Geant4 in Atlas

John Apostolakis

On behalf of the Atlas Simulation Team

26th September 2019

Based on material prepared by Marilena Bandieramonte

For Geant4 Technical Forum of 18th January 2019 and
29th March 2019

Current production

- MC production is continuing with **no major changes** from the simulation side:
  - Default production release uses **G4 10.1 patch03**, CLHEP 2.2, 64-bit, gcc 4.9, SLC6, C++14. Some samples produced with gcc6.2.
- Compiling **G4** as part of our nightly builds
  - Significant number of updates to ATLAS user code (geometry and detector response), including several speed ups.

- Still running tails of (much) older production campaigns: (G4 9.4+patches, 9.6p3)

- **Changes in 2019:**
  - Moved Run 2 development branch to use Geant4 10.1.patch03.atlas07 (G4 Solid updates – 4% speedup).

  - Run 3 development has been based on Geant4 10.4.patch03.atlas01

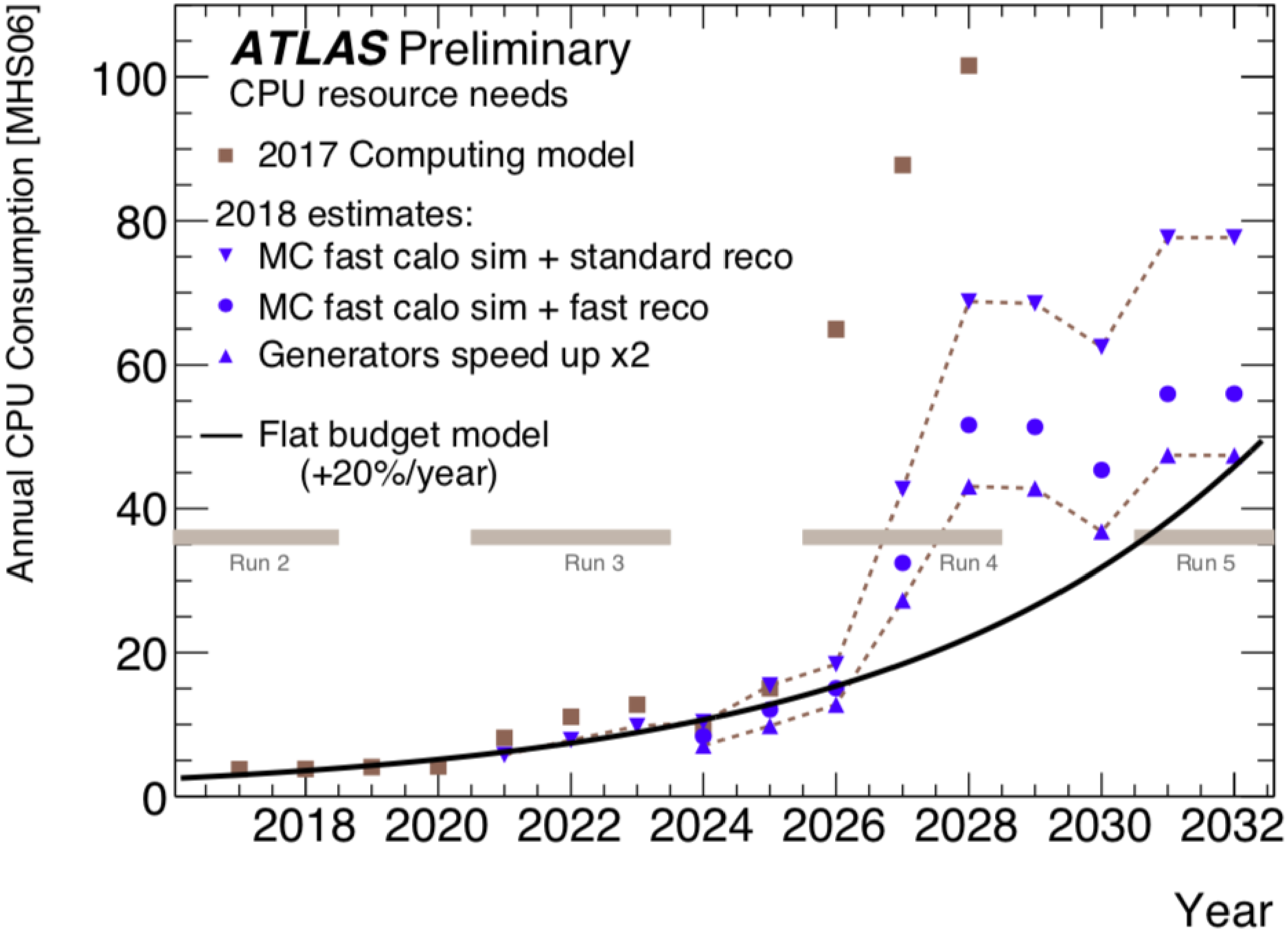
Production plans

- **Upcoming changes:**

  - Early testing of Geant4.10.5: We built AthSimulation with Geant4.10.5. It will be used for testing purposes

- **The next MC** campaign (preparing for LHC Run 3) will use Geant4 10.4.patch03.atlas01 or later.
  - we are testing Geant4 10.5 and will test Geant4 10.6 (when available). We will decide in mid-2020 on the G4 version to use for MC to match 2021 data, with the possibility of updating the G4 version again for MC produced in 2022.

Projection of CPU needs

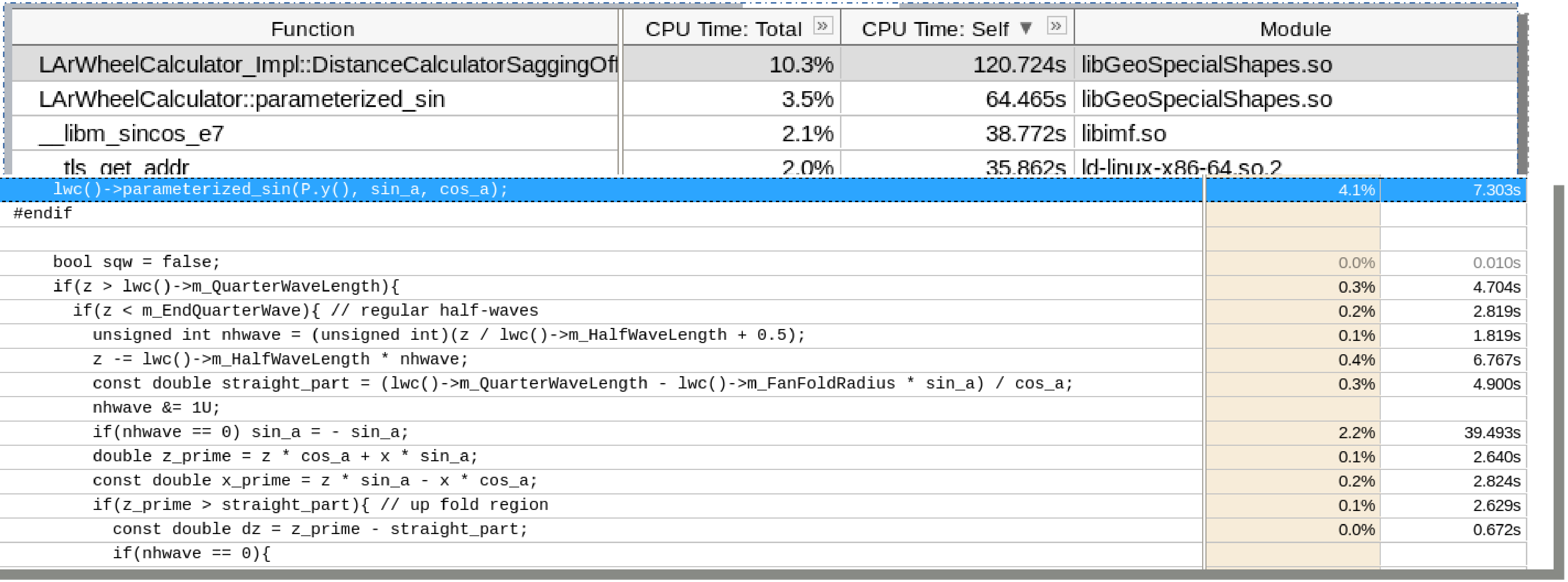


- CPU consumption will increase dramatically for HL-LHC.
- Most of simulation will rely on FastCaloSim, but full Geant4 sim will be heavily used regardless (e.g. 25% of all CPU time).
- Any performance optimizations of ATLAS simulation have a big impact on the overall picture.

