Introduction to SDSC Systems

Mahidhar Tatineni
UCLA Nov 8, 2019

Agenda

- **9:00 AM – 9:10 AM: Welcome**
- **9:10 AM – 10:10 AM: Introduction to SDSC Resources – Current and Future**
  - Comet, hardware overview
  - Software, stack, queues, etc
  - Overview of upcoming Expanse system
- **10:10 AM – 10:20 AM: Short break**
- **10:20 AM -12:00 PM: Python for High Performance Computing (HPC)**
  - Use of Jupyter notebooks on Comet
  - Scaling using IPython parallel
  - Using Numba to run pure python codes on GPUs
  - Distributed parallel computing using Dask
- **12:00 PM – 1:00 PM: Lunch (provided)**
- **1:00 PM – 2:30 PM: Introduction to Machine Learning and Deep Learning on Comet**
  - Overview of machine learning / deep learning tools available on Comet.
  - Examples using R, Python, and Keras with TensorFlow.
- **2:30 PM – 2:40 PM: Short break**
- **2:40 PM – 4:00 PM: Containers for Scientific and High Performance Computing (HPC)**
  - Introduction to Containers – advantages and limitations
  - Overview of containers on HPC systems
  - Singularity Essentials
  - Hands-on examples showing usage of containers – CPU, GPU, and MPI cases
  - Hands-on example showing the import of Docker container
- **Info on building images**
- **4:00 PM: Adjourn**

https://github.com/sdsc-scicomp/2019-11-08-comet-workshop-ucla

NSF Supercomputers at SDSC

- **Future:**
  - **EXPANSE** – *Coming October 2020*
- **Current:**
  - **Comet** - In production since 2015
  - Today's tutorials will be using Comet


Thank you to our collaborators, partners, users, and the SDSC team!
NSF
DELL
intel
AEON COMPUTING
NVIDIA
XSEDE
Extreme Science and Engineering Discovery Environment
Mellanox TECHNOLOGIES
Ilkay Altintas
Amit Chourasia
Trevor Cooper
Jerry Greenberg
Eva Hocks
Tom Hutton
Christopher Irving
Marty Kandes
Amit Majumdar
Dima Mishin
Sonia Nayak
Mike Norman
Wayne Pfeiffer
Scott Sakai
Fernando Silva
Bob Sinkovits
Subha Sivagnanam
Michele Strong
Shawn Strande
Mahidhar Tatineni
Mary Thomas
Nicole Wolter
Frank Wuerthwein
EXPANSE
COMPUTING WITHOUT BOUNDARIES
SAN DIEGO SUPERCOMPUTER CENTER
IN PRODUCTION OCTOBER 2020

Disclaimer:

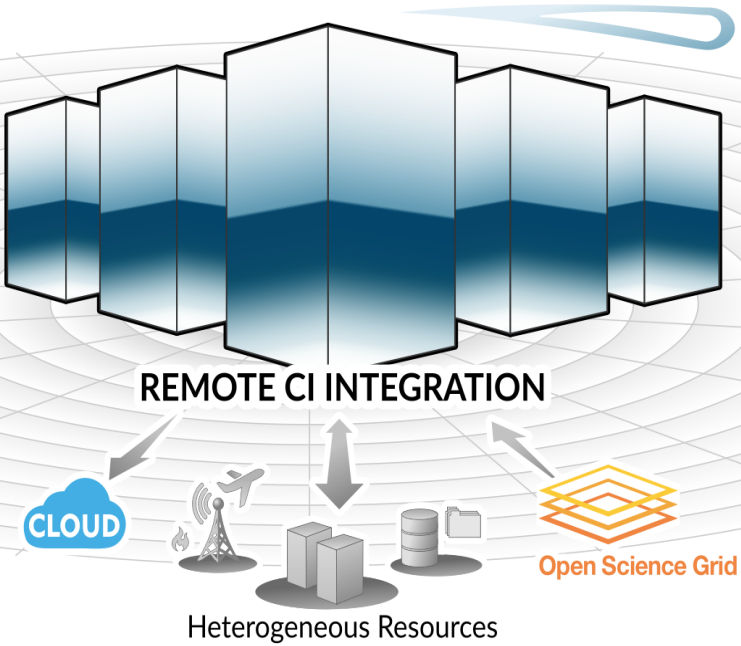
The Expanse system has not yet been procured and technical details are subject to change.

