Secure Multiparty Computation Sprint 5

Developer | Hasnain Abdur Rehman| hasnain@bu.edu
Developer | Pierre-François Wolfe | pwolfe@bu.edu
Developer | Samyak Jain | samyakj@bu.edu
Developer | Suli Hu | sulihu@bu.edu
Developer | Yufeng Lin | yflin@bu.edu
Mentor/Client | John Liagouris | liagos@bu.edu
Mentor/Client | Vasiliki Kalavri | vkalavri@bu.edu
Subject-Matter Expert | Mayank Varia | varia@bu.edu

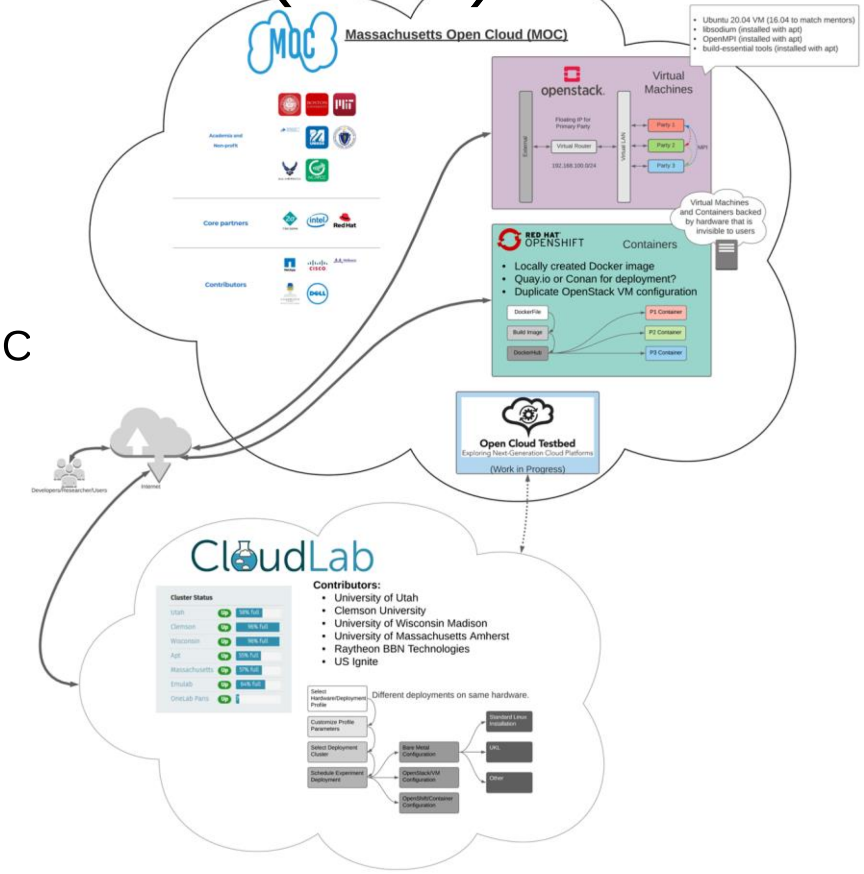
BOSTON UNIVERSITY

Presentation Outline

- Project Recap
- Project Goals & Sprint 5 Stories/Tasks
- Work Accomplished & Information Learned
  - Bare-Metal → Additional CloudLab Topologies
  - Containers → OpenShift Implementation Steps
  - Automation → Ansible Playbook
  - Profiling/Benchmarking → Score-P and Inspection tools
- Project Organization Assessment (Burndown)
- Future goals (Mentor priorities)

Recap of Multi-Party Computation (MPC)

- MPC enables...
  - Shared Computation on Private Data
  - Protects the Privacy of Data
  - Mutually Agreed Computation
- Our mentors…
  - Are using three party Secret Sharing MPC
  - Perform Relational Queries with MPC
  - Keep all parts secure vs. splitting into secure and insecure steps
- Our mission…
  - Profile this new MPC library
  - Identify bottlenecks
  - Compare deployment scenarios and find the best performance



- Ubuntu 20.04 VM (16.04 to match mentors)
- libsodium (installed with apt)
- OpenMPI (installed with apt)
- build-essential tools (installed with apt)

- Locally created Docker image
- Quay.io or Conan for deployment?
- Duplicate OpenStack VM configuration

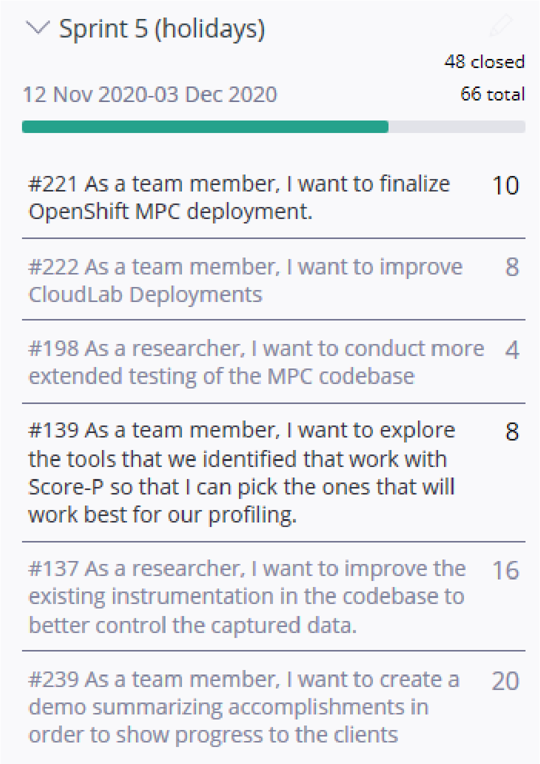
Virtual Machines and Containers backed by hardware that is invisible to users

Contributors:
- University of Utah
- Clemson University
- University of Wisconsin Madison
- University of Massachusetts Amherst
- Raytheon BBN Technologies
- US Ignite

Different deployments on same hardware.

Project Goals & Sprint 5 Stories/Tasks

- OpenShift
  - Solve final deployment bugs
- CloudLab
  - Create and test Ring topology
  - Test LAN and Ring on single and multi cluster
- Automation
  - Ansible playbook for unified VM (OpenShift) and bare-metal (CloudLab) deployment
- Data Collection/Analysis
  - Score-P captures and some tools for inspection



Sprint 5 (holidays)

48 closed

12 Nov 2020-03 Dec 2020

66 total

#221 As a team member, I want to finalize OpenShift MPC deployment.	10
#222 As a team member, I want to improve CloudLab Deployments	8
#198 As a researcher, I want to conduct more extended testing of the MPC codebase	4
#139 As a team member, I want to explore the tools that we identified that work with Score-P so that I can pick the ones that will work best for our profiling.	8
#137 As a researcher, I want to improve the existing instrumentation in the codebase to better control the captured data.	16
#239 As a team member, I want to create a demo summarizing accomplishments in order to show progress to the clients	20

Openshift Container Platform

- Original OpenShift deployment approaches didn’t work
  - Pulling from DockerHub to OpenShift
  - Pushing to OpenShift Internal Container Registry
  - Pushing to Quay and pulling to OpenShift
- Finally, settled on pushing a DockerFile directly to OpenShift and starting a binary build!

Openshift Container Platform

- Main Impediment: Getting SSH Daemon (server) running on Openshift.
  - Made a Dockerfile running SSHd correctly on local Docker setup.
  - But when deployed on Openshift, container would crash loop!

- SSHd would return error “no host keys found --exiting”
- This meant ‘ssh-keygen -A’ command wasn’t running fine.
- Managed to pause the Container from crashing, and got a shell inside.

Strange error on OpenShift when running ‘ssh-keygen’

- Error: “No user exists for uid 1000500300”
- Stuck on this error for good amount of time.Tried bypassing, didn’t work.
- Found ‘Openshift Specific Guidelines’ which said containers are assigned random uid’s
    - For security purposes to stop processes trying to escape the container
- Since this random uid is not in /etc/passwd, programs like ssh would fail to run.
- Another thing to fix -- ssh would not have sufficient permissions for files it needs if running via a random uid.

Fixing error on OpenShift

- Fix:
  - Create specific files/directories and own them by root group, and give group read/write access.
  - Files ran in those directories would run as under an arbitrary user.
- Edited the docker file to:
  - Make some specific directories writable -- to the root group -- including the ones ssh requires sufficient permissions to.
  - Made ‘passwd’ file writeable to the root group too, and fixed the UID by editing the file.

