Ongoing GPU work at RAL

Jyoti Prakash Biswal

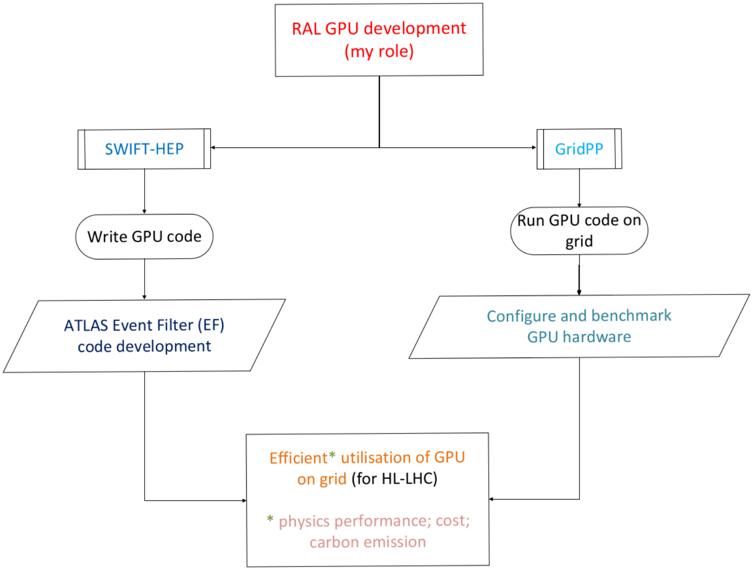
Rutherford Appleton Laboratory

SWIFT-HEP #8 (jointly with ExaTEPP)

Warwick University

11-12 November 2024

Overview

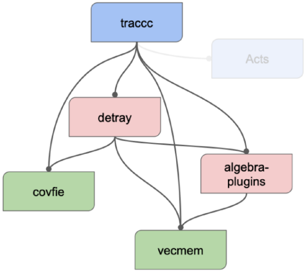

RAL GPU development (my role)
SWIFT-HEP
GridPP
Write GPU code
Run GPU code on grid
ATLAS Event Filter (EF) code development
Configure and benchmark GPU hardware
Efficient* utilisation of GPU on grid (for HL-LHC)
* physics performance; cost; carbon emission

† Sam and Alison have already presented on the CMS FPGA work at RAL.

traccc synopsis

- The ACTS (A Common Tracking Software) project aims to integrate reliable and efficient tracking software while adapting its modules for contemporary computing systems and ensuring sustainable code maintenance.
- *traccc* is a comprehensive tracking implementation for GPUs, developed independently of experiments but with significant ATLAS involvement.
    - It has evolved from ACTS, with plans for re-integration.
- The work is divided into several projects, each focused on specific tasks.

- **detray**: tracking geometry handling.
- **covfie**: generic vector field handling
- **algebra-plugins**: small matrix linear algebra abstractions.
- **vecmem**: common memory management.
- ***traccc***: main repository, mostly algorithmic code.



The primary objective is to show that track reconstruction can be performed on GPUs while maintaining full reconstruction and physics quality.

What is **EF** tracking in ATLAS?

- The high pile-up conditions planned for the High Luminosity LHC (HL-LHC) give rise to specific challenges for the **trigger** algorithms.
- Event Filter (EF) tracking will run on a farm of commodity CPUs with/without GPUs/FPGAs.
- EF tracking reconstruction chain:
  1. An initial data preparation stage – Inner Tracker (ITk) data preparation.
  2. Seed formation and pattern recognition step (track Seeding and pattern recognition).
  3. Track extension, fitting and ambiguity resolution.
- Robust against pile-up: capable of reconstructing vertices and tracks needed for pile-up suppression.
- The overall tracking efficiency should be close to offline efficiency.

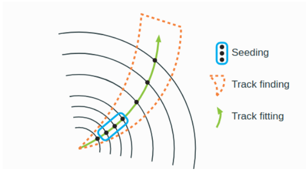
Why is *traccc* useful for SWIFT-HEP?

- Like SWIFT-HEP, *traccc* focuses heavily on GPU development.
- Getting involved with *traccc* will be handy in –
  - Getting familiar with the functioning of the GPU codes in use.
  - Contributing to the ATLAS GPU effort.
  - Gaining experience for future developments.
- In a nutshell, *traccc* provides most of the areas for GPU software workflow.

