LHCb ideas and path toward vectorization and parallelization

Niko Neufeld

CERN, Wednesday, Nov 28th 2012

Fourth International Workshop for Future Challenges in Tracking and Trigger Concepts

Disclaimer & Acknowledgements

- This talk contains very few technical results
  - Short notice, LHCb week ☺ and … most work in LHCb is simply at a very early stage today
- Material presented comes of of many discussions with my colleagues in and outside LHCb, both from the online and offline world
- Very few of what is presented is LHCb specific

From the Intro to the LHCb Manycores workshop April 2012

- How well can many-cores solve important LHCb problems (many small events, secondary vertices, tracking, particle-ID)
- How do we evaluate the overall (cost-)effectiveness of these technologies?
- How can we make a code-base working with and without these (albeit at different performance) so that things can be run also “off-line” on other sites
  - Not all Grid sites will have “our” chosen coprocessor-cards
- How will these codes be maintained?
- Can we develop frameworks allowing non-expert developers to contribute → you think Boost and STL are tricky to master? Have a look at a decent technical book on CUDA

Two approaches for better CPU use

1. "Evolutionary": better use of modern processor-resources within existing frameworks
   - Make classical event-parallel processing more efficient: late fork-ing, x32, …
   - Try to make key, well isolated, pieces of code thread-safe
2. "Revolutionary": re-think everything
   - New frame-works for many-core architectures: GPU, MIC, etc…

LHCb specialties

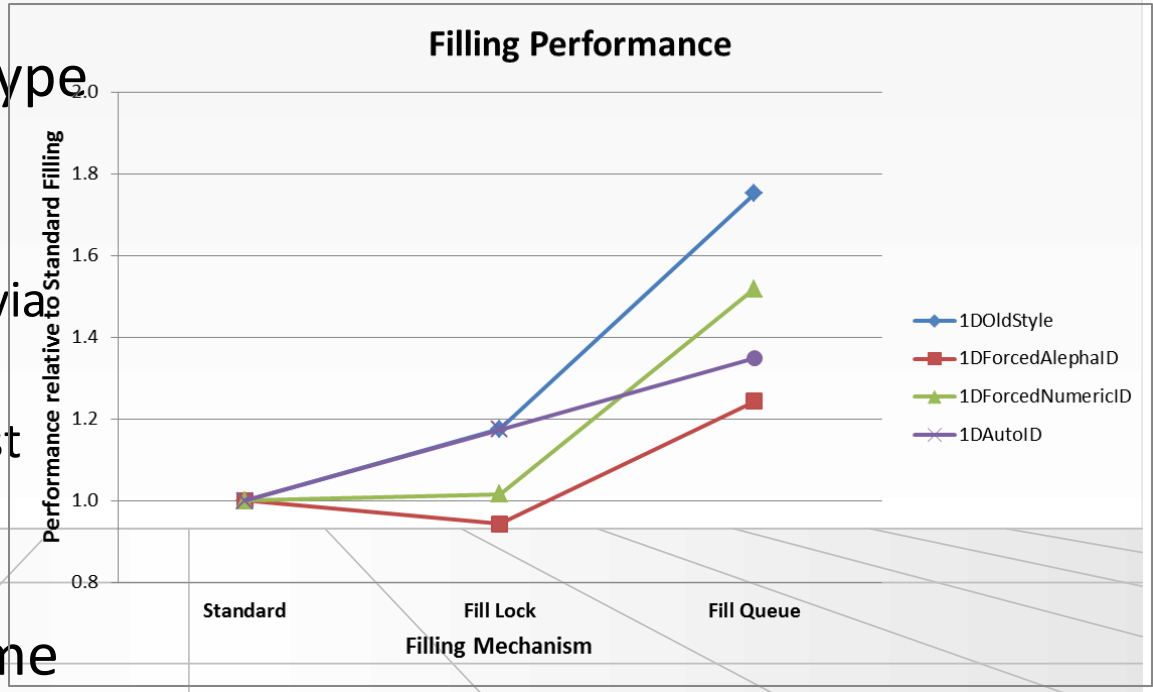
- Main specialty are the small events (60 kB today, ~ 100 kB after the upgrade) with little pileup (today about 1.5 to 2, later up to 4) and consequently relatively short processing time (about 30 ms for the sequential trigger code)
- → it seems that more can be gained by processing many events in parallel rather than individual events in a parallel fashion

Reducing memory footprint

- Many efforts under way...
- Analysis of the ROOT Persistence I/O Memory Footprint
- Working on GaudiMP (c.f. talk on Friday morning)
- Kernel Same page Merging (KSM)
  "Reducing the Memory Footprint of Parallel Applications with KSM"
- "X32" ABI i.e. 32 bits pointers in 64 bit linux

Targeted paralellisation: Histogram filling

- Solutions b (pthread mutex) and c (TBB concurrent queue) have been implemented in a prototype of the histogram service
  - Choice between old functionality and b) or c) via job options
  - Tested with histogram test option file from Gaudi Examples
- The test delivers also some sort of performance summary



