

# Introduction to Comet

*SDSC's 2PF HPC Resource*

*Mahidhar Tatineni*  
UCLA, April 4, 2017

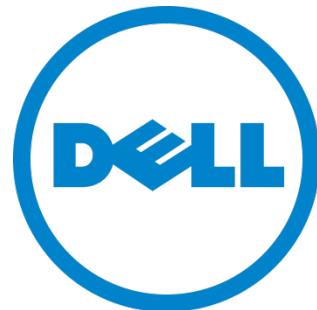




# UC San Diego



**SDSC**  
SAN DIEGO SUPERCOMPUTER CENTER



Ψ

INDIANA UNIVERSITY

*This work supported by the National Science Foundation, award ACI-1341698.*



**SDSC** SAN DIEGO  
SUPERCOMPUTER CENTER

UC San Diego

# Comet

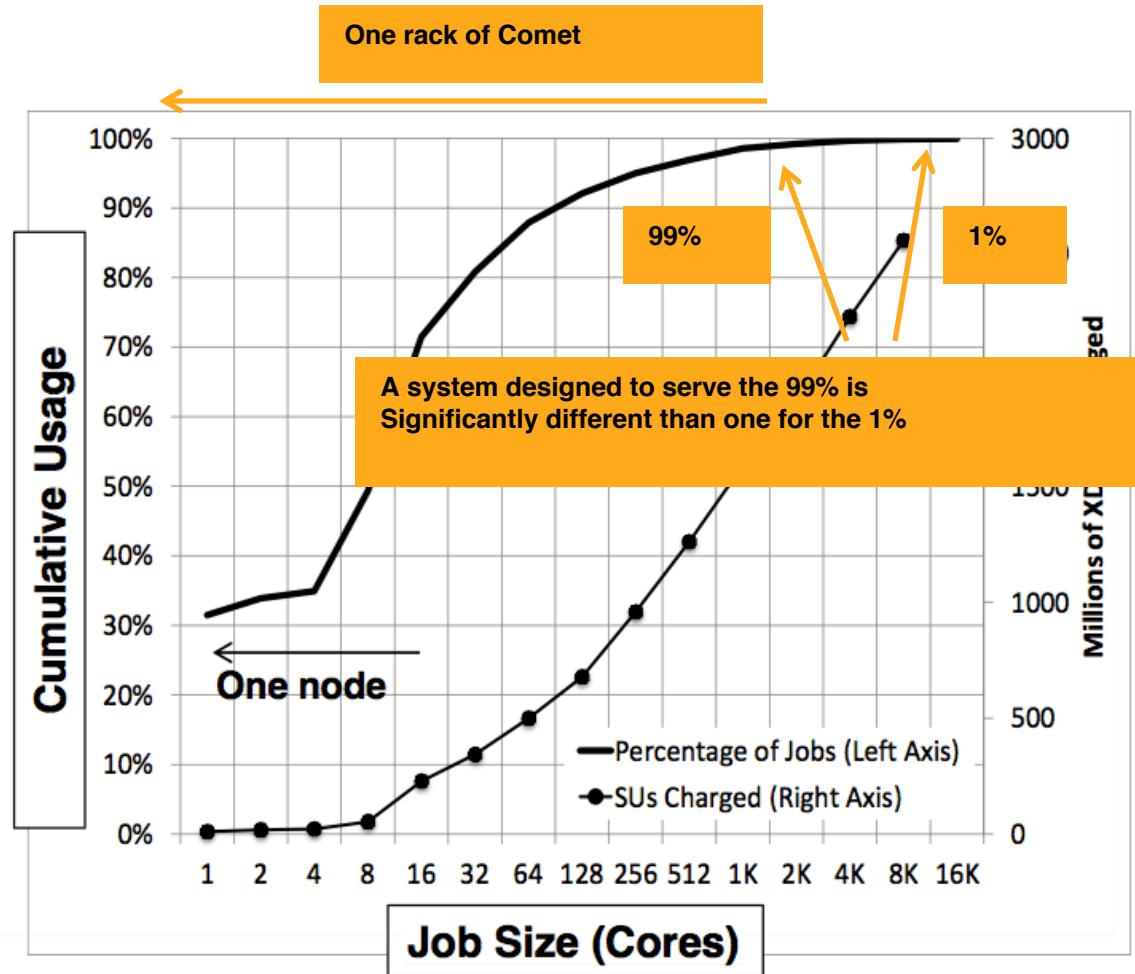
## “HPC for the long tail of science”



**iPhone panorama photograph of 1 of 2 server rows**

# *Data extracted from NSF's XDMoD data base informed a design that reflects the way researchers actually use HPC systems*

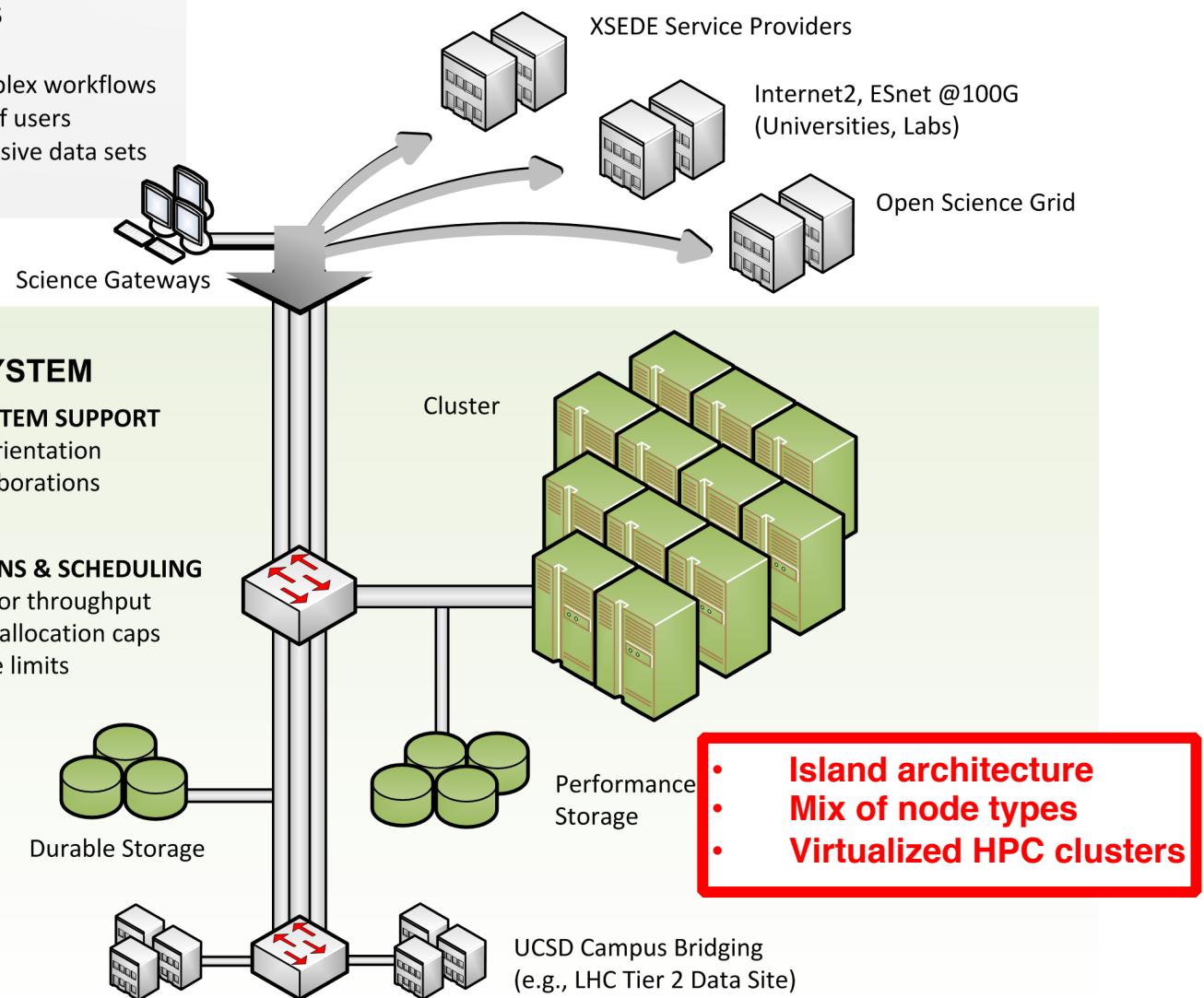
- 99% of jobs run on NSF's HPC resources in 2012 used <2,048 cores
- And consumed >50% of the total core-hours across all NSF resources



