Outline

- Expanse Overview & Innovative Features
- Getting Started
- Modules
- Account Management
- Compiling and Linking Code
- Running Jobs
- Hands-on Examples
    - MPI Jobs
    - OpenMP Jobs ⎤
    - GPU/CUDA Jobs ⎥ Not covering today
    - Hybrid MPI-OpenMP Jobs ⎦
- Final Comments

Basic Information

- Expanse User Guide:
  - https://www.sdsc.edu/support/user_guides/expanse.html

- You need to have an Expanse account in order to access the system. There are a few ways to do this:
  - Submit a proposal through the XSEDE Allocation Request System
  - PI on an active allocation can add you to their allocation (if you are collaborators working on the same project).
  - Request a trial account, instructions @ https://portal.xsede.org/allocations/startup.

- Online repo and information:
  - https://github.com/sdsc-hpc-training-org/expanse-101
  - https://hpc-training.sdsc.edu/expanse-101/

Outline

- **Expanse Overview & Innovative Features**
- Getting Started
- Modules
- Account Management
- Compiling and Linking Code
- Running Jobs
- Hands-on Examples
  - MPI Jobs
  - OpenMP Jobs
  - GPU/CUDA Jobs
  - Hybrid MPI-OpenMP Jobs
- Data and Storage

- 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

LONG-TAIL SCIENCE

Multi-Messenger Astronomy
Genomics
Earth Science
Social Science

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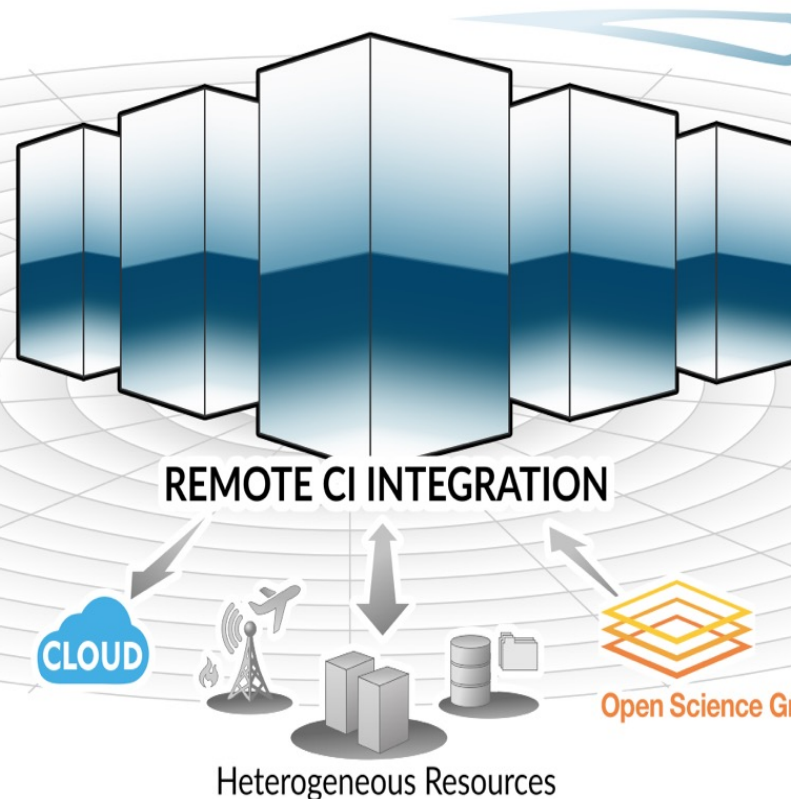
12PB Perf. Storage: 140GB/s, 200k IOPS
Fast I/O Node-Local NVMe Storage
7PB Ceph Object Storage
High-Performance R&E Networking

INNOVATIVE OPERATIONS

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

REMOTE CI INTEGRATION

CLOUD

Open Science Grid

Heterogeneous Resources

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

Outline

- Expanse Overview & Innovative Features
- **Getting Started**
- Modules
- Account Management
- Compiling and Linking Code
- Running Jobs
- Hands-on Examples
  - MPI Jobs
  - OpenMP Jobs
  - GPU/CUDA Jobs
  - Hybrid MPI-OpenMP Jobs
- Final Comments

