Table of Contents

PilotMain

Introduction

The main PanDA pilot module is called pilot.py and is the code that is launched directly by the wrapper. The pilot module downloads a job definition (payload) from the server (or discovers it locally) and prepares for its execution. The pilot forks a separate process (RunJob*) which are the modules that directly executes the payload. The pilot monitors (handled by the Monitor module) the RunJob* process and makes sure that the payload is updating its output files.

The main workflow, discussed in functional detail below, contains a sequential multi-job loop. This refers to the ability of running multiple jobs one after the other until the pilot runs out of time, as defined by schedconfig.timefloor variable. During this time, the pilot is allowed to start new jobs.

Description

This module is the entry point for the PanDA Pilot workflow. It runs job recovery, performs special checks, downloads queuedata, jobs and launches the Monitor instance which in turn launches the payload execution module (RunJob*).

Main workflow

Main functions prior to multi-job loop:

1. pUtil.handleQueuedata()

Download (SiteInformation.getQueuedata()) and update queuedata if requested (by SiteInformation.postProcessQueuedata())

2. Experimenet.specialChecks()

[ATLAS]

```
a) displayArchitecture()
b) displayChangeLog()
c) setPilotPythonVersion(): set ATLAS_PYTHON_PILOT
d) testCVMFS(): not called on HPC:s.
```

3. environment.getBenchmarkDictionary()

Run benchmark test if required by experiment site information object.

4. createSiteWorkDir()

Create the initial pilot workdir (Panda_Pilot_*).

5. Experiment.checkSpecialEnvVars()

[Useless] Check special environment variables.

6. WatchDog class

[Useless if threads are used?]

7. Signal handling

Registration of supported signals: SIGTERM, SIGQUIT, SIGSEGV, SIGXCPU, SIGUSR1, SIGBUS

8. runJobRecovery()

[Reimplemented as DeferredStageout]

Job recovery mechanism was designed to recover remains of jobs run by different pilot on the same WN. The current version only handles remaining output files that were not transferred properly.

Note: the job recovery is only needed if alternative stage-out is not used.

9. diskCleanup()

Perform local disk cleanup using Cleaner.cleanup();

```
a) purgeEmptyDirs(): empty directories are removed in other Panda_Pilot_* directories if not touc
b) purgeWorkDirs(): lingering athena directories (Panda_Pilot_*/PandaJob*) are removed if not tou
c) purgeMaxedoutDirs(): removing lingering maxed out directories (i.e. if MAXEDOUT job state file
hours.
```

```
d) PanDA Pilot directory (Panda_Pilot_*) clean-up will be _investigated_ if directory not touched
for jobs in running, starting, holding states for over one week.
```

Main functions inside multi-job loop:

1. createSiteWorkDir()

Create the site.workdir and write the path to file. chmod to 0770.

2. Experiment.verifyProxy()

Make sure the proxy lifetime is long enough.

3. node.collectWNInfo()

Collect information about the WN.

4. getsetWNMem()

Get the memory limit from queuedata or from the -k pilot option and set it. For non-CGROUPS sites, use resource.setrlimit().

5. checkLocalDiskSpace()

Do we have enough local disk space left to run the job? (skip this test for ND true pilots - job will be failed in Monitor.monitor_job() instead if out of disk space).

6. getJob()

Download a new job from the dispatcher (or from a pre-placed file). Loop over getNewJob();

```
a) getDispatcherDictionary(), construct a dictionary for passing to jobDispatcher and get the pro
b) Read pre-placed job definition.
c) Call pUtil.httpConnect() to download job definition from server.
d) backupDispatcherResponse(), backup response (will be copied to workdir later).
```

7. Experiment.postGetJobActions()

Perform any special post-job definition download actions.

```
a) [ATLAS] verifyNCoresSettings(): Verify that nCores settings are correct.
```

8. Monitor instance

Launch pilot monitoring.

Notes

```
1. Pilot should return standard shell exit code (pUtil.shellExitCode())
2. Keep track of when pilot was launched (pilot_startup). Used by [[Monitor][Monitor]] to measure
3. checkLocalSE() is implemented by currently not used. It could be used to verify that the local
4. Pilot option -y <loggingMode> is deprecated. See its use in [[Monitor][Monitor]] which forward
5. The main function (runMain()) is protected with a try-statement.
6. Currently the pilot module houses the large job recovery algorithm. This will be removed to su
```

Major updates:
-- PaulNilsson - 25-Sep-2012 -- PaulNilsson - 17-May-2016

Responsible: PaulNilsson

Never reviewed

This topic: PanDA > PilotMain
Topic revision: r3 - 2016-06-20 - PaulNilsson