



A **UT**/ORNL PARTNERSHIP
NATIONAL INSTITUTE FOR COMPUTATIONAL SCIENCES



Regression Testing on Petaflop Computational Resources

CUG 2010, Edinburgh

Mike McCarty

Software Developer

May 27, 2010

Additional Authors

- **Troy Baer (NICS)**
- **Lonnie Crosby (NICS)**

Outline

- **What is NICS and Kraken?**
- **Background on Regression Testing**
- **Regression Testing Framework**
- **Post Processing**
- **Analysis of Preliminary Result**
- **Discuss Future Work**

National Institute for Computational Sciences University of Tennessee



- NICS is the latest NSF HPC center
- Kraken #3 on Top 500
 - 1.030 Petaflop peak; 831.7 Teraflops Linpack

First academic PF



Kraken XT5

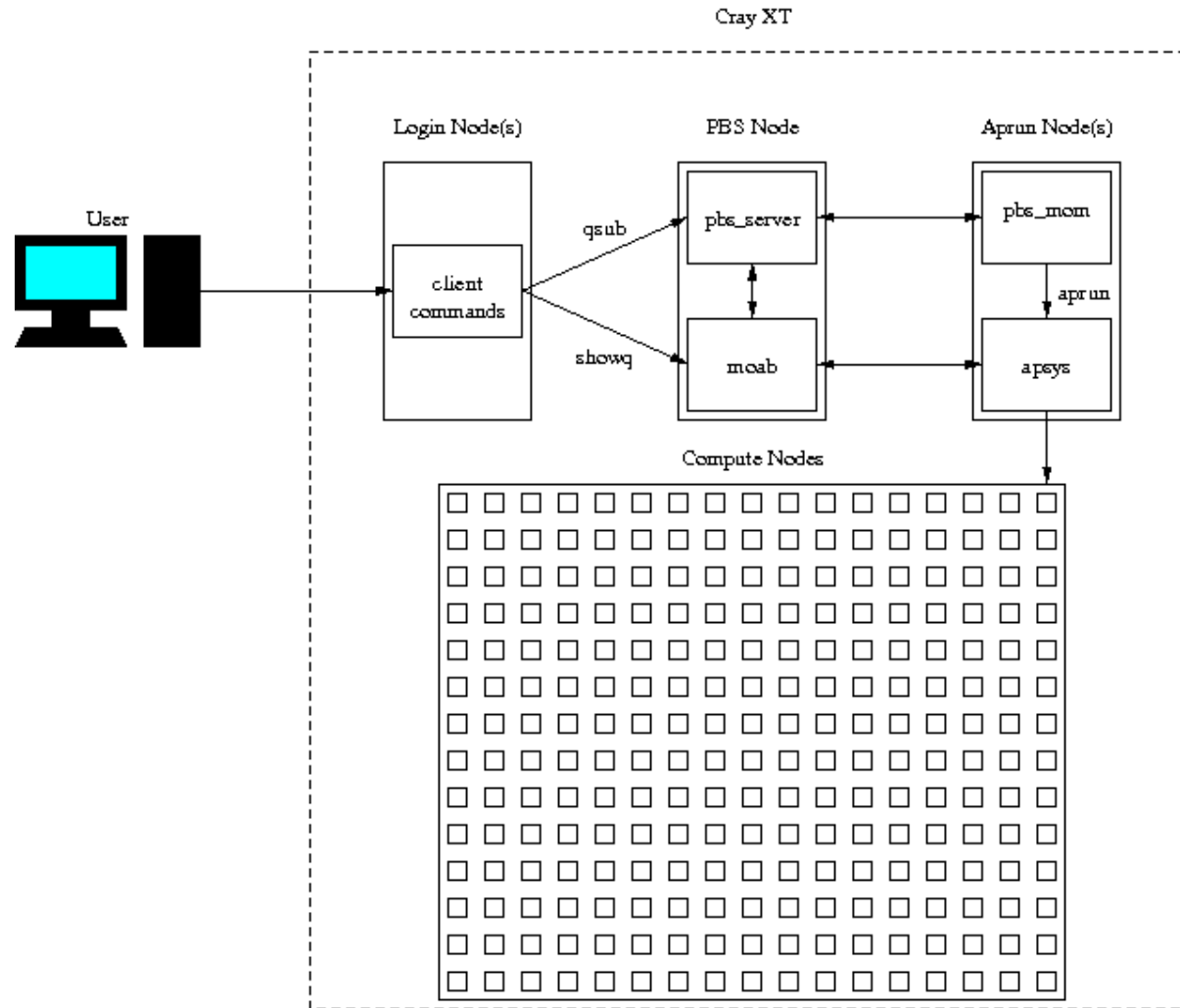


	Kraken
Compute processor type	AMD 2.6 GHz Istanbul-6
Compute cores	99,072
Compute sockets	16,512 hex-core
Compute nodes	8,256
Memory per node	16 GB (1.33 GB/core)
Total memory	129 TB