# Comet Built to Serve the 99%

## CHALLENGES OUR PROPOSAL ADDRESSES

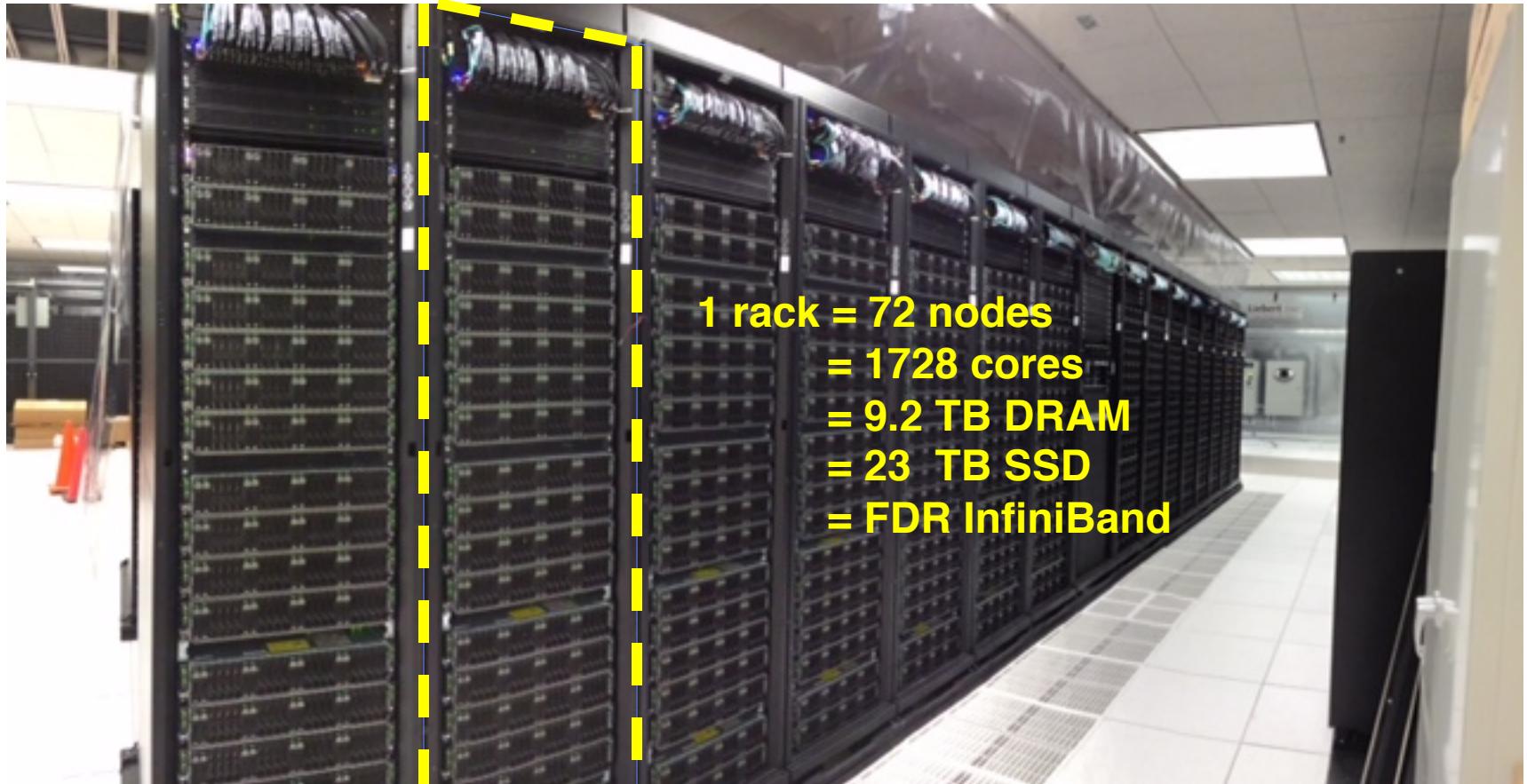
- ✓ Attract new users and communities
- ✓ Support diverse applications with complex workflows
- ✓ Ensure responsiveness for thousands of users
- ✓ Transfer, store, analyze, and share massive data sets
- ✓ Integrate with XSEDE



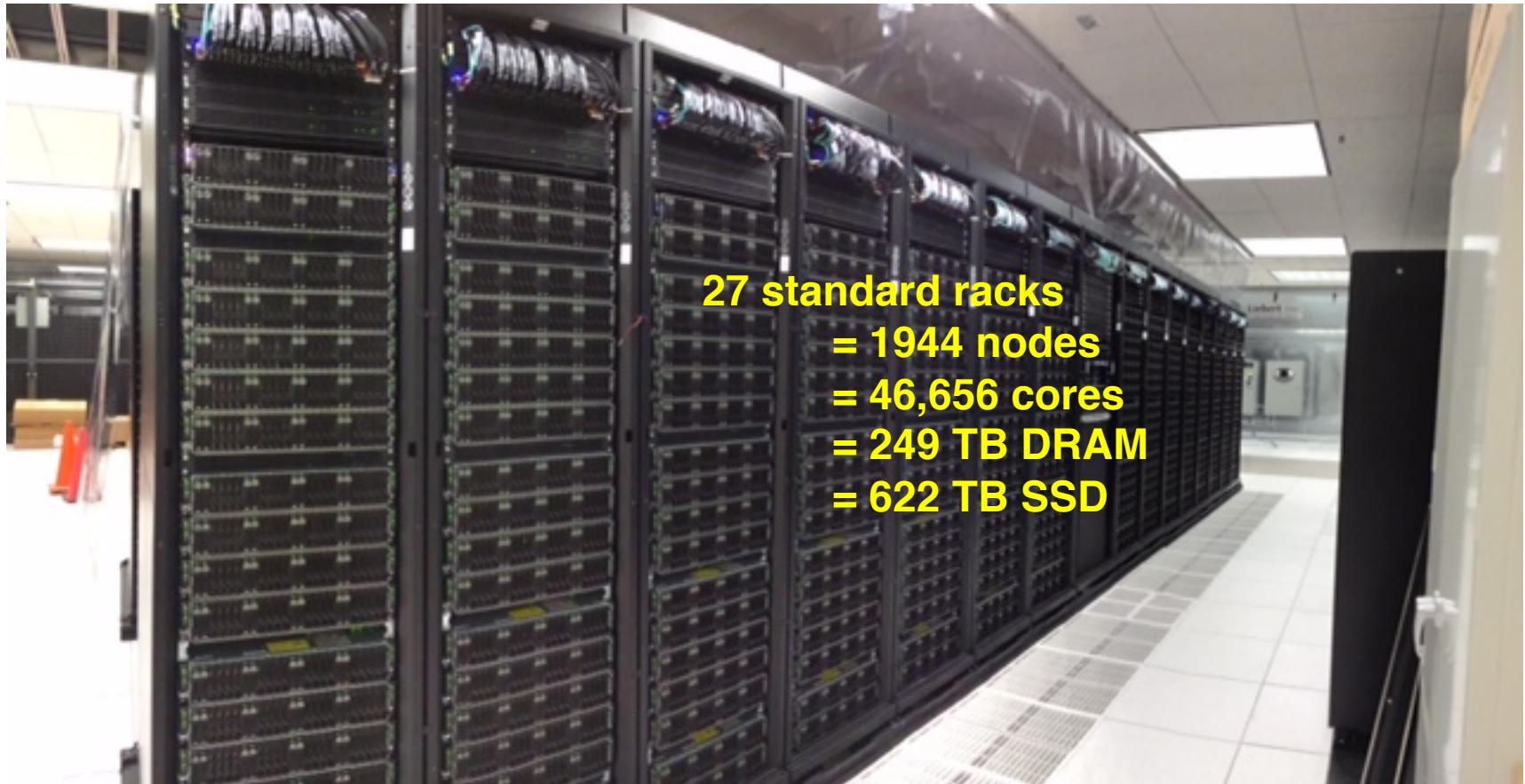
# Comet: System Characteristics

- 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)
- **36 GPU nodes**
  - Same as standard nodes *plus*
  - Two NVIDIA K80 cards, each with dual Kepler3 GPUs
- **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**

