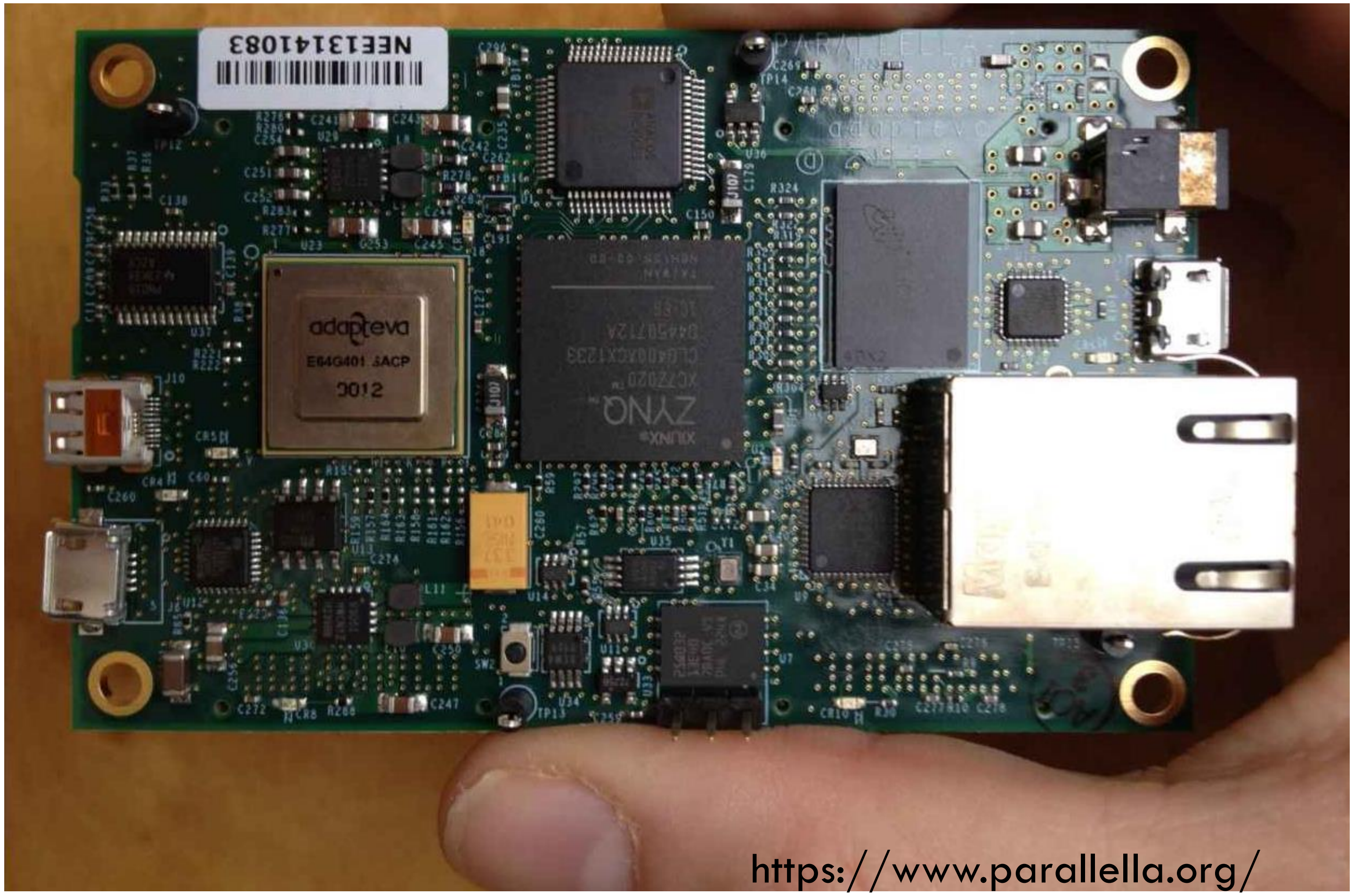


The role of Heterogeneous computing and accelerators in HEP data processing ”CERN openlab workshop 23/4/17”

Niko Neufeld
niko.neufeld@cern.ch

Acknowledgements & disclaimer

I have ”stolen” slides and material from various sources in our community. I try to acknowledge correctly but I will for sure have forgotten at few.

I collectively thank all my colleagues and friends in the HEP computing community and beyond for what I have learned in discussions and presentations

It’s really “challenging” to cover all of HEP. And then in 15’ … so my excuses to NA62, PANDA, and many others… I take the examples from the LHC experiments

Heterogeneous computing

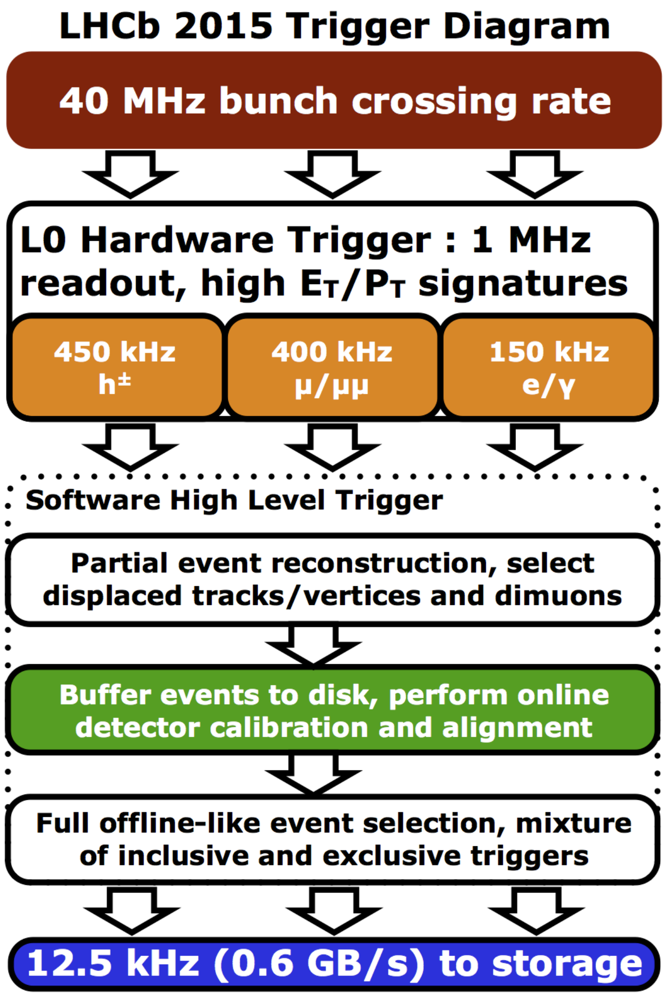
Heterogeneous computing means using something else than 64-bit Linux servers

