Fermi Large Area Telescope:
Science Analysis Software's Use of Gaudi

Heather Kelly
Fermi Science Analysis Software Group
heather@slac.stanford.edu

http://www-glast.slac.stanford.edu/software

Ancient History: 1995-1999

- Software effort adopted C++ with a handful of developers with varying amounts of C++ experience.
- Development and support on Windows using Microsoft Visual C++, including Visual SourceSafe as code repository
- By 1999, we had
  - Detailed Monte Carlo simulation with hard-coded geometry description
  - Prototype reconstruction code
  - Output to ASCII ntuples
- This was before the proposal

- 2000 Richard Dubois takes the software reins.

Why Use a Framework?

- By 1999, we had plenty of code
  - but it was a maintenance nightmare
    - Needed organization and clear lines of division between components.
- Our MC generator was custom built
  - Planned move to Geant4
  - Looked to re-use existing code where possible
    - HEP offered a number of packages: CLHEP, G4
- Provide one code system for simulation, test data analysis, and flight operations.
- Flexible & Extensible

Gaudi Framework

- **Object Oriented C++ Framework**
  “The implementation of an architecture which defines a structure flexible enough to support all types of physics data processing needs...from simulation to analysis to visualization.”
- **Promotes the creation of small maintainable components which can be loaded at runtime**
  - **Components = algorithms, services, I/O, etc**
- **Division of data from algorithms**
- **Supports Linux and Windows**

Gaudi Framework Contd.

- **Standard interfaces for components**
  Algorithms, Services have defined interfaces
  Can extend and create additional interfaces as needed.
- **Gaudi provides a number of basic services:**
  - **Event Data Service, Transient Data Store (TDS)**
  - **Messaging and Logging Services**
  - **Persistency Service**
    I/O where various output formats may be supported
- **Provides standard event loop**
- **JobOptionsSvc**
  Job parameters are handled via an input ASCII file or optional Python interface.

Example JobOptions

```
ApplicationMgr.DLLs = {"GaudiAlg", "GaudiAud", "GlastSvc",
   "HelloWorldGaudi"};

ApplicationMgr.ExtSvc ={
   "EventSelector/EventSelector","EventLoopSvc" };

ApplicationMgr.Runable=  "EventLoopSvc";

AuditorSvc.Auditors = { "ChronoAuditor" };

ApplicationMgr.TopAlg  ={ "HelloWorld" };

// Set output level threshold (2=DEBUG, 3=INFO, 4=WARNING,
   5=ERROR, 6=FATAL )
MessageSvc.OutputLevel  = 3;

ApplicationMgr.EvtMax    = 10;   // events to be processed
EventPersistencySvc.CnvServices ={"EventCnvSvc"};
```

