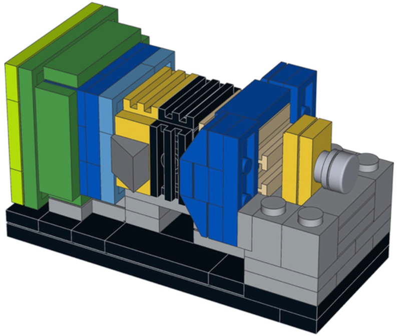
Fast ML inference framework for real-time analysis at LHCb



Maarten van Veghel on behalf of the LHCb RTA project

netherlands eScience center

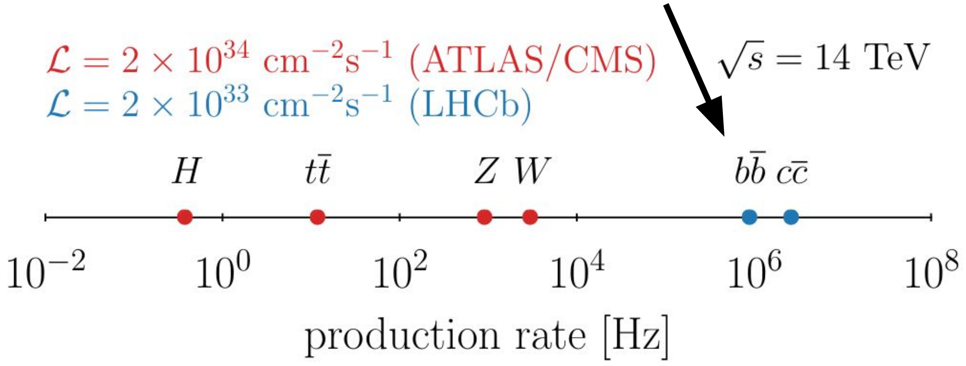
Nikhef

LHCb RTA

LHCb

High throughput demands of LHCb Run 3

- **LHCb** studies mainly decays of ***beauty*** and ***charm*** **hadrons** with **high signal rates**



