HPC and Computing Accelerators GPUs and MIC;
Use case: Solving trajectography problems in operational conditions

Dr. Didier El Baz H.D.R.
LAAS-CNRS
Toulouse France

Outline

- 1. Computing Accelerators
- 2. Graphics Processing Unit based accelerators
- 3. Many Integrated Cores
- 4. Study case: Trajectography Problems
- 5. Conclusions

1. Computing Accelerators

- What?
- When?
- Why?
- How?

1.1 Computing Accelerators: what?

- Powerful parallel computing devices
  speedup scientific computations.
  NVIDIA, AMD, intel.

Figure 1. NVIDIA K40 GPU

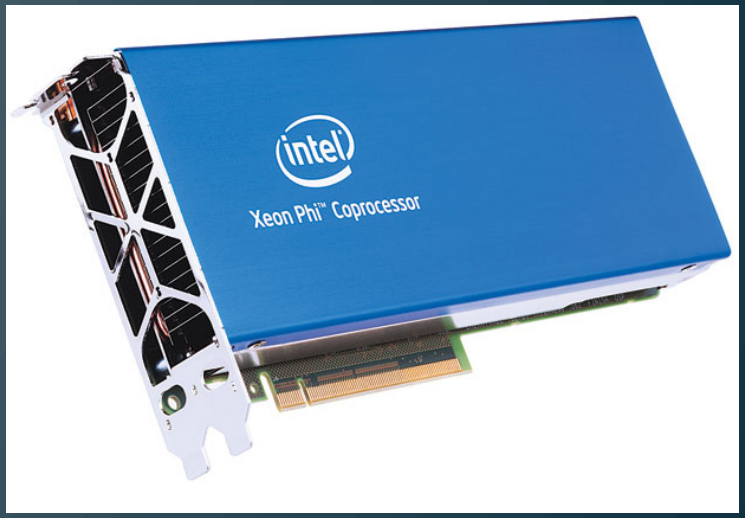

Figure 2. intel Xeon Phi

1.1 Computing Accelerators: when?

- Originally, Visual Processing Units, VPUs, specialized electronic circuits designed to rapidly manipulate memory in order to accelerate the creation of images.
- Ten years ago, Graphics Processing Units (GPUs) have started to be used intensively for general computing purpose under the impulse of NVIDIA → GPGPU.

Figure 3. GeForce 256 world's first GPU (1999)

1.1 Computing Accelerators: why?

- HPC

  - use powerful parallel computing devices

    massive parallelism (GPU), vectorization capabilities (Xeon Phi)

    Tianhe 1A (M2050), Tianhe 2 (Xeon Phi), Titan (K20x)
- Accelerate numerical simulation codes on your PC.
- -structural mechanics (Abaqus/Standard), CFD (Altair AcuSolve)
- Accelerate data treatment in operational conditions

  - aeronautics, oil industry (Tesla K80)…
- Real time treatment → embedded system (Jetson TX1)

  deep learning,

  - automotive industry (Jetson TX1).

1.2 Computing Accelerators: How?

- Today GPUs are very powerfull computing accelerators with thousends of cores.
  - NVIDIA K40, 2880 computing cores, 1.6 teraFLOPS (DP, GPU boost), Memory bandwidth: 288 GB/sec.
  - NVIDIA K80, 4992 CUDA cores, 2.91 teraFLOPS (DP, GPU boost), Memory bandwidth 480 GB/sec.
- New accelerators have appeared: Many integrated Core architecture (MIC), Xeon Phi, 61 cores, vectorization, 1.2 teraFLOPS (DP).

2. Graphics Processing Unit accelerators

- GPU based computing accelerators are highly parallel multithreaded, many-core architectures.
- NVIDIA K40, 2014, $3500 (GK110 Kepler, 15x192=2880 CUDA cores, 1.6 teraFLOPS, 288 gB/sec

Figure 4. NVIDIA K40 GPU

2.1. GPU architecture

- GPU (graphic card): originally designed for visualization purpose (image processing).
- Many cores parallel architectures.
- Multithreaded architecture.
- Two kind of cards:
  - Gaming: GeForce and Quadro products;
  - HPC: Tesla products.

