Introduction to Singularity

Le Yan
Goals

• Understand
  – The targeted use cases of Singularity
  – How Singularity works

• Learn
  – How to build Singularity images on local computers
  – How to run Singularity containers on HPC clusters
Pain Points for HPC Users

• Dependencies of an application are
  – Not available on the host OS
    • For example, GLIBC version too low
  – Complex and difficult to resolve/install
• Reproducibility is not always guaranteed
• Difficult to share workflows, pipelines and environments with colleagues
Pain Points for HPC Users

• Dependencies of an application are
  – Not available on the host OS

  The purpose of Singularity is to eliminate or ease these difficulties.

• Reproducibility is not always guaranteed

• Difficult to share workflows, pipelines and environments
Virtualization

- Virtualization is “the act of creating a virtual (rather than actual) version of something” (Wikipedia)
- So that multiple applications (that have different dependencies) can share the hardware resources on one physical computer

https://blog.netapp.com/blogs/containers-vs-vms/

https://en.wikipedia.org/wiki/Virtualization
Virtual Machines vs. Containers

<table>
<thead>
<tr>
<th>Virtual machines</th>
<th>Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very flexible, e.g. one can run a Windows guest OS on Linux or vice versa</td>
<td>Less flexible, on Linux systems only</td>
</tr>
<tr>
<td>Heavyweight, need to install all files of a guest OS</td>
<td>Very lightweight, will use the kernel of the host OS</td>
</tr>
</tbody>
</table>

https://blog.netapp.com/blogs/containers-vs-vms/
What is Singularity

- Singularity is an open-source container software.
- Allows users to pack an application/workflow/pipeline and all of its dependencies into a single image (file).
- “Container for HPC”
  - Assumes that the user does not have root privileges on the host OS.
  - There are a few others, e.g. Charliecloud, Shifter.
## Containers: Docker vs. Singularity

<table>
<thead>
<tr>
<th></th>
<th>Docker</th>
<th>Singularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privileges</td>
<td>Assumes that the user has root privileges in the production environment</td>
<td>Assumes that the user does not have root privileges in the production environment</td>
</tr>
<tr>
<td>Use Cases</td>
<td>Designed for system services</td>
<td>Designed for HPC use cases</td>
</tr>
</tbody>
</table>
Singularity Vocabulary

• Singularity – the software
  – As in “Singularity 3.5”

• Image – a compressed, read-only file
  – As in “build a Tensorflow 2.6 image”

• Container
  – The technology
    • As in “containers vs virtual machines”
  – An instance of an image
    • As in “process my data in a Singularity container of Tensorflow”
## Why Singularity: HPC Users’ Perspective

<table>
<thead>
<tr>
<th>Pain points using HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependencies of an application are not available on the host OS, e.g. GLIBC version too low</td>
</tr>
<tr>
<td>Dependencies of an application are complex and difficult to resolve/install</td>
</tr>
<tr>
<td>Reproducibility is not guaranteed</td>
</tr>
<tr>
<td>Difficult to share workflows, pipelines and environments</td>
</tr>
</tbody>
</table>
Why Singularity: HPC Users’ Perspective

<table>
<thead>
<tr>
<th>Pain points using HPC</th>
<th>With Singularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependencies of an application are not available on the host OS, e.g. GLIBC version too low</td>
<td>Build an image with a different OS</td>
</tr>
<tr>
<td>Dependencies of an application are complex and difficult to resolve/install</td>
<td>Get an image/recipe (from the developers or other users)</td>
</tr>
<tr>
<td>Reproducibility is not guaranteed</td>
<td>Share an image/recipe</td>
</tr>
<tr>
<td>Difficult to share workflows, pipelines and environments</td>
<td>Share an image/recipe</td>
</tr>
</tbody>
</table>
Demo 1
First encounter
Singularity Basics: Command Line

• Native command line interface
  – Syntax: singularity <command> <options> <arguments>

• To get help information for a specific command:
  – singularity <command> --help
Singularity Basics: Overlay File System

What the user sees within the container:
- `/etc`
- `/usr`
- `/lib`
- `/home`
- `/work`
- `/project`

Singularity image:
- `/etc`
- `/usr`
- `/lib`

Host OS:
- `/etc`
- `/usr`
- `/lib`
- `/home`
- `/work`
- `/project`
Singularity Basics: Overlay File System

What the user sees within the container:
- `/etc`
- `/usr`
- `/lib`  

Singularity image:
- `/etc`
- `/usr`
- `/lib`

Host OS:
- `/etc`
- `/usr`
- `/lib`
- `/home`
- `/work`
- `/project`

/home is bound by default.
The bind paths (e.g. /work, /project) need to be specified with the “-B” option.
Singularity Basics: Privilege Escalation

• If you do not have root privileges outside the container, you do not have them inside the container either.

• Need to build images on your local computer (where you have root privileges)
Singularity Workflow

• Step 1: Install Singularity on a local Linux machine (or a Linux VM on a Windows machine)
  – Root privilege is needed
• Step 2: Build Singularity images on the local machine
  – Root privilege is needed
• Step 3: Upload images onto the HPC cluster
  – Root privilege is NOT needed
• Step 4: Run containers on the HPC cluster
  – Root privilege is NOT needed
Installing Singularity

• On Linux
  – Install binary (recommended)
    • Use either `apt-get` or `rpm/yum`
  – Install from source
    • [https://github.com/sylabs/singularity](https://github.com/sylabs/singularity)

• On Windows or Mac
  – Install a Linux VM first
Building Singularity Images (1)

- Use the "build" command to build Singularity images
- Need root privileges
- Syntax:
  - `singularity build [build options...] <image file path> <BUILD TARGET>`