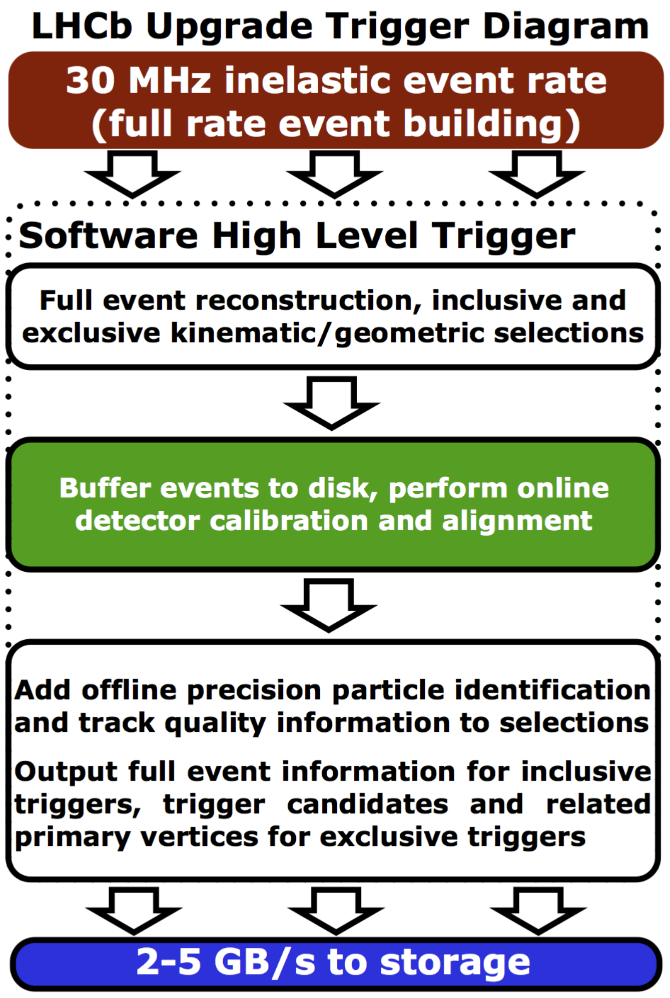
- ❑I do not consider using AMD, ARM64 or OpenPower as "heterogeneous" computing, even though there are of course **significant** "eco-system" issues with some of those (excepting x86 AMD)

Examples of heterogeneous computing are:

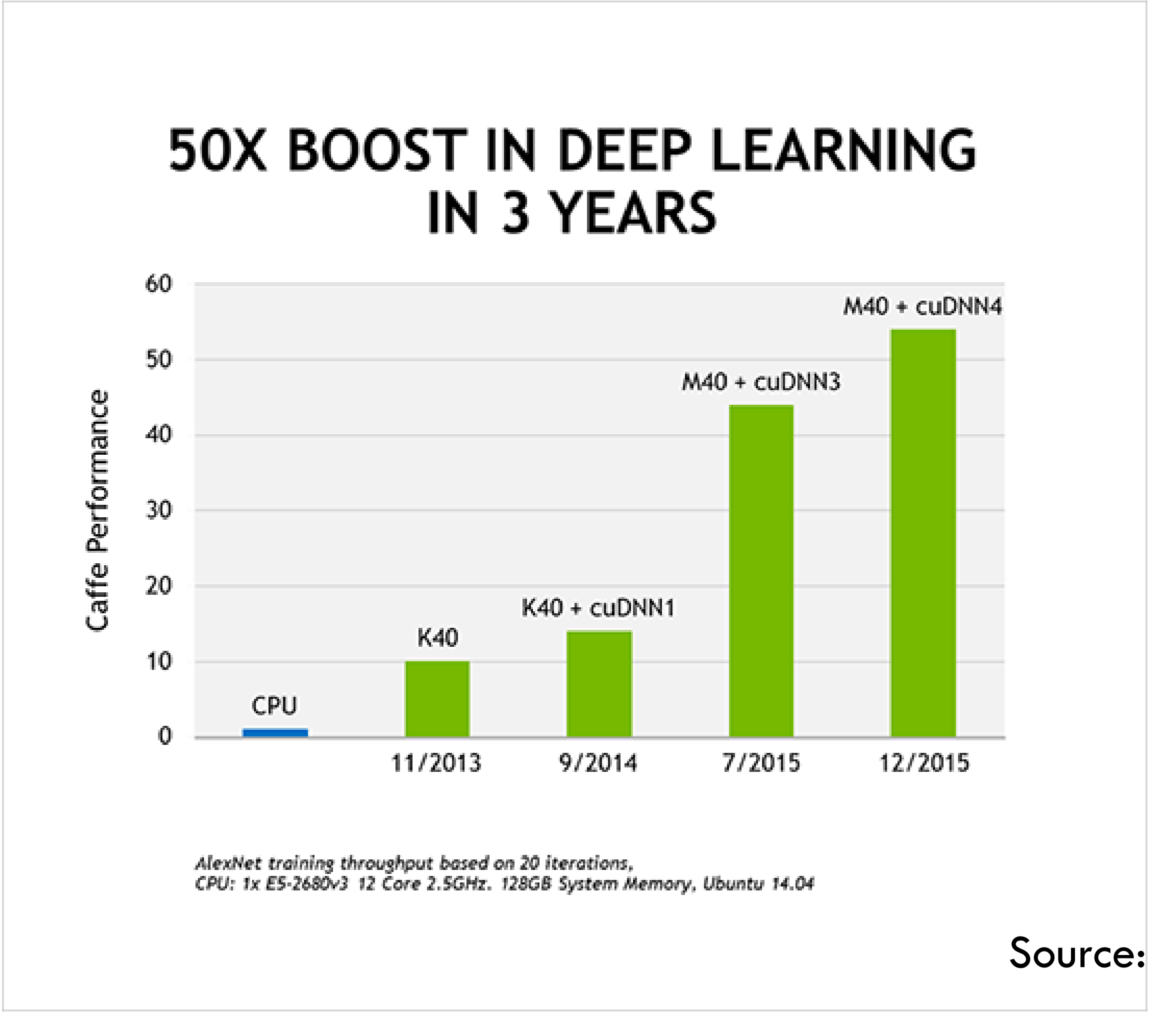
- ❑Accelerators working with a host CPU: GPGPUs, FPGAs
- ❑Self-booting accelerators connected to a fabric: Intel's Knights family
- ❑External custom-made systems using combinations o FPGAs and ASICs: example ATLAS Fast TracKer (FTK)

LHCb Trigger in from Run2 to Run3


LHCb 2015 Trigger Diagram
40 MHz bunch crossing rate
L0 Hardware Trigger : 1 MHz readout, high E_T/P_T signatures
450 kHz $h^\pm$
400 kHz $\mu/\mu\mu$
150 kHz e/γ
Software High Level Trigger
Partial event reconstruction, select displaced tracks/vertices and dimuons
Buffer events to disk, perform online detector calibration and alignment
Full offline-like event selection, mixture of inclusive and exclusive triggers
12.5 kHz (0.6 GB/s) to storage


