PySlurm

Using Slurm via Python

Mark Roberts & Stéphan Gorget

Background

- Why chose Python/Pyrex ?
  - Python
    - Rapid prototyping
    - Used in a lot of commonly used scientific projects
    - Nice data structures
    - Execution speed not essential
  - Pyrex
    - Interface to C libraries
    - C like language
    - Some operation speed ups i.e. integers and loops

Background

- Quick 30 minute evening hack
  - To get to data not available via commands
- Only basic Slurm 1.x API functions
- Simple wrapped C functions
  - Meant wrapping/unwrapping pointers
- All developed on
  - Basic laptop
  - Home server

Recent Effort

- Object design goal
- Python 2.6+
- Based on Cython (www.cython.org)
    - Pyrex fork
    - Optimisations
    - Python3 compatibility
    - Used in mpi4py, SciPy, Pytables, YT
- Slurm 2.3/2.4/2.5
- Blue Gene L/P/Q and Cray XK6

Module Setup

- Default slurm path (/usr)
  - python setup.py build
  - python setup.py install
- Non-default slurm path
  - python setup.py build –slurm=PATH_TO_SLURM
- Seperate slurm library and include paths
  - python setup.py build –slurm-lib=LIB_PATH –slurm-inc=INC_PATH
- Blue Gene Flags
  - Add either –bgl or –bgp or –bgq

API support

- Controller/scheduler
- Job control
- Nodes
- Partitions
- Blocks
- Reservations
- Triggers
- Topology
- Front End Node
- Statistics
- Hostlists

API support

- NOT CURRENTLY SUPPORTED
  - Job submission
  - Job launch
  - Resource allocation
  - Callbacks

Common Use

```
>>> import pyslurm
>>> pyslurm.slurm_kill_job(51, 9, 0)          # Send Signal 9 to Job 51
>>> pyslurm.slurm_set_debug_level(1)          # Set SLURM_DEBUG
>>> # Set Debug Flags
>>> pyslurm.slurm_set_debugflags(pyslurm.DEBUG_FLAG_FRONT_END)
>>> pyslurm.slurm_set_debugflags(pyslurm.DEBUG_FLAG_RESERVATION)
>>> pyslurm.slurm_set_schedlog_level(SCHED_DEBUG)
>>> # Return Tuple Of Slurm Controllers
>>> pyslurm.get_controllers()
>>> # Get String Equivalent from Slurm Error Number
>>> pyslurm.slurm_strerror(pyslurm.slurm_get_errno())
```

Example – Get Job Info

```
>>> import pyslurm
>>>
>>> jobs = pyslurm.job()
>>> jobDict = jobs.get()          # Job data in dictionary, key = JobID
>>>
>>> print "JobIDs - %s" % jobs.ids()
JobIDs - [6, 7, 8, 9]
>>> print "Job Running: %s" % jobs.find('job_state', pyslurm.JOB_RUNNING)
Job Running: [6]
>>> jobDict[6]['partition']
'night'
>>> jobDict[6]['gres']
['gpu:1']
>>> jobDict[6]['resv_id']         # Cray reservation ID
141
```

Common Methods

- Still a work in progress !

load	**# Retrieve Slurm data into object**
get	**# Retrieve object data**
lastUpdate	**# Last time data was updated**
find	**# Locate a field**
find_id	**# Locate a field and associated value**
fields	**# Get all field names (structure)**
	# in an object

Example – Drain A Node

```
>>> import pyslurm
>>>
>>> NodeDict = { 'node_names':  'bgq[000-001]',
                 'node_state':  pyslurm.NODE_STATE_DRAIN,
                 'reason':  'API Test ' }
>>>
>>> Nodes = pyslurm.node()
>>> if Nodes.update(NodeDict):
…        print 'Node update failed'
...    else:
...        print 'Node update succeeded'
...
Node update succeeded
```

