Reducing CPU Consumption of Geant4 Simulation in ATLAS

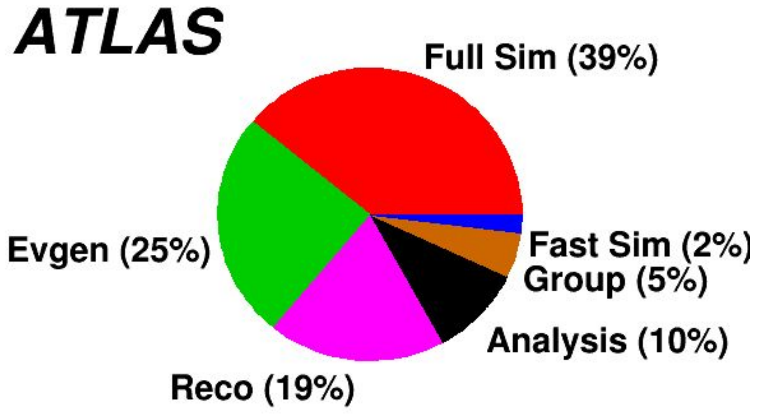
John Chapman (University of Cambridge)
on behalf of the ATLAS Simulation Group

Joint WLCG & HSF Workshop 2018
Napoli, Italy - 28th March 2018

Current Situation

Setting the scene

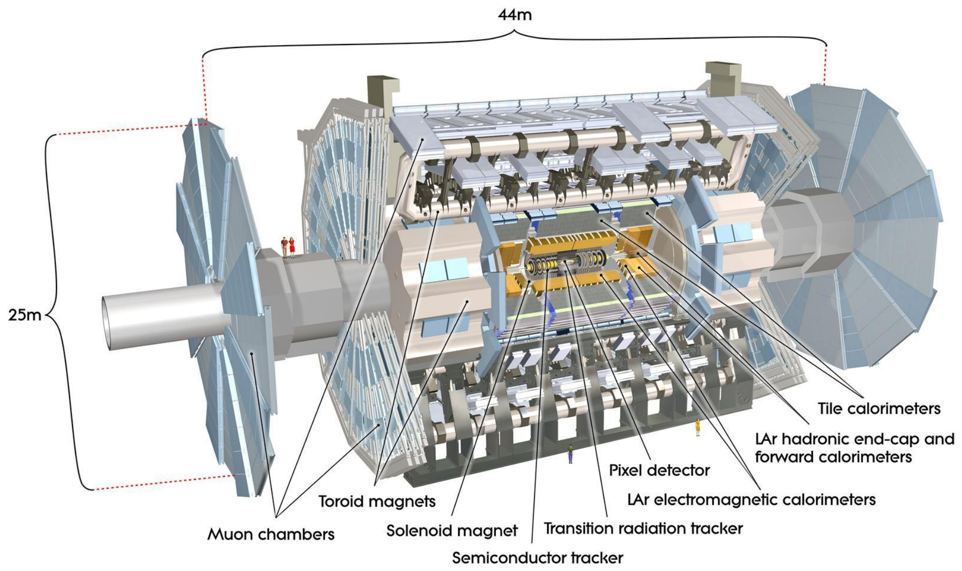
The largest component of ATLAS CPU time is spent on Geant4 Simulation. This will continue to be the case into LHC Run 3 due to the increased simulation demands.
Speeding up our simulation will allow for the production of larger Monte Carlo statistics.
Increasing statistics will increase the precision of physics analyses by reducing the statistical error, with only a slight increase to the systematic error if approximations are used.



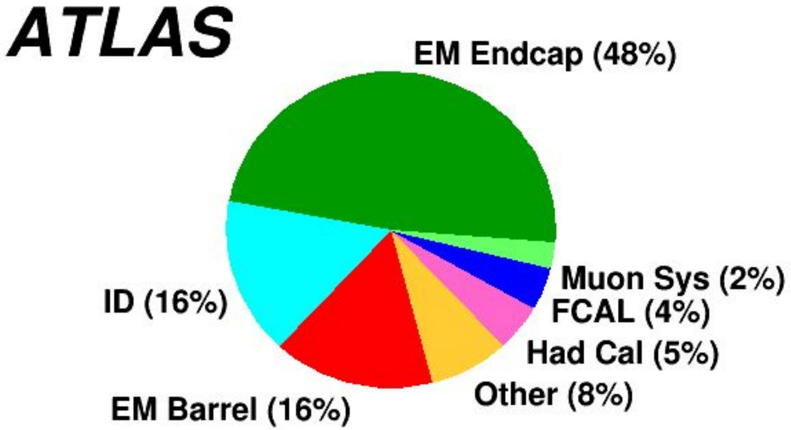
Wall clock time fraction for grid and HPC jobs
July 2015 - July 2016

