Geant4 in Atlas

Marilena Bandieramonte
marilena.bandieramonte@cern.ch

On behalf of the Atlas Simulation Team

18th January 2019

Current production I

- 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 later releases built using 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:
  - **MC15**
    - Geant4 9.6 patch03, CLHEP 2.1, 64-bit, gcc 4.7, SLC6, C++11
  - **MC12**
    - Geant4 9.4+ patches for "MC12" production

Current production II

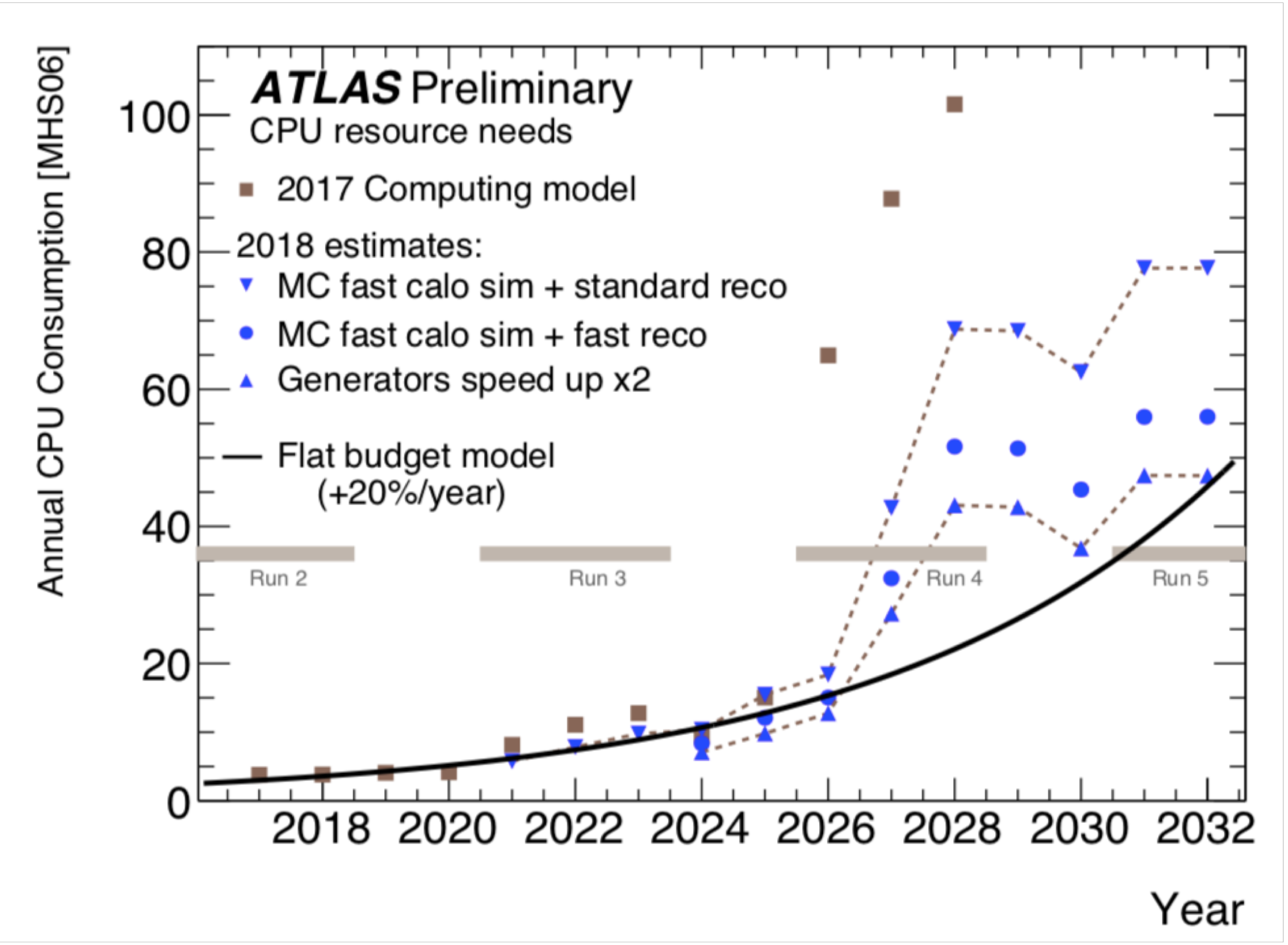
- **Upcoming changes:**
  - Hope to update 21.0 to use Geant4 10.1.patch03.atlas07 (G4 Solid updates – 4% speedup) soon - some difficulties due to other externals changes.
  - Aiming to update master to use Geant4 10.4.patch02.atlas01
  - Early testing of Geant4.10.5: We built AthSimulation with Geant4.10.5. It will be used for testing purposes

- **The next MC** campaign (preparing for LHC Run 3) will most likely use Geant4 10.4.patch02.atlas01
  - we are testing Geant4.10.5 but aren't ready to make a decision on that yet.

Requests or features

- Allow Geant4 to deal with **zero-lifetime particles** (needed for quasi-stable particle simulation) - Currently testing patches from Makoto – Thanks!
- Improving the **robustness of commands executed via G4UIManager** – What is the current status of this? Which G4 versions will be patched?

CPU needs projection



Plot from Davide Costanzo presented at: HL-LHC_Weekly-2018-12-04

- CPU consumption will increase dramatically for HL-LHC.
- Most of simulation will rely on FCS, 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.

