Introduction to containerization using Singularity on Comet

Mahidhar Tatineni
UCLA
May 15, 2018

Singularity: Provides Flexibility for OS Environment

- Singularity (http://singularity.lbl.gov)is a relatively new development that has become very popular on Comet.
- Singularity allows groups to easily migrate complex software stacks from their campus to Comet.
- Singularity runs in user space, and requires very little special support – in fact it actually reduces it in some cases.
- We have roughly 15 groups running this on Comet.
- Applications include: Tensorflow, Torch, Fenics, and custom user applications.
- Docker images can be imported into Singularity.

Use Case: PDB REDO project

- **Assisted PDB-REDO team in running the complete PDB-REDO pipeline with with 101,570 entries using Gordon and Comet supercomputers**
- **Complex software stack with 50+ independent libraries. Docker container imported via Singularity.**
- **Skill in working at-scale - scheduled computations for 100K+ entries with multiple steps using pylauncher**
- **Singularity gave flexibility to move the complex stack between Gordon and Comet (for large memory) and bind mount scratch directories (different on two machines).**
- **Published paper:**
  - van Beusekom B, Touw WG, Tatineni M, Somani S, Rajagopal G, Luo J, Gilliland GL, Perrakis A, Joosten RP Homology-based hydrogen bond information improves crystallographic structures in the PDB. Protein Science. 2018;27:798-808.

Singularity Use Cases

- Applications with newer library OS requirements than available on the HPC system – e.g. Tensorflow, Torch, Caffe.
- Commercial application binaries with specific OS requirements.
- Importing docker images to enable use in a shared HPC environment. Sometimes this is entire workflows with a large set of tools bundled in one image.
- Limitation: Cannot run services, modify underlying system configuration.

Singularity Image Sources

- **SDSC staff have some useful images in:**
  - /share/apps/compute/singularity
  - /share/apps/gpu/singularity
- **Users can build their own images on their laptops/desktops/cloud - as long as you have singularity installed and have root access on your own machine (or VM or cloud instance)**
- **Pull an image from Singularity Hub**
- **Import a docker image**
- **Comet specific documentation available at:**
  - http://www.sdsc.edu/support/user_guides/tutorials/about_comet_singularity_containers.html

Downloading prebuilt images

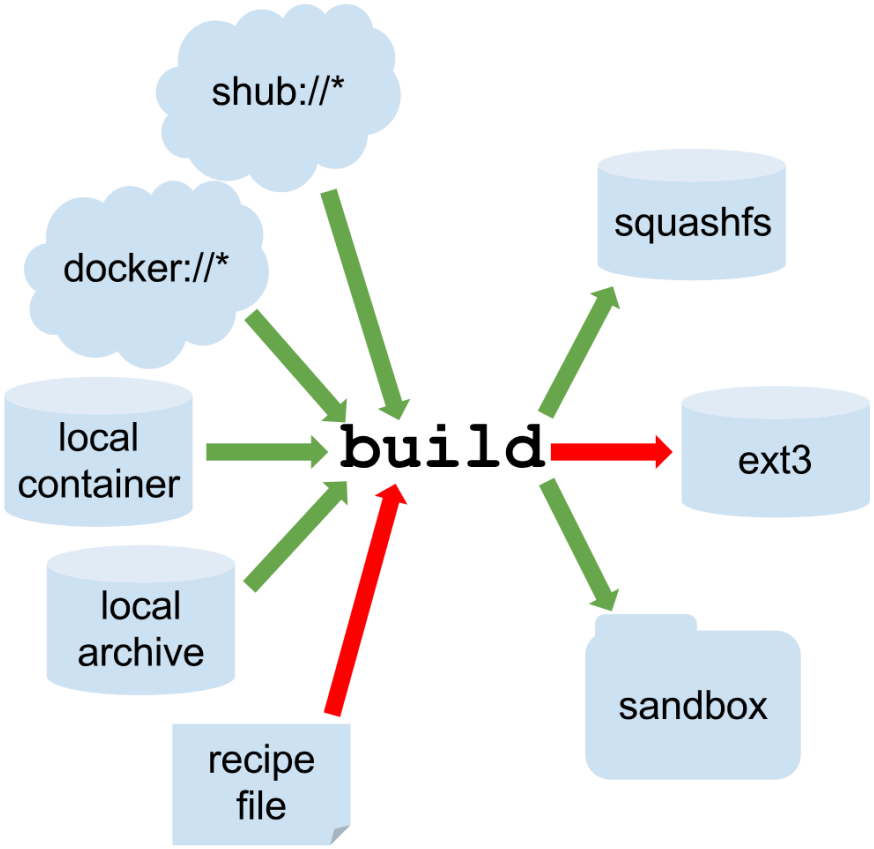
- **singularity pull command can be used to download images from**
  - *Singularity Hub*: e.g. singularity pull shub://vsoch/hello-world
  - *Docker Hub :* e.g. singularity pull --name funny.simg docker://godlovedc/lolcow

