Fast Simulations in ATLAS

Zach Marshall (LBNL) for ATLAS

GeantV / Fast Sim Working Meeting

26-27 May 2016

Time in the Simulation

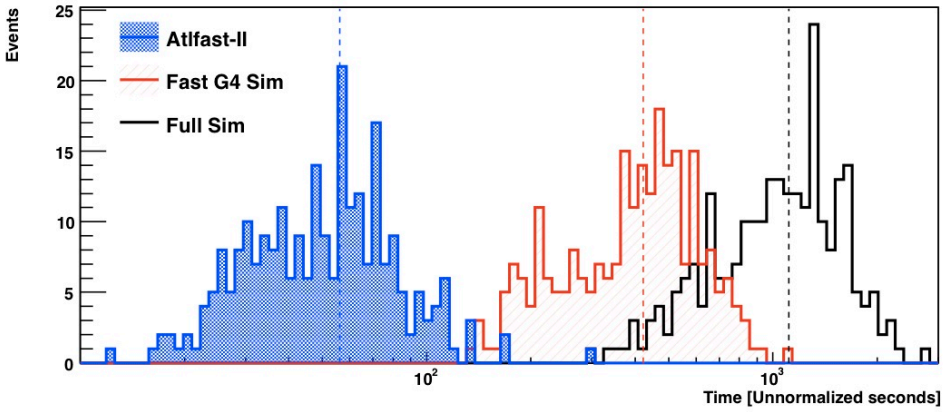
- Most time spent moving low E particles in the calorimetry
  - e/γ below 1 MeV
- Next most time spent moving other particles in the calorimetry
- Next is moving particles in the muon system and tracker
  - In the muon system most time is spent on neutrons from calorimeter showers; once the calorimeter fast sim is used that time goes way down

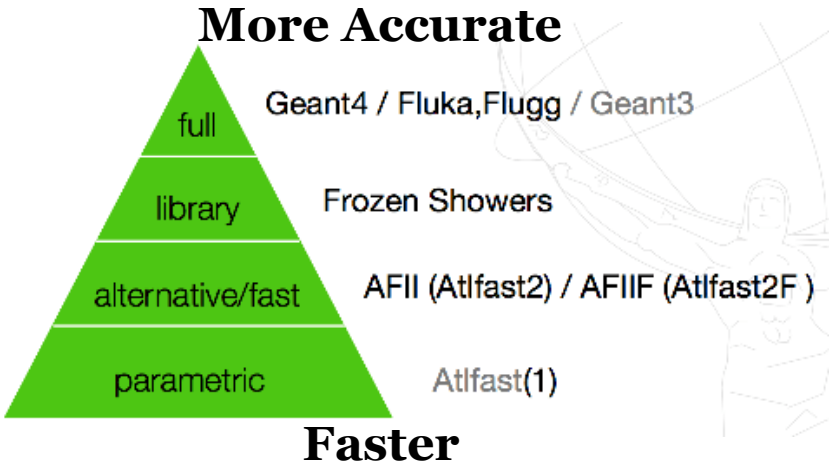
- This dictates our priorities for fast simulation

- Once simulation is sufficiently fast, other steps (digitization, reconstruction) are a significant fraction of our CPU budget, so we also work on fast digitization and fast reconstruction

