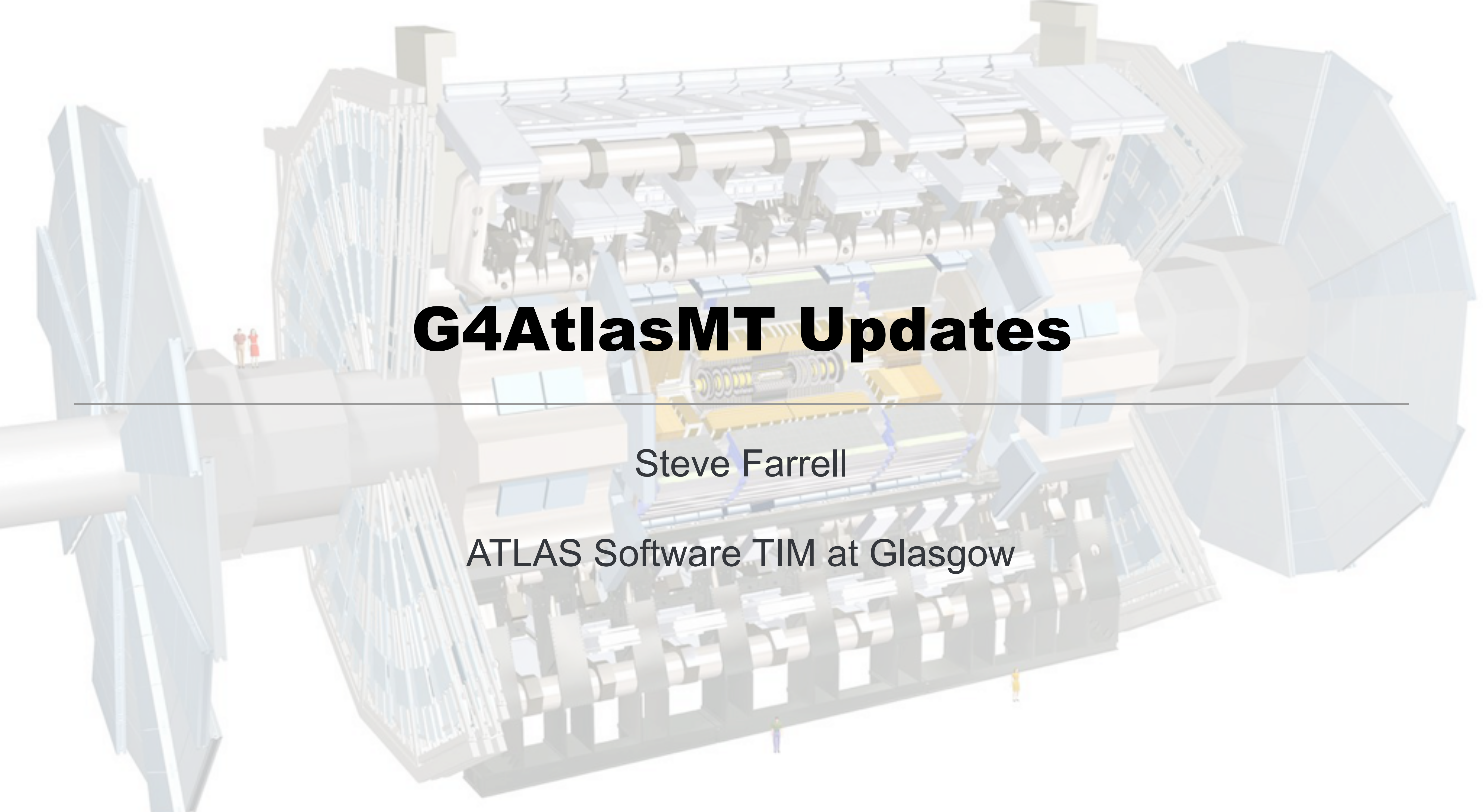


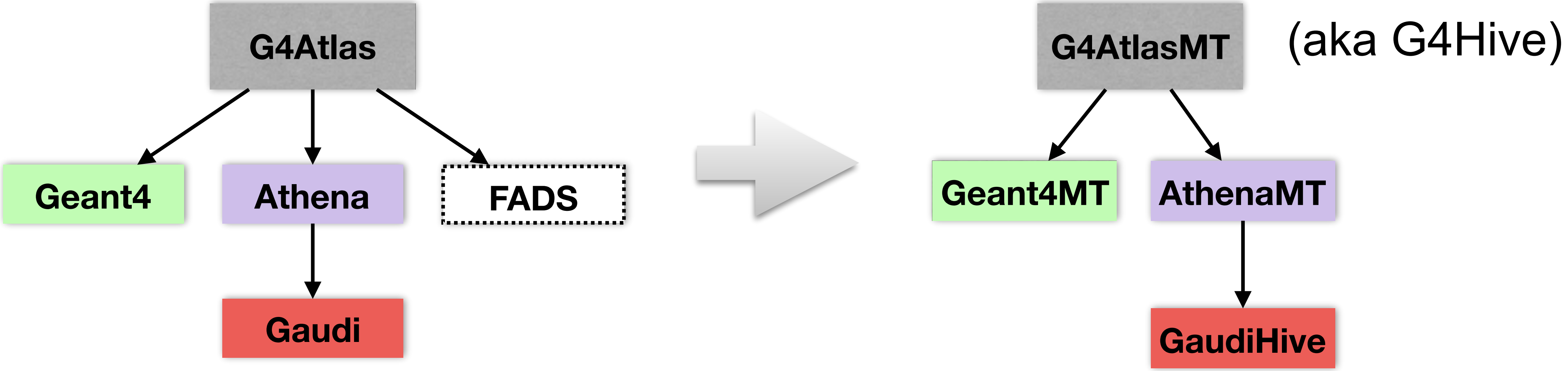
G4AtlasMT Updates

Steve Farrell

ATLAS Software TIM at Glasgow

Jun 6, 2016

Introduction



- Previous report from November TIM:
  - https://indico.cern.ch/event/395887/contributions/947413/attachments/1185054/1717711/Farrell_G4Hive_v3.pdf
- Good progress has been made on multi-threaded G4Atlas, the “old-style” ATLAS simulation
  - Nearly a “complete”, “realistic” simulation configuration in place
  - Updated performance measurements
  - ISF-MT not yet heavily pursued, but still progressing (partially indirectly)
- Still some open issues
  - Few blockers
  - Non-essential missing features
  - Design choices that could be simplified
- Bonus: first tests with KNL architecture!!

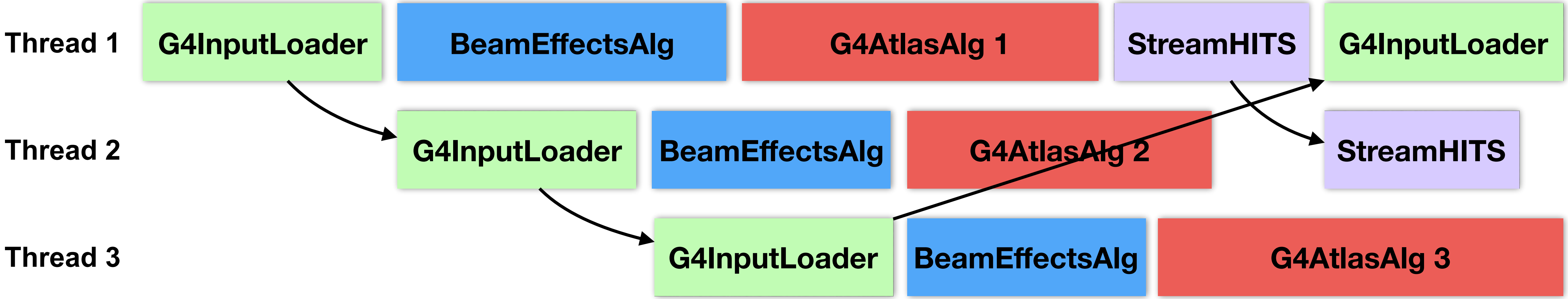
Overview of progress

- Geometry and physics lists
  - Tool-based design from the G4Atlas infrastructure migration seems to be working well in MT
- Sensitive detectors
  - Nearly all done and working
  - LAr SDs are not yet working, but making progress
  - Design is still holding up
- User actions
  - Migration ~done, but not all work in MT
  - Design works, but migration has taught us we may want to simplify some areas
- Truth code: ***now working in MT!***
- Magnetic field: ***now working in MT!***
- Fast simulations
  - Good progress made on frozen showers

Many successes!

