TEST-DRIVEN EVALUATION OF GALAXY SCALABILITY ON THE CLOUD

Enis Afgan[1,2], Derek Benson[3], Nuwan Goonasekera[1]

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:

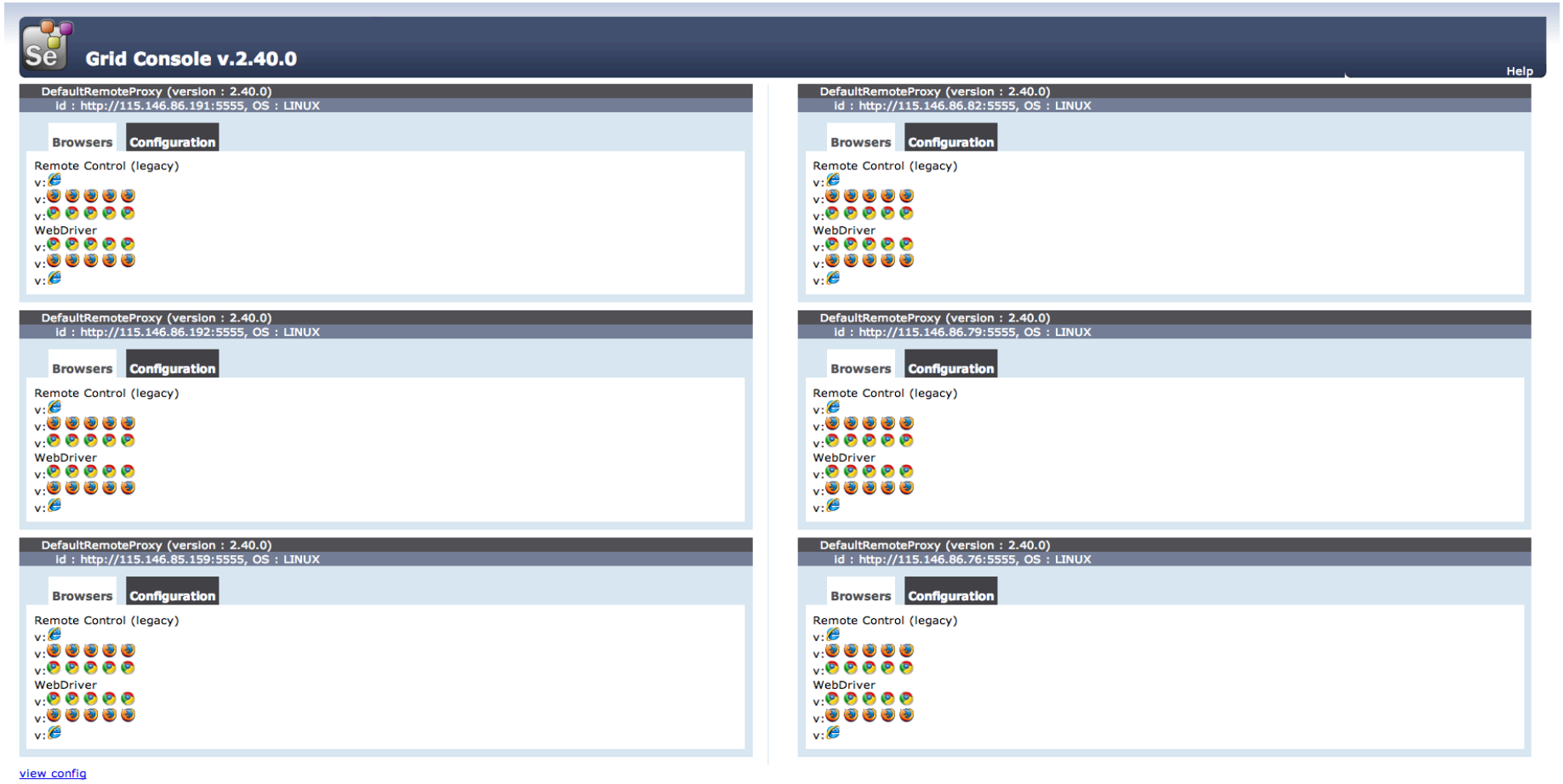
- Do the tutorials run to completion? (Tedious and error prone to check manually)
- Need a quick way of knowing whether things are in reasonable shape
- Need an end-user perspective on how things work

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

```
1  from gvl_test_base import GVLTestBase
2  from selenium_snippets.galaxy.history import History
3  from selenium_snippets.galaxy.get_data import GetData
4  from selenium_snippets.galaxy.rna_analysis import RNAAnalysis
5  from selenium_snippets.galaxy import snippet_base
6
7  class SimpleTest(GVLTestBase):
8
9      def __init__(self, galaxy_test_context):
10         super(SimpleTest, self).__init__(galaxy_test_context)
11
12     @snippet_base.ui_action()
13     def execute_gvl_testcase(self):
14         history = History(self.context)
