CIML Summer Institute, Day 1

Day 1: Running Jobs

June 27, 2022

Mary Thomas

San Diego Supercomputer Center

SDSC SAN DIEGO SUPERCOMPUTER CENTER
UC San Diego

Outline

- **Introduction**
- Compiling and Linking Code
- Running Jobs
- Additional Examples
  - MPI Jobs
  - OpenMP Jobs
  - Hybrid MPI-OpenMP Jobs
- Final Comments

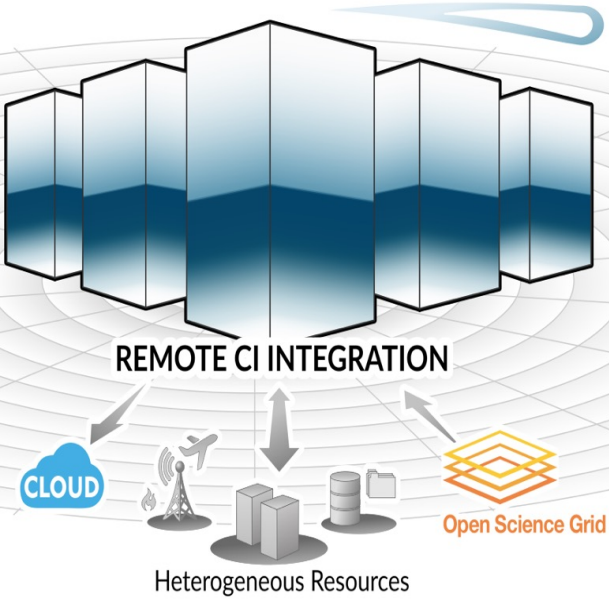
Expanse

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

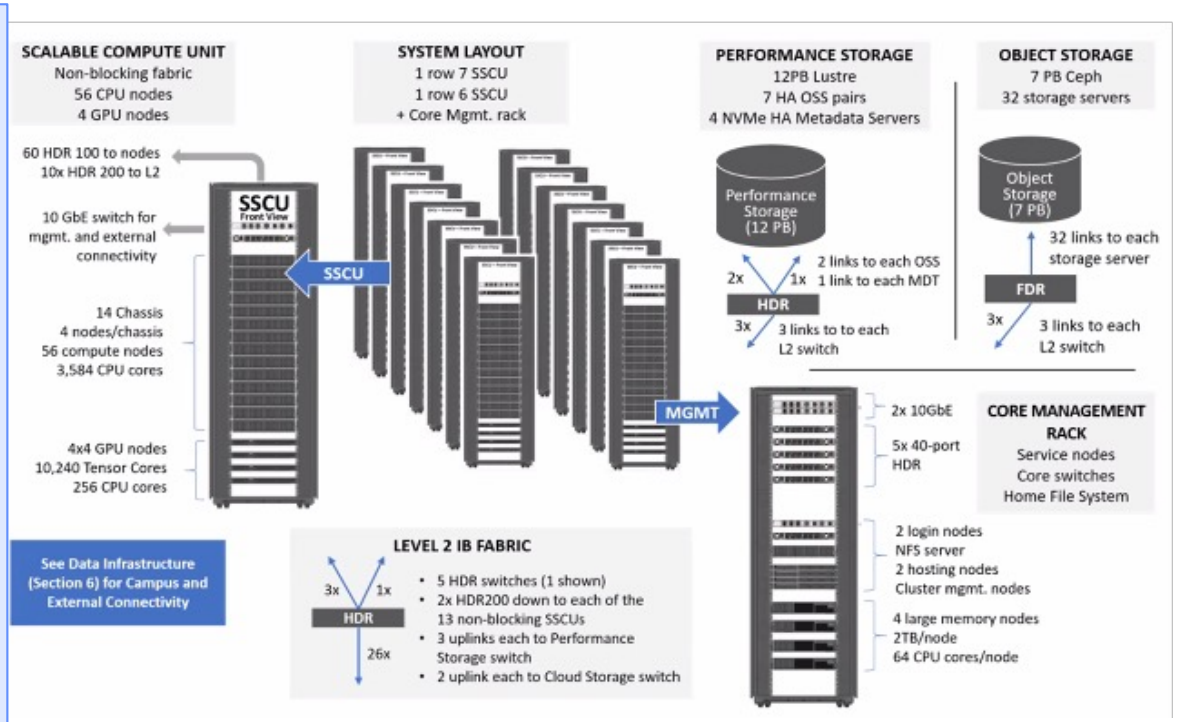
INNOVATIVE OPERATIONS
Composable Systems
High-Throughput Computing
Science Gateways
Interactive Computing
Containerized Computing
Cloud Bursting

For more details see the Expanse user guide @ https://www.sdsc.edu/support/user_guides/expanse.html
and the "Introduction to Expanse" webinar @ https://www.sdsc.edu/event_items/202006_Introduction_to_Expanse.html

Expanse Heterogeneous Architecture

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*



Outline

- Introduction
- **Compiling and Linking Code**
- Running Jobs
- Additional Examples
  - MPI Jobs
  - OpenMP Jobs
  - Hybrid MPI-OpenMP Jobs
- Final Comments

Supported Compilers on Expanse

- CPU nodes
  - GNU, Intel, AOCC (AMD) compilers
  - multiple MPI implementations (OpenMPI, MVAPICH2, and IntelMPI).
  - A majority of applications have been built using gcc/10.2.0 which features AMD Rome specific optimization flags (-march=znver2).
  - Intel, and AOCC compilers all have flags to support Advanced Vector Extensions 2 (AVX2).
- GPU Compiling:
  - Expanse GPU nodes have GNU, Intel, and PGI compilers.
  - Note: Expanse login nodes are not the same as the GPU nodes → all GPU codes must be compiled by requesting an interactive session on the GPU nodes.