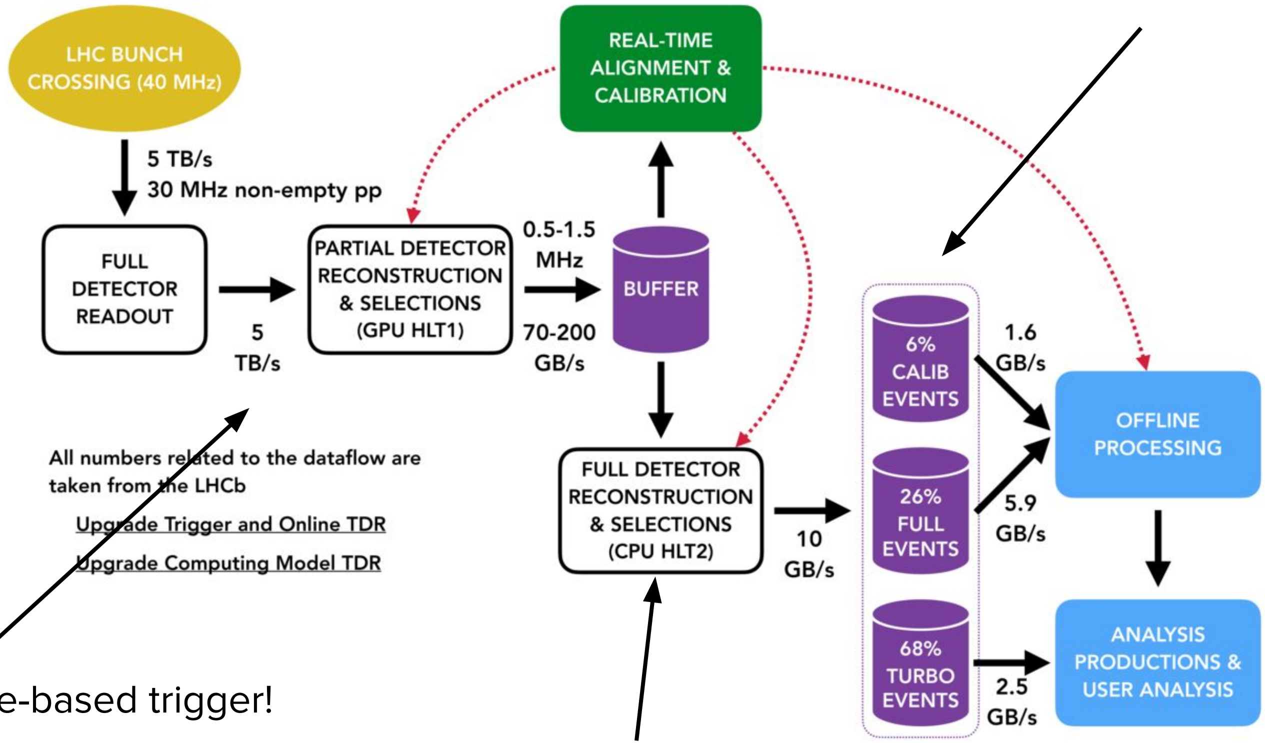
LHCb-PROC-2022-010

- **DAQ running at 40 MHz** to cope with **high signal rate**
  - *Reconstruction and selection* with as **many features** as possible, as **early** as possible
- Extract information from tracking sub-detectors and subsequently ***reconstruct*** and ***select***
  - Make use of **Machine Learning** (inference) **at earliest level as much as possible**

Data flow of the current detector

LHCB-FIGURE-2020-016

Event-model flexibility!



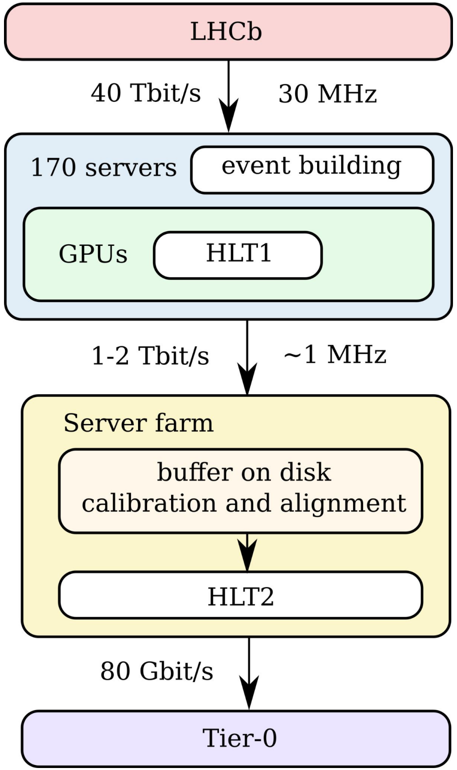
Direct software-based trigger!

Full offline-quality reconstruction!

