

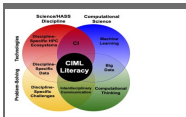
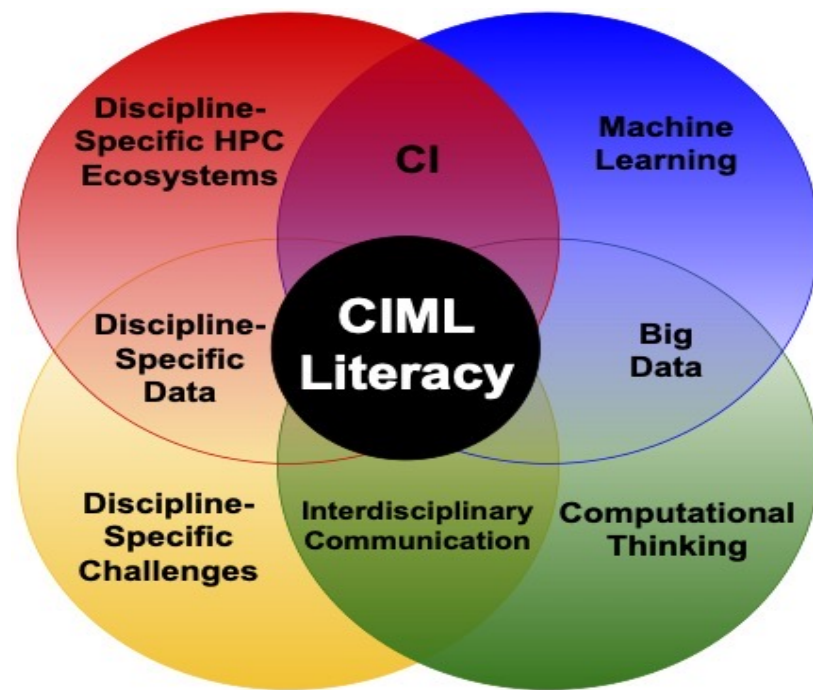
CIML Summer Institute, Day 1

Prep Day: Running Jobs

June 22, 2022

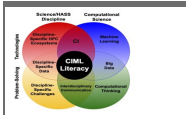
Mary Thomas

San Diego Supercomputer Center

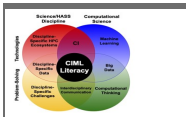


Outline

- Compiling and Linking Code
- Running Jobs
- Additional Examples
 - CPU Jobs
 - GPU/CUDA Jobs
- Final Comments



