NCTest Capacity Testing

Architecture, Methodology, and Results

Executive Summary

NCSU's Institute for Next Generation IT Systems (ITng) was contracted by the Center for Urban Affairs and Community Service to do a capacity test of the NCTest web application. The goal of the testing is to determine if NCTest can support 250,000 concurrent users. NCTest is a modern Asynchronous Javascript and XML (AJAX) web application similar to gmail or Google calendar. AJAX makes testing challenging since the test equipment must maintain state during the test as opposed to a simple request/response scenario found with vintage HTML/HTTP based web applications.

Given this complexity, each simulated user in the test infrastructure must run a browser to interact with the AJAX web application. The need to run all users in separate browsers implies that many RAM and processing resources are required to simulate 250,000 users. To mitigate this, the clients were accelerated and focused on a small portion of the NCTest infrastructure. Assuming linear scaling, results from this scenario can be multiplied over the entire NCTest infrastructure to estimate it's ability to support 250,000 users.

It is estimated that 250,000 concurrent users will consume about 10% of the NCTest infrastructure with the caveat that the setup section of the NCTest app should be staged over a 30 minute period due to its network requirements. This 30 minute period can be reduced since the setup loading requirements have been reduced, but this change occurred after testing was complete.

Please read below for more details on the testing methodology, findings and results.

Table of Contents

Background

ITng Services was asked by the Center for Urban Affairs and Community Service (CUACS) to prove NCTest's ability to support 250,000 concurrent users. This document describes the architecture and methodology necessary to implement the test.

NCTest is an AJAX web app developed with the Google Web Toolkit (GWT) development environment.

Design Principles

This testing architecture is guided by the following principles:

- Allow reuse when possible. After this test is complete, all testing scripts and configuration will be delivered to CUACS for developing future tests.
- Use well-known and actively developed testing frameworks and libraries.
- Allow the system to scale as testing needs scale.
- Use systems that use fewer RAM and CPU resources per simulated user while maintaining test quality.
- Mimic user characteristics as closely as possible.
- Use the simplest solution to achieve the desired goal.

Architecture

Components

The testing architecture will rely on the following components:

- Selenium (seleniumhq.org)
- Firefox browser (https://www.mozilla.org/en-US/firefox/new/)
- Xvfb (https://en.wikipedia.org/wiki/Xvfb)
- Fedora 17 (fedoraproject.org)
- python programming language (http://www.python.org/)

The following sections list details of each of the components.

Selenium

Selenium is a set of linux libraries and a python package that allows automation of browser operations which in turn exercise web application functions. Selenium is programmed by writing

Python code. Other programming languages are supported too, but python is currently the language of choice of ITng services.

Firefox

The NCTest web application supports Firefox (as does Selenium). Firefox was chosen as the browser for all NCTest capacity tests. Install the selenium IDE add-on on a development machine to automate the recording of selenium scripts. This automates a lot of the work in automating a web app work flow. The resulting script is almost guaranteed not to work, but it is a good start and saves a lot of coding by hand.

Xvfb

Firefox requires a display to run. However, the scale of this test precludes the use of physical displays. Instead, Xvfb (x-windows virtual frame buffer) is used to create a virtual display used by Firefox. Beyond the concept of creating a virtual display, Xvfb also has several interesting features such as screen capture in the event of an error.

Fedora 17

Fedora 17, currently the latest release of the Fedora Linux distribution, was chosen as the OS for the client emulation of web users. Other linux distributions are equally as viable, but ITng Services is most familiar with Fedora.

The following packages are loaded on a Fedora installation with no graphical desktop. Each package is listed in the form of a command to install the package.

1. yum install screen (for convenience)
2. yum install Xvfb (virtual frame buffer for headless operation of firefox)
3. yum install firefox
4. yum install selenium-core
5. yum install selenium-server
6. yum update (to make sure everything is up to date)

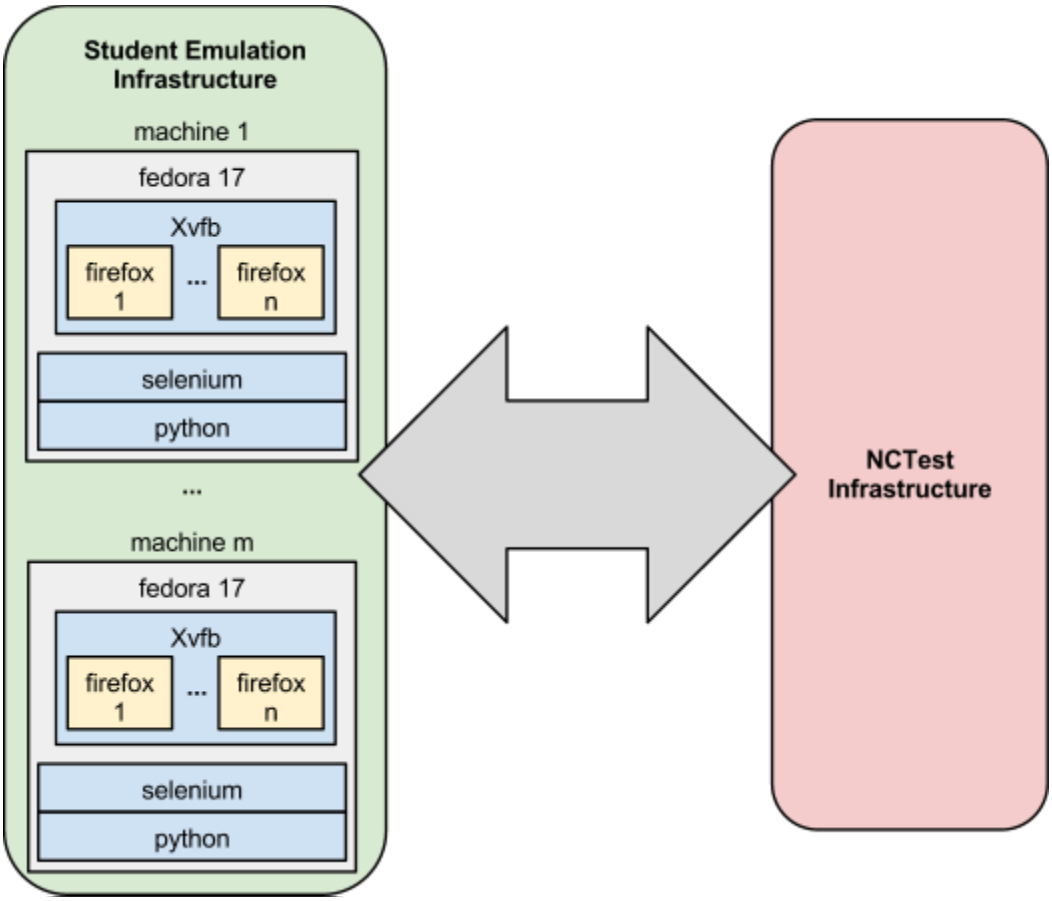
Python

As mentioned above, Python is the programming language for all automation of the testing. The following is a list of python packages necessary for the testing. Each package is listed in the form of a command to install the package.

1. easy_install pip
2. pip install selenium
3. yum install gcc; pip install fabric (only needed from the controller machine)

Topology

The following diagram describes the topology of the test.