# **~67 TF supercomputer in a rack**



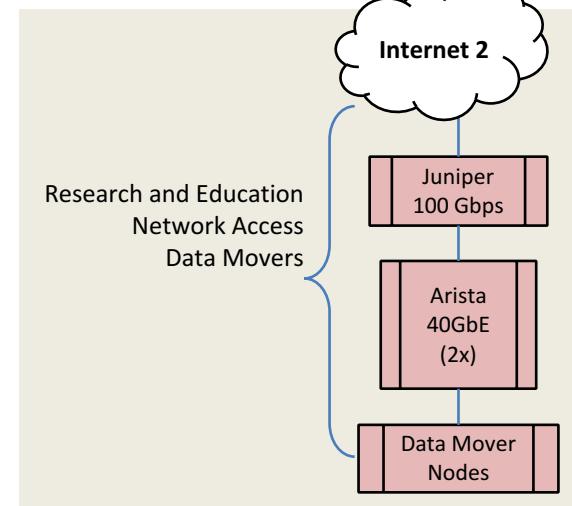
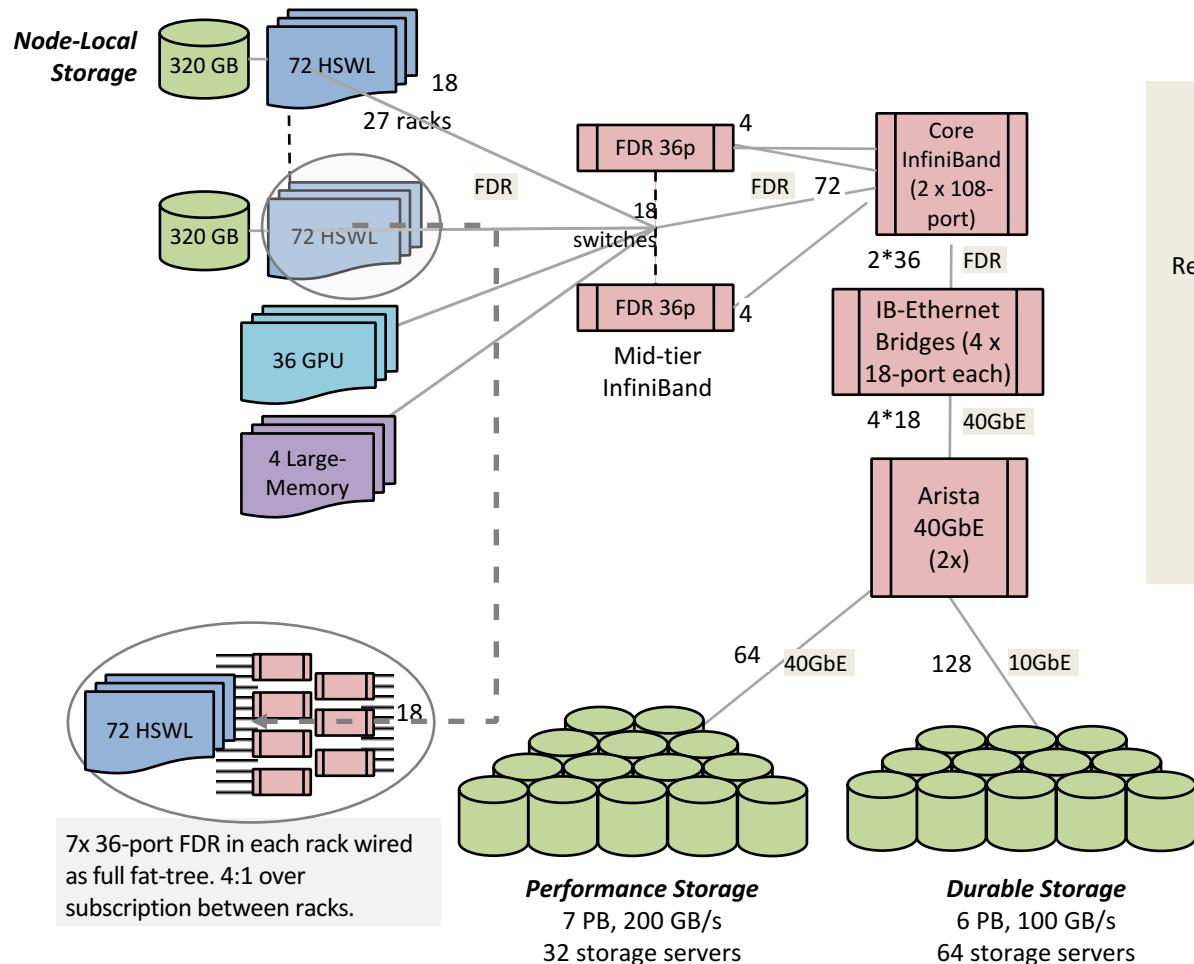
# And 27 single-rack supercomputers



27 standard racks  
= 1944 nodes  
= 46,656 cores  
= 249 TB DRAM  
= 622 TB SSD

# Comet Network Architecture

## InfiniBand compute, Ethernet Storage



**Additional Support Components**  
*(not shown for clarity)*

- Ethernet Mgt Network (10 GbE)
- NFS Servers for Home Directories
- Virtual Image Repository
- Gateway/Portal Hosting Nodes
- Login Nodes
- Rocks Management Nodes

# **Comet Flexibility Addresses Diverse Needs**

- **Wide range of hardware options**
  - Large number of regular compute nodes (**1,944**) with **128GB** of memory and **210GB of local flash**.
  - Subset of compute nodes have **1.5TB of local flash**
  - 4 large memory (**1.5TB RAM**) nodes
  - **36 GPU nodes** with 4 GPUs each (in 2 K80s)
- **Flexible Software Environment**
  - **Rich set of applications** (>100) in regular compute environment
  - **Hadoop/Spark capability** can be enabled within regular scheduler environment.
  - Supports **Singularity based containerization** to enable other Linux based environments (for example Ubuntu). Users can upload their own images!
  - Virtual Clusters (VC) – see operational bullet below.
- **Flexible Operations**
  - **Flexible scheduler environment** – shared and exclusive queues, long running jobs, focus on quick turn around time
  - Research Groups/communities, who have people in their group with **expert system administration skills**, can build their **own virtual clusters** with a custom OS and custom operational setup.

# Extensive Software/Applications Stack

- **Applications packaged into “Rocks Rolls”** that are built and deployed on any of the SDSC systems. **Benefits wider community deploying software** on their Rocks clusters.
- Efficient system administration pooling software install/testing efforts from different projects/machines
- Users benefit from a familiar applications environment across SDSC systems.
- **Rolls made available on Github\*** for all applications installed on Comet => easy replication of environment if users/groups wish to do so.

```
mahidhar ~ mahidhar@comet-ln3:~ ssh 89x35
-----
/opt/modulefiles/applications -----
abaqus/6.11-2          lammps/20151207(default)
abaqus/6.14-1(default)  llvm/3.6.2(default)
abinit/7.10.5(default)  mafft/7.187(default)
abyss/1.5.2(default)    matlab/2015b(default)
amber/15(default)       matt/1.00(default)
apbs/1.4.2(default)     migrate/3.6.11(default)
bamtools/2.3.0(default) migrate/3.6.8
bbcp/14.09.02.00.0(default) miRDeep2/0.0.7(default)
bbftp/3.2.1(default)    miso/0.5.3(default)
beagle/2.1(default)     mkl/11.1.2.144(default)
beast/1.8.0              mkl/11.2.2.164
beast/1.8.1              mlden/5.0.7(default)
beast/2.1.3(default)    mpc/1.0.3(default)
beadtools/2.22.1(default) mpfr/3.1.2(default)
biopython/1.65(default)  mpiblast/1.6.0(default)
blastools/1l(default)   namd/2.10(default)
bismark/0.13.1(default) namd/2.9
blast/2.2.30(default)   node/0.12.6(default)
blat/35l(default)        nwchem/6.6(default)
bowtie/1.1.1(default)   octave/4.0.0(default)
bowtie2/2.2.4(default)  openbabel/2.3.2(default)
bwa/0.7.12(default)    picard/1.130(default)
bx-python/0.7.3(default) plink/1.9(default)
cp2k/2.6.2(default)    polymake/2.14(default)
cpmd/3.17.1(default)   proj/4.9.1(default)
cuda/6.5                 pysam/0.8.3(default)
cuda/7.0(default)        qchem/4.3.2(default)
cufflinks/2.2.1(default) qe/5.3.0(default)
dendropy/4.0.3(default) qiime/1.9.1(default)
edena/3.131028(default) R/3.2.3(default)
eigen/3.2.7(default)    randfold/2.0(default)
fastqc/0.11.3(default)  rapidminer/6.1.0(default)
raxml/8.2.3(default)
```



