

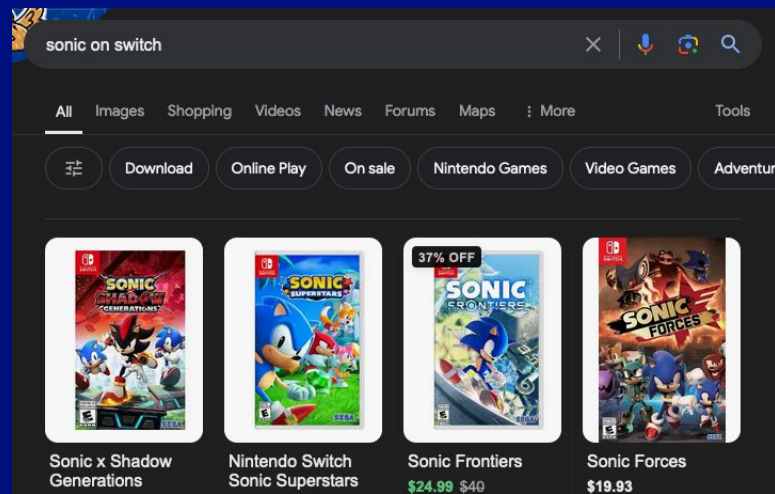
Cluster Management with Containerization on Switches

HPC-DO

Robin Lee Simpson
Anvitha Ramachandran
Dohyun Lee

Switches? SONiC? What are they?

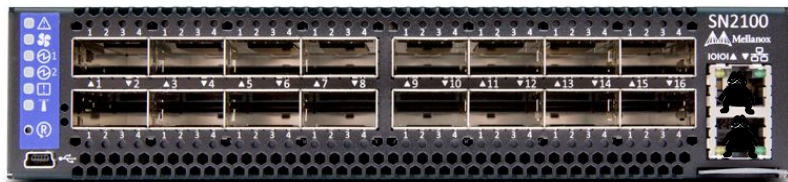
- Routes packets between nodes
- Software for Open Networking in the Cloud
 - Open Source Operating System
 - Mellanox, Arista
- Often underutilized
 - What are switches doing in the meantime?
 - CPU vs ASIC



Uncovering Untapped Potential

- Deploy containers on switches via podman and docker
- Leverage the cpus on the switches via SONiC with auxiliary tasks:
 - Run Cloud-Init services on the switch
 - Utilize Telegraf and Grafana to aggregate logs and metrics
 - Initialize a proxy server for S3 Storage
 - Create an IPv6 DHCP and DNS Provider
 - Implement an automatic client discovery and configuration

Testing and Methodology



Physical Switch

- SN2100 w/ Intel x86 2.40 GHz Quad Core, 8 GB DDR3L, 16GB SATA
- Connected to 9 Compute Nodes and 1 Head Node Cluster running on Intel 6438Y+



Virtual Emulation

- QEMU VM
- Quad Core
- Set up via Ansible and Libvirt

Why the VM?

- All scenarios are running on the switch
 - If one breaks it all breaks
- Fast testing and unlimited modifications
 - No need to go downstairs to reimage the switches or nodes
- Uniformity/Reproducibility
- Fun Fact: We broke the nodes and the switches several times

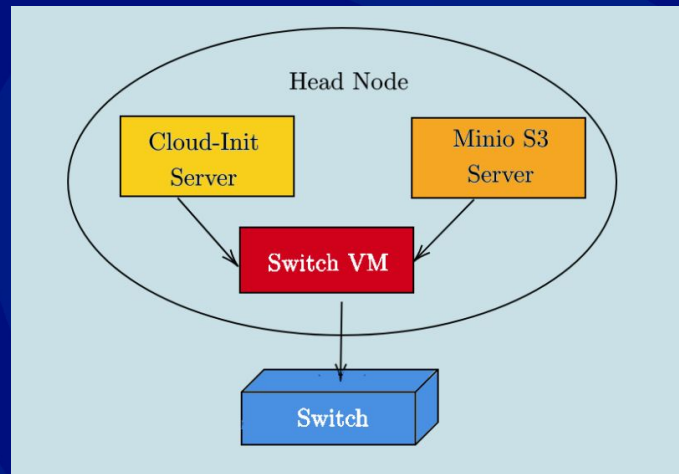
```
- name: sonic-vm
  memory: 16777216
  cpus: '4'
  arch: 'x86_64'
  interfaces:
    - type: 'network'
      network: 'ochami-mgmt'
      mac: '02:00:00:28:b7:61'
      model_type: 'virtio'
    - type: 'network'
      network: 'ochami-svc'
      mac: '02:00:00:28:b7:71'
      model_type: 'virtio'

libvirt_nets:
- name: ochami-mgmt
  autostart: true
  forward_mode: 'nat'
  bridge_name: 'br-ochami-mgmt'
  mac_addr: '02:00:00:ff:20:d7'
  ipaddr: '10.1.0.1'
  netmask: '255.255.255.0'
  ip_start: '10.1.0.2'
  ip_end: '10.1.0.254'
  hosts:
    - name: 'ochami-vm'
      mac: '02:00:00:a3:46:70'
      ipaddr: '10.1.0.2'
    - name: 'sonic-vm'
      mac: '02:00:00:28:b7:61'
      ipaddr: '10.1.0.3'
  booturl: 'http://10.100.0.1:9000/efi/BOOTX64.EFI'
```

