CIML Summer Institute:
Expanse CPU Computing
June 22, 2021

Mary Thomas
SDSC

EXPANSE
COMPUTING WITHOUT BOUNDARIES

SAN DIEGO SUPERCOMPUTER CENTER

NSF Award 1928224

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

Not covering today

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

Outline

- **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
```

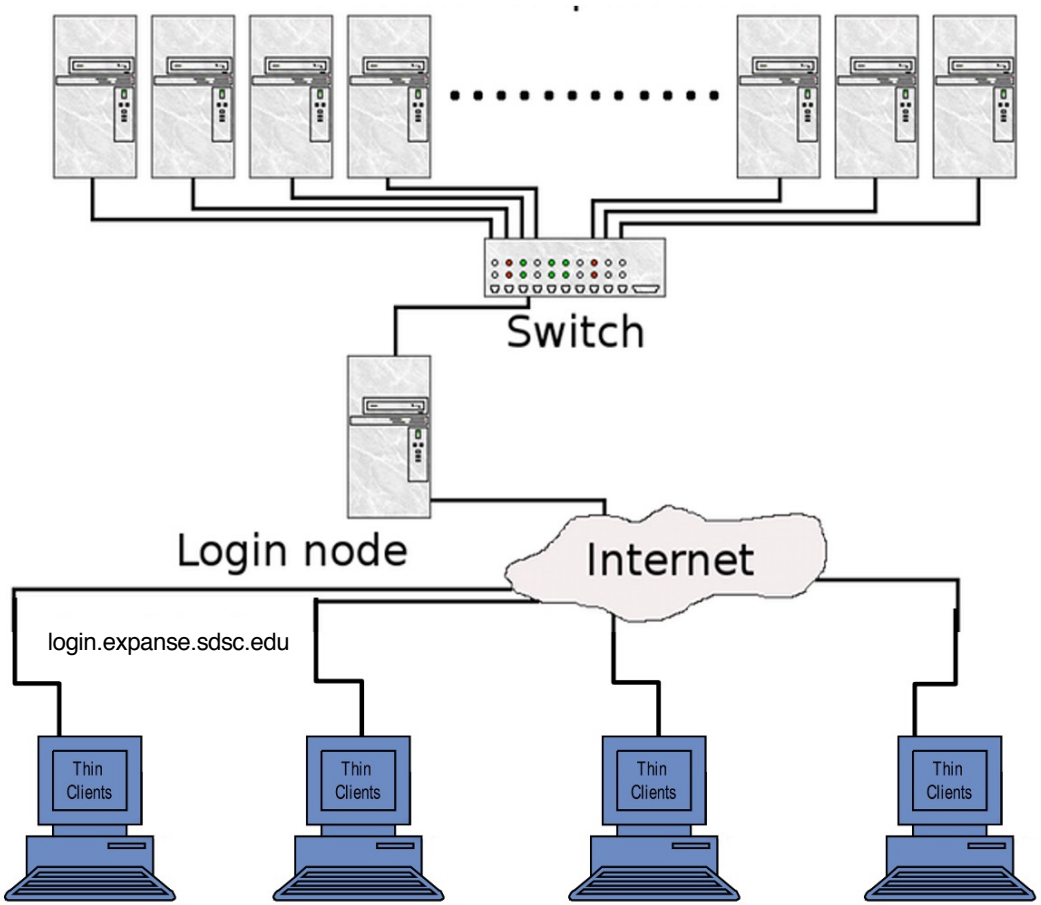
- When you log in to *login.expanse.sdsc.edu*, you will be assigned one of the two login nodes login0[1-2]-expanse.sdsc.edu. Both systems are identical.

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



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
Welcome to Bright release         9.0
                                                        Based on CentOS Linux 8
                                                                  ID: #000002
--------------------------------------------------------------------------------

                             WELCOME TO

                _______  __ ____  ___     _   _______ ______
               / ____/ |/ // __ \/   |  / | / / ___// ____/
              / __/  |   // /_/ / /| | /  |/ /\__ \/ __/
             / /___ /   |/ ____/ ___ |/ /|  /___/ / /___
            /_____//_/|_/_/   /_/  |_/_/ |_//____/_____/

--------------------------------------------------------------------------------
Use the following commands to adjust your environment:
'module avail'            - show available modules
'module add <module>'     - adds a module to your environment for this session
'module initadd <module>' - configure module to be loaded at every login
--------------------------------------------------------------------------------
Last login: Mon Jun 21 18:21:08 2021 from 76.176.117.51
[mthomas@login02 ~]$ whoami
mthomas
[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

- 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

[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
```