Regression Testing

- **What is regression testing?**
 - Regression testing is any type of software testing that seeks to uncover 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



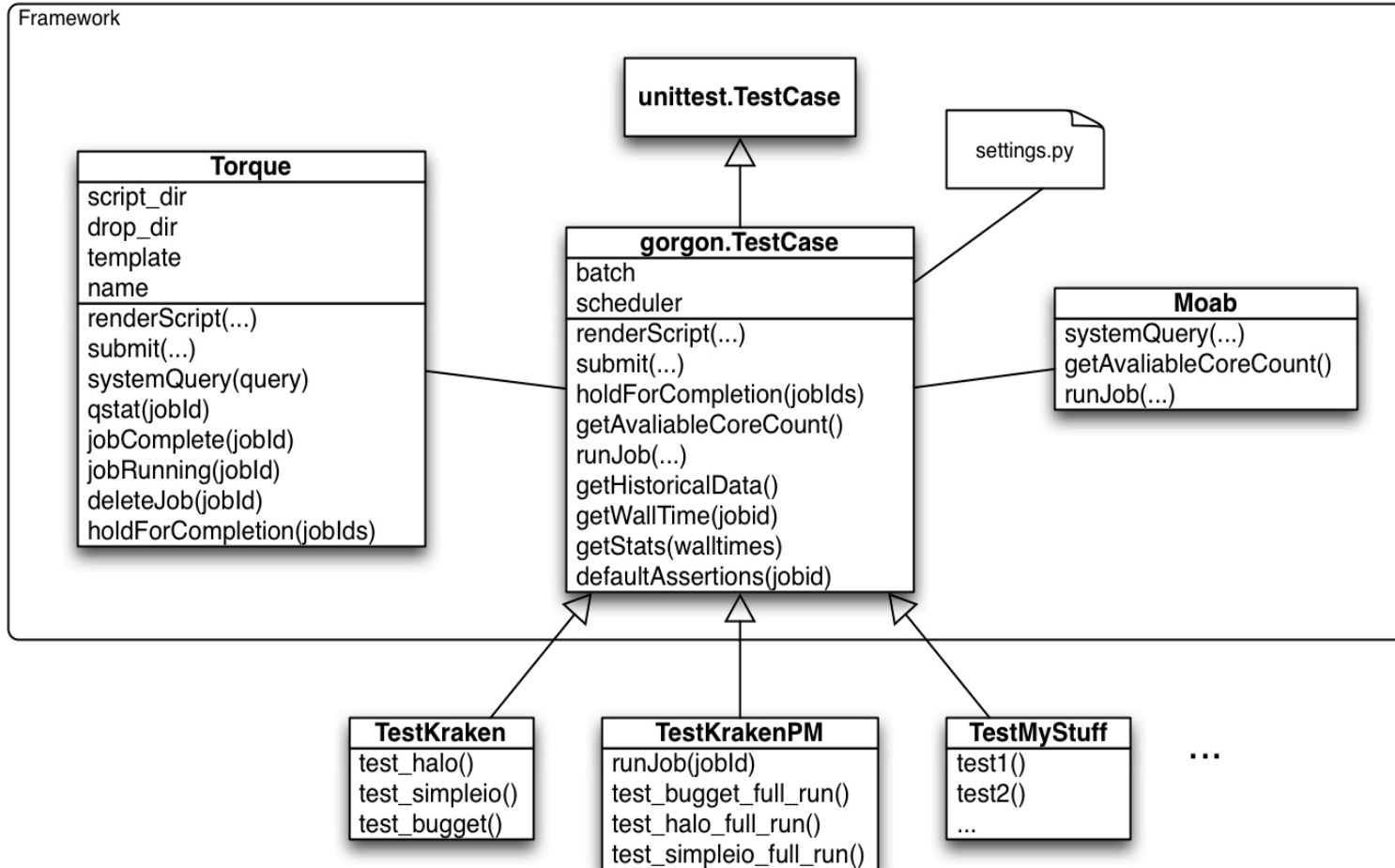
Automate Testing

- **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 convenient format for reporting.**
- **Automatically generate plots and reports from a centralized database.**

