

# Running jobs on Expanse

## March 4, 2021

Mahidhar Tatineni  
SDSC

EXPANSE  
COMPUTING WITHOUT BOUNDARIES

SAN DIEGO SUPERCOMPUTER CENTER



NSF Award 1928224

# Outline

- System overview and login
- Slurm scheduler – partition information
- Running jobs
  - MPI
  - OpenMP
  - MPI/OpenMP hybrid
  - GPU jobs
- Filesystems
  - Example local scratch usage
- Summary

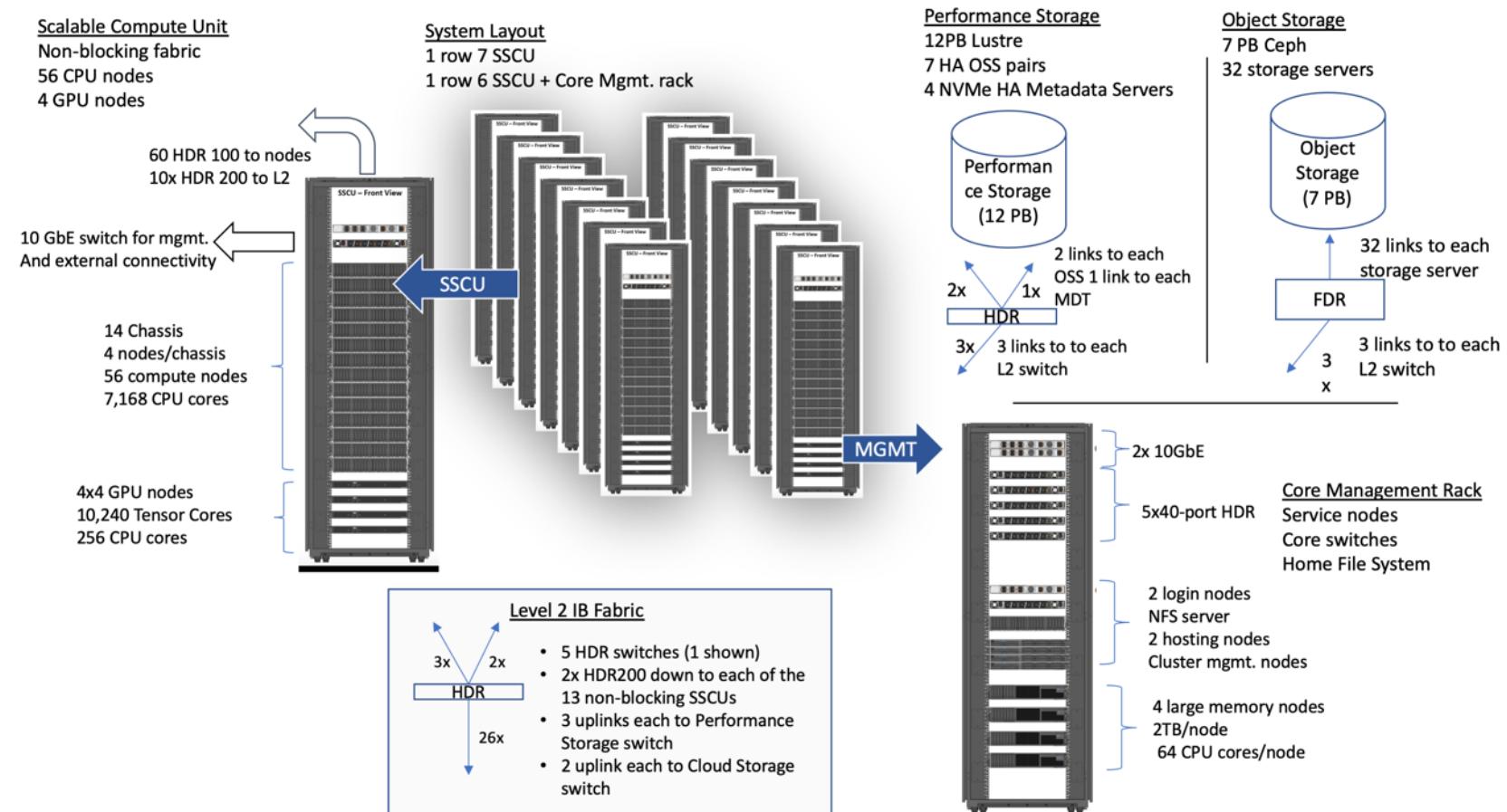
# Outline

- System overview and login
