
CERN

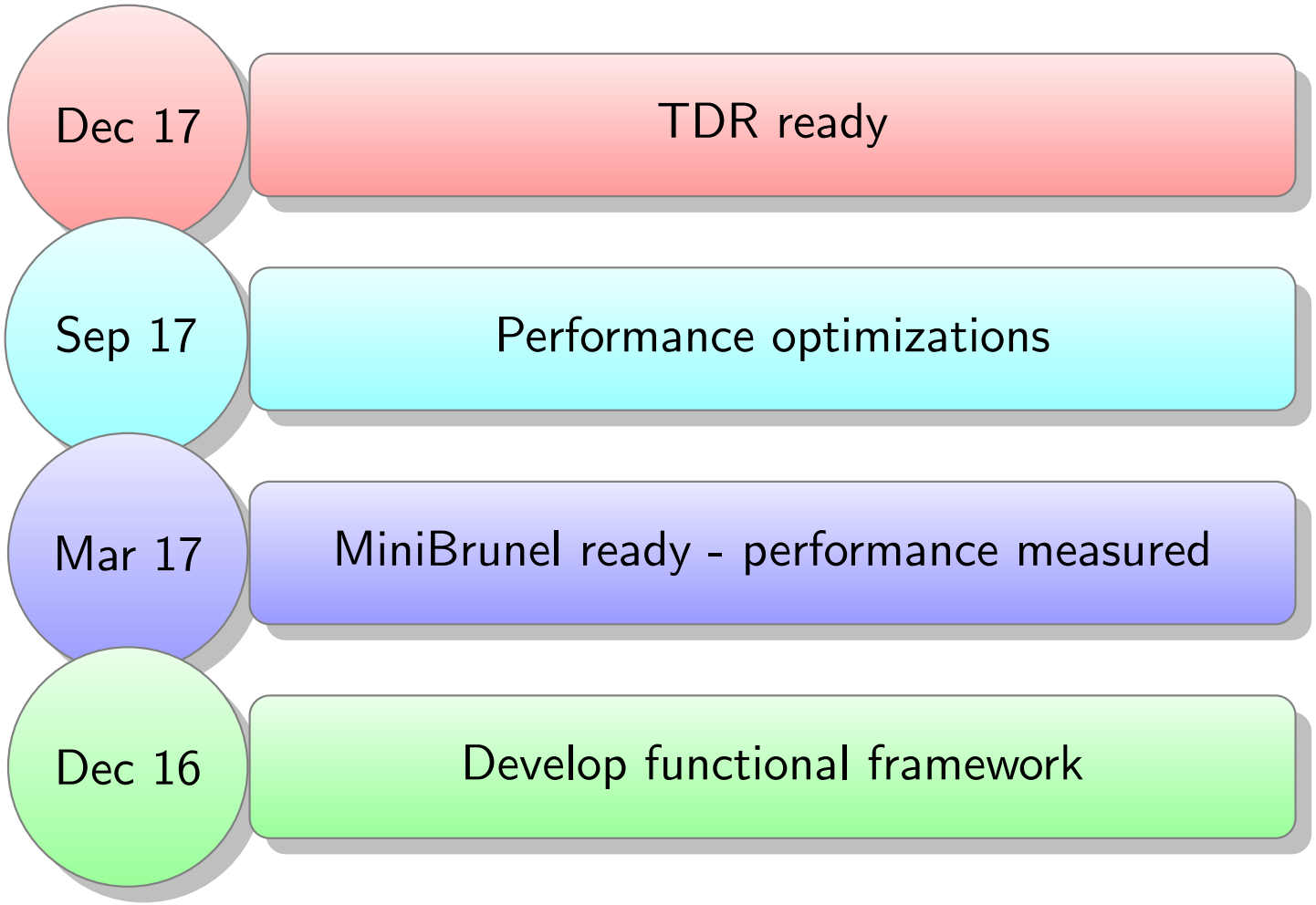
Software upgrade status

Sébastien Ponce
sebastien.ponce@cern.ch

Overall context

- Run3 needs have triggered the modernization of the LHCb software
  - in order to gain in efficiency
  - in order to allow the usage of new hardware
- Effort is targeted on the TDR
  - due by the end of 2017
  - where decisions have to be taken
- Internal milestone end of March
  - current status
  - evaluation of the strategy

The master plan



Areas concerned

- LHCb core framework
- Event model
- Conditions
- Detector Description
- Collaboration training

