Software frameworks for HL-LHC reconstruction

Andy Morris

On behalf of the Alice, Atlas, CMS and LHCb collaborations

Aix Marseille Univ, CNRS/IN2P3, CPPM, Marseille, France

Andy.M@cern.ch – [6/Jun/2024]

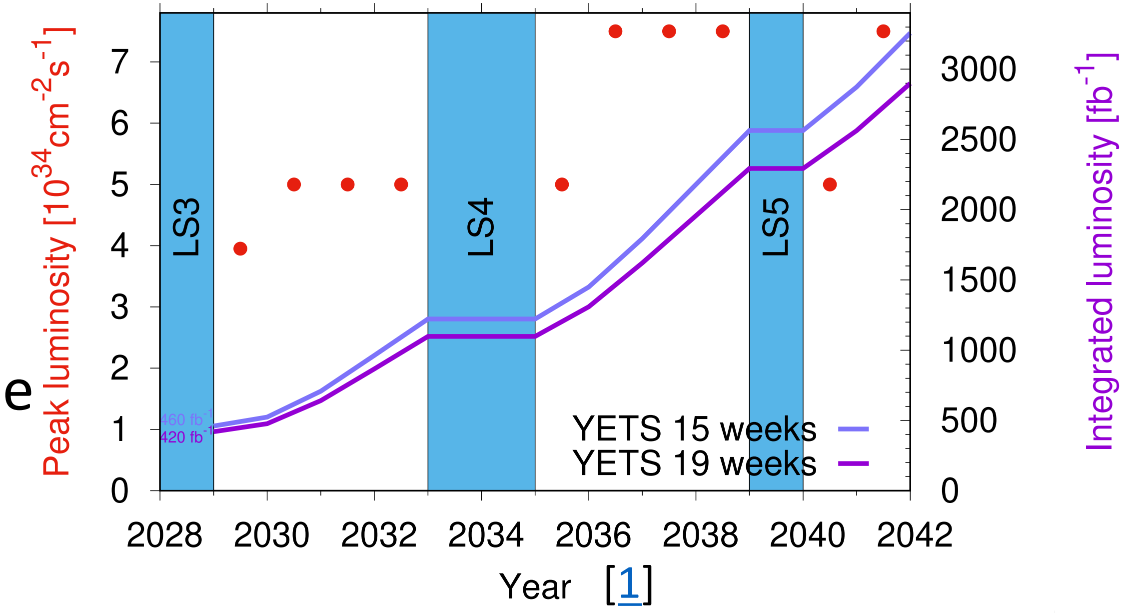
Introduction

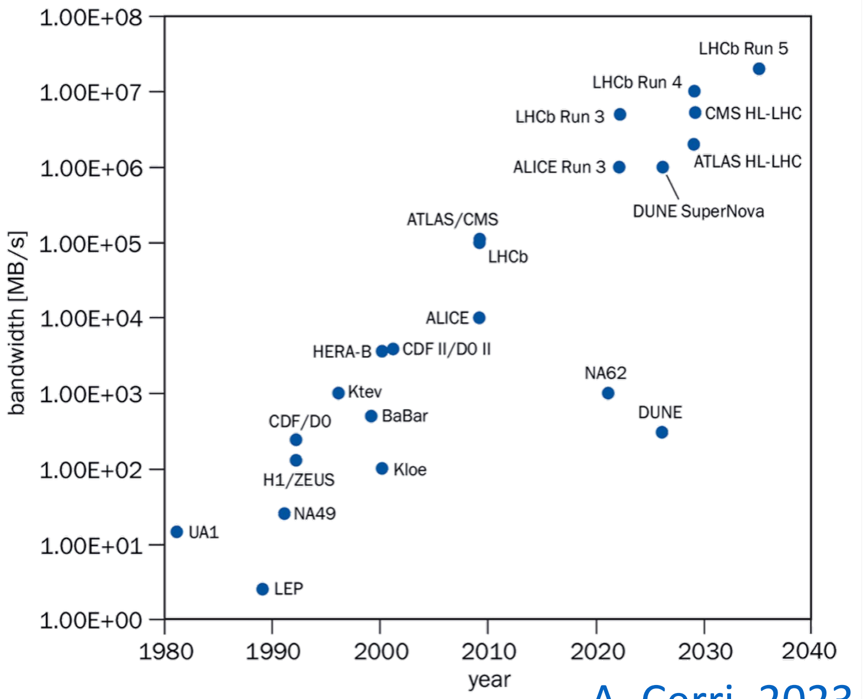
- High-luminosity LHC (HL-LHC) is an upgrade to the LHC allowing for higher luminosity - starting in 2029 (Run 4)

- Peak luminosity increasing ~ 2.5-4× Run 3

- With higher intensity comes greater demand on software
  - Higher bandwidths!
  - We need higher throughput!