2.1. GPU architecture

- Streaming Multiprocessor (SM) based GPU architecture

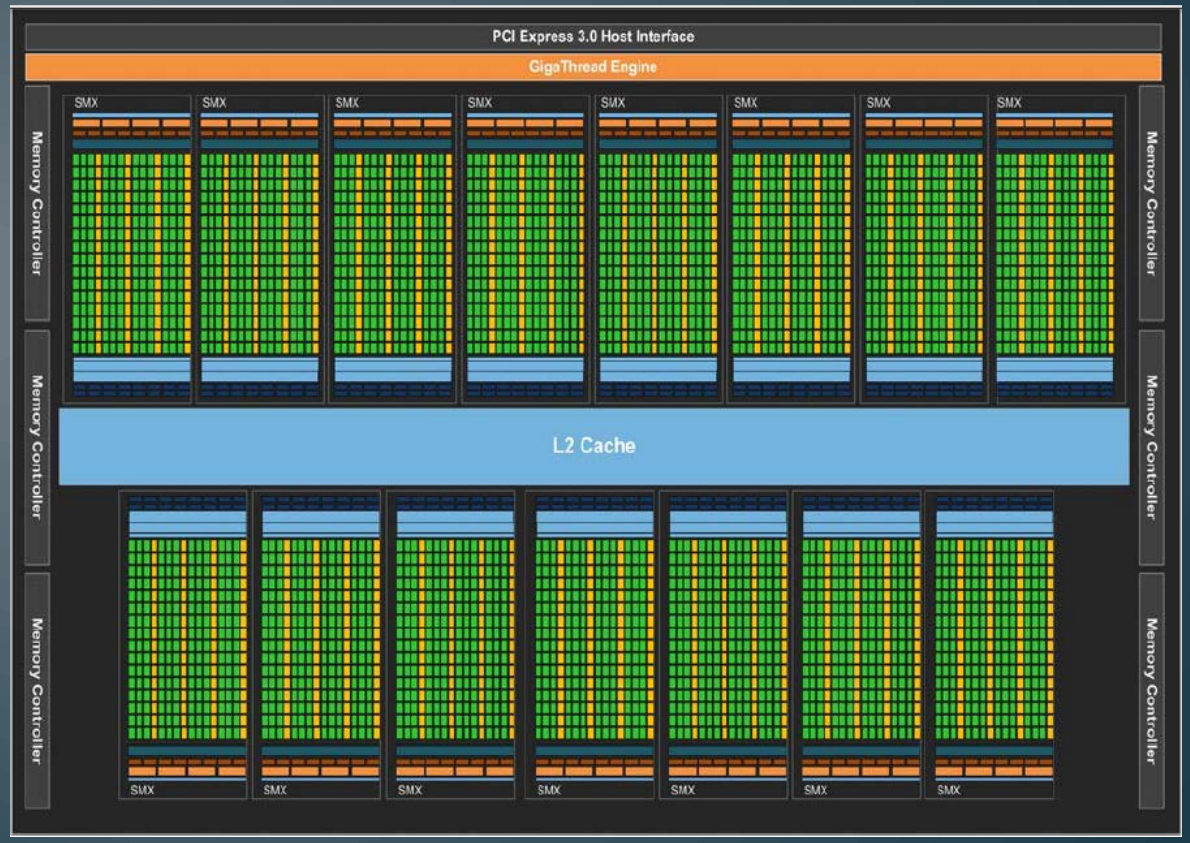

Figure 5. NVIDIA Kepler GK110 architecture

2.1. GPU architecture

- Streaming Multiprocessor architecture

SMX — Streaming Multiprocessor

Core — Single Precision cuda core

DP Unit — Douple precision unit

LD/ST — Loading and Storing unit

SFU — Special function unit