The Full Program

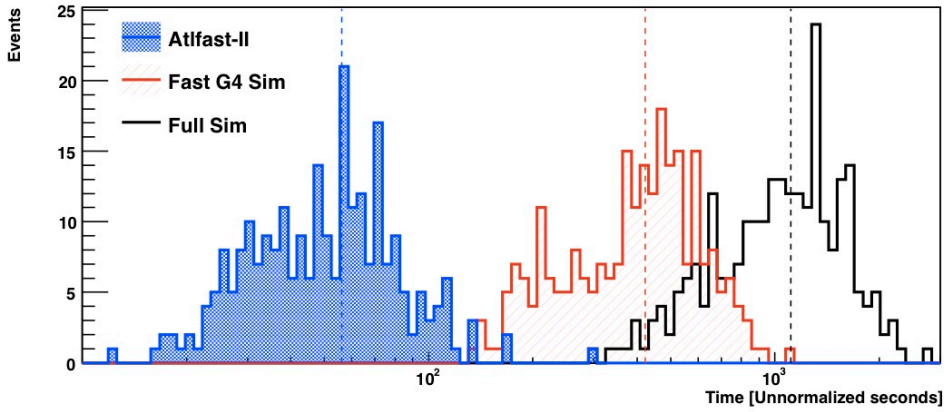
- Frozen Showers
  - "Shower library" (pre-simulated showers) for low energy (<1 GeV) electrons and photons.
- ATLFAST-II aka FastCaloSim
  - Parameterization of calorimeter showers – histograms for e/γ/h, deposits in the readout structure (faster than using the detailed geometry).
- ATLFAST-IIF aka FATRAS+FastCaloSim
  - Adds a fast simulation of tracking in the ID and muon system with simplified geometry.
- ATLFAST-I
  - Mostly abandoned. Simple EVNT input to four-vector output with parameterized smearing.

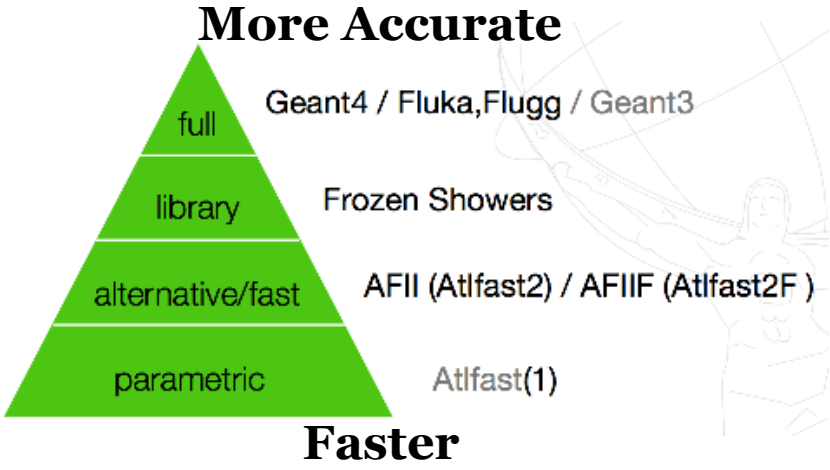




The Full Program

- Frozen Showers
  - Part of our DEFAULT simulation! Everybody uses this unless they specifically ask not to.
- ATLFAST-II aka FastCaloSim
  - Our STANDARD fast simulation. Common for signal models and large backgrounds in Run 1; several billion events.
- ATLFAST-IIF aka FATRAS+FastCaloSim
  - VERY popular for potential **upgrade** studies. Moving quickly for phase-2 upgrades…
- ATLFAST-I
  - Considering revitalization. Don't want to "just" reinvent Delphes/PGS.





ATLAS Framework

- ATLAS uses Gaudi as our framework
- Simulation algorithm at the top
  - Controls the ATLAS stack
- Simulator tools owned by the algorithm
  - Algorithm deals out particles to the appropriate simulator
  - Allows us full control over which particles are simulated with what precision
  - Geant4 is one of the simulators

Option #1: Frozen Showers

- Operates as a proper G4 fast simulation
  - Takes over a track with the right PDG ID, energy, and detector region
  - Configurable – by default in only the forward calorimeter
  - Kills the track
  - Places a number of energy spots in the calorimeter SDs
- Called many, many times
  - Reduces overhead via isApplicable and region checking
- Probably would not do this differently if we started over
- Pre-simulated showers in the same setup, stored in a custom ROOT format, loaded and applied during the job
  - Could imagine sharing specific aspects (e.g. shower format), but it quickly becomes framework-dependent