- Each experiment has its own framework tailored for its needs
  - Different event sizes and event rates for the software





A. Cerri, 2023

Bandwidth of data processed (not stored!)

Scope of this talk

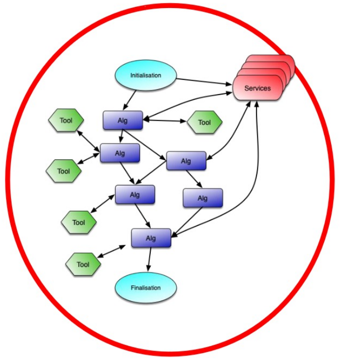
- Discuss the plans of all 4 large LHC experiments – some extra focus on LHCb
  - Heavy reference to the Future frameworks workshop held last November in Marseille [2]

- Looking towards the future – what are the main concerns
  - What kind of framework would best suit the HL-LHC experiments
    - How realistic are they?
  - What direction do we want to take with our framework to ensure high throughput without too much compromise in other areas?

- Simulation frameworks are being considered in general – but not presented here
  - Focusing instead mostly on real-time software and reconstruction

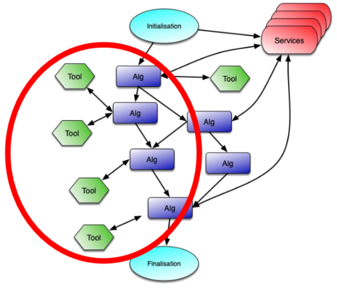
Considerations for the future

- How do we want to achieve the required throughput?
  - Acceleration – GPUs? FPGAs?
    - Cost is the greatest consideration – throughput/CHF
  - Event scheduling – Multiple events at the same time?
- How do we want testing to proceed?
- What do we want the algorithm configuration to look like?
- What about ML considerations?
  - Bookkeeping of models needs a framework of its own
- How should these be prioritised?
  - What to do given the personpower available

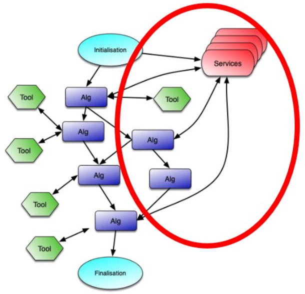


The whole application?!

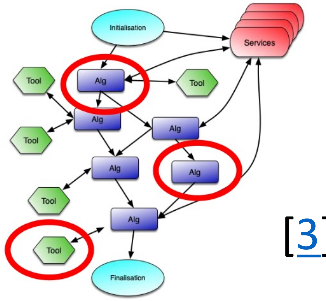
GPU offloading
At what level?



A substantial chunk?
Still pretty good.



Algorithms that need services?
Lots of states to manage across devices!