Review of algorithms

- We now have four algorithms
  - BeamEffectsAlg applies beam-related smearing to the gen event and saves the new collection as input to G4AtlasAlg.
- We still run *mostly* event-level parallelism
  - All the simulation work is done in G4AtlasAlg, which we clone for each thread
  - The new BeamEffectsAlg gets cloned as well, but it must run before G4AtlasAlg
  - I/O algs are *not cloned*
- Re-entrancy?
  - Not worth it because of G4 design/restrictions



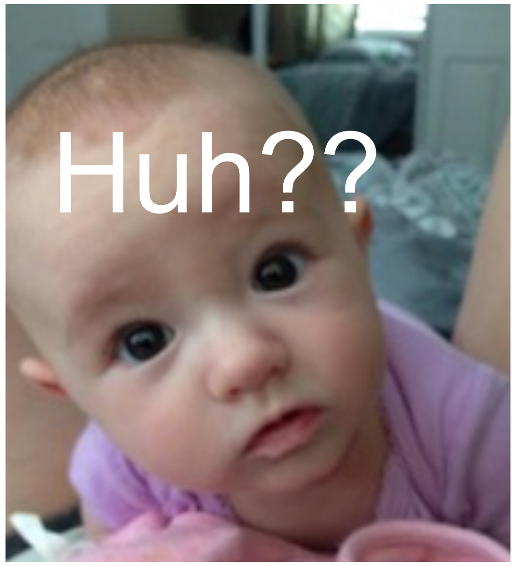
Just a cartoon. Not to scale.

LAr sensitive detectors

- LAr SDs collaborate on hit collections. In the new SD infrastructure, initial design was not thread safe.
  - WriteHandle lives in SD tool, which attempts to collect hits from all SDs
  - But SD tools are “shared” in MT (they hang off a service)
  - Pointers to SDs were not thread-local
- My proposed solution: use an SD wrapper which is thread-local and holds the WriteHandle as well as the actual SDs
  - This SD functions as the single SD of the SD tool, restoring consistency with the SD tool pattern for MT
  - Not the only solution, but it does work and produces consistent outputs
- This implementation is in branch tags and ready for merging, but thread safety is waiting on a migration of the LAr calculators
  - See https://its.cern.ch/jira/browse/ATLASSIM-2606
  - See https://its.cern.ch/jira/browse/ATLASSIM-2290

Digression on teaching

- I get the sense that many (probably most) developers are still unclear about the essential concepts of AthenaMT, despite the amount of time we spend talking about them
  - cloning of algorithms
  - public/private tools in MT
  - levels of thread safety and the requirements for components
- This makes design decisions difficult
  - Every design decision is complicated now by these concepts
  - Understanding is really essential
- I suspect this is because the comprehension doesn't sink in until you actually get your hands dirty
  - Maybe we should put together a small tutorial homework that demonstrates the concepts



User actions

- I think all actions are migrated to the new infrastructure, though a few are not really thread safe and will not allow to run in MT
  - e.g. actions which write out custom ROOT files or use THistSvc
- …which reminds me: can we have an implementation of THistSvc which manages context-local histograms automatically?
  - user code is otherwise fairly complicated; e.g., see: http://acode-browser.usatlas.bnl.gov/lxr/source/atlas/Simulation/G4Utilities/G4UserActions/src/LengthIntegrator.cxx?v=head#0464
- We've been thinking about ways to simplify the design
  - Reduce tool interfaces in exchange for more dependencies
  - Inheriting from G4 base classes instead of custom interfaces
- I pushed a thread-termination update into Gaudi
  - We can now have end-run actions!
- See https://its.cern.ch/jira/browse/ATLASSIM-2226

Truth code

- The G4Atlas code was pretty old and not very Athena-centric
  - Truth strategy objects hang off a global singleton TruthStrategyManager
  - Static state in the AtlasTrajectory, elsewhere
  - Global storage of current list of secondary particles
- Rather than re-write everything (which we'll probably do later), I actually managed to get it working multi-threaded with some "minor" refactoring
  - Cleanup of statics
  - Moved some TruthStrategyManager code into stateless standalone functions:
    - See MCTruthBase/TruthStrategyUtils.h
  - Using G4Step method to query current list of secondaries
    - Thanks to an upcoming patch requested in Geant4
- The new implementation works and is validated!
  - See https://its.cern.ch/jira/browse/ATLASSIM-2409
- Longer term plans involve merging with the ISF solution in a thread-safe way
  - Some investigation has already been done into this, but it will require some refactoring
  - Real progress was made at sim workshop last week

Magnetic field

- G4Atlas infrastructure migration resulted in new tools/services for G4 fields
  - G4MagField services create G4 fields
  - G4FieldManager tools create field managers with steppers and fields
  - DetectorGeometrySvc owns the field mgr tools and invokes the setups
- After just a little refactoring, it works now in MT!
  - Finally after a ~year I can confirm the mag field svc is thread safe
- See https://its.cern.ch/jira/browse/ATLASSIM-2373
- See https://its.cern.ch/jira/browse/ATLASSIM-2793

This reminds me: it’s difficult/impossible to prove thread-safety in components. We should provide some “simple” Athena setup for stress-testing thread safety of components

Fast simulations

- Frozen showers
  - Necessary to be considered a “complete” simulation application
  - Good progress made at the Simulation Workshop at Cambridge last week
  - Turned out to be easier than expected
  - Follows a sensitive detector pattern, with a frozen shower service that was made thread-safe
  - Basic implementation is done, but currently debugging some data-race related crashes (e.g. RDBAccessSvc)
- AF2
  - Next most common
  - No progress yet
- FATRAS
  - Least commonly used
  - Larger scale problem, not sure how to proceed yet

MT in the ISF

- Migration of core ISF code to AthenaMT started at the Simulation Workshop
- Data-flow refactoring
  - Previously, collections were created and stored within services
  - Moving event store interactions instead into main ISF SimKernel algorithm
  - Containers/elements now passed into the services/tools for processing
- Hits collection merging
  - Multiple simulators were writing to same hits collection in StoreGate
    - Common pattern in ATLAS code
  - Now, simulators write separate collections which get merged at the end of the event
  - This design choice appears in many places. Not clear without spending considerable effort which solution(s) are the best

Other open issues

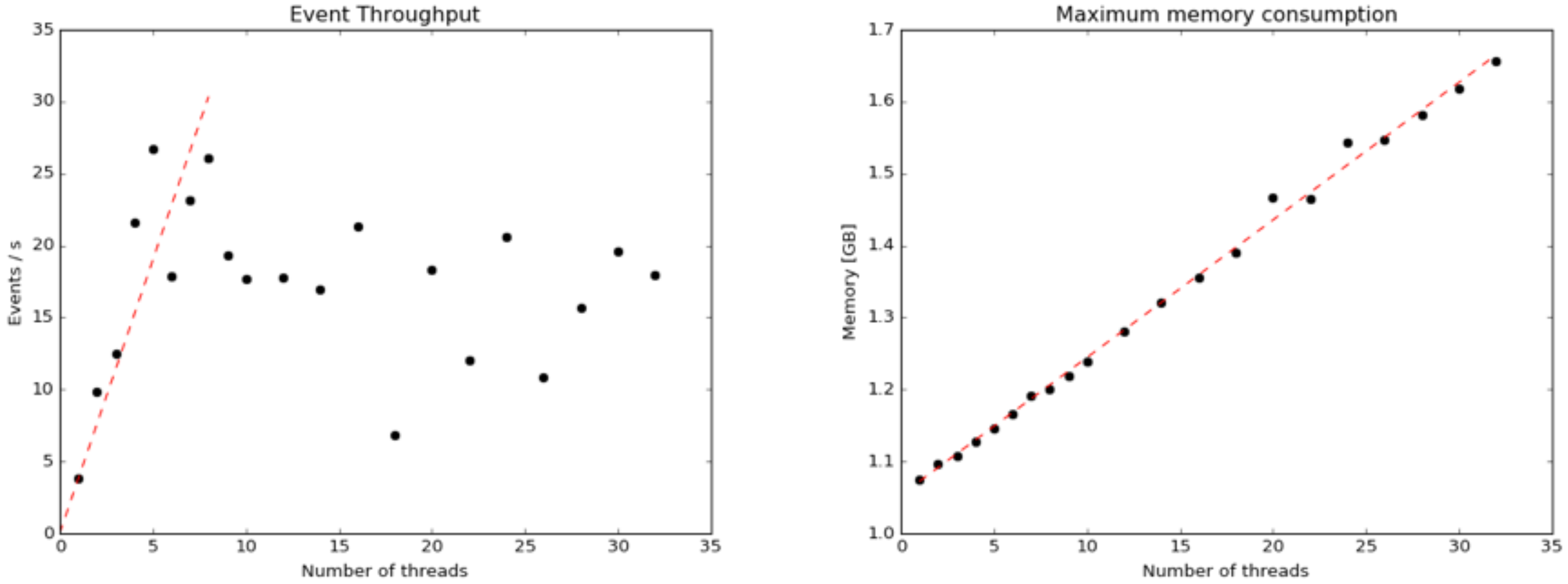
- There may be some non-essential straggler pieces in the simulation
  - e.g., VertexRangeChecker is a FADS thing
  - We have a non-FADS version which we can maybe adopt
- DecisionSvc has been an issue for a while
  - Used for aborting/skipping simulation events
  - Grabs decision algorithms once in the job, and then queries them when queried for a decision by the output stream
  - Obviously this completely falls over in multi-threading!
  - This thing will need a re-write. Any thoughts?