\*<https://github.com/sdsc/?query=roll>

# *Compilers, Applications, Libraries*

Category	List of Software/Libraries
Compilers	Intel, PGI, GNU, MVAPICH2,OPENMPI, IntelMPI
Bioinformatics	BamTools, BEAGLE, BEAST, BEAST 2, bedtools, Bismark, BLAST, BLAT, Bowtie, Bowtie 2, BWA, Cufflinks, DPPDiv, Edena, FastQC, FastTree, FASTX-Toolkit, FSA, GARLI, GATK, GMAP-GSNAP, IDBA-UD, MAFFT, MrBayes, PhyloBayes, Picard, PLINK, QIIME, RAxML, SAMtools, SOAPdenovo2, SOAPSnp, SPAdes, TopHat, Trimmomatic, Trinity, Velvet
Chemistry	CPMD, CP2K, GAMESS, Gaussian, MOPAC, NWChem, Q-Chem, VASP
Molecular Dynamics	AMBER, Gromacs, LAMMPS, NAMD
Engineering	ABAQUS
Data Analysis/Analytics	Hadoop 1, Hadoop 2 (with YARN), Spark, R, Weka, KNIME, Hadoop-RDMA, Spark-RDMA, ENVI; <i>Singularity Enabled Apps:</i> Torch, Keras, Tensorflow, Caffe, FEniCS
Visualization	VisIt, IDL; <i>Singularity Enabled:</i> Paraview
Numerical libraries	ATLAS, FFTW, GSL, LAPACK, MKL, ParMETIS, PETSc, ScaLAPACK, SPRNG, Sundials, SuperLU, Trilinos
Debugging/Profiling	DDT, PAPI, TAU, mpiP
GPU enabled software	AMBER, GROMACS, NAMD, LAMMPS, BEAST, HOOMD, mvapich2-gdr, RELION

# **Data Intensive Computing & Visualization Stack**

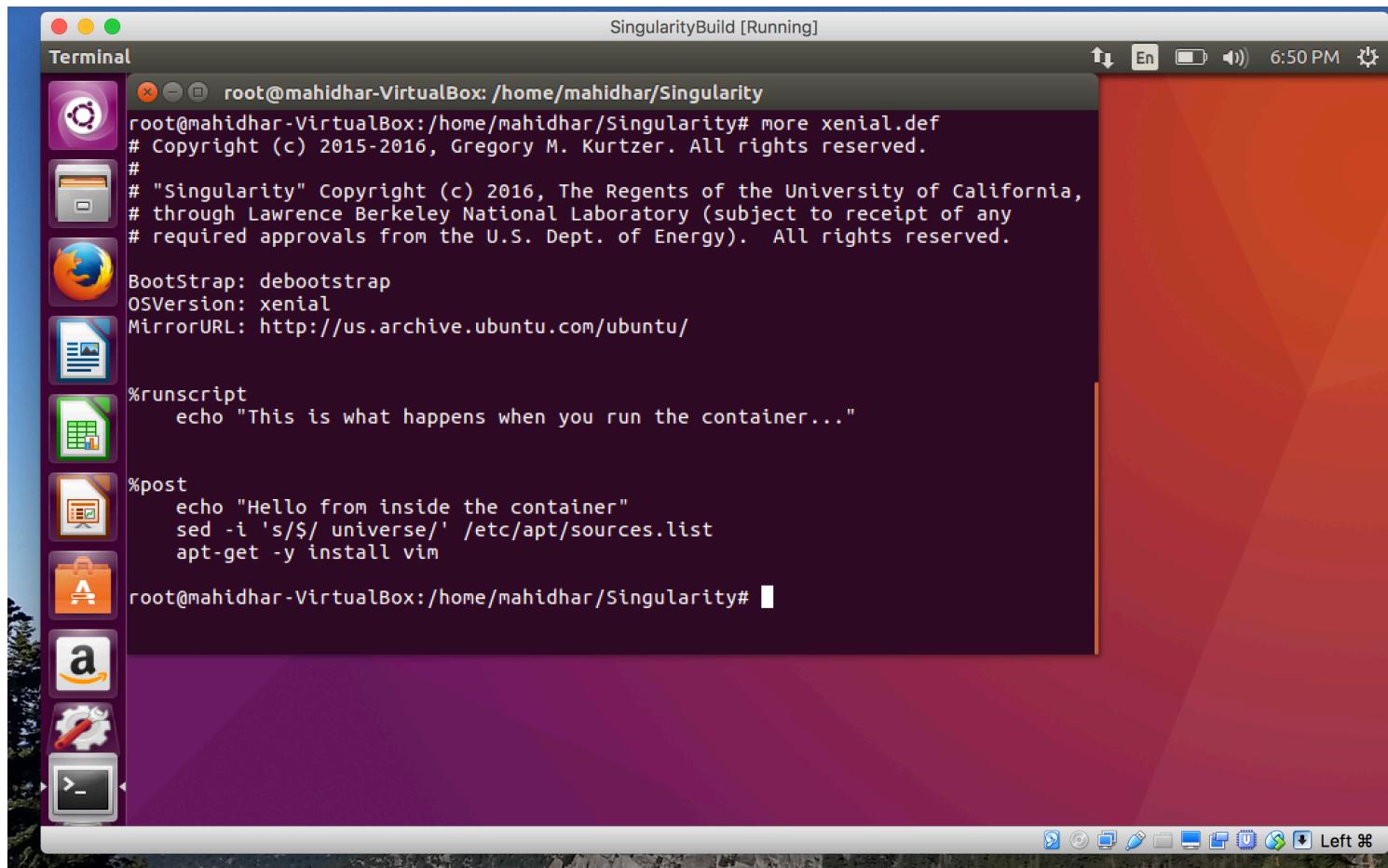
- Comet's high performance computing and data components enable data intensive computing
- High speed Lustre filesystem with an aggregated peak measured data rate of 100 GB/s
- Several libraries and packages enable data intensive computing and visualization:
  - **R** – Software environment for statistical computing and graphics
  - **Weka** – Tools for data analysis and predictive modeling
  - **RapidMiner** – Environment for machine learning, data mining, text mining, and predictive analytics
  - **Matlab, Octave, IDL**
  - **VisIt**
  - **Paraview (via Singularity)**
- The myHadoop infrastructure was developed to enable use **Hadoop and Spark** for distributed data intensive analysis **via the regular scheduler**.
- Several compute and GPU based data analytics tools enabled via Singularity (more details in Singularity slide).

