GPU developments @RAL-Tier1

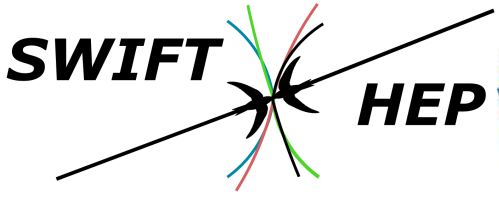
Jyoti Prakash Biswal

Rutherford Appleton Laboratory

GridPP52 Collaboration Meeting

University of Cumbria, Ambleside Campus

28-30 August 2024



GPUs: An overview

Why do we need GPUs?

- GPUs are designed for parallel processing (the number of cores on each chip is much denser than the CPU), which allows them to handle multiple tasks at once.
- Useful for **high computing performance**, **accelerating AI and ML tasks**, **video and graphics processing**, and scientific computing.
- The price is a lot more, **but the performance is a lot better**.

GPU vendors

- Three foremost market leaders: AMD, Intel, and Nvidia.
- As of June 2024, Nvidia's market share is $\sim$ 88%, and AMD's is $\sim$ 12%. Intel is almost negligible!
- Quadro RTX 5000 (Nvidia); Radeon RX 7600 (AMD); Iris Pro Graphics 580 (Intel).

GPU programming languages

- CUDA (developed by Nvidia).
- HIP (Heterogeneous Interface for Portability) – by AMD.
- oneAPI DPC++ (supports Intel GPUs natively).
- Kokkos, Alpaka, SYCL, ...
- High-level language support: Python, CuPy, Numba, ...

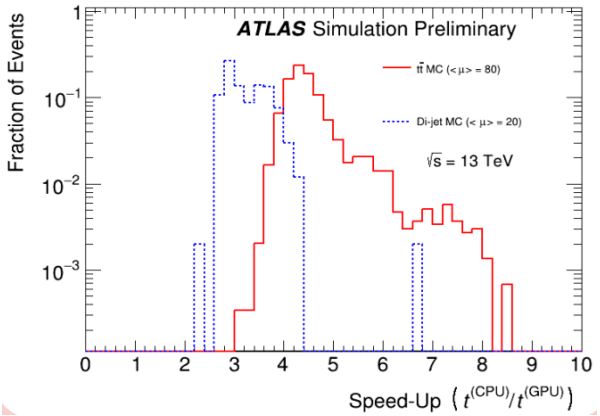
How to choose GPUs?

1. Throughput (the amount of operations that can be processed per unit time) improvement.
2. Running cost (power consumption).
3. Purchase cost.
4. Development cost.

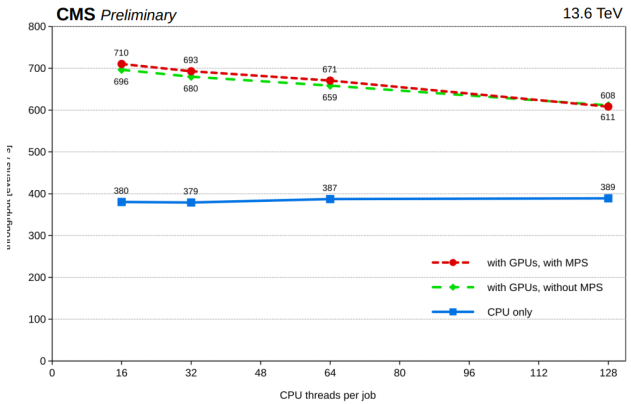
- Your decision can be influenced by how you prioritise these factors!

