High Performance Computing meets High Energy Physics

Daniel Hugo Cámpora Pérez
dcampora@cern.ch

PASC, 28th June, 2017

Universidad de Sevilla
CERN

Outline

HTCC overview

The network challenge

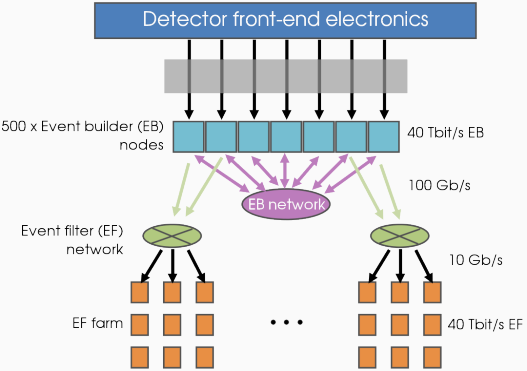
The software challenge

Conclusions

HTCC overview

The LHCb data acquisition (DAQ) system

- Data comes at a rate of *30 MHz*
- A throughput of *40 Tbit/s* needs to be processed in real-time
- All data will be processed in software



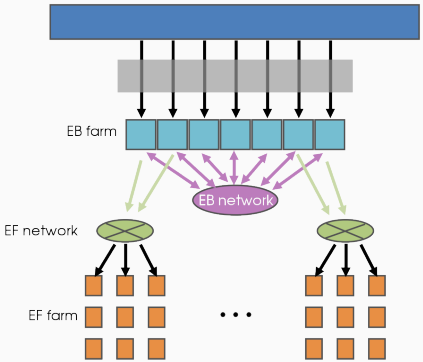
High Throughput Computing Collaboration (HTCC)

Apply upcoming Intel® technologies in an *Online* computing context at the Large Hadron Collider

- Data acquisition (DAQ) and event building
- Accelerator assisted decision taking on collected data

Use LHCb upgrade as an example, but applicable and useful for other experiments too!

LHCb DAQ architecture using Intel®



EB farm

- Intel® Xeon®
- Intel® Xeon® + FPGA
  w/ Intel® Omni-Path Architecture
- Intel® Xeon Phi™ w/ Intel® Omni-Path Architecture

EB network

- Intel® Omni-Path Architecture

EF network

- Intel® Omni-Path Architecture and / or 100 GbE

EF farm

- Intel® Xeon®
- Intel® Xeon® + FPGA
- Intel® Xeon Phi™

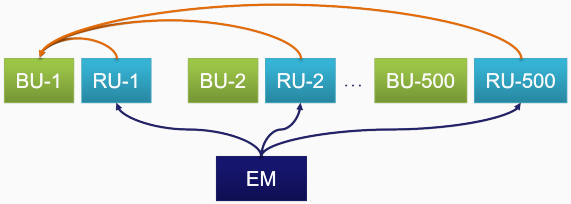
The network challenge

Event building network topology

Work with 4 units:

- Readout unit (RU) to read data from PCI-E readout board
- Builder unit (BU) to merge data from RU and send to filter unit
- Filter unit (FU) to select the interesting data (not considered here)
- Event manager (EM) to dispatch the work over BUs (credits)

RU / BU perform *allgather* communication operations to aggregate the data chunks from each collision

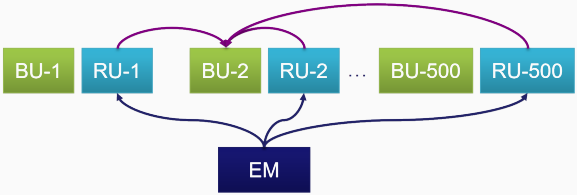


Event building network topology

Work with 4 units:

- Readout unit (RU) to read data from PCI-E readout board
- Builder unit (BU) to merge data from RU and send to filter unit
- Filter unit (FU) to select the interesting data (not considered here)
- Event manager (EM) to dispatch the work over BUs (credits)

RU / BU perform *allgather* communication operations to aggregate the data chunks from each collision