The Framework

- **Automates the role of the administrator**
 - Test specification
 - PBS script rendering
 - Job(s) submission
 - Test assertions based on output or end state
- **Post processing and report generation automation are less trivial since they are application specific.**

Framework Design



PBS Script Rendering Parameters

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

How does it work?

```
from TestCase import TestCase
[]
class TestKraken(TestCase):

    def test_halo(self):
        jobids = []
        for size in range(1056, 16000, 1056):
            self.renderScript(size = size
                              , name = "halo"
                              , walltime = "00:10:00"
                              )

            jobids.append(self.submit())
        self.waitForCompletion(jobids)



if __name__ == "__main__":
    print "Running tests..."
    import unittest
    unittest.main()
```



Full Machine Tests and Assertions

```
def test_bugget_full_run(self):  
    size = self.getAvaliableCoreCount()  
    self.renderScript(size = size  
                      , name = 'bugget'  
                      , walltime = "00:20:00"  
                      , options = "-f"  
                      , profile = False  
                      , machine = "Kraken"  
                      )  
  
    jobid = self.submit()  
    self.runJob(jobid)  
    self.holdForCompletion([jobid])  
  
    self.assertTrue(BuggetUtils.analyzeOutput([jobid]))  
    self.defaultAssertions(jobid)
```

Parses output file &
determines pass/failure
(application specific)



Performs real time analysis
(generalized for all tests)

