WISE 2.0—An Improved ELBA Toolkit

By Raghav Bhat

Table of Contents

Testing & Debugging

How I Helped Improve the Toolkit

Key Contributions

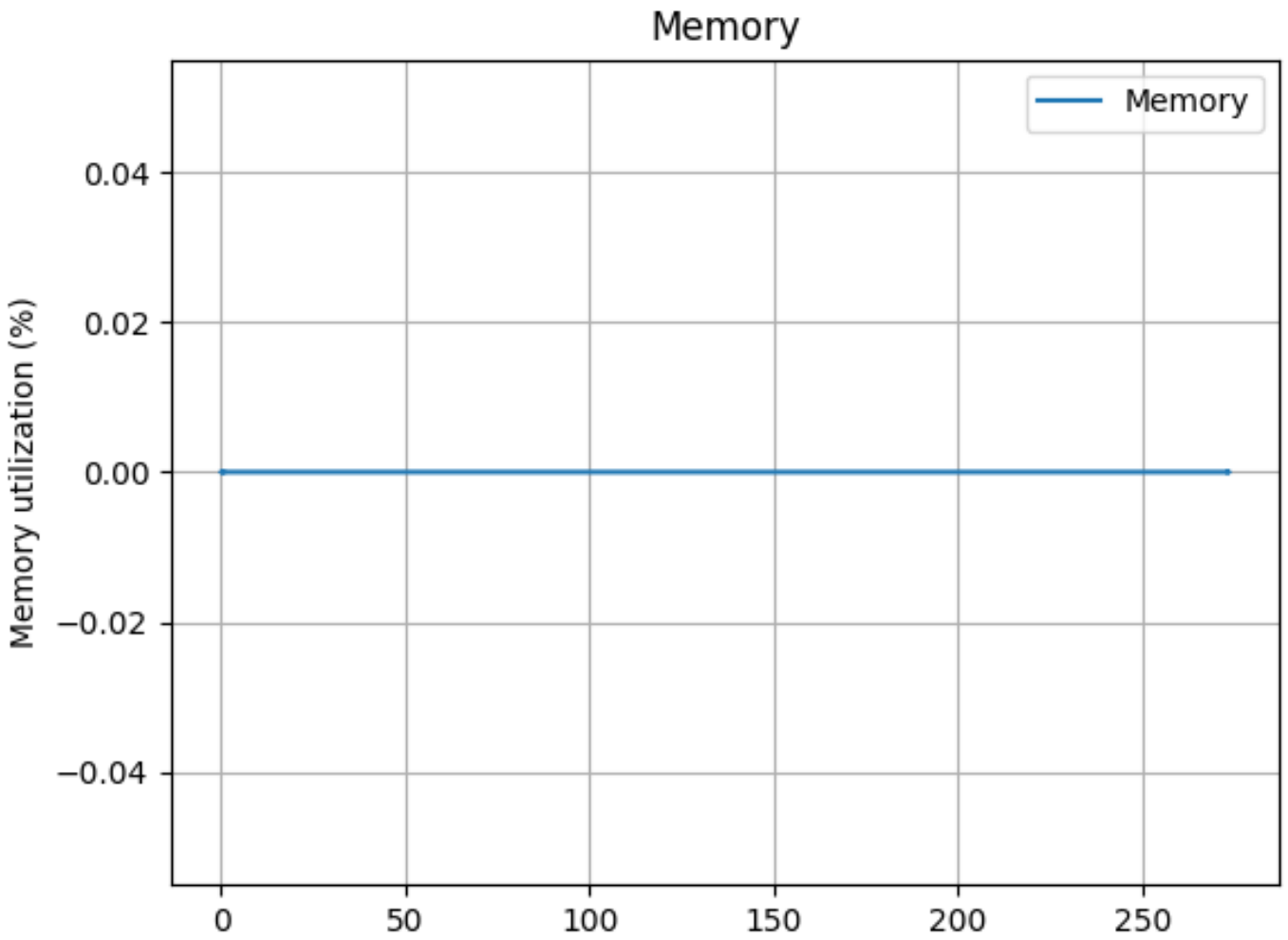
Worked with Rodrigo to Debug WISE Tutorial & Toolkit.