GPUs from an LHC perspective

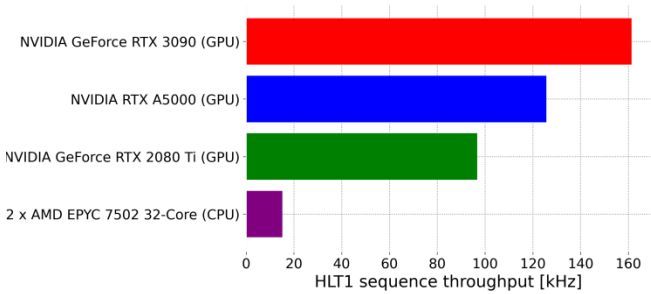
- The four major LHC experiments have been evaluating solutions integrating GPUs for a while –
  - Online/Offline data reconstruction.
  - Fast physics simulation.
  - Machine/Deep Learning applications (*e.g.*, analysis).
- Most general deployments are online processing and high-level trigger (HLT) – track seeding, track fitting, and vertexing (ref.). **The LHC experiments have GPUs in operation at their HLT farms**.
- Main features for High Energy Physics (HEP) algorithms to be ported:
  - Static, predictable workflows: minimal control flow branches (*i.e.*, if-else statements).
  - Intelligent usage of the local memory to minimise **latency** (within the internal network).



ATLAS: GPU-accelerated topological clustering (calorimeter) – Source.



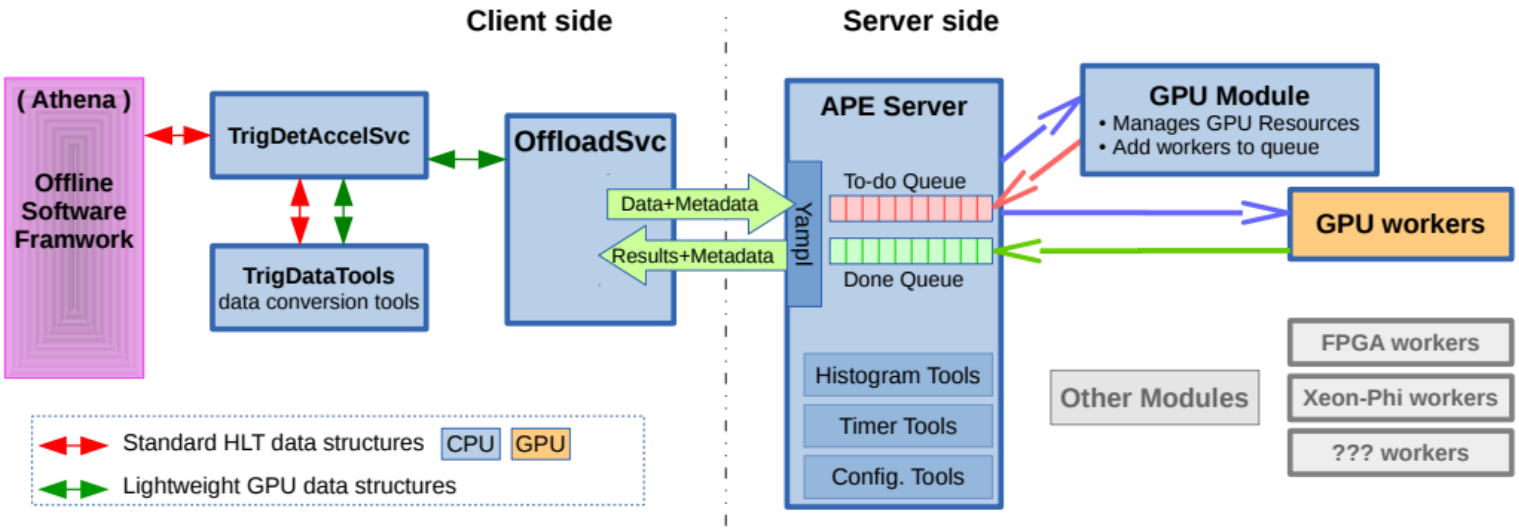
CMS: HLT throughput, measured under different conditions – Source.
MPS: Multi-Process Server.



LHCb: The throughput of the reference HLT1 sequence for three different GPUs – Source.

