Manycore feasibility studies at the LHCb trigger

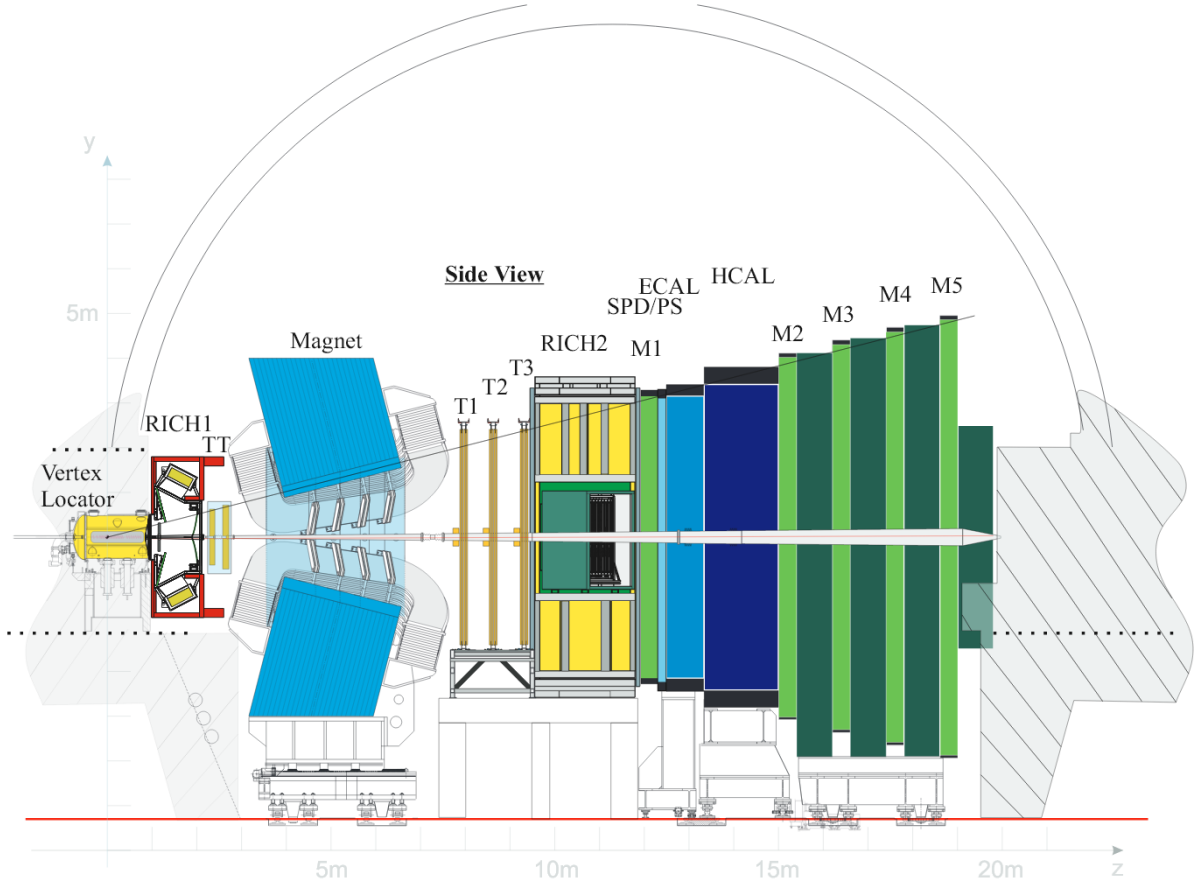
AN ONGOING TALE

LHCb ONLINE

Be our guest

- What?
- How?