- Slurm scheduler – partition information
- Running jobs
  - MPI
  - OpenMP
  - MPI/OpenMP hybrid
  - GPU jobs
- Filesystems
  - Example local scratch usage
- Summary

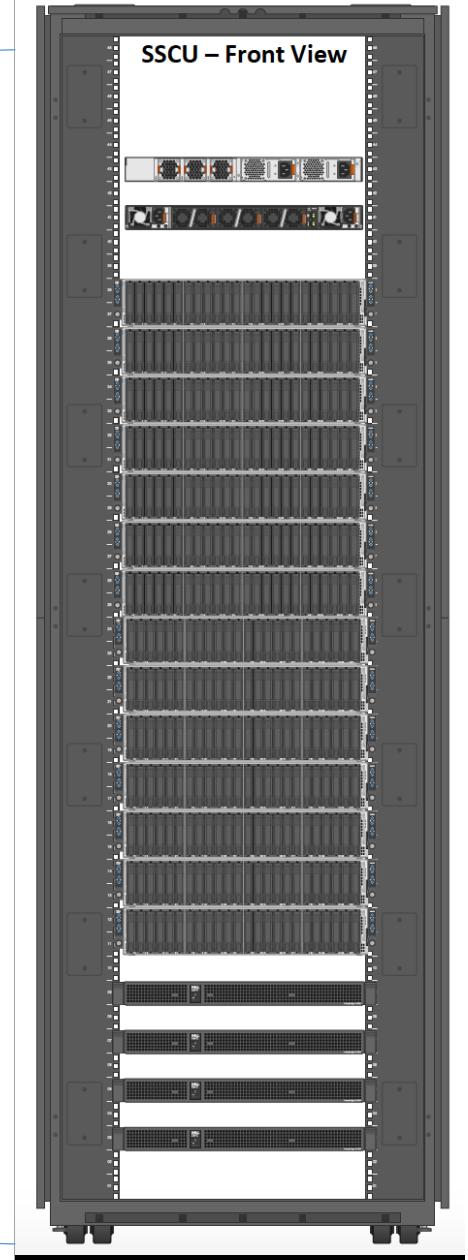
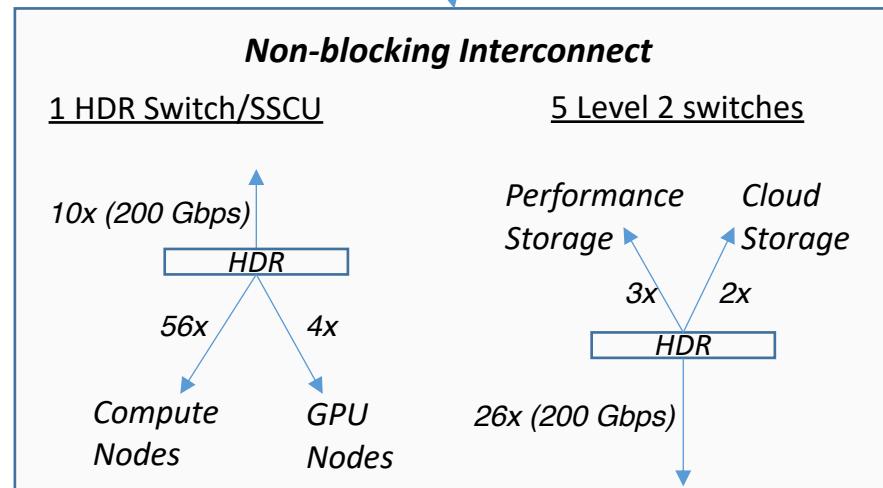
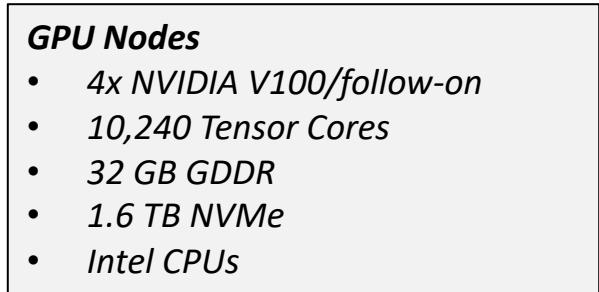
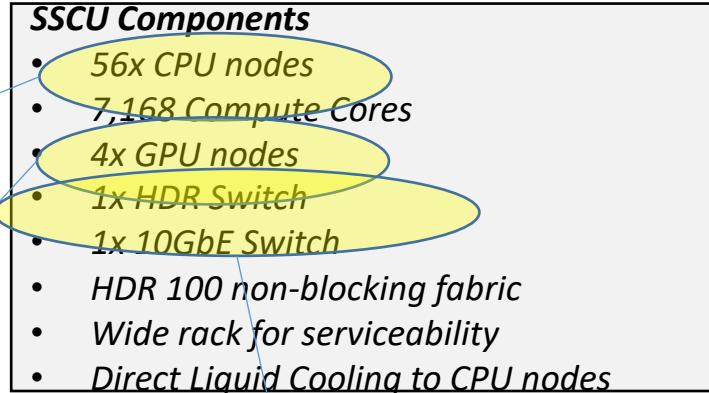
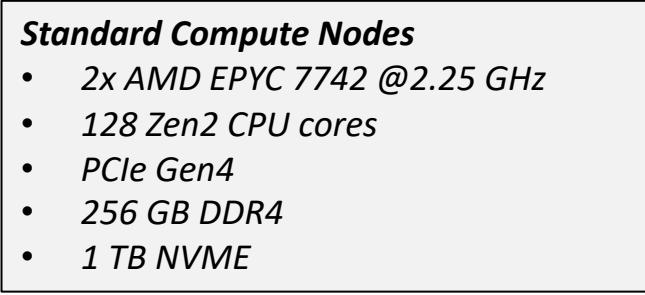
# Expanse is a heterogeneous architecture designed for high performance, reliability, flexibility, and productivity

