Building a High Availability NFS Server

Mentors: Michael Gilbert, David Fox, Martin Baltezore, Jason Shortino

August 11, 2021

Arshita Sandhiparthi
Emily Ramirez-Serrano

LLNL-PRES-825698
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

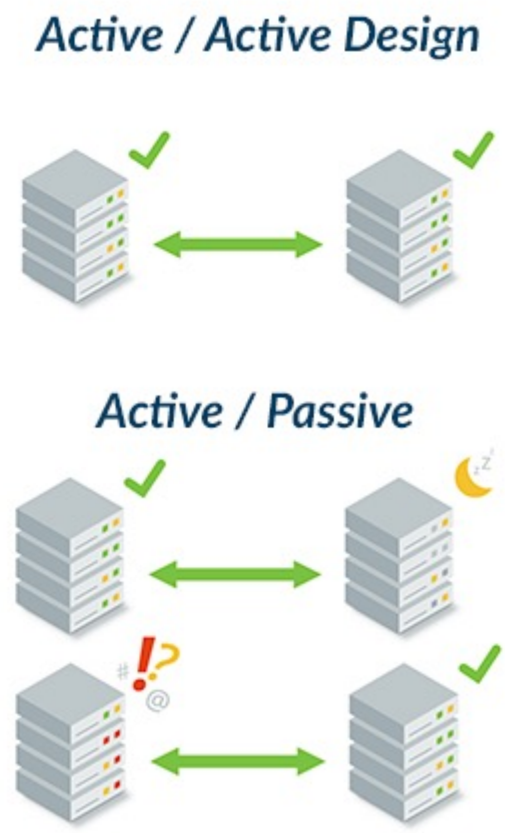
Team Members

Arshita Sandhiparthi
University of the Pacific
Political Science & Computer Science
Graduating Spring 2022

Emily Ramirez Serrano
Northern Arizona University
Computer Science
Graduating Spring 2022

High Availability (HA)

- Why HA?
  - Continuous operation
  - Reliable protection
  - Automatic failover procedures in outages or node failure
- The Biggest Use Case
  - The Lustre file system
- Problem
  - Don't have a system set up to failover NFS on mgmt nodes
  - Need to explore CentOS



ZFS

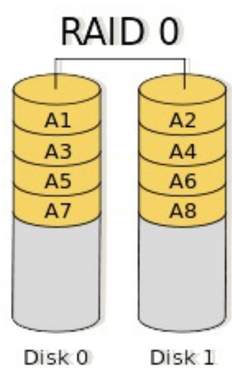
OpenZFS

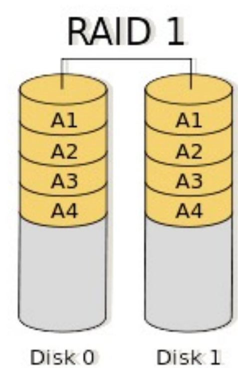
- ZFS
  - zpools
  - RAIDz1
  - multihost
- SAN Arrays
  - Storage Area Network
  - Logical Unit Numbers (LUN)
  - Multipath

```
[root@stc2 ~]# zpool status
  pool: stc2_pool
 state: ONLINE
  scan: resilvered 126K in 00:00:00 with 0 errors on Thu Aug  5 12:09:46 2021
config:

        NAME          STATE     READ WRITE CKSUM
        stc2_pool     ONLINE       0     0     0
          raidz1-0    ONLINE       0     0     0
            stc1      ONLINE       0     0     0
            stc2      ONLINE       0     0     0
            stc3      ONLINE       0     0     0

errors: No known data errors
```





openzfs.github.io/openzfs-docs

Pacemaker

- Pacemaker
  - HA Resource Manager software
- Fencing and Shoot The Other Node In The Head (STONITH)
  - Powerman
  - Small Computer System Interface (SCSI)
- Safely manage resources across the system

```
Node List:
  * Online: [ radon1 radon3 radon4 ]

Full List of Resources:
  * ClusterIP   (ocf::heartbeat:IPaddr2):        Started radon1
  * WebSite     (ocf::heartbeat:apache):         Started radon3
  * fence_pm    (stonith:fence_powerman):        Started radon1
```

