Singularity on HPC

What is Singularity?

Singularity is an open-source container platform that is designed for high-performance computing environments. It allows users to create and run lightweight, portable, and reproducible containers that encapsulate a complete runtime environment, including software dependencies, libraries, and configurations. Singularity containers are built from a single image file, which can be signed and verified for authenticity and integrity, and can be run by any user without the need for root access.

Features:

- Verifiable reproducibility and security, using cryptographic signatures, an immutable container image format
- Isolation and dedicated hardware access like GPU
- Mobility of container images
- Integration with resource managers like SLRUM

NOTE: Singularity is compatible with docker and podman images.

Links:

Official Link

User Guide

Versions Available:

The following versions are available on the cluster:

- Singularity v3.7.2

How to load Singularity?

To load Singularity, use the following commands:

```
#Load the Singularity module
module load singularity/3.7.2
```

To verify if the module is loaded correctly, use the following command,

```
# List all the module loaded in the environment
module list
```

In a fresh environment, this should show singularity and go module loaded.

Singularity Permissions

Singularity relies on UID-GID mapping to ensure consistent permissions between the host and the container. This mapping guarantees that the user within the container has the same permissions as the user outside who creates the container. Consequently, trying to gain root privileges within a cluster-based container will result in a permissions error. However, executing the same task within a container on a local PC, where the user has sudo permissions, will succeed.

To install or run commands requiring sudo permissions, such as

```
sudo apt install cmatrix
```

Users must locally install Singularity on a system where they have sudo permissions. They can then install the desired package and update the container file accordingly. After completing these steps, the user can export the container file to a cluster and execute the program within the cluster environment. This approach allows users to use the required permissions during the initial setup and subsequently run the container on the cluster without encountering permission-related issues.

For local installations, follow the official documentation or user guide below,

User Guide

Deb and Rpm packages with installation guide linked here.

For users who do not want to install singularity locally, they can use remote building tools offered free of cost by SyLabs.

For users currently using docker or podman for containerization, the same task can be performed on those platforms as well.

Cloud Libraries for containers

- Singularity Hub
- Docker Hub
- Nvidia's NGC catalog
- BioContainers

How to use Singularity?

Here, we will show basic skills to get started with singularity. If users are interested in in-depth tutorials, the following resources might come in handy:

- SyLabs Tutorials

Primarily, singularity image uses a lot of space which might exceed uses home quota, so moving the cache directly to scratch is suggested,

```
export SINGULARITY_CACHEDIR=/scratch/$USER/singularity/cache
export SINGULARITY_TMPDIR=/scratch/$USER/singularity/temp
```

or users can routinely remove the cache using

```
singularity cache clean --days 1
```

For this tutorial, I am going to use the PyTorch image which I pulled from Nvidia NGC repository to train a feed-forward neural network using GPU in our cluster.

```
singularity pull docker://nvcr.io/nvidia/pytorch:20.01-py3
```

This command will pull the image hosted on the NGC repository to a local sif file in the user's current working directory. Singularity will also pull all the components (blobs) to make this image and cache in the current cache directory. This is the reason that we redirect the caching to another large directory. The image is itself 3GB so it might take a while to download and assemble.

Meanwhile, let us clone my GitHub repository where a neural network and data files are located.

```
git clone https://github.com/NISCHALPI/SingularityHPC.git
```

Now jump to a compute node with gpu using this command,

```
srun -n 1 -c 12 --mem 12G -p gpu --qos gpu --gres gpu --pty bash
```

Note, this can also be executed by using a standard sbatch script instead of an interactive terminal. But, for this tutorial, I am opting for an interactive terminal.

Assuming users have already loaded singularity, user can run singularity in three ways:

Like an executabe:

```
./pytorch_20.01-py3.sif
```

This will land the user inside the singularity container or entry point of the image which in this case is the bash shell. A prompt “Singularity>” should show that the user is inside the container. The UID and GID are the same inside the container, we can verify using

```
id
```

Using singularity shell:

```
singularity shell --nv pytorch_20.01-py3.sif
```

This has the same effect as the executable case.

Using the exec command:

```
singularity exec --nv pytorch_20.01-py3.sif echo “hello world”
```

This is a bit different because this executes the command inside the container and redirects the output to the standard output of the current shell. The program is run inside the container environment! This can help users change the entry point of the image.

Few notes:

By default, the home directory of the user is binded to the home directory inside the container. If a user wants to access other directories, this is the syntax for the bind,

```
singularity shell --bind <host directory>:<container directory> image
```

Users can also make virtual volumes and networks inside the container which is out of the scope of this tutorial.

GPU Neural Networks Training

Now, let us train a neural network using GPU in the cluster. First, let us talk about the --nv flag. This flag is passed to singularity for the container to communicate with the

Nvidia drivers for the host system. Since we are training in GPU, we have to pass the --nv flag before any command.

To run the python script to train the neural network, jump to the container shell using the above command and execute the following.

```
#I am assuming this cloned directory is in the home directory
cd SingularityHPC
```

Now run the neuralnetwork.py file inside the directory,

```
# Here the python is the python installed in the container
python neuralnetwork.py
```

This should print the test error rate. If the user encounters an NVIDIA driver error, it is possibly because of the --nv flag not being passed.

How do we know it is the GPU that is doing the heavy lifting?

Users can see if pytorch detects the GPU by using the following:

```
# Run python shell
python
```

```
# Run the following commands
import torch
# This should return true on gpu nodes
torch.cuda.is_available()
```

This is just a simple example of Singularity. We can do a lot of stuff in a container without ever having to install a single piece of software in the HPC environment. For example, users can run Jupyter Notebook containers and even web servers. All the containers are mostly reproducible and very portable which is all the reason to use them.

MPI Programs: MPI programs in containers are executed similarly as though it was a normal MPI job. For detailed information, see the following tutorial.

Follow the mentioned resources for additional information about singularity.

Where to find help?

If you are confused or need help, contact OIT (Office of Information Technology) at the following address.

https://ua-app01.ua.edu/researchComputingPortal/public/oitHelp