ATLAS and GPUs

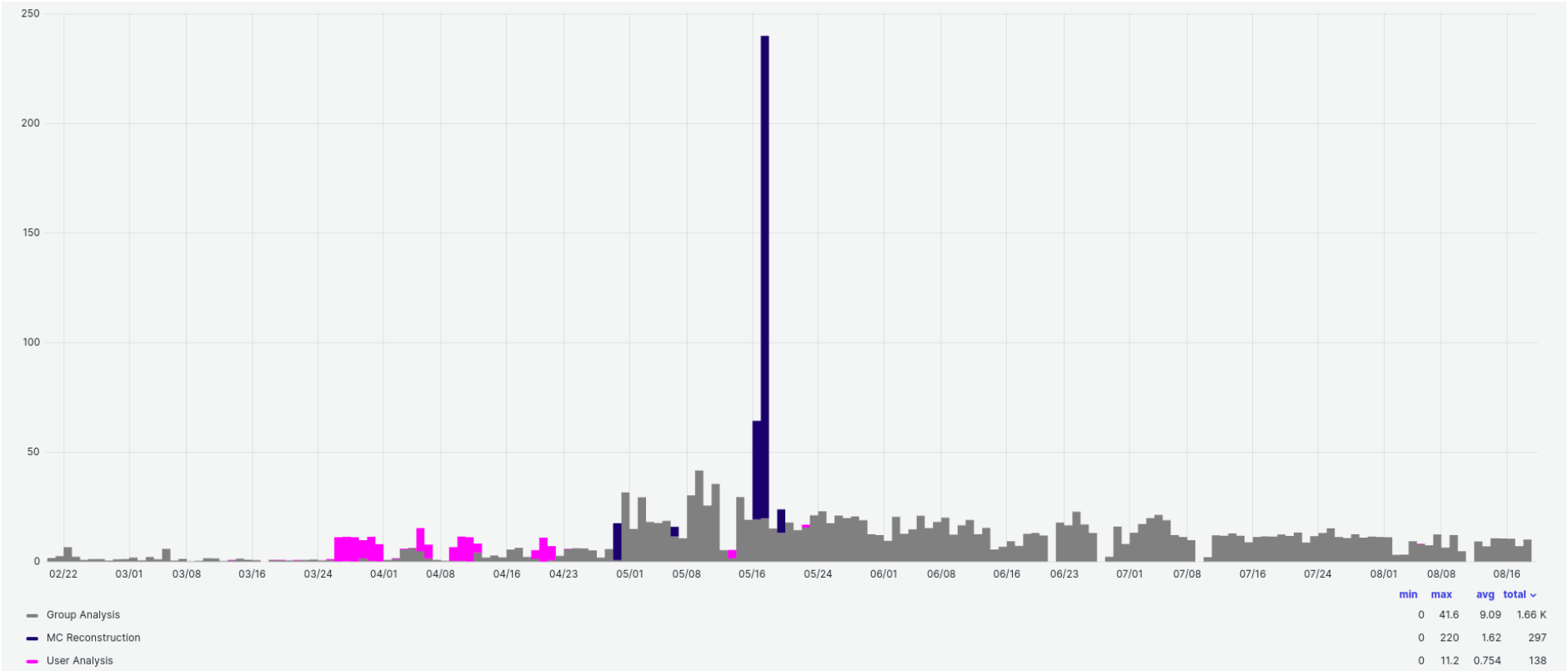
- Two major directions:
  - traccc: an open (but extensive ATLAS involvement) project on R&D for the use of GPUs in tracking.
    - I am involved in this (joined the effort recently).
  - HLT efforts for Phase-II in ID tracking and calorimetry.
    - It is also part of the decision to use either FPGA, GPU or CPU in the HLT farm for Phase-II.
    - ATLAS Phase-II Upgrade Scoping Document.
    - ATLAS Trigger on GPUs.



ATLAS framework (Athena) clients request offloading to the *TrigDetAccelSvc*. *TrigDetAccelSvc* uses *TrigDataTools* for data conversions between the client and server structure. The module appends the data to a unique data space and gives it to a new worker. The worker is appended to a to-do queue that is managed by the server.

Existing UK ATLAS GPU queues – Manchester

- **UKI-NORTHGRID-MAN-HEP_GPU**: BigPanda; CRIC.
- ANALY_MANC_GPU (the original queue running only analysis jobs): BigPanda; CRIC.
- monit-grafana (combined).
- The plots shown below have 1-day binning and cover the past **six months**.