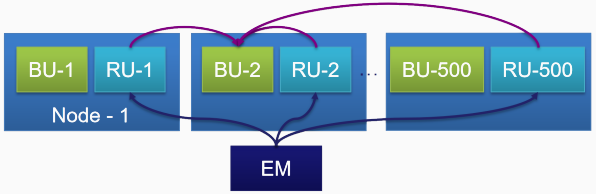


Event building network topology

Work with 4 units:

- Readout unit (RU) to read data from PCI-E readout board
- Builder unit (BU) to merge data from RU and send to filter unit
- Filter unit (FU) to select the interesting data (not considered here)
- Event manager (EM) to dispatch the work over BUs (credits)

RU / BU perform *allgather* communication operations to aggregate the data chunks from each collision

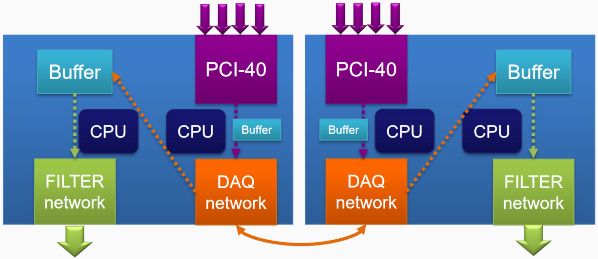


IO nodes hardware

Three IO boards at 100 Gb/s per node:

- PCI-40 for fiber input
- Event building network
- Output to filter farm

Handling these communications stresses the memory, requiring 400 Gb/s of total traffic per node



DAQPIPE

A benchmark to evaluate event building solutions

- Provides EM / RU / BU units

Support various APIs

- MPI
- libfabric
- PSM2
- Verbs
- TCP / UDP
- RapidIO

Manage communication scheduling models

- Barrel shift ordering (with N on-fly)
- Random ordering (with N on-fly)
- Send all in one go

DAQPIPE (2)

Three message sizes on the network

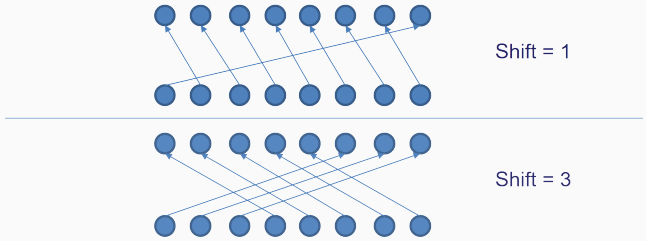
- Command 64 B
- Metadata 10 KB
- Data 1 MB

Configurable parameters

- Message size (512 KB / 1 MB)
- Number of pending gather (credits)
- Active connections by gather
- Processes per node

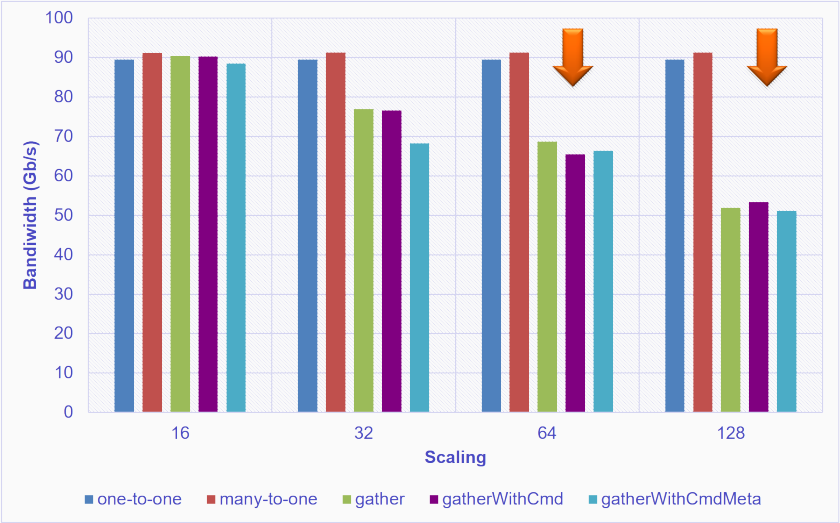
Barrel shift

In order to avoid congestion, a variant is to synchronously send data to predetermined receivers, with a configurable shift. Similar discussion as in [1].