LHCb Upgrade Trigger Diagram
30 MHz inelastic event rate (full rate event building)
Software High Level Trigger
Full event reconstruction, inclusive and exclusive kinematic/geometric selections
Buffer events to disk, perform online detector calibration and alignment
Add offline precision particle identification and track quality information to selections
Output full event information for inclusive triggers, trigger candidates and related primary vertices for exclusive triggers
2-5 GB/s to storage

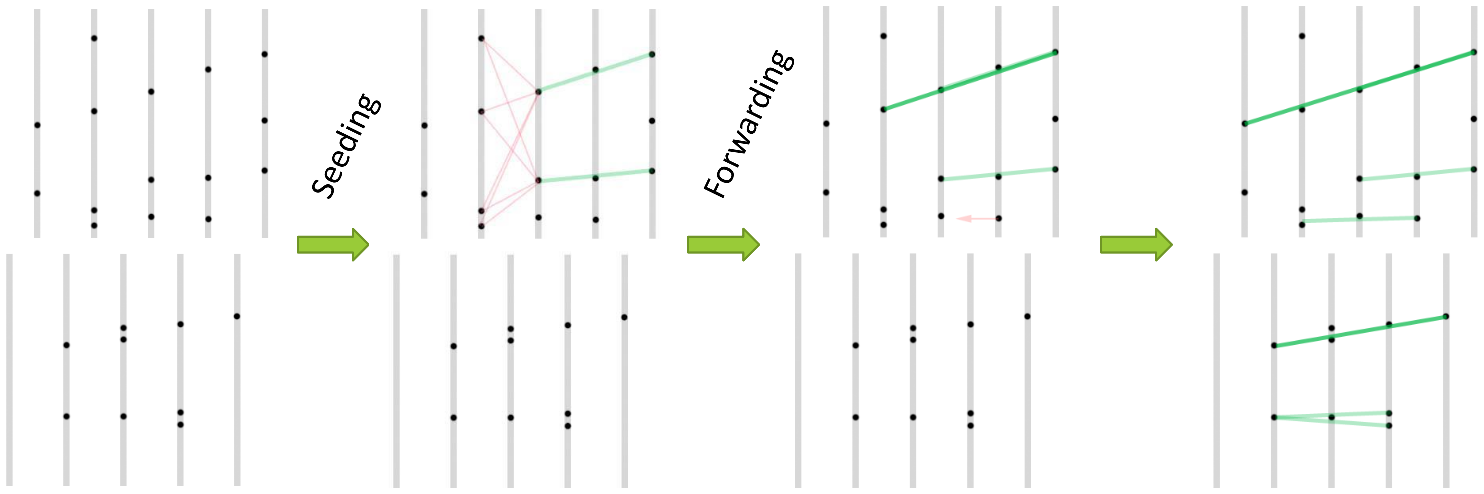
Want a factor 50 in performance: use GPGPUs?



