THE GAP PROJECT:
GPU for Realtime Applications in High Energy Physics and Medical Imaging

M. Bauce, A. Messina, M. Rescigno, S. Giagu, S. Capuani, M. Palombo, G. Lamanna, M. Fiorini, G. Di Domenico

1 Sapienza Università di Roma

IEEE - Realtime, Nara, May 26-30, 2014
Basic GPU architecture

- Less CONTROL units, many more ARITHMETIC LOGIC units.
- Optimized to execute simultaneously the same operation on many different data (Single-Instruction on Multiple Data)
- SMX executes kernels/functions using hundreds of threads concurrently
While a CPU executes programs serially a GPU is tailored for highly parallel operation.

GPUs have many parallel execution units (cores) and higher transistor counts, while CPUs have few execution units and higher clock speeds.

GPUs have significantly faster and more advanced memory interfaces as they need to move around a lot more data than CPUs.

A GPU is for the most part deterministic in its operations.
The GAP Project

GPU Application Project for Realtime in HEP and medical imaging is a 3 years project funded by the Italian Ministry of research, started in April ’13

Collaboration between: Sapienza università di Roma, INFN Pisa, Università di Ferrara

Aim to demonstrate that is possible to use off-the-shelf computer commodities to accelerate computation in scientific application:

The main challenge will be to extend it to real time application for the first time.

APE group - INFN Roma
Different scales of Realtime

see P. Vicini's talk on 29/05
What is a GPU suitable for?

- Every algorithm that can be divided into tasks operating on independent data can gain from parallelization.

**Amdahl’s law**

- Different color → Different tasks

- Weak point is the memory usage: small amount available, overhead for data cross-reading algorithm.
What is a GPU suitable for?

- Every algorithm that can be divided into tasks operating on independent data can gain from *parallelization*.
- Weak-point is the memory usage: small amount available, overhead for data cross-reading algorithm.
A physics scenario: NA62

- Kaon decays in flight
  - High intensity unseparated hadron beam (6% kaons).
  - Need a good kinematics reconstruction.
  - Efficient veto and PID system for background contribution

- Trigger/DAQ:
  - Three levels
  - L0 max latency: 1ms
  - 10 MHz events on main detectors

- RICH:
  - 17 m long, 3 m in diameter, filled with Ne at 1 atm
  - 10 MHz events: about 20 hits per particle
  - Distinguish between pions and muon.
Several approaches to the ring fitting problem in HEP.

- Incomplete rings (non symmetric hits distribution)
- Trackless, difficult to merge detector info at L0
- Fast reconstruction, not iterative
- Accurate results, offline quality
- Low latency for synchronous trigger

State-of-Art

- Geometric fits
- Algebraic fits
- Statistical fits
- Conformal fits

Single ring

Multi-body decays
**ALMAGEST**: a new algorithm based on Ptolemy’s theorem:

*If a quadrilateral is inscribable in a circle then the product of the measures of its diagonals is equal to the sum of the products of the measures of the pairs of opposite sides.*

\[
\overline{AD} \times \overline{BC} + \overline{AB} \times \overline{DC} = \overline{AC} \times \overline{BD}
\]
1) Select a triplet (3 starting points)
1) Select a triplet (3 starting points)

2) Loop on the remaining points: if the next point does NOT satisfy the Ptolemy’s theorem reject it.
1) Select a triplet (3 starting points)

2) Loop on the remaining points: if the next point does NOT satisfy the Ptolemy’s theorem reject it.

3) If the point satisfy the Ptolemy’s theorem consider it for the fit.
1) Select a triplet (3 starting points)

2) Loop on the remaining points: if the next point does NOT satisfy the Ptolemy’s theorem reject it.

3) If the point satisfy the Ptolemy’s theorem consider it for the fit.

4) repeat again
1) Select a triplet (3 starting points)

2) Loop on the remaining points: if the next point does NOT satisfy the Ptolemy’s theorem reject it.

3) If the point satisfy the Ptolemy’s theorem consider it for the fit.

4) repeat again, and again
1) Select a triplet (3 starting points)

2) Loop on the remaining points: if the next point does NOT satisfy the Ptolemy’s theorem reject it.

3) If the point satisfy the Ptolemy’s theorem consider it for the fit.

4) Repeat again

5) Perform a single ring fit
1) Select a triplet (3 starting points)

2) Loop on the remaining points: if the next point does NOT satisfy the Ptolemy’s theorem reject it.

3) If the point satisfy the Ptolemy’s theorem consider it for the fit.

4) repeat again

5) Perform a single ring fit

6) Repeat excluding the points already assigned to a ring
1) Select a triplet (3 starting points)

2) Loop on the remaining points: if the next point does NOT satisfy the Ptolemy’s theorem reject it.

3) If the point satisfy the Ptolemy’s theorem consider it for the fit.

4) Repeat again

5) Perform a single ring fit

6) Repeat excluding the points already assigned to a ring

Add complications
1) Select a triplet (3 starting points)

2) Loop on the remaining points: if the next point does NOT satisfy the Ptolemy’s theorem reject it.

3) If the point satisfy the Ptolemy’s theorem consider it for the fit.

4) repeat again

5) Perform a single ring fit

6) Repeat excluding the points already assigned to a ring

Add complications
1) Select a triplet (3 starting points)

2) Loop on the remaining points: if the next point does NOT satisfy the Ptolemy’s theorem reject it.

3) If the point satisfy the Ptolemy’s theorem consider it for the fit.