Build a Centos 7 image:

```
sudo singularity build centos7.sif docker://centos:7
```
Building Singularity Images (2)

- The "BUILD TARGET" defines the method how an image is built
  - A URI to a base OS/container image
    - Docker Hub images begins with `docker://`
    - Singularity Hub images begins with `shub://`
  - Path to a Singularity sandbox (see next slide)
  - Path to a Singularity recipe (definition file)
    - More on this later
Building Singularity Images (3)

- By default, a compressed, read-only image will be built.
- The "--sandbox" option tells Singularity to build a sandbox to which changes can be made.
- To make changes to the sandbox, use the `singularity shell` command with the "--writable" option.
  - Otherwise any change made will be wiped when the session ends.
  - Then you can
    - Manage files and directories
    - Install packages/dependencies

Build a Centos 7 image as a sandbox:
```bash
sudo singularity build --sandbox centos7 docker://centos:7
```
Demo 2
Build Singularity Images
Building Workflow

1. Find a base container/OS image
2. Build a sandbox
3. Make changes to the sandbox (e.g. create directories, install packages)
4. Test in the production environment (HPC cluster)
5. Capture all actions in a recipe and rebuild the read-only image

If necessary
Singularity Definition Files (Recipes)

- Capture all the interactive building steps

**BootStrap:**
```
From: centos:7

%labels
  Author lyan1@lsu.edu

%post
  yum install -y python3 vim

# Create bind points for HPC environment
  mkdir /project /work

%environment
  export LC_ALL=C

%runscript
  echo "Hello, world!"
```

**Header:** describes the base container image

**Label:** metadata for the container

**Post:** commands executed within the container after the base OS has been installed at build time.

**Environment:** define environment variables

**Runscript:** commands that will be run when the container is run by “singularity run”
Base OS/Container Images (1)

• Repositories:
  – Docker hub: https://hub.docker.com
  – Singularity hub: https://singularity-hub.org
  – NVIDIA GPU Cloud: https://ngc.nvidia.com
  – QUAY: https://quay.io
  – Distribution repo
    • YUM/RHEL
    • Debian/Ubuntu
• See examples from “singularity build --help”
Base OS/Container Images (2)

• Pay attention to the OS version
  – Singularity uses the kernel of the host OS
  – Do not deviate too much from the host OS kernel
    • Otherwise you will get error messages like “FATAL: kernel too old/new”
Inspecting Singularity Images

• Use the “inspect” command to query
  – How an image is built
  – What the runscript is
  – What environment variables are set

• Syntax: `singularity inspect [options] <container image>`
  – The options are self-explanatory: “--labels”, “--runscript”, “--deffile”, “--environment” etc.
Demo 3
Build Singularity containers from recipes
Singularity on HPC Clusters

• Singularity is installed on all compute nodes on all HPC clusters
  – No Singularity on the head nodes

• You need to
  – Send an email to sys-help@loni.org and ask to be added to the “singularity” user group (one time)
  – The image file needs to be owned by the “singularity” user group
    • Use the “chgrp” command
    • Incorrect group ownership will generate an error: “FATAL: failed to retrieve group information for cvmfs: group: unknown group cvmfs”

• You will NOT be able to build Singularity images on the HPC clusters
Running Singularity on HPC Clusters

• Syntax
  - `Singularity <command> [options] <container image>`

• Commands
  - `shell`: run an interactive bash shell
  - `run`: launch the runscript
  - `exec`: execute a command
Frequently Used Options

• “–B” or “--bind”: directory binding
  – To bind a directory, it needs to be present both within and without the container
  – The home directory is bound automatically
  – Can be called multiple times

• “–nv”: enable NVIDIA GPU support
Running Singularity As Batch Jobs

• Singularity can run in a job script just like any other application

```bash
#!/bin/bash
#PBS -A loni_loniadmin1
#PBS -q checkpt
#PBS -l nodes=1:ppn=20
#PBS -l walltime=24:00:00
#PBS -N TF_benchmark

cd $PBS_O_WORKDIR

SIMG=/home/admin/singularity/tensorflow-2.2-gpu-dockerhub.simg

Singularity exec -B /project -nv $SIMG \ python3 benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py \ --num_gpus=2 -batch_size=32 -model=resnet50 \ --variable_update=parameter_server
```
Demo 4
Run Singularity containers on clusters
Singularity Workflow

1. Install Singularity on local computer – need root
2. Build a Singularity image – need root
   A. Build a sandbox from an existing base image
   B. Make changes to the sandbox
   C. Test the image on the production environment (HPC clusters)
      • Go back to step 2B if necessary
   D. Capture all actions in a recipe and build a read-only image from it
      • Alternative: build the read-only image from the sandbox (harder to share with others)
3. Run on the HPC cluster
   Need to be added to the “singularity” user group
   The image needs to be owned by the “singularity” group
### Why Singularity: HPC Users’ Perspective

<table>
<thead>
<tr>
<th>Pain points using HPC</th>
<th>With Singularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependencies of an application are not available on the host OS, e.g. GLIBC version too low</td>
<td>Build an image with an different OS</td>
</tr>
<tr>
<td>Dependencies of an application are complex and difficult to resolve/install</td>
<td>Get an image/recipe (from the developers or other users)</td>
</tr>
<tr>
<td>Reproducibility is not guaranteed</td>
<td>Share an image/recipe</td>
</tr>
<tr>
<td>Difficult to share workflows, pipelines and environments</td>
<td>Share an image/recipe</td>
</tr>
</tbody>
</table>
Services Provided by HPC

• Prebuilt Singularity images
  – Located under /home/admin/singularity on all clusters

• Recipes
  – https://github.com/lsuhsphpchelp/singularity

• Troubleshooting and consulting
Questions?