Fast Simulations in ATLAS

Zach Marshall (LBNL) and Andrea Dotti (SLAC) for ATLAS

Simulation CWP Meeting

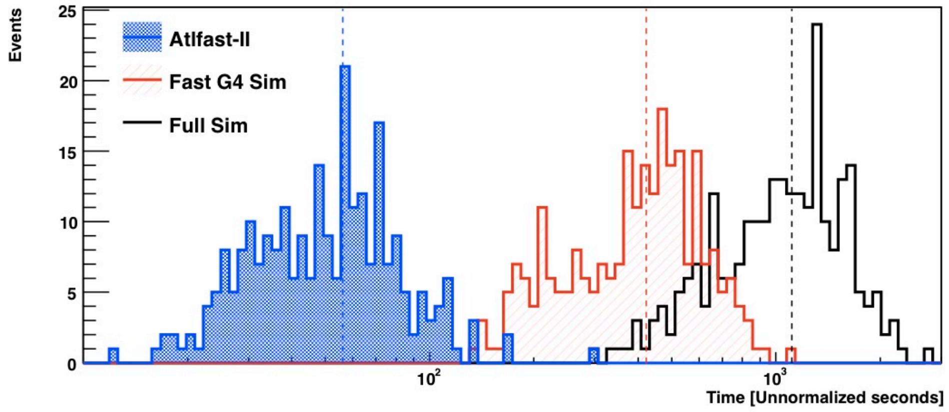
27 March 2017

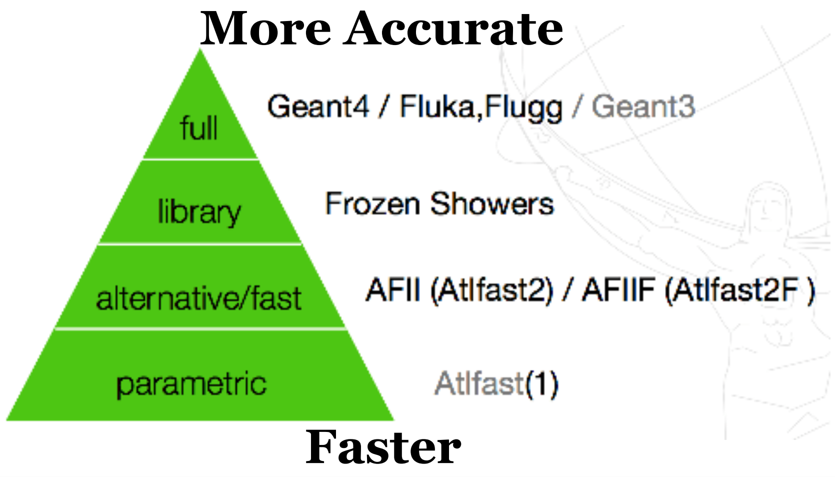
Big Picture

- >90% of ATLAS simulation uses *some* fast simulation
  - Most is at the level that users don't even realize it's there
- About 30-40% of ATLAS simulation uses a *serious* fast simulation with a time gain of >10x
  - Most *events* are large Standard Model backgrounds that need to be useable for all analyses, so the adoption threshold is very high for people to use fast sim for those large backgrounds
  - BSM physics samples are mostly done with fast simulation
- Goal for ATLAS is to have >50% done with fast simulation
- We also want to have a *very* fast simulation – a good tool!