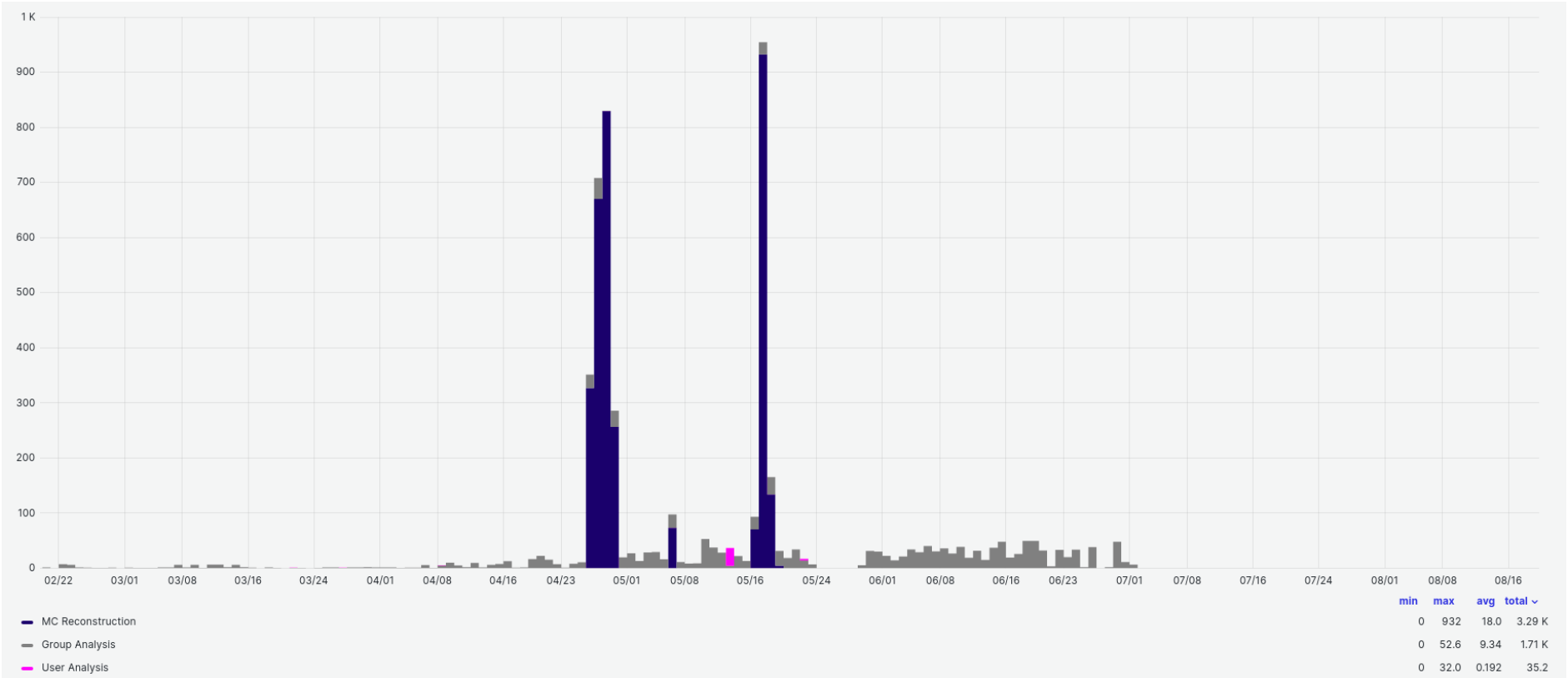


Slots of running jobs.

Note: The GPU queues were unified this year to run production jobs. The unified queue in Manchester seems to have problems – being investigated. Regular nightly tests are submitted with GPU code ⇒ the HLT reprocessings there would be good, too!

Existing UK ATLAS GPU queues – QMUL

- **UKI-LT2-QMUL_GPU**: BigPanda; CRIC.
- ANALY_QMUL_GPU (the original queue running only analysis jobs): BigPanda; CRIC.
- monit-grafana (combined).
- The plots shown below have 1-day binning and cover the past **six months**.



Slots of running jobs.

Note: The GPU queues were unified this year to run production jobs. Due to the Data Centre refurbishment @QMUL, not much has happened. Regular nightly tests are submitted with GPU code ⇒ the HLT reprocessings there would be good, too!