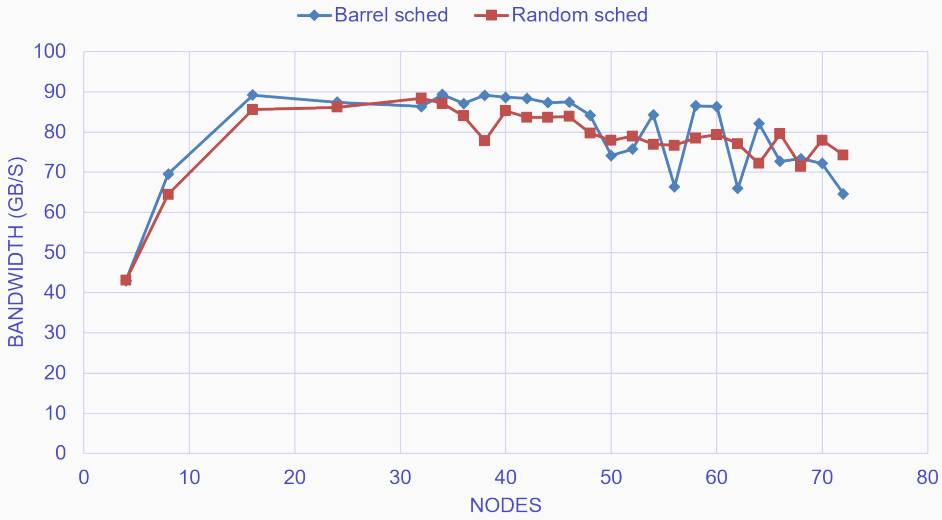


[1] Bandwidth-optimal All-to-all Exchanges in Fat Tree Networks, B. Prisacari

Micro-benchmarks on Intel® Omni-Path Architecture



Scaling on Infiniband

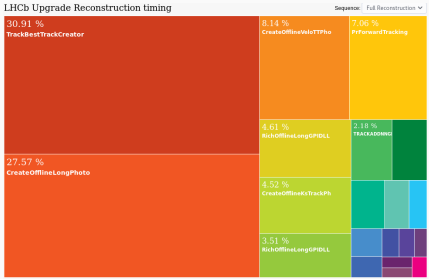


The software challenge

Status of LHCb codebase

More than 5 MLOCs of C++ (and some Python)

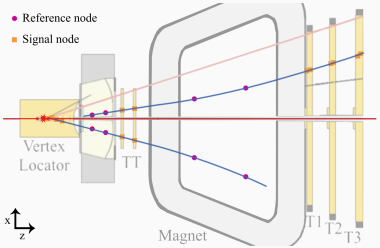
- Under redesign
  for SIMD and shared-mem parallelism
- Baseline remains Xeon® CPUs
- New
  framework uses TBB to dispatch algorithms
  to process events in a multi-threaded fashion
- Possible avenues to accelerate algorithms:
  - Offload critical functions to FPGA
  - Rewrite most time-consuming algorithms in a parallel fashion and use Xeon Phi™



The Kalman filter

The most time consuming algorithm, taking 60% of the time in the Online reconstruction, is the *Kalman filter*. The Kalman filter is a well-known linear quadratic estimator. For every particle *node*,

- *Predict* - The state of the system is projected according to a given model
- *Update* - The state is adjusted taking into account a measurement



Cross Kalman

We have developed a cross-architecture Kalman filter, targeting SIMD architectures. Our design considers three key components:

- Control flow
- Data structures
- Arithmetic backend

Control flow

Every particle can be considered a succession of nodes with an implicit computing order. Given that we receive hundreds of particle trajectories as an input, we are given the scheduling problem of assigning particle *nodes* to processors, where we attempt to minimize the number of computing iterations.

This problem is a variant of the number partitioning problem NPP, which is NP-complete [1].

Nevertheless, a Decreasing Time Algorithm (DTA) behaves well as a scheduling algorithm.