Running tests

```
mmccarty@krakenpf7(XT5):/lustre/scratch/mmccarty/regression_tests> python -/sandbox/gorgon/regression_tests/trunk/TestKrakenPM.py
Running tests...
.EF
=====
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

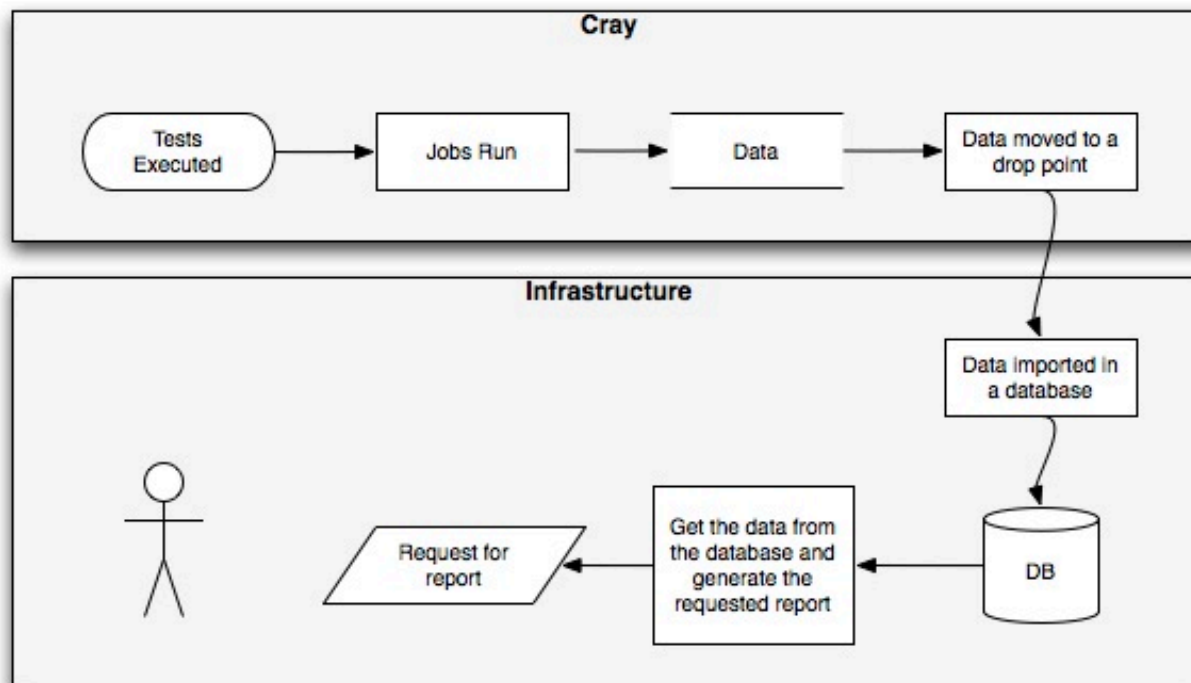
=====
FAIL: test_simpleio_full_run (__main__.TestKrakenPM)
-----
Traceback (most recent call last):
  File "/nics/a/home/mmccarty/sandbox/gorgon/regression_tests/trunk/TestKrakenPM.py", line 52, in test_simpleio_full_run
    self.assertTrue(False)
AssertionError

-----
Ran 3 tests in 2.922s

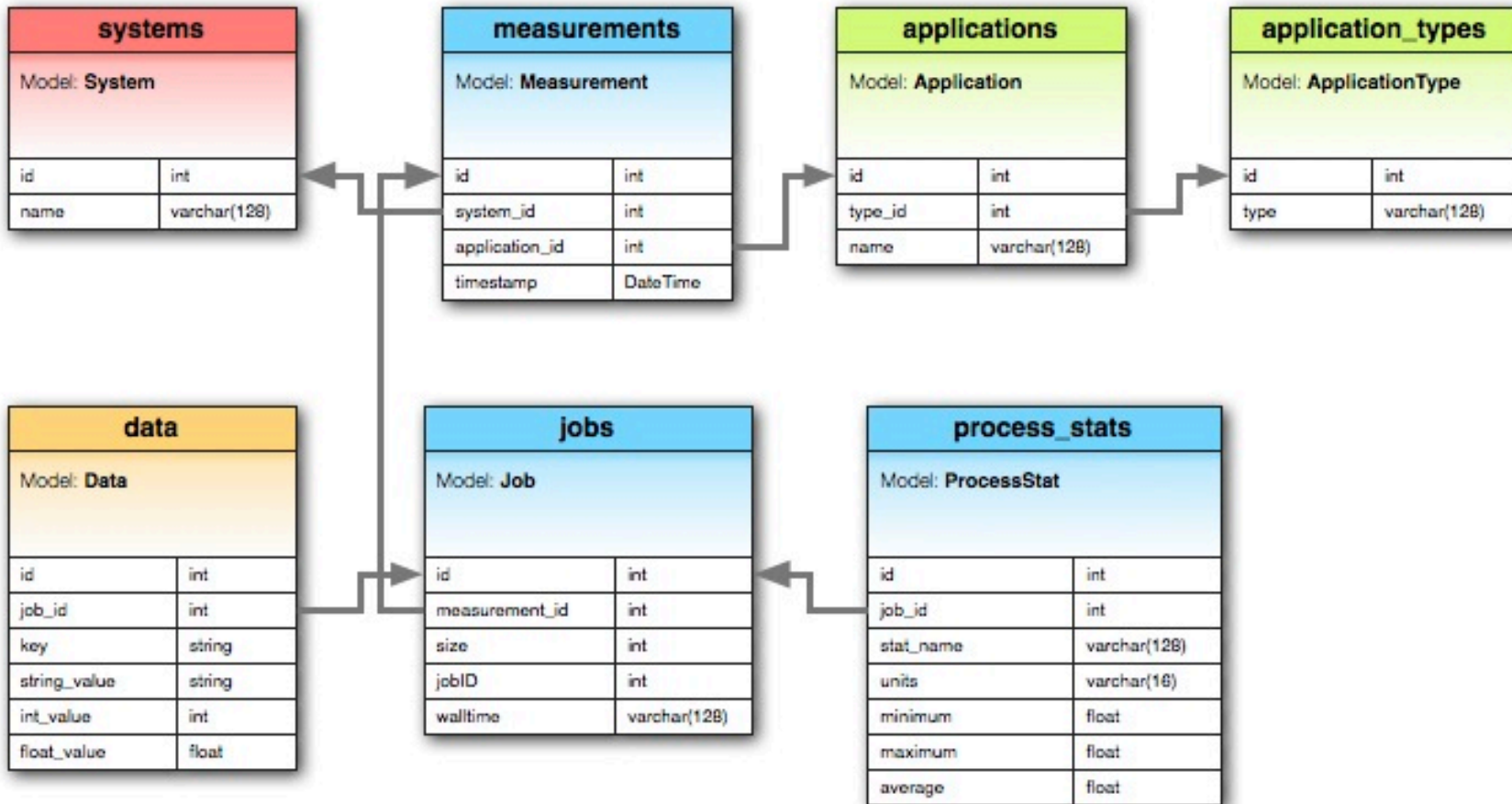
FAILED (failures=1, errors=1)
mmccarty@krakenpf7(XT5):/lustre/scratch/mmccarty/regression_tests> █
```

