oarpy Documentation

Release 0.1.0b2

de nolf

Apr 08, 2022

Contents

CHAPTER 1

oarpy

1.1 oarpy package

1.1.1 Subpackages

oarpy.tests package

Submodules

oarpy.tests.test_all module

oarpy.tests.test_all.**test_suite**()
Test suite including all test suites

oarpy.tests.test_oarjob module

class oarpy.tests.test_oarjob.**test_oarjob**(*methodName='runTest'*)
Bases: unittest.case.TestCase

definition(*seconds*, *name*)

setUp()
Hook method for setting up the test fixture before exercising it.

skipoar()

tearDown()
Hook method for deconstructing the test fixture after testing it.

test_immediate()

test_interrupt()

test_jobfactory()

test_notimmediate()

test_status()

test_suspend()

oarpy.tests.test_oarjob.**test_suite**()
Test suite including all test suites

oarpy.tests.test_oarresource module

class oarpy.tests.test_oarresource.**test_oarresource**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_cores()

test_from_cli()

test_to_cli()

oarpy.tests.test_oarresource.**test_suite**()
Test suite including all test suites

oarpy.tests.test_oarshell module

oarpy.tests.test_oarshell.**test_suite**()
Test suite including all test suites

class oarpy.tests.test_oarshell.**test_utils**(*methodName='runTest'*)
Bases: unittest.case.TestCase

skipoar()

test_cli()

test_installed()

test_oarstat()

Module contents

1.1.2 Submodules

1.1.3 oarpy.oarjob module

class oarpy.oarjob.**Job**(*jobid*)
Bases: object

Manage existing jobs but does not create new jobs

command

definition

exists

exit_code

Returns

- None: interrupted by user
- ==0: success
- !=0: error

fixed_stats(***kwds*)

has_error

interrupt()
Stop a running, waiting or holding job

is_finished

is_intermediate

is_running

is_terminated

is_waiting

jobid

log_base

log_directory

name

needsresume

project

remove_logs()

resource

resume()
Resume suspended job or schedule job for execution

runtime
Effective execution time, excluding queue time

stats

status

stderr

stderr_file

stdout

stdout_file

stop()
Stop a running, waiting or holding job

suspend()
Remove job from schedule or suspend running job (currently not permitted)

time_enqueued
Time this job was enqueue (not scheduled for execution)

time_scheduled
Time this job was scheduled, excluding queue and runtime

time_to_start
Time until the job starts

wait(*states=('Terminated', 'Error')*, *timeout=None*, *refresh=1*, *silent=False*)

Parameters

- **states** (*str or tuple*) – state or states we are waiting for
- **timeout** (*int or None*) – if not none, second before the timeout.
- **refresh** (*int*) – time (in seconds) between two observations of the job state
- **silent** – if False then write to stdout advancement ('.')

working_directory

class oarpy.oarjob.**JobFactory**(*command=None*, *resource=None*, *name=None*, *project=None*, *working_directory=None*, *log_directory=None*, *log_base=None*, ***resource_parameters*)

Bases: object

Define and submit new jobs

cli_arguments

cli_string

log_base

name

submit(*hold=False*)

oarpy.oarjob.**search**(*name=None*, *project=None*, *owner=None*, *state=None*, *start=None*, *end=None*, ***properties*)

Parameters

- **name** (*str*) –
- **project** (*str*) –
- **owner** (*str*) –
- **start** (*datetime*) –
- **end** (*datetime*) –
- **state** (*str*) –

Return list list of jobs

oarpy.oarjob.**submit**(*hold=False*, ***parameters*)

1.1.4 oarpy.oarresource module

class oarpy.oarresource.**Resource**(*host=None*, *nodes=None*, *cpu=None*, *core=None*, *gpu=False*, *walltime=None*, *mem_core_mb=None*, ***properties*)

Bases: object

cli_arguments

cli_string

cores

classmethod from_cli(*cmd*, *properties=None*)

memory

walltime

oarpy.oarresource.**str2walltime**(*s*)

oarpy.oarresource.**walltime2str**(*tdelta*)

