
CERN

Software framework for the LHCb upgrade

Sébastien Ponce, Stefan Roiser, Concezio Bozzi, on behalf of the LHCb Collaboration
sebastien.ponce@cern.ch

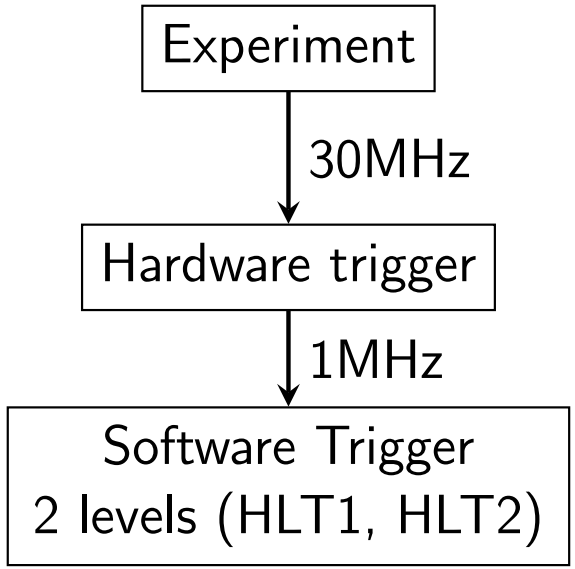
Outline

LHCb Run 3 landscape

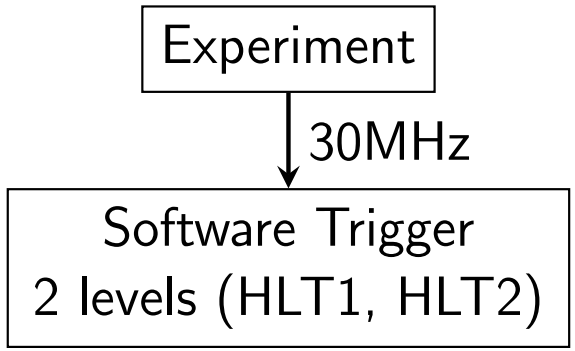
- Upgrade of the detector itself to take more luminosity (x5)
  - still 30MHz collisions
  - more pile-up (now 5.5, was 1.1)

LHCb Run 3 landscape

- Upgrade of the detector itself to take more luminosity (x5)
  - still 30MHz collisions
  - more pile-up (now 5.5, was 1.1)
- New trigger system
  - no hardware, fully software
  - input rate x30 !

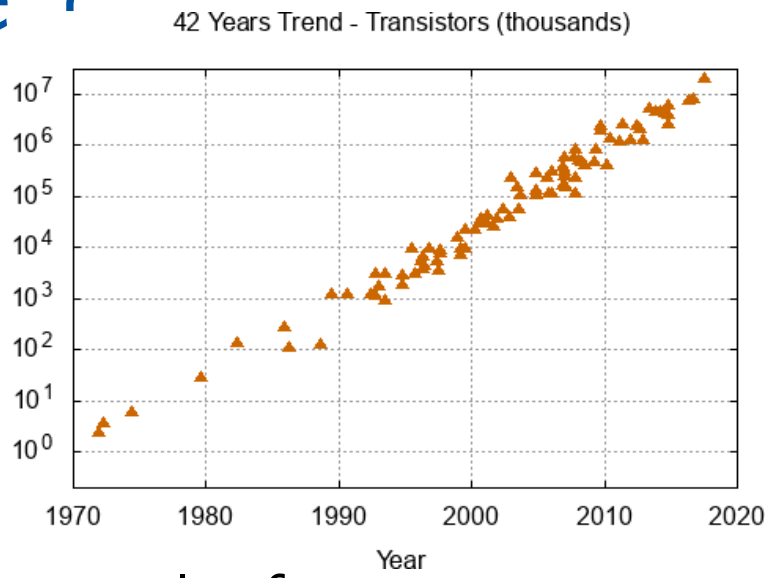


Run2



Run3

How to make it possible ?

