Introduction to SDSC Comet

Mahidhar Tatineni

UC Riverside Nov 2, 2018

Comet
“HPC for the long tail of science”

iPhone panorama photograph of 1 of 2 server rows

Overview

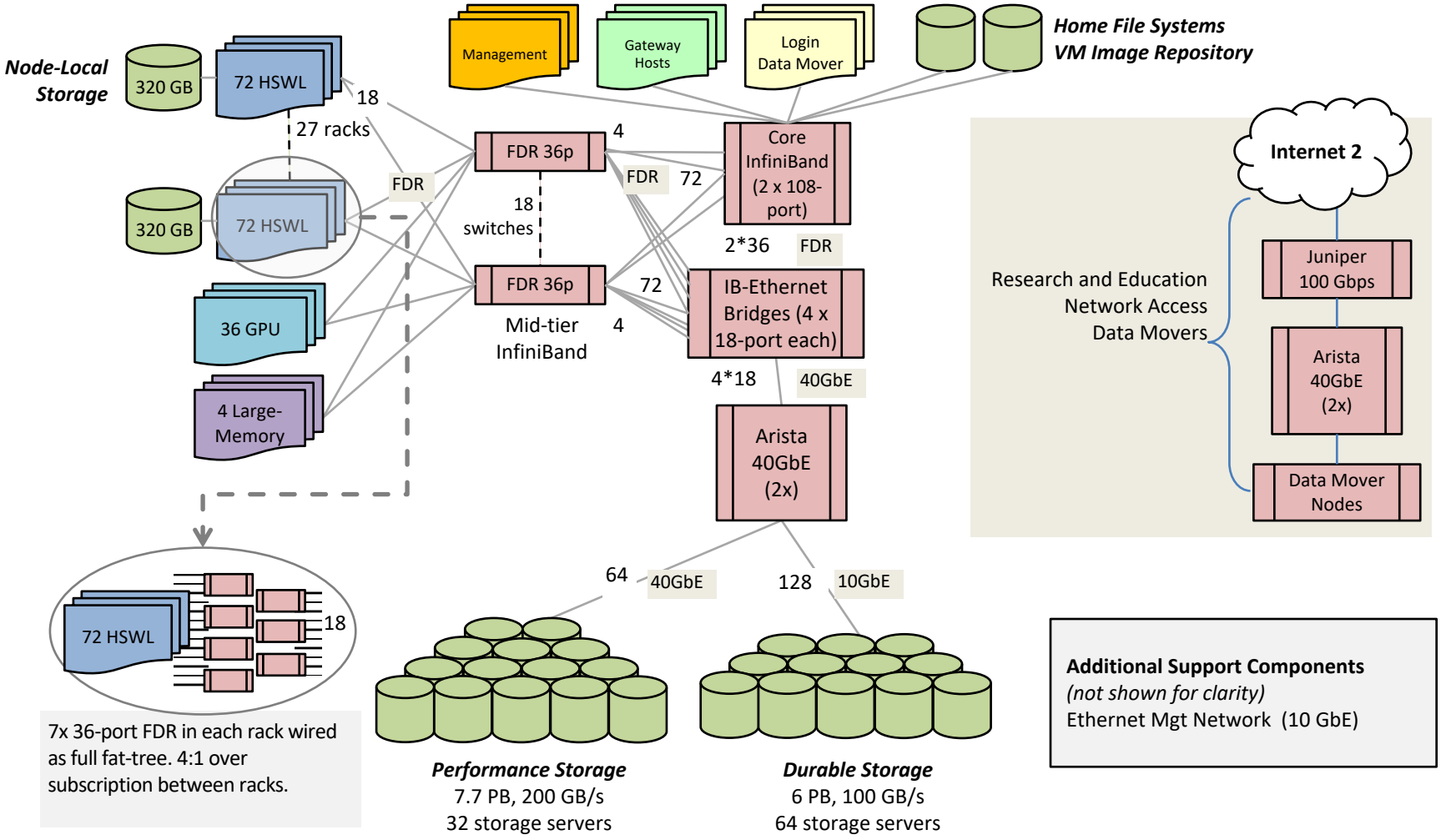
- Comet architecture details
- Software stack
- Running jobs on CPU, GPU resources - hands on examples
- Overview of containerization on Comet using Singularity
- Building singularity containers and running jobs - hands on examples

Comet: System Characteristics

- **Total peak flops ~2.1 PF**
- **Dell primary integrator**
  - Intel Haswell processors w/ AVX2
  - Mellanox FDR InfiniBand
- **1,944 standard compute nodes (46,656 cores)**
  - Dual CPUs, each 12-core, 2.5 GHz
  - 128 GB DDR4 2133 MHz DRAM
  - 2*160GB GB SSDs (local disk)
- **72 GPU nodes**
  - 36 nodes same as standard nodes *plus* Two NVIDIA K80 cards, each with dual Kepler3 GPUs
  - 36 nodes, with 4 P100 GPUs per node
- **4 large-memory nodes**
  - 1.5 TB DDR4 1866 MHz DRAM
  - Four Haswell processors/node
  - 64 cores/node
- **Hybrid fat-tree topology**
  - FDR (56 Gbps) InfiniBand
  - Rack-level (72 nodes, 1,728 cores) full bisection bandwidth
  - 4:1 oversubscription cross-rack
- **Performance Storage (Aeon)**
  - 7.6 PB, 200 GB/s; Lustre
  - Scratch & Persistent Storage segments
- **Durable Storage (Aeon)**
  - 6 PB, 100 GB/s; Lustre
  - Automatic backups of critical data
- **Home directory storage**
- **Gateway hosting nodes**
- **Virtual image repository**
- **100 Gbps external connectivity to Internet2 & ESNet**