# **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, Paraview, Torch, Fenics, and custom user applications.
- Singularity was not available as an option during Comet's design. Adds another element of flexibility to Comet.
- Docker images can be imported into Singularity.
- Not possible to run services within Singularity environment ->**Virtual Clusters cover this class of applications.**

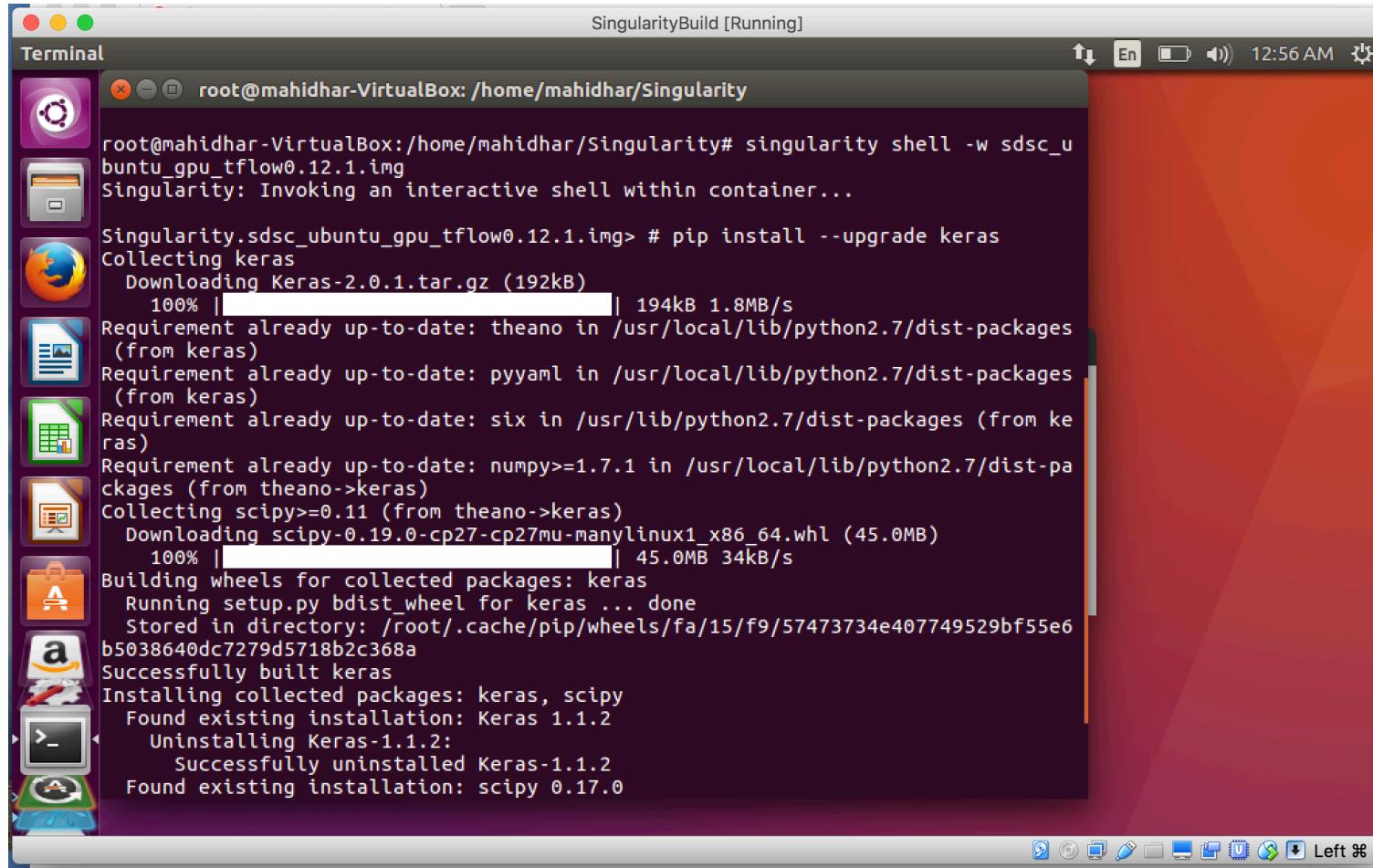
14

# *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*



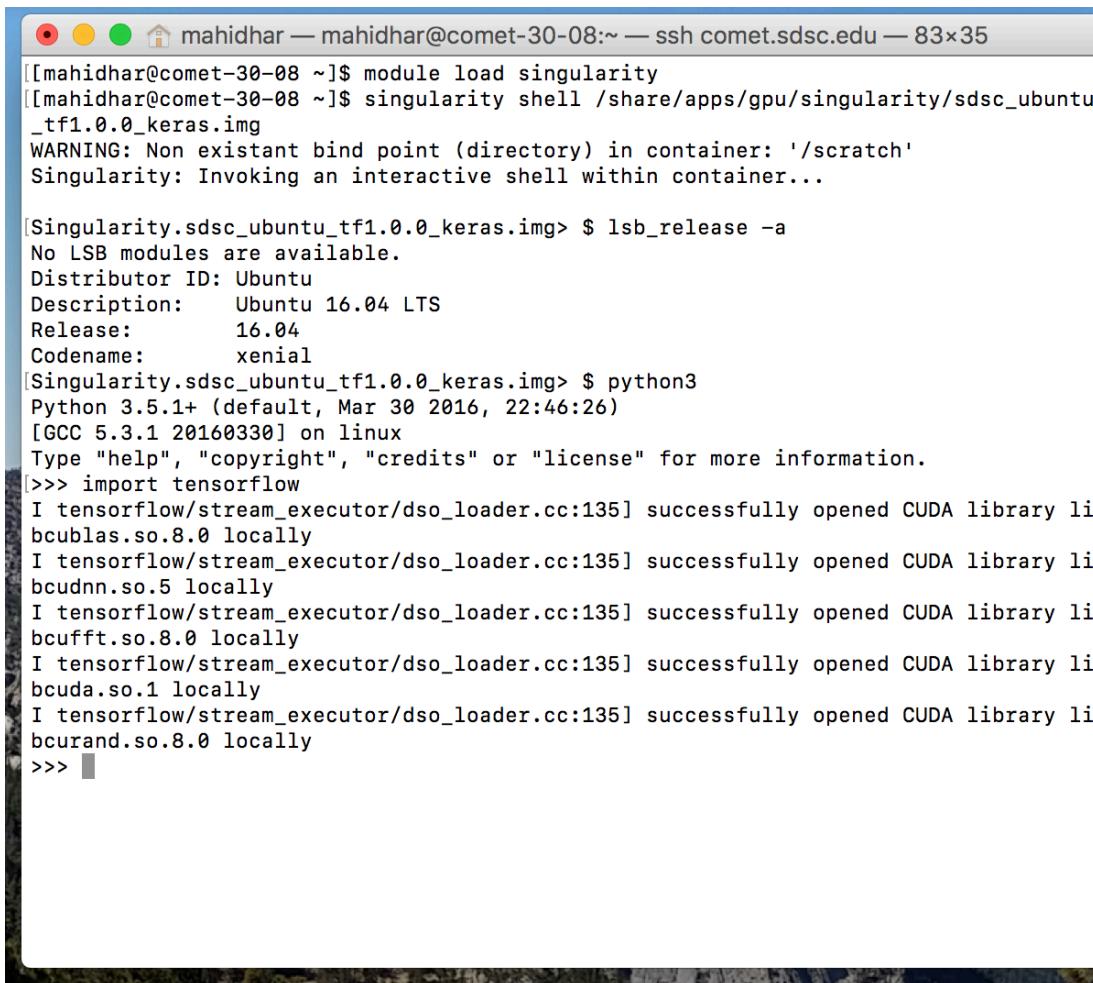
The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "SingularityBuild [Running]". The terminal content shows a root shell session on a virtual machine named "mahidhar-VirtualBox". The user runs "singularity shell -w sdsc\_ubuntu\_gpu\_tflow0.12.1.img", which invokes an interactive shell within the container. Inside the container, the user runs "# pip install --upgrade keras", which installs Keras version 2.0.1. The terminal also shows the download progress of Keras and other dependencies like theano, pyyaml, six, numpy, and scipy. Finally, it installs the collected packages, upgrading Keras from 1.1.2 to 0.17.0.

```
root@mahidhar-VirtualBox:/home/mahidhar/Singularity# singularity shell -w sdsc_ubuntu_gpu_tflow0.12.1.img
Singularity: Invoking an interactive shell within container...
Singularity.sdsc_ubuntu_gpu_tflow0.12.1.img> # pip install --upgrade keras
Collecting keras
  Downloading Keras-2.0.1.tar.gz (192kB)
    100% |██████████| 194kB 1.8MB/s
Requirement already up-to-date: theano in /usr/local/lib/python2.7/dist-packages (from keras)
Requirement already up-to-date: pyyaml in /usr/local/lib/python2.7/dist-packages (from keras)
Requirement already up-to-date: six in /usr/lib/python2.7/dist-packages (from keras)
Requirement already up-to-date: numpy>=1.7.1 in /usr/local/lib/python2.7/dist-packages (from theano->keras)
Collecting scipy>=0.11 (from theano->keras)
  Downloading scipy-0.19.0-cp27-cp27mu-manylinux1_x86_64.whl (45.0MB)
    100% |██████████| 45.0MB 34kB/s
Building wheels for collected packages: keras
  Running setup.py bdist_wheel for keras ... done
  Stored in directory: /root/.cache/pip/wheels/fa/15/f9/57473734e407749529bf55e6b5038640dc7279d5718b2c368a
Successfully built keras
Installing collected packages: keras, scipy
  Found existing installation: Keras 1.1.2
    Uninstalling Keras-1.1.2:
      Successfully uninstalled Keras-1.1.2
  Found existing installation: scipy 0.17.0
```