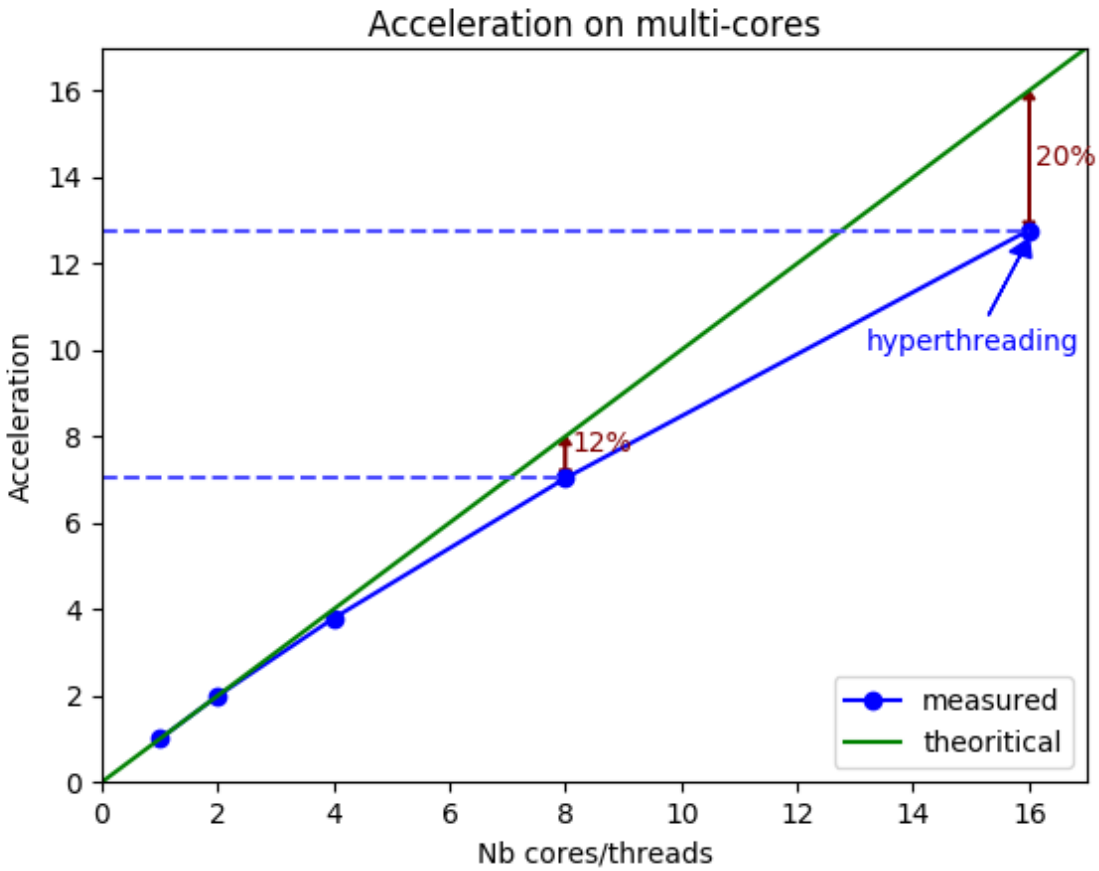
Outline

Outline

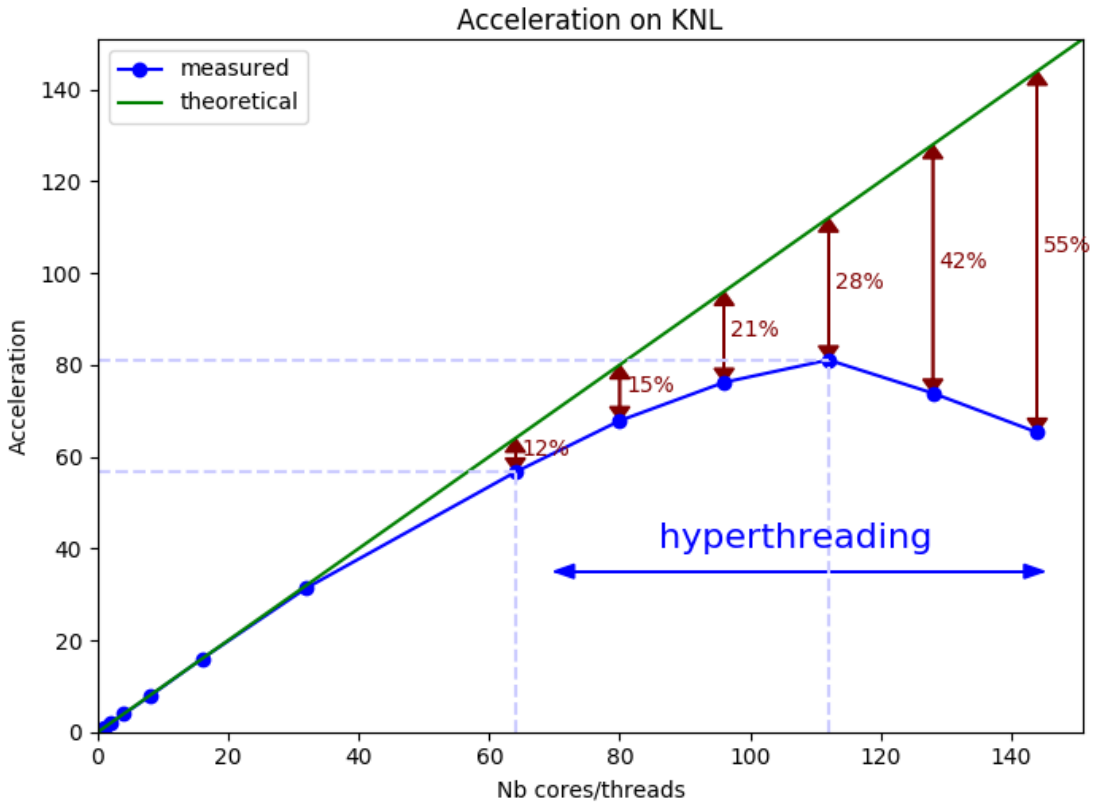
Framework - Where do we stand ?