Short list `:D`

Performance measurements

- Now that the application is realistic, we can really start to trust the performance measurements
  - Previous results are from last year. Time to redo them!
- What kind of data do we want?
  - Throughput scaling
  - Memory scaling
  - Timing information at algorithm granularity
- Other stuff
  - Compare MP and MT concurrency
- Hardware to use
  - 16-core machine at CERN
  - NERSC’s Cori phase 1
  - Intel Knight’s Landing early access machines

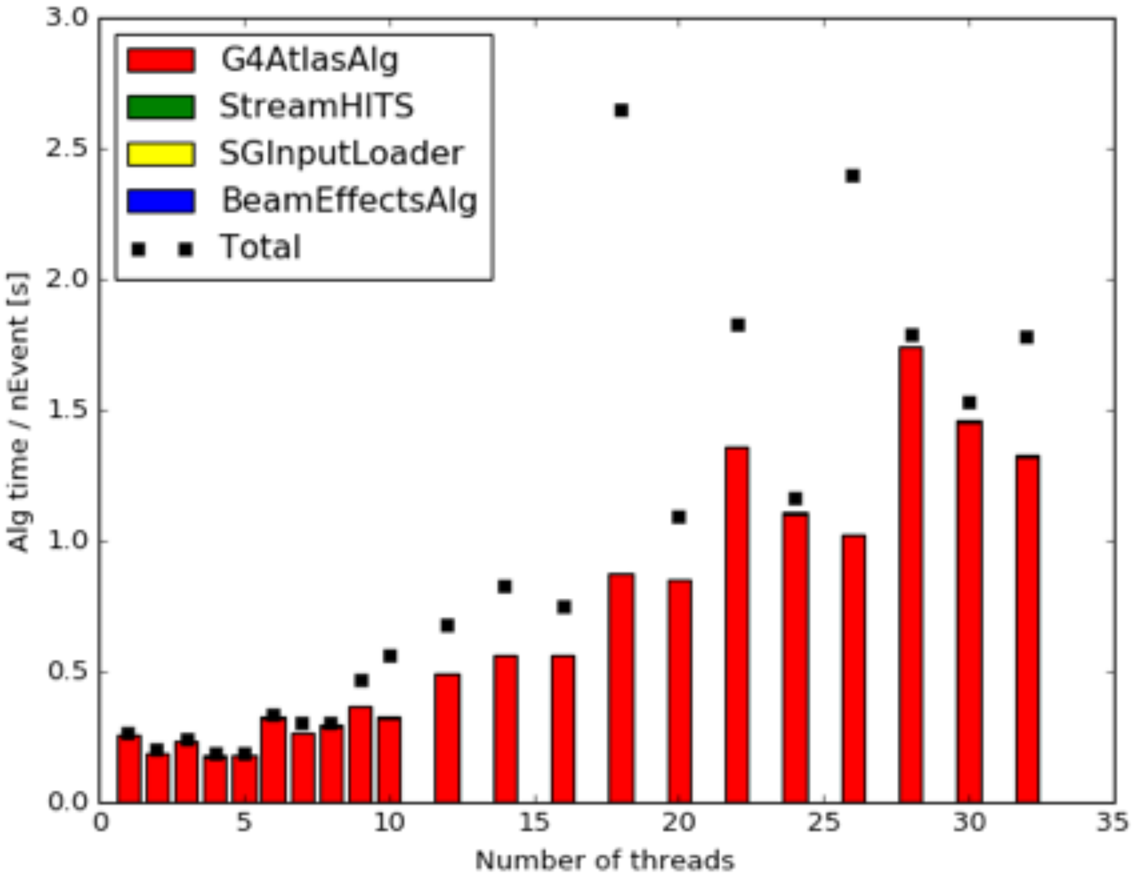
First checks, single-mu sample

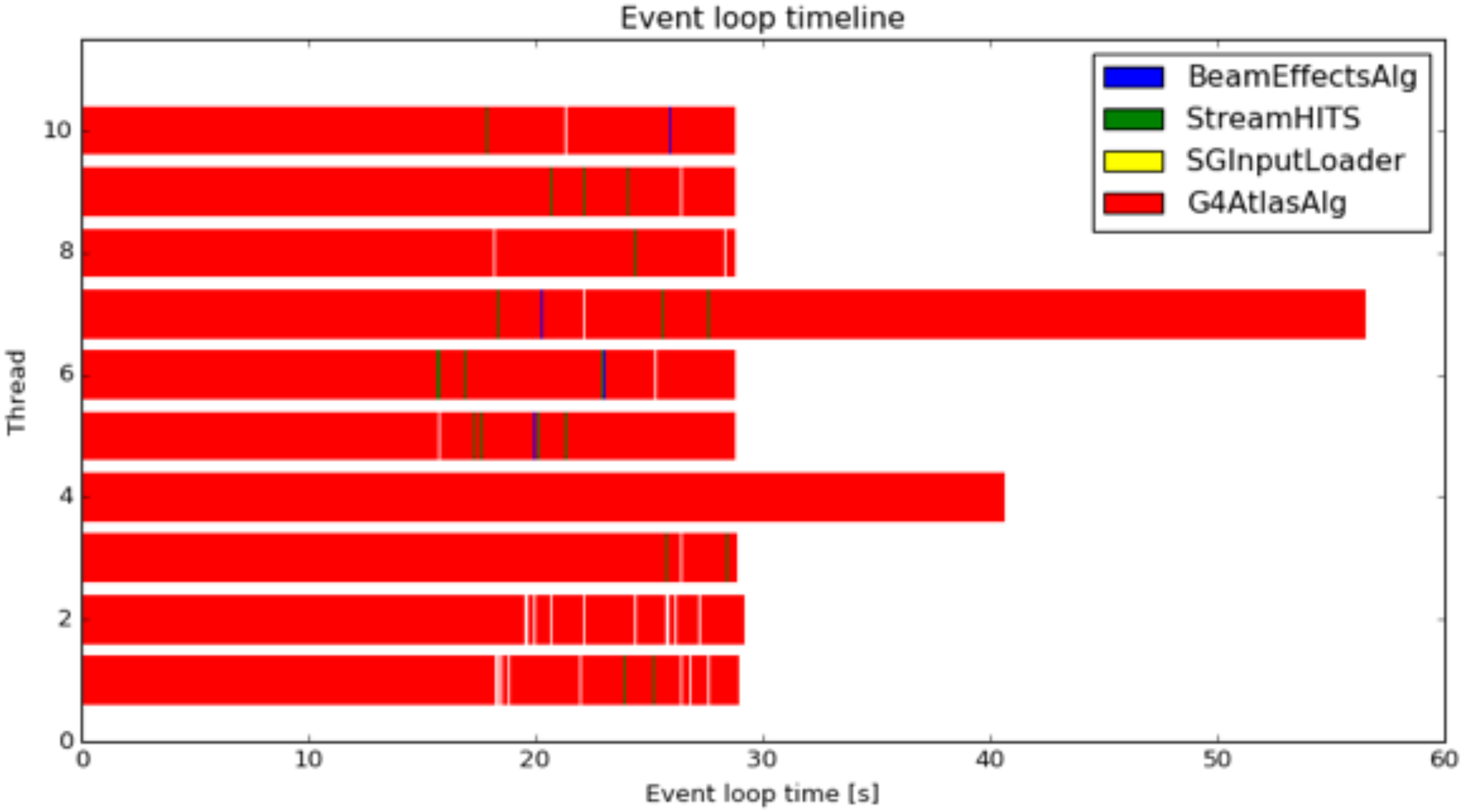


- Terrible throughput!
- Uh-oh, what did I do??

