PanDA Pilot Development

Dr Paul Nilsson

UNIVERSITY OF TEXAS ARLINGTON

PanDA Pilot Code Structure

- Main PanDA Pilot code: pilot.py, launched by pilot wrapper(s)
- Experiment specific code are placed in special plug-in classes for "easy" implementation for a new PanDA system user
  - ATLAS code are located in ATLASExperiment (e.g. payload setup) and ATLASSiteInformation (e.g. handling of queuedata, i.e. site specific information), inheriting from Experiment and SiteInformation respectively
- Stage-in/out handled by "Site Mover"-classes (e.g. all code related to xrdcp is located in xrdcpSiteMover, inheriting from SiteMover)
- Many other support classes and modules (e.g. for PanDA server communication, utilities, etc)

PanDA Pilot Refactorization

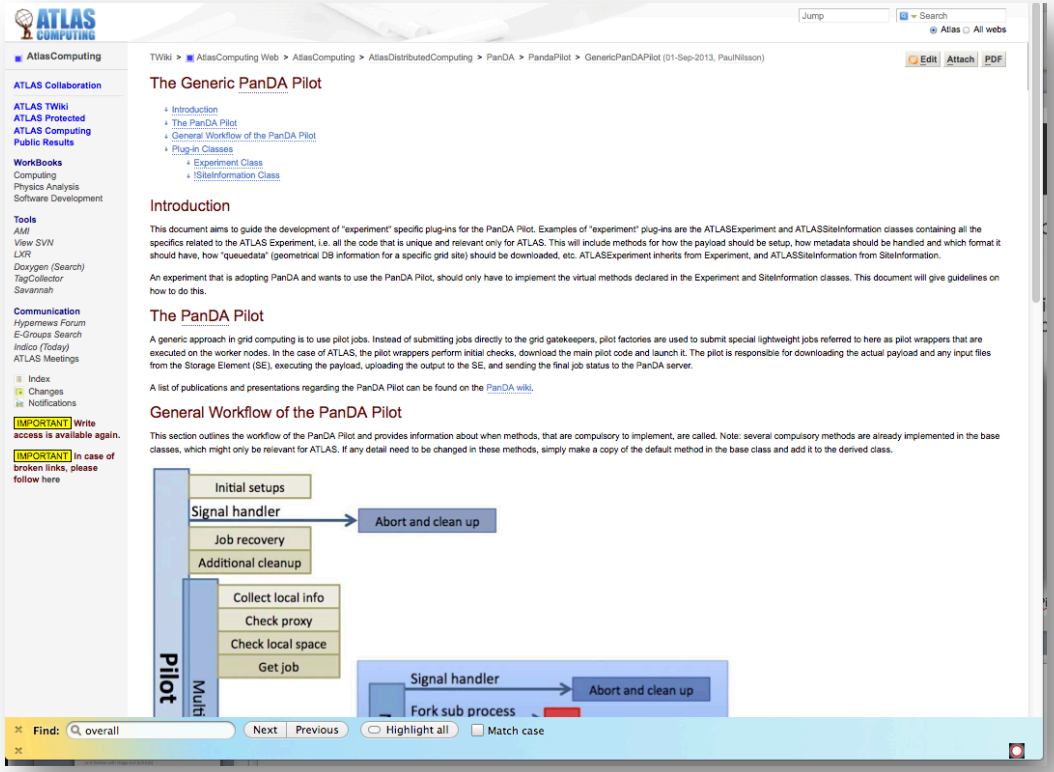
- Ongoing effort to improve overall code quality and efficiency
- Refactorization necessary for several reasons
  - Adaptation of PanDA Pilot for non-ATLAS use
  - Complex feature requests
  - Reduce/remove legacy code (in general: refactoring means reorganization, not just rewriting)
  - Reduce complexity to make it easier for code contributors
- Trivial vs non-trivial pilot refactorizations
  - Site Movers contain repeated code fragments (trivial)
  - Pilot code needs to support users other than ATLAS (non-trivial)
- "Only" 47k LOC, so not a hopeless effort..

Refactorization Status

- General pilot refactoring effort is progressing well
- New pilot versions have minor or major changes related to refactoring on basically every check-in
- PanDA Pilot is now effectively a "generic" pilot that is "not" ATLAS specific
  - NB 1: this means that the major changes are done, there are plenty of minor changes that remain (code relevant only for ATLAS is still present but is of little or no concern to other users)
  - NB 2: Some ATLAS specific code is deeply integrated with specific modules, namely the site movers and the DQ2 tracing code. Cannot easily be refactored