The LHCb detector



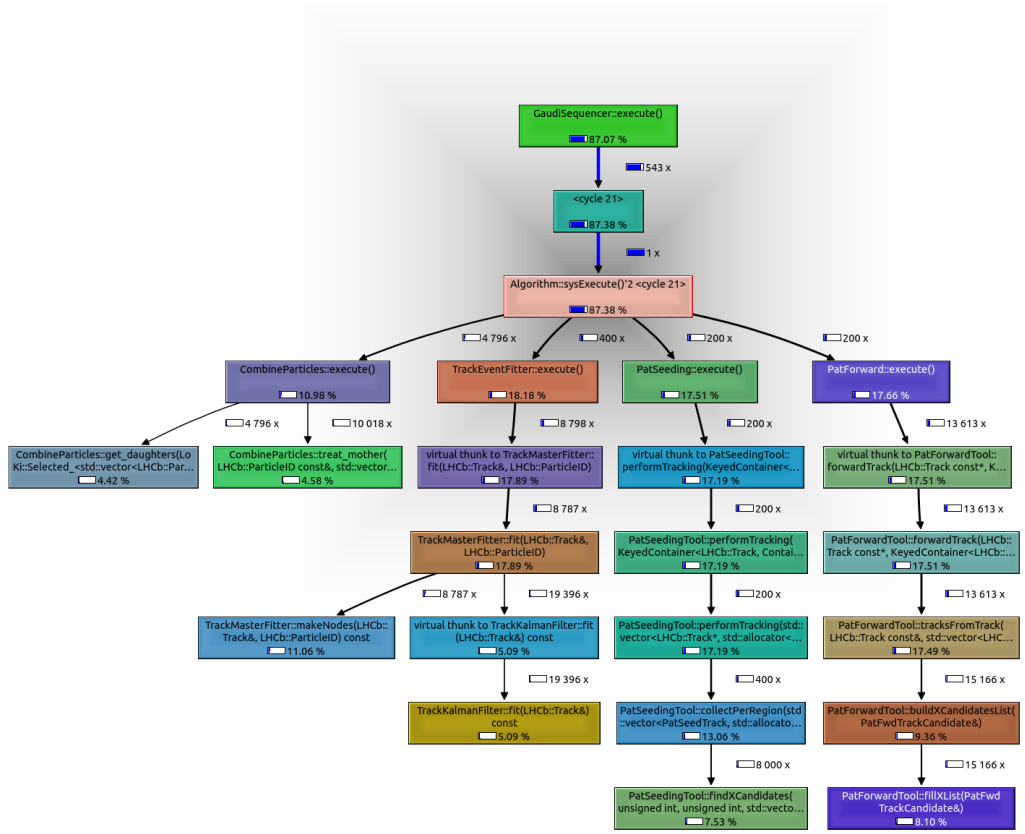
The meat in the HLT

Current HLT timing figures

- CombineParticles – 10.98 %
- TrackEventFitter – 18.18 %
- PatSeeding – 17.51 %
- PatForward – 17.66 %

LS2, 2020 Upgrade

- ~ 13 ms available per-event (L1)



Courtesy of Ben Couturier

Vertex Locator

Current HLT

- FastVELO – Stefano Gallorini
  - Track pattern-recognition on GPGPUs in the LHCb experiment

