Using the Frontier Supercomputer

SC23
Denver, CO | i am hpc.

All
User Assistance - OLCF
Oak Ridge National Lab.

Outline

- Programming Environment
- Parallel Programming Models
- Resource Scheduler & Parallel Job Launcher

Programming Environment

What is the Programming Environment?

At the highest level, the PE is your shell's build- and run-time environment.

- Compilers, compiler wrappers, tools, scientific libraries, runtimes, etc.

OLCF offers many software packages that are managed through session environment variables.

- Search paths
  - `PATH, LD_LIBRARY_PATH, LIBRARY_PATH, PKG_CONFIG_PATH, …`
    - See output of running `env`
- Program options via environment variables
  - `OMPI_*, CC, FC, …`

Much of the available software cannot coexist simultaneously in your environment.

LMOD Environment Modules

Frontier runs LMOD to manage environment complexity

Build- and runtime-environment software managed with LMOD

- https://lmod.readthedocs.io

Usage:

```
$ module -t list                  # list loaded modules
$ module avail                    # show modules that can be loaded given current env
$ module help <package>           # help info for package (if provided)
$ module show <package>           # show environment changes made by module
$ module load <package> <package>...     # add package(s) to environment
$ module unload <package> <package>...   # remove package(s) from environment
$ module reset                    # restore system defaults
$ module restore <collection>     # load a saved collection
$ module spider <package>                 # deep search for modules
$ module purge                    # clear all modules from environment
```

Compilers on Frontier

Vendor	Programming Environment	Compiler Module	Language	Compiler Wrapper	Compiler
Cray	PrgEnv-cray	cce	C	cc	craycc
			C++	CC	craycxx or crayCC
			Fortran	ftn	crayftn
AMD	PrgEnv-amd	amd	C	cc	amdclang
			C++	CC	amdclang++
			Fortran	ftn	amdflang
GCC	PrgEnv-gnu	gcc	C	cc	${GCC_PATH}/bin/gcc
			C++	CC	${GCC_PATH}/bin/g++
			Fortran	ftn	${GCC_PATH}/bin/gfortran

LMOD Environment Modules

- Use **module load** to load to add software packages that are compatible with your environment.

```
[user@login1.frontier ~]$ module -t list

craype-x86-trento
libfabric/1.15.2.0
craype-network-ofi
perftools-base/22.12.0
xpmem/2.6.2-2.5_2.22__gd067c3f.shasta
cray-pmi/6.1.8
cce/15.0.0
craype/2.7.19
cray-dsmml/0.2.2
cray-mpich/8.1.23
cray-libsci/22.12.1.1
PrgEnv-cray/8.3.3
darshan-runtime/3.4.0
hsi/default
DefApps/default
```

LMOD Environment Modules

- Use **module load** to load to add software packages that are compatible with your environment.

```
[user@login1.frontier ~]$ module load PrgEnv-amd

Lmod is automatically replacing "cce/15.0.0" with "amd/5.3.0".

Lmod is automatically replacing "PrgEnv-cray/8.3.3" with "PrgEnv-amd/8.3.3".

Due to MODULEPATH changes, the following have been reloaded:
  1) cray-mpich/8.1.23
```

LMOD Environment Modules

- Use **module load** to load to add software packages that are compatible with your environment.

```
[user@login1.frontier ~]$ module -t list

craype-x86-trento
libfabric/1.15.2.0
craype-network-ofi
perftools-base/22.12.0
xpmem/2.6.2-2.5_2.22__gd067c3f.shasta
cray-pmi/6.1.8
amd/5.3.0
craype/2.7.19
cray-dsmml/0.2.2
cray-mpich/8.1.23
cray-libsci/22.12.1.1
PrgEnv-amd/8.3.3
darshan-runtime/3.4.0
hsi/default
DefApps/default
```

LMOD Environment Modules

If you need to add the tools and libraries related to ROCm, the framework for targeting AMD GPUs, to your path, you will need to use a version of ROCm that is compatible with your programming environment.

Programming Environment Module	Module that gets you ROCm Toolchain	How you load it:
`PrgEnv-amd`	`amd`	amd is loaded automatically with `module load PrgEnv-amd`
`PrgEnv-cray` or `PrgEnv-gnu`	`amd-mixed`	`module load amd-mixed`

You will also need the CrayPE Accelerator Module for most GPU related tasks.

```
[user@login1.frontier ~]$ module load craype-accel-amd-gfx90a
```

Additional information about the Programming Environment can be found on the OLCF User Documentation.

OAK RIDGE National Laboratory

Parallel Programming Models

November 13, 2023

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

High Performance Computing

High Performance Computing (HPC) is about doing work efficiently in parallel.

Profiling and Optimizing your Laundry Day Workflow

Just you and one washer and one dryer

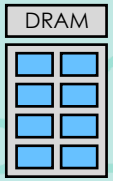
Optimizing just you and one washer and one dryer

You your housemate, 2 washers, and 2 dryers

Parallel Programming Models

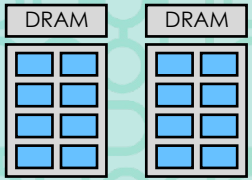
A =

```
for(int i=0; i<N; i++){

    B[i] = A[i]*A[i]

}
```



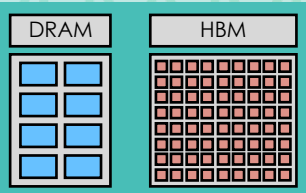
Shared-memory models

- E.g., OpenMP
  - All process threads can access same memory (single-node)



Distributed-memory models

- E.g., Message Passing Interface (MPI)
  - All processes (i.e., MPI ranks) have access to their own memory (multi-node)