ATLAS GPU queue @RAL-Tier1

- **Aim:** to set up an ATLAS GPU queue at RAL.
- **Why is this a challenge?**
  - We do not have any GPUs yet. The GPUs will be provisioned from the STFC Cloud.
  - STFC Cloud GPUs are under high demand ⇒ Can't be simply kept for long!
  - Processing STFC Cloud GPU in batch farm is not trivial.
  - In collaboration with **Thomas Birkett** & co.
- The whole procedure involves some significant architectural changes at the Tier1.
  - This is to have the Coyote fully functional – it is not running regularly at the moment!
  - Coyote deals with the batch bursting at the Tier1.
    - Coyote is a Python script that interacts with the OpenStack APIs, creates worker nodes, and runs at a defined interval.

      [Rooster was the system that woke up the machines. *Coyote eats roosters, hence the naming.*]

ATLAS GPU queue @RAL-Tier1: A timeline

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

- There are quite a few GPU machines at PPD – the hepacc's [not that up to date link].
- Not all of them are occupied all the time, hence, the idea/proposal is to build an ATLAS-specific GPU queue out of these machines.
- The GPU queue in question is aimed to use the ATLAS grid jobs – initially by the PPD members (once tested and verified), and eventually by others.

The first quarter (until mid-October 2024)

- Setting up of one or two VMs.
- Addition of these VMs to the preproduction batch farm.
- In the meantime, a VM could be provisioned on the STFC Cloud for job (CUDA/non-CUDA) submission.

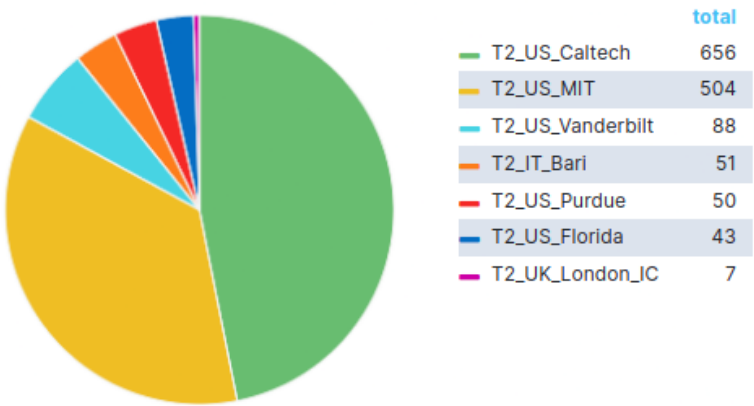
The second quarter (until Christmas 2024)

- Panda queue configuration and debugging.
- Functionally, get the queue running.
- Jobs are to be submitted via ARC.

The third quarter (until April 2025)

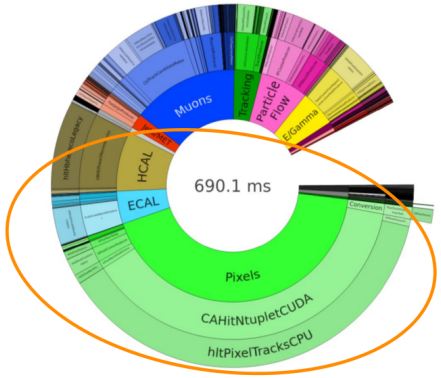
- Run the jobs at different sites and observe the run time.
- Could help in setting up the GPUs for other Tier2 sites.
- By this time, there will be more GPU codes for testing.

GPUs in CMS and LHCb

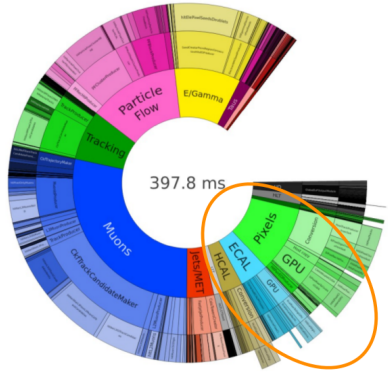


CMS: **GPU pool size per resource provider** (last ∼1 year) – Link to CMS GPUs monitor.

US is doing a lot, but it is nice to see Imperial (UK) up there.



No GPU, 32 threads, 24 streams



GPU, no MPS, 32 threads, 24 streams

CMS: **HLT with GPU offloading (2022 numbers)** – Source.

40% less time per event.

20% better performance per initial cost.