Several firefox instances are driven with Selenium in a virtual frame buffer in an installation of Fedora 17. This structure is repeated in other machines until the desired number of concurrent firefox instances are achieved.

Automation

A script will be called on all machines in the topology. This script will take the following parameters:

- start user#
- end user#
- test start time (in epoch)

These scripts may be called via ssh from a 'control' machine. The scripts must be called in ample time to create the randomized wait times (see wait times) and start all processes/threads. These processes will then wait for their start times to begin. The start times will be randomized somewhat within a two minute window so as not to synchronize all calls to the infrastructure. The user number are described is more detail below (see credentials).

User Parameters

The following sections will describe parameters necessary for characterizing the emulated users.

Credentials

The credentials for a test are as follows:

- userid for teacher
- password for teacher
- student name

The idea is a teacher will log into the NCTest application on a group of machines in a testing lab and then select a student for each machine. The test automation script will need to accomplish the same authentication before taking the test. The 250,000 concurrent user goal for testing refers to 250,000 students.

For the sake of the test, it will be assumed that a teacher has 100 students. For 250,000 students, 2,500 teachers are required. The following table shows the contents of a configuration file for users that each test script will read from. This file is depicted as a table, but will most likely be represented as a comma separated value (CSV) file.

This table shows how the user number, userid, password, and student parameters are laid out. This configuration must be mirrored in the NCTest infrastructure and is the responsibility of NCTest personnel.

User Number	Userid	Password	Student Number
1	1	1	1
…	…	…	…
100	1	1	100
101	2	2	1

…	…	…	…
250000	2500	2500	100

Wait Times

Some students complete tests faster than others. To emulate this, test completion data is used to create a stochastic model. This model is used to determine randomized wait times between test questions.

The following is a procedure to determine the wait times between test questions in the user work flow. This procedure will be run once during the setup phase for each emulated user on each machine.

1. **Determine overall time to take the test (T_{total})**

   Appendix A lists 'time on test' histogram data found in a document titled 'Test Time Statistics for the 2012 NC Field Tests - 2011-12'. Using this data, the startup script will do the following:

   a. get a random number between 0 and 1 (inclusive) over a uniform distribution
   b. a row in the histogram is selected by comparing the random number with the cumulative % column
   c. the Time Bin is found from the selected row
   d. the middle point of the bin ((max time - min time) / 2) is used for the total test time (T_total)

2. **Determine time between questions in a test ($T_{question}$)**

   For a given 'student', the wait time between test questions is calculated by:
   $T_{question} = T_{total} / \#Questions$

Note the following characteristics of this procedure:

- This procedure calculates a minimal amount of random numbers. Random number calculation can be computationally heavy and could consume significant amount of computing resources and time for large numbers of emulated users.
- For any given student, the time between all questions is equal.
- The actual test completion time for a user will probably be greater than the target test completion time. This is because the application latency is not considered when calculating the time between questions.

Metrics

All metrics will be gathered/sampled from the NCTest infrastructure during the test. The following is a suggested list of metrics. Gathering of these metrics are the responsibility of NCTest personnel.

- number of concurrent users
- cpu utilization
- RAM utilization
- network utilization
- disk I/O

Scaling

The limit to scaling this test beyond the 10,000 users is a function of the web app infrastructure limits, the limits to how many users can be emulated per machine, the number of machines, the capacity of the network supporting the emulated users, and the network capacity between the emulated users and the NCTest infrastructure. This initial test of 10,000 emulated users will inform many of these unknown limits.

Results

Configuration

Client-side Setup

Selenium

The Selenium scripts (listed) in the nctest.py script in the appendix is the result of much effort and the details are beyond the scope of this document.

OS Tuning

Both the default number of concurrent processes and number of open files were reached while attempting to scale up the number of concurrent emulated users. This was fixed by modifying the /etc/security/limits.conf file with the following lines:

```
msbrown3       soft    nproc   32768
msbrown3       hard    nproc   32768
msbrown3       soft    nofile  1000000
msbrown3       hard    nofile  1000000
```

This modification allows the number of concurrent processes to be as much as 32768 and the number of open files to be 1M.

Hardware Tuning

Since many firefox instances require over 100MB of RAM for the test, it was assumed that system RAM would be the limitation for each server used to emulate users. Therefore we loaded seven IBM Bladecenter HS22 blades with 96GB of RAM. We were able to fit 750 concurrent users in each of these systems.

However, it was determined that CPU was the bottleneck since the delay between questions was >30sec regardless of the amount of test acceleration (where acceleration is the divisor of expected wait time between questions). After some experimentation, it was found that 200 concurrent users can run in the blades while barely saturating the CPUs. This resulted in a consistent 3 seconds delay between test questions.

After determining this limit, RAM from the 7 blades were spread into 12 blades such that each blade has at least 48GB of RAM.

Effective Number of Emulated Users

Since 2400 users were run at approximately 20 times the pace of the average user described from the empirical test duration data in the Appendix, we are able to emulate approximately 48,000 concurrent users.

Userids are chosen such that a small portion of the NCTest infrastructure is utilized during the test. The idea is to measure the percent utilization and then linearly extrapolate to 70% utilization to determine a reasonable number of users that can be supported by the infrastructure.

Workflow Workarounds

Some of the questions in the test were difficult or impossible to automate using Selenium. These questions are skipped. This occurred with 2 of the 57 questions in the test (questions 8 and 39) and are documented in the Selenium code. This should have negligible impact on the findings of the capacity tests.

Server-side Setup

Hardware Configuration

Traffic was managed via a hardware load balancer, with one database server to manage logins, and one to store student responses. The web traffic was processed using one web server.

Software Configuration

The database software used during the test was MariaDB, while the web server ran Apache-Tomcat.

Findings and Conclusions

Setup Phase of Testing

During the 8.5 minute setup phase of the tests, 2400 users generate 8% CPU utilization and 150Mbps network utilization on one web server. Assuming linear scaling and use of the entire NCTest web infrastructure, accommodating 250,000 users would consume 22% of all web servers' CPUs and generate 15.625Gbps inbound on the NCTest network. Since the network is limited by a single 10Gbps link from the load balancer to the NCSU network, this becomes a bottleneck. Therefore, a minimal setup window for 250,000 users must be at least 13.28 minutes. To mitigate the risk of over-running the NCTest infrastructure network, we recommend a 30 minute window for test setup statewide when anticipating a high number of concurrent users.

Note: Testing was conducted using a model of the NCTest content as it existed during the 2011-12 school year. Since testing completed, modifications to the application have reduced its overall size, chiefly by reducing the size of the setup load by approximately 1.4 megabytes. This should yield a margin of additional safety to the conclusions presented in this report. Time pressures preclude retesting the application to quantify this margin.

Question and Answer Phase of Testing

After scaling the Selenium tests to 200 users per blade over 12 blades while setting the delay between questions to a minimum (approx 3 seconds), the web server sees a processor utilization of 8% and the database server experiences a processor utilization of 18%.