- **The pull option simply downloads the image to your system. You can download to a custom name.**

Singularity Pull

```
root@mahidhar-VirtualBox: /tmp/NEW
root@mahidhar-VirtualBox:/tmp/NEW# singularity pull shub://vsoch/hello-world
Progress |==================================| 100.0%
Done. Container is at: /tmp/NEW/vsoch-hello-world-master-latest.simg
root@mahidhar-VirtualBox:/tmp/NEW#
root@mahidhar-VirtualBox:/tmp/NEW#
root@mahidhar-VirtualBox:/tmp/NEW#
root@mahidhar-VirtualBox:/tmp/NEW# singularity pull --name hello.img shub://vsoc
h/hello-world
Progress |==================================| 100.0%
Done. Container is at: /tmp/NEW/hello.img
root@mahidhar-VirtualBox:/tmp/NEW#
root@mahidhar-VirtualBox:/tmp/NEW# ls -lt
total 127644
-rwxr-xr-x 1 root root 65347615 Apr 30 16:23 hello.img
-rwxr-xr-x 1 root root 65347615 Apr 30 16:22 vsoch-hello-world-master-latest.sim
g
root@mahidhar-VirtualBox:/tmp/NEW#
```

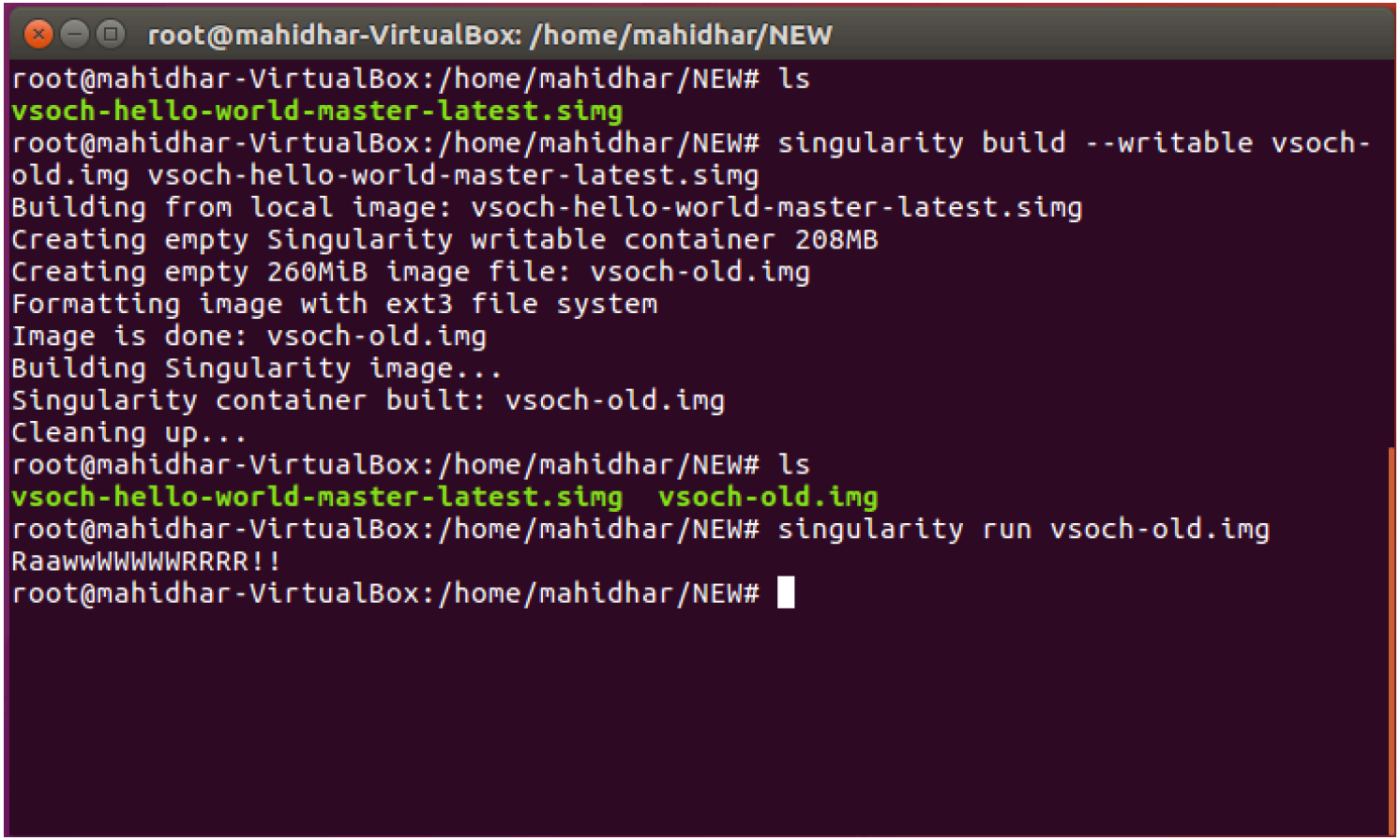
Singularity ”build” option



Reference: Singularity user guide: https://singularity.lbl.gov/docs-build-container

Build by converting existing image

- Comet is currently running older version. So we can convert the image we just pulled.