Summer 2024 LA-UR-24-28424

Scenario 1: Running Cloud-Init Services

- Utilized MinioS3 as an object store
 - Store config payload
- Repurposed existing containers and networks from Supercomputing Institute Bootcamp
 - Why create new containers and instances?
- Virtual:
 - Mount the image
 - Modify the grub.cfg to point to cloud-init server
- Physical:
 - SSH or access the switch
 - Find where grub.cfg is
 - Modify grub.cfg to point to cloud-init server

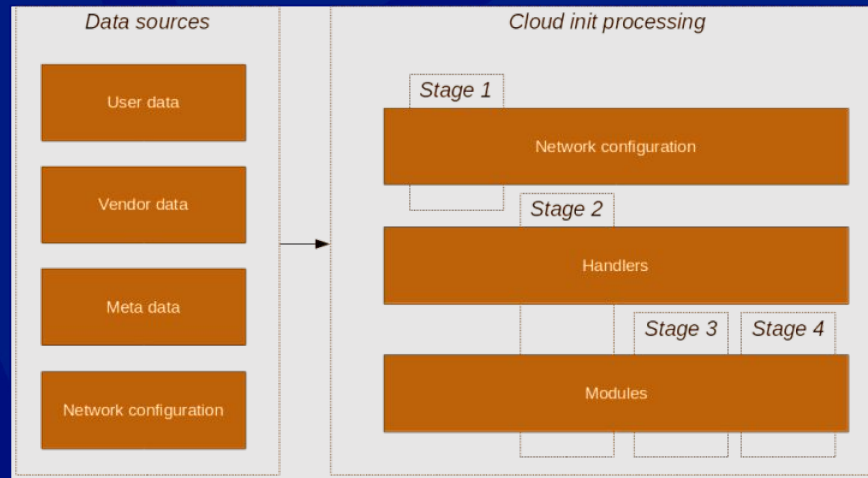
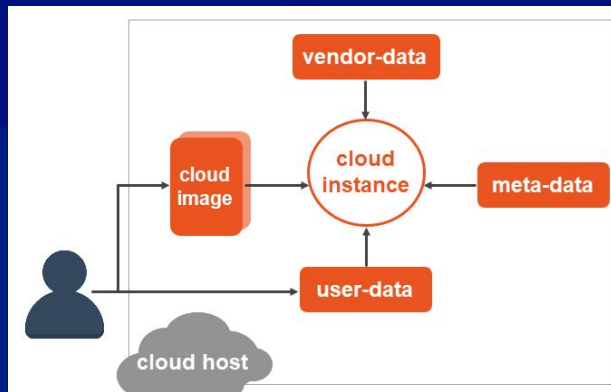


```
linux /image-master.580196-1abd0deae/boot/vmlinuz
→ -6.1.0-11-2-amd64 root=UUID=3f88f0f9-5902-4dc5-a606
→ -44767bd38318 rw console=tty0 console=ttyS0,115200n8
→ quiet processor.max_cstate=1 intel_idle.max_cstate=0
→ net.ifnames=0 biosdevname=0
→ loop=image-master.580196-1abd0deae/fs
→ .squashfs loopfstype=squashfs
→                                     systemd.
→ unified_cgroup_hierarchy=0          apparmor=1
→ security=apparmor varlog_size=4096 usbcore.
→ autosuspend=-1 swiotlb=65536
```

Listing 2: Grub Config

Scenario 1: What and Why?

- Automates the setup process.
 - Configure network settings (Vlans, routing protocols, security settings)
 - Assign hostnames
 - Install software and apply updates
 - Custom scripts
- Not limited to physical systems
- Runs during first boot of an instance
- Extensible with custom modules and plugins
- `ds=nocloud-net;s=http://<CLOUD_INIT_SERVER>` in `grub.cfg`



Scenario 1: Virtual Testing

```
1 #cloud-config
2 packages:
3   - python3
4   - python3-pip
5
6 runcmd:
7   - python3 /root/get.py --bucket-name sonic --key-name test.
8     ↪ txt --output-file /home/admin/cloud-init-test.txt
9
10 final_message: "System up after $UPTIME seconds"
```

Simple Configuration to test
get from Minio

```
Linux sonic-vm 6.1.0-11-2-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.38-4 (2023-08-08) x86_64
Cloud Init Test
Last login: Wed Jul 17 21:43:48 2024
admin@sonic-vm:~$ ls
aws  cache_out  cache_test  cloud-init-test.txt  dummy-interface.sh  ex.txt  go  scenario1  test_mount
admin@sonic-vm:~$
```

Successful post-boot get from Minio

Scenario 1: Physical Testing

```
Skibidi

#cloud-config
runcmd:
  - echo "Booted Cloud" > /etc/motd
  - touch /root/dab.txt

final_message: "System up after $UPTIME seconds"
```

