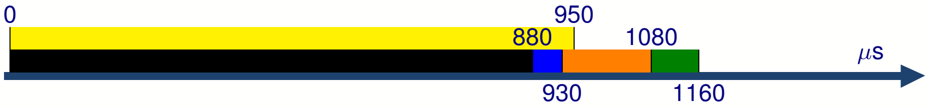
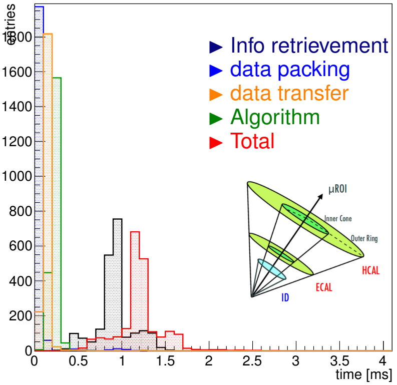
GPU in Atlas HLT

M. Bauce

October 24, 2014

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: ~1.2 ms

$\cdots$ A month ago in Pisa $\cdots$

What we learnt so far

- Integrating a GPU in an HEP experiment has a cost in terms of execution time:
  - The server-client scheme is the most flexible and transparent solution
  - The overhead is of the order of $\sim$100 μs, within the typical HLT time budget
  - A critical aspect is handling complex pre-existing object-oriented data structures

- Two (complementary) viable approaches:
  1. Consider more complex algorithms that can better exploit the time already invested in the data transfer
  2. Manage *simpler data structure* on GPU, ignoring framework structures

$\cdots$ A month ago in Pisa $\cdots$

Interaction with the Atlas collaboration

What we learnt was actually interesting also to Atlas to proceed:

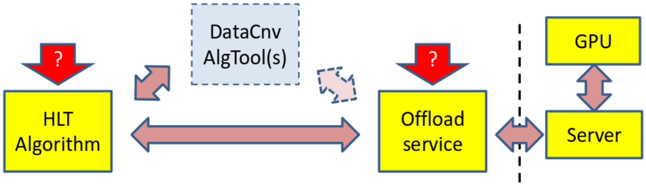
- Only complete implementation of Athena-GPU communication and Atlas trigger algorithm execution
- We used undocumented, continuously evolving version of APE code, tailored solution

▶ Atlas decided to start a structured effort on this path, putting all the pieces together for benchmark measurements in a 1-year timescale: creation of a **GPU Demonstator**

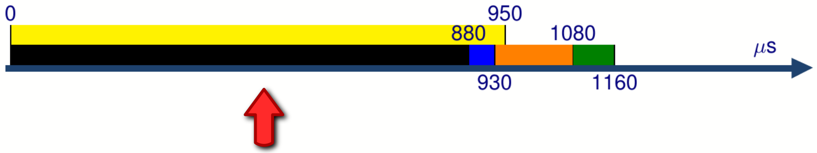
(Sort of) kick-off meeting: `https://indico.cern.ch/event/344464/`

Athena/GPU data conversion (from Dimitry slides)

- Data conversion problem:
  - GPU EDM: SoAs, C structs, vector data types
  - Athena EDM: non-trivial C++ objects with pointers
- Question:
  - what would be the right place to convert the data ?



Possible work/data flow (from Dimitry slides)



- CPU→GPU
  - Athena service to prepare payload for GPU: retrieve and convert data to flat buffer
  - Athena service to send payload, header and instructions
  - Wait for a response · · ·
- GPU→CPU
  - Athena service to get notification of ended execution; retrieving data buffer
  - Athena service to convert result to Object Oriented structure and put them back into Athena
  - Continue · · ·

► RoI-by-RoI execution or event-by-event execution? Multiple RoI data handling?

Interaction with the Atlas collaboration

Athena interface

- Offloading: official code in SVN (Dimitry)
- Serialization: *ongoing* (Dimitry)
- *Data extraction for development: pending*

APE server

- First *stable* APE release: Sami
- APE (partial) integration into Athena: Sami
- Test of *dummy* algorithm: Sami/Matteo

GPU algorithms

- Modification of previous test (mulso) to be compliant with new APE code: ongoing
- Investigation of other algorithm: *ongoing**

Additional thoughts on the algorithm

- Algorithm chosen as first test (muIso) processes RoI-per-RoI, not suitable for parallel scaling
- muIso execution on multiple RoI results in reading large fraction of the detector: fast enough?
- Considering EF algorithms: full event available, less synchronization issues...