15         history.create_new_history("Hello world")
16         file = "https://swift.rc.nectar.org.au:8888/v1/AUTH_377/public/RNAseqDGE_BASIC/C2_R1.chr4.fq"
17         GetData(self.context).run_upload_file('url', file, 'fastqsanger')
18         history.wait_for_datasets_to_finish()
19         RNAAnalysis(self.context).run_tophat(file, "D melanogaster (dm3)")
20         history.wait_for_datasets_to_finish()
21
```

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

```
@snippet_base.ui_action()
def edit_history_item(self, name):
    driver = self.driver
    self.switch_to_galaxy_history_frame()
    driver.find_element_by_xpath("//div[@class='dataset-primary-actions'][following-sibling::div[contains(., '"
                                 + name
                                 + "')]]//a[contains(@data-original-title, 'Edit attributes')]").click()
    self.switch_to_galaxy_outer_frame()
    self.wait_for_galaxy_content_frame()
```

Performance Testing

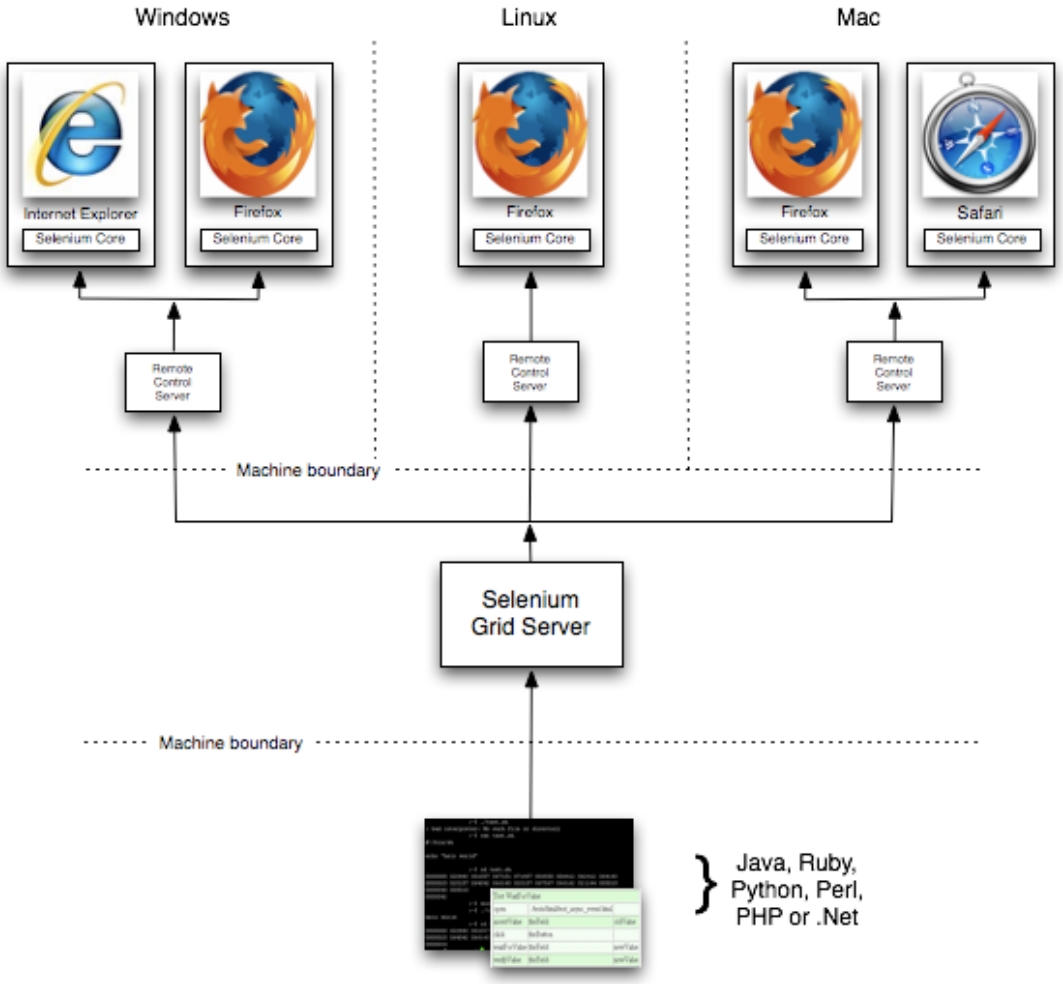
Why?