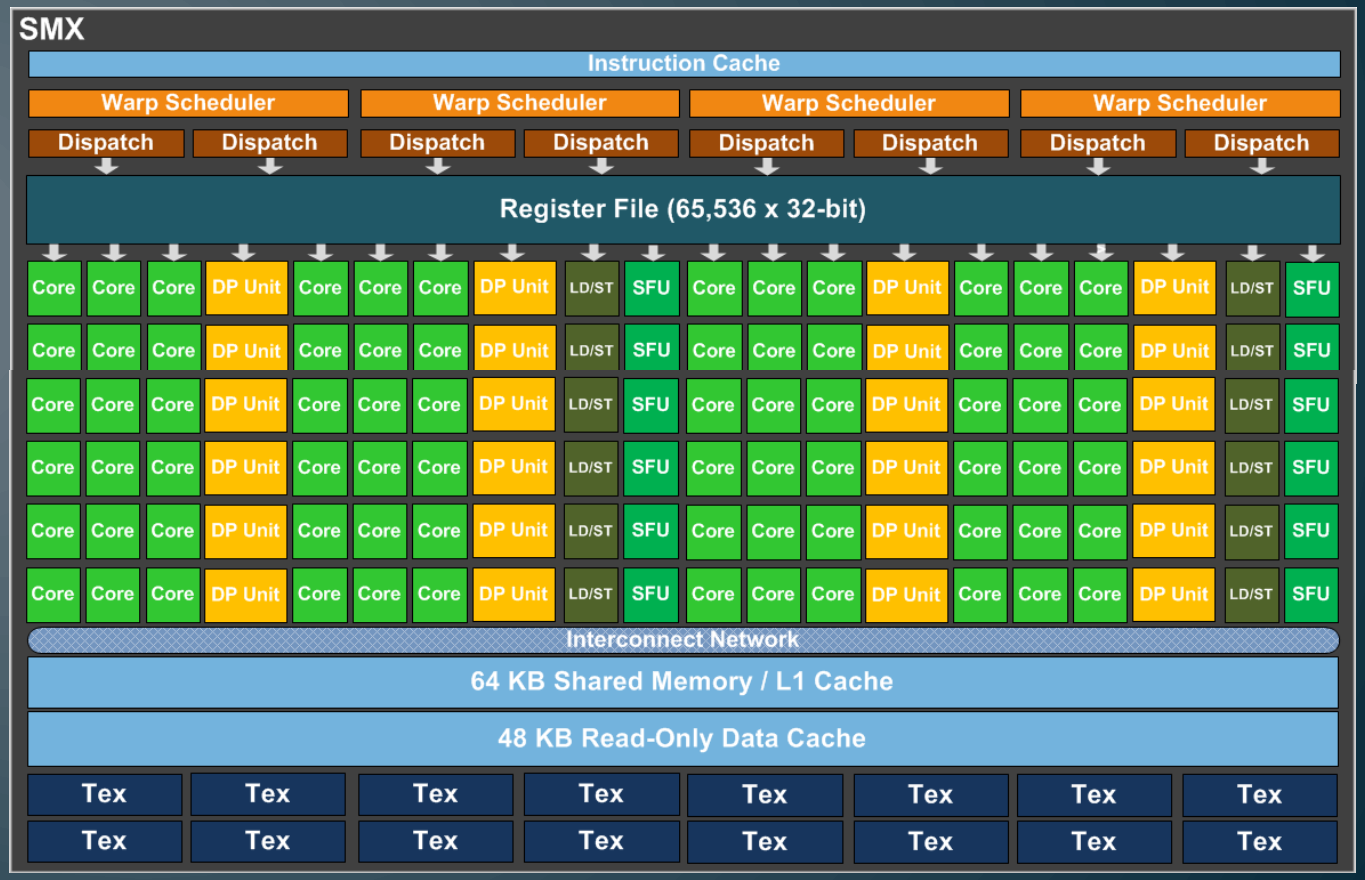

Figure 6. SM architecture (modified)

2.2. Advantages

- The exploitation of GPUs for HPC applications presents many advantages:

➢ GPUs are powerful accelerators featuring thousands of computing cores;

➢ GPUs are widely available;

➢ GPUs are relatively cheap devices;

➢ GPUs are compact devices;

➢ GPUs accelerators require less energy than classical computing devices.