Post Processing

- **Some tests may require additional post processing**
- **“Standard” post processing can be automated**

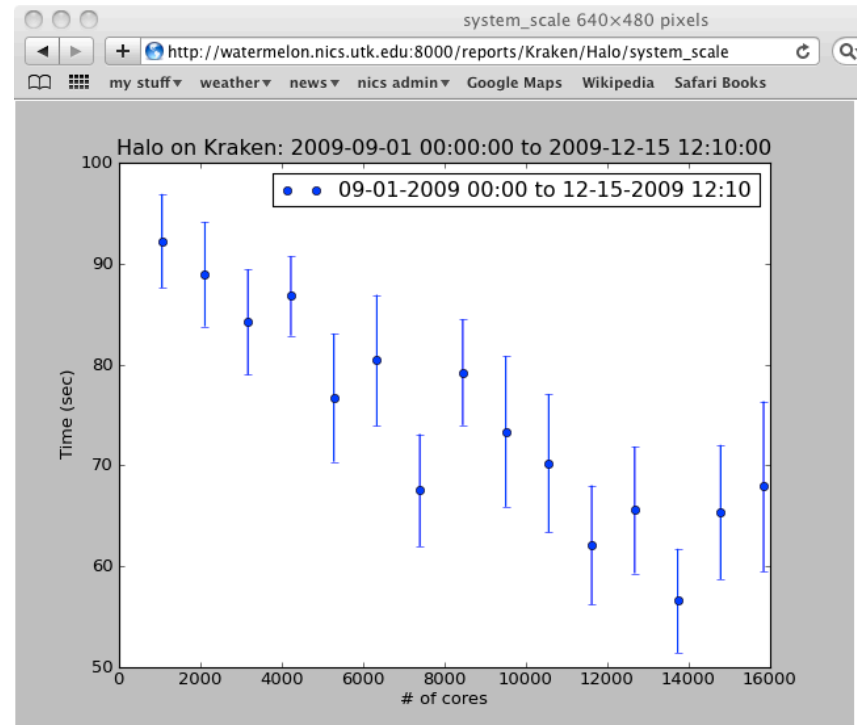


Database Structure

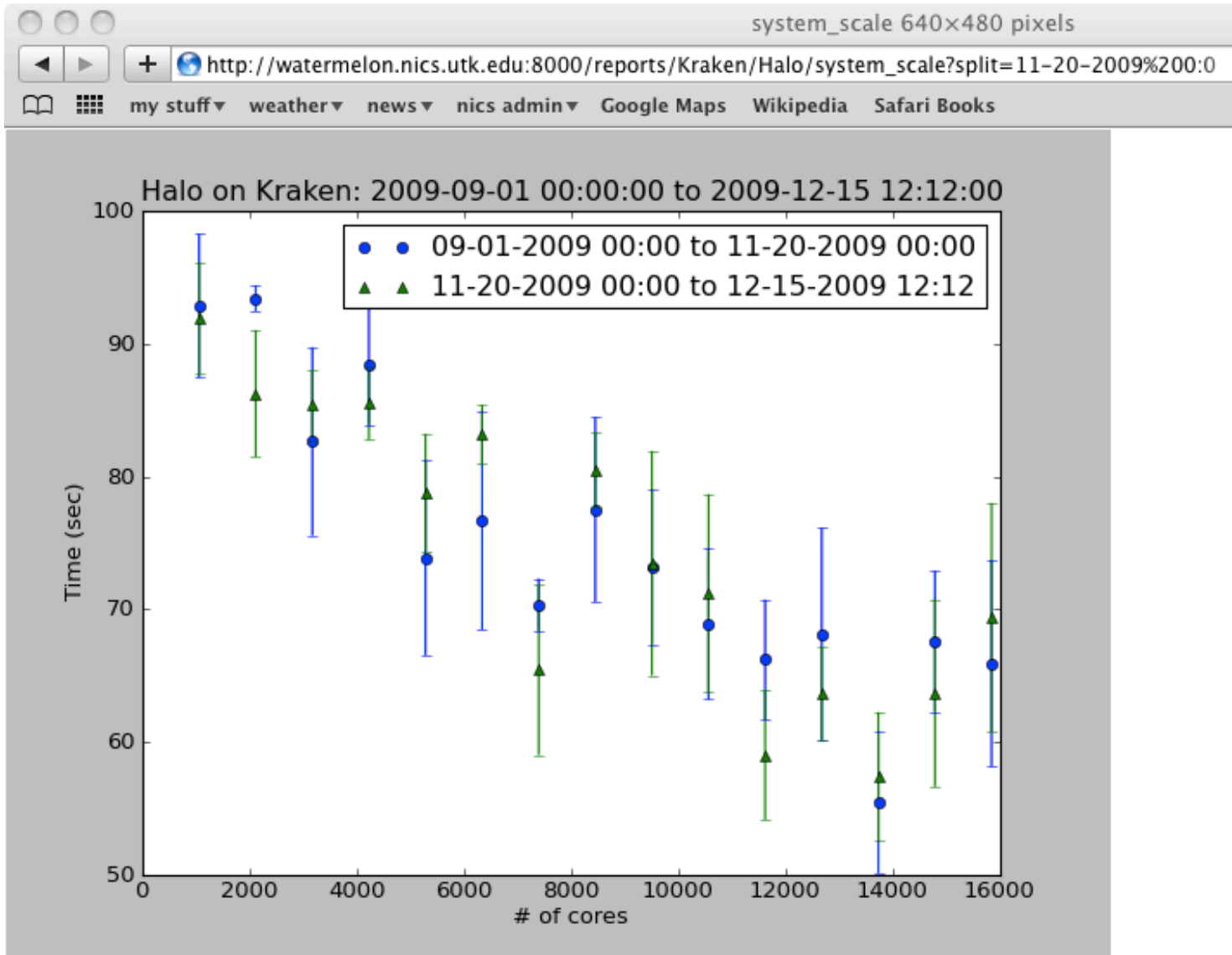


