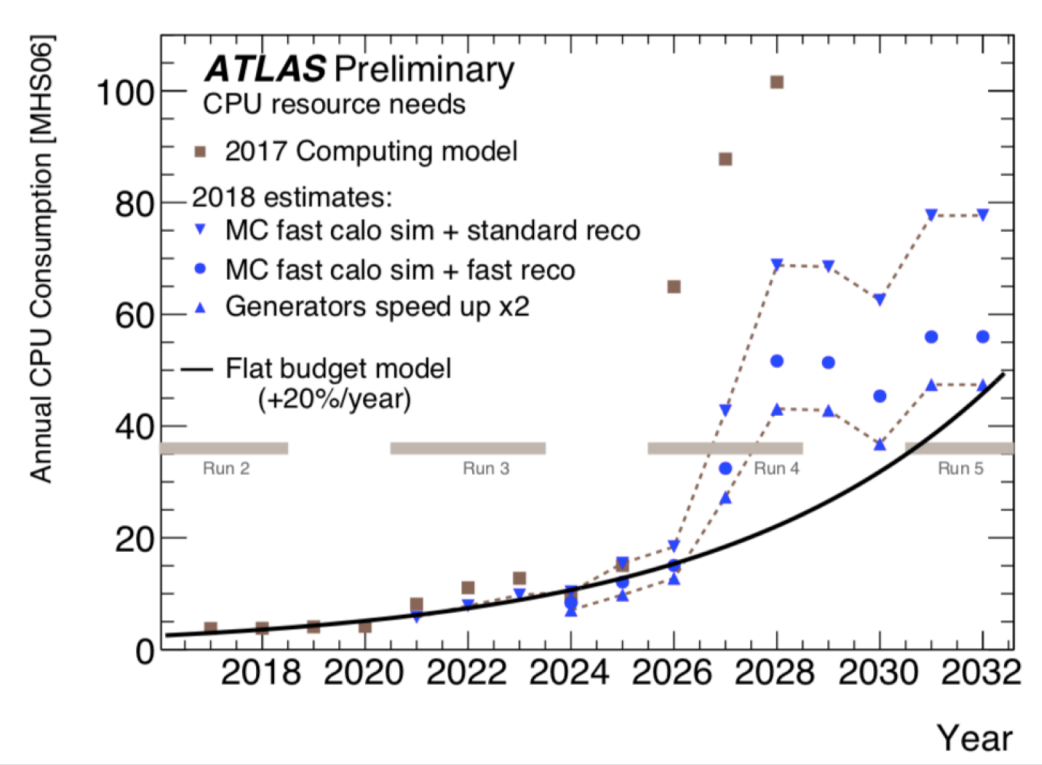
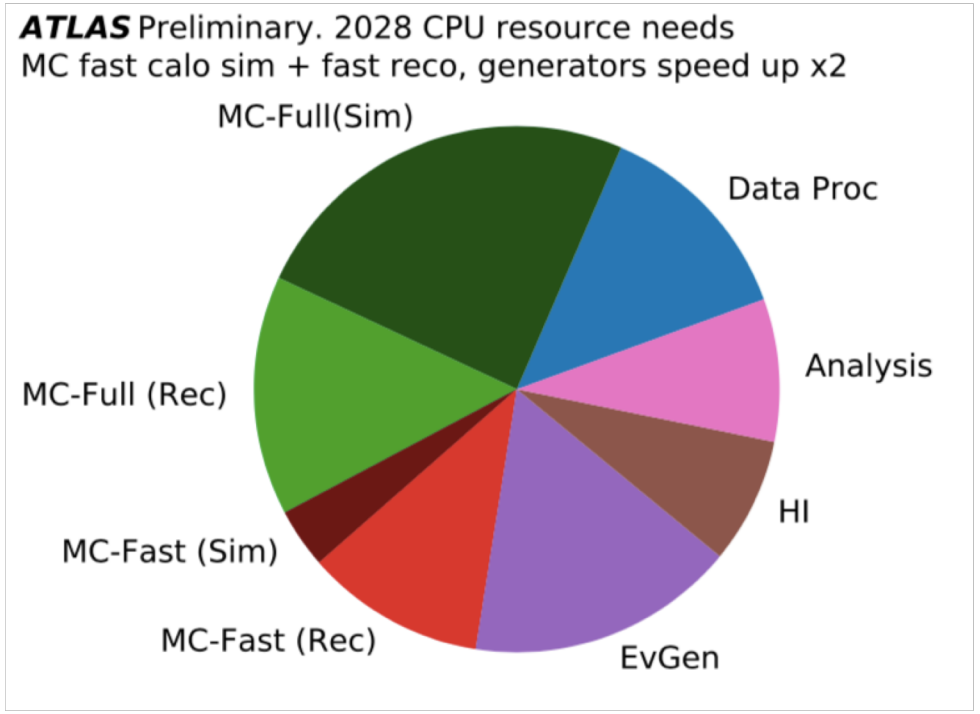
Geant4 in Atlas

Geant4 Technical Forum
29th March 2019

Marilena Bandieramonte
on behalf of the ATLAS Simulation Team
marilena.bandieramonte@cern.ch

Introduction





- Estimated CPU resources (in MHS06) needed for the years 2018 to 2032 for both data and simulation processing.
- Most of simulation will rely on FCS, but full Geant4 sim will be heavily used regardless
- Any performance optimizations of ATLAS simulation have a big impact on the overall picture.

