Introduction to SDSC Systems, Launching and Managing Jobs

Mahidhar Tatineni (mahidhar@sdsc.edu)

SDSC Summer Institute

August 01, 2016

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 machines:
    - comet.sdsc.edu
    - gordon.sdsc.edu

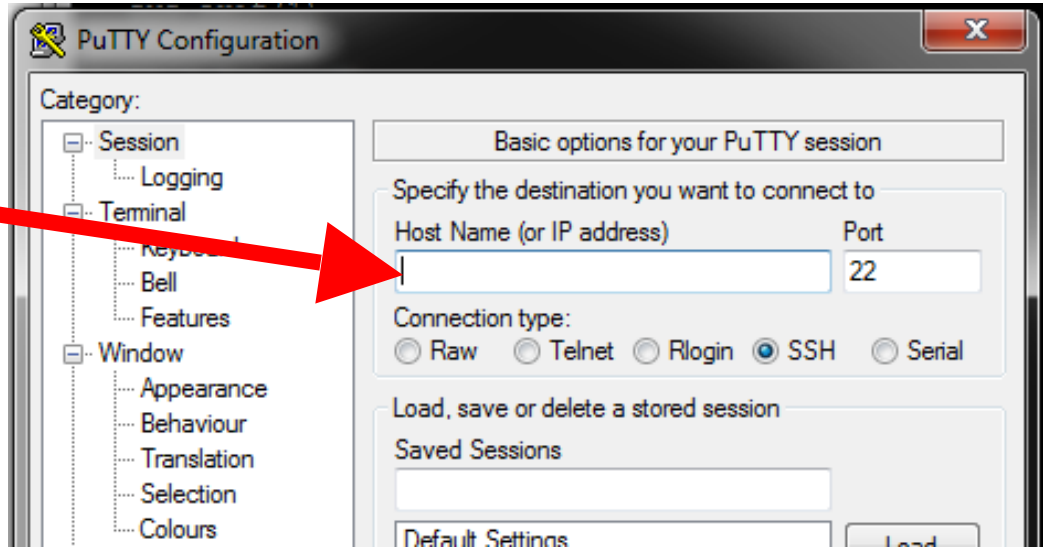
Logging into Comet

Mac/Linux:

```
ssh etrainXY@comet.sdsc.edu
```

Windows (PuTTY):

comet.sdsc.edu



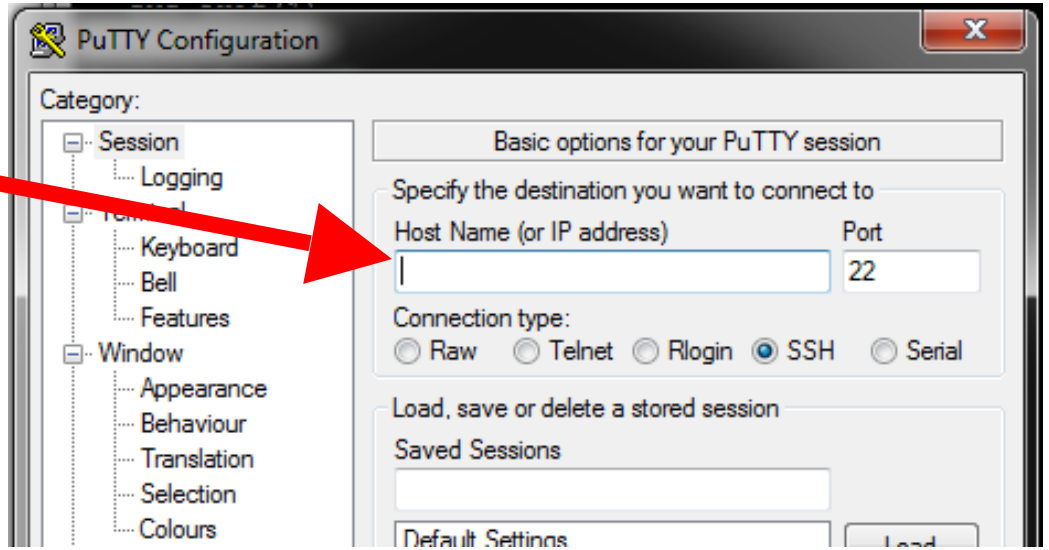
Logging into Gordon

Mac/Linux:

```
ssh etrainXY@gordon.sdsc.edu
```

Windows (PuTTY):

```
gordon.sdsc.edu
```



Training Accounts

- **Make sure we have the training intro exercises. Login to Comet and copy the files to your home directory:**

```
cp  -r  /share/apps/examples/SI2016  $HOME/
```

```
ls $HOME/SI2016/INTRO
```

```
COMET   GORDON
```

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. The CIPRES gateway has supported over 17700 users and enabled ~2300 publications since it began operations.**

- **NSG portal hosted by SDSC PIs enables HPC jobs for neuroscientists.**

Data Transfer (scp, globus-url-copy)

- **scp is o.k. to use for simple file transfers and small file sizes (<1GB). Example:**

  $ scp w.txt train40@gordon.sdsc.edu:/home/train40/w.txt
  100% 15KB 14.6KB/s 00:00

- **globus-url-copy for large scale data transfers between XD resources (and local machines w/ a globus client).**
  - Uses your XSEDE-wide username and password
  - Retrieves your certificate proxies from the central server
  - Highest performance between XSEDE sites, uses striping across multiple servers and multiple threads on each server.

Data Transfer – globus-url-copy

- **Step 1: Retrieve certificate proxies:**

  $ module load globus

  $ myproxy-logon –l xsedeusername

  Enter MyProxy pass phrase:

  A credential has been received for user xsedeusername in /tmp/x509up_u555555.

- **Step 2: Initiate globus-url-copy:**

```
$ globus-url-copy -vb -stripe -tcp-bs 16m -p 2 gsiftp://gridftp.stampede.tacc.xsede.org:2811//work/00342/
mahidhar/fromtres.tar.gz gsiftp://oasis-dm.sdsc.edu:2811//oasis/scratch-comet/mahidhar/temp_project/
fromcomet.tar.gz
Source: gsiftp://gridftp.stampede.tacc.xsede.org:2811//work/00342/mahidhar/
Dest:   gsiftp://oasis-dm.sdsc.edu:2811//oasis/scratch-comet/mahidhar/temp_project/
  fromtres.tar.gz  ->  fromcomet.tar.gz

    124585864 bytes       198.02 MB/sec avg       198.02 MB/sec inst
```

Data Transfer – Globus

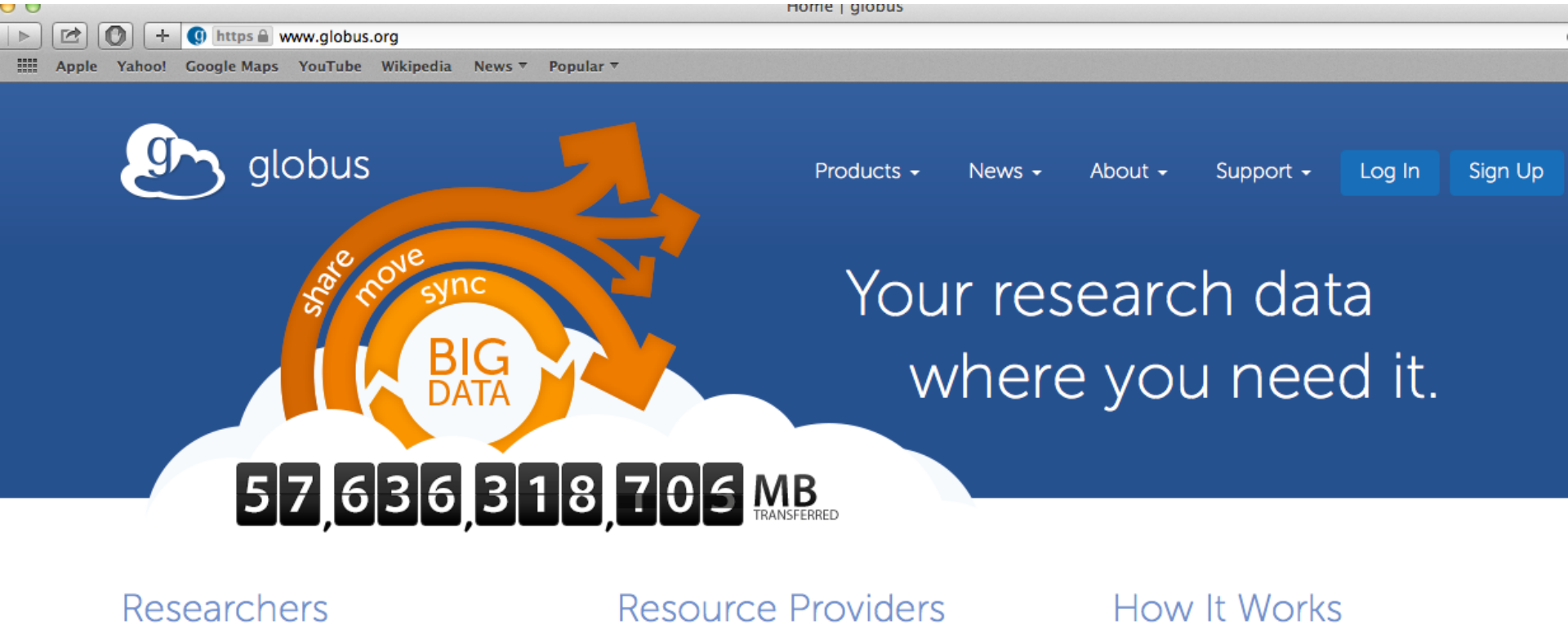
- Works from Windows/Linux/Mac via globus online website:
  - https://www.globus.org