Plotting results

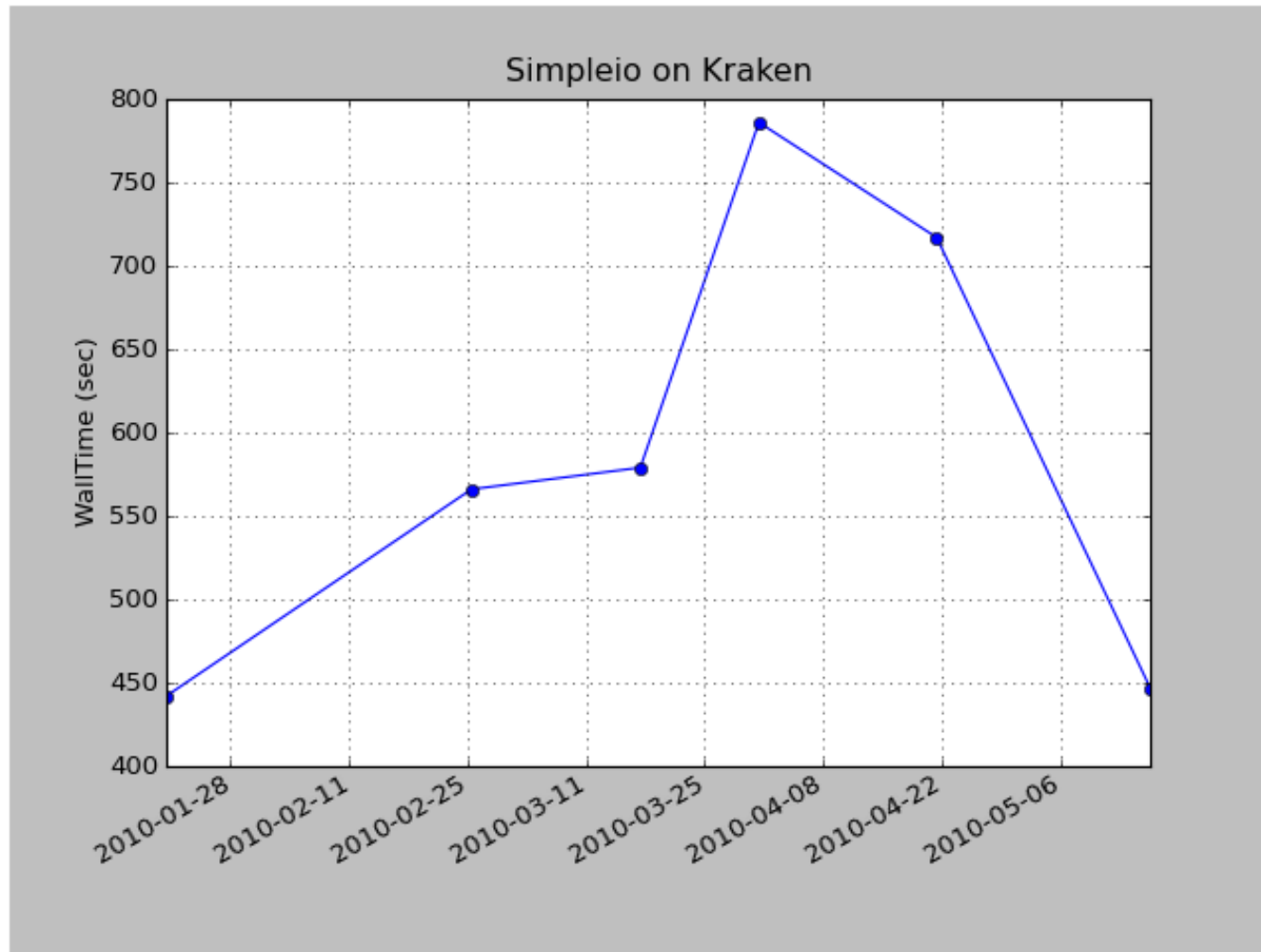
- Plots viewed in a RESTful web interface written in Django
- Plots are generated on the fly using data stored in the database



