An Automated Release Manager for the Fermi Large Area Telescope Software Systems

Thomas E. Stephens (Wyle IS/NASA GSFC)
Navid Golpayegani (NASA GSFC)

Abstract

The Fermi Gamma-ray Space Telescope (Fermi) Large Area Telescope (LAT) collaboration maintains a large software system that covers all aspects of the instrument operation from simulations of the instrument response to event reconstruction and data analysis. Much of this software is supported and developed across a variety of operating systems and platforms (Windows, Linux and Mac OS X, both 32 and 64 bit). In order to ensure that the software works across the full range of supported systems, the LAT collaboration has developed an automated Release Manager system to checkout, compile and test any new code across all these systems regardless of which system it was developed on.

This poster describes the newest version of this Release Manager system developed in conjunction with the move by the collaboration to the use of SCons as our build tool of choice (described elsewhere at this conference). Built upon the Qt framework, the Release Manager leverages the batch submission system at the SLAC National Accelerator Laboratory (SLAC) to build and test any new code changes on all relevant platforms. Here we describe the design of the system as well as issues encountered in its implementation.

Software Systems

The LAT collaboration software is contained in two major and two minor software systems or release packages:

- Science Tools – The Science Tools contain all the collaboration software related to the scientific analysis of the Fermi LAT data. This is the package used by the individual scientists to do data analysis as well as tools used by various automated pipelines in the collaboration (flaring source detection, catalog generation, etc.)
- GlastRelease – Named before the mission name was changed to Fermi, the GlastRelease package contains all of the simulation and data reconstruction software for the mission. It contains a high fidelity spacecraft model and physics simulation used for generating large simulation data sets to study the instrument response. The GlastRelease package also contains all the software and algorithms used to reconstruct the data received from the spacecraft (or simulation) into useful scientific data. The software in the GlastRelease package is used by the data reconstruction pipeline to process the data as it arrives from the spacecraft and prepare it for the data archive and use by the Science Tools.
- Command, Health and Safety (CHS) – This package contains the software responsible for generating commands sent to the instrument to control operation as well as receiving and analyzing telemetry data downloaded from the instrument during operation.
- TMineRelease – This package contains a classification tree data mining package that is used as part of the data analysis and reconstruction done by the software in the GlastRelease package. It was split out into its own package to facilitate rapid development without encumbering the much larger GlastRelease package.

Supported Operating Systems

The LAT collaboration software is supported across a variety of operating systems and environments. These operating system include systems used by developers, end users within the LAT collaboration, and the various operating environments that the production software needs to run in for simulation and data processing.

Currently we support development and/or operation of the various software systems on the following operating systems:
- Redhat Enterprise Linux 4 – 32 and 64 bit systems
- Redhat Enterprise Linux 5 – 32 and 64 bit systems
- Windows – Visual Studio 2003 compilers (soon to be discontinued)
- Windows – Visual Studio 2008 compilers
- Mac OS X versions 10.4 (Tiger) and 10.6 (Snow Leopard)

Build Types

The Release manager supports three basic build types: Integration, Release Candidate, and Release.

Integration Builds

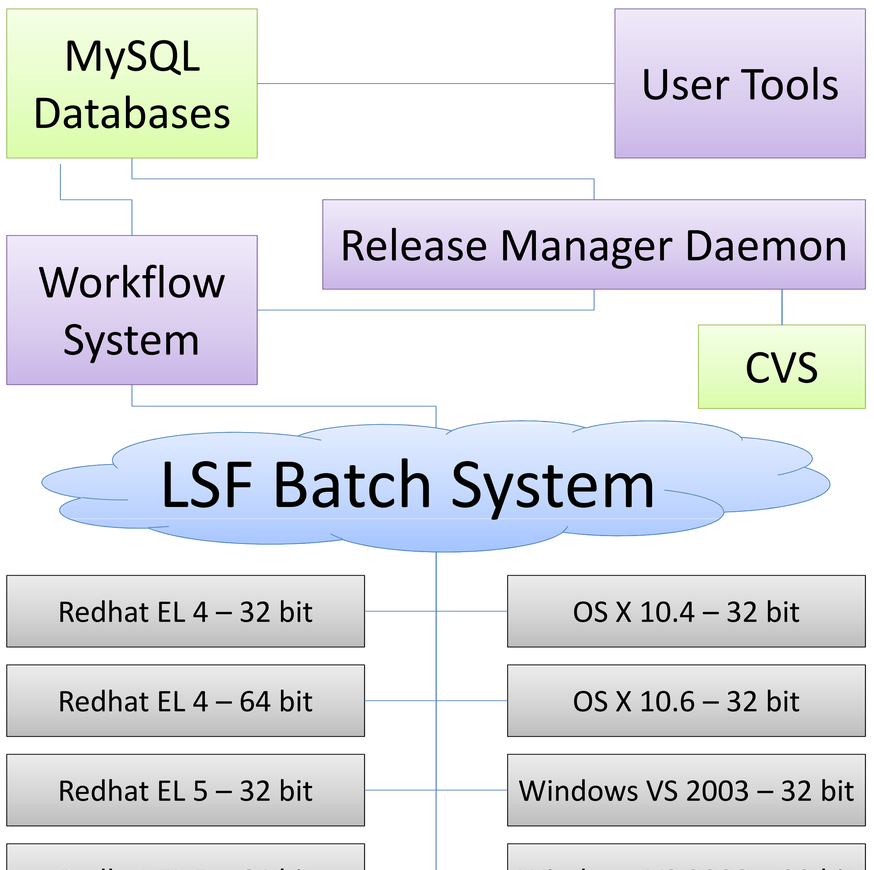
This type of build is automatically triggered whenever a software component receives a new tag in the CVS repository. The Release Manager Daemon regularly checks the repository looking for new tags on the various sub-packages that make up each release package. When one or more new tags are discovered, a new Integration build is triggered. Debug versions of the packages are built automatically but optimized version can be triggered manually if desired. The Integration builds are primarily designed to provide rapid feedback to developers on changes made to the code and to verify that changes made work on all supported operating systems.