EXPANSE

COMPUTING WITHOUT BOUNDARIES
5 PETAFLOP/S HPC and DATA RESOURCE

HPC RESOURCE
13 Scalable Compute Units
728 Standard Compute Nodes
52 GPU Nodes: 208 GPUs
4 Large Memory Nodes

DATA CENTRIC ARCHITECTURE
12PB Perf. Storage: 140GB/s, 200k IOPS
Fast I/O Node-Local NVMe Storage
7PB Ceph Object Storage
High-Performance R&E Networking



LONG-TAIL SCIENCE
Multi-Messenger Astronomy
Genomics
Earth Science
Social Science

INNOVATIVE OPERATIONS
Composable Systems
High-Throughput Computing
Science Gateways
Interactive Computing
Containerized Computing
Cloud Bursting

Computing Without Boundaries: Cyberinfrastructure for the Long Tail of Science

- **NSF Solicitation 19-534: Advanced Computing Systems & Services: Adapting to the Rapid Evolution of Science and Engineering Research**
- **Category 1: Capacity System, NSF Award # 1928224**
- **NSF Program Officer: Robert Chadduck**
- **System name: *Expanse***
- **Awardee: San Diego Supercomputer Center at UC San Diego**
- **PIs: Mike Norman (PI), Ilkay Altintas, Amit Majumdar, Mahidhar Tatineni, Shawn Strande**
- **$10M Acquisition**
- **Operations and Maintenance request of 25% of acquisition.**
- **5-year operations, commencing 10/1/2020**
- **Potential 5-year follow-on request pending successful 3rd-year review**
- **Expected delivery and integration: Q2-Q3 2020**
- **Primary Vendors: Dell (HPC system); Aeon Computing (storage)**
- **Compute and interconnect: Intel, NVIDIA, Mellanox**

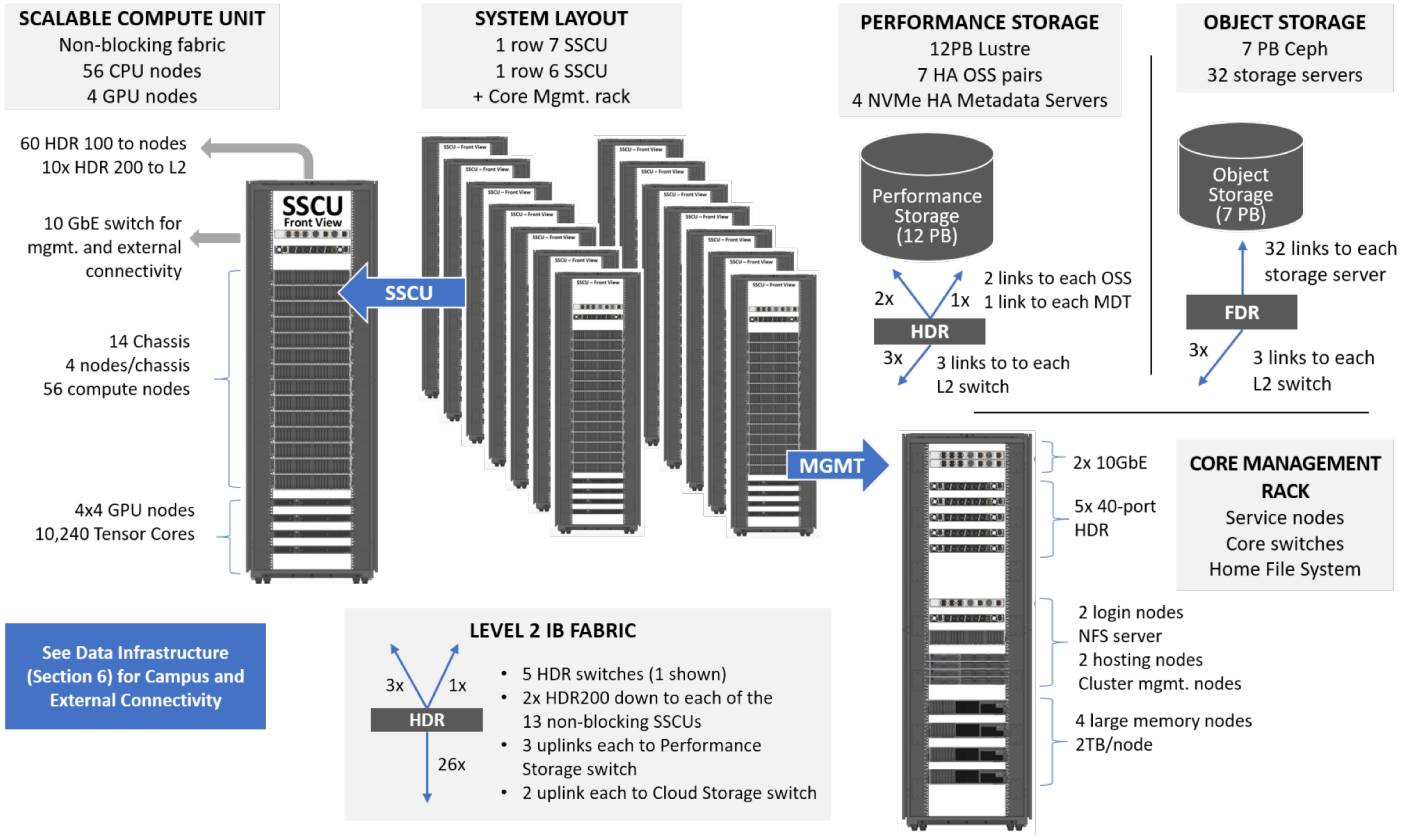
Expanse is responsive to NSF 19-534

"Resources proposed in this category…**deployments of production computational resources**…provide **maximum capacity and throughput** to support the broad range of computation and data analytics needs in S&E research.

