Pilot Factory - a Condor-based System for Scalable Pilot Job Generation in the Panda WMS Framework

Po-Hsiang Chiu, Maxim Potekhin
Brookhaven National Laboratory
Upton, NY11973,USA

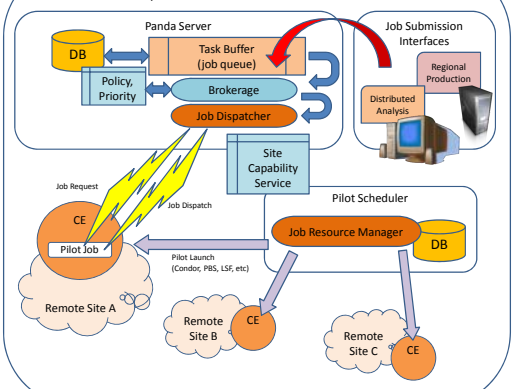
Panda's Pilot Framework for Workload Management

Workload jobs are assigned to successfully activated and validated Pilot Jobs (lightweight processes which probe the environment and act as a 'smart wrapper' for payload jobs), based on Panda-managed brokerage criteria. This 'late binding' of workload jobs to processing slots prevents latencies and failure modes in slot acquisition from impacting the jobs, and maximizes the flexibility of job allocation to resources based on the dynamic status of processing facilities and job priorities. The pilot also encapsulates the complex heterogeneous environments and interfaces of the grids and facilities on which Panda operates.

Job Submission

Jobs are submitted via a client interface (written in Python), whereby the jobs sets, associated datasets, input/output files etc can be defined. This client interface was used to implement Panda front ends for both production and distributed analysis. Jobs received by the Panda server are placed in the job queue, and the brokerage module operates on this queue to prioritize and assign work based on job type, available resources, data location and other criteria.

Simplified View of core Panda Architecture



Submission of Pilot Jobs

Panda makes extensive use of Condor (particularly Condor-G) as a Pilot job submission infrastructure of proven capability and reliability. Other submission methods (such as PBS) are also supported.

Other principal design features

- Through a system-wide job database, a comprehensive and coherent view of the system and job execution is afforded to the users.
- Integrated data management is based on the concept of 'datasets' (collections of files), and movement of datasets for processing and archiving is built into the Panda workflow
- Asynchronous and automated pre-staging of input data minimizes data transport latencies
- Integration of site resources is made easy, with minimum site requirements consisting of being able to accept pilot jobs and allow outbound HTTP traffic from the worker nodes
- Panda system security is based on standard grid tools and approaches (such as X.509 certificate proxy-based authentication and authorization)

Motivations for the Pilot Factory

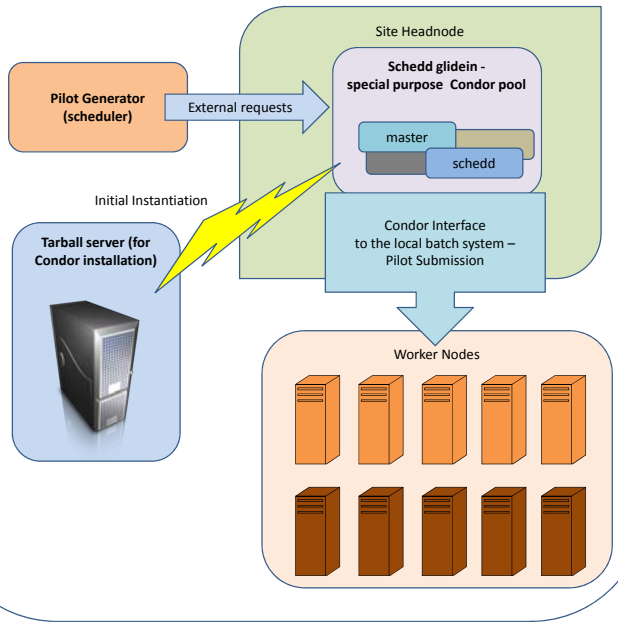
Pilot jobs submission via Condor-G is not different from submission of any other kind of jobs. With the increase in demand of production and distributed analyses jobs, more pilot jobs will need to be submitted, which leads to heavier traffic through the GRAM gatekeeper. That presents a potential scalability problem for the target site's headnode. One way of circumventing this is to implement local pilot submission on the headnode, thus significantly reducing the load on the gatekeeper.