https://www.sdsc.edu/support/user_guides/expanse.html#compiling

GNU Compilers: CPU & GPU

Table of recommended GNU compilers:

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

- For AVX support, compile with -march=core-avx2.
- AVX support only available in version 4.7 or later --> explicitly load the gnu/4.9.2 module
- For more information on the GNU compilers:

  $*man [gfortran | gcc | g++]*

GNU Compilers

- The GNU compilers can be loaded by executing the following commands at the Linux prompt or placing in your startup files (~/.cshrc or ~/.bashrc)

```
[user@login02]$ module purge
[user@login02]$ module load slurm
[user@login02]$ module load cpu
[user@login02]$ module load gcc/10.2.0
[user@login02]$ module load openmpi/4.0.4
[user@login02]$ module list
Currently Loaded Modules:

  1) slurm/expanse/20.02.3   2) cpu/1.0   3) gcc/10.2.0   4) openmpi/4.0.4
```

AMD AOCC Compilers: CPU Only

Language	Serial	MPI	OpenMP	MPI + OpenMP
Fortran	flang	mpif90	ifort -openmp	mpif90 -openmp
C	clang	mpiclang	icc -openmp	mpicc -openmp
C++	clang++	mpiclang	icpc -openmp	mpicxx -openm

The AMD Optimizing C/C++ Compiler (AOCC) is only available on CPU nodes. AMD compilers can be loaded using the module load command:

$ *module load aocc*

For more information on the AMD compilers:

$ [flang | clang] -help

Using the AOCC Compilers

- If you have modified your environment, you can change your environment by swapping modules or executing the module purge & load commands at the Linux prompt
- Place the load commands in your startup file (~/.cshrc or ~/.bashrc) or batch script

```
[user@login02 ~]$ module list
Currently Loaded Modules:
 1) slurm/expanse/21.08.8   2) cpu/0.15.4   3) intel/19.1.1.217   4) openmpi/4.0.4
[user@login02 ~]$ module purge
[user@login02 ~]$ module list
No modules loaded
[user@login02 ~]$ module load slurm
[user@login02 ~]$ module load cpu
[user@login02 ~]$ module load intel
[user@login02 ~]$ module load openmpi/4.0.4
[user@login02 ~]$ module list
Currently Loaded Modules:
  1) slurm/expanse/21.08.8   2) cpu/0.15.4   3) intel/19.1.1.217   4) openmpi/4.0.4
[user@login02 MPI]$ module swap intel aocc
Due to MODULEPATH changes, the following have been reloaded:
  1) openmpi/4.0.4
[user@login02 ~]$ module list
Currently Loaded Modules:
  1) slurm/expanse/20.02.3   2) cpu/1.0   3) aocc/2.2.0   4) openmpi/4.0.4
[user@login02 ~]$ which mpirun
[mthomas@login01 env_info]$ which mpirun
/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/clang-10.0.0/openmpi-4.0.4-nz2g566opuddnwt5ymjorz2cdgb4spfm/bin/mpirun
```

Loaded the wrong module.
Use swap

Intel Compilers: CPU and GPU

Default/Suggested Compilers to used based on programming model and languages:

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

The Intel compilers and the MVAPICH2 MPI compiler wrappers can be loaded by executing the following commands at the Linux prompt:

$module load intel mvapich2

For more information on the Intel compilers:

$[ifort | icc | icpc] -help

Using the Intel Compilers

```
[user@login02 ~]$ module list
[user@login02 MPI]$ module list
Currently Loaded Modules:
  1) slurm/expanse/20.02.3   2) cpu/1.0   3) aocc/2.2.0   4) openmpi/4.0.4
[user@login02 ~]$ module purge
[user@login02 ~]$ module list
No modules loaded
[mthomas@login01 env_info]$ which mpirun
/usr/bin/which: no mpirun in (/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin)
[user@login02 ~]$ module load slurm
[user@login02 ~]$ module load cpu
[user@login02 ~]$ module load intel
[user@login02 ~]$ module load openmpi/4.0.4
[user@login02 ~]$ module list
Currently Loaded Modules:
  1) slurm/expanse/20.02.3   2) cpu/0.15.4   3) intel/19.1.1.217   4) openmpi/4.0.4
[user@login02 ~]$  which mpirun
/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen2/intel-19.1.1.217/openmpi-4.0.4-f5mc6sg2jtrw7qqdksf6tru4vo4tawrv/bin/mpirun
[mthomas@login01 env_info]$
```

Accessing Intel Compiler Features

- Advanced Vector Extensions (AVX2): to enable AVX2 support
  - compile with the `-march=core-avx2` option.
  - https://en.wikipedia.org/wiki/Advanced_Vector_Extensions (128/256bit SIMD, Vector ops (MPI broadcast, gather, …)
  - Note that -march=core-avx2 alone does not enable aggressive optimization, so compilation with -O3 is also suggested.