- The functional framework is functional :-)
  - used in many algorithms (close to 100)
  - used in 2017 production
- "MiniBrunel"
  - MiniBrunel is not so mini !
    - includes Kalman filter, full Rich reconstruction
  - it has an HLT1 version
- Performance extensively measured
  - very good behavior of multithreading
  - coherent with upgrade performance document

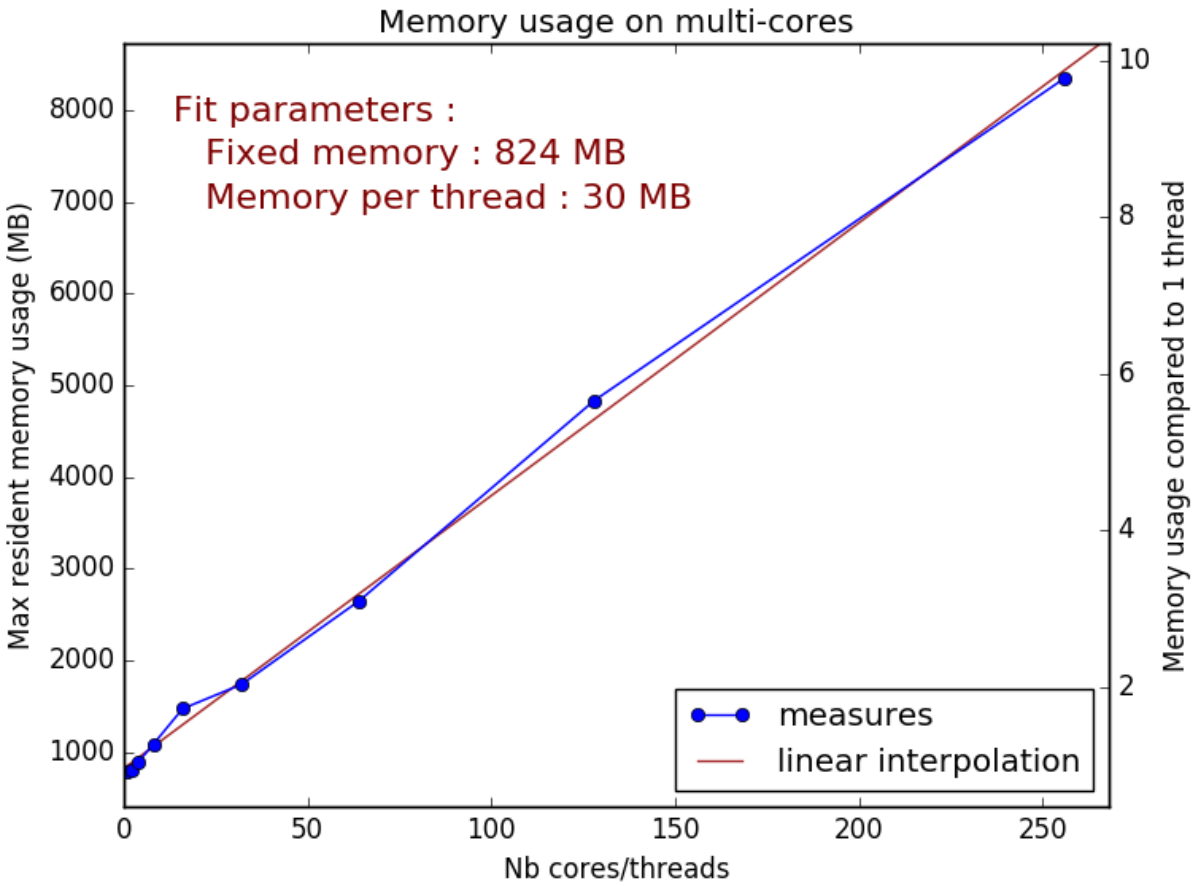
Acceleration with multithreading



Acceleration on KNL



Memory usage with multithreading



Event Model : what changes ?

- Change access pattern to the Event Store
  - write once, read-only after write
  - imposed by multi-threading
- Allow for object composition to compensate
- Introduce Structures of Arrays (SoA)
  - to boost gains due to auto vectorization
- Review usage of doubles
  - replace with floats when possible
- Test all this in MiniBrunel

Event Model - Where do we stand ?

