Introduction to Using Gordon

2012 International Summer School on AstroComputing
San Diego, CA July 9-20

Robert Sinkovits
Gordon Applications Lead
San Diego Supercomputer Center

An apology in advance

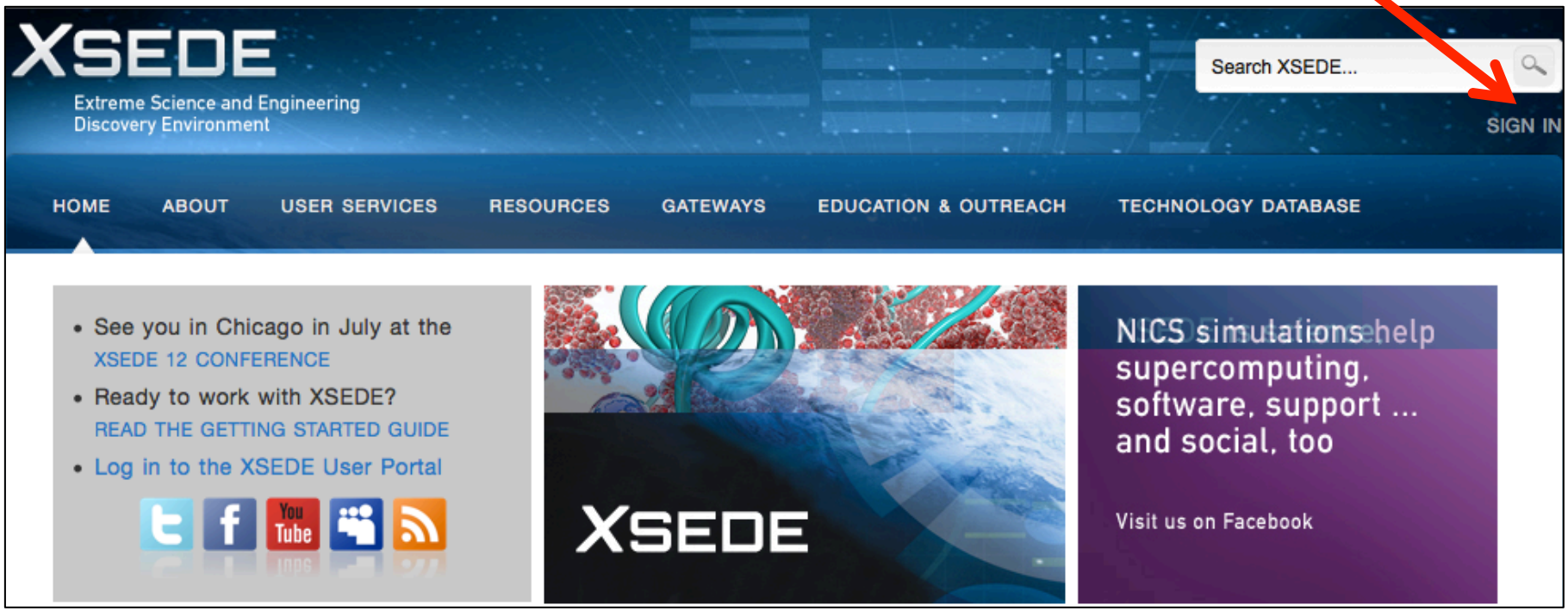
Those of you who are already experienced users of NSF, DoE, DoD, PRACE, etc. supercomputing resources may find much of the material being presented here to be very basic.

Our intention is to bring everyone up to speed early in the summer school so that the on-hands exercises will go as smoothly as possible

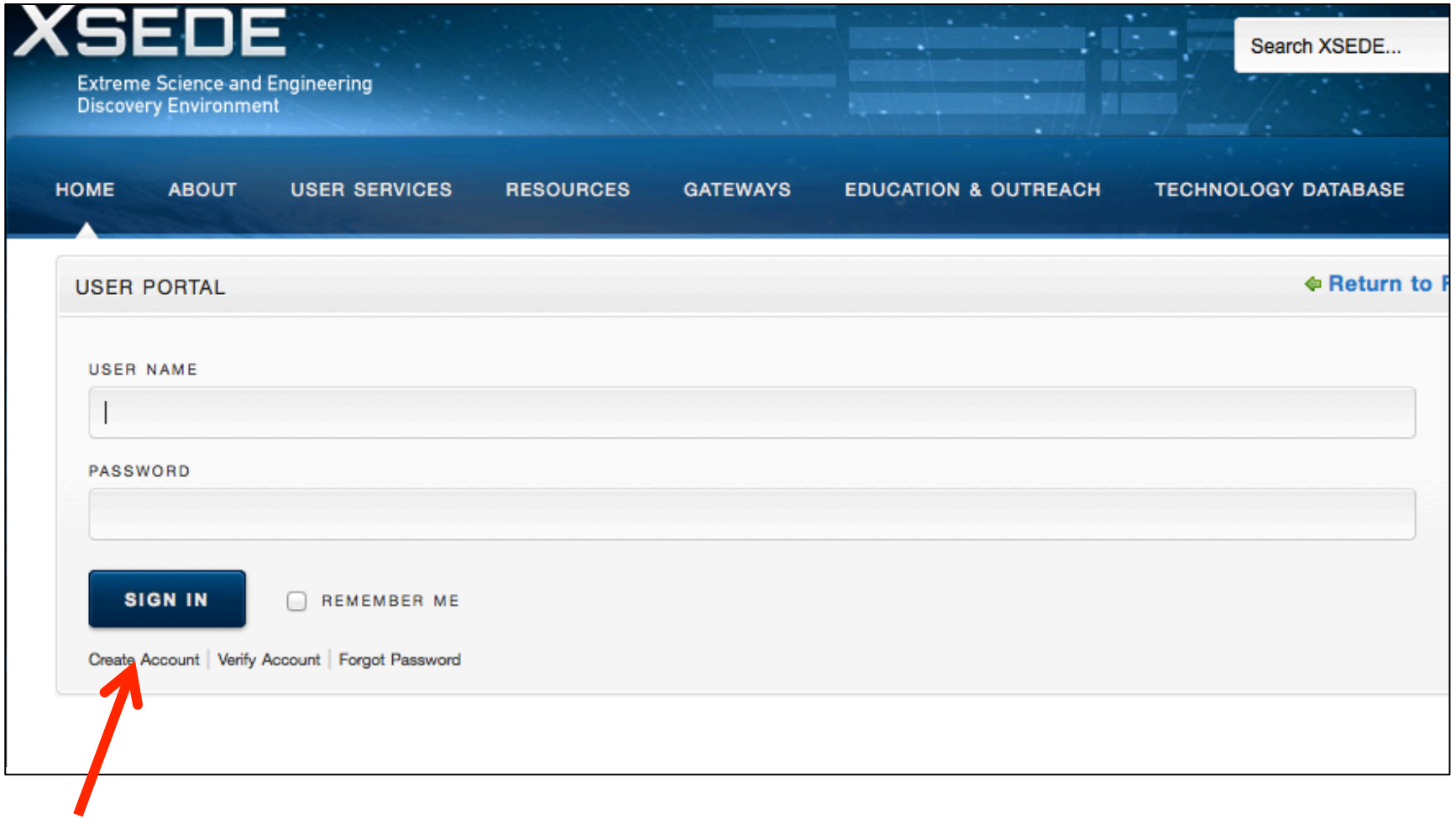
If you don't already have an XSEDE portal account

Go to https://www.xsede.org

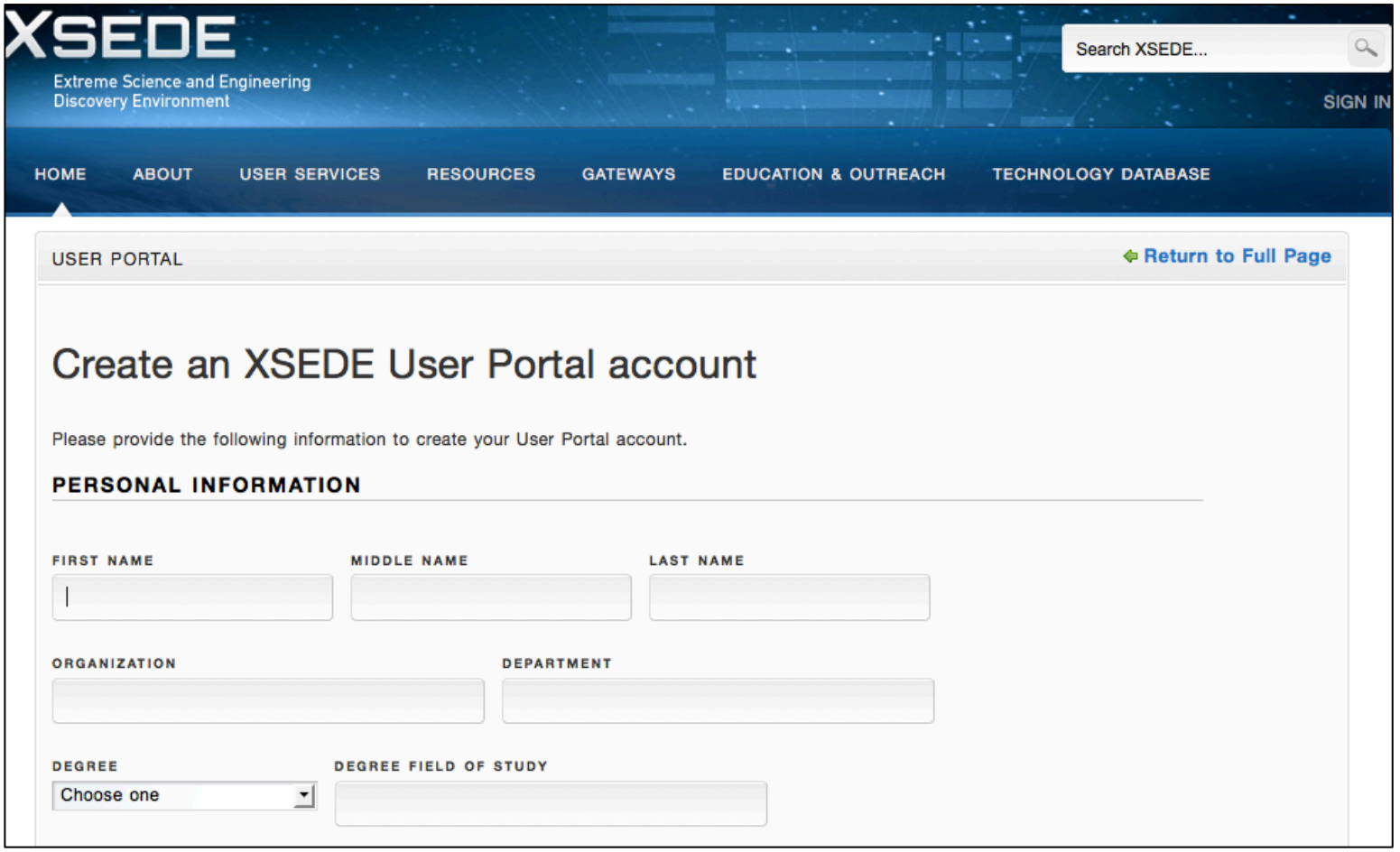
Click the "SIGN IN" link



… click the “Create Account” link



… Fill out the information, create account and email your XSEDE portal name to rpwagner@sdsc.edu



Why Gordon?

Designed for data and memory intensive applications that don't run well on traditional distributed memory machines

- Large shared memory requirements
- Serial or threaded (OpenMP, Pthreads)
- Limited scalability
- High performance data base applications
- Random I/O combined with very large data sets
- Large scratch files

Gordon is a national resource made possible through a grant from the National Science Foundation. One of three OCI Track 2D awards for innovative systems

Available to all U.S. academic researchers on a competitive basis and on a limited basis for-fee to most non-academic users

Design
Deployment
Support

Processors
Motherboards
Flash drives

Integrator

vSMP Foundation

Funding
OCI #0910847

3D Torus

Gordon Hardware overview

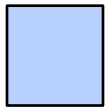
- **1024 dual-socket compute nodes**
  - 2 x Intel EM64T Xeon E5 (Sandy Bridge) processors
  - 64 GB DDR3-1333 memory
  - 80 GB local Flash memory
  - ***64 TB total DRAM, 341 TFlop peak performance***
- **64 dual-socket I/O nodes**
  - 2 x Intel Westmere processors
  - 48 GB DDR3-1333 memory
  - 16 x 300 GB Intel 710 (Westmere) Solid State Drives (SSD)
  - ***300 TB total flash memory***
- **Dual-rail 3D torus InfiniBand QDR (40 Gbit/s) network**
- **4 PB Lustre-based parallel file system**
  - ***Capable of delivering up to 100 GB/s to Gordon***

A great Gordon application will …

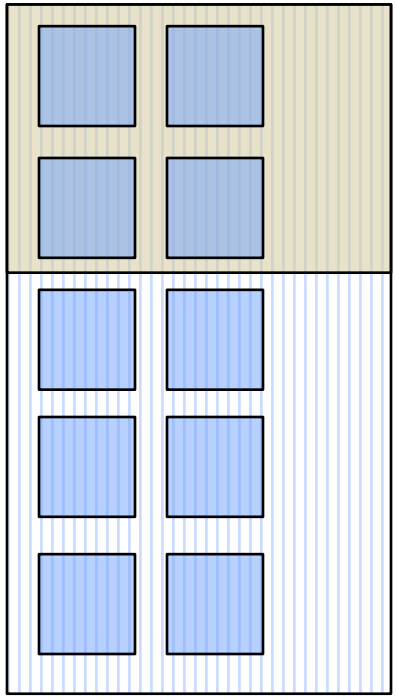
- Make use of the flash storage for scratch/staging
  64 I/O nodes each w/ 16 x 300 GB SSDs (4.4 TB usable)
- Need the large, logical shared memory
  ~ 1 TB DRAM & 256 cores, w/ larger configurations possible
- Be a threaded app that scales to very large number of cores
- Require large physical memory per node
  64 GB/node, 4 GB/core
- Be able to use the AVX instructions (8 flops/cycle/core)
- Need a high-bandwidth, low-latency inter-processor network

Foxglove Calculation using Gaussian 09 with vSMP - MP2 Energy Gradient Calculation

The Foxglove plant (Digitalis) is studied for its medicinal uses. *Digoxin*, an extract of the Foxglove, is used to treat a variety of conditions including diseases of the heart. There is some recent research that suggests it may also be a beneficial cancer treatment.