- Intel Math Kernal Lib (MKL) libraries are available as part of the "intel" modules on Expanse.
  - Once this module is loaded, the environment variable INTEL_MKLHOME points to the location of the mkl libraries and
  - Use MKL Link Advisor to see what libraries are recommended for your compiler and system configuration:
  - https://software.intel.com/content/www/us/en/develop/articles/intel-mkl-link-line-advisor.html

PGI Compilers

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

- PGI (formerly The Portland Group, Inc.): created a set of commercially available Fortran, C and C++ compilers for high-performance computing systems.
- It is now owned by NVIDIA: PGI is on available on the GPU nodes.
- To compile code, you need to obtain an interactive node.
- For AVX support, compile with –fast
- For more information on the PGI compilers run: *$ man [pgf90 | pgcc | pgCC]*

Note: PGI Compilers will be discussed in the Section 2.5 of the CIML Summer Institute

PGI Compilers: GPU Only

```
[user@login02 ~]$ module reset
[user@login02 ~]$ module load gpu
[user@login02 ~]$ module load pgi
[user@login02 ~]$
[user@login02 ~]$ which pgcc
/cm/shared/apps/spack/gpu/opt/spack/linux-centos8-skylake_avx512/gcc-8.3.1/pgi-20.4-
2tsjnv2icisxmgdy4mijl4t5mkbr32ea/linux86-64/20.4/bin/pgcc
[user@login02 ~]$ which mpicc
/cm/shared/apps/spack/gpu/opt/spack/linux-centos8-skylake_avx512/gcc-8.3.1/pgi-20.4-
2tsjnv2icisxmgdy4mijl4t5mkbr32ea/linux86-64/20.4/mpi/openmpi-3.1.3/bin/mpicc
```

PGI supports the following high-level languages:

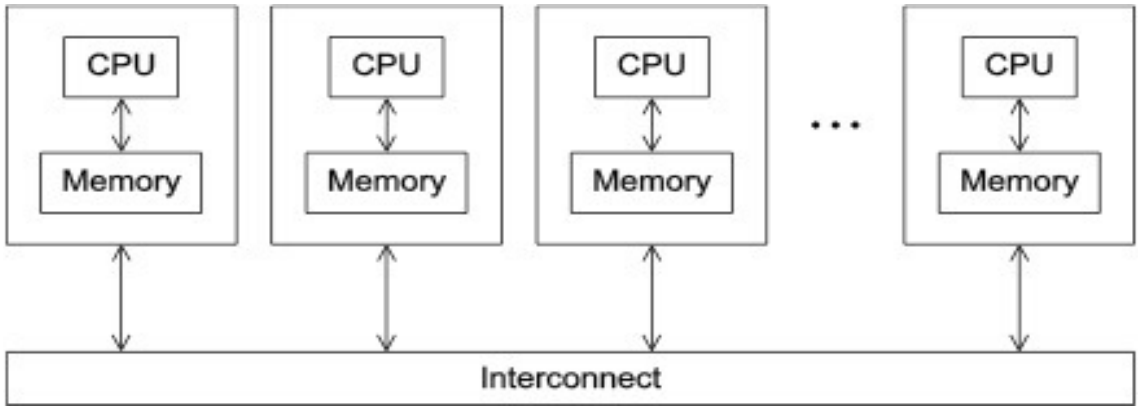
- Fortran 77, 90/95/2003, 2008 (partial)
- High Performance Fortran (HPF)
- ANSI C99 with K&R extensions
- ANSI/ISO C++
- CUDA Fortran
- OpenCL
- OpenACC
- OpenMP

Outline

- Introduction
- Compiling and Linking Code
- **Running Jobs**
- Additional Examples
  - MPI Jobs
  - OpenMP Jobs
  - Hybrid MPI-OpenMP Jobs
- Final Comments

Parallel Models: Distributed Memory

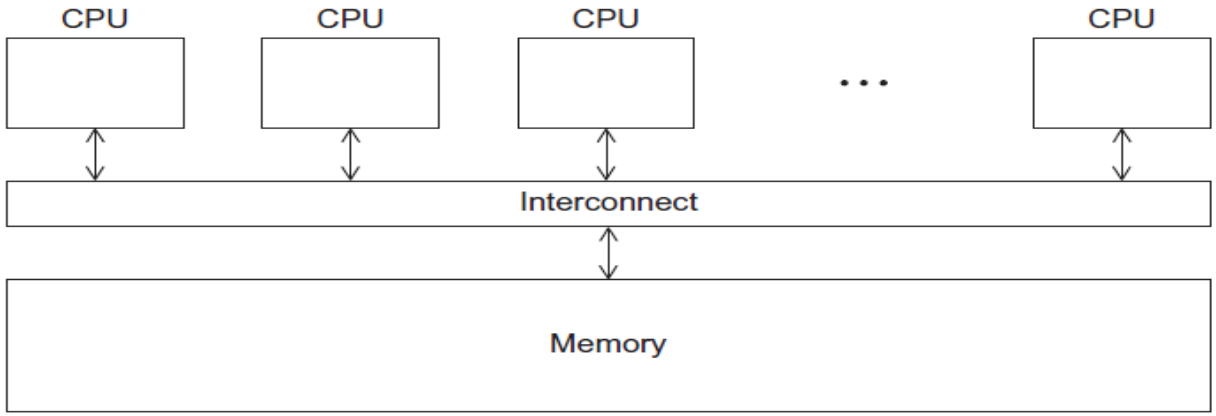
- Programs run asynchronously, pass messages for communication and coordination between resources.
- Examples include: SOA-based systems, massively multiplayer online games, peer-to-peer apps.



- Different types of implementations for the message passing mechanism: HTTP, RPC-like connectors, message queues
- HPC historically uses the Message Passing Interface (MPI)

Parallel Models: Shared Memory

- CPUs all share same localized memory (SHMEM);
  - Coordination and communication between tasks via interprocessor communication (IPC) or virtual memory mappings.
- May use uniform or non-uniform memory access (UMA or NUMA); cache-only memory architecture (COMA).



