TEST-DRIVEN EVALUATION OF GALAXY SCALABILITY ON THE CLOUD

Enis Afgan¹,², Derek Benson³, Nuwan Goonasekera¹

1. VLSCI, University of Melbourne, Melbourne, Australia
2. Galaxy Team
3. Research Computing Centre, University of Queensland
Overview

• GVL Quality Assurance
  • Automated QA tests

• GVL Scaling
  • Many QA tests in parallel = Scalability test
Quality Assurance

Why?

With each new GVL release:
• Do the tutorials run to completion? (Tedious and error prone to check manually)
• Need a quick way of knowing whether things are in reasonable shape
• Need an end-user perspective on how things work

How?

• Using Selenium
• Run full workflows that exercise a complete set of tools
• Check whether tool output == expected output
• Also exercise typical use cases in UI
Demo

- Selenium in action
Writing a simple test

```python
from gvl_test_base import GVLTestBase
from selenium_snippets.galaxy.history import History
from selenium_snippets.galaxy.get_data import GetData
from selenium_snippets.galaxy.rna_analysis import RNAAnalysis
from selenium_snippets.galaxy import import snippet_base

class SimpleTest(GVLTestBase):
    def __init__(self, galaxy_test_context):
        super(SimpleTest, self).__init__(galaxy_test_context)

@snippet_base.ui_action()
def execute_gvl_testcase(self):
    history = History(self.context)
    history.create_new_history("Hello world")
    GetData(self.context).run_upload_file('url', file, 'fastasanger')
    history.wait_for_datasets_to_finish()
    RNAAnalysis(self.context).run_tophat(file, "D melanogaster (dm3)")
    history.wait_for_datasets_to_finish()
```

A library of test snippets available. Composed as desired for a more complex test. Similar to Galaxy’s Twill based tests of internal API.
Issues encountered

• Some elements have no ids = brittle xpaths
Performance Testing

Why?

• How many workers do you need for an RNASeq workshop with 20 users? What size should the workers be?
• How does the GVL scale for different workloads?
• What combinations of storage, instance types, workers etc. are recommended?
• We had mostly anecdotal evidence – needed a more data-driven approach

How?

• 1 thread = 1 user = QA
• Many threads = Multiple Users = Performance
• Use Selenium Grid
Also tried PhantomJS+Ghostdriver as a lightweight Selenium backend – lots of potential – but didn’t work out
What was done?

- Desirable combinations tested.
- All run on the NCI zone (identical hardware)
- Each test had independent resources (e.g. brand new Galaxy/Cloudman instance launched, independent gluster servers, nfs servers used etc.)
- Transient cloud conditions not controlled for
Combinations tested

- storage_type = { gluster, transient, volumes, nfs }
- machine_type = { m1.medium, m1.large, m1.xlarge }
- workers = { 0 to 5 workers }
- workloads = { rnaseq basic tutorial, deseq basic tutorial, microbial assembly tutorial, variant detection basic tutorial }
- simultaneous users = { 1, 5, 10, 20 }
Test loop

```python
for storage_type in storage_types: (4)
    for machine_type in machine_types: (3)
        for worker in workers: (5)
            for workload in workloads: (4)
                for user in number_of_users: (4)
                    time_stuff()
```

- Total = 4 * 3 * 6 * 4 * 4 = 1152
- Total completed so far: 837
- Successful completion for: 655

- Reasons for failure:
  - Turnaround time for a job capped at 1.5 hours
  - Transient capacity issues on the cloud (couldn’t get the machines on demand)
  - The occasional selenium hiccup
What it records

- Time taken for each segment of the test

- Records atop logs at 10 second intervals
  - Provides snapshot of CPU, memory, process and network usage

5 GB of atop logs and timing logs (mostly atop)
Results

A list of configurations to use for a particular scenario (e.g. how many workers for a 20 user rna-seq workshop?)

<table>
<thead>
<tr>
<th>Average of time taken workers</th>
<th>machine_type 1. m1.medium</th>
<th>2. m1.large</th>
<th>3. m1.xlarge</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1425.393948</td>
<td>1569.968806</td>
<td>1266.41437</td>
</tr>
<tr>
<td>1</td>
<td>1346.412998</td>
<td>1176.365253</td>
<td>1084.031046</td>
</tr>
<tr>
<td>2</td>
<td>1269.079958</td>
<td>1124.207206</td>
<td>993.8295797</td>
</tr>
<tr>
<td>3</td>
<td>1938.757942</td>
<td>1380.971698</td>
<td>937.1919329</td>
</tr>
<tr>
<td>4</td>
<td>1558.144051</td>
<td>1133.047089</td>
<td>819.9848138</td>
</tr>
<tr>
<td>5</td>
<td>1177.656928</td>
<td>986.7495938</td>
<td>722.8780805</td>
</tr>
</tbody>
</table>

Graph showing average time taken in seconds for different numbers of workers and machine types.
Results (contd…)

- One worker pays off the most irrespective of the instance type or workload
Results (contd...)  

- Transient vs Volumes  
  - Volumes were slightly outperforming transient storage.  
  - We asked the NeCTAR team why? Turned out that transient storage was rate limited to 25MB/sec to prevent any one VM from hogging the disk bandwidth.

Above effect not visible on zone NCI.
Results (contd…)

• Gluster vs Volumes
  • Differences turned out to be marginal.
  • Not congruent with our previous experiences
  • Reasons unknown so far
Results (contd…)

- Total CPUs in cluster appears to contribute most to overall performance.
  - E.g. Two large (4 core) instances roughly = Single xlarge (8 core) instance
- Therefore – less likely to overprovision if you use many smaller instances (with autoscaling), as opposed to a few larger instances

![Graph showing average time (secs) vs. No. of Workers for different instance types](image-url)
Repository

https://bitbucket.org/gvl/gvl-stress-test

Raw Result Data:

url: https://swift.rc.nectar.org.au:8888/v1/AUTH_377/gvl_performance_results

Detailed Report:

Work in progress
What next?

- Amazon/EC2 vs NeCTAR/Openstack?
- Gluster vs NFS vs PVFS/OrangeFS vs …?
- No. of web runners?
- No. of Job Handlers?
- No. of Nginx workers?

- More in-depth analysis of the data we have right now.