List Current environment: list, li

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

Module Command Examples

Use “module show” to find out what a particular module will change in the environment

```
[[mahidhar_test@login01 ~]$ module show cmake
-------------------------------------------------------------------------------------------------------------
   /cm/shared/apps/spack/cpu/lmod/linux-centos8-x86_64/Core/cmake/3.18.2.lua:
-------------------------------------------------------------------------------------------------------------
whatis("Name : cmake")
whatis("Version : 3.18.2")
whatis("Target : zen")
whatis("Short description : A cross-platform, open-source build system. CMake is a family of tools designed to build, test and package
software. ")
help([[A cross-platform, open-source build system. CMake is a family of tools
designed to build, test and package software.]])
prepend_path("PATH","/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/gcc-8.3.1/cmake-3.18.2-rfzatdti4qlsrf2zezwad75fnccy4f7d/bin")
prepend_path("ACLOCAL_PATH","/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/gcc-8.3.1/cmake-3.18.2-
rfzatdti4qlsrf2zezwad75fnccy4f7d/share/aclocal")
prepend_path("CMAKE_PREFIX_PATH","/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/gcc-8.3.1/cmake-3.18.2-
rfzatdti4qlsrf2zezwad75fnccy4f7d/")
setenv("CMAKEHOME","/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/gcc-8.3.1/cmake-3.18.2-rfzatdti4qlsrf2zezwad75fnccy4f7d")


[mahidhar_test@login01 ~]$
```

Module: check Environment

Once you have loaded the modules, you can check the system variables that are available for you to use.

```
[mahidhar_test@login01 ~]$ module load cmake
[mahidhar_test@login01 ~]$
[mahidhar_test@login01 ~]$ echo $PATH
/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/gcc-8.3.1/cmake-3.18.2-
rfzatdti4qlsrf2zezwad75fnccy4f7d/bin:/cm/shared/apps/slurm/current/sbin:/cm/shared/apps/slurm/
current/bin:/home/mahidhar_test/.local/bin:/home/mahidhar_test/bin:/usr/local/bin:/usr/bin:/usr/loc
al/sbin:/usr/sbin:/opt/dell/srvadmin/bin
[mahidhar_test@login01 ~]$
[mahidhar_test@login01 ~]$ echo $LD_LIBRARY_PATH
/cm/shared/apps/slurm/current/lib64/slurm:/cm/shared/apps/slurm/current/lib64
[mahidhar_test@login01 ~]$
[mahidhar_test@login01 ~]$ echo $CMAKEHOME
/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/gcc-8.3.1/cmake-3.18.2-
rfzatdti4qlsrf2zezwad75fnccy4f7d
[mahidhar_test@login01 ~]$
```

Module: command not found

- Sometimes encountered when switching from one shell to another or attempting to run the module command from within a shell script or batch job.
- Module command may not be inherited to the shell
- To keep this from happening, execute the following command:
  - command line (interactive shells)
    - source /etc/profile.d/modules.sh
  - OR add to your shell script (including Slurm batch scripts)