- Comet, Gordon endpoints already exist (xsede#comet and xsede#gordon). Authentication can be done using XSEDE-wide username and password for the NSF resources.

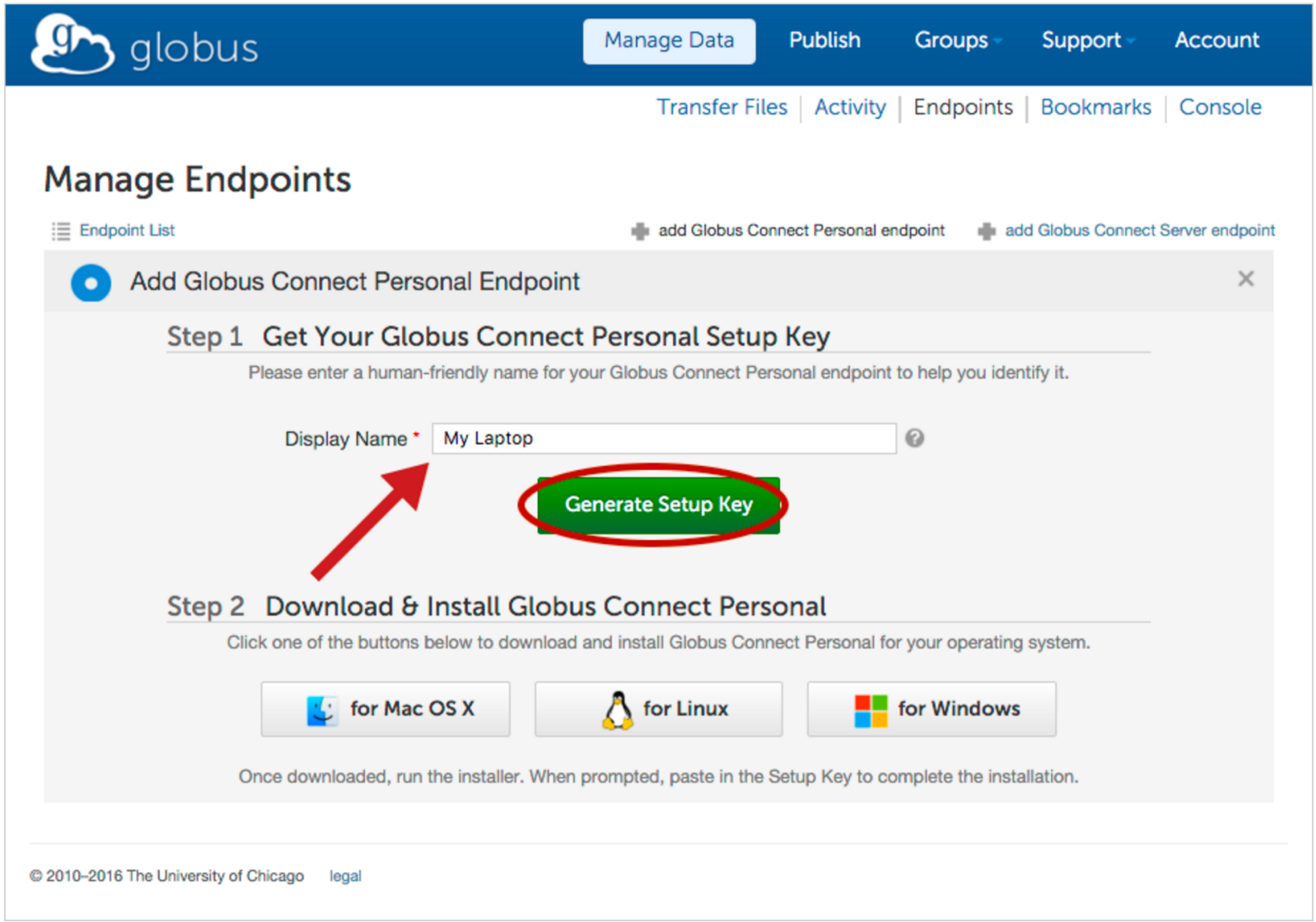
- Globus Connect application (available for Windows/Linux/Mac can turn your laptop/desktop into an endpoint.