Setting the scene

The regions of the detector which take the most time during simulation are determined by particle flux and geometry complexity. The EM Endcap dominates (the CPU requirements of the FCAL are suppressed due to the use of Frozen Showers in that region of the detector).



44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Solenoid magnet
Transition radiation tracker
Muon chambers
Semiconductor tracker



Subdetector CPU fraction for 50 ttbar events
MC16 Candidate Release

Speed-ups already in place

- Frozen Showers in FCAL
- AtlasRK4 Stepper
- 250ns neutron time cut
- Only simulate primary particles with $|\eta| < 6.0$

Configuration	MinBias	ttbar
Nominal production configuration: shower libraries in the forward calorimeter, nominal range cuts, NystromRK4 stepper, FTFP_BERT_ATL physics list, 250ns neutron time cut, simulation of primary particles with pseudo-rapidity below 6.0	1.0	1.0
No shower libraries	1.5	1.3
ClassicalRK4 stepper instead of NystromRK4	1.09	1.07
No neutron time cut	1.02	1.01

Table 4: Performance of various configurations of the ATLAS simulation for minimum bias and ttbar production events. The Geant4 version used for this test was G4 10.2p03. No significant performance improvements were introduced in patch 03 with respect to patch 02. (Taken from Detector Simulation White paper.)

Improvements under–investigation

ATLAS Strategy

There are two ways to speed up the simulation:

1. **Do the same thing, but faster.** (Simulation output unchanged.)
2. **Do something simpler.** (Simulation output changes.)

Speed-ups in category 2 require much more careful validation, so we prefer to focus on improvements in category 1 first.

Category 1:

- Improved G4Solid implementations.
- Optimize choice of G4Solids used to create given volumes.
- Big Library (static-linking)
- Profile-guided optimization.

Category 2:

- Geometry Simplification?
- Hadronic cross-section tables
- More aggressive G4 cuts.
- Russian roulette for neutrons.

Will discuss some of these on the following slides.

Detector geometry optimization

There are two principal directions in the detector geometry optimization:

- Enhancement of Geant4 solids by taking into account the specifics of the shapes used in the detector geometry description.
- Simplification (optimization) of the detector geometry description itself.

Detector geometry: Main shapes

Main shapes used in the ATLAS geometry description

- **Box** – 11814 solids
  - 7756 boxes have volume < 1 cm^3
  - of those 472 have volume < 1 mm^3
- **Trd** – 26310 solids
  Only one(!) Trd solid has general shape, all others have specific shapes:
  - 9331 are boxes, that can be easily detected and used in GeoModel
  - 16944 have parallel X sides
  - 34 have parallel Y sides
- **Tube** – 6774 solids
  - 798 tubes have volume < 1 cm^3
  - of those 123 have volume < 1 mm^3

Elimination of small (thin) objects may give essential improvement in performance. For example, **elimination of the glue layers gives up to 30% speed up in the simulation of the Tile calorimeter.** (Effect on physics to be evaluated.)

Recent Geant4 10.4 provides revised implementation of G4Box, G4Trd and G4Trap. Revised G4Trap has several internal specializations. Similarly, specializations can be introduced for G4Trd.

Revised solids have been used in the ATLAS custom version of Geant4 based on Geant4 10.1.p03. **It gave ~4% improvement in the performance.**

Detector geometry: Boolean volumes

Boolean solids do not have internal optimisation. The time required for calculations in boolean solids scales at best with the number of components, often worse than this in fact because of rather complicated logic of the calculations.

Received wisdom in ATLAS was that descriptions using Boolean Solids were more robust than equivalent descriptions using Extruded Solids.