- Most common HPC API's for using SHMEM:
- Portable Operating System Interface (POSIX); Open Multi-Processing (OpenMP) designed for parallel computing – best for multi-core computing.

Running Jobs on Expanse

- "***batch mode***:" you submit jobs to be run on the compute nodes using the ***sbatch*** command as described below.
- Remember that computationally intensive jobs should be run only on the compute nodes and not the login nodes.
- Expanse places limits on the number of jobs queued and running on a per group (allocation) and partition basis.
- Please note that submitting a large number of jobs (especially very short ones) can impact the overall scheduler response for all users.

Methods for Running Jobs on Expanse

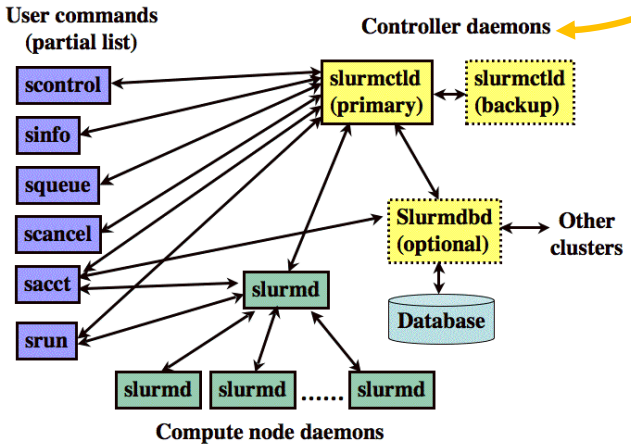
- Batch Jobs: Submit batch scripts from the login nodes to a batch service:
  - Expanse uses the Simple Linux Utility for Resource Management (SLURM) batch environment.
  - Partition (queue)
  - Time limit for the run (maximum of 48 hours)
  - Number of nodes, tasks per node; Memory requirements (if any)
  - Job name, output file location; Email info, configuration
- Interactive Jobs: Use the srun command to request 'live' nodes from Slurm for command line, interactive access
    - See below

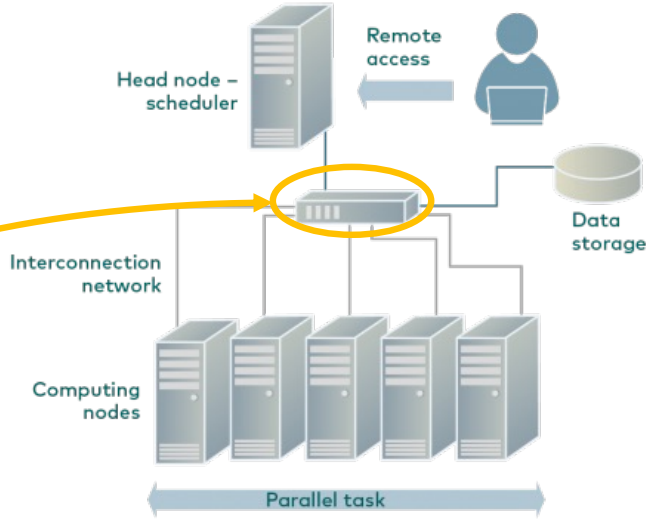
https://www.sdsc.edu/support/user_guides/expanse.html#running

Slurm Resource Manager

Simple Linux Utility for Resource Management

- "Glue" for parallel computer to schedule and execute jobs
- Role: Allocate resources within a cluster
  - Nodes (unique IP address)
  - Interconnect/switches
  - Generic resources (e.g. GPUs)
  - Launch and otherwise manage jobs





- Functionality:
  - Prioritize queue(s) of jobs;
  - decide when and where to start jobs;
  - terminate job when done;
  - Appropriate resources;
  - manage accounts for jobs

https://slurm.schedmd.com/sbatch.html

Common Slurm Commands

- 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, monitor its state:

  $ **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

https://slurm.schedmd.com/sbatch.html

Slurm Partitions on Expanse

Partition Name	Max Walltime	Max Nodes/Job	Max Running Jobs	Max Running + Queued Jobs	Charge Factor	Notes
compute	48 hrs	32	32	64	1	Used for exclusive access to regular compute nodes; *limit applies per group*
shared	48 hrs	1	4096	4096	1	Single-node jobs using fewer than 128 cores
gpu	48 hrs	4	4	8 (32 Tres GPU)	1	Used for exclusive access to the GPU nodes
gpu-shared	48 hrs	1	24	24 (24 Tres GPU)	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	30 min	2	1	2	1	Priority access to shared nodes set aside for testing of jobs with short walltime and limited resources
gpu-debug	30 min	2	1	2	1	Priority access to gpu-shared nodes set aside for testing of jobs with short walltime and limited resources; *max two gpus per job*
preempt	7 days	32		128	.8	Non-refundable discounted jobs to run on free nodes that can be pre-empted by jobs submitted to any other queue
gpu-preempt	7 days	1		24 (24 Tres GPU)	.8	Non-refundable discounted jobs to run on unallocated nodes that can be pre-empted by higher priority queues

