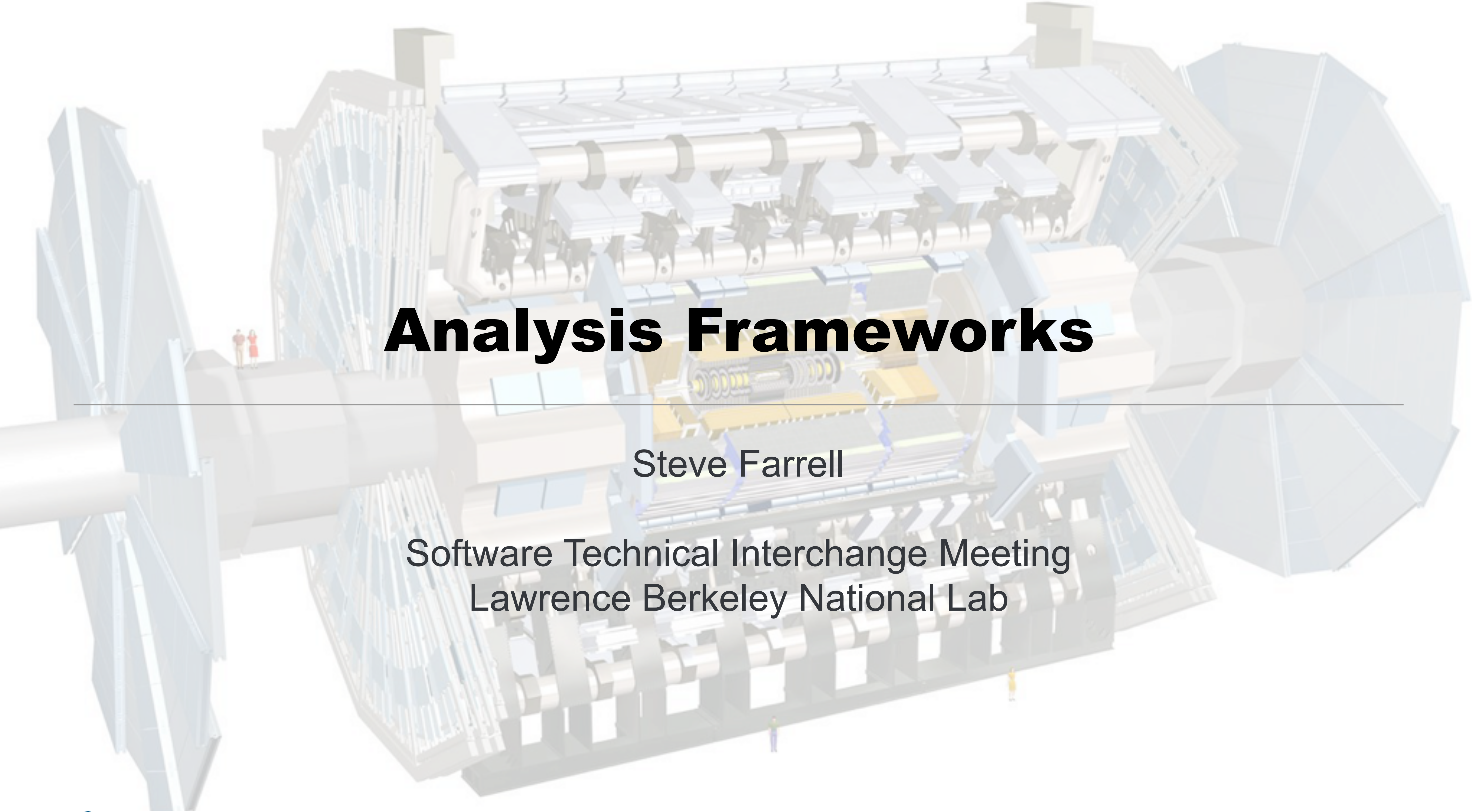

Analysis Frameworks

Steve Farrell

Software Technical Interchange Meeting
Lawrence Berkeley National Lab

Ernest Orlando Lawrence Berkeley National Laboratory

Nov 10, 2015

Introduction

- Good progress is being made on AthenaMT, in a wide range of areas
  - People seem to be aware of the magnitude and importance of the work
- But of course it’s not all about trigger/reco/simulation
  - Quality of analysis software is extremely important for ATLAS’s scientific mission
- The big questions:
  - How can we continue to run analysis software effectively alongside/within AthenaMT?
  - How can we leverage new features/capabilities to improve our analysis software for the future?

Background

- The FFReq analysis addendum described some requirements of the framework for analysis
    - https://cds.cern.ch/record/2013708?
- Its main points:
    - Framework components should be instantiable outside the framework
    - Components/tools should be configurable via a standardized file format like JSON
    - Framework should support multiple submission backends
- This discussion today continues the one from the ATLAS frameworks workshop in July at CERN
    - https://indico.cern.ch/event/394278/session/0/contribution/7/attachments/789344/1081914/fraat_statusReport3.pdf
- Topics to discuss
    - Analysis framework landscape
    - Dual-use tools in the future framework
    - Extending the dual-use concept
    - Concurrency in analysis

Frameworks used for analysis

- AthAnalysisBase
- AnalysisTop
  - common executable in RootCore
  - application steered completely by config file
- HWW xAOD framework
  - 2 stages: PxAOD production in Athena, analysis in RootCore
- CxAOD framework
  - RootCore framework produces CxAOD files
  - Analyzed with a toolkit (I think..?)
- xTau Framework
- SUSYTools
  - a super-tool with several high-level methods (e.g. fillElectron)
- QuickAna
  - a super-tool toolkit and tool scheduler