Outline

- 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 ~]$
```

Charging

- Charge unit for all SDSC machines, including Expanse, is the Service Unit (SU).
- Your charges are based on the resources that are tied up by your job, may not reflect how resources are used.
- Charges are based on either:
  - Number of cores
  - Fraction of the memory requested, whichever is larger.
- The minimum charge for any job is 1 SU. *This is important to note - can quickly use up SUs if you run a lot of very short jobs.*
- More details in Expanse user guide:

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

Outline

- 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

Using the AOCC Compilers

- If you have modified your environment, you can reload by executing the module purge & load commands at the Linux prompt, or placing the load command in your startup file (`~/.cshrc` or `~/.bashrc`)
- Note: The examples below are for the simple "hellompi" examples shown below

```
[mthomas@login02 ~]$ module list
Currently Loaded Modules:
  1) shared   2) cpu/1.0   3) DefaultModules   4) hdf5/1.10.1   5) intel/ 19.1.1.217
## need to change multiple modules
[mthomas@login02 ~]$ module purge
[mthomas@login02 ~]$ module list
No modules loaded
[mthomas@login02 ~]$ module load slurm
[mthomas@login02 ~]$ module load cpu
[mthomas@login02 ~]$ module load gcc
[mthomas@login02 ~]$ module load openmpi/4.0.4
[mthomas@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
[mthomas@login02 MPI]$ module swap intel aocc
Due to MODULEPATH changes, the following have been reloaded:
  1) openmpi/4.0.4
[mthomas@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
[mthomas@login02 ~]$
```

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

Using the Intel Compilers

- If you have modified your environment, you can reload by executing the module purge & load commands at the Linux prompt, or placing the load command in your startup file (`~/.cshrc` or `~/.bashrc`)

```
[mthomas@login02 ~]$ module list
[mthomas@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
[mthomas@login02 ~]$ module purge
[mthomas@login02 ~]$ module list
No modules loaded
[mthomas@login02 ~]$ module load slurm
[mthomas@login02 ~]$ module load cpu
[mthomas@login02 ~]$ module load intel
[mthomas@login02 ~]$ module load openmpi/4.0.4
[mthomas@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
[mthomas@login02 ~]$
```

Using the Intel Compilers

- For Intel compilers, to enable Advanced Vector Extensions (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

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

PGI Compilers: GPU Only

- PGI (formerly The Portland Group, Inc.), was a company that produced a set of commercially available Fortran, C and C++ compilers for high-performance computing systems.
- It is now owned by NVIDIA.
- To compile code, you need to obtain an interactive node.
- For AVX support, compile with -fast