SSH running on Openshift

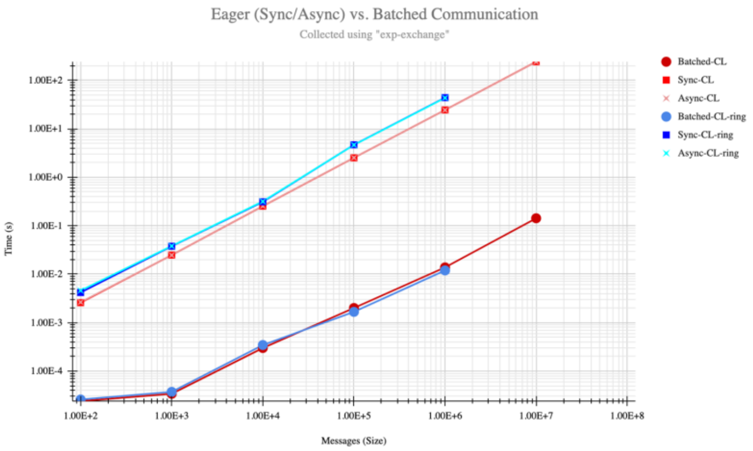
- Once this was done, ssh-keygen command ran well, and SSH Daemon got up and running as well!
- Port 22 is privileged on Openshift.
  - Had to use port 2022.
  - Then created a service to expose an internal IP address/hostname.
  - Edited the service using ‘oc edit svc my_container’ command to map service port 2022 to the target port 22.
- Finally we have 3 containers running which can ssh into each other!

Bare-Metal Topologies (CloudLab)

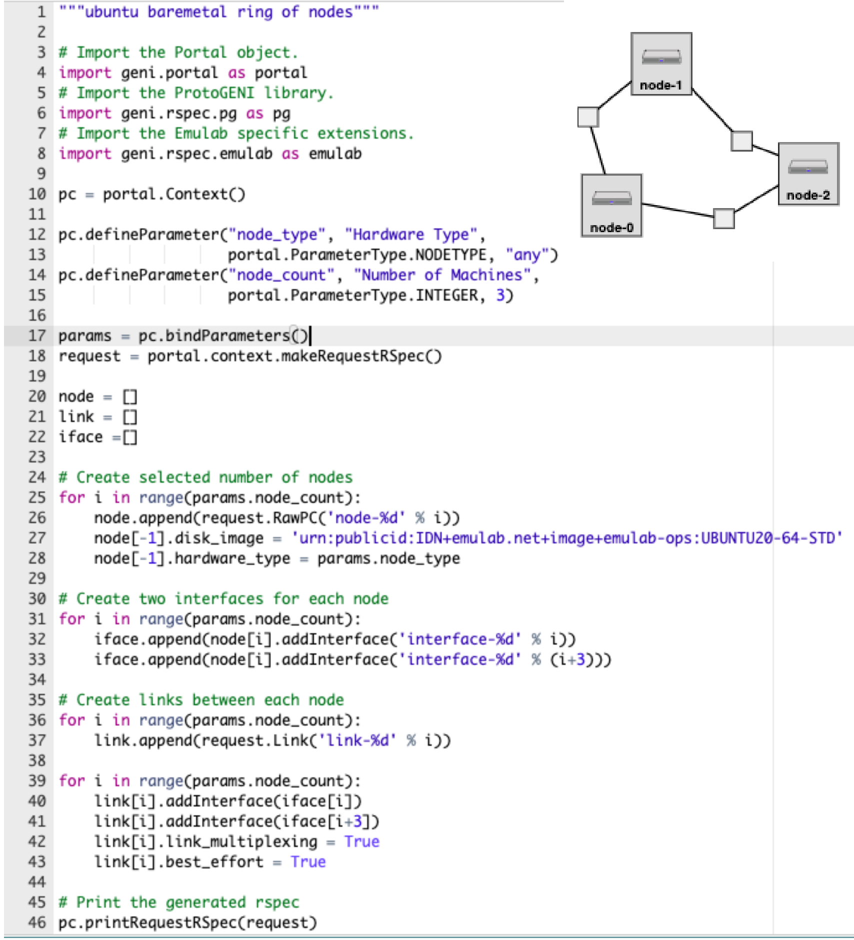
- Single Cluster and Multi Cluster
  - Two cluster practical limit (challenges with 3 locations)
- Topologies
  - LAN topology (previously tested
  - Ring topology (using reference designs)
- Useful options to be aware of (geni-lib)
  - lan.bandwidth
  - node.Site

Single cluster ring

- Successfully created ring topology
- Each node has 2 interfaces each
- link_multiplexing and best_effort options used to create virtual links
- Plot shows ring vs LAN deployments of the 3 nodes/parties

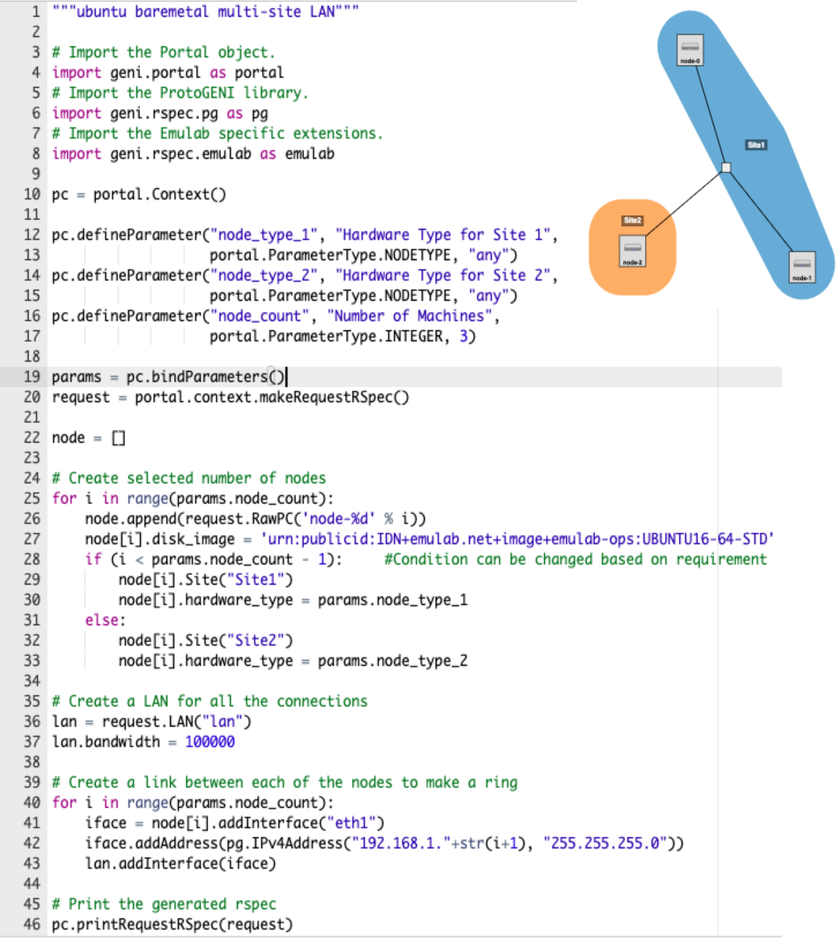

Eager (Sync/Async) vs. Batched Communication

Collected using "exp-exchange"


```
"""ubuntu baremetal ring of nodes"""

# Import the Portal object.
import geni.portal as portal
# Import the ProtoGENI library.
import geni.rspec.pg as pg
# Import the Emulab specific extensions.
import geni.rspec.emulab as emulab

pc = portal.Context()

pc.defineParameter("node_type", "Hardware Type",
                   portal.ParameterType.NODETYPE, "any")
pc.defineParameter("node_count", "Number of Machines",
                   portal.ParameterType.INTEGER, 3)

params = pc.bindParameters()
request = portal.context.makeRequestRSpec()

node = []
link = []
iface =[]

# Create selected number of nodes
for i in range(params.node_count):
    node.append(request.RawPC('node-%d' % i))
    node[-1].disk_image = 'urn:publicid:IDN+emulab.net+image+emulab-ops:UBUNTU20-64-STD'
    node[-1].hardware_type = params.node_type

# Create two interfaces for each node
for i in range(params.node_count):
    iface.append(node[i].addInterface('interface-%d' % i))
    iface.append(node[i].addInterface('interface-%d' % (i+3)))

# Create links between each node
for i in range(params.node_count):
    link.append(request.Link('link-%d' % i))

for i in range(params.node_count):
    link[i].addInterface(iface[i])
    link[i].addInterface(iface[i+3])
    link[i].link_multiplexing = True
    link[i].best_effort = True

# Print the generated rspec
pc.printRequestRSpec(request)
```