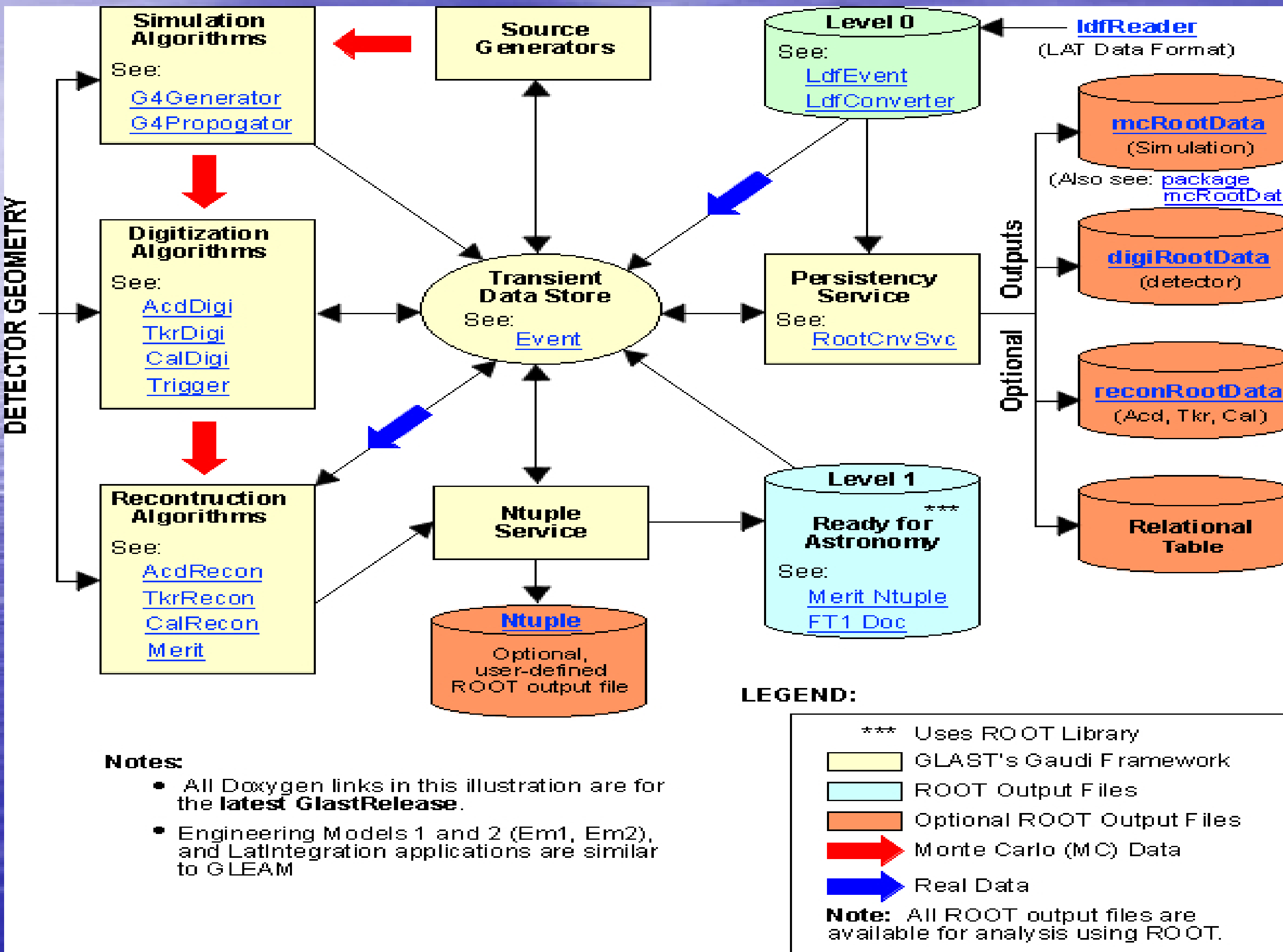
Migration

- Took a few months to get our code working within the Gaudi framework
- Most of the work involved using the TDS.
  - Had to extract the data that was embedded in our existing algorithm classes,
  - And figure out how to use the TDS and Event Data Service correctly
- Skipped using the PersistencySvc, in favor algorithms for ROOT I/O

By 2005…

- ~25 developers distributed across 9 time zones
- Support Windows & Red Hat Linux using Code Management Tool (CMT) as our build system
- ~35 MB (169 MB by 2010) of source code for:
  - Detailed Monte Carlo Simulation using Geant4
  - Reconstruction and Calibration algorithms
  - I/O infrastructure for MC as well as test data to ROOT ntuples and full MC, digitization, and reconstruction data.