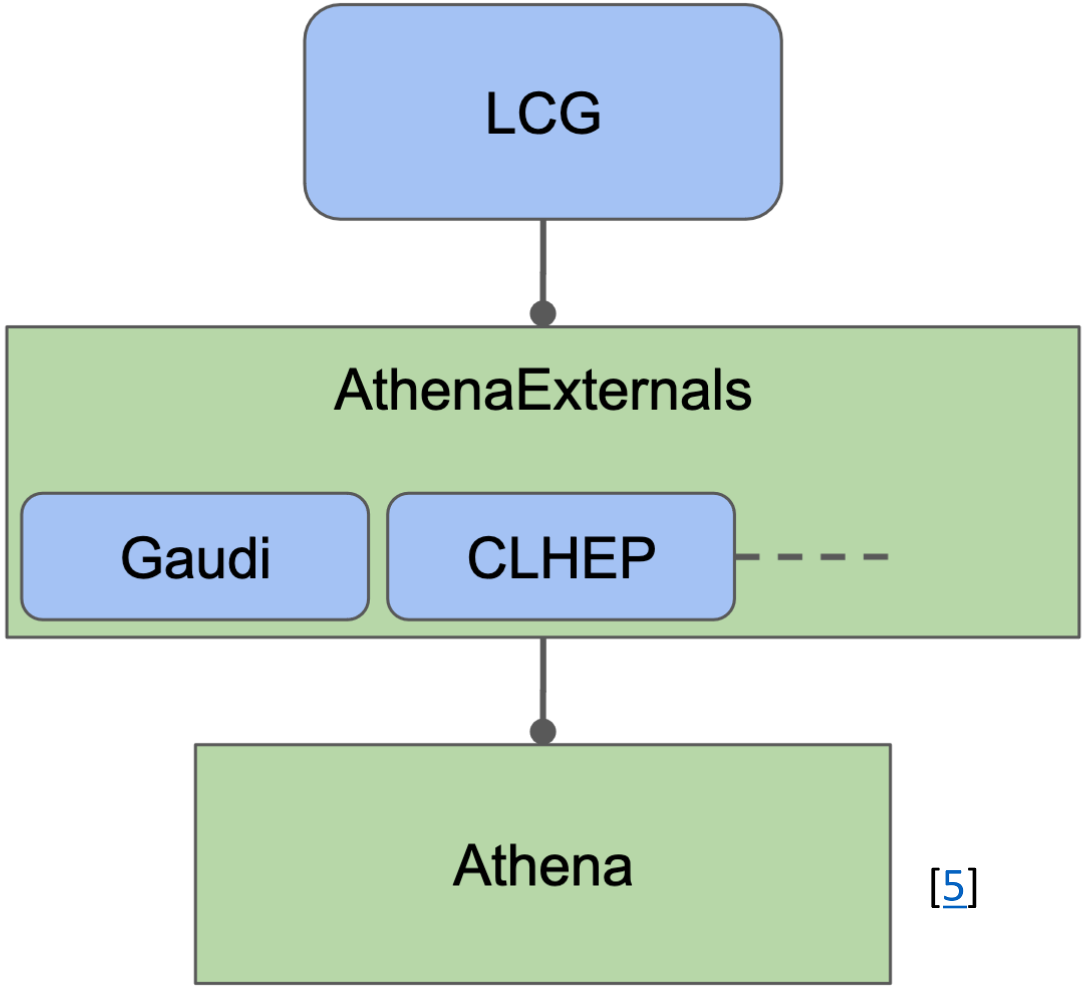


Bits and pieces?
That's a lot of internal data movement…

[3]

How is it currently done – ATLAS

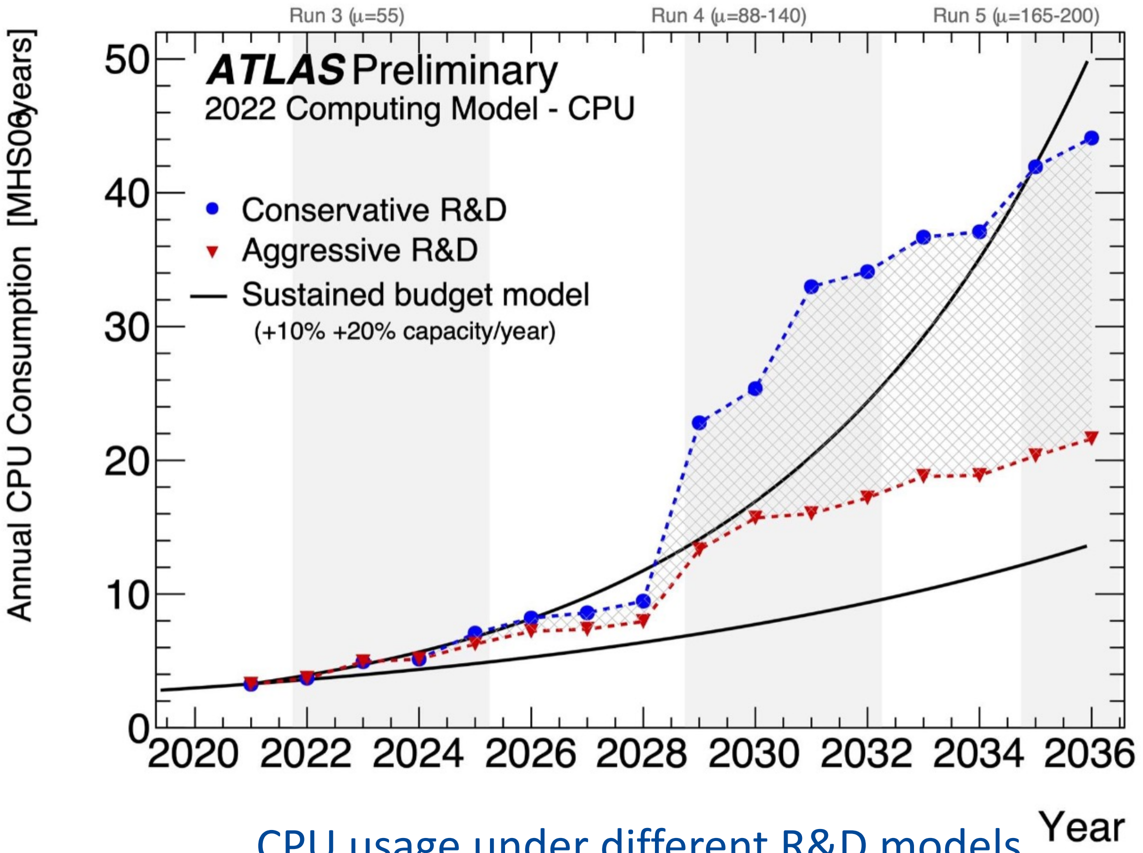
- Framework is Gaudi/Athena [4]
  - Scheduling provided by Avalanche
- Multithreading both within events and across events
  - Each event loaded into the transient event store
- Multiprocessing
  - Allows further parallelism if resources are available