- TES is now read-only
  - less impact than foreseen
  - most code can be adapted by splitting objects
- Composition can be achieved using “range v3”
  - often enough for transient data
  - used successfully to port the RICH code
- SoA components have been developed
  - ready to be tested
  - starting to measure benefits
- Switching double to floats has been tested
  - Vectorized Kalman filter goes 2x faster

Example of SoA on PixelTracking

```
const PrPixelHit* bestHit(const PrPixelModuleHits& modulehits, ...)
class PrPixelModuleHits final {
  std::vector<PrPixelHit> m_hits;
};
class PrPixelHit final {
  float m_x;
  float m_y;
  float m_z;
};
```

	before SOA	after SOA
Function / Call Stack	PrPT_bestHit	PrPT_bestHit
Clockticks	46958000000	26652000000
Instructions Retired	25760000000	24952000000
CPI Rate	1.8229	1.06813
MEM_LOAD_UOPS_RETIRED.L3_MISS_PS	94002820	0
MEM_LOAD_UOPS_L3_MISS_RETIRED.LOCA	90002700	0

In this test, SOA was crafted manually

SOAContainer & SOAView

```
// AOS - style object
struct Hit {
  float m_x;
  float x() const noexcept { return m_x; }
};

// SOA - style
struct HitFields {  // fields defined as types
  typedef struct : public SOATypelist::wrap_type<float> {} f_x;
}
// Skin decorating HitFields
template ... struct HitSkin : ... , HitFields {
  auto & x() const noexcept { return this->template get<f_x> (); }
}
SOAContainer <std::vector, HitSkin, HitFields::f_x> hits;
hits.reserve(...);
hits.emplace_back(...);
```

Detector Description

Current implementation

- is not thread safe
- is not maintained

Plans

- Investigate DD4Hep as replacement
- Use a minimal geometry
  - to speed up tracking

DetDesc - Where do we stand ?

- Full geometry has been converted to DD4Hep
  - validation is ongoing
- The minimal geometry has been defined
  - is default in MiniBrunel
- Efficiency of the code has been reviewed
  - not optimal, opportunity of optimizations

MiniBrunel's flops consumers

Self	Called	Function
129 441 204	7 191 178	ROOT::Math::Transform3D::operator()(ROOT::M...
118 452 234	2 374 308	PrPixelTracking::bestHit(PrPixelModuleHits con...
45 316 280	9 063 256	ROOT::Math::Cartesian3D<double>::Mag2() c...
33 944 515	6 788 903	double ROOT::Math::DisplacementVector3D<R...
26 288 988	1 314 851	(anonymous namespace)::xPointParameters(L...
24 397 215	1 626 481	ROOT::Math::Transform3D::operator()(ROOT::M...
21 904 575	7 301 525	ROOT::Math::Cartesian3D<double>::Scale(do...

Event Type	Incl.
F32 op	153 655 462
F64 op	630 742 861
SIMD 128b FP op	22 543 928
SIMD 256b FP op	25 380 052
SIMD FP op	47 923 980
Scalar FP op	736 474 343
FP op (FLOP)	784 398 323

- A lot of flops in the geometry
- Not much of vectorization
- Mostly coming from `isInside`
- Bounding boxes in global coordinates would help

Conditions - The goals

- Adapt our conditions' interfaces to multithreaded environment
  - multiple concurrent events
  - may not have same conditions
- Change our transient representation
  - triggered by the move to DD4Hep
- Change our persistent representation
  - from XML to something simpler to parse
- Change our tools to manage condition files
  - from COOL/Coral to something simpler and maintained

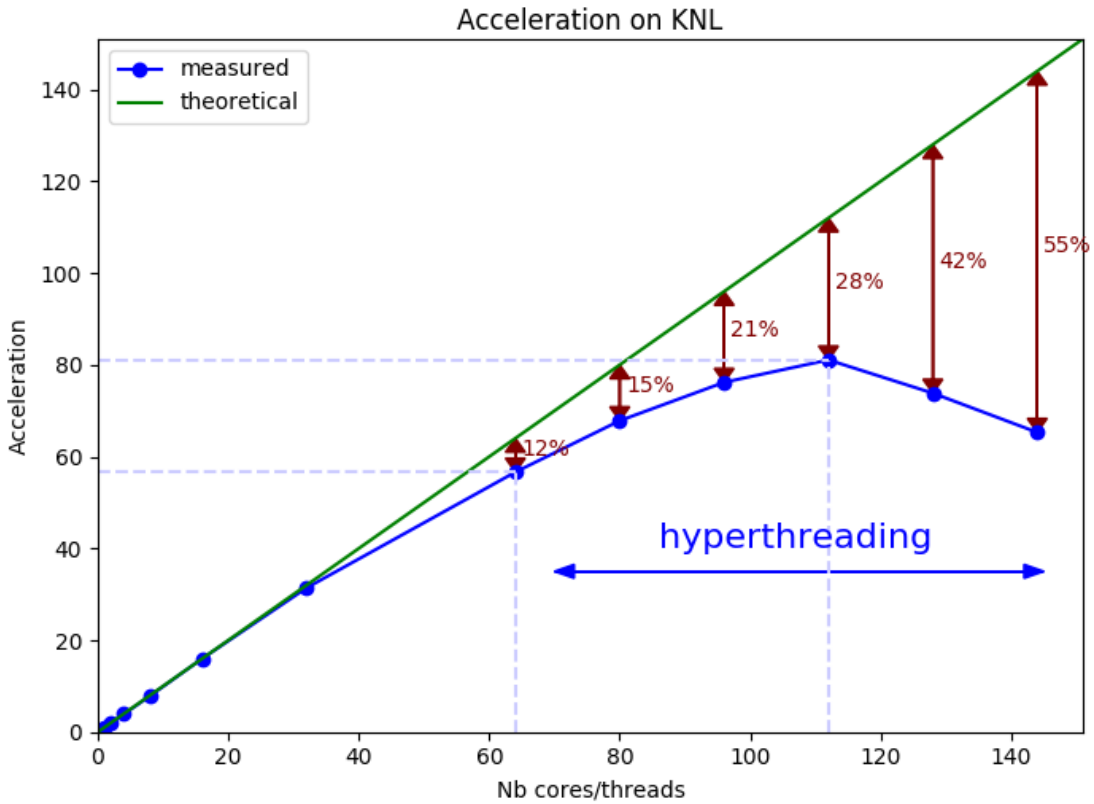
Conditions - Where do we stand ?