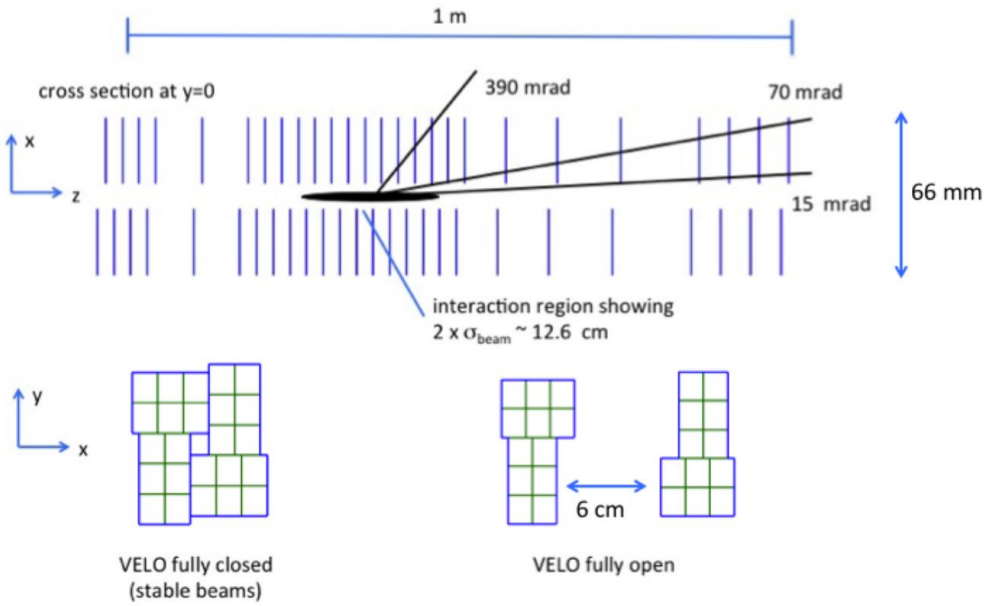
Upgrade HLT

- Vertexing – Gerco Onderwater
  - Exploring Retina-like tracking approach
- VELO Pixel
  - Vectors
  - Tracking strategies

VELO Pixel – An old new friend

- 256x256 chips, 12 chips per module
- 48 modules, 24 on each side
- O(1000) hits per collision
- Non-uniform distribution of hits
- No magnetic field – Tracks are lines!

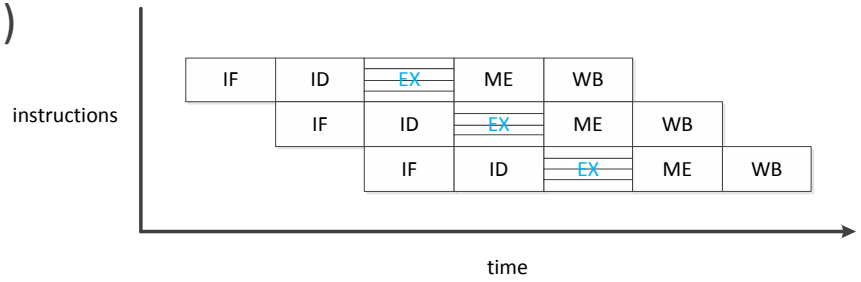
- Tracklets are required to have three hits
- Reconstruction checks for acceptance window, then performs square root fit to select tracks
- Forward track Reconstruction Efficiency should be maximized



The starting point

Sections of the code can be ported to parallel architectures

- Vectorisation improved the current baseline, **1.14x** (12% speedup)
- Bit-level exact same result
  - SSE / SSE2 is supported in all 64-bit architectures
  - Runtime optimization based on architecture is possible
  - Preparing the ground for the future!



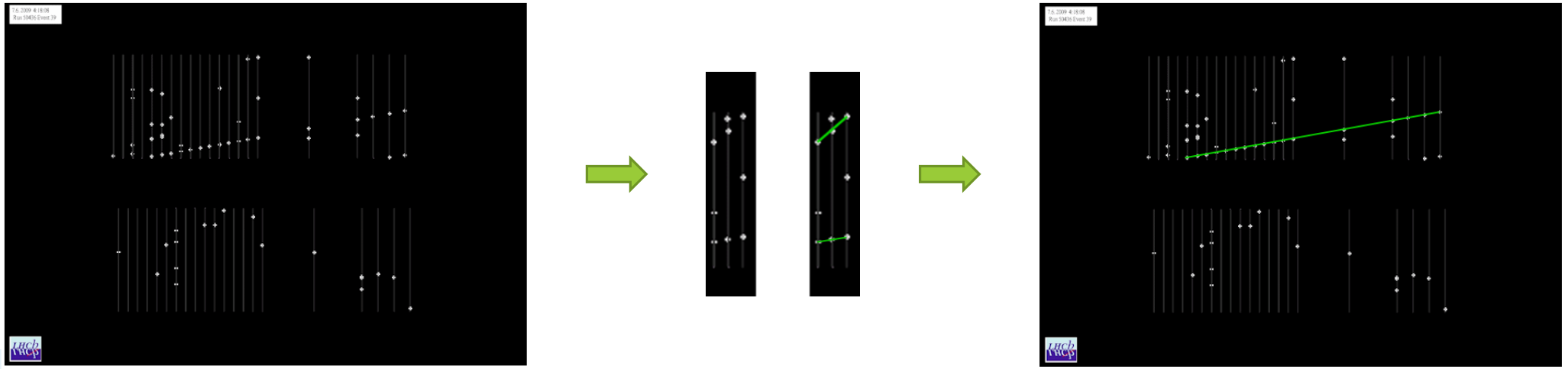
Literature tells us manycore has something in store for us