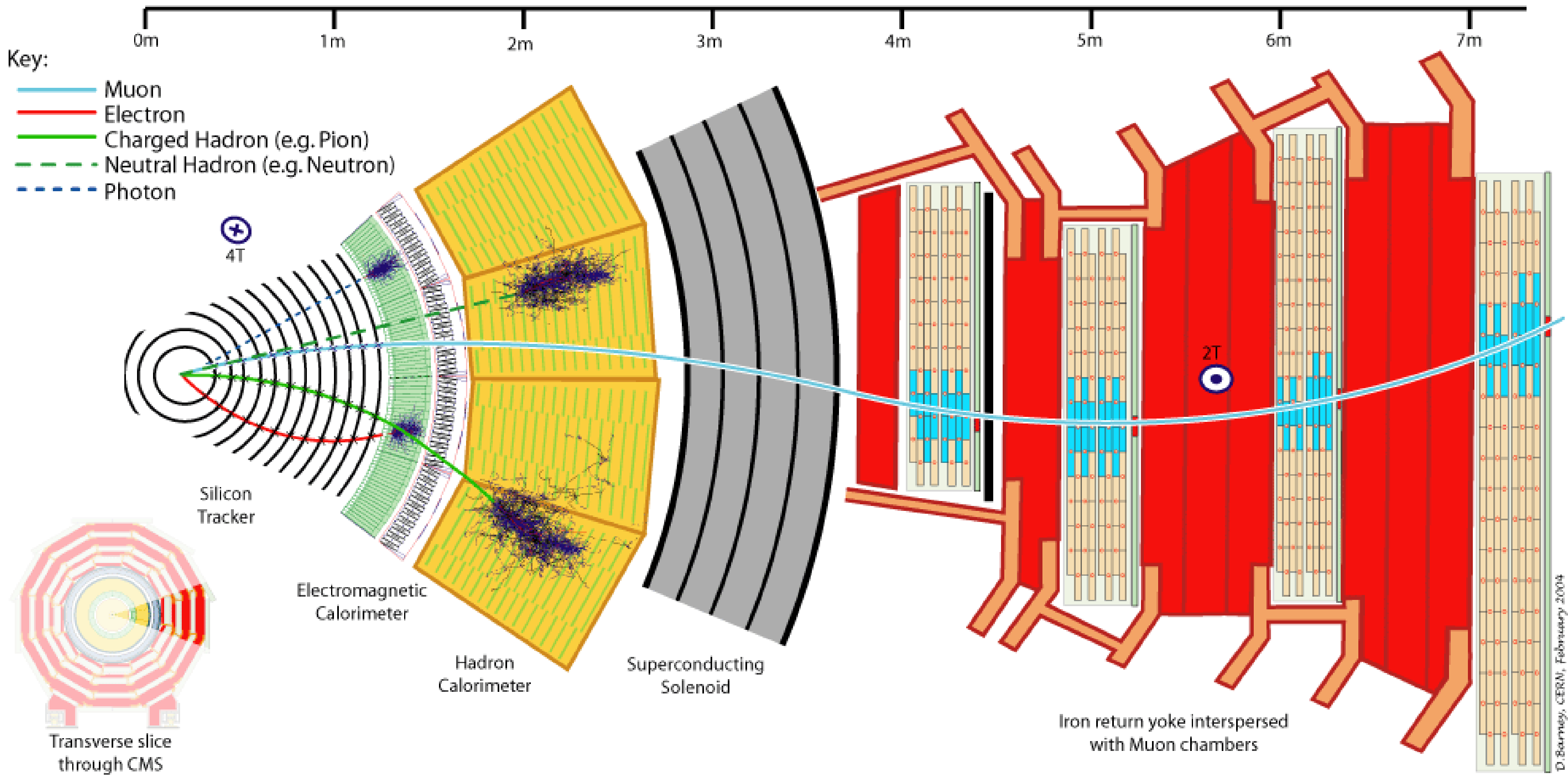
Source: nVidia

But does it work for most CPU demanding HEP problems? Example: tracking

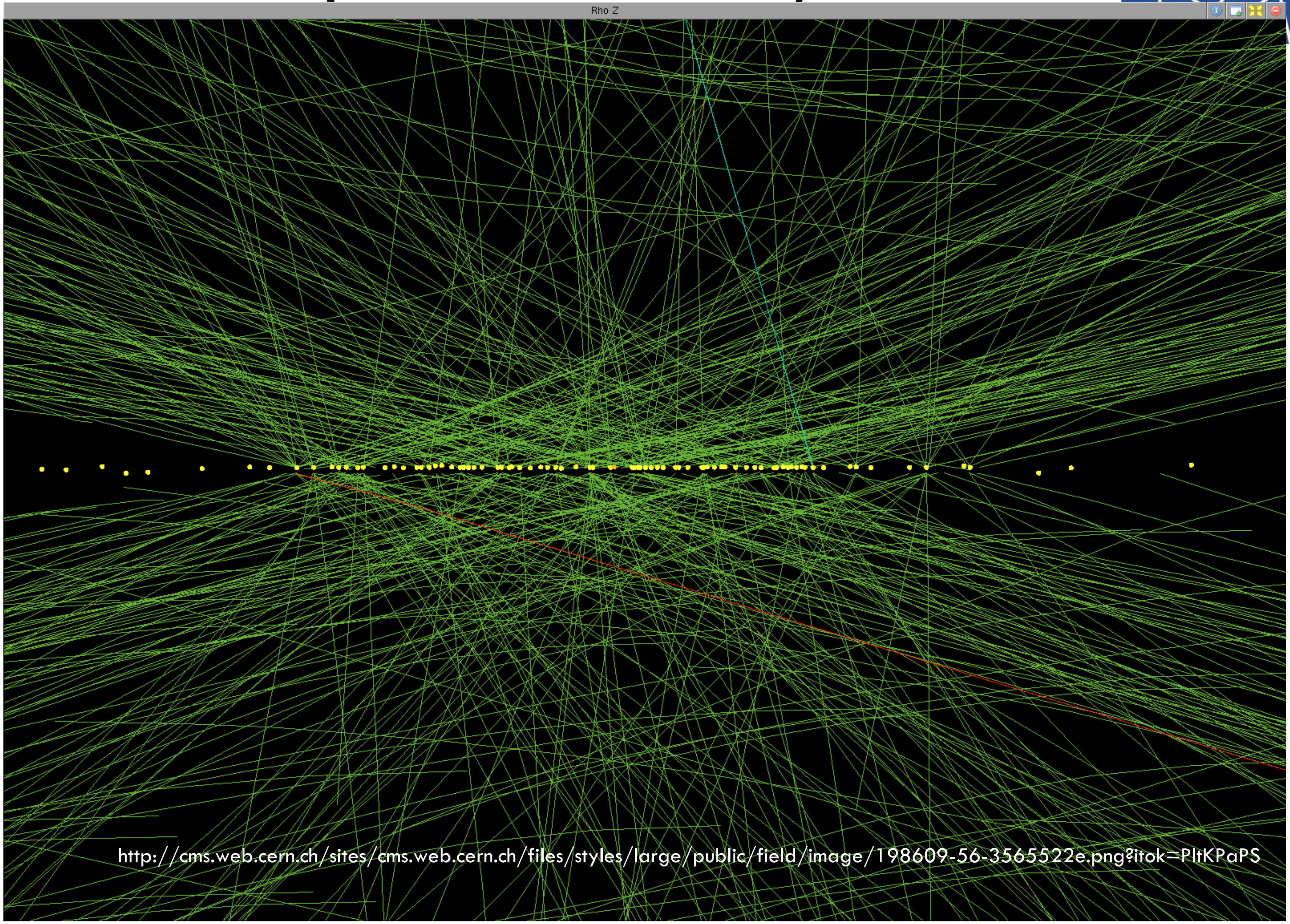
- Iterative algorithm that finds straight lines in collision event data in VeloPixel sub-detector
- Triplets of hits with best criterion are searched (seeding)
- Triplets are extended to tracks if a fitting hit can be found



The full nine yards in CMS

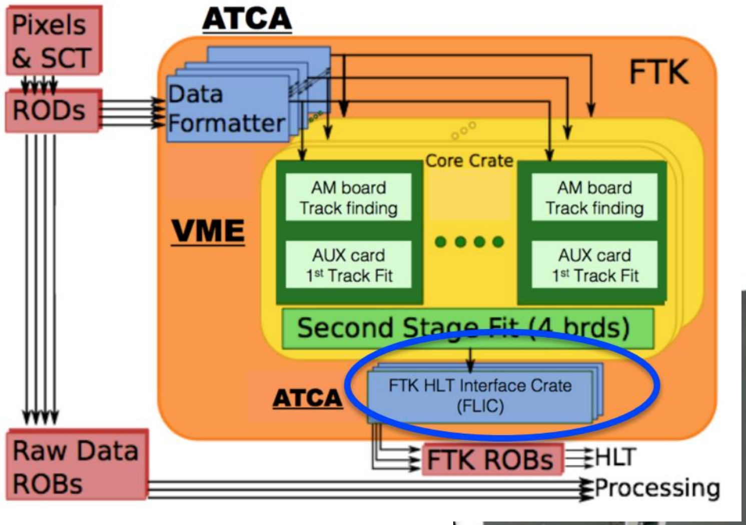

0m
1m
2m
3m
4m
5m
6m
7m
Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
Transverse slice through CMS
D.Barney, CERN, February 2004