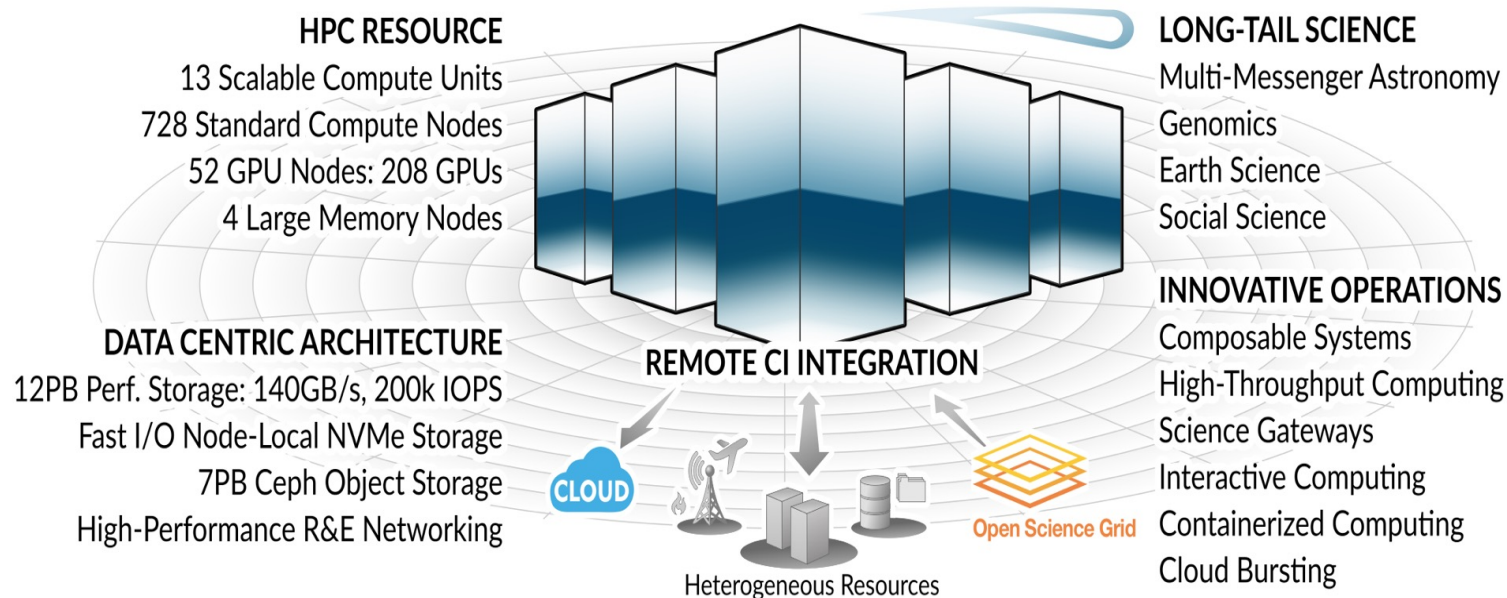
Expanse



SDSC SAN DIEGO
SUPERCOMPUTER CENTER
UC San Diego

EXPANSE

COMPUTING WITHOUT BOUNDARIES
5 PETAFL0P/S HPC and DATA RESOURCE



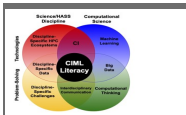
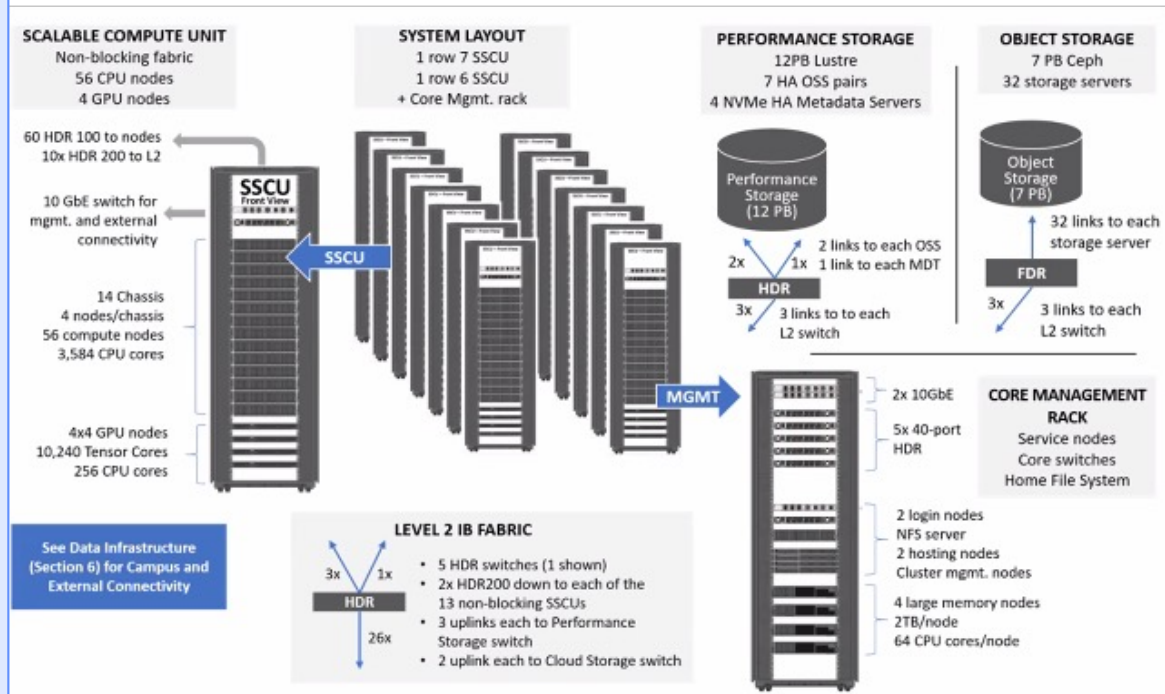
For more details see the Expanse user guide @ https://www.sdsc.edu/support/user_guides/expanse.html
and the "Introduction to Expanse" webinar @ https://www.sdsc.edu/event_items/202006_Introduction_to_Expanse.html