Outline

- **Current production and G4 releases** :
  - No major changes w.r.t. the last Forum:
    - Ready to use Geant4 10.4.patch03 in Athena master branch
  - Testing Geant4.10.5
  - Testing Geant4.10.4 with VecGeom

- **Geant4 Simulation Optimization:**
  - EM range cuts
  - Neutron Russian Roulette
  - Photon Russian Roulette

- **Geant4 based MT simulation validation:**
  - Tile data race issue
  - LAr data race issue
  - CaloCalibration Hits
  - Intel Inspector issues

Initial tests with Geant4 10.5

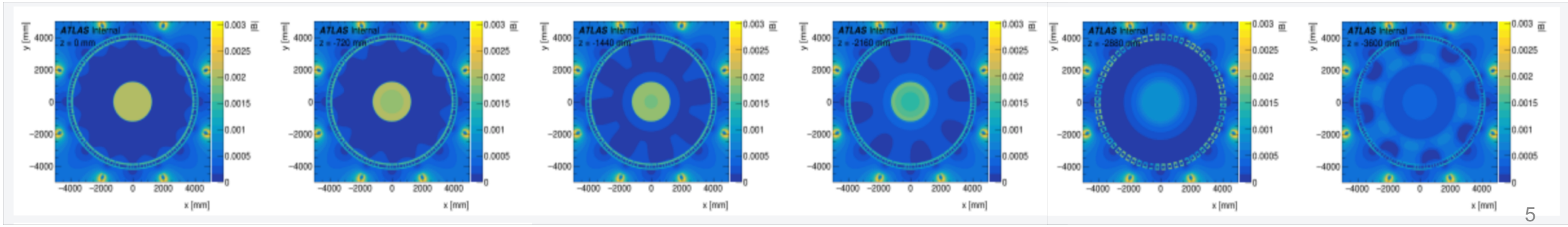
- Geant4 10.5 includes new exceptions for looping particles:
  - Only stable particles are killed if they 'loop' *(i.e. take more than the maximum - default 1000 - integration steps in one physics step)*
  - Warning exceptions with full details of the particle, location and volume are now generated when stable particles are killed (predominantly in **vacuum** or other **low density** media).
  - Controlled by two energy thresholds and a step number parameter
    - 100 MeV (warning energy): below this, tracks are killed silently.
    - 250 MeV (important energy): above this, tracks are given multiple chances (10 physics steps).
- Test of the number of raised exceptions revealed a **very high number** (occurring also on other media and at lower energies --- > simple debugging message left uncommented!

```
10:43:09 -------- WWWW ------- G4Exception-START -------- WWWW -------
10:43:09 *** G4Exception : GeomNav1002
10:43:09       issued by : G4PropagatorInField::ComputeStep
10:43:09  Unfinished integration of track (likely looping particle)   of momentum (-0.257287,-0.356727,-0.876095) (
magnitude = 0.980303 )
10:43:09  after 1000 field substeps  totaling 2521.10548568 mm  out of requested step 43819.9875614 mm  a fraction of
5.753 %
10:43:09  in volume SectionC01 with material Vacuum ( density = 1e-06 g / cm^3 )
10:43:09 *** This is just a warning message. ***
10:43:09 -------- WWWW -------- G4Exception-END --------- WWWW -------
```

Initial tests with Geant4 10.5

- The analysis of this bug reveals where the propagation requires more than 1,000 integration sub-steps:
  - In this cases G4 magnetic field code is spending a lot of cycles to move a track to the end of its physical step
  - This gives us insights that we should be able to leverage to reduce the incidence of such large number of iterations
  - Further investigations revealed that **98%** of these exceptions are caused by muons
    - Likely where the magnetic field is not uniform
    - In Air

Work in progress!



Initial tests with Geant4 10.5

- Can this be related to the **non-uniform field** and the long requested **step size**?
- We tried reducing the maximum allowed step size in volumes where GeomNav1002 exception appears:
  - Number of exceptions in 100 events:

```
default: 17871
10 mm step size: 4628
50 mm step size: 17462
100 mm step size: 18469
```

  - Timing is affected:

```
default: 199960.303 +/- 8125.795 ms
10 mm step size: 211621.515 +/- 8349.587
50 mm step size: 200197.576 +/- 8124.066
100 mm step size: 198711.717 +/- 8076.505
```

- These are GeomNav1002 exceptions caused mainly by muons and presumably the muons are not killed so everything should be fine.
- **Ideas on other tests we could try?**