Design considerations

To work efficiently, personnel operating the Pilot Scheduler must have a way to monitor the submitted pilots and their output. The new submission approach that satisfies this requirement was inspired from the idea of Condor-C, an alternative job submission mechanism that allows for jobs to be "forwarded" between a Condor *schedd* (i.e. a job request agent) to another. In order to use Condor-C, we would still need a communication gateway on the headnode itself in order to relay jobs directly to the native Condor system. This gateway also needs to be flexible enough to interface with the other batch systems for sites that do not employ Condor for job management. *Schedd-based glidein* is therefore conceived to support these features. Using *schedd-based glidein,* the Pilot Factory further removes the need of installing a pilot submission system on the headnode, leaving a thin layer - glidein schedd to redirect pilots to the native batch system. Once a glidein schedd is installed and running, it can be utilized exactly the same way as local *schedds* and therefore, pilots submitted in such a way are almost equal to those submitted within local Condor pool, resembling Condor's vanilla universe jobs.

Pilot Factory is effectively a new pilot-generating system using dynamic job queue as a gateway to dispatch pilots from within remote site's perimeter. The factory first deploys glidein schedds to the head nodes of the participating sites, followed by Pilot Generator submitting pilots directly to these glideins, from which these pilots are then forwarded the native batch system. Pilot Factory consists of three major components: (1) A glidein launcher, responsible for the dynamic deployment of glideins to eligible sites, (2) A glidein monitor that detects occasional failures due to site's temporary downtimes, upon which it then invokes the glidein launcher to perform another glidein deployment, and (3) A pilot generator that disseminates pilots through the schedd glideins running on remote resources.

Pilot Factory Components



Pilot Factory Components and Operation

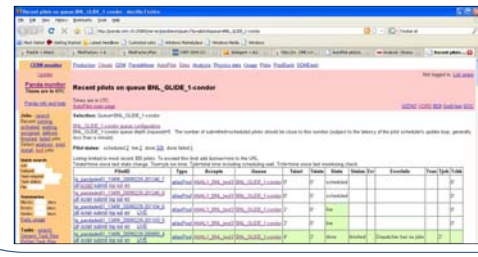
The current implementation of Pilot Factory consists of three major components:

1. Glidein launcher that dynamically installs and configures a set of schedd-related Condor daemons on a Globus-controlled headnode.
2. Glidein monitor . Both it and the Glidein launcher are coupled to the schedd glidein script, which is implemented based on the existing Condor condor_glidein script and has a very similar usage mode and functionality, but adds a few more options to aid in remote installation.
3. a pilot submitter that performs the following tasks:
   a) Configure a proper Condor submit file for pilot jobs
   b) Submit pilot jobs with a regulated control of submission rate
   c) Obtain the outputs and/or other byproducts of pilot jobs upon their completion.

More information and examples can be found here:
http://www.usatlas.bnl.gov/twiki/bin/view/AtlasSoftware/PilotFactory.html
http://www.usatlas.bnl.gov/twiki/bin/view/AtlasSoftware/ScheddGlidein

Pilot Factory deployment status

Over the period of the past few months, an instance of the Pilot Factory was run in test mode in conjunction with the BNL instance of the Panda server. Status of the pilots generated by the Pilot Factory can be checked using the Panda Monitor:



Examples of commands used with the Pilot Factory

Starting a number of glideins on a few sites:
- condor_glidein **-count 1** -arch 6.8.1-i686-pc-Linux-2.4 -setup_jobmanager=jobmanager-fork **gridgk01.racf.bnl.gov**/jobmanager-**fork -type schedd** –forcesetup
- condor_glidein **-count 1** -arch 6.8.1-i686-pc-Linux-2.4 -setup_jobmanager=jobmanager-fork **gridgk02.racf.bnl.gov**/jobmanager-**fork -type schedd** –forcesetup
- condor_glidein **-count 3** -arch 6.8.1-i686-pc-Linux-2.4 -setup_jobmanager=jobmanager-fork **nostos.cs.wisc.edu**/jobmanager-**fork -type schedd** –forcesetup

Check the list of available glideins:
- condor_status -schedd -c 'is_glidein=?=true'

Monitor glideins using the Pilot Scheduler:
- ./pilotScheduler.py --glidein-monitor --client-schedd=myschedd01@gridgk10.racf.bnl.gov

Submit Pilots using the Pilot Scheduler:
- ./pilotScheduler.py --queue=bnl-glidein-cc --nqueue=5 --pilot=myPilot --pandasite=mySite --client-schedd=myschedd01@gridgk10.racf.bnl.gov --server-schedd=gridgk10.racf.bnl.gov --server-collector=gridgk10.racf.bnl.gov:9660