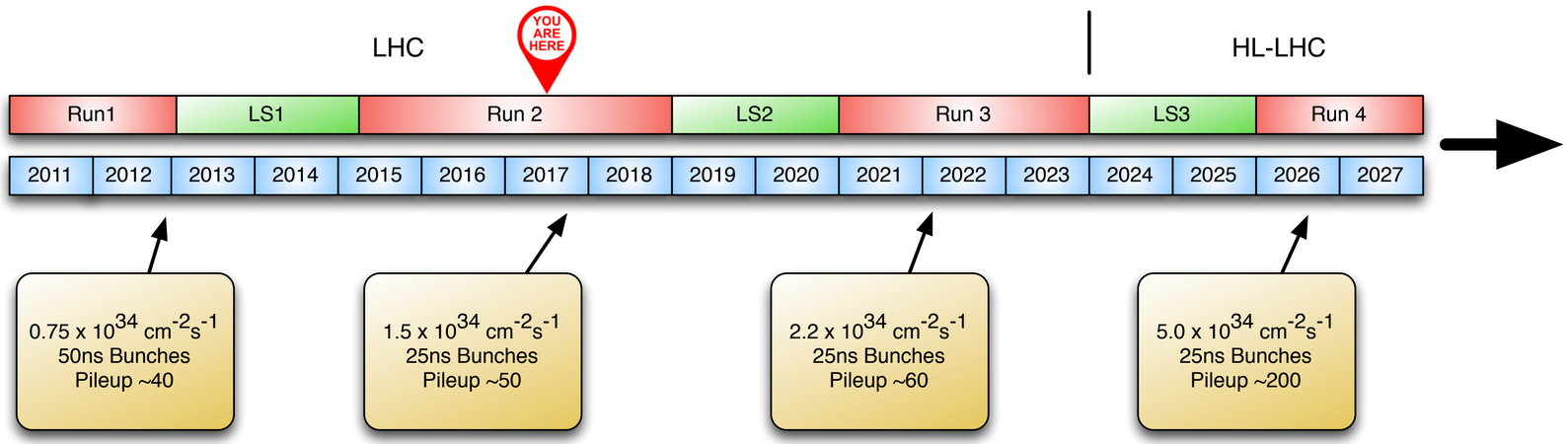
Modernising ATLAS Software and Metadata

Graeme Stewart, on behalf of the ATLAS Collaboration

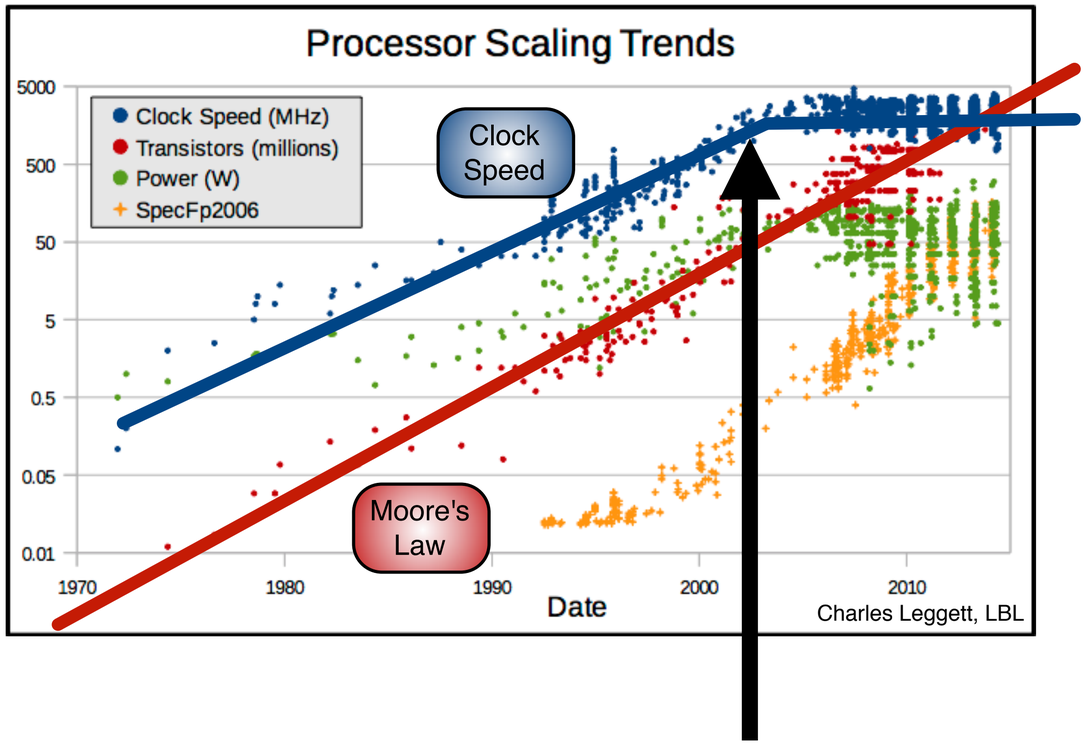
Challenge of High Luminosity LHC



- High luminosity LHC will deliver about x10 increase in luminosity over what we have today to ATLAS
  - Needed for precision physics program and to increase the discovery reach of ATLAS
- Pileup of 200 means a great increase in event complexity
- HLT output of 10kHz increases rate
  - Complexity x Rate = Challenge
- The challenge encompasses not only reconstructing the data, but organising it, accessing it and keeping it accessible for physics

Hardware Evolution