1.1.5 oarpy.oarshell module

oarpy.oarshell.**cli_args2str**(**cli_arguments*)

oarpy.oarshell.**cli_getarg**(*cli_arguments*, *flag*)

oarpy.oarshell.**cli_str2args**(*cmd*)

oarpy.oarshell.**execute**(**args*)

oarpy.oarshell.**executejson**(*cmd*, **args*)

Parameters **cmd** (*str*) –

Return tuple out(dict), err(str or None), exitcode(int)

oarpy.oarshell.**installed**(**args*)

oarpy.oarshell.**jobdel**(*jobid*, *signal=None*)

oarpy.oarshell.**jobhold**(*jobid*)

oarpy.oarshell.**jobresume**(*jobid*)

oarpy.oarshell.**jobstats**(*jobid*)

Parameters **jobid** (*int*) –

Return tuple out(str or None), err(str or None), exitcode(int)

oarpy.oarshell.**jobstatus**(*jobid*)

Parameters **jobid** (*int*) –

Return str or None

- None: means job is not registered
- Hold: not scheduled (needs to be resumed)
- Waiting: scheduled for execution
- **Suspended: running process is suspended** (needs to be resumed)
- Launching: process is starting
- Running: process is running
- Resuming: resuming after hold or suspended
- Finishing: process is stopping
- Terminated: process finished successfully
- Error: process finished successfully

oarpy.oarshell.**oarinstalled**()

oarpy.oarshell.**oarjobstat**(*jobid*, **args*)
oarstat for a single job

Parameters **jobid** (*int*) –

Return tuple out(str or dict or None), err(str or None), exitcode(int)

oarpy.oarshell.**oarstat**(**args*)

Return tuple out(dict), err(str or None), exitcode(int)

oarpy.oarshell.**oarsub**(**args*)

Return tuple out(str or None), err(str or None), exitcode(int)

1.1.6 oarpy.timeutils module

oarpy.timeutils.**add**(*dt*, ***kwargs*)

Parameters **dt** (*datetime*) –

Return datetime

oarpy.timeutils.**astimezone**(*dt*, *tz*)

oarpy.timeutils.**fromtimestamp**(*stamp*, *tz=tzlocal()*)

Parameters

- **stamp** (*int*) – seconds since UNIX epoch
- **tz** (*tzinfo*) – local by default (Optional)

Return datetime tz-aware

oarpy.timeutils.**now**(*tz=tzlocal()*)

Parameters **tz** (*tzinfo*) – local by default

Return datetime tz-aware

oarpy.timeutils.**totimestamp**(*dt*)

Parameters **dt** (*datetime*) – assume local tz when tz-unaware

Return int seconds since UNIX epoch

1.1.7 Module contents

CHAPTER 2

Tutorials

2.1 Quick start

2.1.1 Submit

Run an OAR job that prints “Hello world” (only “command” is a required argument):

[8]:
```
from oarpy.oarjob import submit

job = submit(command='echo "Hello word"', name='quickstart',
             project='oarpy', core=1, gpu=False, walltime={'hours':2})
print(job)
job.wait()

if job.exit_code:
    print('Failed:\n{}'.format(job.stderr))
elif job.exit_code is None:
    print('Interrupted:\n{}'.format(job.stdout))
else:
    print('Succes:\n{}'.format(job.stdout))

job.remove_logs()
```

```
Job(1131384)
 name = quickstart
 project = oarpy
 state = Waiting
 owner = denolf
 runtime = 0:00:00
...
Succes:
Hello word
```

2.1.2 Search

Find the job started above in case you know the job ID:

[9]:
```
from oarpy.oarjob import Job
job = Job(1131384)
print(job)
```

```
Job(1131384)
 name = quickstart
 project = oarpy
 state = Terminated
 owner = denolf
 runtime = 0:00:07
```

Find the job started above in case you do not know the job ID (not all arguments are necessary):

[10]:
```
from oarpy.oarjob import search
from oarpy import timeutils
import os

owner = os.getlogin()
start = timeutils.add(timeutils.now(), minutes=-3)
jobs = search(owner=owner, start=start, name='quickstart',
              project='oarpy', state='Terminated')
print(jobs)
```

```
[Job(1131384)]
```

2.2 Getting started

2.2.1 Hello world

OAR job that prints “Hello world” to the standard output.