[5]

ATLAS – HL-LHC

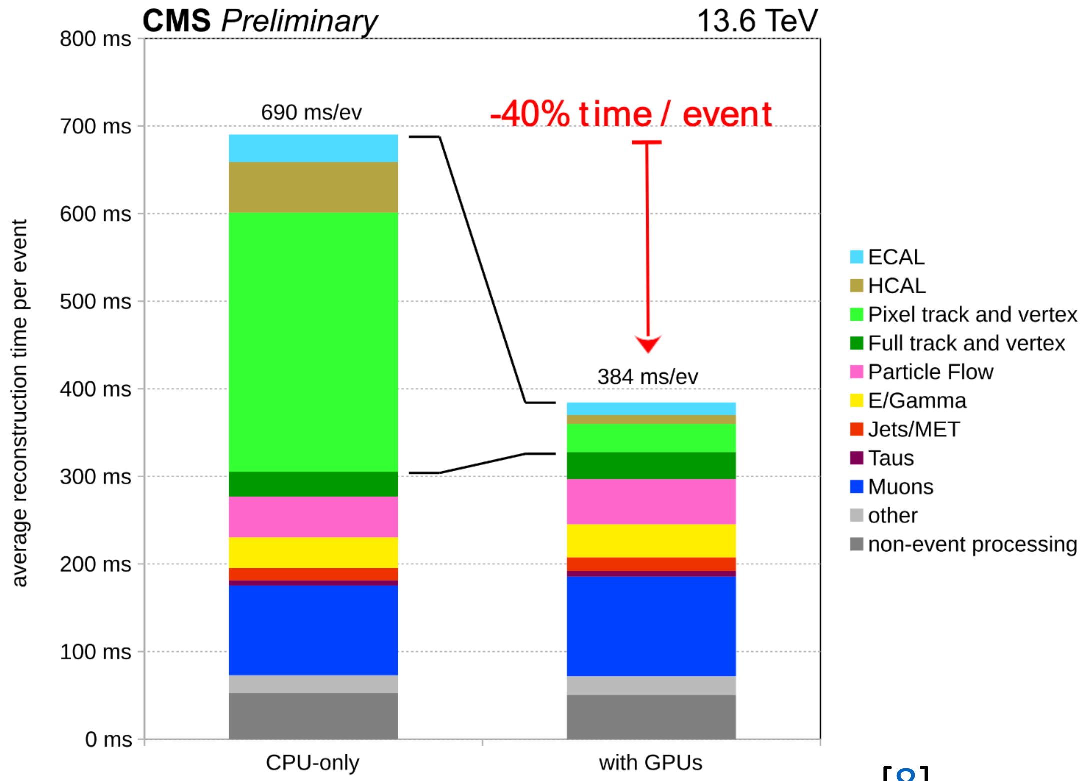
- In the high-lumi era – ATLAS intends to save data at 10kHz with a pileup of 140-200 interations/event!
- Extend Athena with hardware acceleration
    - Compute load will vary depending on the R&D approach [6,7]
    - GPUs are the most likely candidates with FPGAs and TPUs also being explored
    - Scheduling achieved with MPI