https://www.sdsc.edu/support/user_guides/expanse.html#running

Example Batch Script: Show ENV

```
[uswername@login02 calc-prime]$ cat mpi-prime-slurm.sb
#!/bin/bash
#SBATCH --job-name="mpi_prime"
#SBATCH --output="mpi_prime.%j.%N.out"
####SBATCH --partition=compute
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --export=ALL
#SBATCH -t 00:10:00
#SBATCH -A use300

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

## echo job name and id:
echo "SLURM_JOB_NAME:  $SLURM_JOB_NAME"
echo "SLURM_JOB_ID:    $SLURM_JOB_ID"
d=`date`
echo "DATE:  $d"
```

```
[mthomas@login02 calc-prime]$ sbatch --export=NLO=1000 mpi-prime-slurm.sb
Submitted batch job 14126259
[mthomas@login02 calc-prime]$ !sq
squeue -u mthomas
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
14126259     debug mpi_prim  mthomas  R       0:04      1 exp-9-55
[mthomas@login01 env_info]$ cat envinfo.14126259.exp-4-35.out
SLURM_JOB_NAME:  envinfo
SLURM_JOB_ID:    14126259
hostname=  exp-4-35
date=  Sun Jun 26 22:05:15 PDT 2022
whoami=  mthomas
pwd=  /home/mthomas/hpctr-examples/env_info
Currently Loaded Modules:       1) slurm/expanse/21.08.8   2) cpu/0.15.4
---------------------------------
env= LD_LIBRARY_PATH=/cm/shared/apps/slurm/current/lib64/slurm:
[SNIP]
/cm/shared/apps/slurm/current/lib64
SLURM_SUBMIT_DIR=/home/mthomas/hpctr-examples/env_info
HISTCONTROL=ignoredups
DISPLAY=localhost:16.0
HOSTNAME=exp-4-35
[SNIP]
```

Example Code: https://github.com/sdsc-hpc-training-org/hpctr-examples

SLURM Environment Variables

https://slurm.schedmd.com/sbatch.html#lbAJ

Internal ENV vars that exist when job submitted:

INPUT ENVIRONMENT VARS

https://slurm.schedmd.com/sbatch.html#lbAJ

- Upon startup, sbatch will read and handle the options set in the following environment variables.
- SBATCH_JOB_NAME
  - Same as -J, --job-name
- SBATCH_ACCOUNT
  - Same as -A, --account
- SBATCH_TIMELIMIT
  - Same as -t, --time

OUTPUT ENVIRONMENT VARS

https://slurm.schedmd.com/sbatch.html#lbAK

- The Slurm controller will set the following variables in the environment of the batch script.
- SLURM_EXPORT_ENV
  - Same as --export.
- SLURM_JOB_ID
  - The ID of the job allocation.
- SLURM_JOB_NAME
  - Name of the job.

Passing values into the batch script

- For SLURM: use the **--export** flag.
  - For other schedulers check documentation
- Example: pass the value of two variables x and B into the job script named jobscript.sbatch
  - sbatch --export=x=7,B=‘mystring' jobscript.sb
  - OR: sbatch --export=ALL,x=7,B= mystring ' jobscript.sbatch
- The first example will replace the user's environment with a new environment containing only values for x and B and the SLURM_* environment variables. The second will add the values for A and b to the existing environment.

Example: Passing Vars to Batch Script

```
[uswername@login02 calc-prime]$ cat mpi-prime-slurm.sb
#!/bin/bash
#SBATCH --job-name="mpi_prime"
#SBATCH --output="mpi_prime.%j.%N.out"
####SBATCH --partition=compute
#SBATCH --partition=debug
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=24
#SBATCH --export=ALL
#SBATCH -t 00:10:00
#SBATCH -A use300
## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4
## echo job name and id:
echo "SLURM_JOB_NAME:  $SLURM_JOB_NAME"
echo "SLURM_JOB_ID:    $SLURM_JOB_ID"
d=`date`
echo "DATE:  $d"
echo "Var NLO:  $NLO"
## Use srun to run the job, pass variable to code
srun --mpi=pmi2 -n 24 --cpu-bind=rank ./mpi_prime $NLO
```

Batch script showing environment, date, and passing variable

```
[mthomas@login02 calc-prime]$ sbatch --export=NLO=1000 mpi-prime-slurm.sb
Submitted batch job 9113467
[mthomas@login02 calc-prime]$ !sq
squeue -u mthomas
JOBID PARTITION     NAME     USER ST   TIME  NODES NODELIST(
9113467     debug mpi_prim  mthomas  R       0:04      1 exp-9-55
[mthomas@login02 calc-prime]$ cat  mpi_prime. 9113467.exp-9-55.out
SLURM_JOB_NAME:  mpi_prime   SLURM_JOB_ID:    9113467
DATE:  Fri Jan 28 14:54:54 PST 2022
Var NHI:  250000
The argument supplied is 250000
PRIME_MPI.     n_hi= 250000     C/MPI version
         N        Pi          Time
         1         0       0.000361
         2         1       0.000004
         4         2       0.000768
         8         4       0.000003
        16         6       0.000003
        32        11       0.000003
        64        18       0.000003
       128        31       0.000004
       256        54       0.000554
[SNIP]
     16384      1900       0.008385
     32768      3512       0.030101
     65536      6542       0.110856
    131072     12251       0.421177
```

SLURM “srun” Command

srun --mpi=pmi2 -n 24 --cpu-bind=rank ./mpi_prime Y

- Used to launch a parallel job on cluster managed by Slurm.
- If necessary, srun will first create a resource allocation in which to run the parallel job.
- Common arguments used on Expanse:
  - --mpi=<*mpi_type*>Identify the type of MPI to be used. Use ‘pmi2’
  - -n, --ntasks=<*number*>Specify the number of tasks to run.
  - -cpu-bind: bind tasks to CPUs
- what is the difference between mpirun and SLURM srun?
  - srun is optimized for Expanse (via the PMI interface) and more efficiently allocates, organizes, and starts up the MPI processes.