- ALICE reconstruction, local method with automata-based seeding
- NA62 RICH reconstruction on GPU

#1 – Track following!

Local method, based on current VELO algorithm

- **Seeding** – Parallel search of best-fitting triplet per hit
- **Track forwarding** – Tracklets are forwarded backwards
- [**Selection**] – Ghost and clones are removed based on *best* track



In the makings

- **11x** speedup gained on the GPU
- Physics cut needs to be polished (should not affect performance)

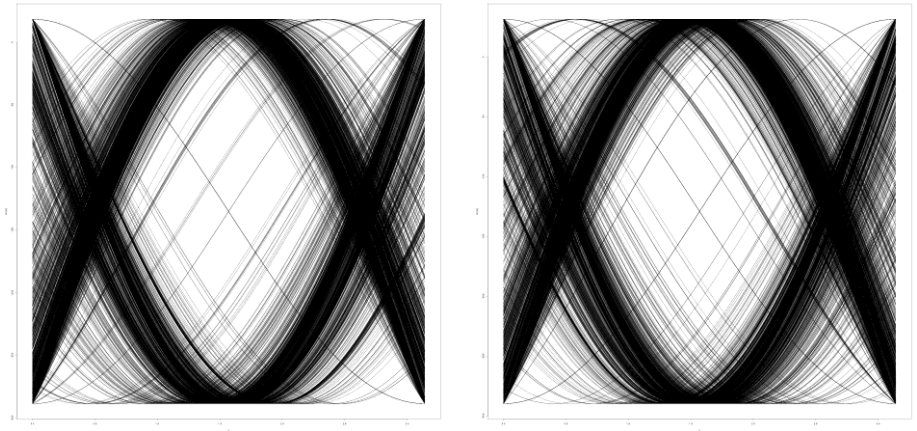
	PrPixel	gpuKalman	selection
Reconstruction efficiency	75.1 % (99 %)	54.9 % (60 %)	37.7 % (40 %)
Ghost Fraction	5.48 %	63.35 %	26.3 %
Clone Fraction	23.05 %	32.4 %	36.4 %
Purity	99.68 %	99.9 %	99.9 %

Intel Xeon CPU E5-2650 @ 2.00GHz (PrPixel v45r0) versus
GeForce GTX 680 (1536 CUDA cores @ 1GHz)

#2 – Hough transform

Global method based on representing all possible lines crossing a point, in another space

- Inherently global method – Parallelisable
- Doesn't require additional preprocessing: LHCb VELO tracks are straight lines!