AthenaMT & Geant4MT validation

- We have been able to run full multi-threaded Geant4 within AthenaMT *(AthSimulation 22.0.0)* for some time now:
  - *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

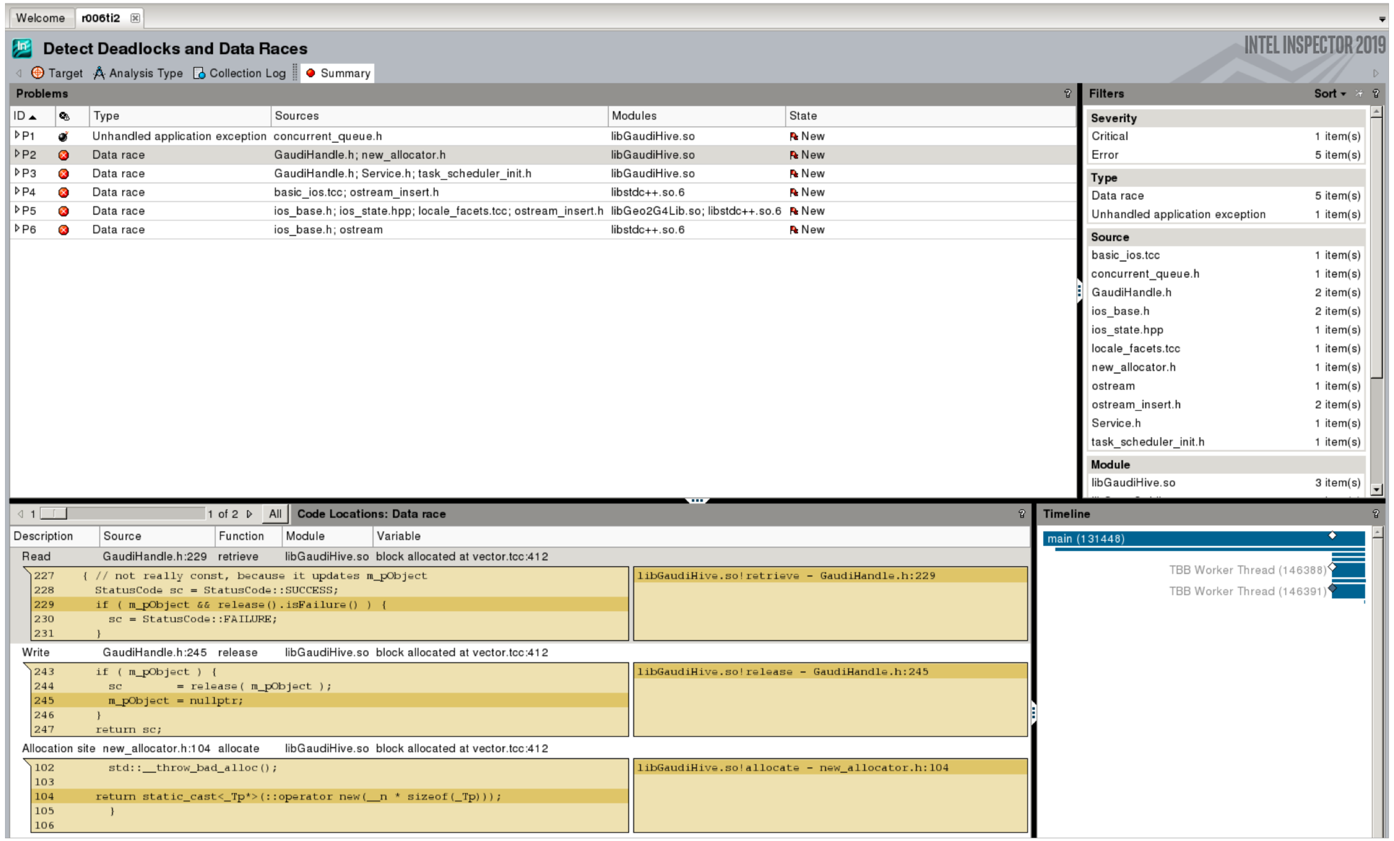
- Validation of output:
  - Fixed: difference in G4 voxelization configuration between MT and ST (simulation diverged)
  - Fixed: thread-safety in particle and vertex barcode service (~50%)
  - Fixed : some events identical, others have differences in SCT hit IDs (~few%)
  - Now: debugging Calorimeter Sensitive Detector code to understand differences in hits (~1-3%)

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

Code optimization and profiling with Intel tools

4th OpenLab-Intel hands-on workshop

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



Read

Write

Alloc

Concurrent Threads

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::DistanceCalculatorSaggingOf	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 I

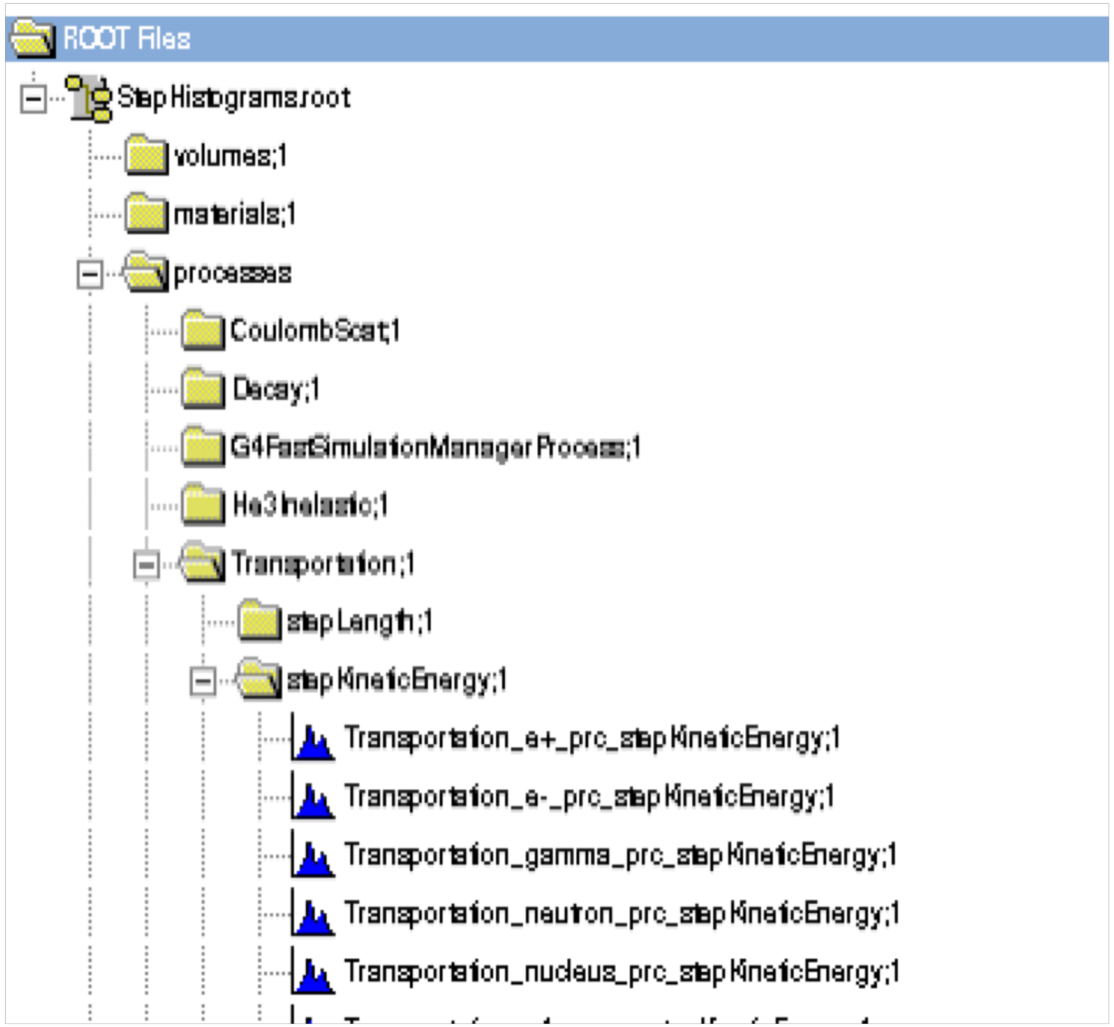
Tool that plots histograms of various step-related quantities:

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

Material

Volume

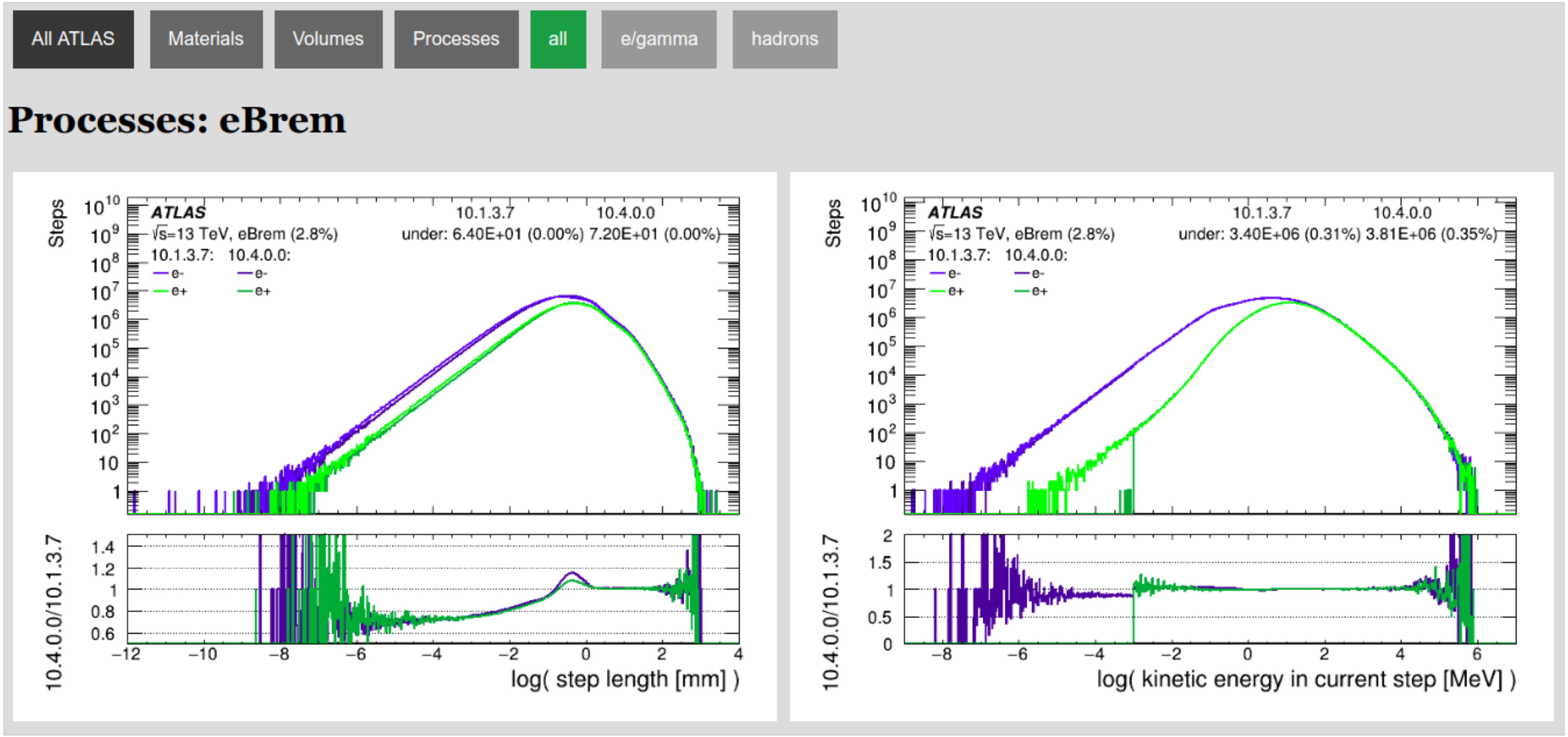
Process



Geant4 debugging tools II

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

O(2000) plots, e.g.: G4 10.1.3.7 vs. G4 10.4.0.0.



Performance optimization: range cuts

- Range cuts are a built-in way of optimizing Geant4 performance.
- Range cuts (in length) are define per region. For each pair of (material, region) they get converted into an energy threshold.