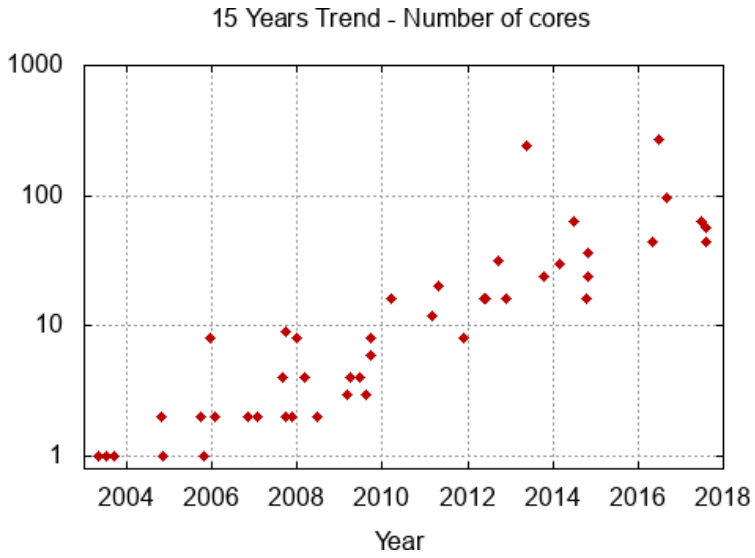


- Hardware evolution[1]
  - Moore's law still holding
  - in numbers of transistors
- Better use of hardware
  - more parallelization
  - many cores, vectorization, hyperscalar features, ...
- Improving our software
  - using latest programming techniques
  - improving our memory accesses
  - reengineering some algorithms[1]

[1] Data source : https://github.com/karlrupp/microprocessor-trend-data, modified to only show transistors

Adapting to many-core

- Multi-core is over,
  we have now many-core[2]
  - easily 40, up to 100
    logical CPU cores
- Consequently multi-process
  is becoming hard
  - as memory pressure is too high
- LHCb software has opted for multithreading
  - parallelizing at the granularity of events
  - running as many events in parallel as cores



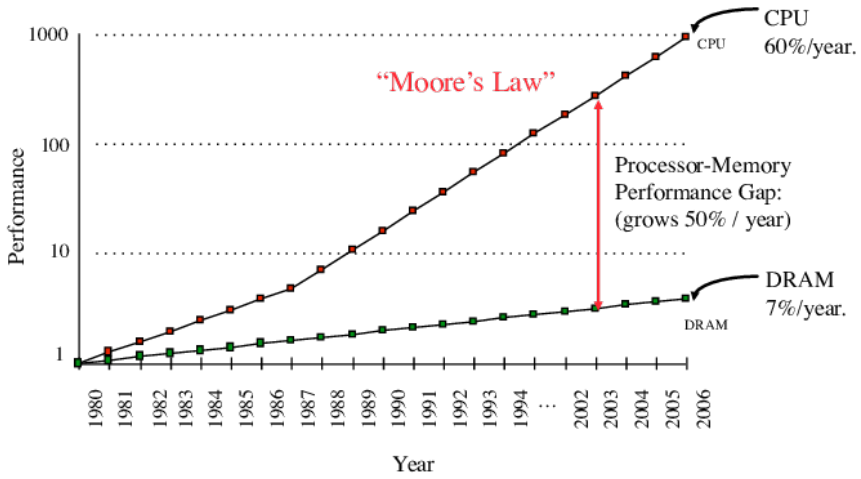
[2] Data source : https://github.com/karlrupp/microprocessor-trend-data, modified to only show transistors

How to handle thread safety ?

- Use a thread safe framework
  - Gaudi hive, and its new "functional" algorithms
- Rely on modern C^{++} standards
  - making extensive use of constness in particular
- Go step by step through the code
  - starting small, learning the techniques
  - spreading the knowledge

Optimization opportunities

- Careful memory usage
  - as memory has become slow compared to CPU
  - and thus cache misses are costful



- Vectorization
  - vector units are getting wider
  - but making good usage of them requires rethinking our data models
- C^{++} new features
  - many are related to optimization

Spreading the knowledge

- Regular tutorial sessions in the collaboration
  - core C^{++} programming
  - LHCb framework and coding practices
  - optimization tools, collaborative tools
- Regular hackathons
  - with support of experts

Monitoring the progress

- Many tools available to measure performance
  - including internal tracing of the code
- Regular (nightly) builds and runs[5]
  - checking validity on all platform
  - checking raw performance
  - recording all performance indicators
    - e.g. cache misses, vectorization levels
- all history available to extract trends