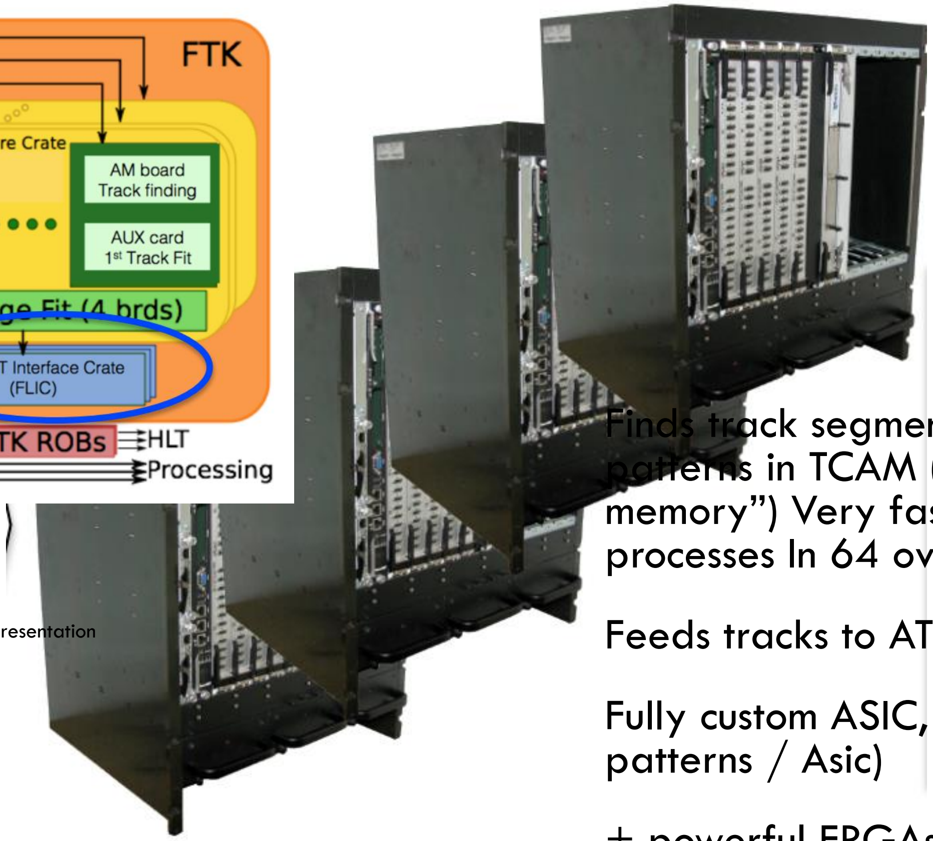
It's more complicated in reality…



http://cms.web.cern.ch/sites/cms.web.cern.ch/files/styles/large/public/field/image/198609-56-3565522e.png?itok=PltKPaPS

Fully custom: ATLAS FTK





http://atlasftk.uchicago.edu/presentations/presentation/yasuyuki.okumura.20141111.00.pdf

Finds track segments using pre-stored patterns in TCAM ("Associative memory") Very fast (microseconds) processes In 64 overlapping wedges

Feeds tracks to ATLAS high level trigger

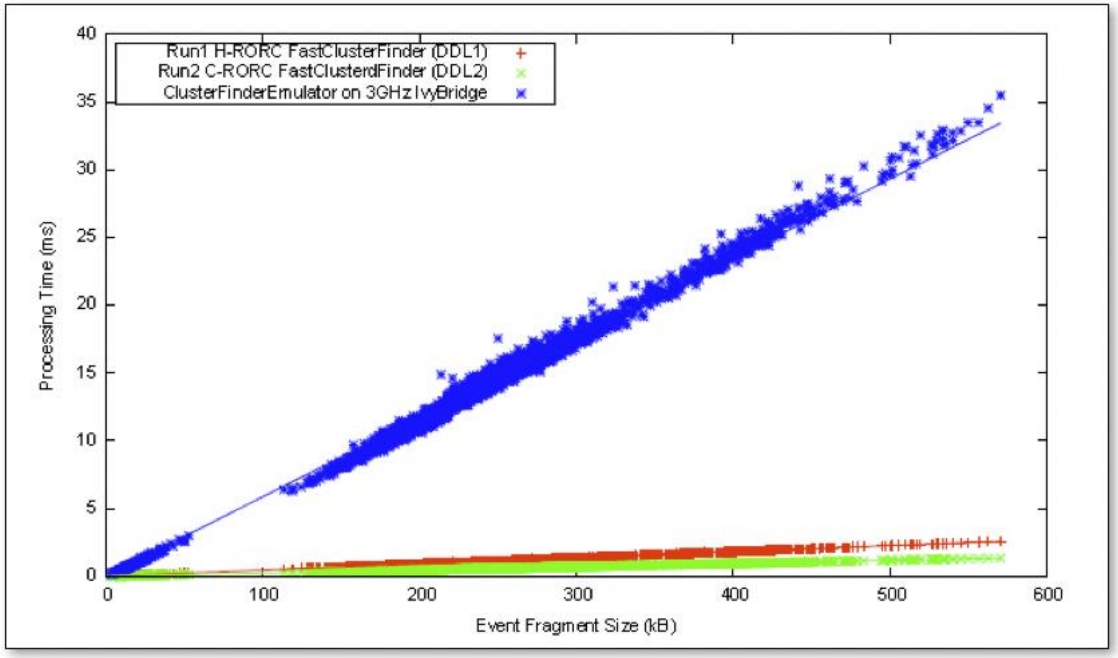
Fully custom ASIC, 2000 units (128k patterns / Asic)

+ powerful FPGAs (Virtex 7)

Acceleration with FPGA in ALICE

A Large Ion Collider Experiment

Technology: Hardware acceleration with FPGA



FPGA

- Acceleration for TPC cluster finder versus a standard CPU
- 25 times faster than the software implementation
- Use of the CRU FPGA for the TPC cluster finder

New (and old) challenges with accelerators in high energy physics

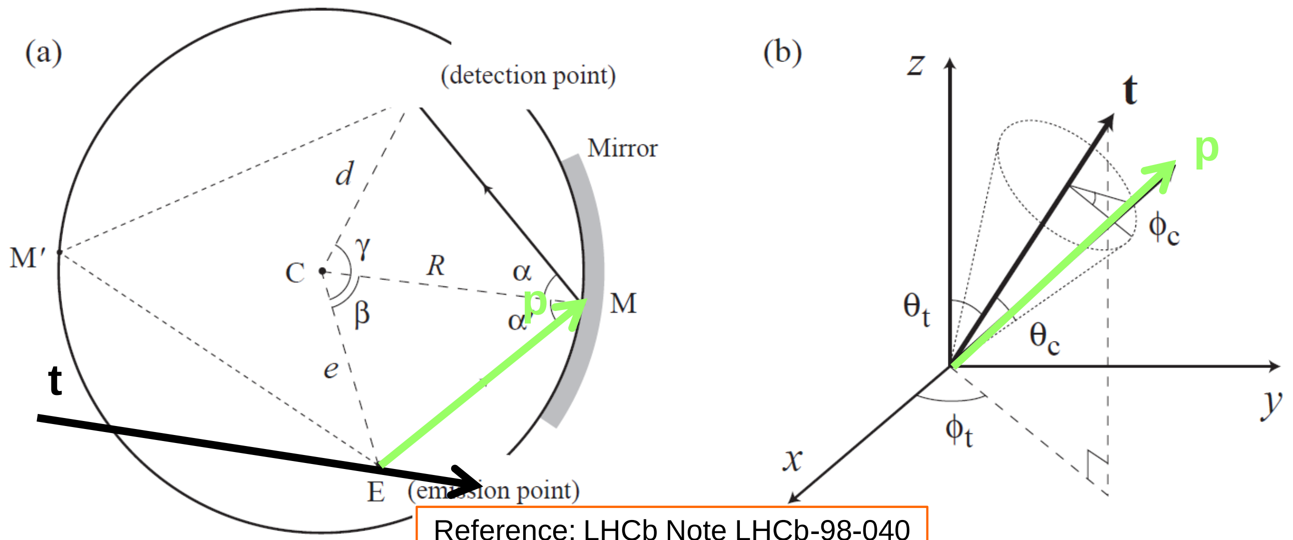
- Sophisticated algorithms need more time, bigger FPGAs, more data
- Long-term maintenance issues with custom hardware and low-level firmware
  - Upgrades usually mean replacing all the hardware
- Exact reproducibility of results without the custom hardware challenging and/or computationally intensive. Can one give a formal proof? Or use frame-works which ensure this? OpenCL goes in the right direction but seems to be suboptimal / little popular on many platforms (except FPGA)

Complex algorithm on FPGA RICH PID Algorithm in LHCb

Calculate Cherenkov angle **Θ_c** for each track **t** and detection point **D**, inverse ray-tracing, hyperbolic functions, etc…

Currently not processed for every event, because it is too costly!