First level trigger at LHCb HLT1

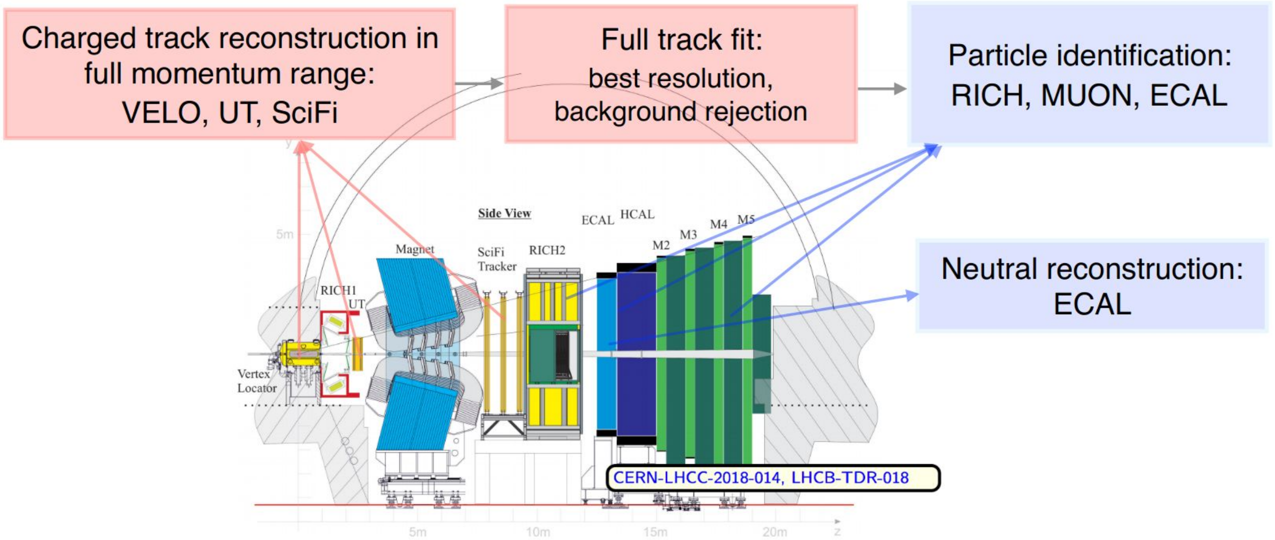
[Comput Softw Big Sci 4, 7 (2020)]

- **About 400 GPUs** reduce the rate of incoming data from 5 TB/s to approximately 100 GB/s
  - About **order 100 kernels** running, with the [Allen] software project
  - Ballpark: with 500 GPUs, **minimum** requirement is **60 kHz per GPU** for 30 MHz non-empty bunch crossings
- **Reconstruction**
  - Charged particles in tracking detectors
    - clustering, tracking, vertexing
      - Track fit and secondary vertex reconstruction
  - Muon stations / calorimeter reconstruction
    - Muon and Electron PID
      - *Including neural nets*
    - Neutrals reconstruction
- **Selection**
  - focused on ***displaced charged tracks***
    - *Including neural nets for two-track combinations*



Second and final level trigger HLT2

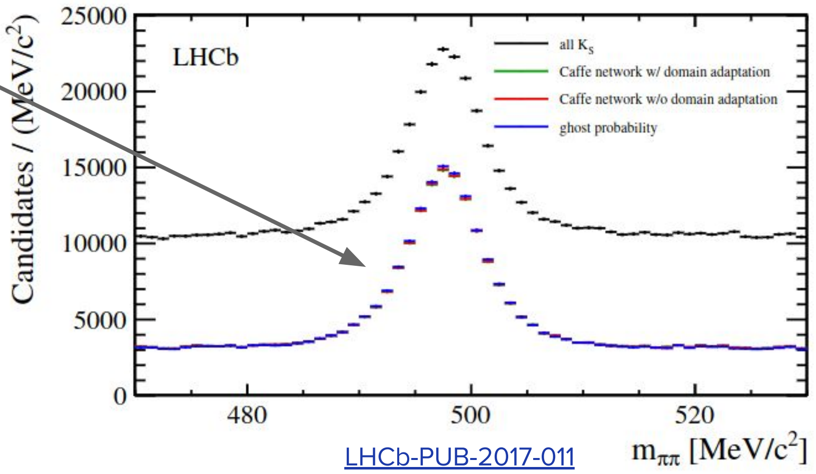
- **Full, offline-quality *(after alignment and calibration)* reconstruction** with full-quality track fit to achieve high momentum resolution, calibrated PID and vertexing **on CPUs**
  - with improvements in muon ID, electron ID and bremsstrahlung reconstruction
- **Order of 1000 selections**
  - including dedicated reconstructions, selective information persistency, …
- Ballpark: about 200 Hz throughput needed assuming about 5000 servers with 1 MHz input