Multi-site LAN

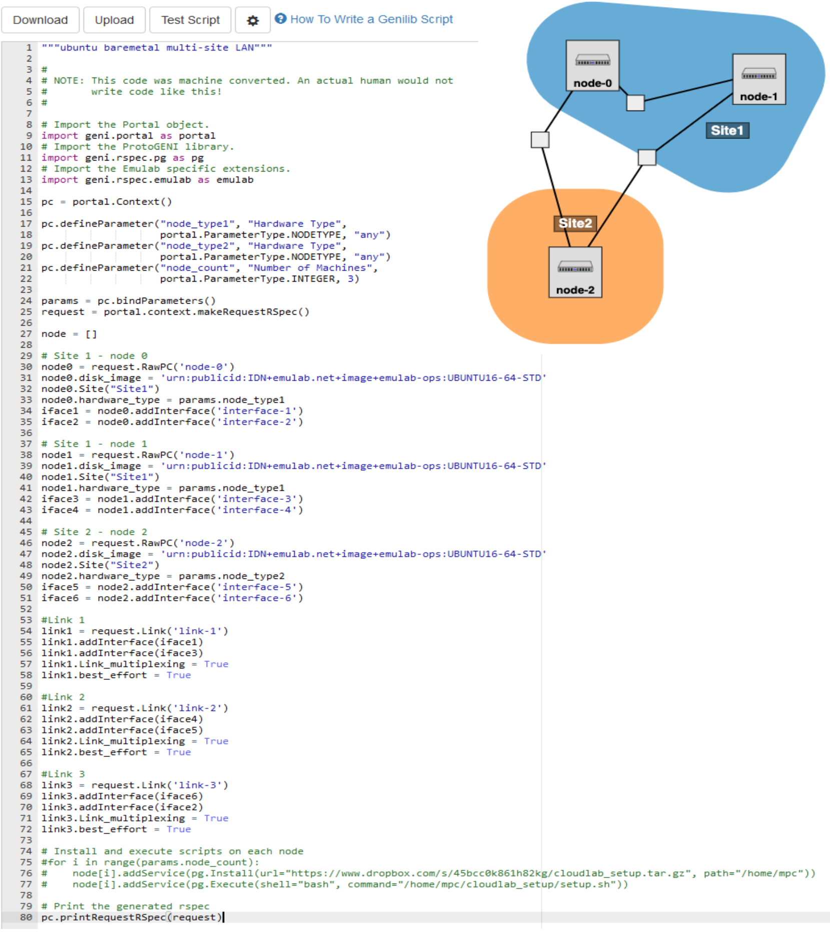
- Deployed nodes across multiple clusters
- Used node.Site() option to specify different clusters
- Cloudlab allows maximum 2 cluster stitching in multi-site experiments
- Need to specify lan.bandwidth for this



```python
"""ubuntu baremetal multi-site LAN"""

# Import the Portal object.
import geni.portal as portal
# Import the ProtoGENI library.
import geni.rspec.pg as pg
# Import the Emulab specific extensions.
import geni.rspec.emulab as emulab

pc = portal.Context()

pc.defineParameter("node_type_1", "Hardware Type for Site 1",
                   portal.ParameterType.NODETYPE, "any")
pc.defineParameter("node_type_2", "Hardware Type for Site 2",
                   portal.ParameterType.NODETYPE, "any")
pc.defineParameter("node_count", "Number of Machines",
                   portal.ParameterType.INTEGER, 3)

params = pc.bindParameters()
request = portal.context.makeRequestRSpec()

node = []

# Create selected number of nodes
for i in range(params.node_count):
    node.append(request.RawPC('node-%d' % i))
    node[i].disk_image = 'urn:publicid:IDN+emulab.net+image+emulab-ops:UBUNTU16-64-STD'
    if (i < params.node_count - 1):     #Condition can be changed based on requirement
        node[i].Site("Site1")
        node[i].hardware_type = params.node_type_1
    else:
        node[i].Site("Site2")
        node[i].hardware_type = params.node_type_2

# Create a LAN for all the connections
lan = request.LAN("lan")
lan.bandwidth = 100000

# Create a link between each of the nodes to make a ring
for i in range(params.node_count):
    iface = node[i].addInterface("eth1")
    iface.addAddress(pg.IPv4Address("192.168.1."+str(i+1), "255.255.255.0"))
    lan.addInterface(iface)

# Print the generated rspec
pc.printRequestRSpec(request)
```

Multi-site ring

- Deployed the ring topology in multi-site setting
- Based on a combination of:
  - Single cluster ring profile
  - Multi-site LAN profile



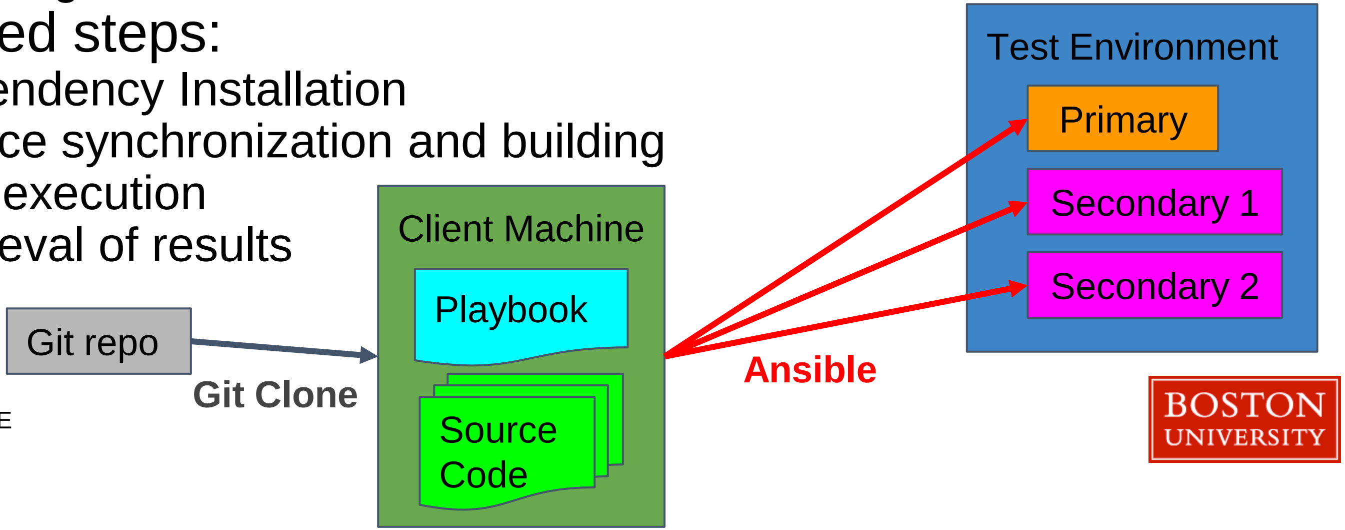
```
"""ubuntu baremetal multi-site LAN"""

#
# NOTE: This code was machine converted. An actual human would not
#       write code like this!
#

# Import the Portal object.
import geni.portal as portal
# Import the ProtoGENI library.
import geni.rspec.pg as pg
# Import the Emulab specific extensions.
import geni.rspec.emulab as emulab

pc = portal.Context()

pc.defineParameter("node_type1", "Hardware Type",
                   portal.ParameterType.NODETYPE, "any")
pc.defineParameter("node_type2", "Hardware Type",
                   portal.ParameterType.NODETYPE, "any")
pc.defineParameter("node_count", "Number of Machines",
                   portal.ParameterType.INTEGER, 3)

params = pc.bindParameters()
request = portal.context.makeRequestRSpec()

node = []

# Site 1 - node 0
node0 = request.RawPC('node-0')
node0.disk_image = 'urn:publicid:IDN+emulab.net+image+emulab-ops:UBUNTU16-64-STD'
node0.Site("Site1")
node0.hardware_type = params.node_type1
iface1 = node0.addInterface('interface-1')
iface2 = node0.addInterface('interface-2')

# Site 1 - node 1
node1 = request.RawPC('node-1')
node1.disk_image = 'urn:publicid:IDN+emulab.net+image+emulab-ops:UBUNTU16-64-STD'
node1.Site("Site1")
node1.hardware_type = params.node_type1
iface3 = node1.addInterface('interface-3')
iface4 = node1.addInterface('interface-4')

# Site 2 - node 2
node2 = request.RawPC('node-2')
node2.disk_image = 'urn:publicid:IDN+emulab.net+image+emulab-ops:UBUNTU16-64-STD'
node2.Site("Site2")
node2.hardware_type = params.node_type2
iface5 = node2.addInterface('interface-5')
iface6 = node2.addInterface('interface-6')

#Link 1
link1 = request.Link('link-1')
link1.addInterface(iface1)
link1.addInterface(iface3)
link1.Link_multiplexing = True
link1.best_effort = True

#Link 2
link2 = request.Link('link-2')
link2.addInterface(iface4)
link2.addInterface(iface5)
link2.Link_multiplexing = True
link2.best_effort = True

#Link 3
link3 = request.Link('link-3')
link3.addInterface(iface6)
link3.addInterface(iface2)
link3.Link_multiplexing = True
link3.best_effort = True

# Install and execute scripts on each node
#for i in range(params.node_count):
#    node[i].addService(pg.Install(url="https://www.dropbox.com/s/45bcc0k861h82kg/cloudlab_setup.tar.gz", path="/home/mpc"))
#    node[i].addService(pg.Execute(shell="bash", command="/home/mpc/cloudlab_setup/setup.sh"))

# Print the generated rspec
pc.printRequestRSpec(request)
```

Original Deployment Strategies

- Local Testing
  - Compile and run with MPI directly
  - Dockerfile (launched with docker-compose) three containers
- OpenShift (Containers)
  - Dockerfile three containers (different pods)
- OpenStack (VMs)
  - Three VMs configured manually or with shell scripts
- CloudLab (Bare-Metal)
  - Three machines configured with shell scripts