Comet Network Architecture

InfiniBand compute, Ethernet Storage



Getting Started

- **System Access – Logging in**
  - Linux/Mac – Use available ssh clients.
  - ssh clients for windows – Putty, Cygwin
    - http://www.chiark.greenend.org.uk/~sgtatham/putty/
  - Login hosts for the SDSC Comet:
    - comet.sdsc.edu

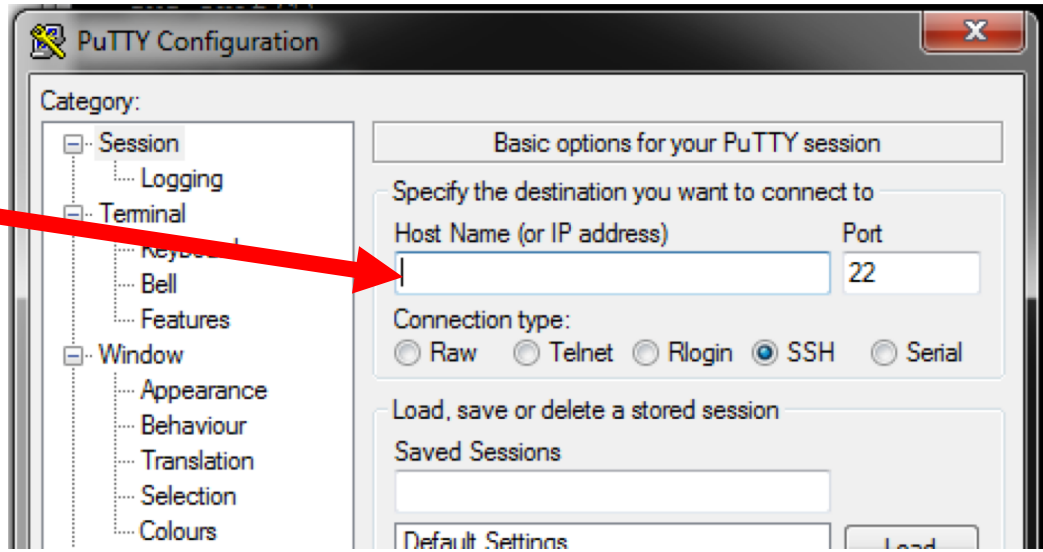
Logging into Comet

Mac/Linux:

```
ssh username@comet.sdsc.edu
```

Windows (PuTTY):

comet.sdsc.edu



Comet: Filesystems

- **Lustre filesystems – Good for scalable large block I/O**
  - Accessible from all compute and GPU nodes.
  - /oasis/scratch/comet - 2.5PB, peak performance: 100GB/s. Good location for storing large scale scratch data during a job.
  - /oasis/projects/nsf - 2.5PB, peak performance: 100 GB/s. Long term storage.
  - ***Not good for lots of small files or small block I/O***.
- **SSD filesystems**
  - /scratch local to each native compute node – 210GB on regular compute nodes, 285GB on GPU, large memory nodes, 1.4TB on selected compute nodes.
  - SSD location is good for writing small files and temporary scratch files. Purged at the end of a job.
- **Home directories (/home/$USER)**
  - Source trees, binaries, and small input files.
  - ***Not good for large scale I/O.***

Comet: System Environment

- **Modules used to manage environment for users.**
- **Default environment:**

```
$ module li
Currently Loaded Modulefiles:
  1) intel/2013_sp1.2.144   2) mvapich2_ib/2.1        3) gnutools/2.69
```

- **Listing available modules:**

```
$ module av
------------------------ /opt/modulefiles/mpi/.intel ------------------------
intelmpi/2016.3.210(default) mvapich2_ib/2.1(default)
mvapich2_gdr/2.1(default)    openmpi_ib/1.8.4(default)
mvapich2_gdr/2.2
-------------------- /opt/modulefiles/applications/.intel --------------------
atlas/3.10.2(default)     lapack/3.6.0(default)     scalapack/2.0.2(default)
boost/1.55.0(default)     mxml/2.9(default)         slepc/3.6.2(default)
…
…
```

Comet: System Environment

- **Loading modules:**

  $ module load fftw/3.3.4

  $ module li

  Currently Loaded Modulefiles:

  1) intel/2013_sp1.2.144 3) gnutools/2.69

  2) mvapich2_ib/2.1 4) fftw/3.3.4

- **See what a module does:**

  $ module show fftw/3.3.4

```
-------------------------------------------------------------------
/opt/modulefiles/applications/.intel/fftw/3.3.4:
module-whatis fftw
module-whatis Version: 3.3.4
module-whatis Description: fftw
module-whatis Compiler: intel
module-whatis MPI Flavors: mvapich2_ib openmpi_ib
setenv FFTWHOME /opt/fftw/3.3.4/intel/mvapich2_ib
prepend-path PATH /opt/fftw/3.3.4/intel/mvapich2_ib/bin
prepend-path LD_LIBRARY_PATH /opt/fftw/3.3.4/intel/mvapich2_ib/lib
prepend-path LIBPATH /opt/fftw/3.3.4/intel/mvapich2_ib/lib
-------------------------------------------------------------------
```

Parallel Programming

- **Comet supports MPI, OpenMP, and Pthreads for parallel programming. Hybrid modes are possible.**
- **GPU nodes support CUDA, OpenACC.**
- **MPI**
  - Default: mvapich2_ib/2.1
  - Other options: openmpi_ib/1.8.4 (and 1.10.2), Intel MPI
  - mvapich2_gdr: GPU direct enabled version
- **OpenMP:** All compilers (GNU, Intel, PGI) have OpenMP flags.
- Default Intel Compiler: **intel/2013_sp1.2.144;** *Versions 2015.2.164,2016.3.210,2018.1.163 available.*

