GPU accelerators in the ATLAS High Level Trigger

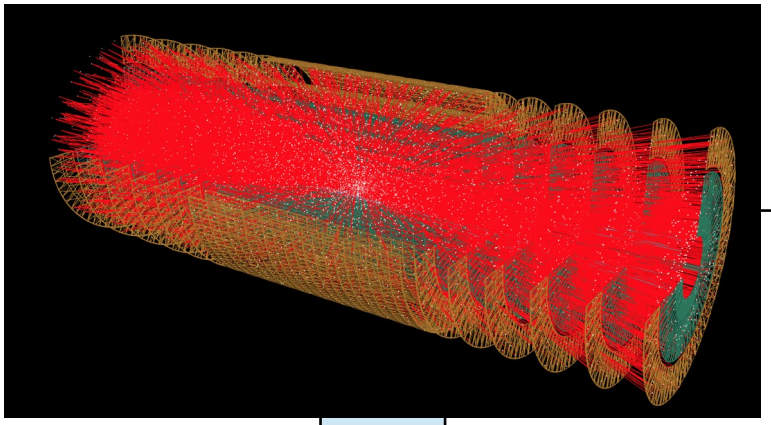
B. Wynne

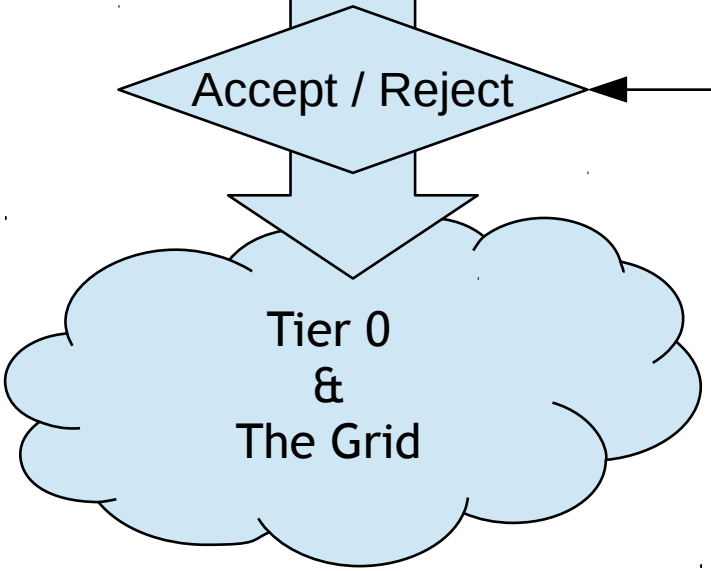
17/02/2020

Introduction

The ATLAS experiment collects vast quantities of data, the overwhelming majority of which is not to be retained for analysis

Decisions to record or reject events are made in real time using a system called the trigger

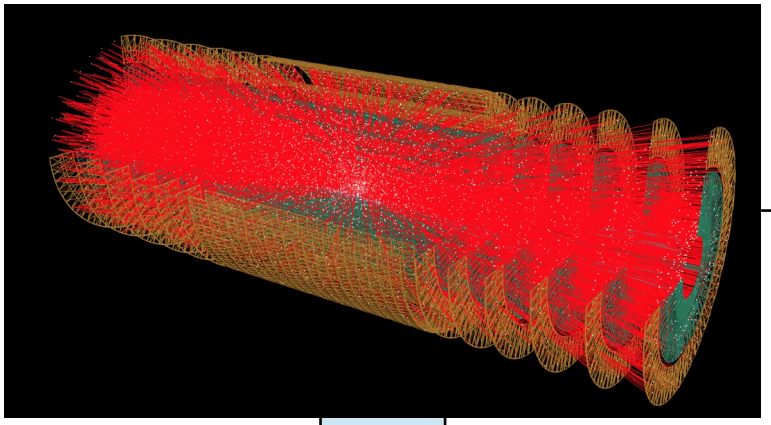




Introduction

The ATLAS experiment collects vast quantities of data, the overwhelming majority of which is not to be retained for analysis

Decisions to record or reject events are made in real time using a system called the trigger