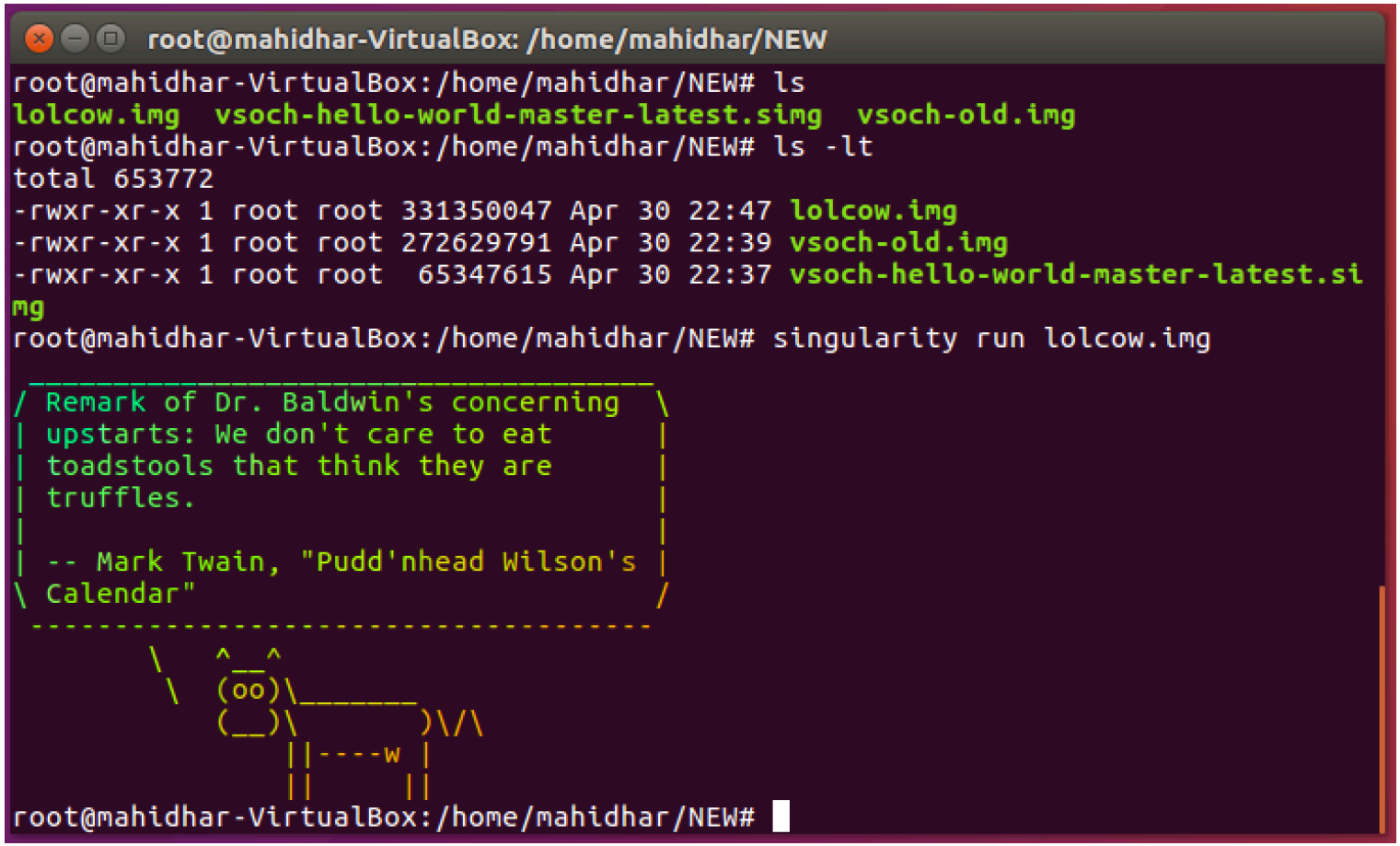
Build from docker image

singularity build --writable lolcow.img docker://godlovedc/lolcow

```
root@mahidhar-VirtualBox: /home/mahidhar/NEW

root@mahidhar-VirtualBox:/home/mahidhar/NEW# singularity build --writable lolcow
.img docker://godlovedc/lolcow
Docker image path: index.docker.io/godlovedc/lolcow:latest
Cache folder set to /root/.singularity/docker
Importing: base Singularity environment
Importing: /root/.singularity/docker/sha256:9fb6c798fa41e509b58bccc5c29654c3ff46
48b608f5daa67c1aab6a7d02c118.tar.gz
Importing: /root/.singularity/docker/sha256:3b61febd4aefe982e0cb9c696d415137384d
1a01052b50a85aae46439e15e49a.tar.gz
Importing: /root/.singularity/docker/sha256:9d99b9777eb02b8943c0e72d7a7baec5c782
f8fd976825c9d3fb48b3101aacc2.tar.gz
Importing: /root/.singularity/docker/sha256:d010c8cf75d7eb5d2504d5ffa0d19696e8d7
45a457dd8d28ec6dd41d3763617e.tar.gz
Importing: /root/.singularity/docker/sha256:7fac07fb303e0589b9c23e6f49d5dc1ff9d6
f3c8c88cabe768b430bdb47f03a9.tar.gz
Importing: /root/.singularity/docker/sha256:8e860504ff1ee5dc7953672d128ce1e4aa4d
8e3716eb39fe710b849c64b20945.tar.gz
Importing: /root/.singularity/metadata/sha256:736a219344fbca3099ce5bd1d2dbfea74b
22b830bac0e85ecca812c2983390cd.tar.gz
Creating empty Singularity writable container 253MB
Creating empty 316MiB image file: lolcow.img
Formatting image with ext3 file system
```