CloudLab uses tcsh instead of bash as default shell. *Tutorial was updated.*

Two critical files were mistakenly not pushed. *WISE Repo was updated.*

Identified 5+ Other Major Issues

Major Issue: Memory Plots Not Showing



Before Fix: Memory Plots

Cause: Lack of Precision in Memory Parser

Observation: Result files were non-empty

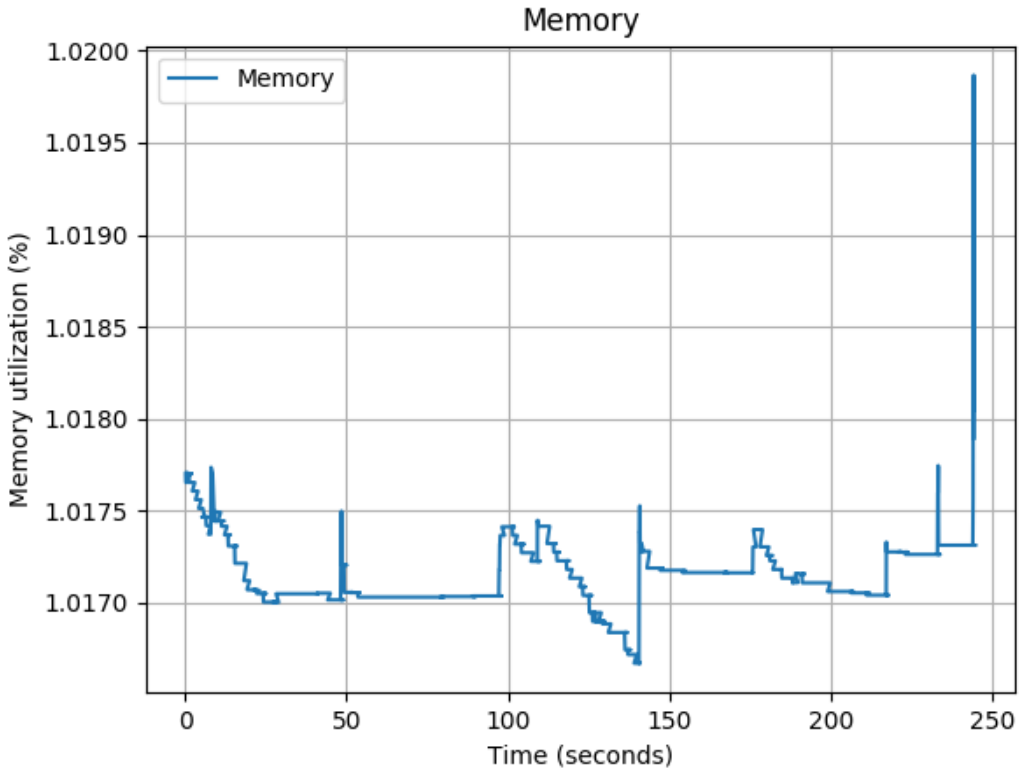
Solution: “Blow-Up” Graphs to Show Differences

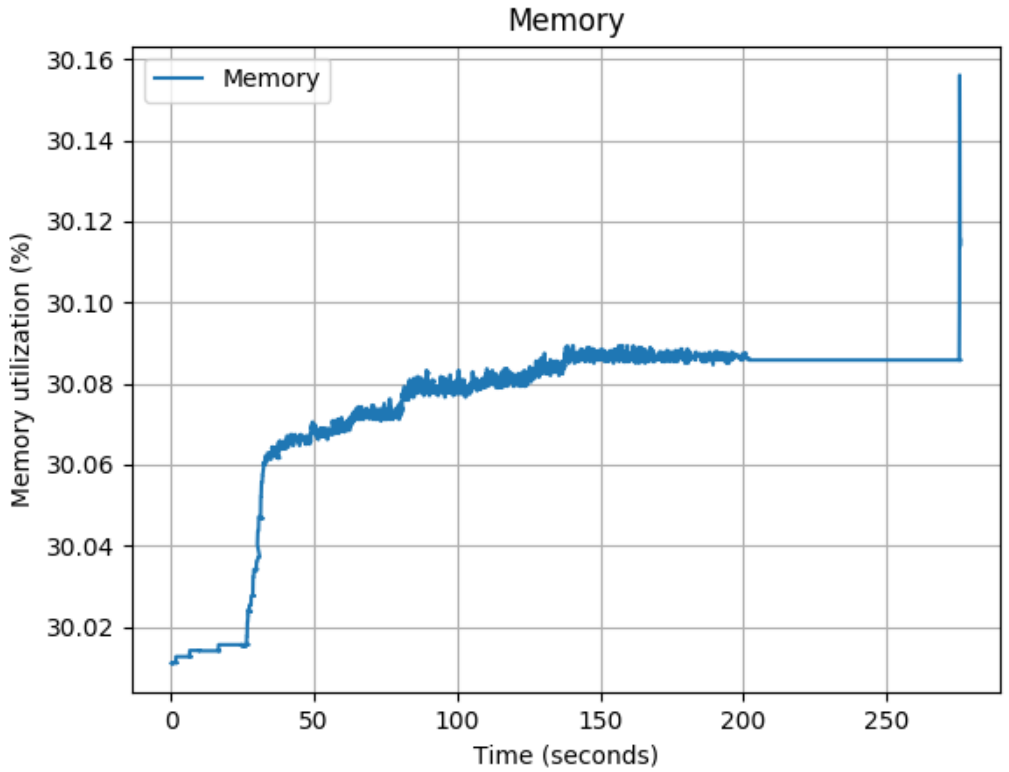
Done by saving more digits in parsing step

Root Cause: The total memory is *sporadically* calculated incorrectly when running on OpenStack

- Services use, on average, 2.4GB of Memory
- Each Virtual Machine for the service is allocated 8GB memory
- The WISE tool occasionally listed system memory as 256GB
- This happens to be the memory of the entire Clemson c8220 cluster
- Note that 2.4 is <1% of 256GB. **Hence, memory usage was rounding down to 0**

Result: Tool Glitch vs. Normal

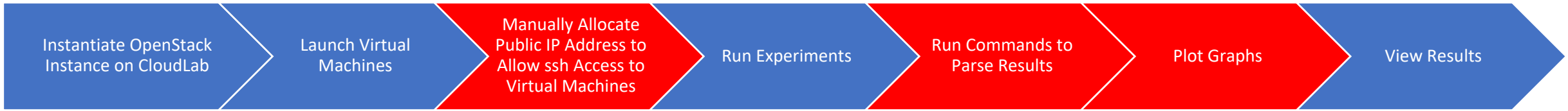




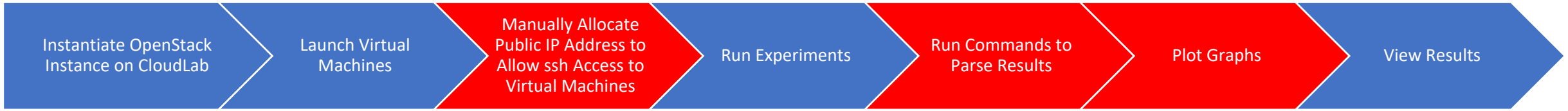
Automation

How I Fully Automated Deployment on OpenStack

How to Currently Run WISE on OpenStack



Our New Pipeline Simplifies Steps



Assigning Public-Facing IP Address to Access Cluster

- Previously, done using the OpenStack Dashboard Web Interface
- Now, automatically done when launching virtual machines
- Used OpenStack Python API to:
  - Request a public-facing IP Address
  - Assign public-facing IP Address to node1 within cluster