Running Jobs on Comet

- **Important note: Do not run on the login nodes - even for simple tests.**
- **All runs must be via the Slurm scheduling infrastructure.**
  - Interactive Jobs: Use **srun** command:

    srun --pty --nodes=1 --ntasks-per-node=24 -p debug -t 00:30:00 --wait 0 /bin/bash
  - Batch Jobs: Submit batch scripts from the login nodes. Can choose:
    - Partition (details on upcoming slide)
    - Time limit for the run (maximum of 48 hours)
    - Number of nodes, tasks per node
    - Memory requirements (if any)
    - Job name, output file location
    - Email info, configuration

Slurm Partitions

Queue Name	Max Walltime	Max Nodes	Comments
compute	48 hrs	72	Used for access to regular compute nodes
gpu	48 hrs	4	Used for access to the GPU nodes
gpu-shared	48 hrs	1	Used for shared access to a partial GPU node
shared	48 hrs	1	Single-node jobs using fewer than 24 cores
large-shared	48 hrs	1	Single-node jobs using large memory up to 1.45 TB
debug	30 mins	2	Used for access to debug nodes

- **Specified using -p option in batch script. For example:**

#SBATCH -p gpu

Slurm Commands

- **Submit jobs using the sbatch command:**

  $ sbatch Localscratch-slurm.sb

  Submitted batch job 8718049

- **Check job status using the squeue command:**

  $ squeue -u $USER

```
             JOBID PARTITION     NAME     USER      ST       TIME  NODES  NODELIST(REASON)
           8718049   compute      localscr mahidhar  PD       0:00      1           (Priority)
```

- **Once the job is running:**

  $ squeue -u $USER

```
             JOBID PARTITION     NAME     USER    ST       TIME  NODES  NODELIST(REASON)
           8718064     debug      localscr mahidhar  R       0:02      1        comet-14-01
```

Comet Compute Nodes

2-Socket (Total 24 cores) Intel Haswell Processors

Hands On Examples using:

(1) MPI

(2) Local scratch

Comet – Compiling/Running Jobs

- **Copy and change to directory (assuming you already copied the TUTORIAL directory):**

  cd /home/$USER/TUTORIAL/MPI

- **Verify modules loaded:**

  module list

  Currently Loaded Modulefiles:

  1) intel/2013_sp1.2.144 2) mvapich2_ib/2.1 3) gnutools/2.69

- **Compile the MPI hello world code:**

  mpif90 -o hello_mpi hello_mpi.f90

- **Verify executable has been created:**

  ls -lt hello_mpi

  -rwxr-xr-x 1 mahidhar sdsc 721912 Mar 25 14:53 **hello_mpi**

- **Submit job from IBRUN directory:**

  cd /home/$USER/TUTORIAL/MPI/IBRUN

  sbatch --res=UCRES hellompi-slurm.sb

Comet: Hello World on compute nodes

The submit script is hellompi-slurm.sb:

```
#!/bin/bash
#SBATCH --job-name="hellompi"
#SBATCH --output="hellompi.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 01:30:00

#This job runs with 2 nodes, 24 cores per node for a total of 48 cores.
#ibrun in verbose mode will give binding detail

ibrun -v ./hello_mpi
```

Comet: Hello World on compute nodes

IBRUN: Command is ../hello_mpi
IBRUN: Command is /share/apps/examples/MPI/hello_mpi
…
…
IBRUN: MPI binding policy: compact/core for 1 threads per rank (12 cores per socket)
IBRUN: Adding MV2_CPU_BINDING_LEVEL=core to the environment
IBRUN: Adding MV2_ENABLE_AFFINITY=1 to the environment
IBRUN: Adding MV2_DEFAULT_TIME_OUT=23 to the environment
IBRUN: Adding MV2_CPU_BINDING_POLICY=bunch to the environment
…
…
IBRUN: Added 8 new environment variables to the execution environment
IBRUN: Command string is [mpirun_rsh -np 48 -hostfile /tmp/rssSvauaJA -export /share/apps/examples/MPI/hello_mpi]
node 18 : Hello world
node 13 : Hello world
node 2 : Hello world
node 10 : Hello world

Using SSD Scratch

```
#!/bin/bash
#SBATCH --job-name="localscratch"
#SBATCH --output="localscratch.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 01:30:00

#Copy binary to SSD
cp IOR.exe /scratch/$USER/$SLURM_JOBID
#Change to local scratch (SSD) and run IOR benchmark
cd /scratch/$USER/$SLURM_JOBID
#Run IO benchmark
ibrun -np 4 $WKDIR/IOR.exe -F -t 1m -b 4g -v –v >  IOR.out.$SLURM_JOBID
#Copy out data you need
cp IOR.out.$SLURM_JOBID $SLURM_SUBMIT_DIR
```

Using SSD Scratch

- **Snapshot on the node during the run:**

```
[mahidhar@comet-20-71 ~]$ squeue -u $USER
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          15580587   compute localscr mahidhar  R       0:11      1 comet-20-71

[mahidhar@comet-20-71 ~]$ cd /scratch/mahidhar/15580587/

[mahidhar@comet-20-71 15580587]$ ls -lt
total 9173887
-rw-r--r-- 1 mahidhar use300 1939865600 Apr 16 23:25 testFile.00000002
-rw-r--r-- 1 mahidhar use300 3865051136 Apr 16 23:25 testFile.00000000
-rw-r--r-- 1 mahidhar use300 2490368000 Apr 16 23:25 testFile.00000001
-rw-r--r-- 1 mahidhar use300 2777677824 Apr 16 23:25 testFile.00000003
-rw-r--r-- 1 mahidhar use300       1088 Apr 16 23:25 IOR.out.15580587
-rwxr-xr-x 1 mahidhar use300     346872 Apr 16 23:25 IOR.exe
```

- **Performance from single node (in log file copied back):**
  - Max Write: 606.49 MiB/sec (635.95 MB/sec)
  - Max Read: 19028.71 MiB/sec (19953.05 MB/sec)

