ASP Data Processing Sequence

Overview

- Conditions for Task Launching
    1. GCN Notices
    2. Level 1 (FT1/2) Data availability
    3. GRB task launcher process and DataCatalog querying
- Individual Task Data Processing
    - GRB_blind_search
    - GRB_refinement
    - GRB_afterglow
    - DRP_monitoring
    - flareDetection
    - PGwave

Conditions for Task Launching

1. GCN Notices arrive asynchronously via the GCN socket connection.
    - On arrival, each Notice is entered into the GCNNOTICES db table.
    - If the Notice refers to a new GRB, a GRB_ID is assigned, and a new entry in the GRB db table is created.
    - For each new entry in the GRB db table, two flags are set to false (0): L1_DATA_AVAILABLE and ANALYSIS_VERSION. L1_DATA_AVAILABLE is set to true if L1 data are available for the GRB_refinement task and the task has been submitted. ANAYLSIS_VERSION is set to true if the GRB_refinement task has run and if the L1 data are available for the GRB_afterglow task. (The names and conditions for altering fields will need to be reviewed.)
2. When new L1 (FT1/2) data are available, ASP tasks are launched.
    - GRB_blind_search, flareDetection, and PGwave are designed to process each downlink and are launched unconditionally.
    - GRB_refinement, GRB_afterglow, and DRP_monitoring require data from specific data time intervals to be available in order to run. Presently (10/15/07), the first two are handled by the GRB task launcher process. DRP_monitoring does not have a task launcher process yet.
3. GRB task launcher process and DataCatalog querying
    - In the BlindSearch process, a script (grb_followup.py) is run that queries the GRB db table for entries satisfying two sets of conditions:
        - L1_DATA_AVAILABLE=0: These are GRB candidates that require the GRB_refinement task to be run.
        - L1_DATA_AVAILABLE=1 and ANALYSIS_VERSION=0: These are GRB candidates that have had GRB_refinement task run, but still require GRB_afterglow to be run.
    - Based on the data in those queries, grb_followup.py computes the time intervals to be considered and queries the DataCatalog for the required FT1/2 files.
    - If those files are available, the corresponding tasks are launched.
    - The GRB table entries are updated by the launched tasks.

Individual Task Data Processing

- GRB_blind_search
    1. The DataCatalog is queried for the FT1 file based on DownlinkId.
    2. The GRB_ASP_CONFIG db table is queried for blind search configuration: log-likelihood thresholds, event number partition size, effective deadtime between burst candidates, etc.
    3. The FT1 file is read in and analyzed.
    4. For each GRB candidate:
        - A working directory is created on NFS.
        - A LatGcnNotice is generated. A text version is written to cwd. A GCN packet version is added to the GCNNOTICES db table.
        - email notification is sent out
        - If the burst corresponds a burst already in the GCNNOTICES db table (via an algorithm TBD), an entry is made and it is marked as an "UPDATE" (ISUPDATE=1), otherwise it is marked as "NEW" (ISUPDATE=0) and a GRB db table entry is created with the candidate burst parameters (INITIAL_RA, INITIAL_DEC, INITIAL_ERR_RADIUS, MET(=GRB_ID)).
    5. grb_followup.py is executed to launch followup tasks for all pending GRB analyses
- GRB_refinement
    1. The DataCatalog is queried for the FT1/2 files based on a time window centered on the candidate burst time derived by GRB_blind_search.
    2. The FT1/2 files from the DataCatalog locations are merged into single FT1 and FT2 files, and these files are written to the NFS working directory created for this burst candidate by GRB_blind_search.
    3. The GRB_ASP_CONFIG table is queried for position refinement configuration parameters.
    4. The extractLatData, refinePosition, LatGrbSpectrum, and tsMap task subprocesses are run, writing their output to the burst-specific working directory on NFS.
    5. Each process writes output to the GRB db table.
    6. The tsMap process creates plots for the GRBMonitoring page.
    7. The registerPlots scriptlet registers the plots with the DataCatalog so that the GRBMonitoring page can find them.
- GRB_afterglow
    1. The DataCatalog is queried for the FT1/2 files based on a time window, nominally 5 hours (TBR), after the end of the prompt burst phase as determined by GRB_refinement.
    2. The FT1/2 files from the DataCatalog locations are merged into single FT1 and FT2 files, and these files are written to the NFS working directory created for this burst candidate by GRB_blind_search.
    3. The GRB_ASP_CONFIG table is queried for afterglow analysis configuration parameters.
    4. The afterglowData, afterglowLivetimeCube, afteglowDiffResps, afterglowExpMap, and afterglowAnalysis task subprocesses are run, writing their output to the burst-specific working directory on NFS.