```
[mahidhar_test@login01 ~]$ module reset
[mahidhar_test@login01 ~]$ module load gpu
[mahidhar_test@login01 ~]$ module load pgi
[mahidhar_test@login01 ~]$
[mahidhar_test@login01 ~]$ 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
[mahidhar_test@login01 ~]$ 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
```

- For more information on the PGI compilers run: man [pgf90 | pgcc | pgCC]

Recommended PGI Compilers

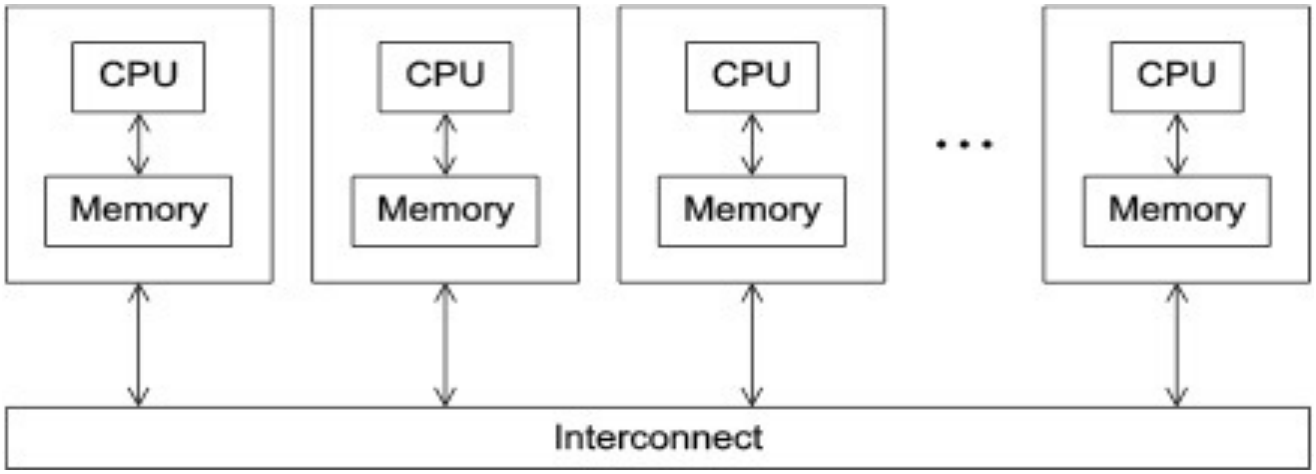
	Serial	MPI	OpenMP	MPI+OpenMP
Fortran	pgf90	mpif90	pgf90 -mp	mpif90 -mp
C	pgcc	mpicc	pgcc -mp	mpicc -mp
C++	pgCC	mpicxx	pgCC -mp	mpicxx -mp

- PGI 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

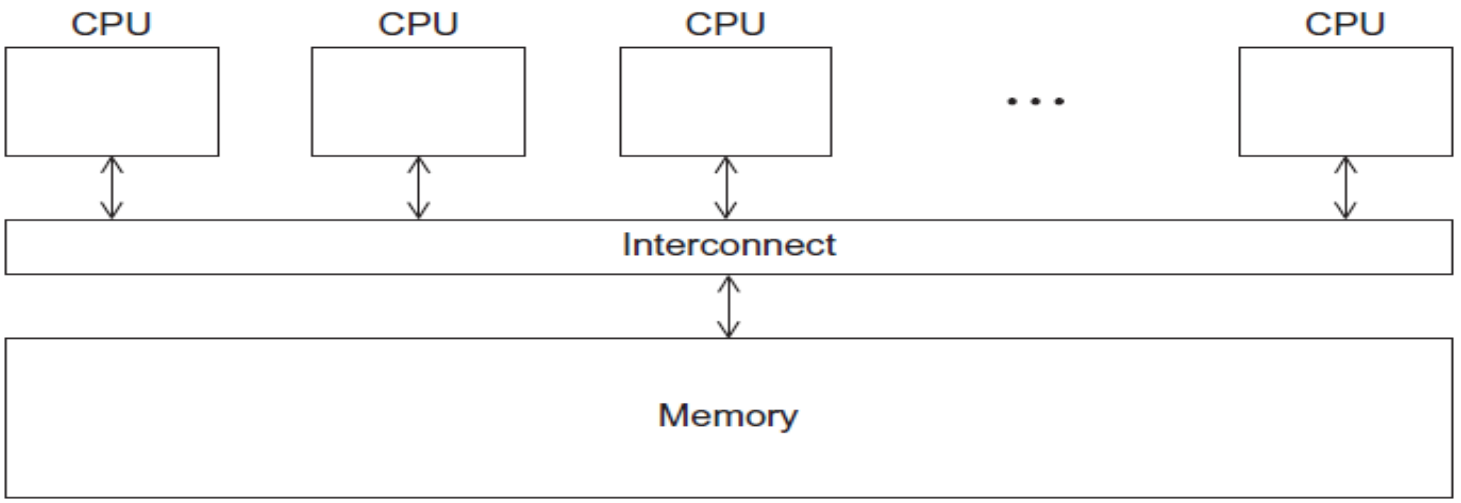
- Getting Started
- Modules
- Account Management