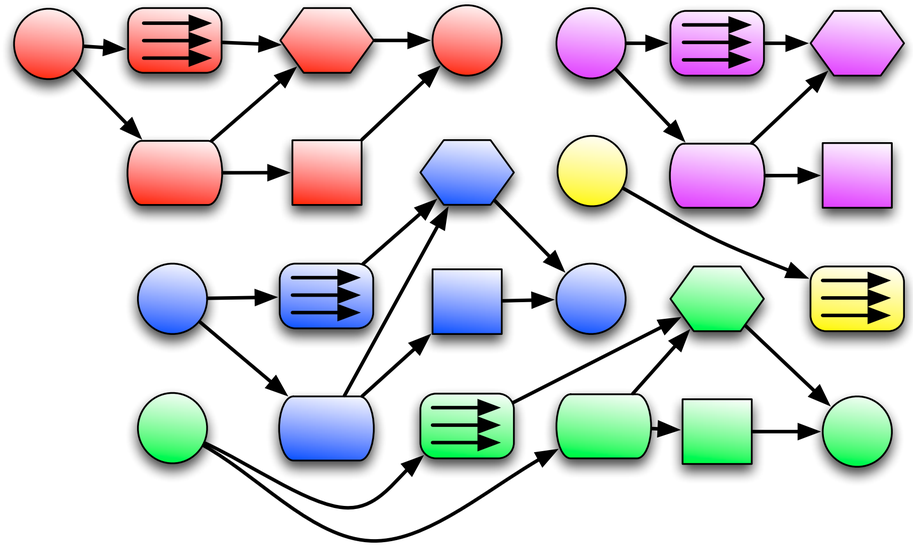
- No significant increases in single core performance in more than a decade
- Moore's Law continues to increase transistor density
  - Performance increases come from multiple/many cores and wide vector registers
  - Not to mention alternative architectures (GPUs, FPGAs)
- These hardware improvements are not trivial to turn into performance improvements in real world HEP code
- Memory wall hits ATLAS the hardest right now
  - 100M channels, complex detector geometry and magnetic field
  - Serial reconstruction hits 4GB of memory usage
- Need to move to multi-threading for most efficient memory sharing and targeting lower power cores and many core architectures

