GPU for boosting performance in High-Energy Physics

Matteo Concas, INFN Torino
on behalf of the
ALICE, ATLAS, CMS and LHCb Collaborations
1. Graphics card architecture and its usage in High-Energy Physics
   ○ GPU: a brief overview and a comparison with CPU

2. GPU in HEP with an illustrative example
   ○ Kalman Filter for track fitting

3. State of the art and future projects about the GPU adoption in HEP field
   ○ Overview of the ALICE, ATLAS, CMS, LHCb most relevant use cases

4. Conclusion and future perspectives in the adoption of GPUs for next Runs of LHC
A Graphics Processing Unit (GPU) is a **programmable architecture**, offering large number of **parallel** independent streams of instructions, originally designed for image processing.

Chip structure is similar to a Central Processing Unit (CPU), the two architectures are designed to cope with **different type of computations**.

Both GPU and CPU use **threads** to distribute parallel workloads among processors.

CPUs can manage up to **hundred** threads, GPUs can scale beyond **many thousands**.
GPU vs CPU: a fair comparison

- Typical scopes are very different
  - the CPU focuses on minimising the latency: rapid context switching and high clock frequency
  - the GPU aims at maximising its throughput: best performance on static control flows, lower frequencies

- A GPU only implements the “Single Instruction on Multiple Threads” (SIMT) paradigm:
  - each thread in a group performs the same sequence of instructions of its neighbours

- GPU are faster and less power-consuming when workflows fit their typical paradigm
  - peak performance can be one order of magnitude higher than CPU, with lower consumption
  - the number of the floating-point operations per second has been growing at rapid pace compared to CPUs

- To use GPUs is a choice of convenience:
  - a computing node can host up to 8 GPUs and up to 4 CPUs on the same motherboard
  - any work offloaded is a win, as it frees the CPU to do more work
  - re-engineering the software and the algorithms is compensated by the cost over performance
Graphics cards for HEP problems

- General-Purpose GPU (GPGPU): usage of graphics cards for generic computing:
  - Machine Learning, Artificial Intelligence applications and real-time image processing
  - Inherently parallel algorithms whose executions scale well with large core densities

- GPUs have been used for far over a decade, also in High-Energy Physics
  - Many kinds of applications, promoting GPU usage for accelerating algorithms on multiple topics
  - Different projects, more or less advanced at a “maturity” level

- ALICE, ATLAS, CMS, LHCb have been evaluating solutions integrating GPUs for a while:
  - Online/Offline data reconstruction
  - Fast physics simulation
  - Machine/Deep Learning applications (e.g. analysis)
Most popular deployments are online processing and high-level triggers:
  ○ Track seeding (e.g. Cellular Automaton) and event reconstruction (e.g. track fitting)
  ○ High core density is useful to accelerate parallel combinatorial computations (e.g. vertexing)

Some experiments are moving to GPUs to meet requirements for their upgrades

Main features for HEP algorithms to be ported:
  ○ Static, predictable workflows: minimal number of control flow branches (i.e. IF-ELSE statements)
  ○ Intelligent usage of the local memory to minimise latency

An interesting example: Kalman Filter on GPU:
  ○ Used in many different HEP scenarios: popular algorithm
  ○ It fits the “Single Instruction on Multiple Threads” paradigm: computes same iterations on multiple instances at the same time
Iterative algorithm
- Uses least mean squares method
- Updates its inner state at each iteration
- Processes “track seeds” built up from space points
- Final result: fitted functions and covariance matrix

Optimal algorithm pattern: no branches in control flow

GPU can run multiple instances on different tracks
- Same operations for each instance, data are different
- High throughput is achieved by maximising its occupancy

Parallelism at multiple levels
- Multiple events processed simultaneously
- Each track processed in parallel
GPGPU at the LHC experiments, an overview

- Recent times: **Raised interest** in the topic due to the challenges coming from the experimental upgrades
  - Knowledge base is consolidating
- Interest in re-thinking existing algorithm to move towards a parallel approach
  - Integration with existing frameworks not obvious
  - Code portability and duplication are critical
- Any HEP algorithm implementation that can provide large bandwidth and efficient usage of GPGPUs is a good candidate for being offloaded on the accelerators
- A selection of some relevant projects currently actively developed will now be presented
LHCb

- Run 3: The full collision rate will be processed in a software trigger running on GPUs for the HLT-1
  - Every reconstruction chain for each sub-detector
- Allen: self-contained framework that can target both CPU and GPU
  - Efficient tracking, vertex finders, Kalman filters and selection algorithms have been implemented on GPUs

How does HLT1 map to GPUs?