Floating IP Allocated When Virtual Machines are Deployed

experiment > scripts > openstack_setup.sh

```
19  ssh -i ~/.ssh/elba -T -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -o \
20      BatchMode=yes $CLOUDLAB_USERNAME@$OPENSTACK_CTLHOST "
21    source admin-openrc.sh
22    for i in {1..$OPENSTACK_NVIRTUALMACHINES}; do
23      cpno=\$((\$i % $OPENSTACK_NCOMPUTINGNODES))
24      if [ \$cpno -eq 0 ]; then
25        cpno=$OPENSTACK_NCOMPUTINGNODES
26      fi
27      echo \"Launching VM node\${i} in computing node cp-\${cpno}...\"
28      openstack server create \
29          --image $OPENSTACK_VMIMAGE \
30          --flavor $OPENSTACK_VMFLAVOR \
31          --key-name $OPENSTACK_KEYNAME \
32          --availability-zone nova:cp-\${cpno}.$CLOUDLAB_EXPNAME.$CLOUDLAB_PROJNAME.$CLOUDLAB_EXI
33          --network tun0-net \
34          node\${i}
35    done
36
37    pip3 install python-openstackclient
38    openstack floating ip create --project admin --subnet ext-subnet ext-net
39    ALLOCATED_IP_ADDRESS_VALUE=\`openstack floating ip list -c 'Floating IP Address' -f value | I
40    openstack server add floating ip node1 \$ALLOCATED_IP_ADDRESS_VALUE
41    echo \"Access node1 via IP address:   \$ALLOCATED_IP_ADDRESS_VALUE\"
42  "
43
```

Tutorial Includes Many Parsing Scripts

- Previously, you would need to run several commands to generate a plottable data format.
- Now, these repeatable steps have been captured in a shell script.

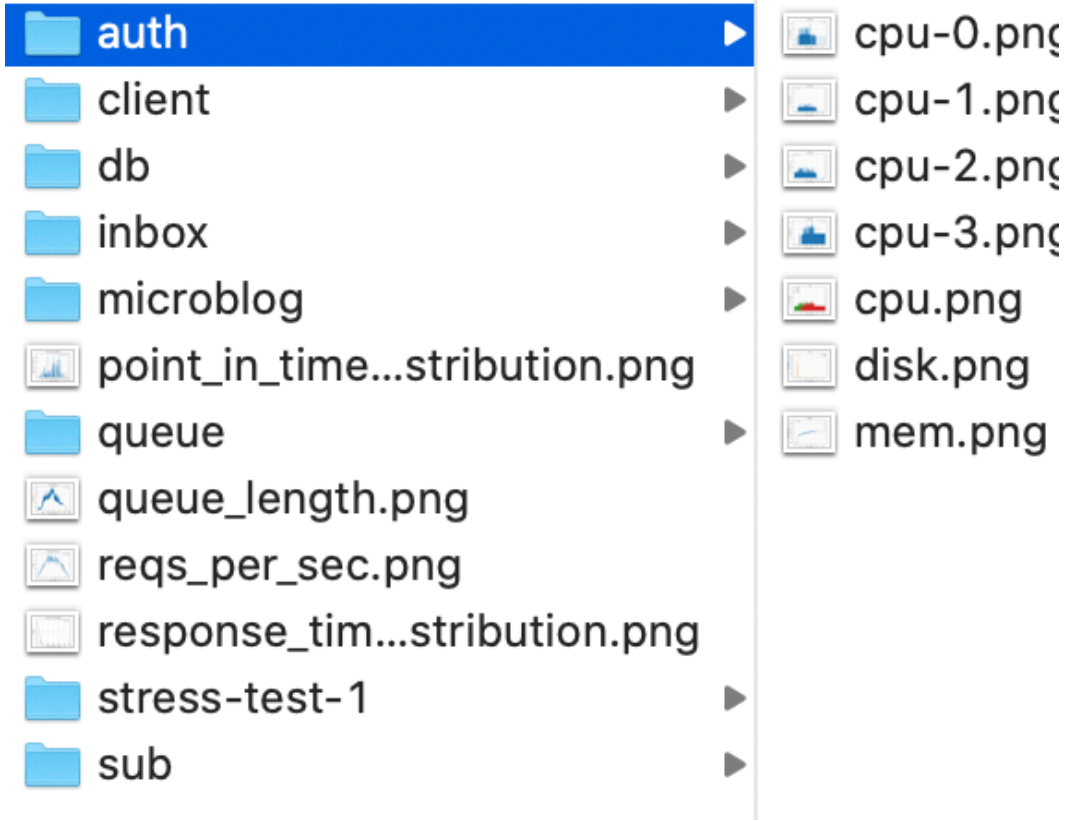
To uncompress the tarballs with the log files of each server:

```
# Uncompress the log files collected from the client server
mkdir -p analysis/client
tar -xzf log-client*.tar.gz -C analysis/client
# Uncompress the log files collected from the database server
mkdir -p analysis/db
tar -xzf log-db*.tar.gz -C analysis/db
# Uncompress the log files collected from the authentication server
mkdir -p analysis/auth
tar -xzf log-auth*.tar.gz -C analysis/auth
# Uncompress the log files collected from the inbox server
mkdir -p analysis/inbox
tar -xzf log-inbox*.tar.gz -C analysis/inbox
# Uncompress the log files collected from the microblog server
mkdir -p analysis/microblog
tar -xzf log-microblog*.tar.gz -C analysis/microblog
# Uncompress the log files collected from the queue server
mkdir -p analysis/queue
tar -xzf log-queue*.tar.gz -C analysis/queue
# Uncompress the log files collected from the subscription server
mkdir -p analysis/sub
tar -xzf log-sub*.tar.gz -C analysis/sub
```