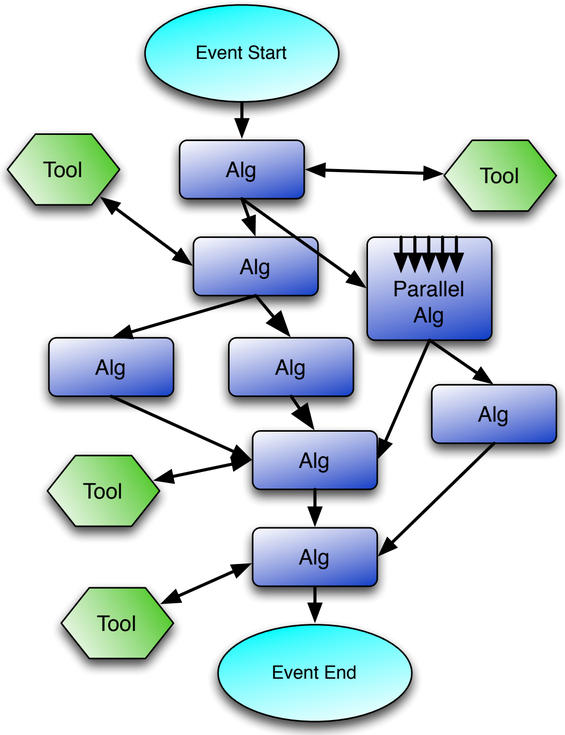


ATLAS software designed here — in its day a great fit for hardware, but not any longer

AthenaMT

- Try to exploit concurrency both between events and within events
- How to manage a multi-threaded migration without requiring developers to become experts in writing thread safe code?
  - Avoid problems primarily through upgrading the framework itself
  - Upgrade Athena to Athena Multi-Threaded, based on evolving the common Gaudi base that we share with LHCb
- This requires strong guidance as to what programming patterns are permissible and advised in a multi-threaded environment, e.g.,
  - No mutable statics or global variables
  - `const` is definitely your friend
- Goal is that a “normal” algorithm that reads data, transforms it and writes out a new collection needs **zero** knowledge of threading