Results from B. Jost

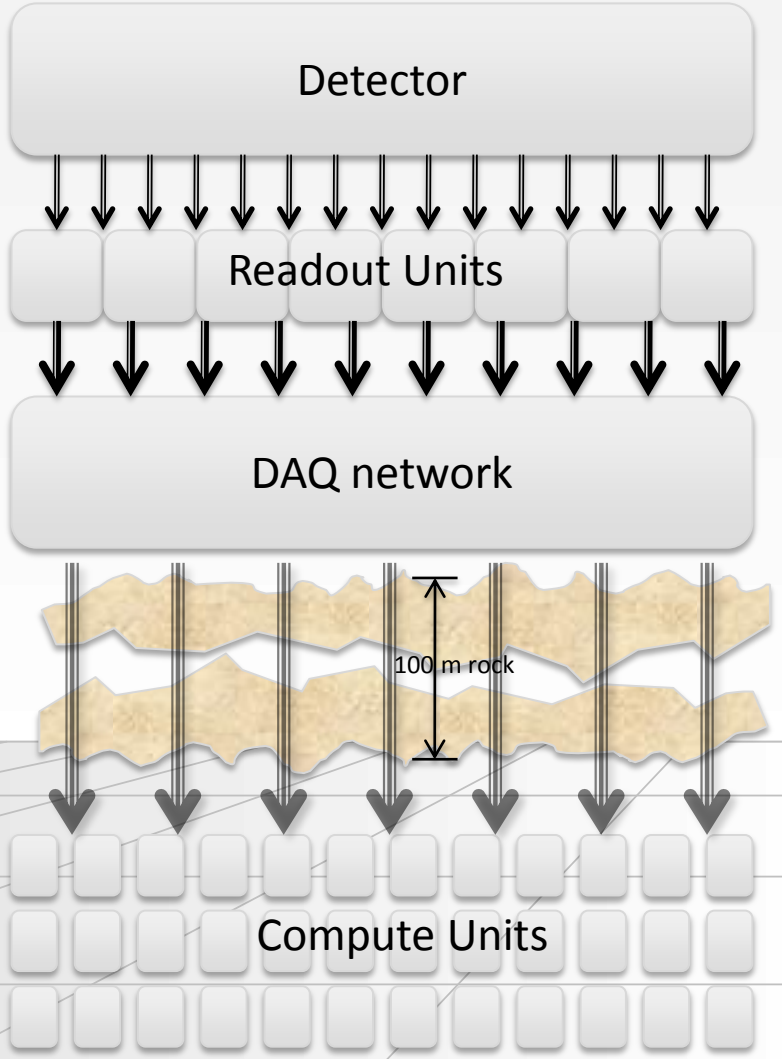
SIMD / vectorization

- Not much explicitly done right now, but we understand that this will be very important
- So far only tried with auto-vectorization (gcc 4.6) → no gain
- Lack of vectorization felt most painfully on Xeon/Phi and GPUs

Two time-scales

- LHCb software today and through LS1 and LS2
  - CPU evolution is not standing still
  - But need to support a large software base for a large user community (ongoing analyses) → can only apply "transparent" improvements
- LHCb software for the upgraded experiment after LS2 (major upgrade)
  - Need to try to anticipate what will be "the" way to go from 2018: GPUs? Xeon/Phi? Something else?
  - Everything is on the table