1 Compute node
= (16 cores/node)
64 GB/node)



Processor footprint - 4 nodes
64 threads

Time to solution:
43,000s

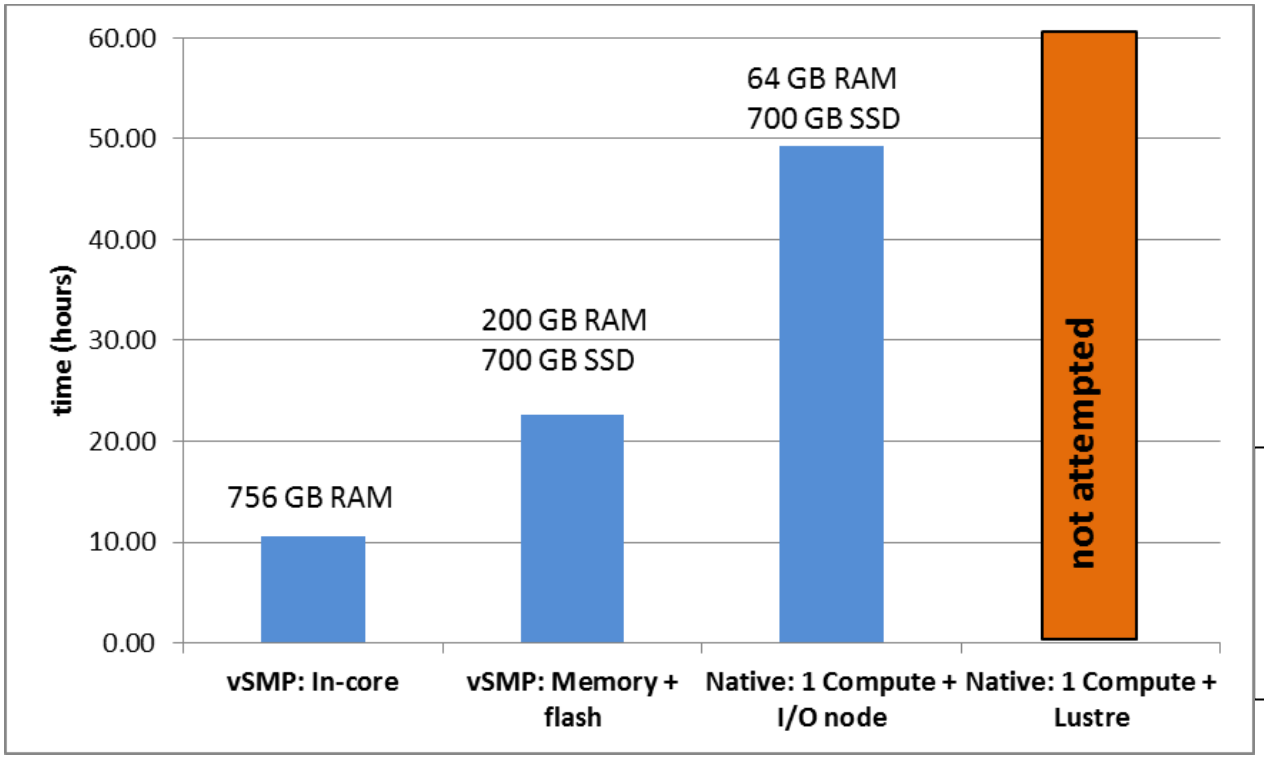
Memory footprint – 10 nodes
700 GB

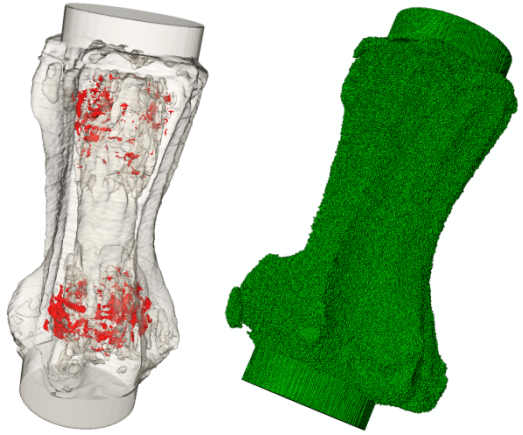
V	M	F
C	T	L

Source: *Jerry Greenberg, San Diego Supercomputer Center. January, 2012.*

Axial compression of caudal rat vertebra using Abaqus and vSMP

The goal of the simulations is to analyze how small variances in boundary conditions effect high strain regions in the model. The research goal is to understand the response of trabecular bone to mechanical stimuli. This has relevance for paleontologists to infer habitual locomotion of ancient people and animals, and in treatment strategies for populations with fragile bones such as the elderly.





- 5 million quadratic, 8 noded elements
- Model created with custom Matlab application that converts 25^3 micro CT images into voxel-based finite element models

V	M	F
C	T	L

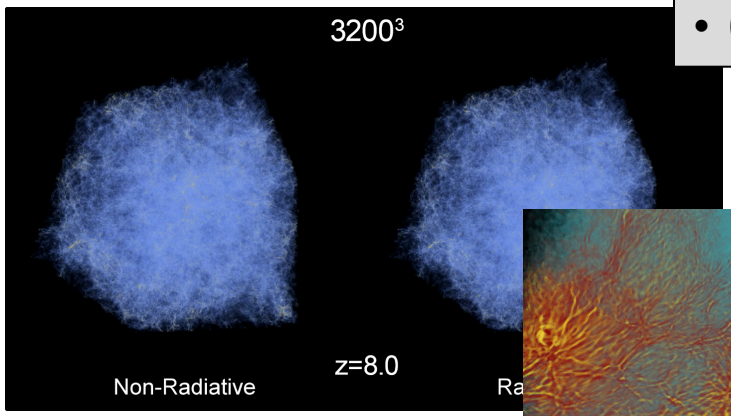
Source: *Matthew Goff, Chris Hernandez. Cornell University. Used by permission. 2012*

Cosmology simulation - matter power spectrum measurement using vSMP

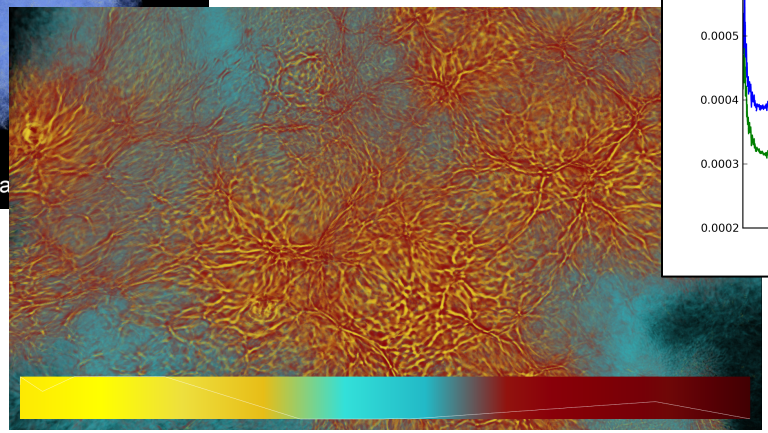
Goal is to measure the effect of the light from the first stars on the evolution of the universe. To quantitatively compare the matter distribution of each simulation, we use radially binned 3D power spectra.

- 2 simulations
- 3200^3 uniform 3D grids
- 15k+ files each

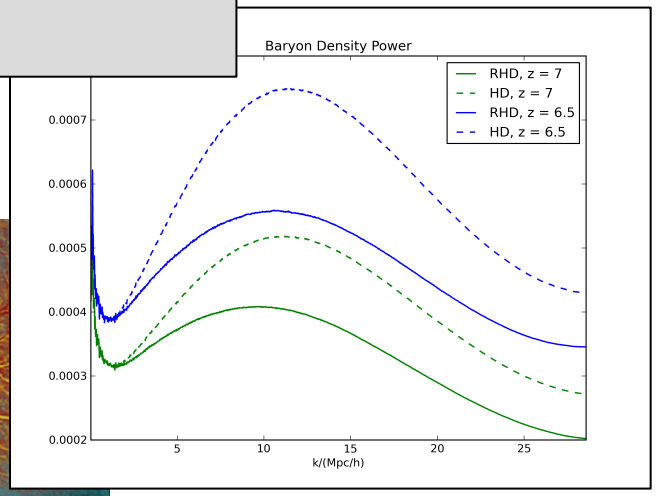
- Existing OpenMP code
- ~256GB memory used
- ~5 ½ hours per field
- 0 development effort



Individual simulations



Difference



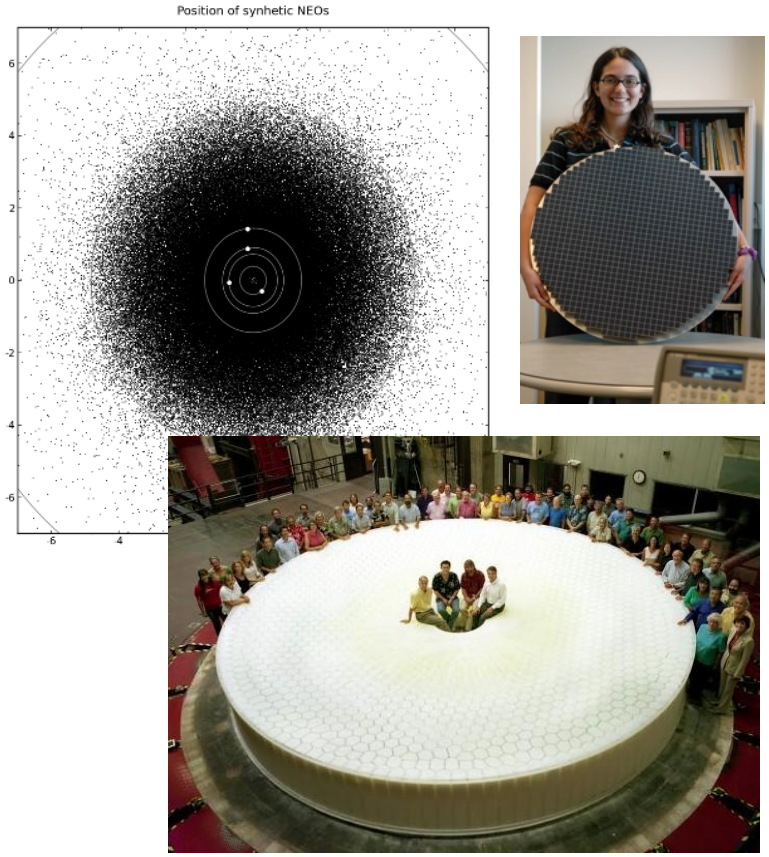
Power spectra

V	M	F
C	T	L

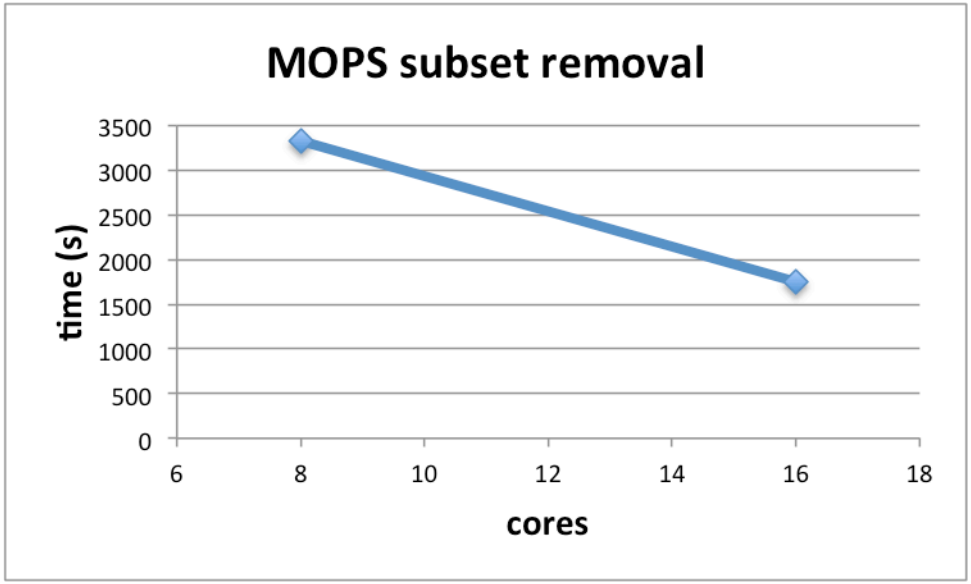
Source: *Rick Wagner, Michael L. Norman. SDSC.*

LSST – Moving Object Pipeline System

Images collected by the Large Synoptic Survey Telescope (LSST) will be be processed using the Moving Object Pipeline System (MOPS). Detections from consecutive nights are grouped together into tracks that potentially represent small portions of the asteroids' sky-plane motion



Run time for subset removal algorithm scales almost linearly out to 16 cores

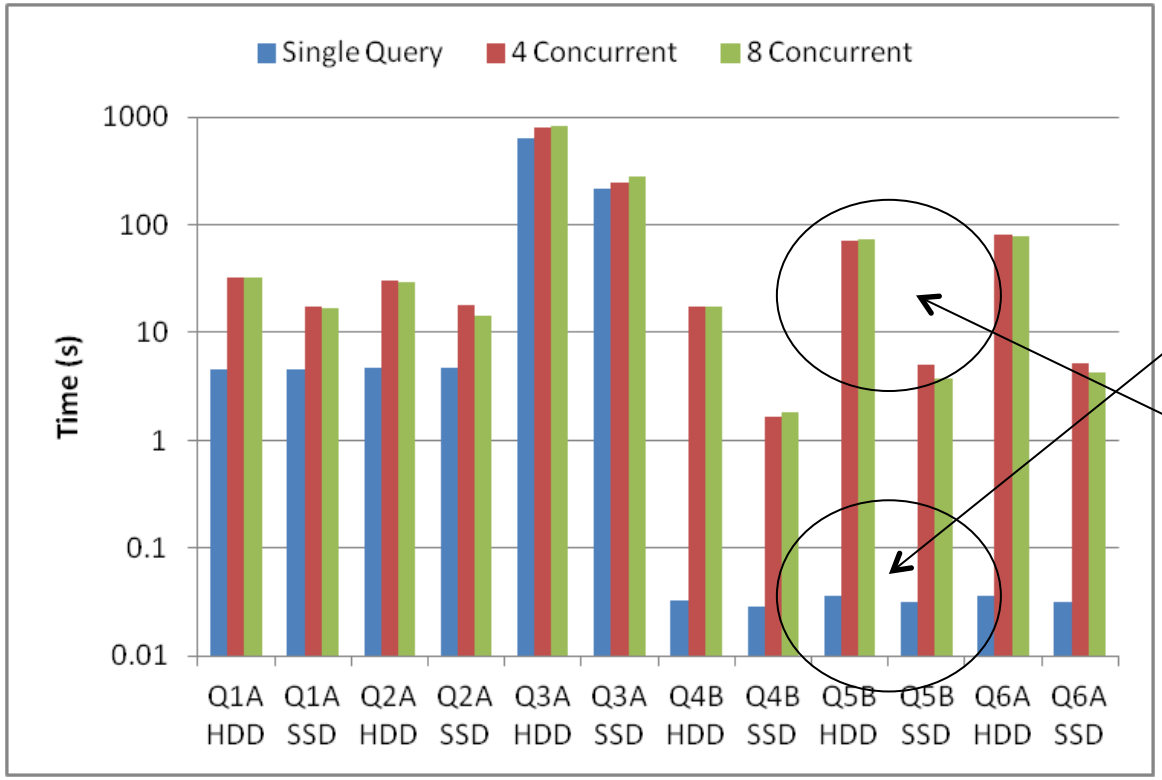


V	M	F
C	T	L

Source: *Jonathan Myers, LSST Used by permission. 6/4/2012*

PDB Query Comparisons, with DB2 Database on two Gordon I/O Nodes: One with HDD's, One with SSD's

The Protein Data Bank (PDB): Is the single worldwide repository of information about the 3D structures of large biological molecules. These are the molecules of life that are found in all organisms. Understanding the shape of a molecule helps to understand how it works.