[1] Mertens S.: The Easiest Hard Problem: Number Partitioning. Computational Complexity and Statistical Physics, 125(2), 125-139 (2003)

Control flow (2)

We use a *static scheduler*

- Data locality is maximized
- Data is guaranteed to be aligned

```
it    in   out  act  vector (#particle-#node)
#540: 0000 0001 1111 { 112-9 80-11 81-11 113-10 }
#541: 0001 1110 1111 { 112-10 80-12 81-12 79-3 }
#542: 1110 0000 1111 { 107-2 109-1 108-2 79-4 }
#543: 0000 0000 1111 { 107-3 109-2 108-3 79-5 }
#544: 0000 0000 1111 { 107-4 109-3 108-4 79-6 }
```

Data structures

We use a *static scheduler*

- Data locality is maximized
- Data is guaranteed to be aligned
- Data is AOSOA

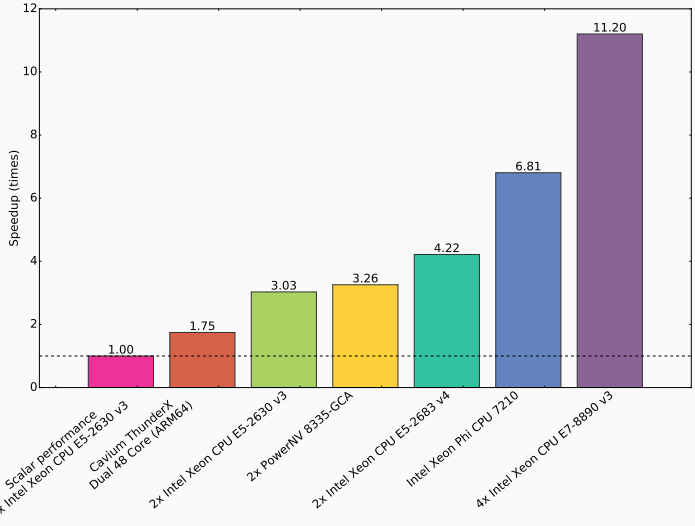
$$
\begin{vmatrix}
x_0 & x_1 & x_2 & x_3 \\
y_0 & y_1 & y_2 & y_3 \\
tx_0 & tx_1 & tx_2 & tx_3 \\
ty_0 & ty_1 & ty_2 & ty_3 \\
\overline{q} & \overline{q} & \overline{q} & \overline{q} \\
p_0 & p_1 & p_2 & p_3 \\
\sigma_{0,0} & \sigma_{1,0} & \sigma_{2,0} & \sigma_{3,0} \\
\vdots & \vdots & \vdots & \vdots \\
\sigma_{0,14} & \sigma_{1,14} & \sigma_{2,14} & \sigma_{3,14} \\
\chi^2{}_0 & \chi^2{}_1 & \chi^2{}_2 & \chi^2{}_3
\end{vmatrix}
$$

Arithmetic backend

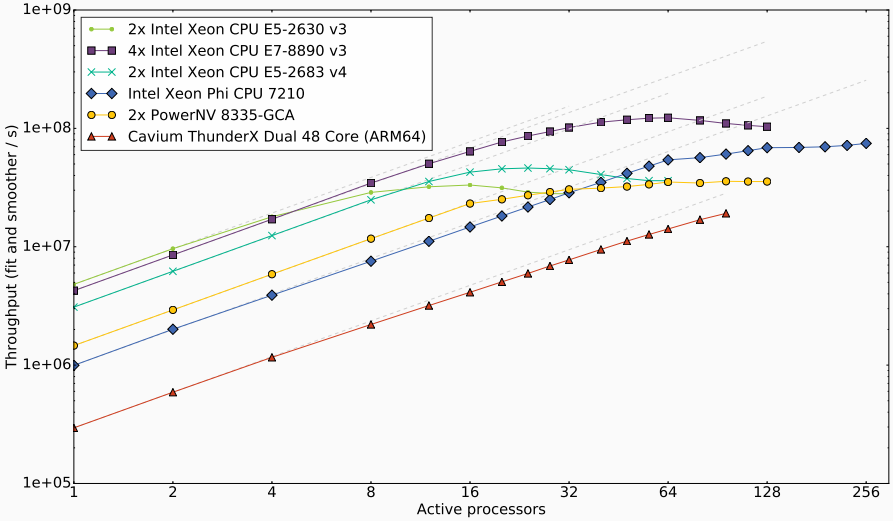
The bulk of the math of the application is written in an architecture-aware programming extension / library,

