MICE Analysis User Software

Durga Rajaram
Illinois Institute of Technology, Chicago

Adam Dobbs
Imperial College, London

October 10, 2016
CHEP 2016
San Francisco
OUTLINE

• MICE
• Software Requirements
• Implementation
  – Design
  – Framework
  – Online
• Performance
• Conclusions
**Muon Ionization Cooling Experiment**

- Muon cooling: essential for future high-luminosity $\mu$-colliders & high-intensity $\nu$-factories
- $\mu$ from $\pi$ decays have large emittance & must be cooled (reduce phase space volume)
- Traditional beam cooling techniques too slow due to the short $\mu$ lifetime
- Ionization Cooling is the only practical means:
  - Reduce momentum by $dE/dX$ in absorber, followed by RF reacceleration to restore $p_{||}$
  - Design, build & commission a realistic section of cooling channel to precisely measure emittance reduction – Lessons => the design of a full cooling channel for a $\nu$ Factory/$\mu$ Collider
MICE Software Goals

• MICE is both an accelerator physics & a particle physics experiment
  – Beam simulations, emittances, transfer matrices, Twiss parameters
  – Traditional HEP detectors: simulation, track reconstruction, particle identification
  – Need common software scope

• Online reconstruction & monitoring during data-taking

• Wide range of geometries, configurations to book-keep

• Framework for analysis tools

• Code testing, Issue tracking, documentation

• Long-term maintainability: MICE is built & operated in stages
Note that reconstruction is same regardless of MC or real data
MAUS:
MICE ANALYSIS AND USER SOFTWARE

- Design inspired by MapReduce

- **Map**
  - Operate on a single event
  - Can run in parallel
  - e.g. Simulate, reconstruct

- **Reduce**
  - Operate on a collection of events
  - e.g., Summary histograms
MAP: SPILL 1
SIMULATE  DIGITIZE  RECON

MAP: SPILL N
SIMULATE  DIGITIZE  RECON

ALL SPILLS
REDUCE: PLOT..

INPUT  OUTPUT
MAUS: MICE ANALYSIS AND USER SOFTWARE

• MapReduce
  – *Technically, in MAUS:* Input-Transform-Merge-Output

• **INPUT:** Read in data
  – DAQ data file, I/O stream, beam library for Monte Carlo

• **MAP:** Process spills & return modified data
  – A spill is the primary data block & consists of several event triggers
  – Monte Carlo simulation, Detector reconstructions
  – Mappers have no internal state & can operate in parallel

• **REDUCE:** Process accumulated data from *all* mapped spills
  – Summary histograms, run performance plots, etc

• **OUTPUT:** Save data
  – Write out in ROOT/JSON format
MAUS FRAMEWORK

• Framework built on plug-in modules
• Developers write modules in C++ or Python
  – Python for higher-level algorithms, or where development time is a factor
  – C++ for lower-level computationally intensive algorithms: particle tracking, fits, likelihoods
  – C++11 support
  – Python-C++ bindings handled by wrappers
• Data representation: ROOT, JSON
  – Default is ROOT, but developers find JSON quite useful for quick debugging
  – Mapper modules are templated to a data type
  – conversion between data types handled by API
  – Significant performance speedup by removing JSON-C++ conversions
CODE MANAGEMENT

- 10-15 developers in the UK, Europe, USA
  - Headed by Adam Dobbs @ Imperial College
- Distributed version control
  - Bazaar repository, hosted on Launchpad
- SCons build system
- QA:
  - Python/C++ style guidelines
  - Unit testing & integration testing
  - Code monitored for line & function coverage: aim ≥ 70% line coverage
- Redmine wiki & issue tracker
- Scientific Linux 6 is officially supported OS
- Several external dependencies
  - Python, ROOT, Geant4, G4Beamline, XBoa …
  - Dependencies built as “third party” libraries during installation; build scripts come with MAUS
CONTINUOUS INTEGRATION

- Unit tests
  - Test individual modules/pieces of code
- Integration tests
  - Test if units work together and with external libraries
- Jenkins CI test server with multiple slaves at RAL and Brunel University
  - Developers run jobs on the test servers, validate & test their user branch before proposing to merge in the mainline trunk
Database

• Varying configurations & running conditions
  – Beam momentum, cooling channel magnets, absorbers, calibrations ….
  – Must be monitored (Controls & Monitoring – EPICS) & stored

• Configuration DB holds
  – Run conditions from data-taking
  – Beamline settings
  – Electronics cabling maps
  – Calibration constants
  – Geometry models

• Postgres DB

• Master DB within control-room LAN, public read-only slave at RAL

• Most APIs in Python, some in C (multi-threaded EPICS does not like Python)

• See talk by J. Martyniak on Wednesday
MAUS Online

- In classic MapReduce, map operations have to terminate before reduction
- However, for online reconstruction:
  - Want to visualize/plot (reduce) continuously as data flows (~after each map operation)
  - Speed was also an issue with more computationally intensive modules coming into MAUS
- New parallel C++ API developed:
  - Interfaces with MAUS modules
  - Jobs distributed by conveyer-like implementation
- Allows either single or multi-threaded
  - Multi-threaded mode during live data-taking
MAUS In Action

• Offline reprocessing & simulation with MAUS performed on GRID
  – Batch production & re-processing on Tier-2 sites
  – MAUS installation for GRID via CVFMS at RAL

• During data-taking, "live" reconstruction happens on a dedicated resource in the control room
  – Reconstruction speed ~ data-taking rate
RECONSTRUCTION PERFORMANCE

Time of Flight

Polarity in the upstream tracker

\[ L = (x_{p_{+}}) - (x_{p_{-}}) < 0 \]  
→ Anticlockwise rotation, normal polarity

Polarity in the downstream tracker

\[ \hat{L} = (x_{p_{+}}) - (x_{p_{-}}) < 0 \]  
→ Anticlockwise rotation, normal polarity
SUMMARY

• MAUS provides a simulation, reconstruction, and accelerator physics analysis framework for MICE
• Implemented based on MapReduce
• Online parallel processing capabilities
• Well-defined & yet flexible framework
• Several industry-standard QA practices adopted
  – Code coverage, continuous integration testing
• Simulation and reconstruction software in place
• Data-taking underway & MAUS is in action feeding analysis
THANKS TO ALL MAUS DEVELOPERS

And many more..