- How many workers do you need for an RNASeq wokshop 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

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

```
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


```
{
    "timing": [{
        "child_actions": [{
            "child_actions": [],
            "action_start": 1395453178.595498,
            "action_type": 1,
            "action_end": 1395453186.418102,
            "action_name": "register_and_login"
        }, {
            "child_actions": [{
                "child_actions": [{
                    "child_actions": [],
                    "action_start": 1395453186.418553,
                    "action_type": 1,
                    "action_end": 1395453189.653837,
                    "action_name": "create_new_history"
                }, {
                    "child_actions": [],
                    "action_start": 1395453189.654075,
                    "action_type": 1,
                    "action_end": 1395453191.343987,
                    "action_name": "run_upload_file"
                }, {
                    "child_actions": [],
                    "action_start": 1395453191.344252,
                    "action_type": 1,
```

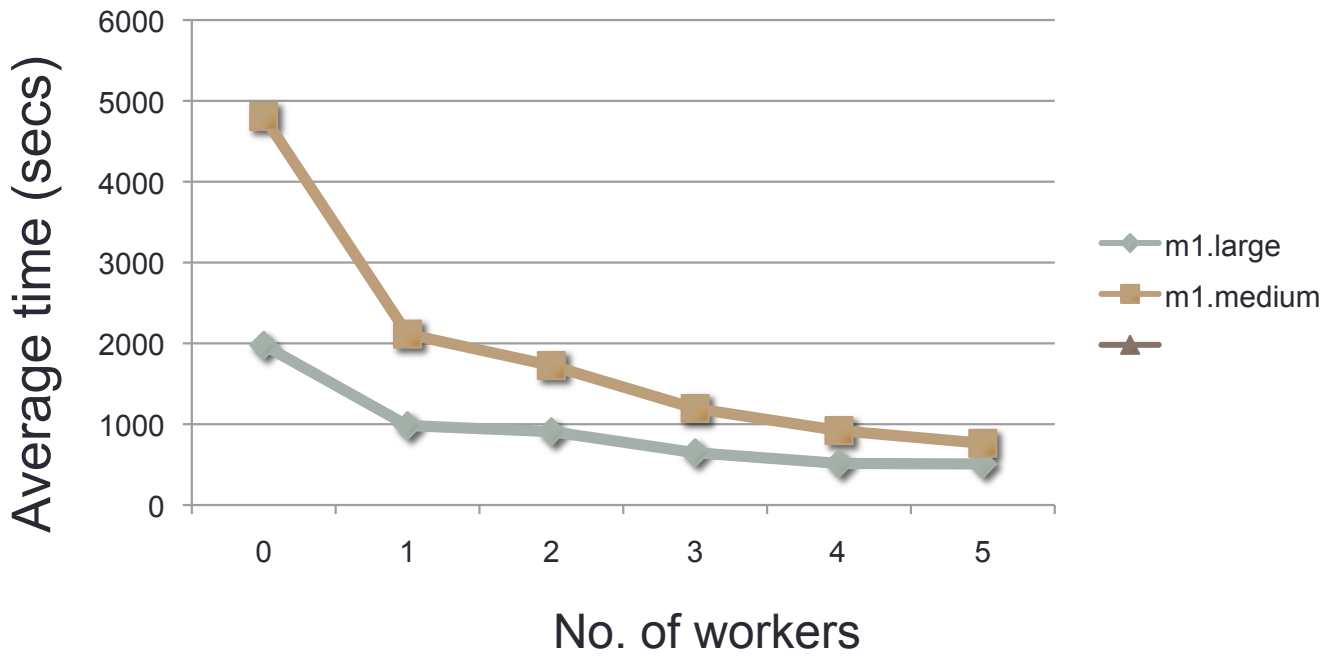

5 GB of atop logs and timing logs (mostly atop)

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

Results

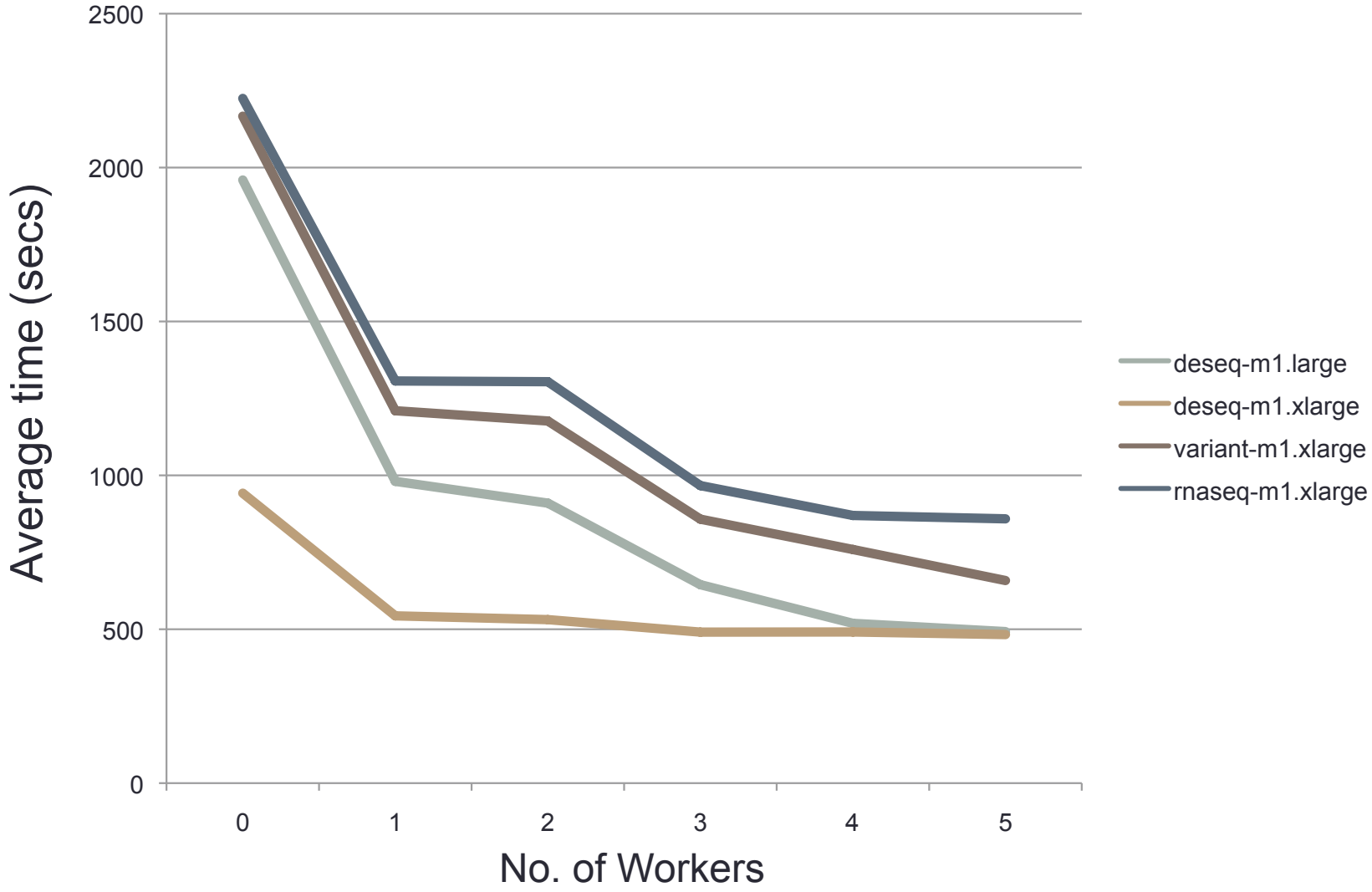
A list of configurations to use for a particular scenario (e.g. how many workers for a 20 user rna-seq workshop?)