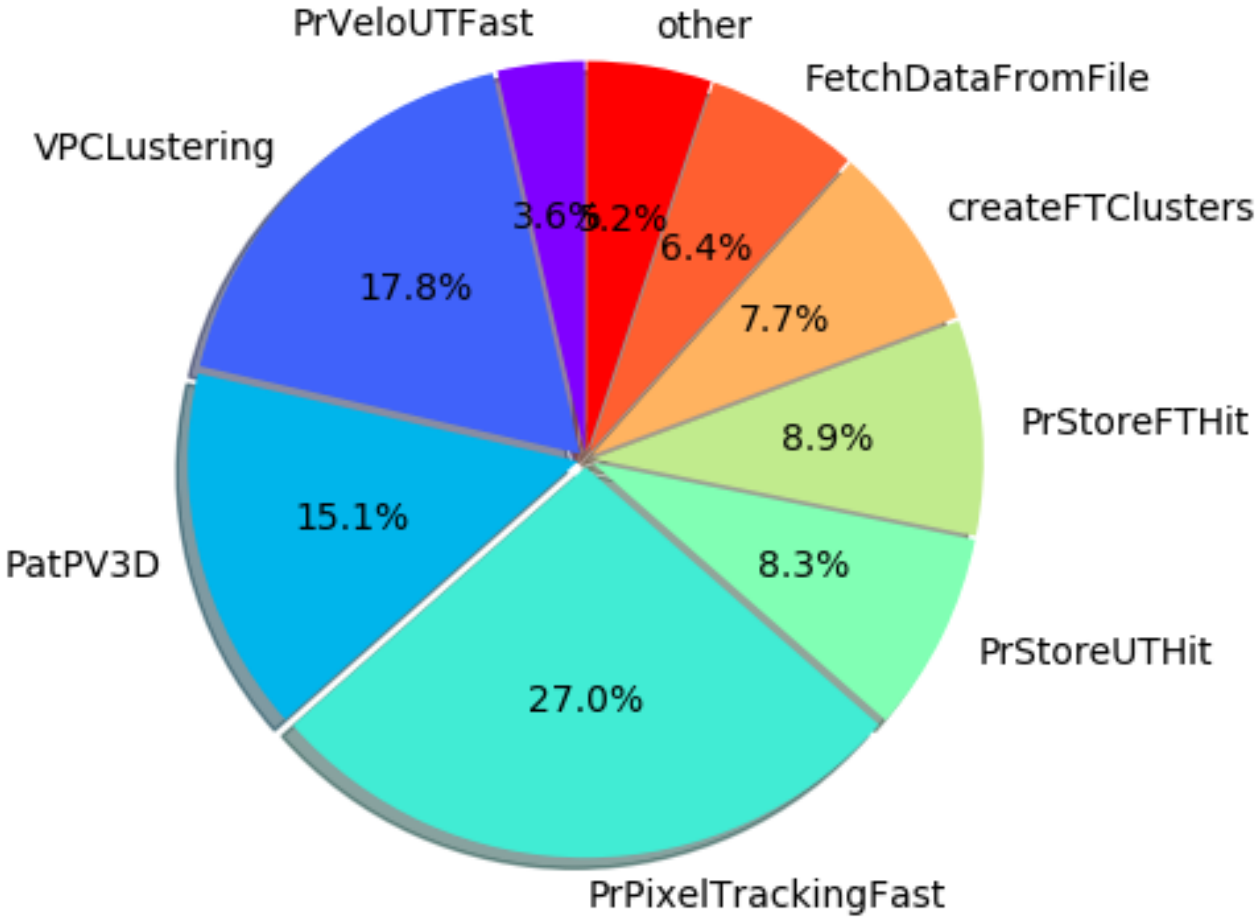
lhcb-head - build: 1900 (2018-06-28)

head of everything against Gaudi mastermaster and LCG_93 (ROOT 6.12.06)
available on: **cvmfs**

Project	Version	x86_64-slc6-gcc7-opt Completed at 14:45:38 Platform diff		x86_64-slc6-gcc7-dbg Completed at 15:49:08 Platform diff		x86_64-centos7-gcc7-opt Completed at 13:22:42 Platform diff	
LCG	93						
Gaudi	master	build	tests	build	tests (1)	build	tests
Online	HEAD	build	tests	build	tests	build	tests (58)
LHCb	HEAD	build	tests	build	tests	build	tests
Lbcom	HEAD	build	tests	build	tests	build	tests
Boole	HEAD	build	tests	build	tests	build	tests
Rec	HEAD	build	tests	build (24)	tests	build	tests
Brunel	HEAD	build	tests (3)	build	tests (21)	build	tests (3)