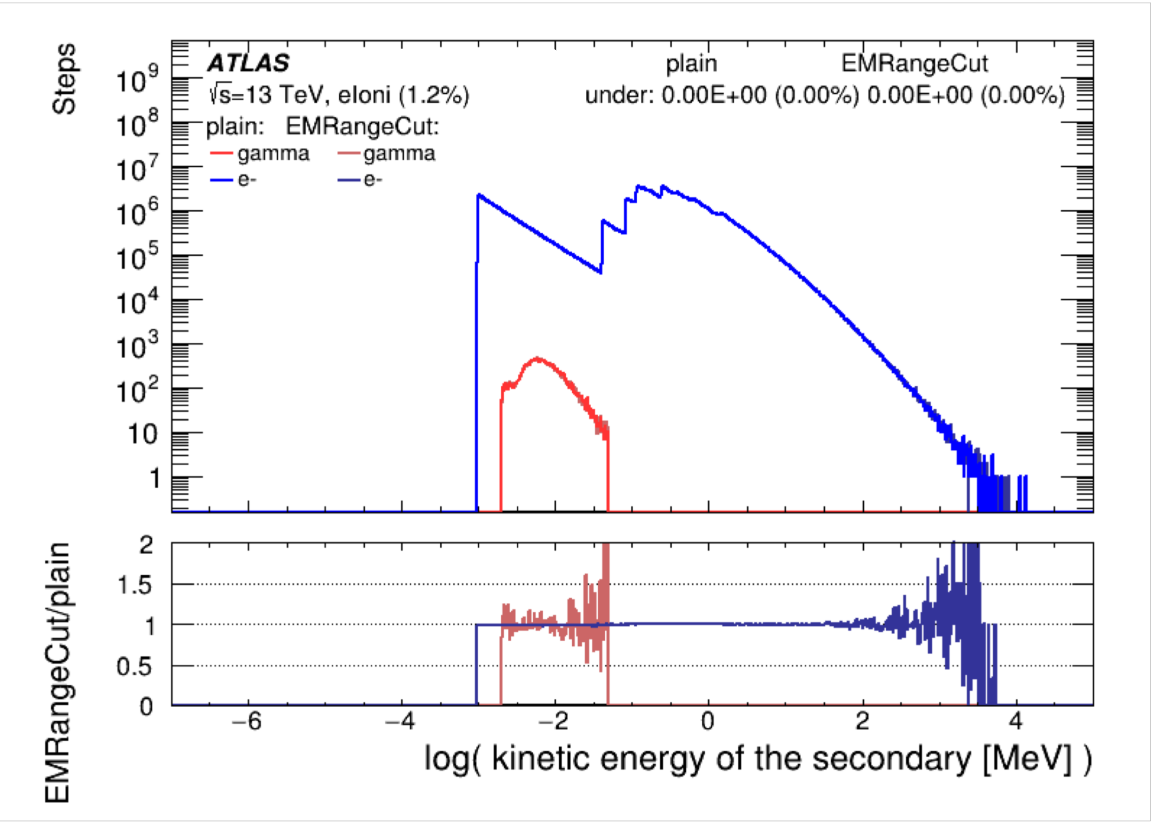
```
Index : 530     used in the geometry : Yes
 Material : LiquidArgon
 Range cuts         :  gamma  30 um     e-  30 um     e+  30 um  proton 1 mm
 Energy thresholds :  gamma  1.10981 keV    e-  41.2472 keV    e+  40.971 keV proton 100 keV
 Region(s) which use this couple :
    FCAL
```

Example: *LiquidArgon, 30 um range cut for electrons*

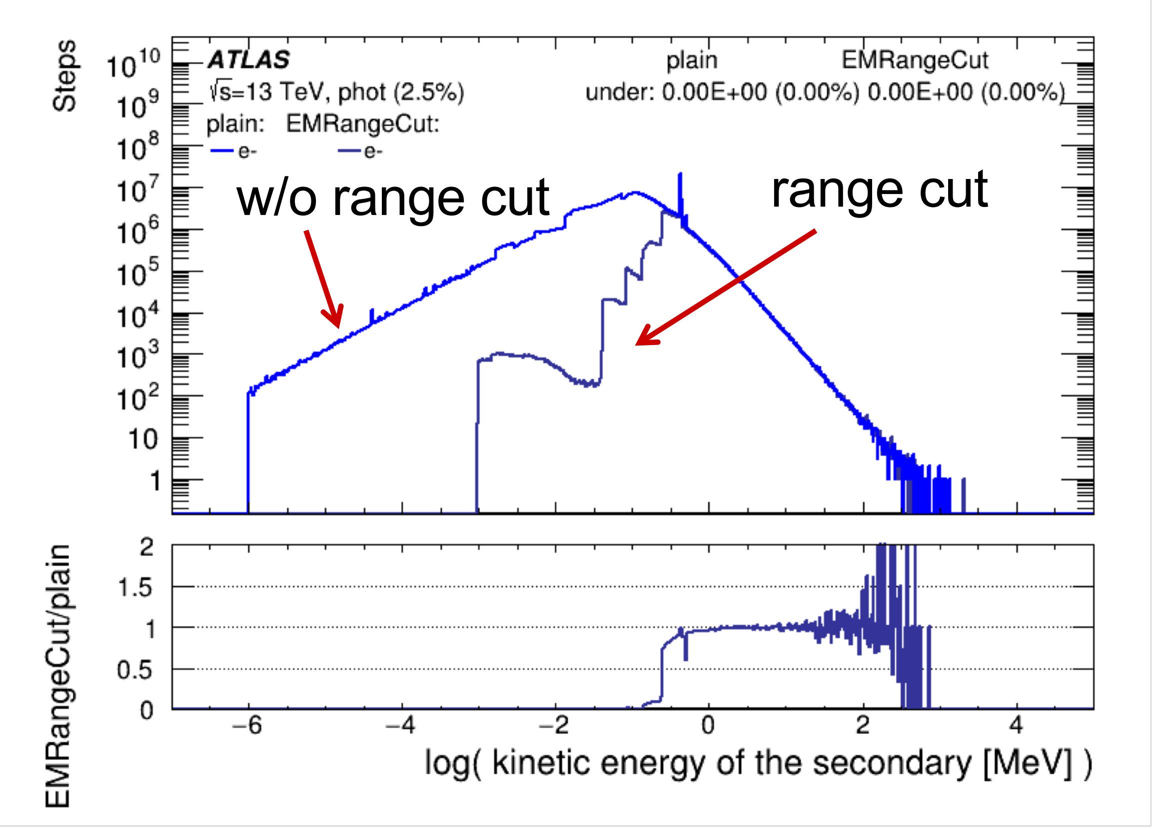
- Secondaries, that are expected to travel less than the range cut are not created and their energy is immediately deposited.