Expanse Heterogeneous Architecture

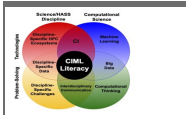
System Summary

- 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



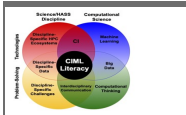
Basic Information

- 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](#)
 - A 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://hpc-training.sdsc.edu/expanse-101/>



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

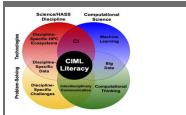
Language	Serial	MPI	OpenMP	MPI + OpenMP
Fortran	flang	mpif90	ifort -openmp	mpif90 -openmp
C	clang	mpiclang	icc -openmp	mpicc -openmp
C++	clang++	mpiclang	icpc -openmp	mpicxx -openm

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:

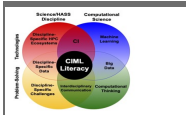
```
$ [flang | clang] -help
```



Using the AOCC Compilers

- If you have modified your environment, you can 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) shared  2) cpu/1.0  3) DefaultModules  4) hdf5/1.10.1  5) intel/ 19.1.1.217
## need to change multiple modules
[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 aocc
[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) aocc/2.2.0  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 ~]$
```



Intel Compilers: GPU and GPU

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

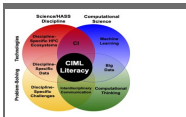
	Serial	MPI	OpenMP	MPI + OpenMP
Fortran	ifort	mpif90	ifort -openmp	mpif90 -openmp
C	icc	mpicc	icc -openmp	mpicc -openmp
C++	icpc	mpicxx	icpc -openmp	mpicxx -openmp

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
```

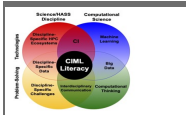


Using the Intel Compilers

- 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

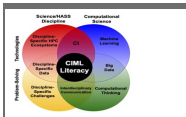
```
[user@login02 ~]$ module list
[user@login02 MPI]$ module list
Currently Loaded Modules:
  1) slurm/expansive/20.02.3  2) cpu/1.0  3) aocc/2.2.0  4)
openmpi/4.0.4
[user@login02 ~]$ module purge
[user@login02 ~]$ module list
No modules loaded
[user@login02 ~]$ module load slurm
[user@login02 ~]$ module load cpu
[user@login02 ~]$ module load intel
[user@login02 ~]$ module load openmpi/4.0.4
[user@login02 ~]$ module list
Currently Loaded Modules:
  1) slurm/expansive/20.02.3  2) cpu/1.0  3) aocc/2.2.0  4)
openmpi/4.0.4
[user@login02 ~]$
```

```
[user@login02 MPI]$ module list
Currently Loaded Modules:
  1) slurm/expansive/20.02.3  2) cpu/1.0  3) aocc/2.2.0  4) openmpi/4.0.4
[user@login02 ~]$ module swap aocc intel
Due to MODULEPATH changes, the following have been reloaded:
  1) openmpi/4.0.4
[user@login02 ~]$ module list
Currently Loaded Modules:
  1) slurm/expansive/20.02.3  2) cpu/0.15.4  3) intel/19.1.1.217  4) openmpi/4.0.4
[user@login02 ~]$ [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
```



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>



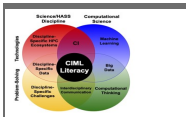
PGI Compilers

- 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**

	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

- For more information on the PGI compilers run:

\$ man [pgf90 | pgcc | pgCC]

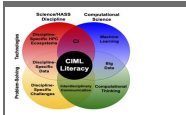


PGI Compilers: GPU Only

```
[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-
2tsjnv2icisxmgdy4mjl4t5mkbr32ea/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-
2tsjnv2icisxmgdy4mjl4t5mkbr32ea/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



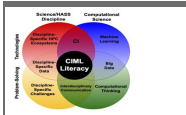
GNU Compilers: CPU & GPU

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 -mavx.
- 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++]`

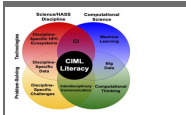


GNU Compilers

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

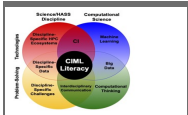
```
[user@login02]$ module purge
[user@login02]$ module load slurm
[user@login02]$ module load cpu
[user@login02]$ module load gcc/10.2.0
[user@login02]$ module load openmpi/4.0.4
[user@login02]$ module list
Currently Loaded Modules:

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

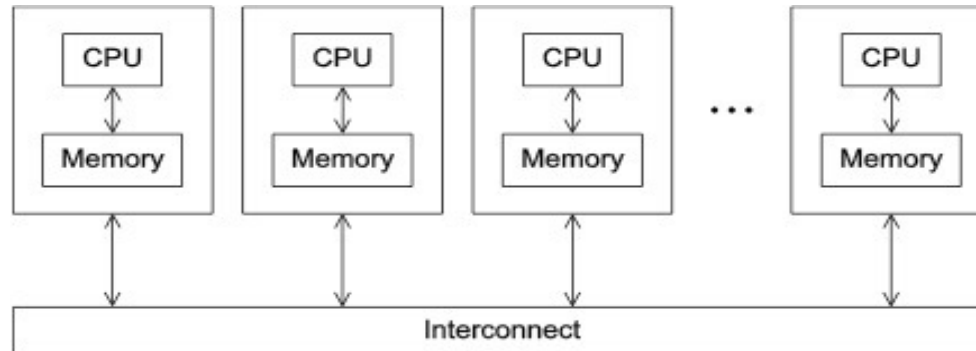


Outline

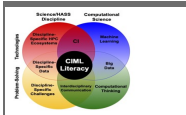
- **Introduction**
- Compiling and Linking Code
- **Running Jobs**
- Additional Examples
 - CPU Jobs
 - GPU/CUDA Jobs
- Final Comments



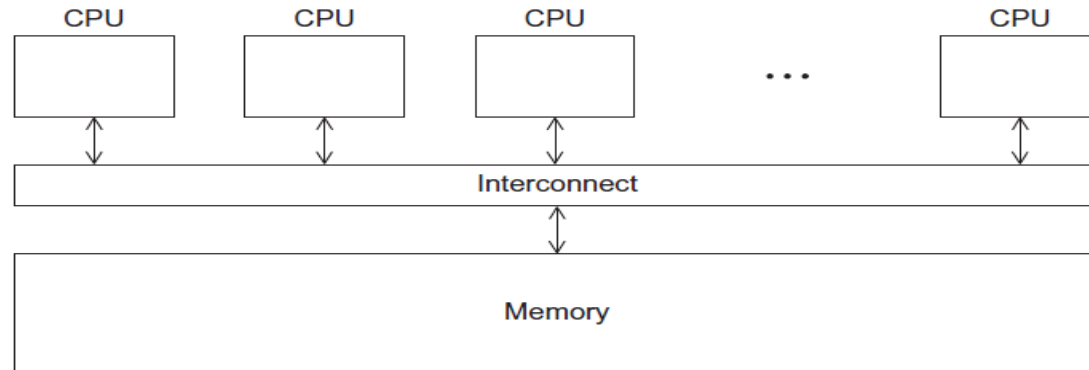
Parallel Models: Distributed Memory



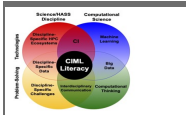
- Programs that run asynchronously, pass messages for communication and coordination between resources.
- Examples include: SOA-based systems, massively multiplayer online games, peer-to-peer apps.
- Different types of implementations for the message passing mechanism: HTTP, RPC-like connectors, message queues
- HPC historically uses the **Message Passing Interface (MPI)**



Parallel Models: Shared Memory



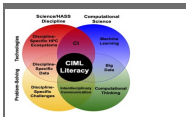
- CPUs all share same localized memory (SHMEM);
 - Coordination and communication between tasks via interprocessor communication (IPC) or virtual memory mappings.
- May use: uniform or non-uniform memory access (UMA or NUMA); cache-only memory architecture (COMA).
- Most common HPC API's for using SHMEM:
 - Portable Operating System Interface (**POSIX**); Open Multi-Processing (**OpenMP**) designed for parallel computing – best for **multi-core computing**.