Single-muon sample results (no calo, magfield)

- Looking more closely, we can see what’s happening

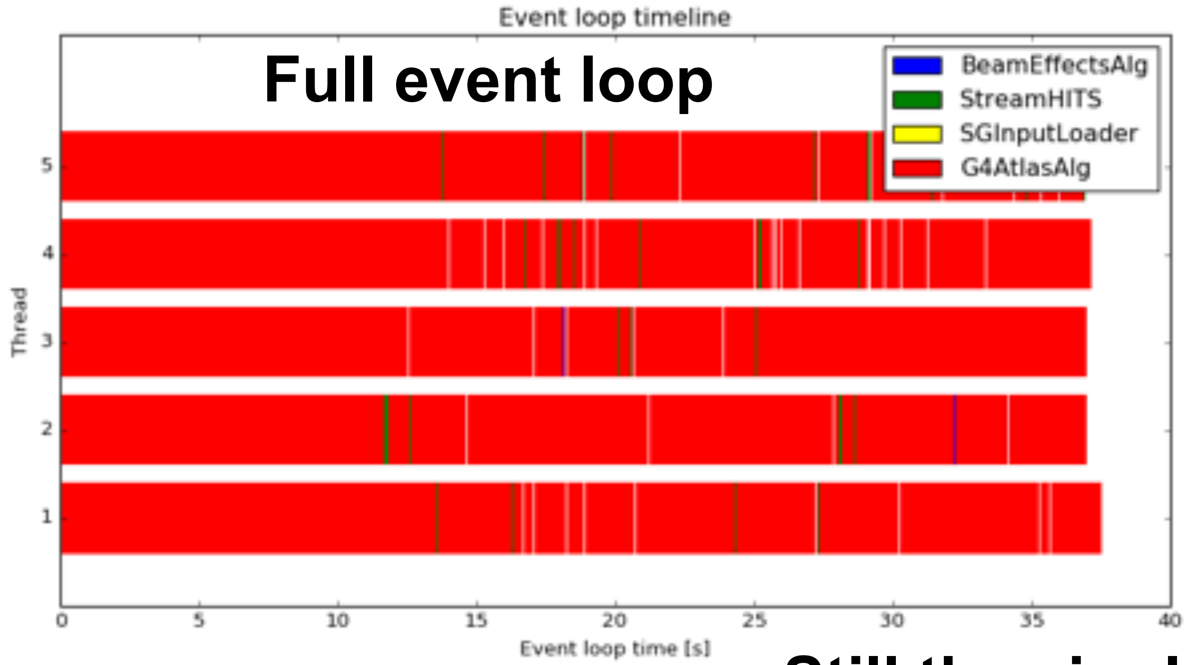


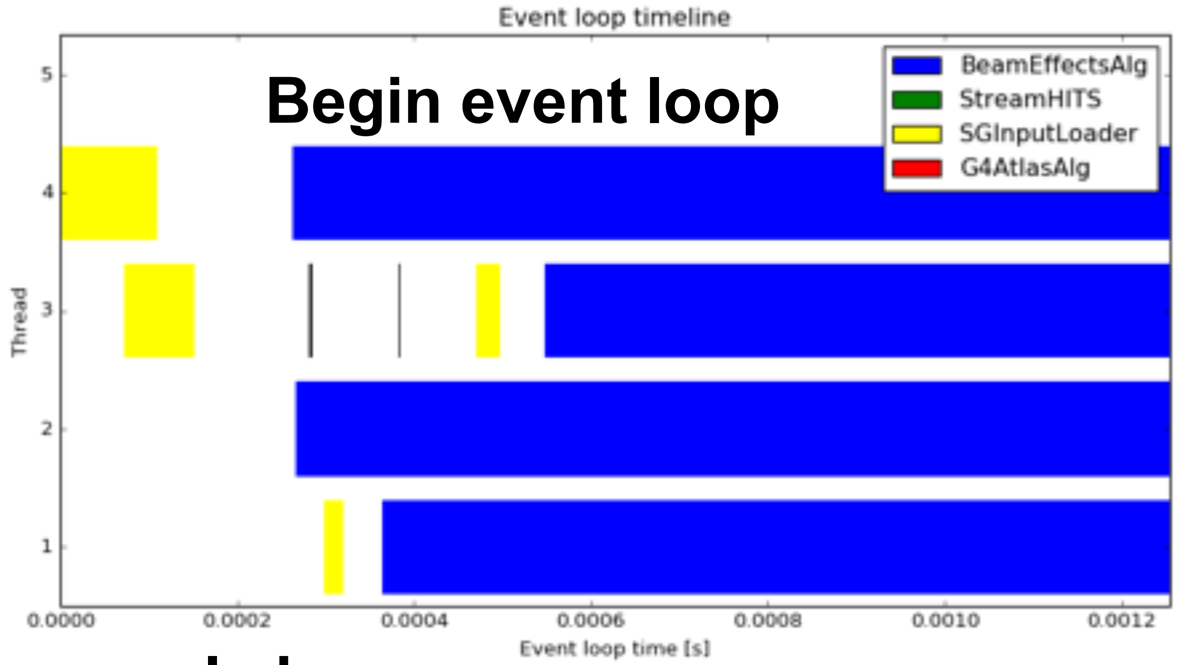


- Apparently, few events are taking up too much time at the end of the loop
  - Whew, so just insufficient stats or random hiccups

