pypbs Documentation

Release 0.2.0-dev

Tyghe Vallard, Michael panciera

May 12, 2015

Contents

pypbs aims to be a super simple api that sits on top of the common pbs commands. Some pbs commands output xml such as qstat and pbsnodes. This output is in turn parsed and used to build the api.

You may also want to check out the pbs_python project pbs_python requires that you compile the project where pypbs is just a wrapper on top of the pbs commands.

Contents:

CHAPTER 1

Additional Commands

1.1 pbsstatus

pbsstatus is a quick view of your cluster's utilization

```
$> pbsstatus

NP Utilization      Cluster Load    Avail CPU       Used CPU        Total CPU       Running Jobs
        28.12%             20.20%           23              9               32                  9
```

1.2 qpeek

qpeek is a a bit of a rewrite for the qpeek perl script that is a little more robust

Qsub a simple job

```
$> cat <<EOF | qsub -l nodes=1,walltime=00:01:00
while sleep 2
do
    echo "HI"
done
EOF
1.host.example.com
```

Cat the output of the job

```
$> qpeek 1.host.example.com
HI
HI
HI
```

Tail the output of the job in real-time

```
$> qpeek --operation follow 1.host.example.com
HI
HI
HI
```

CHAPTER 2

API Examples

Import modules

```
>>> from pypbs import (
    pbsstatus, pbsxml, util, qstat
)
```

2.1 Node XML

Fetch parsed xml for all nodes

```
>>> node_xml = pbsstatus.get_pbsnodes_xml()
```

Fetch parsed xml for specific nodes

```
>>> nodes = [
    'host1.example.com',
    'host2.example.com'
]
>>> node_xml = pbsstatus.get_pbsnodes_xml(nodes)
```

Now, you have a simple xml.etree.ElementTree root element to work with.

2.2 Job XML

Fetch parsed xml for all jobs

```
>>> job_xml = qstat.get_qstat_xml()
```

Fetch parsed xml for specific jobs

```
>>> job_xml = qstat.get_qstat_xml([1, 5])
```

2.3 Parsing XML

Since it is easier to have a dictionary to work with, pypbs.pbsxml exists to help aide in parsing your xml into a dictionary.

```
>>> my_dict = pbsxml.parse_xml(xml, 'name')
```

2.3.1 Parse Node XML

Now you have a nice dictionary to work with. In the below, only one host was in the output, but it will likely contain many.

Notice that you need to specify the key that you want your dictionary to be indexed by. We are using the name of the node as the index below which works well.

```
>>> node_dict = pbsxml.parse_xml(node_xml, 'name')
>>> from pprint import pprint
>>> pprint(node_dict)
{'host.example.com': {
    'gpus': '2',
    'jobs': [
                '0/1367.host.example.com',
                '1/1368.host.example.com',
    ],
    'mom_manager_port': '15003',
    'mom_service_port': '15002',
    'name': 'host.example.com',
    'np': '12',
    'ntype': 'cluster',
    'power_state': 'Running',
    'properties': [
        'bigmem',
        'tesla'
    ],
    'state': 'online',
    'status': {
        'availmem': '266267940kb',
        'cpuclock': 'OnDemand:2301MHz',
        'energy_used': '0',
        'gres': None,
        'idletime': '358820',
        'jobs': {
            '1367.host.example.com': {
                'cput': '590710',
                'energy_used': '0',
                'mem': '872796kb',
                'session_id': '22367',
                'vmem': '290125588kb',
                'walltime': '591782'
            },
            '1368.host.example.com': {
                'cput': '590758',
                'energy_used': '0',
                'mem': '871552kb',
                'session_id': '22456',
                'vmem': '290060052kb',
                'walltime': '591668'
            }
        },
        'loadave': '5.00',
        'macaddr': '00:00:00:00:00:01',
        'mem': '649416kb',
```

```
        'ncpus': '24',
        'netload': '740386347120',
        'nsessions': '2',
        'nusers': '1',
        'opsys': 'linux',
        'physmem': '264455068kb',
        'rectime': '1431440582',
        'sessions': '23298 22519',
        'state': 'free',
        'totmem': '272647060kb',
        'uname': 'Linux host.example.com 2.6.32-431.20.3.el6.x86_64 #1 SMP Fri Jun 6 18:30:54 EDT 201
        'varattr': None,
        'vmem': '290383028kb',
        'walltime': '581956'
    }
}}
```

Now you can convert that dictionary into a consolidated dictionary that represents stats for the entire cluster(or only for the nodes you select)

```
>>> cluster_dict = pbsstatus.cluster_info(node_dict)
>>> pprint(cluster_dict)
{
    'avail_np': 10,
    'load_utilization': 0.20875,
    'np_utilization': 0.1666666666,
    'running_jobs': 2,
    'total_np': 12,
    'used_np': 2
}
```

Notice how the load_utilization is loadave / ncpus and not loadave / np

2.3.2 Parsing Job XML

We use the Job_Id to index our jobs as that ensures we are using a unique key name

```
>>> job_dict = pbsxml.parse_xml(job_xml, 'Job_Id')
>>> from pprint import pprint
>>> pprint(job_dict)
{'1371.host.example.com': {
    'Checkpoint': 'u',
    'Error_Path': 'host.example.com:/workdir/job.e1371',
    'Hold_Types': 'n',
    'Job_Id': '1371.host.example.com',
    'Job_Name': 'Job1',
    'Job_Owner': 'user.name@host.example.com',
    'Join_Path': 'n',
    'Keep_Files': 'n',
    'Mail_Points': 'abe',
    'Mail_Users': 'user@example.com',
    'Output_Path': 'host.example.com:/workdir/job.o1371',
    'Priority': '0',
    'Rerunable': 'True',
    'Resource_List': {
        'neednodes': '1:gpus=1',
         'nodect': '1',
```

```
        'nodes': '1:gpus=1',
        'walltime': '700:00:00'
    },
    'Variable_List': 'PBS_O_QUEUE=batch,PBS_O_HOME=/home/user.name,PBS_O_LOGNAME=user.name,PBS_O_PATH
    'Walltime': {'Remaining': '1935796'},
    'comment': 'Job started on Tue May 05 at 16:43',
    'ctime': '1430858591',
    'egroup': 'group',
    'etime': '1430858591',
    'euser': 'user.name',
    'exec_gpus': 'host.example.com-gpu/0',
    'exec_host': 'host.example.com/3',
    'fault_tolerant': 'False',
    'gpu_flags': '1',
    'hashname': '1371.host.example.com',
    'job_radix': '0',
    'job_state': 'R',
    'mtime': '1430858591',
    'qtime': '1430858591',
    'queue': 'batch',
    'queue_rank': '1226',
    'queue_type': 'E',
    'resources_used': {
        'cput': '162:00:46',
        'energy_used': '0',
        'mem': '649416kb',
        'vmem': '290383028kb',
        'walltime': '162:16:02'},
    'server': 'host.example.com',
    'session_id': '23298',
    'start_count': '1',
    'start_time': '1430858591',
    'submit_host': 'host.example.com',
    'substate': '42'
}}
```

CHAPTER 3

Installation

1. Download the code

```
git clone https://github.com/VDBWRAIR/pypbs.git
```

2. CD to pypbs directory

```
cd pypbs
```

3. (Optional) Highly recommended to install into a virtualenv
   (a) Create virtualenv

```
virtualenv myenv
```

   (b) Activate virtualenv

```
. myenv/bin/activate
```

4. Install project

```
python setup.py install
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

p

P