Minimal

[1]:
```
from oarpy import oarjob

jobdef = oarjob.JobFactory(command='echo "Hello word"')
job = jobdef.submit()
print(job)
job.wait()

if job.exit_code:
    print('Failed:\n{}'.format(job.stderr))
elif job.exit_code is None:
    print('Interrupted:\n{}'.format(job.stdout))
else:
    print('Succes:\n{}'.format(job.stdout))
job.remove_logs()
```

```
Job(1130826)
 name = OAR
```

(continues on next page)

(continued from previous page)

```
 project = default
 state = Waiting
 owner = denolf
 runtime = 0:00:00
...
Succes:
Hello word
```

With resources and postponed execution

[2]:
```
from oarpy import oarjob
from oarpy.oarresource import Resource

resource = Resource(core=1, walltime={'hours':1,'minutes':30}, gpu=False)
jobdef = oarjob.JobFactory(name='helloworld', project='oarpy',
                           command='echo "Hello word"', resource=resource)

job = jobdef.submit(hold=True)
print(job)
job.wait(states='Hold')
# job is waiting for you to resume it
job.resume()
job.wait()

if job.exit_code:
    print('Failed:\n{}'.format(job.stderr))
elif job.exit_code is None:
    print('Interrupted:\n{}'.format(job.stdout))
else:
    print('Succes:\n{}'.format(job.stdout))
job.remove_logs()
```

```
Job(1130827)
 name = helloworld
 project = oarpy
 state = Hold
 owner = denolf
 runtime = 0:00:00
...
Succes:
Hello word
```

2.2.2 Job management

Find all jobs started in the last 5 minutes:

[3]:
```
from oarpy import oarjob
from oarpy import timeutils

start = timeutils.add(timeutils.now(), minutes=-1)
jobs = oarjob.search(start=start)
```

(continues on next page)

(continued from previous page)

```
print(sorted(jobs))
print(set([(job['owner'], job.status) for job in jobs]))
```

```
[Job(1130826), Job(1130827), Job(976220)]
set([(u'ljacques', u'Error'), (u'denolf', u'Terminated')])
```

Retrieve job definition (can be used to resubmit a job):

[4]:
```
from oarpy import oarjob

job = oarjob.Job(1103714)
if job.exists:
    jobdef = job.definition
    stats = job.stats
    print('Initial request:\n {}\n'.format(stats['initial_request']))
    print('Wanted resources:\n {}\n'.format(stats['wanted_resources']))
    print('Properties:\n {}\n'.format(stats['properties']))
    print('Job resource:\n {}\n'.format(jobdef.resource))
    print('Job definition:\n {}\n'.format(jobdef))
```

```
Initial request:


Wanted resources:
 -l "{type = 'default'}/host=1/core=16,walltime=16:0:0"

Properties:
 (((((gpu='YES') AND desktop_computing = 'NO') AND cluster = 'NICE') AND opsys =
↪'debian8') AND interactive = 'MIXED') AND drain='NO'

Job resource:
 -l nodes=1/core=16,walltime=16:00:00 -p "gpu='YES' and drain='NO'"

Job definition:
 -n Gecko_2_0p7um_2474_3031__001_.par --project default -d /mntdirect/_data_visitor/
↪md1189/id17/GeckosHR/Gecko_2/Gecko_2_0p7um_2474_3031__001_/Slices -O OAR.Gecko_2_
↪0p7um_2474_3031__001_.par.%jobid%.stdout -E OAR.Gecko_2_0p7um_2474_3031__001_.par.
↪%jobid%.stderr -l nodes=1/core=16,walltime=16:00:00 -p "gpu='YES' and drain='NO'" /
↪mntdirect/_data_visitor/md1189/id17/GeckosHR/Gecko_2/Gecko_2_0p7um_2474_3031__001_/
↪Slices//./tmpmd1189.sh
```