Plot from Davide Costanzo presented at:

Code optimization and profiling with Intel tools

Intel’s VTune profiling tool can be easily used to thoroughly profile Athena.



Function	CPU Time: Total	CPU Time: Self	Module
LArWheelCalculator_Impl::DistanceCalculatorSaggingOff	10.3%	120.724s	libGeoSpecialShapes.so
LArWheelCalculator::parameterized_sin	3.5%	64.465s	libGeoSpecialShapes.so
__libm_sincos_e7	2.1%	38.772s	libimf.so
tls_get_addr	2.0%	35.862s	ld-linux-x86-64.so.2

```
    lwc()->parameterized_sin(P.y(), sin_a, cos_a);                                      4.1%     7.303s
  #endif

    bool sqw = false;                                                                   0.0%     0.010s
    if(z > lwc()->m_QuarterWaveLength){                                                 0.3%     4.704s
      if(z < m_EndQuarterWave){ // regular half-waves                                   0.2%     2.819s
        unsigned int nhwave = (unsigned int)(z / lwc()->m_HalfWaveLength + 0.5);        0.1%     1.819s
        z -= lwc()->m_HalfWaveLength * nhwave;                                          0.4%     6.767s
        const double straight_part = (lwc()->m_QuarterWaveLength - lwc()->m_FanFoldRadius * sin_a) / cos_a;   0.3%     4.900s
        nhwave &= 1U;
        if(nhwave == 0) sin_a = - sin_a;                                                2.2%    39.493s
        double z_prime = z * cos_a + x * sin_a;                                         0.1%     2.640s
        const double x_prime = z * sin_a - x * cos_a;                                   0.2%     2.824s
        if(z_prime > straight_part){ // up fold region                                  0.1%     2.629s
          const double dz = z_prime - straight_part;                                    0.0%     0.672s
          if(nhwave == 0){
```

parameterized_sin function calculates cosine as: `cos_a = sqrt(1. - sin_a*sin_a);`
That’s very slow and it can be replaced with a parameterized cos calculation.

}

1-2% speedup

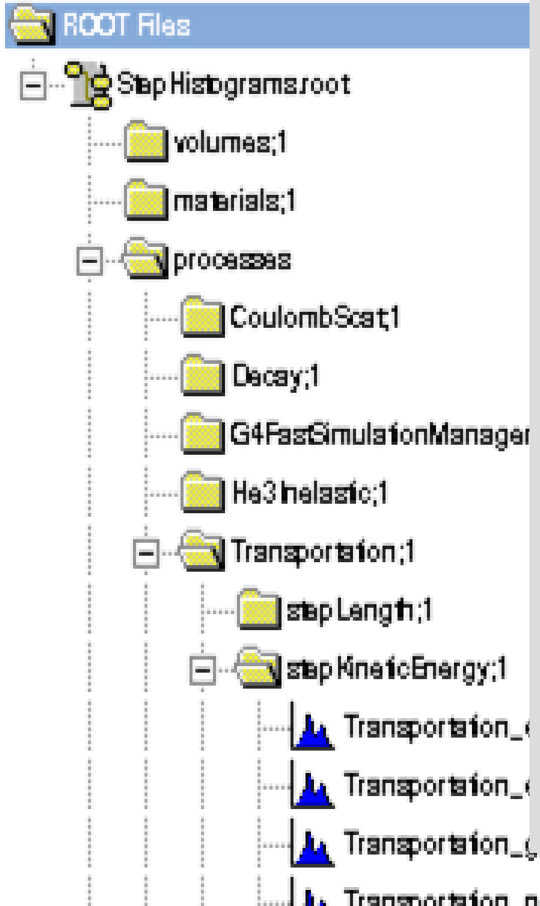
Geant4 debugging tools

All debugging plots are relatively automatically assembled into a web-page.

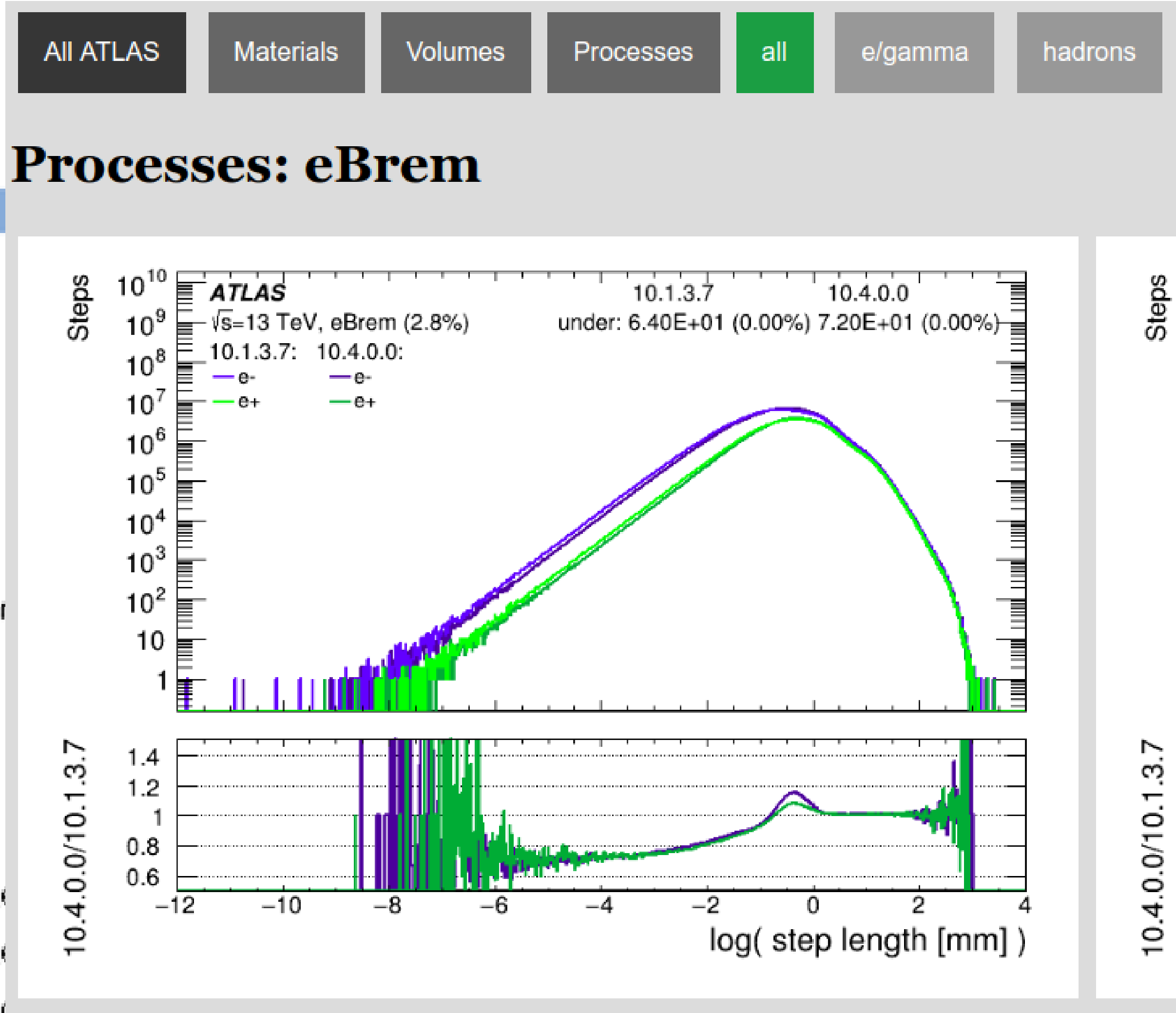
O(2000) plots, e.g.:

Tool that plots histograms of various step-related quantities:

- step length,
- step energy deposit,
- step kinetic energy,
- step position,
- created secondaries,
- ... As a function of:



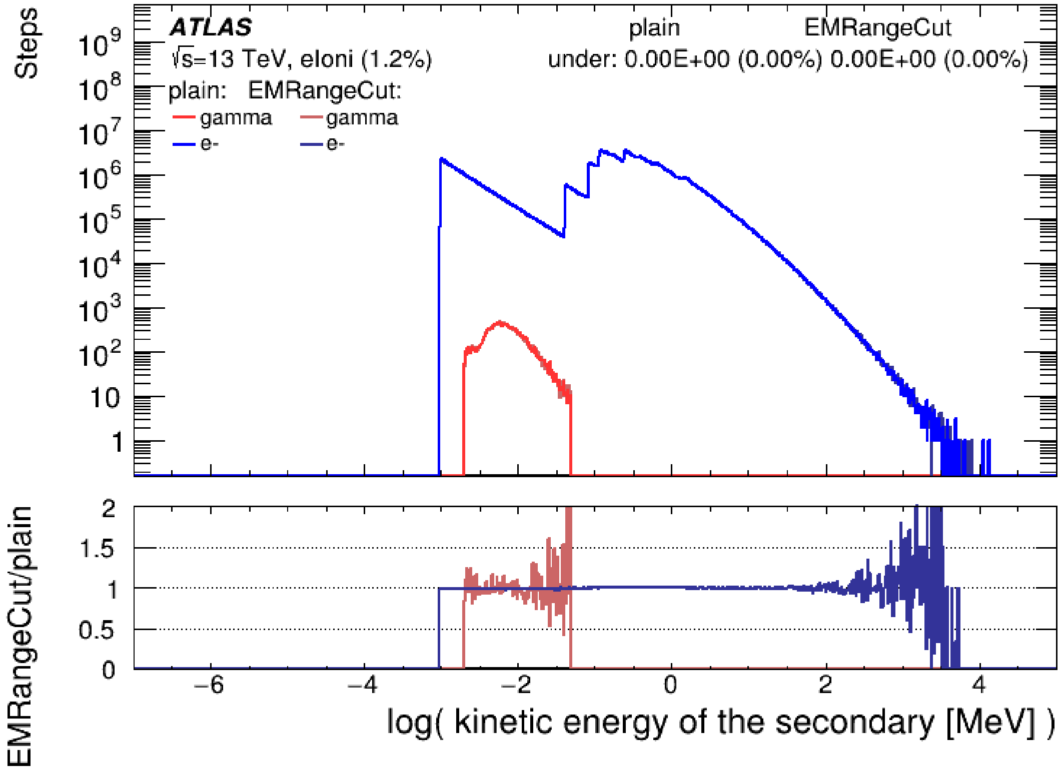
ROOT Files
- StepHistograms.root
  - volumes;1
  - materials;1
  - processes
    - CoulombScat;1
    - Decay;1
    - G4FastSimulationManager
    - He3Inelastic;1
    - Transportation;1
      - stepLength;1
      - stepKineticEnergy;1
        - Transportation_
        - Transportation_
        - Transportation_
        - Transportation_neutron_pro_stepKineticEnergy;1