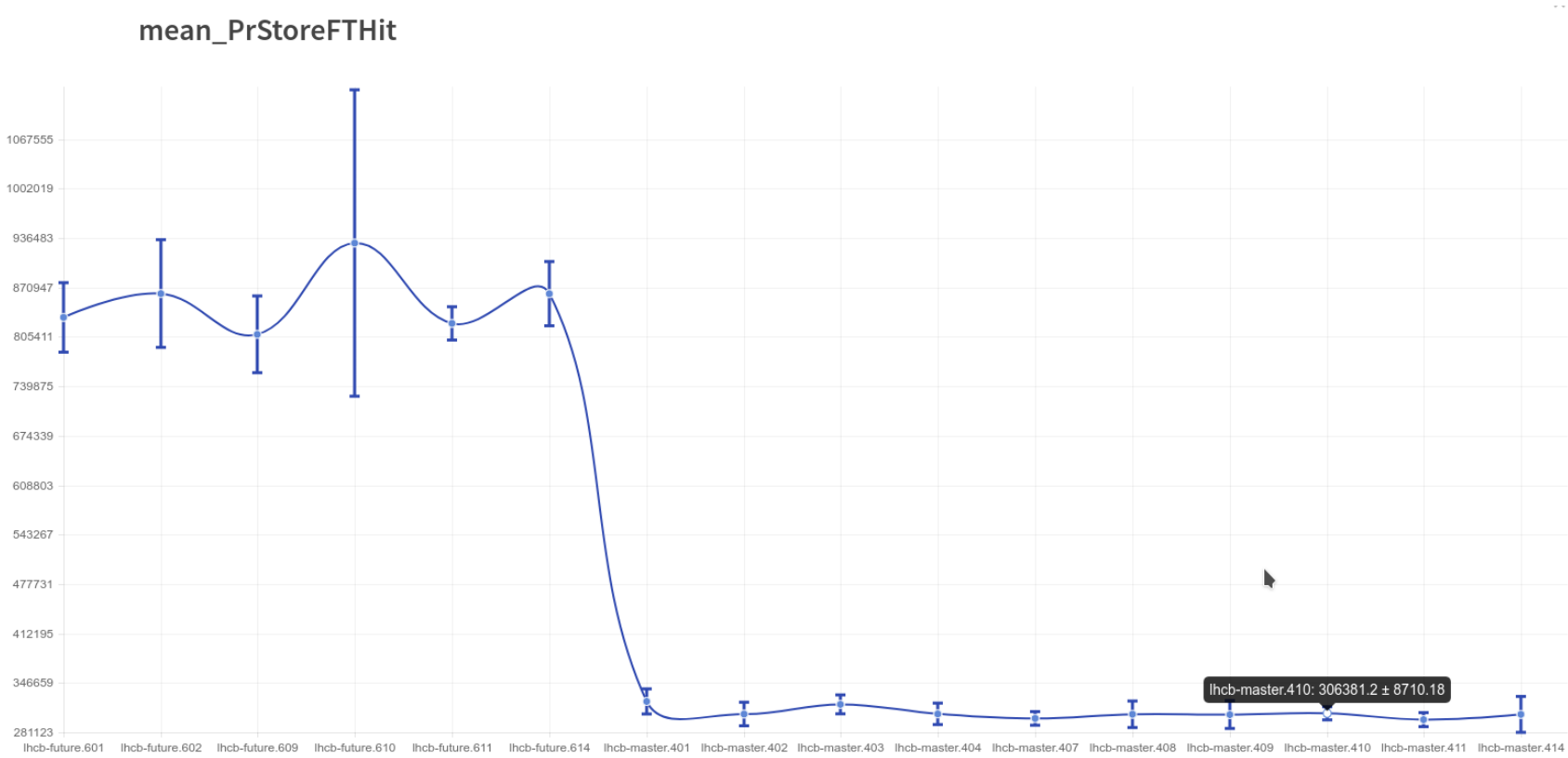
Monitoring the progress

- Many tools available to measure performance
  - including internal tracing of the code
- Regular (nightly) builds and runs[5]
  - checking validity on all platform
  - checking raw performance
  - recording all performance indicators
    - e.g. cache misses, vectorization levels
- all history available to extract trends


