P3S Clients

job.py

This client is used to:

- create a job (or a number of jobs) on the server. By default standalone jobs are created, which are not associated with a workflow. This can be done by reading a description of job(s) contained in a JSON file.
- adjust job attributes
- delete a job from the server

An example (with arbitrary names and variables demonstrates how JSON is used:

```
[     {          "name":          "p3s_job",          "timeout":
"100",          "jobtype":      "type1",          "payload":      "/home/p3s/my_executable.py'
"env":          {"P3S":"TRUE","MYVAR":"FALSE"},          "priority":
"1",          "state":          "defined"     } ]
```

pilot.py

Generates a pilot and performs a small number of other functions. The lifecycle of a pilot is as follows: * The process starts and then attempts to contact the server in order to register. At the time of writing handling timeouts is not imeplemented yet, i.e. the pilot will exit in fail condition if there is not response from the server. This behavior will be improved in future development.

- Upon successful registration, the pilot enters a loop. In each iteration it send a job request to the server. If the request is not granted, the pilot sleeps for a configurable period of time and make another attempts. Having exhausted the maximal number of attempts (also configurable), the pilot exits normally.
- If a matching job is identified by the server, the will receive a message containing the description of the payload (e.g. the path to the script/wrapper/executable that needs to be run). The pilot then initiates job execution using the Python “subprocess” machinery.

workflow.py

Creates and manipulates DAGs and workflows. DAGs serve as templates for actual active workflows.

P3S Interfaces

Server interface

- serverAPI.py: translates calls to methods in scripts to HTTP messages sent to the server utilizing *urllib*. It is used by most all clients.

Please see individual documentation for the clients listed above which is contained in files named like WORKFLOW.md etc. Same information converted into PDF (so it's accessible locally and can be printed) can be found in **p3s/documentation**.

There are additional clients which perform tests or a combination of procedures on these objects, such as * injector.py - scans a directory for new files and created workflow with the new data as input * urltest.py (deprecated) - generic HTTP interface to the server * verifyImport.py - verifies that dependecies are satisfied, i.e. certain packages can be imported

Service Scripts

For the most part these scripts are meant to be used on multiple machines e.g. by utilizing pdsh * pgen.sh serves as a basic generator of pilots, in case multiple pilots need to run on same node. The number of concurrent pilots is configurable on the command line.

- preport.sh reports on the number of pilots running on WNs
- pkill.sh remotely kills pilots