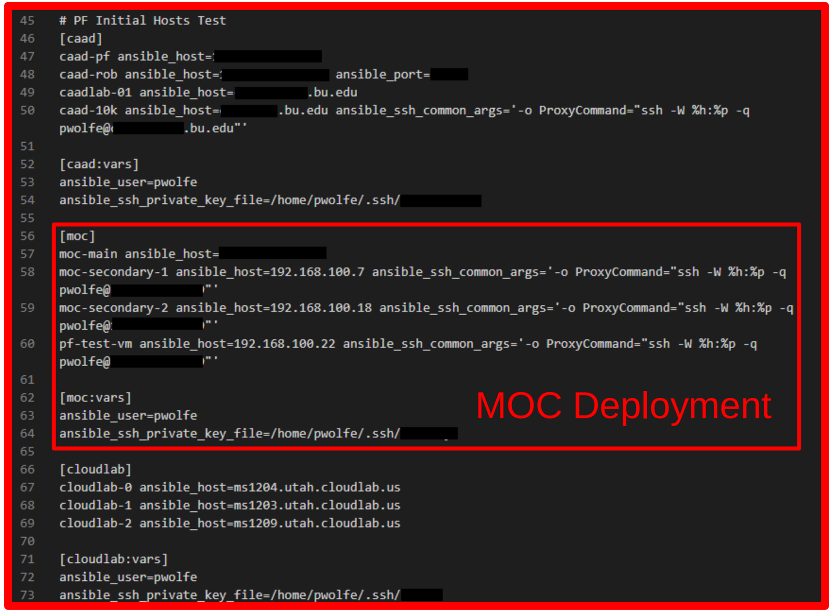
Deployment Automation/Consolidation (Ansible)

- Targets VMs and CloudLab
  - CloudLab particularly benefits from our playbook
- Dockerfile based deployments could be supported
  - Less urgent but would be ideal to add
- Supported steps:
  - Dependency Installation
  - Source synchronization and building
  - Test execution
  - Retrieval of results



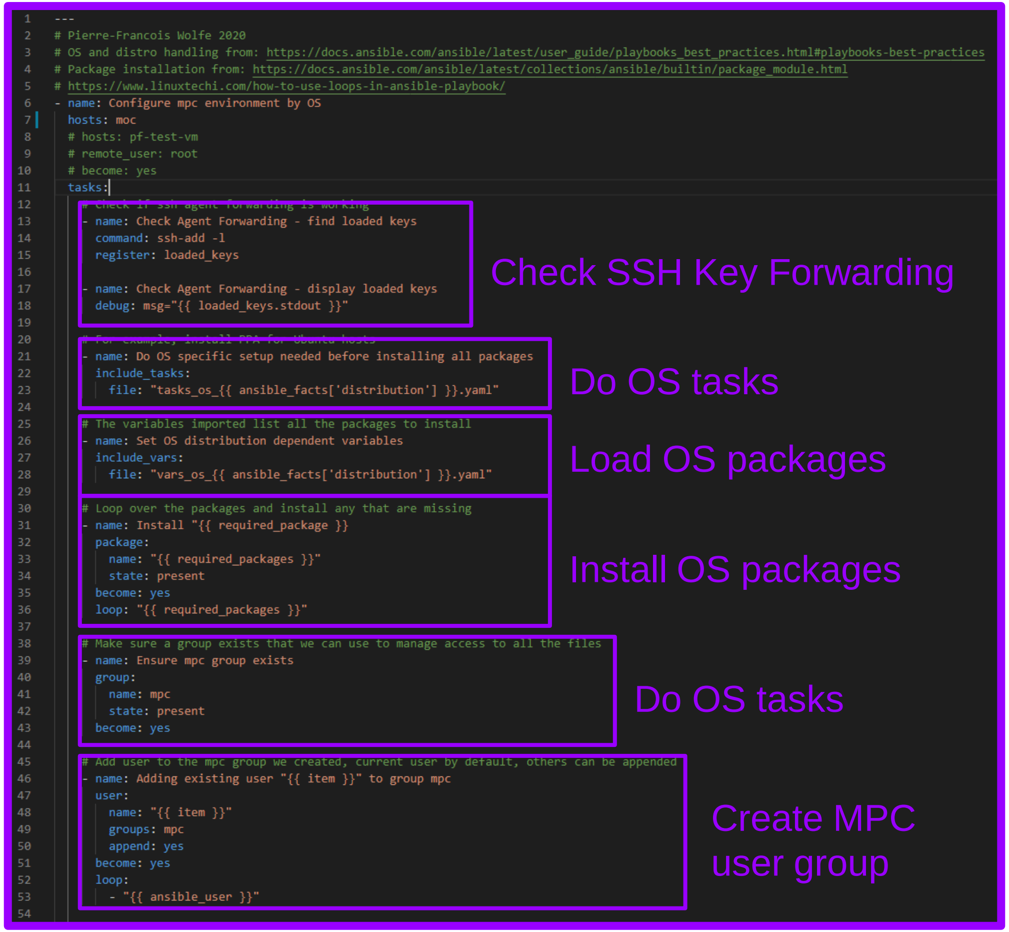
Ansible Detail

1. Install ansible: `pip3 install ansible`
2. Create inventory of hosts
3. Create a playbook
4. Execute the playbook



```
# PF Initial Hosts Test
[caad]
caad-pf ansible_host=[redacted]
caad-rob ansible_host=[redacted] ansible_port=[redacted]
caadlab-01 ansible_host=[redacted].bu.edu
caad-10k ansible_host=[redacted].bu.edu ansible_ssh_common_args='-o ProxyCommand="ssh -W %h:%p -q pwolfe@[redacted].bu.edu"'

[caad:vars]
ansible_user=pwolfe
ansible_ssh_private_key_file=/home/pwolfe/.ssh/[redacted]

[moc]
moc-main ansible_host=[redacted]
moc-secondary-1 ansible_host=192.168.100.7 ansible_ssh_common_args='-o ProxyCommand="ssh -W %h:%p -q pwolfe@[redacted]"'
moc-secondary-2 ansible_host=192.168.100.18 ansible_ssh_common_args='-o ProxyCommand="ssh -W %h:%p -q pwolfe@[redacted]"'
pf-test-vm ansible_host=192.168.100.22 ansible_ssh_common_args='-o ProxyCommand="ssh -W %h:%p -q pwolfe@[redacted]"'

[moc:vars]
ansible_user=pwolfe
ansible_ssh_private_key_file=/home/pwolfe/.ssh/[redacted]

[cloudlab]
cloudlab-0 ansible_host=ms1204.utah.cloudlab.us
cloudlab-1 ansible_host=ms1203.utah.cloudlab.us
cloudlab-2 ansible_host=ms1209.utah.cloudlab.us

[cloudlab:vars]
ansible_user=pwolfe
ansible_ssh_private_key_file=/home/pwolfe/.ssh/[redacted]
```

MOC Deployment



```
---
# Pierre-Francois Wolfe 2020
# OS and distro handling from: https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html#playbooks-best-practices
# Package installation from: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/package_module.html
# https://www.linuxtechi.com/how-to-use-loops-in-ansible-playbook/
- name: Configure mpc environment by OS
  hosts: moc
  # hosts: pf-test-vm
  # remote_user: root
  # become: yes
  tasks:
    # Check if ssh agent forwarding is working
    - name: Check Agent Forwarding - find loaded keys
      command: ssh-add -l
      register: loaded_keys

    - name: Check Agent Forwarding - display loaded keys
      debug: msg="{{ loaded_keys.stdout }}"

    # For example, install PPA for Ubuntu hosts
    - name: Do OS specific setup needed before installing all packages
      include_tasks:
        file: "tasks_os_{{ ansible_facts['distribution'] }}.yaml"

    # The variables imported list all the packages to install
    - name: Set OS distribution dependent variables
      include_vars:
        file: "vars_os_{{ ansible_facts['distribution'] }}.yaml"

    # Loop over the packages and install any that are missing
    - name: Install "{{ required_package }}
      package:
        name: "{{ required_packages }}"
        state: present
      become: yes
      loop: "{{ required_packages }}"

    # Make sure a group exists that we can use to manage access to all the files
    - name: Ensure mpc group exists
      group:
        name: mpc
        state: present
      become: yes

    # Add user to the mpc group we created, current user by default, others can be appended
    - name: Adding existing user "{{ item }}" to group mpc
      user:
        name: "{{ item }}"
        groups: mpc
        append: yes
      become: yes
      loop:
        - "{{ ansible_user }}"
```