Multi-node SSD example

$HOME/TUTORIAL/LOCALSCRATCH2

```
#!/bin/bash
#SBATCH --job-name="localscratch2"
#SBATCH --output="localscratch2.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 00:10:00

#Get a list of hosts
export SLURM_NODEFILE=`generate_pbs_nodefile`
cat $SLURM_NODEFILE > nodes.list.$SLURM_JOBID
uniq nodes.list.$SLURM_JOBID > nodes.unq.list

#Change to local scratch (SSD) and run IOR benchmark
cd /scratch/$USER/$SLURM_JOBID

#Run IO benchmark
ibrun -np 48 $SLURM_SUBMIT_DIR/IOR.exe -F -t 1m -b 4m -v -v -w

#Change back to submit dir
cd $SLURM_SUBMIT_DIR
```

```
#List files on both nodes
for (( nn=1; nn<=$SLURM_NNODES; nn++ ))
 do
   p="`sed -n ${nn}p nodes.unq.list`"
   echo "Files on $p"
   ssh $p /bin/ls /scratch/$USER/$SLURM_JOBID
 done


#Tar back the results from each node
for (( nn=1; nn<=$SLURM_NNODES; nn++ ))
 do
   p="`sed -n ${nn}p nodes.unq.list`"
   echo "Tar files on $p"
   ssh $p /bin/tar -cvf $SLURM_SUBMIT_DIR/node$nn.tar /scratch/$USER/$SLURM_JOBID
 done

rm nodes.unq.list
rm nodes.list.$SLURM_JOBID
```