Range cut example

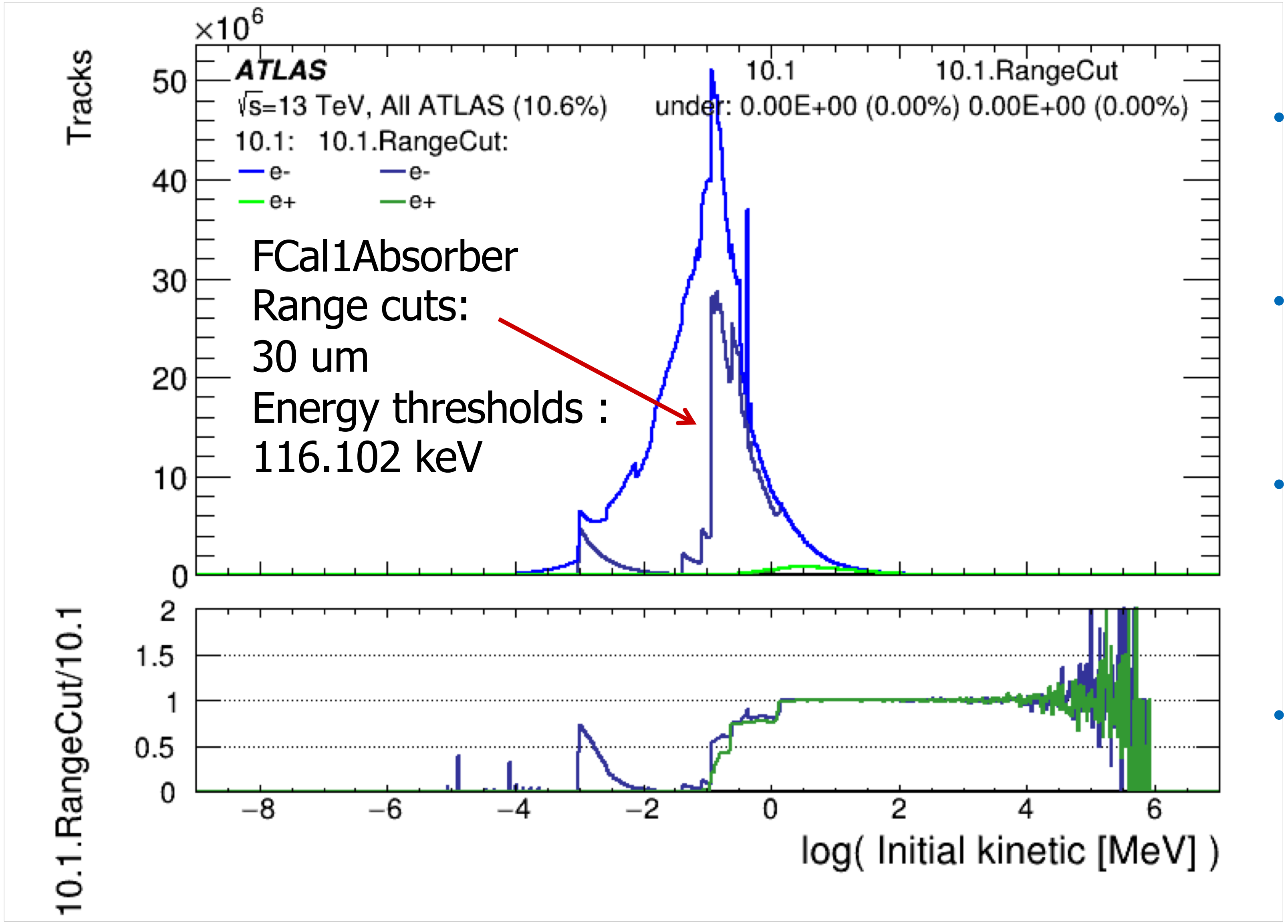
Electron Ionization respects the range cut.
Kinks in the secondary kinetic are clearly visible.