40 MHz

Custom hardware

Identifies large-scale features of interest in low detail

100 kHz

~1.5 kHz

Accept / Reject

Tier 0
&
The Grid

Commodity hardware

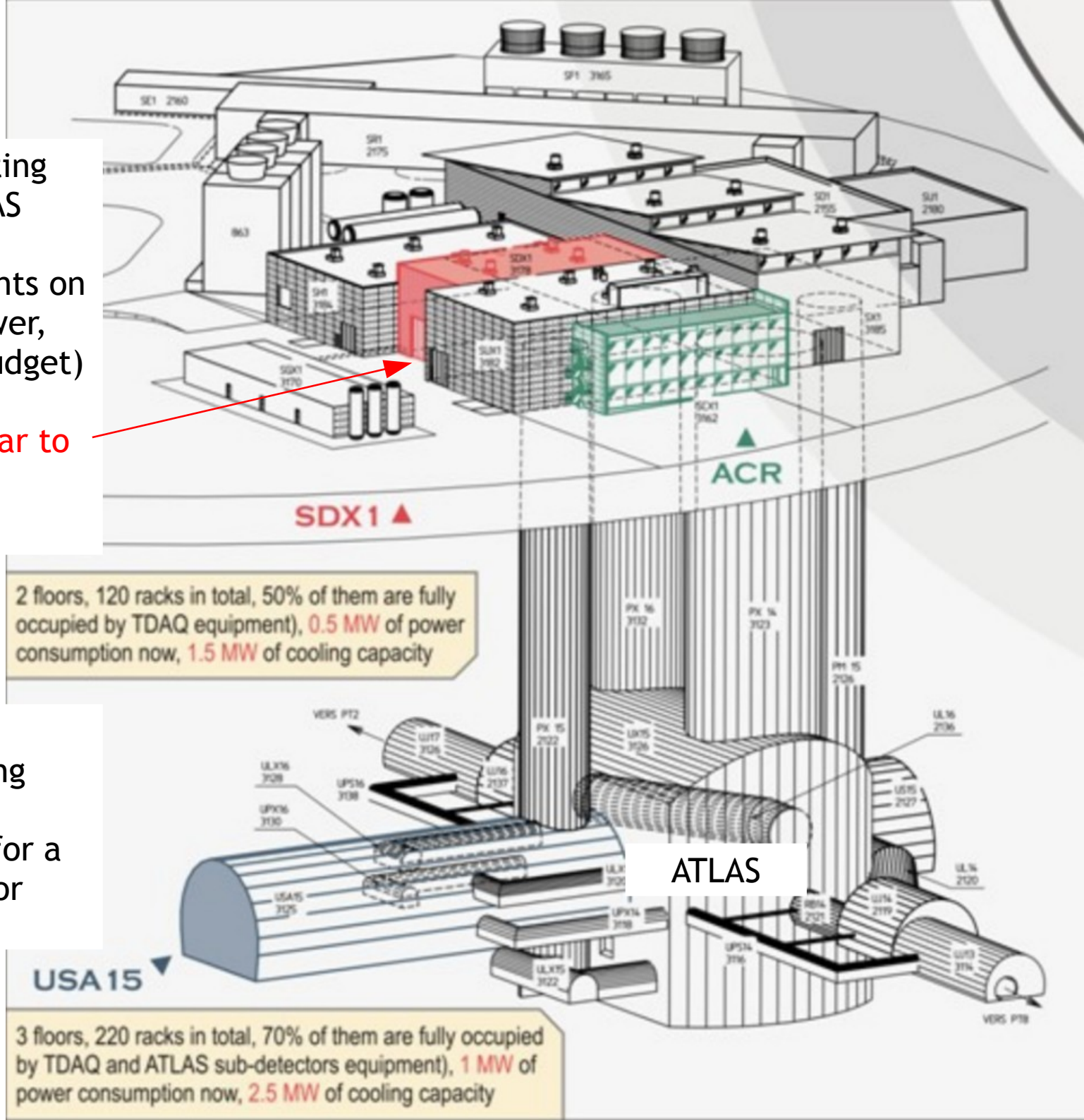
Full detail reconstruction in regions of interest

Introduction

Besides the computing challenge, the ATLAS trigger system has significant constraints on physical space, power, and cooling (and budget)

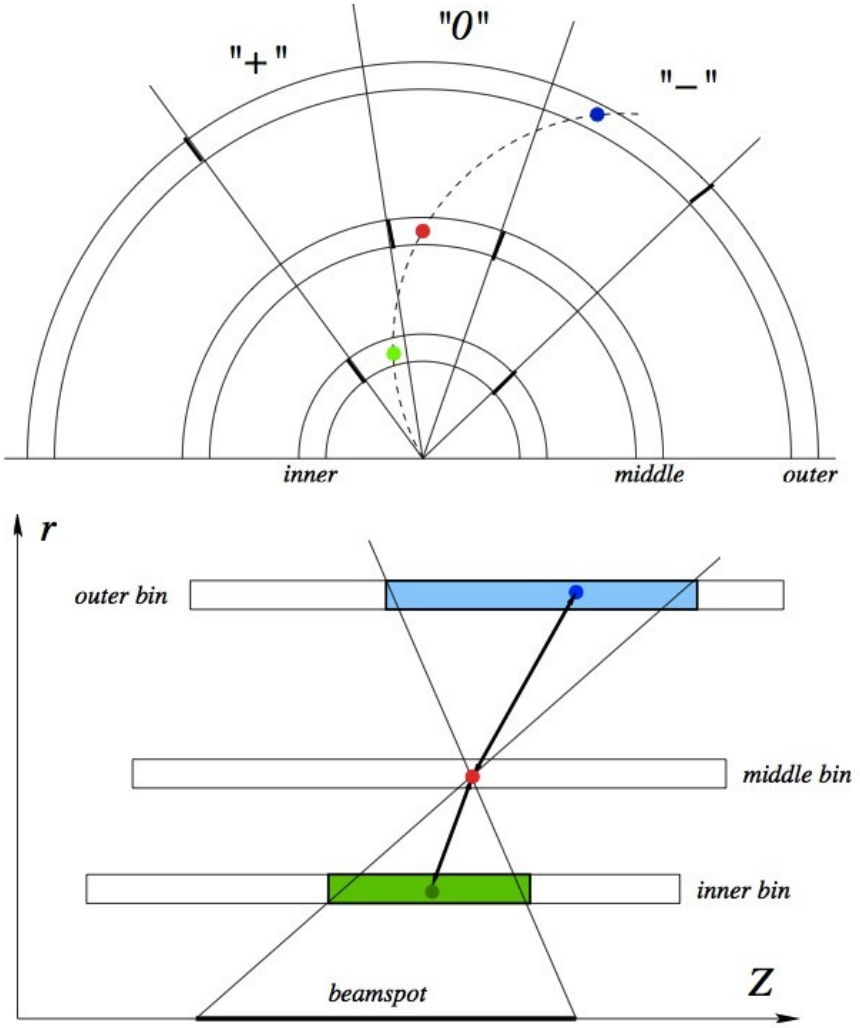
HLT racks are as near to the detector as is practical

Since the HLT has dedicated computing hardware, we can choose to develop for a specific co-processor