Comet GPU Nodes

2 NVIDIA K-80 Cards (4 GPUs total) per node.

[1] CUDA code compile and run example

[2] Hands On Examples using Singularity to enable Tensorflow

Compiling CUDA Example

- **Load the CUDA module:**

  module load cuda

- **Compile the code:**

  cd /home/$USER/TUTORIAL/CUDA

  nvcc -o matmul -I. matrixMul.cu

- **Submit the job:**

  sbatch --res=UCRES cuda.sb

CUDA Example: Batch Submission Script

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

#Load the cuda module
module load cuda

#Run the job
./matmul
```

Introduction to containerization using Singularity on Comet

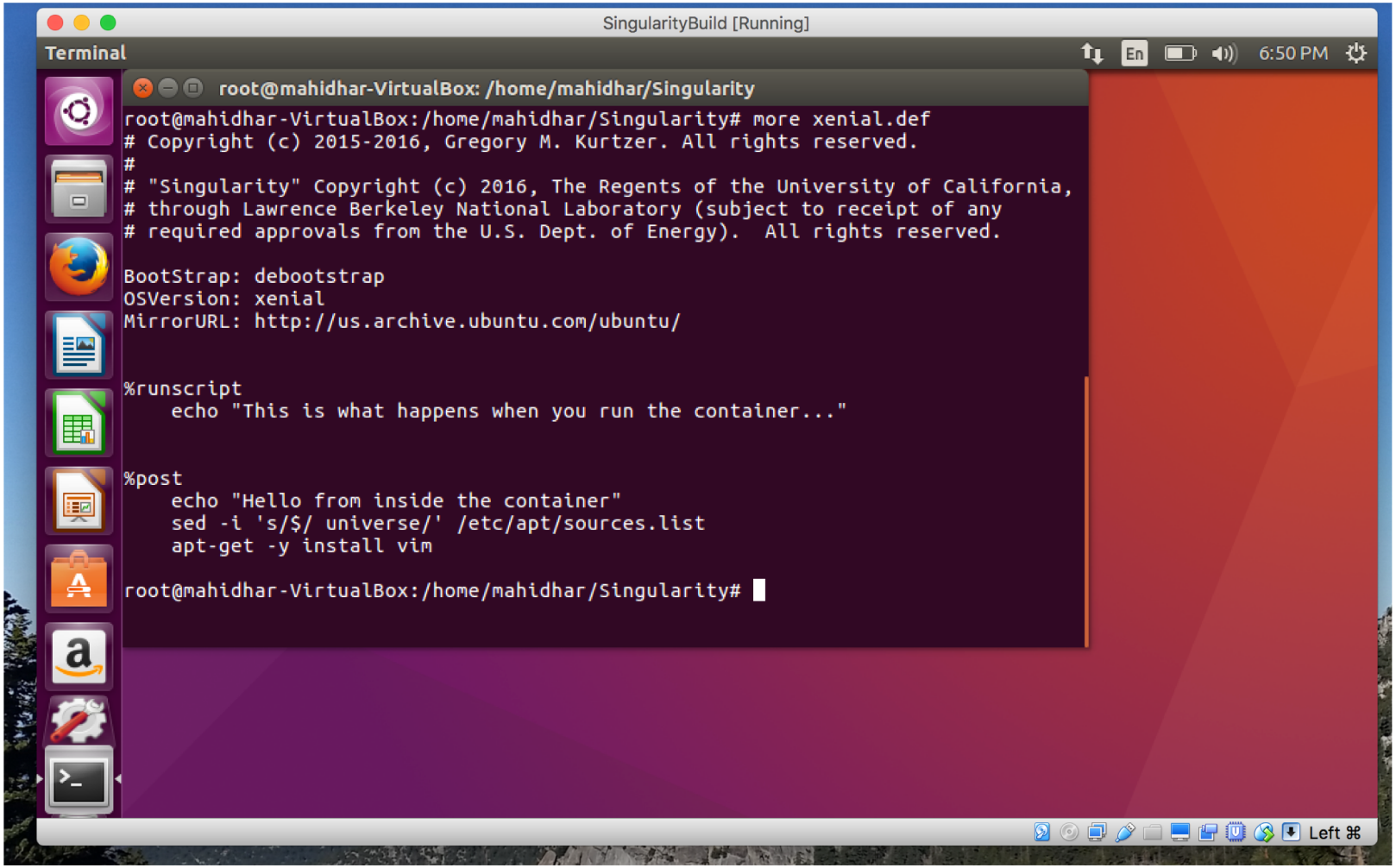
SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

Singularity: Provides Flexibility for OS Environment

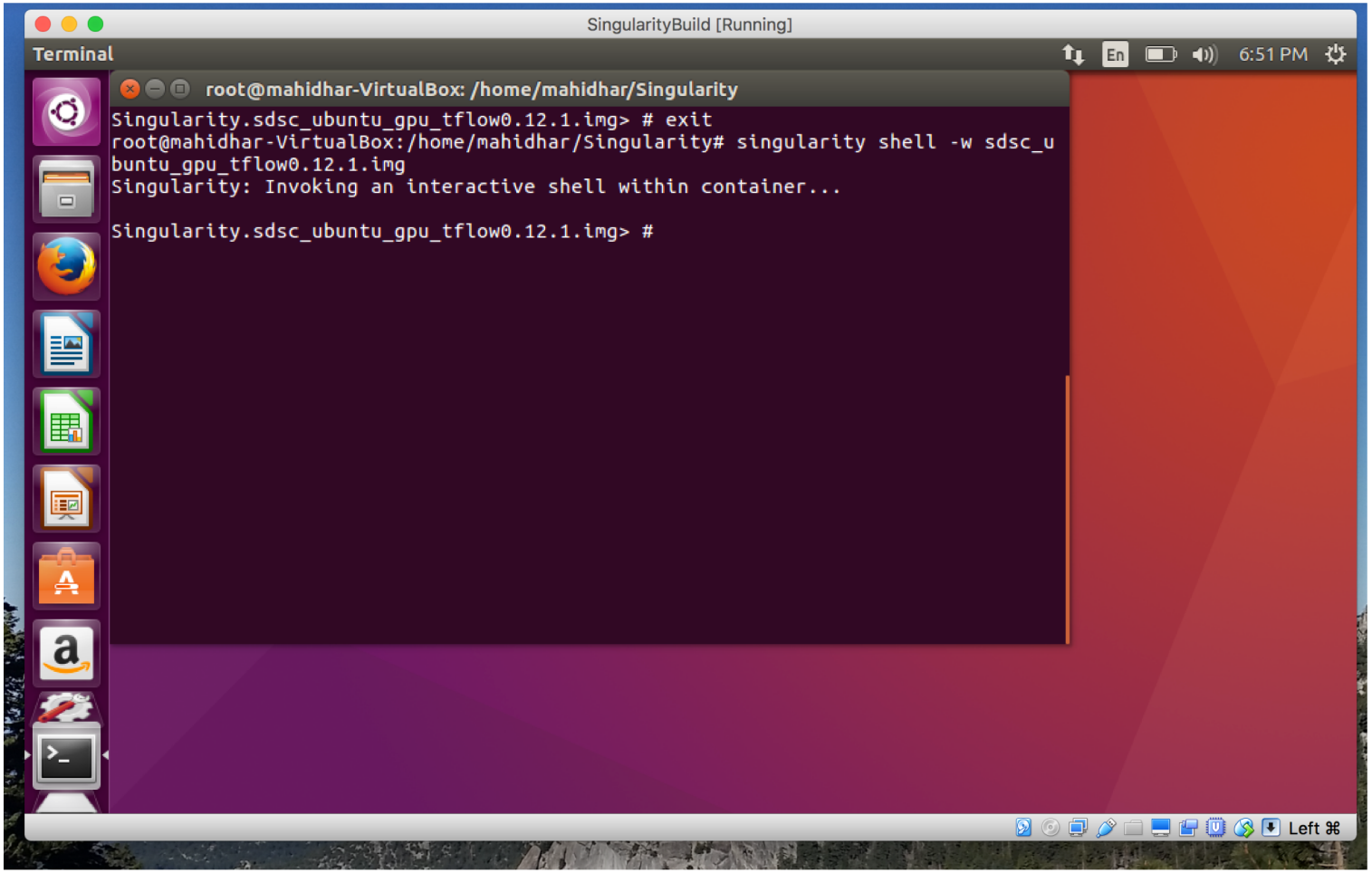
- Singularity (http://singularity.lbl.gov)is a relatively new development that has become very popular on Comet.
- Singularity allows groups to easily migrate complex software stacks from their campus to Comet.