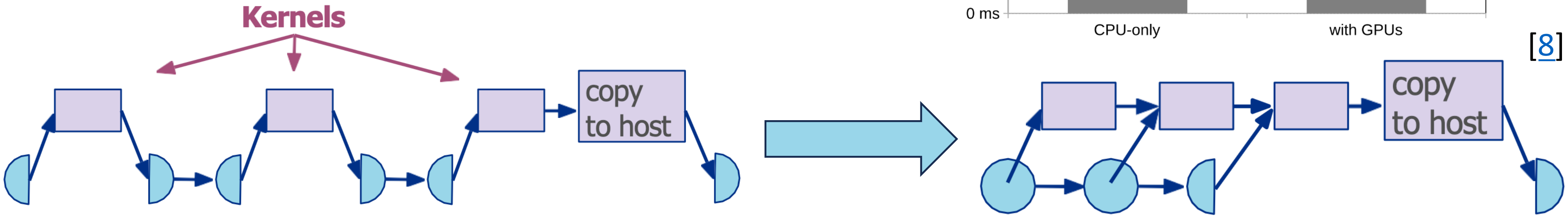
CPU usage under different R&D models assuming 0 GPU acceleration

How is it currently done – CMS

- Trigger GPU accellerated in Run 3
  - CMSSW framework
  - Heterogeneous solution
- Calos and pixel reconstruction performed on GPU
  - Otherwise CPU – including tracking
    - Using Cuda streams and clever sychronization with CMSSW
  - Vastly improved throughput!

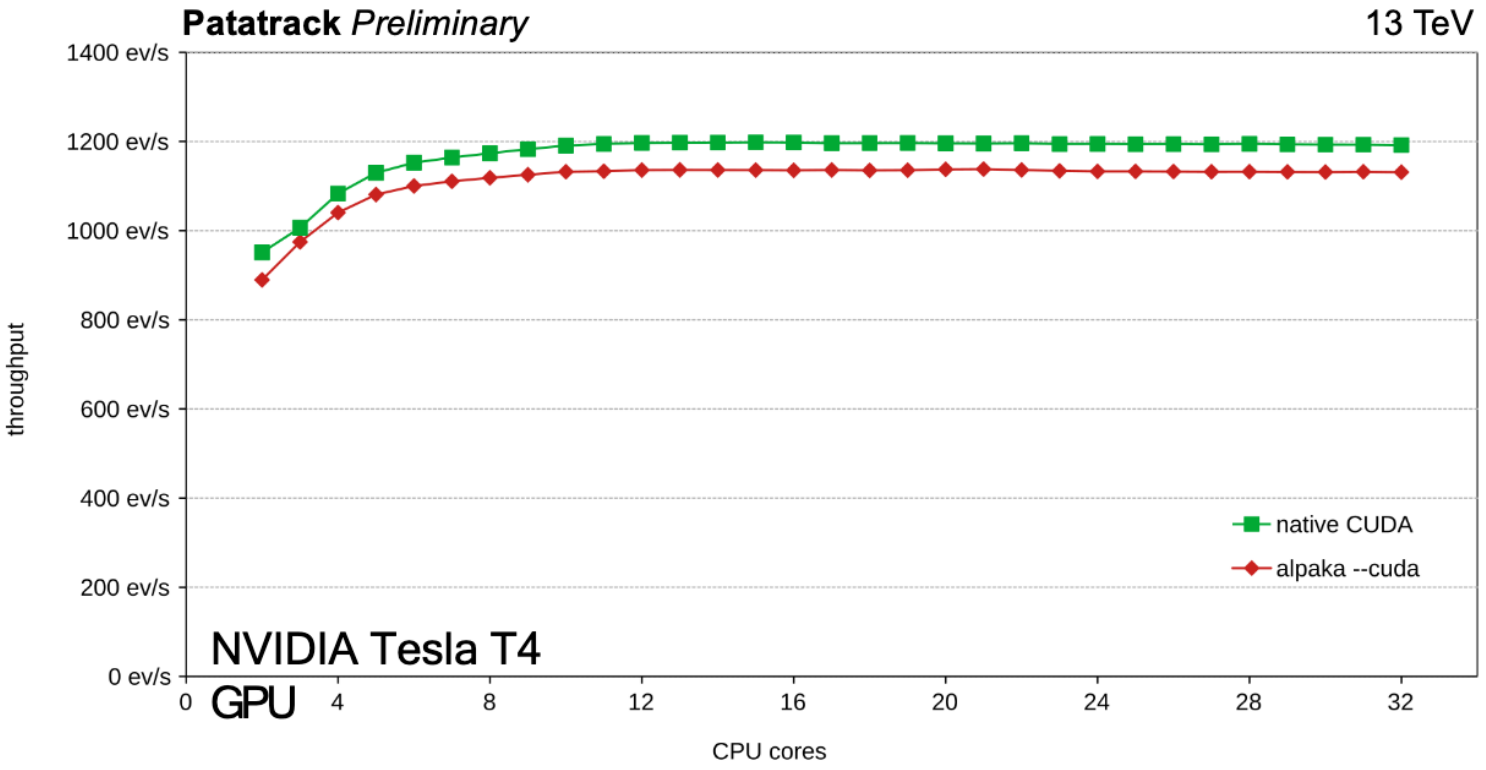


[8]