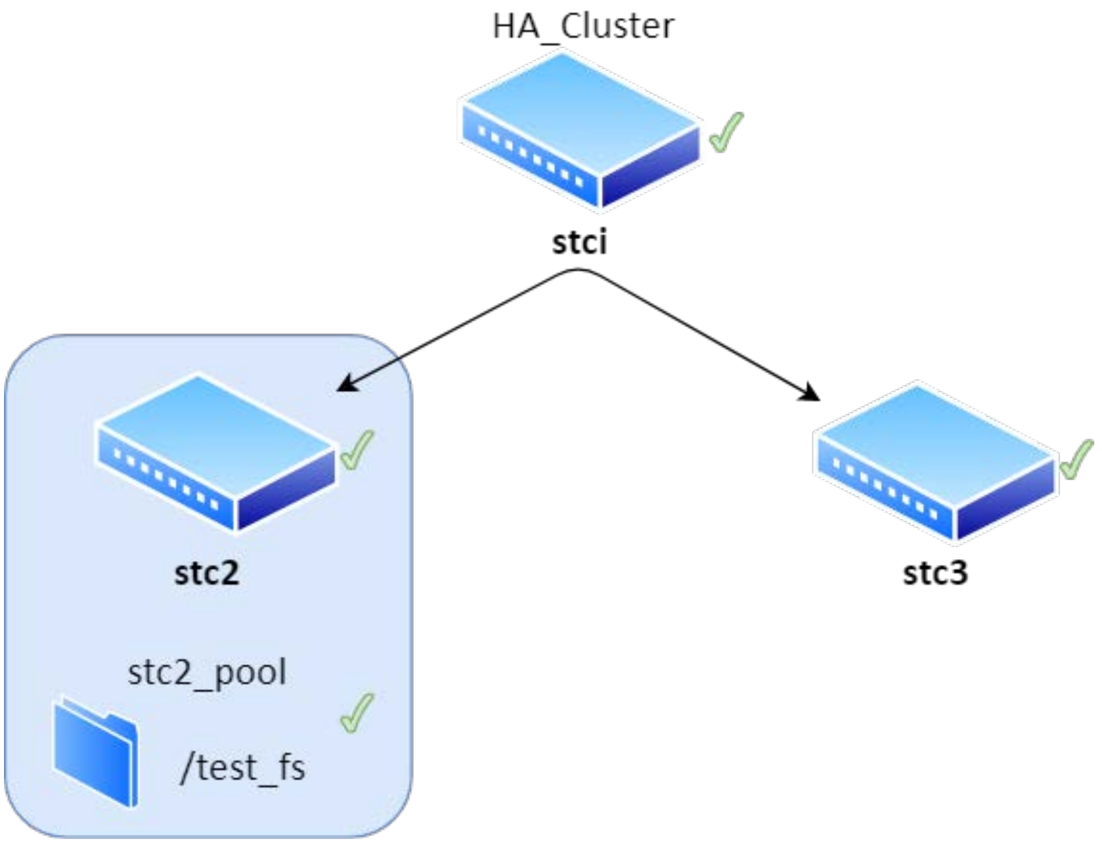
clusterlabs.org/pacemaker

Project Accomplishments

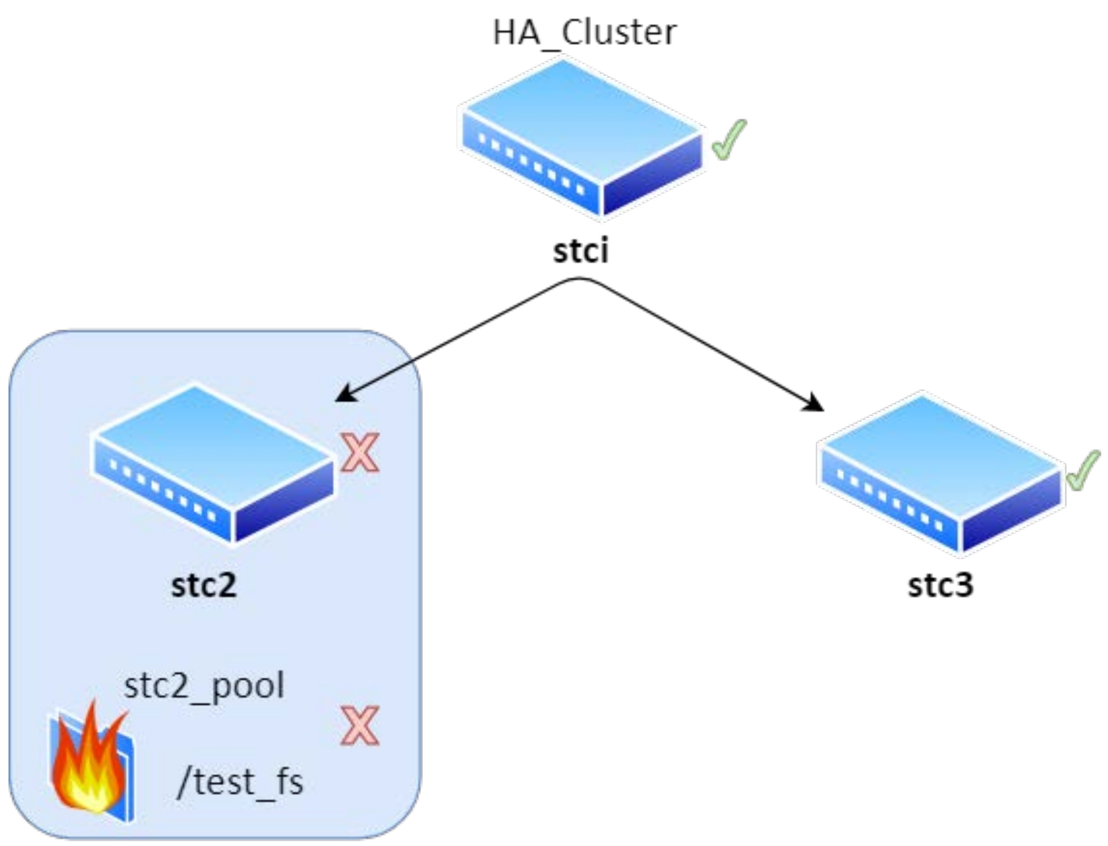
Integrating NFS With ZFS

- Goal: Setup pacemaker to support a HA setup and manage ZFS and NFS resource migration.
- Configuring Pacemaker and ZFS
  - Migrating resources
    - Importing/Exporting ZFS pools
    - Floating IP
  - Using multipath devices
- NFS on top of ZFS
  - ZFS pools are already widely used at the lab but not with NFS