https://slurm.schedmd.com/srun.html

General Steps: Compiling/Running Jobs

- Change to your working directory (e.g. hpctr-examples):

```
[mthomas@login02 ~]$
[mthomas@login02 ~]$ cd /home/$USER/hpctr-examples/mpi
[mthomas@login02 mpi]$ ll hello_mpi.f90
-rw-r--r-- 1 mthomas use300 333 Feb 18 13:42 hello_mpi.f90
[mthomas@login02 mpi]$
```

- Verify that the correct modules are loaded:

```
[mthomas@login02 ~]$ module list
Currently Loaded Modules:
  1) shared   2) cpu/0.15.4   3) slurm/expanse/21.08.8
  4) sdsc/1.0   5) DefaultModules
```

- Compile MPI hello world code:

```
[mthomas@login02 mpi]$
[mthomas@login02 mpi]$ mpif90 -o hello_mpi hello_mpi.f90
[mthomas@login02 mpi]$
```

- Verify executable create date:

```
[mthomas@login02 mpi]$
[mthomas@login02 mpi]$ ls -lt hello_mpi
-rwxr-xr-x 1 mthomas use300 22440 Jun 26 18:45 hello_mpi
[mthomas@login02 mpi]$
```

- Submit job

```
[mthomas@login02 mpi]$
[mthomas@login02 mpi]$ sbatch hellompi-slurm.sb
Submitted batch job 14124495
[mthomas@login02 mpi]$
```

Outline

- Introduction
- Compiling and Linking Code
- Running Jobs
- **Additional Examples**
  - MPI Jobs
  - OpenMP Jobs
  - Hybrid MPI-OpenMP Jobs
- Final Comments

Additional Examples

- Copy code examples from:
  - /cm/shared/examples/sdsc/
  - These are the most recent verions
- Clone examples from
  - https://github.com/sdsc-hpc-training-org/expanse-101
- CPU Jobs & GPU Jobs

Outline

- Expanse Overview & Innovative Features
- Compiling and Linking Code
- Running Jobs
- Additional Examples
  - **MPI Jobs**
  - OpenMP Jobs
  - Hybrid MPI-OpenMP Jobs
- Final Comments

MPI Hello World

- Change to the MPI examples directory:

```
[user@login02 MPI]$ cat hello_mpi.f90
!  Fortran example
   program hello
   include 'mpif.h'
   integer rank, size, ierror, tag, status(MPI_STATUS_SIZE)

   call MPI_INIT(ierror)
   call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierror)
   call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierror)
   print*, 'node', rank, ': Hello world!'
   call MPI_FINALIZE(ierror)
   end
[user@login02 MPI]$
```

MPI Hello World: Compile

Set the environment and then compile the code

```
[user@login02 MPI]$ cat README.txt
[1] Compile:

# Load module environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

mpif90 -o hello_mpi hello_mpi.f90

[2a] Run using Slurm:

sbatch hellompi-slurm.sb

[2b] Run using Interactive CPU Node

srun --partition=debug --account=sds184 --pty --nodes=1 --ntasks-per-
node=128 --mem=248 -t 00:30:00 --wait=0 --export=ALL /bin/bash
```

```
[user@login02 MPI]$ module list

Currently Loaded Modules:
  1) cpu/1.0   2) slurm/expanse/20.02.3

[user@login02 MPI]$ module purge
[user@login02 MPI]$ module load slurm
[user@login02 MPI]$ module load cpu
[user@login02 MPI]$ module load gcc/10.2.0
[user@login02 MPI]$ module load openmpi/4.0.4
[user@login02 MPI]$ module list

Currently Loaded Modules:
  1) slurm/expanse/20.02.3   2) cpu/1.0   3) gcc/10.2.0   4) openmpi/4.0.4

[user@login02 MPI]$ mpif90 -o hello_mpi hello_mpi.f90
[user@login02 MPI]$
```

MPI Hello World: Batch Script

- To run the job, use the batch script submission command.
- Monitor the job until it is finished using the squeue command.

```
[user@login02 MPI]$ cat hellompi-slurm-gnu.sb
#!/bin/bash
#SBATCH --job-name="hellompi-gnu"
#SBATCH --output="hellompi-gnu.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=128
#SBATCH --export=ALL
#SBATCH –A abc123
#SBATCH -t 00:10:00
```

squeue -u mthomas

```
#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_gnu

[user@login02 MPI]$
```

```
[user@login02 MPI]$ sbatch hellompi-slurm-gnu.sb; squeue -u user
Submitted batch job 108910
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            108910   compute hellompi  user PD       0:00      2 (None)
[user@login02 MPI]$ cat hellompi-gnu.108910.exp-12-54.out
 node         4 : Hello world!
 node         5 : Hello world!
 node         7 : Hello world!
 node         0 : Hello world!
 node         2 : Hello world!
 node         3 : Hello world!
 node         9 : Hello world!
 node        10 : Hello world!

[SNIP]

node       247 : Hello world!
 node       248 : Hello world!
 node       249 : Hello world!
 node       186 : Hello world!
 node       220 : Hello world!
 node       203 : Hello world!
 node       135 : Hello world!
```

Outline

- Introduction
- Compiling and Linking Code
- Running Jobs
- Additional Examples
  - MPI Jobs
  - **OpenMP Jobs**
  - Hybrid MPI-OpenMP Jobs
- Final Comments