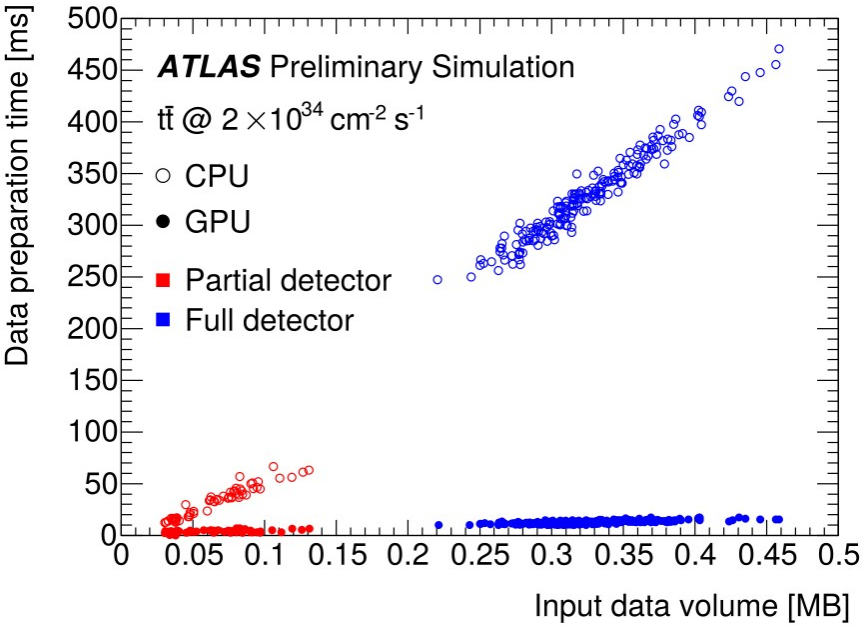


GPU-accelerated track reconstruction

Since track reconstruction is our most CPU-intensive task at high pileup, it's the focus of our attention. In theory it's quite suited to calculation on a GPU, since a relatively small amount of data (~10^5 space points) creates a very large number of different hypotheses to evaluate (~10^9 space point triplets), of which very few are accepted for further processing on the CPU (~10^4 track seeds)



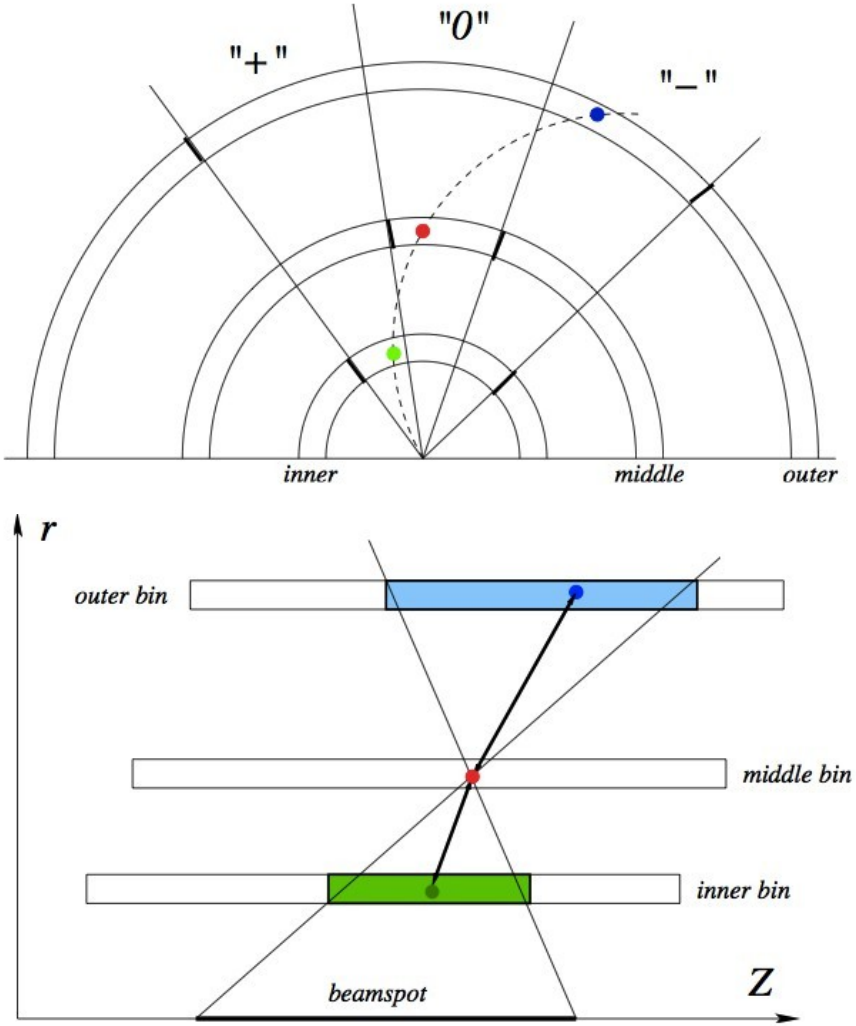
Can achieve a 26x speedup in the decoding of the raw data, and formation of the initial clusters



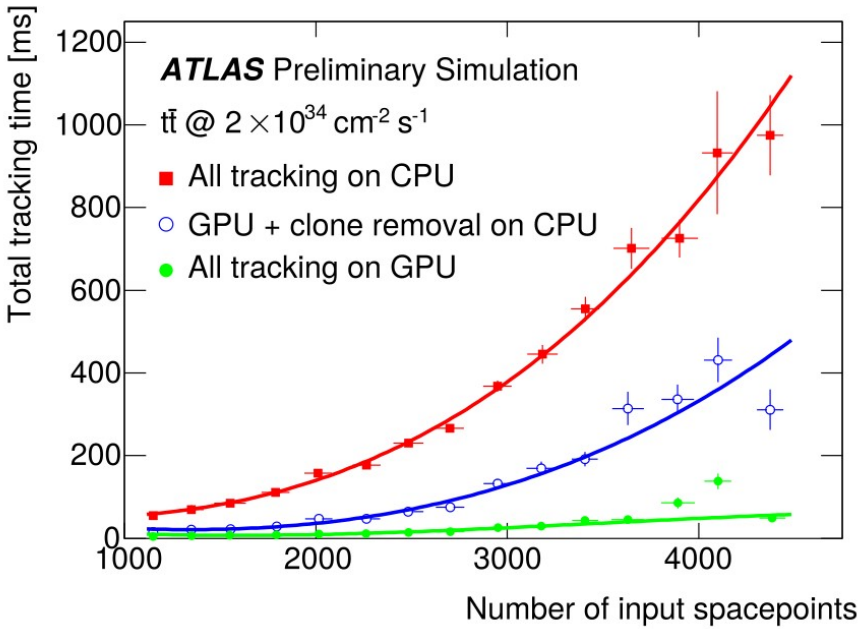
Nvidia C2060 versus Intel E5620 (st)