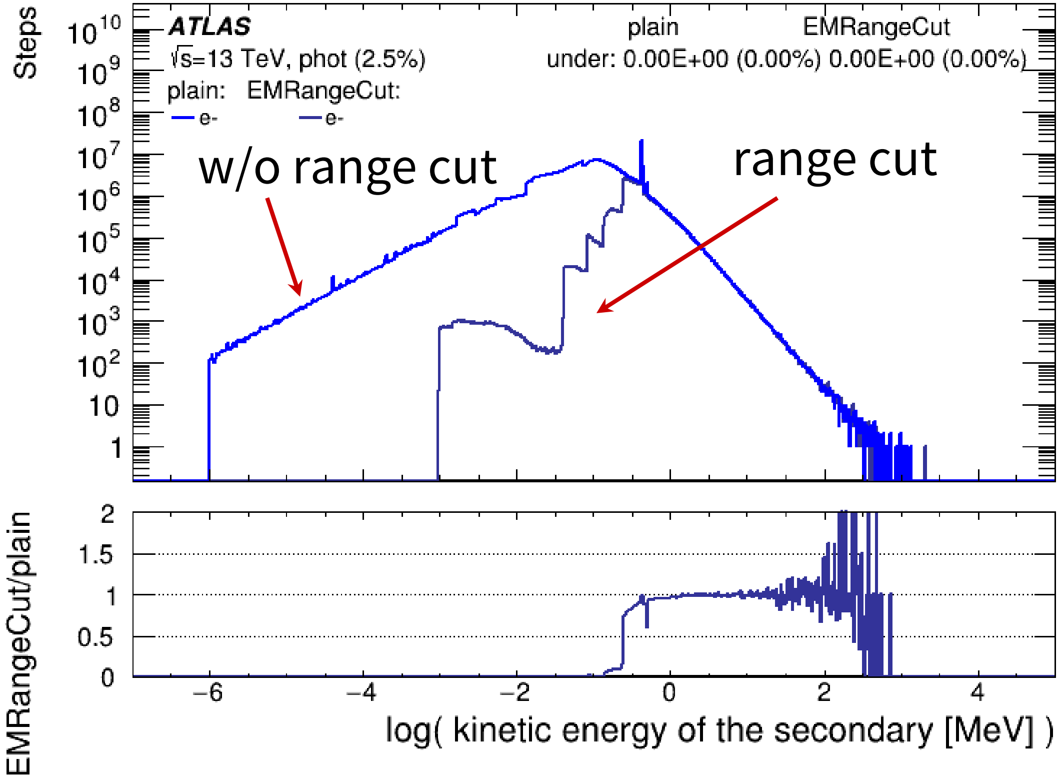
Processes: eBrem

Range cut: e / γ processes

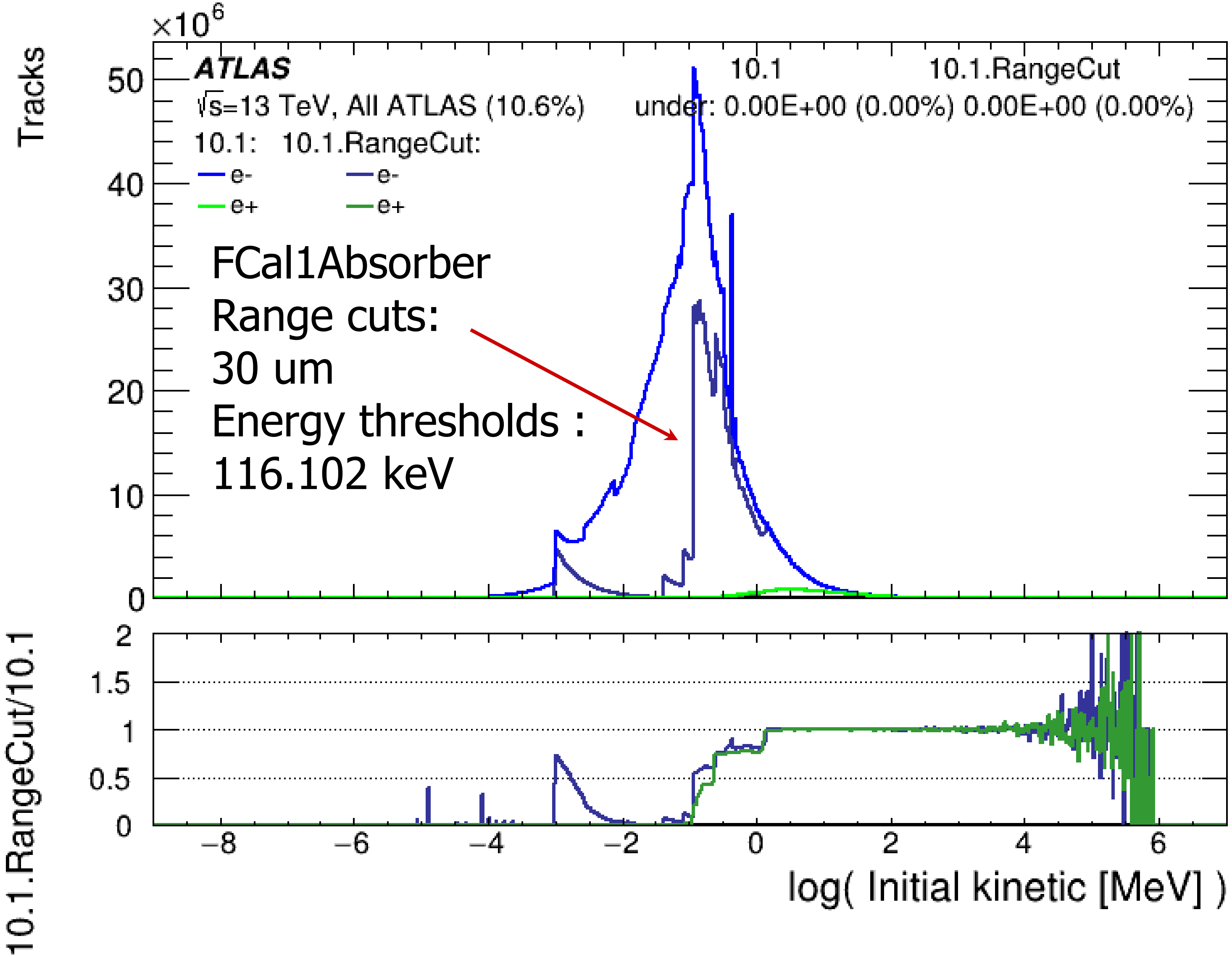
Electron Ionization respects the range cut.
Kinks in the secondary kinetic are clearly visible.



Photoelectric effect ignores range cuts by default.
Electrons down to eV are created and simulated.

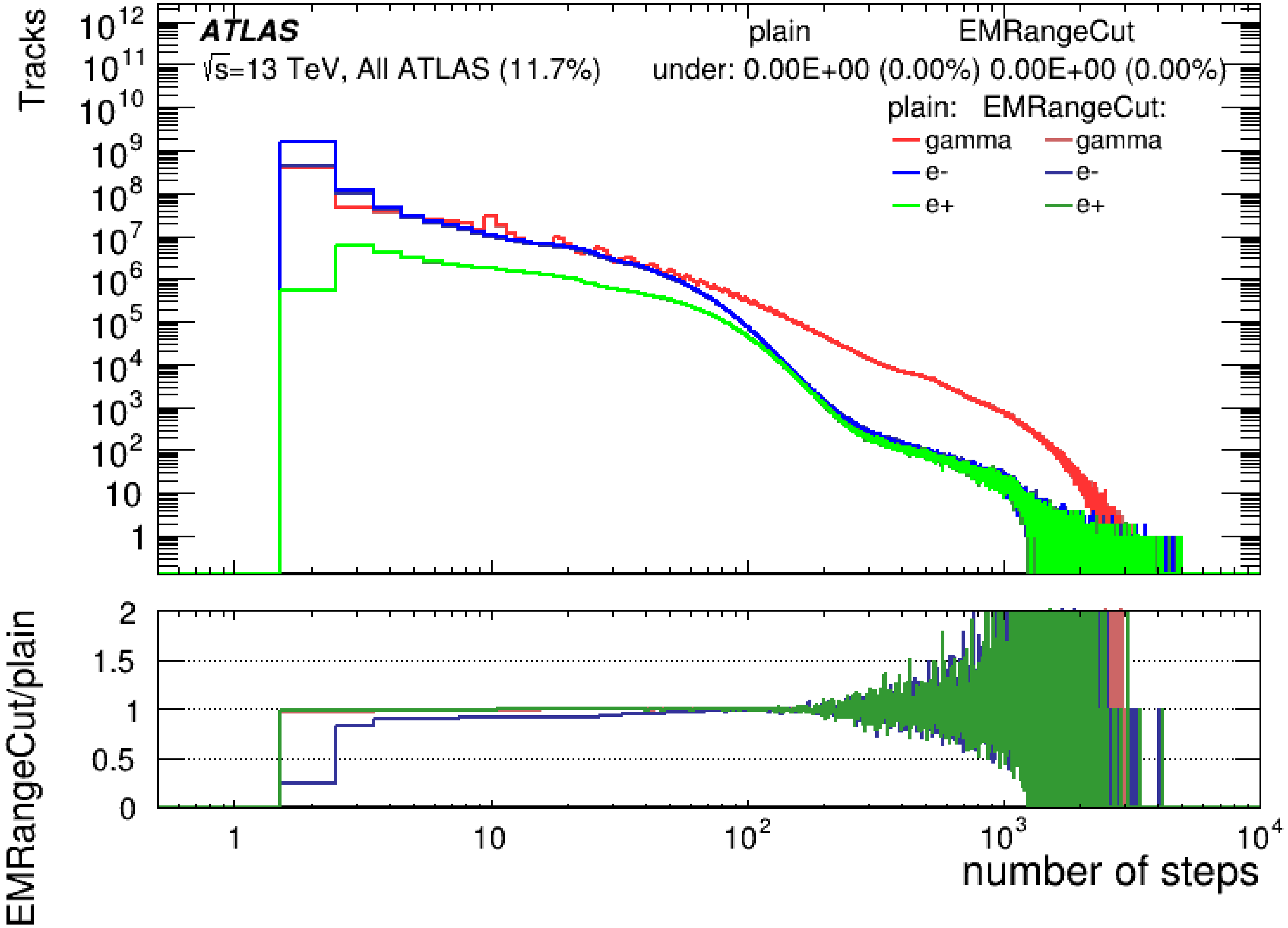


Impact of enforcing range cut in γ processes



- Enforcing the range-cuts for gamma processes in G4
    - (turned off by default)
- **60% fewer electrons** created in total with the range cut in ATLAS.
- The potential speedup of the total simulation time with range cuts for gammas is 6-10%.
- Physics validation undertaken with different thresholds in MeV range