Running Jobs on Expanse

A few observations:

- 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 Running Jobs on Expanse

- **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

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

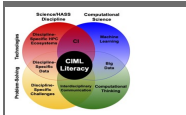
- **Interactive Jobs:** Use the *sr*un command to request 'live' nodes from Slurm for command line, *inter*active access

- **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:**

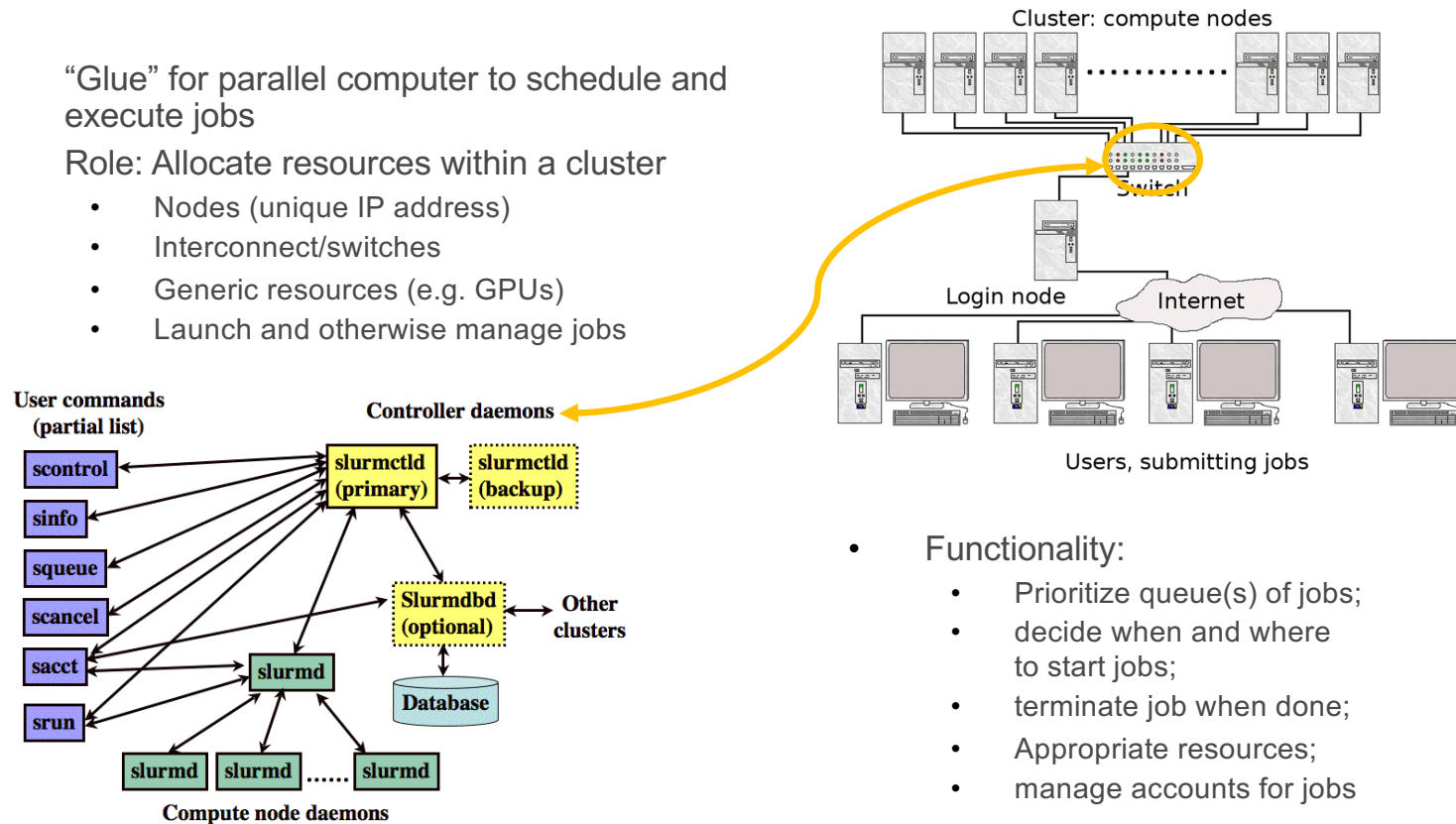
```
srun --pty --account=XYZ123 --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 -p gpu-debug --gpus=1 -t 00:10:00 /bin/bash
```



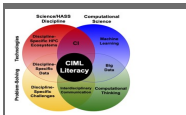
Slurm Resource Manager

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



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

SDSC SAN DIEGO
SUPERCOMPUTER CENTER
UC San Diego

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 it's state:

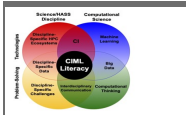
```
$ squeue -u $USER
```

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>



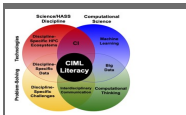
Slurm Partitions on Expanse

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

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; <i>limit applies per group</i>
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; <i>max two gpus per job</i>
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