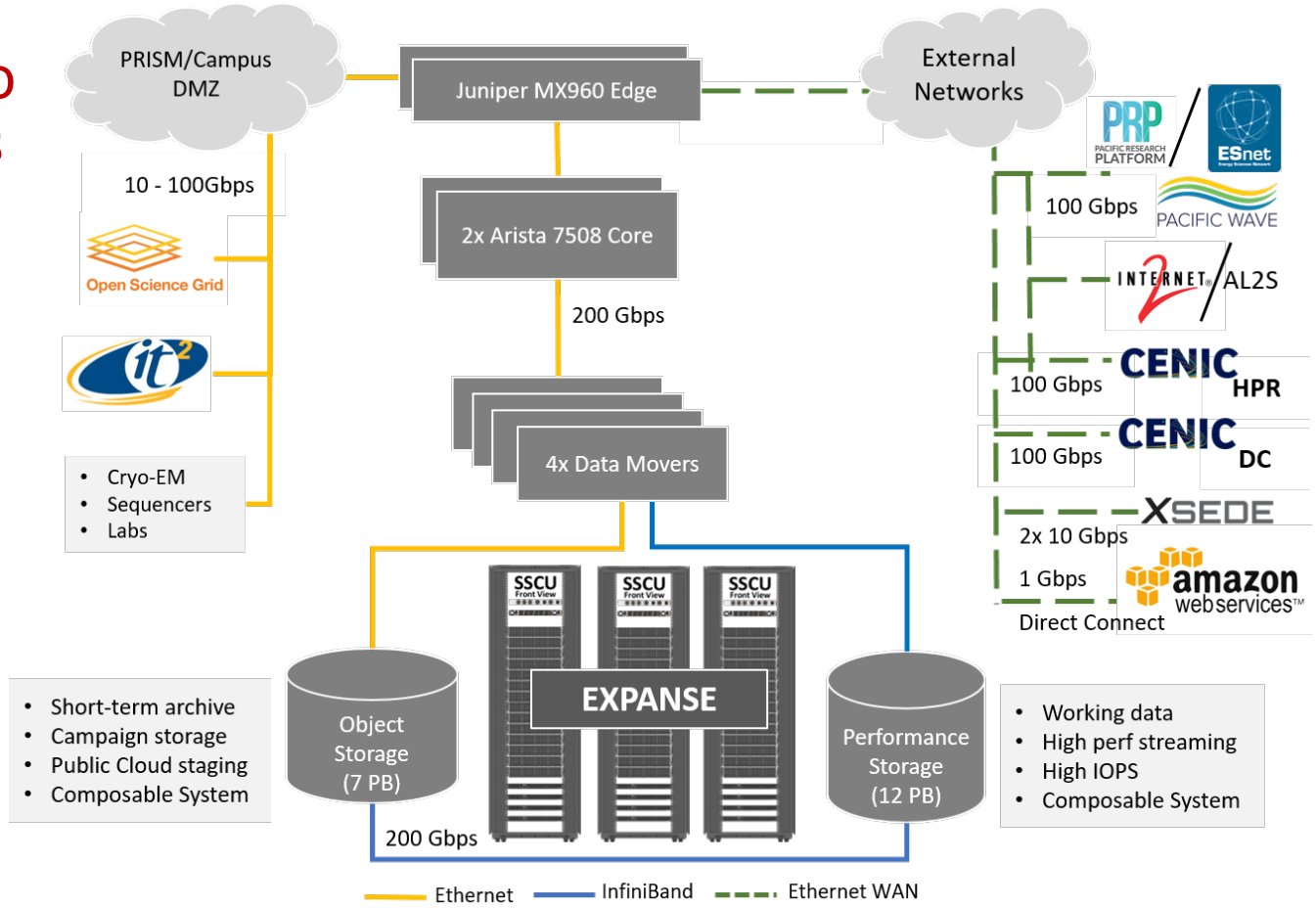
This category particularly targets…**small- to mid-scale jobs (from one to a few thousand cores per job)** across broad areas of S&E, including **support for "long-tail science" applications**, as well as new classes of applications, such as **artificial intelligence/machine learning/deep learning** applications…encouraged to explore novel models that leverage federated and/or **distributed resources, regional and/or campus supported resources, and/or commercial cloud services.**

NSF strongly urges the community to…**think broadly about the nature and composition of the CI ecosystem**. …federated approaches with opportunities for leveraging the increasing availability and **capabilities at the network edge** (including campuses); and **composable services provisioning virtualized on-premise computing infrastructure and commercial cloud services."**

Expanse is a heterogeneous architecture designed for high performance, reliability, flexibility, and productivity