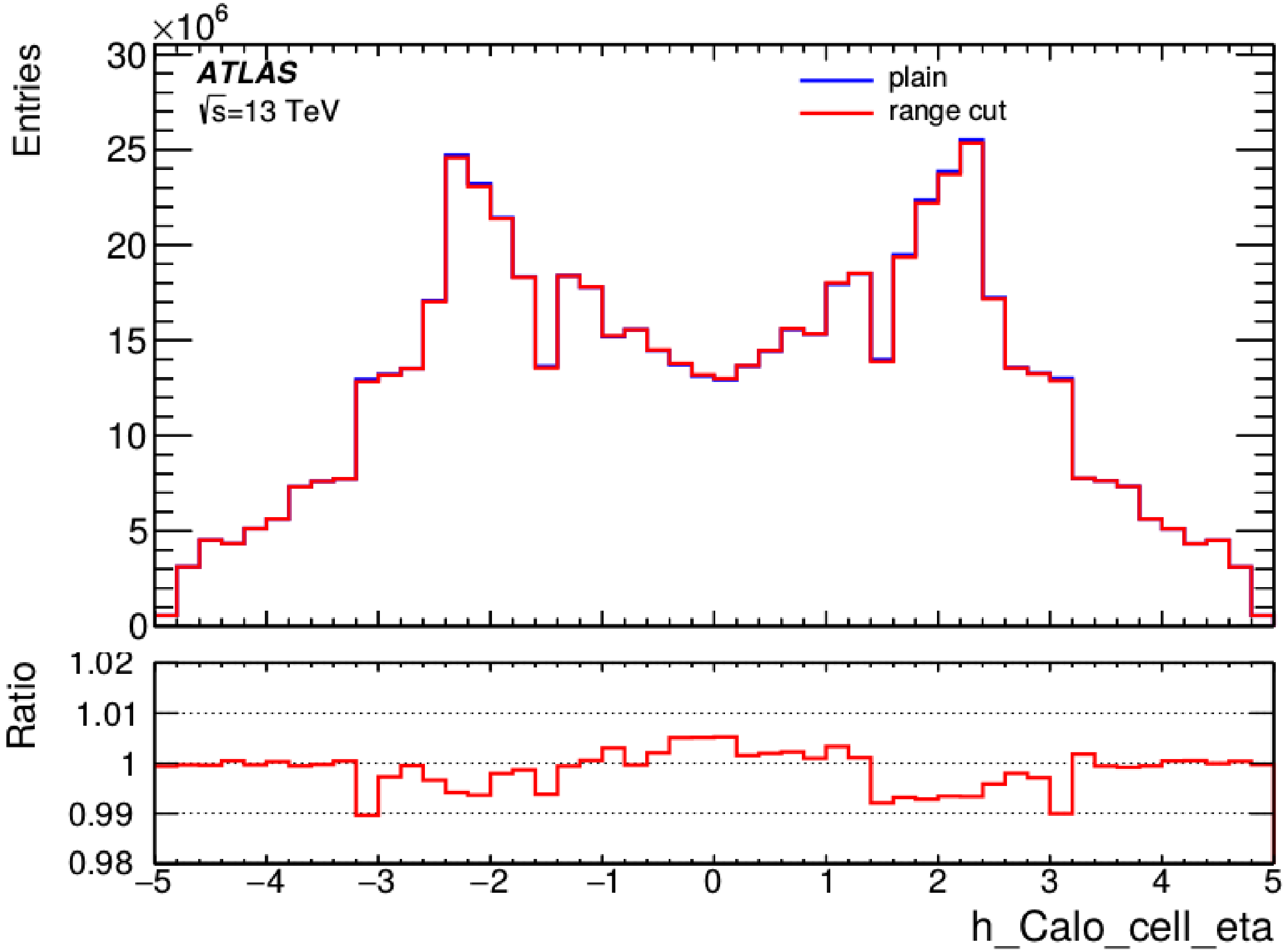
What kind of electrons are these?



- Most of electrons affected by the new range cuts take two steps (including the init step). Some take three steps.
- Two steps means that they are created and immediately die in the next step.
- Range cuts are designed exactly for such cases. Impact on physics should be very low.

Simple hit-count analysis

A simple **hit count analysis** show no significant difference in the number of hits in calorimeters with the range cuts.



- However, this does not take into the actual energy deposit.
- Fewer particles are created 'by construction' when range cuts are applied so fewer hits are expected.
- Full reconstruction needed for confirmation (i.e. PhysVal), but encouraging to see that killing 60% of electrons has such a low impact.

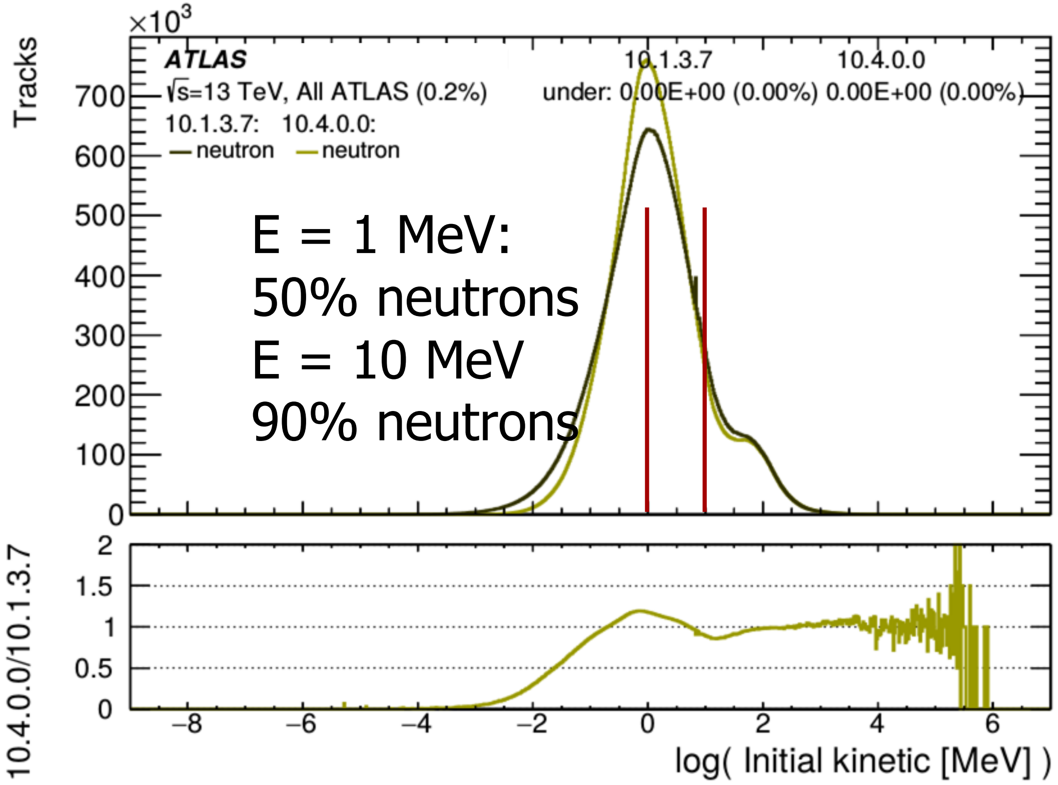
Performance optimization: Neutron Russian Roulette

Randomly kill the majority of neutrons below some energy and weight the energy deposits of remaining neutrons accordingly:

Energy threshold (E),

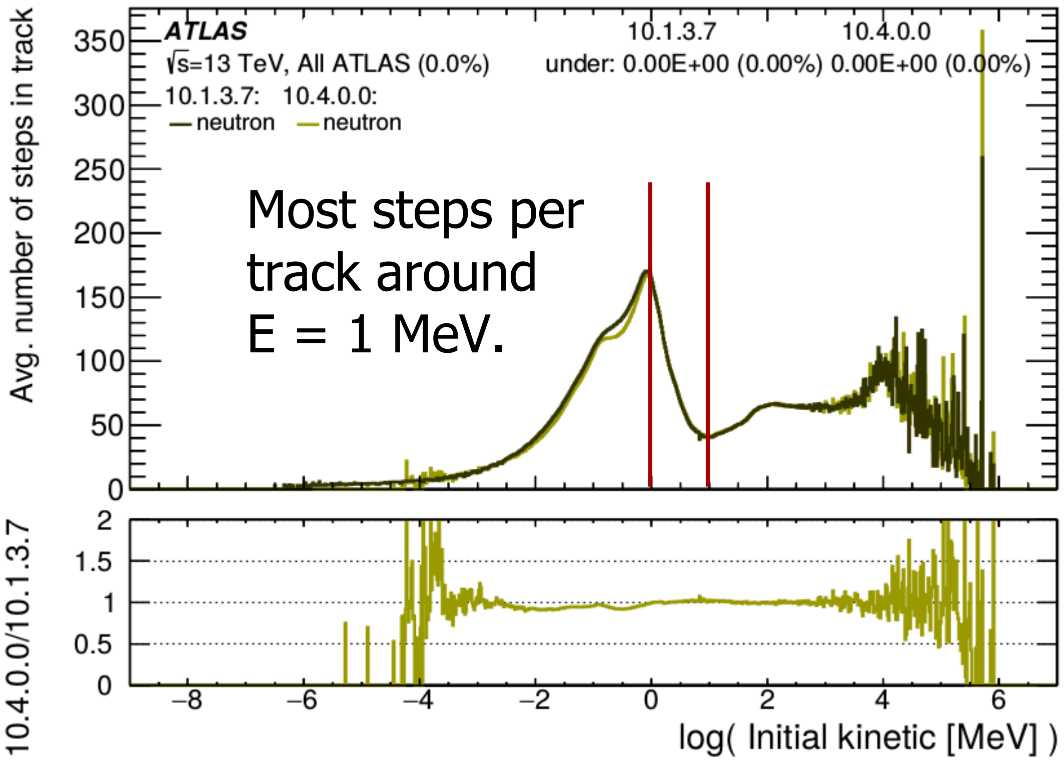
Weight (w): neutrons below E are killed with P((w-1)/w) and weighted with w,

Weighting energy deposits is the tricky part (~25 modified files in Athena).