Data Transfer – Globus Online

- **Step 1: Create a globus account (at www.globus.org)**

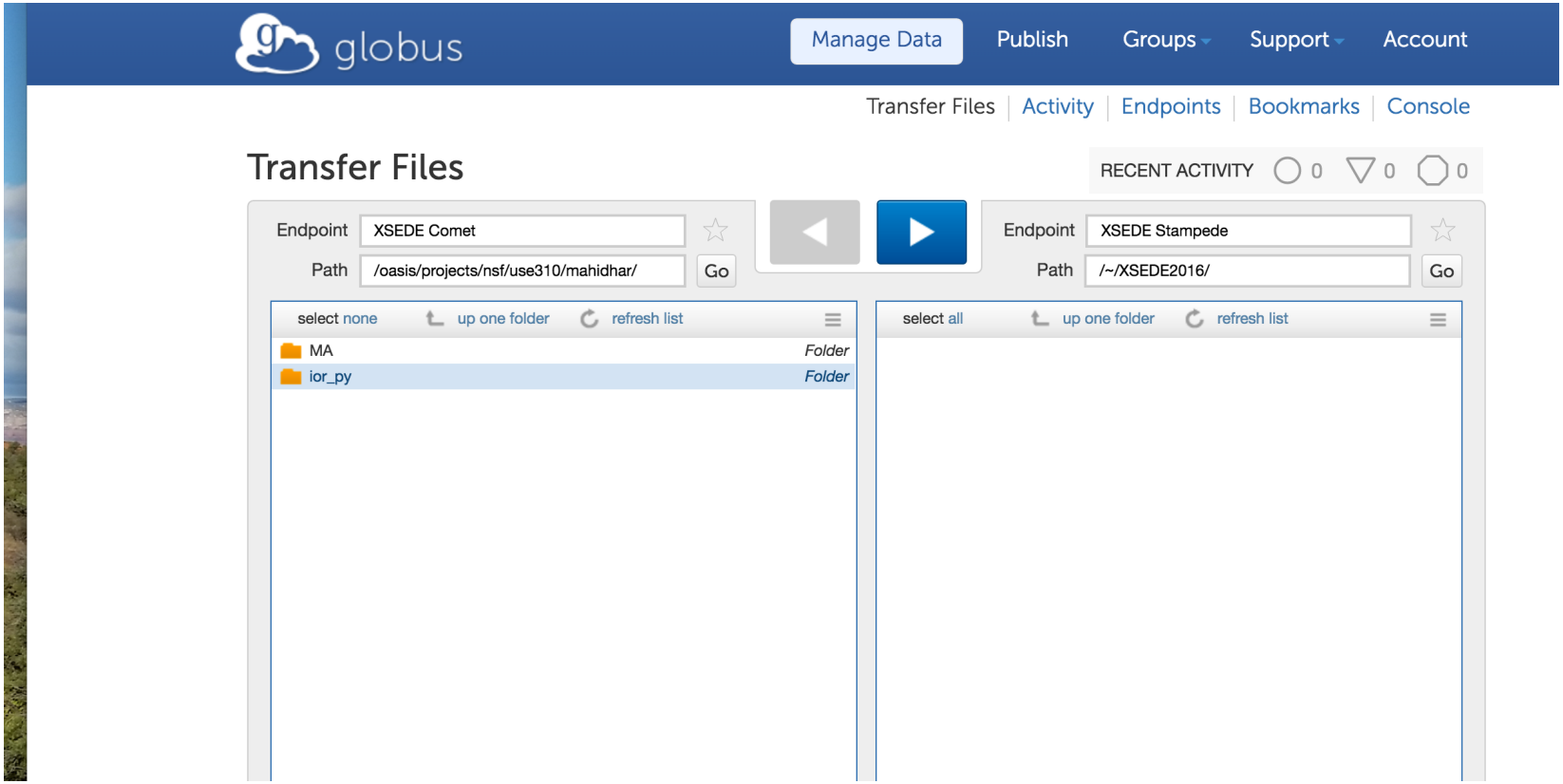


Data Transfer – Globus



Data Transfer – Globus

- **Step 3: Pick Endpoints and Initiate Transfers!**



HPC Resources at SDSC

NSF : Comet, Gordon
Parallel Filesystems: Data Oasis
UCSD IDI Resource: TSCC

NSF

UC San Diego

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

DELL

intel

INDIANA UNIVERSITY

AEON COMPUTING

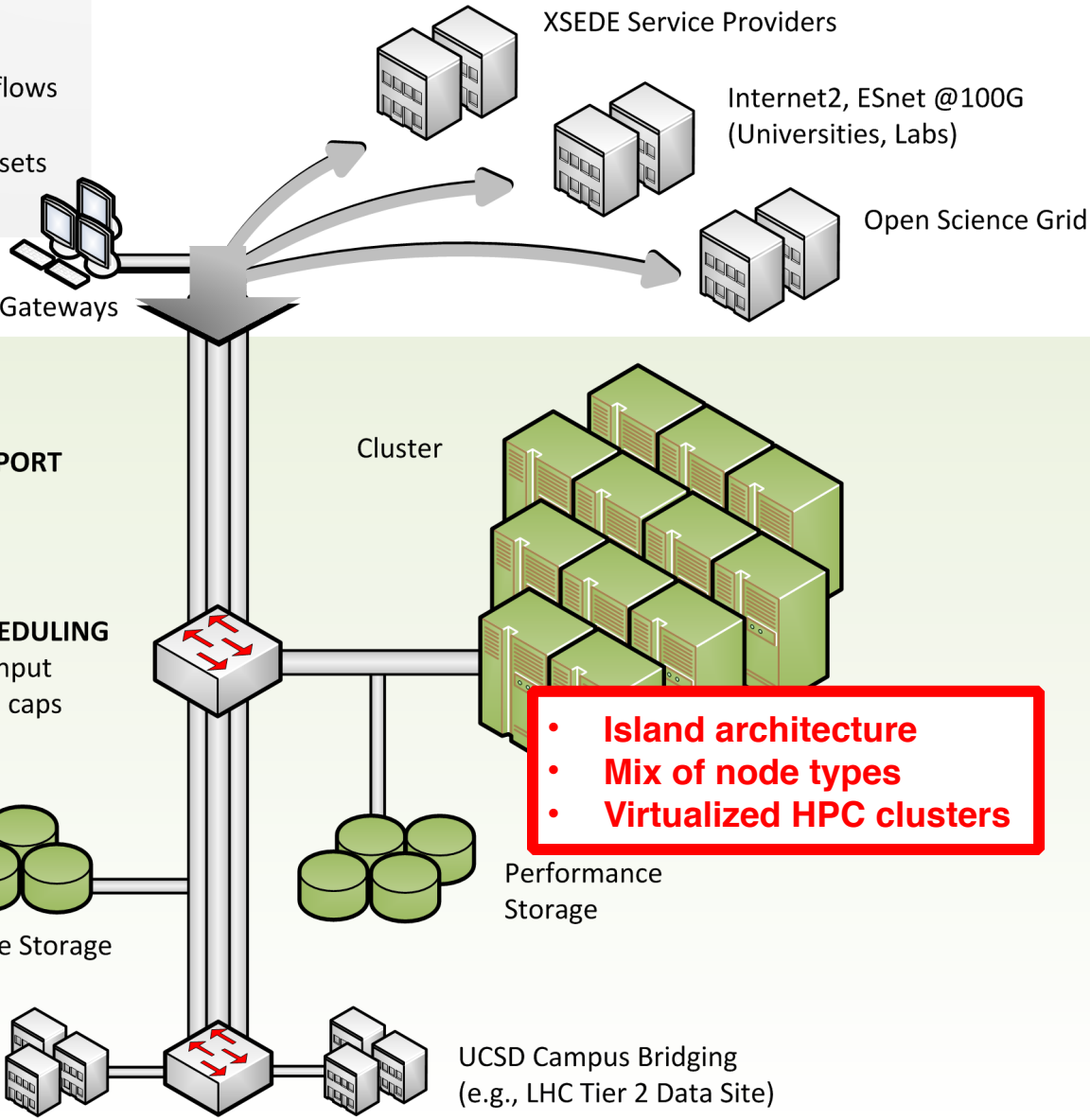
Mellanox
TECHNOLOGIES

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

Comet Serves 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 COMPUTE SYSTEM

Cluster architecture
Fast standard nodes
Large-memory nodes
GPU-accelerated nodes
FDR InfiniBand

Storage architecture
Performance Storage
Durable Storage

Software
Science Gateways
Rich base of installed apps
Virtualization

USER & SYSTEM SUPPORT
New user orientation
XSEDE collaborations
FutureGrid

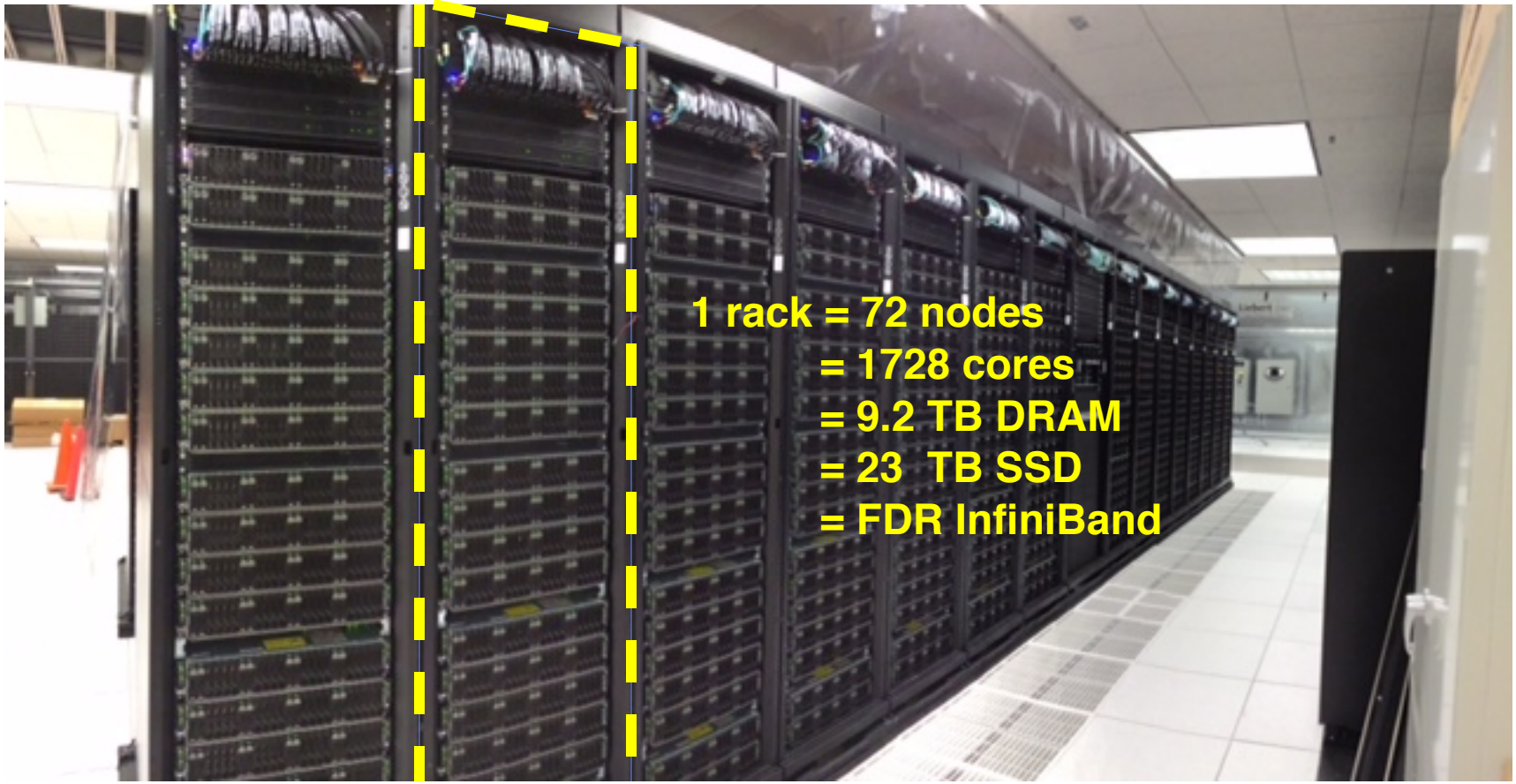
ALLOCATIONS & SCHEDULING
Optimized for throughput
Per-project allocation caps
Per-job core limits