The dataflow for the upgraded DAQ

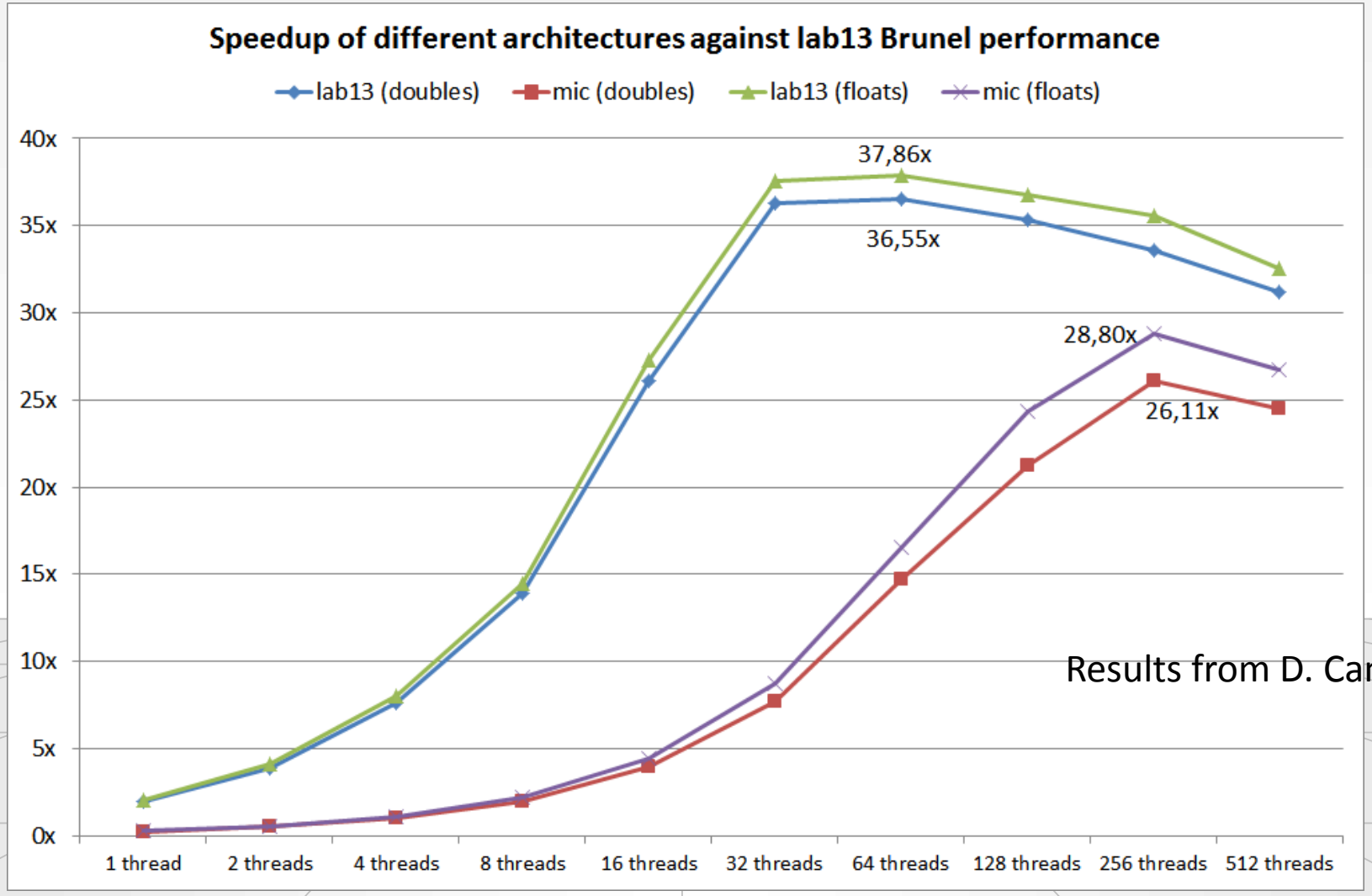


- GBT: custom radiation- hard link over MMF, 3.2 Gbit/s (about 10000)
- Input into DAQ network (10/40 Gigabit Ethernet or FDR IB) (1000 to 4000)
- Output from DAQ network into compute unit clusters (100 Gbit Ethernet / EDR IB) (200 to 400 links)

Two worlds

- Online:
  - Trigger system for the LHCb upgrade (next slide)
  - Full control over hardware (servers, interconnect)
  - Limited essentially only by power, cooling, money and – most importantly – imagination and creativity
  - Continuous support by comparatively small & close-knit teams of system and development experts
- Offline:
  - Need to work with what is “there” – only one of many clients to Grid/Cloud resources
  - Need to provide software which runs the same way on large batch-farms and on the laptop of individual researchers
  - Works with a large, “anonymous” or at least remote, user-base

Online Pixel reconstruction using TBB



Results from D. Campora

The compute unit

- Receives event-fragments and assembles complete events (actually multiple events simultaneously)
  - No separate event-builder PC forseen
- Runs the selection algorithm
- Using some rough back-of-the-envelope estimates and Moore's law about 1600 servers (dual-socket) of the 2017 will be needed for the baseline upgrade event-filter (10 MHz)
- Each server needs to absorb about 8-9 Gigabit (1GB) of data per second (depends on event-size)

Challenges on the compute unit

- The CU must absorb 8 Gbit/s
- Feeding all data through a co-processor card will at least double the through-put to the system bus
  - Unless “snooping” is used and part of the processing can only be done on co-processor card, this will be tricky at these rates

The server of 2017

- It will be based on PCIe Gen 3 as the I/O system
- Current generation of Intel CPU (and the next two on the road-map) offer 40 PCIe Gen 3 lanes per socket → theoretical I/O of 320 Gbit/s
  - Need to verify what this means in practice but should get close to 90% → partially done on SandyBridge using GPUs and InfiniBand cards
  - Data can be DMA-ed directly to processor cache (by-passing slow main-memory)
- Optimal use of off-load engines and data distribution requires some control over the data-flow (not good for blind push)

LHCb “strategy”

In a few preliminary discussions we have very quickly settled on the following “musts”

- Any physics algorithm must be able to run on any offline / online available hardware producing the exactly same results (crucial for systematics etc…)
- We do not want a vendor / technology lock-in (for competitive prices online but also because we can not (strongly) influence what is offered on the Grid)
- We need a data-processing framework, which can process many events in parallel