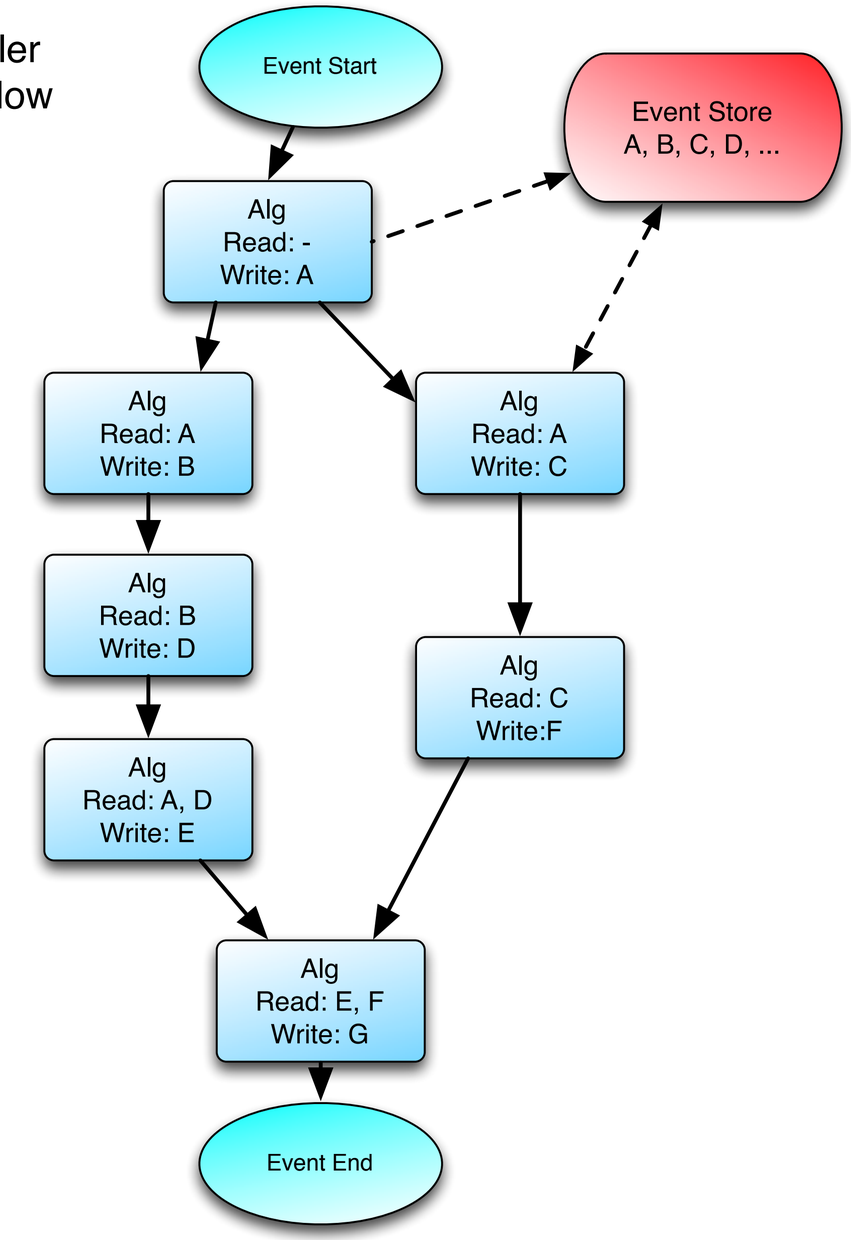


Exploit concurrency between events (upper diagram) as well as concurrency within events (upper and lower diagrams)

Data Dependencies

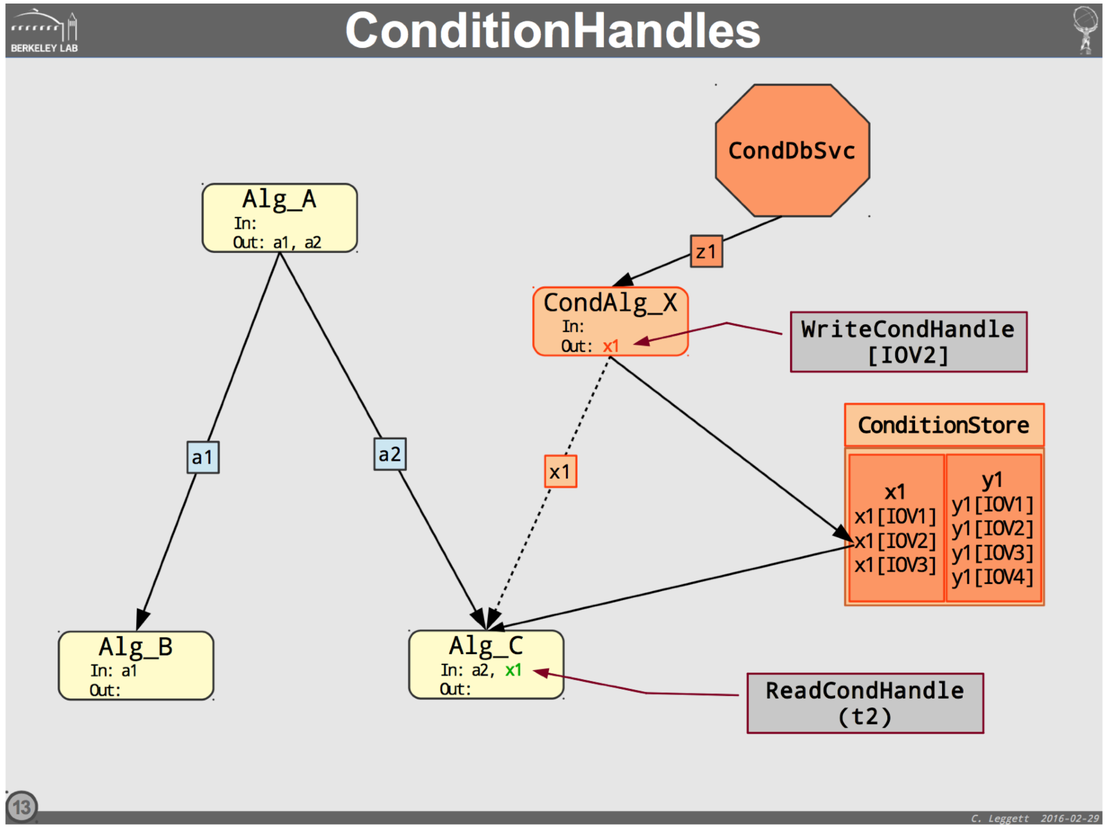
- The first step to avoiding data races is to ensure that algorithms' data dependencies are expressed properly
  - Abstract all access to the event store though a data handle
  - If an algorithm writes data object X then no consumer of X can run until that algorithm has completed
- We also ensure that all tools used by algorithms are private
  - Public tools were often used as a backdoor to circumvent the event store and would break in a multi-threaded program
- Special handling of decorations in ATLAS's reconstruction format (xAOD) have been added
  - Treat each decoration as a different object, declared explicitly to the scheduler
  - Modification of any existing part of an object is forbidden