Connectivity to R&E Networks Facilitates Compute and Data Workflows



Expanse is a heterogeneous architecture designed for high performance, reliability, flexibility, and productivity

SDSC Scalable Compute Units (SSCU) (13)
- 56 Standard compute nodes
- 4 GPU nodes

Compute Nodes
- Two-socket, Intel Next Gen
- 256 GB DDR4
- 1 TB NVMe

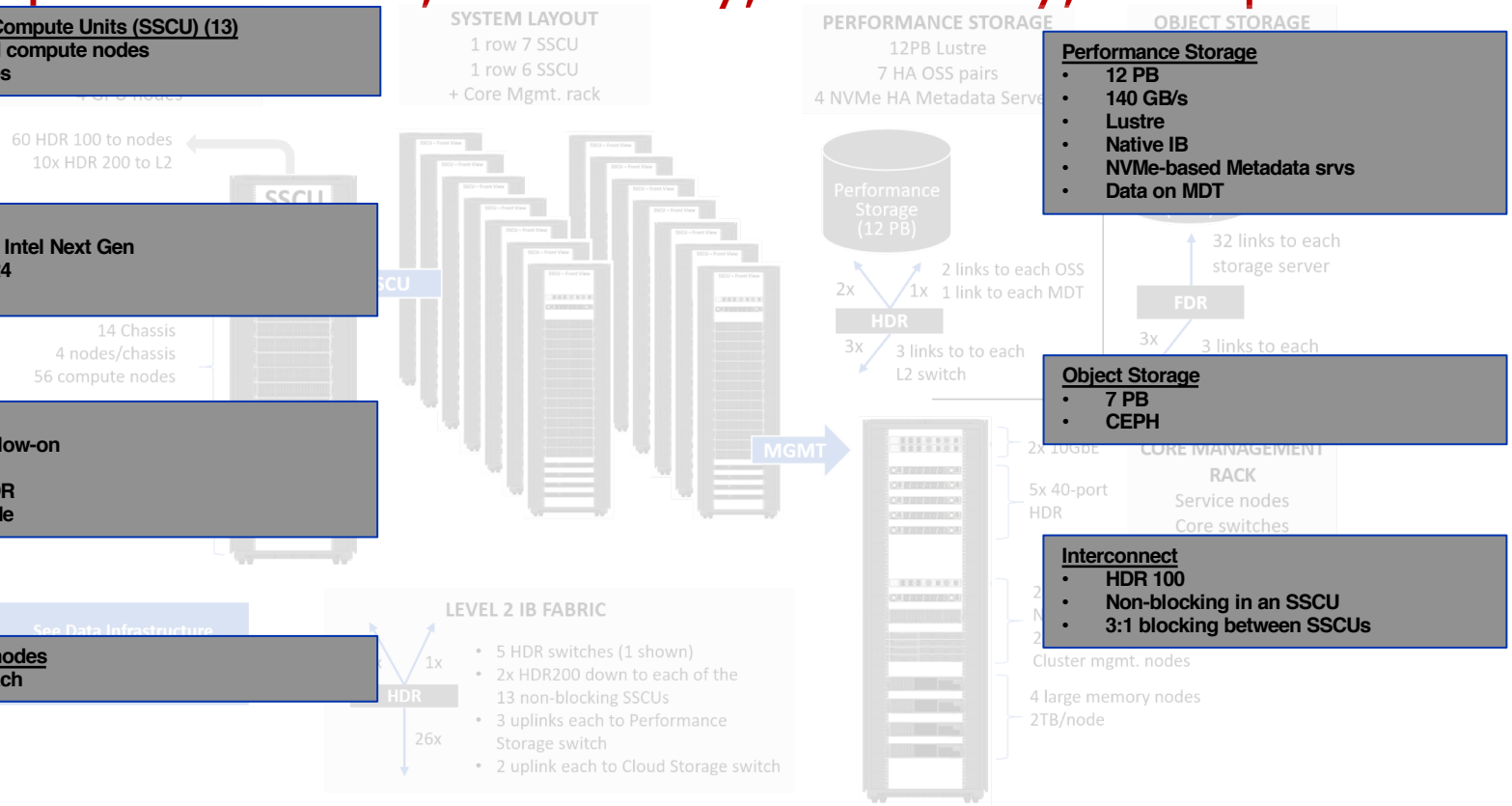
GPU Nodes
- 4x V100/follow-on
- SMX2
- 32 GB GDDR
- 1.6 TB NVMe

Large Memory nodes
- 4 @ 2TB each

Performance Storage
- 12 PB
- 140 GB/s
- Lustre
- Native IB
- NVMe-based Metadata srvs
- Data on MDT

Object Storage
- 7 PB
- CEPH

Interconnect
- HDR 100
- Non-blocking in an SSCU
- 3:1 blocking between SSCUs



Expanse architecture and operational policies will support the long tail of science