The NCTest infrastructure includes 36 web servers and nine database servers. Therefore, this load consumes minimal amount of the web server infrastructure (0.2%) and approximately 2% of the database infrastructure. The test load represents approximately 19.2% (48,000) of the maximum concurrent users (250,000). Assuming linear scaling of the NCTest infrastructure, it is conceivable that the NCTest application and infrastructure will support at least 250,000 concurrent users under normal circumstances during the question phase of the test while only consuming 10.4% of the database server infrastructure and 1.04% of the web server infrastructure.

Cost Feasibility of Test Methodology

To fully automate all 250,000 users using the empirical test duration data in Appendix A, 10 times the amount of hardware is required - 120 IBM Bladecenter HS22 blades with two CPUs and 48GB of RAM in each. This hardware costs over $750,000 including university discount. Given the hardware costs, extrapolating capacity with a reduced user set size running at an accelerated rate while targeting a subset of the full infrastructure is a cost effective method for determining user capacity.

Appendix A - Time On Test Histogram Data

Time Bin (Minutes)	Frequency	%	Cum Freq	Cum %
30-35	4902	6.79	4902	6.79
35-40	5991	8.30	10893	15.09
40-45	6332	8.77	17225	23.86
45-50	5858	8.12	23083	31.98
50-55	4391	6.08	27474	38.06
55-60	5583	7.73	33057	45.79
60-65	4981	6.90	38038	52.69
65-70	4767	6.60	42805	59.30
70-75	4186	5.80	46991	65.10
75-80	3800	5.26	50791	70.36
80-85	3217	4.46	54008	74.82
85-90	2822	3.91	56830	78.73
90-95	2491	3.45	59321	82.18
95-100	2217	3.07	61538	85.25
100-105	1654	2.29	63192	87.54
105-110	1418	1.96	64610	89.50
110-115	1352	1.87	65962	91.38
115-120	1098	1.52	67060	92.90
120-125	995	1.38	68055	94.28
125-130	846	1.17	68901	95.45
130-135	661	0.92	69562	96.36
135-140	534	0.74	70096	97.10
140-145	474	0.66	70570	97.76
145-150	416	0.58	70986	98.34
150-155	367	0.51	71353	98.84
155-160	267	0.37	71620	99.21
160-165	200	0.28	71820	99.49
165-170	158	0.22	71978	99.71
170-175	112	0.16	72090	99.87
175-180	97	0.13	72187	100.00

Appendix B - Test Code

'Fabric' was used to drive processes on all the client machines from an additional machine. All Fabric interactions were coded in fabfile.py. A 'run many' script was called to generate calls to multiple 'nctest' scripts. Python was used for all code and is listed below.