Gleam Data Flow using Gaudi



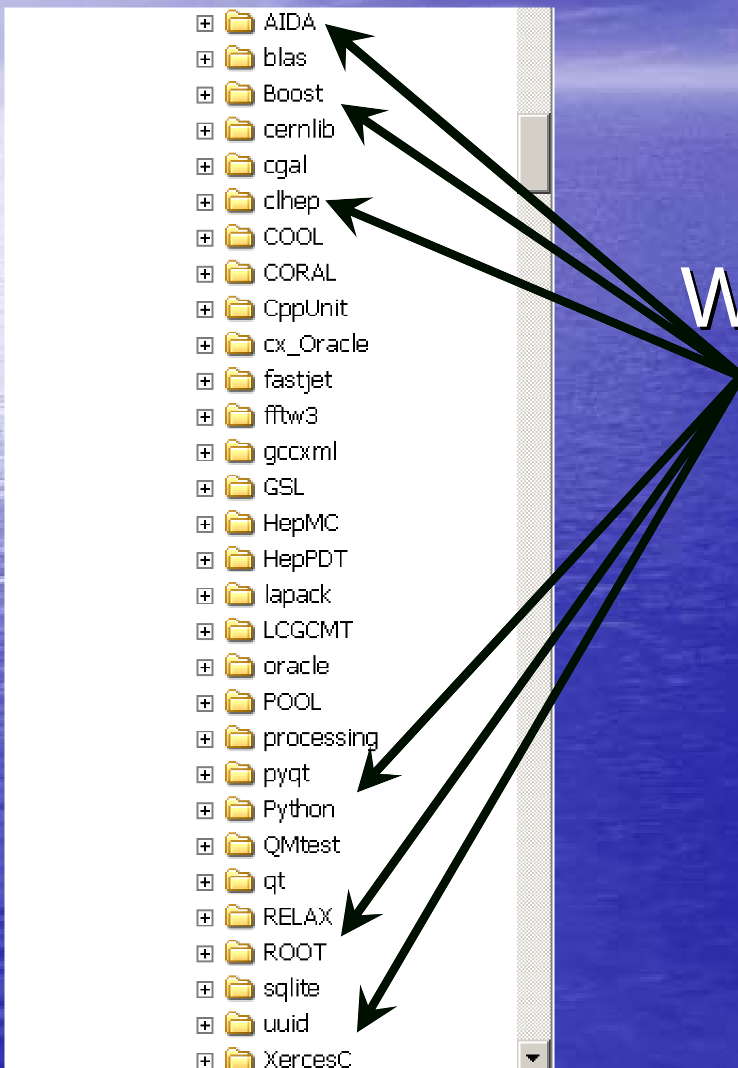
How does Gaudi Play with Geant4?

- Both have event loops, and we need to utilize Gaudi’s event loop.
- We usurp G4’s RunManager, and ask for one event at a time through our G4Generator algorithm.

Fermi uses Core Packages of Gaudi

- GaudiKernel – defines basic interfaces
- GaudiSvc – basic services including event loop and TDS
- GaudiUtil – helpful utilities for loading libs at runtime
- GaudiAlg - defines Gaudi Algorithms
- GaudiAud – auditors, such as ChronoSvc
- May be adding GaudiPython soon
- Gaudi v21r7, includes and binaries ~350 MB not counting their ROOT external

Gaudi Has Many Externals



We use these

- Gaudi provides many interfaces, many Fermi does not need.
- In the interest of simplifying building Gaudi, we limit what externals we use.

Native Gaudi Components We Use

- Event Data Store (EDS)
- Detector Data Store (DDS), for our calibration data
- ChronoSvc
  - monitors the cpu usage for all algorithms and provides a report at the end of the job
- JobOptionsSvc
  - setting of runtime parameters for all components as well as determining the sequence of execution.
- MessageSvc
  - Logging
    - log << MSG::INFO << "Hello World!" << endreq;

Fermi-Defined Gaudi Components

- G4Generator – interface to Geant4
- Monte Carlo and digitization algorithms
- GlastDetSvc – service providing detector geometry description where details are stored in XML
- CalibSvc - calibrations
- GlastRandomSvc – handles initialization of random number seeds
- AcdRecon, TkrRecon, CalRecon - Reconstruction algorithms
- ntupleWriterSvc – ROOT ntuple
- ROOT I/O algorithms
- LDFConverter - Flight and test data input

Advantages and Disadvantages

ADVANTAGES

- Provides standardized, general interfaces
- Free, source code available
- Pick and Choose Components
- Python Interface
- Object Oriented
- Longevity

DISADVANTAGES

- Not well documented Most docs from 2000
- Steep learning curve for development
- Slow start up during loading
- Requires use of DataObject interface

Current Plans

- Migrating to Gaudi v21r7 from v18r4
  - Closer relationship to ROOT
    - Uses rootmaps on windows to dynamic load libs
  - Looking for VC++ 2008 and gcc 4 support
- Fermi is moving from CMT to SCons
  - We'll continue to use CMT to build Gaudi as one of our ext. libs

References

- Fermi Workbook
  - http://glast-ground.slac.stanford.edu/workbook/instrAnalysis_home.htm
- Gaudi Home Page
  - **http://proj-gaudi.web.cern.ch/proj-gaudi**
- BNL Gaudi Mailing List
  - https://lists.bnl.gov/mailman/listinfo/gaudi-talk