- Need to go back one step: retrieve bare objects and re-think one trigger algorithm
  - focus on one reconstruction test: tracking? fitting? vertexing?
- e.g. track reconstruction on MS: Hough transform? Retina?

Conclusions

- Interacting with Sami: help testing a stable version of APE
- Converting to new *standards* the first test code
- Studying MS (tracking) algorithms for possible applications (suggestions welcome)

SPARES SLIDES

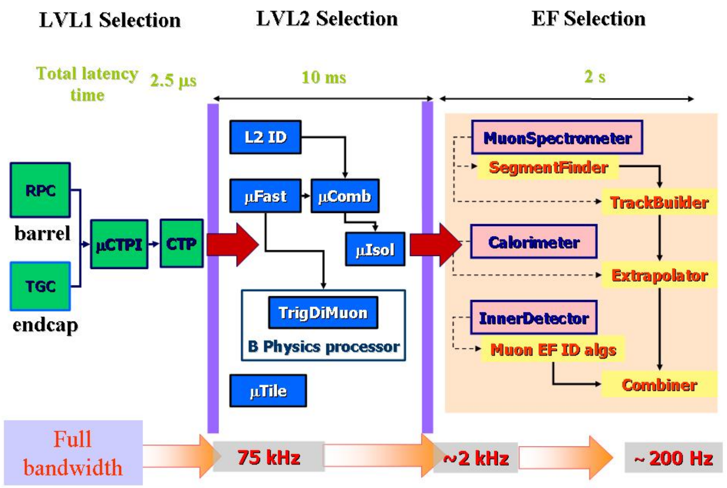
Atlas *taskforce* plans

- modify the algorithm of interest by adding a flag (DoOffloading) which would then retrieve the OffloadingSvc https://svnweb.cern.ch/trac/atlasoff/browser/Offloading
- when in this configuration, the algorithm would prepare data and sent it to OffloadingSvc rather than doing the processing itself.
- critically important decisions needs to be made at this stage about the data organisation and which algorithms would be accelerated
- a dummy algorithm of APE should receive this data and respond with some predefined pattern and this pattern should be properly decoded back in HLT algorithm. This step would require the definition of a protocol/commands that will steer processing in the APE server.

In addition we could think of a not-so-dummy algorithm to save the offloaded data to disc and replay this data standalone to facilitate quicker development of APE modules (GPU or other accelerated code).

J. Baines

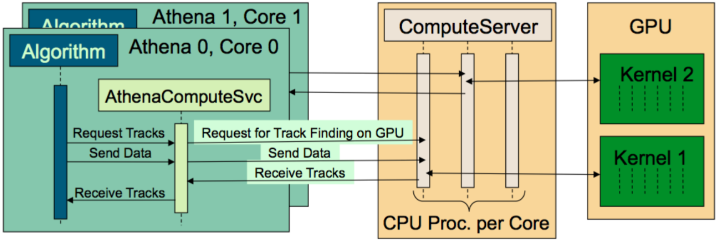
Muon Trigger Flow



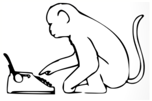
► Starting from next run L2 and EF will be merged in a High Level Trigger (HLT), combining various algorithms.

Framework Architecture Implementation

Atlas Trigger has a complex framework: how to integrate a GPU in it?