Running image on build host (VirtualBox in this case)



Build from definition file (examples)

- Example: ***meep.def***
- Meep is a open-source electromagnetic equation propagation software.
- Difficult to compile in default Comet environment.

meep.def

- **Some dependency installs. For example:**

  apt-get -y install libopenblas-base
  apt-get -y install libopenblas-dev
  apt-get -y install libgmp-dev
  apt-get -y install libgsl-dev
  apt-get -y install libpng16-dev
  apt-get -y install swig
  apt-get -y install guile-2.0-dev

  …

  …

meep.def

- **Some installs from source:**

```
wget https://www.open-mpi.org/software/ompi/v1.8/downloads/openmpi-1.8.4.tar.gz
 tar -xzvf openmpi-1.8.4.tar.gz
 cd openmpi-1.8.4
 ./configure --prefix=/opt/openmpi-1.8.4
  make all install
 export PATH="/opt/openmpi-1.8.4/bin:${PATH}"
 export LD_LIBRARY_PATH="/opt/openmpi-1.8.4/lib:${LD_LIBRARY_PATH}"
```

meep.def

- Setup container environment variables

```
cd /.singularity.d/env
echo 'export PATH="/opt/openmpi-1.8.4/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/openmpi-1.8.4/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/harminv-1.4.1/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/harminv-1.4.1/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/libctl-4.0.0/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/libctl-4.0.0/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/zlib-1.2.11/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/hdf5-1.10.1/hdf5/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/hdf5-1.10.1/hdf5/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/h5utils-1.13/bin:${PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/fftw-3.3.7/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/fftw-3.3.7/lib:{LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/mpb-1.6.1/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/mpb-1.6.1/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
echo 'export PATH="/opt/meep-1.4.3/bin:${PATH}"' >> 90-environment.sh
echo 'export LD_LIBRARY_PATH="/opt/meep-1.4.3/lib:${LD_LIBRARY_PATH}"' >> 90-environment.sh
```

Build from meep.def

```
root@mahidhar-VirtualBox: /tmp/NEW
root@mahidhar-VirtualBox:/tmp/NEW# !389
singularity build --writable meep.img ./meep.def
Using container recipe deffile: ./meep.def
Sanitizing environment
Adding base Singularity environment to container
I: Retrieving InRelease
I: Checking Release signature
I: Valid Release signature (key id 790BC7277767219C42C86F933B4FE6ACC0B21F32)
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
I: Found additional base dependencies: gcc-5-base gnupg gpgv libapt-pkg5.0 liblz
4-1 libreadline6 libstdc++6 libusb-0.1-4 readline-common ubuntu-keyring
I: Checking component main on http://us.archive.ubuntu.com/ubuntu...
I: Retrieving adduser 3.113+nmu3ubuntu4
I: Validating adduser 3.113+nmu3ubuntu4
I: Retrieving apt 1.2.10ubuntu1
I: Validating apt 1.2.10ubuntu1
I: Retrieving base-files 9.4ubuntu4
I: Validating base-files 9.4ubuntu4
I: Retrieving base-passwd 3.5.39
I: Validating base-passwd 3.5.39
I: Retrieving bash 4.3-14ubuntu1
```