Logging into Expanse

- Expanse supports Single Sign-On through the XSEDE User Portal
- From the command line using an XSEDE-wide password,
    - Coming soon the Expanse User Portal.
- 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
- Secure shell (SSH) command examples:

```
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 systems are identical.

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

- See the SDSC *Indispensable Security: Tips to Use SDSC's HPC Resources Securely"*
  - https://www.sdsc.edu/event_items/202007_CometWebinar.html

System Access: Clients

- Linux/Mac –
    - use terminal + installed ssh app
- Windows:
    - Win10 terminal app + installed ssh app
    - Older Windows OS's: ssh clients apps Putty, Cygwin
- Login hostname for SDSC Expanse:
    - login.expanse.sdsc.edu
    - 198.202.113.252

Switch

Login node

Internet

login.expanse.sdsc.edu

Thin Clients | Thin Clients | Thin Clients | Thin Clients
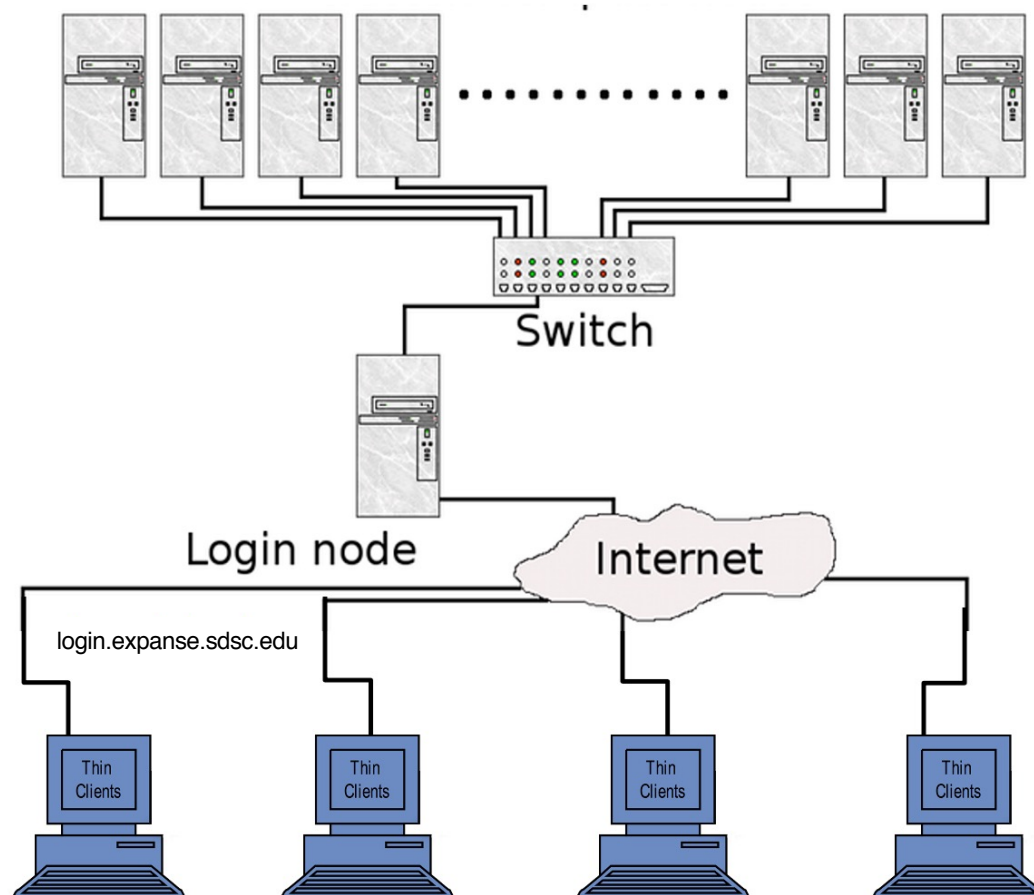
SDSC tutorial on HPC security: https://www.sdsc.edu/event_items/202007_CometWebinar.html

Example of a terminal connection:

