Increasing the portability and reproducibility of a scientific application using containers and Spack

Mentors: Jian Sun, Brian Vanderwende, John Dennis
SAMURAI - an application with build challenges

1. Sensitive to package dependency versions and build tool versions
2. Build scripts make assumptions about location and configuration of installed packages
3. Relies on fragile dependency with non-standard build structure
4. Build fails on some systems
5. Difficult to debug build errors
Problem Statement

**Build complexity:** some software stacks have 10s or 100s of dependencies

**Portability:** deploying applications on different platforms requires system configuration/rebuilding

(a) impractical to build large applications manually
(b) can be difficult to port builds to new systems

Containers - achieve lightweight portability

Traditional

Container

Virtual Machine (VM)

App

App

OS

bin/library

Hypervisor

Virtualized Hardware

OS

bin/library

Hardware
Solution - combine containers with Spack

Container

App

Dependencies

Docker → develop and distribute a container

Spack (package manager) → populate the container

Singularity → deploy the container on HPC

Prototype locally

Store in the cloud

Deploy anywhere
Objectives

1. **Containerize Samurai**: a fragile application with build challenges
   a. **Use Spack to setup** the container **software environment**
   a. **Optimize containers for** multiple HPC architectures (**CPU and GPU**)
   a. **Verify correct program output** and **evaluate performance** on Cheyenne and Casper; **compare to bare-metal** Samurai

1. **Develop a lightweight container** that is easy to distribute

1. Extra Credit: Demonstrate how containerized scientific applications can be easily deployed on different platforms (NCAR clusters, DOE clusters, University clusters, AWS)

1. Extra credit: create a container/suite of containers for all multiple apps
What goes in a container?

Opaque View of Container

Virtualized OS
(isolated file system, specific OS lib/bin, sandboxed processes)

Container Internals

Software

- program x
- package y
- file z

Base Image

OS

Build Image

- file z
- OS

Dockerfile

FROM <base_image>

RUN <command>

COPY <local_file>

COPY <file_from_image>

ENTRYPOINT <shell>
Augmenting a container with Spack

Spack lets you install software how you want and manages package install complexity

$ spack install netcdf-c@4.8.1%nvhpc^hdf5@1.12%gcc

Version
Customized Dependencies
Package
Compiler

Container Internals
Program
Packages
Compiler
spack.yaml
Spack
Base Image
OS

Spack Commands
spack install
spack compiler add
spack env activate
Containerizing Samurai

Developer Container
~10 GB

- samurai
- lrose
- fftw
- netcdf
- cmake
- oneapi/nvhpc
- spack.yaml
- Spack
- Ubuntu

Lightweight Container
~200 MB

- samurai exe
- runtime libs
- Ubuntu

Base Image
Ubuntu Base Image
~10 GB

Base Image
Ubuntu
Portability and Performance - Cheyenne

System Configuration
- 1 full node
- 36 cpu threads
- 128 GB memory

Samurai Execution Time
Supercell Test Case

<table>
<thead>
<tr>
<th></th>
<th>CPU - Regular</th>
<th>CPU - Containerized</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCtransform</td>
<td>49.08</td>
<td>53.10</td>
</tr>
<tr>
<td>SAttransform</td>
<td>32.57</td>
<td>35.74</td>
</tr>
<tr>
<td>Htransform</td>
<td>10.23</td>
<td>12.56</td>
</tr>
<tr>
<td>calcHTranspose</td>
<td>2.20</td>
<td>2.67</td>
</tr>
</tbody>
</table>

Samurai Execution Time
Hurricane Test Case

<table>
<thead>
<tr>
<th></th>
<th>CPU - Regular</th>
<th>CPU - Containerized</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCtransform</td>
<td>70.65</td>
<td>74.80</td>
</tr>
<tr>
<td>SAttransform</td>
<td>67.23</td>
<td>68.64</td>
</tr>
<tr>
<td>Htransform</td>
<td>13.34</td>
<td>14.67</td>
</tr>
<tr>
<td>calcHTranspose</td>
<td>2.16</td>
<td>2.52</td>
</tr>
</tbody>
</table>
System Configuration
- 1 full node
- 1 cpu thread
- 1 V100 GPU
- 300 GB memory

Portability and Performance - Casper

Samurai Execution Time
- SuperCell Test Case

<table>
<thead>
<tr>
<th>Component</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU - Regular</td>
<td>16.82</td>
</tr>
<tr>
<td>GPU - Containerized</td>
<td>16.03</td>
</tr>
</tbody>
</table>

- Hurricane Test Case

<table>
<thead>
<tr>
<th>Component</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU - Regular</td>
<td>20.84</td>
</tr>
<tr>
<td>GPU - Containerized</td>
<td>21.49</td>
</tr>
</tbody>
</table>
Outcomes

1. Demonstrated *containerized Samurai* provides *competitive performance* on both Casper and Cheyenne

1. Customized containers to run on both CPUs and GPUs

1. Generated *lightweight container* that is *easy to distribute*
Future Work

Containerized App Suite

- app1
- app dependencies
- compiler a
- spack.yaml
- Base Image
- Spack
- Ubuntu

- app2
- app dependencies
- compiler b
- spack.yaml

Contributions to Extreme-scale Scientific Software Stack (E4S)
Many thanks to

**Mentors:** Jian Sun, Brian Vanderwende, and John Dennis

**SIParCS organizers:** Virginia Do, Francesgladys Pulido, AJ Lauer, and Jerry Cycone