- **Island architecture**
- **Mix of node types**
- **Virtualized HPC clusters**

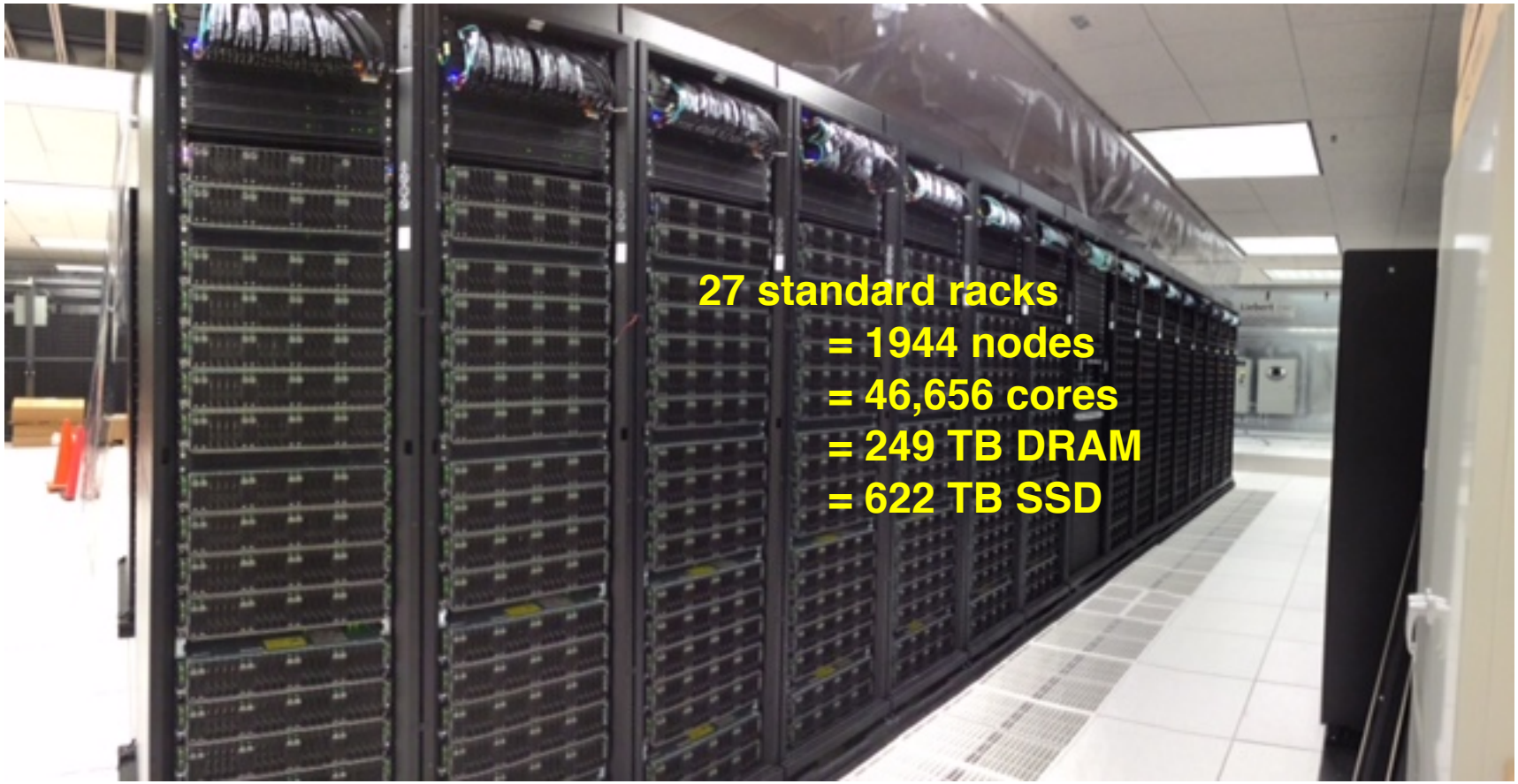
Comet: System Characteristics

- **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
  - 16 cores/processors => 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



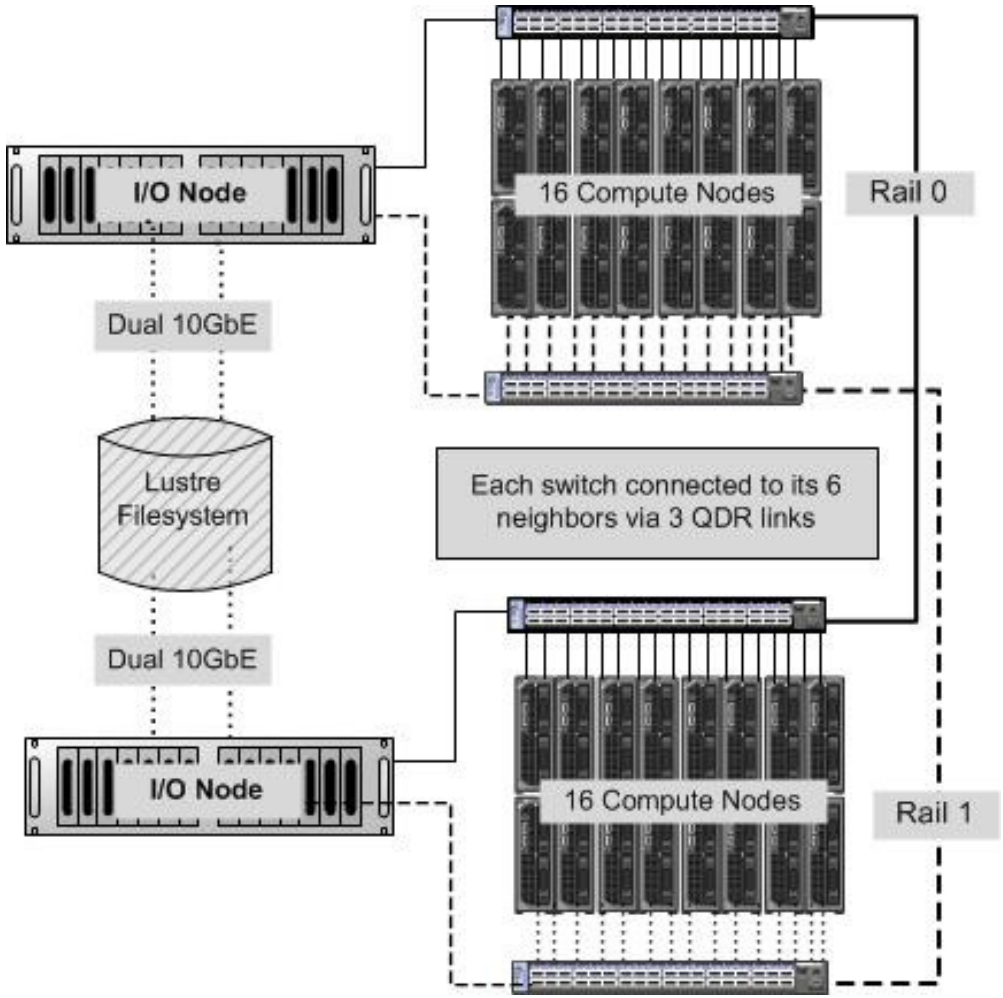
Gordon – A Data Intensive Supercomputer

- Designed to accelerate access to massive amounts of data in areas of genomics, earth science, engineering, medicine, and others
- Emphasizes memory and IO over FLOPS.
- Appro integrated 1,024 node Sandy Bridge cluster
- 300 TB of high performance Intel flash
- Large memory supernodes via vSMP Foundation from ScaleMP
- 3D torus interconnect from Mellanox
- In production operation since February 2012
- Funded by the NSF and available through the NSF Extreme Science and Engineering Discovery Environment program (XSEDE)