Check SSH Key Forwarding

Do OS tasks

Load OS packages

Install OS packages

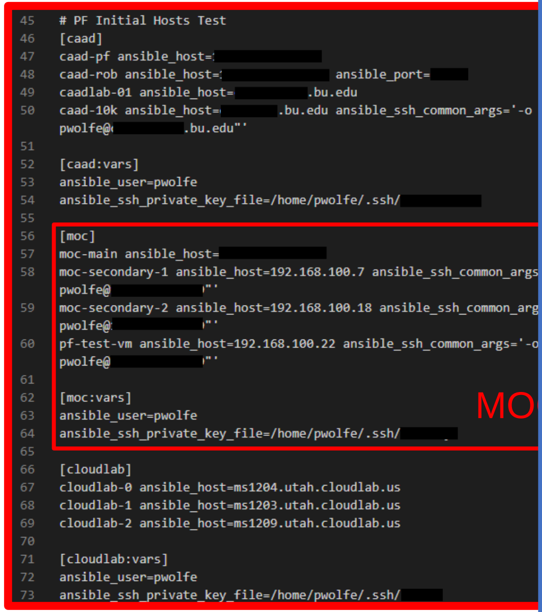
Do OS tasks

Create MPC user group

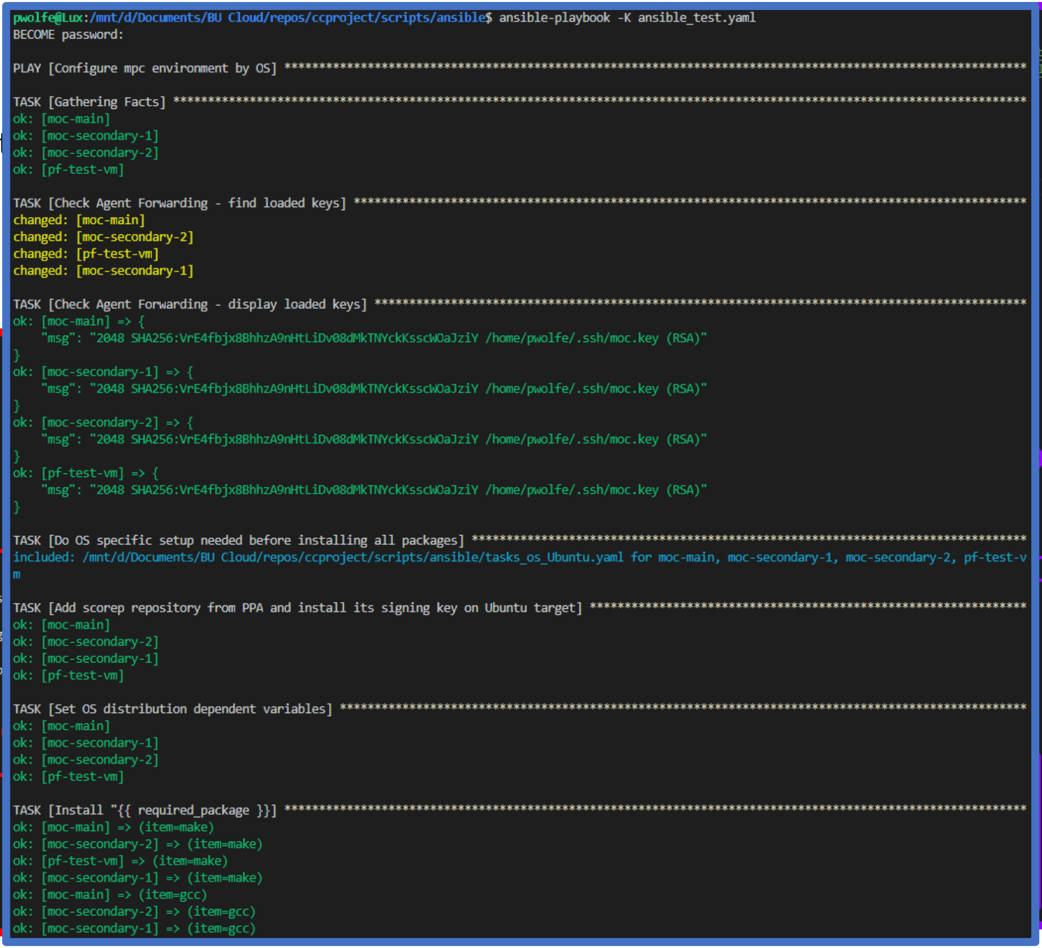
etc… (more tasks follow)

Ansible Detail

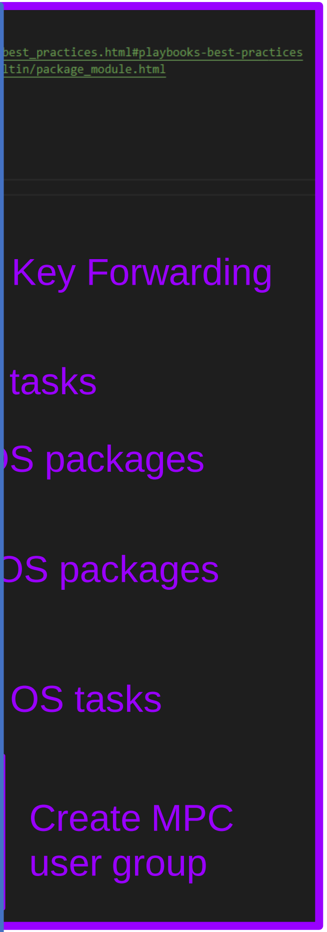
1. Install ansible: `pip3 inst`
2. Create inventory of hosts
3. Create a playbook
4. Execute the playbook



```
# PF Initial Hosts Test
[caad]
caad-pf ansible_host=[redacted]
caad-rob ansible_host=[redacted] ansible_port=[redacted]
caadlab-01 ansible_host=[redacted].bu.edu
caad-10k ansible_host=[redacted].bu.edu ansible_ssh_common_args='-o
pwolfe@[redacted].bu.edu"'

[caad:vars]
ansible_user=pwolfe
ansible_ssh_private_key_file=/home/pwolfe/.ssh/[redacted]

[moc]
moc-main ansible_host=[redacted]
moc-secondary-1 ansible_host=192.168.100.7 ansible_ssh_common_args
pwolfe@[redacted]"'
moc-secondary-2 ansible_host=192.168.100.18 ansible_ssh_common_arg
pwolfe@[redacted]"'
pf-test-vm ansible_host=192.168.100.22 ansible_ssh_common_args='-o
pwolfe@[redacted]"'

[moc:vars]
ansible_user=pwolfe
ansible_ssh_private_key_file=/home/pwolfe/.ssh/[redacted]

[cloudlab]
cloudlab-0 ansible_host=ms1204.utah.cloudlab.us
cloudlab-1 ansible_host=ms1203.utah.cloudlab.us
cloudlab-2 ansible_host=ms1209.utah.cloudlab.us

[cloudlab:vars]
ansible_user=pwolfe
ansible_ssh_private_key_file=/home/pwolfe/.ssh/[redacted]
```



```
pwolfe@Lux:/mnt/d/Documents/BU Cloud/repos/ccproject/scripts/ansible$ ansible-playbook -K ansible_test.yaml
BECOME password:

PLAY [Configure mpc environment by OS] ***********************************************************************

TASK [Gathering Facts] ***************************************************************************************
ok: [moc-main]
ok: [moc-secondary-1]
ok: [moc-secondary-2]
ok: [pf-test-vm]

TASK [Check Agent Forwarding - find loaded keys] *************************************************************
changed: [moc-main]
changed: [moc-secondary-2]
changed: [pf-test-vm]
changed: [moc-secondary-1]

TASK [Check Agent Forwarding - display loaded keys] **********************************************************
ok: [moc-main] => {
    "msg": "2048 SHA256:VrE4fbjx8BhhzA9nHtLiDv08dMkTNYckKsscWOaJziY /home/pwolfe/.ssh/moc.key (RSA)"
}
ok: [moc-secondary-1] => {
    "msg": "2048 SHA256:VrE4fbjx8BhhzA9nHtLiDv08dMkTNYckKsscWOaJziY /home/pwolfe/.ssh/moc.key (RSA)"
}
ok: [moc-secondary-2] => {
    "msg": "2048 SHA256:VrE4fbjx8BhhzA9nHtLiDv08dMkTNYckKsscWOaJziY /home/pwolfe/.ssh/moc.key (RSA)"
}
ok: [pf-test-vm] => {
    "msg": "2048 SHA256:VrE4fbjx8BhhzA9nHtLiDv08dMkTNYckKsscWOaJziY /home/pwolfe/.ssh/moc.key (RSA)"
}

TASK [Do OS specific setup needed before installing all packages] ********************************************
included: /mnt/d/Documents/BU Cloud/repos/ccproject/scripts/ansible/tasks_os_Ubuntu.yaml for moc-main, moc-secondary-1, moc-secondary-2, pf-test-v
m

TASK [Add scorep repository from PPA and install its signing key on Ubuntu target] ***************************
ok: [moc-main]
ok: [moc-secondary-2]
ok: [moc-secondary-1]
ok: [pf-test-vm]

TASK [Set OS distribution dependent variables] ***************************************************************
ok: [moc-main]
ok: [moc-secondary-1]
ok: [moc-secondary-2]
ok: [pf-test-vm]

TASK [Install "{{ required_package }}] ***********************************************************************
ok: [moc-main] => (item=make)
ok: [moc-secondary-2] => (item=make)
ok: [pf-test-vm] => (item=make)
ok: [moc-secondary-1] => (item=make)
ok: [moc-main] => (item=gcc)
ok: [moc-secondary-2] => (item=gcc)
ok: [moc-secondary-1] => (item=gcc)
```



```
est_practices.html#playbooks-best-practices
ltin/package_module.html
```