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

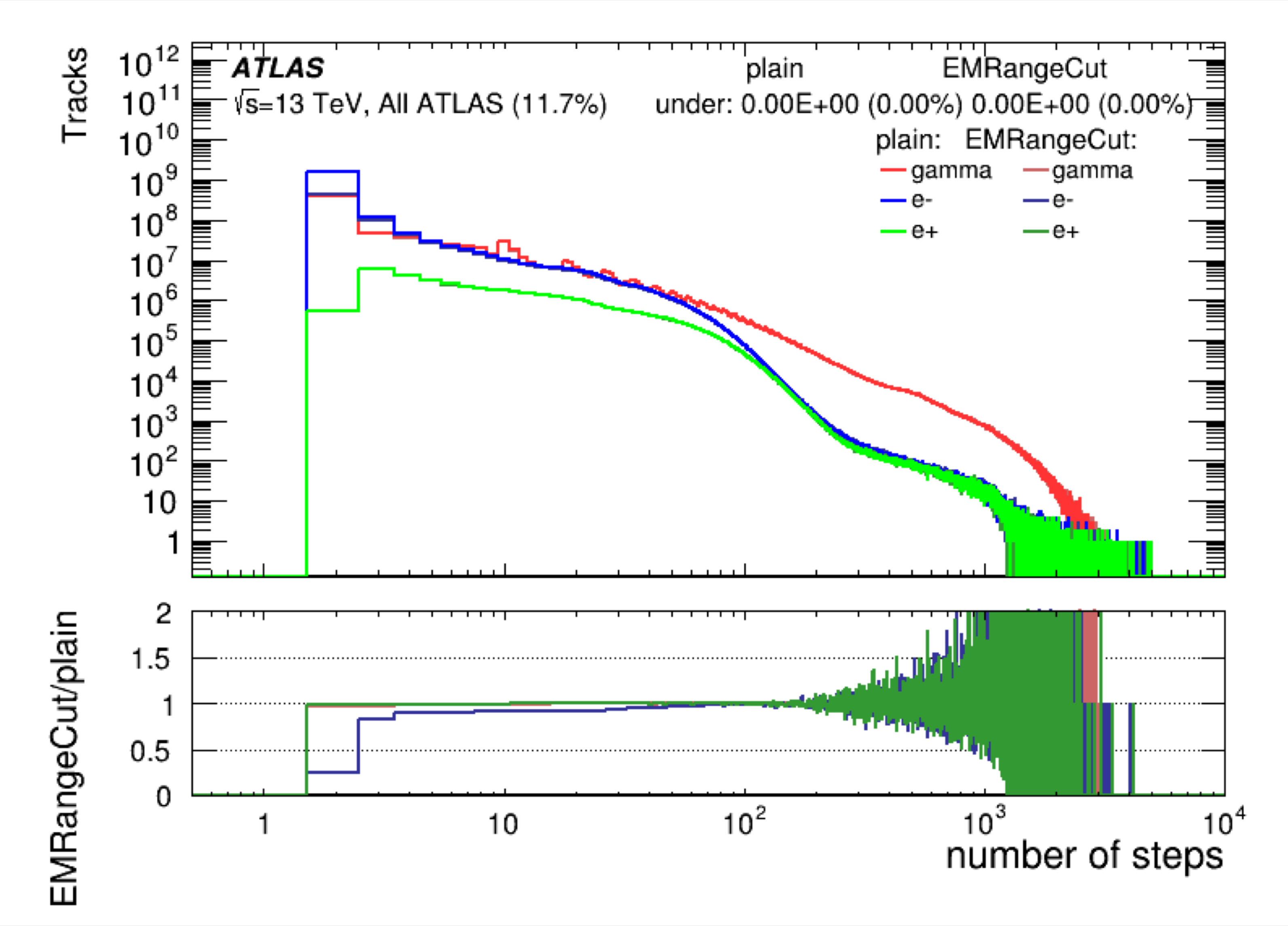


Impact of new range cut



- Range-cuts are turned off by default for gamma processes in G4.
- 60% less 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%.
- Currently running physics validation

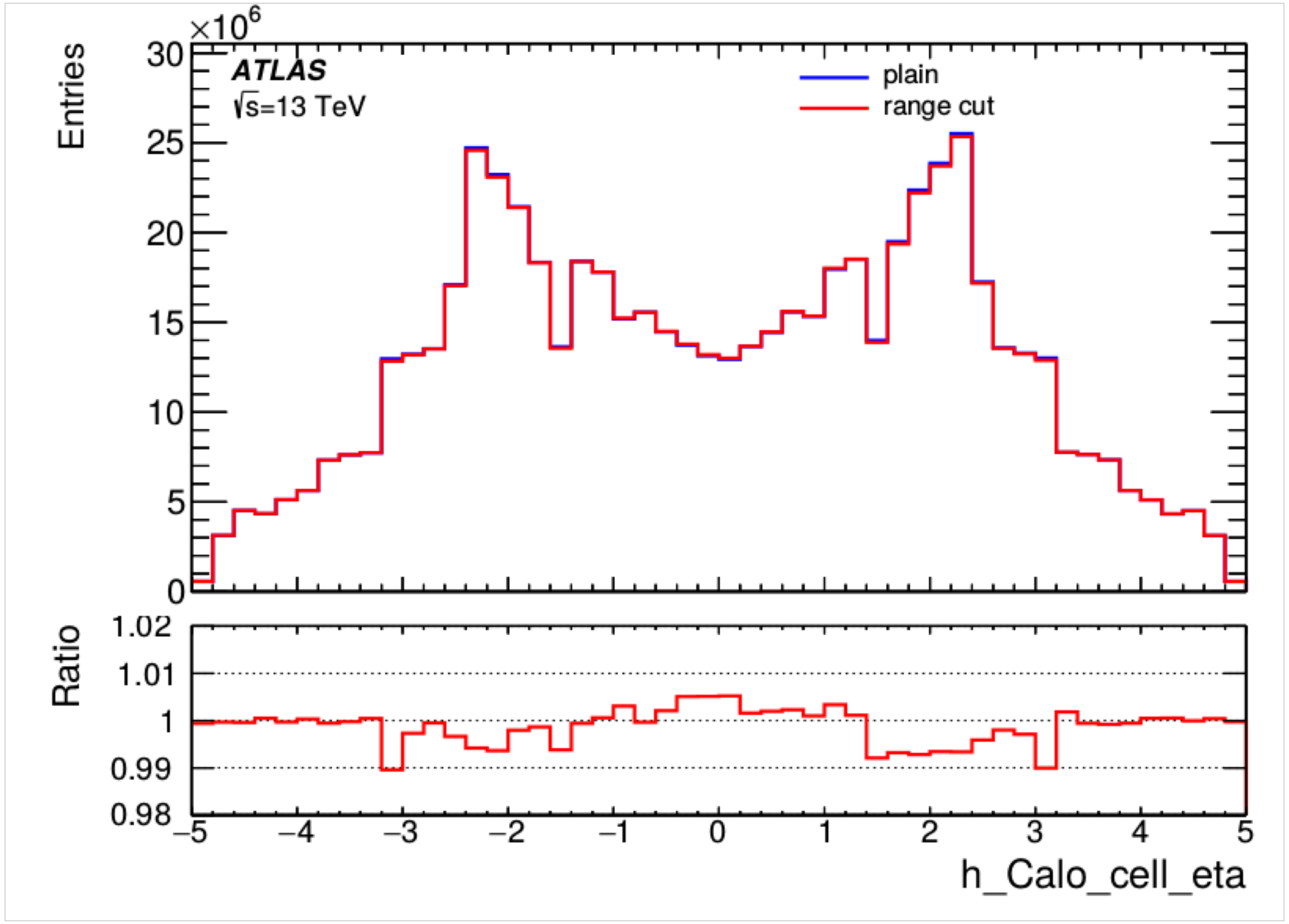
What kind of electrons are these?