Gordon System Specification

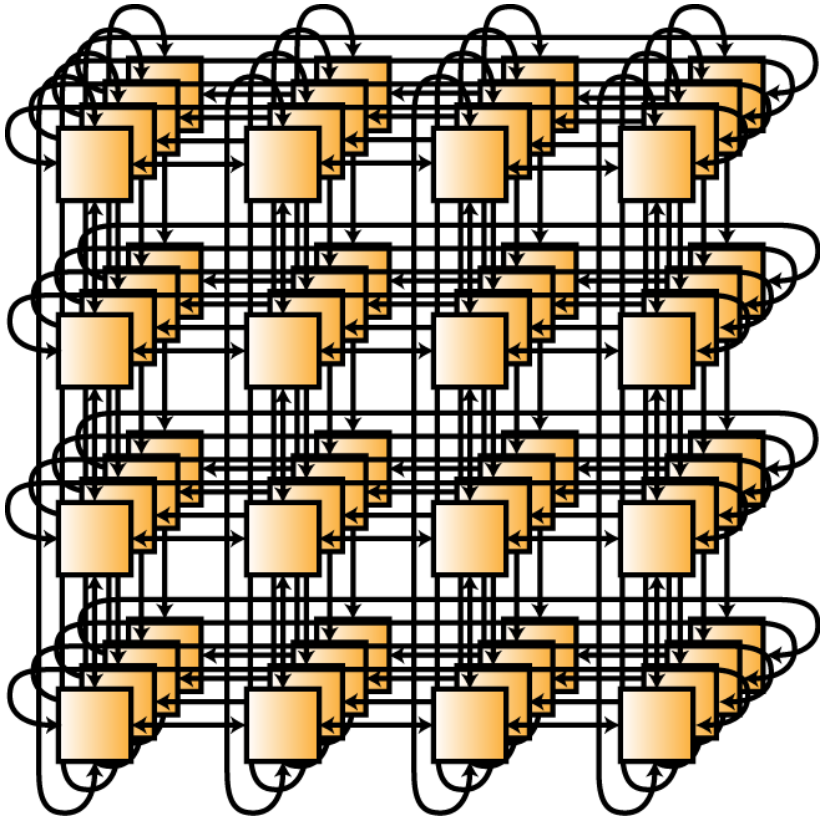
Intel Sandy Bridge Compute node	
Sockets	2
Cores	16
Clock speed	2.6
DIMM slots per socket	4
DRAM capacity	64 GB
Intel Flash I/O Node	
NAND flash SSD drives	16
SSD capacity per drive/Capacity per node/total	300 GB / 4.8 TB / 300 TB
Flash bandwidth per drive (read/write) Flash bandwidth per node (write/read)	270 MB/s / 210 MB/s 4.3 /3.3 GB/s
SMP Super-Node	
Compute nodes	32
I/O nodes	2
Addressable DRAM	2 TB
Addressable memory including flash	12TB
Gordon	
Compute Nodes	1,024
Total compute cores	16,384
Peak performance	341TF
Aggregate memory	64 TB
InfiniBand Interconnect	
Aggregate torus BW	9.2 TB/s
Type	Dual-Rail QDR InfiniBand
Link Bandwidth	8 GB/s (bidirectional)
Latency (min-max)	1.25 µs – 2.5 µs
Disk I/O Subsystem	
Total storage	/oasis/scratch (1.6 PB), /oasis/projects/nsf(1.5PB)
I/O bandwidth	100 GB/s
File system	Lustre

Subrack Level Architecture



16 compute nodes per subrack

3D Torus of Switches

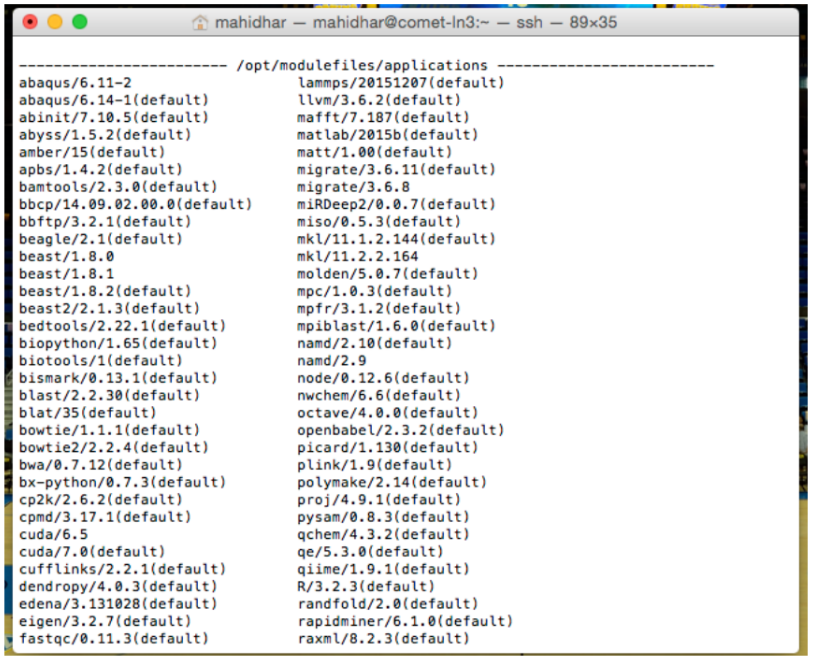


- Linearly expandable
- Simple wiring pattern
- Short Cables- Fiber Optic cables generally not required
- Lower Cost :40% as many switches, 25% to 50% fewer cables
- Works well for localized communication
- Fault Tolerant within the mesh with 2QoS Alternate Routing
- Fault Tolerant with Dual-Rails for all routing algorithms

3rd dimension wrap-around not shown for clarity

Software – Applications, Libraries

- Users can manage environment via modules.
- Applications packaged into “Rocks Rolls” that can built and deployed on any of the SDSC systems. Benefits wider community deploying software on their Rocks clusters. Available at **https://github.com/sdsc**.
- Efficient system administration pooling software install/testing efforts from different projects/machines – Comet benefits from work done for Trestles, Gordon, and Triton Shared Computing Cluster (TSCC).
- Users benefit from a familiar applications environment across SDSC systems => Comet, Gordon, and TSCC.



Compilers, Applications, Libraries

Category	List of Software/Libraries
Compilers	**Intel, PGI, GNU, MVAPICH2,OPENMPI**
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***
Visualization	**VisIt, IDL**
Numerical libraries	**ATLAS, FFTW, GSL, LAPACK, MKL, ParMETIS, PETSc, ScaLAPACK, SPRNG, Sundials, SuperLU, Trilinios**
Debugging/Profiling	**DDT, PAPI, TAU, mpiP**
GPU enabled software	**AMBER, GROMACS, NAMD, LAMMPS, *Caffe*, BEAST, *HOOMD, mvapich2-gdr***

Apps in red are unique to Comet and Gordon

SDSC HPC Resources: Running Jobs

Running Batch Jobs

- Comet uses SLURM for scheduling. The bulk of our examples today will be on Comet.

- Gordon and TSCC use the TORQUE/PBS resource manager for running jobs. On Gordon we have the Catalina scheduler to control the workload.

- Both schedulers allows the user to submit one or more jobs for execution, using parameters specified in a job script.