Build from meep.def

```
root@mahidhar-VirtualBox: /home/mahidhar/NEW
 /usr/bin/install -c -m 644 meep.pc '/opt/meep-1.4.3/lib/pkgconfig'
make[2]: Leaving directory '/opt/meep-1.4.3'
make[1]: Leaving directory '/opt/meep-1.4.3'
Adding deffile section labels to container
Adding runscript
Finalizing Singularity container
Calculating final size for metadata...
Skipping checks
Creating empty Singularity writable container 2251MB
Creating empty 2813MiB image file: meep.img
Formatting image with ext3 file system
Image is done: meep.img
Building Singularity image...
Singularity container built: meep.img
Cleaning up...
root@mahidhar-VirtualBox:/home/mahidhar/NEW# singularity shell meep.img
Singularity: Invoking an interactive shell within container...

Singularity meep.img:~> which meep
/opt/meep-1.4.3/bin/meep
Singularity meep.img:~> exit
exit
root@mahidhar-VirtualBox:/home/mahidhar/NEW# clear
```

Special note on GPUs

- Option 1 is to match the driver and CUDA files on the host system. This is what we do for most of our images. Allows us to build from source if needed.

- Option 2 is to use the “--nv” flag which allows you to leverage the driver on the host system.

GPU def example

- Some parts to install drivers (367.48 is the one on Comet)

…

```
dpkg -i nvidia-367_367.48-0ubuntu1_amd64.deb
dpkg -i nvidia-367-dev_367.48-0ubuntu1_amd64.deb
dpkg -i nvidia-modprobe_367.48-0ubuntu1_amd64.deb
dpkg -i libcuda1-367_367.48-0ubuntu1_amd64.deb
```

…

GPU def example

- Some parts to install CUDA libraries

…

```
dpkg -i cuda-cusolver-8-0_8.0.44-1_amd64.deb
dpkg -i cuda-cusolver-dev-8-0_8.0.44-1_amd64.deb
dpkg -i cuda-cublas-8-0_8.0.44-1_amd64.deb
dpkg -i cuda-cublas-dev-8-0_8.0.44-1_amd64.deb
dpkg -i cuda-cufft-8-0_8.0.44-1_amd64.deb
```

…

Singularity “inspect” command

mahidhar@mahidhar-VirtualBox: ~/NEW

```
mahidhar@mahidhar-VirtualBox:~/NEW$ sudo singularity inspect -l -r -d -t -e -j -hf ./vsoch-old.img

{
    "data": {
        "attributes": {
            "deffile": "Bootstrap: docker\nFrom: ubuntu:14.04\n\n%labels\nMAINTAINER vanessasaur\n
WHATAMI dinosaur\n\n%environment\nDINOSAUR=vanessasaurus\nexport DINOSAUR\n\n%files\nrawr.sh /rawr
.sh\n\n%runscript\nexec /bin/bash /rawr.sh\n",
            "help": null,
            "labels": {
                "org.label-schema.usage.singularity.deffile.bootstrap": "docker",
                "MAINTAINER": "vanessasaur",
                "org.label-schema.usage.singularity.deffile": "Singularity",
                "org.label-schema.schema-version": "1.0",
                "WHATAMI": "dinosaur",
                "org.label-schema.usage.singularity.deffile.from": "ubuntu:14.04",
                "org.label-schema.build-date": "2017-10-15T12:52:56+00:00",
                "org.label-schema.usage.singularity.version": "2.4-feature-squashbuild-secbuild.g7
80c84d",
                "org.label-schema.build-size": "333MB"
            },
            "environment": "# Custom environment shell code should follow\n\nDINOSAUR=vanessasauru
s\nexport DINOSAUR\n\n",
            "runscript": "#!/bin/sh \n\nexec /bin/bash /rawr.sh\n",
            "test": null
        },
        "type": "container"
    }
}
mahidhar@mahidhar-VirtualBox:~/NEW$
```