5. Each process writes output to the GRB db table. (not implemented yet)

- DRP_monitoring
  1. The DataCatalog is queried for FT1/2 files based on a daily or weekly time window. The exact start times of these windows are TBD.
  2. A local working directory on NFS is created for each day or week interval.
  3. The getIntervalData subprocess merges the FT1/2 files from the DataCatalog locations into single FT1 and FT2 files and writes them to the time interval specfic working directory on NFS.
  4. The getIntervalData subprocess reads a parameter definition file from DRPMONTIORINGROOT with the analysis configuration (i.e., the IRFs to use, etc), and creates a local copy. (This parameter file should be replaced by an appropriate configuration table in the Oracle db.)
  5. getIntervalData, livetimecube, and diffuseResponses subprocesses are run, writing their output to the working directory on NFS.
  6. The launchRoiAnalysis pipeline scriptlet loops over the ROI_IDs, and for each ROI_ID, the roiAnalysis subtask is launched:
     - The getRoiData subprocess
       - queries the SOURCEMONITORINGROI db table for the ROI parameters (RA, Dec, ROI radius, source region radius),
       - creates a subdirectory on NFS fro that ROI,
       - runs gtselect to extract events for that ROI,
       - queries the SOURCEMONITORINGPOINTSOURCE and SOURCEMONITORINGDIFFUSESOURCE db tables for sources to include in the xml model definition
       - writes the parameter file for gtexpmap.
     - exposureMap, exposureSubMap, drpExpMap, and combineExpMaps subprocess compute the unbinned exposure maps, which are written to the local NFS time interval subdirectory. (This infrastructure allows the exposure maps to be partitioned into submaps and those submaps dispatched to the batch queues to be run in parallel.)
     - sourceAnalysis runs unbinned likelihood analysis, computing the source model parameters over the 30 MeV to 200 GeV range. Results are written to the xml model output file in the local interval and region-specific working directory on NFS.
     - The energyBandAnalysis subtask runs the fitEnergyBand subprocess for each energy band in the DRP_monitoring XML task definition (It would be good to have these bands configurable in at db table.)
     - The fitEnergyBand subprocess performs an unbinned likelihood analysis for the specified energy band and registers the flux values for each DRP monitored source for the time interval with the SOURCEMONITORINGDATA db table.
- flareDetection
  1. The DataCatalog is queried for the FT1 and FT2 file based on DownlinkId.
  2. A working directory on NFS is created for each downlink.
  3. The createHealpixMaps subprocess bins FT1 data into 48 (rather large) sky pixels and computes the exposure at the center of each pixel.
  4. The computePixelFluxes subprocess computes the fluxes and registers the fluxes for the pixels with non-zero exposure with the SOURCEMONITORINGDATA db table, with each entry identified by its HEALPix coordinate for that Downlink time.
- PGwave
  1. The DataCatalog is queried for the FT1 and FT2 file based on DownlinkId.
  2. A working directory on NFS is created for each downlink.
  3. The getPgwInputData subprocess merges the FT1/2 files from the DataCatalog into single FT1 and FT2 files and writes them to the local working directory on NFS for that downlink.
  4. The runpgw sub process:
     - Creates a counts map using gtbin
     - Runs the pgwave application, which writes FITS and text files with the source candidates to the local NFS directory.
     - Runs gtsrcid to make preliminary source identifications and writes these results to the local NFS directory.