- Compiling and Linking Code
- **Running Jobs**
- Hands-on Examples
  - MPI Jobs
  - OpenMP Jobs
  - GPU/CUDA Jobs
  - Hybrid MPI-OpenMP Jobs
- Final Comments

Parallel Models: Distributed Memory



- Programs that 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

- 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
    ```

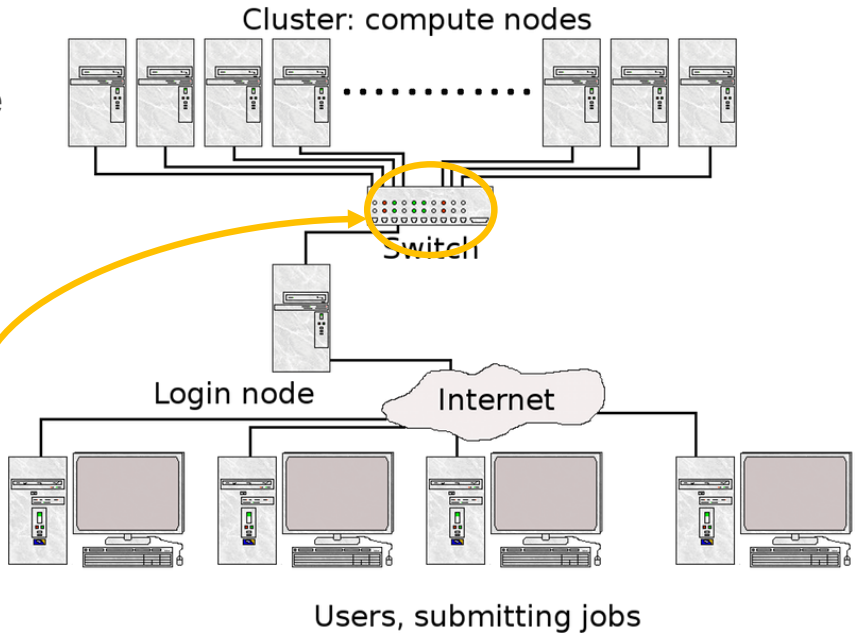
  - GPU:

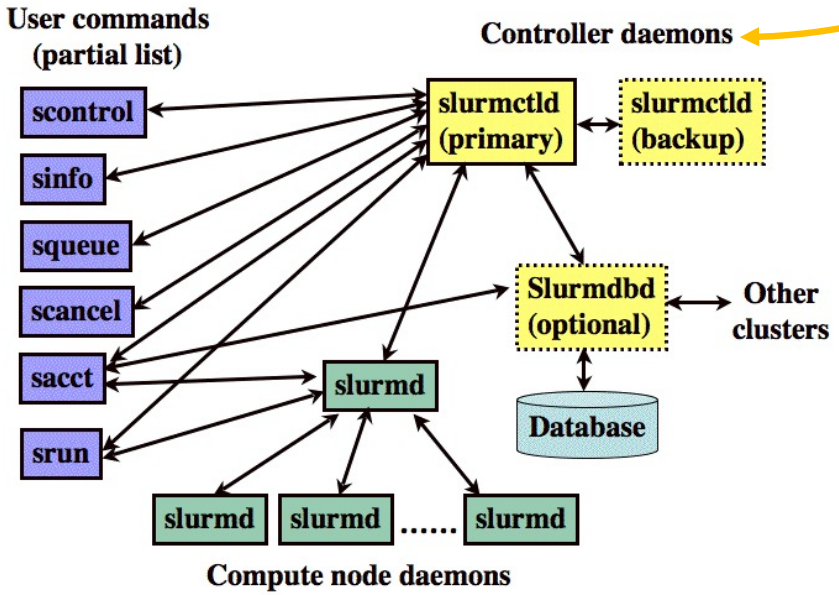
    ```
    srun --pty --account=XYZ123 --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 -p gpu-debug --gpus=1 -t 00:10:00 /bin/bash
    ```

Slurm 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

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 submited 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

Example Batch Script

