ATLAS: Gaudi Requirement and Usage

Charles Leggett

Gaudi Workshop
2017-09-25

ATLAS: Where We Are

- ATLAS has begun migration of development branch (release 22) to AthenaMT, which uses Gaudi v29 (currently, almost)
  - Athena production branch (release 21) will stay with Gaudi v27 with minor updates for bugfixes, etc
    - no major updates or feature requests
  - Athena development branch will track Gaudi master for some time
    - is likely to cause increasing amounts of backward incompatbililty
- Maintain our own AtlasGaudi git repository for ATLAS specific tag schemes and to decouple from the Gaudi MR schedule
  - probably an error, as witnessed by the pain of submitting MRs to both repos
  - will probably revisit this
    - ATLAS branch in main Gaudi repo?
    - no impact on developers

ATLAS AthenaMT Migration Schedule

	Core	User
2015	Baseline Functionality	Basic Demonstrators
2016	Most functionality available	A few algorithms / services
2017	All core functionality available	Start migration
2018	Performance improvements and optimization	Bulk of migration
2019	Bug fixes	Continuing bulk of migration
2020	rewrite from scratch	Validation
2021	**Run 3 Production**	

OK
(mostly)

not so OK

ATLAS Migration to MT

- ATLAS has started its migration to MT
  - require a critical mass of components before things can be tested
- usage of DataHandles: no direct access to the EventStore
  - in general, the easiest
  - refactor UpdateHandle access pattern
- removal of Public AlgTools
  - either make them Private or Services
    - making them all Private can have significant memory implications
  - can require massive re-writing of interfaces
- Thread safety / concurrent event management for Services
- Conditions
  - callback functions into Condition Algorithms
- Detector Description
  - same model as Conditions

this is the hard part

eg: Trigger uses 823 Public Tools

- Trigger
  - Event Views, Scheduler, and Sequences
  - many, many Trigger chains == many many Algorithms
  - very complicated setup
    - need tools to understand and verify configuration
  - See talk by Tomasz on Tuesday

First Uses of AthenaMT

- We have already started seeing our first real-world usage of AthenaMT
  - G4Hive: Geant simulation with AthenaMT
  - demonstrable memory savings
- Excellent testbed for various core features, without being overly complex
  - only one "Algorithm"
  - tight integration with G4
  - good example of special thread initialization
- Allows us to investigate MT solutions for some critical Services in a well controlled environment
  - multithreaded RNG
    - how to predictably distribute generators to components
    - seeding issues: what to seed with, overuse of seeds
    - verifiability
  - concurrent I/O
  - job monitoring

Disruptive Updates to Gaudi

- At some point, we're going to need a feature freeze on Gaudi for release 22
  - how much backward incompatibility can we tolerate?
  - there's already a big difference between user code for release 21 and 22, and will be much more so when migration is complete
    - automatic sweeps in git from 21 to 22 will become harder
- C++: would be nice to settle on gcc 7 and c++17 standard
- Many framework level changes are hidden from Users
  - *eg*, we can swap out implementations of the event store or condition service as long as the DataHandles look the same
  - or at least easily disguised or fixed
    - git is our friend
- Some critical components are more tightly integrated with framework level implementations
  - eg, Trigger

Validation

- At some point, we're going to want to move the production branch to AthenaMT
  - it's a challenge to get developers to work on AthenaMT migration when they can ignore it
  - always better to maintain one branch instead of two

- Multi-stage validation of results of AthenaMT
  - release 21 vs 22 (no Scheduler, identical jobOpts)
  - release 21 vs 22 (Scheduler, 1 thread, no data dep handling)
    - need easy way to transform current release 21 job into one that uses the Scheduler, ignoring data deps, same Algorithm ordering
  - release 21 vs 22 (Scheduler, 1 thread, w/ DataHandles)
  - AthenaMT in serial (Scheduler, 1 thread, 1 concurrent event) vs various MT combinations

- Tools needed to aid validation
  - RNG correctness?
  - output histograms?
  - bitwise comparison?

Tools for Performance Monitoring

- Major claim of AthenaMT was decreased memory usage for similar event throughput
  - will need to demonstrate this
  - a lot of prototype/initial implementations of various components do the opposite
- Tools to monitor memory usage and efficiency
- Tools to ease debugging of thread issues
  - compile time static checkers
  - code verification
  - runtime analyzers
- Critical path analysis for Scheduler
  - many ways to schedule a given event. which is best
  - different optimizations for different types of events
  - visualization tools essential
    - help understanding runtime stalls / failures
- Optimizations for different hardware
  - 16 cores *vs* 160