fabfile.py

```
from __future__ import division
import os.path
import time
from copy import copy

from fabric import api
from fabric.api import env,parallel
from fabric.context_managers import hide, show

env.hosts = [
    'nctest1.oscar.ncsu.edu',
    'nctest2.oscar.ncsu.edu',
    'nctest3.oscar.ncsu.edu',
    'nctest4.oscar.ncsu.edu',
    'nctest5.oscar.ncsu.edu',
    'nctest6.oscar.ncsu.edu',
    'nctest7.oscar.ncsu.edu',
    'nctest8.oscar.ncsu.edu',
    'nctest9.oscar.ncsu.edu',
    'nctest10.oscar.ncsu.edu',
    'nctest11.oscar.ncsu.edu',
    'nctest12.oscar.ncsu.edu',
]
env.code_dir = "/tmp/nctest"
env.start_test = 1  # Indexed starting at one
env.workers_per_server = 200
env.xserver_start_port = 20  # First port to start xservers on
env.worker_chunk_size = 20  # Number of workers per xserver
env.start_time = 25 * env.worker_chunk_size # Seconds from now to start the test in each worker
env.accelerator = 0 # divisor for time between tests (0 for no wait time)

FILES_TO_COPY = [
    'run_many.py',
    'nctest.py',
    'watchcmd.sh',
    'chromedriver',
    'firefox_profile/',
    'google-chrome.repo',
]

YUM_DEPENDENCIES = [
    "Xvfb",
    "screen",
    "firefox",

    # Fonts required by firefox, when no other x11 stuff is installed
```

```
    "xorg-x11-fonts-Type1",
]

@api.task
@parallel
def install_dependencies():
    api.sudo("yum install -q -y %s" % ' '.join(YUM_DEPENDENCIES))
    api.sudo("easy_install pip")
    api.sudo("pip install selenium")

@api.task
@parallel
def copy_code():
    """
    Copies code to hosts in preparation to run a test.
    """
    with api.settings(warn_only=True):
        api.run("mkdir %s" % env.code_dir)
    for filename in FILES_TO_COPY:
        api.put(filename, env.code_dir)
    api.sudo("mv " + env.code_dir + "/google-chrome.repo /etc/yum.repos.d/")
    api.sudo("yum install -q -y google-chrome-stable")
    api.run("chmod u+x /tmp/nctest/chromedriver")

@api.task
@parallel
def clean_tmp():
    """
    Cleans out the /tmp directory.
    """
    with api.settings(warn_only=True):
        api.run("rm -rf %s" % env.code_dir)
        api.run("rm -rf /tmp/tmp*")
        api.run("rm -rf /tmp/xvfb-run*")

@api.task
@parallel
def run():

    """
    with api.settings(warn_only=True):
        api.run("killall Xvfb firefox")
    api.run('screen -d -m -S xvfb Xvfb :99')
    time.sleep(4)
    """

    start_test = env.start_test + env.hosts.index(env.host_string) * env.workers_per_server
    xserver_port = env.xserver_start_port

    for i in xrange(0, env.workers_per_server, env.worker_chunk_size):
        api.local('ssh {server} "cd {code_dir}; xvfb-run -n {xserver_port} python {script}
{start_test} {num_tests} {start_time} {accelerator}" &'.format(
            server = env.host_string,
            xserver_port = xserver_port,
            code_dir = env.code_dir,
            script = os.path.join(env.code_dir, 'run_many.py'),
            start_test = start_test + i,
            num_tests = min(env.workers_per_server-i, env.worker_chunk_size),
            start_time = env.start_time,
            accelerator = env.accelerator,
```

```
        ))
        xserver_port += 1

    """
    with api.cd(env.code_dir):
        api.run('DISPLAY=:99 screen -d -m -S run_many xvfb-run {script} {start_test} {num_tests}
{start_time}'.format(
            script = os.path.join(env.code_dir, 'run_many.py'),
            start_test = start_test,
            num_tests = env.workers_per_server,
            start_time = env.start_time,
        ))
    """

@api.task
def collect_logs():
    """
    """
    pass

@api.task
def monitor():
    """
    Monitor summary
    """

    with hide('running', 'stderr', 'status', 'user','aborts'):

        print 'Waiting to start: ',
        api.run("grep -i 'Waiting at initial time point' /tmp/nctest/log/* | wc -l")

        print 'Started tests: ',
        api.run("grep -i 'Question 1$' /tmp/nctest/log/* | wc -l")

        print 'Completed tests: ',
        api.run("grep -i 'Test Complete' /tmp/nctest/log/* | wc -l")

        print 'Exited tests: ',
        api.run("grep -i 'Exiting$' /tmp/nctest/log/* | wc -l")

@api.task
def monitor_ram():
    """
    Monitor ram
    """

    with hide('running', 'stderr', 'status', 'user','aborts'):

        print 'RAM usage: ',
        api.run("free -go")

@api.task
def monitor_questions():
    """
    Monitor progress through test questions
    """

    with hide('running', 'stderr', 'status', 'user','aborts'):

        print 'Question 1: ',
```

```
api.run("grep -i 'Question 1$' /tmp/nctest/log/* | wc -l")
print 'Question 2: ',
api.run("grep -i 'Question 2$' /tmp/nctest/log/* | wc -l")
print 'Question 3: ',
api.run("grep -i 'Question 3$' /tmp/nctest/log/* | wc -l")
print 'Question 4: ',
api.run("grep -i 'Question 4$' /tmp/nctest/log/* | wc -l")
print 'Question 5: ',
api.run("grep -i 'Question 5$' /tmp/nctest/log/* | wc -l")
print 'Question 6: ',
api.run("grep -i 'Question 6$' /tmp/nctest/log/* | wc -l")
print 'Question 7: ',
api.run("grep -i 'Question 7$' /tmp/nctest/log/* | wc -l")
print 'Question 8: ',
api.run("grep -i 'Question 8$' /tmp/nctest/log/* | wc -l")
print 'Question 9: ',
api.run("grep -i 'Question 9$' /tmp/nctest/log/* | wc -l")
print 'Question 10: ',
api.run("grep -i 'Question 10$' /tmp/nctest/log/* | wc -l")
print 'Question 11: ',
api.run("grep -i 'Question 11$' /tmp/nctest/log/* | wc -l")
print 'Question 12: ',
api.run("grep -i 'Question 12$' /tmp/nctest/log/* | wc -l")
print 'Question 13: ',
api.run("grep -i 'Question 13$' /tmp/nctest/log/* | wc -l")
print 'Question 14: ',
api.run("grep -i 'Question 14$' /tmp/nctest/log/* | wc -l")
print 'Question 15: ',
api.run("grep -i 'Question 15$' /tmp/nctest/log/* | wc -l")
print 'Question 16: ',
api.run("grep -i 'Question 16$' /tmp/nctest/log/* | wc -l")
print 'Question 17: ',
api.run("grep -i 'Question 17$' /tmp/nctest/log/* | wc -l")
print 'Question 18: ',
api.run("grep -i 'Question 18$' /tmp/nctest/log/* | wc -l")
print 'Question 19: ',
api.run("grep -i 'Question 19$' /tmp/nctest/log/* | wc -l")
print 'Question 20: ',
api.run("grep -i 'Question 20$' /tmp/nctest/log/* | wc -l")
print 'Question 21: ',
api.run("grep -i 'Question 21$' /tmp/nctest/log/* | wc -l")
print 'Question 22: ',
api.run("grep -i 'Question 22$' /tmp/nctest/log/* | wc -l")
print 'Question 23: ',
api.run("grep -i 'Question 23$' /tmp/nctest/log/* | wc -l")
print 'Question 24: ',
api.run("grep -i 'Question 24$' /tmp/nctest/log/* | wc -l")
print 'Question 25: ',
api.run("grep -i 'Question 25$' /tmp/nctest/log/* | wc -l")
print 'Question 26: ',
api.run("grep -i 'Question 26$' /tmp/nctest/log/* | wc -l")
print 'Question 27: ',
api.run("grep -i 'Question 27$' /tmp/nctest/log/* | wc -l")
print 'Question 28: ',
api.run("grep -i 'Question 28$' /tmp/nctest/log/* | wc -l")
print 'Question 29: ',
api.run("grep -i 'Question 29$' /tmp/nctest/log/* | wc -l")
print 'Question 30: ',
api.run("grep -i 'Question 30$' /tmp/nctest/log/* | wc -l")
print 'Question 31: ',
```

```
        api.run("grep -i 'Question 31$' /tmp/nctest/log/* | wc -l")
        print 'Question 32: ',
        api.run("grep -i 'Question 32$' /tmp/nctest/log/* | wc -l")
        print 'Question 33: ',
        api.run("grep -i 'Question 33$' /tmp/nctest/log/* | wc -l")
        print 'Question 34: ',
        api.run("grep -i 'Question 34$' /tmp/nctest/log/* | wc -l")
        print 'Question 35: ',
        api.run("grep -i 'Question 35$' /tmp/nctest/log/* | wc -l")
        print 'Question 36: ',
        api.run("grep -i 'Question 36$' /tmp/nctest/log/* | wc -l")
        print 'Question 37: ',
        api.run("grep -i 'Question 37$' /tmp/nctest/log/* | wc -l")
        print 'Question 38: ',
        api.run("grep -i 'Question 38$' /tmp/nctest/log/* | wc -l")
        print 'Question 39: ',
        api.run("grep -i 'Question 39$' /tmp/nctest/log/* | wc -l")
        print 'Question 40: ',
        api.run("grep -i 'Question 40$' /tmp/nctest/log/* | wc -l")
        print 'Question 41: ',
        api.run("grep -i 'Question 41$' /tmp/nctest/log/* | wc -l")
        print 'Question 42: ',
        api.run("grep -i 'Question 42$' /tmp/nctest/log/* | wc -l")
        print 'Question 43: ',
        api.run("grep -i 'Question 43$' /tmp/nctest/log/* | wc -l")
        print 'Question 44: ',
        api.run("grep -i 'Question 44$' /tmp/nctest/log/* | wc -l")
        print 'Question 45: ',
        api.run("grep -i 'Question 45$' /tmp/nctest/log/* | wc -l")
        print 'Question 46: ',
        api.run("grep -i 'Question 46$' /tmp/nctest/log/* | wc -l")
        print 'Question 47: ',
        api.run("grep -i 'Question 47$' /tmp/nctest/log/* | wc -l")
        print 'Question 48: ',
        api.run("grep -i 'Question 48$' /tmp/nctest/log/* | wc -l")
        print 'Question 49: ',
        api.run("grep -i 'Question 49$' /tmp/nctest/log/* | wc -l")
        print 'Question 50: ',
        api.run("grep -i 'Question 50$' /tmp/nctest/log/* | wc -l")
        print 'Question 51: ',
        api.run("grep -i 'Question 51$' /tmp/nctest/log/* | wc -l")
        print 'Question 52: ',
        api.run("grep -i 'Question 52$' /tmp/nctest/log/* | wc -l")
        print 'Question 53: ',
        api.run("grep -i 'Question 53$' /tmp/nctest/log/* | wc -l")
        print 'Question 54: ',
        api.run("grep -i 'Question 54$' /tmp/nctest/log/* | wc -l")
        print 'Question 55: ',
        api.run("grep -i 'Question 55$' /tmp/nctest/log/* | wc -l")
        print 'Question 56: ',
        api.run("grep -i 'Question 56$' /tmp/nctest/log/* | wc -l")
        print 'Question 57: ',
        api.run("grep -i 'Question 57$' /tmp/nctest/log/* | wc -l")
        print 'Complete: ',
        api.run("grep -i 'Test Complete' /tmp/nctest/log/* | wc -l")
        print 'Exited: ',
        api.run("grep -i 'Exiting$' /tmp/nctest/log/* | wc -l")

@api.task
def copy_logs():
```

```
    with api.settings(warn_only=True):
        api.run("mkdir remote_logs")
    api.get("/tmp/nctest/log/*", "remote_logs/%(path)s")

@api.task
@parallel
def kill():
    """
    Kills all firefox instances
    """
    with api.settings(warn_only=True):
        api.run("killall firefox")
        api.run("killall Xvfb")
        api.run("killall python")
```