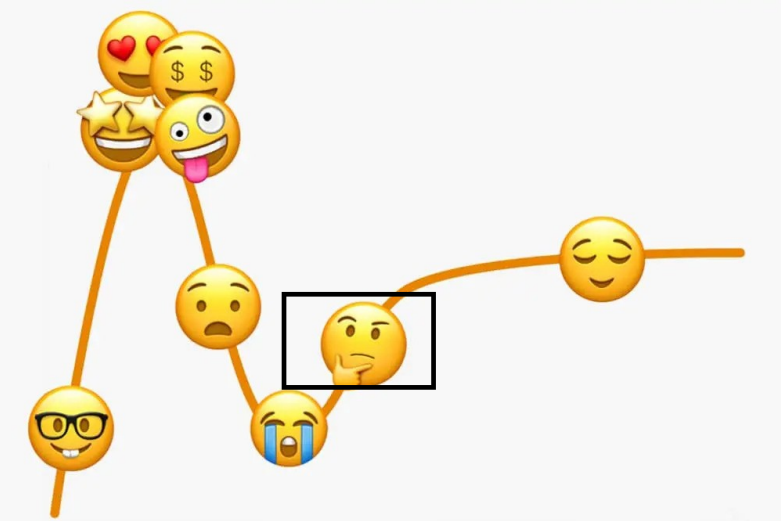
LHCb

- LHCb does not use GPUs for offline processing.
- They have one site (Italy) that provides GPUs for user jobs, but it is not used.
- However, GPUs (175 of them; ref.) are used for HLT.

- CMS and LHCb now have big GPU deployments at their HLTs.
  - CMS uses only CPUs for regular **grid jobs**.
  - LHCb follows the same approach as CMS when it comes to **grid jobs**.

Future outlook

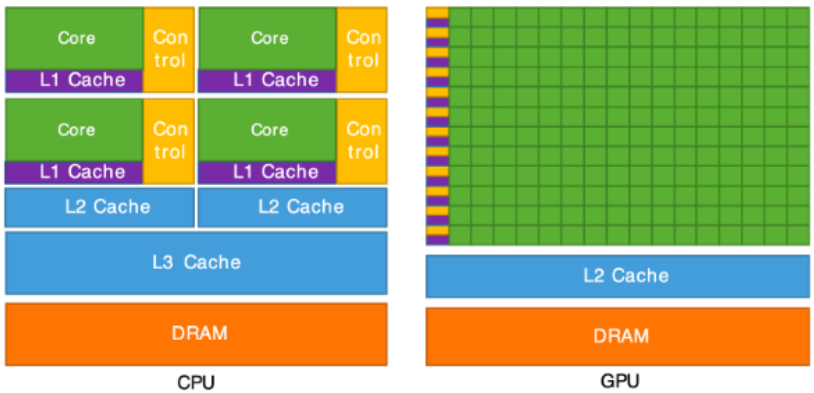
- There are enough GPU codes that we can viably run on the grid.
- The research will culminate in the next year or so – how much GPUs will be incorporated into the workflow.
- While we are still a few years away from achieving GPU benchmarking, we are making significant strides in understanding what the GPU setup will look like.
- We are trying to focus on the learning curve, *i.e.*, the slope of enlightenment from the Gartner Hype Cycle (below).



Backup

General information on GPUs

- **What is a GPU?**
  - **G**raphics **P**rocessing **U**nit.
  - GPUs have many efficient cores for parallel processing.
  - GPUs gain high speed and efficiency by using multiple smaller cores simultaneously for specialised computational tasks.
- **How does a GPU work?**
  - Overall, GPUs are designed with parallelism and Floating Point Operations Per Second (FLOPS) in mind.
  - Thousands of cores allow numerous Arithmetic Logic Units (ALUs) to process various data types simultaneously.
  - Data in the GPU pipeline spreads across simpler cores, unlike the CPU's sequential tasks on complex cores.
- **Usages of GPU:** Gaming/graphics; Artificial Intelligence (AI); Scientific computing; ...



Main differences in hardware architecture between CPU (left) and GPU (right) –

computation; instruction processing; L1 cache; higher-level cache; memory (DRAM-Dynamic random-access memory).