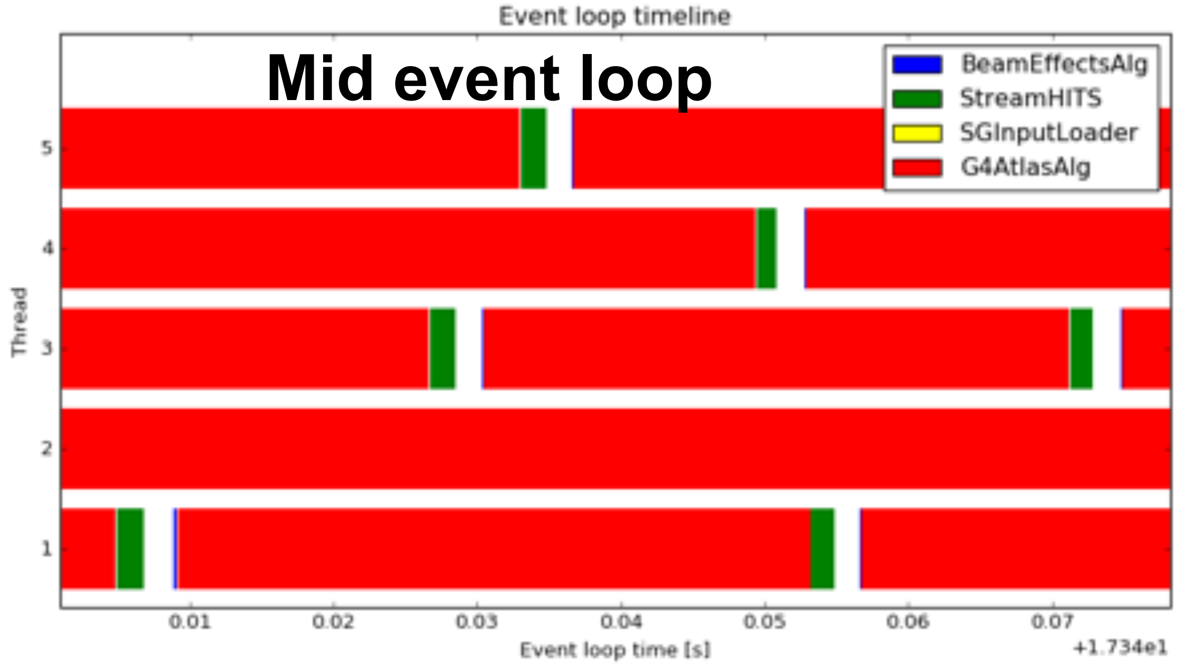
Visualizing algorithm timeline

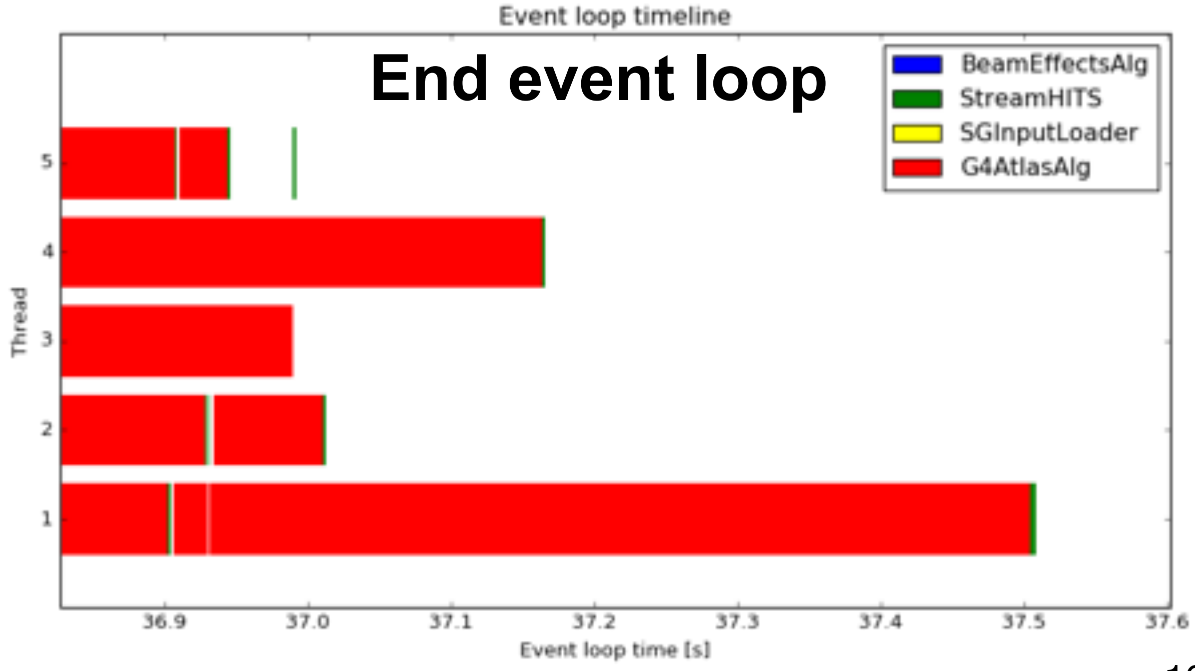
- Using ipython + matplotlib, can interactively look at what’s really happening in the event loop => very useful!



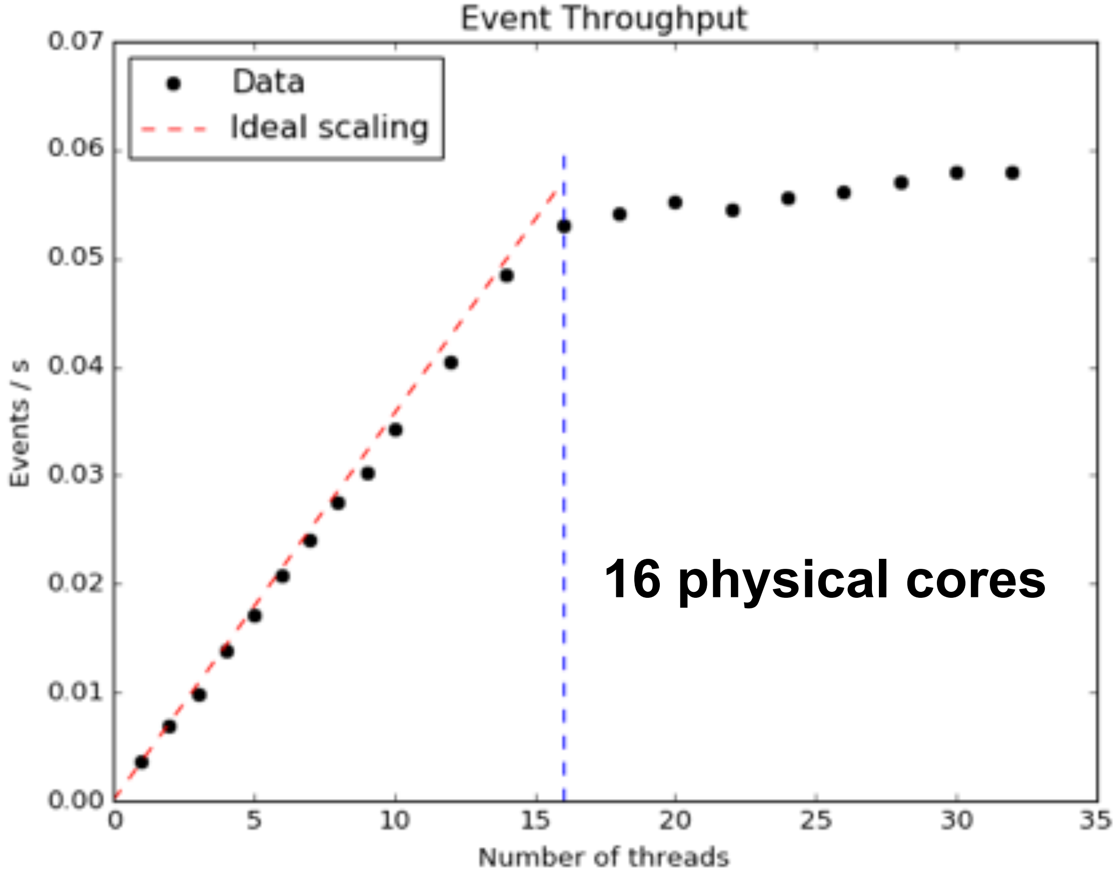


Still the single-mu sample here

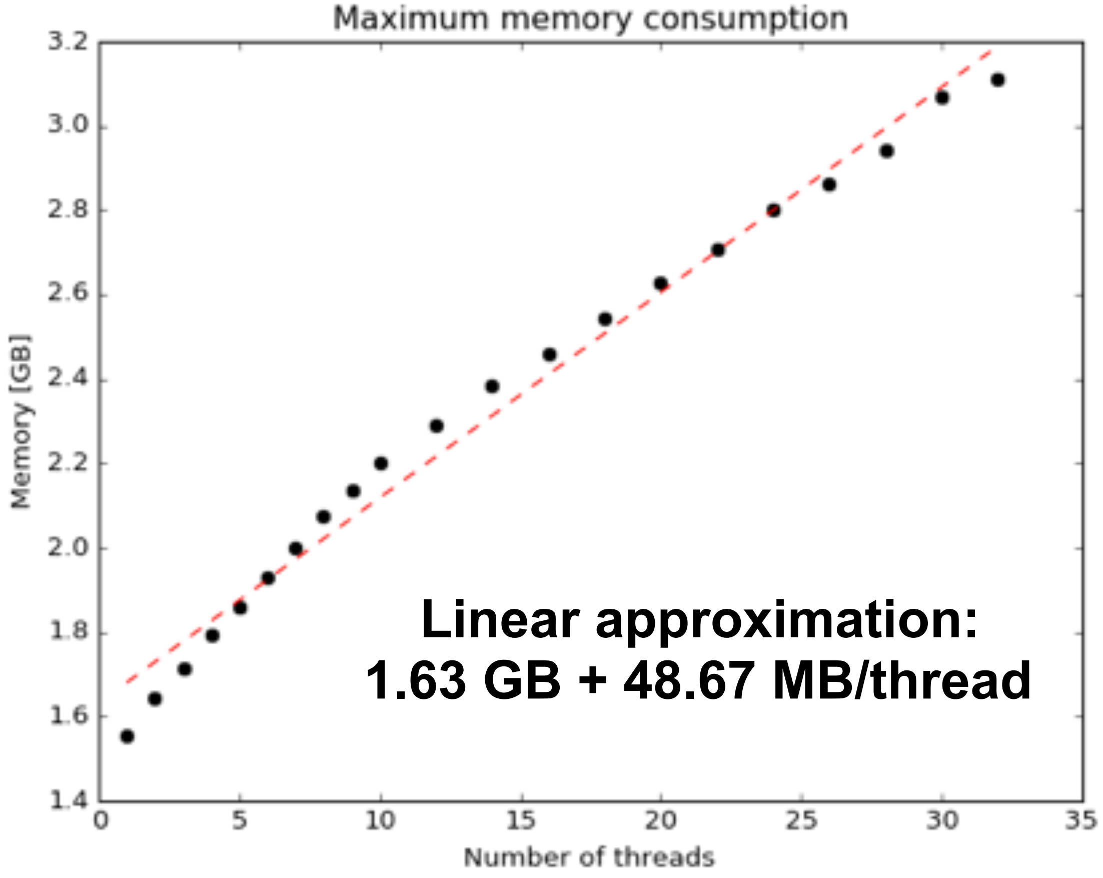




TTBar results (with calo, mag-field, no LAr SDs)


