Pilot Updates for Event Server

Paul Nilsson

UNIVERSITY OF TEXAS ARLINGTON

runJob vs runEvent

- runJob module: forked as subprocess and monitored by the pilot/Monitor module for "normal" jobs
  - runJob determines (in sequential order) payload setup, stages in any input files, executes the payload and stages out the output files
  - The pilot/Monitor monitors the progress of the payload; verifies it is not hanging, that it does not consume too much disk space, etc
- runEvent module: forked and monitored by the pilot/Monitor for "event" jobs
  - runEvent will reuse several functions developed for runJob
  - runEvent determines payload setup (DBRelease stage-in?), mainly runs in a loop, asking the event server for events [to be done]
  - If it receives events, it will process them [to be done]
  - Independent to the event processing loop, a separate thread is looking for new output files in a special output directory; if a new output file is found it will be staged out and then removed [in progress]
  - When there are no more events to be processed, the main event loop waits for the asynchronous stage-out loop to finish

Status and Planning

- runEvent module created and added to [major] dev pilot version 59a
  - Integration and development delayed because of glExec integration and code merge [almost done]
  - runEvent based on runJob, plus previously developed test script
  - Only activated in test mode (for now)
- runEvent module could be run independently of the pilot, practical for interactive testing off the grid
  - Once the a PanDA test event job has been defined, it can be put in a file which is read by runEvent (this is how runJob works)
  - SYNTAX: python runEvent.py <arguments to be determined..>
- Eventually merge stand-alone event processing script with runEvent (as soon as it makes sense)

Considerations (1)

- Event job concept
  - Job download as usual, job definition contains an identifier specifying the job type to be "normal" or "event" (used for subprocess selection; runJob or runEvent)
  - ***Definition:*** "Event bunch" = all the N "event batches" the pilot will download
  - One PanDA job id per "event bunch" for easy job monitoring on the PanDA Monitor, like in a "normal" PanDA job, corresponding to all the "event batches" the pilot will process
- Pilot should abort event processing when there is an SE problem
  - I.e. stop downloading further events after current event processing is done (or even abort the current processing)
  - Error tolerance in stage-out loop (give up after two stage-out events)
- Pilot should report to server when it has finished an "event batch" (better: after staging out corresponding file)
  - Server will know which events have not been processed if the pilot does not report back

Considerations (2)

- How to handle/upload job metadata (payload + output)
- More questions will follow..