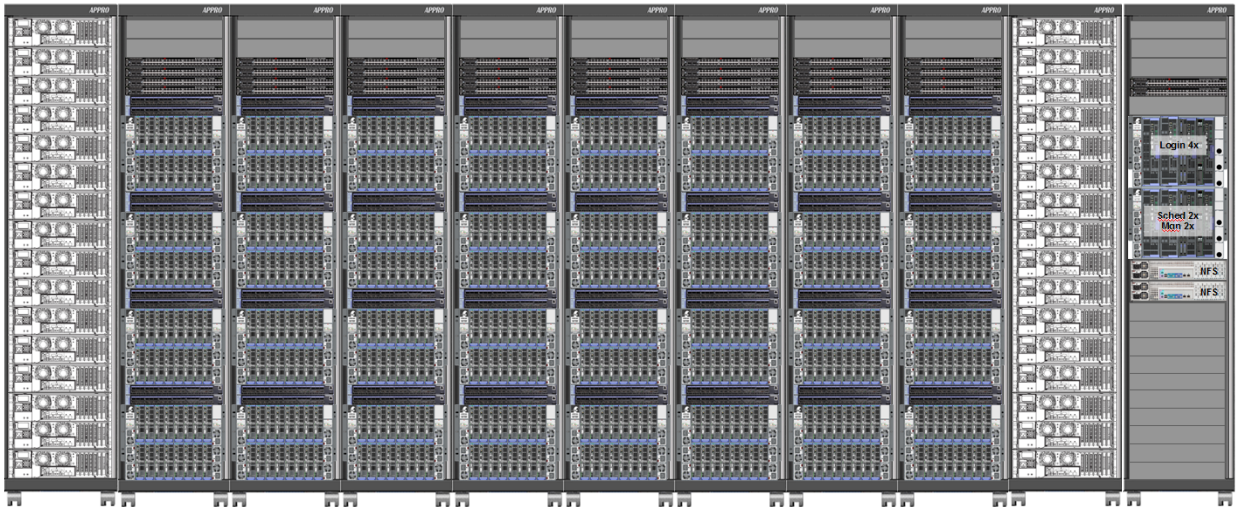
RCSB PDB PROTEIN DATA BANK

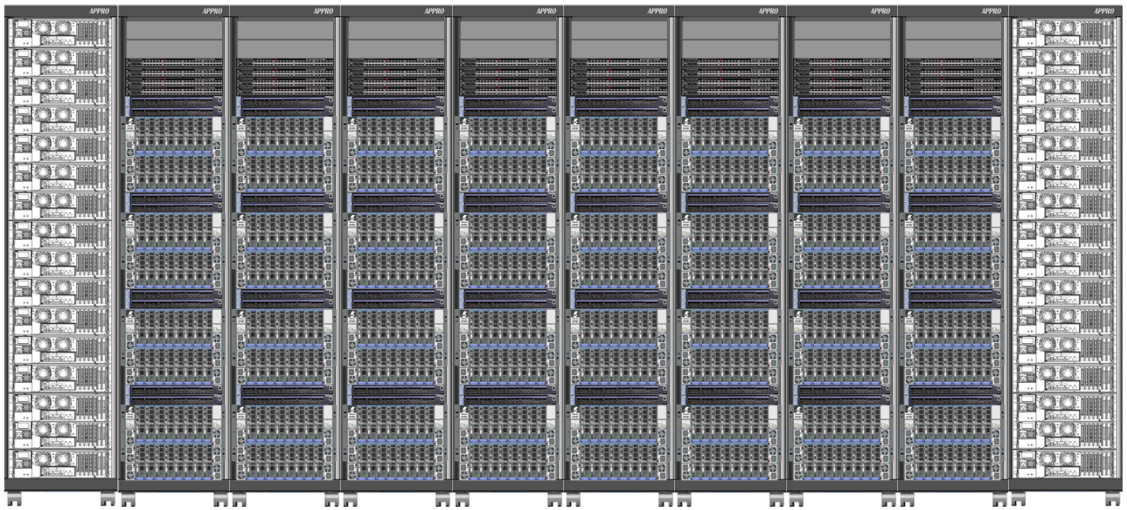
- For single queries, HDD and SSD perform about the same.
- For concurrent queries, SSD's achieve big speedup.
- Q5B is > 10x, and performance varies by type of query

V	M	F
C	T	L

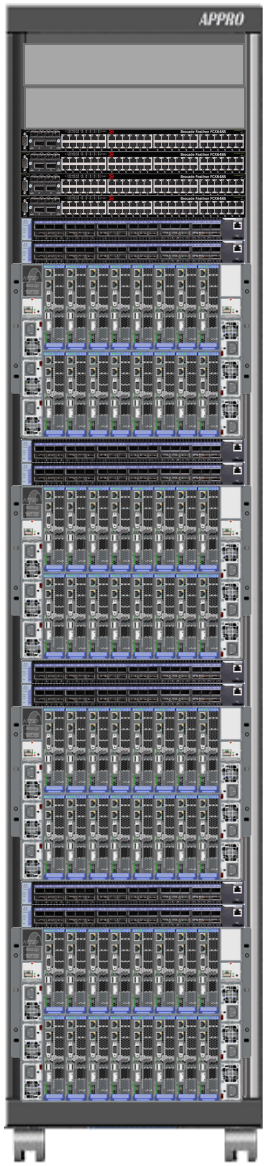
Source: *Vishwinath Nandigam, San Diego Supercomputer Center. 2011*

Gordon Rack Layout

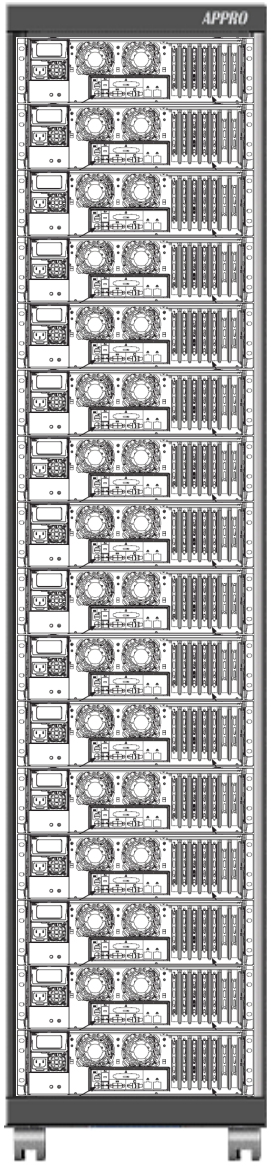




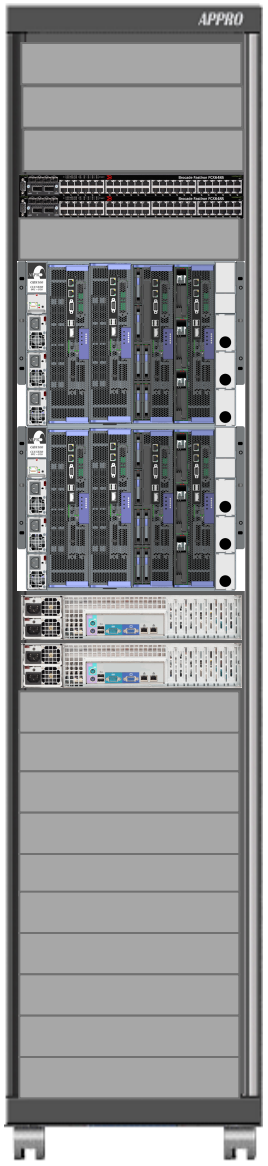
16 compute node racks
4 I/O node racks
1 service rack



CN Rack



ION Rack



Service Nodes Rack

Compute node racks:
4 Appro subracks
64 blades

ION racks:
16 Gordon I/O nodes

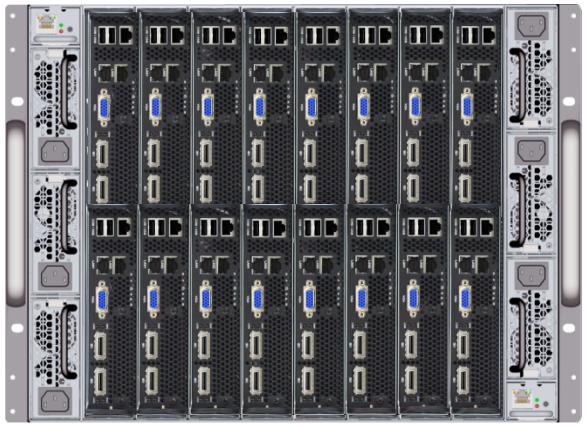
Service rack:
4 login nodes
2 NFS servers
2 Scheduler nodes
2 management nodes

Based on Appro GreenBlade™ 8000 Series

designed for improved reliability and energy efficiency

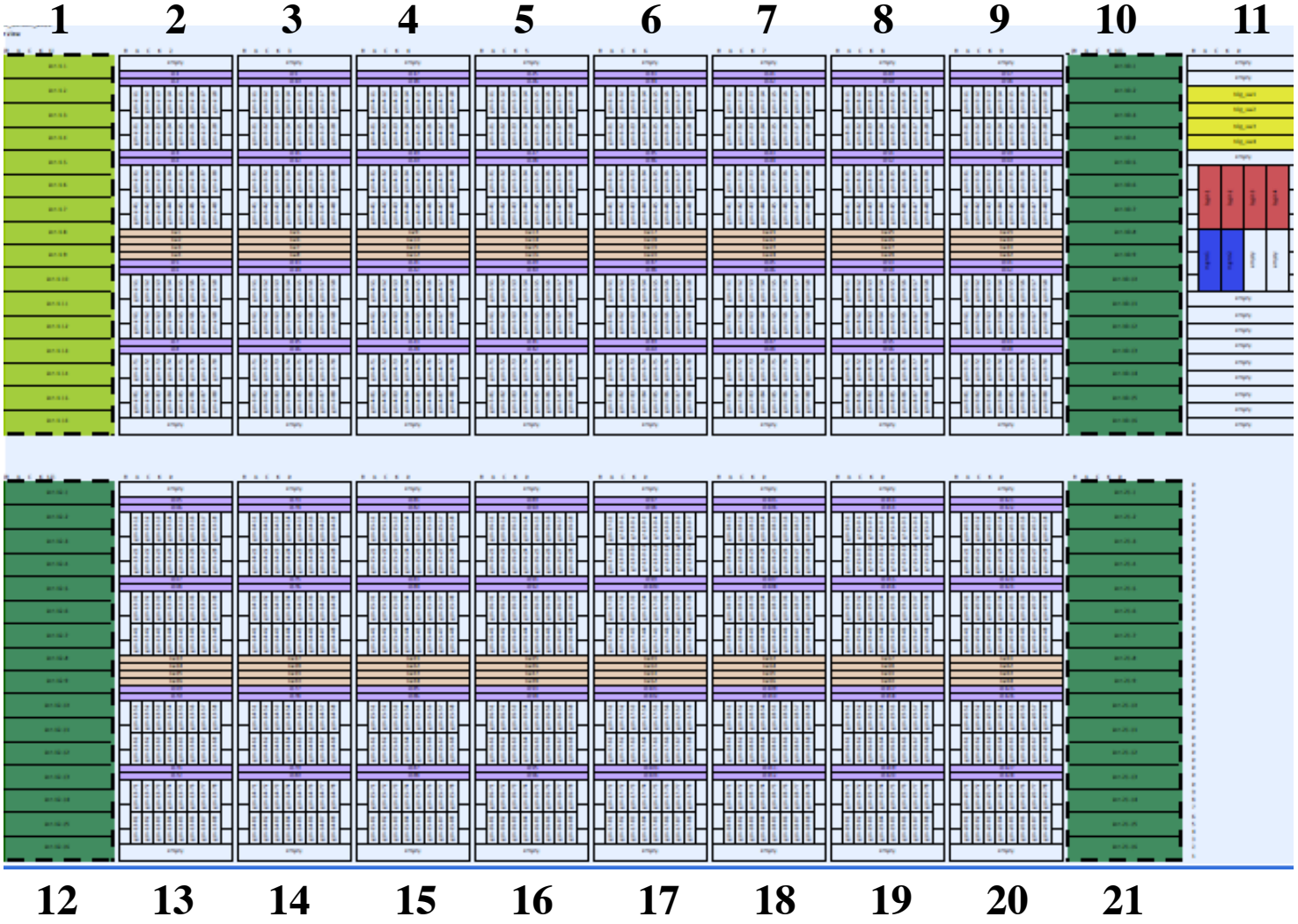
Front View

Rear View

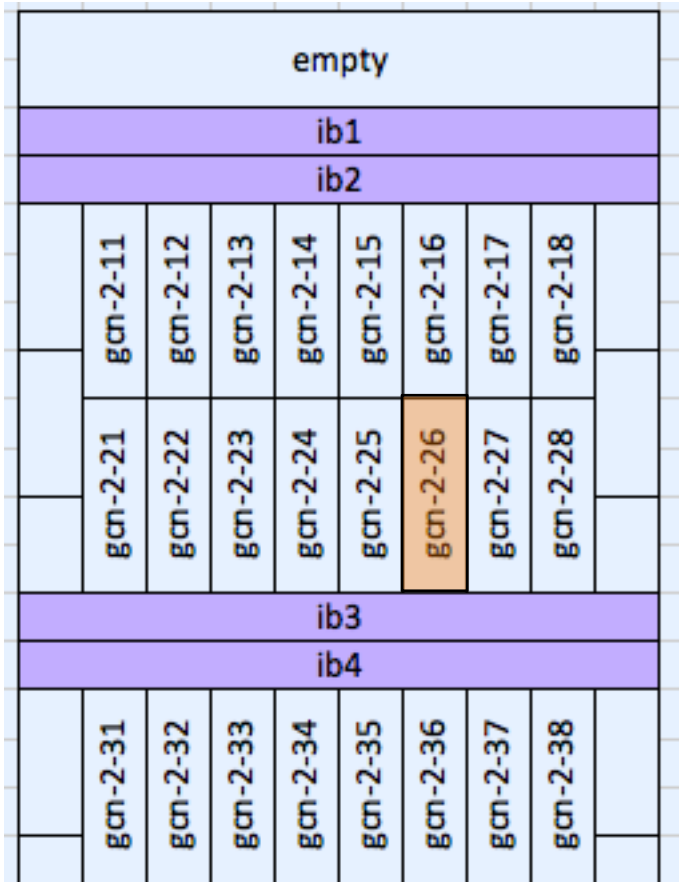


- 8RU Subrack
- Supports 16x 2P Intel Sandy-Bridge Blades
- Support for up to six high-efficiency 1625W hot-swappable PS in N+1 configuration
- Support for dual-redundant platform management modules
- Supports six hot-swappable, redundant fan modules
- Shared reduces power consumption by up to 20W per blade over previous design