Singularity “run” command

- **Lets try on Comet:**

  module load singularity

  ls -lt /share/apps/examples/workshop/images/vsoch-old.img

  singularity run /share/apps/examples/workshop/images/vsoch-old.img

```
[mahidhar@comet-ln2 workshop]$ module li
Currently Loaded Modulefiles:
  1) intel/2013_sp1.2.144   2) mvapich2_ib/2.1        3) gnutools/2.69
[mahidhar@comet-ln2 workshop]$ module load singularity
[mahidhar@comet-ln2 workshop]$ ls -lt /share/apps/examples/workshop/images/vsoch
-old.img
-rwxr-xr-x 1 mahidhar use300 272629791 May  1 09:52 /share/apps/examples/worksho
p/images/vsoch-old.img
[mahidhar@comet-ln2 workshop]$
[mahidhar@comet-ln2 workshop]$ singularity run /share/apps/examples/workshop/ima
ges/vsoch-old.img
WARNING: Non existent bind point (directory) in container: '/oasis'
WARNING: Non existent bind point (directory) in container: '/projects'
WARNING: Non existent 'bind path' source: '/scratch'
RaawwWWWWWRRRR!!
[mahidhar@comet-ln2 workshop]$
```

Singularity shell

- Good for interactive tests
- Helps with compiling in the image environment.
- Very useful when the image provides the right dependencies that are required for building a particular application
- We have images with dependencies for ***Torch, Caffe, OpenCL, and Julia.***

Example of compiling using image

```
[mahidhar@comet-ln2 TUTORIAL]$ singularity shell ./ubuntu-openmpi.img
WARNING: Non existent 'bind path' source: '/scratch'
Singularity: Invoking an interactive shell within container...

Singularity ubuntu-openmpi.img:/oasis/scratch/comet/mahidhar/temp_project/Singularity/TUTORIAL>
mpif90 -o hello_mpi_ubuntu ./hello_mpi.f90

Singularity ubuntu-openmpi.img:/oasis/scratch/comet/mahidhar/temp_project/Singularity/TUTORIAL>
mpirun -np 2 ./hello_mpi_ubuntu
 node         0 : Hello world
 node         1 : Hello world

Singularity ubuntu-openmpi.img:/oasis/scratch/comet/mahidhar/temp_project/Singularity/TUTORIAL>
```

Singularity exec

- Allows for command to be executed in image environment.
- This is the primary method of running in batch on Comet using the singularity images.
- Sometimes we have a simple python script and we can call a single exec command
- Can bundle workflow into a script and then execute the script via exec command.

Tensorflow via Singularity

```
#!/bin/bash
#SBATCH --job-name="TensorFlow"
#SBATCH --output="TensorFlow.%j.%N.out"
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=6
#SBATCH --gres=gpu:1
#SBATCH -t 01:00:00

#Run the job
#

module load singularity
singularity exec /share/apps/gpu/singularity/sdsc_ubuntu_gpu_tflow.img lsb_relea
se -a
singularity exec /share/apps/gpu/singularity/sdsc_ubuntu_gpu_tflow.img python -m
 tensorflow.models.image.mnist.convolutional
```

Tensorflow via Singularity

- **Change to the examples directory:**
  cd /home/$USER/TUTORIAL/TensorFlow

- **Submit the job:**
  sbatch TensorFlow.sb

Tensorflow Example: Output

```
Distributor ID: Ubuntu
Description: Ubuntu 16.04 LTS
Release: 16.04
Codename: xenial
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcudnn.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcurand.so locally
I tensorflow/core/common_runtime/gpu/gpu_init.cc:102] Found device 0 with properties:
name: Tesla K80
major: 3 minor: 7 memoryClockRate (GHz) 0.8235
pciBusID 0000:85:00.0
Total memory: 11.17GiB
Free memory: 11.11GiB
I tensorflow/core/common_runtime/gpu/gpu_init.cc:126] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:136] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:838] Creating TensorFlow device (/gpu:0) -> (device: 0, name:
Tesla K80, pci bus id: 0000:85:00.0)
Extracting data/train-images-idx3-ubyte.gz
…
Step 8500 (epoch 9.89), 11.6 ms
Minibatch loss: 1.601, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.9%
Test error: 0.9%
```