- A prototype of thread safe conditions have been proposed for Gaudi
  - under discussion at Gaudi level
  - looks promising from LHCb point of view
- Conditions Management switching to git
  - faster, smaller, easier than COOL/Coral
  - allows to drop a lot of code
  - ready to be used, being commissionned for 2017 run
- Changing file format will be easier once geometry is separated

Optimization Status

- Running on multiple/many cores validated
- Detailed timing of MiniBrunel
  - efficiency timing of HTL1
  - impact of O2/O3/Ofast/sse/avx/avx2
  - per algorithm/function timing
- Vectorization studied in details
  - where is code vectorized/not vectorized
  - where do we spend scalar flops ?
- Cache efficiency and mispredictions
  - Tools are in place, first tests done
- Studies on algorithm efficiencies
  - e.g. efficiency of cuts

Acceleration on KNL



Optimization Status

- Running on multiple/many cores validated
- Detailed timing of MiniBrunel
  - efficiency timing of HTL1
  - impact of O2/O3/Ofast/sse/avx/avx2
  - per algorithm/function timing
- Vectorization studied in details
  - where is code vectorized/not vectorized
  - where do we spend scalar flops ?
- Cache efficiency and mispredictions
  - Tools are in place, first tests done
- Studies on algorithm efficiencies
  - e.g. efficiency of cuts