- Singularity runs in user space, and requires very little special support – in fact it actually reduces it in some cases.
- We have roughly 20 groups running this on Comet.
- Applications include: Tensorflow, Torch, Fenics, and custom user applications.
- Docker images can be imported into Singularity.

Singularity: Provides Flexibility for OS Environment



- **Above snapshot shows a definition file on a virtual box on personal laptop. The definition file lets you build a singularity image.**

Singularity: Provides Flexibility for OS Environment



- **Can open image in write mode on laptop via singularity and install packages (like Tensorflow). So an existing image on Comet can serve as a starting point.**

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

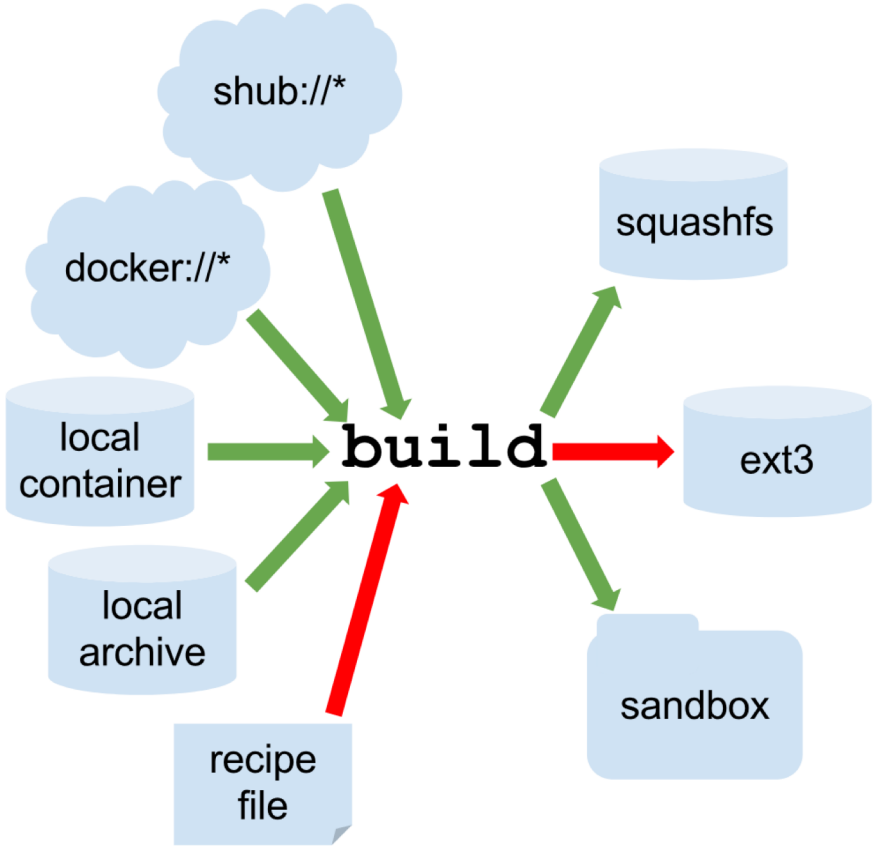
- **singularity pull command can be used to download images from**
  - *Singularity Hub*: e.g. singularity pull shub://vsoch/hello-world
  - *Docker Hub :* e.g. singularity pull --name funny.simg docker://godlovedc/lolcow

- **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#
```