Documentation

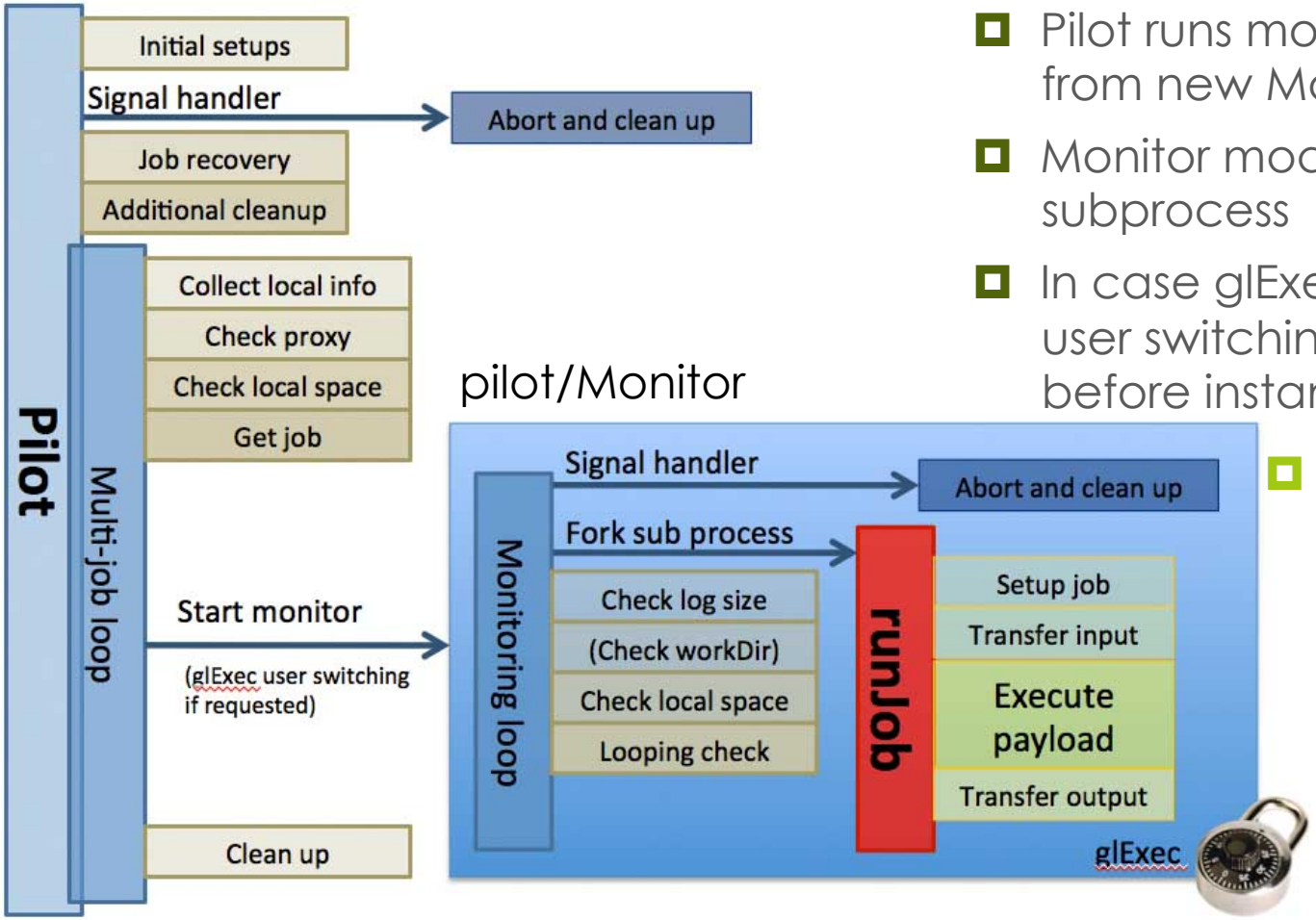
- Generic PanDA Pilot documentation in development
  - Will serve as a manual on how to use PanDA Pilot for a joining experiment and what methods should be implemented
  - Will contain a walk-through on pilot workflow, indicating which methods need to be implemented, described in the workflow
- Current pilot documentation to be revamped as well



