Regression Testing on Petaflop Computational Resources

Mike McCarty
Software Developer
May 27, 2010
What? and Why?

• What is regression testing?
  – Regression testing is any type of software testing that seeks to uncover software errors by partially retesting a modified program or system.

• Why should we do regression testing?
  – To track how the performance of a system changes overtime.
  – To test the system after maintenance to make sure it is ready for production.
Depth of testing (Where?)

Diagram showing the interaction between user, login nodes, PBS nodes, aprun nodes, and compute nodes.
Automate Testing

• Regression testing can be a labor intensive task.
  – Testing how scaling curves change for various applications over time
  – Comparing recent test results to previous results; while performing maintenance on a system.

• Rerunning test periodically to build up regression data over time.

• Store result data in a convent format for reporting.

• Automatically generate plots and reports.
The Framework

• Automates the role of the user
  – Test specification
  – PBS script rendering
  – Job(s) submission
  – Test assertions based on output or end state

• Post processing and report generation automation less trivial since they are application specific.
Framework Architecture

Torque
- script_dir
- drop_dir
- template
- name
- renderScript(...)
- submit(...)
- systemQuery(query)
- qstat(jobId)
- jobComplete(jobId)
- jobRunning(jobId)
- deleteJob(jobId)
- holdForCompletion(jobIds)

unittest.TestCase
- batch
- scheduler
- renderScript(...)
- submit(...)
- holdForCompletion(jobIds)
- getAvailableCoreCount()
- runJob(...)
- getHistoricalData()
- getWallTime(jobId)
- getStats(walltimes)
- defaultAssertions(jobId)

settings.py

gorgon.TestCase

Moab
- systemQuery(...)
- getAvailableCoreCount()
- runJob(...)

TestKraken
- test_halo()
- test_simpleio()
- test_budget()

TestKrakenPM
- runJob(jobId)
- test_budget_full_run()
- test_halo_full_run()
- test_simpleio_full_run()

TestMyStuff
- test1()
- test2()
- ...

...
# PBS Script Rendering Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>The number of processors for the job to run on.</td>
<td>12</td>
</tr>
<tr>
<td>machine</td>
<td>The name of the machine that the test will be ran on.</td>
<td>None</td>
</tr>
<tr>
<td>name</td>
<td>The name of the executable.</td>
<td>None</td>
</tr>
<tr>
<td>project</td>
<td>The name of the project to charge the job too.</td>
<td>None</td>
</tr>
<tr>
<td>walltime</td>
<td>The walltime limit for the job.</td>
<td>00:10:00</td>
</tr>
<tr>
<td>pbs_additions</td>
<td>A raw string for specifying custom PBS variables.</td>
<td>Empty String</td>
</tr>
<tr>
<td>env_vars</td>
<td>A raw string for specifying environment variables.</td>
<td>Empty String</td>
</tr>
<tr>
<td>preamble</td>
<td>A raw string for specifying a preamble of code that is inserted into the PBS script before the aprun command is issued.</td>
<td>Empty String</td>
</tr>
<tr>
<td>aprun_options</td>
<td>Options for the aprun command.</td>
<td>&quot;-n $PBS_NNODES&quot;</td>
</tr>
<tr>
<td>options</td>
<td>Options for the application's executable.</td>
<td></td>
</tr>
<tr>
<td>profile</td>
<td>Boolean that controls whether or not to look for a profile from FPMPI.</td>
<td>True</td>
</tr>
</tbody>
</table>
How does it work?

class TestKraken(unittest.TestCase):
    def setUp(self):
        self.batch = Torque(script_dir, drop_dir)

    def test_halo(self):
        jobids = []
        for size in range(1056, 16000, 1056):[i]:
            self.batch.render_script(size = size
                                , name = "halo"
                                , walltime = "00:10:00"
                        
                jobids.append(self.batch.submit())
        self.batch.hold_for_completion(jobids)

mmccarty@krakenpf6(XT5):/lustre/scratch/mmccarty/regression_tests> python ~/sandbox/gorgon/regression_tests/trunk/test.py
Running tests...
.
Ran 1 test in 153.490s
OK
mmccarty@krakenpf6(XT5):/lustre/scratch/mmccarty/regression_tests>  