E = 1 MeV:
50% neutrons
E = 10 MeV
90% neutrons

Initial kinetic energy distribution of neutrons



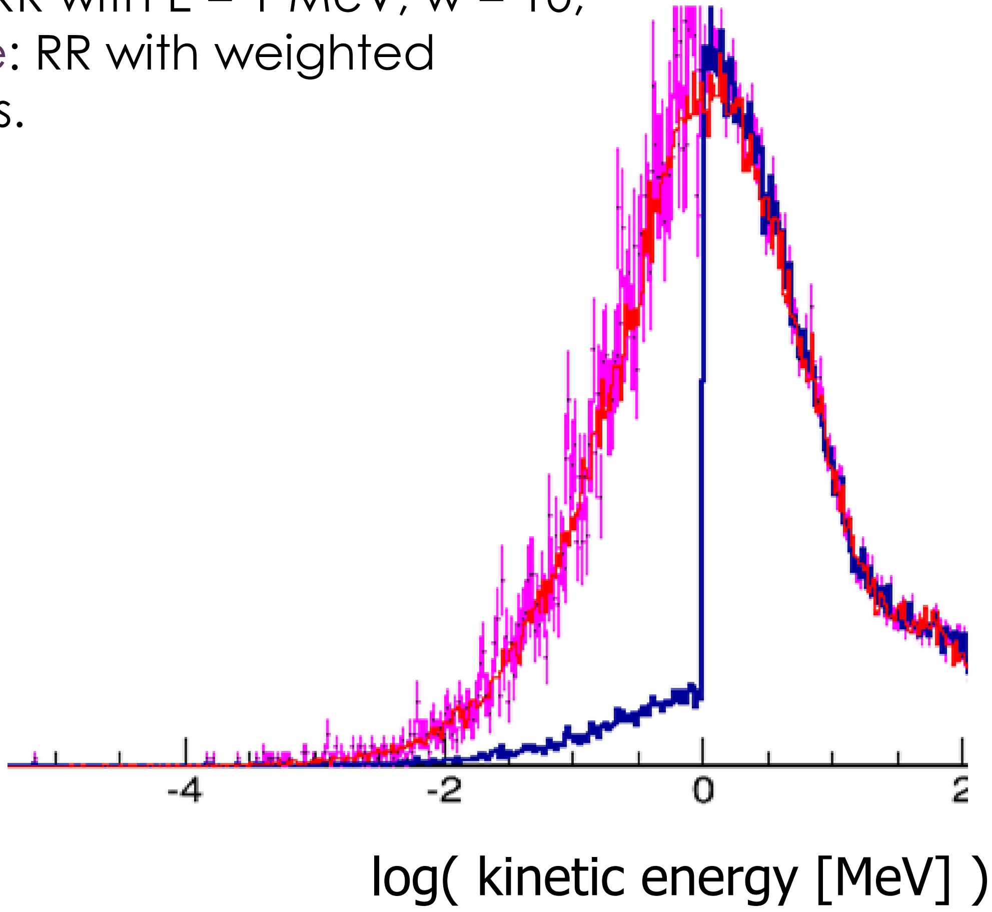
Most steps per track around E = 1 MeV.

Avg. number of steps per track vs initial energy

Expected speedup for NRR

Initial kinetic energy distribution:

Red: plain G4,
Blue: RR with E = 1 MeV, w = 10,
Purple: RR with weighted entries.



Two setups tested:

test1: E = 1 MeV, w = 10,
test2: E = 10 MeV, w = 10.

Expected speedups of the total simulation time are 10% and 20% respectively.

A simple calorimeter hit-level analysis show no significant discrepancies.

Physics validation undertaken for both setups.

Complications in validations of biasing

Two recent validations of performance improvements have encountered ‘larger’ fluctuations in results:

- Photon Russian Roulette
- e range cut for gamma processes in Geant4

Extra samples were simulated to check whether there are ‘real’ differences or not:

- Disjoint samples of 100k events (without optimizations)
- Samples with the same input events, but different random number seeds

A mechanism to reduce the ‘divergence’ of descendant tracks when secondary particles are killed in a simulation (or other ‘history’ changes occur elsewhere in the shower tree upsetting the RNG sequence) is expected to significantly reduce the effort required to undertake such validations.

We know of a trial implementation that could fulfill this stability by ‘pinning’ the RNG state to a G4Track.

Request **feasibility study** for a G4 simulation mode that **avoids fluctuations** due to RNG divergence from ‘downstream’ changes of particle history, e.g. from choice in secondary production and biasing (RR.) in a different branch of the history (not in an ancestor particle.)

Other WIP items

- Geometry optimization effort continues after 2018 gains ~4% (report @Lund):

- Benchmarked VecGeom Solids using Geant4 10.4 and 10.5
  - Using only Cons and Polycons solids from VecGeom gave a 2% -4% speedup (in sample of 500t-tbar events.)
  - Using all solids from VecGeom gave a small slowdown.

- "Big library": static linking of single ATLAS library with static build of Geant4

- Ensuring that multi-threaded simulation (standalone Geant4MT and AthenaMT) produces the exact output of single-threaded simulation
  - Careful comparison of hits uncovered thread-safety issues
  - Fixes regained performance totaling 2-5% level.

AthenaMT & G4MT validation

- Been able to run full multi-threaded G4 within AthenaMT, but outside of ISF, *for some time* (AthSimulation 22.0.0 onwards):
  - *Inter-event parallelism* rather than intra-event parallelism
  - Memory savings come from shared geometry & XS tables
  - Geant4MT requires thread-local initialization by design
  - TBB – on which AthenaMT is based – prefers tasks to be "thread *unaware*" →
    - tricky coupling between AthenaMT and Geant4MT

- Validation of output:
  - Fixed: difference in G4 voxelization configuration between MT and ST (simulation diverged)
  - Fixed: thread-safety in particle and vertex barcode service (~50%)
  - Fixed: some events identical, others have differences in SCT hit IDs (~few%)
  - Fixed: data-race in Calorimeter Sensitive Detector code (~1-3%)
  - Fixed: simulation with CaloCalibrationHit (~50% of Dead material hits)
  - Confirmed reproducibility of simulation with **SUSY/Exotics G4Extensions** enabled (Fixed monopole code thread-unsafe issues)

- Stability fixes:
  - Fixed: crashes due to missing thread-local G4 initialization when TBB spawns extra threads

Update on Readying MT for production

- Multi-threaded simulation is required for Run-4, but is certainly nice to have for Run-3, in order to ensure that hardware with reduced memory per CPU can be fully utilized.

- **Intensive** work to ensure that multi-threaded simulation (Geant4MT and AthenaMT) reproduces the exact output of single-threaded simulation
  - Careful comparison of hits uncovered thread-safety issues. Output Hits now confirmed to be **bitwise identical** in tests of 5k ttbar events.

- Working hard to implement ISF-based G4 Multi-threaded simulation
  - Need to **fully understand initialization sequence in MT-mode**, in order to duplicate it in Athena/ISF simulation using TBB for worker tasks.