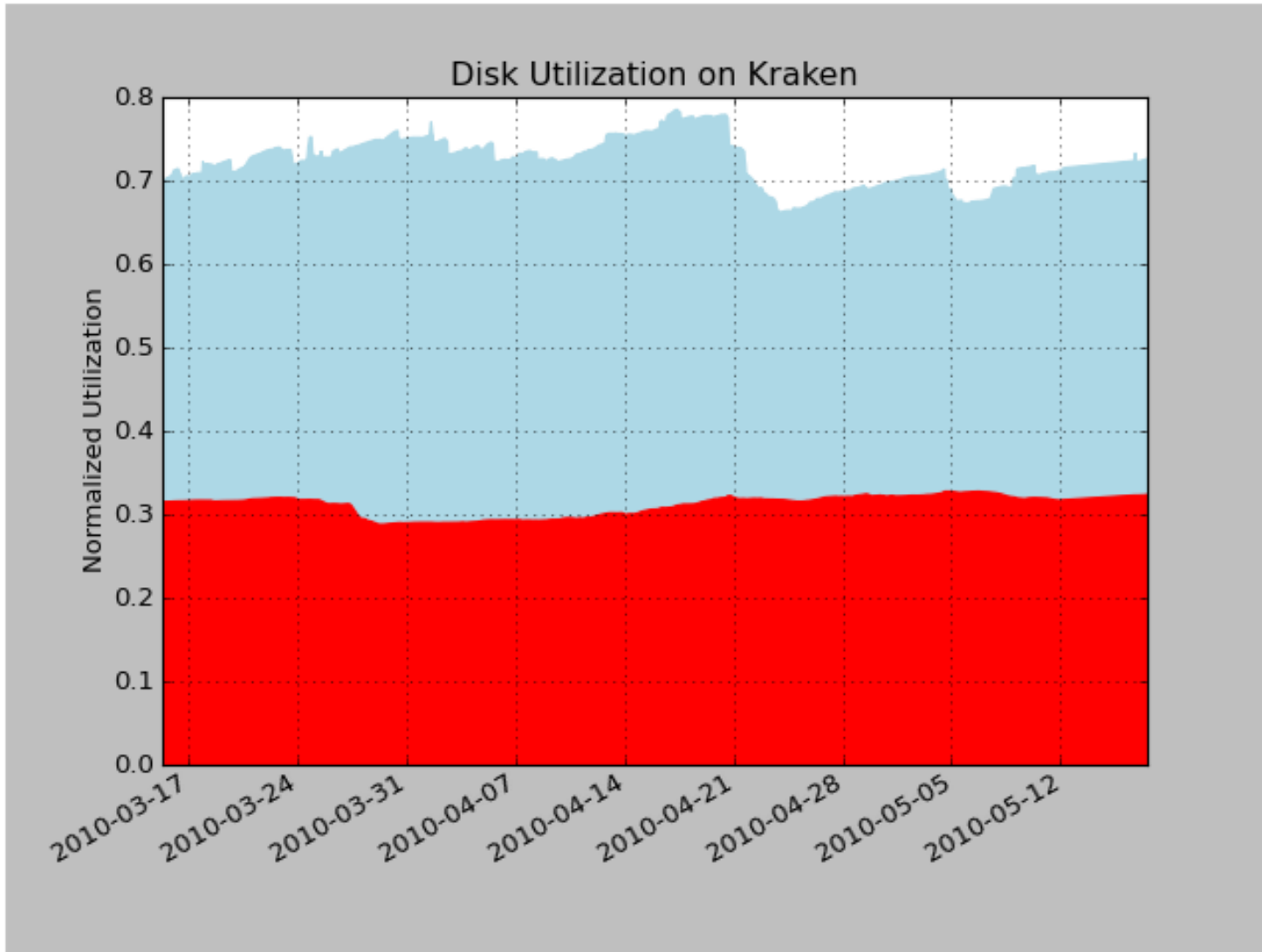
Plotting (continued)