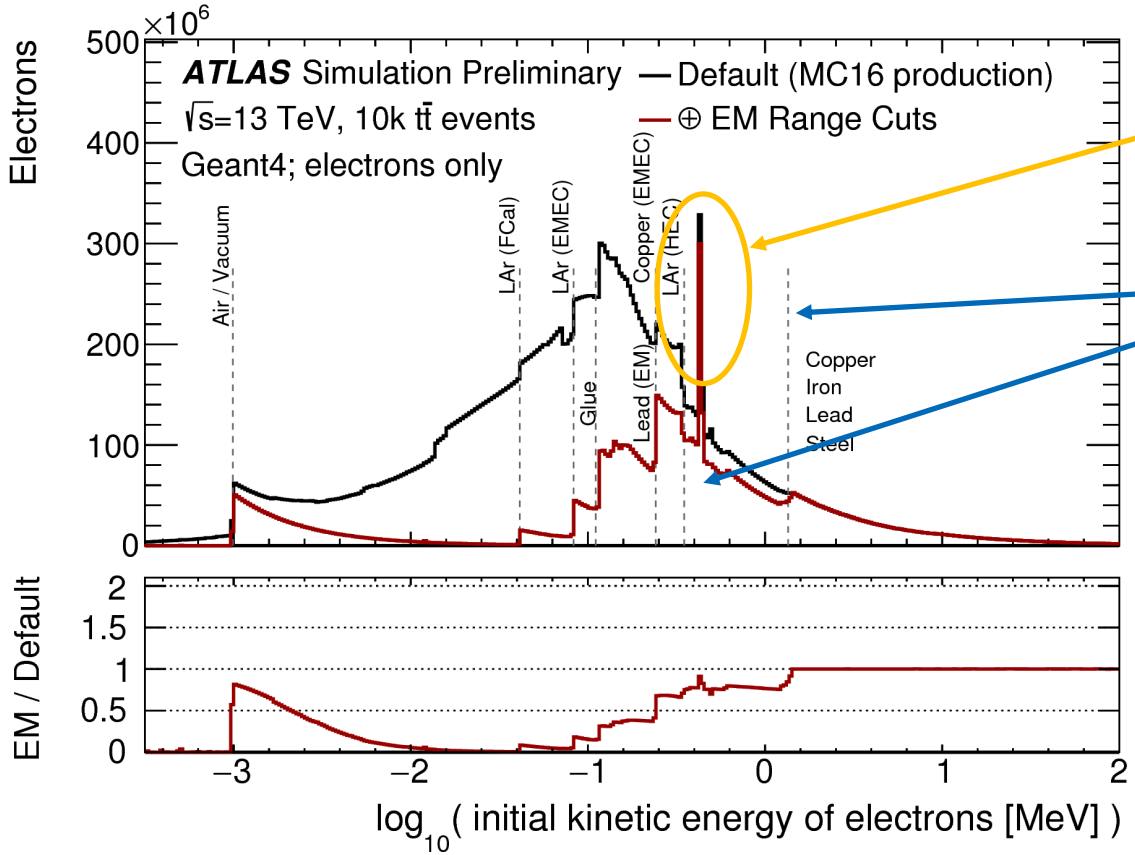
Work in progress!

Test of Geant4 10.4 with VecGeom

- We have now a build of Athena on top of **G4 10.4 with VecGeom**
- First test replacing only **Polycone and Cons**
  - Based on profiling of solid usage, plus VecGeom's Polycone being composed from VecGeom Cons solids.
- Speed up comparing two builds, each with 500 events.:
  - We can see a speed-up with VecGeom in the **1-3% range**
- Now trying a full VecGeom build, to see the full effect.
- After that we can decide whether to do more testing of partial replacements or to try out VecGeom with Geant4 10.5 next.

EM range cuts

- **The total simulation time** is linearly correlated to the **number of steps**
- By default, **range cuts** are off in Geant4 for '*compton*', '*photo-electric*', *and* '*conversion*' processes
- Turning them on drastically decreases the amount of **simulated low energy electrons**



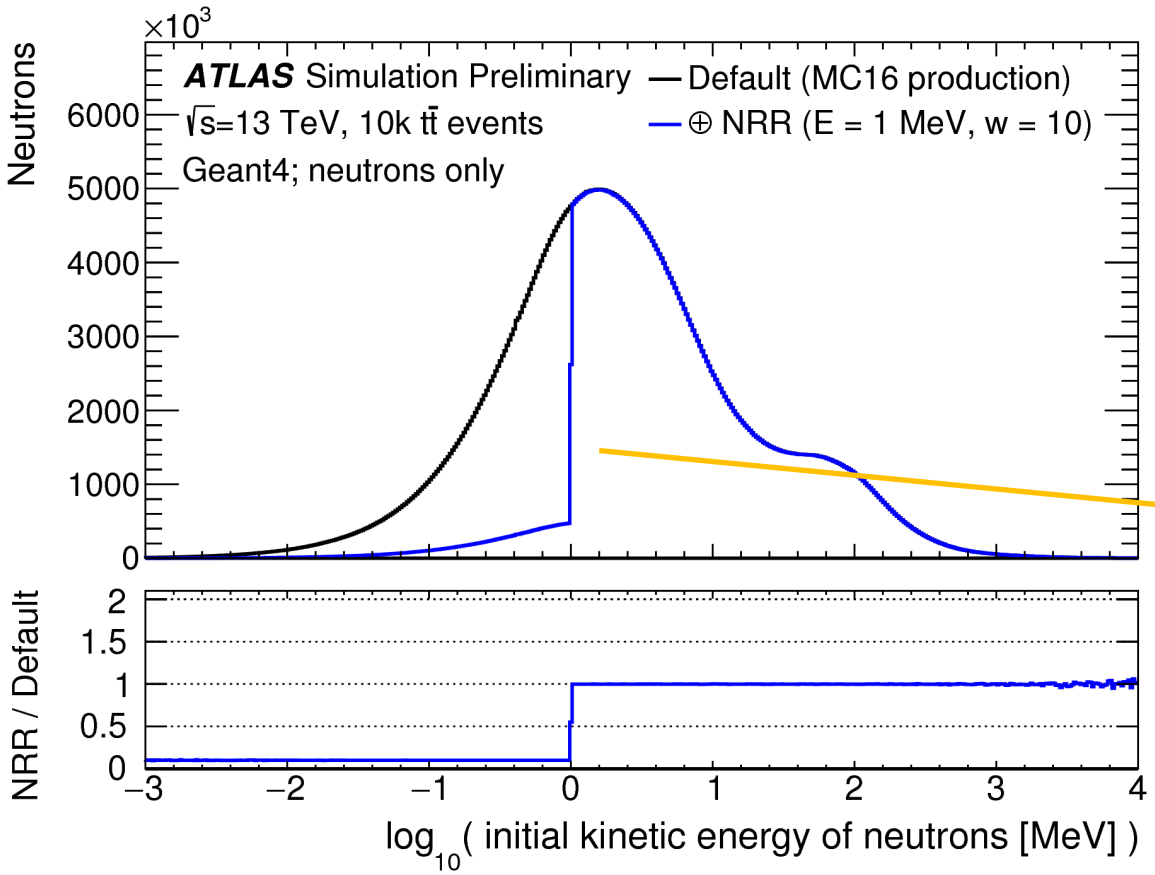
Interesting (!) peak at ~500 keV

Vertical lines corresponding to range-cuts on different materials

- This setting leads to a speedup of **~8%** in total simulation time
- Results on physics validation in the next slides