https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/GenericPanDAPilot

glExec Integration and Code Merge

- Work done with help from Ramon Medrano Llamas and Fernando Barreiro



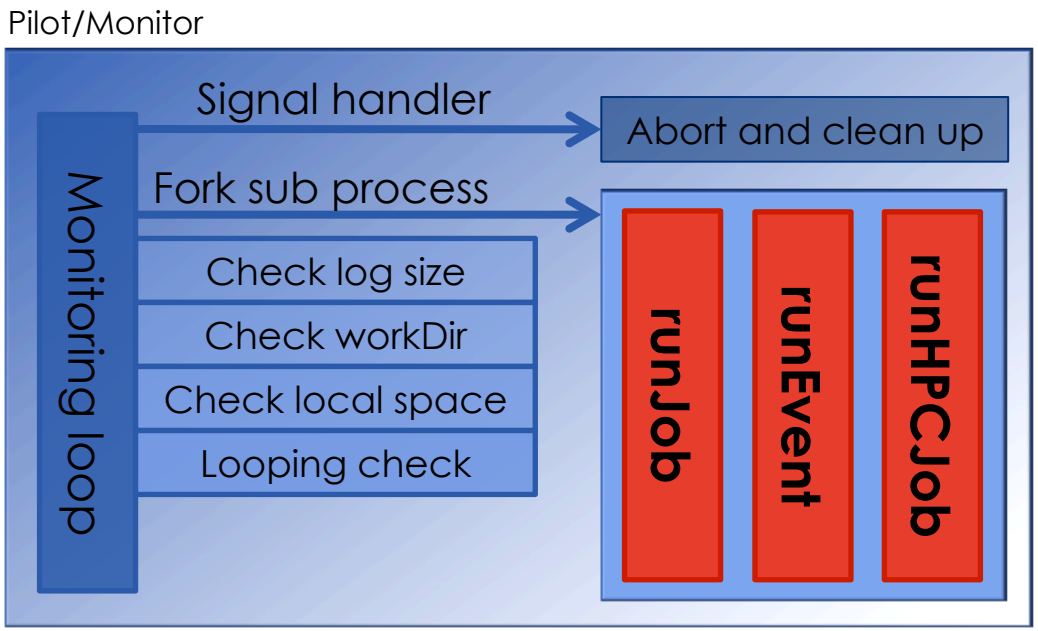
- Multi-job intact, monitoring loop refactored
  - Pilot runs monitoring loop from new Monitor module
  - Monitor module forks runJob subprocess
  - In case glExec is needed, user switching done by pilot before instantiating Monitor
- ***Status***: merge done, sorting out bugs, testing every single algorithm. *glExec part not tested on grid yet (in progress by Ramon)*

Subprocess Modules

- Pilot/Monitor forks and monitors a ***subprocess*** responsible for the payload
- The subprocess can be any module that needs full attention and supervision of the Pilot/Monitor
  - **runJob** in use since 2005
  - Introducing **runEvent** and **runHPCJob**

To read events from an Event Server

Possible subprocess module for HPC use



Subprocesses: runJob vs runEvent

- runJob module: forked as subprocess and monitored by the pilot/Monitor 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 follows 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 [finished] output files; if one is found it will be staged out and then removed (work dir tarred up as usual?) [in progress]
  - When there are no more events to be processed, the main event loop waits for the asynchronous stage-out loop to finish

Event Server Support: Status and Planning

- runEvent module created and added to dev pilot version 59a
  - Integration and development delayed because of glExec integration and code merge [almost done]
  - runEvent based on runJob, plus previously developed threaded test script
  - Only activated in test mode (for now) – will not be activated in 59a
- 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 (Vakho Tsulaia et al) with runEvent, as soon as it makes sense to do so
- More details in Torre's talk

HPC Support

- PanDA Pilot cannot be executed on HPC system out of the box
  - No middleware
  - No outbound connections
  - No TCP connections
  - Process forking not allowed
- Two approaches possible (both with non-trivial pilot changes)
  - Pilot can be executed on the front end node(s), using new suggested subprocess module runHPCJob acting as a job factory, submitting jobs into the HPC [Danila Oleynik]
  - Bypass HPC limitations; get rid of middleware use (as far as possible), pilot contacting a Control Tower running on the front-end (instead of PanDA server), use threading in pilot/ Monitor instead of process forking (also solved the TCP limitation) [Doug Benjamin]

Log File Stage-out

- Log file is currently staged out to the same SE as the output files
- It has been suggested (Simone Campana et al) to stage-out log files to special secondary SE area (either on the same site or centrally at CERN) and eventually perhaps only to this special area
- DB and queuedata changes required; activation, base path (possibly reuse sepath), SE info, ..
- Stage-out using xrdcp or perhaps WebDAV (new site mover required); initially xrdcp will be used
- Suggested storage path format: example using job id 1838566890 gives path xroot://eos.cern.ch/atlas/logs/18/38/56/68/90/mylog.tgz (already implemented)
- Some refactoring is needed in the pilot
- Should this be the default for any PanDA system user?

Error Code Handling

- Currently TRF/payload error codes only properly handled for ATLAS jobs
- TRF/payload errors picked up by the pilot and forwarded to server
  - Not experiment specific
  - Experiment TRF/payload need only create special job report json file
- Monitor side needs update to handle general TRF/ payload error codes
  - Solutions (incl. plug-in developed by Graeme Stewart) have been discussed already but stalled due to other priorities
  - Let's resume these discussions as soon as possible

In Development (a selection)

- Finish glExec code merge (pilot version 59a PICARD), largely done, but lots of testing remains (= weeks)
- Test stage-out to alternative SE (T-2 to T-1); coding done, test delayed due to several reasons (external reasons now resolved and the pilot has been updated)
- New subprocess modules in development (runEvent) or in planning stage (runHPCJob) (pilot/Monitor changes done)
- Cleanup of schedconfig.copysetup vs schedconfig.envsetup
  - Both can contain environment setup scripts
  - Envsetup should in principle be sourced before pilot is run, "impossible" if queuedata is only downloaded by pilot. Use special trick by Jose Caballero to update env variables in pilot
  - Will eventually require schedconfig changes (copy setup part of copysetup to envsetup, if set)