Last updated 9/14/21

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



Example Batch Script

```
[uswename@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

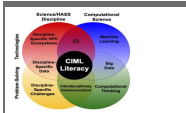
#This job runs with 1 node, 24 cores per node for a total of 24 cores.
## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

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

## Use srun to run the job, pass variable to code
srun --mpi=pmi2 -n 24 --cpu-bind=rank ./mpi_prime
```

Simple batch script showing environment, date, etc.

```
[mthomas@login02 calc-prime]$ sbatch --export=NLO=1000 mpi-prime-slurm.sb
Submitted batch job 9113467
[mthomas@login02 calc-prime]$ lsq
squeue -u mthomas
      JOBID PARTITION  NAME  USER ST  TIME  NODES NODELIST(REASON)
      9113467  debug mpi_prim mthomas R   0:04    1 exp-9-55
```



Output: mpi-prime

28 January 2022 12:21:30 PM

PRIME_MPI

n_hi= 262144

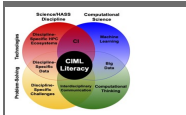
C/MPI version

An MPI example program to count the number of primes.

The number of processes is 24

N	Pi	Time
1	0	0.025196
2	1	0.001954
4	2	0.024015
8	4	0.025980
16	6	0.025996
32	11	0.026022
64	18	0.025977
128	31	0.025999
256	54	0.025999
512	97	0.026009
1024	172	0.013043
2048	309	0.012683
4096	564	0.028262
8192	1028	0.025999
16384	1900	0.025998
32768	3512	0.041243
65536	6542	0.213824
131072	12251	0.676576
262144	23000	1.570949

PRIME_MPI - Master process: Normal end of execution.



SLURM Environment Variables

<https://slurm.schedmd.com/sbatch.html#lBAJ>

Internal ENV vars that exist when job submitted:

INPUT ENVIRONMENT VARS

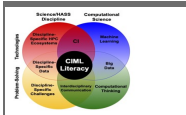
<https://slurm.schedmd.com/sbatch.html#lBAJ>

- 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

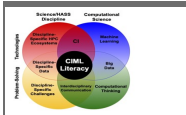
<https://slurm.schedmd.com/sbatch.html#lBAK>

- 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

- Passing variables into a SLURM job when you submit the job
- Use the **--export flag**.
- Examples
 - 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

```
[uswename@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

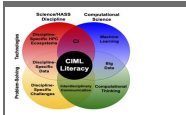
#This job runs with 1 node, 24 cores per node for a total of 24 cores.
## Environment
module purge
module load slurm
module load cpu
module load gcc/10.2.0
module load openmpi/4.0.4

## 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 Y
```

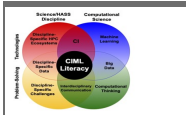
Batch script showing
environment, date, and passing
variable

```
[mthomas@login02 calc-prime]$ sbatch --export=NLO=1000 mpi-prime-slurm.sb
Submitted batch job 9113467
[mthomas@login02 calc-prime]$ !sq
queue -u mthomas
      JOBID PARTITION  NAME  USER ST   TIME  NODES NODELIST(REASON)
      9113467   debug mpi_prim mthomas R    0:04    1 exp-9-55
```



What is Interactive HPC-Computing

- In **computer** science, **interactive computing** refers to software which accepts input from the user as it runs.
 - commonly used programs, such as word processors or spreadsheet applications.
- **Interactive HPC computing** involves *real-time* user inputs to perform tasks on a set of compute node(s) including:
 - Code development, real-time data exploration, and visualizations.
 - Used when applications have large data sets or are too large to download to local device, software is difficult install, etc.
 - User inputs come via command line interface or application GUI (Jupyter Notebooks, Matlab, R-studio).
 - Actions performed on remote compute nodes as a result of user input or program out.
- Expanse supports running applications via command line and Web-based services (Jupyter Notebooks, Matlab, R-studio)