Neutron Russian Roulette

- Randomly kill neutrons that below some energy and weight the energy deposits of remaining neutrons accordingly *(Eth, w)*

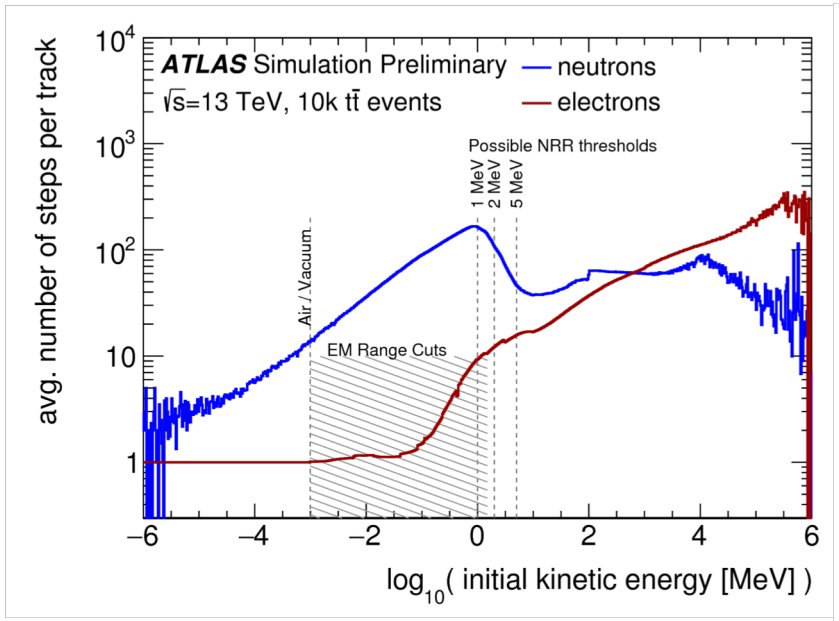


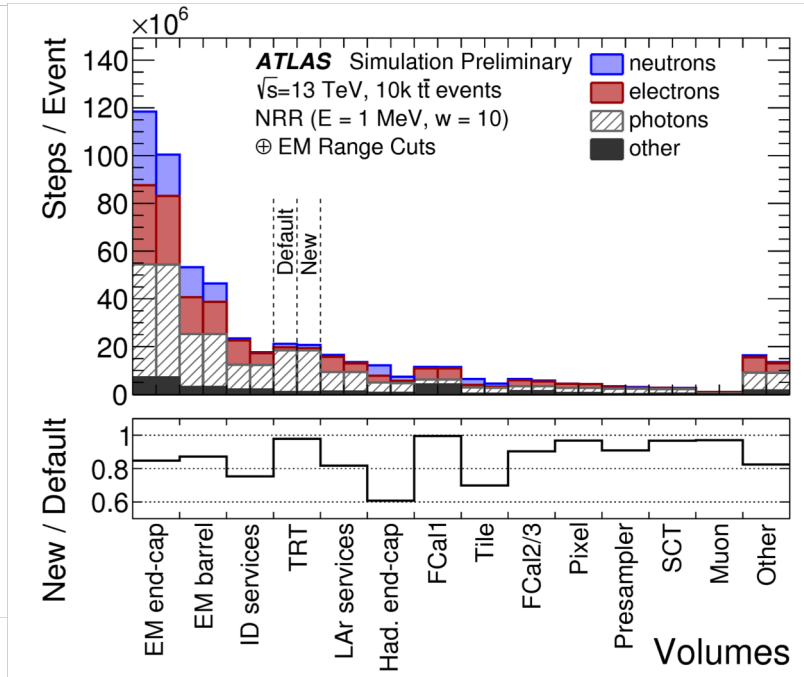
- Neutrons below E_{th} are killed with probability **P((w-1)/w)**.
- If they survive, their weight is set to ***w*** → they will deposit more energy

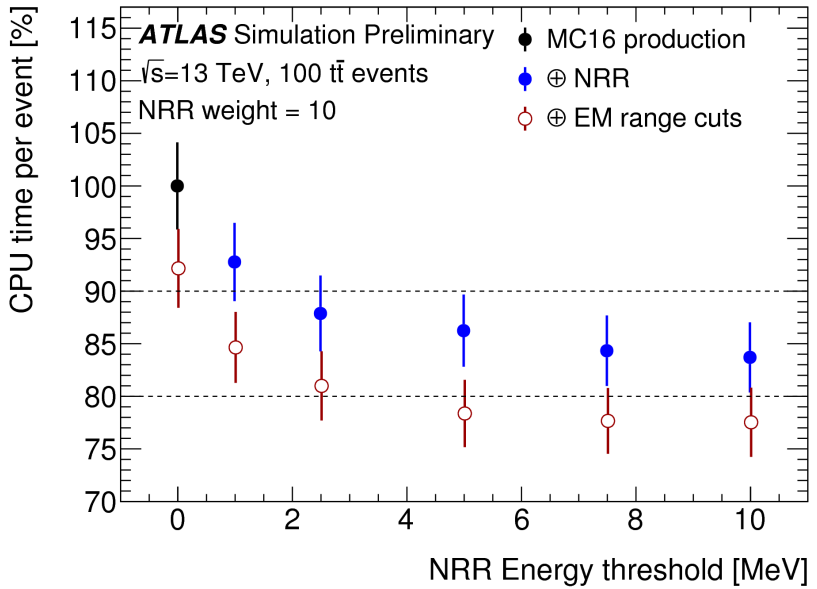
E_{th} at 1 MeV would affect ~40% of all neutrons

