Random Thoughts

Zach Marshall, with lots of John's thoughts on Digi

LBNL TIM Workshop

10 November 2015

Generation

- Nobody at this workshop is speaking about event generation ☹
- We already have a parallel generation system for some generators *inside of athena*
  - Pick up the number of cores from AthenaMP
  - Run MadGraph or Powheg with that number of cores
  - Swap back to a serial job to run Pythia (Pythia reads from files; it is a very serial task)
- There are some more interesting things that we could try for some of these event generators
  - Writing matrix elements continuously on the fly from N-1 cores and showering on the other 1 core
- It's not an insignificant portion of our CPU, so we should really think a bit more about what's running in there
  - Us, profile Pythia??

AthSimulationBase

- Some of you have heard of AthSimulationBase
  - http://atlas-nightlies-browser.cern.ch/~platinum/nightlies/info?tp=g&nightly=AthSimulation-1.X.0
  - Originally built to help with HPC "kit" distribution, obvious bonus that it's easier to build for an alternative platform
- Only the 417 packages "needed" to run full simulation
  - <10% of the Athena libraries of the standard release
  - There are some not-great dependencies in there (XMLCatalog, XMLCoreParser, RecEvent, quite a bit of Tracking and Vertexing stuff)
- **Need help** from experts who can help us reduce the externals
  - We don't know what we need locally or what we might need on the grid that we shouldn't remove (PyCMT? AtlasAuthentication? ChkLib? AtlasRELAX? AtlasExpat? AtlasGSL? AtlasShift? AtlasDCACHE?)
  - These ***externals*** are going to be the big problem for porting this thing to alternative architectures

Simulation Biasing

- We frequently want to generate a large number of events in the tail of some distribution
- Often the event properties are only determined after simulation (e.g. after we know how much energy the muon lost in the calorimeter, or how far into the tail the jet response is)
- We could imagine a biasing method to re-simulate (up to N times) an event based on the post-simulation properties
  - This is already done in Evgen for PythiaB
  - Only some properties work; if things are affected by pileup we're dead
- This would be a **very** CPU hungry job, but it could save us other resources (disk, digi/reco CPU…)

Some Fast Sim Thoughts

- To a naïve user, Geant4 might *sound* very vectorizable, when in fact it's a pretty complicated piece of code that is not so trivial to work through
  - The idea is obvious, though: we have a whole bunch of particles and we act on those particles. Surely we can just put the particles into vectors!
- Actually, while this is not super true for things like Geant4 and even FATRAS, it *is* very true for FastCaloSim
  - Could target a vectorized version of that guy once the new tune is done

Digitization

- Digitization is a bit like reconstruction and a bit like simulation and a bit like neither of them
  - Most of digi is implemented as ***one algorithm*** with ***tools*** for each subdetector that do the work (except for L1 algs and some other small contributions). Note each tool is called multiple times (once per pileup event), so we need to be careful about thread-safety in these guys
  - It has a lot more I/O and a faster event loop (like reconstruction ☺)
  - It is inherently ***local***, so it is an obvious target for ***more paralellism***, and could be a heavy user of vectorization (might require EDM work!!)
- Generally we don't use more than one IOV per job
  - This infrastructure is available, but it hasn't been used in some time; remember that there are a limited number of conditions that we want to apply in *digi* (not later in reco or earlier in sim) that might vary
  - Overlay is the main worry here, though we don't have much experience
  - Could we imagine a "force conditions to load" method for digi and sim to be used before the fork / in a thread-shared way in MP/MT jobs??

All Those Events…

- PileUpEventLoopMgr holds the keys to the pileup event kingdom (caches for pileup events)
  - This is where most of John's concerns for GaudiHive-like setups lie
  - That thing is basically a service, so we could have context-specific data (pileup caches per thread), similar to what we have now
  - Might be potential for wins in re-writing this thing – obviously using the same low-p_T events across multiple threads is an opportunity