Take a time to skim through these logs files and understand their format and content.

Automatically Plotting All Logs

- Wrote plot scripts in Matplotlib to plot CPU utilization, queue lengths, etc.
- Script neatly saves.
- Called from the Parsing Scripts.
  - Call could be replaced by custom plotting script.

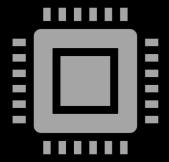


Noisy Neighbor Experiment

Adding Support for Noisy Neighbor Experiment to WISE

Contribution

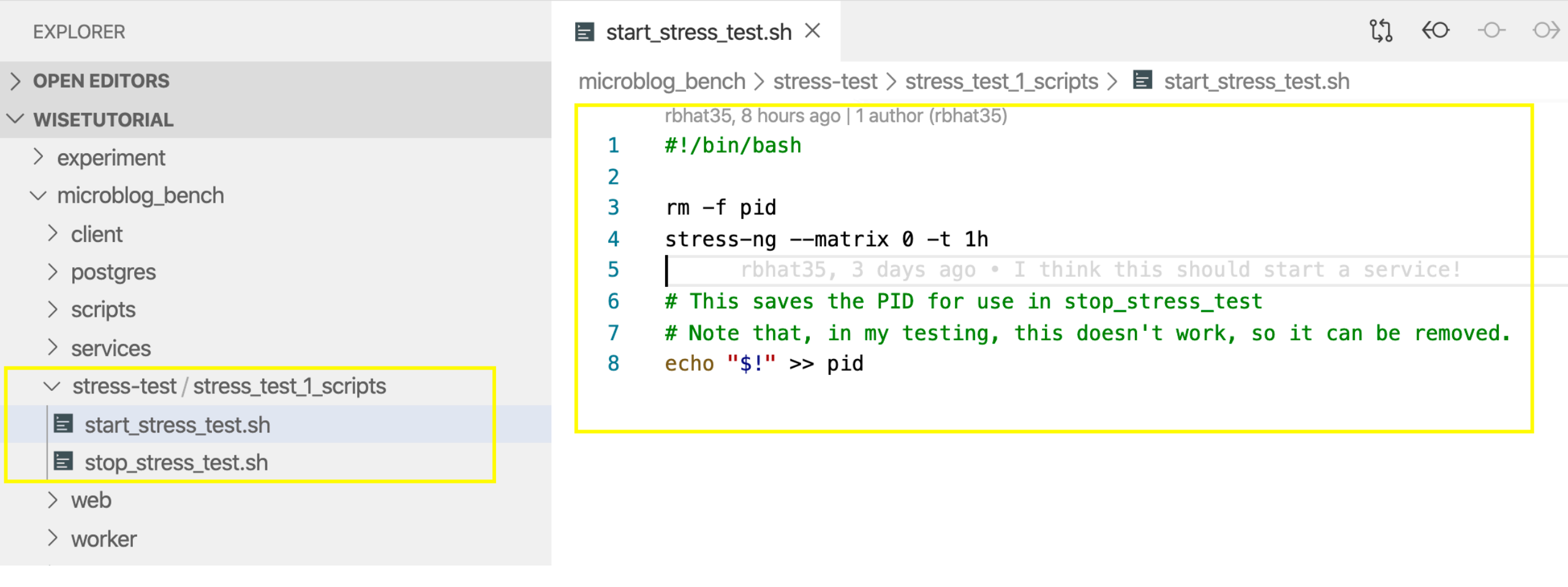
Not Previously Supported: Experiments to observe noisy neighbor phenomenon.



Implementation is Configurable—choose to stress CPU, Disk, Memory, Network, etc.

WISE Monitor processes run inside stress-testing machine.

Hence, CPU, Memory, Disk logs are generated, just like for other services.