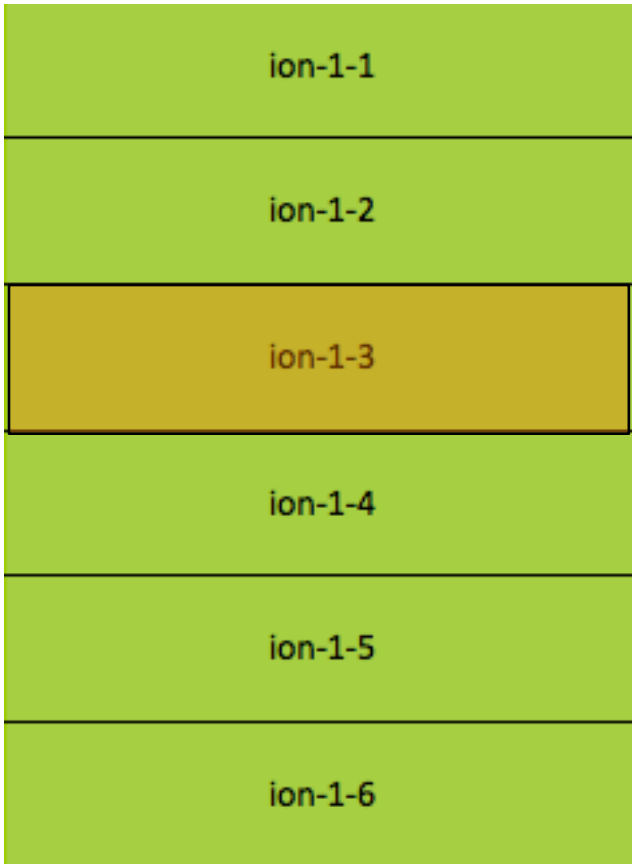
Gordon Rack Layout



Gordon naming conventions

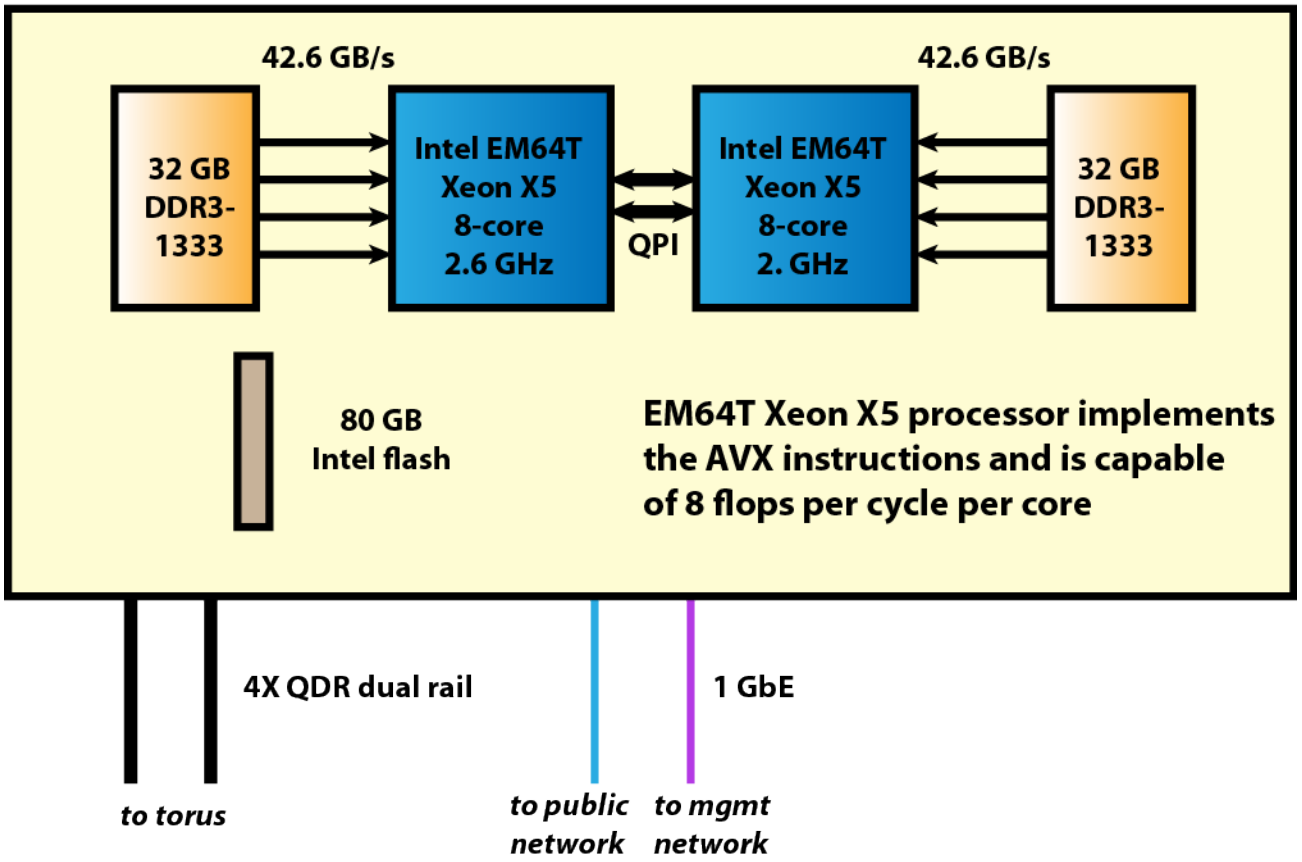


Gordon Compute Node
rack 2, row 2, node 6



I/O Node
rack 1, node 3

Gordon compute node



summary

64 GB DRAM

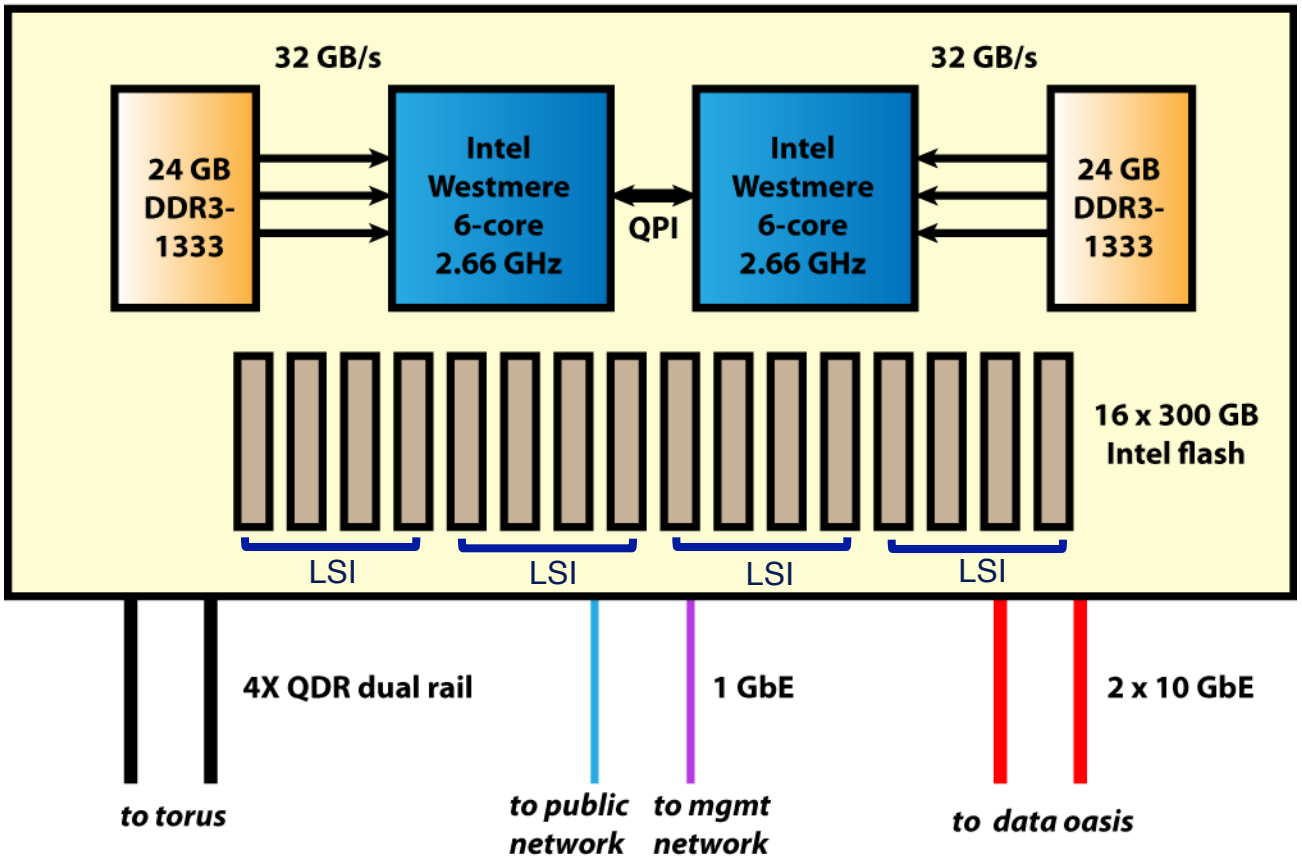
16 cores

2.6 GHz

80 GB flash

For more information on AVX, see http://software.intel.com/en-us/avx/

Gordon I/O node



summary

48 GB DRAM

12 cores

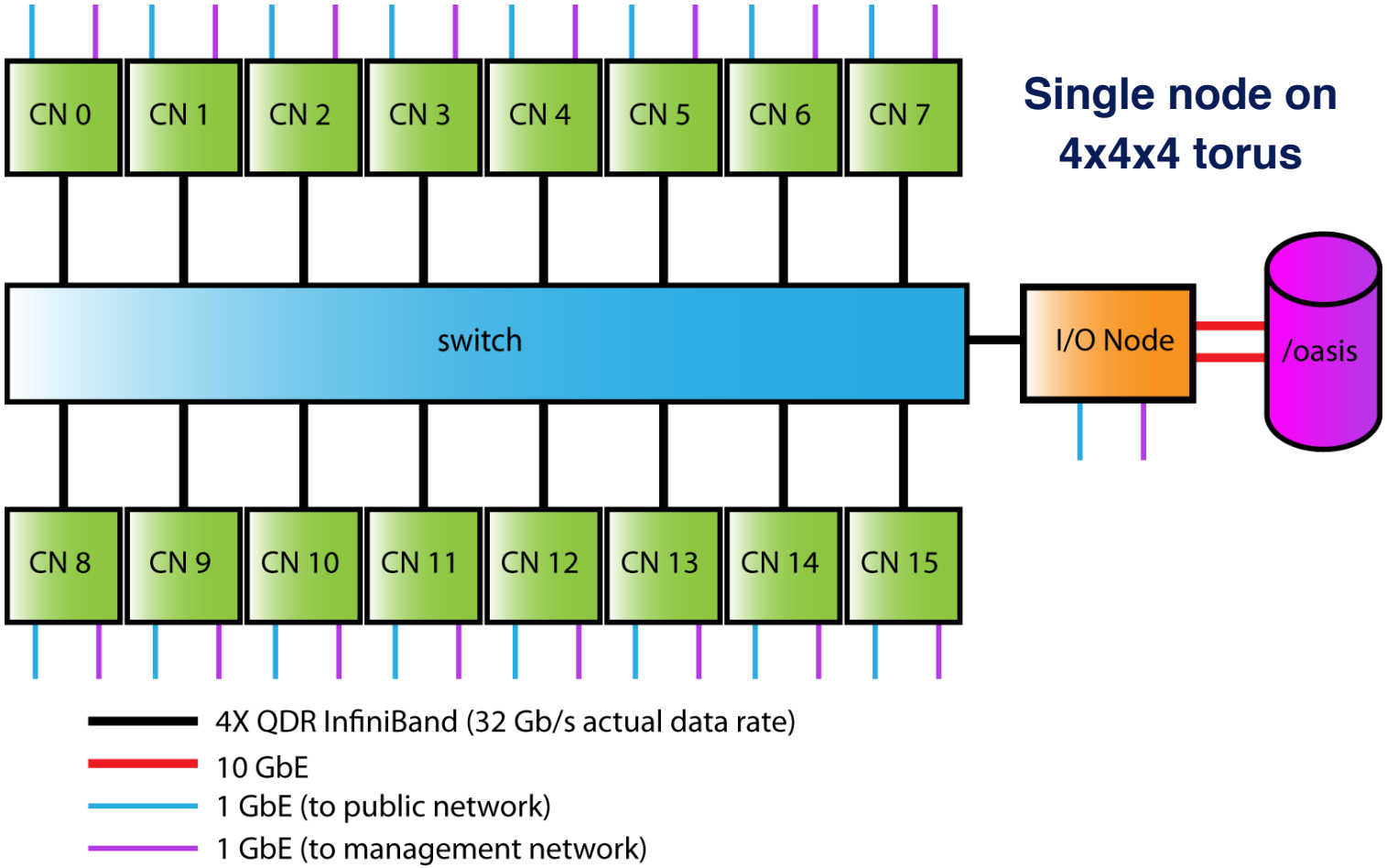
2.66 GHz

4.8 TB flash

Bonded into single channel

~ 1.6 GB/s bandwidth

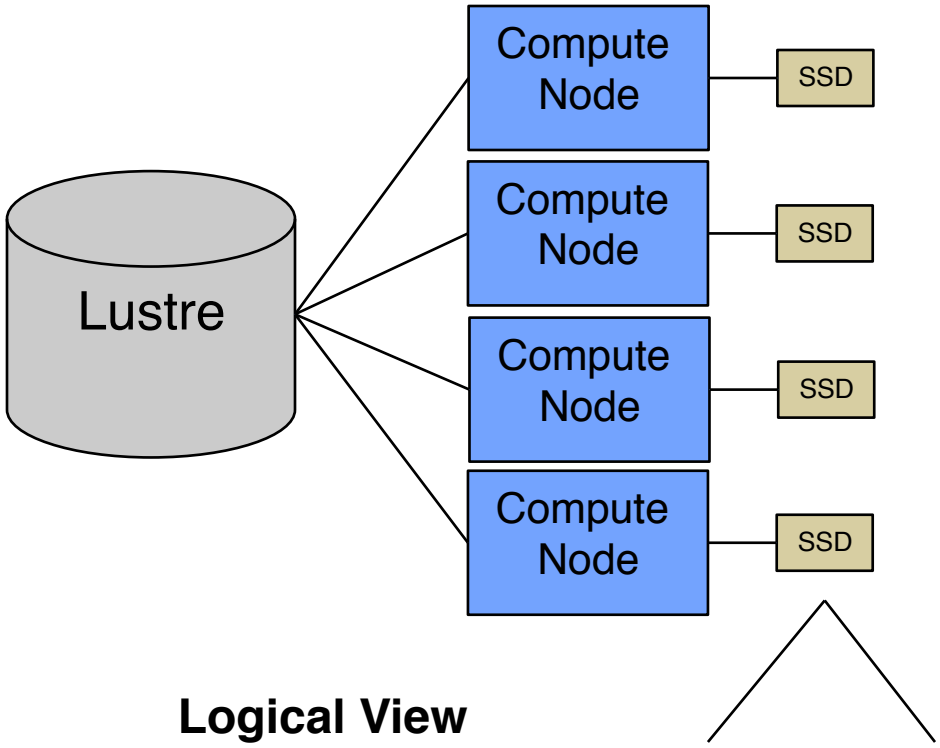
Simplified single rail view of Gordon connectivity showing routing between compute nodes on same switch, I/O node, and data oasis.