This is no longer the case and **Extruded Solids are usually more performant.**

Below is a list of boolean volumes in different parts of the Atlas detector:

- **Inner Detector** – 465 boolean volumes
  - 3 volumes are composed of 42 components, others are composed of 2 – 11 components.
- **Tile Calorimeter** – **15863** boolean volumes
  - Almost all boolean solids can be replaced with Extruded Solids. (In progress)
- **Muon System** – 2869 boolean volumes
  - >170 volumes have 30 - 82 components.

Detector geometry: Overlaps

The existence of overlaps in the detector geometry description means that there are some issues in the geometry definition that may lead to incorrect result of the simulation.

Geant4 tracking algorithms are quite sensitive to overlaps, which can be the cause of various issues during tracking. I.e. skipping of material, stuck tracks, etc.

Below is the result of overlap check in the GDML dumps:

- Inner Detector – 1983 overlaps
- Calorimeters – 172 overlaps
- Muon System – 332 overlaps

A number of fake overlaps have been observed in the Tile calorimeter geometry.

If the result of a boolean operation (subtraction or intersection) is a null object, the Geant4 overlap check may report an overlap. In such a case the warning message on overlap will be preceded by a warning message on a failure to pick up a random point on the surface of the object.

Null objects do not create problems for tracking, however it indicates that the geometry description can be improved.

4%

From new G4Solid classes.

Potentially more from both optimizing the choices of G4Solid classes used and from simplifying the description.

Big library

Geant4 stand-alone applications shows 20% improvement with static builds (the larger the number of threads the higher the benefit).

Got even better with KNL (interesting for CORI).

Cannot use a full static build w/ Athena, but:

1. Compile G4 in “static mode” (e.g. *.a libs)
2. Create a single large *.so library with all athena packages that depend on G4 and libG4*.a

D. Smith (CERN IT) has shown that a **~10% gain** can still be achieved with grouping all code in a single .so library.

Similar strategy, based on this work, has been already adopted by CMS with said **10% gain**.

Big library: Strategy

1. **Identify** all packages which depend on G4.
2. **Compile** G4 external in static mode (e.g. create .a archives).
3. For all the packages identified by (1) modify CMakeLists.txt to **create** a cmake library of OBJECT type.
4. Create a meta-package that will create the *big library* using all OBJECT libraries and static link from G4.

PkgA/CMakeLists.txt

```
atlas_subdir( PkgA )
atlas_add_library( PkgA
     src/*.cxx
     [...]
     OBJECT )
// ...
```

PkgB/CMakeLists.txt

```
atlas_subdir( PkgB )
atlas_add_library( PkgB
     src/*.cxx
     [...]
     OBJECT )
// ...
```

PkgC/CMakeLists.txt

```
atlas_subdir( PkgC )
atlas_add_library( PkgC
     [...]
     OBJECT_LIBRARIES
     PkgA
     PkgB )
// ...
```

~10%

Improvement should be orthogonal to others.

Study of AutoFDO in ATLAS Simulation.

AutoFDO (Feedback-Directed Optimization) is an optimization technique available in Linux which provides performance gains and is able to collect profile data on production system.

It uses sampling - based profile collected using the perf tool to drive feedback directed optimizations.

A standalone tool is used to convert the perf.data file into gcov format. gcc reads in the gcov file and interprets the profile into a set of hashmaps.

The use of this tool has already been investigated in CMS simulation workflows and has shown a **10% improvement** (NB using statically-linked libraries).

Aiming to achieve at least a 5% improvement in simulation time in ATLAS Simulation. In that case it will be considered for use in production release builds.

The official version of AutoFDO did not seem to work for us, so we are using our own patched version.

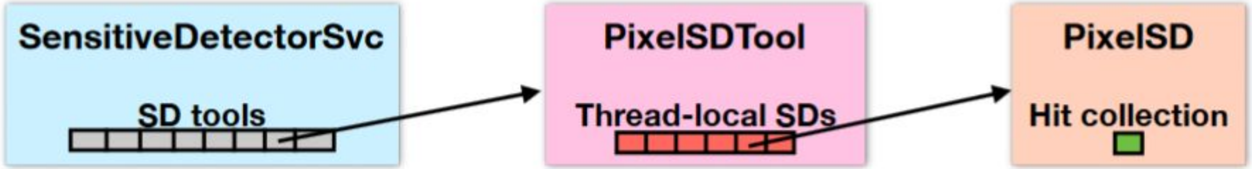
Hope to have some results soon.