run_many.py

```
import sys
import os.path
import random
from subprocess import Popen
from time import sleep, time

if __name__ == "__main__":

    if len(sys.argv) == 5:
        start_time = time() + float(sys.argv[3])
    else:
        print "Usage: %s <start test number 1-10000> <number of tests to run> <start time in
seconds from now> <accelerator>" % sys.argv[0]
        sys.exit(-1)

    nctest_script = os.path.abspath(os.path.join(os.path.abspath(__file__), "..", "nctest.py"))

    start_test_num = int(sys.argv[1])
    nprocesses = int(sys.argv[2])
    accelerator = int(sys.argv[4])

    processes = []
    try:
        for i in xrange(nprocesses):

            arguments = ['python', nctest_script, str(start_test_num+i), str(start_time +
(random.random() * 20)), str(accelerator)]
            process = Popen(arguments)
            processes.append(process)
            sleep(20)

        # Start processes and wait
        for process in processes:
            process.wait()

    # Always kill all processes when finished
    finally:
        for process in processes:
            try:
                if process.poll() is None:
                    process.kill()
            except Exception as e:
```

```
            print
            print "Error in killing:"
            print e
```

nctest.py

```
import time, time, sys, os, random
import logging as log
import os

from selenium import webdriver
from selenium.webdriver.support.ui import Select
from selenium.common.exceptions import NoSuchElementException, StaleElementReferenceException,
WebDriverException
from selenium.webdriver.firefox.firefox_binary import FirefoxBinary
from selenium.webdriver.firefox.firefox_profile import FirefoxProfile
from selenium.webdriver.common import utils

class NctestSelenium(object):
def __init__(self, username, password, testnum, accelerator, start_time=None, use_xvfb=False):
        self.username = username
        self.password = password
        self.testnum = testnum
        self.use_xvfb = use_xvfb
        self.accelerator = accelerator

        if start_time:
            self.target_time = start_time
        else:
            self.target_time = time.time()

        # find amount of time between tests
        if self.accelerator == 0:
            self.time_point_delay = 0
        else:
            # get the delay between questions based on CUACS test duration
            # frequency distribution
            # (57 is the number of questions)
            self.delay_index = random.random()
            if (self.delay_index >= 0) and (self.delay_index < 0.0679):
                self.time_point_delay = (30*60 + 35*60)/(2*57)
            elif (self.delay_index >= 0.0679) and (self.delay_index < 0.1509):
                self.time_point_delay = (35*60 + 40*60)/(2*57)
            elif (self.delay_index >= 0.1509) and (self.delay_index < 0.2386):
                self.time_point_delay = (40*60 + 45*60)/(2*57)
            elif (self.delay_index >= 0.2386) and (self.delay_index < 0.3198):
                self.time_point_delay = (45*60 + 50*60)/(2*57)
            elif (self.delay_index >= 0.3198) and (self.delay_index < 0.3806):
                self.time_point_delay = (50*60 + 55*60)/(2*57)
            elif (self.delay_index >= 0.3806) and (self.delay_index < 0.4579):
                self.time_point_delay = (55*60 + 60*60)/(2*57)
            elif (self.delay_index >= 0.4579) and (self.delay_index < 0.5269):
                self.time_point_delay = (60*60 + 65*60)/(2*57)
            elif (self.delay_index >= 0.5269) and (self.delay_index < 0.5930):
                self.time_point_delay = (65*60 + 70*60)/(2*57)
            elif (self.delay_index >= 0.5930) and (self.delay_index < 0.6510):
```

```
                self.time_point_delay = (70*60 + 75*60)/(2*57)
            elif (self.delay_index >= 0.6510) and (self.delay_index < 0.7036):
                self.time_point_delay = (75*60 + 80*60)/(2*57)
            elif (self.delay_index >= 0.7036) and (self.delay_index < 0.7482):
                self.time_point_delay = (80*60 + 85*60)/(2*57)
            elif (self.delay_index >= 0.7482) and (self.delay_index < 0.7873):
                self.time_point_delay = (85*60 + 90*60)/(2*57)
            elif (self.delay_index >= 0.7873) and (self.delay_index < 0.8218):
                self.time_point_delay = (90*60 + 95*60)/(2*57)
            elif (self.delay_index >= 0.8218) and (self.delay_index < 0.8525):
                self.time_point_delay = (95*60 + 100*60)/(2*57)
            elif (self.delay_index >= 0.8525) and (self.delay_index < 0.8754):
                self.time_point_delay = (100*60 + 105*60)/(2*57)
            elif (self.delay_index >= 0.8754) and (self.delay_index < 0.8950):
                self.time_point_delay = (105*60 + 110*60)/(2*57)
            elif (self.delay_index >= 0.8950) and (self.delay_index < 0.9138):
                self.time_point_delay = (110*60 + 115*60)/(2*57)
            elif (self.delay_index >= 0.9138) and (self.delay_index < 0.9290):
                self.time_point_delay = (115*60 + 120*60)/(2*57)
            elif (self.delay_index >= 0.9290) and (self.delay_index < 0.9428):
                self.time_point_delay = (120*60 + 125*60)/(2*57)
            elif (self.delay_index >= 0.9428) and (self.delay_index < 0.9545):
                self.time_point_delay = (125*60 + 130*60)/(2*57)
            elif (self.delay_index >= 0.9545) and (self.delay_index < 0.9636):
                self.time_point_delay = (130*60 + 135*60)/(2*57)
            elif (self.delay_index >= 0.9636) and (self.delay_index < 0.9710):
                self.time_point_delay = (135*60 + 140*60)/(2*57)
            elif (self.delay_index >= 0.9710) and (self.delay_index < 0.9776):
                self.time_point_delay = (140*60 + 145*60)/(2*57)
            elif (self.delay_index >= 0.9776) and (self.delay_index < 0.9834):
                self.time_point_delay = (145*60 + 150*60)/(2*57)
            elif (self.delay_index >= 0.9834) and (self.delay_index < 0.9884):
                self.time_point_delay = (150*60 + 155*60)/(2*57)
            elif (self.delay_index >= 0.9884) and (self.delay_index < 0.9921):
                self.time_point_delay = (155*60 + 160*60)/(2*57)
            elif (self.delay_index >= 0.9921) and (self.delay_index < 0.9949):
                self.time_point_delay = (160*60 + 165*60)/(2*57)
            elif (self.delay_index >= 0.9949) and (self.delay_index < 0.9971):
                self.time_point_delay = (165*60 + 170*60)/(2*57)
            elif (self.delay_index >= 0.9971) and (self.delay_index < 0.9987):
                self.time_point_delay = (170*60 + 175*60)/(2*57)
            elif (self.delay_index >= 0.9987) and (self.delay_index < 1.0000):
                self.time_point_delay = (175*60 + 180*60)/(2*57)

            self.time_point_delay = self.time_point_delay / float(self.accelerator)

        self.log("self.target_time = "+str(self.target_time))
        self.log("self.time_point_delay = "+str(self.time_point_delay))

        if self.use_xvfb:
            from pyvirtualdisplay import Display
            self.display = Display(visible=0, size=(800, 600))
            self.display.start()

        self.log("Loading Browser")
        self.driver = self.load_driver()
        self.base_url = "https://data.ncsu.edu"
        #self.base_url = "https://152.1.168.115:8080"

    def load_driver(self):
```

```
        profile = FirefoxProfile("firefox_profile/")
        binary = NctestFirefoxBinary(wait_time=300)
        driver = webdriver.Firefox(profile, firefox_binary=binary)
        self.log("Firefox PID: %s" % binary.process.pid)

        """
        driver = webdriver.Chrome(executable_path="./chromedriver")
        """

        """