PrVeloUTFast
other
FetchDataFromFile
VPCLustering
createFTClusters
3.6%
5.2%
6.4%
17.8%
7.7%
PrStoreFTHit
8.9%
15.1%
PatPV3D
8.3%
PrStoreUTHit
27.0%
PrPixelTrackingFast

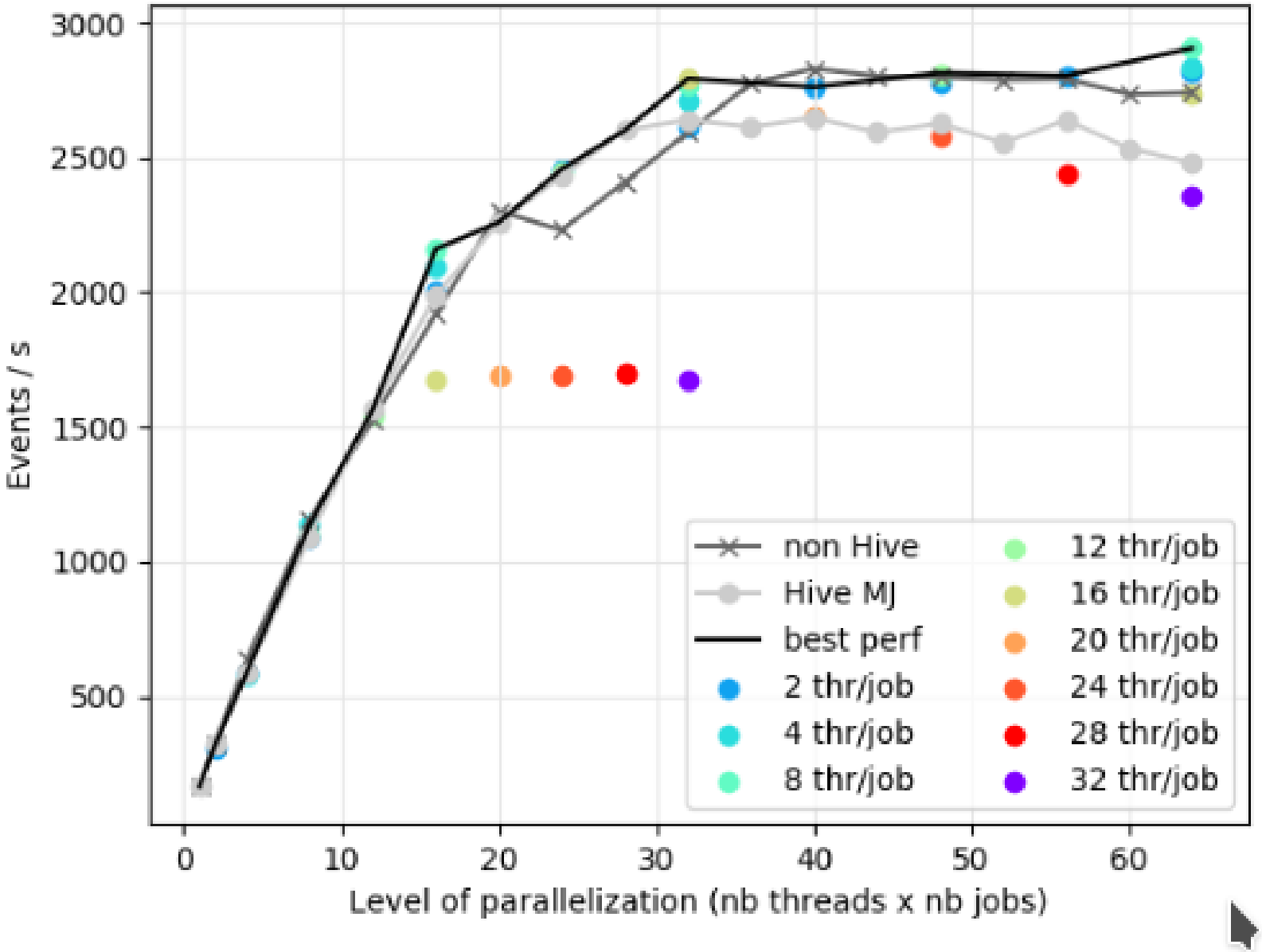
Monitoring the progress

- Many tools available to measure performance
  - including internal tracing of the code
- Regular (nightly) builds and runs[5]
  - checking validity on all platform
  - checking raw performance
  - recording all performance indicators
    - e.g. cache misses, vectorization levels
- all history available to extract trends