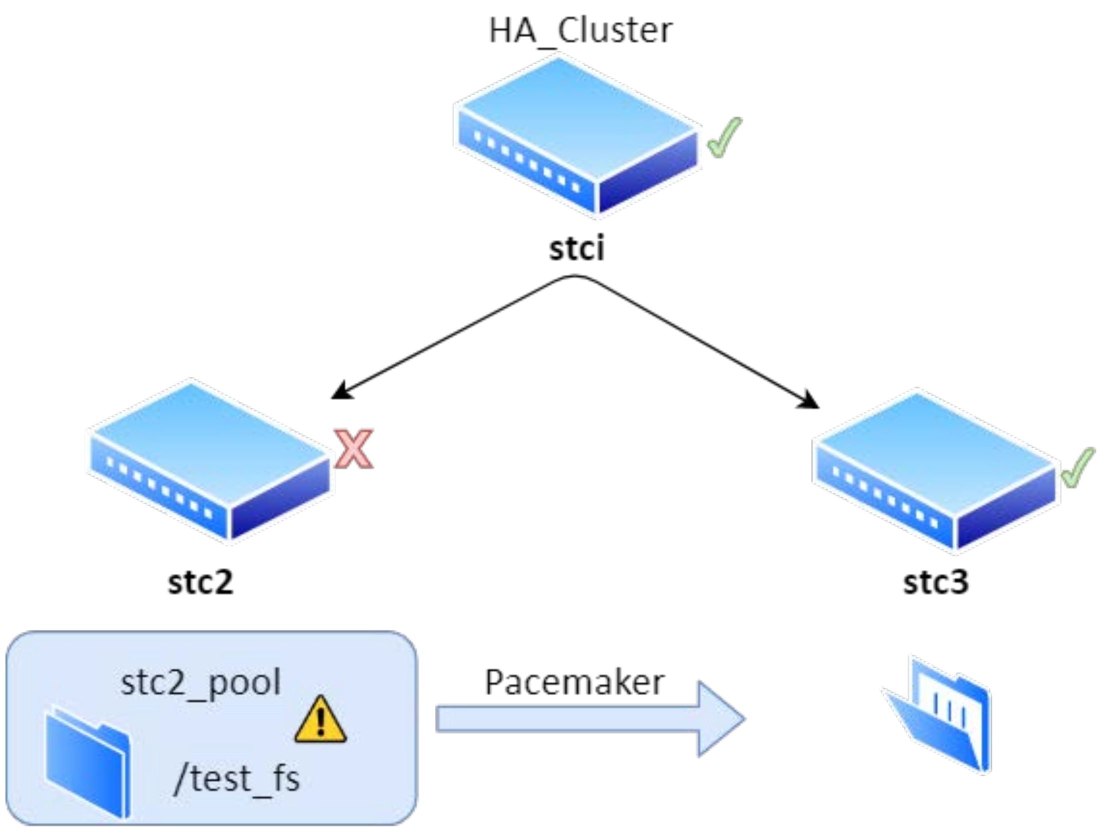
Integrating NFS With ZFS



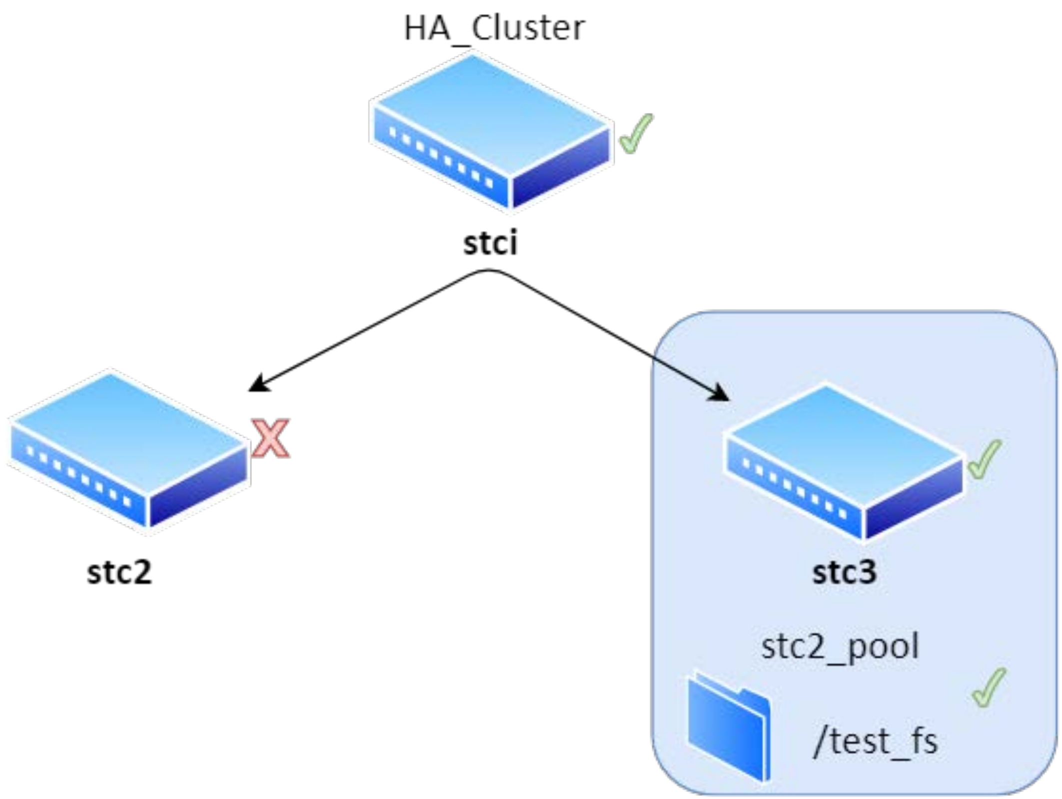
Integrating NFS With ZFS



Integrating NFS With ZFS



Integrating NFS With ZFS



Integrating NFS With ZFS

Before fencing stc2

```
Cluster name: ha_cluster
Cluster Summary:
  * Stack: corosync
  * Current DC: stc (version 2.0.5-9.el8_4.1-ba59be7122) - partition with quorum
  * Last updated: Fri Aug  6 14:50:57 2021
  * Last change:  Fri Aug  6 14:50:54 2021 by hacluster via crmd on stc4
  * 4 nodes configured
  * 3 resource instances configured

Node List:
  * Online: [ stc stc2 stc3 stc4 ]

Full List of Resources:
  * f_scsi2     (stonith:fence_scsi):    Started stc
  * virtual_ip  (ocf::heartbeat:IPaddr2):        Started stc2
  * stc2-zfs    (ocf::heartbeat:ZFS):    Started stc2

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

```
[root@stc2 test_fs]# ls
blah
[root@stc2 test_fs]#
```

On stc2

```
[root@stc3 test_fs]# ls
[root@stc3 test_fs]#
```

On stc3

After fencing stc2

```
Cluster name: ha_cluster
Cluster Summary:
  * Stack: corosync
  * Current DC: stc (version 2.0.5-9.el8_4.1-ba59be7122) - partition with quorum
  * Last updated: Fri Aug  6 14:52:26 2021
  * Last change:  Fri Aug  6 14:52:19 2021 by hacluster via crmd on stc3
  * 4 nodes configured
  * 3 resource instances configured

Node List:
  * Online: [ stc stc3 stc4 ]
  * OFFLINE: [ stc2 ]

Full List of Resources:
  * f_scsi2     (stonith:fence_scsi):    Started stc
  * virtual_ip  (ocf::heartbeat:IPaddr2):        Started stc3
  * stc2-zfs    (ocf::heartbeat:ZFS):    Started stc3

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

```
[root@stc2 test_fs]# ls
[root@stc2 test_fs]#
```

On stc2

```
[root@stc3 test_fs]# ls
blah
```

On stc3

Challenges

- CentOs8 Compatibility
  - Fencing agents (powerman)
    - Custom fencing resource
    - Too simplistic for ZFS management
- Pacemaker and ZFS
  - Importing and Exporting ZFS pools
  - SCSI Fencing
  - ZFS set up took a lot of time
- Lack of Documentation
  - Had to dig around for a lot of information