Completely Configurable—Just Edit One File

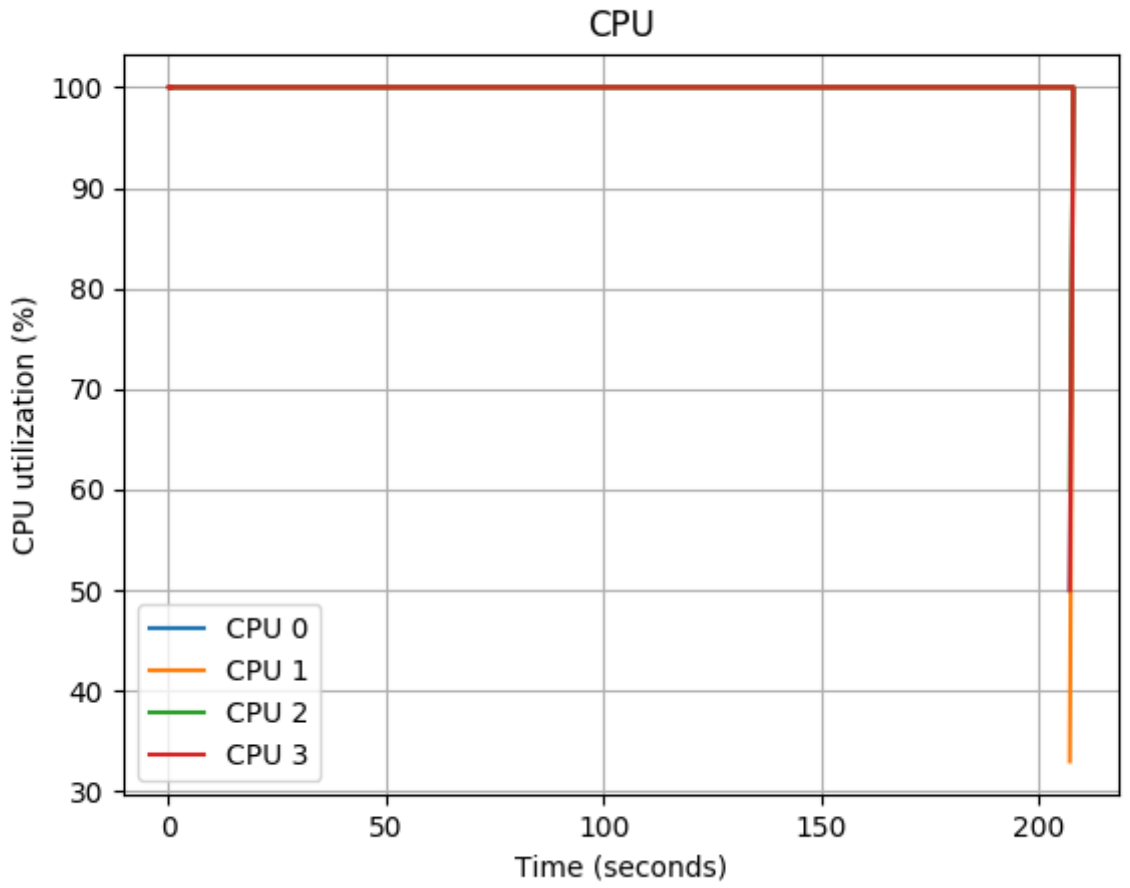
Config File to State Host

```
34    readonly SUB_PORT=9094
35    readonly CLIENT_HOSTS="10.254.0.6"
36
37    # Hostname of stress-testing nodes
38    readonly STRESS_TEST_1="10.254.3.72"
39
40    # Apache/mod_wsgi configuration.
41    readonly APACHE_PROCESSES=8
```

Deployed to Service When Running Experiment

```
583  echo "[$(date +%s)] Stress Test 1 tear down:"
584  for host in $STRESS_TEST_1; do      You, 5 hours ago • Changing the hostname
585    echo "  [$(date +%s)] Tearing down Stress Test 1 on host $host"
586    ssh -T -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no \
587        -o BatchMode=yes $USERNAME@$host "
588      # Stop server.
589      chmod +x $wise_home/microblog_bench/stress-test/stress_test_1_scripts/st
590      $wise_home/microblog_bench/stress-test/stress_test_1_scripts/stop_stress
591
592      # Stop resource monitors.
593      sudo pkill collectl
594      sleep 8
595
596      # Collect log data.
597      mkdir logs
598      mv $wise_home/collectl/data/coll-* logs/
599      gzip -d logs/coll-*
600      cat /proc/spec_connect > logs/spec_connect.csv
601      cat /proc/spec_sendto > logs/spec_sendto.csv
602      cat /proc/spec_recvfrom > logs/spec_recvfrom.csv
603      tar -C logs -czf log-stress-test-1-\$(echo \$(hostname) | awk -F'[-.]' '
604
605      # Stop event monitors.
606      sudo rmmod spec_connect
```

Result—Running CPU Stress Test on All Cores

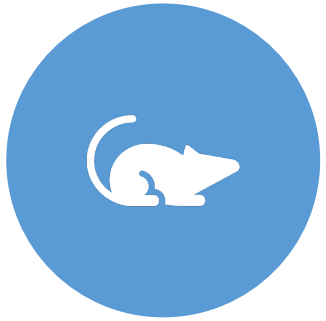


Key Challenge

Shell scripts are notoriously difficult to debug; no compiler, error messages not informative.

Had to edit an 800+ line shell script to run stressing script.

Each experimental run takes about 1 hour.



Had to run the Experiments 7-8 times before I got it right.

This Was By Far the Most Challenging, and Frustrating, Part of my Project

<u>Key Takeaways from This Portion (Many Software Engineering insights):</u>

- Understand *every line* of a script before you modify it.
- Avoid writing bash; it has its uses, of course, but it does not scale well and is not easily modified.
- When you have no debugger, building and testing in extremely small chunks is invaluable.
- Log as much as you can!