Scheduler Graph Flow



Non-Event Data

- Event data is not all that's needed for event processing
- Non-event data can come from state that has an independent lifetime from a single event (an Interval of Validity)
  - High voltage corrections
  - Noise bursts
  - Magnetic field ramps
  - Geometry shifts
- We have developed the concept of conditions handles that provides a similar abstraction as data handles
  - Conditions store handles different intervals of validity for objects
  - 'Raw' data can be processed before use by reconstruction algorithms



Retooling and Migration

- ATLAS has 4M lines of C++ that need to be migrated to be thread-friendly (plus 1.5M lines of supporting python)
- Much of ATLAS's infrastructure for managing that code and building releases in 2015 was out of date
  - SVN for code management
    - Organised in a package hierarchy that did not reflect how code should actually be built
    - Therefore requiring an additional Tag Collector component to assemble a release
  - CMT build system, HEP specific tool developed in the early 2000s
- In addition to using out of date tools, our builds were complicated, hard for ATLAS users to manage, slow and dependent on CERN specific infrastructure
  - It was time to retool, taking advantage of the advances in new tools from open source and industry

CMake

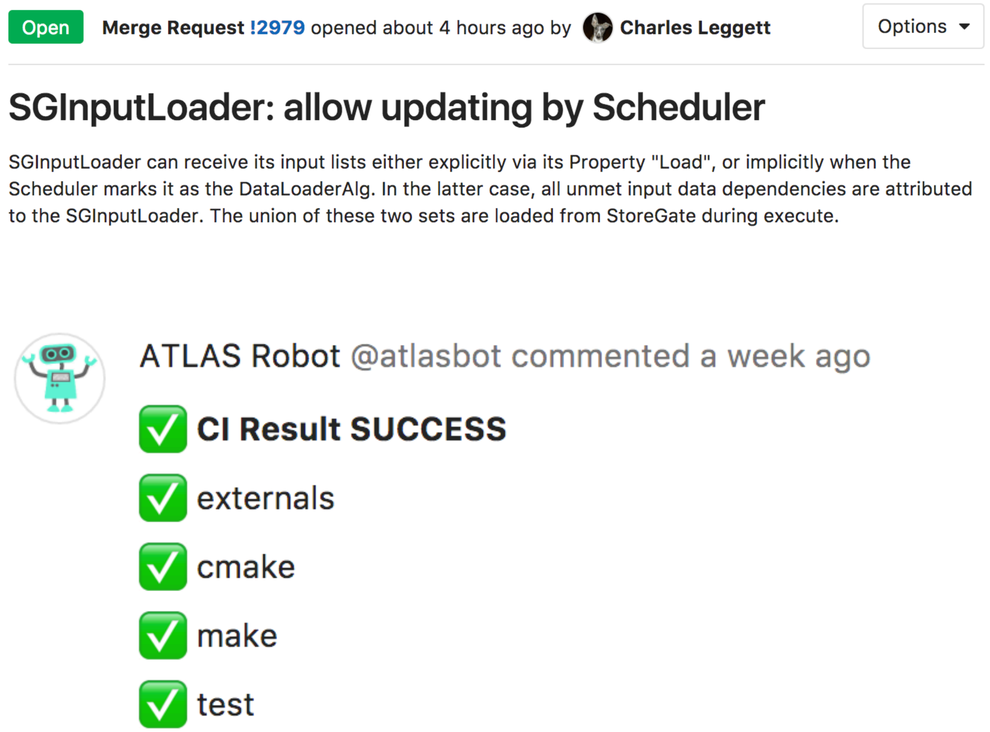
- CMake is a cross platform build tool
- Widely used now within HEP and outside of HEP
  - A lot of help available online
- Modular structure allows for collaboration in the code used to discover and configure external dependencies
  - Easy to find contributions that work for us
- Automatic translation script was written to convert from CMT files to CMake that took most of the load of the basic migration
- First live release with CMake was made summer 2016
- Performance is excellent
  - Time to configure full Athena build, with 2100 packages, is around 8 minutes
  - Parallelism allows a machine to exploit many cores during the build process (better than CMT)
    - 9 hours to about 3 hours (coupled with simplifications in the build process)