Simple Config

```
[ 16.976639] rc.local[1074]: + onie_version=2018.05-5.2.0004-115200
[ 16.992547] rc.local[1074]: + program_console_speed
[ 17.012069] rc.local[1121]: + grep -Eo console=ttty(S|AMA)[0-9]+,[0-9]+
[ 17.022952] rc.local[1120]: + cat /proc/cmdline
[ 17.037879] rc.local[1122]: + cut -d , -f2
[ 17.052765] rc.local[1074]: + speed=115200
[ 17.068378] rc.local[1074]: + [ -z 115200 ]
[ 17.080463] rc.local[1074]: + CONSOLE_SPEED=115200
[ 17.099279] rc.local[1126]: + grep keep-baud
[ 17.117658] rc.local[1125]: + grep agetty /lib/systemd/system/serial-getty@.service
[ 17.137800] rc.local[1126]: ExecStart=-/sbin/agetty -o '-p -- \\u' --keep-baud 115200,57600,38400,9600 - $TERM
[ 17.161367] rc.local[1074]: + [ 0 = 0 ]
[ 17.176722] rc.local[1074]: + sed -i s|\\-keep-\\baud .* %I| 115200 %I|g /lib/systemd/system/serial-getty@.service
[ 17.196647] rc.local[1074]: + systemctl daemon-reload
[ 18.010489] cloud-init[1185]: Cloud-init v. 22.4.2 running 'modules:config' at Tue, 23 Jul 2024 13:59:10 +0000. Up 17.91 seconds.
[ 18.164941] rc.local[1074]: + [ -f /host/image-master.592546-c2055abbc/platform/firsttime ]
[ 18.184333] rc.local[1074]: + [ -f /var/log/fsck.log.gz ]
[ 18.201042] rc.local[1187]: + gunzip -d -c /var/log/fsck.log.gz
[ 18.220653] rc.local[1188]: + logger -t FSCK
[ 18.237163] rc.local[1074]: + rm -f /var/log/fsck.log.gz
[ 18.256248] rc.local[1074]: + exit 0

Debian GNU/Linux 12 sonic ttyS0

sonic login:
```

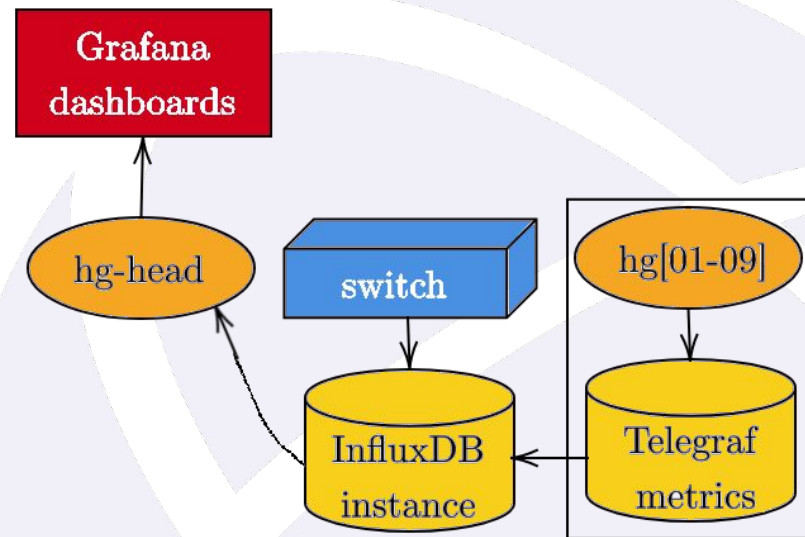
Cloud-Init Running

```
Linux sonic 6.1.0-11-2-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.38-4 (2023-08-08) x86_64
Booted Cloud
Last login: Mon Jul 22 21:16:43 UTC 2024 on ttyS0
```

Post-boot

Scenario 2: Metric Collection

- Objective: Create cluster monitoring system
- Collect information about node health from a Telegraf service running on the node
- Run InfluxDB instance on the switch
- Display metrics on a Grafana dashboard