MiniBrunel time distribution

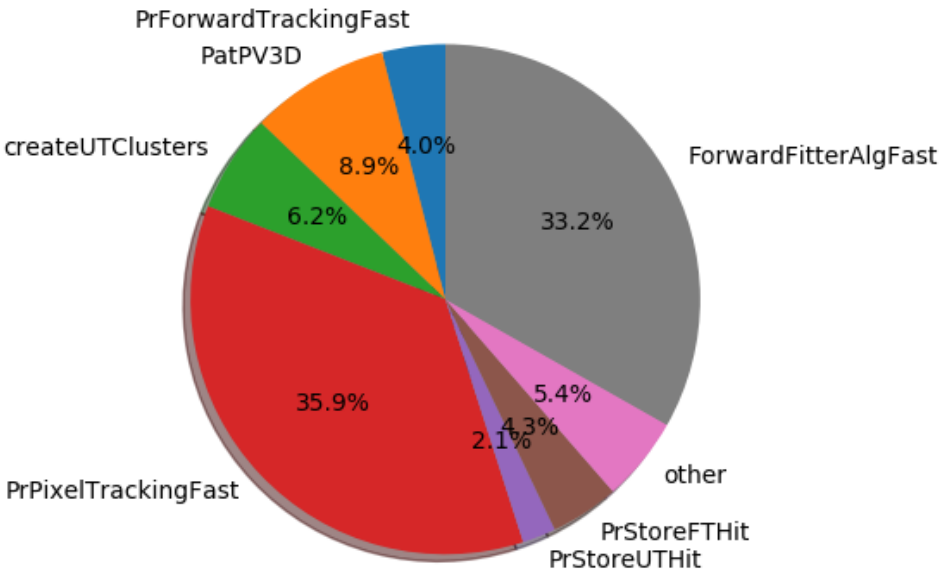


Figure: HLT1 on Minimum bias

MiniBrunel time distribution

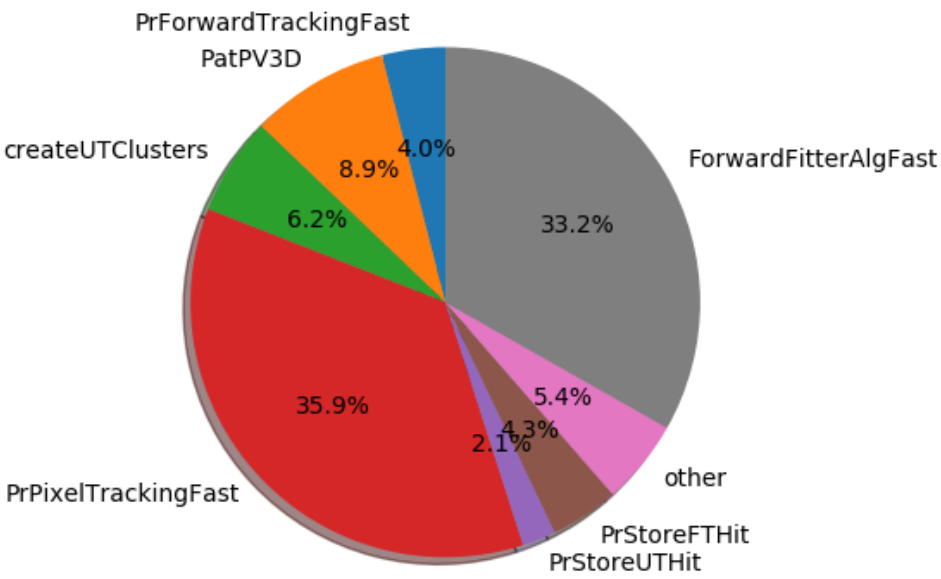


Figure: HLT1 on Minimum bias

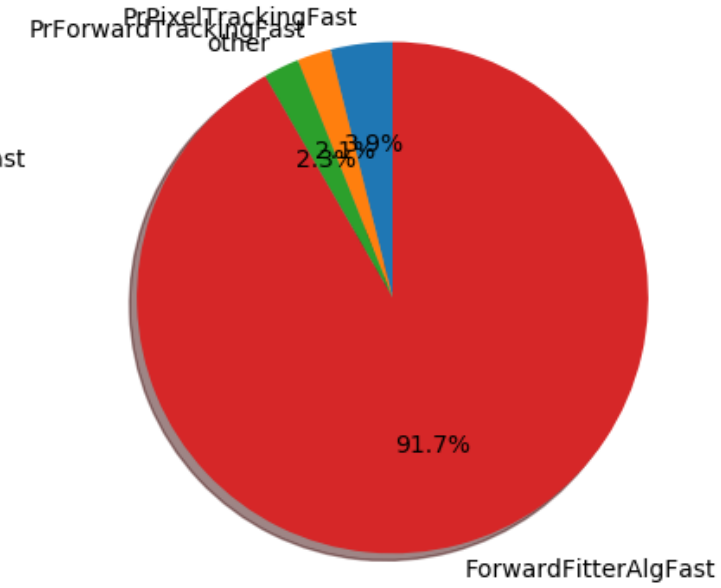
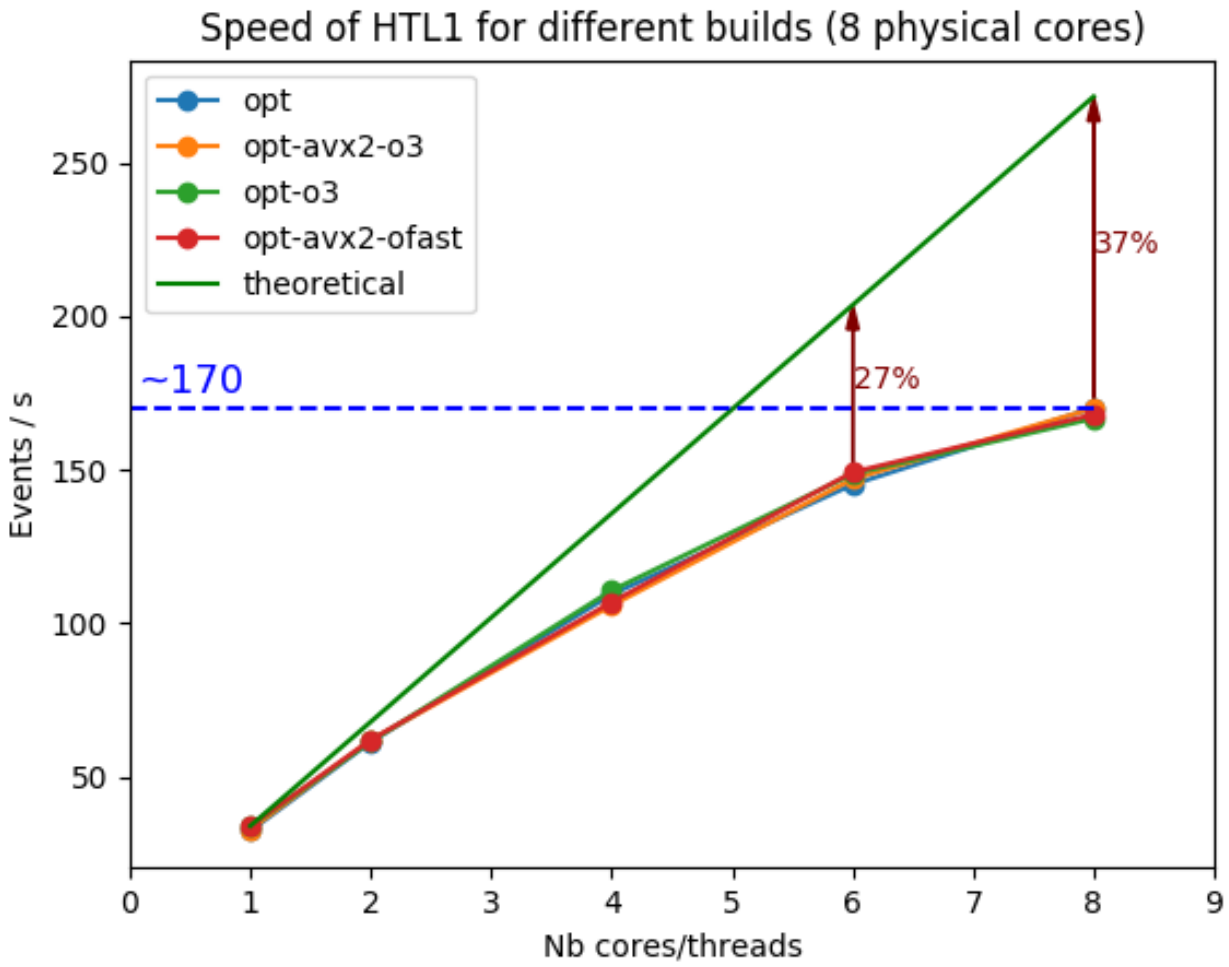