<table>
<thead>
<tr>
<th>Characteristics of LHCb HLT1</th>
<th>Characteristics of GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrinsically parallel problem:</td>
<td>Good for</td>
</tr>
<tr>
<td>- Run events in parallel</td>
<td>- Data-intensive parallelizable applications</td>
</tr>
<tr>
<td>- Reconstruct tracks in parallel</td>
<td>- High throughput applications</td>
</tr>
<tr>
<td>Huge compute load</td>
<td>Many TFLOPS</td>
</tr>
<tr>
<td>Full data stream from all detectors is read out</td>
<td>GPUs have higher latency than CPUs, not as predictable as FPGAs</td>
</tr>
<tr>
<td>⇒ no stringent latency requirements</td>
<td>Connection via PCIe ⇒ limited I/O bandwidth</td>
</tr>
<tr>
<td>Small raw event data (~100 kB)</td>
<td>Thousands of events fit into O(10) GB of memory</td>
</tr>
<tr>
<td>Small raw event data (~100 kB)</td>
<td></td>
</tr>
</tbody>
</table>

Perfect fit!

“Allen: A High Level Trigger on GPUs for LHCb Physics and throughput performance” - CHEP 2019
- Effort to support heterogeneity in CMSSW
  - investigating various "performance portability" approaches
- HLT: 24% of current online reconstruction GPU-ready
  - Patatrack: pixel-based tracks and vertices. Both on GPU and CPU with improved throughputs
  - Calorimeters local reconstruction ported on GPU
- Clustering algorithm based on energy density (CLUE) for the HGCal detector for the CMS upgrade (HL-LHC)
  - results from HGCAL simulation: speedups x50 on GPU
  - rethinking algorithms: also more performance
  - the same parallel code is deployable on CPU
- Machine learning applications, also for inference
- Investigating possible techniques for simulation

"Heterogeneous online reconstruction at CMS" - CHEP 2019

"GPU-based Clustering Algorithm for CMS High Granularity Calorimeter" - CHEP 2019
Previous studies some time ago, renewed interest in GPGPU

- HL-LHC will require more computing power: challenging scenarios for Run 4 and 5
- GPUs might answer those requirements

- It is now easier to design and maintain single code-base deployable on heterogeneous resources and to integrate with existing framework

- Many frontiers are being explored:
  - Online trigger, and offline reconstruction: tracking (eg. for seedfinding), and track following: custom GPU code and ML frameworks
  - ACTS framework: extension to a GPU offloading is under development
  - Madgraph: Event generation using GPUs is now being upgraded
  - Fast calorimeter simulation using different GPU programming interfaces
  - Different ML techniques for analysis could use a GPU backend

In Run 3 ALICE will take 50 kHz of Pb-Pb data in a triggerless fashion

- High data compression factors require online reconstruction
- Data reduction from ~3.5TB/s bandwidth down to < 100 GB/s
- GPUs are the pivotal architecture to efficiently process such a reduction
- Time Projection Chamber and Inner Tracking System reconstruction on GPU is the base goal
- TPC: most relevant use case, its speed-up on GPU is a factor of x20. One GPU replaces ~40 CPU cores
- More optimistic scenarios foresee to enqueue more reconstruction algorithms on GPUs whenever not fully used for data reduction
- This will allow for a better data classification that will improve analysis performance
Conclusions

- Next Runs at the LHC will be extremely challenging in terms of computing requirements
- Experiments are considering GPUs to address the increasing demand for computing power
- Many different ongoing efforts with diverse scopes by all the larger LHC experiments
  - They accelerate a parallel workflow to push performance beyond what can be achieved with CPUs
- GPUs enable the deployment of computing scenarios hardly addressable using standard CPUs
  - Using GPUs allows for freeing resources and reduce the number of required nodes
- Next generations of data centres and HPC facilities will increase the number of GPUs in their computing pledges
  - Efficiently exploiting such potential would be beneficial, since our need for computing power is always growing
Kalman Filter in a nutshell

- **Initialisation**: starting condition $p_0(r, C)$, large errors to minimise biases
- **Prediction** of the subsequent hit using starting information
- **Update** $p_i(r, C)$ at each iteration by adding information extracted from hits position and predict next hit position
- **Repeat** for each detecting surfaces crossed by the particle
- Last iteration provides the result

Every iteration is composed by the same matrix products that updates the $p_i$ model
pixel tracks and vertices global reco

CPU
- dual socket Xeon Gold 6130
- 2 x 16 cores (2 x 32 threads)
- throughput measured on a full node
- 4 jobs with 16 threads

GPU
- single NVIDIA Tesla T4
- 2560 CUDA cores
- single job with 10-16 concurrent events

transfer from GPU to CPU
- on demand
- small impact on event throughput

conversion to legacy data formats
- on demand, to be minimised
- small impact on event throughput
- high cost in CPU usage