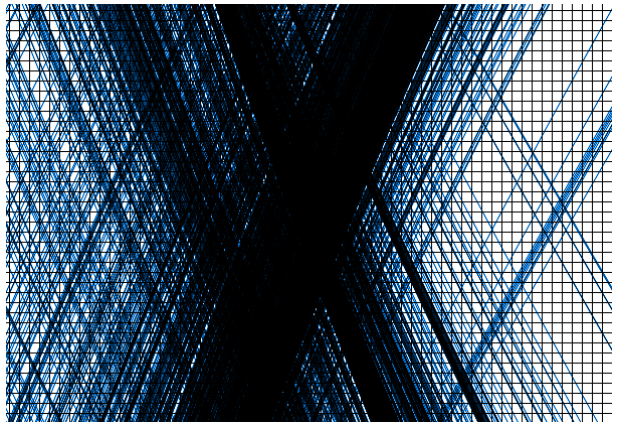


LHCb velopix event in XZ and YZ plane, in polar coordinates

Bringing the process to manycore

Our current GPU design

- 2D histograms (XZ, YZ) + set intersections, or 3D histogram
- Thread assignment to column processing (no Read-After-Write)
- Tiling
- Tweakable granularity (bin size)
- Minimise global memory accesses, shared memory for tracklets formation

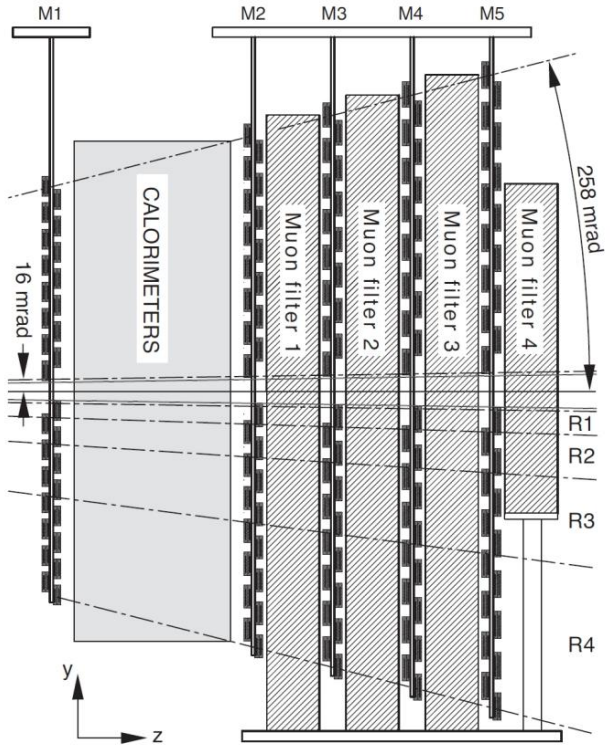


Bin-size determines granularity,
linked to error acceptance

More *Moore*

Muon ID – Some shortcomings:

- Code is extremely *branchy*
- Search window occurs within Field of Interest
  - Determined with input track
  - Long / Downstream track search in M2 + M3 [+ M4 [+ M5]]
    - Min 2 hits
    - Algorithm flowchart
- **Dependencies**, not blind search in all subdetector space
- Little room for parallelization
  - Fine-tuned for FPGA at the moment



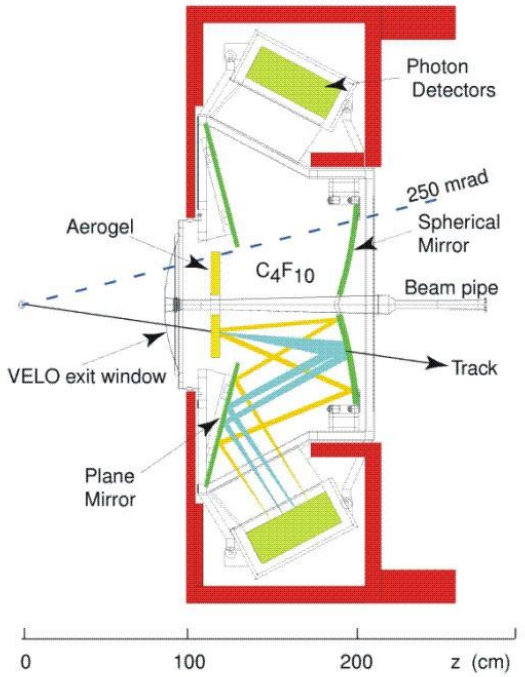
Thanks to X. Cid Vidal for fruitful discussions

Getting RICH

Looks extremely interesting, LHCb HLT with RICH anybody?