Very challenging on FPGA (quartic equations, hyperbolic functions etc…



Reference: LHCb Note LHCb-98-040

Compare Verilog - OpenCL

Verilog

OpenCL

Development time

2.5 months vs 2 weeks

3400 lines Verilog vs 250 lines C

Faster

Easier

Performance

CQRT: x35 – x30

RICH: x35 – x26

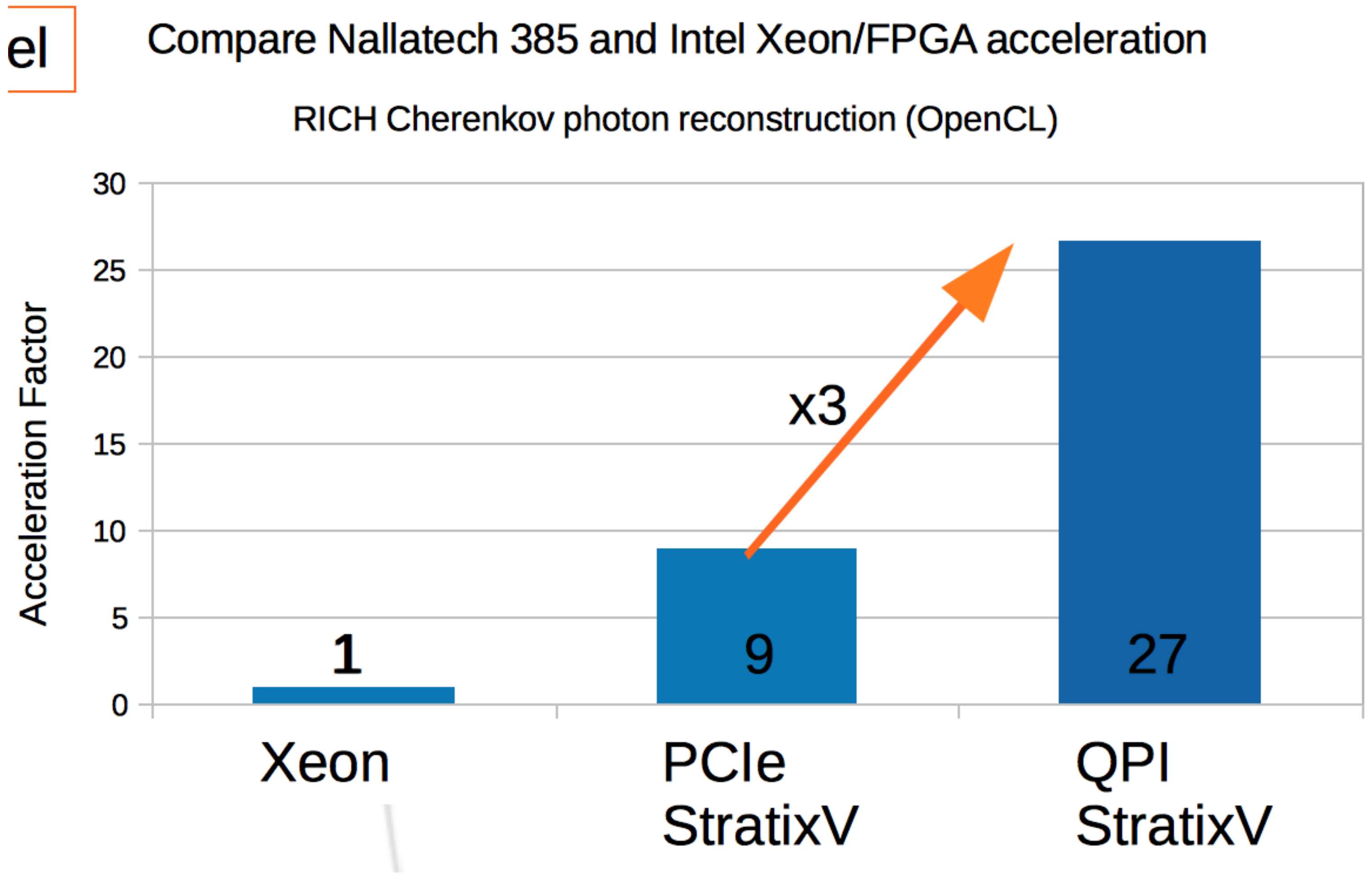
Comparable performance

	Verilog RTL	OpenCL
FPGA Resource Type	FPGA Resources used [%]	FPGA Resources used [%]
ALMs	88	63
DSPs	67	82
Registers	48	24