GPU-accelerated track reconstruction

Since track reconstruction is our most CPU-intensive task at high pileup, it's the focus of our attention. In theory it's quite suited to calculation on a GPU, since a relatively small amount of data (~10^5 space points) creates a very large number of different hypotheses to evaluate (~10^9 space point triplets), of which very few are accepted for further processing on the CPU (~10^4 track seeds)



Actual calculation of the track properties is more difficult, but still 12x speedup



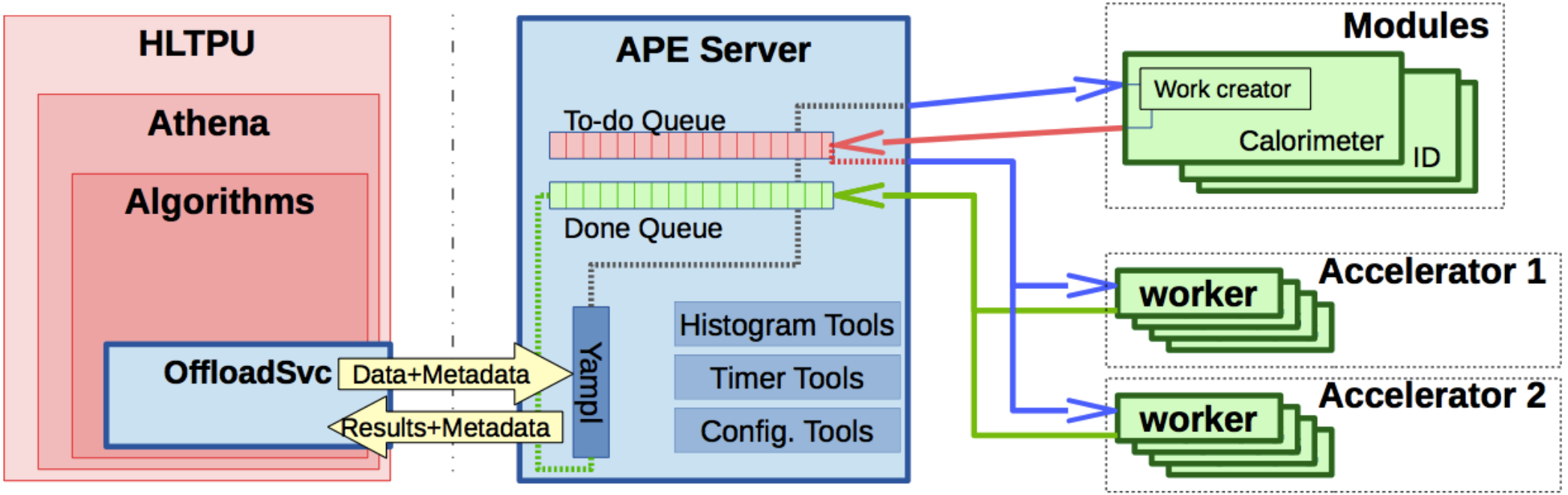
Nvidia C2060 versus Intel E5620 (st)

ATLAS HLT GPU demonstrator

While the potential is clearly there to speed up a specific algorithm by offloading it to a GPU, the effect on the performance of the overall system is not easy to determine

The GPU demonstrator project aimed to use these algorithms within a realistic HLT workflow in the Athena framework

Requires a mechanism for offloading algorithms: the APE server. This is a separate process outside of Athena, communicating using YAMPL. One APE server can manage multiple different accelerator devices, and serve multiple Athena clients



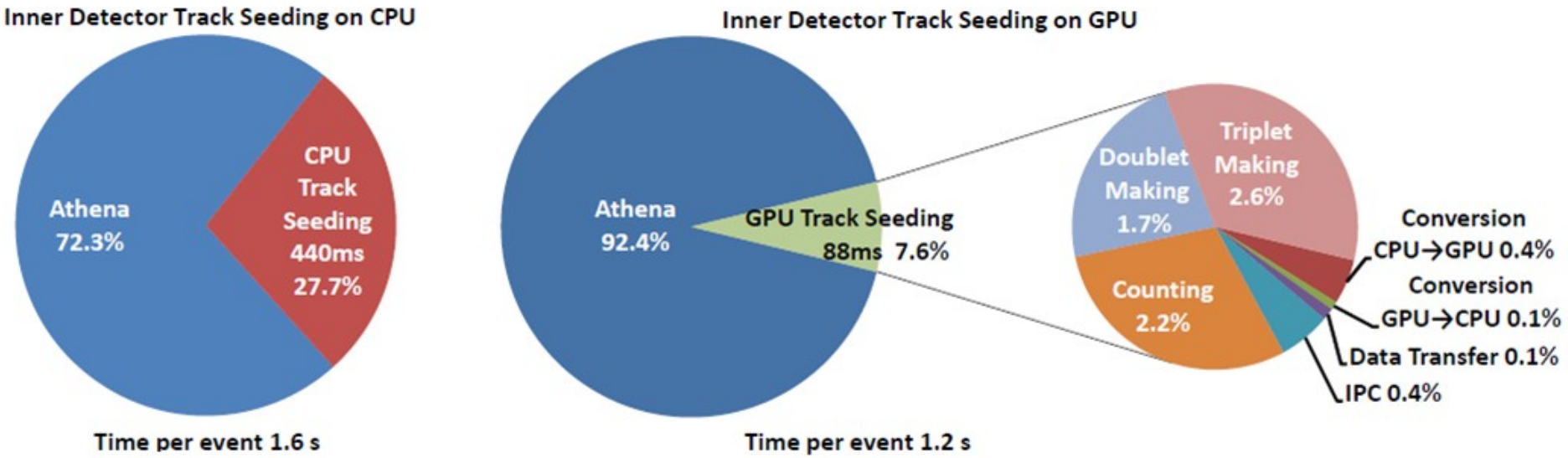
ATLAS HLT GPU demonstrator

While the potential is clearly there to speed up a specific algorithm by offloading it to a GPU, the effect on the performance of the overall system is not easy to determine