Figure: HLT1 on signal

Impact to compiler optimizations



Optimization Status

- Running on multiple/many cores validated
- Detailed timing of MiniBrunel
  - efficiency timing of HTL1
  - impact of O2/O3/Ofast/sse/avx/avx2
  - per algorithm/function timing
- Vectorization studied in details
  - where is code vectorized/not vectorized
  - where do we spend scalar flops ?
- Cache efficiency and mispredictions
  - Tools are in place, first tests done
- Studies on algorithm efficiencies
  - e.g. efficiency of cuts

Flops usage

Extension of callgrind allowing to count flops

- and differentiate scalar from SIMD
- but also different vector widths
- and floats from doubles

Self	Called	Function
129 441 204	7 191 178	ROOT::Math::Transform3D::operator()(ROOT::M...
118 452 234	2 374 308	PrPixelTracking::bestHit(PrPixelModuleHits con...
45 316 280	9 063 256	ROOT::Math::Cartesian3D<double>::Mag2() c...
33 944 515	6 788 903	double ROOT::Math::DisplacementVector3D<R...
26 288 988	1 314 851	(anonymous namespace)::xPointParameters(L...
24 397 215	1 626 481	ROOT::Math::Transform3D::operator()(ROOT::M...
21 904 575	7 301 525	ROOT::Math::Cartesian3D<double>::Scale(do...

Event Type	Incl.
F32 op	153 655 462
F64 op	630 742 861
SIMD 128b FP op	22 543 928
SIMD 256b FP op	25 380 052
SIMD FP op	47 923 980
Scalar FP op	736 474 343
FP op (FLOP)	784 398 323

Self	Called	Function	Lo
12 631 168	35 884	LHCb::Math::avx::similarity_5_5(double const*,...	lib
8 049 080	17 498	LHCb::Math::avx::average(double const*, dou...	lib
5 752 656	2 876 328	double __vector(2) Eigen::internal::pmul<doub...	lib
4 625 032	2 312 516	double __vector(2) Eigen::internal::padd<doub...	lib
4 288 896	39 712	LHCb::Math::avx::filter(double*, double*, doubl...	lib
3 813 600	28 867	PrPixelTrack::fit()	lib
3 562 680	890 670	operator*(Vec4f const&, Vec4f const&)	lib

Event Type	Incl.
F32 op	4 137 226
F64 op	0
SIMD 128b FP op	3 813 600
SIMD 256b FP op	0
SIMD FP op	3 813 600
Scalar FP op	323 626
FP op (FLOP)	4 137 226

Optimization Status

- Running on multiple/many cores validated
- Detailed timing of MiniBrunel
  - efficiency timing of HTL1
  - impact of O2/O3/Ofast/sse/avx/avx2
  - per algorithm/function timing
- Vectorization studied in details
  - where is code vectorized/not vectorized
  - where do we spend scalar flops ?
- Cache efficiency and mispredictions
  - Tools are in place, first tests done
- Studies on algorithm efficiencies
  - e.g. efficiency of cuts

Example of SoA on PixelTracking

```
const PrPixelHit* bestHit(const PrPixelModuleHits& modulehits, ...)
class PrPixelModuleHits final {
  std::vector<PrPixelHit> m_hits;
};
class PrPixelHit final {
  float m_x;
  float m_y;
  float m_z;
};
```