Event Throughput

16 physical cores


Maximum memory consumption

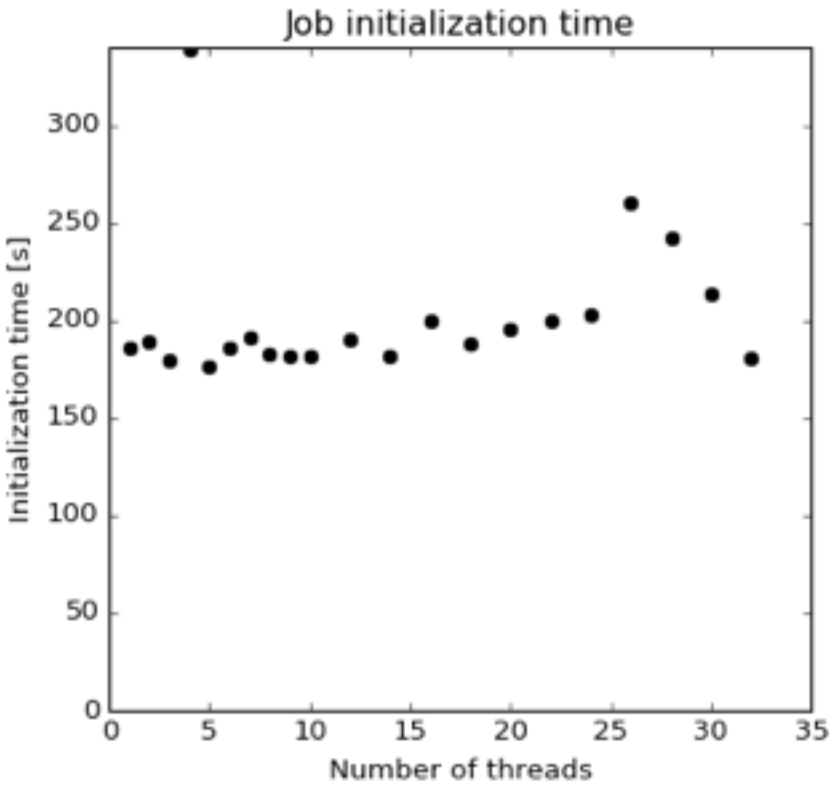
Linear approximation:
1.63 GB + 48.67 MB/thread

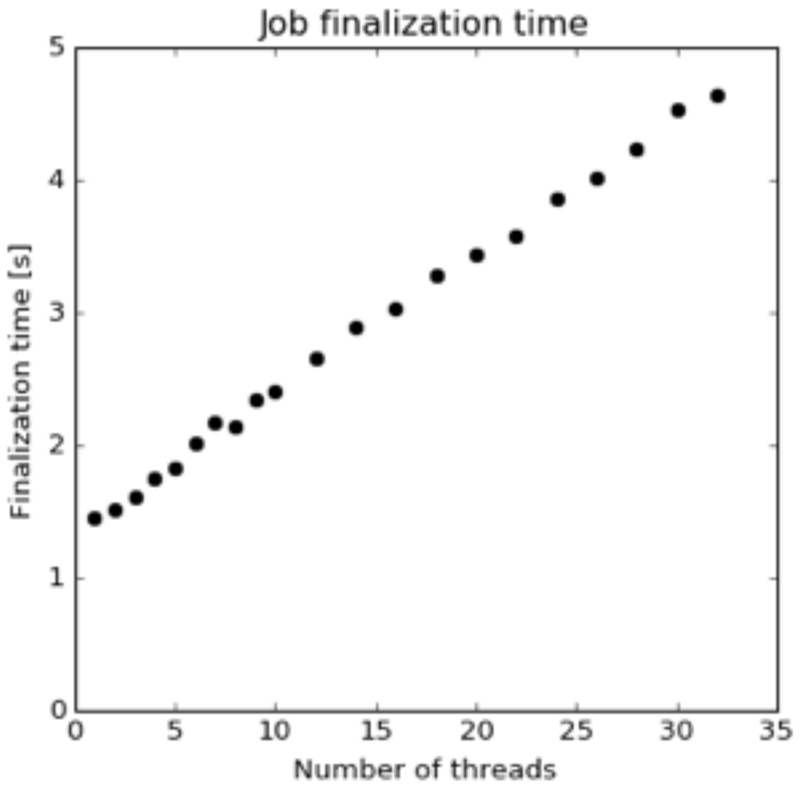
- Scaling results are quite good
  - Throughput shows nearly perfect scaling up to the number of cores
    - Some degradation at 16 is probably expected due to interference from monitoring
  - Only 50 MB per event => should be good for KNL
    - ~5 GB at 72 threads
    - ~9 GB at 144 threads
    - ~16 GB at 288 threads

NOTE: LAr SDs will
affect memory footprint

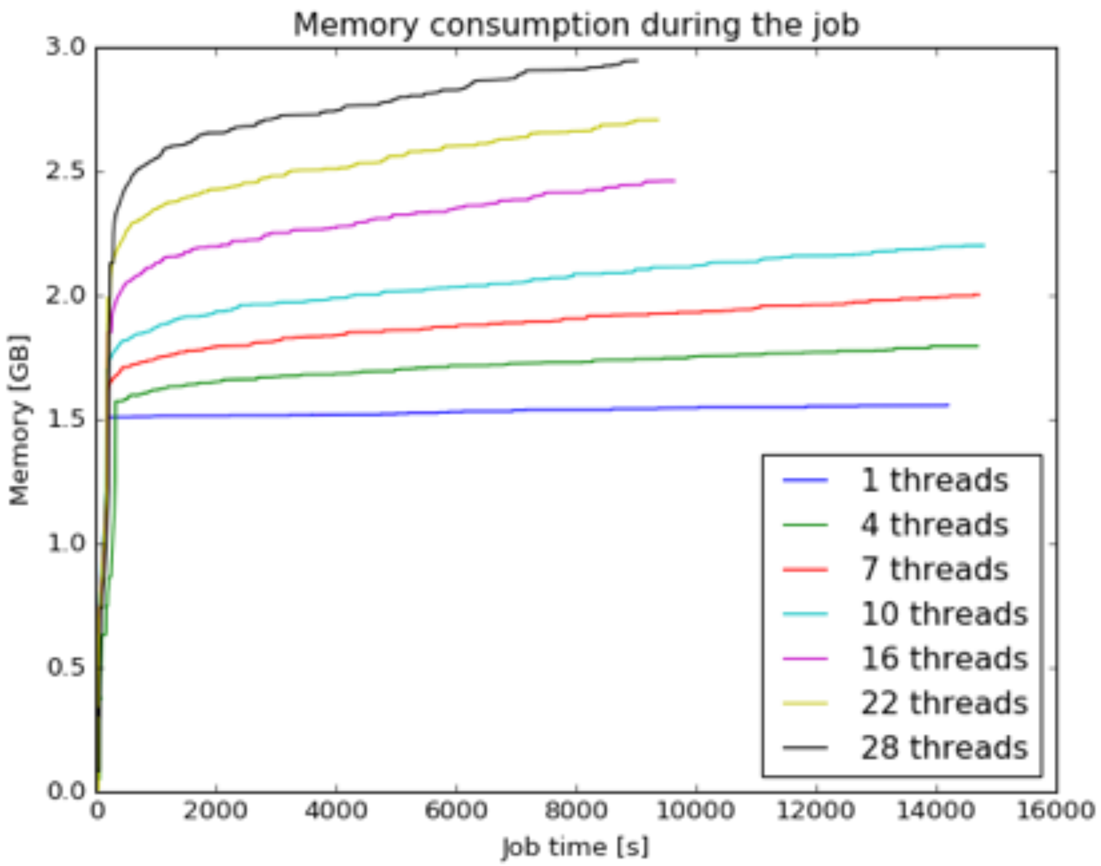
TTBar sample results

- Initialization and finalization times
  - Not very surprising, besides a few funny fluctuations