```
(base) quantum:~ mthomas$ ssh -l mthomas login.expanse.sdsc.edu
Last login: Wed Oct  7 11:04:17 2020 from 76.176.117.51
[mthomas@login02 ~]$
[mthomas@login02 ~]$
[mthomas@login02 ~]$ whoami
mthomas
[mthomas@login02 ~]$
[mthomas@login02 ~]$ pwd
/home/mthomas
[mthomas@login02 ~]$
[mthomas@login02 ~]$
```

Typically you would also see a logon message – often called the MOTD (message of the day, located in /etc/motd). This has not been implemented at this point on Expanse

Using Login Nodes Properly

- The login nodes are meant for file editing, simple data analysis, & tasks that use minimal compute resources.
- All computationally demanding jobs should be submitted and run through the batch queuing system.
- Do not use the login nodes for:
  - computationally intensive processes,
  - as hosts for running 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
- Login nodes are not the same as the batch nodes.
  - Users should request an interactive sessions to compile ;arge programs.

Outline

- Expanse Overview & Innovative Features
- Getting Started
- **Modules**
- Account Management
- Compiling and Linking Code
- Running Jobs
- Hands-on Examples
  - MPI Jobs
  - OpenMP Jobs
  - GPU/CUDA Jobs
  - Hybrid MPI-OpenMP Jobs
- Final Comments

Expanse Environment Modules

- Expanse uses *Lmod*, a *Lua* based module system.
  - https://lmod.readthedocs.io/en/latest/010_user.html
- Users setup custom environments by loading available modules into the shell environment, *including needed compilers and libraries* and the batch scheduler.
- What modules let you do:
  - Dynamic modification of your shell environment
  - User can set, change, or delete environment variables
  - User chooses between different versions of the same software or different combinations of related codes.

Modules on Expanse

- Users will need to load the scheduler (e.g. slurm)
- Users will *not* see all available modules when they run command "module available" *without loading a compiler*.
- Use the command "module spider" option to see if a particular package exists and can be loaded, run command

  > *module spider <package>*

  > *module keywords <term>*

- For additional details, and to identify module dependencies modules, use the command

  > module spider <application_name>

- The module paths are different for the CPU and GPU nodes. Users can enable the paths by loading the following modules:

  > module load cpu (for cpu nodes)

  > module load gpu (for gpu nodes)

  - Avoid loading both modules

Module Command Examples

```
[[mahidhar_test@login01 ~]$ module reset
Resetting modules to system default. Reseting $MODULEPATH back to system default. All extra directories will be removed from
$MODULEPATH.
[mahidhar_test@login01 ~]$ module list


Currently Loaded Modules:
  1) shared   2) slurm/expanse/20.02.3   3) cpu/0.15.4   4) DefaultModules
```

List Current environment: list, li

```
[mahidhar_test@login01 ~]$ module avail


------------------------------ /cm/shared/apps/spack/cpu/lmod/linux-centos8-x86_64/Core ------------------------------
   abaqus/2018               bzip2/1.0.8        gcc/7.5.0        intel/19.1.1.217   parallel/20200822
   anaconda3/2020.11             cmake/3.18.2        gcc/9.2.0        libtirpc/1.2.6     pciutils/3.7.0
   aocc/2.2.0               curl/7.72.0        gcc/10.2.0 (D)   matlab/2020b       pigz/2.4
   aria2/1.35.0               emboss/6.6.0        gmp/6.1.2        mpfr/4.0.2         subversion/1.14.0
   arm-forge/21.0.1-linux-x86_64   gaussian/16.C.01   go/1.15.1        nbo/7.0-openblas   zstd/1.4.5
   byacc/master               gaussian09/09.E.01   idl/8.4          openjdk/11.0.2


------------------------------------- /cm/local/modulefiles -------------------------------------
   boost/1.71.0    cmjob    lua/5.3.5    shared (L)    singularitypro/3.5    slurm/expanse/20.02.3 (L)


------------------------------------- /cm/shared/apps/xsede/modulefiles -------------------------------------
   cue-login-env    xdinfo/1.5-1    xdusage/2.1-1