Geant4MT in AthenaMT

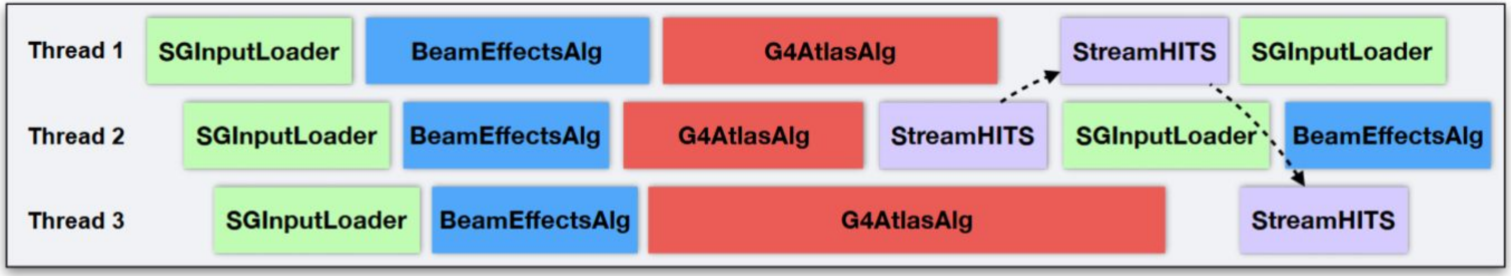
- Chip manufacturers are producing evermore highly parallelized devices - we have to take advantage of this.
- ATLAS can now run a complete simulation setup using Geant4MT in multi-threaded Athena (AthenaMT).
- Now in the process of stress-testing the code to check for race-conditions and evidence of non-reproducibility.

Geant4MT in AthenaMT: Application details

- Composition model of Gaudi/Athena components that create + manage thread-local G4 components



- Thread-local workspaces setup via new thread initialization tool framework mechanism
- Processing algorithms are cloned to execute on different events concurrently
- One instance of the output stream algorithm (StreamHITS) services all worker threads. Now being replaced by a SharedWriter which can handle multiple threads in parallel to avoid a bottleneck.



Future/ Other work

- Optimizations which will affect physics results:
  - More aggressive Geant4 cuts
  - Switch of B-Field in Calo (except for Muons).
  - Russian roulette for neutrons
  - Fast hadronic cross-section lookup with precomputed mapping information.
  - Geometry simplification.
- May not be possible if precision loss is too high.
- Manual tuning like strength reduction: division→multiplication, sin()→polynomials
- In parallel with speeding up Geant4 Simulation in ATLAS, we are working hard on an improved Fast Calorimeter Simulation (FastCaloSim V2) and a Fast Chain incorporating fast simulation, fast digitization and fast reconstruction techniques. (This would be a whole talk itself.)

Summary

- ATLAS is pursuing a number of strategies to speed up our Geant4 Simulation jobs.
- Working with Geant4 experts we are incorporating G4Solid improvements into our production version of Geant4. This should gain us 4%.
- The Big Library concept (all Athena code with Geant4 dependencies is compiled into a single library and statically-linked against Geant4) should gain us an additional 10%.
  - Will go into the current production release when possible.
- Profile-guided optimization using AutoFDO is being investigated. This could potentially gain us another 10%.
- Optimizing how our detector geometry is constructed could gain a few more percent.
  - E.g. Removal of Tile Calorimeter glue volumes could save ~1% overall.
  - This will be for Run 3 campaigns though.
- Geant4MT working in AthenaMT
  - This will be for Run 3 campaigns, but we are testing it on the grid already.
- Many other ideas, which will change physics output and therefore need to be considered more carefully.
- Alternative Fast Calorimeter Simulation actively in development and Fast Chain option to also being developed.

Thanks to...

- A. Dotti, S. Farrell, C. Macron, E. Tcherniaev and Y. Wang for providing material for this talk
- Others in the ATLAS Simulation Group for helpful suggestions.