The Current Program

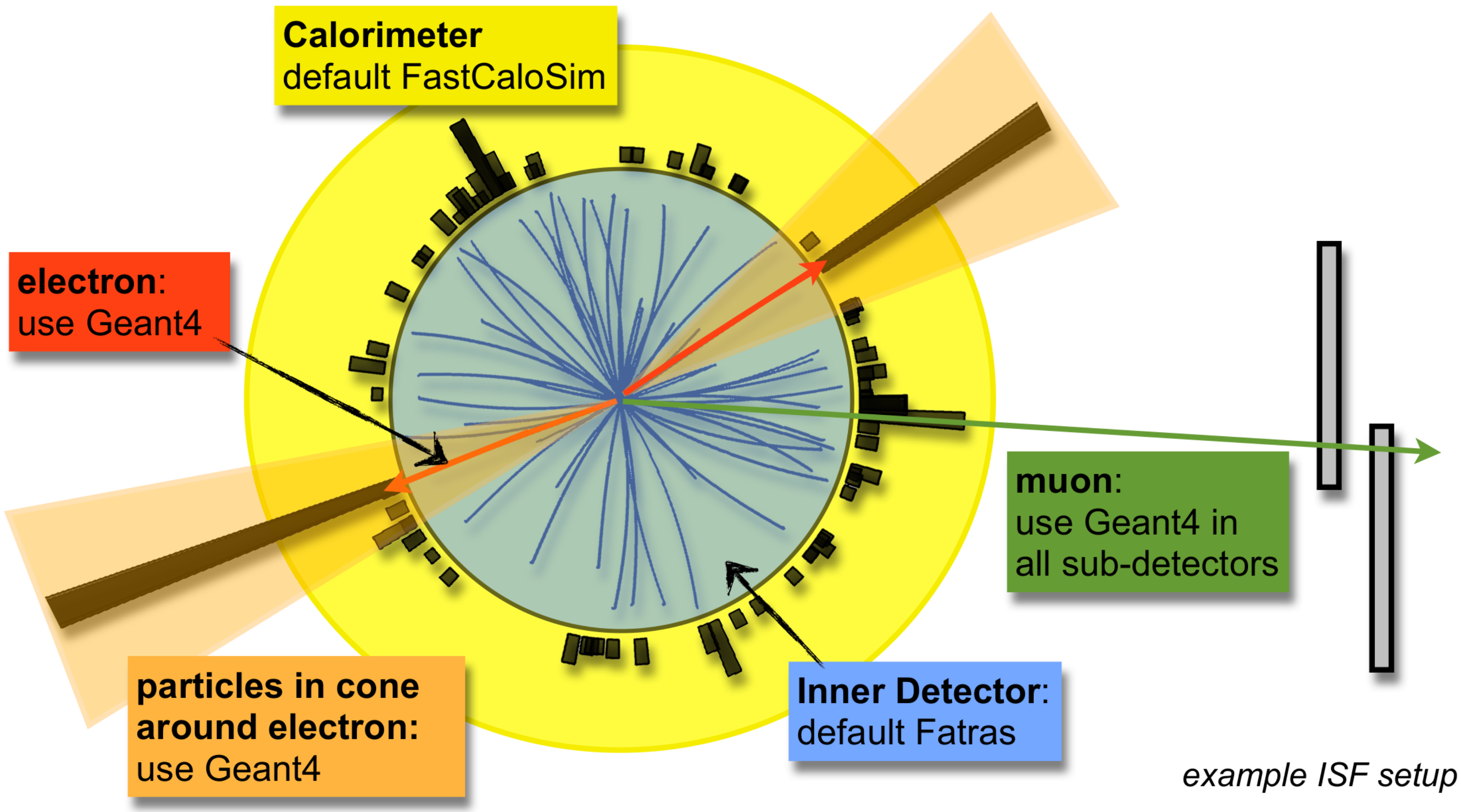
- Frozen Showers
  - "Shower library" (pre-simulated showers) for low energy (<1 GeV) electrons and photons. Part of our **default** simulation! Everybody uses this unless they specifically ask not to.
- ATLFAST-II aka FastCaloSim
  - Parameterization of calo showers – histograms for e/γ/h, deposits in the readout structure (faster than using the detailed geometry). Our **standard** fast simulation.
- ATLFAST-IIF aka FATRAS+FastCaloSim
  - Adds a fast simulation of tracking in the ID and muon system with simplified geometry. VERY popular for potential **upgrade** studies. Moving quickly for phase-2 upgrades…
- ATLFAST-I
  - Mostly abandoned. Simple EVNT input to four-vector output with parameterized smearing. Shouldn't "just" reinvent Delphes/PGS.





How Integrated Are They?

- We normally think of using a single simulation type per sub-detector, but we could divide the event up differently, e.g.:



example ISF setup

Enter the ISF

- The ISF is ATLAS's solution to providing a **single interface**
- Allows us to mix events in complex ways
  - This is already causing some struggles in terms of understanding object calibration and performance recommendations for users. If this cannot be overcome, then this is just an expensive toy ☹
- Chose to not depend on Geant4 at this top level
  - Means that fast simulations don't depend on Geant4
  - Originally we were arguing about not pulling in large G4 dependencies, but our application is large enough that this might not be a real concern (and of course this would be a VMEM worry but not an RSS worry)

Fast Sims in Geant4

- An incredible number of physics models are available in G4
- Some fast simulations would benefit from using them
  - Generating the secondary photon from brem spectrum
  - Generating secondaries from a hadronic interaction
  - Looking up cross sections for interactions
- To this end, more deeply integrating our fast simulations in G4 could prove useful
- Another good solution would be to add some (public) interfaces to some of the Geant4 classes that could act as unit-test-like functions that the fast simulations could call
  - Some methods for example take a track and a step and then use only a small amount of info from those classes (which are fairly complex)
  - Fast sims are set up right now to 'fake' these objects on the fly
  - A simplified interface would ensure that we pass only the required information in and don't suddenly discover that information we missed was being used

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
  - A detector-specific smearing, and part of the gain is going 'directly' to analysis objects – ATLAS tracks and so on – which are not general
- We've talked about the possibility of using generative neural networks for fast simulation, for low-energy pileup, 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

Summary

- We have a number of different fast simulation options and need to evolve our fast simulation toolkit
- 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
- 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)

DETAILS OF FAST SIMULATIONS

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

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**