Summary

- **Good progress on Optimizing** Atlas Geant4 performance:
  - Range cuts for secondary electrons originating from photons (6-10%)
  - Validation Russian Roulette for neutrons (potential for 10-20%)
  - General improvements of the existing code (few %).
  - Further 'technical' improvements including the "Big Library" will be studied

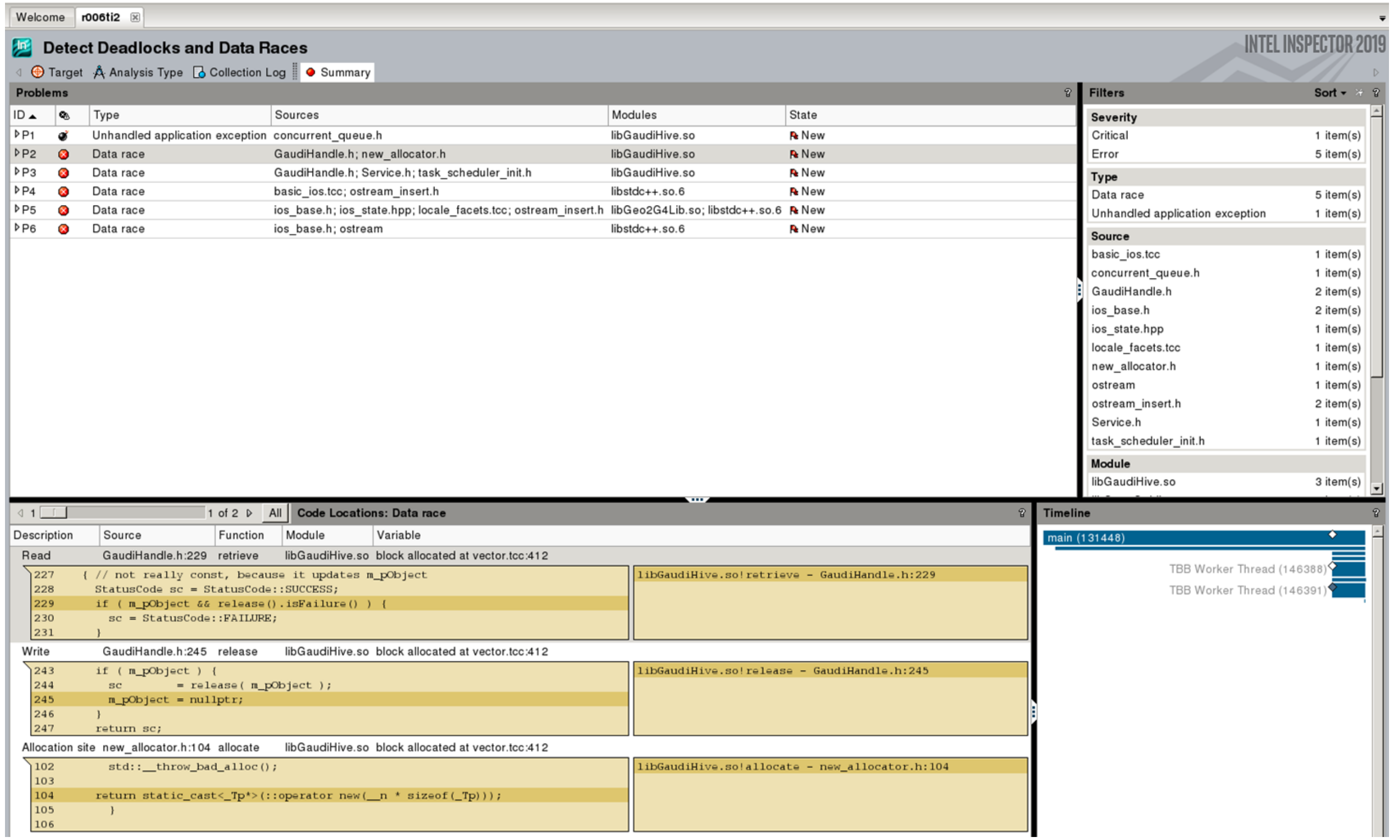
- **Challenges**
  - Validation of options which change RNG seeds is challenging
  - Interest in simulation mode that reduces variance due to "history changes"

- **Good progress on Validation** of AthenaMT with Geant4MT:
  - MT simulation is an important near term goal (LS2)
  - Simulation in MT mode is working – validation is underway
  - Good news for Geant4: no bugs were found (so far) on G4 side!
  - Working on ensuring correct initialization for TBB-powered ISF MT simulation

Thanks for your attention.

Code optimization and profiling with Intel tools

- ~ 10 race-conditions
- ~ 2 lock hierarchy violations/deadlocks
- ~ 2-3 unhandled exceptions



Detect Deadlocks and Data Races

Problems

ID		Type	Sources	Modules	State
P1		Unhandled application exception	concurrent_queue.h	libGaudiHive.so	New
P2		Data race	GaudiHandle.h; new_allocator.h	libGaudiHive.so	New
P3		Data race	GaudiHandle.h; Service.h; task_scheduler_init.h	libGaudiHive.so	New
P4		Data race	basic_ios.tcc; ostream_insert.h	libstdc++.so.6	New
P5		Data race	ios_base.h; ios_state.hpp; locale_facets.tcc; ostream_insert.h	libGeo2G4Lib.so; libstdc++.so.6	New
P6		Data race	ios_base.h; ostream	libstdc++.so.6	New

Filters

Severity	
Critical	1 item(s)
Error	5 item(s)
Type	
Data race	5 item(s)
Unhandled application exception	1 item(s)
Source	
basic_ios.tcc	1 item(s)
concurrent_queue.h	1 item(s)
GaudiHandle.h	2 item(s)
ios_base.h	2 item(s)
ios_state.hpp	1 item(s)
locale_facets.tcc	1 item(s)
new_allocator.h	1 item(s)
ostream	1 item(s)
ostream_insert.h	2 item(s)
Service.h	1 item(s)
task_scheduler_init.h	1 item(s)
Module	
libGaudiHive.so	3 item(s)

Code Locations: Data race

Description	Source	Function	Module	Variable
Read	GaudiHandle.h:229	retrieve	libGaudiHive.so	block allocated at vector.tcc:412

```
  { // not really const, because it updates m_pObject
    StatusCode sc = StatusCode::SUCCESS;
    if ( m_pObject && release().isFailure() ) {
      sc = StatusCode::FAILURE;
    }
```

```
libGaudiHive.so!retrieve - GaudiHandle.h:229
```

Description	Source	Function	Module	Variable
Write	GaudiHandle.h:245	release	libGaudiHive.so	block allocated at vector.tcc:412

```
    if ( m_pObject ) {
      sc          = release( m_pObject );
      m_pObject = nullptr;
    }
    return sc;
```

```
libGaudiHive.so!release - GaudiHandle.h:245
```

Description	Source	Function	Module	Variable
Allocation site	new_allocator.h:104	allocate	libGaudiHive.so	block allocated at vector.tcc:412

```
      std::__throw_bad_alloc();

    return static_cast<_Tp*>(::operator new(__n * sizeof(_Tp)));
      }

```

```
libGaudiHive.so!allocate - new_allocator.h:104
```

Timeline

```
main (131448)
TBB Worker Thread (146388)
TBB Worker Thread (146391)
```