EM range cut + NRR

- Threshold at 1 MeV for NRR affect neutrons with the biggest number of steps ☺
  - Increasing the threshold would not bring same benefits ☹
- EM range cut below 1 MeV affects electrons with lower number of steps ☺
- Combining [1MeV,w=10] NRR and EM range cuts, we get **20-40%** fewer steps in calorimeters ☺



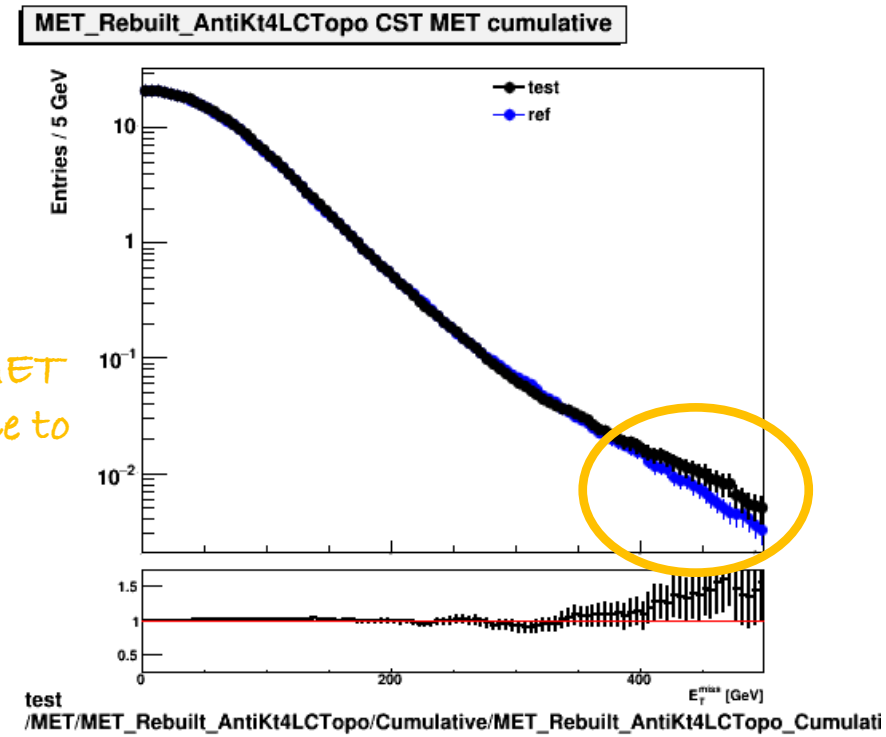


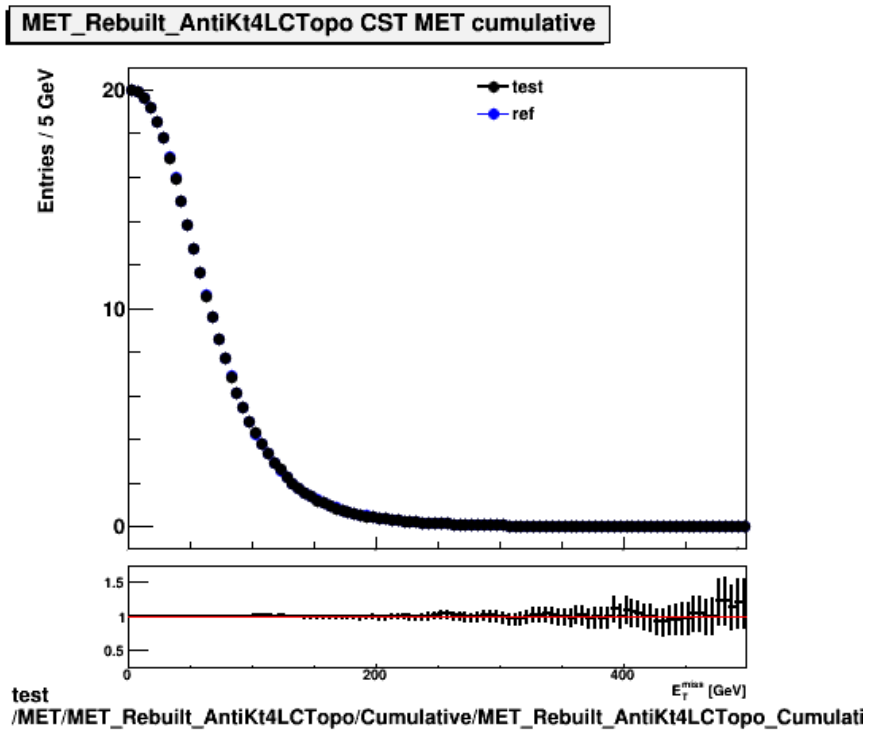


Physics validation

- Neutron Russian Roulette:
  - OK up to a E_{th}= 2 MeV and w=10 for all physics
  - Currently testing with higher thresholds but lower weights (i.e. E_{th} = 5 MeV and w=2)
    - all attempts failed so far
- EM range cuts is almost everywhere ok, some plots still to be understood
  - Additional validation without pileup shows better agreement in MET

Cumulative MET plot, tails prone to fluctuations





Photon Russian Roulette

Preliminary results!

