Containerization on Petascale HPC Cluster

State of Practice Talk in SC20
Nov 17, 2020

PRESENTED BY:

Amit Ruhela, Matt Vaughn, Stephen Lien Harrell, Gregory J. Zynda, John Fonner, Richard Todd Evans, Tommy Minyard

Texas Advanced Computing Center
Austin Texas
Agenda

- Introduction to Containerization in HPC
- Methodology
- Performance Evaluation
  - Microbenchmarks
  - Applications
- Conclusions
What is Containerization?

Standardized way to encapsulates software code and all its dependencies that can run uniformly and consistently on any infrastructure.
Containerization Architecture

Virtual Machines

<table>
<thead>
<tr>
<th>APP1</th>
<th>APP2</th>
<th>APP3</th>
<th>APP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bins/Libs</td>
<td>Bins/Libs</td>
<td>Bins/Libs</td>
<td>Bins/Libs</td>
</tr>
<tr>
<td>Guest OS</td>
<td>Guest OS</td>
<td>Guest OS</td>
<td>Guest OS</td>
</tr>
</tbody>
</table>

Type-2 Hypervisor

Host Operating System

Infrastructure

Containers

<table>
<thead>
<tr>
<th>APP1</th>
<th>APP2</th>
<th>APP3</th>
<th>APP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bins/Libs</td>
<td>Bins/Libs</td>
<td>Bins/Libs</td>
<td>Bins/Libs</td>
</tr>
</tbody>
</table>

Container Runtime Environment

Host Operating System

Infrastructure
Benefits of Containerization

- Low overheads
- User-space Communication
- Performance
- Abundant Compute capabilities
- Batch Scheduling
GOALS

Does the performance of container-based solutions on HPC clusters match bare metal runs at varying problem scales?
Containerization Options
Containerization Options

1. Docker
Containerization Options

1. Docker
   - Scalability and Security Concerns
Containerization Options

1. Docker
   - Scalability and Security Concerns

2. Singularity
Containerization Options

1. Docker
   - Scalability and Security Concerns

2. Singularity

3. Charliecloud
**Containerization Options**

1. **Docker**
   - Scalability and Security Concerns
2. **Singularity**
3. **Charliecloud**
4. **Podman**

...
### Experimental Setup - Hardware Configuration

| **Processor** | 56-core Intel Xeon Platinum 8280 processors ("Cascade Lake")
Two sockets each containing 28 cores
Core Frequency: 2.7GHz.
1 hardware threads/core. |
| **Memory** | 192 GB main memory and 144 GB /tmp partition on a 240GB SSD |
| **Interconnect** | Mellanox HDR-200 between switches and HDR-100 to compute nodes |
# Experimental Setup

<table>
<thead>
<tr>
<th>Library</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Singularity</td>
<td>3.6.0-1.el7</td>
</tr>
<tr>
<td>CharlieCloud</td>
<td>0.19~pre</td>
</tr>
<tr>
<td>Podman</td>
<td>2.0.4</td>
</tr>
<tr>
<td>Mpi Library</td>
<td>MVAPICH2 2.3.4</td>
</tr>
<tr>
<td>Microbenchmarks</td>
<td>Intel(R) MPI Benchmarks 2019 Update 6</td>
</tr>
<tr>
<td>Application</td>
<td>MIMD Lattice Computation (MILC) v7.7.3</td>
</tr>
</tbody>
</table>
Microbenchmark - Bcast

Observations:
1. Latency at small messages with containerized approaches is on-par with bare-metal runs.
2. The trend is similar at large message size indicating Singularity and Charliecloud have no difference in performance once the containers are initialized.

Nodes = 2,048, PPN=1
Cluster: Cascade Lake + InfiniBand
Microbenchmark - Allreduce

Observations:
1. Latency for small messages with containerized approaches is on-par with bare-metal runs.
2. The trend is similar at large message size.

Nodes = 4,096  PPN=1
Cluster: Cascade Lake + InfiniBand
Nodes = 4096, PPN=1
Cluster : Cascade Lake + InfiniBand

Observations :
1. Time to instantiate containers escalate with nodes count
2. Singularity incurs larger overheads compared to Charliecloud
Application - MILC (CG TIME)

Observation:
TIME to solve conjugate gradient is similar for both baremetal and containerized runs.
Application - MILC (Total TIME)

Observation:
Containerized runs show small overheads over baremetal runs, particularly due to containers instantiation.
Application - MILC (Memory)

- MVAPICH2 2.3.4 Native
- MVAPICH2 2.3.4 Charliecloud v0.19 pre
- MVAPICH2 2.3.4 Singularity v3.6.0-1.el7

Grid : 72x72x72x144
PPN = 54
Cluster : Cascade Lake + InfiniBand

Observation:
Memory consumption is similar for baremetal and containerized runs.
Experiments at small scale with a virtual setup similar to Stampede2 configuration indicates 5% - 10% overheads at microbenchmarks level.

Overheads might be result of fuse-overlayfs and additional inter-process isolation, which is under investigations.
Summary

Containerization eludes the build time complexity for HPC applications with

- No significant overheads compared to bare metal runs in terms of latency and memory.
- No known security issues in HPC environments

We validated the on-par performance of Singularity, Charliecloud, and Podman at benchmarks and application on a large scale of up to 4,096 nodes.
Thanks for Listening

E-mails : {aruhela, vaughn, sharrell, gzynda, jfonner, rtevans, minyard}@tacc.utexas.edu