mean_PrStoreFTHit
1067555
1002019
936483
870947
805411
739875
674339
608803
543267
477731
412195
346659
281123
lhcb-future.601
lhcb-future.602
lhcb-future.609
lhcb-future.610
lhcb-future.611
lhcb-future.614
lhcb-master.401
lhcb-master.402
lhcb-master.403
lhcb-master.404
lhcb-master.407
lhcb-master.408
lhcb-master.409
lhcb-master.410
lhcb-master.411
lhcb-master.414
lhcb-master.410: 306381.2 ± 8710.18

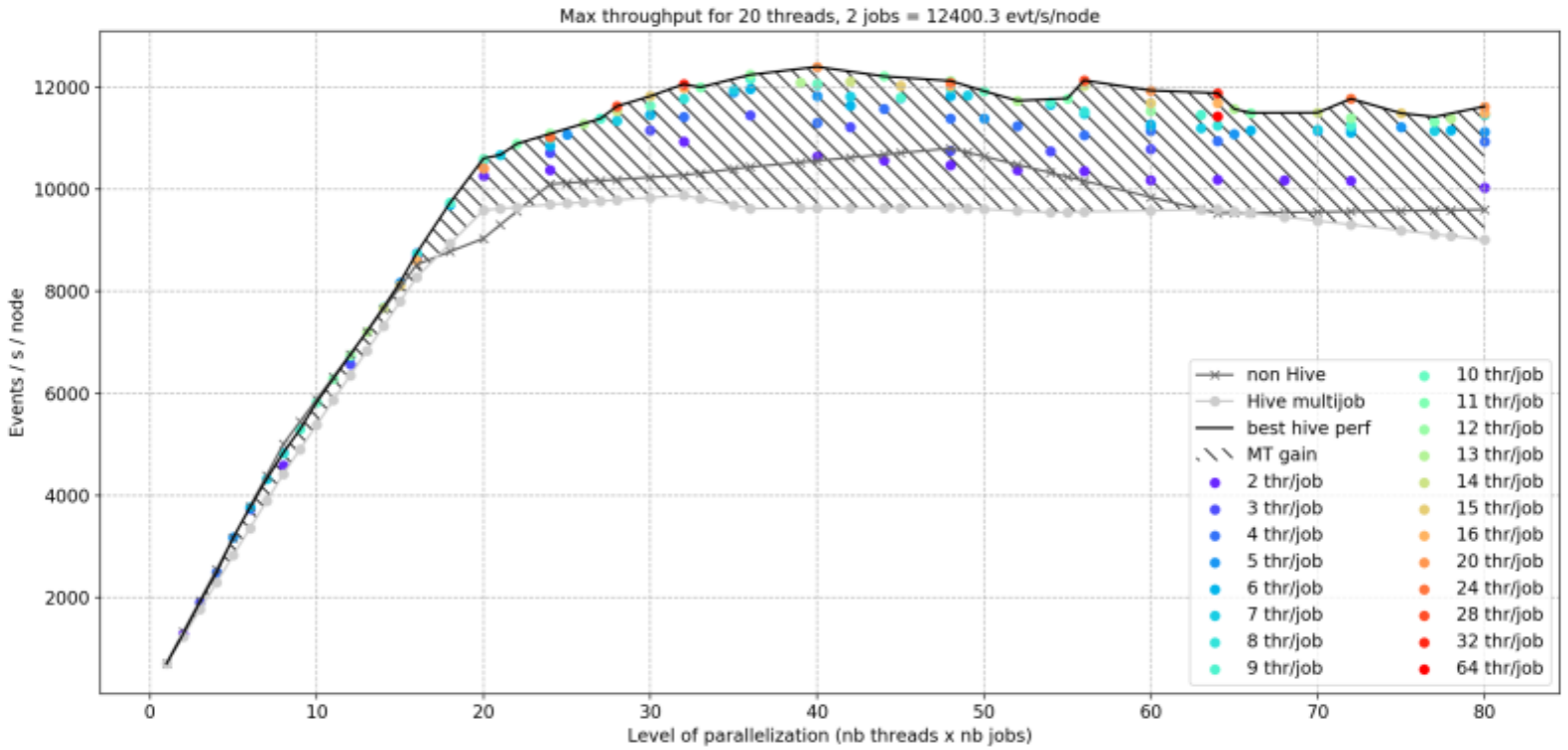
What we've achieved so far

- 2016 : first thread safe code in LHCb
- May 2017 : first benchmark with multithreading
  - 500 evts/s/node
- November 2017 : scalable code
  - 3000 evts/s/node
- Feb 2018 : realistic reconstruction with cuts
  - 12400 evts/s/node
- June 2018 : latest improvements
  - 14800 evts/s/node