- O(100) times slower than Online tracking reconstruction
- Existing methods (ie. ENA) have been tried in the past on CPU, little success
- Analytical solution requires several interfaceable algorithms, only some are potentially suitable for exploiting the power of manycore

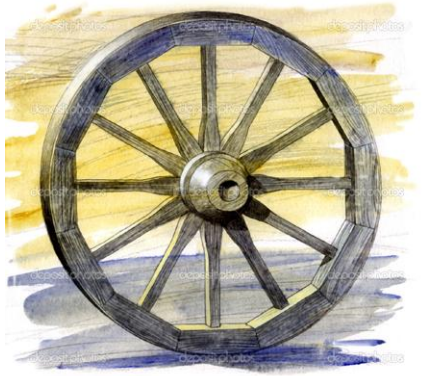
- Looking good
  - Ray Tracing – Prototyping on GPU
  - Minimum likelihood parallelisation



Offloading to libraries

Whenever possible, we should use libraries

- Standardized
- Maintained
- Scalable – ie. Vectorised



Many fits done in LHCb code, slight different requirements

- Putting them all in one place!

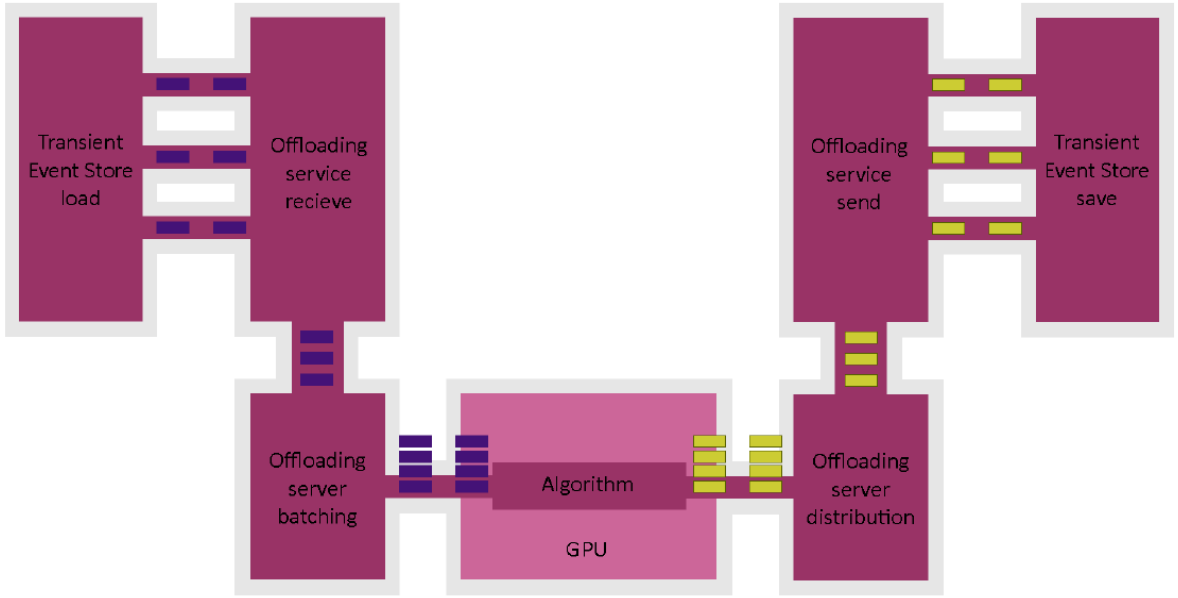
Be our guest

- What?
- How?