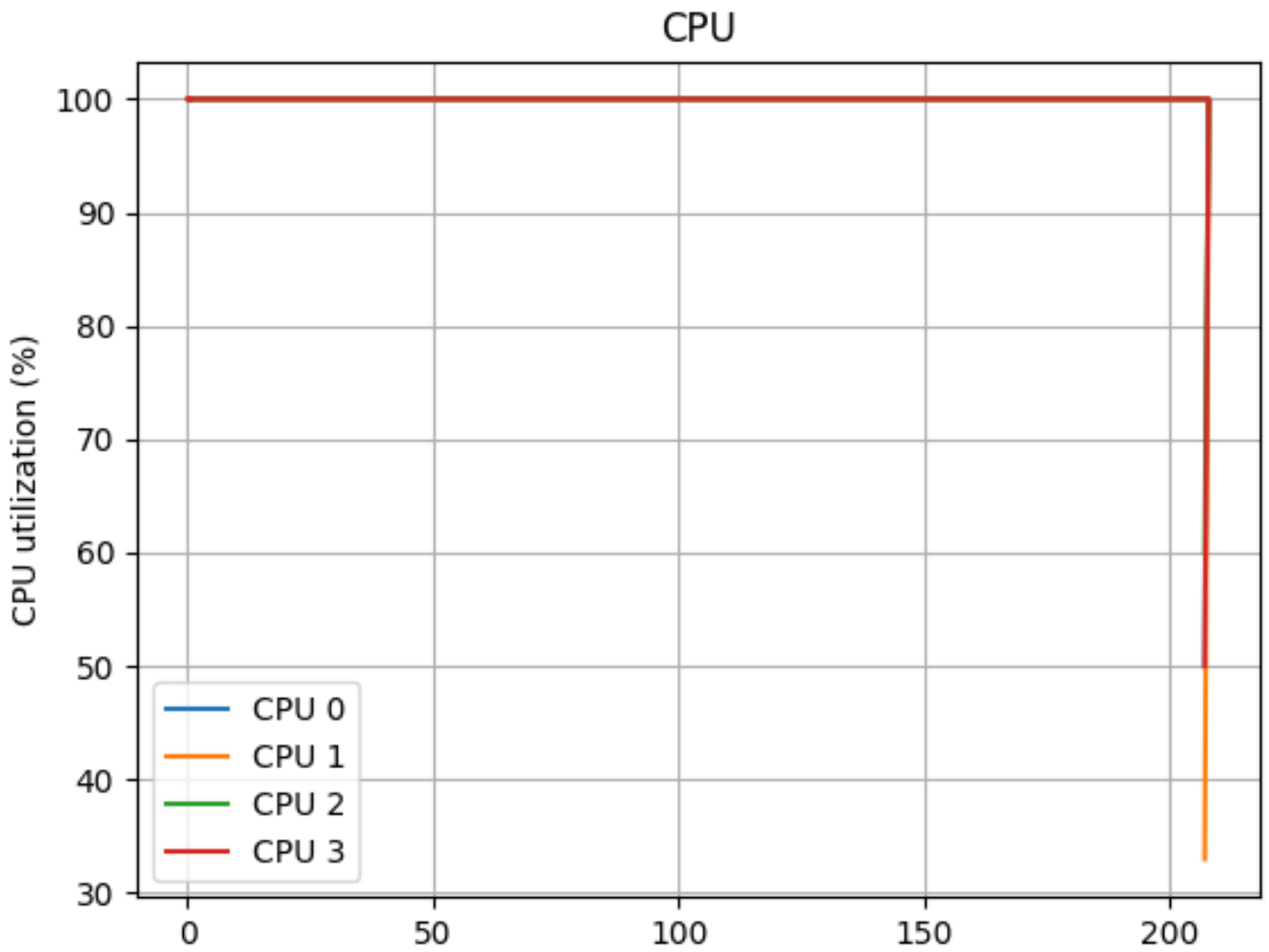
Results

I tested the entire pipeline out, stressing CPU while running the benchmark (standard workload configuration).

Baseline vs. Stress Test VM Allocation between OpenStack Nodes

Service	Node
Controller	1
WEB_HOSTS	2
POSTGRESQL_HOST	3
WORKER_HOSTS	1
MICROBLOG_HOSTS	2
AUTH_HOSTS	3
INBOX_HOSTS	1
QUEUE_HOSTS	2
SUB_HOSTS	3
CLIENT_HOSTS	1
STRESS_TEST_1	N/A

Service	Node
Controller	1
WEB_HOSTS	2
POSTGRESQL_HOST	3
WORKER_HOSTS	1
MICROBLOG_HOSTS	2
AUTH_HOSTS	3
INBOX_HOSTS	1
QUEUE_HOSTS	2
SUB_HOSTS	3
CLIENT_HOSTS	1
STRESS_TEST_1	2