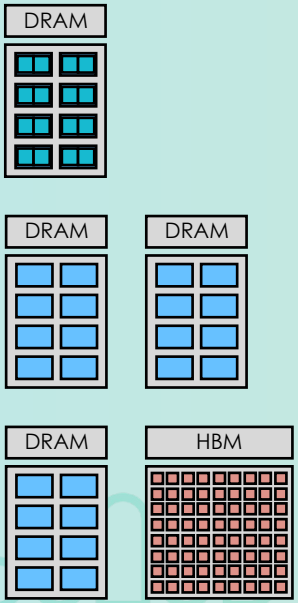
GPU

- CUDA, HIP
- OpenACC, OpenMP offload (directive-based models)
- Kokkos (portability)

Parallel Programming Models

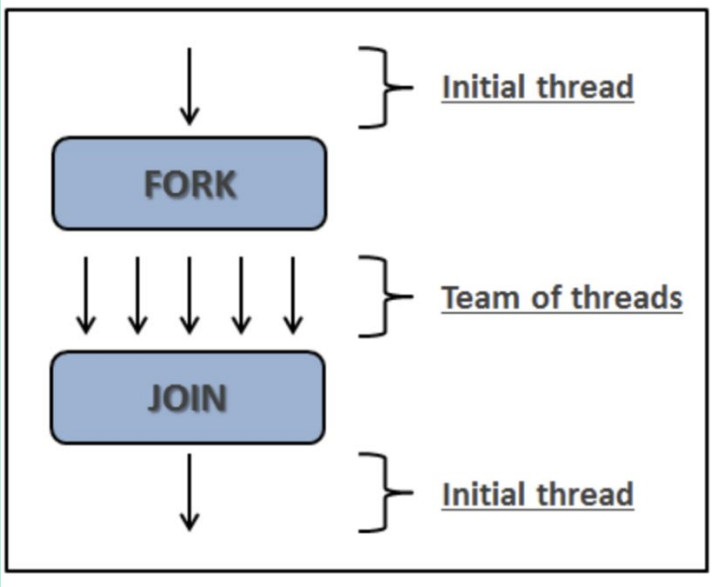
A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



Shared-memory models

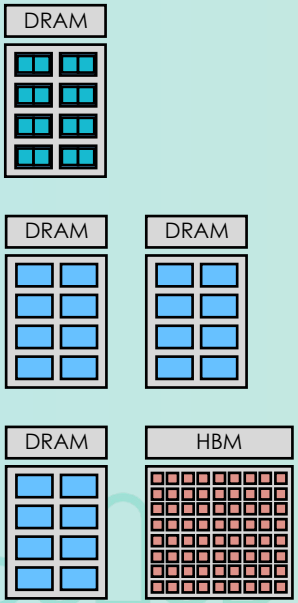
- E.g., OpenMP



Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



Shared-memory models

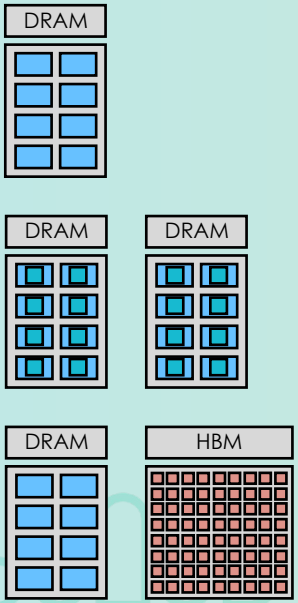
- E.g., OpenMP

```
#pragma omp parallel default(none) shared(A, B)
    {
        #pragma omp for
         for(int i=0; i<N; i++){
           B[i] = A[i]*A[i]
         }
    }
```

Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



Shared-memory models

- E.g., OpenMP
  - All process threads can access same memory (single-node)

Distributed-memory models

- E.g., Message Passing Interface (MPI)
  - All processes (i.e., MPI ranks) have access to their own memory (multi-node)

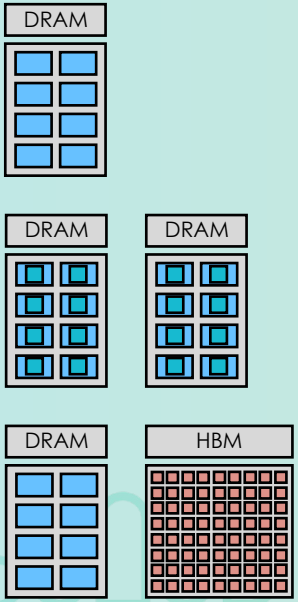
GPU

- CUDA, HIP
- OpenACC, OpenMP offload (directive-based models)
- Kokkos (portability)

Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



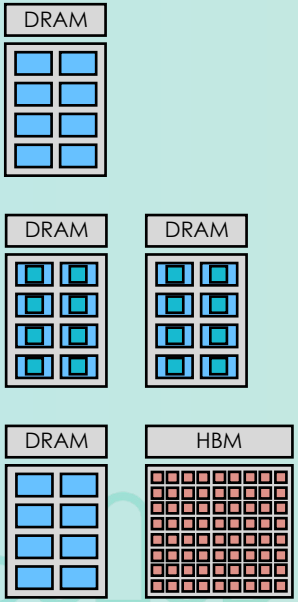
There are 4 things to know about MPI

1. It is a standard for a message passing library. The standard describes how each library call needs to function. The functions, among other things, allow you to move data between processors with different pools of memory.
2. It sets up processes on each processing element (CPU) and gives them an identifying rank, so messages with data can be sent between them.
3. All processes have their own memory and data.
4. MPI executes the same code on all processes. So for example, if you put a loop that multiplies two vectors inside an MPI region *as is*, it will multiply *all* of the elements of those two vectors in each process. If you want each processor to handle one specific part of that loop, you have to provide the logic in the code for how the iterations of the loop are divided between the processes.

Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



. . .

MPI_Init()

#sets up communication

Code that you want to run on many processors

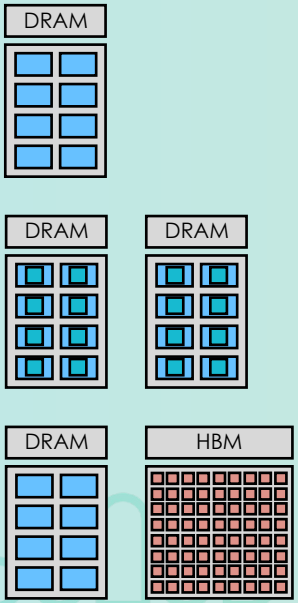
MPI_Finalize

. . .

Parallel Programming Models

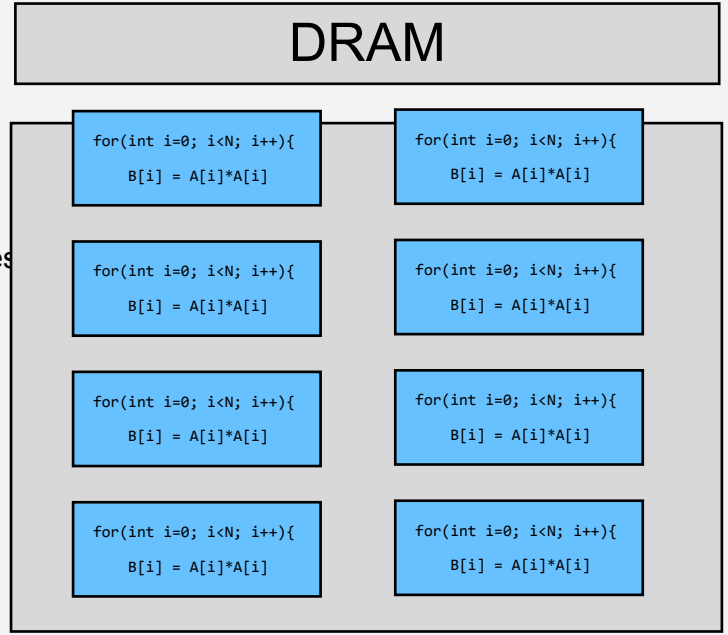
A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



Putting that loop In an MPI region will not by itself divide the iterations among the processes

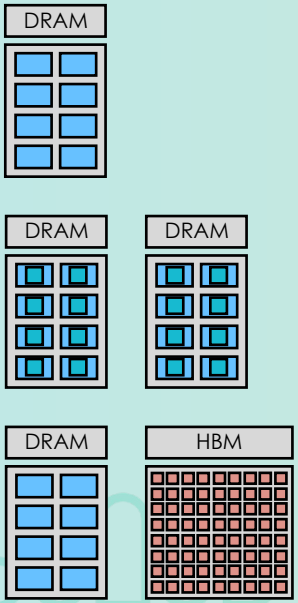
The programmer needs to supply the logic for how to distribute the work.



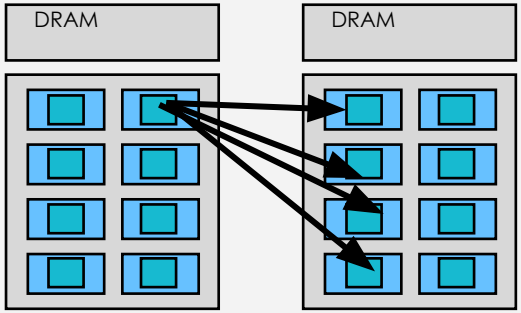
Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



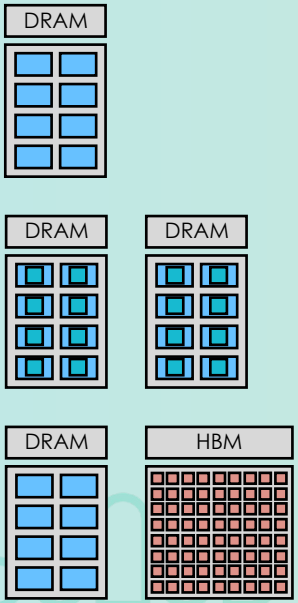
Processor can communicate collectively, for example, by broadcasting data from one processor to many.



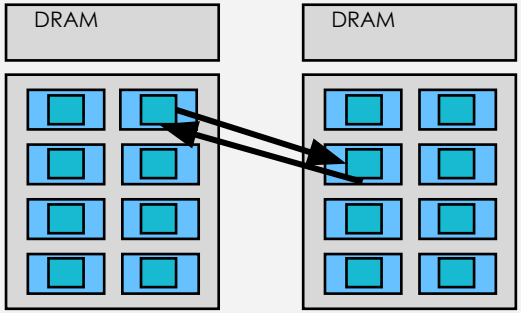
Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



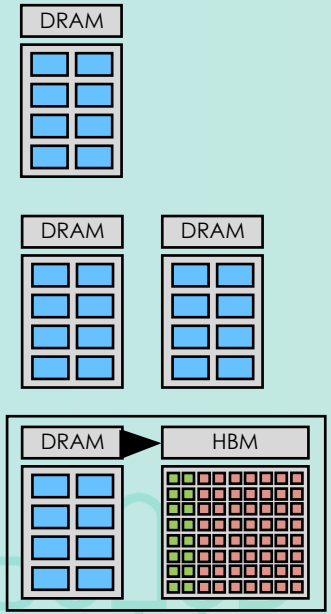
Communication of data can flow from one process to another in a point to point communication.



Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



Shared-memory models

- E.g., OpenMP
  - All process threads can access same memory (single-node)

Distributed-memory models

- E.g., Message Passing Interface (MPI)
  - All processes (i.e., MPI ranks) have access to their own memory (multi-node)

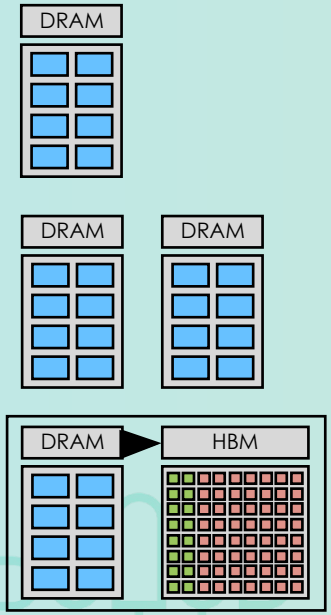
GPU

- CUDA, HIP
- OpenACC, OpenMP offload (directive-based models)
- Kokkos (portability)

Parallel Programming Models

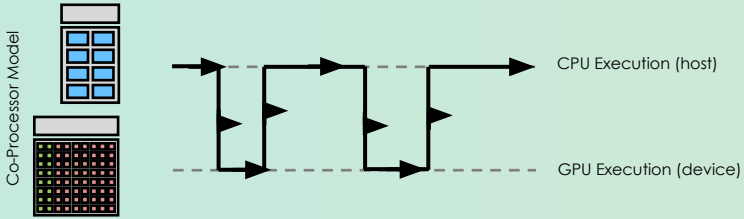
A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



GPU Programming Models

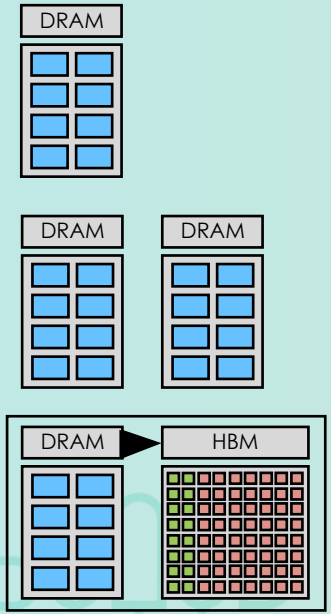
- With GPU programming, work is split between the CPU (host) and GPU (device).
- Mostly bottleneck regions of applications are ported to the GPU for optimization.
- Applications are initially profiled to determine compute intensive regions which are then offloaded to the device.



Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



GPU Programming Models: CUDA

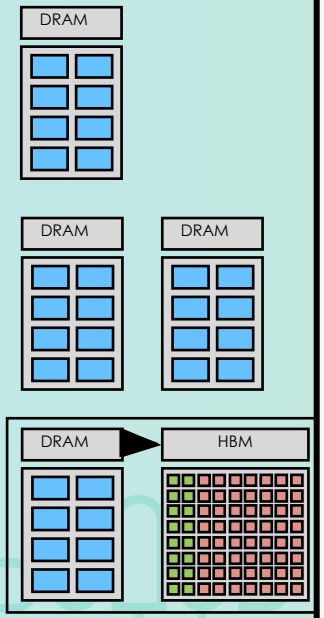
- **__global__ is a CUDA C++ keyword which indicates a function that**
  - **Runs on the device**
  - **Is called from the host code or other device code.**

```
__global__ void mykernel (void)  {

    }
```

Parallel Programming Models: CUDA

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



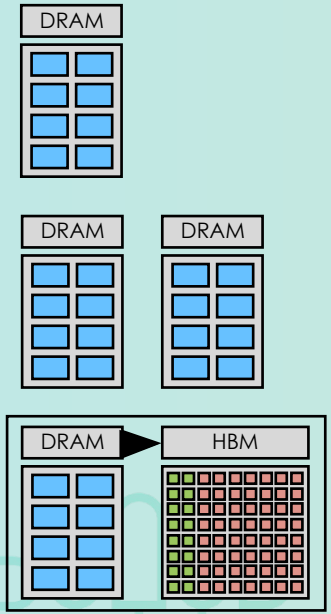

```
__global__ void mykernel (void)  {

        }
        #include <stdio.h>

        __global__
        void saxpy(int n, float a, float *x, float *y)
        {
          int i = blockIdx.x*blockDim.x + threadIdx.x;
          if (i < n) y[i] = a*x[i] + y[i];
        }

        int main(void)
        {
          int N = 1<<20;
          float *x, *y, *d_x, *d_y;
          x = (float*)malloc(N*sizeof(float));
          y = (float*)malloc(N*sizeof(float));

          cudaMalloc(&d_x, N*sizeof(float));
          cudaMalloc(&d_y, N*sizeof(float));

          for (int i = 0; i < N; i++) {
            x[i] = 1.0f;
            y[i] = 2.0f;
          }

          cudaMemcpy(d_x, x, N*sizeof(float), cudaMemcpyHostToDevice);
          cudaMemcpy(d_y, y, N*sizeof(float), cudaMemcpyHostToDevice);

          // Perform SAXPY on 1M elements
          saxpy<<<(N+255)/256, 256>>>(N, 2.0f, d_x, d_y);

          cudaMemcpy(y, d_y, N*sizeof(float), cudaMemcpyDeviceToHost);

          float maxError = 0.0f;
          for (int i = 0; i < N; i++)
            maxError = max(maxError, abs(y[i]-4.0f));
          printf("Max error: %f\n", maxError);

          cudaFree(d_x);
          cudaFree(d_y);
          free(x);
          free(y);
        }
```

SC23 Denver, CO | i am hpc.

Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



GPU Programming Models: OpenACC

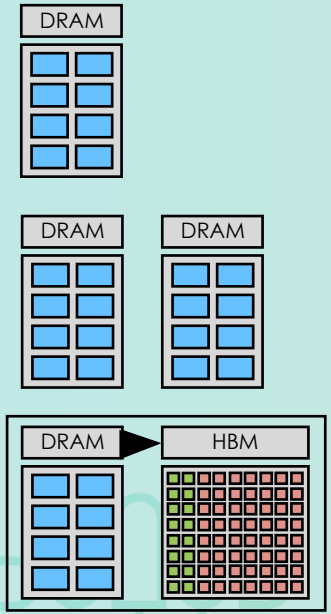
- **OpenACC is a directive based parallel programming model for GPU offloading**
- **The directive `!$acc parallel loop` indicates the device code which instructs the compiler to offload to GPU**

```
!$acc parallel loop gang default(present)
 do i=1,N

    B(i) = A(i)*A(i)

 end do
```

Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



GPU Programming Models: OpenMP Offload

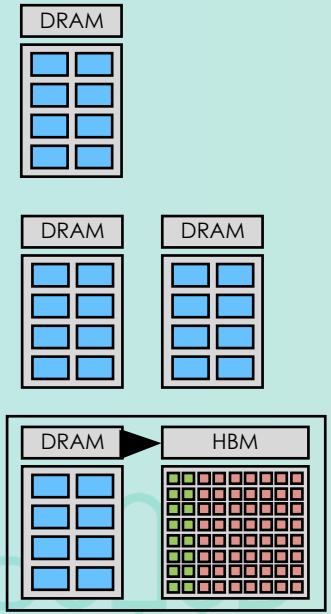
- **OpenMP offload is another one of directive based parallel programming models for GPU offloading.**
- **It is similar to the OpenACC offloading paradigm.**
- **As in OpenACC, the `!$omp target teams distribute parallel do` construct indicates the region of code to offload to the device.**

```
!$omp target teams distribute parallel do

do i=1,N

     B(i) = A(i)*A(i)

end do
!$omp end target teams distribute parallel
do
```

Parallel Programming Models

A =

```
for(int i=0; i<N; i++){
    B[i] = A[i]*A[i]
}
```



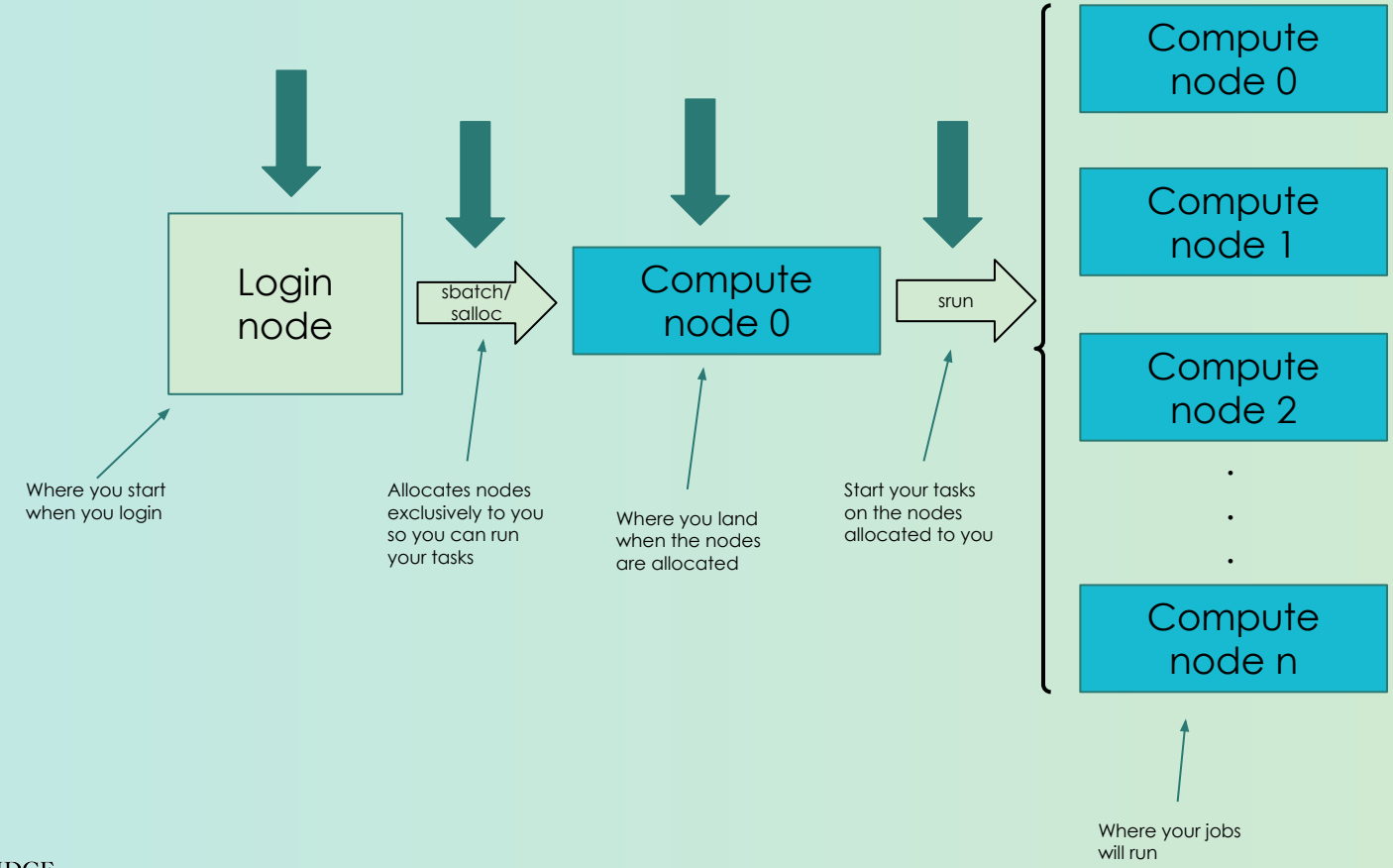
Frontier Support for Parallel Programming Models