GPU Manager

Gaudi tool to offload algorithms

- Socket client-server transmission
- Scheduler First-Come First-Served, gathers multiple events and ships them for concurrent processing
- Some goodies
  - Algorithm exceptions propagated to callers
  - Centralized profiling, logging
  - Centralized performance measurement
  - File input / output configurable
  - Outside framework execution possible



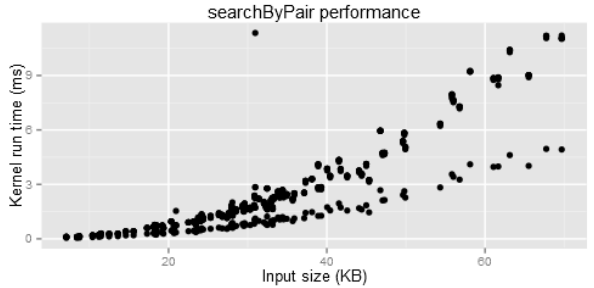
$$TS(AoS) \xrightarrow{prepare} SoA \xrightarrow{kernel\_execute} SoA(result) \xrightarrow{store} TS(AoS)$$

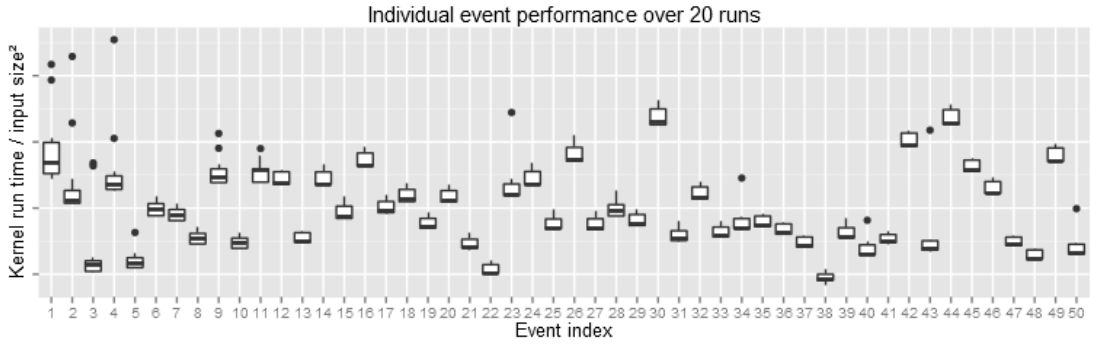
In collaboration with A. Badalov
University La Salle – Ramón Llull

What's *new*, folks?

- Version prepared for svn, inclusion as an LHCb project
- Inclusion with other ongoing manycore projects, getting more people in the boat

- Current framework doesn't support several events in parallel
  - Size of LHCb events is O(100 KiB)
  - Multithreaded version of the framework on its way – GaudiMT

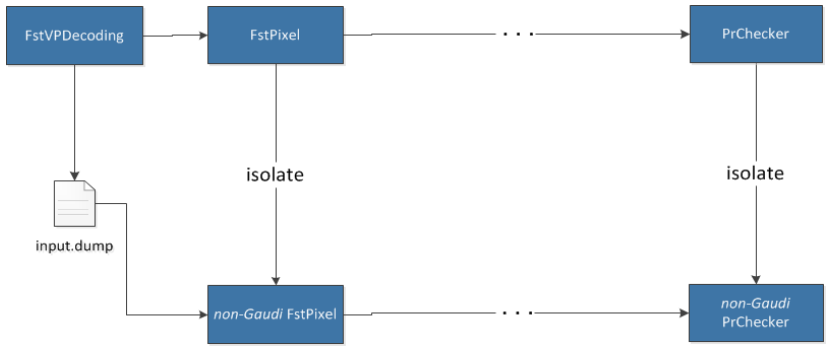




Independence day

- As a GPU developer, I want to stay *away* from our big framework for as long as possible!
  - Not suited for GPU development