Key Forwarding

tasks

S packages

S packages

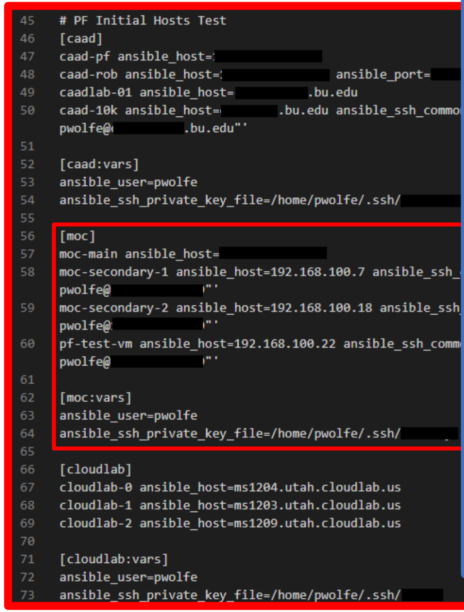
OS tasks

Create MPC user group

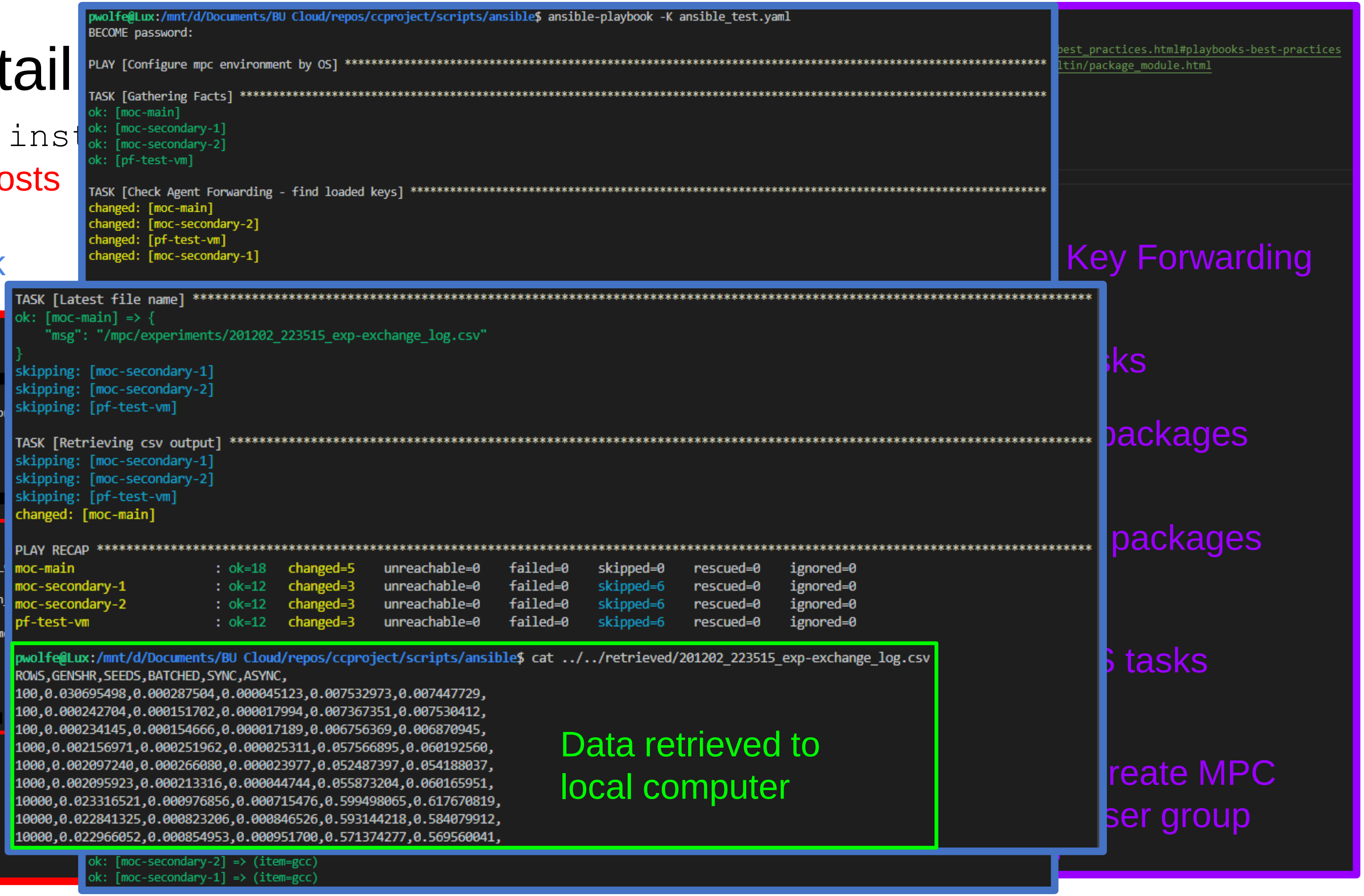
etc… (more tasks follow)

Ansible Detail

1. Install ansible: `pip3 inst`
2. Create inventory of hosts
3. Create a playbook
4. Execute the playbook



```
# PF Initial Hosts Test
[caad]
caad-pf ansible_host=[redacted]
caad-rob ansible_host=[redacted] ansible_port=[redacted]
caadlab-01 ansible_host=[redacted].bu.edu
caad-10k ansible_host=[redacted].bu.edu ansible_ssh_commo
pwolfe@[redacted].bu.edu"'

[caad:vars]
ansible_user=pwolfe
ansible_ssh_private_key_file=/home/pwolfe/.ssh/[redacted]

[moc]
moc-main ansible_host=[redacted]
moc-secondary-1 ansible_host=192.168.100.7 ansible_ssh_
pwolfe@[redacted]"'
moc-secondary-2 ansible_host=192.168.100.18 ansible_ssh
pwolfe@[redacted]"'
pf-test-vm ansible_host=192.168.100.22 ansible_ssh_comm
pwolfe@[redacted]"'

[moc:vars]
ansible_user=pwolfe
ansible_ssh_private_key_file=/home/pwolfe/.ssh/[redacted]

[cloudlab]
cloudlab-0 ansible_host=ms1204.utah.cloudlab.us
cloudlab-1 ansible_host=ms1203.utah.cloudlab.us
cloudlab-2 ansible_host=ms1209.utah.cloudlab.us

[cloudlab:vars]
ansible_user=pwolfe
ansible_ssh_private_key_file=/home/pwolfe/.ssh/[redacted]
```



```
pwolfe@Lux:/mnt/d/Documents/BU Cloud/repos/ccproject/scripts/ansible$ ansible-playbook -K ansible_test.yaml
BECOME password:

PLAY [Configure mpc environment by OS] *****************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [moc-main]
ok: [moc-secondary-1]
ok: [moc-secondary-2]
ok: [pf-test-vm]

TASK [Check Agent Forwarding - find loaded keys] *******************************************************************************
changed: [moc-main]
changed: [moc-secondary-2]
changed: [pf-test-vm]
changed: [moc-secondary-1]
```

```
TASK [Latest file name] ********************************************************************************************************
ok: [moc-main] => {
    "msg": "/mpc/experiments/201202_223515_exp-exchange_log.csv"
}
skipping: [moc-secondary-1]
skipping: [moc-secondary-2]
skipping: [pf-test-vm]

TASK [Retrieving csv output] ***************************************************************************************************
skipping: [moc-secondary-1]
skipping: [moc-secondary-2]
skipping: [pf-test-vm]
changed: [moc-main]

PLAY RECAP *********************************************************************************************************************
moc-main                   : ok=18   changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
moc-secondary-1            : ok=12   changed=3    unreachable=0    failed=0    skipped=6    rescued=0    ignored=0
moc-secondary-2            : ok=12   changed=3    unreachable=0    failed=0    skipped=6    rescued=0    ignored=0
pf-test-vm                 : ok=12   changed=3    unreachable=0    failed=0    skipped=6    rescued=0    ignored=0
```

```
pwolfe@Lux:/mnt/d/Documents/BU Cloud/repos/ccproject/scripts/ansible$ cat ../../retrieved/201202_223515_exp-exchange_log.csv
ROWS,GENSHR,SEEDS,BATCHED,SYNC,ASYNC,
100,0.030695498,0.000287504,0.000045123,0.007532973,0.007447729,
100,0.000242704,0.000151702,0.000017994,0.007367351,0.007530412,
100,0.000234145,0.000154666,0.000017189,0.006756369,0.006870945,
1000,0.002156971,0.000251962,0.000025311,0.057566895,0.060192560,
1000,0.002097240,0.000266080,0.000023977,0.052487397,0.054188037,
1000,0.002095923,0.000213316,0.000044744,0.055873204,0.060165951,
10000,0.023316521,0.000976856,0.000715476,0.599498065,0.617670819,
10000,0.022841325,0.000823206,0.000846526,0.593144218,0.584079912,
10000,0.022966052,0.000854953,0.000951700,0.571374277,0.569560041,
```