CMS – HL-LHC

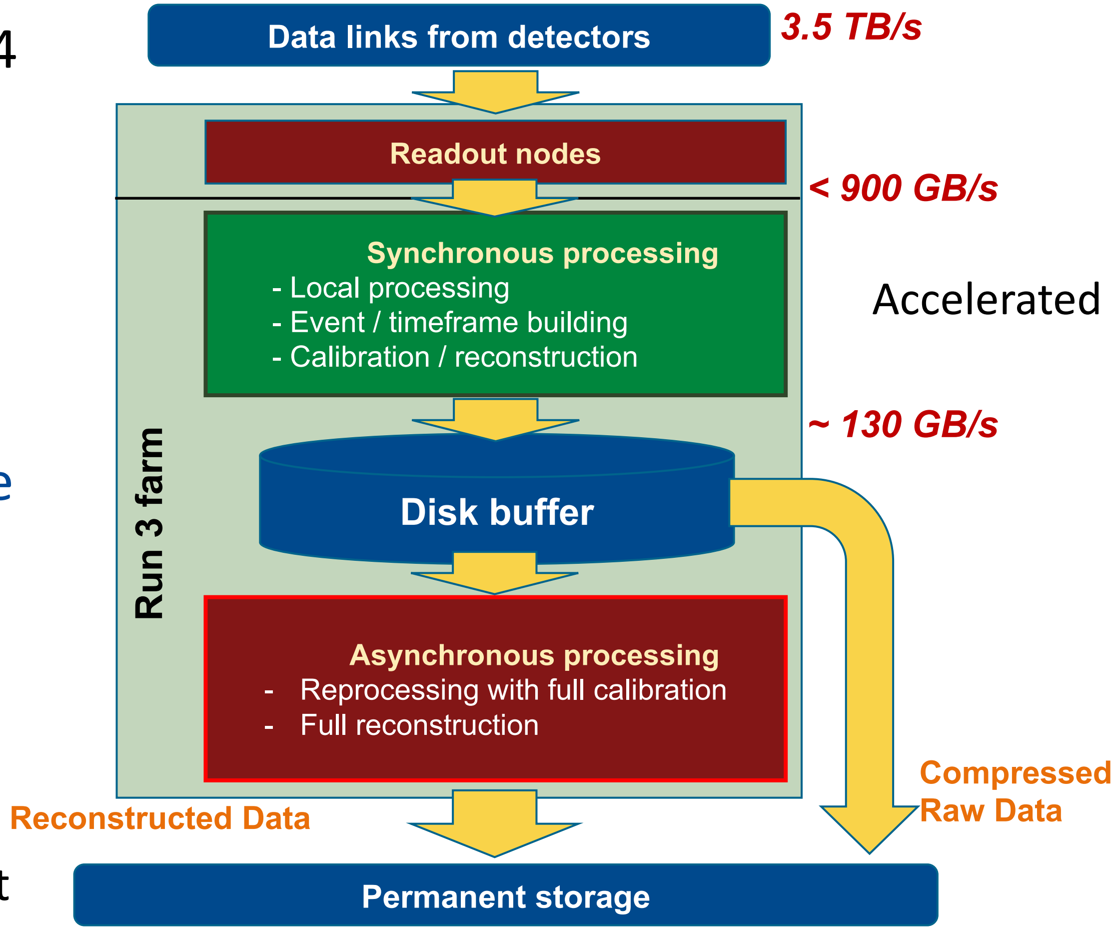
- Investigations ongoing into portability
  - In particular Alpaka is of interest here
  - Abstraction layer across architectures – near native performance!
  - CMS authors actively contributing to Alpaka



- Trying to optimise the framework to work on a wide range of HPC centres [8]
  - GPUs not always available
  - CPU architecture not guarenteed