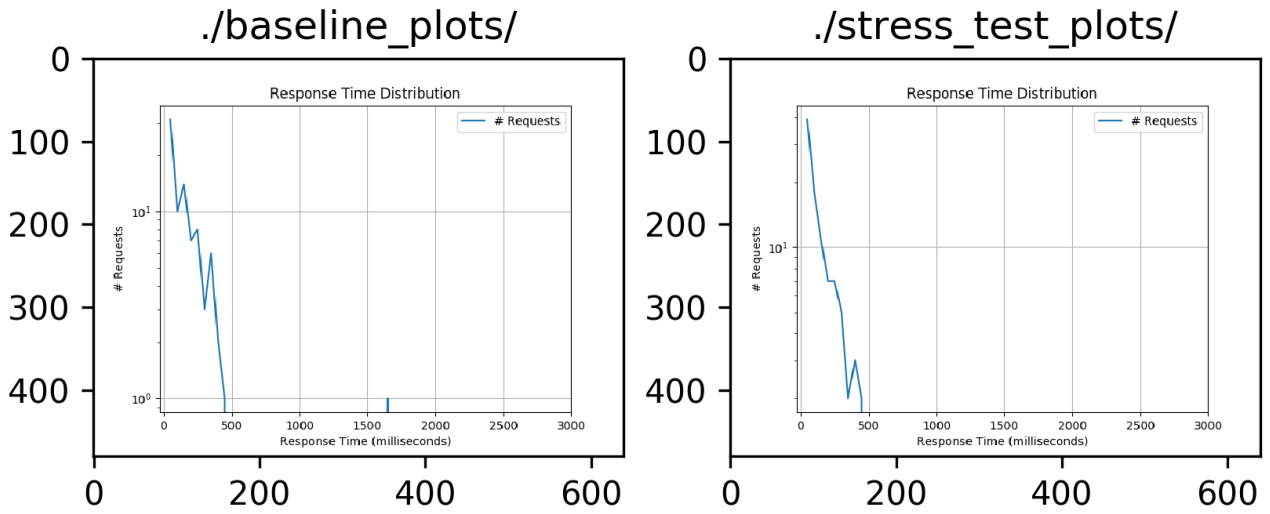


CPU Utilization Over Time during Stress Test Experiment

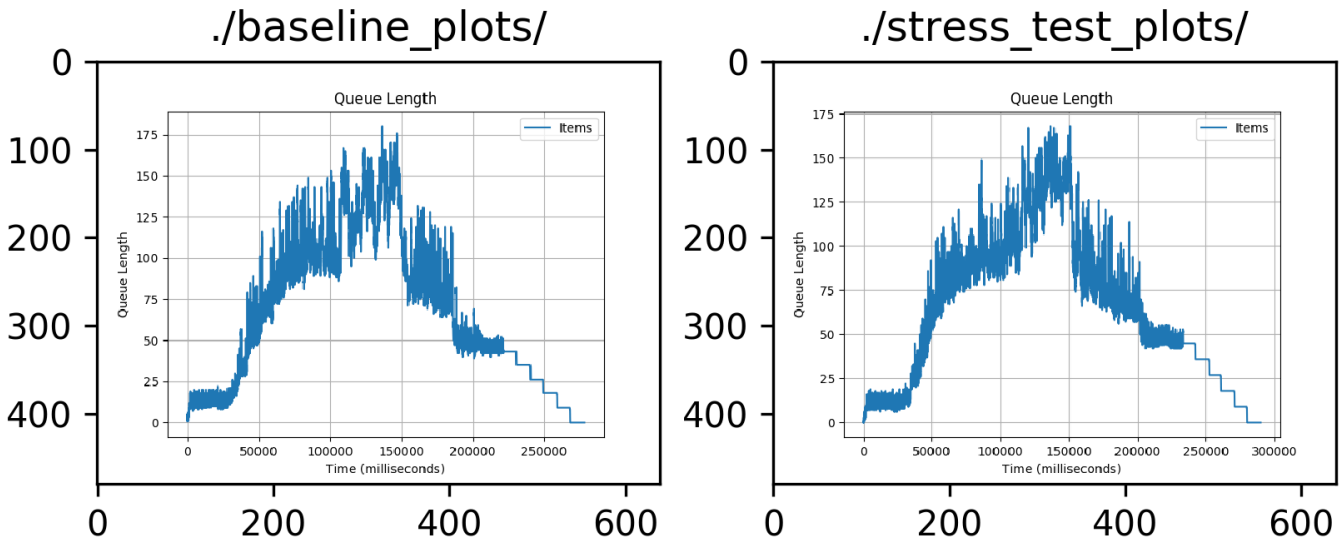
Expectation: Observe Noisy Neighbor

- We are expecting to observe millibottlenecks related to instruction cache misses.
  - Multiple VMs share the same cache. Due to consolidation, this increase the number of cache misses.
  - Our stress test does many floating-point matrix multiplications, so we can expect many more cache misses and thus degraded performance on collocated services and our benchmark as a whole.
- Idea: Look for Changes in Performance that might be caused by the high CPU usage on Node 2 by our stress test.
- Particularly, we will look at other services running in Node 2
  - Queue, Microblog, & Web Hosts