OpenMP Hello World

Change to the OPENMP examples directory:

```
[mthomas@login01 examples]$ cd OPENMP/
[mthomas@login01 OPENMP]$ ll
total 89
drwxr-xr-x 2 mthomas use300     7 Oct  7 11:28 .
drwxr-xr-x 7 mthomas use300     7 Oct  8 00:03 ..
-rwxr-xr-x 1 mthomas use300 19640 Oct  7 11:28 hello_openmp
-rw-r--r-- 1 mthomas use300   236 Oct  7 11:28 hello_openmp.f90
-rw-r--r-- 1 mthomas use300   672 Oct  7 11:28 hello_openmp_shared.108737.exp-6-56.out
-rw-r--r-- 1 mthomas use300   442 Oct  7 11:28 openmp-slurm-shared.sb
-rw-r--r-- 1 mthomas use300   168 Oct  7 11:28 README.txt
[mthomas@login01 OPENMP]$ cat hello_openmp.f90
      PROGRAM OMPHELLO
      INTEGER TNUMBER
      INTEGER OMP_GET_THREAD_NUM

!$OMP PARALLEL DEFAULT(PRIVATE)
      TNUMBER = OMP_GET_THREAD_NUM()
      PRINT *, 'HELLO FROM THREAD NUMBER = ', TNUMBER
!$OMP END PARALLEL


      END
```

OpenMP Hello World: Compile (using aocc compiler)

Set the environment and then compile the code

```
[mthomas@login01 OPENMP]$ cat README.txt
[1] Compile:

#load module environmentmodule
module purge
module load slurm
module load cpu
module load aocc

flang -fopenmp -o hello_openmp hello_openmp.f90

[2] Run:

sbatch openmp-slurm-shared.sb

[mthomas@login01 OPENMP]$
```

```
[mthomas@login01 OPENMP]$ module list

[mthomas@login01 OPENMP]$ module purge
[mthomas@login01 OPENMP]$ module load slurm
[mthomas@login01 OPENMP]$ module load cpu
[mthomas@login01 OPENMP]$ module load aocc

Currently Loaded Modules:
  1) slurm/expanse/20.02.3   2) cpu/1.0   3) aocc/2.2.0

[mthomas@login01 OPENMP]$

[mthomas@login01 MPI]$ mpif90 -o hello_mpi hello_mpi.f90
```

OpenMP Hello World: Controlling #Threads

A key issue when running OpenMP code is controlling thread behavior.
If you run from command line, it will work, but it is not recommended because you will be using Pthreads, which automatically picks the number of threads - in this case 24.

```
[expanse-ln2:~/expanse1010/PENMP]./hello_openmp
 Hello from Thread Number[           0 ] and Welcome Webinar!
 Hello from Thread Number[           2 ] and Welcome Webinar!
  .   .   .   .
 Hello from Thread Number[          22 ] and Welcome Webinar!
 Hello from Thread Number[          11 ] and Welcome Webinar!
 Hello from Thread Number[          23 ] and Welcome Webinar!
```

To control thread behavior, there are several key environment variables: OMP_NUM_THREADS controls the number of threads allowed, and OMP_PROC_BIND binds threads to “places” (e.g. cores) and keeps them from moving around (between cores).

```
[expanse-ln2:~/expanse1010PE/NMP] export OMP_NUM_THREADS=4; ./hello_openmp
HELLO FROM THREAD NUMBER = 3
HELLO FROM THREAD NUMBER = 1
HELLO FROM THREAD NUMBER = 2
HELLO FROM THREAD NUMBER = 0
```

See: https://www.ibm.com/support/knowledgecenter/SSGH2K_13.1.3/com.ibm.xlc1313.aix.doc/compiler_ref/ruomprun.html

OpenMP Hello World: Batch Script

```
[mthomas@login01 OPENMP]$ cat openmp-slurm-
shared.sb
#!/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
[mthomas@login01 OPENMP]$
```

- Note: Expanse supports shared-node jobs (more than one job on a single node).
- Many applications are serial or can only scale to a few cores.
- Shared nodes improve job throughput, provide higher overall system utilization, and allow more users to run on nodes.

```
[expanse-ln2:~/expanse101/OPENMP] cat openmp-slurm-
shared.sb
#!/bin/bash
#SBATCH --job-name="hell_openmp_shared"
#SBATCH --output="hello_openmp_shared.%j.%N.out"
#SBATCH --partition=shared
#SBATCH --share
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --mem=80G
#SBATCH --export=ALL
#SBATCH -t 01:30:00

#SET the number of openmp threads
export OMP_NUM_THREADS=16

#Run the openmp job
./hello_openmp
```