------------------------------------- /usr/share/modulefiles -------------------------------------
   DefaultModules (L)    cpu/0.15.4 (L)    gct/6.2    globus/6.0    gpu/0.15.4    nostack/0.15.4
```

Show available modules

Modules: Popular commands

Command	Description
module list	List the modules that are currently loaded
module avail	List the modules that are available in environment
module spider	List of the modules and extensions currently available
module display <module_name>	Show the environment variables used by <module name> and how they are affected
module unload <module name>	Remove <module name> from the environment
module load <module name>	Load <module name> into the environment
module swap <module one> <module two>	Replace <module one> with <module two> in the environment
module help	get a list of all the commands that module knows about do:
Shorthand notation: ml foo ml -bar	"ml" == module load foo "ml -bar" == module unload bar

SDSC Guidance: add module calls to your environment and batch scripts

Outline

- Expanse Overview & Innovative Features
- Getting Started
- Modules
- **Account Management**
- Compiling and Linking Code
- Running Jobs
- Hands-on Examples
    - MPI Jobs
    - OpenMP Jobs
    - GPU/CUDA Jobs
    - Hybrid MPI-OpenMP Jobs
- Final Comments

Multiple Allocations

- Many users will have access to multiple accounts and hence *projects*:
  - an allocation for a research project and a separate allocation for classroom or educational use
- Users should verify that the correct *project* is designated for all batch jobs.
- Awards are granted for a specific purposes and should not be used for other *projects*.
- To charge your job to one of your *projects*, replace << project >> with one from your list and put this PBS directive in your job script:
  - #SBATCH -A << project >>

Allocation Information

```
module load sdsc
expanse-client user
expanse-client user -r expanse_gpu
```

```
[mahidhar_test@login01 ~]$ module load sdsc
[mahidhar_test@login01 ~]$ expanse-client user

Resource   expanse

+---+---------------+----------+------------+------+-----------+----------------+
|   | NAME          | PROJECT  | TG PROJECT | USED | AVAILABLE | USED BY PROJECT |
+---+---------------+----------+------------+------+-----------+----------------+
| 1 | mahidhar_test | ddp386   |            |   4  |   10000   |             86 |
+---+---------------+----------+------------+------+-----------+----------------+

[mahidhar_test@login01 ~]$ expanse-client user -r expanse_gpu

Resource   expanse_gpu

+---+---------------+----------+------------+------+-----------+----------------+
|   | NAME          | PROJECT  | TG PROJECT | USED | AVAILABLE | USED BY PROJECT |
+---+---------------+----------+------------+------+-----------+----------------+
| 1 | mahidhar_test | ddp386   |            |   0  |    2500   |             21 |
+---+---------------+----------+------------+------+-----------+----------------+

[mahidhar_test@login01 ~]$
```

Outline

- Expanse Overview & Innovative Features
- Getting Started
- Modules
- Account Management
- **Compiling**
- Running Jobs
- Hands-on Examples
  - MPI Jobs
  - OpenMP Jobs
  - GPU/CUDA 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.

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*

Intel Compilers: GPU 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

- In this tutorial, we include hands-on examples that cover many of the cases in the table:
  - (1) MPI
  - (2) OpenMP
  - (3) HYBRID

GNU Compilers: CPU and GPU

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

```
[mthomas@login01 MPI]$ module purge
[mthomas@login01 MPI]$ module load slurm
[mthomas@login01 MPI]$ module load cpu
[mthomas@login01 MPI]$ module load gcc/10.2.0
[mthomas@login01 MPI]$ module load openmpi/4.0.4
[mthomas@login01 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
```

- For AVX support, compile with -mavx.
- Note that AVX support is only available in version 4.7 or later, so it is necessary to explicitly load the gnu/4.9.2 module until such time that it becomes the default.
- For more information on the GNU compilers: man [gfortran | gcc | g++]

Using the GNU Compilers

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

Outline

- Expanse Overview & Innovative Features
- Getting Started
- Modules
- Account Management
- Compiling and Linking Code
- **Running Jobs**
- Hands-on Examples
    - MPI Jobs
    - OpenMP Jobs
    - GPU/CUDA Jobs
    - Hybrid MPI-OpenMP Jobs
- Final Comments

Running Jobs on Expanse

- When you run in the 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 Runing Jobs on Expanse

- Expanse uses the **Simple Linux Utility for Resource Management (SLURM)** batch environment.
  - **Batch Jobs**: Submit batch scripts to Slurm from the login nodes:
    - 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 obtain nodes for 'live,' command line interactive access:
  - CPU:

    srun --partition=debug --account=XYZ123 --pty --nodes=1 --ntasks-per-node=128 --mem=248 -t 00:30:00 --wait=0 --export=ALL /bin/bash

  - GPU:

    srun --pty --account=XYZ123 --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 -p gpu-debug --gpus=1 -t 00:10:00 /bin/bash

Slurm Partitions on Expanse

Partition limits subject to change based on Early User Period evaluation

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

cified using -p option in batch script.

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:

  $ 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

Hands On Examples

- Clone examples:
  - https://github.com/sdsc-hpc-training-org/expanse-101.git
- CPU:
  - MPI
  - OpenMP
  - HYBRID
- GPU
- Large Memory Nodes

General Steps: Compiling/Running Jobs

- Change to a working directory (for example the expanse101 directory):
  cd /home/$USER/expanse101/MPI

- Verify that the correct modules are loaded:
  module list
  Currently Loaded Modulefiles:

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

- Compile the MPI hello world code:
  mpif90 -o hello_mpi hello_mpi.f90

- Verify executable has been created (check that date):
  ls -lt hello_mpi
  -rwxr-xr-x 1 user sdsc 721912 Mar 25 14:53 **hello_mpi**

- Submit job
  sbatch hello_mpi_slurm.sb

Outline

- Expanse Overview & Innovative Features
- Getting Started
- Modules
- Account Management
- Compiling and Linking Code
- Running Jobs
- Hands-on Examples
  - **MPI Jobs**
  - OpenMP Jobs
  - GPU/CUDA Jobs
  - Hybrid MPI-OpenMP Jobs
- Final Comments

MPI Hello World

- Change to the MPI examples directory:

```
[mthomas@login01 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
[mthomas@login01 MPI]$
```

MPI Hello World: Compile

Set the environment and
then compile the code

```
[mthomas@login01 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  --pty --account=use300 --nodes=1 --ntasks-per-node=128 --mem=248G -t
00:30:00 --wait=0 --export=ALL /bin/bash
```

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

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

[mthomas@login01 MPI]$ module purge
[mthomas@login01 MPI]$ module load slurm
[mthomas@login01 MPI]$ module load cpu
[mthomas@login01 MPI]$ module load gcc/10.2.0
[mthomas@login01 MPI]$ module load openmpi/4.0.4
[mthomas@login01 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

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

```
[mthomas@login01 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 -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_gnu

[mthomas@login01 MPI]$
```

```
[mthomas@login01 MPI]$ sbatch hellompi-slurm-gnu.sb; squeue -u mthomas
Submitted batch job 108910
       JOBID PARTITION    NAME    USER ST    TIME  NODES NODELIST(REASON)
      108910   compute hellompi  mthomas PD     0:00     2 (None)
[mthomas@login01 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!
```

Using An Interactive mode

```
[mthomas@login01 MPI]$ module purge
[mthomas@login01 MPI]$ module load slurm
[mthomas@login01 MPI]$ module load cpu
[mthomas@login01 MPI]$ module load gcc/10.2.0
[mthomas@login01 MPI]$ module load openmpi/4.0.4
[mthomas@login01 MPI]$ srun  --pty  --nodes=1  --ntasks-per-node=24 -p debug -t 00:30:00 --wait 0 /bin/bash
[mthomas@exp-9-55 MPI]$ module list
```

Request interactive node for 30 minutes

```
[mthomas@exp-9-55 MPI]$ mpirun -np 16 ./hello_mpi
node       1 : Hello world!
node      15 : Hello world!
node       7 : Hello world!
node      14 : Hello world!
node      11 : Hello world!
node       6 : Hello world!
node       4 : Hello world!
node       5 : Hello world!
node      12 : Hello world!
node      13 : Hello world!
node       0 : Hello world!
node       8 : Hello world!
node       9 : Hello world!
node      10 : Hello world!
node       2 : Hello world!
node       3 : Hello world!
```

- Exit interactive session when your work is done or you will be charged CPU time.
- Beware of oversubscribing your job: asking for more cores than you have. Intel compiler allows this, but your performance will be degraded.

Outline

- Expanse Overview & Innovative Features
- Getting Started
- Modules
- Account Management
- Compiling and Linking Code
- Running Jobs
- Hands-on Examples
  - GPU/CUDA Jobs
  - OpenMP Jobs
  - MPI Jobs
  - Hybrid MPI-OpenMP Jobs
  - Large Memory
- **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.

Thank You

Resources

- Expanse User Guide
  - https://www.sdsc.edu/support/user_guides/expanse.html
- GitHub Repo for this webinar: clone code examples for this tutorial – clone example code:
  - https://github.com/sdsc-hpc-training-org/expanse-101
- SDSC Training Resources
  - https://www.sdsc.edu/education_and_training/training
  - https://github.com/sdsc-hpc-training/webinars
- XSEDE Training Resources
  - https://www.xsede.org/for-users/training
  - https://cvw.cac.cornell.edu/expanse/