git and gitlab

- Open source community moved to distributed version control
  - Of which git is the most popular, also in the HEP community
- ATLAS git migration was performed in early 2017
  - Custom script imported release package tags (i.e., code that actually was in a numbered ATLAS release) into git
    - Trimmed out a lot of useless files and abandoned packages
  - Helped by git's very efficient storage the repository size shrank from 62GB to 220MB
    - Can be cloned in under a minute, even over long distance connections
- Combined with CERN's GitLab instance for social coding
  - Allows developers to work independently, build and test changes
  - Propose *merge requests* into the main repository

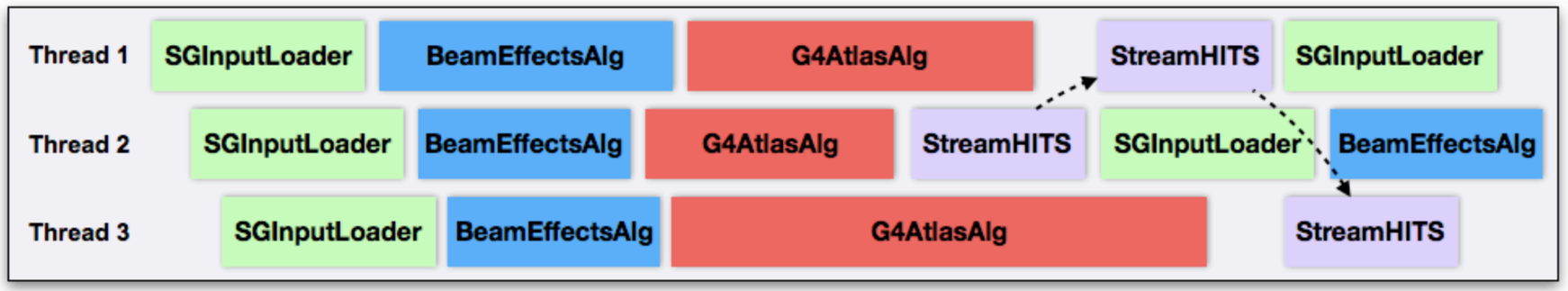
Code Review and Continuous Integration

- Code update requests are made in GitLab from a developer's private fork
- Merge request is first checked by Continuous Integration system, powered by Jenkins
  - Configure, Build, Unit Tests, (Short) Integration Tests
- The the code is reviewed
  - Level 1 shifter does basic checks
  - Level 2 does deeper check of design and quality
  - Experts can be called in when needed
  - Finally, Release Coordinator accepts the change, if good
- Next steps are to broaden the range of CI tests — more automated checks of code quality

