GLAST Data Processing Pipeline

Status Report
9 September, 2004
Daniel Flath
Required Functionality

- automatically process Level 0 data through reconstruction (Level 1)
- provide near real-time feedback to IOC
- facilitate the verification and generation of new calibration constants
- re-process existing data
- produce bulk Monte Carlo simulations
- backup all data that passes through
Major Components

- relational database (management system)
- database access layer
- user interface
- scheduler
- execution layer
Database Overview

• Currently Oracle, using Stored Procedures for all query access
• Perl Module provided as entrypoint to stored procedures (API)
• Primarily two sets of tables:
  – Pipeline Management tables allow users to configure processing flow of a pipeline
  – Processing History tables record status of processing and data
Task Configuration Tables

• Task Table
  – Linked to by TaskProcess and Dataset Tables

• A task is comprised of 1 record in the task table and 0 or more records in the TaskProcess and Dataset tables

• The latter are related by a linked list of Read and Write flags that determine processing flow
TaskProcess and Dataset Tables

- A TaskProcess record contains all information necessary to run a job (script version, location, etc.)
- It’s links to Dataset records allow the pipeline to determine where to find input and where to write output datasets (files) for the job
- These filenames are provided to the users’ wrapper scripts at job execution
Processing History Tables

• Records in the Run table represent instances of processing for a Task
• TPInstance, and DSInstance records are in the same way analogous to TaskProcess and Dataset records
TPInstance and DSInstance Tables

- TPInstance tracks the state of processing for a single job (Execution status, CPU time, memory used, etc.)
- DSInstance records act as file descriptors for datasets used in the processing chain that is a run (File size, location on disk, file format, data type, archive location, etc.)
User Interface

• Suite of web pages that provide:
  – Pipeline Configuration
  – Processing Status Display
  – Processing Management
  – Reprocessing configuration

• Backed by utilities that allow creation of runs, rolling back of failed processes, deletion of runs and data, etc.

• Many of the utilities already exists as command line utils. Need to be callable from the UI.
Scheduler

- Run periodically in cron
- Queries for runs that have successfully finished running their last scheduled taskprocess
- Finds the next taskprocess to run and schedules it to the execution layer
- Marks runs failed for which the last scheduled taskprocess failed for any reason
Execution Layer

- Provides access to computing resources
- At SLAC, this is LSF/Batch
- Takes requests to process jobs from Scheduler
- Monitors status of scheduled jobs
- Reports changes in job status (start of execution, completion, failure, etc) back to Scheduler
Execution Layer Implementation

• Currently a simple interface to LSF/Batch written directly into the scheduling code
  – Makes blocking (synchronous) submissions to batch and waits on completion

• Soon to be replaced by Navid’s new LSF/Batch interface
  – Provides a full implementation of the Exec Layer Spec (last slide) including a callback mechanism for asynchronously reporting processing status updates
Status

• First version delivered to I&T several weeks ago
• Second version with I&T requested features delivered Friday (Sept 3)
  – Supports variable substitution in file paths
  – Uses work-around for chaining different Tasks on a single dataset
• Need to add a few more features to current version
• Next version already in progress
  – will allow Heather to begin running simulation in large doses
Yet to do in Current Version

• Implement archiving of files at completion of a run (Navid working on archiver now)
• Use a database connection pooling proxy to avoid running over the limit of simultaneous connections (Alex installing this software now)
• Email notification of Task owner on failed runs
Next Version Major Improvements

- Ability to share datasets between tasks
- Parallel processing of independent TaskProcesses within a Task
- Management code to become more modular
- Implement Navid’s new Batch interface in the execution layer
Where to Find It…

- Current configuration utility written by Navid is at:  https://www.slac.stanford.edu/www-glast-dev/cgi/Login
- Matt is working on a fancy new version that allows editing of existing records
- Code is in cvs:/DPF/PDB
User Responsibilities

• Configure their pipelines using the front-end provided.
• Monitor & manage processing using provided tools (assistance will be given on issues for which no tool yet exists).
• Ensure executables provided are visible to the pipeline account (SLAC/glast).
• Wrap each executable using the perl template that provides access to the datasets for the run being processed.
  (cvs:DPF/PDB/taskProcessTemplate.pl)