## System Summary

- 13 SDSC Scalable Compute Units (SSCU)
- 728 x 2s Standard Compute Nodes
- 93,184 Compute Cores
- 200 TB DDR4 Memory
- 52x 4-way GPU Nodes w/NVLINK
- 208 V100s
- 4x 2TB Large Memory Nodes
- HDR 100 non-blocking Fabric
- 12 PB Lustre High Performance Storage
- 7 PB Ceph Object Storage
- 1.2 PB on-node NVMe
- Dell EMC PowerEdge
- Direct Liquid Cooled



# The SSCU is Designed for the Long Tail Job Mix, Maximum Performance, Efficient Systems Support, and Efficient Power and Cooling



# Logging into Expanse

- CPU and GPU resources are allocated separately, the login nodes are the same. To log in to Expanse from the command line, use the hostname:  
**login.expanse.sdsc.edu**
- Users can login directly using the Secure shell (SSH) command or a ssh client (such as putty). Example:

```
ssh your_username@login.expanse.sdsc.edu  
ssh -l your_username login.expanse.sdsc.edu
```

- When you log in to login.expanse.sdsc.edu, you will be assigned one of the two login nodes login0[1-2]-expanse.sdsc.edu. Both nodes are identical.
- Users can also login via the XSEDE single sign on host (login.xsede.org):

```
ssh your_xsede_portal_username@login.xsede.org  
gsissh login.expanse.sdsc.edu
```