- A Scalable Unit supports more than 3,000 cores in a full bisection, HDR 100 network
- Rapid access “kick the tire” accounts given within 1 working day of request
- Software support and allocation policies favorable to science gateways
- Scheduler optimized for high throughput and shared-node jobs
- Public cloud integration for projects that share data, need access to novel technologies, and integrate cloud resources into workflows
- Integration with the Open Science Grid for high-throughput computing
- Support for containerized computing
- Composable systems for complex, distributed workflows
- Interactive computing capability for rapid development, data analysis and visualization

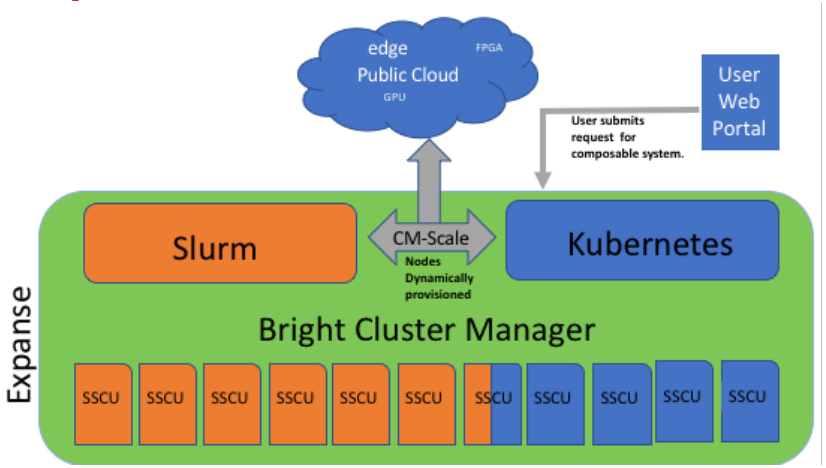
Integration with public cloud supports projects that share data, need access to novel technologies, and integrate cloud resources into workflows

- Slurm + in-house developed software + Terraform (Hashicorp)
- Early work funded internally and via NSF E-CAS/Internet2 project for CIPRES (Exploring Cloud for the Acceleration of Science, Award #1904444).
- Approach is cloud-agnostic and will support the major cloud providers
- Users submit directly via the Slurm, or as part of a composed system
- Options for data movement: data in the cloud; remote mounting of file systems; cached filesystems (e.g., StashCache), and data transfer during the job.
- New CC* Award (1925558), Triton Stratus, will integrate campus cluster with commercial cloud

*** Funding for user cloud resources is not part of the Expanse award. Researcher must have access to these via other NSF awards and funding.**

Composable Systems will support complex, distributed, workflows – making Expanse part of a larger CI ecosystem

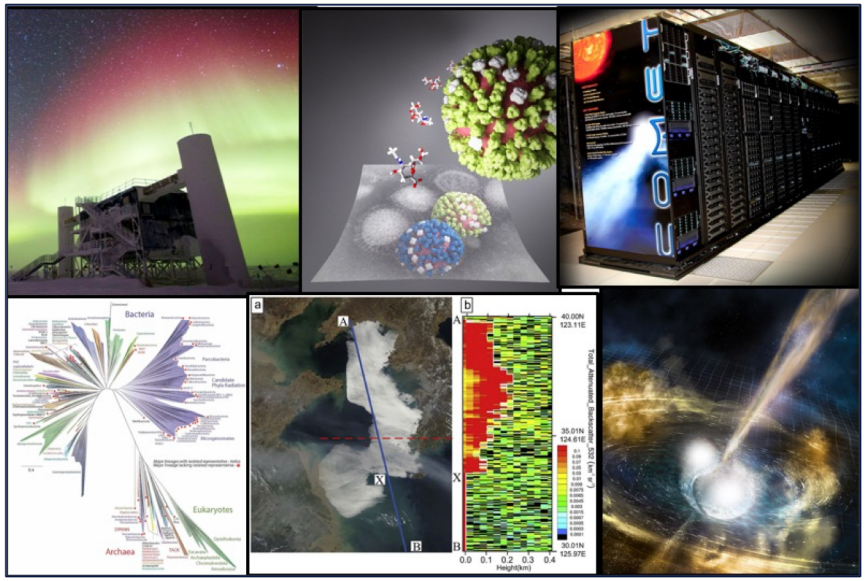
- **Bright Cluster Manager + Kubernetes**
- **Core components developed via NSF-funded CHASE-CI (NSF Award # 1730158), and the Pacific Research Platform (NSF Award # 1541349)**
- **Requests for a composable system will be part of an XRAC request**
- **Advanced User Support resources available to assist with projects - this is part of our operations funding!**



User support, training, outreach, and education will help users make the most of Expanse's traditional and innovative features

- **Fully integrated as an XSEDE Level 1 Resource bringing substantial support and expertise to the community**
- **A new program, HPC@MSI targeted at Minority Serving Institutions will make use of Directors Discretionary time that can be awarded via a rapid review process**
- **A novel machine learning approach will mine the Expanse knowledge base to improve the relevancy and quality of self-service support**
- **Advanced Support from SDSC staff for projects that need deeper engagements. This will complement what is available via XSEDE ECSS**