Remove the log files of all succesfully finished jobs of a particular user and project:

[5]:
```
from oarpy import oarjob
jobs = oarjob.search(owner='testuser', project='oarpy', state='Terminated')
for job in jobs:
    job.remove_logs()
```

2.3 OAR jobs from python

The library provides three main classes:

- oarpy.oarjob.Job: manage existing jobs (status, stop, suspend, resume)
- oarpy.oarjob.JobFactory: define and launch jobs (creates oarpy.oarjob.Job)

- oarpy.oarresource.Resource: optional OAR resources for JobFactory (nodes, cores, gpu)

[1]:
```
import logging
logging.basicConfig()
def debug(b):
    if b:
        logging.getLogger('oarpy').setLevel(logging.DEBUG)
    else:
        logging.getLogger('oarpy').setLevel(logging.INFO)

from time import sleep
from oarpy import oarjob
from oarpy.oarresource import Resource
```

2.3.1 Monitor jobs

Jobs can be monitored based on their job ID:

[2]:
```
# Invalid job ID
job = oarjob.Job(0)
assert(not job.exists)

# Valid job ID
job = oarjob.Job(1)

# Context manager is optional (reduces queries)
with job.fixed_stats():
    if job.exists:
        # Specific statistics
        print(job['assigned_network_address'])
        print(job['assigned_resources'])
        print(job['walltime'])

        # All statistics
        for k,v in job.stats.items():
            print('{}: {}'.format(k,v))

        # Statistics exposed as attributes
        job.is_finished
        job.is_running
        job.is_waiting
        job.is_intermediate
        job.needsresume
        job.time_to_start
        job.time_enqueued
        job.time_scheduled
        assert(job.status==job['state'])
```

```
[u'hpc2-0701']
[2353]
2:00:00
resubmit_job_id: 0
owner: forstner
submissionTime: 2018-08-16 10:16:00+02:00
message: R=1,W=2:0:0,J=I,Q=interactive (Karma=0.000)
jobType: INTERACTIVE
queue: interactive
```

(continues on next page)

(continued from previous page)

```
launchingDirectory: /users/forstner
exit_code: None
properties: ((((desktop_computing = 'NO') AND gpu = 'NO') AND cluster = 'NICE') AND␣
↪opsys = 'debian8') AND drain='NO'
state: Terminated
stopTime: 2018-08-16 10:16:11+02:00
job_user: forstner
assigned_network_address: [u'hpc2-0701']
walltime: 2:00:00
events: [{u'job_id': 1, u'event_id': u'1', u'date': 1534407371, u'type': u'SWITCH_
↪INTO_TERMINATE_STATE', u'to_check': u'NO', u'description': u'[bipbip 1] Ask to␣
↪change the job state'}]
array_index: 1
assigned_resources: [2353]
array_id: 1
dependencies: []
startTime: 2018-08-16 10:16:02+02:00
reservation: None
stdout_file: OAR.1.stdout
types: []
Job_Id: 1
cpuset_name: forstner_1
name: None
initial_request:
scheduledStart: None
wanted_resources: -l "{type = 'default'}/core=1,walltime=2:0:0"
project: default
stderr_file: OAR.1.stderr
command:
```

Jobs can be searched for based on date, name, project, owner and other properties:

[3]:
```
from oarpy import timeutils

end = timeutils.now()
start = timeutils.add(end,minutes=-10)
jobs = oarjob.search(start=start, end=end)

print(str(len(jobs))+' jobs found')

examples = {}
for job in jobs:
    if job.status not in examples:
        examples[job.status] = job

for job in examples.values():
    with job.fixed_stats():
        print(job)
        print(' time_to_start: {}'.format(job.time_to_start))
        print(' time_enqueued: {}'.format(job.time_enqueued))
        print(' time_scheduled: {}'.format(job.time_scheduled))
        print(' runtime: {}'.format(job.runtime))
```

```
15 jobs found
Job(1124788)
 name = PyHST
 project = default
```