Read

Write

Alloc

Concurrent Threads

Case study: barcode service for multiple threads

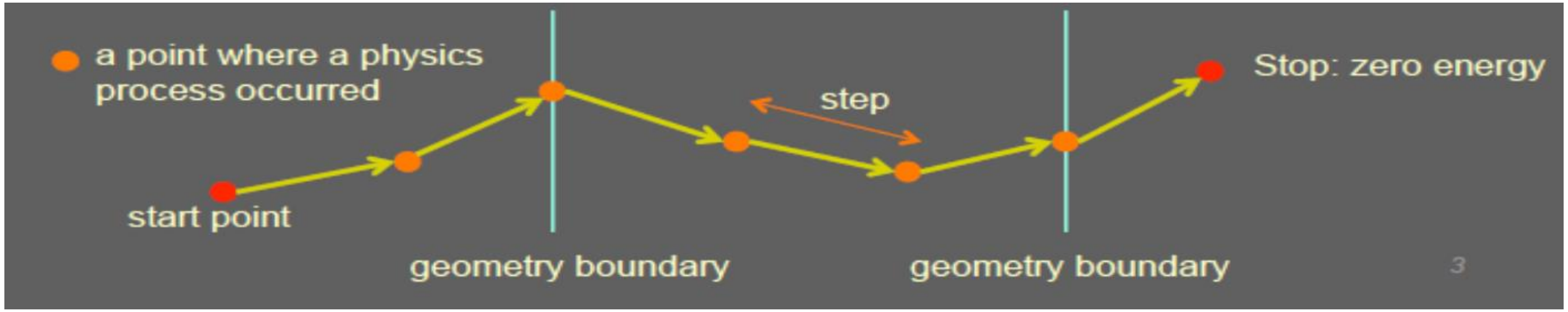
- Barcode service provides **unique particle and vertex barcodes**:

  - internal barcode counters are incremented each time a new barcode is requested
    - returned barcode is simply the incremented value
  - counters are reset at the beginning of each event
  - Service was made thread-safe by:
    - storing the counters in a tbb::concurrent_unordered_map with the std::thread::id as the key and initializing a key-value pair for each thread, and
    - replacing the BeginEvent incident used to trigger the counter reset with a resetBarcodes() call inside the algorithm execute()

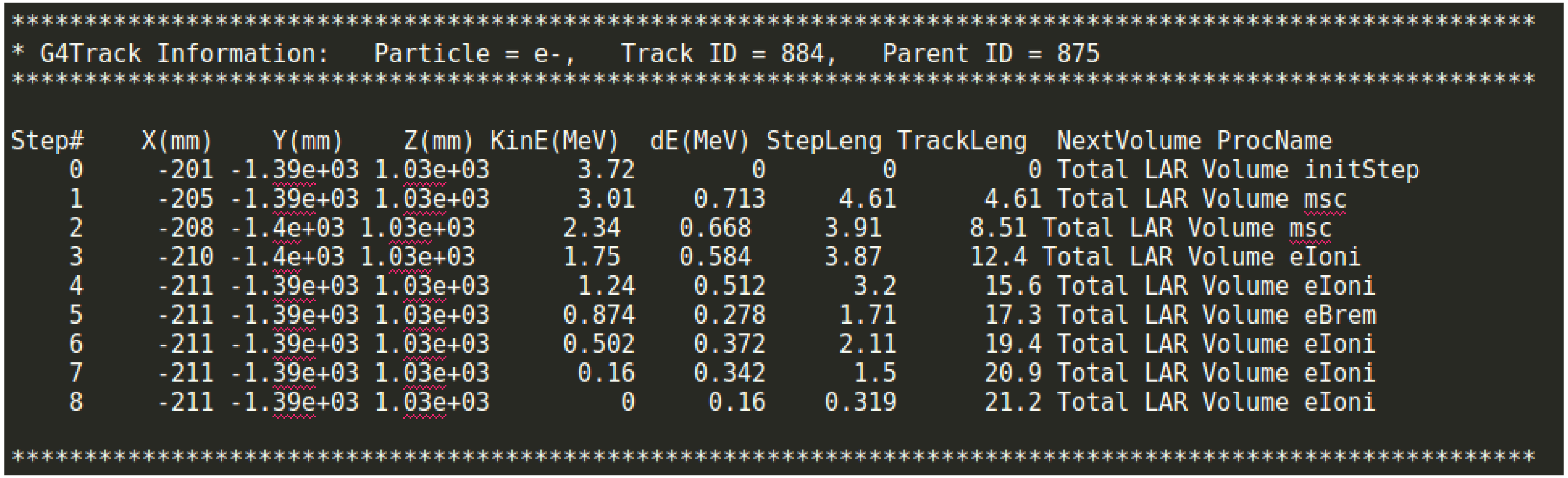
  - Services in AthenaMT should be stateless
    - The use of tools such as Intel Inspector is helping us to detect threading bugs

Geant4 simulation in ATLAS

'Steps' are the smallest units in a Geant4 simulation.



It is possible to intercept information about each step with User Actions:



```
********************************************************************************************************
* G4Track Information:   Particle = e-,   Track ID = 884,   Parent ID = 875
********************************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0     -201 -1.39e+03 1.03e+03      3.72        0        0         0 Total LAR Volume initStep
    1     -205 -1.39e+03 1.03e+03      3.01    0.713     4.61      4.61 Total LAR Volume msc
    2     -208 -1.4e+03 1.03e+03      2.34    0.668     3.91      8.51 Total LAR Volume msc
    3     -210 -1.4e+03 1.03e+03      1.75    0.584     3.87      12.4 Total LAR Volume eIoni
    4     -211 -1.39e+03 1.03e+03      1.24    0.512      3.2      15.6 Total LAR Volume eIoni
    5     -211 -1.39e+03 1.03e+03     0.874    0.278     1.71      17.3 Total LAR Volume eBrem
    6     -211 -1.39e+03 1.03e+03     0.502    0.372     2.11      19.4 Total LAR Volume eIoni
    7     -211 -1.39e+03 1.03e+03      0.16    0.342      1.5      20.9 Total LAR Volume eIoni
    8     -211 -1.39e+03 1.03e+03         0     0.16    0.319      21.2 Total LAR Volume eIoni

********************************************************************************************************
```

Validation of the range cut for gamma processes in Geant4

- Running the simulation with this option gives an expected speedup of **about 6-7%** while the impact on physics should be negligible by design.

- Range cuts are already **turned on** for the majority of other processes.
  - Some simple physics tests were already performed and the agreement was good enough in our opinion to proceed with the physics validation

- Range cuts for gamma processes (conv, phot, compt) are **turned off** by default in Geant4.
  It is possible to turn them on with a simple postExec:

```
--postExec="from G4AtlasApps.SimFlags import simFlags; simFlags.G4Commands
+= ['/process/em/applyCuts true']"
```

Performance

The raw number of steps in same 1000 ttbar events has changed as follows:

- electron steps: (7.56e9 - 5.88e9) / 7.56e9 = 22%
- all steps: (2.64e10 - 2.46e10) / 2.64e10 = 6.8%

Assuming that CPU time is proportional to the number of steps a **6-7% speedup is expected**.

Local test

Two jobs with 100 ttbar events were submitted locally on a quiet machine for timing purposes:

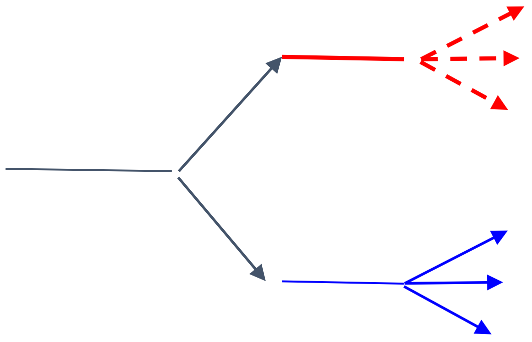
- no range cut: Ave/Min/Max= 3.67(+- 1.52)/ 1.12/ 9.3[min]
- w/ range cut: Ave/Min/Max= 3.46(+- 1.39)/ 1.2/ 8.57[min]

Local speedup is about 6%.

Grid jobs

10000 ttbar events were submitted on the GRID to perform the Calo Hits Analysis
jobs with the range cut are in general **faster by about 10% in this example**

‘Independence’ of tree branches



Multiple ways to simulate:

- with all the tracks or
- replacing the (detailed) simulation of the red branch, or
- replacing the interaction that resulted in the red dashed particles.

To reduce fluctuations, what is needed is that **the simulation of an unrelated branch** of the tree - (e.g. the blue one) **is unaffected** by the choices in simulating the red branch - even if the red branch was simulated before the blue one.