```
[mthomas@login01 ENV_INFO]$ cat env-slurm.sb
#!/bin/bash
#SBATCH --job-name="envinfo"
#SBATCH --output="envinfo.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --export=ALL
#SBATCH -t 00:01:00

## Environment
module purge
module load slurm
module load cpu

##  perform some basic unix commands
echo "----------------------------------"
echo "hostname= " `hostname`
echo "date= " `date`
echo "whoami= " `whoami`
echo "pwd= " `pwd`
echo "module list= " `module list`
echo "----------------------------------"
echo "env= " `env`
echo "----------------------------------"
```

Simple batch script showing environment, date, etc.

```
[mthomas@login01 ENV_INFO]$ cat envinfo.108867.exp-6-56.out
---------------------------------
hostname=  exp-6-56
date=  Wed Oct 7 23:45:43 PDT 2020
whoami=  mthomas
pwd=  /home/mthomas/DEMO/ENV_INFO
Currently Loaded Modules:
  1) slurm/expanse/20.02.3   2) cpu/1.0
---------------------------------
env=  SLURM_MEM_PER_CPU=1024
LD_LIBRARY_PATH=/cm/shared/apps/slurm/current/lib64/slurm:/cm/shared/ap
ps/slurm/current/lib64 LS_COLORS=rs=0

[SNIP]

 MODULESHOME=/usr/share/lmod/lmod LMOD_SETTARG_FULL_SUPPORT=no
HISTSIZE=5000 LMOD_PKG=/usr/share/lmod/lmod
LMOD_CMD=/usr/share/lmod/lmod/libexec/lmod SLURM_LOCALID=0
LESSOPEN=||/usr/bin/lesspipe.sh %s LMOD_FULL_SETTARG_SUPPORT=no
LMOD_DIR=/usr/share/lmod/lmod/libexec BASH_FUNC_module%%=() { eval
$($LMOD_CMD bash "$@") && eval $(${LMOD_SETTARG_CMD:-:} -s sh) }
BASH_FUNC_ml%%=() { eval $($LMOD_DIR/ml_cmd "$@") } _=/usr/bin/env
---------------------------------
```

Outline

- 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

Hands On Examples

- Clone examples:
  - https://github.com/sdsc-hpc-training-org/expanse-101.git
- CPU:
  - OpenMP
  - MPI
  - 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 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
  - Hybrid MPI-OpenMP Jobs
  - GPU/CUDA 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 --account=sds184 --pty --nodes=1 --ntasks-per-
node=128 --mem=248 -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 --partition=debug --account=sds184 --pty --nodes=1 --ntasks-per-node=128 --mem=248 -t
00:30:00 --wait=0 --export=ALL /bin/bash
[mthomas@exp-9-55 MPI]$
```

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
  - MPI Jobs
  - **OpenMP Jobs**
  - Hybrid MPI-OpenMP Jobs
  - GPU/CUDA 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]$ 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
```

```
[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]$
```

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]$
```

```
[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
```

- 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.

OpenMP Hello World: submit job & monitor

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

```
[mthomas@login01 OPENMP]$ sbatch openmp-slurm-shared.sb ; squeue -u mthomas
Submitted batch job 108911
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            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

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

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

Expanse GPU Hardware

GPU Nodes	
GPU Type	NVIDIA V100 SMX2
Nodes	52
GPUs/node	4
CPU Type	Xeon Gold 6248
Cores/socket	20
Sockets	2
Clock speed	2.5 GHz
Flop speed	34.4 TFlop/s
Memory capacity	*384 GB DDR4 DRAM
Local Storage	1.6TB Samsung PM1745b NVMe PCIe SSD
Max CPU Memory bandwidth	281.6 GB/s

Using GPU Nodes

- GPU nodes are allocated as a separate resource. The conversion rate is (TBD) Expanse Service Units (SUs) to 1 V100 GPU-hour.
- Login nodes are not the same as the GPU nodes:
  - → GPU codes must be compiled by requesting an interactive session on a GPU nodes.
- Batch: GPU nodes can be accessed via either the "gpu" or the "gpu-shared" partitions.
  - #SBATCH -p gpu
  - or #SBATCH -p gpu-shared
- Interactive GPU node:
  - srun --pty --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 -p gpu-debug --gpus=1 -t 00:10:00 /bin/bash

GPU/CUDA: Interactive Node

- Change to the OpenACC directory