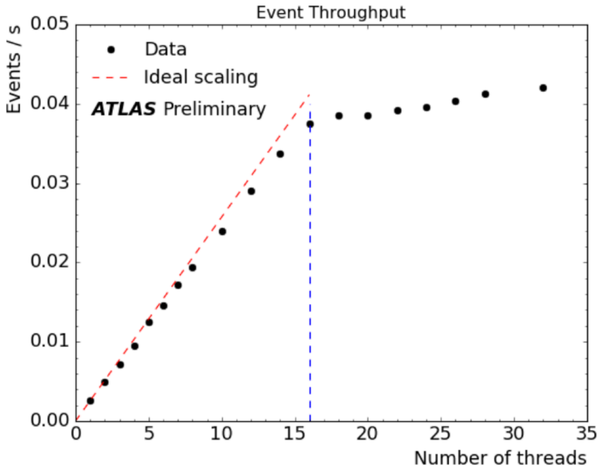


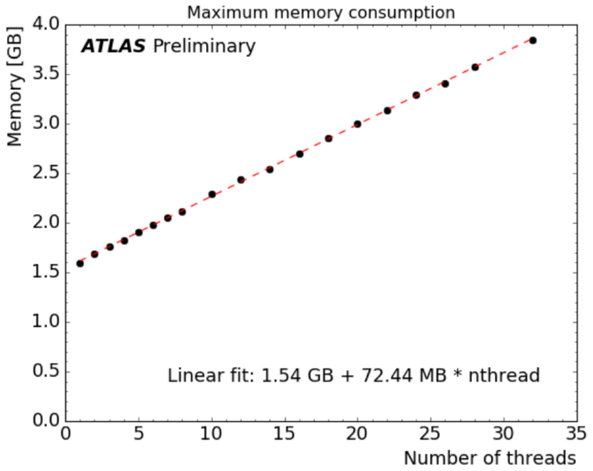
AthenaMT Simulation

- MT reconstruction is a large target that touches many domains and requires a lot of work to get running
- MT simulation is an easier target, as it leans on threading improvements already made in Geant 4 and relies on a smaller set of ATLAS packages
  - Also, CPU heavy, i/o light workflow is a good match to many-core architectures
  - Even this is still considerable work, requiring significant redesign of many pieces (e.g., sensitive detector code)
- Each thread runs one G4 simulation job, with some serialisation in the i/o layer
- Use thread local storage to have a local workspace for each event

