# The (ATLAS) ISF:
# Could It Be a Common Project?

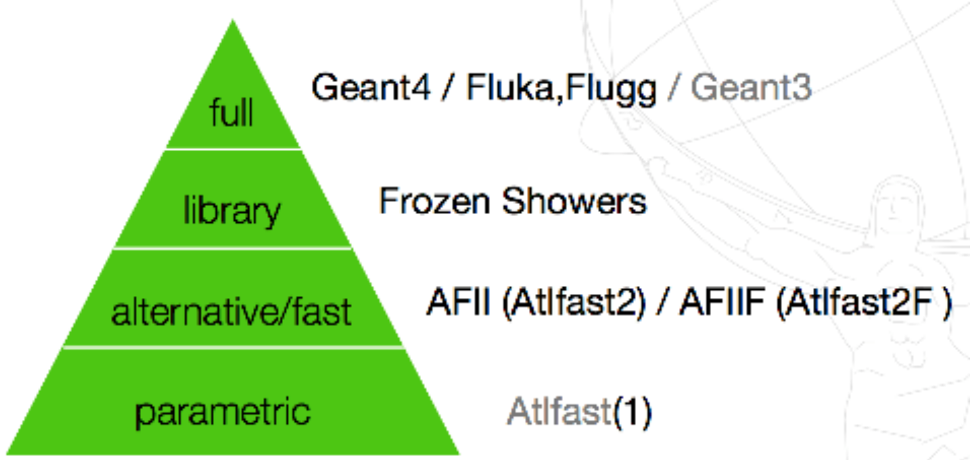
Zach Marshall (LBNL)