Applications of ML ***in online environment of LHCb***

- **Classification of reconstructed objects** (at all levels)
  - **Reconstruction**
    - Charged tracks
      - Real vs fake (ghost rejection)
    - Type of charged tracks
      - pion / muon / electron / ...
  - **Selection** level
    - Higher level objects
      - combination of tracks coming from heavy flavour decays
    - Typically trained / used for **selecting specific signals** with trigger lines
  - **Typical feature counts of 10-20**
- Other tasks like ***pattern recognition*** and ***anomaly detection*** are possible and studied

Ghost rejection MLP from previous LHCb Run 2



LHCb-PUB-2017-011

ML infrastructure ***in online environment of LHCb***

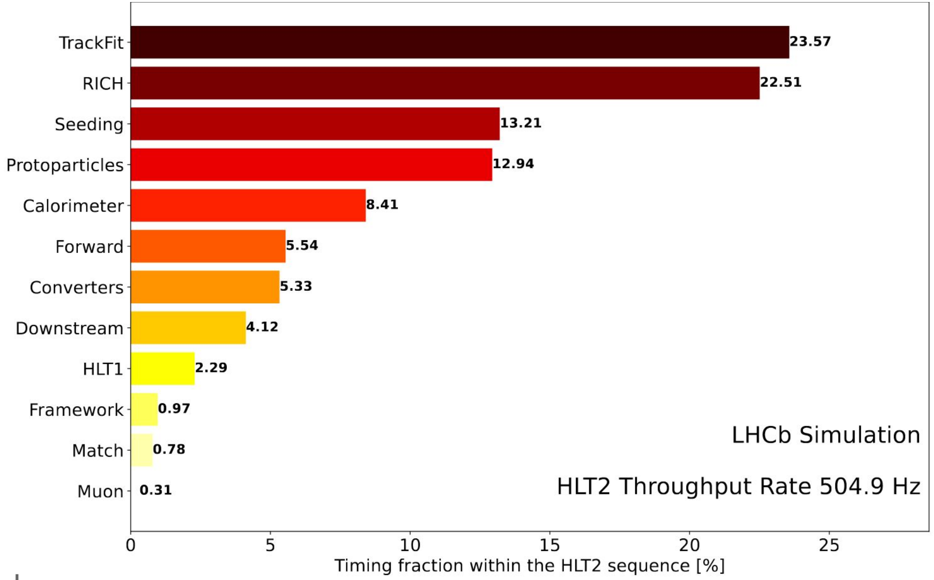
- **Online environment needs**
  - **Most of all high speed!**
  - **Fast turn around time** of training and deployment, ...
  - **Common tools / standardization**
    - avoid customization / hard coded solutions as much as possible
    - improve maintainability and ease of use
  - **Production level code needs a lot of testing**
    - Run ML pipelines in CI/CD (Gitlab/Jenkins)
      - Also for fast turnaround time!
- **Most needed in HLT2 (CPU)**
  - Most applications, most interactions with ‘users’
  - **First focus on fastest algorithms, also have simplest models!**
- But in the **future more emphasize on general libraries and GPUs**, developments ongoing
  - More challenging setup with demands on GPU/CPU compatible libraries and speed