(Some) SSDs are a good fit for data-intensive computing

	Flash Drive	Typical HDD	Good for Data Intensive Apps
Latency	< .1 ms	10 ms	✔
Bandwidth (r/w)	270 / 210 MB/s	100-150 MB/s	✔
IOPS (r/w)	38,500 / 2000	100	✔
Power consumption (when doing r/w)	2-5 W	6-10 W	✔
Price/GB	$3/GB	$.50/GB	-
Endurance	2-10PB	N/A	✔
Total Cost of Ownership	Jury is still out.		

Exporting Flash
Model A: One SSD per Compute Node



Logical View

- One 300 GB flash drive exported to each compute node appears as a local file system
- Lustre parallel file system is mounted identically on all nodes.
- Data is purged at the end of the run

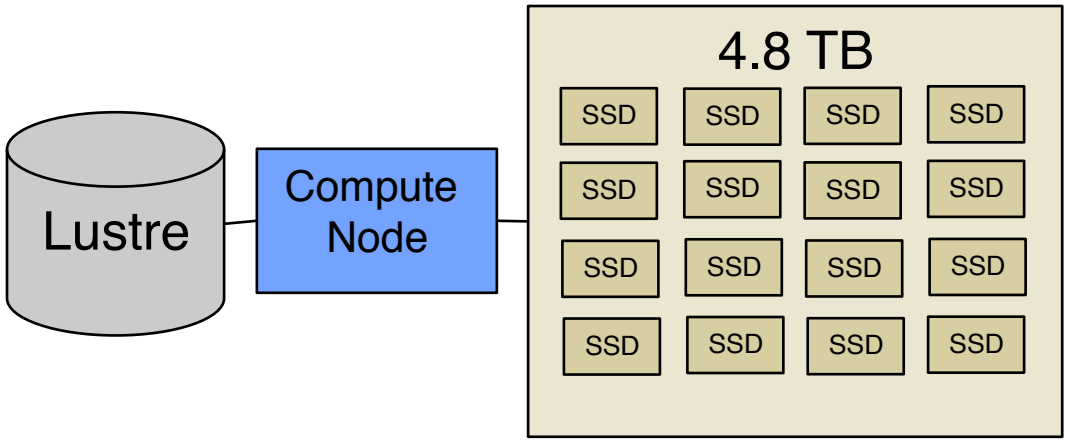
Use cases:

- Applications that need local, temporary scratch
- Gaussian
- Abaqus

This is what we'll be using for most of the summer school

File system appears as:
/scratch/$USER/$PBS_JOBID

Exporting Flash
Model B: 16 SSD's for 1 Compute Node



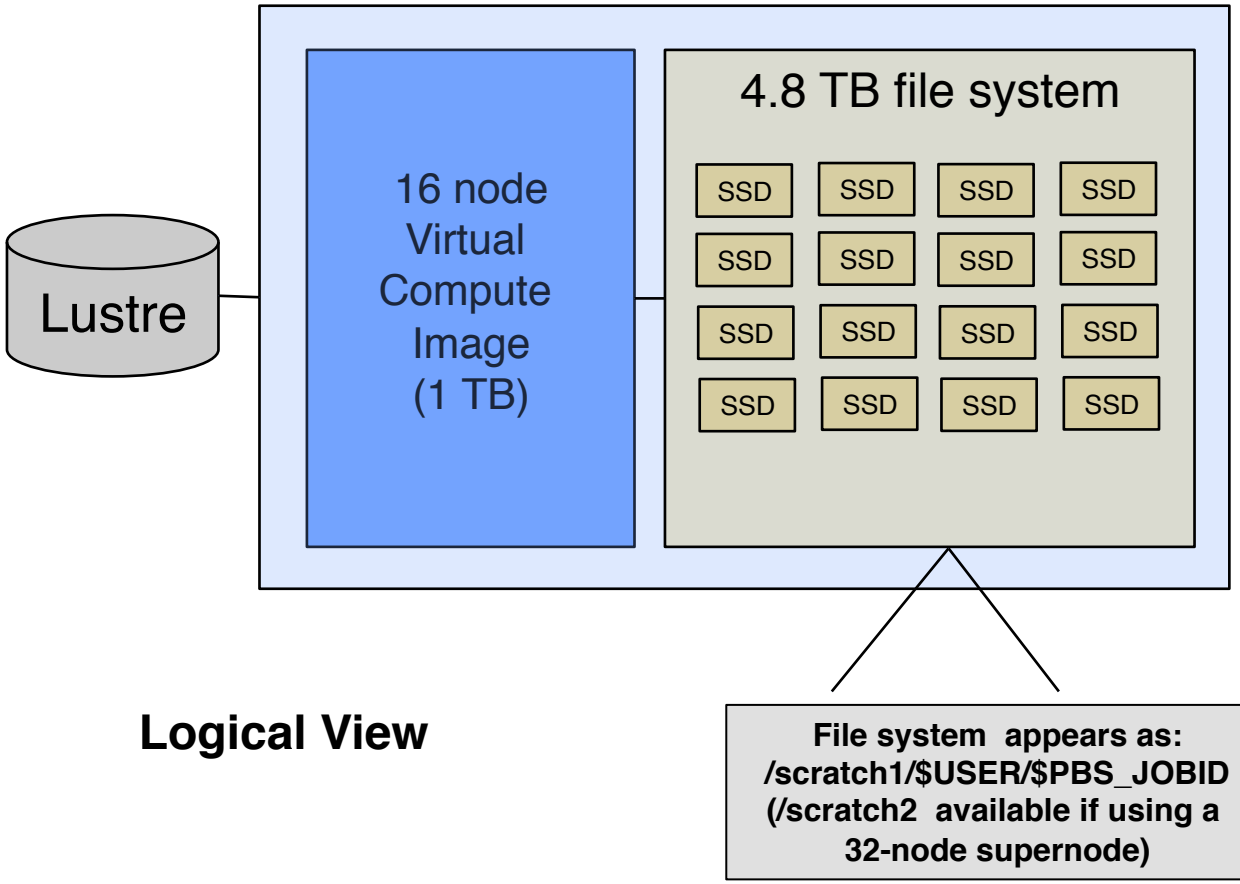
- 16 SSD's in a RAID0 appear as a single 4.8 TB file system to the compute node.
- Flash I/O and Lustre traffic uses Rail 1 of the torus.

Use cases:
- Database
- Data mining
- Gaussian
- Abaqus

Logical View

File system appears as:
/scratch/$USER/$PBS_JOBID

Exporting Flash
Model C: 16 SSD's within a vSMP Supernode



Logical View

File system appears as:
/scratch1/$USER/$PBS_JOBID
(/scratch2 available if using a 32-node supernode)

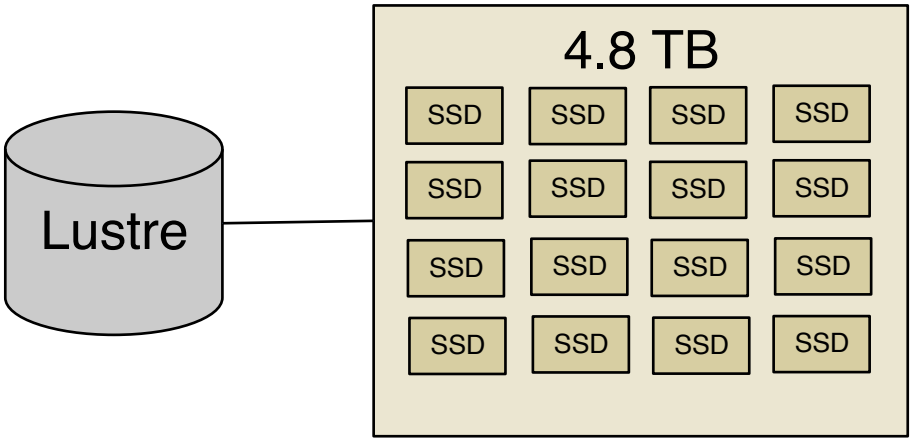
- 4.8 TB flash as a single XFS file system
- Flash I/O uses both rail 0 and rail 1
- Data purged at the end of a run

Use cases:

- Serial and threaded applications that need large memory and local disk
- Abaqus
- Genomics (Velvet, Allpaths, etc)

Exporting Flash

Model D: Dedicated I/O node
****A new allocations model for data intensive computing ****



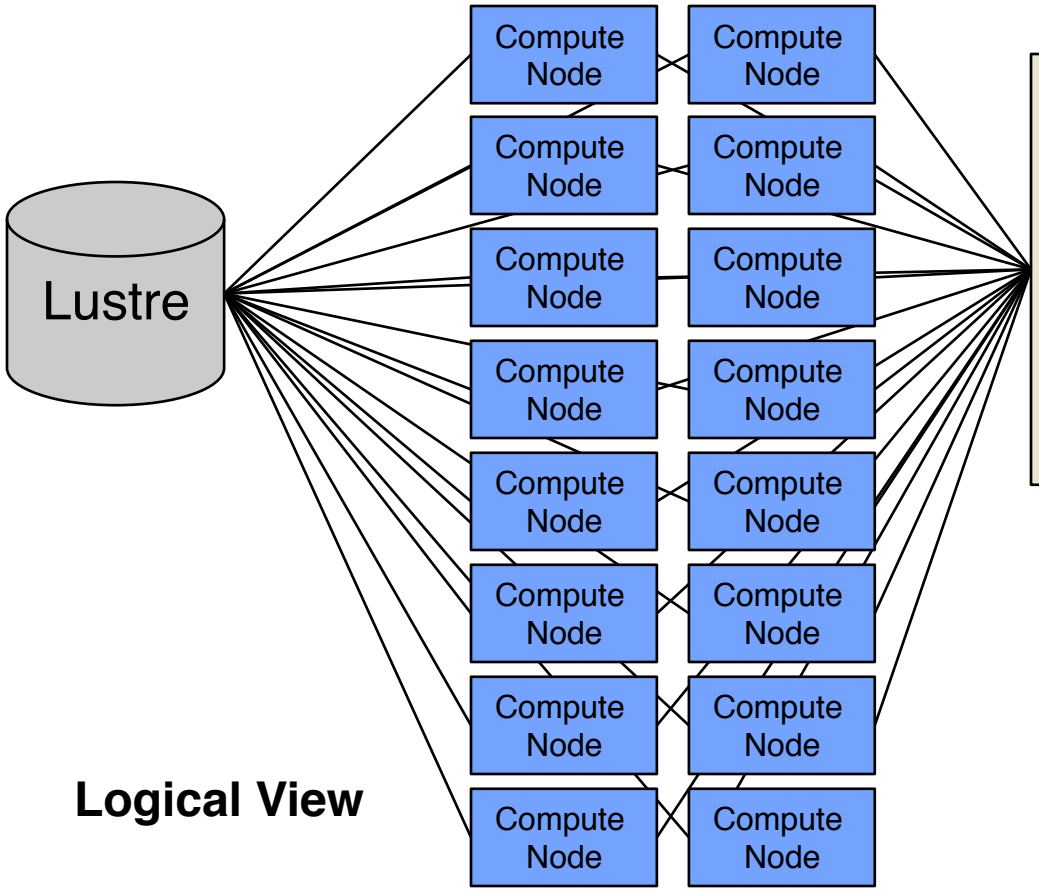
Logical View

- I/O node is allocated for up to one year
- Users run applications directly on the I/O node
- Users may request dedicated compute nodes or access them through the scheduler
- Data is persistent

Use cases:

- Database
- Data mining

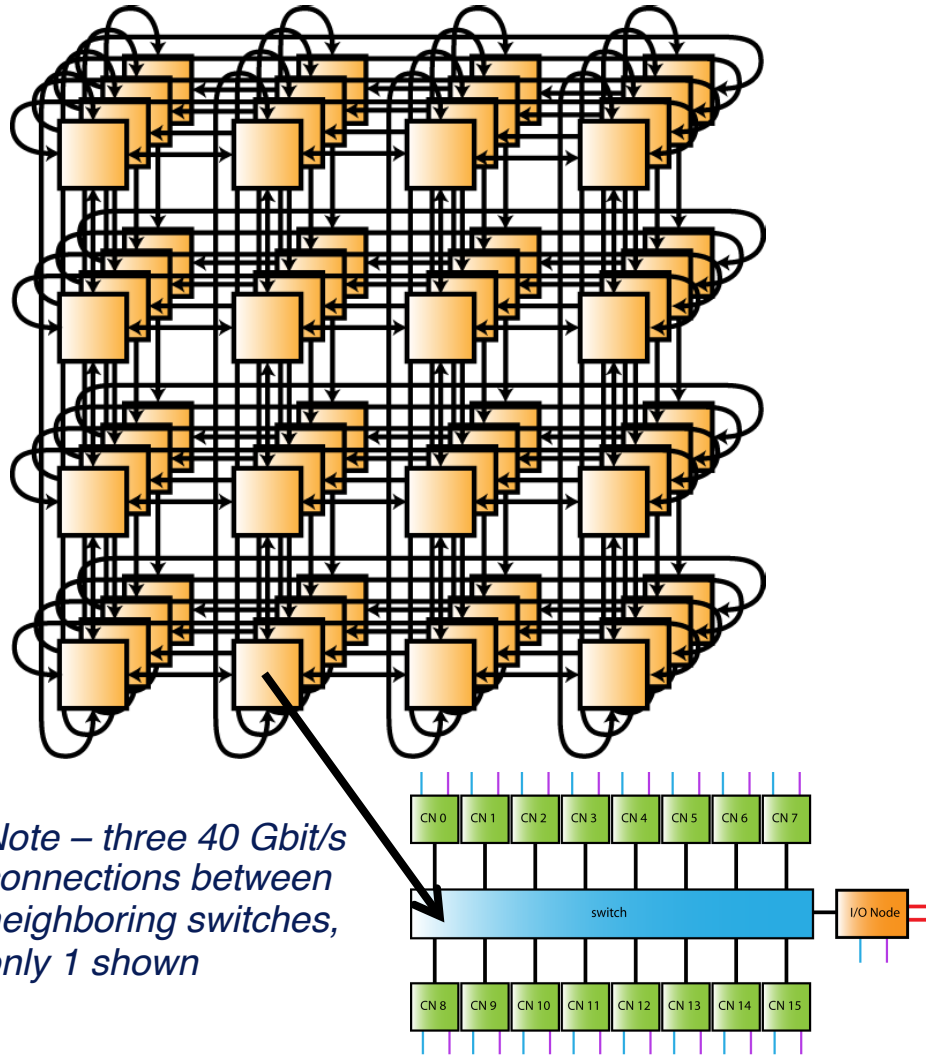
Exporting Flash
Model E: 16 SSD's/ 16 compute node –
Flash mounted as OCFS parallel file system
(tested but not currently deployed)



- 16 SSD's in a RAID0 appear as a single 4.8 TB file system to all compute nodes

Use cases:
- MPI applications

3D Torus Interconnect



Note – three 40 Gbit/s connections between neighboring switches, only 1 shown

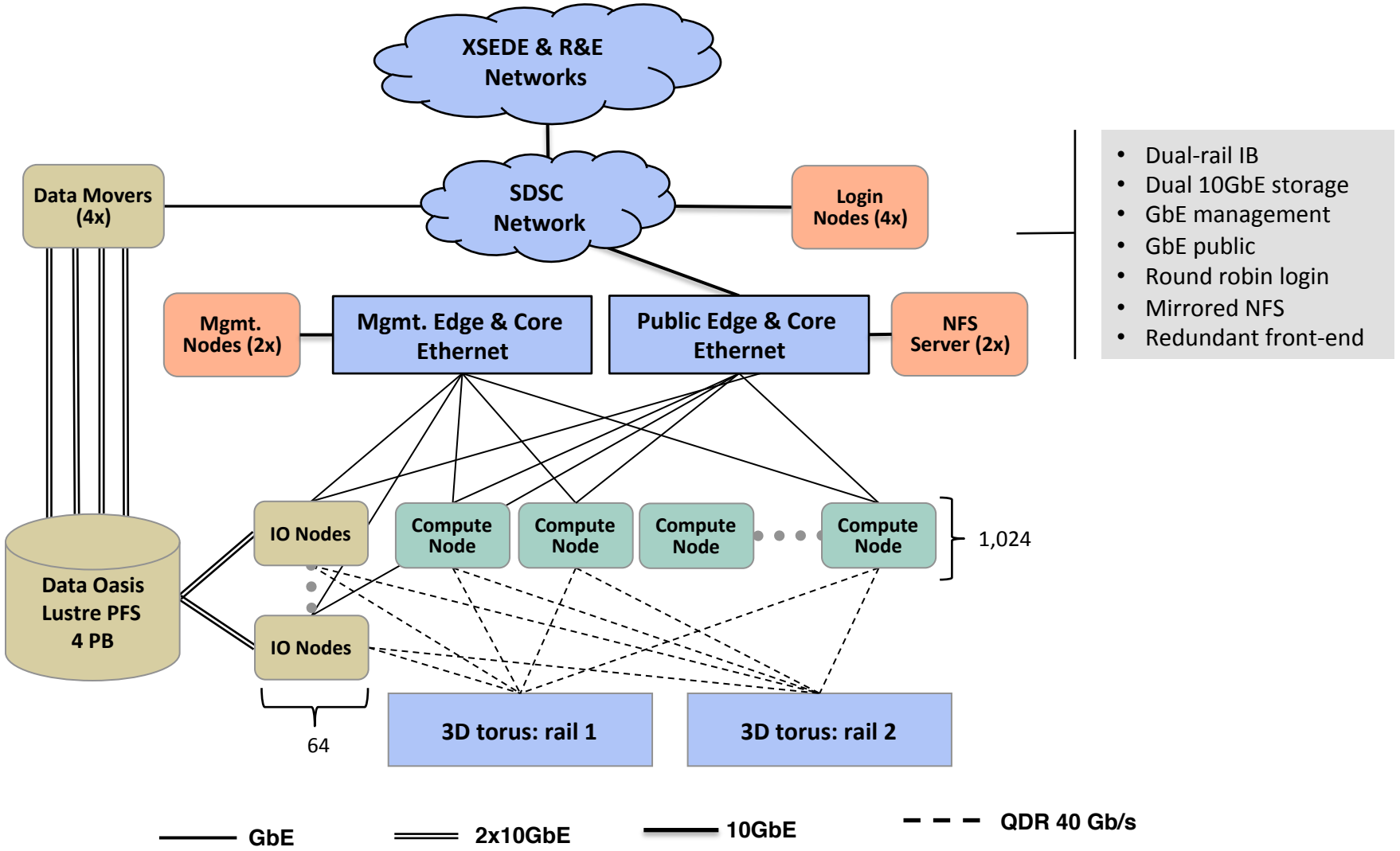
Gordon switches connected in dual rail 4x4x4 3D torus

Maximum of six hops to get from one node to furthest node in cluster

Fault tolerant, requires up to 40% fewer switches and 25-50% fewer cables than other topologies

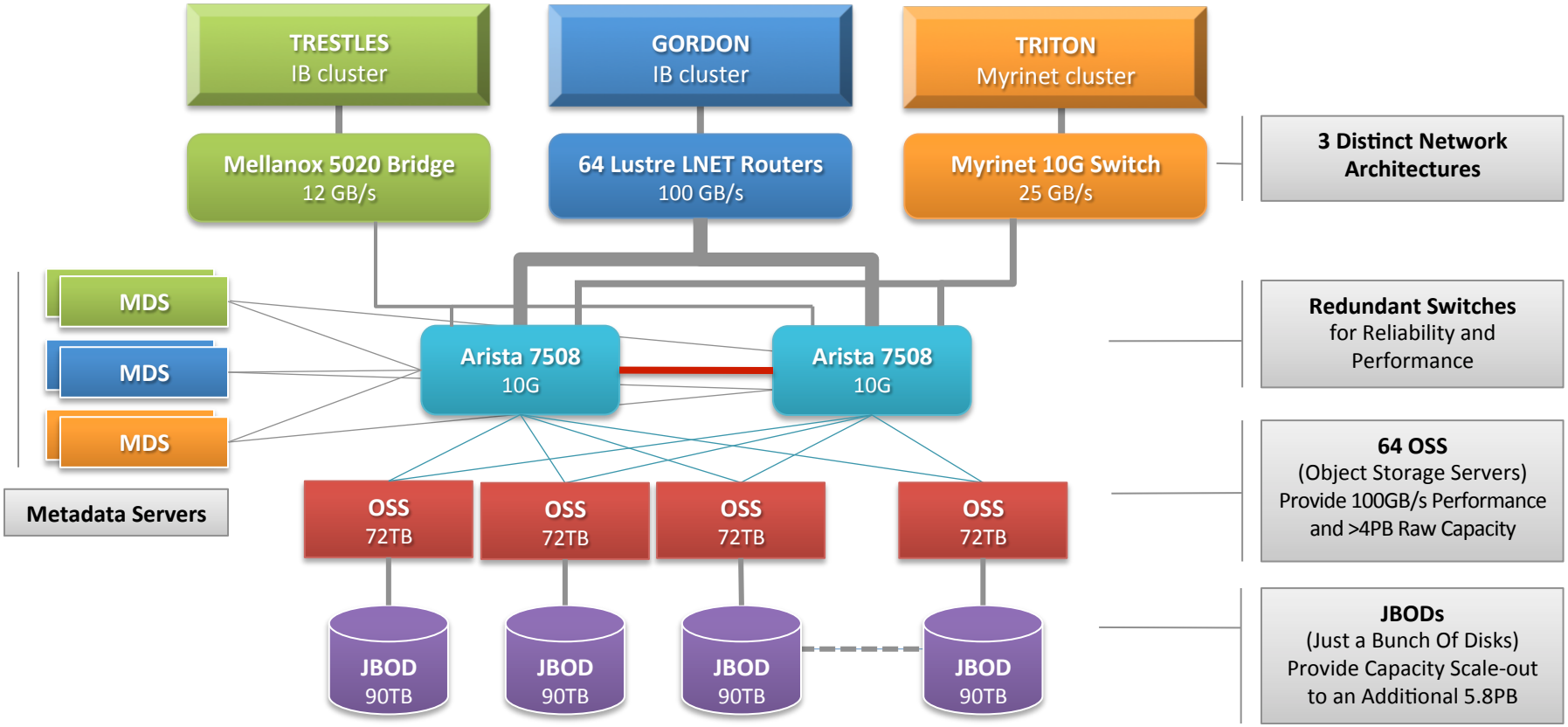
Scheduler will be aware of torus geometry and assign nodes to jobs accordingly

Gordon Network Architecture

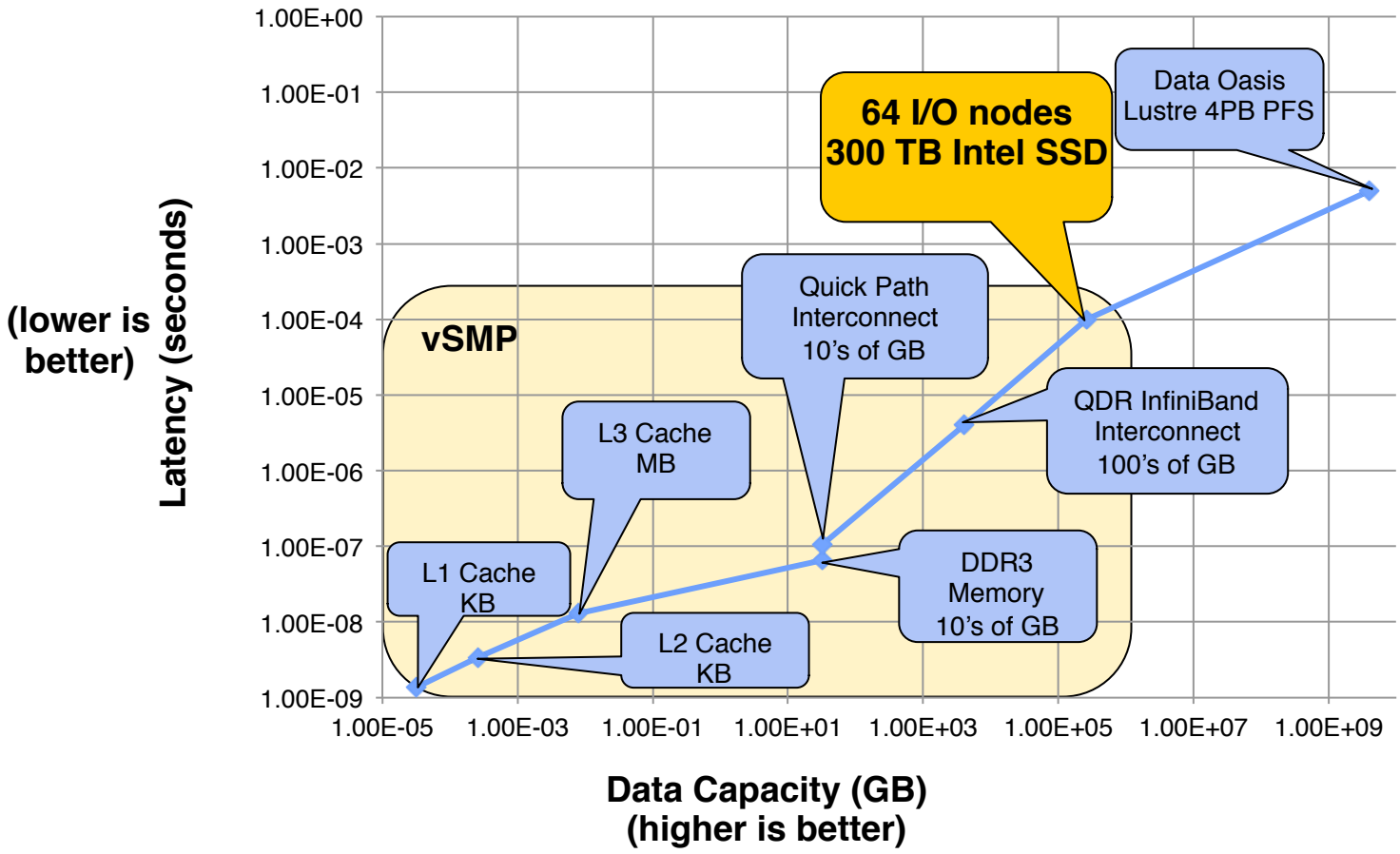


- Dual-rail IB
- Dual 10GbE storage
- GbE management
- GbE public
- Round robin login
- Mirrored NFS
- Redundant front-end