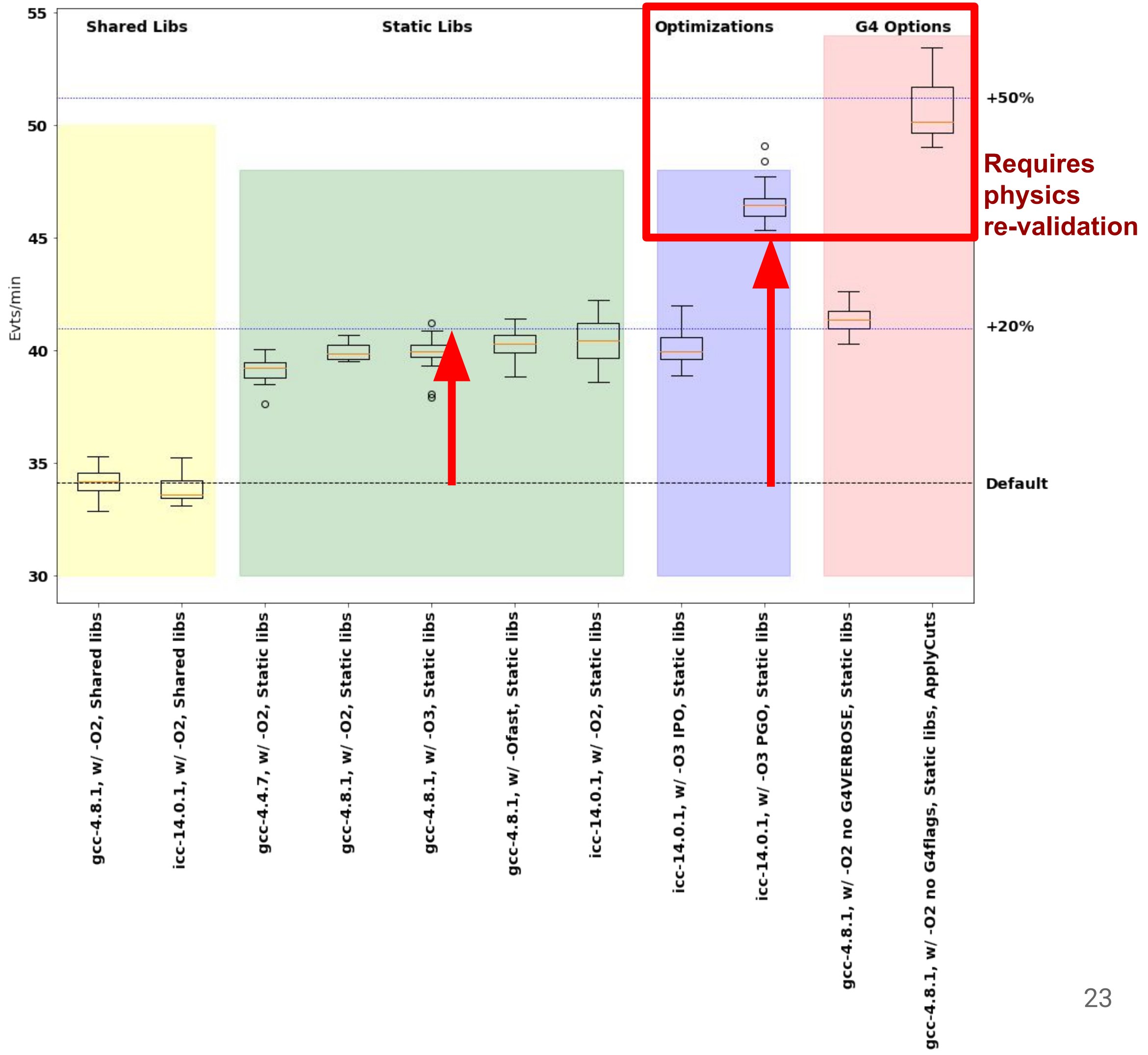
Backups

Reminder: Why we do that?

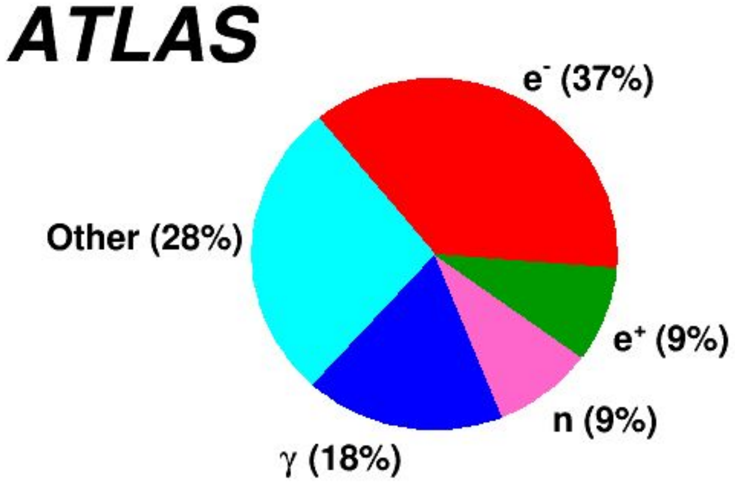
Static builds provide +20% performances

Cannot use w/ Athena a full static build, but:

1) Compile G4 in “static mode” (e.g. *.a libs)
2) Create a single large *.so library with all athena package that depends on G4 and libG4*.a

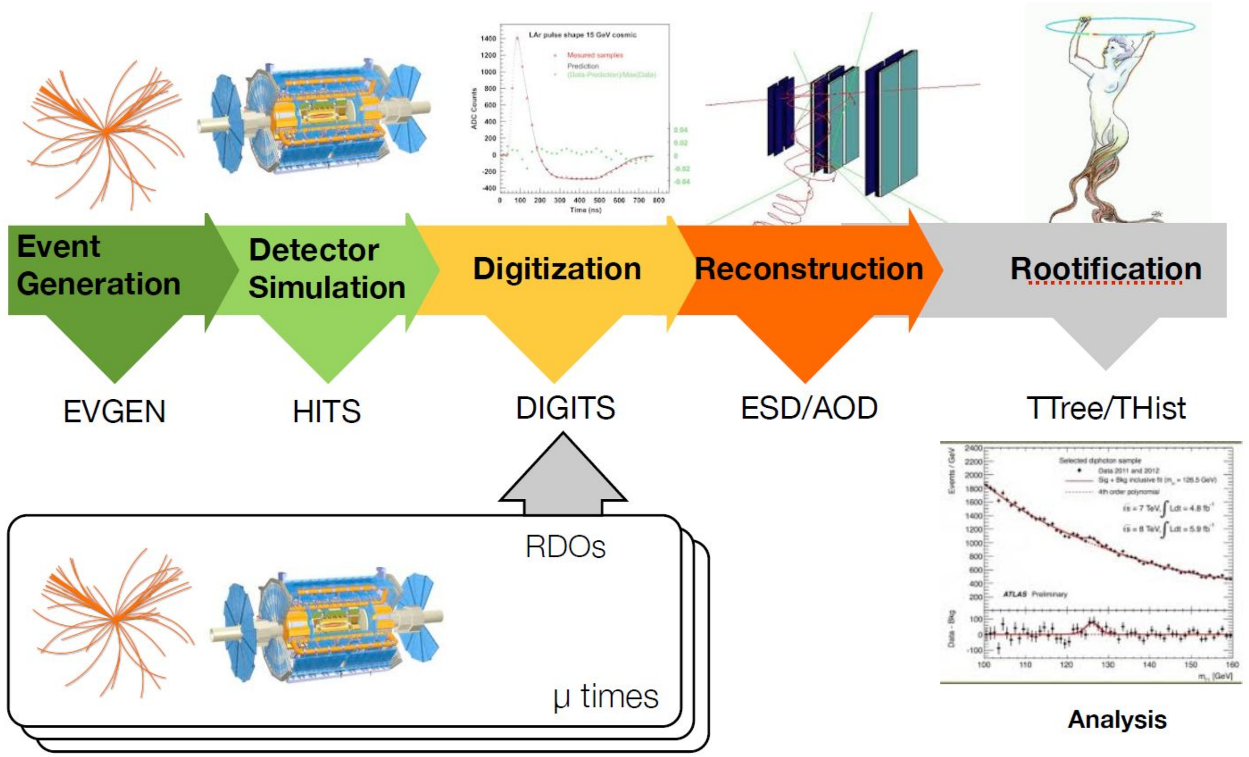


CPU Fraction by particle species



Particle species CPU fraction for 50 ttbar events
MC16 Candidate Release

Monte Carlo Production Chain


LAr pulse shape 15 GeV cosmic
Event Generation
Detector Simulation
Digitization
Reconstruction
Rootification
EVGEN
HITS
DIGITS
ESD/AOD
TTree/THist
RDOs
μ times
ATLAS Preliminary
Analysis