Parallel Programming Model	Frontier Support	GNU Compiler	Cray Compiler	AMD Compiler
OpenMP	✓	✓	✓	✓
MPI	✓	✓	✓	✓
CUDA	x	x	x	x
OpenMP Offload	✓	✓	✓	✓
OpenACC	✓	✓	✓ *	x
HIP	✓	x	✓	✓
Kokkos	✓	✓	✓	✓

OAK RIDGE National Laboratory

Resource Scheduler & Parallel Job Launcher

Login, and Compute Nodes



Batch Scripts

A **batch script** can be used to submit a job to run on the compute nodes at a later time. In this case, stdout and stderr will be written to a file(s) that can be opened after the job completes.
Here's an example of a simple batch script:

Line	Actual batch script	Description
	`#!/bin/bash`	[optional] shell interpreter line
	`#SBATCH -A <project_id>`	OLCF project to charge
	`#SBATCH -J <job_name>`	Job name
	`#SBATCH -o %x-%j.out`	stdout file name (%x is job name, %j is job id)
	`#SBATCH -t 00:05:00`	Walltime requested (HH:MM:SS)
	`#SBATCH -p <partition>`	Batch queue (usually 'batch')
	`#SBATCH -N 2`	Number of computed nodes requested
		Blank line
	`srun -N2 -n4 --ntasks-per-node=2 ./a.out`	srun command to launch parallel job

Submitting your job:

```
$ sbatch submit.sl
Submitted batch job
400454
```

Common Slurm options and commands

Common Sbatch options

`-A <project_id>`	Project ID to charge
`-J <job_name>`	Name of job
`-p <partition>`	Partition / batch queue
`-t <time>`	Wall clock time <HH:MM:SS> (or you can just give minutes)
`-N <number_of_nodes>`	Number of compute nodes
`-o <file_name>`	Standard output file name
`-e <file_name>`	Standard error file name
`--threads-per-core=<threads>`	Number of active hardware threads per core [1 (default) or 2]