EF tracking *traccc* pipelines

- **Athena** is the ATLAS software framework that manages almost all ATLAS production workflows: event generation, simulation, reconstruction and derivation production.
    - It is also used online in the ATLAS High Level Trigger (HLT).
- Two pipelines:
    - Pass clusters to *traccc* directly in Athena and perform tracking, *i.e.*, *traccc* with Athena.
    - Create *traccc* input files and test performance in *traccc* standalone.
- Full reconstruction chain in Athena:

$$\boxed{\text{clusters}} \rightarrow \boxed{\text{seeds}} \rightarrow \boxed{\text{track finding}} \rightarrow \boxed{\text{track fitting}} \rightarrow \boxed{\text{ambiguity resolution}}$$



- Both pipelines have merits –
    - Athena will make things easier from the ATLAS point of view.
    - The standalone part has its applicability for the experiment-independence: Open Data Detector (ODD), Tracking Machine Learning (TML) data.
- ATLAS is also looking at CPU-only and FPGA-accelerator solutions/pipelines (along with GPU).

traccc work @RAL

- In the last few months, both the pipelines have been successfully tested on (1) lxplus-gpu and (2) hepacc (RAL GPU).
    - The hepacc machines have different GPU models than the ones at lxplus.
    - CUDA environment is the main catch!
    - Also, *traccc* keeps evolving $\Rightarrow$ the pipeline tests on different GPUs are non-trivial.
- The standalone version, for ATLAS testing, requires pre-made spacepoints.
    - ITk geometry via detray is used.
    - The spacepoints are produced via non-GPU codes.
- The Athena-one works on RAL GPU machines after the proper CUDA library environment is set up.

Possible areas for contribution

- Performance measurements.
- Optimisation of the pipelines – what to look at, understanding of the code, bottlenecks...
- Granularity – how many blocks to run?
- **Note:** although many people are involved in this work, it is a good way to get inducted to the project.
- All these are also beneficial for future main work.
- More specific/independent areas are being discussed.

ATLAS GPU queue @RAL-Tier1

- **Aim:** to set up an ATLAS GPU queue at RAL.
- **Why is this a challenge?**
  - The GPUs need to be provisioned from the STFC Cloud.
  - STFC Cloud GPUs are under high demand $\Rightarrow$ Can't be simply kept for long!
  - Processing STFC Cloud GPU in Batch Farm is not trivial.
  - In collaboration with the team from the Scientific Computing Department (SCD) at RAL.

During the ongoing quarter: run a job on the Tier1 Batch Farm that utilises GPU workloads (step-wise details below):

GPU job types

- CUDA $\sim$ 12.4.
- Relevant Python version(s).
- Potential use of CVMFS for binary distribution.
- 8.5.0 gcc or newer.

Resources needed: Grid Services

- Restart conversations with the STFC Cloud team.
- Condor queue creation/routing for GPU.
- Creation of ARC-CE queue for ATLAS to submit to.

Tier1 deliverables

- Utilisation of CUDA logic to print "Hello, World!“
- Accounting by wallclock time.

Resources needed: ATLAS

- Medium long-term GPU model (PyTorch, TensorFlow, or basic CUDA).
- Timescales.

GPU *test* queue @RAL-PPD (Tier2)

- Several GPU machines are at the Particle Physics Department (PPD) – the hepacc's [not that up-to-date link].
- Not all of them are occupied all the time. Hence, the idea/proposal was to build an ATLAS-specific GPU queue out of these machines.
- The GPU queue (heplnx206.pp.rl.ac.uk:2811/nordugrid-Condor-gpu) in question is aimed to use the LHC grid jobs – initially by the PPD members (once tested and verified) and eventually by others.
- It is catering to local testing (expertise) – a job has already been successfully submitted and tracked via *arcsub*.
- In the recent past, the GPU queue was added on CRIC (hence, ATLAS BigPanda):
  - Name: UKI-SOUTHGRID-RALPP_GPU.
  - Links: CRIC; ATLAS BigPanda.
  - Currently in TEST mode – jobs are being analysed.
  - The setup is largely borrowed from the existing Manchester GPU queue.

Investigation is ongoing to make the queue full-fledged.

CMS GPU Update

via Sam Harper

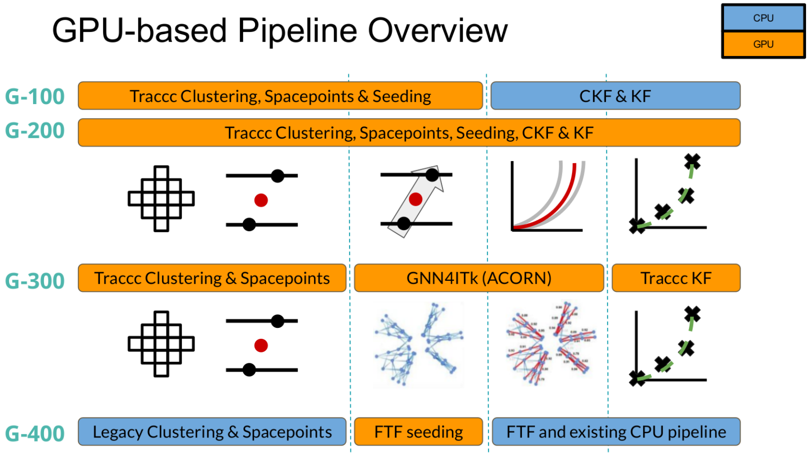
- CMS has migrated fully from CUDA to **alpaka** at the HLT.
  - Running in production at Point 5 this year.
- CMS has developed a common data format syntax.
  - Structures of Arrays (SoA), but looks like Arrays of Structures (AoS).
- Efforts are ongoing to port more and more algorithms.
  - Have pixel tracking, Electromagnetic Calorimeter (ECAL) local reconstruction, Hadron Calorimeter (HCAL) local reconstruction, and HCAL Particle Flow (PF) clustering.
  - Porting ECAL local calibrations, electron seeding, and primary vertex (PV) reconstruction.
- RAL CMS leads the **ECAL** **alpaka** code development – there is significant experience in the group now.
- Recent CHEP talk on this topic.