        driver = webdriver.Opera(executable_path="../bin/selenium-server-standalone-2.25.0.jar")
        """

        driver.implicitly_wait(60) # Amount of time to wait for an
                                   # element to be found
        return driver

    def load_page(self, page):

        # Use javascript instead of driver.get to load the page
        # Doing it this way fixes an odd case that prevents the "Start Test"
        # button from being pressed.
        self.driver.execute_script('window.location.href = "%s"' % page)
        #self.driver.get(page)

    def log(self, *args):
        #print self.username, self.password, self.testnum, '::', ' '.join(args)
        log.debug(' '.join(args))

    def driver_command(self, xpath, command, *args):
        '''
        Runs driver.find_element_by_xpath(xpath).<command>(*args), with added
        error and retry support.

        Specially supports the select_by_visible_text by wrapping
        find_element_by_xpath with Select().
        '''
        MAX_RETRIES = 10
        SLEEP_TIME = 1

        #self.log(command, xpath)

        retry_count = 0
        while True:

            try:
                if command == "select_by_visible_text":
                    element = Select(self.driver.find_element_by_xpath(xpath))
                else:
                    element = self.driver.find_element_by_xpath(xpath)
                return getattr(element, command)(*args)

            #except StaleElementReferenceException, WebDriverException:
            except Exception as e:
                #self.log(str(e))
                if retry_count >= MAX_RETRIES:
                    raise
                self.log("Error running command!  Retrying in 1 second.")
                retry_count += 1
```

```
                time.sleep(SLEEP_TIME)

    def time_point(self):
        self.target_time += self.time_point_delay
        t = time.time()
        if t < self.target_time:
            delay = self.target_time - t
            self.log("Waiting for %s s" % delay)
            time.sleep(delay)

    def run(self):

        self.log("Loading Initial Page")
        self.load_page("%s/nctest/NCTest.html#testAdminLogin" % self.base_url)

        # Intentional wait to put time between firefox starting and test
        # starting.  This will hopefully reduce errors.
        time.sleep(10)

        # Click "Caution: You're using Linux"
        self.driver_command("//div/div/div/div/div/div/div/div/table/tbody/tr/td/div", "click")

        # Login Page
        self.driver_command("//select", "select_by_visible_text", "Science Grade 5 Field Test")
        self.driver_command("(//input[@type='text'])[6]", "clear")
        self.driver_command("(//input[@type='text'])[6]", "send_keys", self.username)
        self.driver_command("//input[@type='password']", "clear")
        self.driver_command("//input[@type='password']", "send_keys", self.password)
        self.driver_command("//tr[8]/td/table/tbody/tr/td[2]/div", "click")

        # Select Test Page
        self.driver_command("//select", "select_by_visible_text", "student, test (%s)" %
self.testnum)
        self.driver_command("//fieldset/div/table/tbody/tr/td[2]/div", "click")

        # This page has a loading dialog that says:
        #
        #        Please Wait.
        #    The test is loading...
        #
        # If we try to push the start test button before this goes away, it
        # will fail silently.  Instead we wait until the dialog is gone by
        # searching for the dialog's text once a second.
        self.log("Waiting for loading dialog to dissapear")
        while True:
            html_element = self.driver.find_element_by_tag_name("html")
            page_text = html_element.text
            if page_text.find("Please Wait") == -1:
                self.log("Dialog is gone!")
                break
            else:
                self.log("Dialog is present, waiting...")
                time.sleep(1)

        # Start Test Button
        self.log("Pressing Start Button")
        self.driver_command("//div[4]/div/div/div/div/div/div", "click")

        # Tutorial Pages
        self.log("Starting Tutorial")
```

```
try:
    self.driver_command("(//button[@type='button'])[2]", "click")
    #self.driver_command("//div[4]/div/div/div/div/div/div", "click")
except NoSuchElementException:
    # Try clicking the "Start Test" button again.  If it didn't work
    # the first time, then we're on the previous page, which is why we
    # couldn't find this button.
    self.driver_command("(//button[@type='button'])[2]", "click")
    #self.driver_command("//div[4]/div/div/div/div/div/div", "click")

    self.driver_command("(//button[@type='button'])[2]", "click")

self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")
self.driver_command("(//button[@type='button'])[2]", "click")

# Button "Close Tutorial and Begin Assesment"
self.log("Waiting at initial time point!")
self.time_point()
self.driver_command("(//button[@type='button'])[3]", "click")

# Question 1
self.log("Question 1")
self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
self.driver_command("//td[5]/div", "click")
self.time_point()

# Question 2
self.log("Question 2")
self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
self.driver_command("//td[5]/div", "click")
self.time_point()

# Question 3
self.log("Question 3")
self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
self.driver_command("//td[5]/div", "click")
self.time_point()

# Question 4
```

```
        self.log("Question 4")
        self.driver_command("//tr[3]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 5
        self.log("Question 5")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 6
        self.log("Question 6")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 7
        self.log("Question 7")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 8
        # Drag and drop style.  We just hit next.
        self.log("Question 8")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 9
        self.log("Question 9")
        self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 10
        self.log("Question 10")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 11
        self.log("Question 11")
        self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 12
        self.log("Question 12")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 13
        # Select style question
        self.log("Question 13")
        self.driver_command("//div[6]/div/div/div/div[3]", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()
```

```
        # Question 14
        self.log("Question 14")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 15
        self.log("Question 15")
        self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 16
        self.log("Question 16")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 17
        self.log("Question 17")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 18
        self.log("Question 18")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 19
        self.log("Question 19")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 20
        # Select style question
        self.log("Question 20")

        # This used to work for this style select question, but doesn't anymore
        #self.driver_command("//tr[6]/td/table/tbody/tr/td/div", "click")
        #self.driver_command("//tr[6]/td/table/tbody/tr/td/div/div", "click")
        #self.driver_command("//td[5]/div", "click")

        self.driver_command("//div[2]/div/div/div/table/tbody/tr[2]/td/table/tbody/tr/td/
div", "click")
        self.driver_command("//td/table/tbody/tr/td/div/div", "click")
        self.driver_command("//td[5]/div", "click")

        self.time_point()