4) repeat again

5) Perform a single ring fit

6) Repeat excluding the points already assigned to a ring

Add complications
Two levels of parallelism on data:
- Several triplets run in parallel
- Several events at the same time

Very high parallelism on GPU:
- Huge number of computing cores, > 2000
- Huge memory bandwidth
A simple example

- The real parameters of the generated rings:
  1. (6.65, 6.15) $R=11.0$
  2. (8.42, 4.59) $R=12.6$

- The fitted parameters of the two ring:
  1. (7.29, 6.57) $R=11.6$
  2. (8.44, 4.32) $R=12.3$

- Fitting time on Tesla C1060: $1.5 \mu s$/event
Algorithm performances

- Total computing time of the order of few $\mu$s per event (on single GPU - Tesla K20).
- Reasonable efficiency (using 8 triplets): room for improvement (further tests ongoing).

- Single and multiple ring algorithms have similar latencies (packing together 256 events).
Data processing real challenging

- Vectorization of the data-analysis code
- Task parallelization during processing
- Include cutting-edge technologies: GPU, MIC, etc. under investigation

- Energy: 7-8 TeV (14 TeV)
- Inst. luminosity: $10^{34} \text{ cm}^{-2}\text{s}^{-1}$
- Bunch crossing rate: 40 MHz
- Event size: 1.5 MB
- Input Data rate: 30 TB/s
New challenges in the upgraded luminosity regime

- Atlas is considering new technologies to include in the upgrade
New challenges in the upgraded luminosity regime

Atlas is considering new technologies to include in the upgrade

Foreseen merge of L2 and EF

Possibility to exploit GPU to run high level trigger algorithms, aiming to offline-like quality resolution.

Two issues to take into account:

1. Integration of GPU in the complex framework
2. Exploit high-parallelizable algorithms
Atlas Trigger has a complex framework: how to integrate a GPU in it?

- **Technical implementation of GPU in Atlas SW through a **client-server** structure.
- **Flexible solution useful for several parallel tasks:** can be expanded to a GPU farm.
Muon Isolation algorithm

1. Retrieve calorimeter cell information
2. Pack data for transfer
3. Send them to APE server
4. Perform algorithm
5. Send back the output to the main trigger framework
6. Total time: $\sim 1.2 \text{ ms}$
Muon Isolation algorithm

1. Retrieve calorimeter cell information
2. Pack data for transfer
3. Send them to APE server
4. Perform algorithm \( \Delta t \sim 250 \, \mu s \)
5. Send back the output to the main trigger framework
6. Total time: \( \sim 1.2 \, \text{ms} \)

Overhead well within time budget!
Atlas Standalone test: Track fitting

- Track fitting on GPU
  - Kalman Filter and track fit on tracks from detector
  - Parallel fit of an increase number of track

GPU implementation less dependent on the track multiplicity:
  - promising for upcoming data taking conditions, with increased luminosity.
Nuclear Magnetic Resonance

Conventional DTI:
- $7.86 \times 10^6$ linear systems to solve
- $\sim 15$ s on a CPU

Kurtosis Tensor imaging:
- $7.86 \times 10^6$ non-linear fit to perform
- $\sim 7200$ s on a CPU

Typical NMR brain scan:
- $128 \times 128$ pixel images
- 10-40 slices
- $\geq 15$ diffusion gradient directions
- On average: $128 \times 128 \times 32 \times 15 = 7.86 \times 10^6$ calculations

Computationally intensive, massively parallel:
- A non-linear fit: $\sim 7$ ms each non-linear fit
- $10^6$-$10^7$ independent fit on each pixel
- only $10 - 80$ ns to read data from memory

8 threads on an Intel Xeon E5-2609 CPU at 2.4 GHz
Estimated improvements:

- High-res. images $512 \times 512$
- 20-200 speed-up factor after algorithm optimization

Image simulations:
- matrix size: $32 \times 32$ up to $2048 \times 2048$
- about $2 \times 10^3$ axons
- 30 points ob $b$-value, from 0 to 5000 s/mm$^2$

Significant improvement observed using GPU
Computed Tomography

- Feldkamp-Davis-Kress algorithm to solve the differential equation below
- Three steps intrinsically parallel
- Big improvement on GPU wrt. parallel OpenMP
- Expected big benefits from next generation GPU (K40)

\[
\hat{f}(s_1, s_2, s_3) = \frac{1}{2} \int_{0}^{2\pi} d\beta \int_{-um}^{um} du \cdot \frac{D}{\sqrt{D^2 + u^2 + v^2}} \cdot g_{\beta}(u, v) \cdot h(u - u')
\]

<table>
<thead>
<tr>
<th>Step</th>
<th>GTS 450</th>
<th>GTX680</th>
<th>Titan</th>
<th>OpenMP</th>
<th>OpenMP/Titan</th>
</tr>
</thead>
<tbody>
<tr>
<td>weighting</td>
<td>0.32</td>
<td>1.5</td>
<td>0.39</td>
<td>0.22</td>
<td>0.55</td>
</tr>
<tr>
<td>filtering</td>
<td>49.17</td>
<td>15.01</td>
<td>13.57</td>
<td>1.26</td>
<td>0.09</td>
</tr>
<tr>
<td>backlproj</td>
<td>17.45</td>
<td>4.37</td>
<td>3.08</td>
<td>80.85</td>
<td>26.3</td>
</tr>
<tr>
<td>Total</td>
<td>66.94</td>
<td>20.88</td>
<td>17.04</td>
<td>82.33</td>
<td>4.83</td>
</tr>
</tbody>
</table>
GPU: quickly developing technology now available for realtime applications.

- trigger algorithm optimized for GPU parallel execution are a viable solution in typical HEP experiments:
  - Tested interface for integration in complex TDAQ frameworks
- allow implementation of complex and good-resolution algorithm for image reconstruction in medical diagnostic

... room for improvements and applications in other scenarios.

ありがとうございました

Thanks!