HEP-score benchmark updates: Gauss on ARM, Geant4 speedup

Andrea Valassi (CERN IT)

HEP-SCORE Task Force meeting, Wednesday 1st February 2023
https://indico.cern.ch/event/1209411

Thanks to V. Innocente for his LHCb simulation profiling and Geant4 studies and suggestions!
Thanks to D. Giordano, R. Sobie and the Benchmarking WG for the Workshop organization and benchmarking infrastructure!
Thanks to C. Bozzi, M. Cattaneo, M. Clemencic, G. Corti, B. Couturier, A. Davis, P. Ilten, M. Kreps, D. Popov, S. Ponce in LHCb!
Thanks to A. Sailer and EP-SFT for the new LCG stack on ARM!

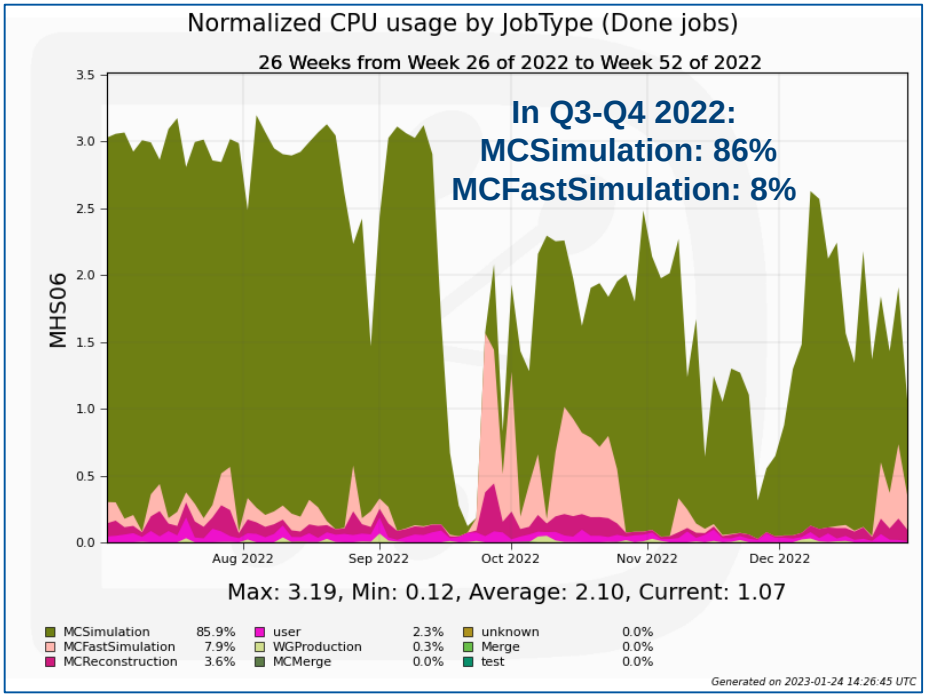
Outline – three areas of work

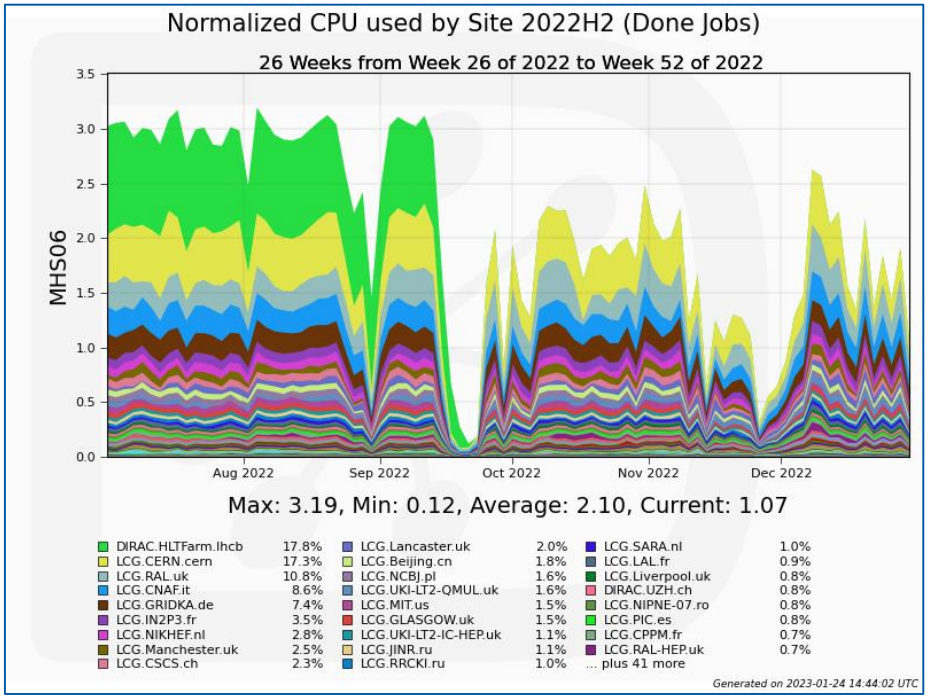
- 1. New LHCb container lhcb-sim-run3 v0.1 for x86 – done
- 2. Port LHCb simulation to ARM
  - Eventually a new lhcb-sim-run3 version for x86 and ARM – WIP
- 3. Speed up LHCb simulation with a Geant4 patch
  - Eventually a new lhcb-sim-run3 version for x86 and ARM – WIP