- VCL
- UMESIMD
- OpenCL
- CUDA

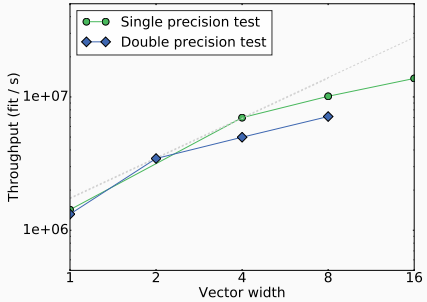
Speedup

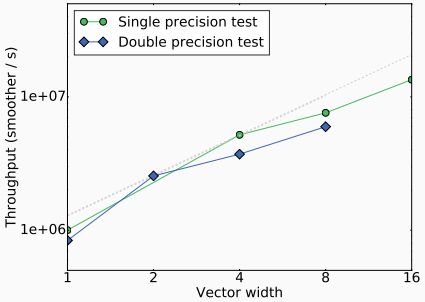

Speedup (times)
0
2
4
6
8
10
12
1.00
1.75
3.03
3.26
4.22
6.81
11.20
Scalar performance
2x Intel Xeon CPU E5-2630 v3
Cavium ThunderX
Dual 48 Core (ARM64)
2x Intel Xeon CPU E5-2630 v3
2x PowerNV 8335-GCA
2x Intel Xeon CPU E5-2683 v4
Intel Xeon Phi CPU 7210
4x Intel Xeon CPU E7-8890 v3

Scalability



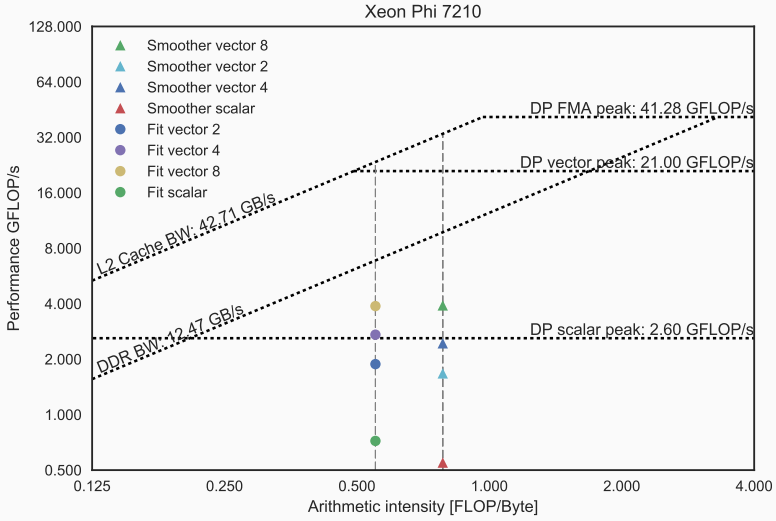
Vector width effect





- Tests performed on an Intel® Xeon Phi™ 7210
- Single precision results show a 1% deviation wrt. expected results

Roofline model



Conclusions

Conclusions

The LHCb DAQ faces a test in the upcoming years

- Network, software, storage, ...

We are currently evaluating network technologies and hardware architectures that fulfill our requirements

- 500-node farm with sustained 100 Gb/s bidirectional bandwidth, all-to-all traffic pattern
- Reconstruction software and farm capable of processing 40 Tbit/s of data

We have shown good cross-architecture performance of the main contributor to the software reconstruction

- There is no *one-fits-all* solution, if one requires best performance. As of now, each architecture requires a slightly different approach
- Many-core architectures are an option only if the framework and algorithms are prepared to take advantage of it

Thanks

Many thanks to

- The HTCC collaboration
- S. Balat for the results on DAQPIPE
- O. Bouizi and S. Harald for low-level code discussions and early results on Xeon Phi
- W. Hulsbergen and R. Aaij for mathematical discussions
- F. Lemaitre for the vectorized transposition code
- N. Neufeld, A. Riscos Núñez for their supervision