3000
2500
2000
1500
1000
500
Events / s
0
10
20
30
40
50
60
Level of parallelization (nb threads x nb jobs)
non Hive
Hive MJ
best perf
2 thr/job
4 thr/job
8 thr/job
12 thr/job
16 thr/job
20 thr/job
24 thr/job
28 thr/job
32 thr/job

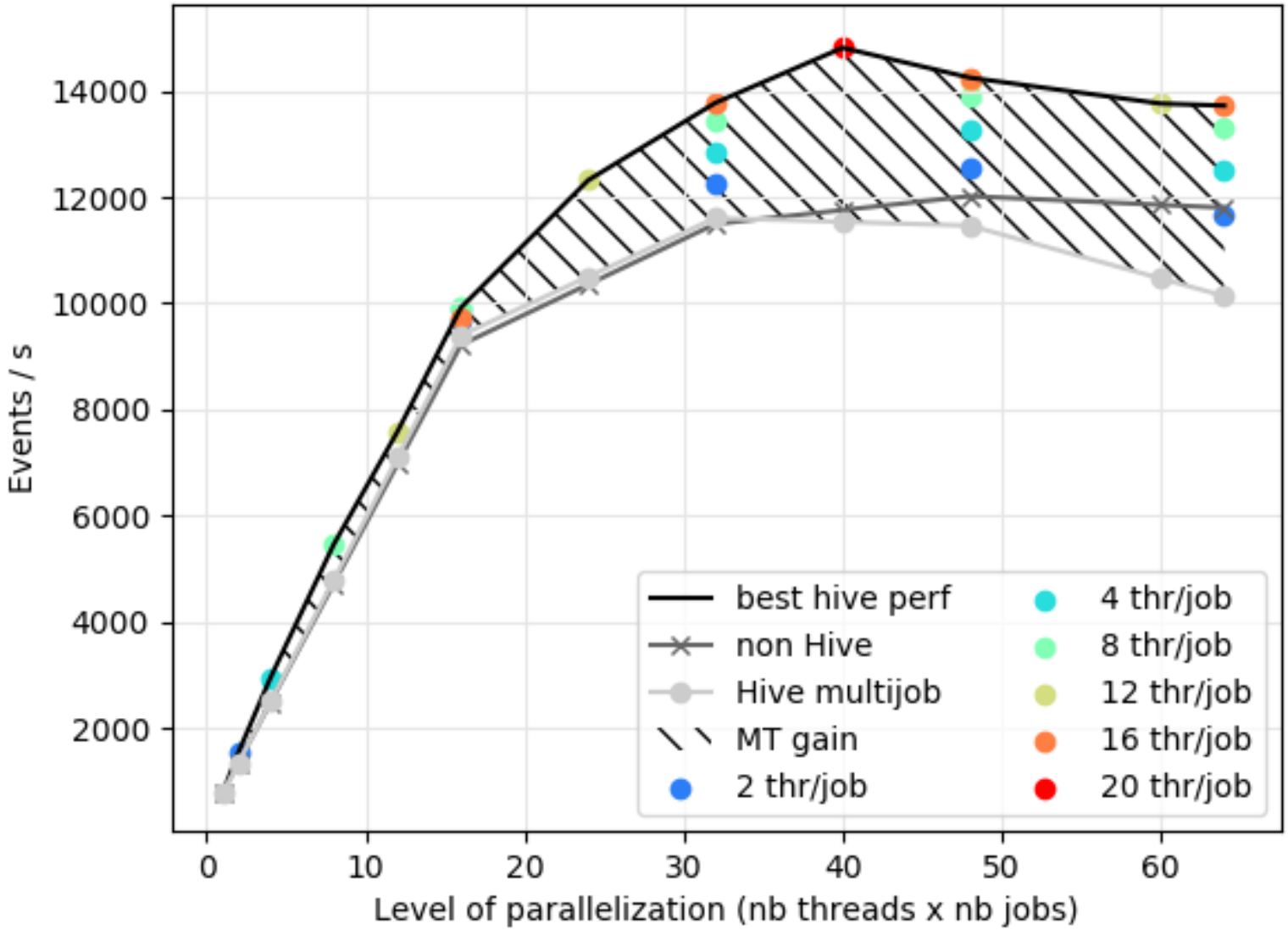
What we've achieved so far

- 2016 : first thread safe code in LHCb
- May 2017 : first benchmark with multithreading
  - 500 evts/s/node
- November 2017 : scalable code
  - 3000 evts/s/node
- Feb 2018 : realistic reconstruction with cuts
  - 12400 evts/s/node
- June 2018 : latest improvements
  - 14800 evts/s/node


Max throughput for 20 threads, 2 jobs = 12400.3 evt/s/node
Events / s / node
Level of parallelization (nb threads x nb jobs)
non Hive
Hive multijob
best hive perf
MT gain
2 thr/job
3 thr/job
4 thr/job
5 thr/job
6 thr/job
7 thr/job
8 thr/job
9 thr/job
10 thr/job
11 thr/job
12 thr/job
13 thr/job
14 thr/job
15 thr/job
16 thr/job
20 thr/job
24 thr/job
28 thr/job
32 thr/job
64 thr/job

What we've achieved so far

- 2016 : first thread safe code in LHCb
- May 2017 : first benchmark with multithreading
  - 500 evts/s/node
- November 2017 : scalable code
  - 3000 evts/s/node
- Feb 2018 : realistic reconstruction with cuts
  - 12400 evts/s/node
- June 2018 : latest improvements
  - 14800 evts/s/node


14000
12000
10000
8000
6000
4000
2000
Events / s
0
10
20
30
40
50
60
Level of parallelization (nb threads x nb jobs)
best hive perf
non Hive
Hive multijob
MT gain
2 thr/job
4 thr/job
8 thr/job
12 thr/job
16 thr/job
20 thr/job

Current state : halfway there

- Infrastructure is in place
- Training effort is starting to pay off
- We have a running HLT1 sequence
- Computing performance has improved a lot
  - we started around 3 KHz/node
  - we are 8 to 15 KHz/node depending on cuts

The road to 30MHz