(continues on next page)

(continued from previous page)

```
 state = Terminated
 owner = in1081
 runtime = 0:00:54
 time_to_start: 0:00:00
 time_enqueued: 1:46:34.100600
 time_scheduled: 0:00:00
 runtime: 0:00:54
Job(1125510)
 name = None
 project = default
 state = Running
 owner = bona
 runtime = 0:02:28.569919
 time_to_start: 0:00:00
 time_enqueued: 0:00:25
 time_scheduled: 0:00:00
 runtime: 0:02:28.571296
Job(1122760)
 name = None
 project = default
 state = Waiting
 owner = in1096
 runtime = 0:00:00
 time_to_start: 15:25:03.953672
 time_enqueued: 21:21:38
 time_scheduled: 0:00:00
 runtime: 0:00:00
```

2.3.2 Define jobs

The minimal job definition requires only a shell command

[4]:
```
jobdef = oarjob.JobFactory(command="ls")
print(jobdef)
```

```
-n OAR -O %jobname%.%jobid%.stdout -E %jobname%.%jobid%.stderr ls
```

A job can be identified by name and project

[5]:
```
jobdef = oarjob.JobFactory(name='test', project='oarpy', command='ls')
print(jobdef)
```

```
-n test --project oarpy -O %jobname%.%jobid%.stdout -E %jobname%.%jobid%.stderr ls
```

Working and log directories can be specified (current directory by default). These directories must exist.

[6]:
```
jobdef = oarjob.JobFactory(name='test', project='oarpy', command='ls',
                           working_directory='/tmp/oarpy', log_directory='/tmp/oarpy/
↪log')
print(jobdef)
```

```
-n test --project oarpy -d /tmp/oarpy -O /tmp/oarpy/log/%jobname%.%jobid%.stdout -E /
↪tmp/oarpy/log/%jobname%.%jobid%.stderr ls
```

A job definition can also specify the resources required to execute the job:

- nodes: number of nodes (default: 1)

- cpu: number of cpu's per node (default: 1)
- core: number of cores per cpu (default: 1)
- gpu: boolean (default: False)
- mem_core_mb: minimal memory per core (default: 8000 MB)
- walltime: is a number (default: 2 hours) or a dictionary with at least one of keys "days, seconds, minutes, hours, weeks"
- custom properties: e.g. cpu_vendor=('=','INTEL')

This starts nodes × cpu × core processes distributed over the specified nodes and cpu's.

[7]:
```
resource = Resource(nodes=1,core=8,walltime={'minutes':1})
jobdef = oarjob.JobFactory(name='test', project='oarpy',
                           command='ls', resource=resource)
print(jobdef)
```

```
-n test --project oarpy -O %jobname%.%jobid%.stdout -E %jobname%.%jobid%.stderr -l␣
↪nodes=1/core=8,walltime=00:01:00 ls
```

2.3.3 Launch jobs

Function to define test jobs

[8]:
```
def definition(seconds):
    command = 'python -c "from time import sleep\nfor i in range({}):\n print(i)\n␣
↪sleep(1)"'
    resource = Resource(core=1,walltime={'seconds':seconds*3})
    return oarjob.JobFactory(name='test{}'.format(seconds), project='oarpy',
                             resource=resource, command=command.format(seconds))
```

Immediate execution

Schedule job for execution, wait until done and show output:

[9]:
```
if True:
    print("Schedule job")
    job = definition(5).submit()
    print(job)
    print("Wait until finished ...")
    job.wait()
    print(job)
    if job.exit_code:
        print('Failed:\n{}'.format(job.stderr))
    else:
        print('Succes:\n{}'.format(job.stdout))
    job.remove_logs()
```

```
Schedule job
Job(1125513)
 name = test5
 project = oarpy
 state = Waiting
 owner = denolf
 runtime = 0:00:00
```

(continues on next page)

(continued from previous page)

```
Wait until finished ...
...
Job(1125513)
 name = test5
 project = oarpy
 state = Terminated
 owner = denolf
 runtime = 0:00:29
Succes:
0
1
2
3
4
```