Data retrieved to local computer

```
ok: [moc-secondary-2] => (item=gcc)
ok: [moc-secondary-1] => (item=gcc)
```

```
est_practices.html#playbooks-best-practices
ltin/package_module.html
```

Key Forwarding

ks

packages

packages

S tasks

reate MPC
ser group

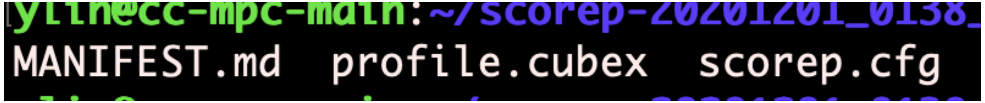
etc… (more tasks follow)

Benchmarking/Profiling and Analysis

- Score-P
  - Framework that allows for data collection from MPI (and other sources)
  - Was able to be used in conjunction with ansible (modify Makefile and run the playbook)
- Analysis
  - CUBE GUI: Inspecting *.cubex benchmark file
  - Others? (What about the trace data? Any insights when testing different parts of a program?)

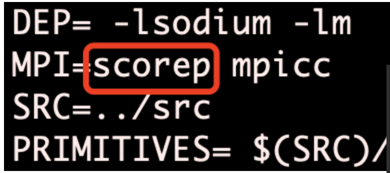
Running MPI with Score-P

- Issues before running:
  - Extra dependency needed on ubuntu
  - Need to correct permissions (sudo chown/chmod)

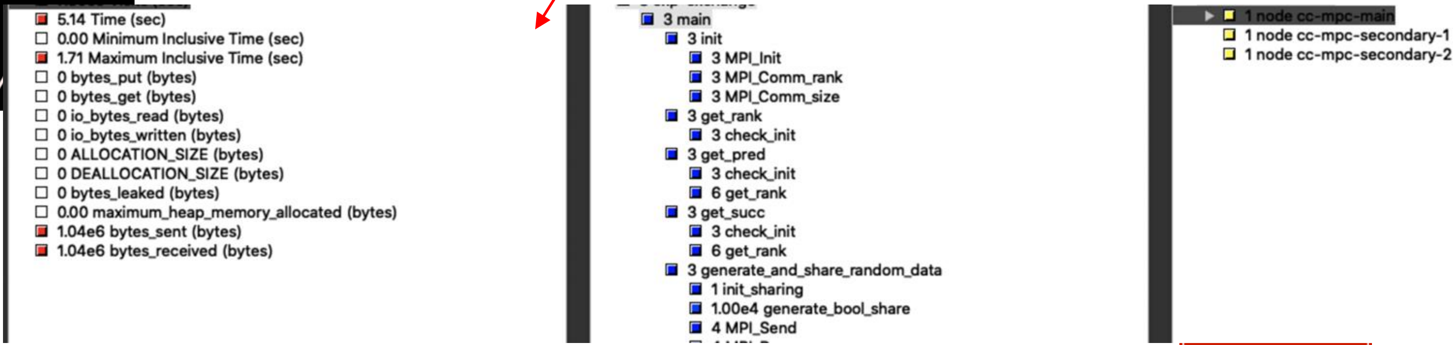


```
yline@cc-mpc-main:~/scorep-20201201_0138_
MANIFEST.md   profile.cubex   scorep.cfg
```

Add scorep prefix in makefile



```
DEP= -lsodium -lm
MPI=scorep mpicc
SRC=../src
PRIMITIVES= $(SRC)/
```



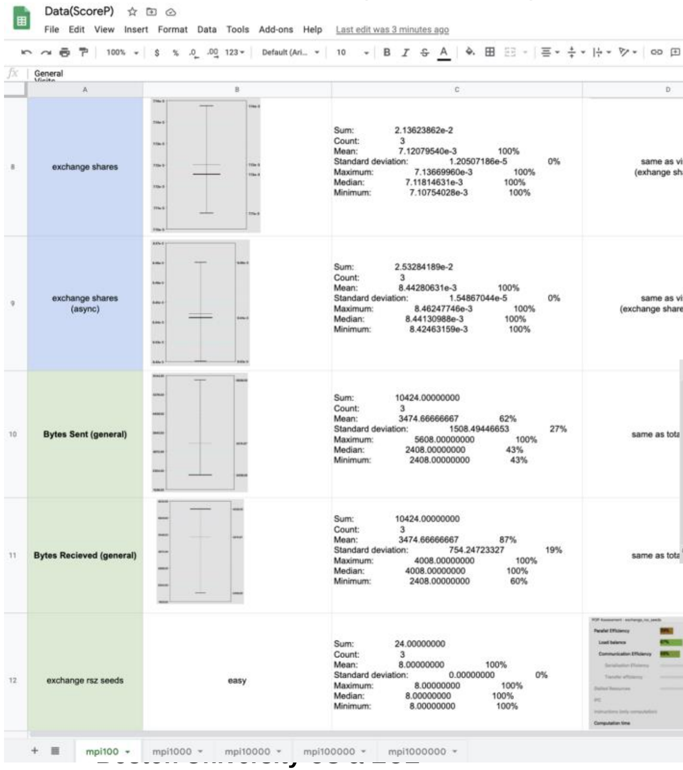
```
■ 5.14 Time (sec)
□ 0.00 Minimum Inclusive Time (sec)
■ 1.71 Maximum Inclusive Time (sec)
□ 0 bytes_put (bytes)
□ 0 bytes_get (bytes)
□ 0 io_bytes_read (bytes)
□ 0 io_bytes_written (bytes)
□ 0 ALLOCATION_SIZE (bytes)
□ 0 DEALLOCATION_SIZE (bytes)
□ 0 bytes_leaked (bytes)
□ 0.00 maximum_heap_memory_allocated (bytes)
■ 1.04e6 bytes_sent (bytes)
■ 1.04e6 bytes_received (bytes)
```

```
■ 3 main
  ■ 3 init
    ■ 3 MPI_Init
    ■ 3 MPI_Comm_rank
    ■ 3 MPI_Comm_size
  ■ 3 get_rank
    ■ 3 check_init
  ■ 3 get_pred
    ■ 3 check_init
    ■ 6 get_rank
  ■ 3 get_succ
    ■ 3 check_init
    ■ 6 get_rank
  ■ 3 generate_and_share_random_data
    ■ 1 init_sharing
    ■ 1.00e4 generate_bool_share
    ■ 4 MPI_Send
```

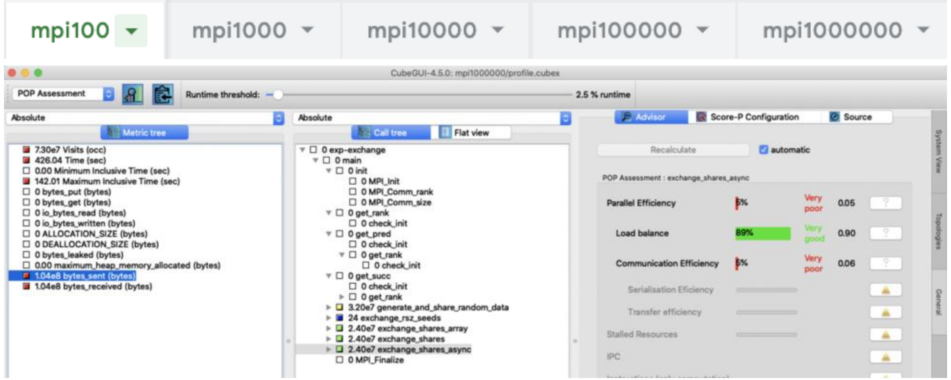
```
▶ □ 1 node cc-mpc-main
  □ 1 node cc-mpc-secondary-1
  □ 1 node cc-mpc-secondary-2
```

Scalasca to evaluate cubex file

Data Analysis by CUBE


Data(ScoreP)
File Edit View Insert Format Data Tools Add-ons Help
Last edit was 3 minutes ago
100% $ % .0 .00 123 Default (Ari... 10 B I S A
General
Visits