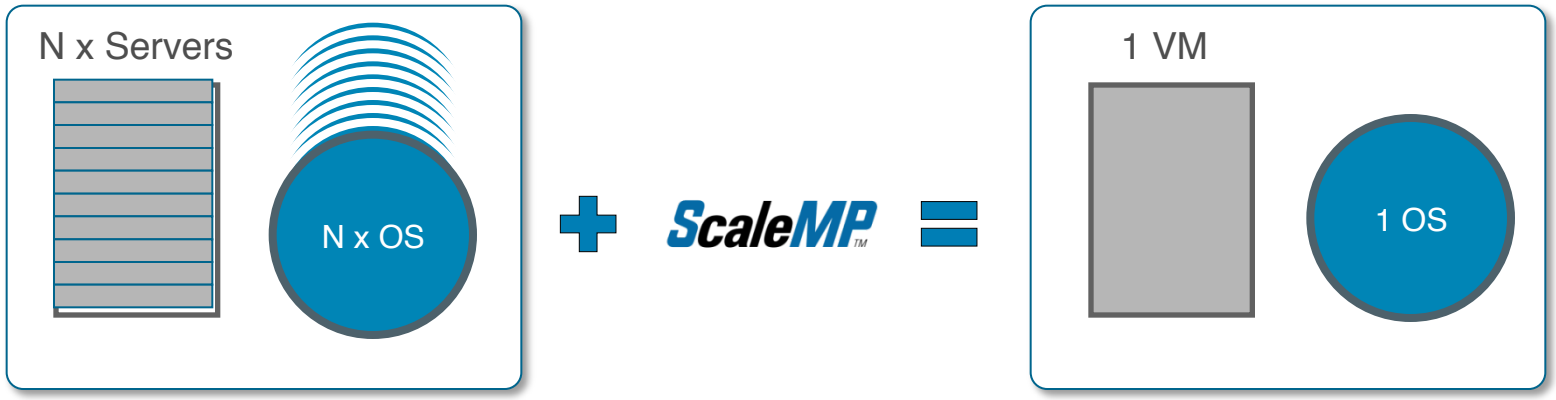
Data Oasis Heterogeneous Architecture
Lustre-based Parallel File System



vSMP and Flash Bridge the Latency & Capacity Gap

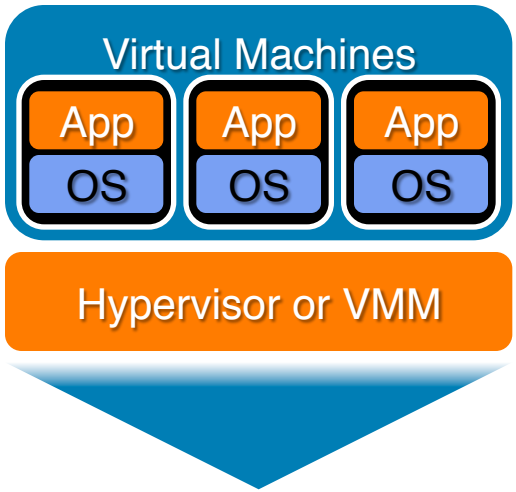


Introduction to vSMP



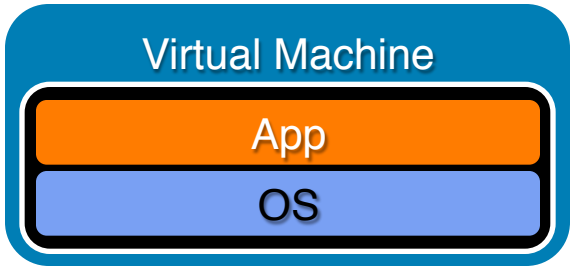
Virtualization software for aggregating multiple off-the-shelf systems into a single virtual machine, providing improved usability and higher performance

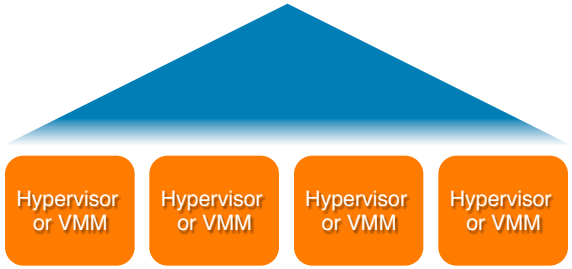
PARTITIONING



redhat vmware CITRIX QUMRANET Microsoft Xen Source™

AGGREGATION

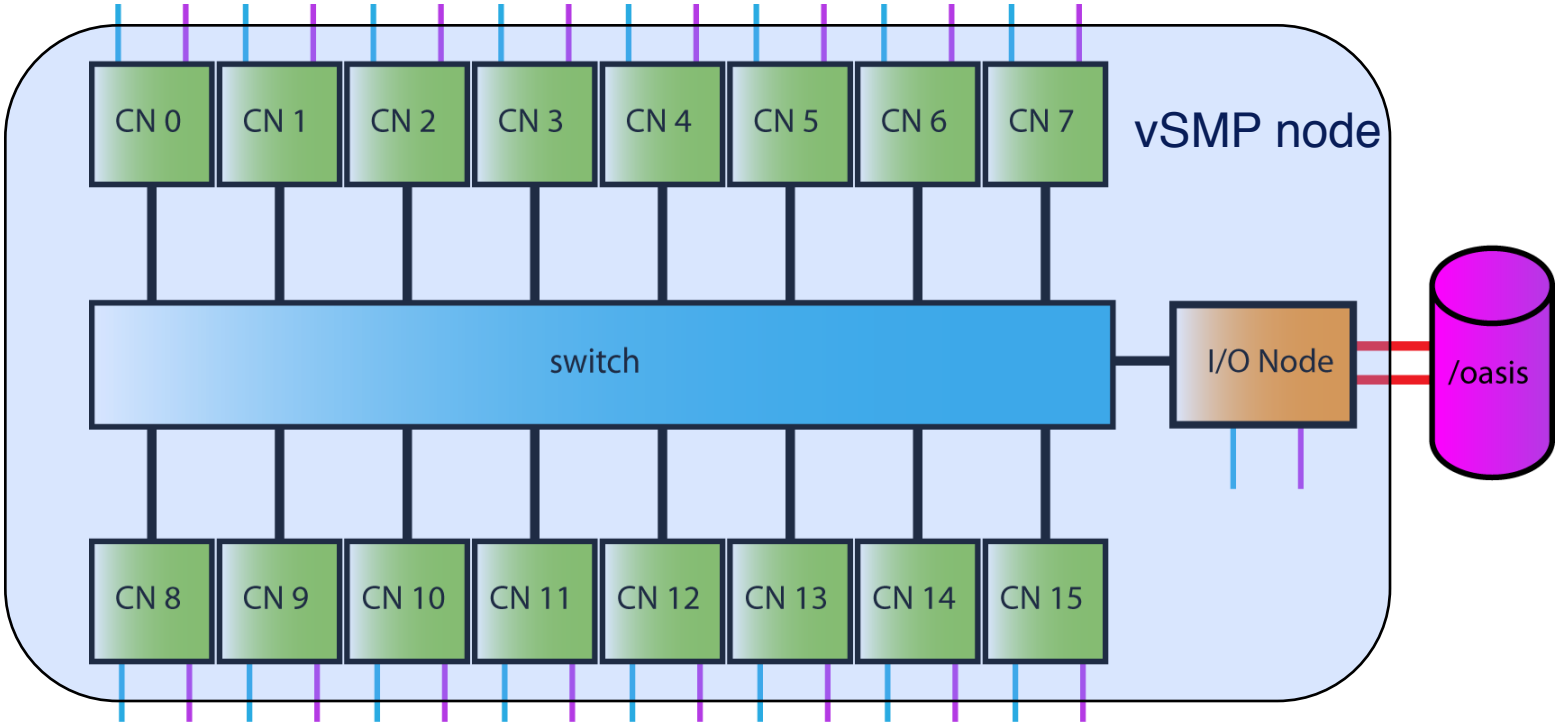




ScaleMP™

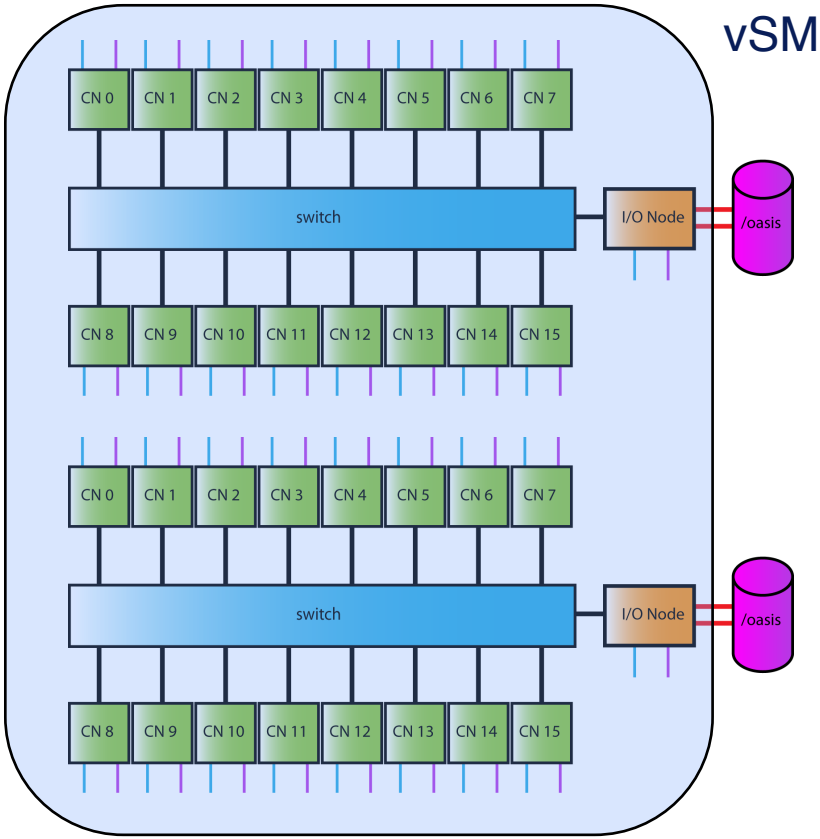
See Paikowsky presentation on Wed afternoon

vSMP node configured from 16 compute nodes and one I/O node



To user, logically appears as a single, large SMP node

vSMP node configured from 32 compute nodes and two I/O node



vSMP software provides flexibility to deploy logical shared memory nodes in sizes demanded by users.

Can potentially configure vSMP nodes on the fly (but not quite there yet)

To user, logically appears as a single, large SMP node with ~2 TB memory (32 x 64 GB) and 512 compute cores (32 x 16)

Overview of a vSMP node

```
[diag@gcn-17-51 ~]$ vsmpversion

vSMP Foundation: 4.0.220.0 (Apr 03 2012 19:05:33)
System configuration:
    Boards:      17
        16 x processor board
        1 x memory board
    Processors:  32 (out of 34), Cores: 256 (out of 268)
        32 x Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz Stepping 06
        2 x Intel(R) Xeon(R) CPU X5650 @ 2.67GHz Stepping 02
    Memory (MB): 957803 (out of 1096988), Cache: 100273, Private: 38912
        16 x 65491MB
        1 x 49132MB
    Host bridge:
        16 x VID/DID=8086/3c00
        1 x VID/DID=8086/3406
    Link Rate:   2 x 40Gb/s
    Boot device: [HDD] ATA INTEL SSDSA2CW08
Serial number:   0
System key:
Supported until:
```

vSMP node built from
16 compute nodes + 1 I/O nodes

Overview of a vSMP node

```
[diag@gcn-17-51 ~]$ grep processor /proc/cpuinfo | tail
processor       : 246
processor       : 247
processor       : 248
processor       : 249
processor       : 250
processor       : 251
processor       : 252
processor       : 253
processor       : 254
processor       : 255
[diag@gcn-17-51 ~]$ head /proc/meminfo
MemTotal:       967227852 kB
MemFree:        908057780 kB
Buffers:          170032 kB
Cached:         23341688 kB
SwapCached:            0 kB
Active:         44385452 kB
Inactive:       11914668 kB
Active(anon):   32792416 kB
Inactive(anon):     3464 kB
Active(file):   11593036 kB
[diag@gcn-17-51 ~]$
```

Logging in to Gordon

- Login to gordon.sdsc.edu
- You will be automatically directed to one of four identical login nodes gordon-ln[1-4].sdsc.edu
- Secure shell users should feel free to append their public RSA key to their ~/.ssh/authorized_keys file to enable access from authorized hosts without having to enter their password (already done for some of you)
- Login nodes have same architecture as compute nodes
- Do not use login nodes for computationally intensive jobs

Gordon file systems

Gordon provides four different file systems that differ by capacity, performance, backup policy and persistence

- Home
- Oasis project
- Oasis scratch
- Flash scratch

When I/O performance and storage capacity are important, the lecturers will tell you which file system to use

Home file system

- $HOME (/home/$USER)
- Persistent storage. Use for source code, small data sets,
- Backed up nightly
- Quotas not currently enforced, but keep usage below 100 GB
- NFS filesystem, mounted on multiple machines, shared by many users. Not meant for I/O intensive jobs

```
[sinkovit@gordon-ln3 ~]$ echo $USER
sinkovit

[sinkovit@gordon-ln3 ~]$ echo $HOME
/home/sinkovit
```

Data Oasis (Lustre) parallel file systems

- Scratch: /oasis/scratch/$USER/temp_project
- Project: /oasis/projects/nsf/[project]/$USER
- Semi-persistent (scratch) to persistent (projects)
- High-performance, parallel file systems, capable of delivering 1.6 GB/s per object storage server
- Default allocation of 500 GB, shared among all users of an XSEDE allocations award. Multiple TB upon request
- Ideal for large sequential reads/writes, large files
- Poor performance for random I/O, large numbers of files

```
[sinkovit@gordon-ln3 ~]$ ls -d /oasis/scratch/sinkovit/temp_project
/oasis/scratch/sinkovit/temp_project

[sinkovit@gordon-ln3 ~]$ ls -d /oasis/projects/nsf/sds136/sinkovit
/oasis/projects/nsf/sds136/sinkovit
```

Flash file systems

- /scratch/$USER/$PBS_JOBID
- Created automatically upon start of Torque job
- Persistent only for duration of job
- High bandwidth, excellent random I/O performance
- Size depends on flash export model, 280 GB minimum
- Ideal for scratch space, staging repeatedly used data
- Be sure to account for copy-in/copy-out time

```
[sinkovit@gcn-13-28 sinkovit]$ df -h /scratch
Filesystem            Size  Used Avail Use% Mounted on
/dev/sdb1             280G   33M  280G   1% /scratch

[sinkovit@gcn-13-28 sinkovit]$ ls -ld /scratch/$USER/$PBS_JOBID
drwx------ 2 sinkovit root /scratch/sinkovit/157431.gordon-fe2.local
```

Using modules

The modules package provides for dynamic management of your Linux environment. Unless there is a compelling reason for you to run your own versions of common software, we strongly encourage you to use modules to load and run the versions that have been installed by our staff.