- Earlier we copied the examples into our home directory:

  /home/$USER/SI2016/INTRO

Comet – Compiling/Running Jobs

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

  cd /home/$USER/SI2016/INTRO/COMET

- **Verify modules loaded:**

  module list

  Currently Loaded Modulefiles:

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

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

  cd /home/$USER/SI2016/INTRO/COMET/MPI

  mpif90 -o hello_mpi hello_mpi.f90

- **Verify executable has been created:**

  ls -lt hello_mpi

  -rwxr-xr-x 1 etrain61 gue998 721912 Jul 31 12:16 **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 00:10:00

#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

[1] Submit the job:

```
cd /home/$USER/SI2016/INTRO/COMET/MPI/IBRUN
sbatch hellompi-slurm.sb
```

[2] Check status:

```
squeue -u $USER
```

[3] Output once it completes running:

```
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 /s
hare/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/SI2016/INTRO/COMET/OPENMP

- **Compile using –openmp flag:**

  ifort -o hello_openmp -openmp hello_openmp.f90

- **Verify executable was created:**

[etrain61@comet-ln2 OPENMP]$ **ls -lt hello_openmp**
-rwxr-xr-x 1 etrain61 gue998 728112 Jul 31 12:22 **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 00:10:00

#SET the number of openmp threads
export OMP_NUM_THREADS=24

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

Output from OpenMP Job

[1] Submit the job:

```
sbatch openmp-slurm.sb
```

[2] Check job status:

```
squeue -u $USER
```

[3] Check output:

```
[etrain61@comet-ln2 OPENMP]$ more hello_openmp.3624662.comet-06-17.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 00:10:00

#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

cd /home/$USER/SI2016/INTRO/COMET/HYBRID

sbatch hybrid-slurm.sb

```
[user@comet-08-11 HYBRID]$ more hellohybrid.435461.comet-17-41.out
Hello from thread 0 out of 6 from process 2 out of 8 on comet-17-41.local
Hello from thread 3 out of 6 from process 2 out of 8 on comet-17-41.local
Hello from thread 4 out of 6 from process 2 out of 8 on comet-17-41.local
Hello from thread 5 out of 6 from process 2 out of 8 on comet-17-41.local
Hello from thread 0 out of 6 from process 3 out of 8 on comet-17-41.local
Hello from thread 2 out of 6 from process 2 out of 8 on comet-17-41.local
Hello from thread 1 out of 6 from process 3 out of 8 on comet-17-41.local
Hello from thread 2 out of 6 from process 3 out of 8 on comet-17-41.local
...
...
Hello from thread 4 out of 6 from process 7 out of 8 on comet-17-42.local
Hello from thread 2 out of 6 from process 7 out of 8 on comet-17-42.local
Hello from thread 3 out of 6 from process 7 out of 8 on comet-17-42.local
Hello from thread 5 out of 6 from process 7 out of 8 on comet-17-42.local
Hello from thread 1 out of 6 from process 6 out of 8 on comet-17-42.local
```

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 00:30:00

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

#Run IO benchmark
ibrun -np 4 $SLURM_SUBMIT_DIR/IOR.exe -F -t 1m -b 4g -v -v > output.$SLURM_JOBID
.log

#Copy back log file
cp output.$SLURM_JOBID.log $SLURM_SUBMIT_DIR/
```

Using SSD Scratch

- **Submit job:**

  cd /home/$USER/SI2016/INTRO/COMET/LOCALSCRATCH

  sbatch Localscratch-slurm.sb

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

```
[etrain61@comet-05-36 3624811]$ pwd
/scratch/etrain61/3624811
[etrain61@comet-05-36 3624811]$ ls -lt
total 3004535
-rw-r--r-- 1 etrain61 gue998 1222639616 Jul 31 12:36 testFile.00000002
-rw-r--r-- 1 etrain61 gue998 1245708288 Jul 31 12:36 testFile.00000003
-rw-r--r-- 1 etrain61 gue998 1534066688 Jul 31 12:36 testFile.00000001
-rw-r--r-- 1 etrain61 gue998 1339031552 Jul 31 12:36 testFile.00000000
-rw-r--r-- 1 etrain61 gue998       1107 Jul 31 12:36 output.3624811.log
```

- **Performance from single node (in log file copied back):**

```
Max Write: 178.20 MiB/sec (186.85 MB/sec)
Max Read:  13360.00 MiB/sec (14008.98 MB/sec)
```

Did we really get 13.4 GB/s from the SSD? *Hint: This data was just written before the read.*

Comet GPU Nodes

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

Hands On Example using OpenACC

GPU Node – OpenACC Example

- **Setup PGI compiler:**

  module purge

  module load pgi

- **Compiling:**

cd /home/$USER/SI2016/INTRO/COMET/OpenACC

pgcc -o laplace2d.openacc.exe -acc -Minfo=accel laplace2d.c

```
laplace:
    58, Generating copy(A[:][:])
        Generating create(Anew[:][:])
    63, Generating Tesla code
    64, Loop is parallelizable
    66, Loop is parallelizable
        Accelerator kernel generated
        64, #pragma acc loop gang /* blockIdx.y */
        66, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
        70, Max reduction generated for error
    74, Generating Tesla code
    75, Loop is parallelizable
    77, Loop is parallelizable
        Accelerator kernel generated
        75, #pragma acc loop gang /* blockIdx.y */
        77, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
```

GPU Nodes – OpenACC script

```
#!/bin/bash
#SBATCH --job-name="hpl"
#SBATCH --output="hpl.%j.%N.out"
#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH --gres=gpu:1
#SBATCH -t 00:10:00

./laplace2d.openacc.exe
```

GPU Nodes – OpenACC Example

```
main()
Jacobi relaxation Calculation: 4096 x 4096 mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 15.914405 s
```

Gordon : Filesystems

- **Lustre filesystems – Good for scalable large block I/O**
  - Accessible from both native and vSMP nodes.
  - /oasis/scratch/gordon – 1.6 PB, peak measured performance ~50GB/s on reads and writes.
  - /oasis/projects – 1.5PB
- **SSD filesystems**
  - /scratch local to each native compute node – 300 GB each.
  - /scratch on vSMP node – 4.8TB of SSD based filesystem.
  - Few “bigflash” nodes with 4.8TB of SSD space.
- **NFS filesystems (/home)**
- **Lets login:**

  ssh etrain*XY*@gordon.sdsc.edu

Gordon: Compiling/Running Jobs

- **Job Queue basics:**
  - Gordon uses the TORQUE/PBS Resource Manager with the Catalina scheduler to define and manage job queues.
  - Native/Regular compute (Non-vSMP) nodes accessible via “normal” queue.
  - vSMP node accessible via “vsmp” queue.

- **Workshop examples illustrate use of Gordon for two use cases.**

  - **hello_native.cmd** – script for running hello world example on native nodes (using MPI).

  - vSMP example to access the virtual large memory node.

Gordon – Compiling/Running Jobs

- **Change to workshop directory:**

  cd /home/$USER/SI2016/INTRO/GORDON

- **Verify modules loaded:**

  module list

  Currently Loaded Modulefiles:

  1) intel/2013.1.117 2) mvapich2_ib/1.9 3) gnubase/1.0

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

  mpif90 -o hello_world hello_mpi.f90

- **Verify executable has been created:**

  ls -lt hello_world

  -rwxr-xr-x 1 mahidhar hpss 735429 May 15 21:22 hello_world

Gordon: Hello World on native (non-vSMP) nodes

The submit script (located in the workshop directory) is hello_native.cmd

```
#!/bin/bash
#PBS -q normal
#PBS -N hello_native
#PBS -l nodes=4:ppn=1:native
#PBS -l walltime=0:10:00
#PBS -o hello_native.out
#PBS -e hello_native.err
#PBS -V
##PBS -M youremail@xyz.edu
##PBS -m abe
#PBS –A gue998
cd $PBS_O_WORKDIR
mpirun_rsh -hostfile $PBS_NODEFILE -np 4 ./hello_world
```

Gordon: Output from Hello World

- **Submit job using "qsub hello_native.cmd"**

  $ qsub hello_native.cmd

  845444.gordon-fe2.local

- **Output:**

  $ more hello_native.out

  node 2 : Hello world

  node 1 : Hello world

  node 3 : Hello world

  node 0 : Hello world

  Nodes: gcn-15-58 gcn-15-62 gcn-15-63 gcn-15-68

Sample VSMP queue script