Future Work and High End Goals

- Migrate ZFS pool and NFS servers across management nodes
- High availability between multiple management nodes

References

- https://github.com/ewwhite/zfs-ha/wiki
- https://openzfs.github.io/openzfs-docs/Project%20and%20Community/index.html
- https://www.clusterlabs.org/pacemaker/doc/2.1/Clusters_from_Scratch/singlehtml/
- https://books.clusterapps.com/books/deployments/page/nfs-on-zfs-ha-cluster
- https://docs.oracle.com/cd/E19253-01/819-5461/gayog/index.html
- https://wiki.lustre.org/Creating_Pacemaker_Resources_for_Lustre_Storage_Services

Lawrence Livermore National Laboratory

Slurm's Rest API

Mentors: David Fox, Ryan Day
Wesley Hsieh
Fnu Azma

August 11, 2021

LLNL-PRES-825699
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

A brief Introduction

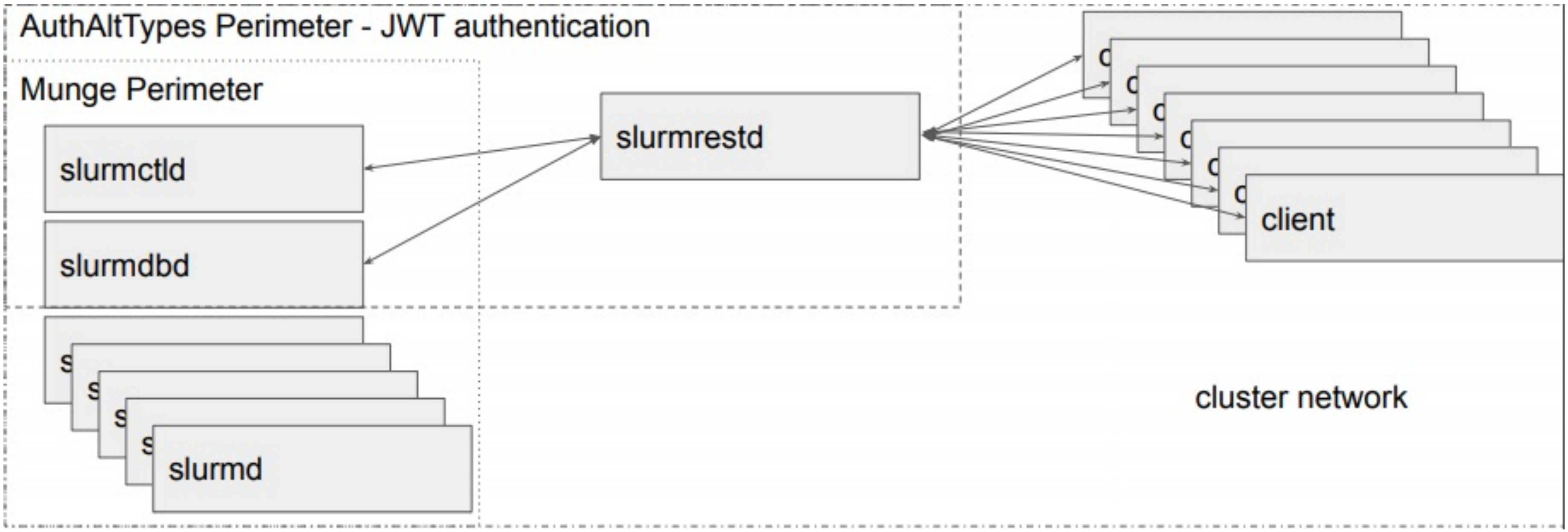
- Wesley Hsieh
- Senior at CSUEB
- Computer Science
- Expected Grad: Dec 2021

- Fnu Azma
- Junior at UCR
- Computer Science and Engineering
- Expected Grad: Dec 2022

Slurm and Slurm’s Rest API

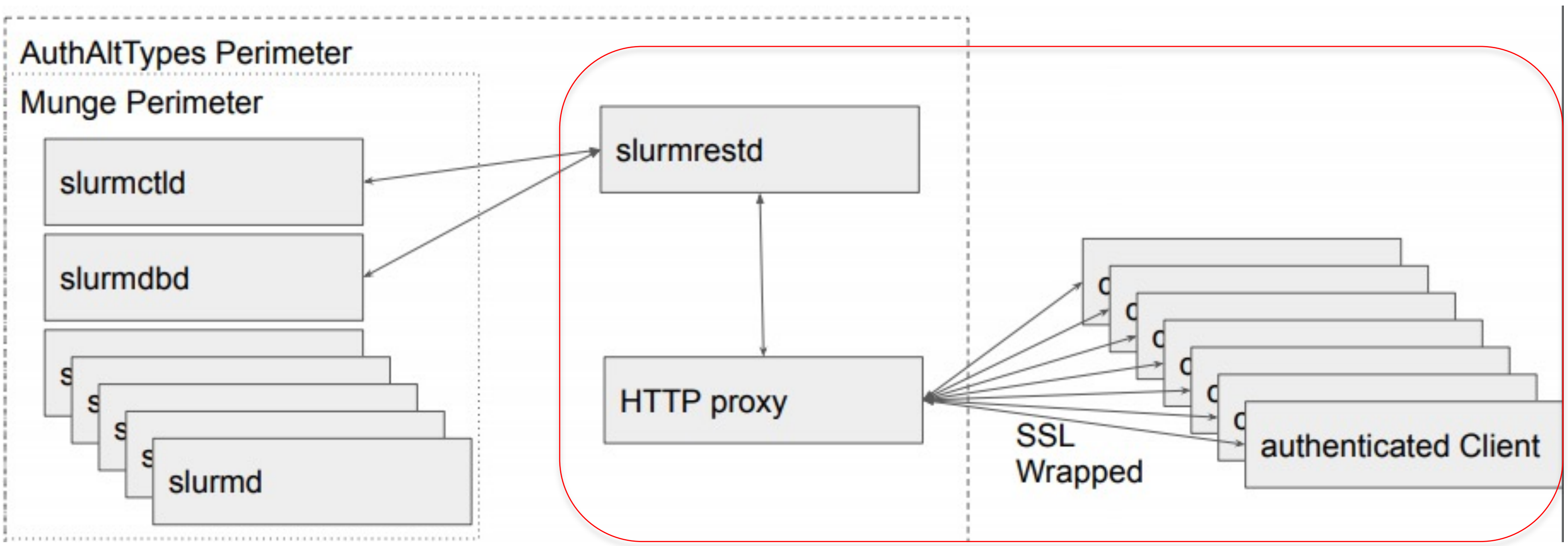
- Slurm:
  - Job scheduler for Linux and Unix systems
  - Features: centralized manager (slurmctld), the executors (slurmd), an “accounting database” (slurmdbd), and it’s own REST API (slurmrestd).
- “A tool that runs inside of the Slurm perimeter that will translate JSON/YAML requests into Slurm RPC requests.”
  - Authentication via Http headers: X-SLURM-USER-TOKEN (auth/jwt) X-SLURM-USER-NAME