Like Comet, which concludes operations in March 2021, Expanse will advance science and engineering discovery



Clockwise from upper left: IceCube Neutrino Detection; Battling Influenza; Comet Surpasses 40,000 Users; Detecting Gravitational Waves; Predicting Sea Fog; Defining a New Tree of Life

In just over 4 years:

- **40,000+ Unique Users**
- **1,200+ Publications**
- **~2,000 Research, education and startup allocations**
- **400+ Institutions**
- **Scientific discoveries and breakthroughs**

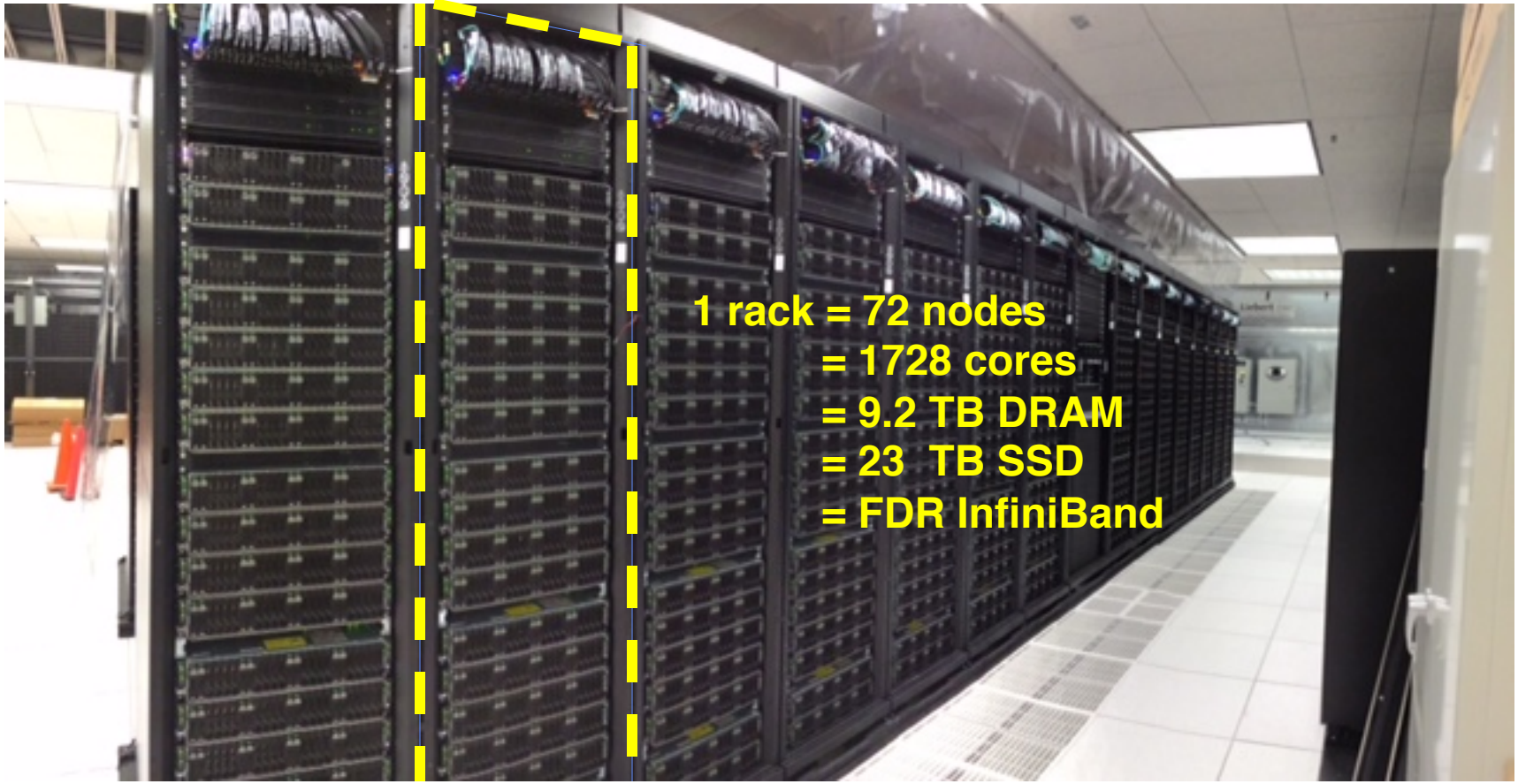
Comet
“HPC for the long tail of science”