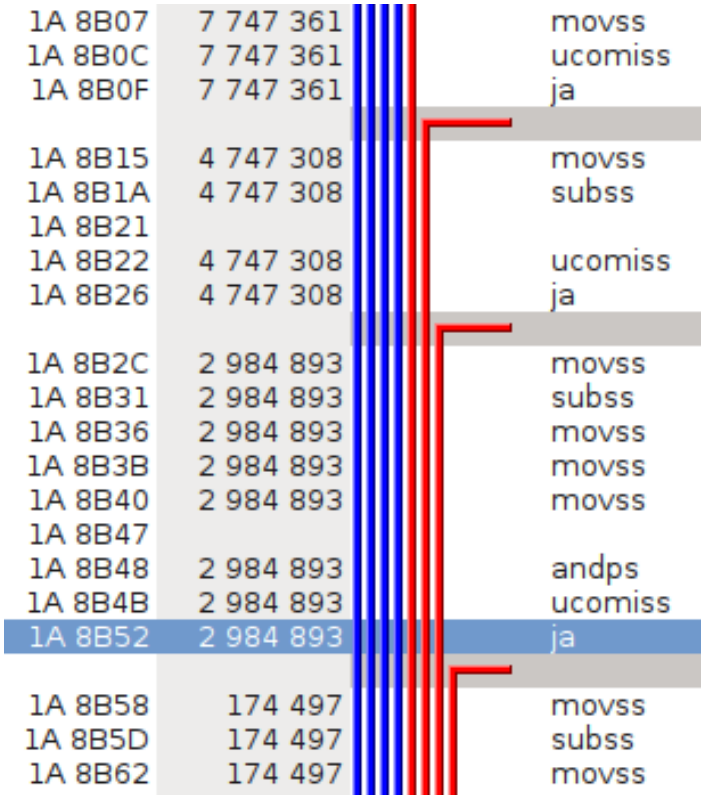
	before SOA	after SOA
Function / Call Stack	PrPT_bestHit	PrPT_bestHit
Clockticks	46958000000	26652000000
Instructions Retired	25760000000	24952000000
CPI Rate	1.8229	1.06813
MEM_LOAD_UOPS_RETIRED.L3_MISS_PS	94002820	0
MEM_LOAD_UOPS_L3_MISS_RETIRED.LOCA	90002700	0

In this test, SOA was crafted manually

Optimization Status

- Running on multiple/many cores validated
- Detailed timing of MiniBrunel
  - efficiency timing of HTL1
  - impact of O2/O3/Ofast/sse/avx/avx2
  - per algorithm/function timing
- Vectorization studied in details
  - where is code vectorized/not vectorized
  - where do we spend scalar flops ?
- Cache efficiency and mispredictions
  - Tools are in place, first tests done
- Studies on algorithm efficiencies
  - e.g. efficiency of cuts

PrPixel Algo efficiency (1)



About ordering of the cuts...

```
// If x-position is above prediction + tolerance, keep loo
if (hit_x + xTol < xPred) continue;
// If x-position is below prediction - tolerance, stop the
if (hit_x - xTol > xPred) break;
const float dy = yPred - hit_y;
// Skip hits outside the y-position tolerance.
if (fabs(dy) > xTol) continue;
```

...are we optimal ?

PrPixel Algo efficiency (2)

```
const PrPixelHit *PrPixelTracking::bestHit() {
  // Do a binary search through the hits.
  unsigned int hit_start = ...
  // Find the hit that matches best.
  for (unsigned int i = hit_start; ...) { ... }
}
```

Where do we spend time ?

- 80% of instructions are in the loop
- but 60% of the time is spent in binary search

Seen thanks to VTune

Outline

Framework

Challenges

- merge back future work into master branch
  - aggressive strategy once 2017 production is branched
- adapt the framework to an online usage
  - started and ongoing
- adapt the framework to simulation usage
  - multi event algorithms

Risks

- ~~MiniBrunel not being representative~~
- personpower to port algorithms (in subsystems)

Event Model

Challenges

- Do we need transparent composition ?
  - for non transient data, the Packing step may save us
  - may be complex, not easy to use for the end user
- Validation of physics with floats
  - where can we use them ?
- Usage of structure of arrays

Risks

- Missing person power
  - has to come from the subsystems

DetDesc

Challenges

- Validation of DD4Hep
- Validation of minimal geometry for tracking
- Integration of DD4Hep in LHCb code

Risks

- Being stuck with unmaintained framework
  - mitigation : preload geometry before starting threads

Conditions

Challenges

- Implement new condition interface in Gaudi
- Adapt transient representation to DD4Hep
  - conditions and geometry are tightly linked

Risks

- Lack of thread safe interface
  - mitigation : keep our conditions stable within a file

Outline

Training - Why ? Who ? What ?

- We are changing considerably the framework
  - the language used ($C^{++}17$ now)
  - the common practices (e.g. TES, functional)
  - the tools around it (e.g. git)
- Everyone is impacted
  - many will have to convert code
  - others will write new algorithms
- We need a substantial training effort
  - on the languages and tools
  - on the best practices

Training - What was done so far

- Development kit
- Workshops and Hackathons
  - user oriented hackathons
  - extended C^{++}courses
  - tutorials on framework and tools
  - practical courses
    - converting code to new framework
    - vectorization
    - efficient cache usage (to come)

LHCb C^{++}courses

LHCb hackathons

And the legendary pasta

Complemented with home made cakes

Training - Effort need to continue

- improve development kit
- other hackathons
- more courses
- participate to intel workshop

Outline

Summary

- The framework is under control
- Event Model, Conditions and DetDesc are progressing well
  - but more effort will be needed
- Performance measurement is well advanced
  - bringing many opportunities of optimizations
  - that will be tested soon
- Training of subsystem developers is essential
  - as we need all of them to be involved
  - as most of the person power has to come from them

www.cern.ch