Slurmrestd Architecture



Courtesy of https://slurm.schedmd.com/PEARC20/REST_API.pdf

HTTP Proxy Front End



Courtesy of https://slurm.schedmd.com/PEARC20/REST_API.pdf

Project Objectives

- Enable Slurm REST API on management nodes
- Slurm REST API – explore sample code, implement in python.
- Configure/enable the use of a proxy server (NGINX) as an added layer of security.

Slurm REST Calls

- DELETE /slurm/v0.0.36/job/{job_id}
- GET /slurm/v0.0.36/diag
- GET /slurm/v0.0.36/job/{job_id}
- GET /slurm/v0.0.36/jobs
- GET /slurm/v0.0.36/node/{node_name}
- GET /slurm/v0.0.36/nodes
- GET /slurm/v0.0.36/partition/{partition_name}
- GET /slurm/v0.0.36/partitions
- GET /slurm/v0.0.36/ping
- POST /slurm/v0.0.36/job/submit
- POST /slurm/v0.0.36/job/{job_id}
- POST /slurmdb/v0.0.36/clusters
- POST /slurmdb/v0.0.36/wckeys
- DELETE /slurmdb/v0.0.36/account/{account_name}

https://slurm.schedmd.com/rest_api.html

What we actually achieved

- Documenting key aspects of the installation process specific to our clusters.
- Basic python script/example code of utilizing REST API calls.
- Basic example of web proxying via NginX

“Producer-Consumer” Python Script

wesley@siliconi:~

```
slurm-job.json', 'test.json']
No more jobs in queue
Submitting job: slurm-job2.json
Job Submitted:
Job ID        :  90
Completed Jobs/Already Submitted
['slurm-job2.json']


Routine check
['slurm-job3.json', 'slurm-job4.json', 'slurm-job5.json', 'slurm-job.json', 't
est.json']
Job still in progress: 90
Routine check
['slurm-job3.json', 'slurm-job4.json', 'slurm-job5.json', 'slurm-job.json', 't
est.json']
Job still in progress: 90
Routine check
['slurm-job3.json', 'slurm-job4.json', 'slurm-job5.json', 'slurm-job.json', 't
est.json']
Job still in progress: 90
Routine check
['slurm-job3.json', 'slurm-job4.json', 'slurm-job5.json', 'slurm-job.json', 't
est.json']
Submitting job: slurm-job3.json
Job Submitted:
Job ID        :  91
Completed Jobs/Already Submitted
['slurm-job2.json', 'slurm-job3.json']
```

Example Job file (JSON)

```
{
"jobs": {
"tasks": 1,
"name": "test1",
"nodes": 4,
"current_working_directory": "/home/wesley",
"environment": {"PATH": "/bin:/usr/bin/:/usr/local/bin/",
"LD_LIBRARY_PATH":"/lib/:/lib64/:/usr/local/lib"}},
"script":"#!/bin/bash\n sleep 15"}
```

Example of job submission:

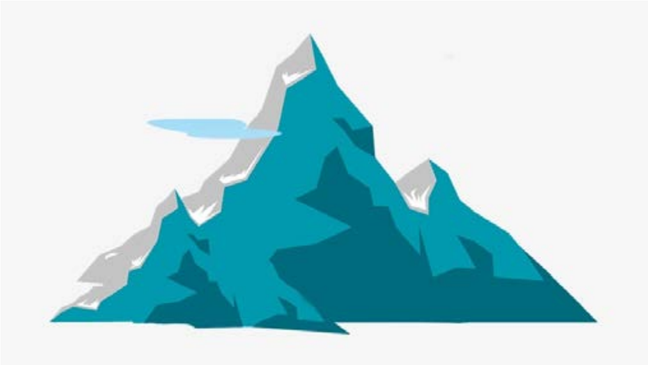
```
[wesley@siliconi ~]$ curl -H "X-SLURM-USER-NAME:$(whoami)" -H "X-SLURM-USER-TOKE
N:$SLURM_JWT" -X POST http://127.0.0.1:5432/slurm/v0.0.36/job/submit -H "Content
-Type: application/json" -d @slurm-job.json
{
   "meta": {
     "plugin": {
       "type": "openapi\/v0.0.36",
       "name": "REST v0.0.36"
     },
     "Slurm": {
       "version": {
         "major": 20,
         "micro": 7,
         "minor": 11
       },
       "release": "20.11.7"
     }
   },
   "errors": [
   ],
   "job_id": 96,
   "step_id": "BATCH",
   "job_submit_user_msg": ""
 }[wesley@siliconi ~]$
```

Example of NginX functionality

```
[wesley@siliconi ~]$ curl -H "X-SLURM-USER-NAME:$(whoami)" -H "X-SLURM-USER-TOKE
N:$SLURM_JWT" http://192.168.95.1:8090/slurm/v0.0.36/ping
{
   "meta": {
     "plugin": {
       "type": "openapi\/v0.0.36",
       "name": "REST v0.0.36"
     },
     "Slurm": {
       "version": {
         "major": 20,
         "micro": 7,
         "minor": 11
       },
       "release": "20.11.7"
     }
   },
   "errors": [
   ],
   "pings": [
     {
       "hostname": "siliconi",
       "ping": "UP",
       "status": 0,
       "mode": "primary"
     }
   ]
```