Analysis of Preventive Maintenance Tests

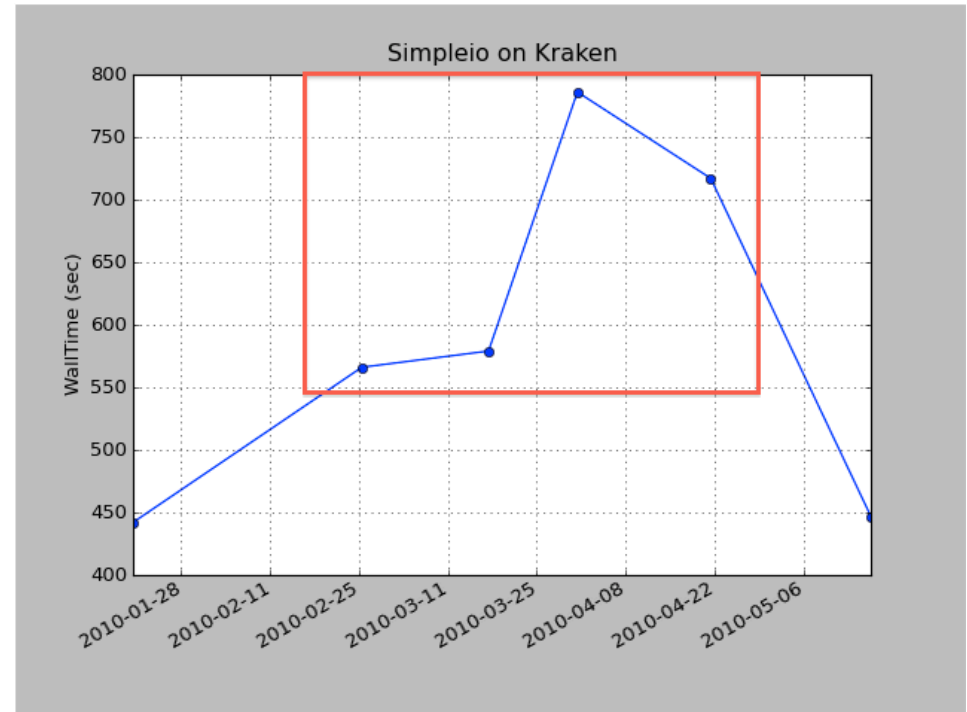


Kraken Disk Usage



Possible explanation

- Cray Bug #759684 Very slow IO after booting (Oracle Lustre)
- Points in the red box are from jobs ran after booting
- The two outer points are from jobs ran after booting, but administrators had already ran SimpleIO manually.



Future Work

- **Add more applications and benchmarks to track the performance for**
 - Memory
 - File System I/O
 - Network
 - MPI
- **Decide when and how often to run tests.**
- **Generalize the framework for other environments and architectures.**

**Thank You
Questions?**