K40: 250 Watts

Jetson TX1: under 10 Watts, ----→

1 TFLOP/s 256-core Maxwell

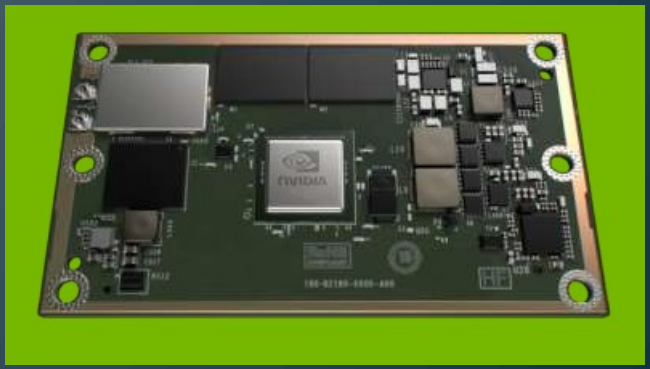

Figure 7. Jetson TX1

2.3. History of GPU Computing accelerators

- Thanks to high-level shading languages like DirectX or OpenGL, graphics accelerators have started to be used for general applications in the early 2000s.
- By that time, problems like stock options pricing and protein folding have been solved on these devices showing noticeable speedup.
- The acronym GPU was then introduced by NVIDIA and people started to speak about General Purpose computing on the GPU (GPGPU).

2.3. History of GPU Computing accelerators (cont'd)

- Programming graphics accelerators via graphics APIs turned out to be difficult.
- Basic programming features were missing and programs were very complex (they had to be expressed in terms of textures, graphic concepts and shader programs).
- Double precision floating point computation was not possible at that time.

2.3. History of GPU Computing accelerators (cont'd)

- GPUs were rethought as highly threaded streaming processors when a new programming model extending C with data-parallel constructs was proposed by Ian Buck, 2006.
- Concepts like kernels, streams, and reduction operators were then introduced.

2.3. History of GPU Computing accelerators (cont'd)

- A new compiler and runtime system permitted one to consider the GPU as a general-purpose processor in a high-level language leading also to substancial performance improvement.
- GPUs are especially well-suited to address problems that can be expressed as data-parallel computations.

2.4. CUDA and OpenCL

- The evolution of GPU's hardware that permits one to program more easily the device combined with the development in 2006 of CUDA and OpenCL has fostered the popularity of GPU computing.
- **Compute Unified Device Architecture (CUDA):** a software and hardware architecture that enables the GPU to be programmed with some high level programming languages like C, C++ and Fortran.
- **OpenCL:** a framework for writing programs that are executed across heterogeneous platforms with CPUs, GPUs and other processors.

2.4. CUDA and OpenCL

- CUDA is a parallel computing platform and programming model designed and developed by NVIDIA.
- CUDA permits one to increase computing performance by harnessing the power of the GPU.
- CUDA greatly simplifies GPU programming.

2.4. CUDA and OpenCL

- With CUDA, one merely writes a serial code intended to the CPU that calls parallel kernels defining the codes to be implemented by threads on the GPU cores.
- CUDA is based on a hierarchy of groups of threads and permits one to use synchronization barrier.
- CUDA functionalities are permanently extended in order to facilitate programming of GPUs. Among the many advantages of CUDA one can quote in particular: fast local memory that can be shared by a block of threads, double precision floating point arithmetic.