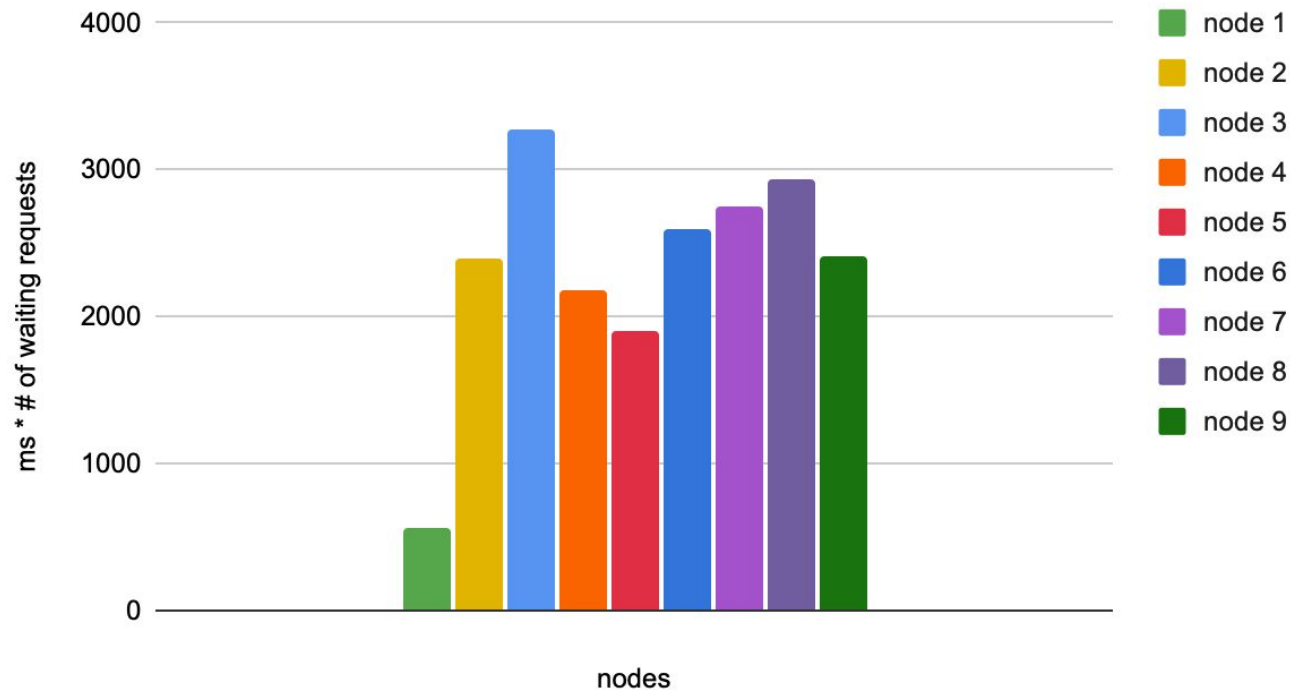


```
root@df51fca458b0:/# du -sh /var/lib/influxdb2/engine/data
233M    /var/lib/influxdb2/engine/data
```

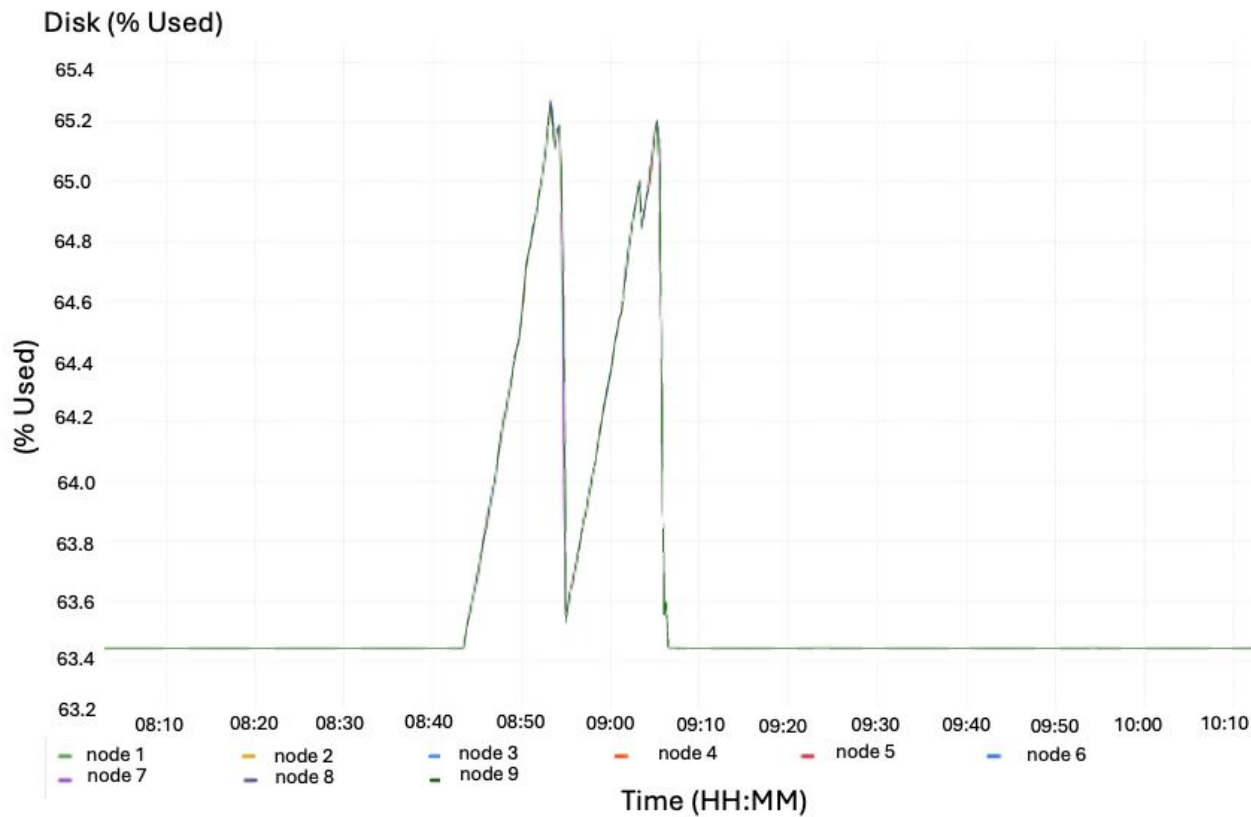
Disk usage for the metrics storage
(from 7/30/2024 - 8/5/2024)