- Most of electrons affected by the new range cuts have two steps. Some have 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.

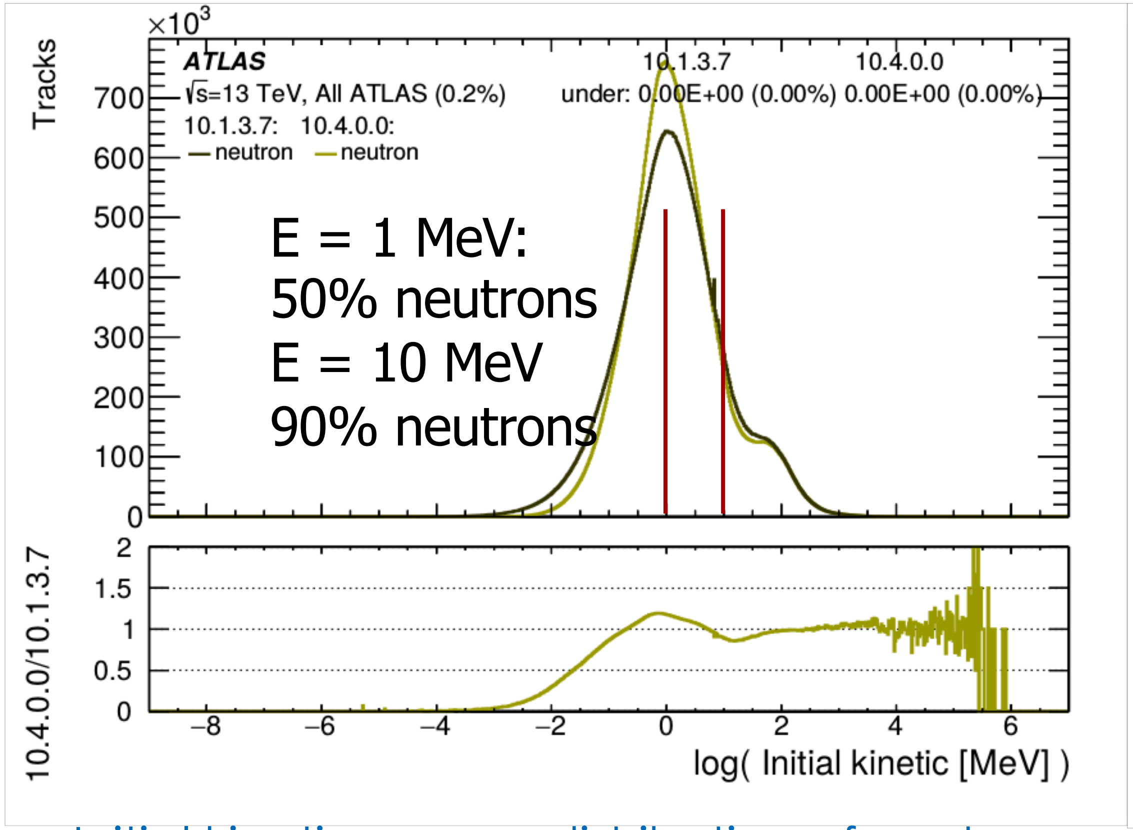


- However, this does not take into the actual energy deposit.
- We get less particles by construction with range cuts and therefore less hits are expected.
- Full reconstruction is needed (i.e. PhysVal), but encouraging to see that killing 60% of electrons has such a low impact.

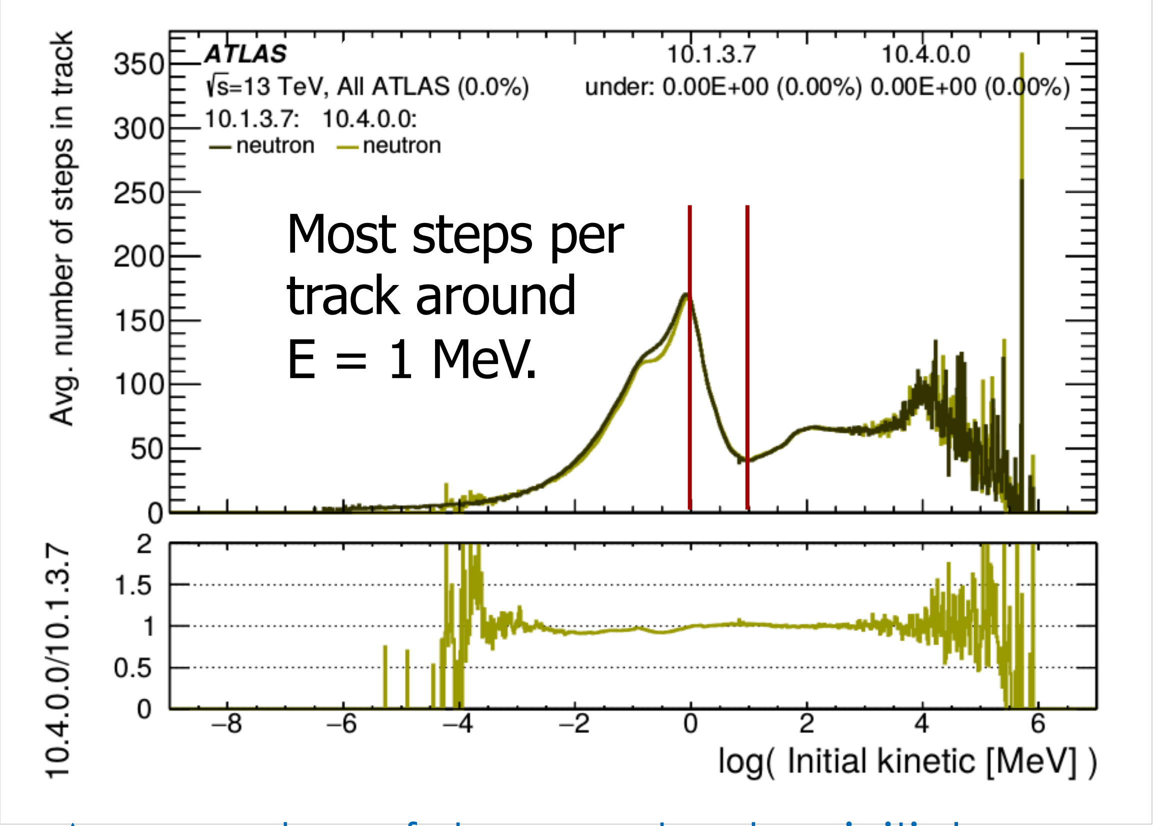
Performance optimization: Neutron Russian Roulette

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).