Outlook

- The *traccc*-related work is coming together, and more ideas are to follow.
- After some initial delay, the Tier1 GPU queue project has progressed, and we hope to keep up the pace.
- The Tier2 GPU is "almost“ there – the current issues are being looked into.
- The CMS group is doing their part.
- Other GPU-related stuff (if any) will be explored as they come by.

Backup

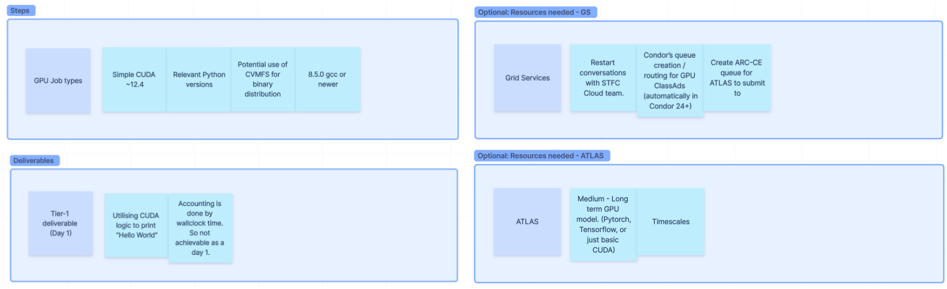
GPU-based Pipeline Overview


CPU
GPU
G-100
Traccc Clustering, Spacepoints & Seeding
CKF & KF
G-200
Traccc Clustering, Spacepoints, Seeding, CKF & KF
G-300
Traccc Clustering & Spacepoints
GNN4ITk (ACORN)
Traccc KF
G-400
Legacy Clustering & Spacepoints
FTF seeding
FTF and existing CPU pipeline

- G-200 is a full end-to-end GPU pipeline, using *traccc*.
- G-100 is the same approach as G-200 but returns seeds only.
- G-300 replaces *traccc* components with the Graph Neural Network (GNN).
- G-400 uses existing Athena GPU code.

]

ATLAS GPU queue @RAL-Tier1



Steps

GPU Job types | Simple CUDA ~12.4 | Relevant Python versions | Potential use of CVMFS for binary distribution | 8.5.0 gcc or newer

Optional: Resources needed - GS

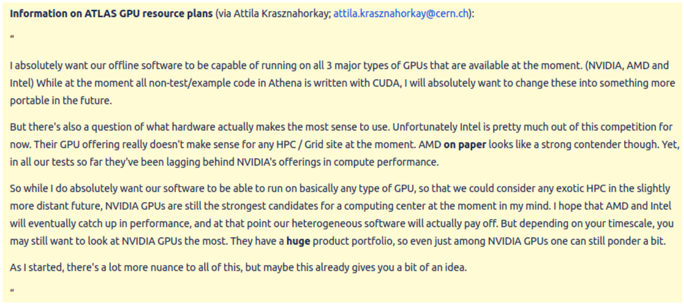
Grid Services | Restart conversations with STFC Cloud team. | Condor's queue creation / routing for GPU ClassAds (automatically in Condor 24+) | Create ARC-CE queue for ATLAS to submit to

Deliverables

Tier-1 deliverable (Day 1) | Utilising CUDA logic to print "Hello World" | Accounting is done by wallclock time. So not achievable as a day 1.

Optional: Resources needed - ATLAS

ATLAS | Medium - Long term GPU model. (Pytorch, Tensorflow, or just basic CUDA) | Timescales



Information on ATLAS GPU resource plans (via Attila Krasznahorkay; attila.krasznahorkay@cern.ch):

"

I absolutely want our offline software to be capable of running on all 3 major types of GPUs that are available at the moment. (NVIDIA, AMD and Intel) While at the moment all non-test/example code in Athena is written with CUDA, I will absolutely want to change these into something more portable in the future.

But there's also a question of what hardware actually makes the most sense to use. Unfortunately Intel is pretty much out of this competition for now. Their GPU offering really doesn't make sense for any HPC / Grid site at the moment. AMD **on paper** looks like a strong contender though. Yet, in all our tests so far they've been lagging behind NVIDIA's offerings in compute performance.

So while I do absolutely want our software to be able to run on basically any type of GPU, so that we could consider any exotic HPC in the slightly more distant future, NVIDIA GPUs are still the strongest candidates for a computing center at the moment in my mind. I hope that AMD and Intel will eventually catch up in performance, and at that point our heterogeneous software will actually pay off. But depending on your timescale, you may still want to look at NVIDIA GPUs the most. They have a **huge** product portfolio, so even just among NVIDIA GPUs one can still ponder a bit.

As I started, there's a lot more nuance to all of this, but maybe this already gives you a bit of an idea.

"