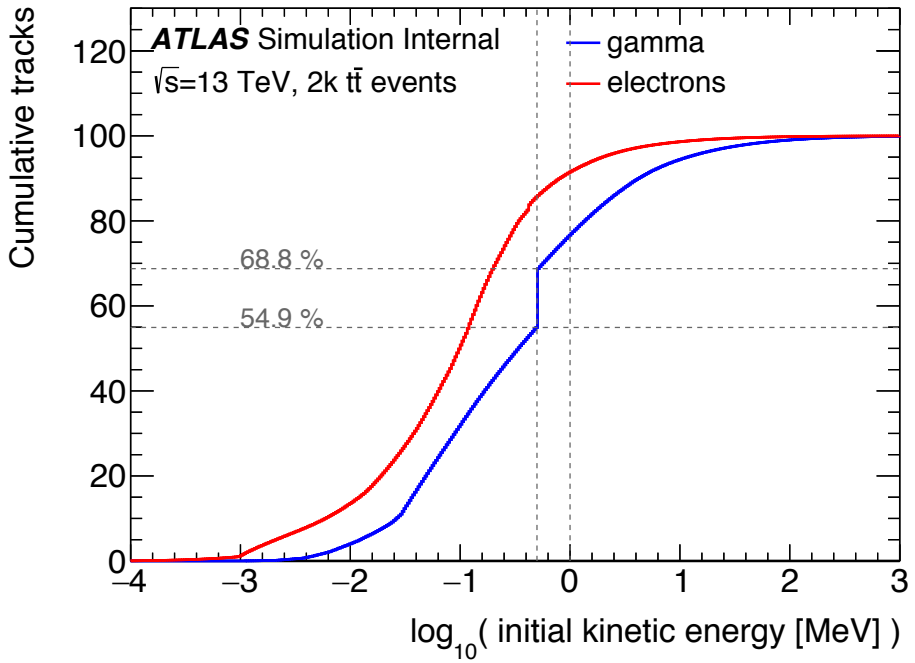
- Started to look into ***Photon Russian Roulette***
  - The idea is to limit it to LAr calorimeters to avoid unwanted effects in ID/TRT.
    - TRT has low- and high-threshold hits, weighted photons could cause high-threshold hits instead of low-threshold hits.

- There is a sharp increase of photons at 511 keV and setting the energy threshold above that brings extra speed-up.

- **Two setups are tested:**
  - test1: E = 1.0 MeV, w = 10, only in volumes named "LAr*"
  - test2: E = 0.5 MeV, w = 10, only in volumes named "LAr*"

- **Preliminary test with 100 ttbar events:**
  default sim: 173778.808 +/- 6752.360
  PRR (1 MeV): 155997.354 +/- 5946.622 (10% faster)
  PRR (0.5 MeV): 167807.253 +/- 6510.748 (3.5% faster)

- Currently opening a physics validation request



AthenaMT & Geant4MT validation

- Improvements in ATLAS Geant4 based simulation are needed especially for the upcoming runs.
- To run effectively on modern architectures a multi-threaded (MT) design is needed
- Multi-threading *(AthenaMT+Geant4MT)* allows for substantial decreases in the memory footprint of jobs with respect to multi-processing jobs *(AthenaMP)* – *better scalability*
  - *Inter-event parallelism* rather than intra-event parallelism
  - Memory savings come from shared geometry & XS tables
- Goal is to validate ATLAS simulation application that has been migrated to a multi-threading processing model in the AthenaMT framework
  - ***Highlights***: *Two very important validation issues have been solved recently (ATLASSIM-4053 and ATLASSIM-4054)*

AthenaMT & Geant4MT validation

- We are able to run full multi-threaded Geant4 within AthenaMT *(AthSimulation 22.0.0)*:

- We can now run a full MT simulation and the **results are validated and consistent** w.r.t. the corresponding sequential one

- Validation of output:
  - Fixed: thread-unsafety causing difference in HITS of LAr sensitive detector (~1-2%)
  - Fixed: thread-unsafety causing difference in HITS of Tile sensitive detector (~1-5%)
  - Now: adding the CaloCalibrationHit to the simulation and testing the behaviour (~50% of Dead material hits)

- Stability fixes:
  - Crashes due to TBB spawning extra threads within the event loop:
    - Not catched by the `ThreadPoolSvc`
    - Crashes when `G4ThreadInitTool::terminateThread` is called for those threads without having called `G4ThreadInitTool::initThread`
    - Temporary protection to avoid dereferencing a null pointer, in place

AthenaMT & Geant4MT validation

- Collection of data races detected in AthenaMT simulation with **Intel Inspector** (ATLASSIM-3990):

ATLASSIM-3991	Data race 1 - read/write		OPEN
ATLASSIM-3992	Data race 2 - read/write		OPEN
ATLASSIM-3993	Data race 3 - read/write		OPEN
ATLASSIM-3994	Data race 4 - read/write		RESOLVED
ATLASSIM-3995	Data race 5 - read/write		OPEN
ATLASSIM-3996	Data race 6 - read/write		OPEN
ATLASSIM-3997	Data race 7 - read/write		RESOLVED
ATLASSIM-3998	Data race 8 - read/write		OPEN
ATLASSIM-3999	Data race 9 - read/write		OPEN
ATLASSIM-4005	Data race 10 - read/write		OPEN
ATLASSIM-4015	Data race 11 - write/write - libRIO.so and AtlasFieldSvc		OPEN
ATLASSIM-4016	Data race 12 - write/write - libG4AtlasAlgLib and libGaudiCoreSvc		OPEN