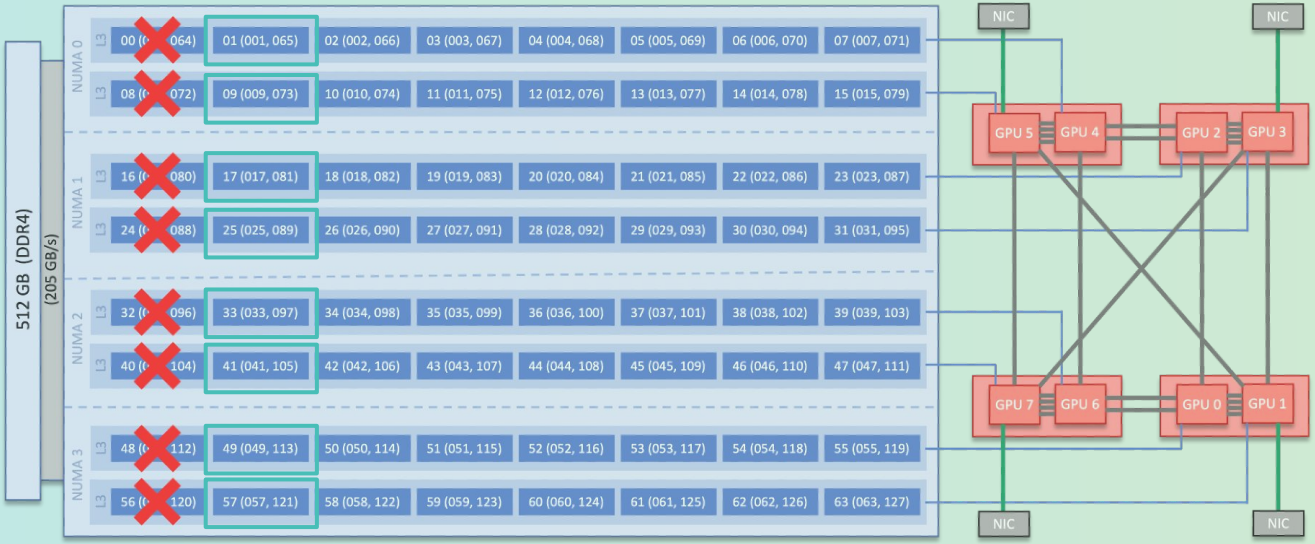
Common Slurm commands

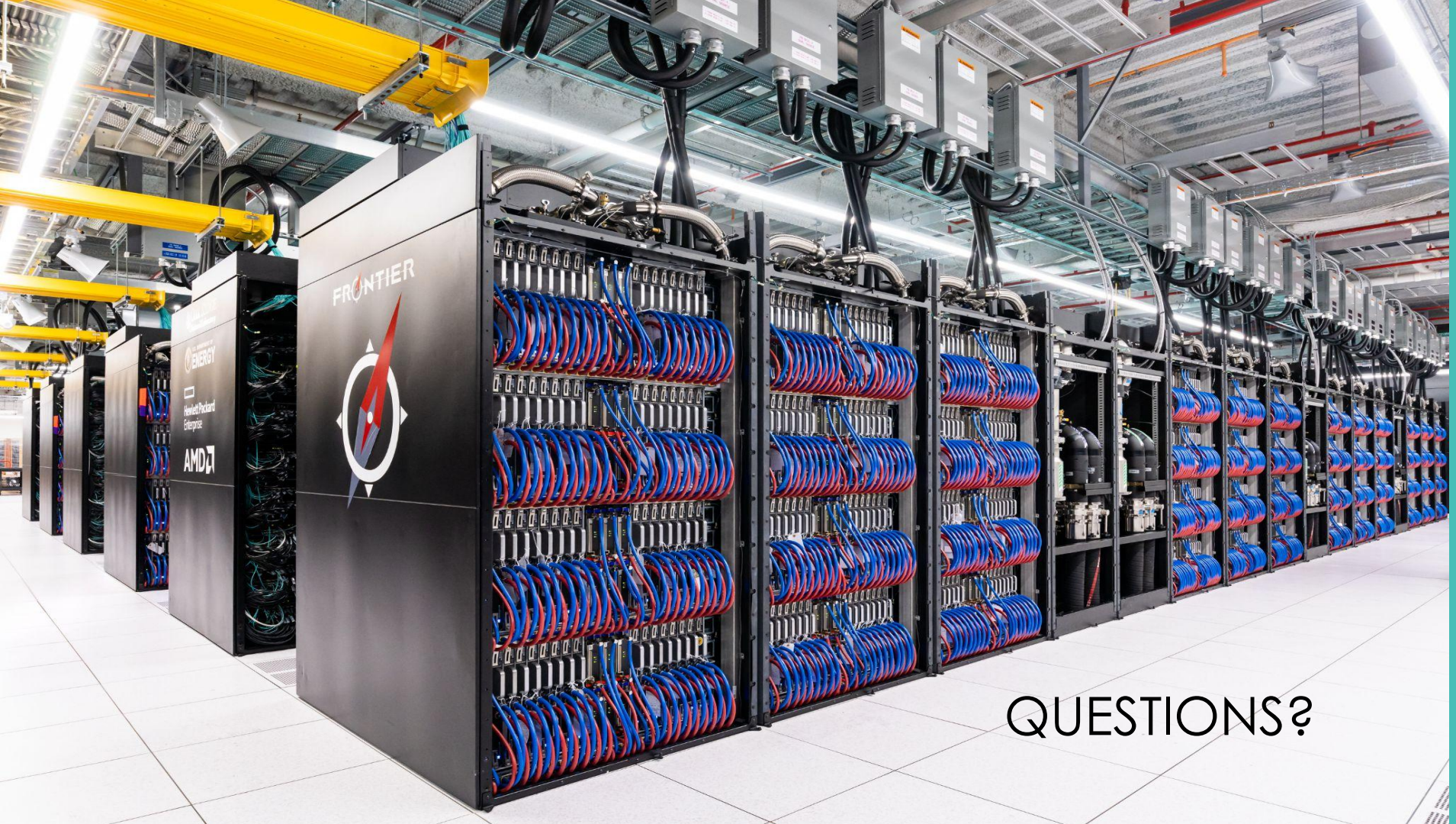
`sbatch`	Submit a job script
`sinfo`	Used to view partition and node information.
`squeue`	Used to view job and job step information for jobs in the scheduling queue.
`sacct`	Used to view accounting data for jobs and job steps in the Slurm database.
`scancel`	Used to signal or cancel jobs or job steps.
`scontrol`	Used to view or modify active job configuration.

Srun example

```
$ srun -N 1 -n 8 -c 1 ./hello_mpi_omp | sort

MPI 000 - OMP 000 - HWT 001 - Node frontier00410
MPI 001 - OMP 000 - HWT 009 - Node frontier00410
MPI 002 - OMP 000 - HWT 017 - Node frontier00410
MPI 003 - OMP 000 - HWT 025 - Node frontier00410
MPI 004 - OMP 000 - HWT 033 - Node frontier00410
MPI 005 - OMP 000 - HWT 041 - Node frontier00410
MPI 006 - OMP 000 - HWT 049 - Node frontier00410
MPI 007 - OMP 000 - HWT 057 - Node frontier00410
```

`-N`	Number of nodes for the job step
`-n`	Number of tasks in the job step
`-c`	Number of cores for each task





QUESTIONS?