- 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. **Here we are upgrading Keras!**

16

# *Singularity: Provides Flexibility for OS Environment*



The screenshot shows a terminal window titled "mahidhar — mahidhar@comet-30-08:~ — ssh comet.sdsc.edu — 83x35". The terminal output is as follows:

```
[mahidhar@comet-30-08 ~]$ module load singularity
[mahidhar@comet-30-08 ~]$ singularity shell /share/apps/gpu/singularity/sdsc_ubuntu
_tf1.0.0_keras.img
WARNING: Non existant bind point (directory) in container: '/scratch'
Singularity: Invoking an interactive shell within container...

[Singularity.sdsc_ubuntu_tf1.0.0_keras.img> $ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04 LTS
Release:        16.04
Codename:       xenial
[Singularity.sdsc_ubuntu_tf1.0.0_keras.img> $ python3
Python 3.5.1+ (default, Mar 30 2016, 22:46:26)
[GCC 5.3.1 20160330] on linux
Type "help", "copyright", "credits" or "license" for more information.
[>>> import tensorflow
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library lib
cublas.so.8.0 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library lib
cudnn.so.5 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library lib
cufft.so.8.0 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library lib
cuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library lib
curand.so.8.0 locally
>>> ]
```

- Copy image to Comet and we are running in an Ubuntu environment on a GPU node!

17

# Access Via Science Gateways (XSEDE)

- Community-developed set of tools, applications, and data that are integrated via a portal.
- **Enables researchers of particular communities to use HPC resources through portals without the complication of getting familiar with the hardware and software details. Allows them to focus on the scientific goals.**
- CIPRES gateway hosted by SDSC PIs enables large scale phylogenetic reconstructions using applications such as MrBayes, Raxml, and Garli. Enabled ~320 publications in 2013 and accounts for a significant fraction of the XSEDE users.
- **NSG portal hosted by SDSC PIs enables HPC jobs for neuroscientists.**

# **High performance virtualization (aka Virtual Clusters)**

# Motivation for Virtual Clusters

- OS and software requirements are diversifying. Growing number of user communities that can't work in traditional HPC software environment.
- Communities that have expertise and ability to utilize large clusters but need hardware.
- Institutions that have bursty or intermittent need for computational resources.

**Goal:** Provide near bare metal HPC performance and management experience for groups that can manage their own clusters.

# **Comet's virtual cluster (VC) technology bridges communities with software needs not typically well-supported by national cyberinfrastructure**

- VC's use a single root I/O virtualization, KVM, and Rocks cluster management software to deliver bare metal resources at near-native IB performance.
- Through our partnership with Indiana University/Future Grid, we have developed an on-ramping process that carefully assesses the readiness of these communities to maintain their own VC.
- VC's are provisioned on the fly through an integration with the Slurm resource manager.



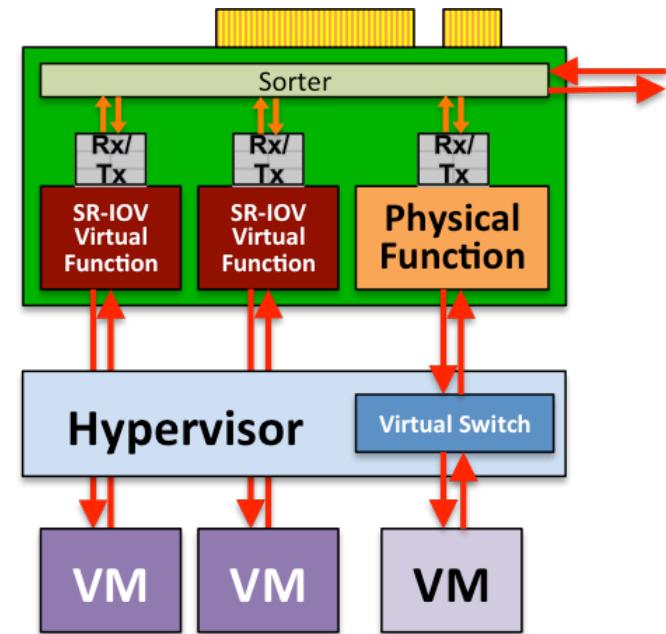
**Open Science Grid Using SDSC's Comet Supercomputer Virtual Clusters ↗**

Successful integration led by a team including UC San Diego Professor Frank Würthwein is seen as a “milestone” for the greater research community.

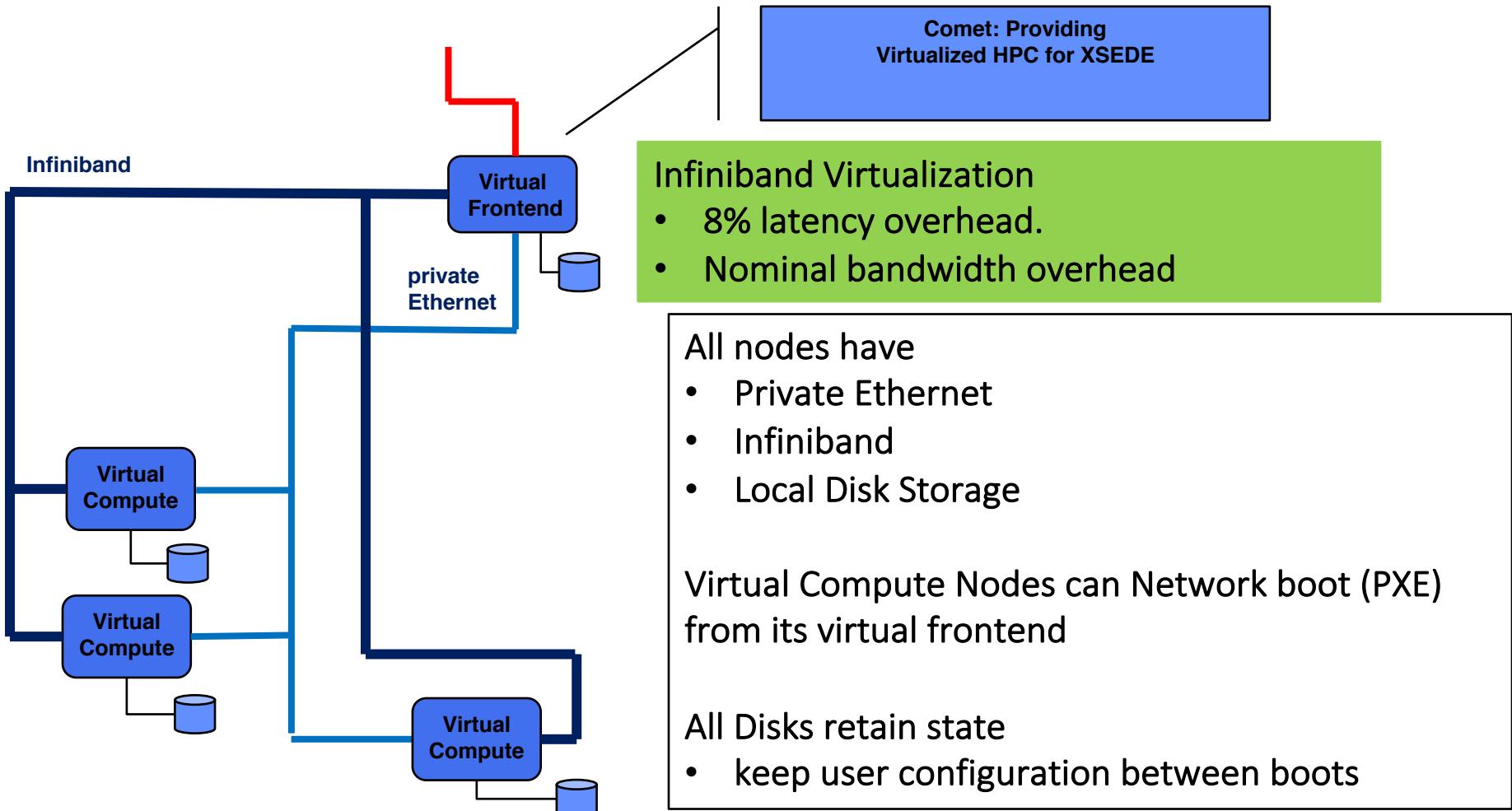
[Read More ↗](#)