Release Candidate Builds

These builds are triggered by a specific tag that is manually applied by the release package owner. They are triggered in preparation for a release build to verify that the selected tagged versions of the sub-packages build and work together properly. They contain the appropriate tags for the release in a combination that may or may not have existed in the Integration builds. Once the tag is applied to the appropriate sub-packages, debug versions of this build are created for each supported OS.

Release Builds

These are he builds intended for distribution to the collaboration and for use in the various automated systems run by the LAT team (data processing, catalog analysis, etc.). These builds are triggered by the existence of the appropriate release tag in CVS which is applied manually by the package owner. The existence of the appropriate tag causes the Release Manager to automatically build debug and optimized versions of the software for each supported operating system.



Schematic of the basic components of the LAT Release Manager System – User tools can request the triggering or deletion of a build by updating the appropriate tables in the database. The Release Manager Daemon monitors both the database and the CVS repository for changes. When a change occurs it triggers the Workflow System to actually execute the build. The workflow system leverages the LSF batch system to run multiple builds in parallel and records state and status in the database. The various operating systems/compilers that the Release Manager system software run on are shown in grey. Not show (for clarity) are the connections between the execution hosts in the batch system and the MySQL database and CVS repository used for fetching the code and logging of the various processes.

Issues and Lessons Learned

No large system is constructed and works without issues. Here we highlight some of the hurdles we had to overcome and lessons learned along the way.

- **LSF on multiple operating systems** - Since the batch queuing system was at the center of the build system, understanding its operation and peculiarities was essential to correct operation. There were subtle differences between the way the LSF system worked with the underlying operating system on the target machines, especially between the Unix-like OSes and Windows.
- **General Windows Support** – Supporting Windows has been both a boon and a thorn in our side. On the one hand, the Windows tools and compilers are high quality and having to support builds with both Visual Studio and GCC compilers has resulted in a strong, robust code base. On the other hand, the Windows environment is very different from the Unix-like OSes and special care has to be taken in many of the configuration issues to account for the differences.
- **Windows Network Storage** – In addition, the use of AFS network mounted storage on Windows resulted in very slow performance of the entire system. Since the builds are run through the batch system and different portions may run on different machines, all the code, libraries, etc need to be on shared disks. The performance on Windows is so bad (8-12 hours instead of ~1) that we are moving all of our Windows builds to a single multi-core machine with a large local disk.

Release Manager Components

The Release Manager consists of three main components: the batch submission system, the workflow system, and the release manager proper. Each of these three components are supported by a series of database tables to drive the processes and store state and metadata about the pending, running and completed builds.

Batch Submission System

The Release Manager utilizes the LSF batch submission system at SLAC National Accelerator Laboratory. Through this system we have access to hardware (either purchased by the Fermi mission or as a shared resource) running all of our supported operating systems that we can use for build and testing of the various software packages.

This is the lowest level component of the system and manages the individual processes of the builds and provides the status information to the other portions of the system.

Workflow System

The Workflow system is a rule based script execution system. Each script or program is considered a stage in the workflow. The workflow moves from one stage to the next by evaluating rules set forth for each stage. The rules are stored in the database as a series of conditions and steps to execute if the conditions are met. Each stage of the workflow consists of a script or program that is passed to the Batch Submission System for execution.

Release Manager System

The Release Manager System consists to two main parts. At the very top, is the Release Manager Daemon that runs and monitors the CVS repository for new tags and triggers the appropriate workflows to actually execute the builds.

At the lower level, this system is composed of the actual programs executed by the Workflow system to checkout code, build the software, test the software and package it for distribution and download as well as command line tools for deleting and triggering builds.

Building a Release

The process of building, testing and packaging a release consists of several steps that are managed by the Workflow System and executed by software that is part of the Release Manager System. The basic steps are:

- **Checkout** – Selects all the sub-packages that are to be part of the build via the appropriate tag in the CVS repository and places them in a central build location
- **Compile** – SCons is invoked to build the package. (See poster P099 for details on our use of SCons.) Output is recorded and stored in the database.
- **Test** – Nearly all of the sub-packages have unit and validation tests that are run to verify that all the code is working properly. Each test is run independently and the output is stored in the database.
- **Package** – Running in parallel with the testing, each build has a source, user and developer distribution package created to allow users and developers do download and work with the specific version of the code built and tested.
- **Cleanup** – Once all other processing is done, a script is run to clean up the build process and store any final metadata in the database.

Using Qt

Qt is used as a framework to build the various programs that comprise the build system in order to provide parallel and asynchronous execution of the various parts of the build system. The main highlights are discussed here.

The Release Manager Daemon leverages the QTimer class to set up an asynchronous polling system to check each of the 12 possible builds (4 Packages and 3 build types) on a configurable polling interval to look for new builds that need to be started.

Once the need for a new build has been determined, the software leverages the QProcess class to launch each of the up to 16 build variations (8 OSes each with a Debug and/or Optimized build) in its own thread for processing.

In addition to allowing the entire system to be multi-threaded, the QProcess class allows the system to be robust against hung processes and other unexpected failures as we utilize the ability to limit the time the process remains active. Processes and build stages that exceed the configured (generous) time limit are cancelled and errors are reported.

Finally, all of the classes and tools associated with the build system make heavy use of the QSqlQuery class to provide easy access to the MySQL databases that hold all of the configuration parameters, build status information and logging.