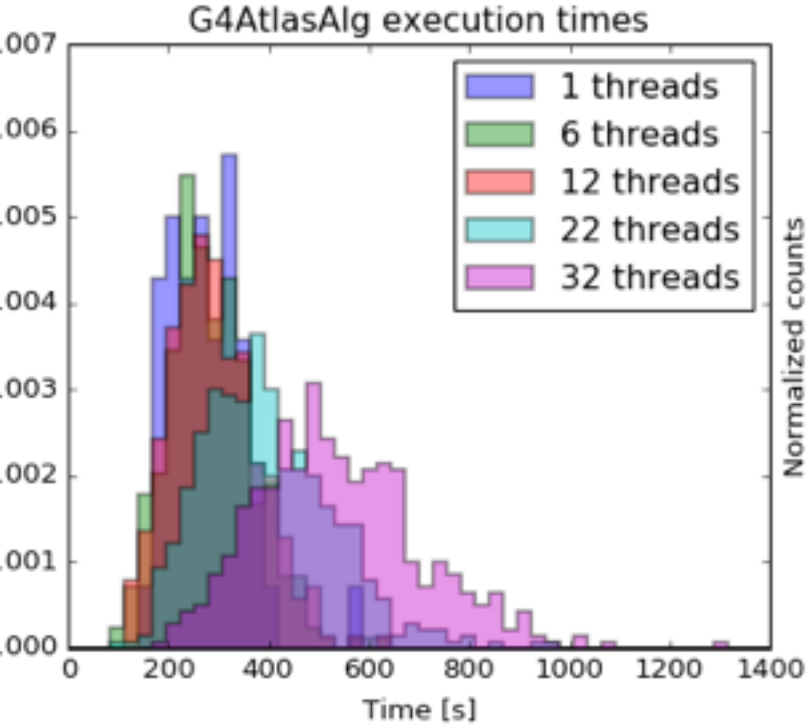


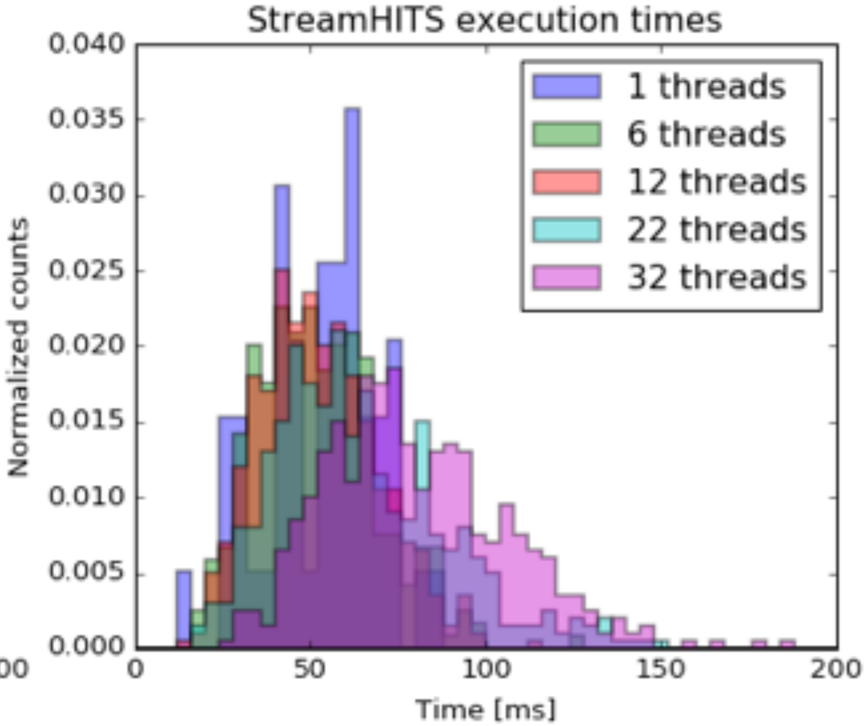


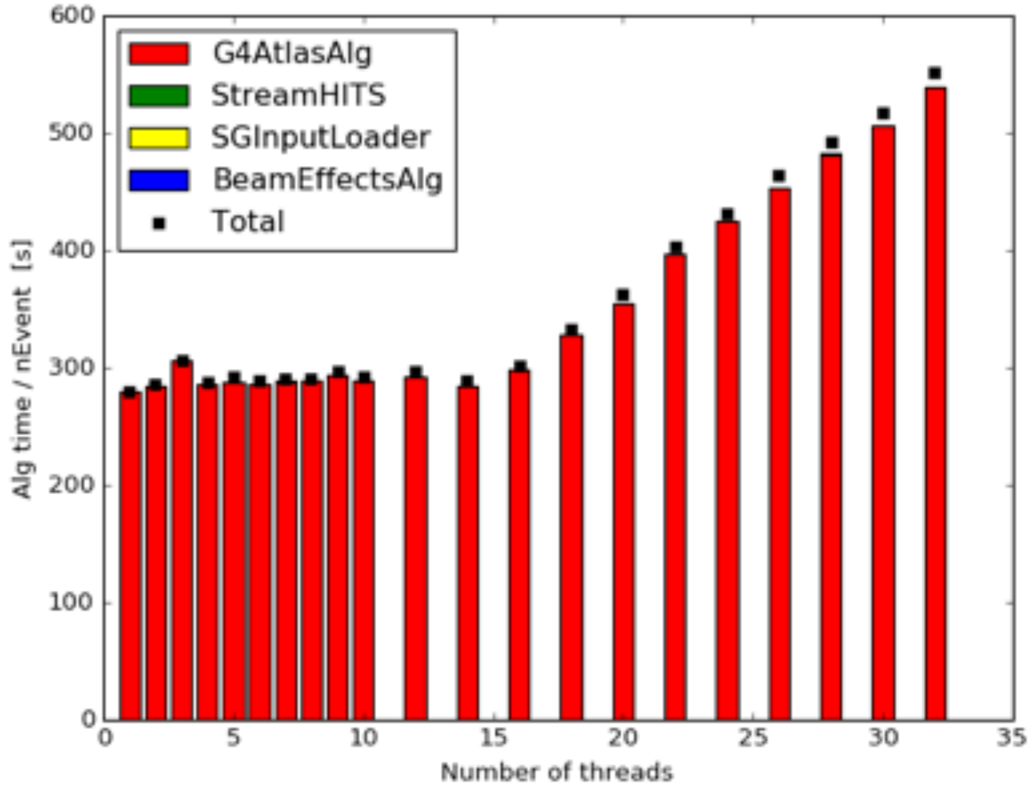
- Memory vs time
  - Not yet plateauing…?