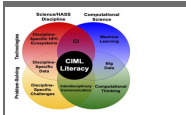


Running Interactive Jobs - CPU

Obtaining an interactive CPU node with 1 node, 128 cores, and 249GB:

```
[user@login01 ~]$ srun --partition=debug --account=abc123 --pty --nodes=1 --ntasks-per-node=128
--mem=248 -t 00:30:00 --wait=0 --export=ALL /bin/bash
srun: job 5825986 queued and waiting for resources
srun: job 5825986 has been allocated resources
[user@exp-9-55 ~]$ hostname
exp-9-55
[user @exp-9-55 ~]$

[user@exp-9-55 ~]$ module list
Currently Loaded Modules:
  1) slurm/exppanse/20.02.3  2) cpu/0.15.4  3) intel/19.1.1.217  4) openmpi/4.0.4
[user@exp-9-55 ~]$ module purge
modul[user@exp-9-55 ~]$ module restore
Restoring modules from user's default
[user @exp-9-55 ~]$ module list
Currently Loaded Modules:
  1) cpu/0.15.4  2) slurm/exppanse/20.02.3  3) intel/19.1.1.217
[user @exp-9-55 ~]$ exit
[user@login01 ~]$
```



Running Interactive Jobs - GPU

The following example will request a GPU node, 10 cores, 1 GPU and 96G in the debug partition for 30 minutes. To ensure the GPU environment is properly loaded, please be sure run both the module purge and module restore commands.

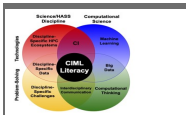
```
[user@login02 ~]$ srun --partition=gpu-debug --pty --account=abc123 --ntasks-per-node=10
--nodes=1 --mem=96G --gpus=1 -t 00:30:00 --wait=0 --export=ALL /bin/bash
srun: job 5826039 queued and waiting for resources
srun: job 5826039 has been allocated resources
[user@exp-7-59 ~]$
[user@exp-7-59 ~]$ hostname
exp-7-59
[user@exp-7-59 ~]$ module purge
[user@exp-7-59 ~]$ module restore
Restoring modules from user's default
[user@exp-7-59 ~]$
```

```
[user@exp-7-59 ~]$ nvidia-smi
Thu Sep 16 00:09:41 2021
```

NVIDIA-SMI 460.32.03 Driver Version: 460.32.03 CUDA Version: 11.2									
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			

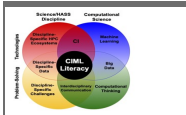
```
Processes:
```

GPU	GI	CI	PID	Type	Process name	GPU Memory
ID	ID	ID				Usage



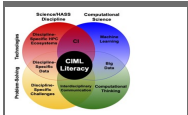
General Steps: Compiling/Running Jobs

- 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`



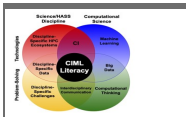
Outline

- **Introduction**
- Compiling and Linking Code
- Running Jobs
- Additional Examples
 - CPU Jobs
 - GPU/CUDA Jobs
- Final Comments



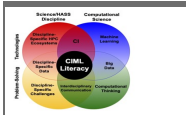
Hands-on Examples

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



Outline

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

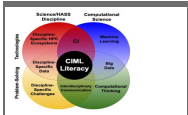


MPI Hello World

- 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/expense/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
```

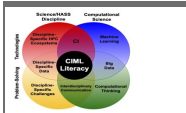
```
[user@login02 MPI]$ module list
```

Currently Loaded Modules:

1) slurm/expense/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
```

#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
```

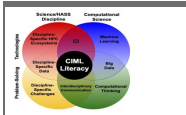
JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(Reason)
108910	compute	hellompi	user	PD	0:00	2	(None)

```
[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!
```



Using An Interactive mode

```
[user@login02 MPI]$ srun --partition=debug --account=sds18  
t 00:30:00 --wait=0 --export=ALL /bin/bash
```

Request
interactive
node for 30
minutes

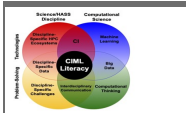
```
--ntasks-per-node=128 --mem=248 -
```

```
[user@exp-9-55 MPI]$  
[user@ exp-9-55 MPI]$ module purge  
[user@ exp-9-55 MPI]$ module load slurm  
[user@ exp-9-55 MPI]$ module load cpu  
[user@ exp-9-55 MPI]$ module load gcc/10.2.0  
[user@ exp-9-55 MPI]$ module load openmpi/4.0.4
```

```
[user@exp-9-55 MPI]$ mpif90 -o hello_mpi_f_gnu hello_mpi.f90  
[user@exp-9-55 MPI]$ mpirun -np 8 ./hello_mpi_f_gnu
```

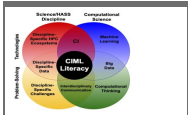
```
node      1 : Hello world!  
node     15 : Hello world!  
node       7 : Hello world!  
node     14 : Hello world!  
node     11 : Hello world!  
node       6 : Hello world!  
node       4 : Hello world!  
node       5 : Hello world!  
node     12 : Hello world!  
node     13 : Hello world!  
node       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.