Reminder: LHCb Grid workloads

- MCSimulation (GEN-SIM) is by far the largest LHCb consumer of CPU for Grid jobs
  - MCFastSimulation (GEN-SIM) is a similar workload involving “ReDecay”
  - Around ~90% overall at the end of 2022, expect this throughout Run3

- GEN-SIM jobs: event generation (e.g. Pythia) + detector simulation (Geant4)
  - SIM is ≫ GEN: in other words, *almost all of LHCb Grid CPU is Geant4*
  - Today, single-threaded, single-process application, Gauss (will change)





1. New lhcb-sim-run3 workload v0.1 for x86

- Uses latest SIM10 software stack for Run3, as agreed with LHCb in Jan 2023
  - See the benchmark script and a typical logfile for details
  - Replaces previous "lhcb-gen-sim-2021", but *only relatively minor differences*
    - As before: single-threaded and single-process software
    - New: *Gauss v56r2 on LCG101_LHCB_7* (was Gauss v55r1 on LCG100_LHCB_5)
    - New: *x86_64_v2-centos7-gcc11-opt* (was x86_64-centos7-gcc9-opt)
    - As before: GEN is Pythia 8.244, inclusive B-physics events (options/10000000.py)
    - ~As before: minimum bias generated for each event (no I/O), pileup nu=3.2 (was 3.8)
    - As before: no spillover from adjacent beam crossings (off-time pileup: prev/next event)
    - ~As before: *SIM is Geant4-10-06-patch02* from May 2020 (10r6p2t6, was 106r2p4)
    - New: CondDB sim-20221220-vc-mu100 (was sim-20210617-vc-md100)
    - New: AppConfig v3r412 (was v3r404), DecFiles v32r2 (v31r7)
  - Default setup processes 5 reproducible events, ~10 minutes on a typical CPU

- The application runs GEN+SIM (starts from random seeds), separate scores exist
  - *The recommended SIM score uses SIM-only throughput for n-1 events*
  - Skip readback of magnetic field and other initialization tasks during the 1st event

- *Status: v0.1 exists and can be tested! But new versions will come (see next slides...)*

2. Port LHCb simulation to ARM

- *The Gauss application now builds and runs successfully on ARM!*
  - Note: ARM results are not bit-by-bit identical to x86, need statistical validation
- This uses a different software stack from that of lhcb-sim-run3 v0.1 on x86
  - The external stack is LCG102b_LHCB_7 (LCG101 was not built for ARM)
  - A few minor patches for ARM are needed in various LHCb packages
- *A new Gauss will be released on cvmfs with identical configurations for x86 and ARM*
  - Will be used (only) to build a new lhcb-sim-run3 container for benchmarking
  - Will not be used for production & will not need large-scale physics validation
    - LHCb will wait for LCG103 instead including a new version of ROOT
- Status: new stack is being built and undergoing functional tests in the LHCb nightlies
  - *A new lhcb-sim-run3 v0.x will be built for x86 and ARM when this is in /cvmfs*

