Table of Contents

Table of Contents

PandaPilotDev

General

Introduction

This wiki is to put here all comments about how pilot works. At the end this will be a full pilot documentation.

TO-DOs

Wrappers

Topics

timefloor

```
get the timefloor from the queuedata, the pilot is allowed to run multi-jobs within this limit
if set to zero, only one job will be executed
```

timed_command

```
it executes a command once, but makes sure it
doesn't take longer than a given amount of time (killing the process
if it times out).  Its just like commands.getstatusoutput, but with
a timeout argument...

The code is being used in several places - in the pilot, in pcache,
in the local site mover.  It basically works, but Paul had to apply
a workaround.

The code (my working version) is here:


http://repo.mwt2.org/viewvc/Python/timed_command.py?view=markup

Or to see with lione numbers,

http://repo.mwt2.org/viewvc/Python/timed_command.py?view=annotate

This code worked for a long time without any problems, then at
some point it stopped working correctly - some change in Python
version or something else in the execution environment?

Paul changed the '60' on line 75 to '5' to work around this
bug.  The only downside to that workaround is that if child
processes exit in less than 5 second, they will be reported
as taking 5 seconds - a few seconds get wasted.  (You see this
in job logs)

The 'guts' of the code is the select statement on line 89.
The idea is to keep the parent process from blocking on waiting for
I/O from the child - we don't want any blocking reads.  So, we use
a 'select' to see if the child has I/O for us.  The timeout arg
passed to the 'select' statment controls the maximum time select
will wait.  S, if there is output avaialble from the child,
the select will exit.  If the timeout is exceeded, the select
will exit.  And if the child process terminates, there *should*
be a SIGCHLD delivered, which should also break the select.
But that last bit is not working any more (something changed in
either the way Python or the ATLAS software handles signals?).
```

```
In the case of timed_command, you couldn't use a thread
even if you wanted to - this thing is doing 'exec' (starting a
separate program) and that needs its own process - if you start
a new thread and 'exec' in that thread, your whole process gets
replaced by the 'exec'ed program - not just that thread, but the
whole process.
```

logging

```
the prints out come from three different sources:
        - the stdout itself
        - the function pUtil.tolog()
        - pUtil.dumpFile(),
          which repeats everything from tolog() but sorted in time.
```

gLExec

```
pilot.py knows if glexec is used or not reading the site config thru
        pUtil.readpar('glexec')
and it is passed to runJob.py as a variable in jobargs.
The site configuration is downloaded in a file called queuedata.dat
This file queuedata.dat is downloaded by atlasProdPilot.py in function
ReadQueueParams()
but only if it has not been downloaded already,
the existence of this file is checked.


Then runJob.py reads the value of 'credname' and 'myproxy' from
the job specs
        job.credname
        job.myproxy

This is allowed after modifying Job.py to add 'credname' and 'myproxy'
as attributes to class Job, so their values can be read with

        self.credname = data.get('credname', 'None')
        self.myproxy = data.get('myproxy', 'None')

These values are got from the server because a  message is sent to the server
thru the dictionary jNode in pilot.py
        'getProxyKey':readpar('glexec').capitalize(),
so the server delivers that info when a new job is requested and it can
be read with the method
        Job.setJobDef()
in module Job.py

If gLExec is used, then the transformation script in runJob.py is executed
with the classes and functions provided by module myproxyUtils.py
instead of a simple
        commands.getstatusoutput(cmd)

$ cat runJob.py
    428             if glexec != 'true':
    429                 # glexec can be False (false, FALSE) or maybe None, or Null (NULL)...
    430                 res = commands.getstatusoutput(cmd)
    431             else:
    432                 tolog('Executing transformation script under glexec...')
    433
    434                 import myproxyUtils
    435                 # retrieving the end-user credentials from a MyProxy server...
    436                 MyPI = myproxyUtils.MyProxyInterface(job.myproxy)
    437                 MyPI.userDN = job.prodUserID
    438                 MyPI.credname = job.credname
    439                 # using those retrieved credentials to switch identity...
```

```
440                 glexec_obj = myproxyUtils.executeGlexec(MyPI)
441                 glexec_obj.payload = cmd
442                 # set permissions for particular files
443                 read_write_exec_req = [stat.S_IRUSR,
444                                        stat.S_IWUSR,
445                                        stat.S_IXUSR,
446                                        stat.S_IRGRP,
447                                        stat.S_IWGRP,
448                                        stat.S_IXGRP,
449                                        stat.S_IROTH,
450                                        stat.S_IWOTH,
451                                        stat.S_IXOTH]
452                 fp = myproxyUtils.FilePermissions('PoolFileCatalog.xml', read_write_exec_
453                 glexec_obj.list_nodes.append(fp)
454                 # ... and run the transformation script with gLExec
455                 glexec_obj.execute()
456                 glexec_output = glexec_obj.output
457                 glexec_status = glexec_obj.status
458                 res = (glexec_status, glexec_output)
```

```
if var proxycheckFlag is 'True' in pilot.py
this value is propagated until SiteMover.py where
then the proxy lifetime is checked
in method verifyProxy() of class SiteMover
```

redirection of stderr

```
stderr is redirected in both pilot.py and runJob.py
In pilot.py:

   4847             # redirect stderr
   4848             pUtil.setPilotstderrFilename("%s/pilot.stderr" % (thisSite.workdir))
   4849             sys.stderr = open(pUtil.getPilotstderrFilename(), 'w')

In runJob.py:

    817         # redirect stder
    818         sys.stderr = open("%s/runjob.stderr" % (jobSite.workdir), "w")
```

Modules

BNLdCacheSiteMover.py

CastorSiteMover.py

castorSvcClassSiteMover.py

Cleaner.py

COGSiteMover.py

config.py

dCacheLFCSiteMover.py

dCacheSiteMover.py

dq2SiteMover.py

FileRegistration.py

futil.py

GSIftpSiteMover.py

HUSiteMover.py

JobInfoXML.py

Job.py

JobState.py

lcgcp2SiteMover.py

lcgcpSiteMover.py

LocalSiteMover.py

Mover.py

mvSiteMover.py

myproxyUtils.py

Node.py

OtherSiteMover.py

PilotErrors.py

pilot.py

PilotTCPServer.py

ProxyGuard.py

pUtil.py

rfcpLFCSiteMover.py

runJob.py

RunJobUtilities.py

SiteMoverFarm.py

SiteMover.py

Site.py

SRMSiteMover.py

stormSiteMover.py

timed_command.py

UberftpSiteMover.py

xrdcpSiteMover.py

xrootdSiteMover.py

Major updates:
-- JoseCaballero - 22-Apr-2010

Responsible: JoseCaballero

Never reviewed

This topic: PanDA > PandaPilotDev
Topic revision: r6 - 2011-04-21 - JoseCaballero