Some Challenges

- Limited web resources
  - A lot of “trial and error” with API calls due to unclear documentation
  - Trusty old `tail /var/log/slurm/slurmctld.log`
- “High Barrier to Entry”
- Lots of command line usage: i.e. vim, curl, tar



Future Improvements, “Where to go from here”

- Configure Slurm’s database to work with slurmrestd
- Running slurmrestd in the background
  - `systemctl start slurmrestd` vs.
  - `slurmrestd -f /etc/slurm/slurmrestd.conf -s openapi/v0.0.36 -vvvvv 127.0.0.1:[port number]`
- Possible considerations to a more fleshed out web proxy service using NginX/Apache
  - Web Application with two-factor authentication (i.e. RSA-token, AD-native authentication)

Citations/Resources

- https://slurm.schedmd.com/rest_api.html
- https://slurm.schedmd.com/rest.html
- https://nginx.org/en/docs/
- https://www.youtube.com/watch?v=RtdJlstFB28
- https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-uswgi-and-nginx-on-ubuntu-18-04
- https://www.programmersought.com/article/48456629330/

Disclaimer and logo

Lawrence Livermore National Laboratory

Survey of HPC Container Tools

Presenters:
Bryan Whitehurst
Rachel Yamamoto

Mentors:
Eric Green
Martin Baltezore
David Fox

August 11, 2021

LLNL-PRES-825699
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

About

- Bryan
  - University of Alabama
  - Junior Computer Science Student
  - Club Tennis

About

- Rachel
  - Case Western Reserve University
  - Major: Computer Science BS
  - Expected Graduation 2024

What are Containers?

- Standard unit of code that packages up software and all its dependencies so that the application can be run quickly and reliably on multiple systems
- Podman, Singularity, Charliecloud, Sarus, Shifter

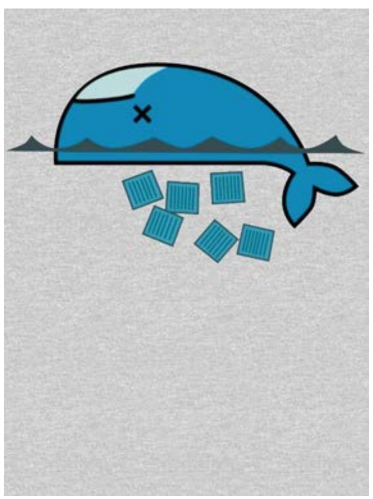
Why do Containers Matter to HPC?

- Portable
  - Ease of transporting software and its dependencies to different systems
- Lightweight
  - Containers are very lightweight compared to VMs
    - Containers use a fraction of the memory required to boot an OS
- Scalable
  - Can distributed to many nodes easily
  - HPC workloads can face a spike in data processing requirements
- Reproducible

Why Not Docker?

- The Docker Runtime doesn't work well in HPC because....
  - Multitenancy
  - Networking

- Docker/OCI Compatible Containers can be run in HPC, just not with the Docker runtime



Our Goals

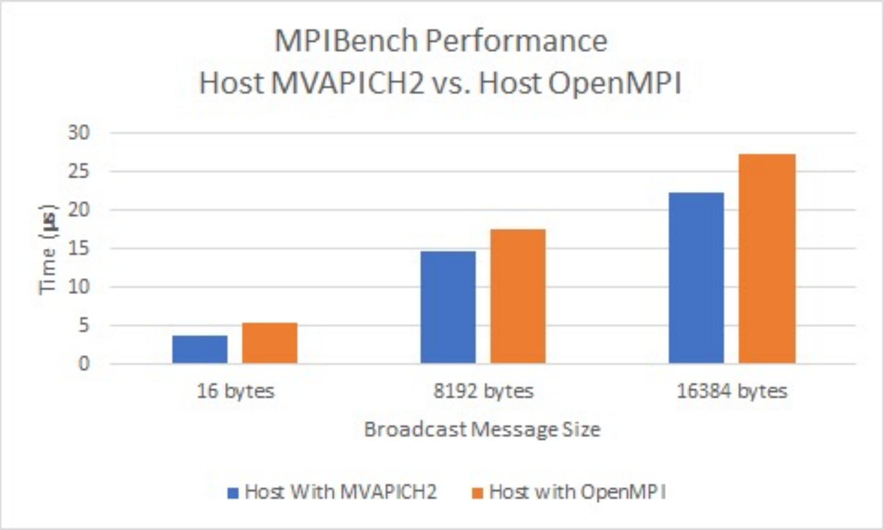
- Install and properly configure container runtimes optimized for HPC
- Run rootless containers using Singularity, Charliecloud, Sarus, Podman, and Shifter
- Configure MPI to work with containers

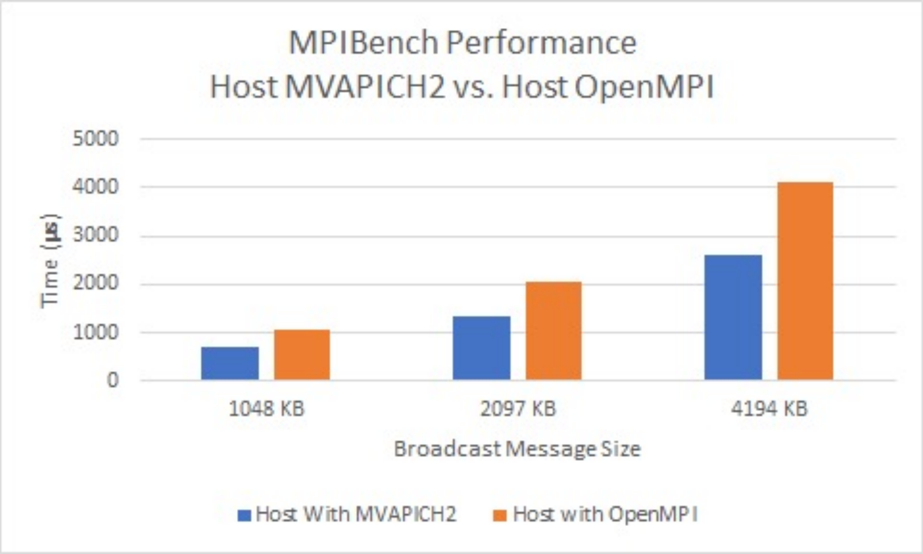
Roadmap: What We Compared

- MVAPICH2 Library performance vs OpenMPI Library performance
- Container performance vs Host System performance
- Singularity vs Charliecloud performance