mmccarty@krakenpf6(XT5):--> qstat | grep mmccarty
400937.nid00016 Halo mmccarty 0 R medium
400938.nid00016 Halo mmccarty 00:00:00 R medium
def test_bugjet(self):
    jobids = []
    for size in range(1056, 16000, 1056)[2:2]:
        script = self.batch.renderScript(size = size
            , name = "bugjet"
            , halftime = "00:15:00"
            , options = "-f"
            , profile = False
            )
        jobids.append(self.batch.submit(script))
        self.batch.holdForCompletion(jobids)

jobs_failed = []
verbose = False
for jobid in jobids:
    output = open("Bugjet.o" + jobid, 'r')
    fail = False
    for l in output.readlines():
        if "3.0-sigma" in l:
            line_data = l.split()
            if line_data[1] != "0.0":
                jobs_failed.append(jobid)
                fail = True
                print "JobID %s: %s processes outside of 3.0 sigma." % (jobid, line_data[1].replace("": ", 
        if "rank" in l and fail and verbose:
            print l.replace("\n", ")
    self.assertTrue(len(jobs_failed) == 0)
Assertion from command line

```
mmccarty@krakenpf6(XT5):/lustre/scratch/mmccarty/regression_tests> python ~/sandbox/gorgon/regression_tests/trunk/test.py
Running tests...
JobID 400918: 1 processes outside of 3.8 sigma.
F..
=====================================================================
FAIL: test_bugget (__main__.TestKraken)
=====================================================================
Traceback (most recent call last):
  File "/nics/a/home/mmccarty/sandbox/gorgon/regression_tests/trunk/test.py", line 62, in test_bugget
    self.assertTrue(len(jobs_failed) == 0)
AssertionError

=====================================================================
Ran 3 tests in 554.473s

FAILED (failures=1)
mmccarty@krakenpf6(XT5):/lustre/scratch/mmccarty/regression_tests>  
```
Running tests

mmccarty@krakenpf7(kt5):/lustre/scratch/mmccarty/regression_tests> python ~/sandbox/gorgon/regression_tests/trunk/TestKrakenPM.py
Running tests...

Error: test_halo_full_run (__main__.TestKrakenPM)
Traceback (most recent call last):
  File "/nics/a/home/mmccarty/sandbox/gorgon/regression_tests/trunk/TestKrakenPM.py", line 37, in test_halo_full_run
    raise
TypeError: exceptions must be classes, instances, or strings (deprecated), not NoneType

Fail: test_simpleio_full_run (__main__.TestKrakenPM)
Traceback (most recent call last):
  File "/nics/a/home/mmccarty/sandbox/gorgon/regression_tests/trunk/TestKrakenPM.py", line 52, in test_simpleio_full_run
    self.assertTrue(False)
AssertionError

Ran 3 tests in 2.922s

FAILED (failures=1, errors=1)
mmccarty@krakenpf7(kt5):/lustre/scratch/mmccarty/regression_tests>
Post Processing

• Some tests may require additional post processing
• “Standard” post processing can be automated
Database Structure

- systems
  - Model: System
  - id: int
  - name: varchar(128)

- measurements
  - Model: Measurement
  - id: int
  - system_id: int
  - application_id: int
  - timestamp: DateTime

- applications
  - Model: Application
  - id: int
  - type_id: int
  - name: varchar(128)

- application_types
  - Model: ApplicationType
  - id: int
  - type: varchar(128)

- data
  - Model: Data
  - id: int
  - job_id: int
  - key: string
  - string_value: string
  - int_value: int
  - float_value: float

- jobs
  - Model: Job
  - id: int
  - measurement_id: int
  - size: int
  - jobID: int
  - walltime: varchar(128)

- process_stats
  - Model: ProcessStat
  - id: int
  - job_id: int
  - stat_name: varchar(128)
  - units: varchar(16)
  - minimum: float
  - maximum: float
  - average: float
Plotting results

- Plots viewed in a RESTful web interface written in Django
- Plots are generated on the fly using data stored in the database
Plotting (continued)
Thank You