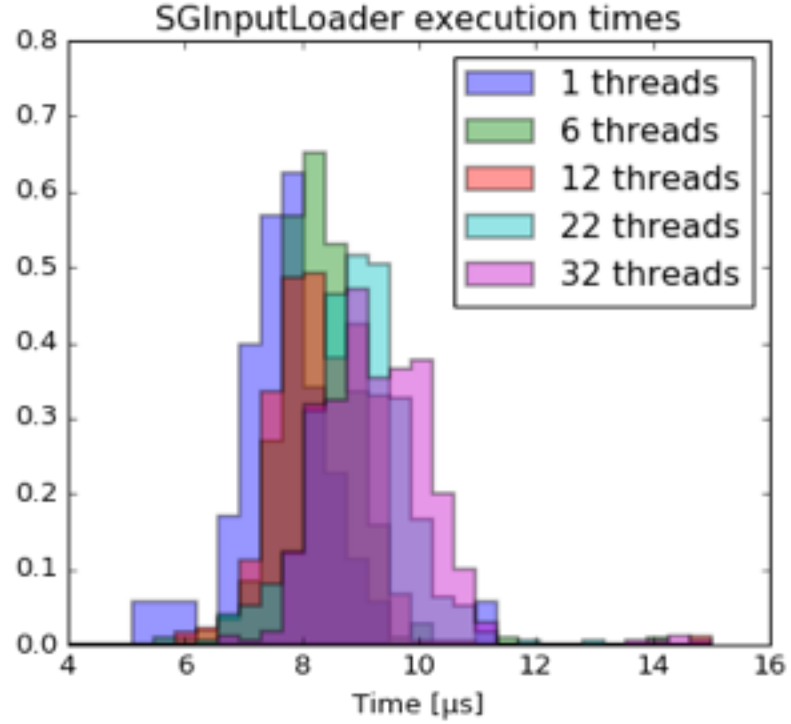


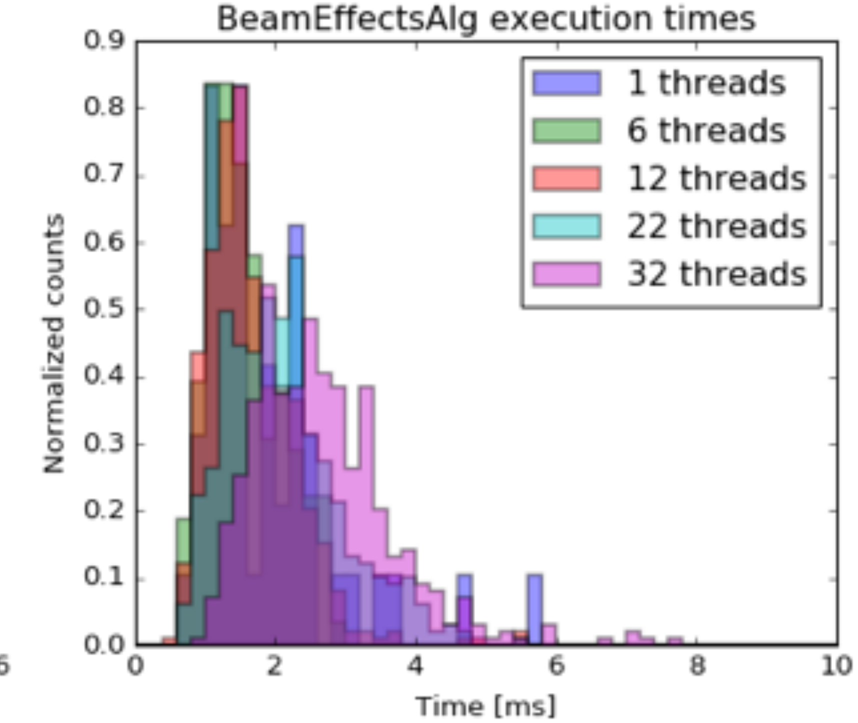
TTBar sample algorithm timings





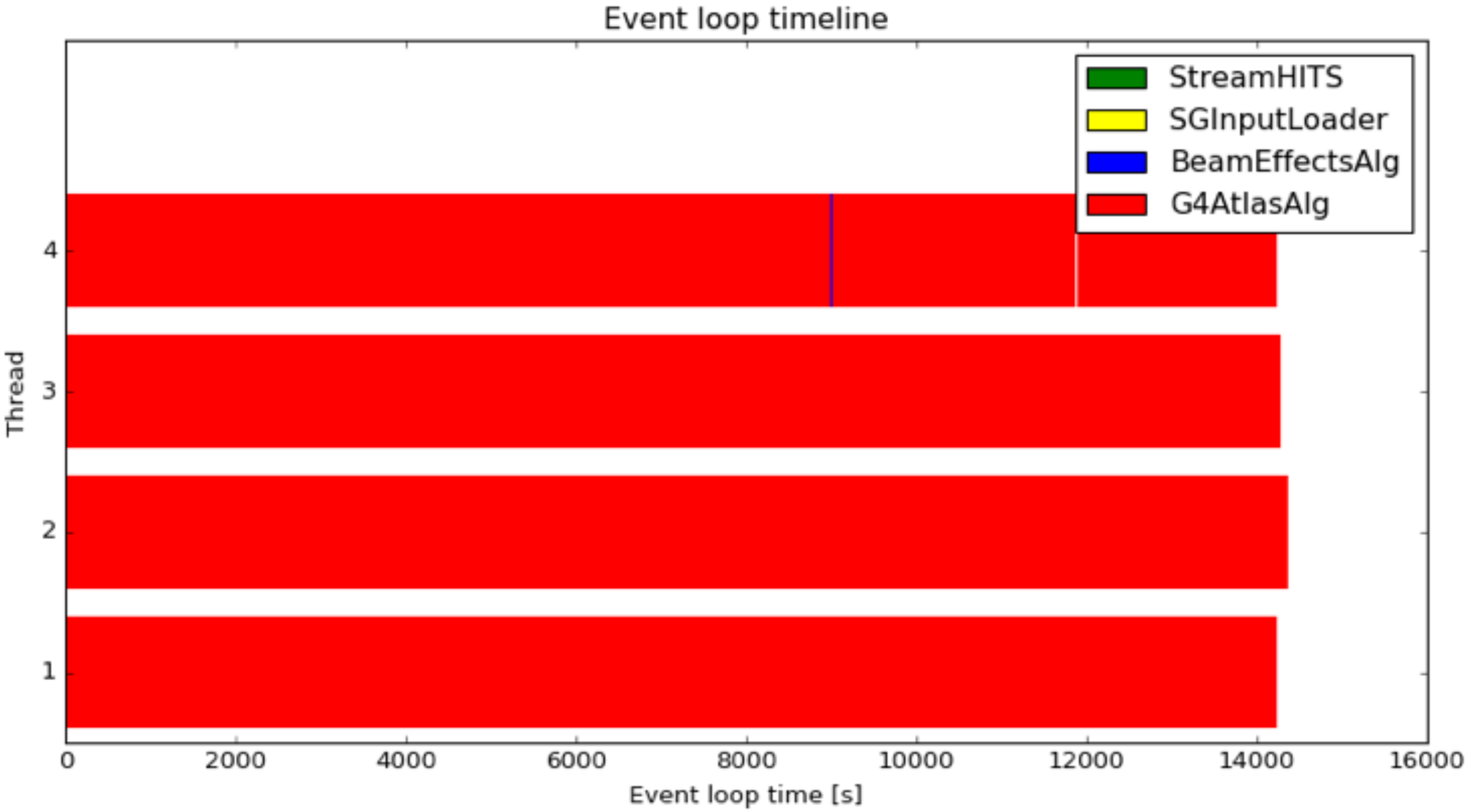






- Slowdown in hyper-threading seems to be almost entirely in the algorithm executions
  - Scheduler not getting overwhelmed

TTBar sample timeline



- Very red :)
- You can find my notebook here:
    - https://github.com/sparticlesteve/g4hive-analysis/blob/master/G4HivePerfAnalysis.ipynb

Knights landing

- One of our major motivating targets is NERSC's Cori Phase II, which will be composed of Intel KNL Xeon Phi machines
    - 72 cores per chip
    - **~100 GB of DDR, 16 GB of high-bandwidth MCDRAM**
- I have early access to a KNL cluster with Intel
    - Will be used for extensive testing of G4AtlasMT
- The good news
    - **Athena and G4AtlasMT run successfully!**
        - Pacman kit installed on a "normal" machine; copied to KNL
        - Practically runs out-of-the-box; ***x86 compatibility confirmed***
- The bad news
    - Processing is currently super slow
        - ~25 min per TTBar event
    - Seems mostly unaffected by usage of MCDRAM
        - Oddly, pure-G4 app showed significant speedup with MCDRAM
    - There's probably a funny bottleneck => *under investigation*

Conclusions

- G4AtlasMT is finally at a stage where we can call it a nearly complete simulation
  - Definitely realistic
- Next development things to be done
  - Continue progress on fast simulations
  - Deep-dive into multi-threaded ISF
- Performance studies heavily underway
  - Results look very nice so far (on traditional machines)
  - Throughput and memory performance is good
- First tests on Knights Landing architecture!
  - Something funny going on, though, which needs further investigation

The end!

Intel Many-Integrated-Core architecture

- A “supercomputer on a chip”
  - Lots of threads, wide vector registers, with low power footprint
  - Particularly suited to highly-parallel, CPU-bound applications

- The Xeon Phi product line:

Knights Corner (KNC)	**Knights Landing (KNL)**	**Knights Hill (KNH)**
current generation	coming soon	maybe 2017
57-61 Pentium cores (~1GHz)	72 Airmont cores (3x faster)	60-72 Silvermont cores
6-16 GB on-chip RAM	8-16 GB MCDRAM	???
coprocessor only	up to 384 GB RAM	
	host or coprocessor	

- Supercomputers:

  - Tianhe-2 @ NSCC-GZ
  - Stampede @ TACC

  - Cori @ NERSC
  - Theta @ ANL

  - Aurora @ ANL