How is it currently done – ALICE

- O^2 software package used in both Run 3 and Run 4 [9,10]
- Also hardware accellerated – Since Run 1!
  - First processing on FPGAs – then the bulk of the work done of GPUs
- Online and offline have different approaches – due to different needs
  - Offline should keep all servers running at 100%
  - Online needs to keep up with input data rate
- Events are scheduled and processed one frame at a time ($\sim$ 120 collisions)
  - Frame size allows GPU parallelism to become efficient
  - Nodes are assigned frames in a round robin approach

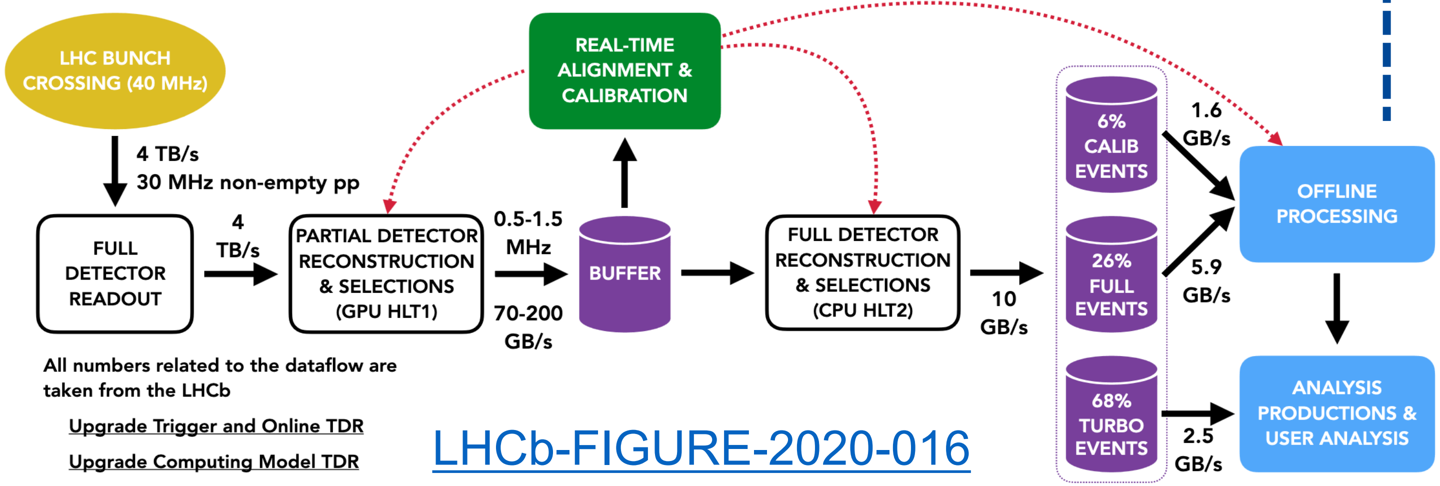


[11]

How is it currently done – LHCb

CURRENTLY 2 FRAMEWORKS

- One based on Gaudi as in ATLAS
- Runs code as a sequence of algorithms
  - Take data from TES – calculate what the user wants – put that data back in the TES



LHCb-FIGURE-2020-016

- Separate framework for first trigger – Allen
  - GPU acceleration designed for high-throughput – Typically ~ 80kevts/s/GPU
  - Cross-architecture
- Algorithms are parallelised
  - Events processed in batches
- Lower memory – a big constraint
  - Different approach compared to the TES in Gaudi

Future considerations and LHCb's plan so far

- Challenge is HLT2 – higher data – quadratic increase in HLT2
  - LHCb looking forwards to its second upgrade – Upgrade II
  - Increasing luminosity × higher HLT1 output rate (needed for signal efficiency)
- Running full reconstruction on GPUs
  - Including full PID, Kalman fit & 4D reconstruction
- Testing and maintenance paramount
- Integrate the Allen and Gaudi frameworks
  - Harmonise the syntax of algorithms between the two – this has already begun
  - Improve memory management – flexibility to choose manager to fit the architecture
  - A common syntax for selections between the trigger levels
  - Work has begun on infrastructure for a common ML framework
- Develop demonstrators for testing and development
  - E.g. Showing the integration of Gaudi and Allen, showing the reconstruction…