2.4. CUDA and OpenCL

- Hierarchy of groups of threads and memory.
- Single Instruction, Multiple Threads (SIMT) parallel programming model.

A parallel code on the device is interleaved with a serial code executed on the host.

The parallel threads are grouped into blocks organised in a grid.

The grid is launched via a single kernel.

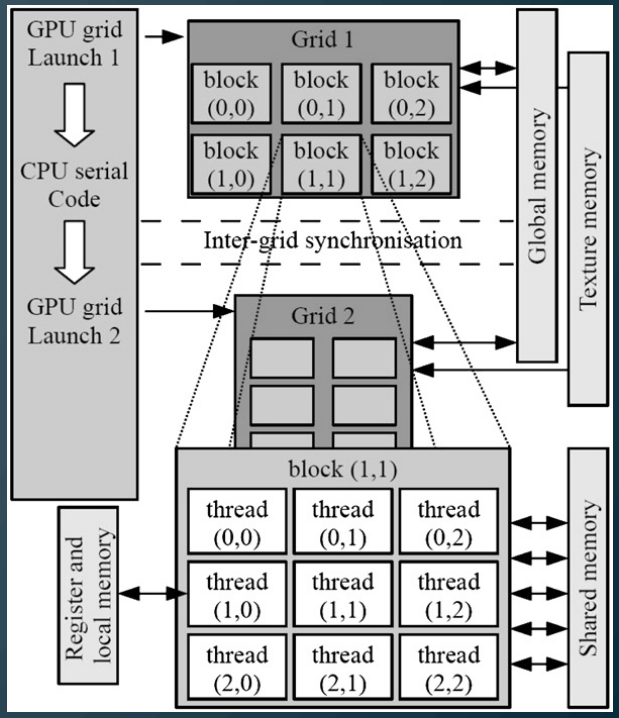

Figure 8 Thread Hierarchy

2.4. CUDA and OpenCL

- B&B on GPU

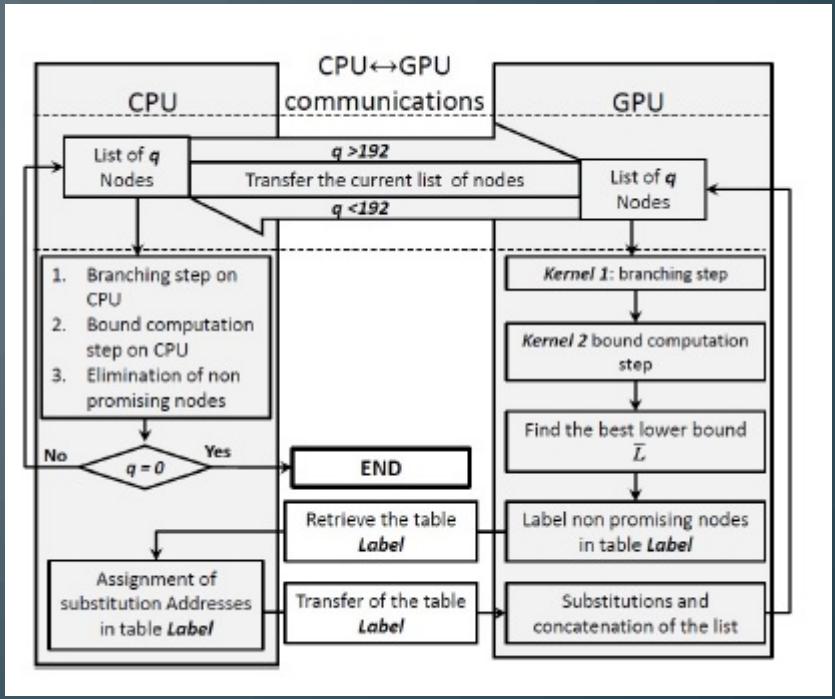

Figure 9 Kernels and data exchanges

2.5. Memory

- Three distinct level of memory :
- Global memory: accessible by every thread within a grid; largest memory (several gigaBytes, highest latency).
- Shared memory: accessible only by each thread within a block.
- Local memory: only accessible by a thread (few kiloBytes, lowest latency)