3. Geant4 patch for LHCb SIM speedup (a)

- This is a follow-up to some email threads with Vincenzo Innocente in July 2022
  - See my SIM meeting talk in Aug and Vincenzo's talk at the Sep 2022 workshop
  - By profiling lhcb-gen-sim-2021, Vincenzo reported that a single Geant4 10.6 function *G4LogicalBorderSurface::GetSurface accounts for 40% of the CPU time*
  - He also noted that this function changed from Geant4 10.6 to 10.7 (std::vector to std::map) and suggested that *upgrading to 10.7 may speed up LHCb simulation*

- While foreseen for later in 2023, it was clear since August that upgrading LHCb SIM to Geant4 10.7 is unfeasible now as it requires validation with large event statistics

- In Jan 2023 I attempted an intermediate solution: keep using G4 10.6 as a baseline, but prepare *a Geant4 patch where only G4LogicalBorderSurface comes from 10.7*
  - Again this was possible thanks to Marco's detailed ARM build instructions
  - Luckily, a short Geant4 patch allowed Geant4 and Gauss to build successfully

- I did a simple test on a few events and the results are even better than expected!
  - *Physics results in the output log are bit-by-bit identical* with and without the patch
  - ***LHCb SIM throughput increases by a factor two (x2.0) for Run3 geometry***
  - G4LogicalBorderSurface::GetSurface disappears from the perf profile

Vincenzo's slides at the Sep 2022 workshop

LHCb Sim

```
40.03%  python  libG4geometry.so      [.] G4LogicalBorderSurface::GetSurface
 4.38%  python  libCLHEP-2.4.4.0.so   [.] CLHEP::RanluxEngine::flat
 2.13%  python  libCLHEP-2.4.4.0.so   [.] CLHEP::RanluxEngine::flatArray
 1.65%  python  libG4geometry.so      [.] G4Navigator::LocateGlobalPointAndSetup
 1.33%  python  libG4tracking.so      [.] G4SteppingManager::DefinePhysicalStepLength
 1.05%  python  libG4geometry.so      [.] G4VoxelNavigation::ComputeStep
 0.99%  python  libG4processes.so     [.] G4VEmProcess::PostStepGetPhysicalInteractionLength
 0.85%  python  libG4tracking.so      [.] G4SteppingManager::InvokePSDIP
 0.72%  python  libG4global.so        [.] G4PhysicsVector::Value
 0.69%  python  libG4processes.so     [.] G4VProcess::ResetNumberOfInteractionLengthLeft
 0.65%  python  libGaussTools.so      [.] virtual thunk to GiGaStepActionSequence::UserSteppingAction(G4Step const*)
 0.65%  python  libG4geometry.so      [.] G4SubtractionSolid::Inside
 0.62%  python  libG4processes.so     [.] G4UniversalFluctuation::SampleFluctuations
 0.59%  python  libDetDescLib.so      [.] LHCb::MagneticFieldGrid::fieldVectorLinearInterpolation
 0.52%  python  libG4tracking.so      [.] G4SteppingManager::Stepping
```

- Spending 40% of the time in these 4 lines of code (G4 10.6)
- code changed (from vector to map) in 10.7

```
104 G4LogicalBorderSurface*
105 G4LogicalBorderSurface::GetSurface(const G4VPhysicalVolume* vol1,
106                                    const G4VPhysicalVolume* vol2)
107 {
108   if (theBorderSurfaceTable != nullptr)
109   {
110     for(auto pos = theBorderSurfaceTable->cbegin();
111         pos != theBorderSurfaceTable->cend(); ++pos)
112     {
113       if( (*pos)->GetVolume1() == vol1 && (*pos)->GetVolume2() == vol2 )
114       { return *pos; }
115     }
116   }
117   return 0;
118 }
```

19/9/22 Vincenzo Innocente: HEPSpec perf 12

3. Geant4 patch for LHCb SIM speedup (b)

Before the patch (G4LogicalBorderSurface from G4 10.6)