```
#!/bin/bash
#PBS -q vsmp
#PBS -N hello_vsmp
#PBS -l nodes=1:ppn=16:vsmp
#PBS -l walltime=0:10:00
#PBS -o hello_vsmp.out
#PBS -e hello_vsmp.err
#PBS -V
##PBS -M youremail@xyz.edu
##PBS -m abe
#PBS -A gue998
cd $PBS_O_WORKDIR
export LD_PRELOAD=/opt/ScaleMP/libvsmpclib/0.1/
lib64/libvsmpclib.so
export PATH="/opt/ScaleMP/numabind/bin:$PATH"
export KMP_AFFINITY=compact,verbose,0,`numabind --
offset 8`
export OMP_NUM_THREADS=8
./hello_vsmp
```

Bundling Jobs

- HPC systems typically have 16 or more cores per node. On several systems these nodes are provided on a “non-shared” basis.
- Often users have several jobs to run but may not be able to use all the cores on a node for each job => bundling jobs can help throughput.
- SDSC User Services developed bundler scripts to enable such use cases:
  - https://github.com/sdsc/sdsc-user/tree/master/bundler
- Can bundle serial and OpenMP jobs.

Job Bundler Example Script

```
#!/bin/bash
…
#PBS -N bundler.gordon
#PBS -q normal
#PBS -l nodes=1:ppn=8:native
#PBS -l walltime=1:00:00
#PBS -v Catalina_maxhops=None,QOS=0

TASKS=tasks              # the name of your tasks list

cd $PBS_O_WORKDIR
module load python    # necessary for bundler.py on Gordon
mpirun_rsh -export \
   -np $PBS_NP \
   -hostfile $PBS_NODEFILE \
   /home/diag/opt/mpi4py/mvapich2/intel/1.3.1/lib/python/mpi4py/bin/python-mpi \
   /home/diag/opt/sdsc-user/bundler/bundler.py $TASKS
```

Job Bundler Example - Tasks

```
$ more tasks
/bin/date > task1.out
/bin/hostname > task2.out
cat /proc/cpuinfo > cpuinfo.dat
df l grep oasis > Lustre.fs.info
df l grep scratch > task5.out
ls /scratch/$USER > task6.out
ps -ef l grep $USER > task7.out
/usr/bin/w > task8.out
```

Output from bundler

Got 8 task slots

MPI process 6 on gcn-3-58.sdsc.edu running task [ps -ef l grep $USER > task7.out]

MPI process 2 on gcn-3-58.sdsc.edu running task [cat /proc/cpuinfo > cpuinfo.dat]

MPI process 3 on gcn-3-58.sdsc.edu running task [df l grep oasis > Lustre.fs.info]

MPI process 0 on gcn-3-58.sdsc.edu running task [/bin/date > task1.out]

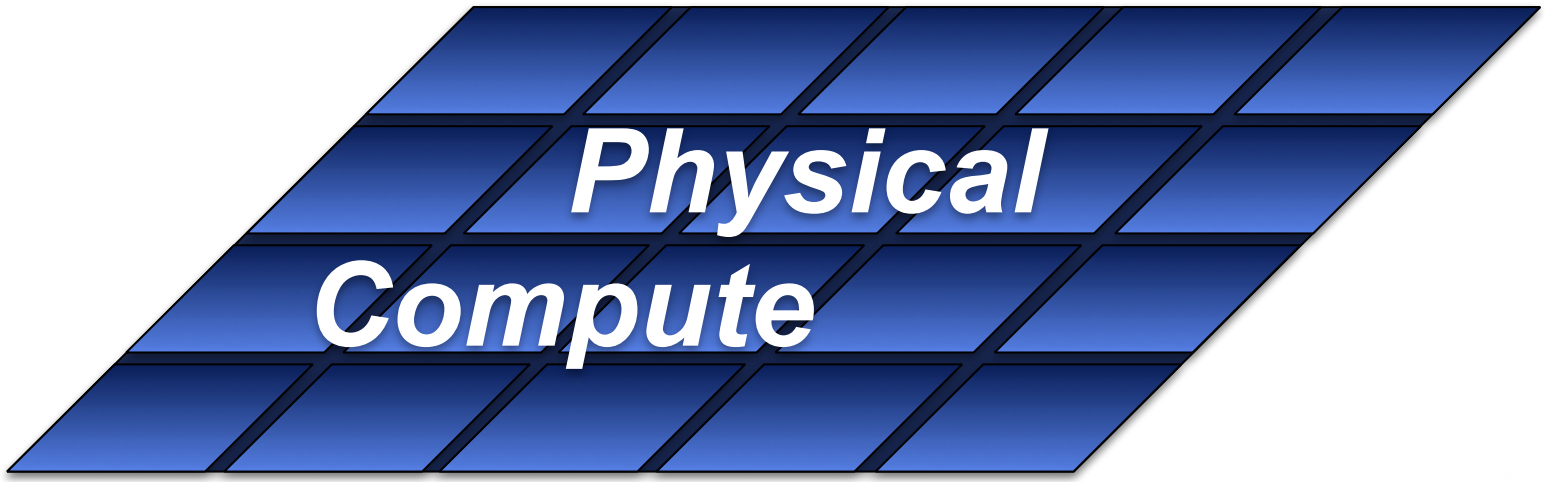
MPI process 7 on gcn-3-58.sdsc.edu running task [/usr/bin/w > task8.out]

MPI process 4 on gcn-3-58.sdsc.edu running task [df l grep scratch > task5.out]

MPI process 5 on gcn-3-58.sdsc.edu running task [ls /scratch/$USER > task6.out]

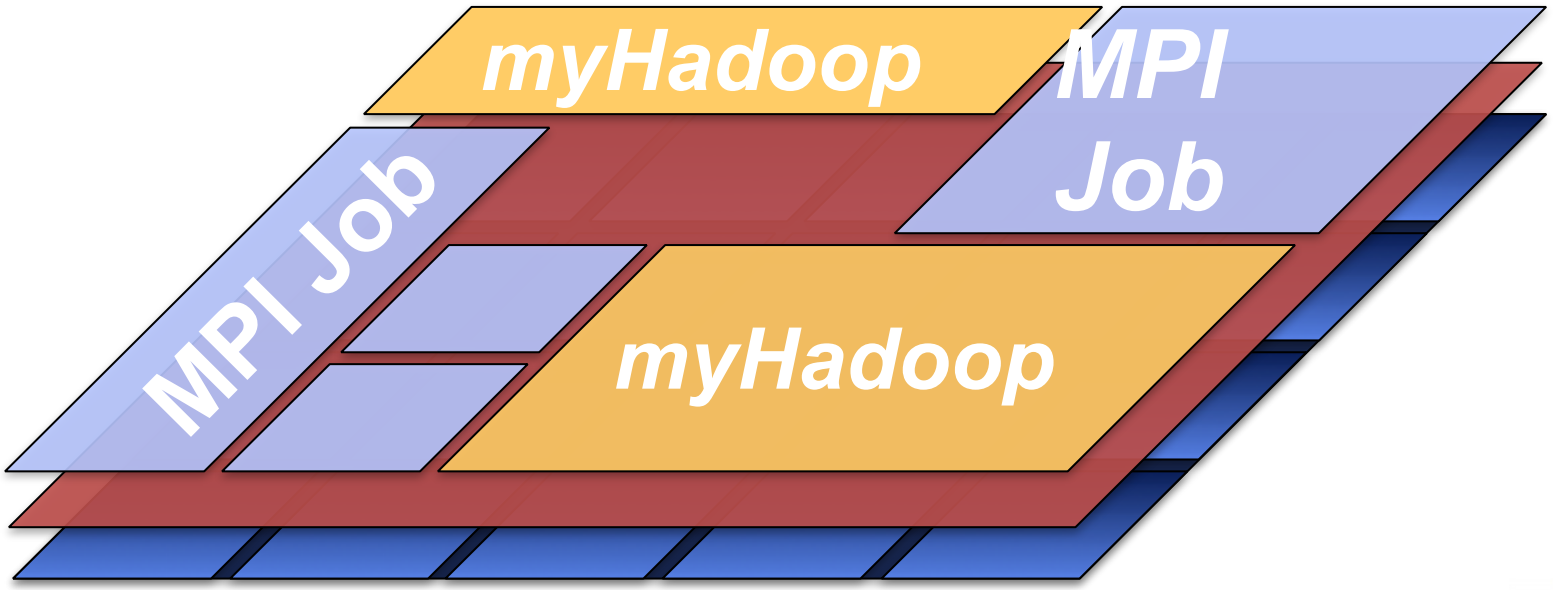
MPI process 1 on gcn-3-58.sdsc.edu running task [/bin/hostname > task2.out]

Add Data Analysis to Existing Compute Infrastructure

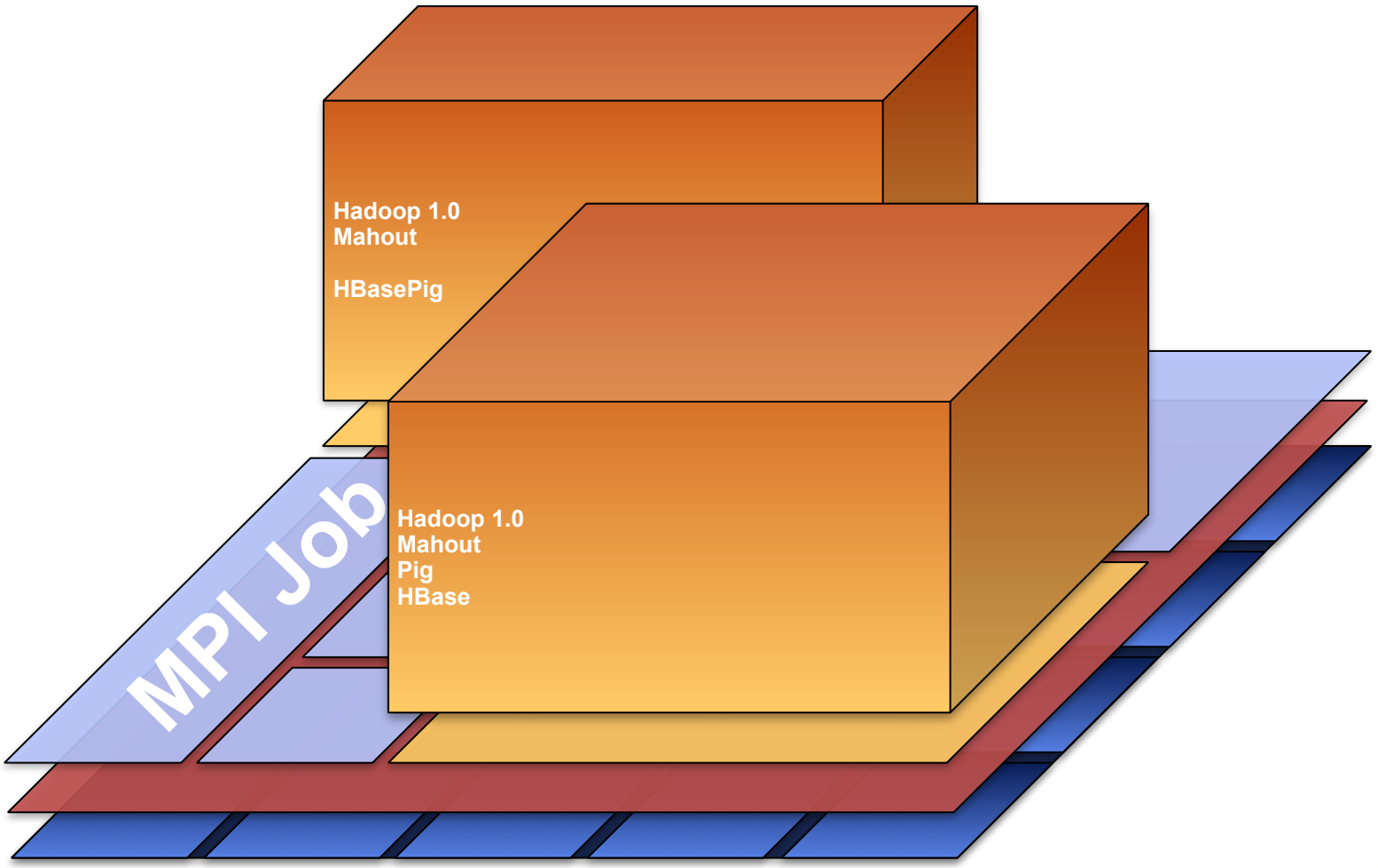


Add Data Analysis to Existing Compute Infrastructure

Add Data Analysis to Existing Compute Infrastructure



Add Data Analysis to Existing Compute Infrastructure



Anagram Example

- **Source:**

  https://code.google.com/p/hadoop-map-reduce-examples/wiki/Anagram_Example

- **Uses Map-Reduce approach to process a file with a list of words, and identify all the anagrams in the file**

- **Code is written in Java. Example has already been compiled and the resulting jar file is in the example directory.**

Anagram Example – Submit Script