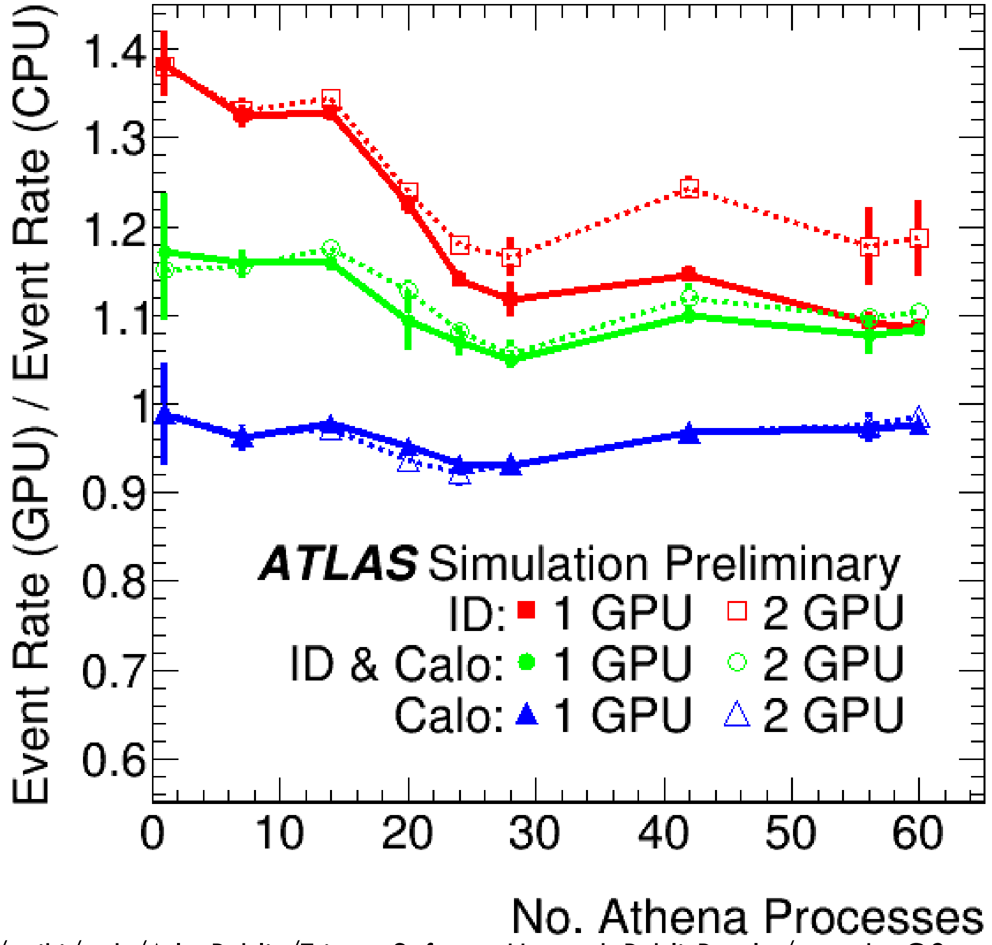
Similar resource usage

Christian Färber, openlab fellow, CERN/Intel HTCC project

More bandwidth is better!

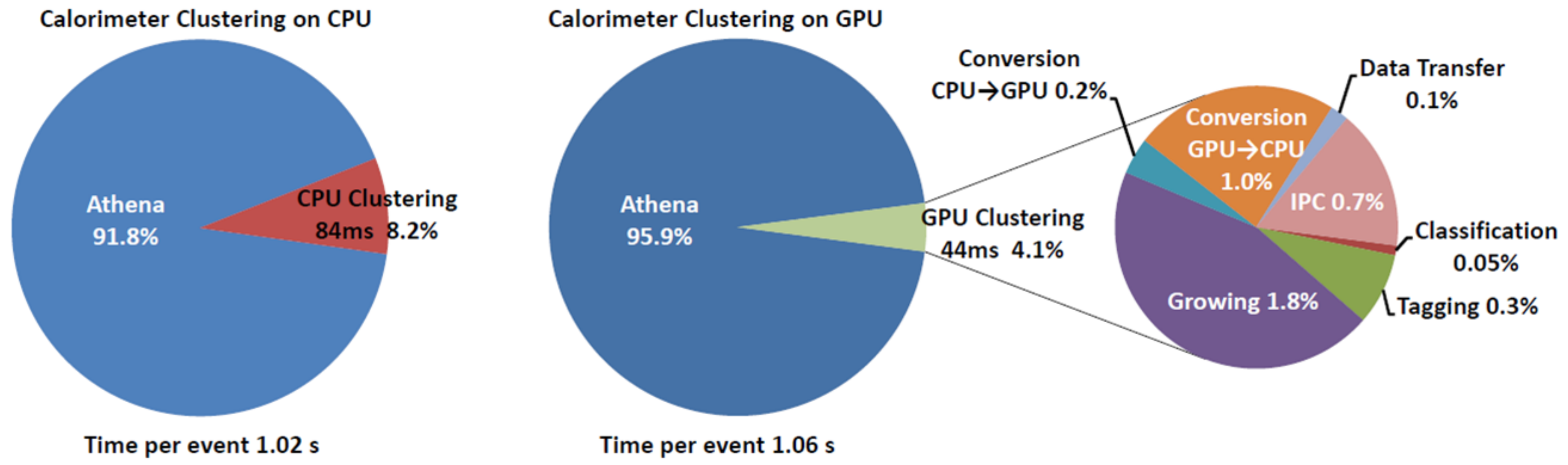


It’s not so simple in practice to get factors of 10 for HEP problems



https://twiki.cern.ch/twiki/pub/AtlasPublic/TriggerSoftwareUpgradePublicResults/speedupG2.png