# Key for Performance: *Single Root I/O Virtualization (SR-IOV)*

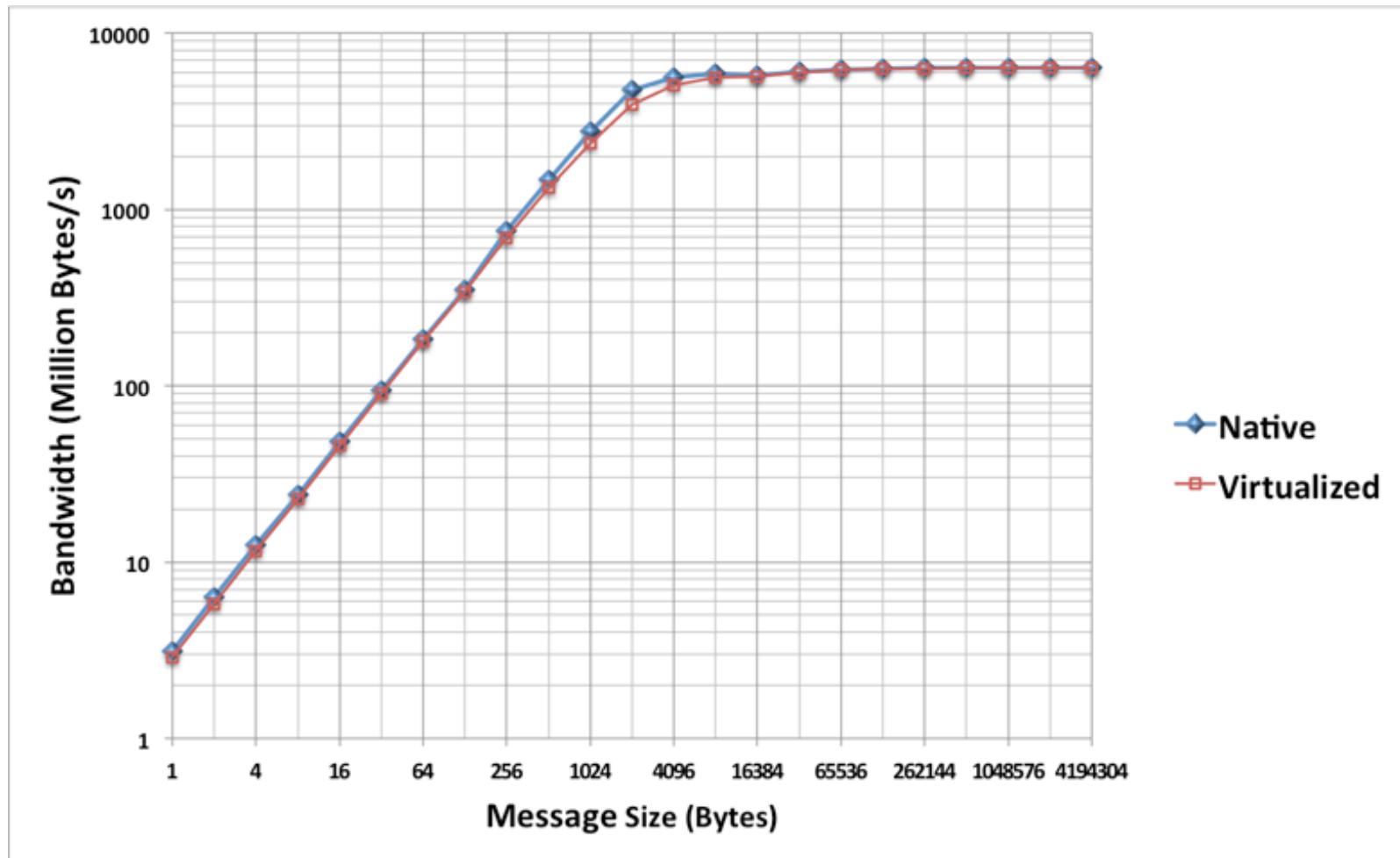
- **Problem:** Virtualization generally has resulted in significant I/O performance degradation (e.g., excessive DMA interrupts)
- **Solution:** SR-IOV and Mellanox ConnectX-3 InfiniBand host channel adapters
  - One physical function → multiple virtual functions, each light weight but with its own DMA streams, memory space, interrupts
  - Allows DMA to bypass hypervisor to VMs
- ***SRIOV enables virtual HPC cluster w/ near-native InfiniBand latency/bandwidth and minimal overhead***



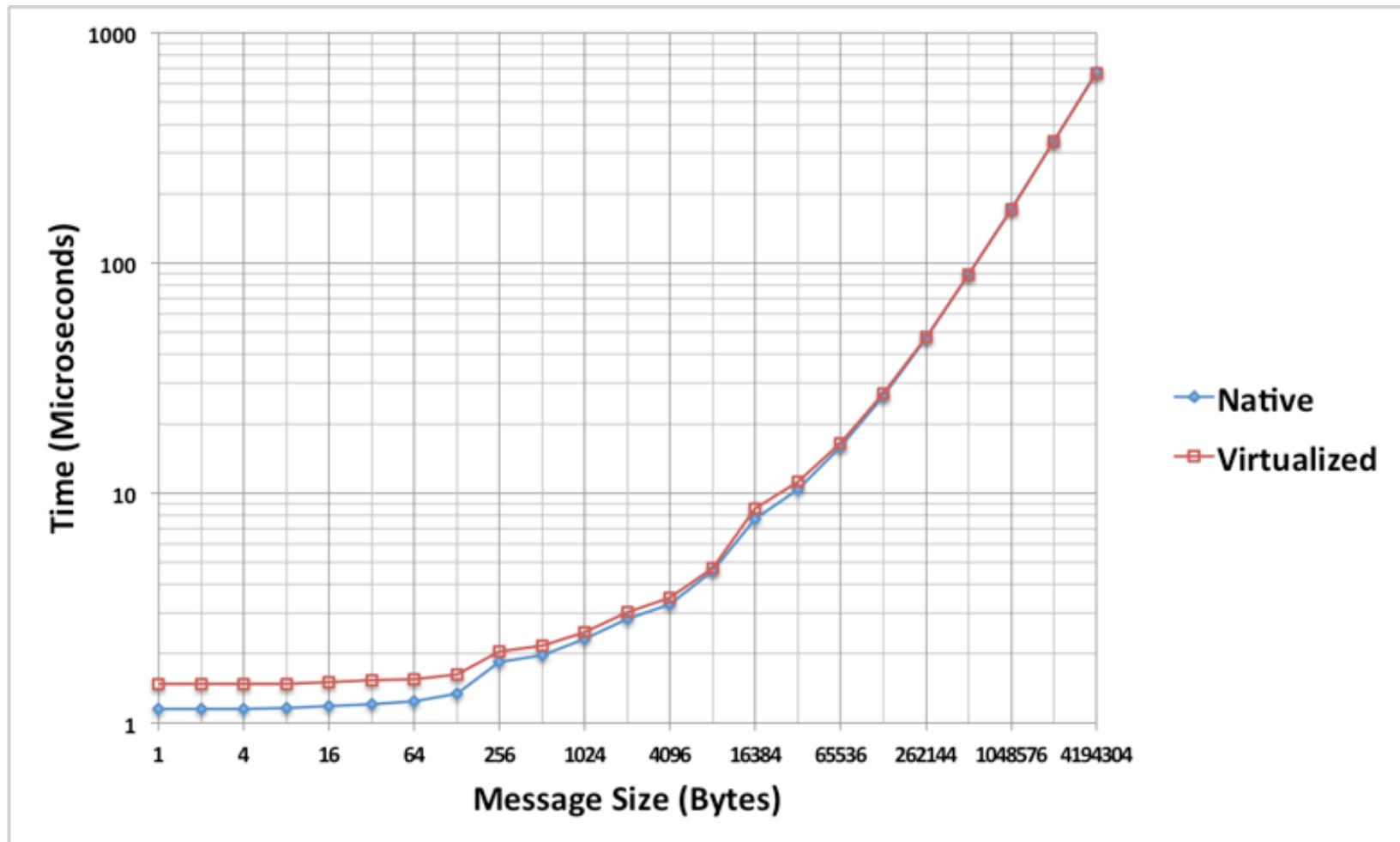
# High Performance Virtual Cluster Characteristics