Example with --nv option for GPUs

```
[mahidhar@comet-30-07 CUDA]$ singularity exec --nv docker://nvidia/cuda:8.0-devel-centos6 ./matrixMul
Docker image path: index.docker.io/nvidia/cuda:8.0-devel-centos6
Cache folder set to /home/mahidhar/.singularity/docker
[7/7] |==================================| 100.0%
Creating container runtime...
tar: usr/include/arpa/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/asm/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/asm-generic/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/bits/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/c++/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/drm/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/gnu/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/linux/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/mtd/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/net/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
tar: usr/include/netash/.wh..wh..opq: implausibly old time stamp 1969-12-31 16:00:00
```

```
WARNING: Non existent bind point (directory) in container: '/oasis'
WARNING: Non existent bind point (directory) in container: '/projects'
WARNING: Non existent bind point (directory) in container: '/scratch'
WARNING: Skipping user bind, non existent bind point (file) in container: '/usr/bin/nvidia-smi'
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "Tesla K80" with compute capability 3.7

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 226.64 GFlop/s, Time= 0.578 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
[mahidhar@comet-30-07 CUDA]$
```

Singularity and MPI

```
#!/bin/bash
#SBATCH --job-name="meep"
#SBATCH --output="meep.%j.%N.out"
#SBATCH --partition=shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4
#SBATCH --export=ALL
#SBATCH -t 00:20:00
### Set Container to use
CONTAINER=/share/apps/examples/workshop/images/meep.img
## List the container to avoid automount issue
ls -lt /share/apps/examples/workshop/images/meep.img
## Run serial job
#time -p singularity exec ${CONTAINER} /opt/meep-1.4.3/bin/meep ./parallel-wvgs-force.ctl
## Run MPI job
module purge
module load gnutools
module load intel
module load openmpi_ib
module load singularity
time -p mpirun -np 4 singularity exec ${CONTAINER} /opt/meep-1.4.3/bin/meep ./parallel-wvgs-force.ctl
```

Multi-Node runs via Singularity

- **Easy for cases with MPI backends - we already saw this in the first talk.**
- **ML/DL frameworks can be a bit more complicated with process launches on remote nodes (need to be done via image)**
- **One example:**
  - Tensorflow - processes are launched once => just need to wrap the launch tasks.

Script snippet from multi-node TF

```
#Start the parameter server
cd /scratch/$USER/$SLURM_JOBID
cp $SLURM_SUBMIT_DIR/example.sh .
cp $SLURM_SUBMIT_DIR/example.py .
./example.sh "ps" 0 2>&1 > $SLURM_SUBMIT_DIR/ps.$SLURM_JOBID.log &


#Run the worker processes remotely
ssh $H2 $SLURM_SUBMIT_DIR/run_worker.sh $SLURM_SUBMIT_DIR
/scratch/$USER/$SLURM_JOBID 0 > $SLURM_SUBMIT_DIR/worker0_$SLURM_JOBID.log
&
sleep 10s

ssh $H3 $SLURM_SUBMIT_DIR/run_worker.sh $SLURM_SUBMIT_DIR
/scratch/$USER/$SLURM_JOBID 1 >
$SLURM_SUBMIT_DIR/worker1_$SLURM_JOBID.log
```

*** For the multinode case, you need to untar the tensorflow.tar file in your home directory.

Commands we didn't use!

- **Image command group**
  - image.export
  - image.import
  - image.create
- **Instance command group**
  - Useful for running services like databases and web servers
  - instance.start
  - instance.list
  - instance.stop
- **Persistent overlay options**

Summary

- Singularity enables several applications on Comet.
- Examples: 1) need newer OS environment, 2) only have binaries that need specific OS, 3) import docker images with pipeline
- Can develop images on laptop and move to Comet. Persistent overlays can help do compilations on final hardware and also have data included.
- Can run multi-node jobs with MPI