MVAPICH2 vs. OpenMPI Runtimes

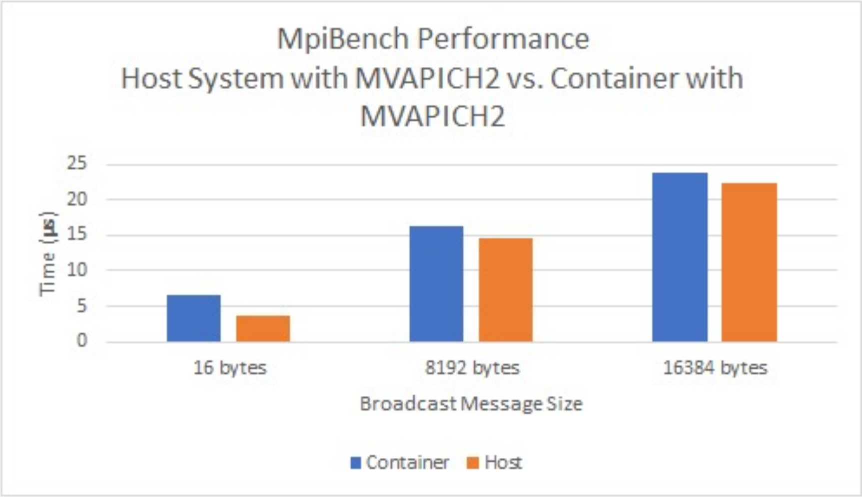
- MVAPICH2 performed **significantly** faster than OpenMPI for small and large message sizes

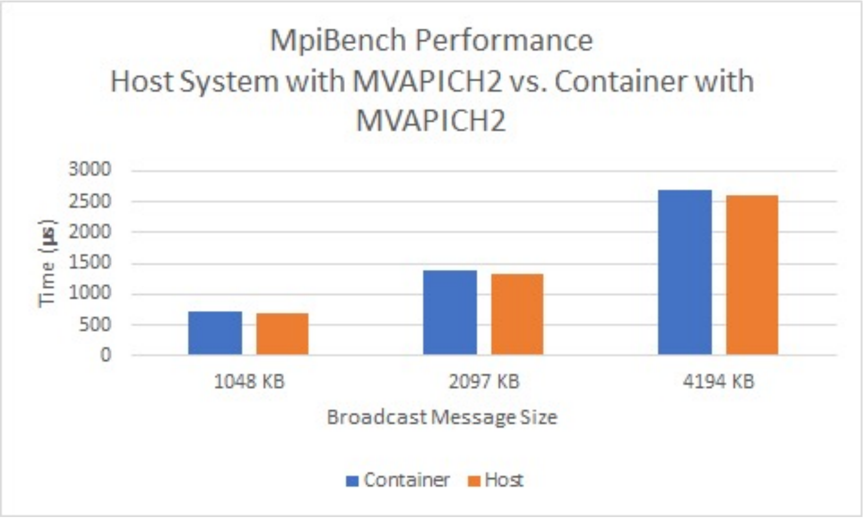




Container vs. Host MPIBench Runtimes

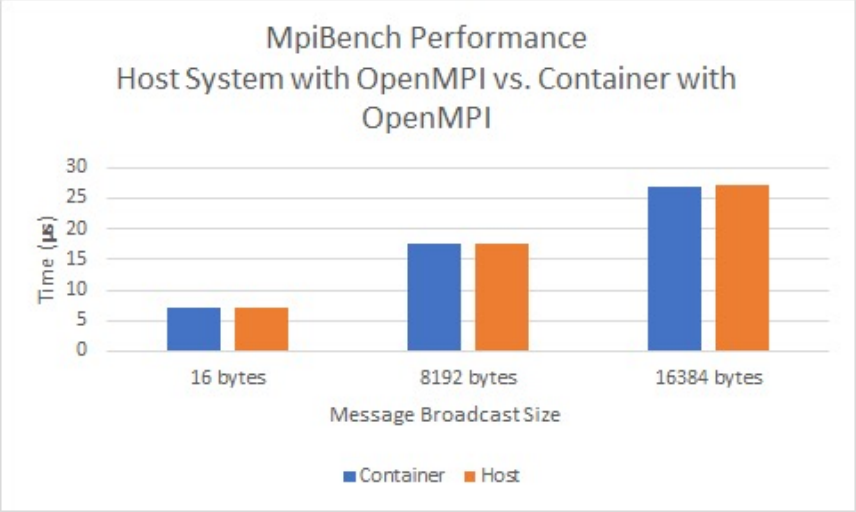
- Containers installed with MVAPICH2 were slower than the host system with MVAPICH2