US ATLAS Computing Workshop

3 August 2016

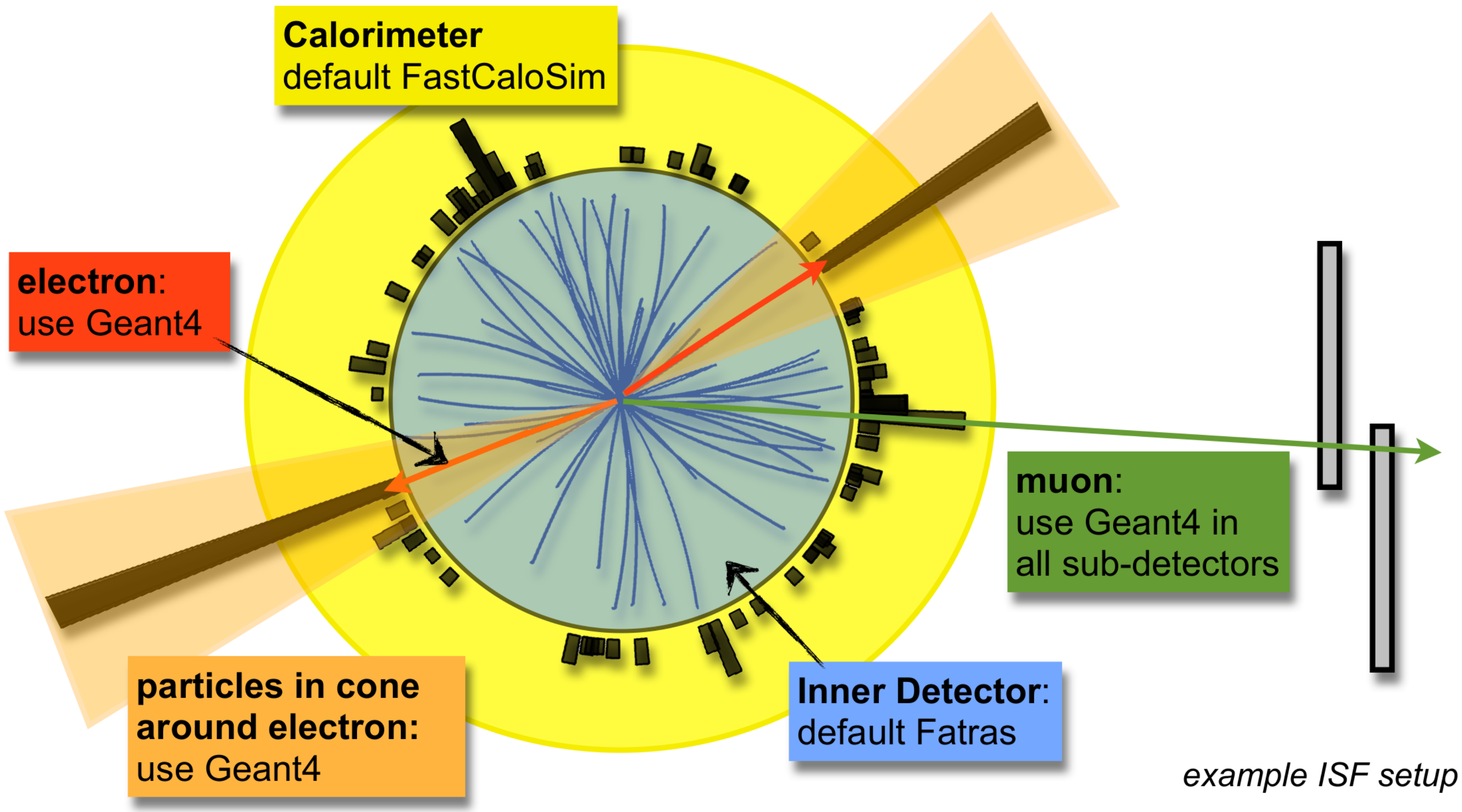
# Integrated Simulation Framework

- **By definition** more accurate sim means slower sim
- In ATLAS, and in many other experiments, there are many different types of fast simulation that have been implemented
  - Frozen showers (in the FCAL) are part of our standard full simulation
  - Our FastCaloSim is the 'standard' fast simulation in ATLAS and treats the calorimetry; FATRAS is a fast sim that treats the ID and muon sys
  - We have a parametric simulation for upgrade and some other setups
- Integrated just means you want these switches to be *easy* for users (give them a single integrated interface)


full
Geant4 / Fluka,Flugg / Geant3
library
Frozen Showers
alternative/fast
AFII (Atlfast2) / AFIIF (Atlfast2F )
parametric
Atlfast(1)