```
#!/bin/bash
#SBATCH --job-name="hadoopanagram"
#SBATCH --output="hadoopanagram.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=24
#SBATCH -t 00:15:00
export HADOOP_CONF_DIR=/home/$USER/cometcluster
export WORKDIR=`pwd`
module load hadoop
myhadoop-configure.sh
export PATH=/opt/hadoop/2.6.0/sbin:$PATH
start-all.sh
hdfs dfs -mkdir -p /user/$USER/input
hdfs dfs -put $WORKDIR/SINGLE.TXT /user/$USER/input/SINGLE.TXT
hadoop jar $WORKDIR/AnagramJob.jar /user/$USER/input/SINGLE.TXT /user/$USER/outp
ut
hdfs dfs -get /user/$USER/output/part* $WORKDIR/
stop-all.sh
myhadoop-cleanup.sh
```

Anagram Example – Sample Output

Make sure you are Comet and then do:

```
cd /home/$USER/SI2016/INTRO/COMET/HADOOP/ANAGRAM_Hadoop2
sbatch anagram.script
```

After the job completes:

```
cat part-00000
…
aabcdelmnu          manducable,ambulanced,
aabcdeorrsst        broadcasters,rebroadcasts,
aabcdeorrst         rebroadcast,broadcaster,
aabcdkrsw           drawbacks,backwards,
aabcdkrw  drawback,backward,
aabceeehlnsst       teachableness,cheatableness,
aabceeelnnrsstu     uncreatableness,untraceableness,
aabceeelrrt         recreatable,retraceable,
aabceehlt  cheatable,teachable,
aabceellr  lacerable,clearable,
aabceelnrtu         uncreatable,untraceable,
aabceelorrrstv      vertebrosacral,sacrovertebral,
…
…
```

Spark on Comet

```
#!/bin/bash
#SBATCH --job-name="graphx-demo"
#SBATCH --output="graphx-demo.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 00:30:00

### Environment setup for Hadoop and Spark
module load spark
export PATH=/opt/hadoop/2.6.0/sbin:$PATH
export HADOOP_CONF_DIR=$HOME/mycluster.conf
export WORKDIR=`pwd`

myhadoop-configure.sh

### Start HDFS.  Starting YARN isn't necessary since Spark will be running in
### standalone mode on our cluster.
start-dfs.sh

### Load in the necessary Spark environment variables
source $HADOOP_CONF_DIR/spark/spark-env.sh

### Start the Spark masters and workers.  Do NOT use the start-all.sh provided
### by Spark, as they do not correctly honor $SPARK_CONF_DIR
myspark start

### Copy the data into HDFS
hdfs dfs -mkdir -p /user/$USER
hdfs dfs -put $WORKDIR/facebook_combined.txt /user/$USER/

run-example org.apache.spark.examples.graphx.LiveJournalPageRank facebook_combined.txt --numEPart=8

### Shut down Spark and HDFS
myspark stop
stop-dfs.sh

### Clean up
myhadoop-cleanup.sh
```

Summary, Q/A

- **Access options – ssh clients, XSEDE User Portal, Science Gateways.**
- **Data Transfer options – scp, globus-url-copy (gridftp), globus online, and XSEDE User Portal File Manager.**
- **SLURM queues on Comet: compute, shared, gpu, gpu-shared, and debug.**
- **PBS queues on Gordon: normal (native, non-vSMP) and vsmp.**
- **Use SSD local scratch where possible. Excellent for codes like Gaussian, Abaqus, Q-Chem.**
- **Hadoop, Spark clusters can be launched within SLURM (Comet) and PBS/Torque (Gordon) framework using myHadoop set up.**