GitLab

Jenkins

HLT2 (CPU) throughput

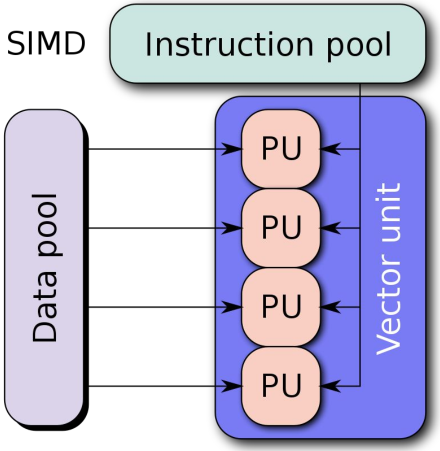
- **Significant speed improvements** have been achieved using
  - **Multithreading / vectorization** also in this CPU-based software
  - *Structure-of-Arrays*
    - Reduce memory usage
  - Parallelization with SIMD
    - Single Instruction/Multiple Data [JINST 15 (2020) 06, P06018]
  - Smarter, more selective algorithms
    - Pre-select on input of time-intensive algorithms
  - Mainly used in reconstruction sequences
    - See reconstruction throughput breakdown on the right before speed ups



- **Developed new ML inference infrastructure to fully make use of that**

Fast inference in HLT2 (CPU)

- Fast inference of **relatively simple models** (MLPs)
  - **Shapes of models fully set at compile time**
  - Custom implementation within Gaudi framework
    - Allows full control (of speed ups)
    - Typical MLP layers supported
    - Integration with (SIMD) event model
  - **Evaluation using SIMD**
    - Automatic batching when running over ranges like *std::vector* with non-SIMD event model
  - **Weights loaded during configuration from database**
    - Allows flexibility with retraining and deployment
- **Training infrastructure**
  - API with **PyTorch**
    - Regression test to ensure similarity
    - Easily extendable to other training software
  - Example of **training runs in CI / Jenkins**



Testing, pipelines and experience in production

- **General aims achieved** for HLT2 (CPU) infrastructure
    - **Speeding up main classifiers** (roughly 10% of reco timing) ✅
        - **factor 2 - 3**
    - **Separate / fully optimized inference from training** ✅
    - **Maintained training pipeline** ✅

- **Running in production since start 2024** for HLT2 (CPU) infrastructure
    - **Already used for fast retraining due to online needs**
        - *Retraining within a day,*
          cross-checked / released / deployed within a few days
    - Multiple developers picked it up and are expanding it
        - Feedback so far is that it’s easy to use and expand

- **General aims achieved**
    - **Fast turnaround time** ✅
    - **Ease of use** ✅

GitLab

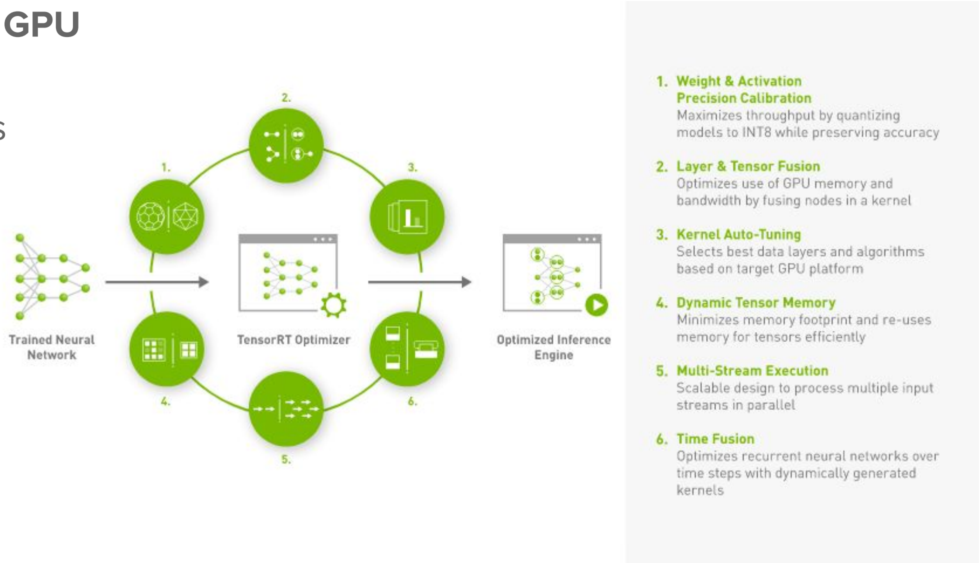
Jenkins

General libraries for ML inference in HLT1 (GPU)