Average of time taken workers	machine_type 1. m1.medium	2. m1.large	3. m1.xlarge
0	1425.393948	1569.968806	1266.41437
1	1346.412998	1176.365253	1084.031046
2	1269.079958	1124.207206	993.8295797
3	1938.757942	1380.971698	937.1919329
4	1558.144051	1133.047089	819.9848138
5	1177.656928	986.7495938	722.8780805



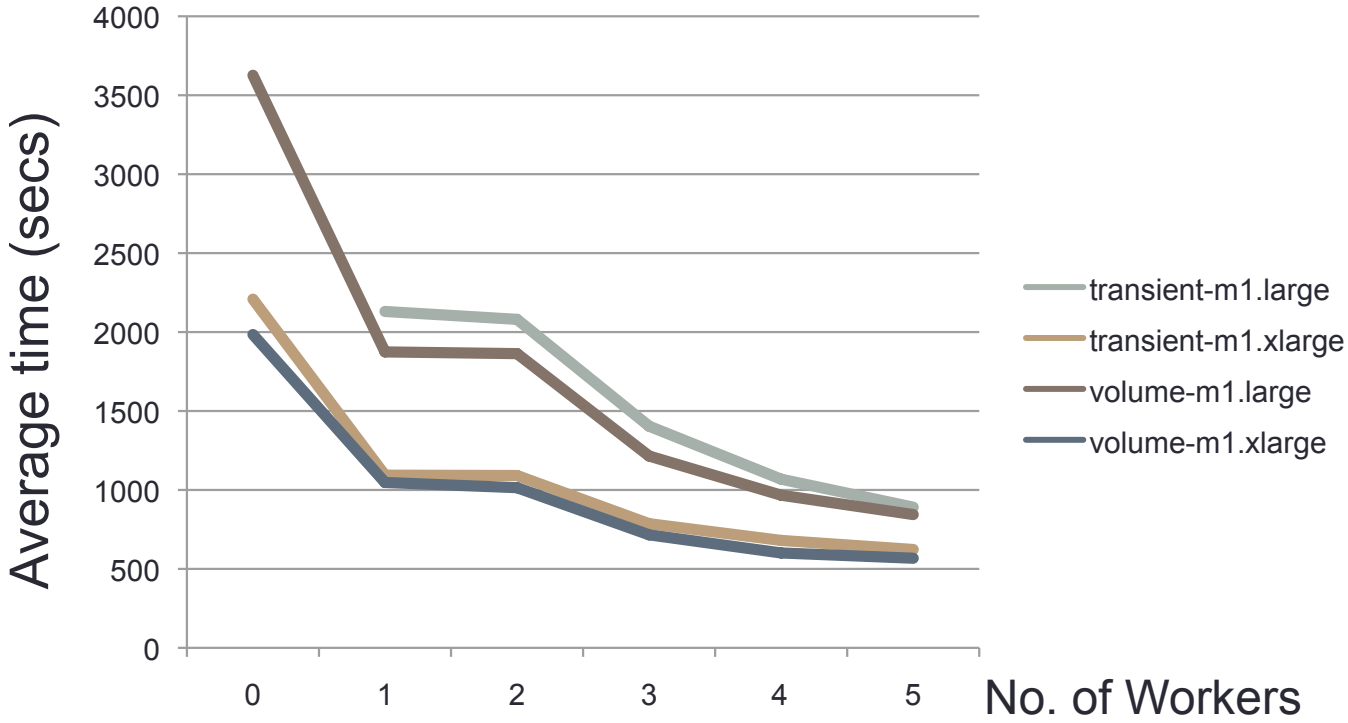
Results (contd…)

- 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.