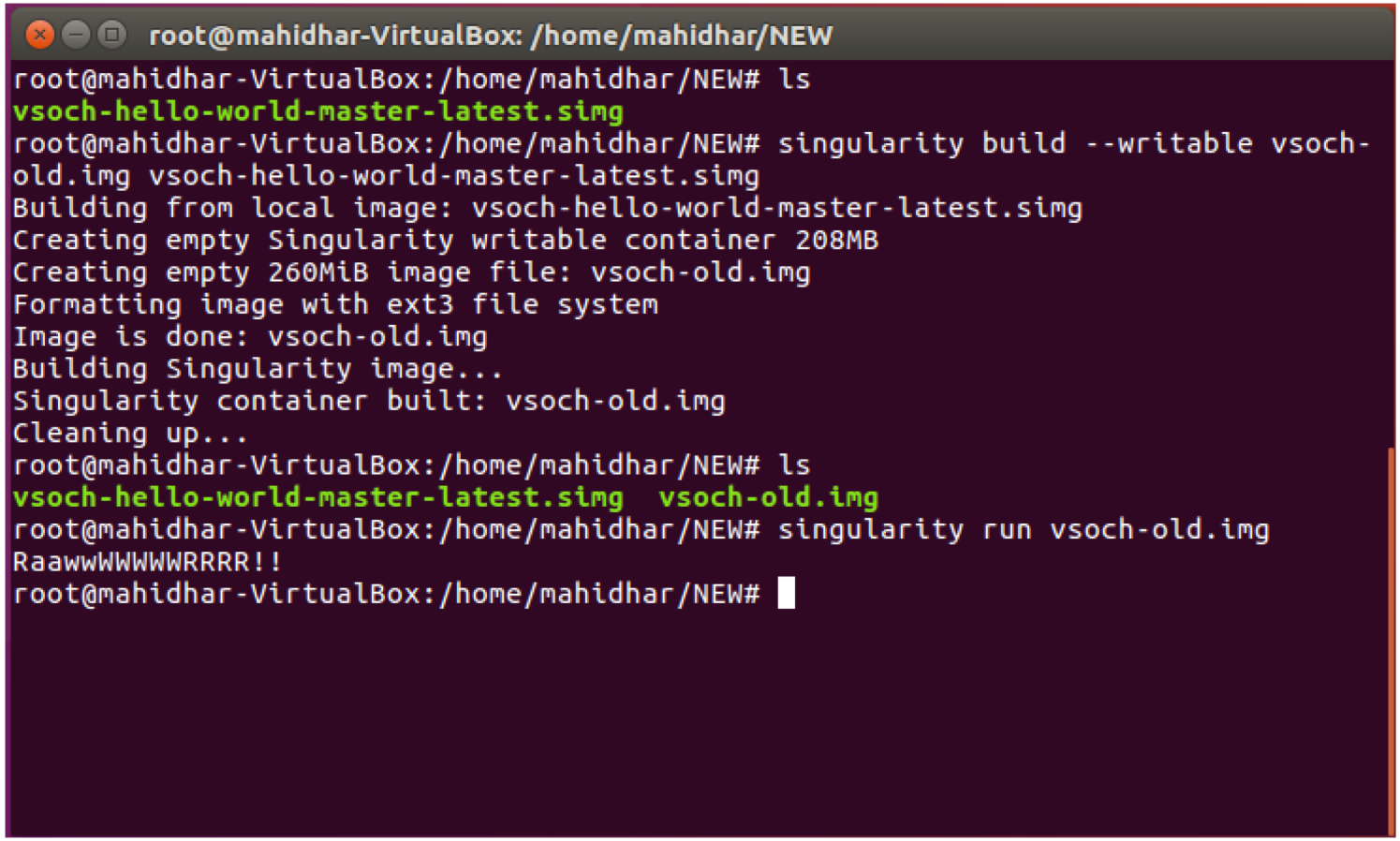
Singularity ”build” option



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

Build by converting existing image

- We can convert the image we just pulled.