- ***Access via Expanse User Portal (See Subha's talk later today).***

# Using SSH Keys

- You can append your public key (e.g. from your laptop) to your `~/.ssh/authorized_keys` file to enable access from authorized hosts without having to enter your password.
- RSA, ECDSA and ed25519 keys are accepted.
- Make sure you have a strong passphrase on the private key on your local machine.
- You can use ssh-agent or keychain to avoid repeatedly typing the private key password.
- Hosts which connect to SSH more frequently than ten times per minute may get blocked for a short period of time
- Two factor authentication with Google Authenticator is available as an option. Set up details available on Expanse website ([https://www.sdsc.edu/support/user\\_guides/expanse.html#access](https://www.sdsc.edu/support/user_guides/expanse.html#access)).
- See SDSC Security Webinar:
  - [https://www.sdsc.edu/event\\_items/202007\\_CometWebinar.html](https://www.sdsc.edu/event_items/202007_CometWebinar.html)

```
ssh <your_username>@login.expanse.sdsc.edu  
ssh -l <your_username> login.expanse.sdsc.edu
```

# Appropriate use of Login Nodes

- The login nodes are meant for file editing, simple data management, and other tasks that use minimal compute resources.
- All computationally demanding jobs should be submitted and run through the batch queuing system. Any intensive code compilation tasks also should be done on compute nodes.
- **Do not use the login nodes for**
  - computationally intensive processes,
  - as hosts for running intensive workflow management tools
  - as primary data transfer nodes for large or numerous data transfers
  - as servers providing other services accessible to the Internet.
  - running Jupyter notebooks
- **For GPU codes, users should request interactive sessions on batch nodes for compilations (the host CPU is different from the login node).**

# Outline

- System overview and login
- Slurm scheduler – partition information
- Running jobs
  - MPI
  - OpenMP
  - MPI/OpenMP hybrid
  - GPU jobs
- Filesystems
  - Example local scratch usage
- Summary

# Expanse Scheduler Information

- Expanse uses the **Simple Linux Utility for Resource Management (SLURM)** batch environment.

- Submit jobs using the `sbatch` command:

```
$ sbatch mycode-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	mycode	user	PD	0:00	1	(Priority)

- Once the job is running:

```
$ squeue -u $USER
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
8718049	debug	mycode	user	R	0:02	1	expanse-14-01

- Cancel a running job:

```
$ scancel 8718049
```

# Expanse Scheduler Information

- Expanse has both node exclusive and shared partitions (just like Comet). Partitions that are familiar from Comet:
  - compute – CPU nodes, exclusive access. ***Need to be careful since this is now 128 cores => will be using up SUs very quickly.*** Make sure to use the full node resources efficiently, and ***do not use for jobs that don't require all 128 cores OR the full memory.***
  - **shared** – CPU nodes, shared access for jobs needing < 128 cores
  - **gpu** – GPU nodes, exclusive access
  - **gpu-shared** – GPU nodes, shared access for jobs needing < 4 GPUs
  - **large-shared** - Large memory nodes
  - **debug** – for shorts tests, quick access
- New partitions on Expanse:
  - **gpu-debug** – for short tests, compilation tasks
  - **preempt** - discounted jobs to run on free nodes that can be pre-empted by jobs in other partitions (compute, shared)
  - **gpu-preempt** - discounted jobs to run on free nodes that can be pre-empted by jobs in other partitions (gpu, gpu-shared)