Example – Query BG Blocks

```
import pyslurm

if __name__ == "__main__":

    a = pyslurm.block()       # Create object for block class & load with data

    block_dict = a.get()      # Return Slurm block data (dictionary)

    display(block_dict)       # Custom parser/display routine

    a.load()                  # Reload/load new data if any
```

Example – Query BG Blocks

```
RMP15Jl150251234 :
        bg_block_id     : RMP15Jl150251234
        blrtsimage      : /bgl/BlueLight/ppcfloor/bglsys/bin/rts_hw.rts
        cnode_cnt       : 32
        conn_type       : (0, 'Mesh')
        ionode_str      : ['0']
        job_running     :
        linuximage      : /bgl/BlueLight/ppcfloor/bglsys/bin/zImage.elf
        mloaderimage    : /bgl/BlueLight/ppcfloor/bglsys/bin/mmcs-mloader.rts
        mp_str          : bps000
        mp_used_str     :
        node_use        : (0, 'UNKNOWN')
        owner_name      : root
        ramdiskimage    : /bgl/BlueLight/ppcfloor/bglsys/bin/ramdisk.elf
        reason          :
        state           : (4, 'Ready')

Block IDs - ['RMP15Jl150251251', 'RMP15Jl150251241', 'RMP15Jl150251240', 'RMP15Jl150251246', 'RMP15Jl150251256',
'RMP15Jl150251260', 'RMP15Jl150251238', 'RMP15Jl150251239', 'RMP15Jl150251264', 'RMP15Jl150251236',
'RMP15Jl150251237', 'RMP15Jl150251234']
```

Example – Create/Query Reservation

```
start_epoch = int(time.mktime(time.strptime( "2013-12-31T18:00:00",  "%Y-%m-%dT%H:%M:%S")))

a = pyslurm.reservation()
res_dict = pyslurm.create_reservation_dict()

res_dict["accounts"] = "mark"
res_dict["users"] = "mark"
res_dict["node_cnt"] = 1
res_dict["nodes"] = "bps001"
res_dict["users"] = "root"
res_dict["start_time"] = start_epoch
res_dict["duration"] = 600

resid = a.create(res_dict)
if pyslurm.slurm_get_errno() != 0:
        print "Failed - Error : %s" % pyslurm.slurm_strerror(pyslurm.slurm_get_errno())
else:
        print "Success - Created reservation %s\n" % resid
        res_display(a.get())
```

Example - Reservations

```
Success - Created reservation mark_23

Res ID : mark_23
        accounts  : ['mark']
        end_time : Wed Jan 01 04:00:00 2014
        Features  : []
        flags        : 0
        licenses   : {}
        name        : mark_23
        node_cnt : 512
        node_list  : bps001
        partition   : compute
        start_time: Tue Dec 31 18:00:00 2013
        users        : ['root']
```

Hostlists

- API functions exist to create/modify hostlists
- PySlurm hostlists are unexpanded
  - As returned from Slurm API
- Use 3rd party modules to expand
  - Clustershell
  - Python-hostlist
- Python-hostlist may be included in PySlurm

Example - Hostlists

```
>>> jobDict[6]['nodes']
 'nid00[007,024,076-079,082-083,260,277,282,286-287]'
>>> import hostlist
>>> hostlist.expand_hostlist(jobDict[6]['nodes'])
[' nid00007', ' nid00024', ' nid00076', ' nid00077', ' nid00078', '
nid00079', ' nid00082', ' nid00083', ' nid00260', ' nid00277', '
nid00282', ' nid00286', ' nid00287']
```

- Can now use a Python list comprehension on the returned data

Project Status

- Slurm BG emulator very useful
- Complete the API port
    - Improve structure/enum coverage
- Exception handling
- Unicode support & Python 3
- Rewrite some areas & optimise others
- Check for memory leaks
- Keep having fun !

Thank you

Moe Jette & Danny Auble

CSCS for Cray XK6 access

https://github.com/phantez/pyslurm

https://github.com/gingergeeks/pyslurm