        # Question 21
        self.log("Question 21")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 22
        self.log("Question 22")
```

```
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 23
        self.log("Question 23")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 24
        self.log("Question 24")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 25
        # Input style question
        self.log("Question 25")
        self.driver_command("(//input[@type='text'])[8]", "clear")
        self.driver_command("(//input[@type='text'])[8]", "send_keys", "3141592")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 26
        self.log("Question 26")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 27
        self.log("Question 27")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 28
        self.log("Question 28")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 29
        self.log("Question 29")
        self.driver_command("//tr[3]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 30
        self.log("Question 30")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 31
        self.log("Question 31")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()
```

```
        # Question 32
        self.log("Question 32")
        self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 33
        # Select style question
        self.log("Question 33")
        self.driver_command("//div[6]/div/div/div/div[3]", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 34
        self.log("Question 34")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 35
        self.log("Question 35")
        self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 36
        self.log("Question 36")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 37
        self.log("Question 37")
        self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 38
        self.log("Question 38")
        self.driver_command("//tr[3]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 39
        # Drag and drop style.  We just hit next.
        self.log("Question 39")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 40
        self.log("Question 40")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 41
        self.log("Question 41")
        self.driver_command("//tr[3]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()
```

```
        # Question 42
        self.log("Question 42")
        self.driver_command("//tr[5]/td/table/tbody/tr", "click")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 43
        self.log("Question 43")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 44
        self.log("Question 44")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 45
        self.log("Question 45")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 46
        self.log("Question 46")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 47
        self.log("Question 47")
        self.driver_command("//tr[3]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 48
        # Select style question
        self.log("Question 48")
        self.driver_command("//div[2]/div/div/div/table/tbody/tr[2]/td/table/tbody/tr/td/
div", "click")
        self.driver_command("//td/table/tbody/tr/td/div/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 49
        self.log("Question 49")
        self.driver_command("//tr[3]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 50
        self.log("Question 50")
        self.driver_command("//td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 51
```

```
        self.log("Question 51")
        self.driver_command("//tr[3]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 52
        self.log("Question 52")
        self.driver_command("//tr[5]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 53
        self.log("Question 53")
        self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 54
        self.log("Question 54")
        self.driver_command("//tr[3]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 55
        self.log("Question 55")
        self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 56
        self.log("Question 56")
        self.driver_command("//tr[4]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # Question 57
        self.log("Question 57")
        self.driver_command("//tr[3]/td/table/tbody/tr/td[4]/div", "click")
        self.driver_command("//td[5]/div", "click")
        self.time_point()

        # End Test Button
        #import ipdb; ipdb.set_trace() #XXX
        #self.driver_command("//div/div/div[2]/div/div/table/tbody/tr/td/div", "click")

        # Confirm End Test
        #self.driver_command("//div[2]/div/table/tbody/tr/td[2]/div", "click")

        self.log("Test Complete!")

    def is_element_present(self, how, what):
        try: self.driver.find_element(by=how, value=what)
        except NoSuchElementException, e: return False
        return True

    def finish(self):
        self.driver.quit()
        if self.use_xvfb:
            self.display.stop()
```

```
class NctestWebkit(NctestSelenium):

    def load_driver(self):
        import webkit
        driver = webkit.WebkitBrowser(gui=True)
        return driver

    def load_page(self, page):
        print "GETTING", page
        self.driver.get(page)

    def driver_command(self, xpath, command, *args):
        self.log(command, xpath)

        if command == "send_keys" or command == "select_by_visible_text":
            self.driver.fill(xpath, args[0])

        elif command == "clear":
            self.driver.fill(xpath, "")

        else:
            getattr(self.driver, command)(xpath)

        self.driver.screenshot("screenshot.jpg")


class NctestFirefoxBinary(FirefoxBinary):
    '''Firefox binary that waits longer for firefox to start.

    The wait isn't long enough if you see this error:

        selenium.common.exceptions.WebDriverException: Message:
        'Can\'t load the profile. Profile Dir: /tmp/tmpDdPGNO
        Firefox output: Xlib:  extension "RANDR" missing on display
        ":99".\n*** LOG addons.xpi: startup\n*** LOG addons.xpi:
        checkForChanges\n*** LOG addons.xpi: No changes found\n'

    This binary also assumes the profile exists and doesn't need to be created.

    '''

    def __init__(self, *args, **kwargs):
        self.wait_time = kwargs.pop("wait_time", 30)
        super(NctestFirefoxBinary, self).__init__(*args, **kwargs)

    def _wait_until_connectable(self):
        '''Blocks until the extension is connectable in the firefox.

        Same as Firefox._wait_until_connectable, but waits self.wait_time
        instead of 30 seconds.

        '''
        count = 0
        while not utils.is_connectable(self.profile.port):
            if self.process.poll() is not None:
                # Browser has exited
                raise WebDriverException("The browser appears to have exited "
                      "before we could connect. The output was: %s" %
                      self._get_firefox_output())
```

```
            if count == self.wait_time:  # Change was made here
                self.kill()
                raise WebDriverException("Can't load the profile. Profile "
                      "Dir: %s Firefox output: %s" % (
                          self.profile.path, self._get_firefox_output()))
            count += 1
            time.sleep(1)
        return True

    def _extract_and_check(self, profile, no_focus_so_name, x86, amd64):

        paths = [x86, amd64]
        built_path = ""
        for path in paths:
            library_path = os.path.join(profile.path, path)
            # This commented out so it just uses the existing profile
            #os.makedirs(library_path)
            #import shutil