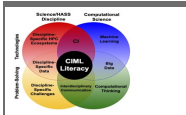
Outline

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



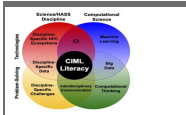
Expanse GPU Hardware

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.
- Login nodes are not the same as the GPU nodes:
→ GPU codes must be compiled by requesting an interactive session on a GPU node.
- 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:
 - `srn --partition=gpu-debug --pty --account=<project> --ntasks-per-node=10 --nodes=1 --mem=96G --gpus=1 -t 00:30:00 --wait=0 --export=ALL /bin/bash`



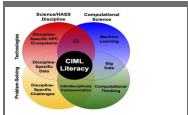
GPU/CUDA: Interactive Node

- Change to the **openacc** directory

```
[user@exp-7-59 openacc]$ ll
total 71
-rw-r--r-- 1 user abc123 2136 Oct 7 11:28 laplace2d.c
-rwxr-xr-x 1 user abc123 52056 Oct 7 11:28 laplace2d.openacc.exe
-rw-r--r-- 1 user abc123 234 Oct 7 11:28 OpenACC.108739.exp-7-57.out
-rw-r--r-- 1 user abc123 307 Oct 8 00:21 openacc-gpu-shared.sb
-rw-r--r-- 1 user abc123 1634 Oct 7 11:28 README.txt
-rw-r--r-- 1 user abc123 1572 Oct 7 11:28 timer.h
```

- Obtain an interactive node:

```
[user@login02 openacc]$ srun --partition=gpu-debug --pty --account=<project> --ntasks-per-node=10 --
nodes=1 --mem=96G --gpus=1 -t 00:30:00 --wait=0 --export=ALL /bin/bash
```



GPU/CUDA: Node Information

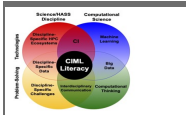
Check node configuration:

```
[user@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          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 |
+-----+

[user@exp-7-59 OpenACC]$
```

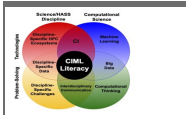


GPU: Compile on interactive node

```
[user@login02 OpenACC]$
cat README.txt
[1] Compile Code:
(a) Get an interactive GPU debug node:
module load slurm
srun --partition=gpu-debug --pty --account=abc123 --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 --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

```
[user@login02 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 -A abc123
```

```
#SBATCH -t 01:00:00
```

```
#Environment
```

```
module purge
```

```
module load slurm
```

```
module load gpu
```

```
module load pgi
```

```
#Run the job
```

```
./laplace2d.openacc.exe
```

```
[user@login02 OpenACC]$ sbatch openacc-gpu-shared.sb
```

```
[user@login02 OpenACC]$ sbatch openacc-gpu-shared.sb ; queue -u user
```

```
Submitted batch job 108915
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(REASON)
108915	gpu-share	OpenACC	user	PD	0:00	1	(None)

```
[user@login02 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
```

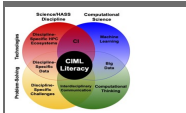
```
700, 0.000345
```

```
800, 0.000302
```

```
900, 0.000269
```

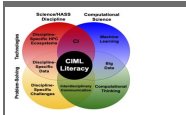
```
total: 1.084470 s
```

```
[user@login02 OpenACC]$
```



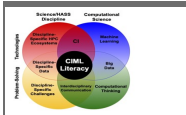
Useful Links

- Expanse User Guide:
 - https://www.sdsc.edu/support/user_guides/expanse.html
- Expanse Overview:
 - https://education.sdsc.edu/training/interactive/202201_parallel_computing_concepts/index.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](#)
 - A 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/>



Resources

- Expanse User Guide & Tutorial
 - https://www.sdsc.edu/support/user_guides/expanse.html
 - <https://hpc-training.sdsc.edu/expanse-101/>
- Clone code examples for this tutorial:
 - <https://github.com/sdsc-hpc-training-org/expanse-101>
- 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