# Expanse Partitions

<b>Partition Name</b>	<b>Max Walltime</b>	<b>Max Nodes/Job</b>	<b>Max Running Jobs</b>	<b>Max Running + Charge Queue Jobs</b>	<b>Charge Factor</b>	<b>Comments</b>
compute	48 hrs	32	64	128	1	* Used for exclusive access to regular compute nodes
shared	48 hrs	1	4096	4096	1	Single-node jobs using fewer than 128 cores
gpu	48 hrs	4	16	24	1	Used for exclusive access to the GPU nodes
gpu-shared	48 hrs	1	16	24	1	Single-node job using fewer than 4 GPUs
large-shared	48 hrs	1	1	4	1	Single-node jobs using large memory up to 2 TB (minimum memory required 256G)
debug	15 min	2	1	2	1	Priority access to compute nodes set aside for testing of jobs with short walltime and limited resources
gpu-debug	15 min	2	1	2	1	** Priority access to gpu nodes set aside for testing of jobs with short walltime and limited resources
preempt	7 days	32		128	.8	Discounted jobs to run on free nodes that can be pre-empted by jobs submitted to any other queue
gpu-preempt	7 days	1			.8	<b>(NO REFUNDS)</b>

# Expanse Scheduler – required parameters

--partition (-p)

--nodes (-N)

--ntasks-per-node OR --ntasks (-n): If both are specified SLURM picks ntasks

--wallclock (-t)

--account (-A): Unlike Comet we do **\*not\*** pick a default

--gpus : Total number of GPUs needed by job (unlike Comet where we're setting –gres).

**Note:** --mem, --mem-per-cpu, --mem-per-gpu are not required. However, it is recommended to explicitly request (using one of these options) what is needed **–the default is 1GB/core even on the “compute” and “gpu” partitions which are exclusive.**

# Outline

- System overview and login
- Slurm scheduler – partition information
- Running jobs
  - MPI
  - OpenMP
  - MPI/OpenMP hybrid
  - GPU jobs
- Filesystems
  - Example local scratch usage
- Summary

# Interactive Jobs

- You can request an interactive session using the srun command. The following example will request one regular compute node, 128 cores, in the debug partition for 30 minutes:

```
srun --partition=debug --pty --nodes=1 --ntasks-per-node=128 --mem=248G -t  
00:30:00 -A xyz123 --wait=0 --export=ALL /bin/bash
```

- The following example will request a GPU node, 10 cores, 1 GPU and 90G (all the memory) in the gpu-shared partition for 30 minutes:

```
srun --partition=gpu-shared --pty --nodes=1 --ntasks-per-node=10 --mem=90G --  
gpus=1 -A xyz123 -t 00:30:00 --wait=0 --export=ALL /bin/bash
```

# Example batch script: MPI Job

(/cm/shared/examples/sdsc/mpi)

```
#!/bin/bash
#SBATCH --job-name="hellompi"
#SBATCH --output="hellompi.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=128
#SBATCH --mem-per-cpu=1800M
#SBATCH --account=XYZ123
#SBATCH --export=ALL
#SBATCH -t 00:10:00

#This job runs with 2 nodes, 128 cores per node for a total of 256 cores.

## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

## Use srun to run the job

srun --mpi=pmi2 -n 256 --cpu-bind=rank ./hello_mpi
```

# Example batch script: OpenMP Job

(/cm/shared/examples/sdsc/openmp)

```
#!/bin/bash
#SBATCH --job-name="hell_openmp_shared"
#SBATCH --output="hello_openmp_shared.%j.%N.out"
#SBATCH --partition=shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=16
#SBATCH --mem=32G
#SBATCH --export=ALL
#SBATCH -t 01:30:00

# AOCC environment

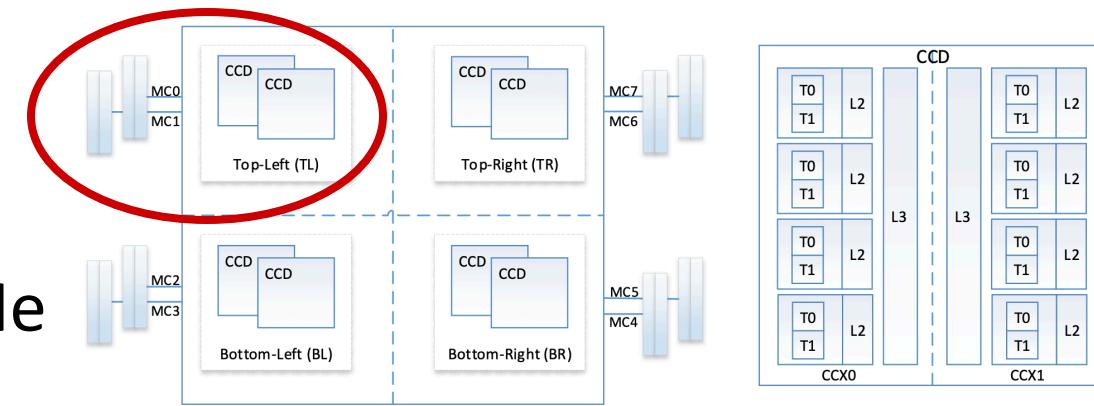
module purge
module load slurm
module load cpu
module load aocc

#SET the number of openmp threads
export OMP_NUM_THREADS=16

#Run the openmp job
./hello_openmp
```