# How Integrated?

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

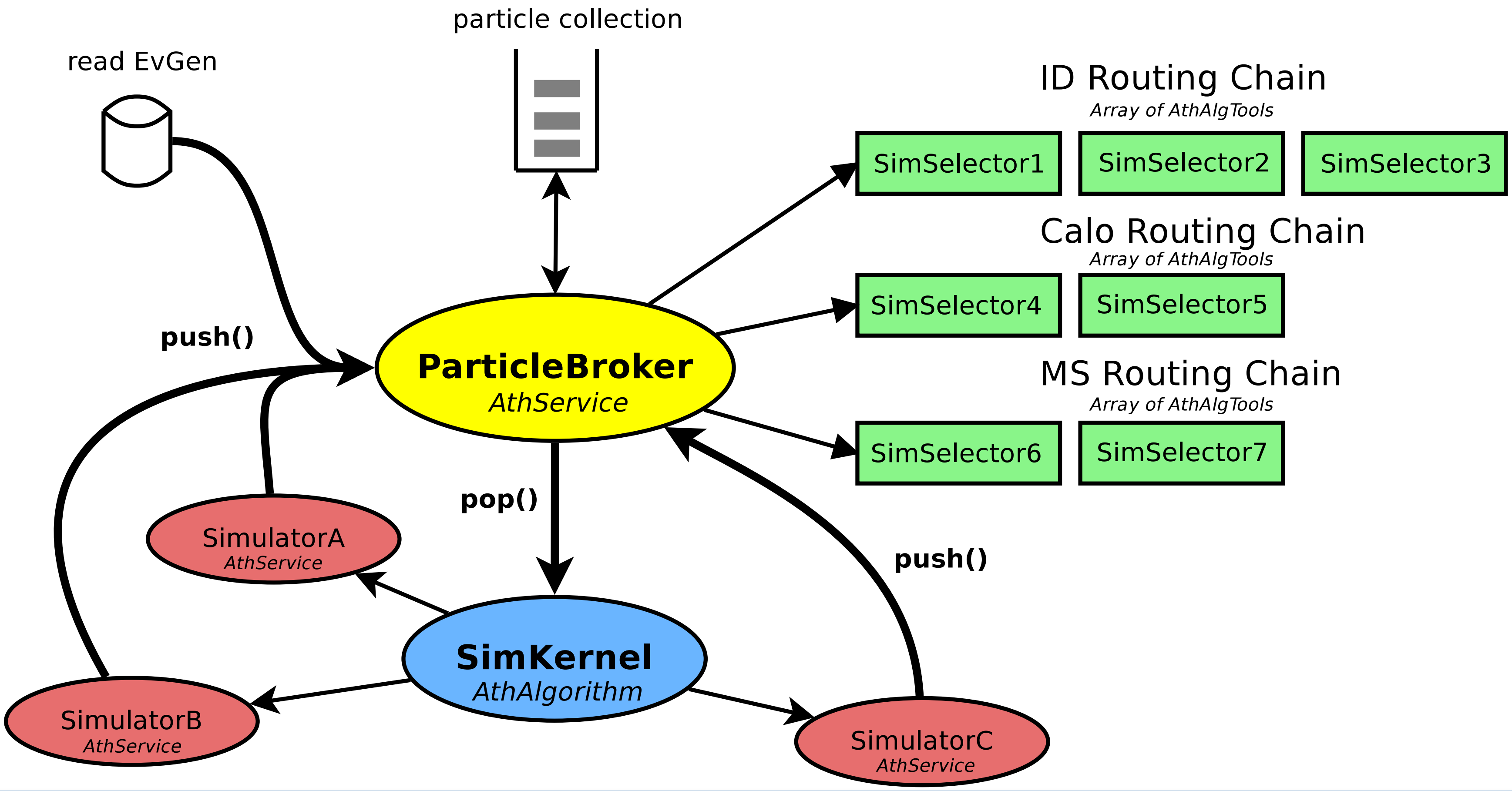


# Enter the ISF

- The ISF is ATLAS's solution to providing a **single interface**
- It goes a bit deeper than that, taking the **top-level stack** away from Geant4
  - That design has the advantage that we can easily separate particles that we never want to pass up to the ISF (e.g. low energy electrons in the calorimeter), so we save some translation into and out of our types
- HepMC Event gets translated into ISF particles
- ISF gives out the particles a vector at a time to the simulators
  - Vector at a time to somewhat reduce the overhead from the translators, begin and end of Geant4 event actions, and so on
- 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)

# In Practice

- Complex set of *selectors* that can dictate where particles will go
- Simulators that do the real work
  - Pull a particle (or array) from the stack, optionally push particle(s) back



## Geant4 Interface Basics

- Geant4 works with a ***stack***.
- Users hook fast simulations in as ***processes***.
- **ISF Version**:
  - Particle picked up from the stack
  - Simulation selectors examined to decide on simulator, based on particle properties (and potentially event properties*)
  - Particle assigned to simulator and propagated
- **Geant4 Version**:
  - Particle picked up from the stack and put into Geant4 (full sim)
  - At each step, fast simulation(s) are given a chance to take over, making their decision based on particle properties
- **Practical differences**:
  - ISF selectors can find out about the **event properties** (this is helpful!)
  - Some performance penalty for putting particles into Geant4 and then taking them out? Not large unless sim is *extremely* fast. Could use the Geant4 isApplicable interface to override ("give me this now")

# Information Transfer

- The ISF stack is based on “ISF particles”
  - The original notion was that we would want a base class with more information attached than is available in Geant4 tracks (the equivalent)
- Geant4 allows an extension via G4VUserTrackInformation
  - This thing could be used to pass around all the information that ISF needs, with a common implementation for the experiments
  - There’s nothing *really* ATLAS specific about the extra information that we have attached to our particles (things like detector region, which is from an enum)
  - Similar tricks exist for other base classes and could be used to move the ISF base classes to Geant4 classes with wrappers
- Learned early that we *must* keep a Geant4 stack
  - There are so many particles moving in an event that we cannot afford to e.g. move all the low-energy junk between different types
  - Might be able to play a similar game with track classification within Geant4 to get the same benefit

# 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 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

# Parallelism

- At the moment, Geant4 works on ***event-wise*** **parallelism**
- Having multiple particles could allow particle-wise parallelism.
  - Similarly, multiple simulators could allow simulator-wise parallelism.
  - In the Geant4 model, since the particles are always moved into the "full simulation" and the fast simulations are only processes, "simulator-wise" parallelism would not exist, or it would look identical to particle-wise parallelism
- In ATLAS, sim-wise parallelism required different simulators to have sensitive detectors write to different collections
  - We then have a merge step at the end
  - Would have to play a similar game for multiple threads moving particles in the same event
- This splitting and merging could be automated if one knows the simulators at the beginning of the event
  - Would imply some Geant4 infrastructure to provide different containers

## Summary

- The ATLAS Integrated Simulation Framework was developed to bring together various flavors of fast simulation within a single interface
- There are ways that this could be done entirely within Geant4
  - We do need to think through a few performance questions
  - Thus far, the fast sim design in GeantV is *very* similar, and ISF design sends vectors of particles to fast sims, so this could be all common
- Development and migration could take time, but in the end we might have a framework that other experiments could use
- During that development, we might be able to benefit from interaction with the Geant4 team
  - Some interface tweaks would probably be helpful
  - Certainly we already know that it would be helpful to have simplified interfaces for some of the fast simulations to use