SIM step for 300 events takes 13940 seconds

Perf profile for 5 events shows G4LogicalBorderSurface at the very top

# Overhead	Command	Shared Object	Symbol
25.88%	python	libG4geometry.so	[.] G4LogicalBorderSurface::GetSurface
6.66%	python	libCLHEP-2.4.5.1.so	[.] CLHEP::RanluxEngine::flat
2.24%	python	libG4processes.so	[.] G4ProductionCutsTable::ScanAndSetCouple
1.79%	python	libm-2.17.so	[.] __ieee754_log_avx
1.65%	python	libCLHEP-2.4.5.1.so	[.] CLHEP::RanluxEngine::flatArray
1.43%	python	libstdc++.so.6.0.29	[.] __cxxabiv1::__vmi_class_type_info::__do_dyncast
1.31%	python	libG4geometry.so	[.] G4Navigator::LocateGlobalPointAndSetup
1.25%	python	libGaussGeo.so	[.] GaussGeo::detElementByLVNameWithAlignment
1.21%	python	libxerces-c-3.2.so	[.] xercesc_3_2::DOMDeepNodeListImpl::nextMatchingElementAfter

After the patch (G4LogicalBorderSurface from G4 10.7)

SIM step for 300 events takes 6960 seconds (a factor x2.0 faster)

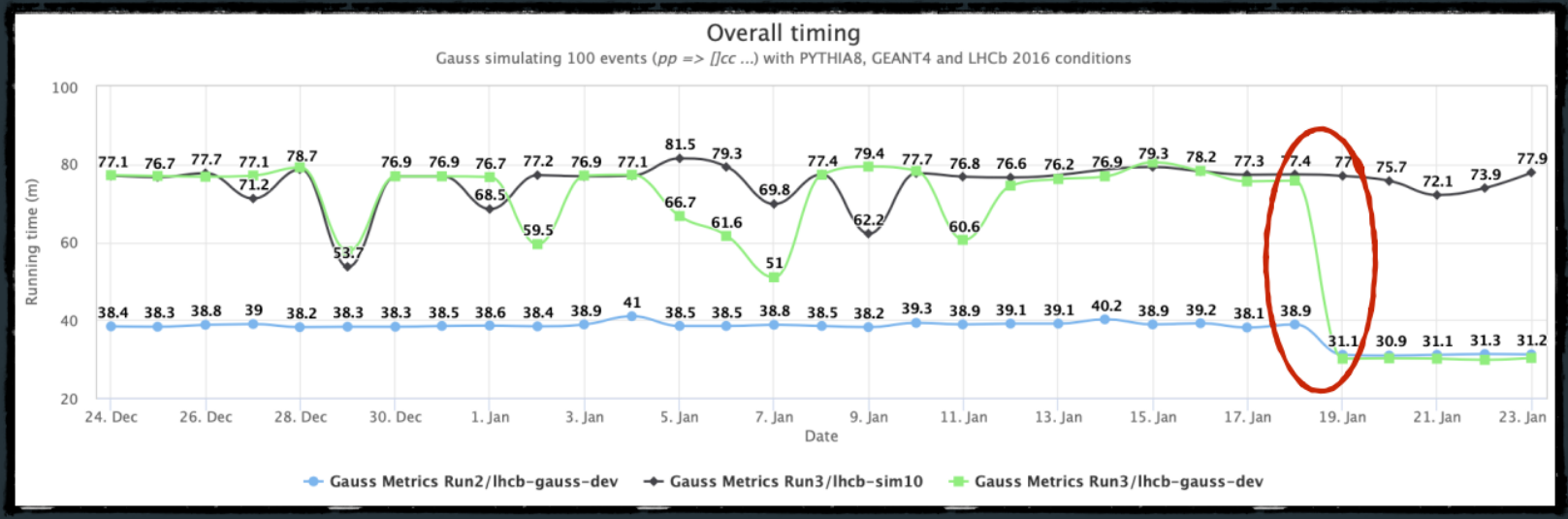
Perf profile for 5 events does not shows G4LogicalBorderSurface::GetSurface

# Overhead	Command	Shared Object	Symbol
9.16%	python	libCLHEP-2.4.5.1.so	[.] CLHEP::RanluxEngine::flat
3.10%	python	libG4processes.so	[.] G4ProductionCutsTable::ScanAndSetCouple
2.58%	python	libm-2.17.so	[.] __ieee754_log_avx
2.30%	python	libCLHEP-2.4.5.1.so	[.] CLHEP::RanluxEngine::flatArray
2.04%	python	libGaussGeo.so	[.] GaussGeo::detElementByLVNameWithAlignment
1.94%	python	libstdc++.so.6.0.29	[.] __cxxabiv1::__vmi_class_type_info::__do_dyncast
1.73%	python	libxerces-c-3.2.so	[.] xercesc_3_2::DOMDeepNodeListImpl::nextMatchingElementAfter