# For Hybrid MPI/OpenMP jobs: recall NPS4 Configuration

- The processor is partitioned into four NUMA domains.
- Each logical quadrant is a NUMA domain.
- Memory is interleaved across the two memory channels
- PCIe devices will be local to one of four NUMA domains (the IO die that has the PCIe root for the device)
- ***Important to keep NUMA architecture in mind. Specifically for hybrid MPI/OpenMP jobs.***



[https://developer.amd.com/wp-content/resources/56338\\_1.00\\_pub.pdf](https://developer.amd.com/wp-content/resources/56338_1.00_pub.pdf)

# Example batch script: Hybrid MPI/OpenMP Job

/cm/shared/examples/sdsc/mpi-openmp-hybrid

```
#!/bin/bash
#SBATCH --job-name="hellohybrid"
#SBATCH --output="hellohybrid.%j.%N.out"
#SBATCH --partition=shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=16
#SBATCH --account=XYZ123
#SBATCH --export=ALL
#SBATCH -t 01:30:00

#Environment
module purge
module load slurm
module load cpu
module load intel
module load intel-mpi

#Run
export OMP_NUM_THREADS=16
mpirun -genv I_MPI_PIN_DOMAIN=omp:compact ./hello_hybrid
```

# Example batch script: Hybrid MPI/OpenMP Job

```
#!/bin/bash
#SBATCH --job-name="hpl-2node"
#SBATCH -o hpl.%j.out
#SBATCH -e hpl.%j.err
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=32
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1800M
#SBATCH -A use300
#SBATCH --export=ALL
#SBATCH -t 10:00:00
module purge
module load cpu
module load slurm
module load aocc
module load openmpi
XHPL=xhpl
mpirun -x OMP_NUM_THREADS --mca pml ucx --mca osc ucx --map-by l3cache ./xhpl
```

# Example batch script: OpenACC job on GPU nodes

```
#!/bin/bash
#SBATCH --job-name="OpenACC"
#SBATCH --output="OpenACC.%j.%N.out"
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --account=XYZ123
#SBATCH --gpus=1
#SBATCH -t 01:00:00

#Environment
module purge
module load slurm
module load gpu
module load pgi

#Run the job
./laplace2d.openacc.exe
```

# Example batch script: Multi-node GPU job

```
#!/bin/bash
#SBATCH --job-name="test"
#SBATCH -o test.%j.out
#SBATCH --qos=gpu-unlim
#SBATCH -e test.%j.err
#SBATCH --partition=gpu
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=10
#SBATCH --mem=370G
#SBATCH --gpus=16
#SBATCH --export=None
#SBATCH -t 04:00:00

## Modules
module purge
module load gpu
module load slurm
module load intel
module load openmpi
module load cuda10.2/toolkit

srun --mpi=pmi2 -n 180 --cpus-per-task=10 $HPL_DIR/bin/CUDA/xhpl
```

# Outline

- System overview and login
- Slurm scheduler – partition information
- Running jobs
  - MPI
  - OpenMP
  - MPI/OpenMP hybrid
  - GPU jobs
- **Filesystems**
  - Example local scratch usage
- Summary

# Home File System

- After logging in, users are placed in their home directory, /home, also referenced by the environment variable \$HOME.
- The home directory is limited in space and should be used only for source code, binaries, and small input files.
- Users will have a quota of 100GB in /home.
- Do **not** run jobs doing intensive IO to/from the home file system, as it is not set up for high performance throughput. Job scripts should use Lustre or local scratch for IO.