Scenario 2 Plots – Weighted I/O Calls

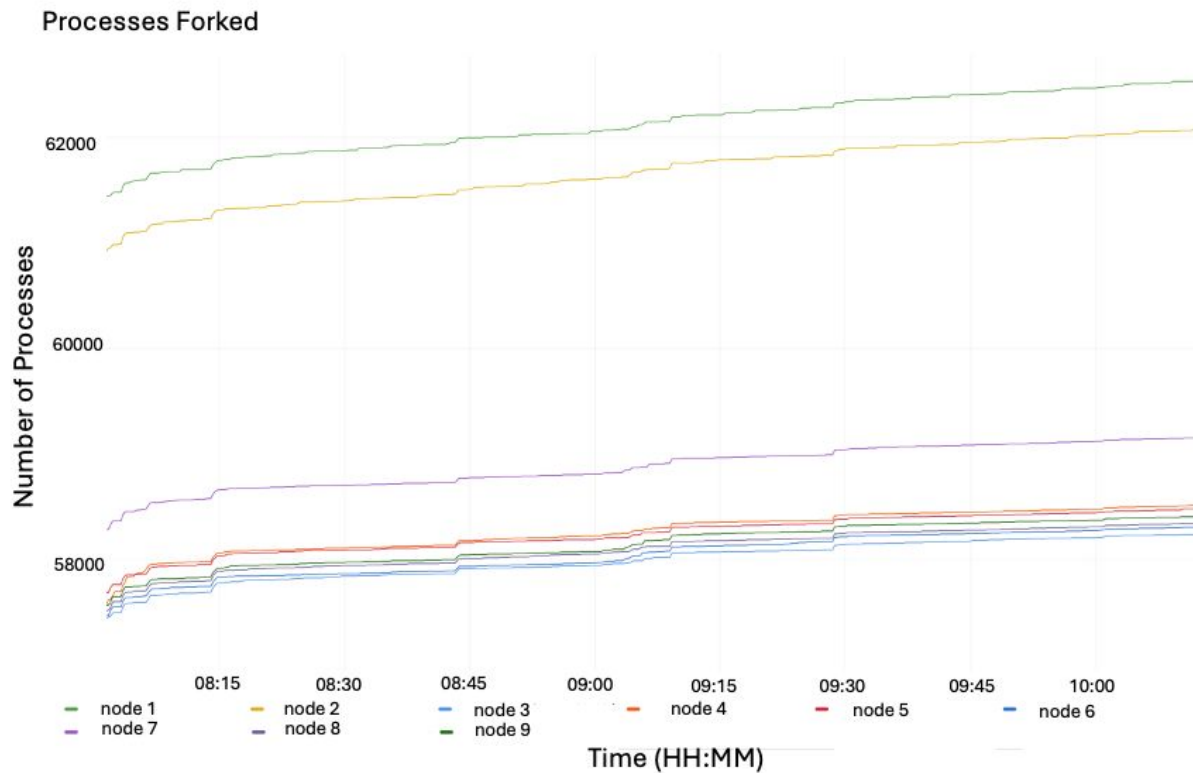
Weighted I/O (mean)