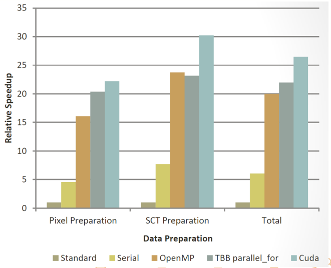


APE project

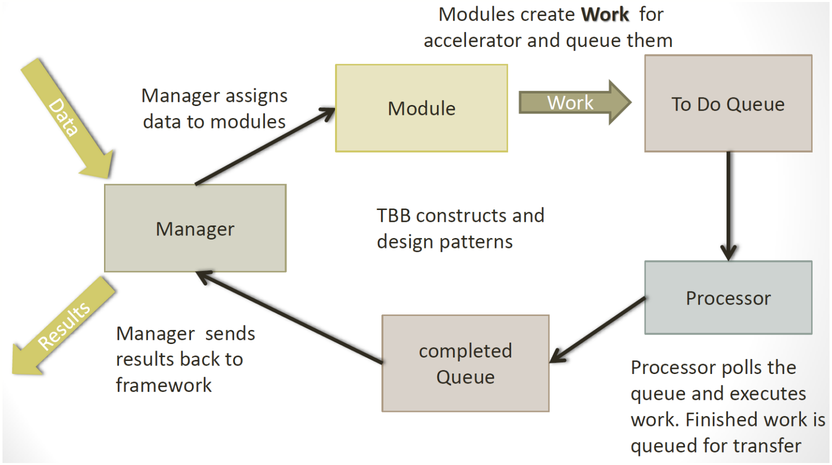


- Tecnical implementation of GPU in Atlas SW through a **client-server** structure.
- Flexible solution useful for several parallel tasks:

  can be expanded to a GPU farm.

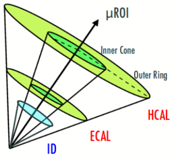


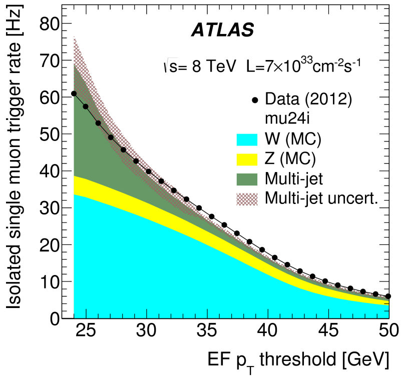
Atlas: APE Processing Data Flow


Modules create **Work** for accelerator and queue them
Data
Manager assigns data to modules
Module
Work
To Do Queue
Manager
TBB constructs and design patterns
Processor
Results
Manager sends results back to framework
completed Queue
Processor polls the queue and executes work. Finished work is queued for transfer

Evolution of Muon Isolation algorithm

- Muon isolation is fundamental to reduce single muon trigger rate
- Relevant fraction of the bandwidth occupied by fake leptons (jets)



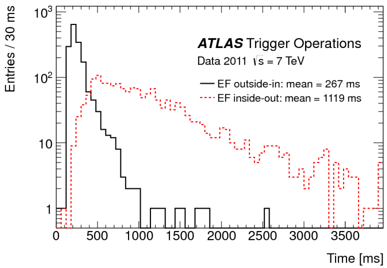


- Refined isolation can replace increase in muon p_T threshold
- Offline muon quality improved from pileup-dependent corrections:
  - Energy deposit evaluated in parallel in control region of the detector
  - Subtract this deposit in the muon isolation calculation

Multivariate approach in Muon Trigger

▶ Neural Network-based muon reconstruction in Atlas starting from an ID track:

- Highly efficient for low-p_T muons, interesting *B*-physics signatures (displaced vertices)
- High track multiplicity, expected to increase with the LHC luminosity increase.
- Neural Network implementation on GPU will provide large speedup factors.



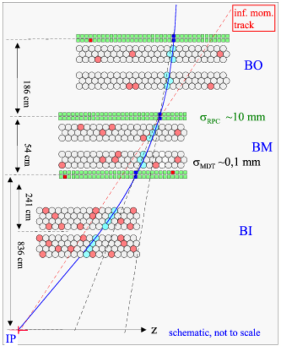
Muon Spectrometer track fitting

Track segment reconstruction in the muon spectrometer takes $\sim 2.5\ \mu$s

Several steps can benefit from parallelization on a dedicated architecture:

- Massively parallel bytestream decoding
- Seed reconstruction and combinatorial suppression
- Track extrapolation to interaction vertex

Need a deep modification of the reconstruction framework



Similar approach developed for the Inner Detector tracking