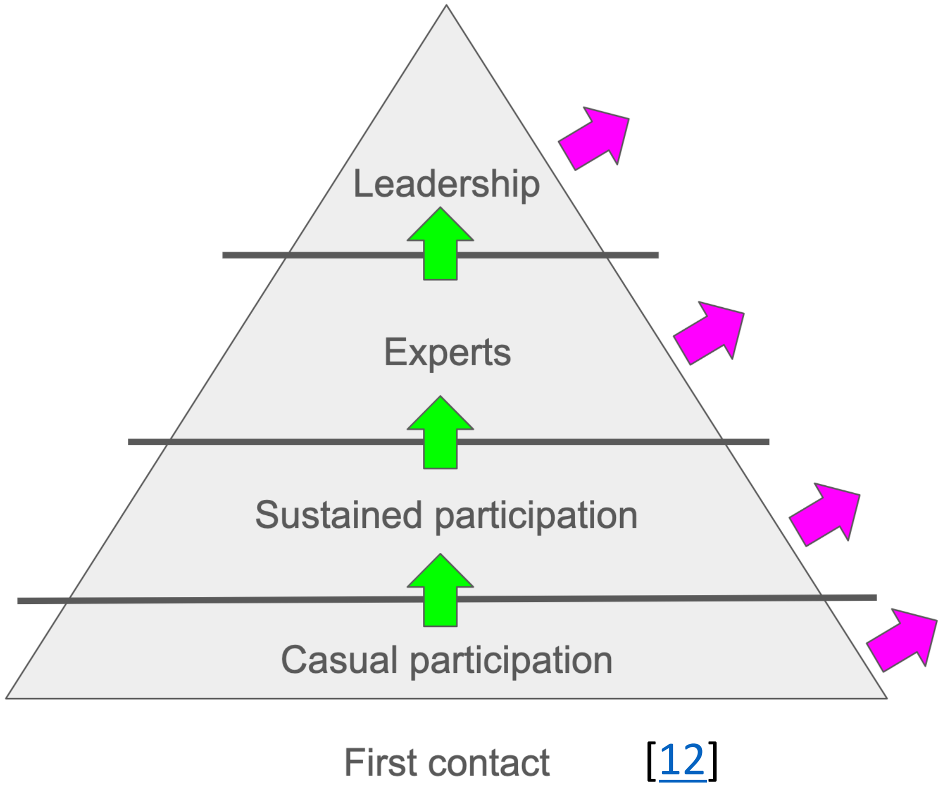
Conclusion

- Planning for the future – high luminosity means a greater computing challenge
  - $\sim \mathcal{O}(10\mathrm{TB/s})$ of data to be processed
  - All large LHC experiments are planning some level of heterogeneity
- Different needs of the experiments lend themselves to separate frameworks tailored for their specific needs
  - A one-size-fits-all solution is unlikely to work efficiently
- The scope of what can be done most heavily relies on personpower
  - Not every good idea will be implemented in time – so prioritisation is a must

Backup

Personpower and documentation

- There is a high turnover rate in academia
  - People come and go quickly – contracts are short
- The decisions/plan must be documented or they risk being forgotten or misunderstood!
- LHCb has started this already with an internal note – this needs to continue
  - The key issue here is knowledge transfer



[12]

Languages to be considered

- Currently –
  - Configuration in python with some yaml
  - Algorithms in C++ and CUDA
    - Precompiler magic and middleware to transpile for CPU and different GPU builds

- Is this a perfect combination?
  - Some interest in changing languages:
    - Julia? – simple to write like python – often quite fast
    - Rust? – similar to C++ with easier memory management
    - A domain specific language we impliment ourselves?
      - Would allow for the same syntax to be used between selections in each trigger

References

- [1] HL-LHC Luminosity reports - https://lhc-commissioning.web.cern.ch/schedule/HL-LHC-plots.htm
- [2] Software Frameworks for LHCb's future conference - https://indico.cern.ch/event/1327907/
- [3] Benedikt Hegner - EP-SFT Plans on Heterogeneous Frameworks
- [4] ATLAS collaboration - Software and computing for Run 3 of the ATLAS experiment at the LHC
- [5] Attila Krasznahorkay - ATLAS's Software Framework Outlook
- [6] ATLAS collaboration - ATLAS Software and Computing HL-LHC Roadmap
- [7] ATLAS collaboration - ATLAS HL-LHC Computing Conceptual Design Report
- [8] Adriano Di Florio - CMS heterogenous experience
- [9] Chiara Zampolli - ALICE data processing for Run 3 and Run 4 at the LHC
- [10] Giulio Eulisse and David Rohr - The O^2 software framework and GPU usage in ALICE online and offline reconstruction in Run 3
- [11] Giulio Eulisse and David Rohr - ALICE Software Stack
- [12] Tim Head - Switzerland, hiking and software: How I try to build sustainable projects