OpenMP Hello World: submit job & monitor

To run the job, type the batch script submission command:

```
[mthomas@login01 OPENMP]$ sbatch openmp-slurm-shared.sb
  Submitted batch job 108911
[mthomas@login01 OPENMP]$ squeue -u mthomas
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASO
108911    shared hell_ope  mthomas PD       0:00      1 (None)
[mthomas@login01 OPENMP]$ ll
total 98
drwxr-xr-x 2 mthomas use300     8 Oct  8 03:37 .
drwxr-xr-x 7 mthomas use300     7 Oct  8 00:03 ..
-rwxr-xr-x 1 mthomas use300 19640 Oct  7 11:28 hello_openmp
-rw-r--r-- 1 mthomas use300   236 Oct  7 11:28 hello_openmp.f90
-rw-r--r-- 1 mthomas use300   672 Oct  8 03:37 hello_openmp_shared.108911.exp-6-56.out
-rw-r--r-- 1 mthomas use300   442 Oct  7 11:28 openmp-slurm-shared.sb
-rw-r--r-- 1 mthomas use300   168 Oct  7 11:28 README.txt
```

```
[mthomas@login01 OPENMP]$ cat
hello_openmp_shared.108911.exp-6-56.out
 HELLO FROM THREAD NUMBER =          7
 HELLO FROM THREAD NUMBER =          6
 HELLO FROM THREAD NUMBER =         12
 HELLO FROM THREAD NUMBER =         10
 HELLO FROM THREAD NUMBER =          1
 HELLO FROM THREAD NUMBER =          2
 HELLO FROM THREAD NUMBER =          5
 HELLO FROM THREAD NUMBER =          0
 HELLO FROM THREAD NUMBER =          4
 HELLO FROM THREAD NUMBER =          3
 HELLO FROM THREAD NUMBER =         13
 HELLO FROM THREAD NUMBER =         15
 HELLO FROM THREAD NUMBER =         14
 HELLO FROM THREAD NUMBER =         11
 HELLO FROM THREAD NUMBER =          9
 HELLO FROM THREAD NUMBER =          8
[mthomas@login01 OPENMP]$
```

Outline

- Introduction
- Compiling and Linking Code
- Running Jobs
- Additional Examples
  - MPI Jobs
  - OpenMP Jobs
  - **Hybrid MPI-OpenMP Jobs**
- Final Comments

Hybrid MPI + OpenMP Hello World

```
#include <stdio.h>
#include "mpi.h"
#include <omp.h>

int main(int argc, char *argv[]) {
  int numprocs, rank, namelen;
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  int iam = 0, np = 1;

  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Get_processor_name(processor_name, &namelen);

  #pragma omp parallel default(shared) private(iam, np)
  {
    np = omp_get_num_threads();
    iam = omp_get_thread_num();
    printf("Hello Webinar particpants from thread %d out of %d from process %d out of
%d on %s\n",
           iam, np, rank, numprocs, processor_name);
  }

  MPI_Finalize();
}
```

Hybrid MPI + OpenMP Jobs

```
[mthomas@login01 HYBRID]$ cat README.txt

[1] Compile:

# Load module environment
module purge
module load slurm
module load cpu
module load intel
module load intel-mpi

export I_MPI_CC=icc
mpicc -qopenmp -o hello_hybrid hello_hybrid.c

[2] Run:

sbatch hybrid-slurm.sb
```

```
[mthomas@login01 HYBRID]$ cat hybrid-slurm.sb
#!/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 -B 2:16:1
#SBATCH --export=ALL
#SBATCH -t 01:30:00

# Load Module 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
```

```
[mthomas@login01 HYBRID]$ mpicc -qopenmp -o hello_hybrid hello_hybrid.c
[mthomas@login01 HYBRID]$ sbatch hybrid-slurm.sb
Submitted batch job 108875
[mthomas@login01 HYBRID]$ squeue -u mthomas
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            108875    shared hellohyb  mthomas PD       0:00      1 (None)
[mthomas@login01 HYBRID]$
```

Hybrid Hello World: Output

Code ran on:

- 1 node,
- 2 cores per node,
- 16 threads per core

```
[expanse-ln2:~/expanse101/HYBRID] cat hellohybrid. 108875.expanse-06-48.out | sort
Hello from thread 0 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 0 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 1 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 1 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 2 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 2 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 3 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 3 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 4 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 4 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 5 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 5 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 6 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 6 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 7 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 7 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 8 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 8 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 9 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 9 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 10 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 10 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 11 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 11 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 12 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 12 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 13 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 13 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 14 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 14 out of 16 from process 1 out of 2 on exp-6-56
Hello from thread 15 out of 16 from process 0 out of 2 on exp-6-56
Hello from thread 15 out of 16 from process 1 out of 2 on exp-6-56
```

Outline

- Introduction
- Compiling and Linking Code
- Running Jobs
- Additional Examples
  - MPI Jobs
  - OpenMP Jobs
  - Hybrid MPI-OpenMP Jobs
- **Final Comments**

When Things Go Wrong, Check Your User Environment

- Do you have the right modules loaded?
- What software versions do you need?
- Is your code compiled and updated
  - Did you compile it last year? Have the libraries changed?
- Are you running your job from the right location?
  - $HOME versus $WORK?

Run jobs from the right location

- Lustre scratch filesystem:
  - /oasis/scratch/expanse/$USER/temp_project
  - Preferred: Scalable large block I/O)
- Compute/GPU node local SSD storage:
  - /scratch/$USER/$SLURM_JOBID
  - Meta-data intensive jobs, high IOPs)
- Lustre projects filesystem:
  - /oasis/projects/nsf
- /home/$USER:
  - Only for source files, libraries, binaries.
  - Do not use for I/O intensive jobs.

Resources

- Expanse User Guide & Tutorial
  - https://www.sdsc.edu/support/user_guides/expanse.html
  - https://hpc-training.sdsc.edu/expanse-101/
- Hands-on/code examples for this tutorial:
  - https://github.com/sdsc-hpc-training-org/hpctr-examples
- SDSC Training Resources
  - https://www.sdsc.edu/education_and_training/training_hpc.html
- XSEDE Training Resources
  - https://www.xsede.org/for-users/training
  - https://cvw.cac.cornell.edu/expanse/

Thank You