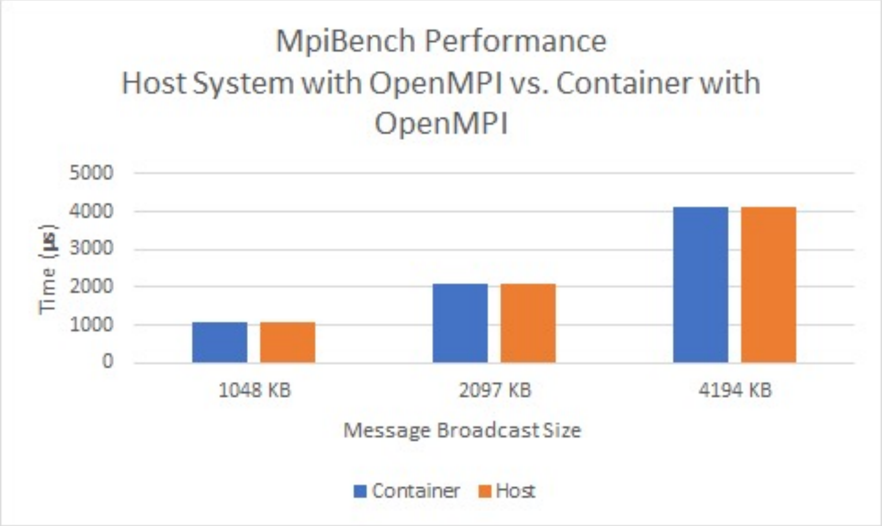




Container vs. Host MPIBench Runtimes

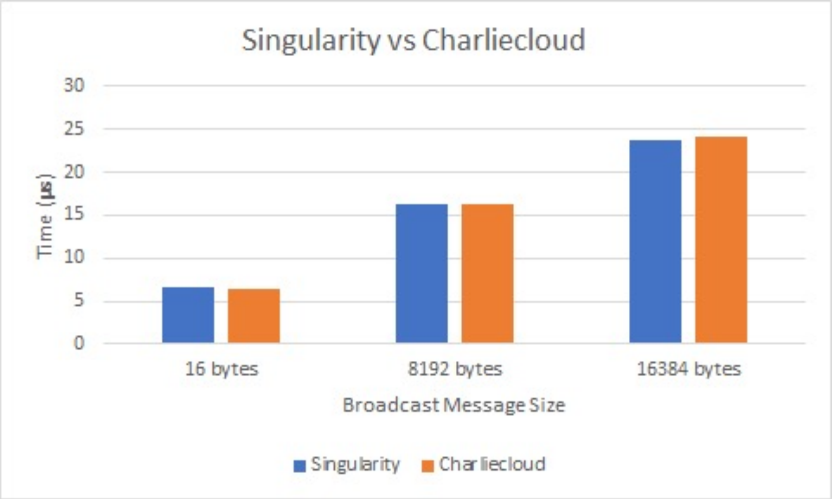
- OpenMPI showed consistent results inside and outside Charliecloud/Singularity containers

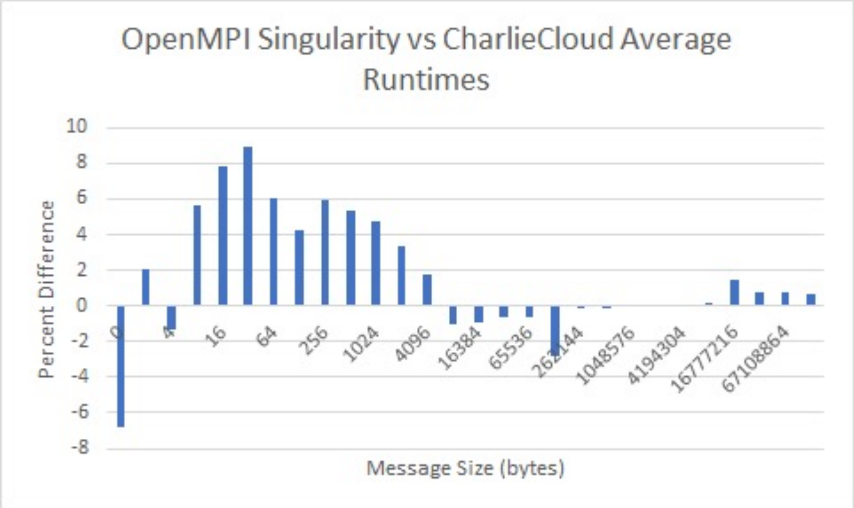


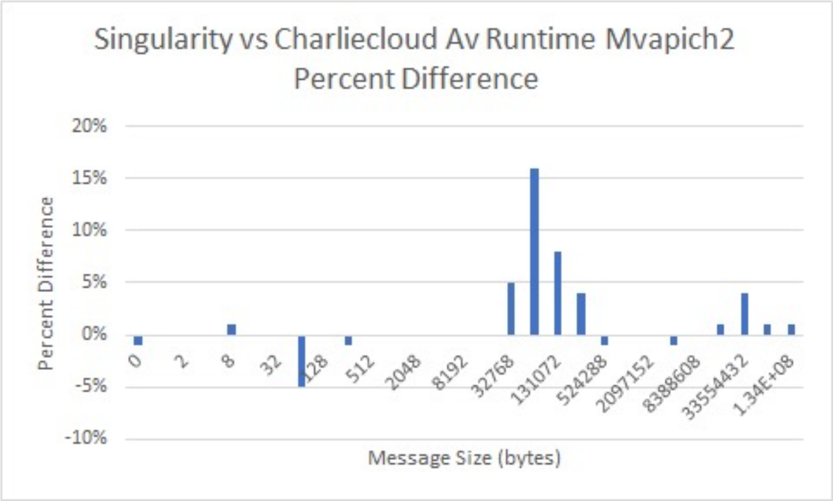


Singularity vs. Charliecloud

- No significant difference between the performance of MPIBench inside Charliecloud and Singularity containers







Challenges

- Sarus
  - Not using the interconnect properly led to high runtimes
- Running Podman Containers stored on NFS
  - Setting up rootless podman to work with NFS
    - Stores images in an NFS based home directory
    - Podman containers cannot run on NFS so you must copy container storage over to each compute node manually
- Establishing MPI and Slurm Compatability
  - Configuring Slurm, OpenMPI, and MVAPICH2 to work with PMI support
  - We had to install OpenMPI and MVAPICH2 from Source – not from the package manager
- Installing Shifter
  - Shifter uses Python 2.7 so it could not be installed on CentOS 8

Future Work

- Shifter
  - Testing runtimes
- Podman
  - Slurm and MPI compatibility
- Sarus high-speed infiniband interface
  - rather than ethernet
- E4s-cl Project
  - Extreme Scale Scientific Software Stack container launcher (e4s-cl)
  - a tool used to run MPI applications in containers
  - Use it to run MPI benchmarks inside the container

References

- https://www.redhat.com/sysadmin/rootless-podman-nfs
- https://podman.io/
- https://www.docker.com/resources/what-container
- https://chrisshort.net/docker-inc-is-dead/

- https://id.pinterest.com/pin/639792690799904646/?amp_client_id=CLIENT_ID(_)&mweb_unauth_id={{default.session}}&amp_url=https%3A%2F%2Fid.pinterest.com%2Famp%2Fpin%2F639792690799904646%2F&from_amp_pin_page=true

- https://sarus.readthedocs.io/en/stable/

- https://hpc.github.io/charliecloud/
- https://containerjournal.com/topics/container-management/containers-hpc-mutually-beneficial/
- https://www.netapp.com/devops-solutions/what-are-containers/
- https://cloud.google.com/containers

Questions

Lawrence Livermore National Laboratory