	A	B	C	D
8	exchange shares		Sum: 2.13623862e-2 Count: 3 Mean: 7.12079540e-3 100% Standard deviation: 1.20507186e-5 0% Maximum: 7.13669960e-3 100% Median: 7.11814631e-3 100% Minimum: 7.10754028e-3 100%	same as visits (exhange shares)
9	exchange shares (async)		Sum: 2.53284189e-2 Count: 3 Mean: 8.44280631e-3 100% Standard deviation: 1.54867044e-5 0% Maximum: 8.46247746e-3 100% Median: 8.44130988e-3 100% Minimum: 8.42463159e-3 100%	same as visits (exchange shares async)
10	**Bytes Sent (general)**		Sum: 10424.00000000 Count: 3 Mean: 3474.66666667 62% Standard deviation: 1508.49446653 27% Maximum: 5608.00000000 100% Median: 2408.00000000 43% Minimum: 2408.00000000 43%	same as tota
11	**Bytes Recieved (general)**		Sum: 10424.00000000 Count: 3 Mean: 3474.66666667 87% Standard deviation: 754.24723327 19% Maximum: 4008.00000000 100% Median: 4008.00000000 100% Minimum: 2408.00000000 60%	same as tota
12	exchange rsz seeds	easy	Sum: 24.00000000 Count: 3 Mean: 8.00000000 100% Standard deviation: 0.00000000 0% Maximum: 8.00000000 100% Median: 8.00000000 100% Minimum: 8.00000000 100%	



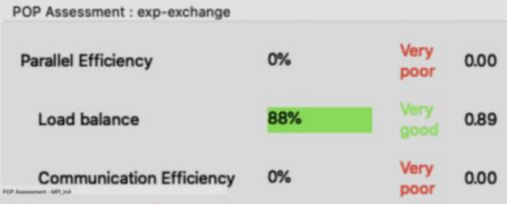
☑ automatic

```
■ 7.30e7 Visits (occ)
■ 426.04 Time (sec)
□ 0.00 Minimum Inclusive Time (sec)
■ 142.01 Maximum Inclusive Time (sec)
□ 0 bytes_put (bytes)
□ 0 bytes_get (bytes)
□ 0 io_bytes_read (bytes)
□ 0 io_bytes_written (bytes)
□ 0 ALLOCATION_SIZE (bytes)
□ 0 DEALLOCATION_SIZE (bytes)
□ 0 bytes_leaked (bytes)
□ 0.00 maximum_heap_memory_allocated (bytes)
■ 1.04e8 bytes_sent (bytes)
■ 1.04e8 bytes_received (bytes)
```

```
▼ □ 0 exp-exchange
  ▼ □ 0 main
    ▼ □ 0 init
        □ 0 MPI_Init
        □ 0 MPI_Comm_rank
        □ 0 MPI_Comm_size
    ▼ □ 0 get_rank
        □ 0 check_init
    ▼ □ 0 get_pred
        □ 0 check_init
      ▼ □ 0 get_rank
          □ 0 check_init
    ▼ □ 0 get_succ
        □ 0 check_init
      ▶ □ 0 get_rank
    ▶ ■ 3.20e7 generate_and_share_random_data
    ▶ ■ 24 exchange_rsz_seeds
    ▶ ■ 2.40e7 exchange_shares_array
    ▶ ■ 2.40e7 exchange_shares
    ▶ ■ 2.40e7 exchange_shares_async
      □ 0 MPI_Finalize
```

POP Assessment : exchange_shares_async

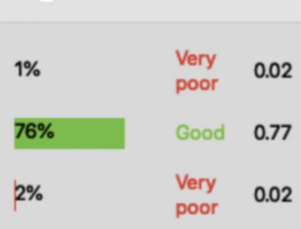
Parallel Efficiency	5%	Very poor	0.05	?
Load balance	89%	Very good	0.90	?
Communication Efficiency	5%	Very poor	0.06	?
Serialisation Eficiency				⚠
Transfer efficiency				⚠
Stalled Resources				⚠
IPC				⚠



POP Assessment : exp-exchange

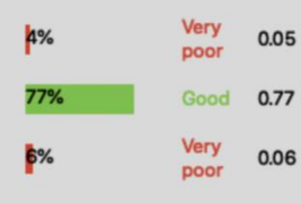
Parallel Efficiency	0%	Very poor	0.00
Load balance	88%	Very good	0.89
Communication Efficiency	0%	Very poor	0.00

100



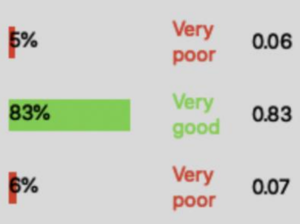
1%	Very poor	0.02
76%	Good	0.77
2%	Very poor	0.02

1000



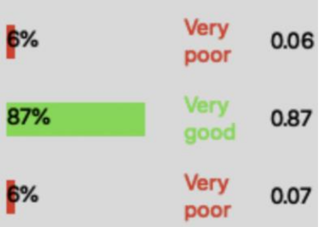
4%	Very poor	0.05
77%	Good	0.77
6%	Very poor	0.06

10000



5%	Very poor	0.06
83%	Very good	0.83
6%	Very poor	0.07

100000

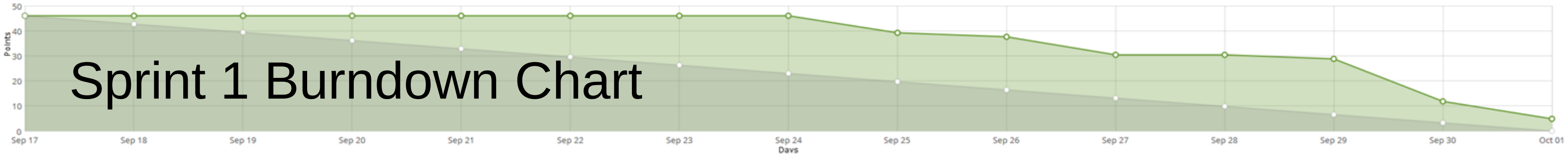


6%	Very poor	0.06
87%	Very good	0.87
6%	Very poor	0.07

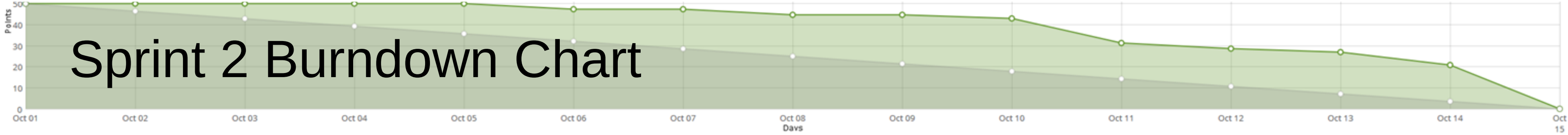
1000000

BOSTON UNIVERSITY

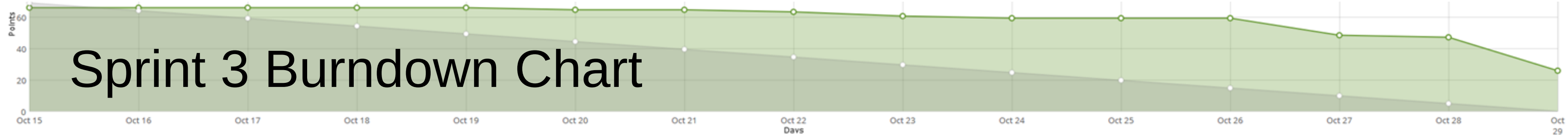
Sprint Demo



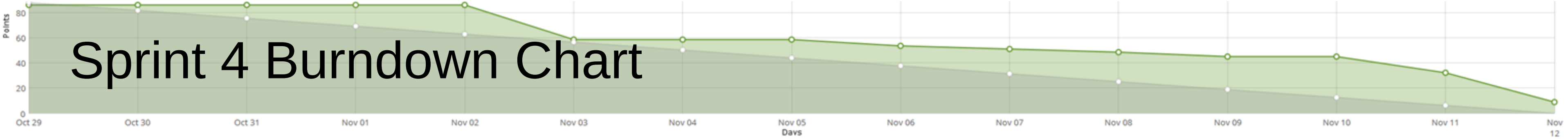
Sprint 1 Burndown Chart



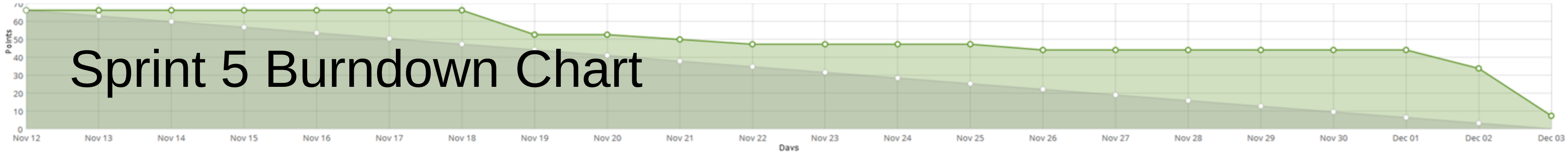
Sprint 2 Burndown Chart



Sprint 3 Burndown Chart



Sprint 4 Burndown Chart



Sprint 5 Burndown Chart

Final Project Details & Mentor/Client Future Efforts

- Final Details
  - Ansible tweaks
  - Documentation organization
  - Handoff meeting with mentors/clients
- Mentor Plans
  - Use framework for new, repeatable experiment
  - Build a frontend interface for other users (leverage our setup work)
  - Use our documentation and pointers for additional development/exploration

Thank you

...any questions?