Initial kinetic energy distribution of neutrons

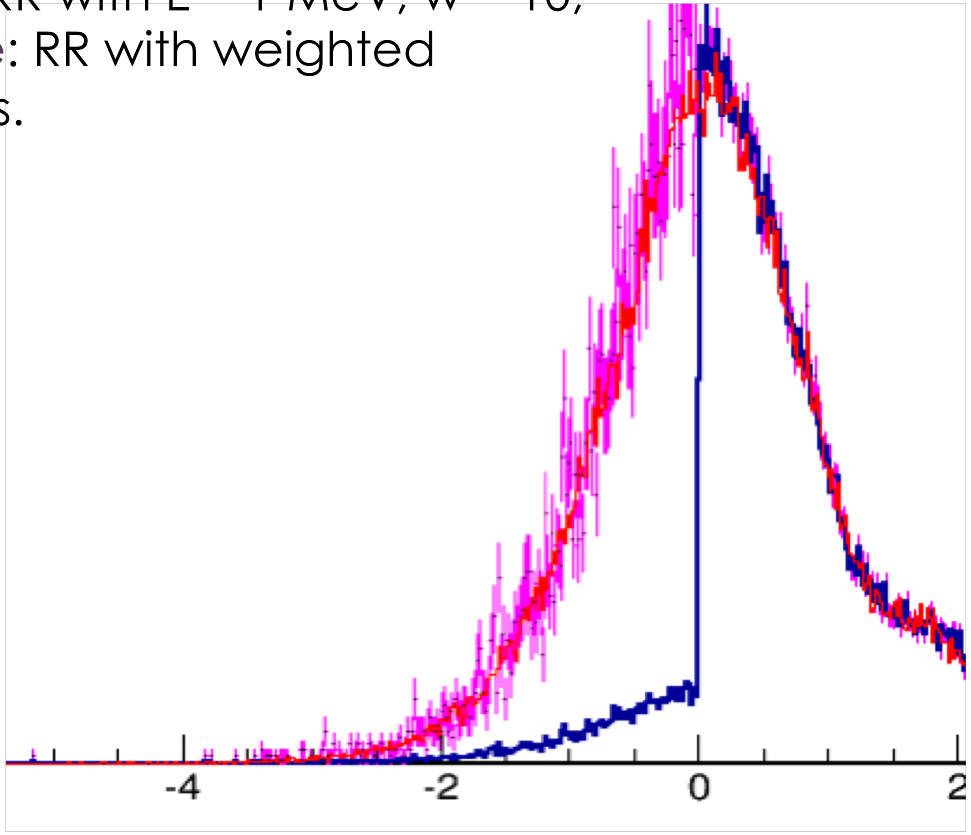


Avg. number of steps per track vs initial energy

Expected speedup for NRR

Initial kinetic energy distribution:

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



log(kinetic energy [MeV])

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 requested for both setups.

Other WIP items

- Geometry work : optimization effort ongoing (~4% speedup)
- “Big library”: static linking of single ATLAS library with static build of Geant4
- Building Athena on top of G4 10.4 with VecGeom
- ATLAS Test Beam Simulations: ATLAS TileCal TB, geometry files retrieved as GDML, standalone simulation code using CALICE approach ”under construction”; details such as mapping of TileCal cells and PMTs still needs some work.

Summary

- **Good progress on Optimizing** Atlas Geant4 performance:

  - Range cuts for secondary electrons originating from photons (6-10%)
  - Russian Roulette for neutrons (10-20%)
  - General improvements of the existing code (few %).
  - Together with other improvements such as the "Big Library" a significant performance increase can be expected

- **Good progress on Validation** of AthenaMT with Geant4MT:
  - Good news for Geant4: no bugs found (so far) on G4 side!

Thanks for your attention.

Marilena Bandieramonte
marilena.bandieramonte@cern.ch

Case study: barcode service for multiple threads

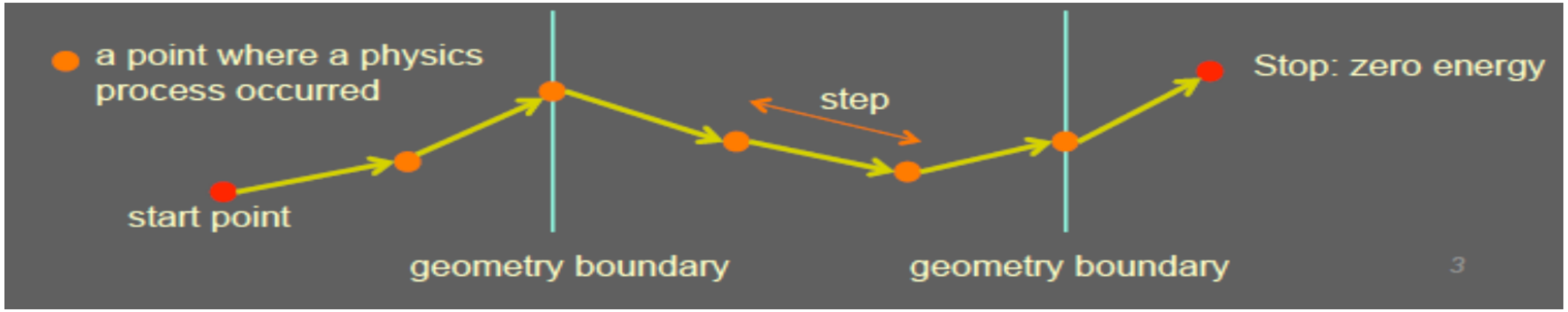
- Barcode service provides **unique particle and vertex barcodes**:

  - 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 with range cut

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**