The GPU demonstrator project aimed to use these algorithms within a realistic HLT workflow in the Athena framework

Offloading the tracking to GPU works quite nicely: we see roughly a 5x speedup for that part of the job

Here CPU is Intel E5-2695 and GPU is Tesla K80

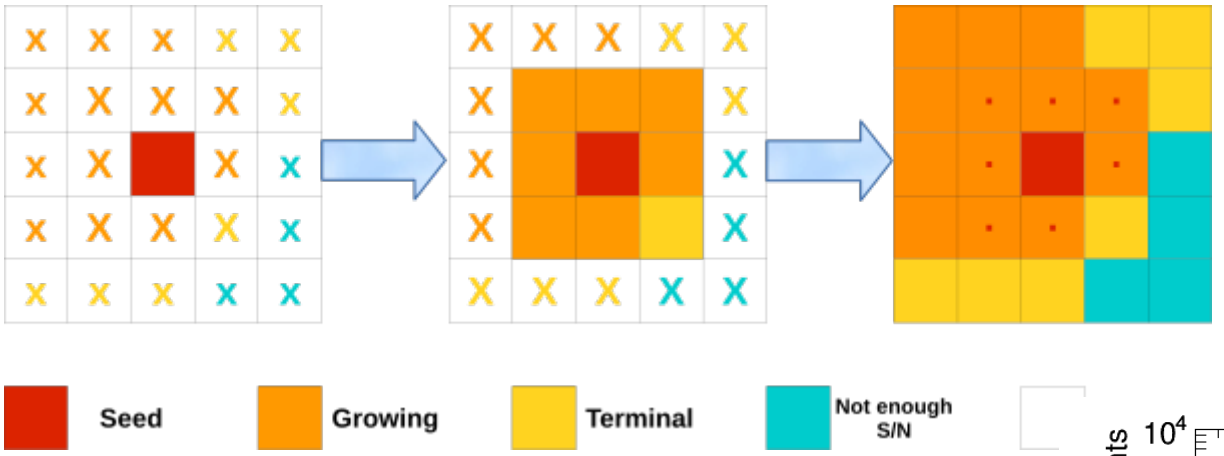


Relatively small overhead from the offload

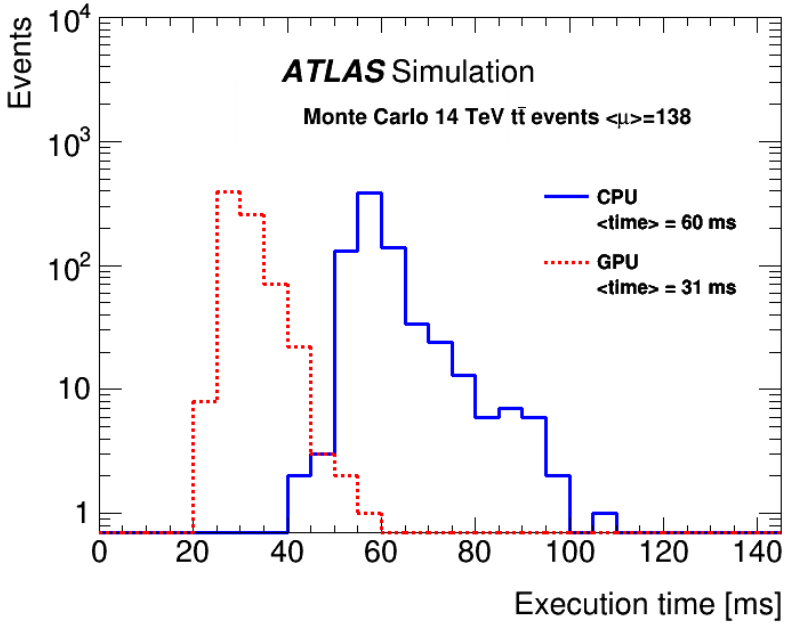
GPU-accelerated calorimeter clustering

We get close to an ideal speedup by offloading the track reconstruction, but overall performance is limited by the 70% of the total work remaining on the CPU

Try to offload additional algorithms: calorimeter clusters can be built up from initial seeds using a cellular automaton



Approximately 2x speedup for GPU (GTX650) vs CPU (AMD FX-8320)

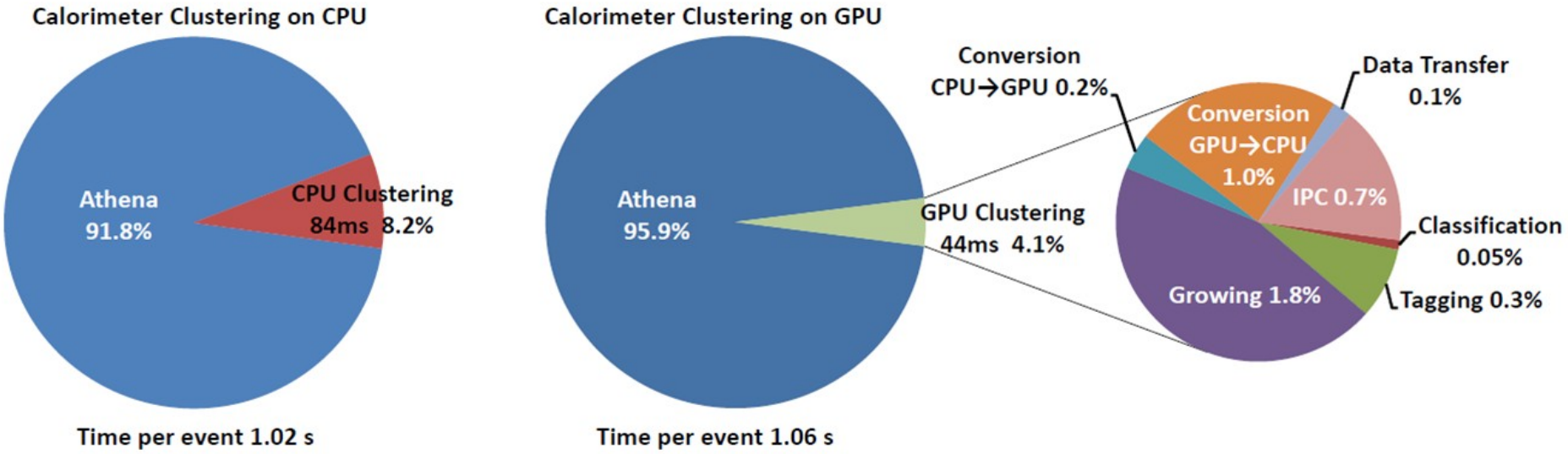


ATLAS HLT GPU demonstrator

While the potential is clearly there to speed up a specific algorithm by offloading it to a GPU, the effect on the performance of the overall system is not easy to determine