            #shutil.copy(os.path.join(os.path.dirname(__file__), path,
            #  self.NO_FOCUS_LIBRARY_NAME),
            #  library_path)
            built_path += library_path + ":"

        return built_path


def credentials(userNum):
    """
    Takes an integer from 0-9999, and returns a namedtuple of username,
    password, testnum.
    """

    if (userNum > 0) and (userNum <= 10000):

#        userids = { 1:294,   2:319,   3:349,   4:397,   5:455,   6:587,   7:701,   8:704,
9:833,  10:877,
#                   11:894,  12:934,  13:950,  14:961,  15:1009, 16:1011, 17:1015, 18:1020,
19:1042, 20:1057,
#                   21:1110, 22:1293, 23:1320, 24:1350, 25:1454, 26:1463, 27:1478, 28:1500,
29:1508, 30:1550,
#                   31:1557, 32:1588, 33:1615, 34:1625, 35:1626, 36:1653, 37:1690, 38:1718,
39:1742, 40:1757,
#                   41:1834, 42:1861, 43:1879, 44:1885, 45:1919, 46:1922, 47:1965, 48:1972,
49:1975, 50:1998,
#                   51:2007, 52:2009, 53:2022, 54:2039, 55:2045, 56:2048, 57:2059, 58:2062,
59:2078, 60:2085,
#                   61:2087, 62:2088, 63:2102, 64:2112, 65:2121, 66:2129, 67:2151, 68:2152,
69:2155, 70:2157,
#                   71:2159, 72:2162, 73:2165, 74:2169, 75:2179, 76:2180, 77:2181, 78:2182,
79:2197, 80:2202,
#                   81:2209, 82:2222, 83:2225, 84:2226, 85:2227, 86:2228, 87:2230, 88:2242,
89:2244, 90:2250,
#                   91:2257, 92:2258, 93:2259, 94:2261, 95:2265, 96:2266, 97:2272, 98:2274,
99:2275, 100:2284}
#
#        userids = {  1:294,    2:319,    3:349,    4:397,    5:455,    6:587,    7:701,
8:704,    9:833,   10:877,
#                    11:894,   12:934,   13:950,   14:961,   15:1009,  16:1011,  17:1015,
18:1020,  19:1042,  20:1057,
```

```
#                    21:1110,  22:1293,  23:1320,  24:1350,  25:1454,  26:2285,  27:1478,
28:1500,  29:1508,  30:1550,
#                    31:1557,  32:1588,  33:1615,  34:1625,  35:1626,  36:1653,  37:1690,
38:1718,  39:1742,  40:1757,
#                    41:1834,  42:1861,  43:1879,  44:1885,  45:1919,  46:1922,  47:1965,
48:1972,  49:1975,  50:1998,
#                    51:2007,  52:2009,  53:2022,  54:2039,  55:2045,  56:2048,  57:2059,
58:2062,  59:2078,  60:2085,
#                    61:2087,  62:2088,  63:2102,  64:2112,  65:2121,  66:2129,  67:2151,
68:2152,  69:2155,  70:2157,
#                    71:2159,  72:2162,  73:2165,  74:2169,  75:2179,  76:2180,  77:2181,
78:2182,  79:2197,  80:2202,
#                    81:2209,  82:2222,  83:2225,  84:2226,  85:2227,  86:2228,  87:2230,
88:2242,  89:2244,  90:2250,
#                    91:2257,  92:2258,  93:2259,  94:2261,  95:2265,  96:2266,  97:2272,
98:2274,  99:2275, 100:2284,
#                   101:2308, 102:2313, 103:2331, 104:2338, 105:2352}

        userids = {  1:5,      2:49,     3:72,     4:78,     5:95,     6:96,     7:117,    8:135,
9:138,   10:159,
                    11:178,   12:181,   13:183,   14:190,   15:192,   16:198,   17:199,   18:205,
19:208,   20:213,
                    21:218,   22:244,   23:246,   24:255,   25:269,   26:273,   27:288,   28:316,
29:320,   30:325,
                    31:329,   32:340,   33:355,   34:358,   35:362,   36:369,   37:390,   38:397,
39:399,   40:449,
                    41:467,   42:474,   43:488,   44:499,   45:502,   46:520,   47:529,   48:533,
49:536,   50:541,
                    51:545,   52:551,   53:563,   54:572,   55:616,   56:618,   57:632,   58:648,
59:652,   60:672,
                    61:679,   62:707,   63:726,   64:730,   65:742,   66:746,   67:753,   68:760,
69:765,   70:775,
                    71:778,   72:791,   73:794,   74:811,   75:820,   76:823,   77:840,   78:850,
79:865,   80:878,
                    81:884,   82:885,   83:901,   84:906,   85:911,   86:919,   87:924,   88:938,
89:947,   90:956,
                    91:968,   92:980,   93:1038,  94:1045,  95:1077,  96:1097,  97:1116,
98:1129,  99:1140, 100:1144,
                   101:1147, 102:1155, 103:1161, 104:1169, 105:1177, 106:1184, 107:1189,
108:1193, 109:1195, 110:1198,
                   111:1212, 112:1213, 113:1219, 114:1220, 115:1225, 116:1228, 117:1321,
118:1327, 119:1336, 120:1351,
                   121:1360, 122:1364, 123:1387, 124:1394, 125:1405, 126:1406, 127:1416,
128:1454, 129:1468, 130:1474,
                   131:1475, 132:1487, 133:1498, 134:1504, 135:1549, 136:1566, 137:1575,
138:1602, 139:1604, 140:1606,
                   141:1612, 142:1619, 143:1630, 144:1648, 145:1651, 146:1683, 147:1688,
148:1690, 149:1695, 150:1698,
                   151:1712, 152:1713, 153:1722, 154:1723, 155:1729, 156:1731, 157:1744,
158:1746, 159:1757, 160:1760,
                   161:1762, 162:1768, 163:1772, 164:1782, 165:1784, 166:1788, 167:1791,
168:1793, 169:1797, 170:1805,
                   171:1807, 172:1824, 173:1828, 174:1847, 175:1853, 176:1856, 177:1871,
178:1896, 179:1903, 180:1937,
                   181:1969, 182:1978, 183:1980, 184:1984, 185:1987, 186:2001, 187:2010,
188:2021, 189:2022, 190:2028,
                   191:2029, 192:2039, 193:2043, 194:2044, 195:2048, 196:2075, 197:2106,
198:2108, 199:2112, 200:2116,
                   201:2117, 202:2123, 203:2149, 204:2157, 205:2170, 206:2183, 207:2198,
208:2216, 209:2224, 210:2228,
```

```
                    211:2229, 212:2243, 213:2244, 214:2251, 215:2270, 216:2283, 217:2285,
218:2308, 219:2313, 220:2331,
                    221:2338, 222:2352, 223:2358, 224:2369, 225:2379, 226:2388, 227:2389,
228:2448, 229:2449, 230:2457,
                    231:2471, 232:2482, 233:2496, 234:2499, 235:2500, 236:2511, 237:2519,
238:2526, 239:2551, 240:2562,
                    241:2564, 242:2566, 243:2570, 244:2575, 245:2588, 246:2593, 247:2595,
248:2604}

        userid = userids[((userNum-1) / 100) + 1]
        studentOffset = userNum - (((userNum/100))*100)
        if studentOffset == 0:
            studentNum = userid * 100
        else:
            studentNum = studentOffset + ((userid - 1) * 100)
        #studentStr = 'student, test (' + str(studentNum) + ')'
        creds = {'userid':str(userid), 'password':str(userid), 'testnum':studentNum}
    else:
        creds = {'userid':'', 'password':'', 'testnum':''}

    return creds


if __name__ == "__main__":

    if len(sys.argv) == 4:
        start_time = float(sys.argv[2])
    else:
        print "Usage: python %s <testnum from 1-10000> <start time in seconds since epoch>
<accelerator>" % sys.argv[0]
        sys.exit(-1)

    test_index = int(sys.argv[1])
    accelerator = float(sys.argv[3])
    creds = credentials(test_index)
    username = creds['userid']
    password = creds['password']
    testnum = creds['testnum']

    logfile = 'log/%s.log' % test_index
    try:
        os.remove(logfile)
    except OSError:
        pass  # File not found
    try:
        os.mkdir('log')
    except OSError:
        pass  # Directory already exists
    log_format = "%(asctime)s : " + str(test_index) + " : %(message)s"
    log.basicConfig(filename=logfile, level=log.DEBUG, format=log_format)
    log.getLogger('selenium.webdriver.remote.remote_connection').setLevel(log.ERROR)

    # Set random seed
    random.seed(test_index)

    try:

        test = NctestSelenium(username, password, testnum, accelerator, start_time)
        #test = NctestWebkit(username, password, testnum, start_time)
```

```
        test.run()

    # Log all tracebacks
    except Exception as e:
        import traceback
        ei = sys.exc_info()
        try:
            tb = traceback.format_exception(ei[0], ei[1], ei[2])
        finally:
            del ei
        for line in tb:
            log.error(line)
        raise

    finally:
        log.info('Exiting')
        try:
            test.finish()
        except Exception:
            pass
```