Implementation and performance of the Fermi LAT level 1 pipeline

Warren Focke (SLAC), Maria-Elena Monzani (SLAC), Anders Borgland (SLAC), Larry Wai (Yahoo) on behalf of the Fermi Large Area Telescope Collaboration

Fermi Level 1 processing performs digitization, reconstruction, and monitoring of Fermi LAT data. It has used 125 CPU*years in the first year of data taking, and has processed $1.4*10^{10}$ events, $2*10^{8}$ photons.

Abstract

Fermi Level 1 processing performs digitization, reconstruction, and monitoring of Fermi LAT data. It must handle incomplete and out of order data and recover gracefully from hardware or software failures. It runs on the SLAC pipeline and the general SLAC batch farm. It uses up to 800 processor cores. It has used 125 CPU*years in the first year of data taking. FITS data products are typically delivered to FSSC 2-3 hours after receiving the data (8 hours after the data are acquired). It has processed 1.4e10 events, 2e8 photons.

Challenges:

- Data may be incomplete
  - runs are in pieces
- Pieces may be out of order
- Can't process 2 pieces of the same run at once
  - file-based locking mechanism
- Can't overload the storage system
  - so we have a throttle to limit the number of runs that can be processed at once– also a file-based lock
  - only applies to the heavy-I/O part of the processing
- Low latency requirement
  - deliver data even if the data is incomplete or there are processing failures – fix it later then deliver more-complete data. This means we frequently do redundant run-level processing.

Implementation Details:

- Implemented in Python
  - 6600 lines (plus 2000 more in GPLtools)
- 3300 line XML pipeline task definition
  - expanded from a template at install time

Parallel Processing:

Every 2.4 hours (average), we receive 2.4 hours of data. Processing 1 hour of data takes 125 hours of CPU, plus more time waiting for I/O. So we have to process it in parallel. This happens at several levels:

- Data arrives in deliveries
- Deliveries contain parts of multiple runs (usually 1 complete and 2 halves)
- Runs split into chunks (~100k events) to avoid gaps and for parallelism
  - digitization and most monitoring happens here
- Chunks split into crumbs (~3k events) for parallelism
  - currently, we only do recon at crumb level
- Chunks and crumbs have varying sizes – that way jobs don't all finish at the same time, and I/O is spread out in time.
- Chunk and crumb files are temporary
  - crumb files remain on disk ~ 1hour
  - chunk files might remain on disk a day or more until all data for the run is received
- Crumb files must be merged into chunks, and chunk files must be merged into runs. This involves a lot of I/O.

Infrastructure:

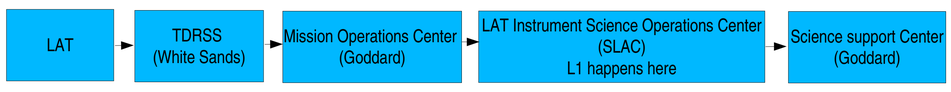
- LSF batch system
  - high-priority queue on general-use batch machines
- The pipeline (see Tony Johnson's poster, "Fermi data processing pipeline, collaboration data servers and web based data monitoring tools.")
- 800 cores CPU
  - We usually don't use them all, other SLAC users may use them when we don't
- 7 AFS servers for chunk-level temporary files
- Fermi xroot cluster (22 servers) for permanent and crumb-level files
- Web-based control
- Failed jobs are automatically retried once to deal with transient errors

Between Level 0 ingest and Level 1 processing, the data passes through the HalfPipe.

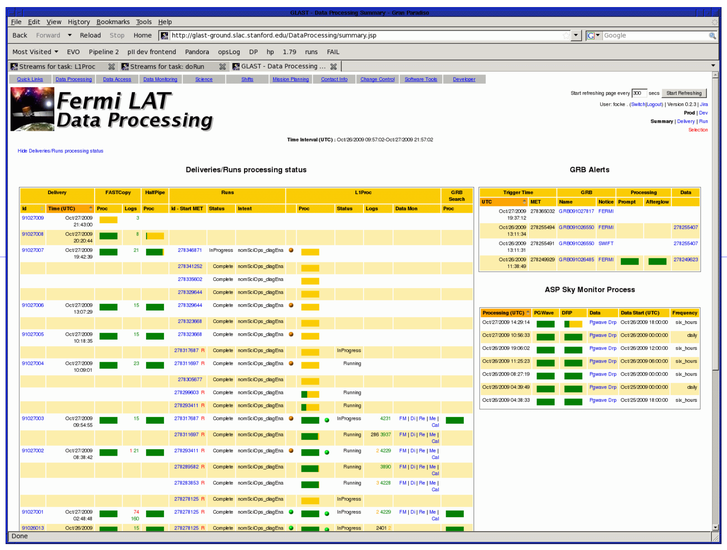
- Ignores duplicate data
- Arranges data into chunks that have no gaps.
- Chunks may be split further for more parallel processing.

Future Directions:

- Prioritize runs
  - They get processed in random order now
  - process oldest first
  - manually or automatically promote runs with a GRB
- Use xroot for all scratch files
- Relax run lock to apply only to run-level processes
- Skip run-level processing if more data is waiting?



Data flow



Web-based monitor. Most elements are links to pipeline or data quality pages.

Elapsed time to deliver processed data. "NASA" = time from acquisition until it arrives at SLAC. "SLAC" = arrival at SLAC until delivery to FSSC.

Typical jobs/time. pink: recon, green: chunk-level jobs, orange: run-level jobs. Other active tasks are listed, but not visible.