iPhone panorama photograph of 1 of 2 server rows

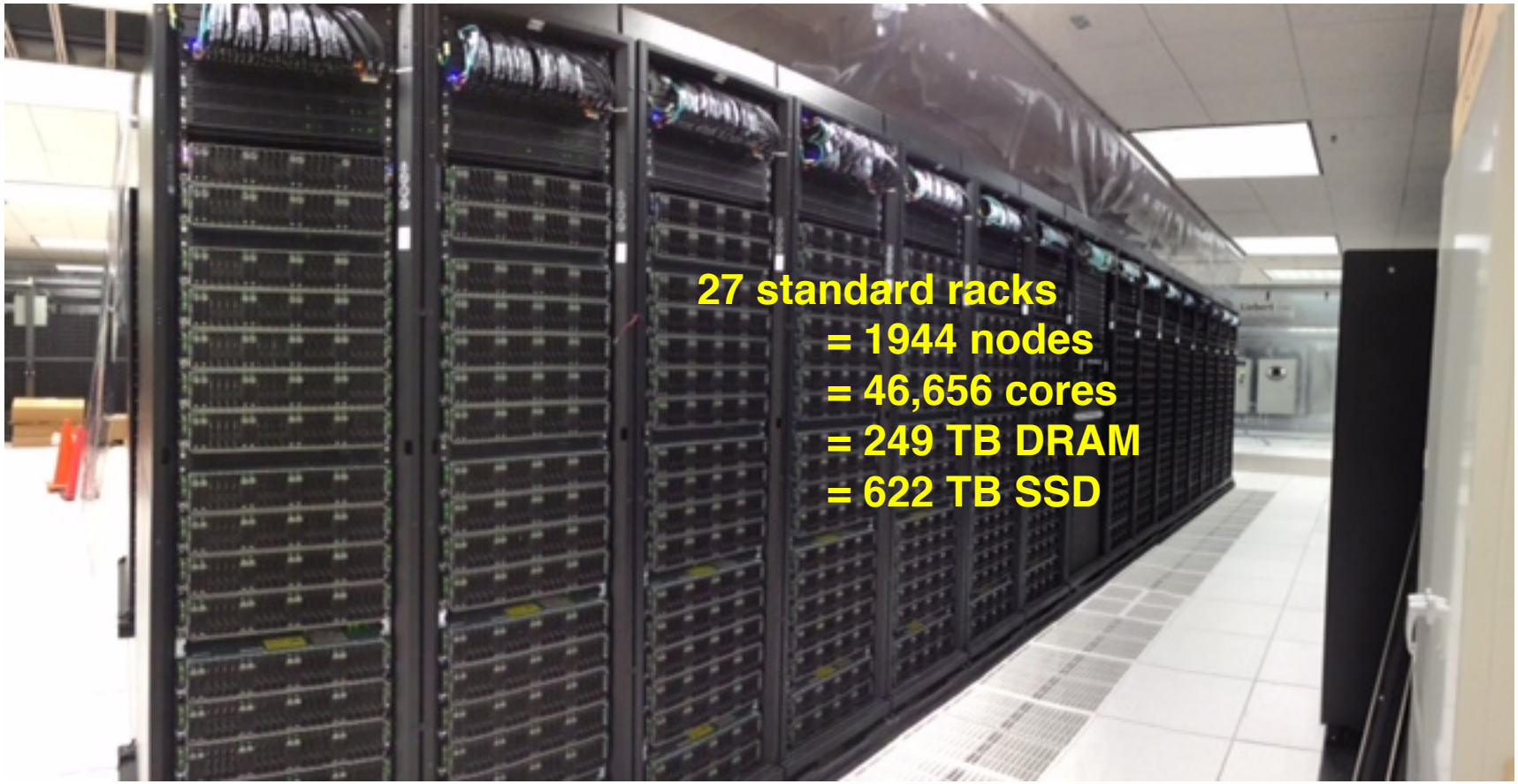
Comet: System Characteristics

- **Total peak flops ~2.7 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**