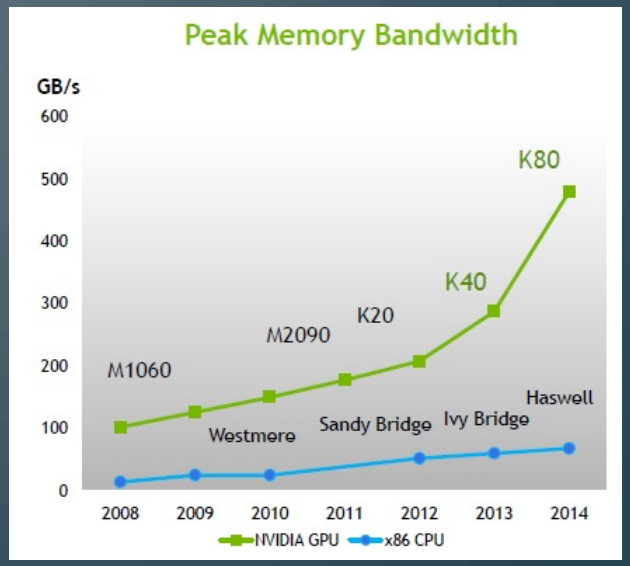

Figure 10 Memory Bandwidth (L2 Cache)

2.6. Heterogeneous Computing Architectures

- Using CPUs, GPUs, in a supercomputer.
- Cluster with Graphics Processing Units (GPU):

Tianhe 1A, TH 1A, NUDT, China, 2009, 1.2 pflops, MPI, OpenMP, CUDA, NVIDIA Tesla M2050.

Figure 11 Tianhe 1A

2.6. Heterogeneous Computing Architectures (cont'd)

- Cluster with Graphics Processing Units (GPU),

  Tsubame II, TIT, NEC and HP., Japan, 2011, NVIDIA Tesla GPU, 2.28 petaflops.

Top 500 Rank: 5

Figure 12 Tsubame II

2.6. Heterogeneous Computing Architectures (cont'd)

- Cluster with Graphics Processing Units (GPU),
  Titan Cray Inc., USA, 2012, NVIDIA K20x, 10 petaflops.
  Top 500 Rank: 2, 8.209 mega Watt, Green 500 rank: 53.

Figure 13 Titan supercomputer

2.7. Applications and optimization

- Widely applied to signal processing and linear algebra.
- Almost all domains in science and engineering are now concerned. We can quote for example astrophysics, seismic, oil industry and nuclear industry, Operations Research (OR).
- Most of the time, GPUs accelerators lead to dramatic improvements in the computation time required to solve complex practical problems.

2.7. Applications and optimization

- Obtaining speedup:
  - Having good thread occupancy, i.e., giving sufficient work to the Streaming Multiprocessors (SM).
  - Maximizing effective memory bandwidth of the GPU.
  - Avoiding divergent branches.
  - Ensuring coalesced memory accesses for all cores of a given SM.

2.7. Applications and optimization

- Multi-GPU computing is possible, i.e., the possibility to exploit several accelerators in a unique application. This leads to massive parallelism.
- Some libraries like CUDA Basic Linear Algebra Subroutines (cuBLAS) have been developed.
- cuBLAS library is a GPU-accelerated version of the complete standard BLAS library.

2.7. Applications and optimization

- CUDA zone:

  https://developer.nvidia.com/cuda-zone

2.8. GPU Roadmap

- Tesla NVIDIA computing accelerators are currently based on Kepler and Maxwell architectures.
- The recent versions of CUDA, like CUDA 7.0, coupled with the Kepler and Maxwell architectures facilitate the dynamic use of GPUs.
- Moreover, data transfers can now happen via high-speed network directly out of any GPU memory to any other GPU memory in any other cluster without involving assistance of the CPU.