```
[mthomas@exp-7-59 OpenACC]$ ll
total 71
-rw-r--r-- 1 mthomas use300  2136 Oct  7 11:28 laplace2d.c
-rwxr-xr-x 1 mthomas use300 52056 Oct  7 11:28 laplace2d.openacc.exe
-rw-r--r-- 1 mthomas use300   234 Oct  7 11:28 OpenACC.108739.exp-7-57.out
-rw-r--r-- 1 mthomas use300   307 Oct  8 00:21 openacc-gpu-shared.sb
-rw-r--r-- 1 mthomas use300  1634 Oct  7 11:28 README.txt
-rw-r--r-- 1 mthomas use300  1572 Oct  7 11:28 timer.h
```

- Obtain an interactive node:

```
[mthomas@login01 OpenACC]$ srun   --pty   --nodes=1   --ntasks-per-node=1   --
cpus-per-task=10   -p gpu-debug  --gpus=1  -t 00:10:00 /bin/bash
```

GPU/CUDA: Node Information

Check node configuration:

```
[mthomas@exp-7-59 OpenACC]$ nvidia-smi
Thu Oct  8 03:58:44 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.51.05    Driver Version: 450.51.05    CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:18:00.0 Off |                    0 |
| N/A   32C    P0    41W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
[mthomas@exp-7-59 OpenACC]$
```

GPU: Compile on Interactive node

```
[mthomas@login01 OpenACC]$
cat README.txt
[1] Compile Code:
(a) Get an interactive GPU debug node:
module load slurm
srun --pty --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 -p gpu-debug --gpus=1 -t 00:10:00 /bin/bash

(b) On the GPU node:
module purge
module load slurm
module load gpu
module load pgi
pgcc -o laplace2d.openacc.exe -fast -Minfo -acc -ta=tesla:cc70 laplace2d.c

Compiler output:
GetTimer:
     20, include "timer.h"
          61, FMA (fused multiply-add) instruction(s) generated
laplace:
     47, Loop not fused: function call before adjacent loop
         Loop unrolled 8 times
         FMA (fused multiply-add) instruction(s) generated
     55, StartTimer inlined, size=2 (inline) file laplace2d.c (37)

[SNIP]
          75, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */
         77, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */
     88, GetTimer inlined, size=9 (inline) file laplace2d.c (54)
(Exit out of debug node after this)

[2] Run job:
sbatch openacc-gpu-shared.sb
```

GPU: Submit Batch Script on CPU node

```
[mthomas@login01 OpenACC]$ cat openacc-gpu-shared.sb
#!/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 --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

[mthomas@login01 OpenACC]$ sbatch openacc-gpu-shared.sb
[mthomas@login01 OpenACC]$ sbatch openacc-gpu-shared.sb ; squeue -u mthomas
Submitted batch job 108915
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            108915 gpu-share  OpenACC  mthomas PD       0:00      1 (None)
```

```
[mthomas@login01 OpenACC]$ cat OpenACC.108915.exp-7-57.out
main()
Jacobi relaxation Calculation: 4096 x 4096 mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 1.084470 s
[mthomas@login01 OpenACC]$
```

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**

Yes, You are Correct: Running Jobs on HPC Systems is Complex

- Multiple layers of hardware and software affect job performance
- Learn to develop and test in a modular fashion
- Build up a suite of test cases:
  - When things go wrong, make sure you can run simple test cases (HelloWorld).
  - This can eliminate questions about your environment.
- Consider using a code repository
  - When things go wrong, you can get back to a working version
- If you need help/have questions, contact XSEDE help desk:
  - **They are very helpful and respond quickly**
  - Support users around the world, so they are truly a 7/24 service
  - Avoid wasting your time.

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/