~67 TF supercomputer in a rack

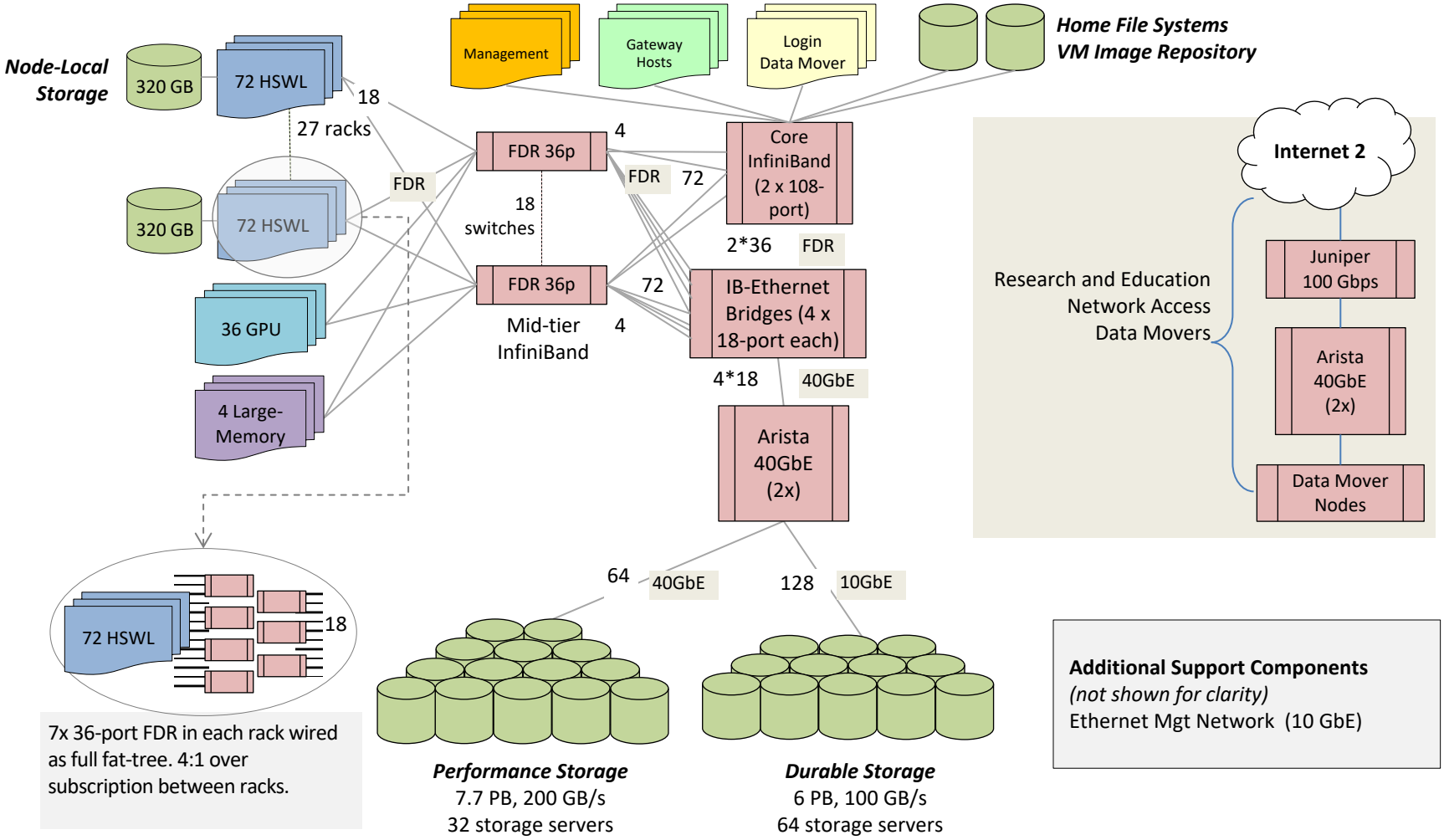


And 27 single-rack supercomputers



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

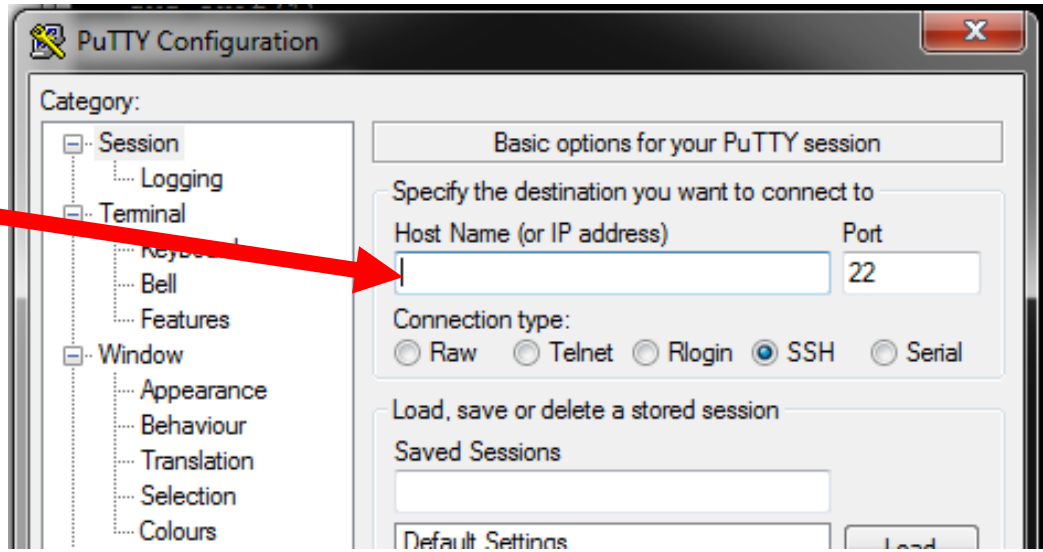
Logging into Comet

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

(Changing soon with upgrade to CentOS 7!)

- **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

(Changing soon with upgrade to CentOS 7!)

- **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.*

Example Files for Class

- **Today's files:**

  /share/apps/examples/workshop

- **TUTORIAL directory:**

  cd $HOME/TUTORIAL

  (If it doesn't exist:
  cp -r /share/apps/examples/workshop/TUTORIAL $HOME)

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 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

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=UCLARES 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
```

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!