# Parallel Lustre Filesystems

- Global parallel filesystem:
  - 12 PB Lustre parallel file system
  - 140 GB/second performance
- Two Lustre filesystems locations on Expanse (both are part of the \*same\* filesystem):
  - Lustre scratch location: /expanse/lustre/scratch/\$USER (purged for files older than 90 days based on create date)
  - Lustre projects location: /expanse/lustre/projects/groupid/\$USER (usage tracked to comply with group allocation).
- SDSC limits the number of files that can be stored in /lustre/scratch filesystem to 2 million files per user.
- Users should contact user support for assistance at [help@xsede.org](mailto:help@xsede.org), if their workflow requires extensive small I/O, to avoid causing system issues associated with load on the metadata server.

# Node local NVMe based scratch filesystem

- All Expanse nodes feature NVMe based local scratch storage. The disk sizes vary based on then nodes:
  - Regular compute nodes: 1TB (~900 GB usable)
  - Large memory nodes: 3.2TB (~2.9TB usable)
  - GPU nodes: 1.6TB (~1.4TB usable)
- NVMe based storge is excellent for IOPs intensive workloads. This can also be useful for scratch files (small and large) generated on a per task basis.
- Users can access the SSDs only during job execution under the following directories local to each compute node:  
`/scratch/$USER/job_${SLURM_JOBID}`
- **The location is \*purged\* at the end of a job** so any important data must be copied out.

# Example batch script: local scratch usage (Gaussian application)

```
#!/bin/bash
#SBATCH --job-name="gaussian"
#SBATCH --output="gaussian.%j.%N.out"
#SBATCH --partition=shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=32
#SBATCH --account=XYZ123
#SBATCH --export=ALL
#SBATCH -t 00:10:00

module purge
module load cpu
module load gaussian/16.C.01
exe=`which g16`
export GAUSS_SCRDIR=/scratch/$USER/job_${SLURM_JOBID}

filename=water_opt_32c.dat
bash ./getcpusets $$
cat $$.$out $filename >file.tmp.$$
/usr/bin/time $exe < file.tmp.$$ > $filename.out
rm -f $$.$out file.tmp.$$
```

# Example batch script: local scratch usage

```
#!/bin/bash
#SBATCH --job-name="localscratch2"
#SBATCH --output="localscratch2.%j.%N.out"
#SBATCH --partition=shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --export=ALL
#SBATCH -t 00:30:00

#Copy binary to SSD
cp IOR.exe /scratch/$USER/job_${SLURM_JOBID}

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

#Run IO benchmark and redirect output to log file in local scratch
mpirun -np 4 ./IOR.exe -F -t 1m -b 4g -v -v > IOR.out.${SLURM_JOBID}

#Copy output log file back to original submit directory
cp IOR.out.${SLURM_JOBID} ${SLURM_SUBMIT_DIR}
```

# Summary

- Expanse uses SLURM for scheduling jobs. All jobs on Expanse must be run via the scheduler. *Do not run jobs on the login nodes.*
- Expanse supports both node exclusive (“compute”, “gpu”, “preempt”) and shared partitions (“shared”, “large-shared”, “gpu-shared”, “debug”, “gpu-debug”).
- Account information is required in job scripts (no default unlike Comet).
- Filesystem options – Lustre parallel filesystem, NVMe based node local scratch filesystem.
- Example job scripts for common applications are in:  
**/cm/shared/examples/sdsc**
- *Expanse has been in Production since December, 2020.* Follow all things Expanse at <https://expanse.sdsc.edu> !