- Ongoing effort needs to continue
  - getting more and more momentum thanks to knowledge spread
- HLT1 and core framework are not all of it
  - trigger selections need to be reworked[7]
  - HLT2 needs the same optimizations as HLT1
  - conditions[3], geometry[4, 6], scheduling, ...[8, 2]
  - the event model has to be reviewed
- All this is already ongoing
- Expected to pay off in some months

Conclusion

- LHCb's software is undergoing a major modernization to face Run3 challenges
- Best usage of modern hardware and parallelization are the main directions
- On top of proactive dissemination of knowledge
- Light is visible at the end of the tunnel
  - and is growing steadily !

References (I)

Johannes Albrecht, Ben Couturier, Christoph Hasse, and Sebastien Ponce.

New approaches for track reconstruction in lhcb's vertex locator.

Concezio Bozzi and Stefan Roiser.

Towards a computing model for the lhcb upgrade.

Marco Clemencic.

A git-based conditions database backend for lhcb.

Ben Couturier, Chris Burr, Lucia Grillo, Marco Clemencic, Markus Frank, and Silvia Borghi.

Perspectives for the migration of the lhcb geometry to the dd4hep toolkit.

References (II)

Ben Couturier and Maciej Pawel Szymanski.

Improvements to the lhcb software performance testing infrastructure using message queues and big data technologies.

Agnieszka Dziurda, Francesco Polci, Giulio Dujany, Jean-Francois Marchand, Lucia Grillo, and Paras Naik.

Lhcb full-detector real-time alignment and calibration: latest developments and perspectives.

Conor Fitzpatrick, Rosen Matev, and Sascha Stahl.

A 30mhz software trigger for the lhcb upgrade.

Aritz Brosa Iartza, Ben Couturier, Laura Promberger, Marco Clemencic, and Niko Neufeld.

Porting the lhcb stack from x86 (intel) to aarch64 (arm).

www.cern.ch