# Comet: MPI bandwidth slowdown from SR-IOV is at most 1.21 for medium-sized messages & negligible for small & large ones



# Comet: MPI latency slowdown from SR-IOV is at most 1.32 for small messages & negligible for large ones



- **Next:** Hands On work with Comet

# 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
- **For NSF Resources – Users can login via the XSEDE user portal:**
  - <https://portal.xsede.org/>

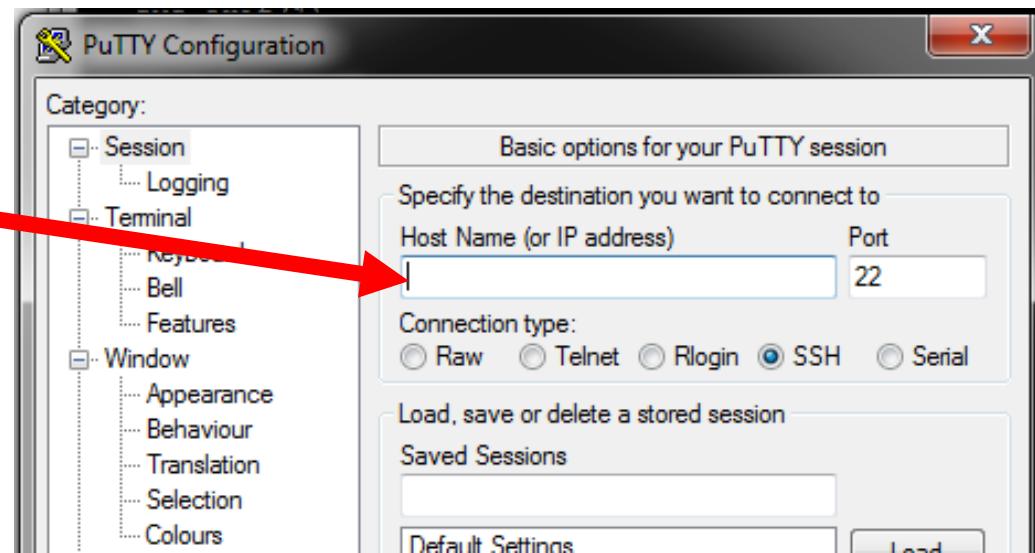
# Logging into Comet

Mac/Linux:

ssh etrainXY@comet.sdsc.edu

Windows (PuTTY):

comet.sdsc.edu



# Example Locations

- **Clone the git repository to get today's workshop scripts/files:**

```
git clone https://github.com/sdsc-scicomp/2017-04-04-comet-workshop-ucla.git workshop
```

- **Following subdirectories should be visible:**

Is \$HOME/workshop

```
hadoop mpi-openmp python README.md spark tensorflow
```

# Comet Compute Nodes

2-Socket (Total 24 cores) Intel Haswell Processors

Hands On Examples using:

- (1) MPI
- (2) OpenMP
- (3) HYBRID

# Comet – Compiling/Running Jobs

- **Copy and change to workshop directory:**

```
cd /home/$USER/workshop/mpi-openmp/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
```

# 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

# Compiling OpenMP Example

- Change to the examples directory:

```
cd /home/$USER/workshop/mpi-openmp/OPENMP
```

- Compile using –openmp flag:

```
ifort -o hello_openmp -openmp hello_openmp.f90
```

- Verify executable was created:

```
[mahidhar@comet-08-11 OPENMP]$ ls -lt hello_openmp
-rwxr-xr-x 1 mahidhar sdsc 750648 Mar 25 15:00 hello_openmp
```

# OpenMP job script

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

#SET the number of openmp threads
export OMP_NUM_THREADS=24

#Run the job using mpirun_rsh
./hello_openmp
```

# Output from OpenMP Job

```
$ more hello_openmp.out
```

HELLO FROM THREAD NUMBER =	7
HELLO FROM THREAD NUMBER =	6
HELLO FROM THREAD NUMBER =	9
HELLO FROM THREAD NUMBER =	8
HELLO FROM THREAD NUMBER =	5
HELLO FROM THREAD NUMBER =	4
HELLO FROM THREAD NUMBER =	0
HELLO FROM THREAD NUMBER =	12
HELLO FROM THREAD NUMBER =	14
HELLO FROM THREAD NUMBER =	3
HELLO FROM THREAD NUMBER =	13
HELLO FROM THREAD NUMBER =	10
HELLO FROM THREAD NUMBER =	11
HELLO FROM THREAD NUMBER =	2
HELLO FROM THREAD NUMBER =	1
HELLO FROM THREAD NUMBER =	15

# Running Hybrid (MPI + OpenMP) Jobs

- Several HPC codes use a hybrid MPI, OpenMP approach.
- “**ibrun**” wrapper developed to handle such hybrid use cases. Automatically senses the MPI build (mvapich2, openmpi) and binds tasks correctly.
- “**ibrun –help**” gives detailed usage info.
- **hello\_hybrid.c** is a sample code, and **hello\_hybrid.cmd** shows “ibrun” usage.

# hello\_hybrid.cmd

```
#!/bin/bash
#SBATCH --job-name="hellohybrid"
#SBATCH --output="hellohybrid.%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.

```
# We use 8 MPI tasks and 6 OpenMP threads per MPI task
export OMP_NUM_THREADS=6
ibrun --npernode 4 ./hello_hybrid
```

# Hybrid Code Output

```
[etrain61@comet-In3 HYBRID]$ more hellohybrid.8557716.comet-14-01.out
```

Hello from thread 0 out of 6 from process 2 out of 8 on comet-14-01.local

Hello from thread 3 out of 6 from process 2 out of 8 on comet-14-01.local

Hello from thread 4 out of 6 from process 2 out of 8 on comet-14-01.local

Hello from thread 5 out of 6 from process 2 out of 8 on comet-14-01.local

Hello from thread 0 out of 6 from process 3 out of 8 on comet-14-01.local

Hello from thread 2 out of 6 from process 2 out of 8 on comet-14-01.local

Hello from thread 1 out of 6 from process 3 out of 8 on comet-14-01.local

Hello from thread 2 out of 6 from process 3 out of 8 on comet-14-01.local

...

...

Hello from thread 4 out of 6 from process 7 out of 8 on comet-14-02.local

Hello from thread 2 out of 6 from process 7 out of 8 on comet-14-02.local

Hello from thread 3 out of 6 from process 7 out of 8 on comet-14-02.local

Hello from thread 5 out of 6 from process 7 out of 8 on comet-14-02.local

Hello from thread 1 out of 6 from process 6 out of 8 on comet-14-02.local

# **Comet GPU Nodes**

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

**Hands On Examples using Singularity to  
enable Tensorflow**

# 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_release -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/workshop/tensorflow
```

- **Submit the job:**

```
sbatch --res=UCLARes 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%

# Summary

- Comet's hardware options, both for compute and for storage, make it a very *flexible platform for research*.
- Baseline configuration supports a diverse HPC and data analytics workload with over *100 applications*. Will continue to add to existing software stack.
- *Singularity* provides a simple solution for users needing a *custom OS and software stack*.
- *Scheduling flexibility* to allow for quick turn around, shared jobs, long running jobs, and support of Science Gateways.

**Thanks!**

**Questions: Email [mahidhar@sdsc.edu](mailto:mahidhar@sdsc.edu)**