The GPU demonstrator project aimed to use these algorithms within a realistic HLT workflow in the Athena framework

The speedup of the calorimeter clustering algorithm was almost completely offset by the overhead associated with the GPU offload, and conversion of data structures



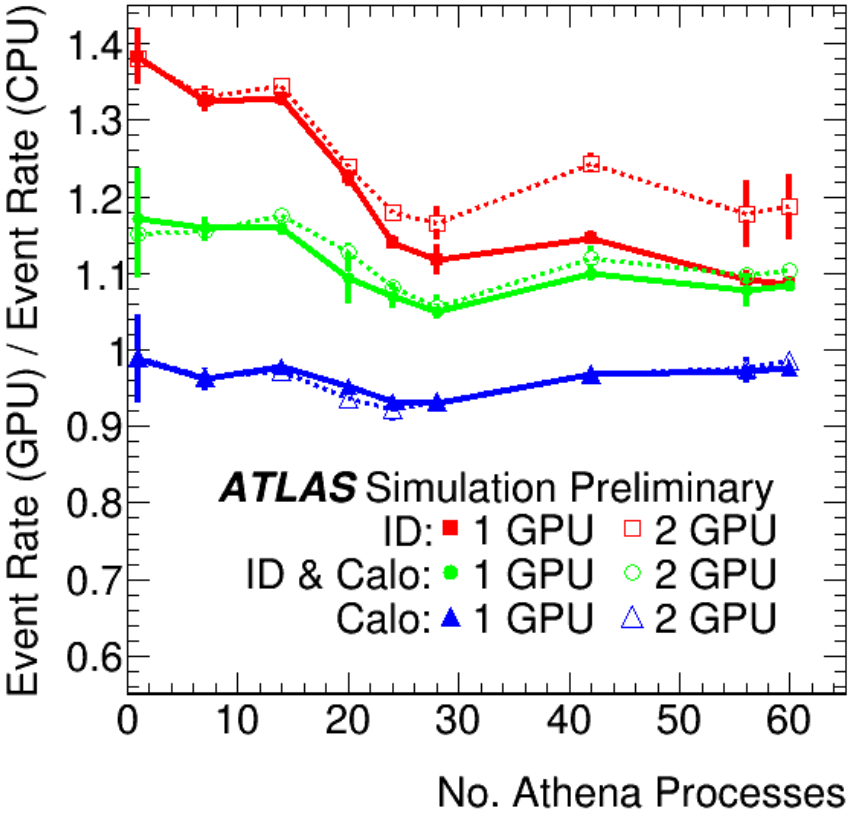
Much larger overhead, particularly from converting data structures

ATLAS HLT GPU demonstrator

While the potential is clearly there to speed up a specific algorithm by offloading it to a GPU, the effect on the performance of the overall system is not easy to determine

The GPU demonstrator project aimed to use these algorithms within a realistic HLT workflow in the Athena framework

Testing our APE server with multiple GPUs and multiple CPU processes as clients shows at best a 40% speedup available for 14 clients