Description

Data race of read-write type:

```
Description: Read
Source: char_traits.h:262
Function: compare
Module: libstdc++.so.6
Variable: tileVolumeString

Call Stack:
libstdc++.so.6!compare - char_traits.h:262
libTileGeoG4SDLib.so!find - basic_string.h:2025
libTileGeoG4SDLib.so!ProcessHits - TileGeoG4SD.cc:61
libG4tracking.so!Hit - G4VSensitiveDetector.hh:122
libG4tracking.so!ProcessOneTrack - G4TrackingManager.cc:126
libG4event.so!DoProcessing - G4EventManager.cc:185
libG4AtlasAlgLib.so!ProcessEvent - G4AtlasWorkerRunManager.cxx:179
libG4AtlasAlgLib.so!execute - G4AtlasAlg.cxx:325
libGaudiPythonLib.so!execute - Algorithm.h:51
libGaudiKernel.so!sysExecute - Algorithm.cpp:496
libGaudiHive.so!execute - AlgoExecutionTask.cpp:60
```

```
Description: Write
Source: char_traits.h:290
Function: copy
Module: libTileGeoG4SDLib.so
Variable: tileVolumeString

  Code snippet:
   288      if (__n == 0)
   289        return __s1;
  >290      return static_cast<char_type*>(__builtin_memcpy(__s1, __s2, __n));
   291        }
   292

Call Stack:
libTileGeoG4SDLib.so!copy - char_traits.h:290
libTileGeoG4SDLib.so!_ZN8G4StringC4EPKc - G4String.icc:39
libTileGeoG4SDLib.so!ProcessHits - TileGeoG4SD.cc:61
libG4tracking.so!Hit - G4VSensitiveDetector.hh:122
libG4tracking.so!ProcessOneTrack - G4TrackingManager.cc:126
libG4event.so!DoProcessing - G4EventManager.cc:185
libG4AtlasAlgLib.so!ProcessEvent - G4AtlasWorkerRunManager.cxx
libG4AtlasAlgLib.so!execute - G4AtlasAlg.cxx:325
```

Work in progress!

Problem:

```
static const G4String tileVolumeString("Tile");
```

Was not thread-safe, substituted with:

```
static const char * const tileVolumeString = "Tile" ;
```

AthenaMT & Geant4MT validation

- Collection of lock hierarchy violations detected in AthenaMT simulation with **Intel Inspector** (ATLASSIM-4001):

- It happens when two threads are trying to access and lock two critical sections in a different order. Possible deadlock.

Thread 1:

```
Description: Lock owned
Source: gthr-default.h:748
Function: __gthread_mutex_lock
Module: libAthAllocators.so
Variable:  block allocated at ArenaBase.cxx:148

Code snippet:
   746  {
   747    if (__gthread_active_p ())
  >748      return __gthrw_(pthread_mutex_lock) (__mutex);
   749    else
   750      return 0;

Call Stack:

libAthAllocators.so!__gthread_mutex_lock - gthr-default.h:748
libAthAllocators.so!allocator - ArenaBase.icc:40
libGeneratorObjectsTPCnv.so!_ZN2SG21ArenaHandleBaseAllocTINS_18ArenaPoolA
libGeneratorObjectsAthenaPoolPoolCnv.so!createTransient - TPConverter.icc
libGeneratorObjectsAthenaPoolPoolCnv.so!PoolToDataObject - T_AthenaPoolCu
libAthenaPoolCnvSvcLib.so!createObj - AthenaPoolConverter.cxx:68
libAthenaBaseComps.so!makeCall - AthCnvSvc.cxx:565
```

Thread 2:

```
Description: Lock owned
Source: gthr-default.h:748
Function: __gthread_mutex_lock
Module: libAthAllocators.so
Variable: block allocated at ArenaBase.cxx:30
Code snippet:

   746 {
   747 if (__gthread_active_p ())
  >748 return __gthrw_(pthread_mutex_lock) (__mutex);
   749 else
   750 return 0;


Call Stack:

libAthAllocators.so!__gthread_mutex_lock - gthr-default.h:748
libStoreGateLib.so!clearStore - SGImplSvc.cxx:309
libStoreGateLib.so!clearStore - SGHiveMgrSvc.cxx:49
libAthenaServices.so!clearWBSlot - AthenaHiveEventLoopMgr.cxx
libAthenaServices.so!drainScheduler - AthenaHiveEventLoopMgr
libAthenaServices.so!nextEvent - AthenaHiveEventLoopMgr.cxx:8
```

Click to edit

Testing!

Other WIP items/request of feedback

- Allow Geant4 to deal with **zero-lifetime particles** (needed for quasi-stable particle simulation):
  - Athena-side workaround seemed to work in a test of a few events.
  - It is being tested in some larger statistics samples now.
  - Any new feedback on the Geant4 side ?
- Improving the **robustness of commands executed via G4UIManager**
  - Any news on this item ?