3. Geant4 patch for LHCb SIM speedup (c)

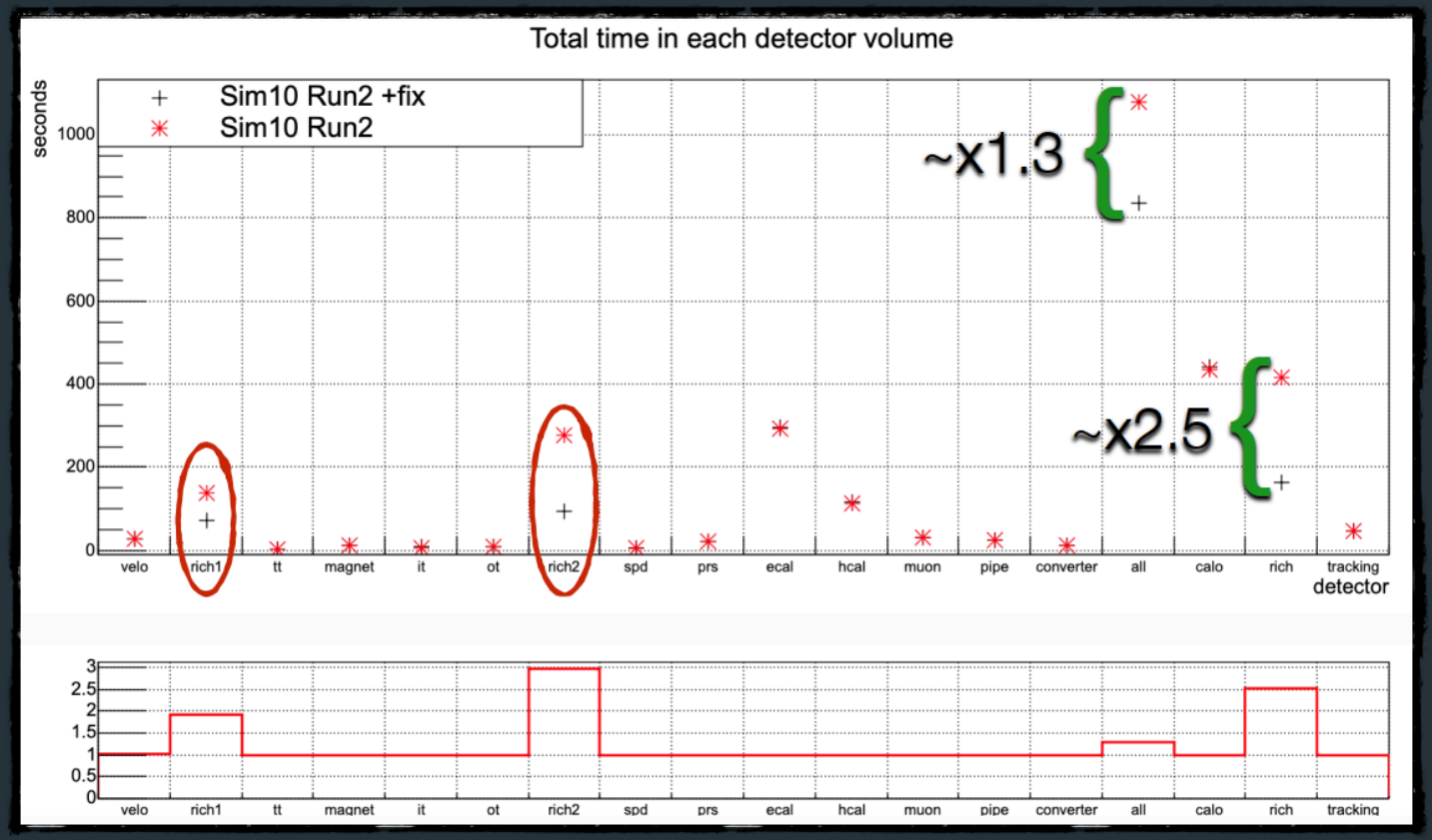
- Some additional comments
  - The LHCb RICH is the subdetector where the issue was coming from
  - The effect of G4LogicalBorderSurface::GetSurface in the LHCb simulation has increased a lot from Run2 Geometry to Run3 geometry (*Run3 RICH has a much more granular substructure with a much larger number of photodetectors*)
- Status: G4 patch is being built and undergoing functional tests in the LHCb nightlies
  - The plan is to prepare new release(s) to eventually use this in production
  - *A new lhcb-sim-run3 v0.x will be built for x86/ARM when a release is in /cvmfs*
- Lesson learnt IMO: reproducible experiment workloads containers in HEP-SCORE make it easier for people outside the collaboration to help improve the software!
  - Thanks to Domenico and the whole benchmarking team for the work on this!
  - Thanks again to Vincenzo for the analysis and the suggestions!
  - Thanks again to MarcoCl and Gloria for the Gauss build and runtime recipes!