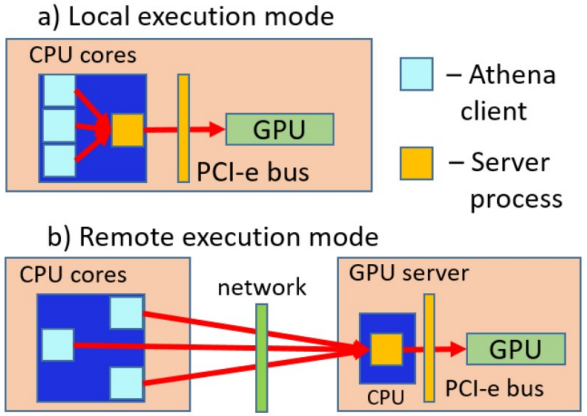


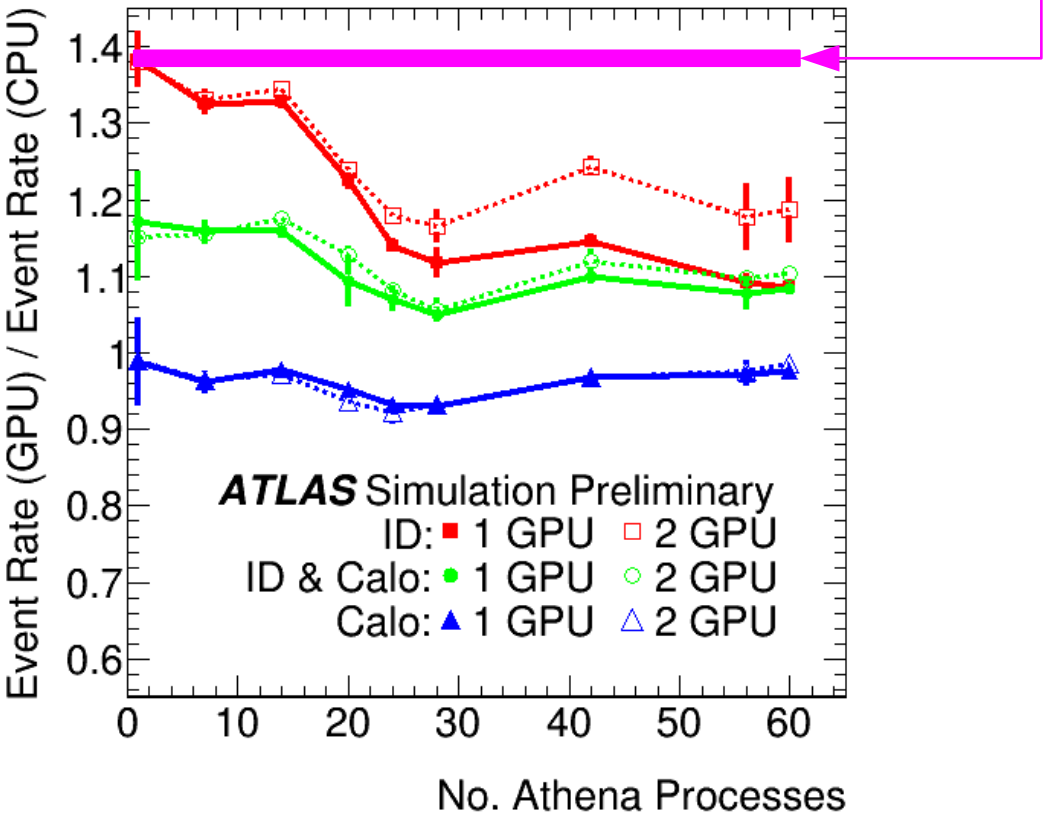
ATLAS HLT GPU demonstrator

While the potential is clearly there to speed up a specific algorithm by offloading it to a GPU, the effect on the performance of the overall system is not easy to determine

The GPU demonstrator project aimed to use these algorithms within a realistic HLT workflow in the Athena framework

More modern GPUs can support 50-60 Athena clients without saturating, where our CPU nodes communicate with the GPU host over the network



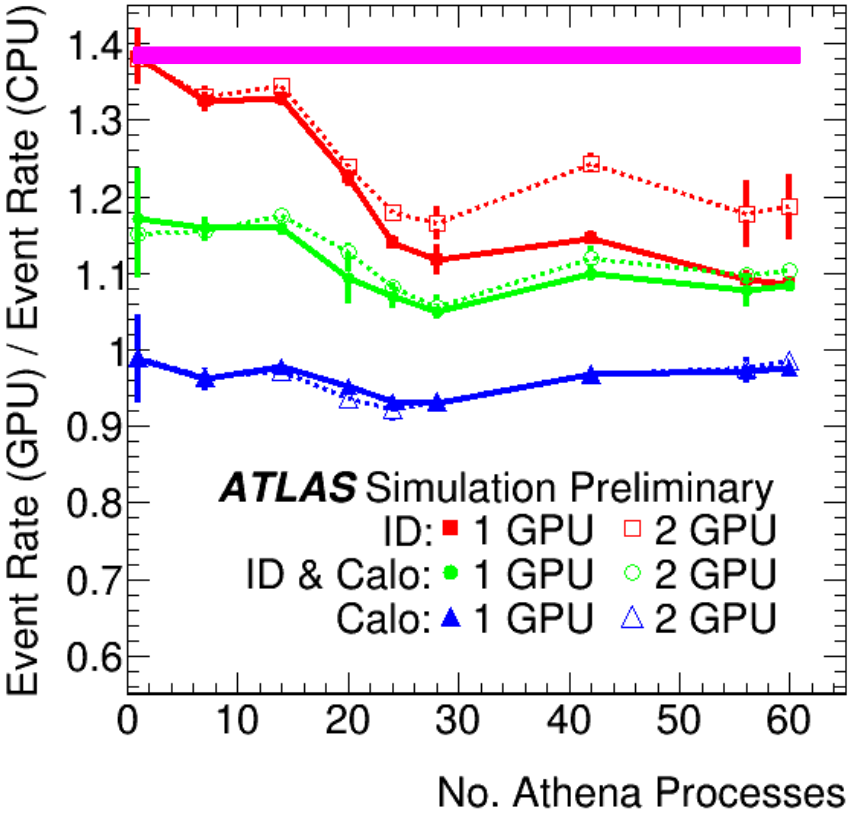


ATLAS HLT GPU demonstrator

While the potential is clearly there to speed up a specific algorithm by offloading it to a GPU, the effect on the performance of the overall system is not easy to determine

The GPU demonstrator project aimed to use these algorithms within a realistic HLT workflow in the Athena framework

Unfortunately, our break-even point for the HLT farm is about a 50% speedup
- Remember that we have to lose some CPUs to make room for GPUs



CUDA in Athena

The APE server made a clear separation between the Athena framework and the accelerator devices

This has a number of advantages, but also some downsides
- Overhead becomes significant for smaller offloaded tasks
- In the current model, the Athena algorithm/CPU thread is idle during the offloaded calculation

Currently prototyping CUDA code compiled as part of Athena itself

Ideally this will reduce overhead, and will also allow the use of asynchronous offloading (not available with APE), where we free the CPU thread while the GPU is running

These things should hopefully encourage development of GPU-accelerated versions of other algorithms, while also reducing the penalty incurred if these implementations are less than ideal

Current issues:
- compilation is tricky since CUDA doesn't understand C++17
- our asynchronous offloading doesn't work that well at present

Single-source development

We have enough trouble maintaining one version of our algorithms!