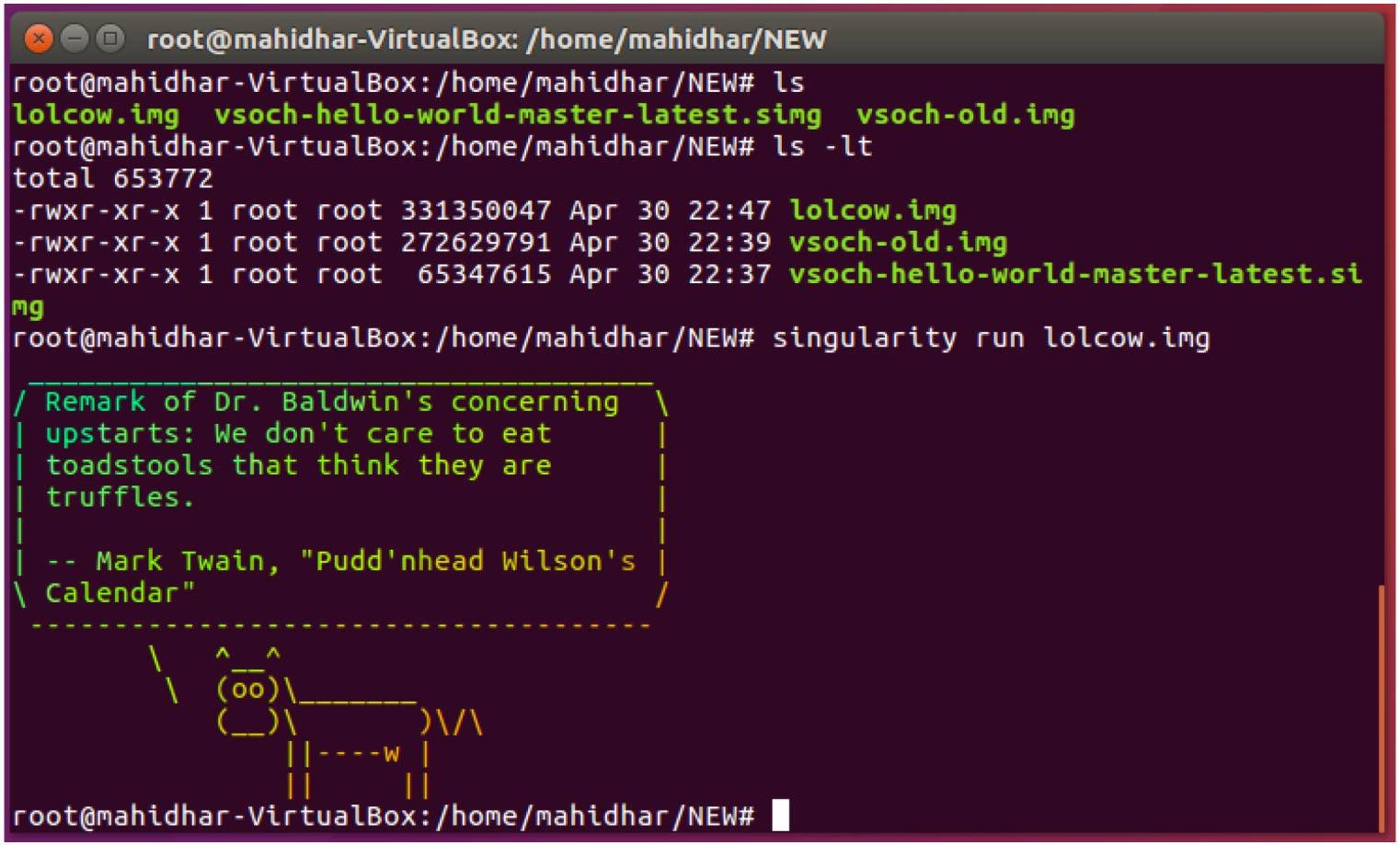
Build from docker image

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

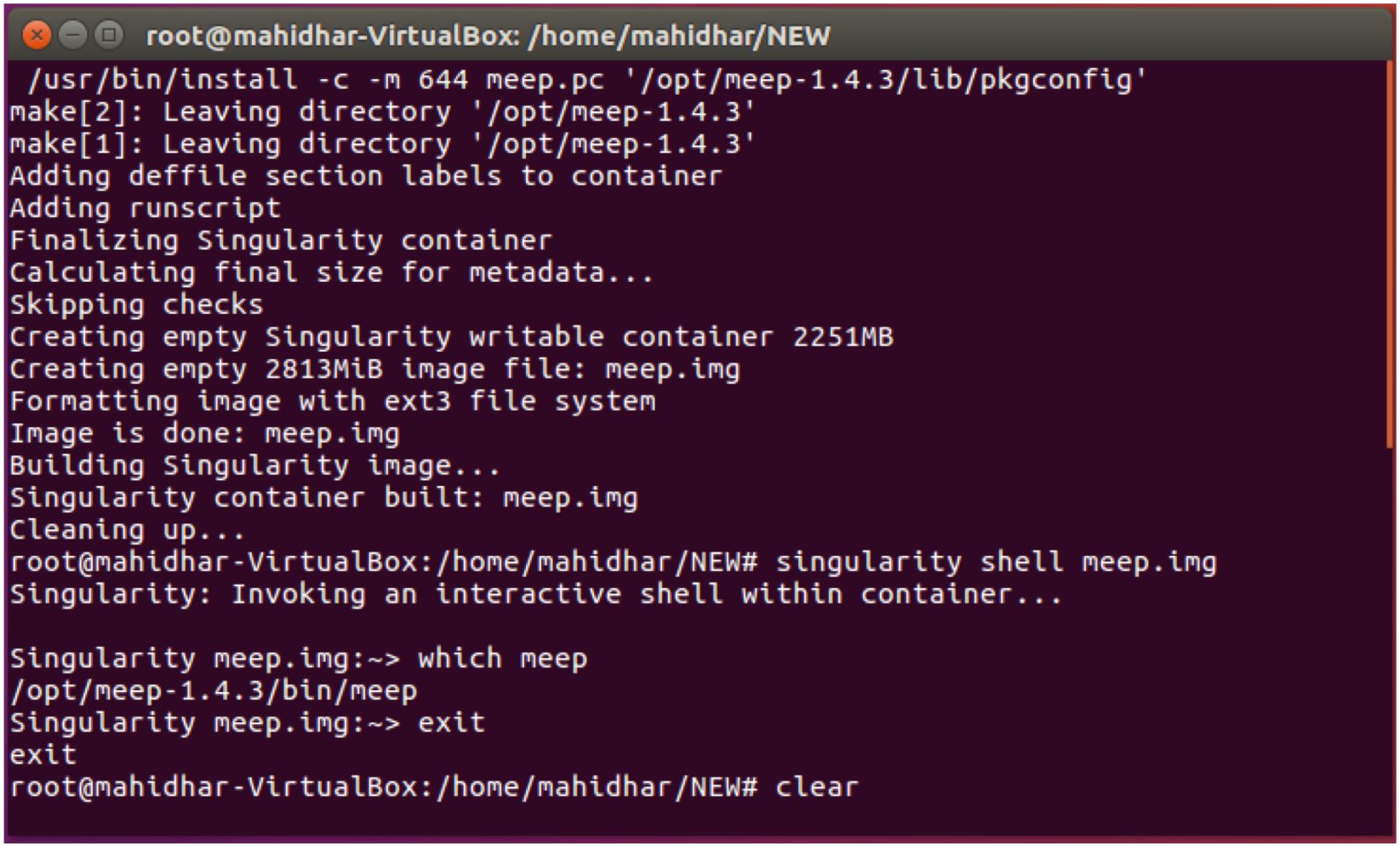
- Setup container environment variables

```
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



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/UCB2018/images/vsoch-old.img

  singularity run /share/apps/examples/UCB2018/images/vsoch-old.img

```
[mahidhar@comet-ln2 TUTORIAL]$ module li
Currently Loaded Modulefiles:
  1) intel/2013_sp1.2.144   2) mvapich2_ib/2.1        3) gnutools/2.69
[mahidhar@comet-ln2 TUTORIAL]$ module load singularity
[mahidhar@comet-ln2 TUTORIAL]$ ls -lt /share/apps/examples/UCB2018/images/vsoch-old.img
-rwxr-xr-x 1 mahidhar use300 272629791 May  1 09:52 /share/apps/examples/UCB2018/images/vsoch-old.im
g
[mahidhar@comet-ln2 TUTORIAL]$ singularity run /share/apps/examples/UCB2018/images/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 TUTORIAL]$
```

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 --res=UCRES 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/UCB2018/images/meep.img
## List the container to avoid automount issue
ls -lt /share/apps/examples/UCB2018/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
```

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**

Machine Learning/Deep Learning on Comet via Singularity

- Bulk of the Singularity usage on Comet is for machine learning/deep learning applications.
- Lot of these packages are constantly upgraded and the dependency list is difficult to update in the standard Comet environment.
- Install options
  - Singularity image provides dependencies and user can compile actual application from source. e.g. Torch
  - Entire dependency stack and the application is in the image. e.g Keras with backend to TensorFlow and some additional python libraries
- Run options
  - Most cases are run on single GPU nodes (4 GPUs at most)
  - Can access this via Jupyter notebooks
  - Multi-node options are possible but difficult to set up via singularity.

Multi-Node runs via Singularity

- **Easy for cases with MPI backends - we saw this earlier.**
- **ML/DL frameworks can be a bit more complicated with process launches on remote nodes (need to be done via image)**
- **Two examples:**
  - Tensorflow - processes are launched once => just need to wrap the launch tasks.
  - MXNET : Lot of remote commands => need to patch the script that does this part.

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.

MXNET

- **Remote launch controlled by:**
  - incubator-mxnet/dmlc-core/tracker/dmlc_tracker/ssh.py

- **Change made:**

```
prog = get_env(pass_envs) + ' cd ' + working_dir + ';
/opt/singularity/bin/singularity exec
/oasis/scratch/comet/username/temp_project/singularity/mxn
et-gpu.img ' + (' '.join(args.command))
```

MXNET interactive snippet

```
scontrol show hostname > hosts

for NODE in `cat hosts`; do ssh $NODE cp -r /home/username/mxnet/incubator-mxnet/example/image-classification /scratch/$USER/$SLURM_JOBID; done

cd /scratch/$USER/$SLURM_JOBID/image-classification

singularity shell /oasis/scratch/comet/username/temp_project/singularity/mxnet-gpu.img

python /home/username/mxnet/incubator-mxnet/tools/launch.py -n 2 --launcher ssh -H hosts python train_mnist.py --gpus 0,1,2,3 --network lenet --kv-store dist_sync

for NODE in `cat hosts`; do ssh $NODE killall -s 9 python ; done
exit
```

Summary of ML/DL information

- Most ML/DL packages on Comet are enabled via Singularity
- Single node runs are easy to wrap with “singularity exec” commands
- Can launch and use Jupyter notebooks from within Singularity containers (will do hands on in afternoon).
- Multi-node runs need some work but feasible in most cases.

Summary

- Comet can be directly accessed using a ssh client.
- Always run via the batch scheduler – for both interactive and batch jobs. ***Do not run on the login nodes.***
- Choose your filesystem wisely – Lustre parallel filesystem for large block I/O. SSD based filesystems for small block I/O, lots of small files. ***Do not use home filesystem for intensive I/O of any kind.***
- Comet can handle MPI, OpenMP, Pthreads, Hybrid, CUDA, and OpenACC jobs. See /share/apps/examples for details!

Summary (Continued)

- 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