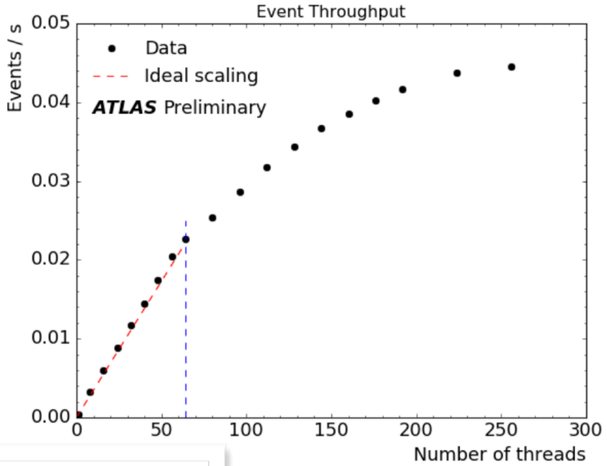


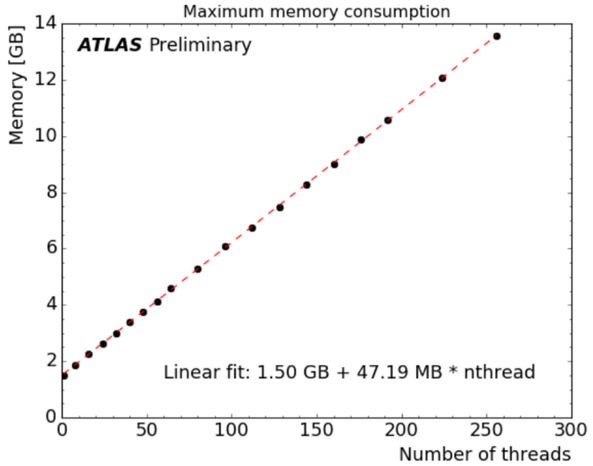
AthenaMT Simulation





Xeon Server, Ivy Bridge



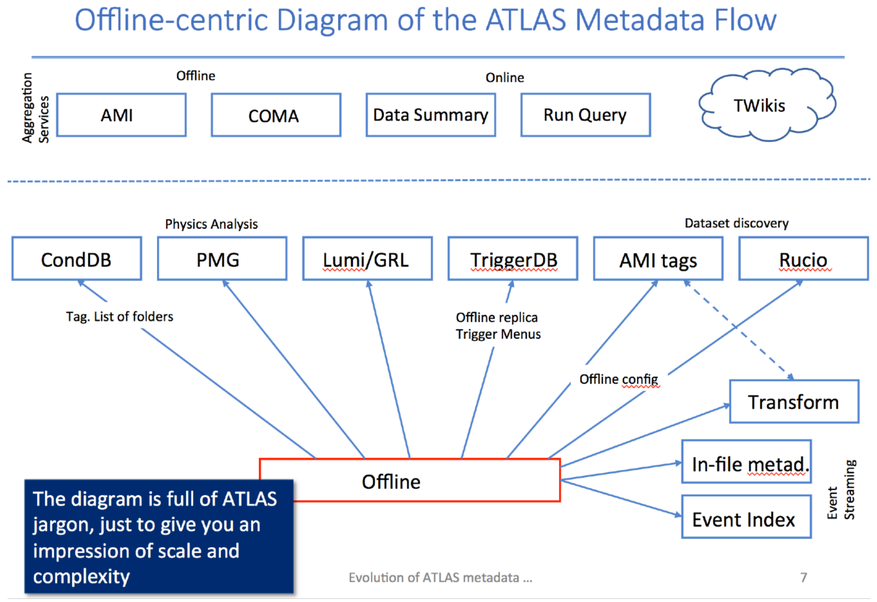


Xeon Phi, Knights Landing

- Scaling on server CPUs is excellent
  - Almost perfect throughput scaling, excellent memory scaling
- Also good scaling on many-core Xeon Phi (Knights Landing)
- Throughput on KNL suffers from some performance issues that are under investigation
  - Illustrates that new architectures do need tuning

ATLAS Metadata

- ATLAS has a mature metadata system in place for Run 2
  - Detector conditions
  - Data and MC processing information (task configuration)
  - Data storage and management information (logical and physical files)
  - Analysis information and physics publication information (from cross sections to which data is used in a publication)
- Stored in various backends and complemented by a whole number of different interfaces and web front ends
- Overall management and adding additional information coherently is a challenge



Sketch showing the sheer number of different systems involved in ATLAS metadata in Run 2

Metadata Evolution

- Reworking of metadata and planning for an adiabatic migration for Run 3 is now in progress
- Idea is to centre metadata around the *event* itself
  - Files, datasets, run periods all composable as sets of events
    - With links between events at different stages of their processing (RAW, xAOD, DxAOD)
  - A better match for the exploitation of highly dynamic computing resources in the future
    - ATLAS Event Service or ATLAS@HOME resources
  - Dynamic collections of events become possible
    - Allows a whiteboard of collections that can be updated with information as data analysis evolves
    - Including versioning to manage the evolution
- Requirements gathering and planning taking place now

Conclusions

- ATLAS has ambitious plans for its software in the next years
  - Targeting Run 3 in the first instance, but with an eye to scaling also to Run 4 and HL-LHC
- Infrastructure upgrades are in place and we are benefiting greatly
  - Big improvements in developer independence, potential productivity and code quality
- Multi-threading is an essential improvement to target modern CPU architectures
  - Hard to achieve with millions lines of serial C++ code, but we are making good progress
  - Most framework elements in place with algorithmic code is starting to move towards thread friendliness
  - Simulation running in multi-threaded mode is now in very good shape
    - Ready for physics validation and performance optimisation
- Metadata improvements will help us to scale better towards HL-LHC and to adapt to dynamic resources