In fact one of the motivations behind our current Athena upgrade was to increase code-sharing between the HLT and our offline reconstruction

Because we have full control of the HLT hardware we do have the option to specialise, but there has not been a compelling reason to do so yet

In the meanwhile, offline reconstruction must run on as many different architectures as possible

OpenMP and OpenACC are incompatible with our framework multithreading (based on Intel TBB)

Currently hoping SYCL will allow CPU/GPU/FPGA portability

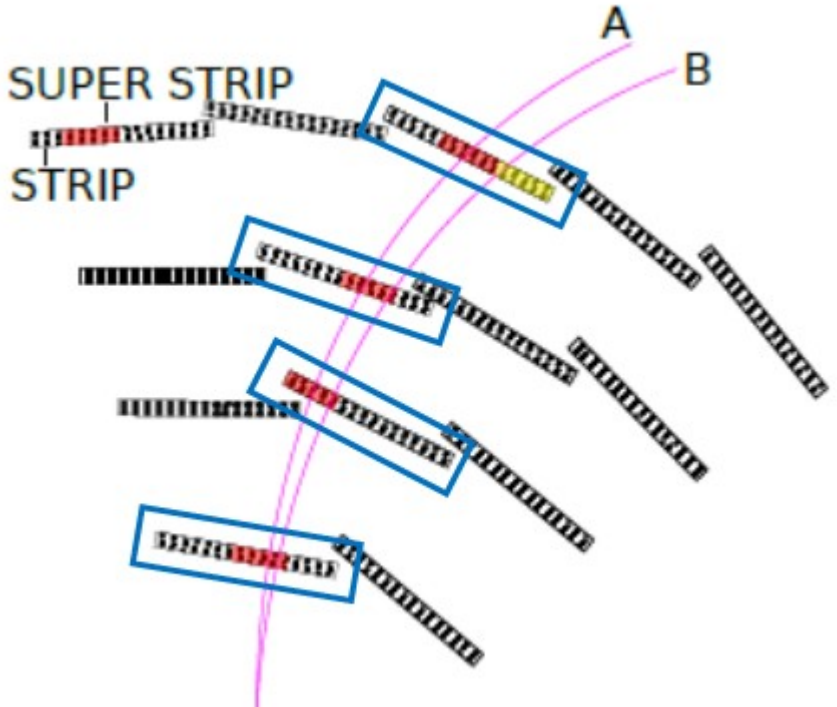
Future projects

The intention for the ATLAS phase 2 upgrade is to use the proposed Hardware Track Trigger (HTT) to perform track reconstruction for the HLT

Pattern-matching approach using custom hardware (FPGA+ASIC)

Given the complexity and expense of such a project, we are currently investigating whether we might be able to achieve similar performance with commodity hardware

We can try an identical pattern-matching method on GPU, but there are limitations arising from total device memory, and memory bandwidth
- can be offset by compression to some extent



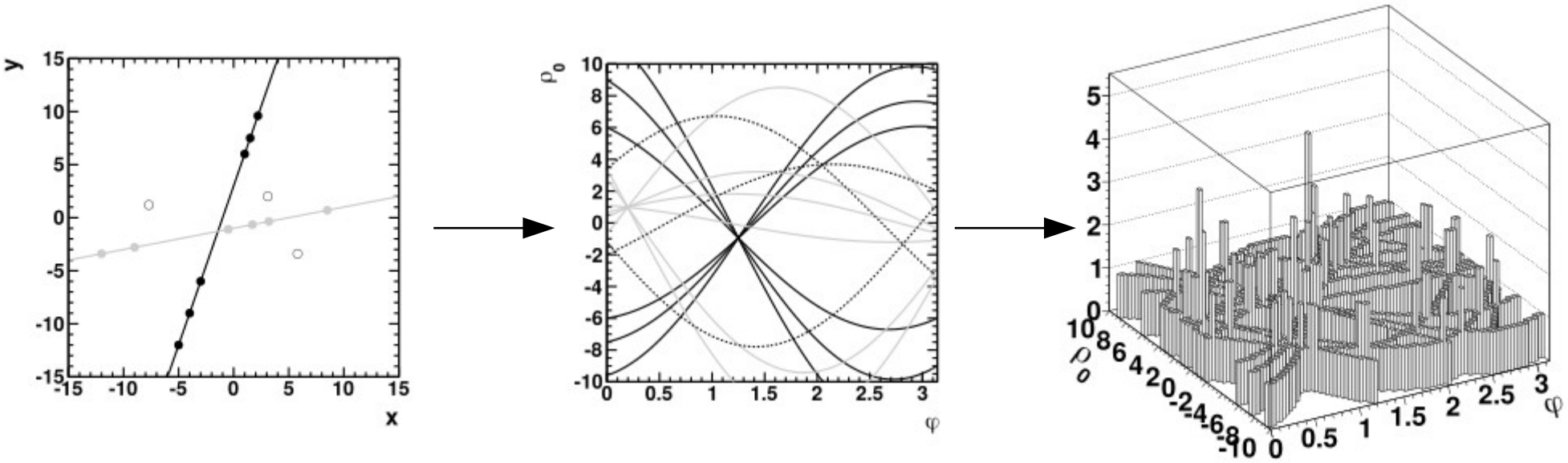
Future projects

The intention for the ATLAS phase 2 upgrade is to use the proposed Hardware Track Trigger (HTT) to perform track reconstruction for the HLT

Pattern-matching approach using custom hardware (FPGA+ASIC)

Given the complexity and expense of such a project, we are currently investigating whether we might be able to achieve similar performance with commodity hardware

An alternative technique uses Hough transforms: convert each tracker hit in the basis of our detector geometry to the basis of track hypotheses, and then look for intersections that might correspond to real tracks



Summary

There has been interest in GPU development in HEP for a long time, based on demonstrations of dramatic speedup for particular algorithms

Track reconstruction is one of our most CPU-intensive tasks, and is also quite amenable to GPU acceleration

However, other algorithms are much less convenient, and collectively still make up the majority of total job time
- limits overall speedup from GPU offload, currently not worthwhile

Potential future development could improve the situation
- Single-source development (SYCL) reducing maintenance headache?
- Offloading directly from Athena might reduce overheads
- General improvement of our data model? We can dream…

Search for HTT alternatives motivating a new round of GPU development

References

Public:

https://twiki.cern.ch/.../TriggerSoftwareUpgradePublicResults

https://cds.cern.ch/.../ATL-DAQ-SLIDE-2016-837.pdf

https://cds.cern.ch/.../TriggeringEventswithGPUatATLAS.pdf

https://cds.cern.ch/.../ATL-DAQ-SLIDE-2014-635.pdf

https://cds.cern.ch/.../ATL-SOFT-SLIDE-2019-809.pdf

ATLAS internal:

https://cds.cern.ch/.../ATL-COM-DAQ-2019-059.pdf

https://cds.cern.ch/.../ATL-COM-DAQ-2019-173.pdf