Postpone execution

Enqueue job, wait until enqueued, schedule for execution, wait until done and show output:

[10]:
```
if True:
    print("Enqueue job")
    job = definition(5).submit(hold=True)
    print(job)
    print("Wait until enqueued ...")
    job.wait(states='Hold')
    print(job)
    print("Schedule job")
    job.resume()
    print("Wait until finished ...")
    job.wait()
    print(job)
    if job.exit_code:
        print('Failed:\n{}'.format(job.stderr))
    else:
        print('Succes:\n{}'.format(job.stdout))
    job.remove_logs()
```

```
Enqueue job
Job(1125516)
 name = test5
 project = oarpy
 state = Hold
 owner = denolf
 runtime = 0:00:00
Wait until enqueued ...
Job(1125516)
 name = test5
 project = oarpy
 state = Hold
 owner = denolf
 runtime = 0:00:00
Schedule job
Wait until finished ...
...
Job(1125516)
```

(continues on next page)

(continued from previous page)

```
 name = test5
 project = oarpy
 state = Terminated
 owner = denolf
 runtime = 0:00:29
Succes:
0
1
2
3
4
```

Suspend

Schedule job for execution, wait until running, suspend/resume, wait until done and show output:

[11]:
```
if True:
    print("Schedule job")
    job = definition(60).submit()
    print(job)
    print("Wait until started ...")
    job.wait(states=('Running', 'Terminated', 'Error'))
    print(job)
    print("Suspend job")
    try:
        job.suspend()
    except RuntimeError:
        print("This operation is currently not permitted")
    else:
        print("Wait until suspended ...")
        job.wait_needsresume(states=('Hold', 'Suspended'))
        print(job)
        print("Resume job")
        job.resume()
    print("Wait until finished ...")
    job.wait()
    print(job)
    if job.exit_code:
        print('Failed:\n{}'.format(job.stderr))
    else:
        print('Succes:\n{}'.format(job.stdout))
    job.remove_logs()
```

```
Schedule job
Job(1125518)
 name = test60
 project = oarpy
 state = Waiting
 owner = denolf
 runtime = 0:00:00
Wait until started ...
...
Job(1125518)
 name = test60
 project = oarpy
```

(continues on next page)

(continued from previous page)

```
 state = Running
 owner = denolf
 runtime = 0:00:24.181162
Suspend job
```

```
ERROR:root:Cannot suspend job (Jobid=1125518,Error=1,EPERM)
/!\ Cannot hold 1125518 : the job is not in the right state (try '-r' option).
```

```
This operation is currently not permitted
Wait until finished ...
...
Job(1125518)
 name = test60
 project = oarpy
 state = Terminated
 owner = denolf
 runtime = 0:01:23
Succes:
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

(continues on next page)

(continued from previous page)

```
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
```

Interrupt

Schedule job for execution, waiting until running, interrupt, wait until done and show output:

[12]:
```
if True:
    print("Schedule job")
    job = definition(60).submit()
    print(job)
    print("Wait until started ...")
    job.wait(states=('Running', 'Terminated', 'Error'))
    sleep(5)
    print("Interrupt")
    job.interrupt()
    print("Wait until finished ...")
    job.wait()
    print(job)
    if job.exit_code:
        print('Failed:\n{}'.format(job.stderr))
    elif job.exit_code is None:
        print('Interrupted:\n{}'.format(job.stdout))
    else:
        print('Succes:\n{}'.format(job.stdout))
    job.remove_logs()
```

```
Schedule job
Job(1125521)
 name = test60
 project = oarpy
 state = Waiting
 owner = denolf
 runtime = 0:00:00
Wait until started ...
...
```

(continues on next page)

(continued from previous page)

```
Interrupt
Wait until finished ...
...
Job(1125521)
 name = test60
 project = oarpy
 state = Error
 owner = denolf
 runtime = 0:00:53
Interrupted:
```

Tutorials Some examples on how to use the library

oarpy API documentation

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

o

Index

P

R

S

T

W