Option #2: FastCaloSim

- G4UserSteppingAction used to find tracks where the fast simulation will be applicable
  - Kills them, passes them up to the ATLAS stack
  - Second tool later restarts using the original simulation output
- Very flexible
  - Easy configuration to allow Geant4 to keep control of simulation for muons, or for particles in a cone, or …
  - Can also do punch-through (restart particles in the muon system)
- Probably could have been done as a G4FastSimulation
  - Technical reasons when we were starting why this was difficult or even impossible, but most of those have been resolved
- Shower energy deposited into calorimeter according to rather complex shape parameterization
  - Energy goes into cell granularity (coarser than geometry granularity)
  - Shapes parameterized in layers (e.g. EM1, EM2)
  - Pretty specific to the experiment. Some common setup (studies of shower shapes, applications of ML) would be interesting, but very little code to share unless we are in the same infrastructure and use similar EDMs.

Option #3: FATRAS

- Fast track simulation using a simplified geometry
  - Geometry uses a density map on simple cylinders; can G4 natively do this at the moment?
  - Extrapolation with the ATLAS extrapolator rather than a G4 stepper
- Currently uses custom (parameterized) EM physics models
  - Faster, coarser, and with higher cut-offs than the G4 models
- Hadronic physics models from Geant4
  - Ideally this would use a simple call to a Geant module in unit-test style; at the moment it's a bit more complicated than that
- Parts of this could generalize
  - Could have been written as a G4FastSimulation module, probably
  - Density-mapped geometry could be generally used by experiments; we would benefit from common tools to map from complex geometries to simple geometries
  - Fast physics modules and use could be generic
  - First attempts are in place: **https://gitlab.cern.ch/acts/acts-fatras**

Note: Pileup

- Traditionally we have overlaid pileup events at the *hits* level
  - Our understanding is that CMS typically does something like this
- Once the simulation is fast enough, it is better to save the disk space and generate and simulate pileup on the fly
  - Our understanding is that LHCb currently does something like this
- It might be possible to share ideas about how to best do this
  - Particularly in a multi-threaded context
  - How best to mark events and hits as pileup
  - How to deal with truth information
  - How to efficiently pass information around
  - What should constitute a 'Geant event'
- Sharing actual code depends on the framework
  - Here we might be able to share more with LHCb…

Other Ideas

- We are resurrecting a parameterized simulation for some uses
  - This relies on our distribution via the particle stack currently
  - Obviously this is a detector-specific smearing, and part of the gain is going 'directly' to final-state objects – ATLAS tracks and so on, which is not general
- We've talked about the possibility of using generative neural networks for fast simulation, or even for faster Geant modules
  - Option 1: use a NN for a specific item (e.g. a cross section or a single interaction) and share the NN between experiments
  - Option 2: use a NN for a general item (e.g. showers in the calorimeter) and share the infrastructure, and technique between experiments
- Just a brief note on multi-threading models while we're here…
  - Geant4 and Gaudi have chosen rather different multithreading models (Master-Worker vs Task-based) that aren't super easy to make play well together
  - If GeantV is going to be a re-write, it might be wise to think about that issue early (now) rather than re-discovering it a few years from now

Summary

- We have a number of different fast simulation options
- Our framework based on owning its own stack allows us full flexibility over the integration of those fast simulations
  - We could have chosen to integrate this more tightly into Geant4 and use its stack for this work, so that Geant4 itself would be our fast simulation framework – this was an early design choice
- Most fast simulations are pretty specific to our detector
  - Some elements of the fast simulation infrastructure could be common, and knowledge certainly could be shared between experiments
  - In many cases it is the detector-specific assumptions that make these fast – the most code that could be shared is a generic base class like G4 has
- Several ideas about new fast simulations
  - Some of these could have shared aspects, and some could even be used directly inside of Geant4 or GeantV (in fact, no reason these neural net ideas could not be integrated inside of both)