Response Time Distributions

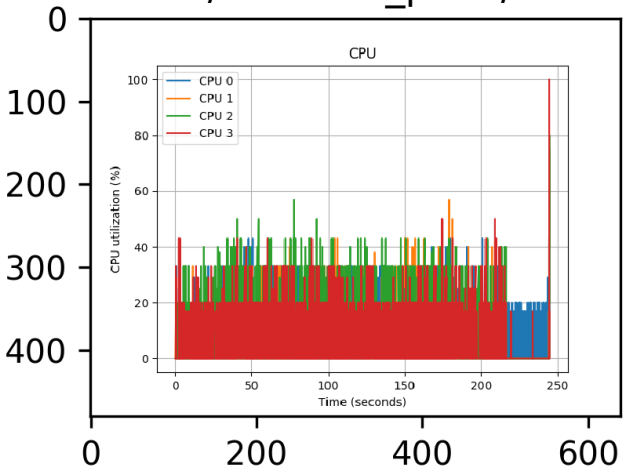


Queue Lengths

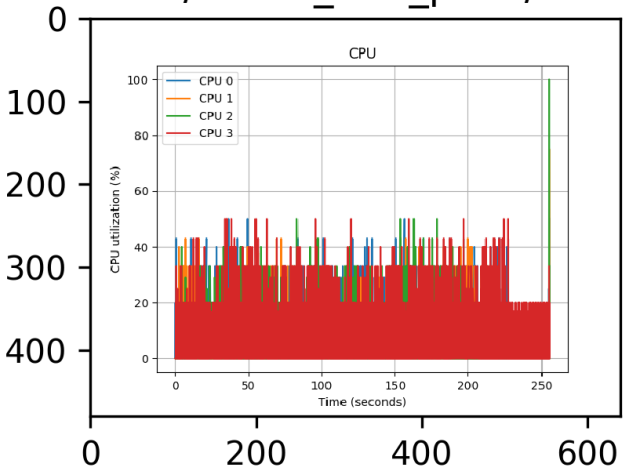


CPU Plots on the Queue Service

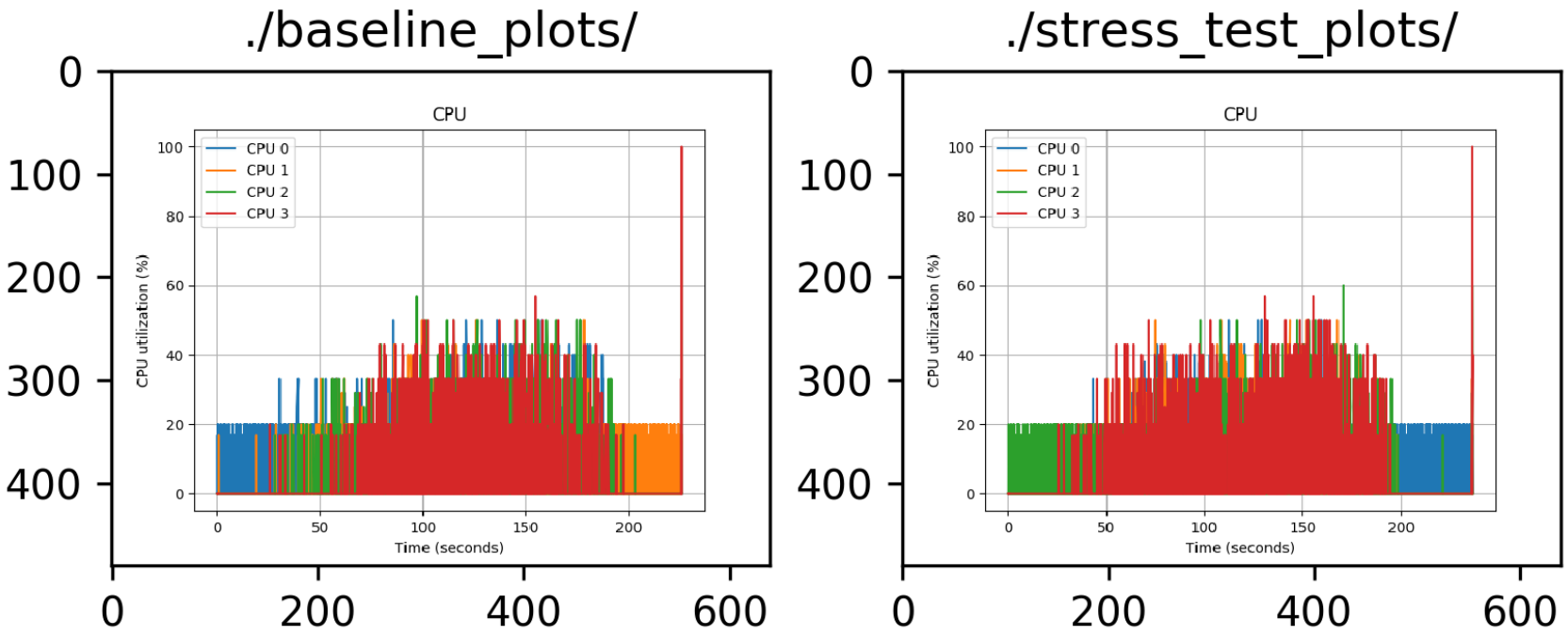
./baseline_plots/



./stress_test_plots/



CPU Plots on the Microblog Service



Possible Explanations

- Likely, there aren't enough Virtual Machines per host machine.
  - The c8220 host machines are very powerful (4 cores, 26GB RAM)—can easily handle 3 VMs.
- Perhaps co-located services aren't very CPU Intensive.
- OpenStack might allocate resources fairly between Virtual Machines on a Given Node, preventing one from "crowding out" others.
  - This idea of "rate limiting" is used by a lot of cloud providers. I learned about it first-hand while working at Microsoft.

Conclusion

Next Steps—Before Final Report

- Document All Contributions to Help Ease Transition to Master Branch.
  - Create a readme with the files for instructions on how to make the changes in master repo. It will include a list of files I edited, which can be diffed with the original ones to find changes.
  - Non-trivial, since my repository has many modifications I had to make to run on my machine.
  - Given time frame (and lack of hosts on CloudLab), this is more feasible than trying to push to master.
- Document steps for configuring stress test.

Next Steps—Given More Time

- *Due to a shortened project schedule, was not able to run an exhaustive set of experiments.*
- Experiment 1: Stress the disk in the same host the database is deployed in. This will create millibottlenecks in disk.
- Experiment 2: Allocate more CPUs on the same host. For example, we would more likely see contention if we ran 2 VMs per core (i.e. 8 VMs per host).
- Allow multiple different stress tests scripts to run on different hosts.
  - i.e. run CPU stress test on one host and memory stress test on another.

Impact of COVID-19

Lost two weeks in the project due to Extended Spring Break, TA Unavailability, and because I had to move *twice* in the span of 2 weeks.

No home internet for **one week** after moving into permanent house.

Of course, transition to online classes & special challenges due to role as a Teaching Assistant and Resident Advisor.

Thank You: Professor Pu & Rodrigo

I learned a lot from this project. I now have a deep understanding of the millibottleneck theory of performance bugs and some first-hand experience of it!