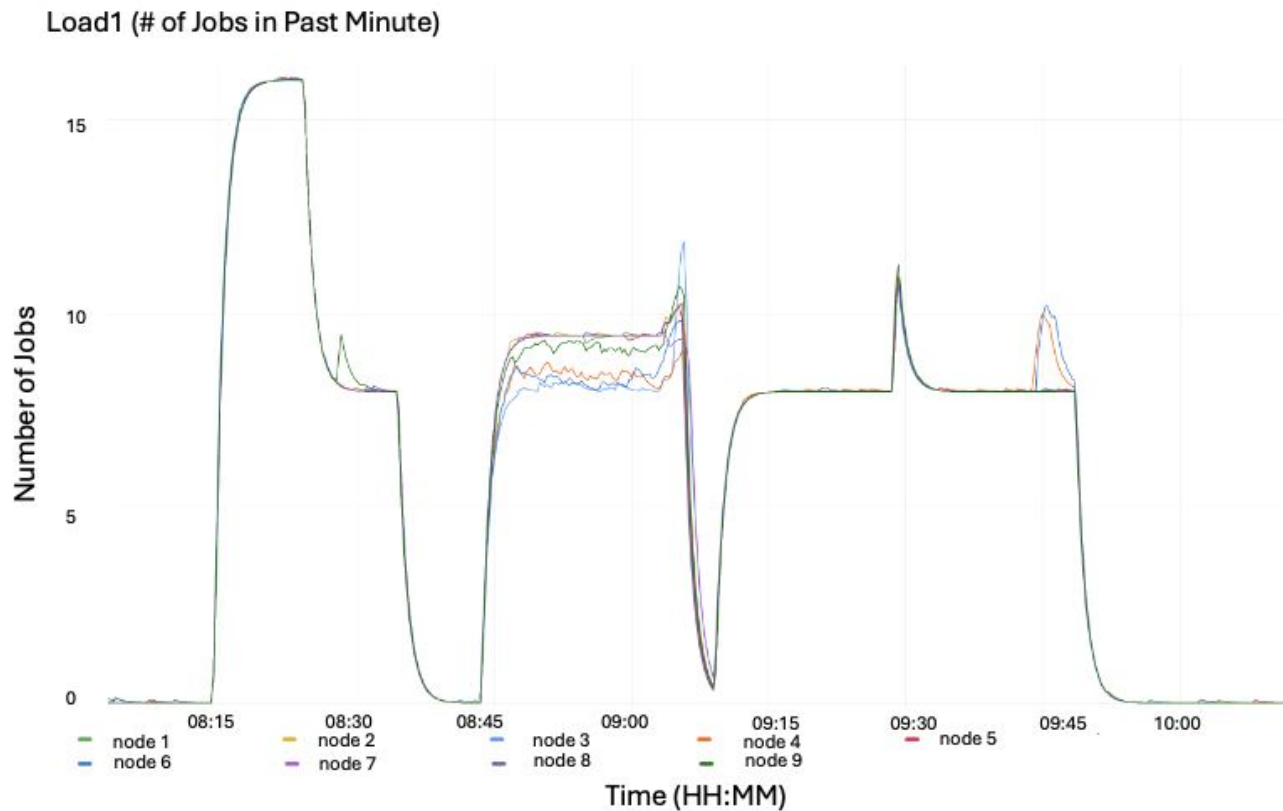
Scenario 2 Plots – Disk Usage



Scenario 2 Plots – Processes Forked

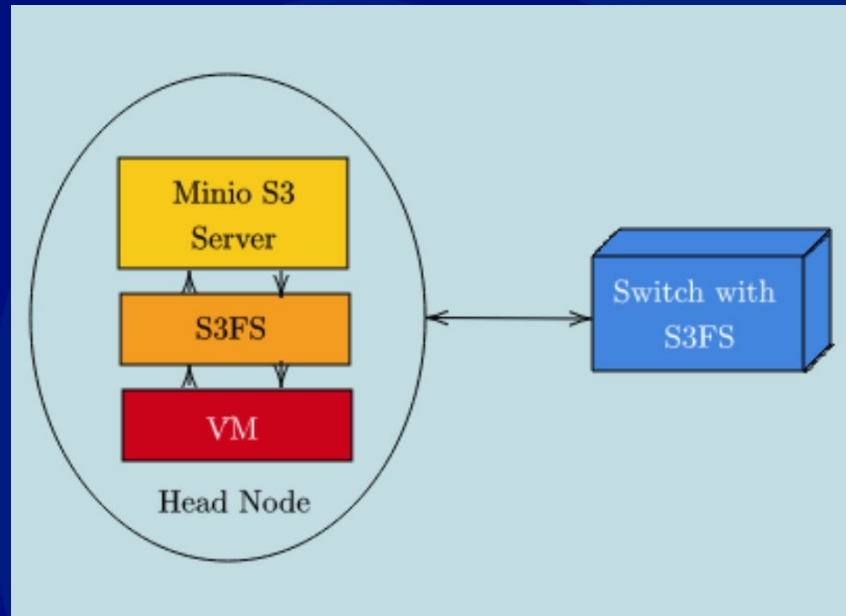


Scenario 2 Plots – Load1



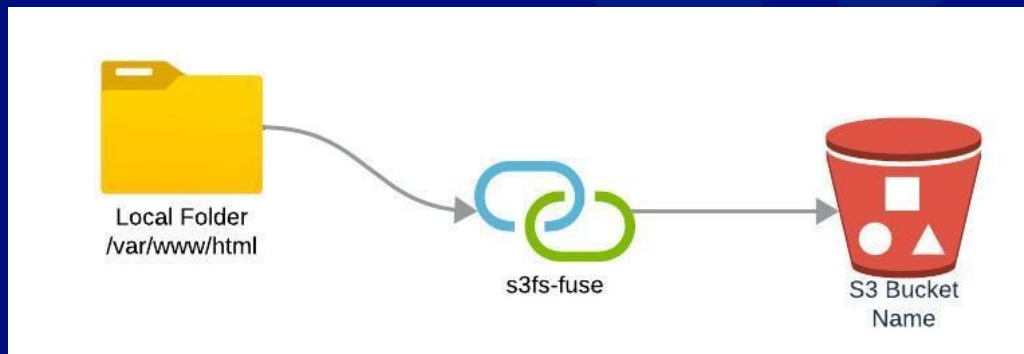
Scenario 3: Proxy Caching

- Utilized the same MinioS3 Storage
- Originally Versity was planned, unsupported, switched to S3FS
- Mounts to local FS, any changes is reflected to S3 store
- Frequently modified and accessed is cached
- Reduces latency



Scenario 3: Proxy Caching

- **-o use_cache:** Specifies the directory to use for local cache storage.
- **-o cache_max_size:** Limits the maximum size of the local cache.
- **-o cache_timeout:** Sets the expiration time for cached data.
- **-o dbglevel=info:** Enables detailed logging, which can help in monitoring and troubleshooting caching behavior.