Login

We are on Frontier for the HPC challenges.

```
ssh csep###@frontier.olcf.ornl.gov
```

Login and Getting Started with Hands on HPC

Suzanne Parete-Koon

OLCF HPC Engineer

Terminal

MAC/Windows Terminal

- Applications> Utilities > Terminal

```
ssh csep###@frontier.olcf.ornl.gov
```

- Enter your PIN follow by the token code when prompted

Jupyter

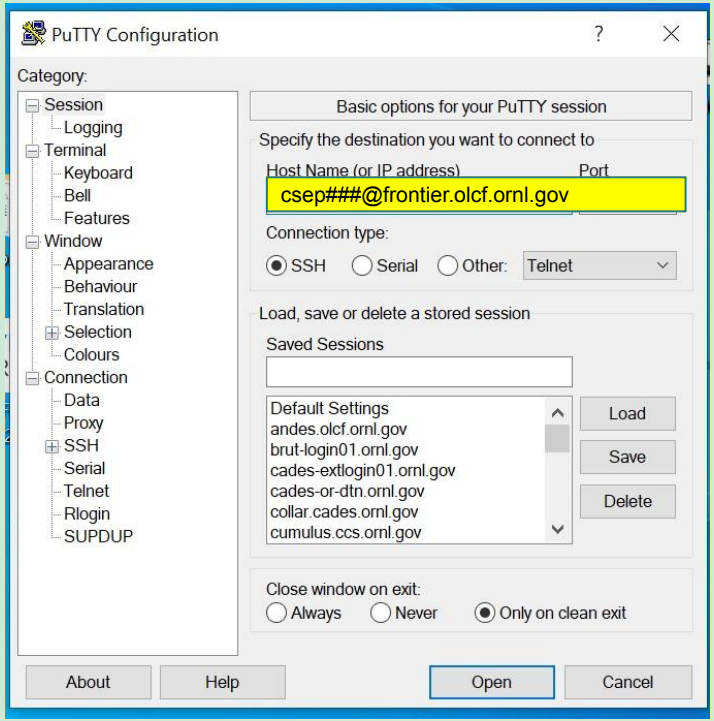
- Open browser
- Go to https://jupyter.olcf.ornl.gov/
- Click the green OAuth Button
- Login with your csep### Username and password.
- Choose “Slate - Crash Course”
- Click Open Terminal
  - Then type

```
ssh csep###@frontier.olcf.ornl.gov
```

- Enter your PIN follow by the token code when prompted

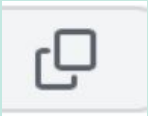
Windows Putty

- Click Open Putty
- Enter csep###@frontier.olcf.ornl.gov
- Click Open
- Enter your PIN + Token Code

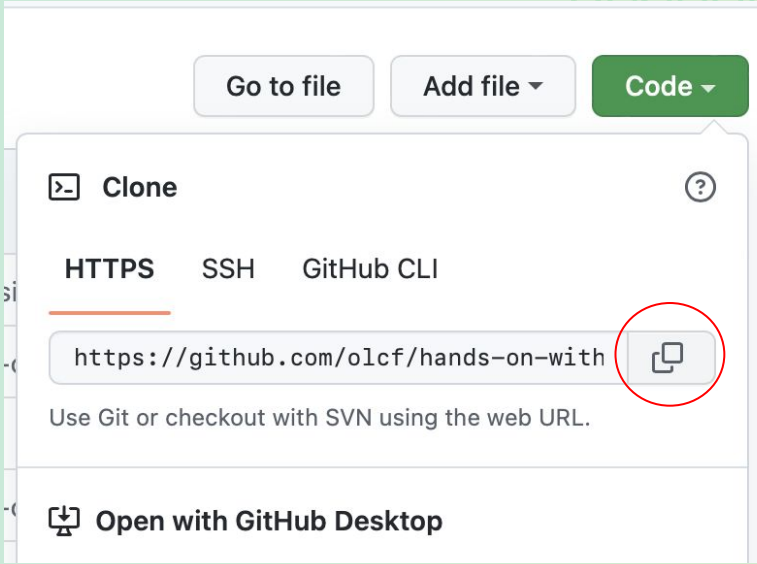


How to get the repo

Direct your browser to https://bit.ly/handsonfrontier

Click on the  to copy the address.

Go to your command line and type
"`git clone`"
then paste in the address.




```
[csep99@login12.frontier ~]$ git clone https://github.com/olcf/hands-on-with-Frontier-.git
Cloning into 'hands-on-with-Frontier-'...
remote: Enumerating objects: 2715, done.
remote: Counting objects: 100% (1352/1352), done.
remote: Compressing objects: 100% (548/548), done.
remote: Total 2715 (delta 1048), reused 934 (delta 796), pack-reused 1363
Receiving objects: 100% (2715/2715), 112.35 MiB | 67.24 MiB/s, done.
Resolving deltas: 100% (1726/1726), done.
[csep99@login12.frontier ~]$
```

Certificate

Do 7 challenges to get a certificate

- You can do them on either Frontier or Anvil
- Frontier only Challenges must complete this afternoon
- We are looking for evidence that you tried and learned
- To get credit for a challenge attempt, enter your full name and email into this google sheet and then paste in the path to **ONE** job file or **ONE** evidence-of-work file for each challenge

- Acceptable files
  - Job output file for successful job
  - Text file with output from a challenge, for example, Password in a haystack asks you to find two passwords, so put them in a text file and give us the path to the text file.
  - Code or text file that shows evidence of progress

What if I spent a lot of time on a challenge but never got the code to work?

- Turn in the path to the code file that shows your best attempt.

You have until Dec 13 to turn in file paths so keep trying!

Certificate Continued

Example For a challenge where you must manually make a .txt file, like password in a Haystack
Make a file in Vim

```
$ vi password_in_haystack.txt
```

Once in vim, type ***i***
Type in the two passwords you found.
Hit ***esc :wq return***
List the file with **ls**

```
$ ls
password_in_haystack.txt
```

Issue the **pwd** command (and hit 'return') to get the full path.

```
$ pwd
/ccs/home/suzanne/hands-on-with-Frontier-/challenges/Password_in_a_Haystack
```

Copy this.

In the google sheet enter this path + the filename on your line under the Password in a Haystack challenge:

/ccs/home/suzanne/hands-on-with-Frontier-/challenges/Password_in_a_Haystack/password_in_haystack.txt