Sim10 overall timing with the fix

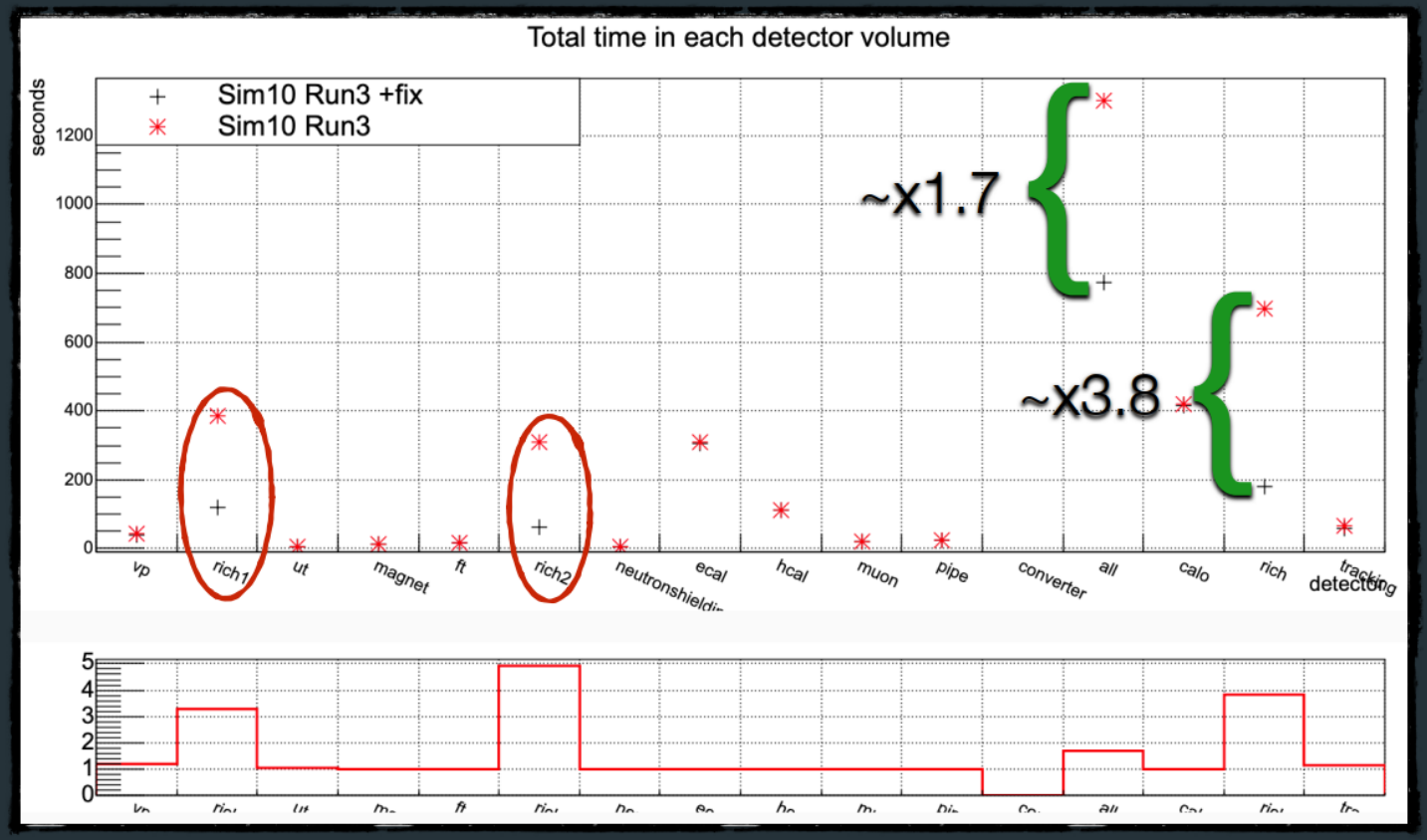


- Sim10 Run2 (+fix) vs. Sim10 Run3 (+fix) vs. Sim10 Run3 (no fix)

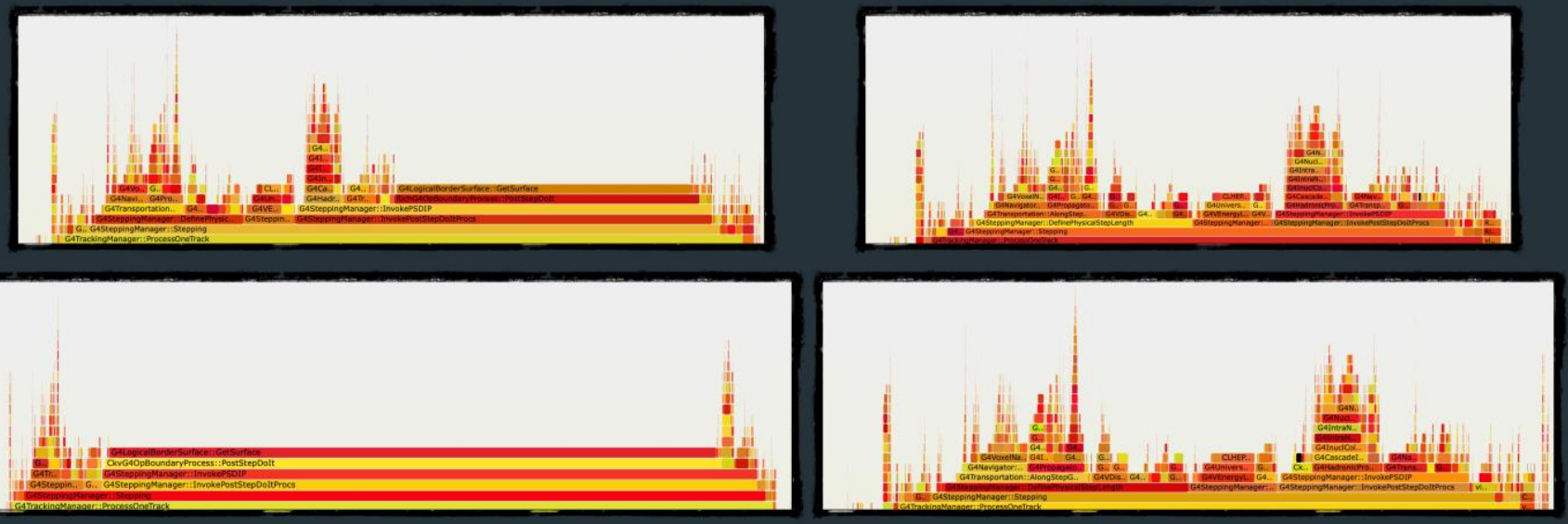
G4 detailed timing: Sim10 Run2 with the fix



G4 detailed timing: Sim10 Run3 with the fix



Sim10 stack traces sampling comparison



Sim10 Run2	Sim10 Run2 +fix	Sim10 Run3	Sim10 Run3 +fix
~40%	~0.3%	~76%	~0.7%

What is the magical fix?

- G4LogicalBorderSurface::GetSurface
- Surfaces stored in: `G4LogicalBorderSurfaceTable`
  - v10.6.x: `std::vector` -> loop over dynamic array to find element
  - v10.7.x: `std::map` -> lookup element: average O(1)
- Surfaces in RICH Run2 ~500, Run3 ~3000
  - Even for Run2 just this changed yielded almost 20% speedup
- Know your data structures!