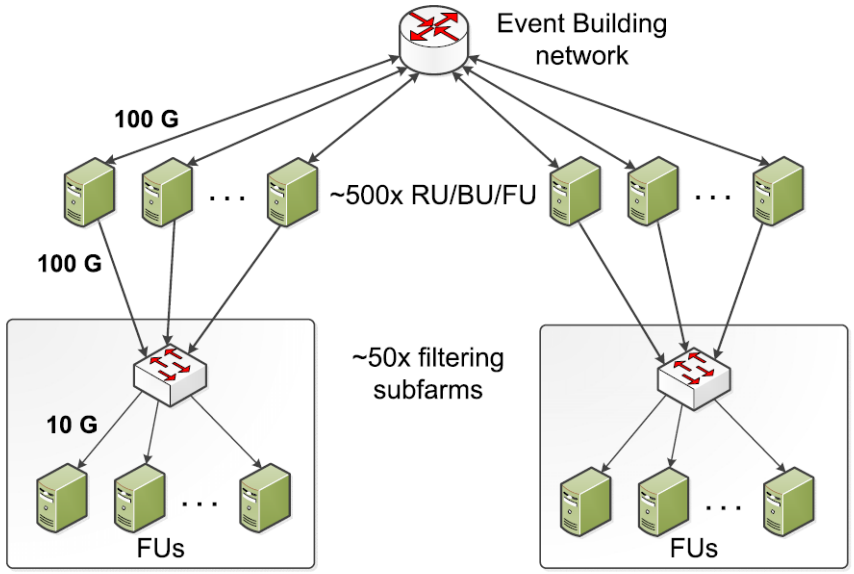
- Standalone method for compiling
  - Compatible with GPU Manager runtime generated input (playback feature)
  - Windows compatible
    - Nsight on Windows has arguably best debugging and profiling capabilities



All is cool and dandy, me too!

We need an infrastructure to do all our testing

- In LHCb, we are moving towards a hardware triggerless DAQ
- Online HLT node replicating data for GPU testing in realtime



CPU / GPU powered filtering farm?

GPUs are discussed often, but…

Coding for them is a big effort

- Analyse - Design - Unit test Physics - compare performance
- Not even considering training here…

We need to provide infrastructure if GPUs are to be even given a chance

- Testing not so straightforward as with other architectures

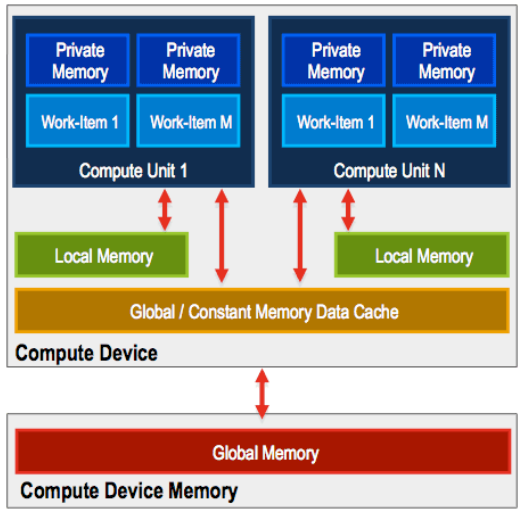
Concluding

Many cores, many opportunities in LHCb

- We have demonstrated the potential of manycore for the LHCb Trigger
- Giving vectors some love!
- Future looks manycore

Obstacles to overcome

- Long dev cycles, can't offload everything to a Summer Student
- Sequential framework
- Demonstrate Physics efficiency of new algorithms
- No *make parallel* button
  - Programming model, memory model, framework integration, algorithm design



Thanks!

Qs? As!

References

Follow us on

- *Fridays at Vidyo* ® - GPU@LHCbTrigger
- https://lbonupgrade.cern.ch/manycore
- GPGPU opportunities at the LHCb trigger

Backup

#3 – Retina algorithm

Photoreceptor inspired algorithm

- Excitation depends on recognition of preexisting pattern
- DB growth can be modeled after VELO contraints
  - Three-cluster tracklets
  - Defining a cone of acceptance
- Parallel search possible, still need for a subsequent merge step

- This method has been successfully implemented in FPGAs