Scenario 3: Set Up

`s3fs <bucket_name> <dir/folder> -o passwd_file=<./passwd-s3fs> -o use_cache=<dir> -o url=<MINIO_IP>`

```
admin@sonic-vm:~$ aws --endpoint-url [REDACTED] s3 mb s3://cachetest/
make_bucket: cachetest
admin@sonic-vm:~$ aws --endpoint-url [REDACTED] s3 cp ex.txt s3://cachetest/ex.txt
upload: ./ex.txt to s3://cachetest/ex.txt
```

Uploading the File to Test

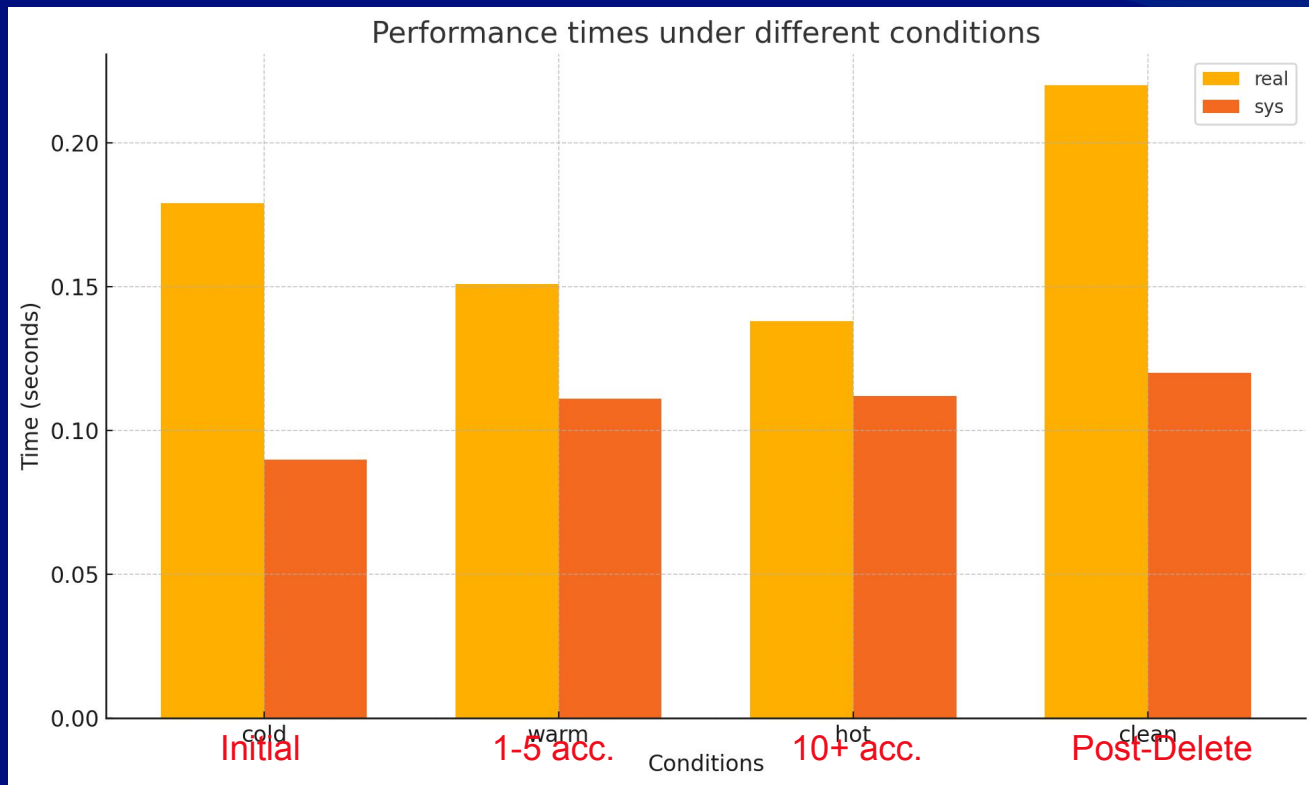
```
admin@sonic-vm:~/cache_test$ ls
ex.txt  test.txt
admin@sonic-vm:~/cache_test$
```

Local 'Copy'

```
admin@sonic-vm:~/cache_out/cachetest$ ls
ex.txt  test.txt
admin@sonic-vm:~/cache_out/cachetest$
```

Cached/Modified

Scenario 3: Caching Performance



100 Megabyte Test File

Scenario 4: IPv6 DNS/DHCP Provider

- Goal: Deploy a container on switch that provides **DHCP/DNS services** for IPv4/6 that allows for **reconfiguration on the fly**
- Why?
 - Human error causes the most downtime in cluster reconfiguration
 - IPv6 is here (maybe)



JAKE-CLARK.TUMBLR

Scenario 4: An Unfortunate Side Quest

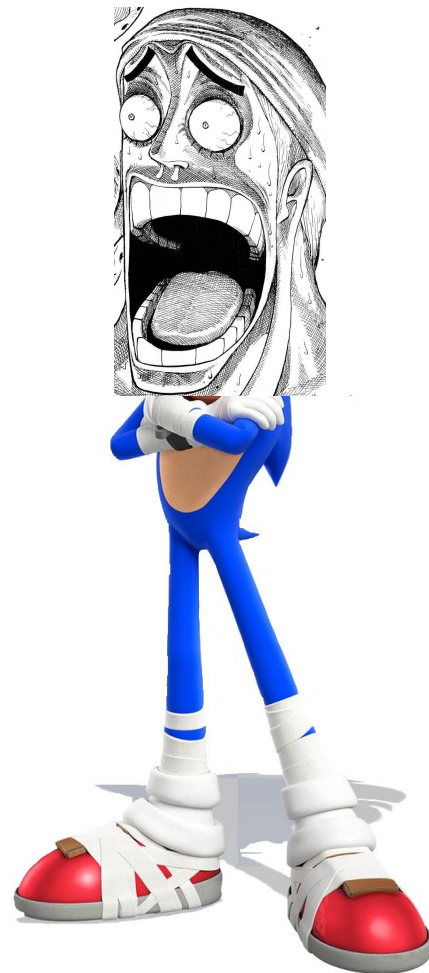
- We all want the most up-to-date tech
- Problem: Old tech doesn't always play well with new tech