- Built using optimal compiler options (e.g. AVX support)
- Linked to appropriate libraries (e.g. MVAPICH2_IB)
- Environment variables automatically set correctly

module list lists currently loaded modules

module load [mod] adds module to your environment

module unload [mod] removes module from your environment

```
[user@gordon-ln1 ~]$ module list
1) binutils/2.22  2) intel/2011  3) mvapich2_ib/1.8a1p1

[user@gordon-ln1 ~]$ module load amber mopac
[user@gordon-ln1 ~]$ module list
  1) binutils/2.22  3) mvapich2_ib/1.8a1p1  5) amber/11
  2) intel/2011     4) mopac/2009

[user@gordon-ln1 ~]$ module unload amber mopac
[user@gordon-ln1 ~]$ module list
  1) binutils/2.22  2) intel/2011  3) mvapich2_ib/1.8a1p1
```

module avail lists the modules that are available. Default is the version that will be loaded if no version number provided

```
[user@gordon-ln1 ~]$ module avail

------ /opt/modulefiles/mpi/.intel ------
mpich2_ib/1.0.7(default) openmpi_ib/1.4.1(default)
mvapich2_ib 1.8a1p1(default)

------ /opt/modulefiles/applications/.intel ------
atlas/3.9.45(default)    hdf5/1.8.3(default)      petsc/3.2.p3(default)
fftw/2.1.5               lapack/3.3.1(default)    scalapack/2.0.1(default)
[ --- additional lines not shown ---]

------ /opt/modulefiles/applications ------
R/2.13.1(default)         ddt/3.1(default)          mopac/2009(default)
amber/11(default)         fsa/1.15.2(default)       namd/2.6
[ --- additional lines not shown ---]

------ /opt/modulefiles/compilers ------
gnu/4.1.2(default)  gnu/4.6.1  intel/2011(default) pgi/11.9(default)
```

Unloading a module makes dependent modules unavailable. For example, note that MPI libraries and numerical libraries are no longer listed after unloading intel module

```
[user@gordon-ln1 ~]$ module unload intel
Unloading compiler-dependent module mvapich2_ib/1.8a1p1

[user@gordon-ln3 ~]$ module avail

------ /opt/modulefiles/applications ------
R/2.13.1(default)         ddt/3.1(default)          mopac/2009(default)
amber/11(default)         fsa/1.15.2(default)       namd/2.6
[ --- additional lines not shown ---]

------ /opt/modulefiles/compilers ------
gnu/4.1.2(default)  gnu/4.6.1  intel/2011(default) pgi/11.9(default)
```

module display [mod] gives information about a module

```
[user@gordon-ln1 ~]$ module display binutils
-------------------------------------------------------------------
/opt/modulefiles/applications/binutils/2.22:

prepend-path      PATH /opt/binutils/bin
prepend-path      LD_LIBRARY_PATH /opt/binutils/lib
prepend-path      LD_LIBRARY_PATH /opt/binutils/lib64
-------------------------------------------------------------------


[user@gordon-ln1 ~]$ module display gaussian
-------------------------------------------------------------------
/opt/modulefiles/applications/gaussian/09.C.01:

prepend-path      PATH /opt/gaussian/g09
setenv            GAUSS_EXEDIR /opt/gaussian/g09
setenv            G09_BASIS /opt/gaussian/g09/basis
-------------------------------------------------------------------
```

module swap [mod1] [mod2] replaces mod1 with mod2

module purge unloads all modules

```
[user@gordon-ln1 ~]$ module list
1) binutils/2.22    2) intel/2011       3) mvapich2_ib/1.8a1p1

[user@gordon-ln1 ~]$ module swap mvapich2_ib openmpi_ib
[user@gordon-ln1 ~]$ module list
1) binutils/2.22    2) intel/2011       3) openmpi_ib/1.4.1

[user@gordon-ln1 ~]$ module purge
[user@gordon-ln1 ~]$ module list
No Modulefiles Currently Loaded.-
```

One last word about modules …

The modules command is a shell function that is defined at login time and locate will not find it in the usual locations (e.g. /bin, /usr/bin, /sbin)

If you ever get the error "*module: command not found*", then run the following (you may need to add this to your shell scripts that execute module commands)

source /etc/profile.d/modules.sh

Gordon software

- Numerical libraries: ATLAS, FFTW, GSL, LAPACK, PETSc, SPRNG, ScaLAPACK, SuperLU, MKL
- Partitioning and frameworks: ParMETIS, Trilinios
- File formats/libraries: netCDF, HDF4, HDF5
- Molecular dynamics / chemistry: AMBER, GAMESS, Gaussian, LAMMPS, NAMD, Gromacs, MOPAC, NWChem
- Visualization: Visit, ParaView (coming soon)
- Languages: R, Octave
- Standard compilers: GNU, Intel, PGI
- MPI: MPICH2, MVAPICH2, OpenMPI

http://www.sdsc.edu/us/resources/gordon/gordon_software_packages.html

Gordon software suggestions

- **Preferred compiler**: Intel
  … but if you really need gnu for compatibility, load 4.6.1 or later to get AVX support

- **Preferred MPI**: MVAPICH2 (module mvapich2_ib)
  … but if you're building an MPI application for vSMP, then use ScaleMP's specially tuned version of MPICH2

- **Preferred numerical library**: MKL
  Tuned for best performance on Intel hardware; contains threaded BLAS, LAPACK, ScaLAPACK, FFTs, etc. Very easy to use, just link with -mkl
  … of course, don't even think about writing your own!

Compiling code for Gordon (Intel)

Intel compilers are the default on Gordon and generally give the best performance

- OpenMP enabled using -openmp
- Use -fast or -xHOST to get AVX support
  -fast = -xHOST -O3 -ipo -no-prec-div -static
- MKL library linked with -mkl

	Serial	MPI	OpenMP	Hybrid
Fortran	ifort	mpif90	ifort -openmp	mpif90 -openmp
C	icc	mpicc	icc -openmp	mpicc -openmp
C++	icpc	mpicxx	icpc -openmp	mpicxx -openmp

Compiling code for Gordon (PGI)

PGI compilers may give good performance on Gordon and also provide AVX support

- OpenMP enabled using -mp
- Use -fast to get AVX support

	Serial	MPI	OpenMP	Hybrid
Fortran	pgf90	mpif90	pgf90 -mp	mpif90 -mp
C	pgcc	mpicc	pgcc -mp	mpicc -mp
C++	pgCC	mpicxx	pgCC -mp	mpicxx -mp

Compiling code for Gordon (GNU)

GNU compilers are provided mainly for compatibility

- OpenMP enabled using -fopenmp
- Must use version 4.6 or later for AVX support
  module load gnu/4.6.1, then compile with -mavx

	Serial	MPI	OpenMP	Hybrid
Fortran	gfortran	mpif90	gfortran -fopenmp	mpif90 -fopenmp
C	gcc	mpicc	gcc -fopenmp	mpicc -fopenmp
C++	g++	mpicxx	g++ -fopenmp	mpicxx -fopenmp

A few comments about compilers

- Always specify an optimization level. In some cases the default is -O0 and performance will be terrible
- Note that the OpenMP options are similar, but vary
  Intel (-openmp), PGI (-mp), GNU (-fopenmp)
- mpicc, mpif90, and mpicxx are wrappers, not compilers
  They call the appropriate serial compilers under the hood
- For production runs, turn off all features that may impact performance (e.g. debug, bounds checking, profiling)

Torque

Torque is a freely available, open-source* distributed resource manager based on the PBS project (from an end user point of view Torque is virtually identical to PBS)

- **qsub** submits jobs to the queue
- **qstat** gets job and queue information
- **qdel** deletes a queued or running job
- **qalter** modifies job attributes
  (limited capabilities for regular users)

See www.clusterresources.com/torquedocs
for more than you'll ever want to know about Torque

qsub submits a job to the queue. Torque options can be placed into a job file or specified on command line

```
[user2@gordon-ln1 ~]$ qsub job_script
[user2@gordon-ln1 ~]$ cat job_script

#!/bin/bash                      <-- File is a bash script
#PBS -q normal                   <-- Running in normal queue
#PBS -N jobname                  <-- Name of job as listed by qstat
#PBS -l nodes=1:ppn=16:native    <-- One node, 16 cores/node, native property
#PBS -l walltime=0:10:00         <-- 10 minutes wall time
#PBS -o jobname.out              <-- stdout file
#PBS -e jobname.err              <-- stderr file
#PBS -v Catalina_res_bind=xxx    <-- Reservation number
#PBS -V                          <-- Export environment variables
#PBS -M username@sdsc.edu        <-- Address for notification
#PBS -m abe                      <-- Notify on abort, begin and end
#PBS –A sds136                   <-- Account to be charged

# run jobs, move files, general bash commands, etc.
mpirun_rsh -hostfile $PBS_NODEFILE –np 16 ./hello_world
```

qsub can also be used to get an interactive node

```
[user2@gordon-ln1 ~]$ qsub -I -X -l nodes=1:ppn=16:native,walltime=10:00:00
-q normal -v Catalina_res_bind=xxx,QOS=0 -A sds136

-I                                <-- Interactive job
-X                                <-- X11 forwarding
-l nodes=1:ppn=16:native,         <-- 1 node, 16 cores/node, native property
   walltime=10:00:00                  10 hour wall clock
-q normal                         <-- Normal queue
–v Catalina_res_bind=xxx,QOS=0    <-- Reservation number
-A sds136                         <-- Account to be charged


[user2@gordon-ln1 ~]$ cat .bash_profile
alias ggn='qsub -I -X -l nodes=1:ppn=16:native,walltime=10:00:00 -q normal
-v Catalina_res_bind=xxx,QOS=0 -A sds136'
```

What happens after you are assigned a node

Users are normally prohibited from directly accessing a node, but an exception is granted after it has been assigned by Torque.

This can be particularly useful if you ever need to monitor a non-interactive job (e.g. using top to determine number of processes, threads, CPU utilization, memory footprint)

```
# Before assignment of node
[sinkovit@gordon-ln4 ~]$ ssh gcn-15-73 cat /etc/security/access.conf
Connection closed by 10.5.102.210

# After assignment of node
[sinkovit@gordon-ln4 ~]$ ssh -q gcn-15-73 cat /etc/security/access.conf
-:ALL EXCEPT root diag sinkovit:ALL
```

qstat gets job and queue information

```
[user2@gordon-ln1 ~]$ qstat -a
                                                  Req'd  Req'd   Elap
Job ID   Username Queue   Jobname NDS  Memory Time S Time
-------- -------- ------  ------- --- ------ ----- - -----
1.gordon user1    normal  g09       1    1gb 26:00 R 23:07
2.gordon user2    normal  task1    32    --  48:00 R 32:15
3.gordon user2    normal  task2     2    1gb 24:00 H   --
4.gordon user3    normal  exp17     1    --  12:00 C 01:32
5.gordon user3    normal  stats1    8    --  12:00 Q   --
6.gordon user3    normal  stats2    8    --  12:00 E 15:27

[user2@gordon-ln1 ~]$ qstat -a -u user2
                                                  Req'd  Req'd   Elap
Job ID   Username Queue   Jobname NDS  Memory Time S Time
-------- -------- ------  ------- --- ------ ----- - -----
2.gordon user2    normal  task1    32    --  48:00 R 32:15
3.gordon user2    normal  task2     2    1gb 24:00 H   --
```

qstat can also provide detailed information about a job

```
[user2@gordon-ln1 ~]$ qstat -f 57016
Job Id: 57016.gordon-fe2.local
    Job_Name = mg6
    Job_Owner = user@gordon-ln2.local
    resources_used.cput = 11:44:57
    resources_used.mem = 266012kb
    [ -- additional lines not shown -- ]

[sinkovit@gordon-ln3 ~]$ qstat -n 57016
                                                  Req'd  Req'd   Elap
Job ID   Username Queue   Jobname NDS  Memory Time S Time
-------- -------- ------  ------- --- ------ ----- - -----
57016    user     normal   mg6     1    1gb    3:00 R  1:57
 gcn-15-56/15+gcn-15-56/14+gcn-15-56/13+gcn-15-56/12+gcn-15-56/11
 +gcn-15-56/10+gcn-15-56/9+gcn-15-56/8+gcn-15-56/7+gcn-15-56/6
 +gcn-15-56/5+gcn-15-56/4+gcn-15-56/3+gcn-15-56/2+gcn-15-56/1+gcn-15-56/0
```

qdel deletes a queued, held, or running job

```
[user2@gordon-ln1 ~]$ qstat -a -u user2
                                                  Req'd  Req'd  Elap
Job ID   Username Queue   Jobname NDS  Memory Time S Time
-------- -------- ------  ------- --- ------ ----- - -----
2.gordon user2    normal  task1    32    --  48:00 R 32:15
3.gordon user2    normal  task2     2    1gb 24:00 H   --

[user2@gordon-ln1 ~]$ qdel 2

[user2@gordon-ln1 ~]$ qstat -a -u user2
                                                  Req'd  Req'd  Elap
Job ID   Username Queue   Jobname NDS  Memory Time S Time
-------- -------- ------  ------- --- ------ ----- - -----
3.gordon user2    normal  task2     2    1gb 24:00 H   --
```

qalter modifies job attributes. Regular users have limited capabilities (e.g. reduce wall time for a running job)

```
[user4@gordon-ln1 ~]$ qstat -a 8
                                                  Req'd  Req'd   Elap
Job ID   Username Queue   Jobname NDS  Memory Time S Time
-------- -------- ------  ------- --- ------ ----- - -----
8.gordon user4    normal  task1    32    --  10:00 R  6:15

[user4@gordon-ln1 ~]$ qalter -l walltime=9:00 8

[user4@gordon-ln1 ~]$ qstat -a 8
                                                  Req'd  Req'd   Elap
Job ID   Username Queue   Jobname NDS  Memory Time S Time
-------- -------- ------  ------- --- ------ ----- - -----
8.gordon user2    normal  task1    32    --  09:00 R  6:15
```

http://hipacc.ucsc.edu/ISSAC2012_Program.html - Sinkovitz

- Login to gordon (ssh –X username@gordon.sdsc.edu)
- Verify that $HOME and /oasis/scratch/$USER/temp_project exist
- Add **contents** of ~sinkovit/ASTRO/add_to_profile to your .bash_profile
  cat ~sinkovit/ASTRO/add_to_profile >> .bash_profile
  (be sure to 'source .bash_profile')
- Launch an interactive job (gastro alias from add_to_profile)
- Verify that X-forwarding works (e.g. 'xclock &')
- While interactive job is running, connect directly to your compute node (gcn-x-xx) from login node
- Verify that /scratch/$USER/$PBS_JOBID exists
- Kill interactive job (CTRL-D or exit)
- Experiment with module commands (load, unload, avail, display)
- Try the qstat command – list all jobs (-a), pick a random job and get detailed information (-f) or node list (-n)
- Copy ~sinkovit/ASTRO/HELLO to your home directory. Follow directions to build executables and launch batch jobs
- Check out Gordon user guide http://www.sdsc.edu/us/resources/gordon/