Dual-use tools in the future framework

- Used extensively in analysis to do all kinds of work
  - Object corrections, selections, weights, corresponding systematics
  - Calculation of MET, overlap removal, taggers
- The dual-use tool software layer was very useful for CP groups providing recommendations via tools, but also useful for analysis tools/frameworks
  - Both QuickAna and SUSYTools work in either RootCore or AthAnalysisBase
  - Some frameworks spread across both RootCore/ AthAnalysisBase; able to move pieces around
- Luckily, the AthenaMT infrastructure changes shouldn't have a huge effect on our ability to maintain this capability

Expanding the dual-use toolkit

- Many folks are interested in the idea of extending the dual-use concept to other types of framework components and services
- All "frameworks" have to tackle the same kinds of problems
  - Component management
  - Scheduling
  - Event data management
  - Configuration
  - I/O
- Can we develop more common solutions to tackle these across different analysis codes?
  - Share design patterns, principles, implementations
- Possible gains
  - Framework becomes more flexible/extensible in general (not just for analysis)
- There are multiple ways to do this
  - Expand the current dual-use pattern to other component types
  - Make components useable outside the framework

Directly instantiable framework components

- Another requirement from the FFReqAD: framework components should be directly instantiable
  - This way we can use them outside of the full framework, standalone or in another framework
  - Difficult/impossible to do currently because of framework dependencies
- This opens the door to interesting design questions about Athena
  - Is it possible to better decouple the framework elements?
  - Should Athena become more like a toolkit?
  - Would we stand to gain general improved flexibility?
- Algorithms: a good possible use-case?

Dual-use component pattern

- Another way to do things: apply the dual-use tool design pattern to other components like *services* and *algorithms*
  - Interfaces and base classes with compiler switches to swap out the Athena dependencies
- What would a dual-use algorithm look like?
  - IAsgAlg interface and AsgAlg base class
  - Replacements for the Athena dependencies
    - e.g., we already have SgTEvent
    - probably need a standalone version of the VarHandles
- What could we do with a dual-use algorithm?
  - Run it in a non-Athena, or even ROOT-based framework
  - Maybe merge with EventLoop Algorithm
  - Integration with PROOF, etc.
- Would this pattern also work for other types of components?

Configuration

- Many frameworks and analysis codes use some kind of configuration file to set properties on tools
  - Some CP tools even rely on them if the configuration is complex enough
  - Formats vary from custom formats to TEnv and more
- One item from the FFReqAD was that the dual-use tools (and even the larger framework) should support some kind of config file in a format like JSON
  - Could allow to harmonize these cases across all tools with a common format and parser
  - Could be used as a layer in the standard framework configuration, generated from current job options and python configurables.
- Or just use alternative/extended version of the JO svc?

What else could be useful?

- If we're talking about making the Athena framework more like a toolkit, it's also useful to take about making our ASG toolkit more like a framework
  - Analyzers clearly *like* to use *some* kind of framework (just not Athena…?)
- What about component management?
  - Currently no uniform factory method for implementing tools, etc. outside of Athena
  - FFReqAD says that algorithms should be able to directly instantiate their private tools outside of the framework
- What about configuration management?
  - Adding support for simple config files is one thing, but some JobOptionsSvc-like infrastructure to handle it and apply it to tools could also be useful

Event processing and concurrency

- FFReqAD: framework should be backend agnostic
  - job submission is separate from framework
  - support submission to PROOF
    - conflicting requirements?
  - New types of computing resources may be possible for running analysis
    - Cori phase 1 with the burst buffer
    - Event-service-like processing

Concurrency

- I don't expect multi-threading to be necessary for analysis
  - Multi-process is still the de-facto way to parallelize analysis code
  - But we can still keep doors open, considering the amount of work it takes to implement solutions
- Whatever code we write that is supposed to work in Athena should *at least* run safely in AthenaMT
  - I.e., things should be thread-safe
  - Systematics tools actually hold *state* which is updated frequently in the event loop
  - As long as systematics tools are *private* tools, things should be thread safe in the AthenaMT event loop
  - Need to add some protection to the SystematicRegistry
- Thread safety outside of this private-tool pattern could still be an issue
  - But I'm doubtful we'll be recommended analyzers to implement their own implementation of multi-threading

QuickAna implementations

- In QuickAna, as in other codes, we have our own custom solutions to these problems
  - Some are still in (slow) development
- Algorithms and scheduling
  - Our algorithms are really just AsgTools with some extra machinery
  - Our scheduler resolves data dependencies in a way that's not too dissimilar to Athena
    - A little more systematic-oriented, though
    - Also, scheduling is fixed early on; not dynamic
- Event processing
  - QuickAna can be wrapped in an Athena algorithm or in an EventLoop algorithm, which allows us to submit to respective supported backends
  - We can even now run QuickAna on Edison @ NERSC, in preparation for running on Cori Phase 1

QuickAna implementations

- Component management and configuration (work in progress)
  - A ConfigSvc takes the place of the JobOptionsSvc, holding properties mapped by tool names
    - Currently can be filled directly, but we envision filling it with a config file
  - An AsgToolSvc handles the factory creation of the tools, using the ROOT dictionary
    - Also configures/initializes the tool automatically via the ConfigSvc
  - Once integrated with a smarter ToolHandle, this becomes quite similar to the infrastructure in Athena
    - An “algorithm” retrieves its private tool; the handle queries the AsgToolSvc which will create it on-the-fly and configure it via the job options in the ConfigSvc
- We’d be happy to have more general implementations of these solutions that we can simply adopt