Kernels and overheads

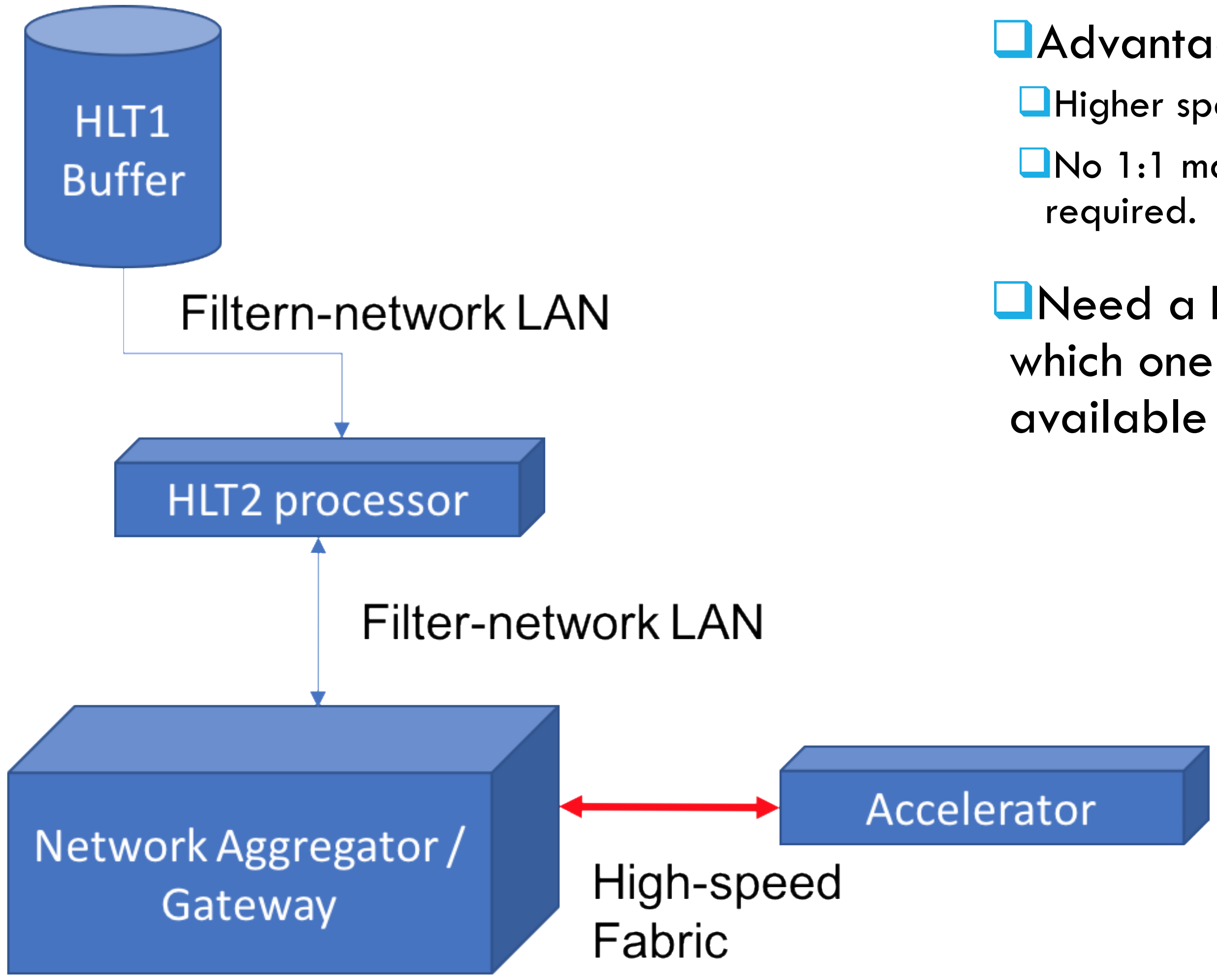


https://twiki.cern.ch/twiki/pub/AtlasPublic/TriggerSoftwareUpgradePublicResults/CaloExecutiontimePiChart3.png

Costing accelerators or *There ain't no such thing as a free lunch*

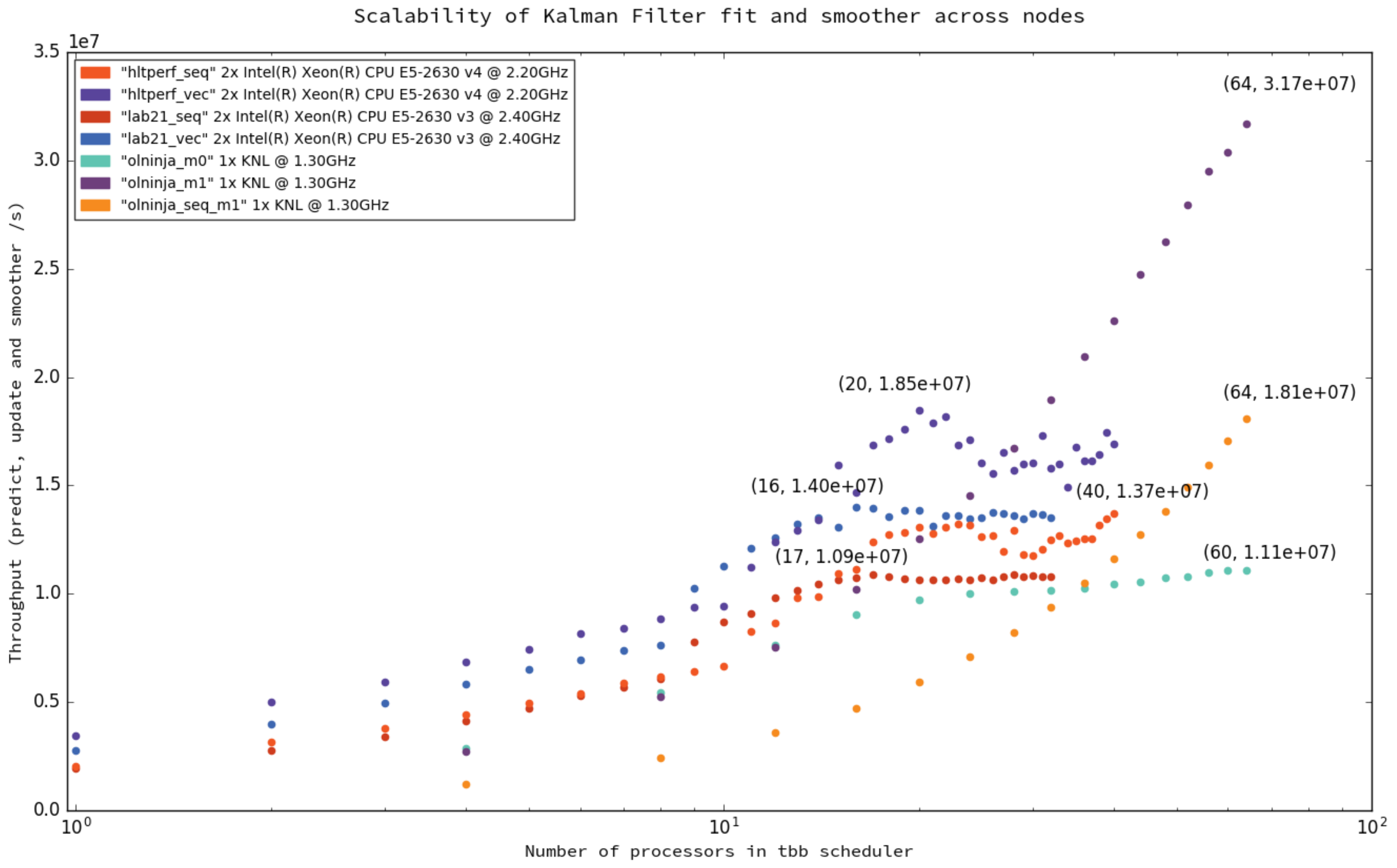
- ❑Required reading on this subject: the ALICE O2 TDR: https://cds.cern.ch/record/2011297/files/ALICE-TDR-019.pdf (page 58ff)
- ❑How much resources / cores are needed for data-transfer to/from host?
- ❑How much resources / cores are needed for data-conversion to / from host and host software framework?
- ❑After all is said and done: speed-up factors between 2 and 4 are measured (comparing full server to full GPGPU in ALICE) → numbers vary of course with specific CPU and GPGPU

How to use fabric-attached accelerators?



- Advantages of fabric attach:
  - Higher speed than PCIe
  - No 1:1 matching of host and GPGPU required.
- Need a high speed fabric though, which one the other hand is normally available in Online systems

Kalman Filter on Xeon® & KNL



Lots of challenges remain

- Many classical algorithms in HEP are not so easily profiting from accelerator architectures
  - Lack of parallelism, divergent branches, too small / too large "events", data-transfer overheads
- Machine learning (training and inference) gets a huge boost from these accelerators but only starting to investigate if traditional problems such as tracking, particle identification can be solved with it → next work-shop ☺
- However there are some success-stories with accelerators already:
  - ALICE GPGPU + FPGA, LHCb FPGA, …
  - Clearly need (much) more bandwidth between accelerator and servers
- Programming remains challenging, in particular but not only for FPGA, scalability is still a challenge (not only on KNL)
- How do we integrate accelerators well with minimal overheads in our large software frameworks?
- How to ensure the reproducibility of the results without accelerators or even across different ones