ($=32$ threads)!
  - For branching code, inactive threads masked out



Peculiarities of GPUs

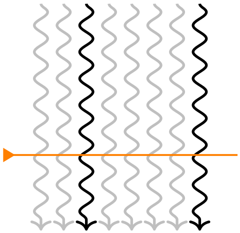
Thread divergence

- GPUs are less good at non-uniform workloads
- Hardware is optimized to work in “lockstep”
  - Initially program counter was per warp (= 32 threads)!
  - For branching code, inactive threads masked out



Peculiarities of GPUs
Thread divergence

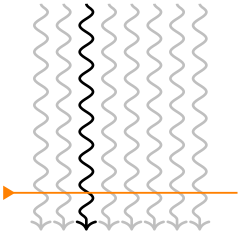
- GPUs are less good at non-uniform workloads
- Hardware is optimized to work in "lockstep"
  - Initially program counter was per warp ($= 32$ threads)!
  - For branching code, inactive threads masked out



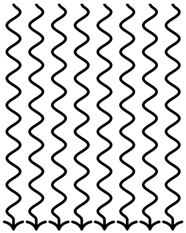
Peculiarities of GPUs

Thread divergence

- GPUs are less good at non-uniform workloads
- Hardware is optimized to work in "lockstep"
  - Initially program counter was per warp ($= 32$ threads)!
  - For branching code, inactive threads masked out
  - Extreme case: only one thread executing at a time



- GPUs are less good at non-uniform workloads
- Hardware is optimized to work in "lockstep"
  - Initially program counter was per warp ($=$ 32 threads)!
  - For branching code, inactive threads masked out
  - Extreme case: only one thread executing at a time
- Somewhat relaxed / improved in recent generations
  - Still, strictly uniform code is fastest



- Related optimization: memory coalescing

- Related optimization: memory coalescing
- On CPUs, improved performance by optimizing for caches
  - Put related values close together, if accessed together
  - Keyword: Array of Structures, AoS

Peculiarities of GPUs

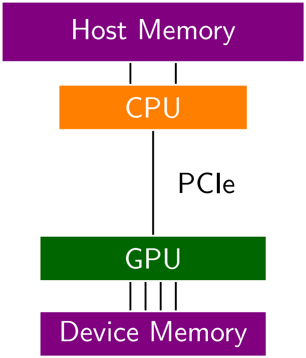
Memory coalescencing

- Related optimization: memory coalescing
- On CPUs, improved performance by optimizing for caches
  - Put related values close together, if accessed together
  - Keyword: Array of Structures, AoS
- On GPUs, best to have threads load adjacent values
  - For example, make 32 threads load entries 0 to 31 of an array
  - Hardware will optimize by "coalescing" loads
  - Keyword: Structure of Arrays, SoA

Peculiarities of GPUs

Separate memory spaces and data transfer

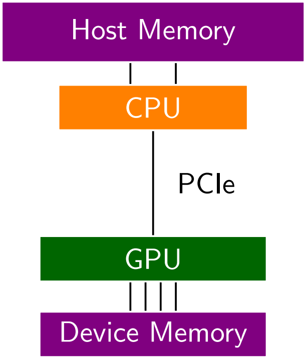
- GPU memory is (usually) separate from main memory



Peculiarities of GPUs

Separate memory spaces and data transfer

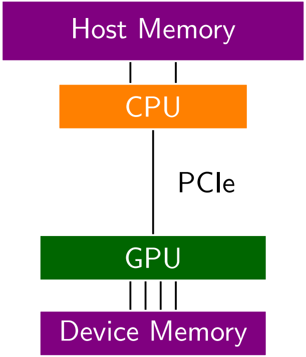
- GPU memory is (usually) separate from main memory
- It is faster than main memory, but limited in capacity
  - Does your problem size fit into GPU memory?



Peculiarities of GPUs

Separate memory spaces and data transfer

- GPU memory is (usually) separate from main memory
- It is faster than main memory, but limited in capacity
  - Does your problem size fit into GPU memory?
- Data must be transferred via interconnect (PCIe)
  - Much slower than memory bandwidths
  - Affects both directions: input data and simulation result



Peculiarities of GPUs

A quick note on floating point precision

- (Consumer) GPUs are sold and optimized for graphics output
  - Need to be fast in single precision floating point arithmetic

Peculiarities of GPUs

A quick note on floating point precision

- (Consumer) GPUs are sold and optimized for graphics output
  - Need to be fast in single precision floating point arithmetic

  - Not so much for `double` precision

  $\rightarrow$ Performance ratio 1:32 for `double` precision computations!

Peculiarities of GPUs

A quick note on floating point precision

- (Consumer) GPUs are sold and optimized for graphics output
  - Need to be fast in single precision floating point arithmetic
  - Not so much for `double` precision

  $\rightarrow$ Performance ratio 1:32 for `double` precision computations!

- Data center GPUs for HPC are optimized for simulation
  - Ratio 1:2 for `double` precision

Applicability for Monte Carlo simulations

Applicability for Monte Carlo simulations

Natural parallelism and computations

- Particle transport is embarrassingly parallel
  - Tracks are simulated independently $\rightarrow$ good for GPU simulation
  - However, leads to very different tracking than GEANT4 (stack-based)
  - Secondary and stopped tracks need to be handled (changing population)

Applicability for Monte Carlo simulations

Natural parallelism and computations

- Particle transport is embarrassingly parallel
  - Tracks are simulated independently $\rightarrow$ good for GPU simulation
  - However, leads to very different tracking than GEANT4 (stack-based)
  - Secondary and stopped tracks need to be handled (changing population)

- Many computations and mathematical functions
  - Logarithms, square roots, exponential, `sin` & `cos`
  - GPUs can provide higher throughput for these

Applicability for Monte Carlo simulations

Non-uniformity

- Monte Carlo simulations governed by random numbers
  - Many interactions require rejection-based sampling

  $\rightarrow$ Thread divergence, bad for performance on GPUs

Applicability for Monte Carlo simulations

Non-uniformity

- Monte Carlo simulations governed by random numbers
  - Many interactions require rejection-based sampling

  $\longrightarrow$ Thread divergence, bad for performance on GPUs

- GEANT4 simulates many different particle types
  - Many different physics processes and models

  $\longrightarrow$ Thread divergence, bad for performance on GPUs

Applicability for Monte Carlo simulations

Non-uniformity

- Monte Carlo simulations governed by random numbers
  - Many interactions require rejection-based sampling

  $\rightarrow$ Thread divergence, bad for performance on GPUs
- GEANT4 simulates many different particle types
  - Many different physics processes and models

  $\rightarrow$ Thread divergence, bad for performance on GPUs
- Divergence also comes from geometry and field propagation

Applicability for Monte Carlo simulations

Data lookup and requirements

- Cross sections require data lookup by kinetic energy
  - Depends on simulation history, which is random

  $\longrightarrow$ No memory coalescing, bad for performance on GPUs

Applicability for Monte Carlo simulations

Data lookup and requirements

- Cross sections require data lookup by kinetic energy
  - Depends on simulation history, which is random
  - → No memory coalescing, bad for performance on GPUs
- Geant4 almost exclusively uses `doubles`
  - Required in some places – a unit vector must be unit!
  - Care must be taken when reducing precision...

Summary

Summary

- GPUs provide great processing power, but are very different from CPUs
  - Designed for massive data parallelism

Summary

- GPUs provide great processing power, but are very different from CPUs
  - Designed for massive data parallelism
- As with any application, performance depends on many factors
  - Some positive characteristics of MC, but many challenges