hpc$_{s}$cheduler Documentation

Release 0.1.2

Lars Buntemeyer

Apr 27, 2022

Contents:

CHAPTER 1

hpc-scheduler

Python code to interact with an HPC scheduler.

- Free software: MIT license
- Documentation: https://hpc-scheduler.readthedocs.io.

1.1 Features

- create jobscripts from templates with python (only SLURM right now)
- interact with the job scheduler (submit and check jobs, check accounting, group jobs)

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install hpc_scheduler, run this command in your terminal:

```
$ pip install hpc-scheduler
```

This is the preferred method to install scheduler, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for scheduler can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/euro-cordex/hpc-scheduler
```

Or download the tarball:

```
$ curl -OJL https://github.com/euro-cordex/hpc-scheduler/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use scheduler in a project:

```
import hpc_scheduler
```

CHAPTER 4

Examples

4.1 Create and submit jobs

```
import os

from hpc_scheduler.scheduler import Scheduler
from configobj import ConfigObj

import logging


def add_job(sc, jobname, logdir, jobdir, header_dict, commands):
    # the fill dictionary is the input for the jobscrip template
    fill             = header_dict
    fill['log_dir']  = logdir
    fill['job_name'] = jobname

    # the name of the jobscript file
    jobscript = os.path.join(jobdir,jobname+'.sh')
    # create a new job
    sc.create_job(jobname=jobname,jobscript=jobscript,
                  commands=commands,header_dict=fill,
                  write=True)


# define scheduler properties
SYS                 = 'SLURM'
pwd                 = os.getcwd()
logDir              = os.path.join(pwd,'logs')
jobDir              = os.path.join(pwd,'jobs')
# this is the file that will hold jobids of jobnames
schLogFile          = os.path.join(pwd,'my_scheduler.jobids.ini')
# the config file for the scheduler
schCfgFile          = os.path.join(pwd,'scheduler.ini')
```

(continues on next page)

(continued from previous page)

```
schCfg              = ConfigObj(schCfgFile)
# the job script template
jobTpl              = schCfg['scheduler']['serial_template']
# a dictionary that contains input for the template
header_dict_default = schCfg['header']


# create a scheduler object, the scheduler will hold a number of jobs
scheduler  = Scheduler(SYS, name='my_scheduler', tpl=jobTpl, logfile=schLogFile)


# example application
# first, we create 10 jobs
nJobs      = 10
for i in range(1,nJobs+1):
    logging.info('creating job nr. {}'.format(i))
    jobname = 'my_job_nr{:03d}'.format(i)
    commands = "echo My job script nr. {}".format(i)
    add_job(scheduler, jobname, logDir, jobDir, header_dict_default, commands)

# now we can submit all jobs
scheduler.submit()
```

4.2 Check jobs

```
from hpc_scheduler.scheduler import Scheduler

# create a scheduler from a jobid logfile
scheduler = Scheduler('SLURM', logfile='my_scheduler.jobids.ini' )

# get a job accounting dictionary
accounting = scheduler.get_jobs_acct()
for jobname, acct in accounting.items():
    print(jobname, acct)

# make a lob of all jobs and status
scheduler.log_jobs_acct()
```

CHAPTER 5

hpc_scheduler

5.1 hpc_scheduler package

5.1.1 Submodules

5.1.2 hpc_scheduler.scheduler module

Scheduler

Classes and methods in `Scheduler` should create jobscripts for different schedulers and help submitting and checking them.

class hpc_scheduler.scheduler.**Job**(*sys*, *jobname=''*, *jobscript=None*, *jobid=-1*, *tpl=None*, *commands=''*, *header_dict={}*, *delimiter='@'*, *control={}*)

Bases: `object`

Class to hold job information

Written by Lars Buntemeyer

Last modified: 06.02.2019

get_acct()

get_log()

grep_log_err()

parse_control()

submit(*write=False*)

write_jobscript(*header_dict=None*)

class hpc_scheduler.scheduler.**Scheduler**(*sys*, *name=''*, *tpl=None*, *logfile=''*, *job_list=[]*, *header_dict={}*)

Bases: `object`

Class to interact with an HPC job scheduler

Written by Lars Buntemeyer

Last modified: 06.02.2019

add_job(*job*)

create_job(*jobname*, *jobscript*, *commands=''*, *header_dict={}*, *write=True*)

get_job(*jobname*)

get_job_list(*filters=[]*)

get_jobids()
Returns a dict containing {jobname:jobid}

Returns:

jobids: A dict containing {jobname:jobid}.

Written by Lars Buntemeyer

Last changes 06.02.2019

get_jobs_acct(*filters=[]*)
Returns a dict containing job accounting

Arguments:

filters: List of strings to filter jobnames before accessing the scheduler database.

Returns:

jobs_acct: A dict containing job accounting information, in the form, e.g.: {jobname:{'State':state,'JobID':jobid,...}}.

Written by Lars Buntemeyer

Last changes 06.02.2019

log_jobs_acct(*filters=None*)
Logs job accounting information.

Arguments:

filters: A string to filter jobnames.

Written by Lars Buntemeyer

Last changes 06.02.2019

read_jobids()

resubmit(*states=[]*)

submit(*jobname=None*)

update_job_list(*job*)

write_jobscripts()

5.1.3 Module contents

Top-level package for scheduler.

CHAPTER 6

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/euro-cordex/hpc_scheduler/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

6.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

6.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

6.1.4 Write Documentation

scheduler could always use more documentation, whether as part of the official scheduler docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/euro-cordex/hpc_scheduler/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

6.2 Get Started!

Ready to contribute? Here's how to set up *scheduler* for local development.

1. Fork the *scheduler* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/scheduler.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv scheduler
$ cd scheduler/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 scheduler tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

6.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/euro-cordex/hpc_scheduler/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ pytest tests.test_scheduler
```

6.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 7

Credits

7.1 Development Lead

- Lars Buntemeyer <lars.buntemeyer@hzg.de>

7.2 Contributors

None yet. Why not be the first?

CHAPTER 8

History

8.1 0.1.0 (2020-09-24)

- First release on PyPI.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index