Summary

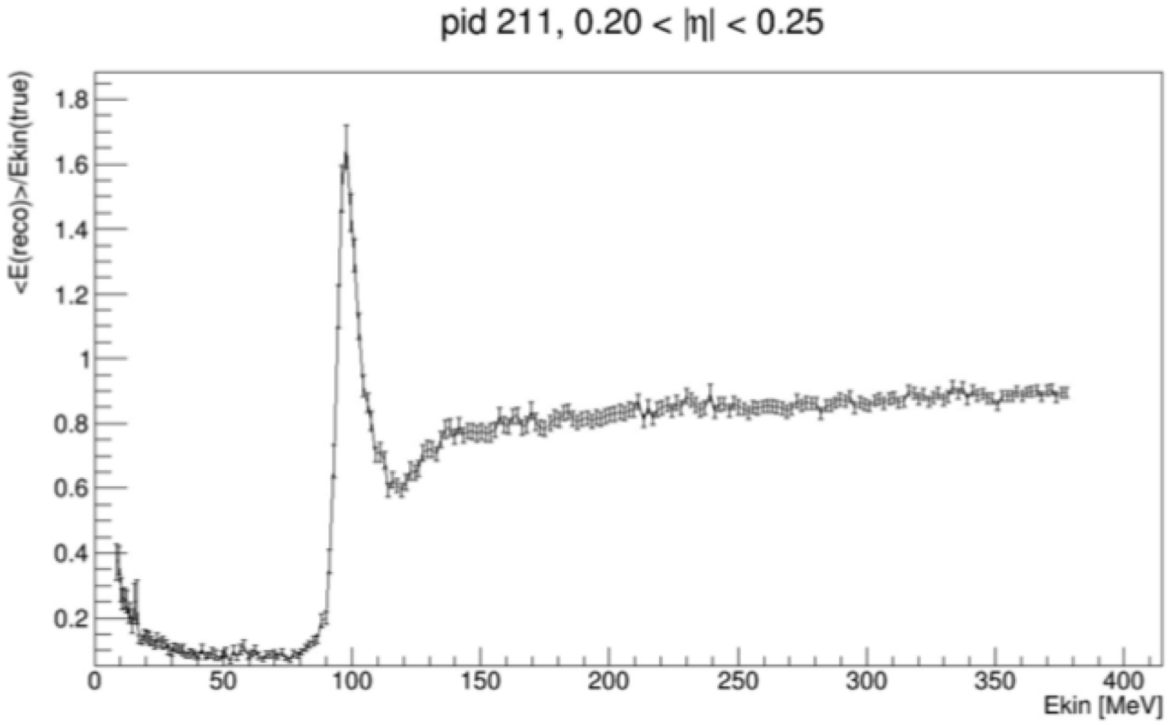
- **Test with VecGeom looks promising:** Polycone and Cons (1-3% speedup)
- **Good progress on Optimizing** ATLAS simulation with Geant4 performance:
  - Range cuts for secondary electrons originating from photons (~8%)
  - Russian Roulette for neutrons (10 - 20%)
  - Russian Roulette for photons (3.5 - 10%)
  - General improvements of the existing code (few %).
  - When the "Big Library" will be in place we expect to see a significant performance increase
- **Good progress on Validation** of AthenaMT with Geant4MT:
  - We can now run a full sequential and MT simulation and the results are fully consistent
  - In order to have a production configuration some items are missing:
    - Frozen Showers code (used in the FCAL) to be thread-safe
    - Check configurations with our extensions to G4 physics are thread-safe
    - Tidying up the `G4AtlasWorkerRunManger` and `G4AtlasMTRunManager` particularly the `RunTermination` methods.

Thanks for your attention.

Marilena Bandieramonte
marilena.bandieramonte@cern.ch

Backup slides

Studies on the Low energy pion “resonance”



Charged pions (especially negative and to some extend also positive) have a surprisingly large energy response when they have a momentum of ~195 MeV (~100 MeV kinetic energy).

This „resonance“ is present for |eta|<1.0

This number is **after sampling fraction correction**. The sampling fraction in the presampler is 0.05, and it is tuned for high energetic showers according to Guillaume → so the energy could be overestimated in some cases.

New hypothesis/explanation:

This is not a resonance. What happens it that incidently the showers at small eta are short enough that they are fully contained in the active material of the presampler. With lower or higher energy, or a different starting point of the shower and different eta, this behaviour changes, and the shower expands to the other layers and energy is lost in inactive material. So at ~195 MeV we are in a „sweet spot“ with an enhanced energy response due to shower containment inside active LAr.

Differences in Tile Hits

- *ATLASSIM-4053*: Differences in the Tile Hits affecting the energy and time:

```
000.TileHitVector_p1_TileHitVec.m_cont.387.m_time.11 115.0 -> 116.0 => diff= [-0.21645022%]
000.TileHitVector_p1_TileHitVec.m_cont.1305.m_time.1 80.0 -> 81.5 => diff= [-0.46439628%]
000.TileHitVector_p1_TileHitVec.m_cont.1310.m_time.1 80.0 -> 81.0 => diff= [-0.31055901%] Py:diff-
root INFO Found [761450] identical leaves Py:diff-root INFO Found [3] different leaves Py:diff-root
INFO [TileHitVector_p1_TileHitVec.m_cont.m_time]: 3 leaves differ
```

- MT-unsafe mutable variable `m_deltaTime` was used to keep granularity in time for hits.
- Now it has been removed as private member in `TileGeoG4SDCalc` and it is calculated and passed directly to clients.

Differences in LAr Hits

- *ATLASSIM-4054:* Differences in the LAr Hits affecting *very rarely* the energy:

```
Ex:
Py:diff-root         INFO comparing [22] leaves over entries...
000.LArHitContainer_p2_LArHitEMB.m_energy.6891 3484255171L -> 1336771523L => diff= [22.27205722%]
Py:diff-root         INFO Found [630943] identical leaves
Py:diff-root         INFO Found [1] different leaves
```

- PsMap class is a singleton and SetMap method was not thread safe.
- This caused data race in the LArBarrelPresamplerCalculator code, as different threads were calling SetMap method at the same time.
- This resulted in differences in the Current values retrieved from the map and consequently in the energy of the LArHitEMB

New segfaults of G4MT jobs

Segfaults during finalization of MT jobs and during execution

- ATLASSIM-4062:
  - When finalizing a MT job, there is a segmentation fault
  - Looks like TBB is creating extra-threads that are not catched by the `ThreadPoolSvc`
  - Crashes when `G4ThreadInitTool::terminateThread` is called for those threads without having called `G4ThreadInitTool::initThread`
  - Temporary protection to avoid dereferencing a null pointer, in place

- ATLASSIM-4078:
  - At some point (42 events have been already processed) TBB starts randomly a thread.
  - In some cases TBB can spawn new threads even after initialization is complete
  - The simulation was aborted because `GeoModel` was released but it is needed to initialize new threads
  - Adding the following option solves the problem temporary
    - `simFlags.ReleaseGeoModel = False`

Workaround in place