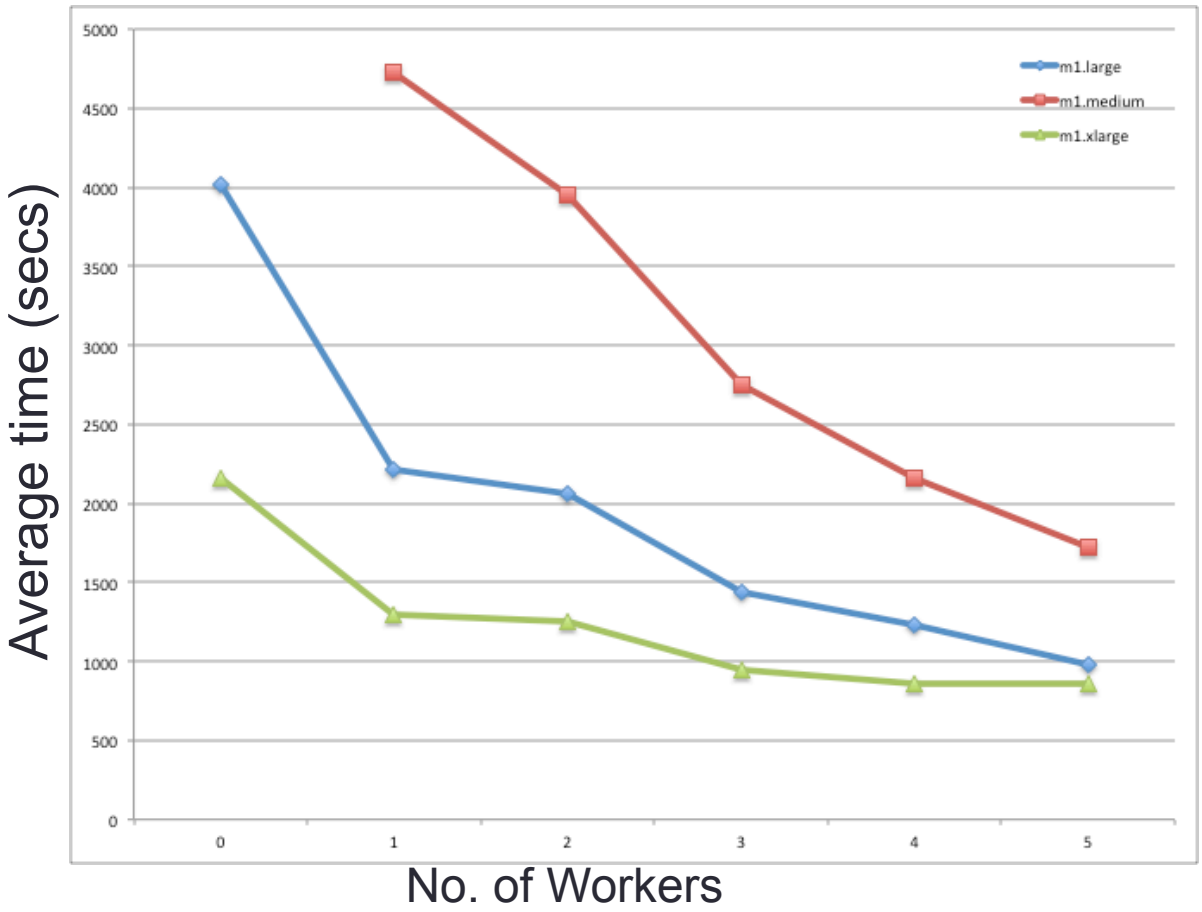
Above effect not visible on zone NCI.

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



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

shortened url: http://bit.ly/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.