- ***Flexibility, maintainability***
  - Hard/hand-coded ML inference is not flexible / not great to maintain
  - Platform to load **standardized ML-model data format**: **ONNX**
    - Supported by many (if not most) training software
    - **At CPU (HLT2) level being integrated with *ONNXRuntime***

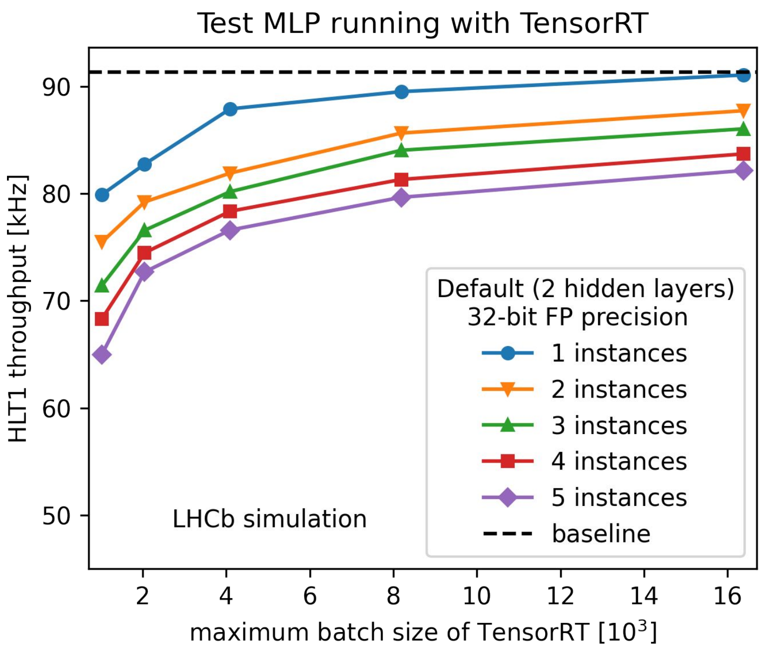
ONNX

- Providing these features with inference on **GPU**
  - LHCb uses NVIDIA RTX A5000
  - **TensorRT** [link] from NVIDIA provides
    - Fast-inference platform / SDK
    - ONNX files can be read by it
    - Optimization possible within package, like quantization

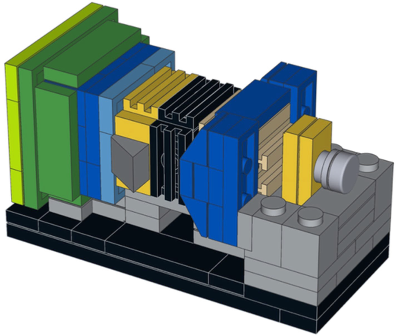


Throughput impact of TensorRT inference

- The **baseline model** tested
  with respect to TensorRT **batch size**
  - **Kernel overhead is main bottleneck**
    - These MLPs are small
- At high batch size it seems getting
  feasible to run a few copies of such neural nets!



Conclusions and outlook



- **LHCb** has **high demands of throughput** of reconstruction and selection on both CPUs and GPUs to cope with high signal rates
    - **Machine learning ideal to reduce rates while keeping signal efficiencies high**

- **ML infrastructure for online use**
    - **General aims**
        - **Fast inference**
        - **Separate inference from training**, using common tools for training like PyTorch
        - **Maintained** training pipeline
    - **Generally achieved for simple, but most demanding models for CPU (HLT2) part** ✅
    - **Ongoing developments for GPU as well**

- **More is needed though**
    - Better infrastructure with **general inference libraries** (*ONNXRuntime*, ...) in selections
        - Selections are typically less demanding in terms of speed ups
    - Testing infrastructure is not really scalable, needs a dedicate solution