SONiC

apt upgrade

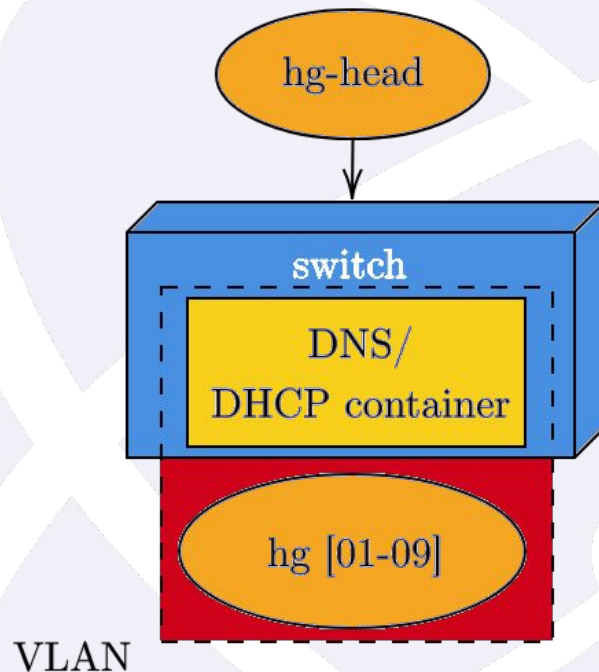


- Upgrading packages led to network issues for SONiC
- Created issues with Monit → more issues with packages
- Solution: Reimage the switch due to time constraints



Scenario 4: Back to DHCP/DNS, issues with dnsmasq

- Decided to use dnsmasq
- Why?
 - As a DHCP server, local hostnames are automatically added to cache
 - Uses less memory and CPU than ISC DHCP
 - Easy to install and run
- DHCP uses port 53, not 67
- dnsmasq doesn't start
 - `dnsmasq: failed to bind DHCP server socket: Address already in use`
- Potential issues with both DNS and DHCP using port 53



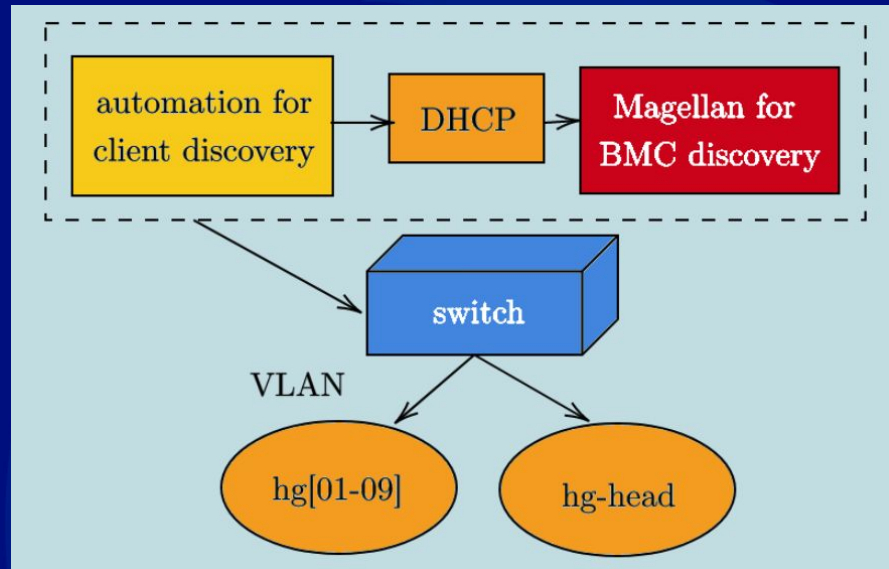
Scenario 4: Trying ISC DHCP, reconfiguration on the fly

- No port issues
- Problem: DHCP requests **reach the physical interfaces** on compute nodes, but **do not reach the VLAN interface**
 - Compute nodes are able to see DHCP requests
- Found similar issue in SONiC repository
 - Potential SONiC networking bug
- SIGHUP is used for reconfiguration on the fly of DHCP/DNS servers
- Tools like iNotify, etc, consul can be used for reconfiguration



Scenario 5: Node Discovery

- Automation to manage client devices in a VLAN
- Assigning IP addresses + initiating BMC discovery via Magellan
- Challenges: Same error regarding DHCP on SONiC switch causes issue with monitoring VLAN
- Can monitor non-virtual interfaces with Telegraf



Future Work

- Look at scenarios' impact on the switch
- Separate collection of switch metrics
- Working with an Arista switch to run the scenarios
- Explore more intensive scenarios to test switch limits
- Implementing more network management services on the switch



Thank you!

- Mentors – Doug Egan, Alex Lovell-Troy, David Rich
- SI instructors – Trevor and Devon Bautista, Shivam Mehta, John Dermer, Sakul Koirala
- Julie Wiens
- OpenCHAMI/Magellan help – David J. Allen