OAK RIDGE National Laboratory

Using the Summit Supercomputer

Suzanne Parete-Koon
User Assistance - OLCF
Oak Ridge National Lab.

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

U.S. DEPARTMENT OF ENERGY

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 for users

Managed by users 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

Summit 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 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 env
```

Using Compilers on Ascent

On Ascent

- Set the module
- Use the recommended compile command
- Compiler section of User docs

C compilers on Ascent

Vendor	Module	Compiler
IBM	`xl`	xlc xlc_r
GNU	system default	gcc
GNU	`gcc`	gcc
LLVM	`llvm`	clang
PGI	`pgi`	pgcc
NVHPC	`nvhpc`	nvc

LMOD Environment Modules

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

```
[nk8@login1.ascent ~]$ module list

Currently Loaded Modules:
  1) xl/16.1.1-10                     3) lsf-tools/2.0
  2) spectrum-mpi/10.4.0.3-20210112   4) DefApps
```

LMOD Environment Modules

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

```
[nk8@login1.ascent ~]$ module list

Currently Loaded Modules:
  1) xl/16.1.1-10                     3) lsf-tools/2.0
  2) spectrum-mpi/10.4.0.3-20210112   4) DefApps



[nk8@login1.ascent ~]$ module load pgi

Lmod is automatically replacing "xl/16.1.1-10" with "pgi/20.4".


Due to MODULEPATH changes, the following have been reloaded:
  1) spectrum-mpi/10.4.0.3-20210112
```

LMOD Environment Modules

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

```
[nk8@login1.ascent ~]$ module list

Currently Loaded Modules:
  1) xl/16.1.1-10                    3) lsf-tools/2.0
  2) spectrum-mpi/10.4.0.3-20210112  4) DefApps



[nk8@login1.ascent ~]$ module load pgi

Lmod is automatically replacing "xl/16.1.1-10" with "pgi/20.4".


Due to MODULEPATH changes, the following have been reloaded:
  1) spectrum-mpi/10.4.0.3-20210112

[nk8@login1.ascent ~]$ module list

Currently Loaded Modules:
  1) lsf-tools/2.0   3) pgi/20.4
  2) DefApps         4) spectrum-mpi/10.4.0.3-20210112



[nk8@login1.ascent ~]$
```

LMOD Environment Modules

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

```
[nk8@login1.ascent ~]$ cd foundational_hpc_skills/intro_to_c/01_simple_c_program/
[nk8@login1.ascent 01_simple_c_program]$ pgcc simple.c
[nk8@login1.ascent 01_simple_c_program]$ ls
a.out  simple.c
[nk8@login1.ascent 01_simple_c_program]$ ./a.out
The value of this integer is 3
[nk8@login1.ascent 01_simple_c_program]$
```

OAK RIDGE National Laboratory

Parallel Programming Models

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

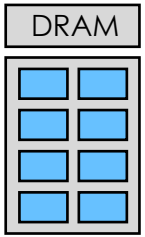
Optimizing just you and one washer and one dryer

You your housemate, 2 washers, and 2 dryers

Parallel Programming Models

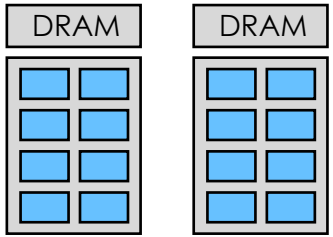
A = 

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



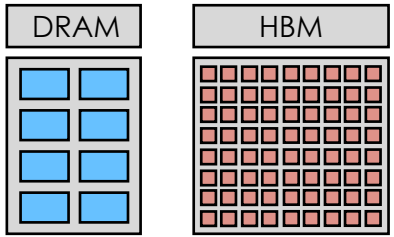
Shared-memory models

- 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)



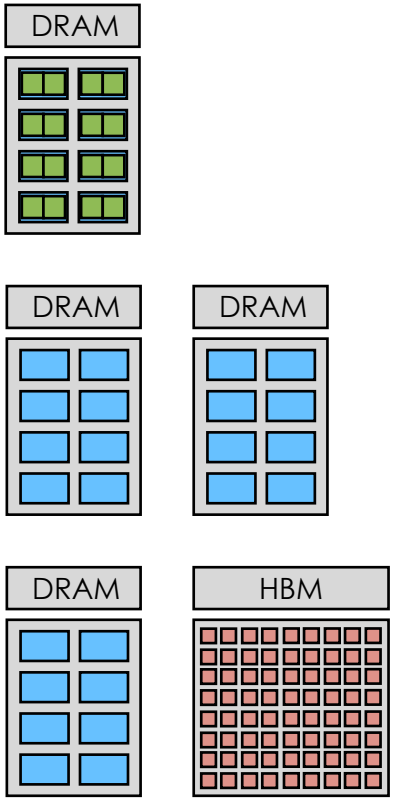
GPU

- 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

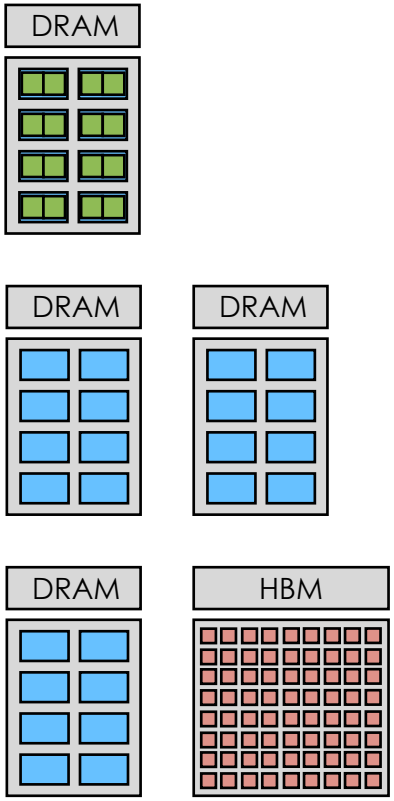
```
#pragma omp parallel default(none)
shared(A, B)
    {
        #pragma omp for
         for(int i=0; i<N; i++){

           B[i] = A[i]*A[i]
         }
    }
```

Parallel Programming Models

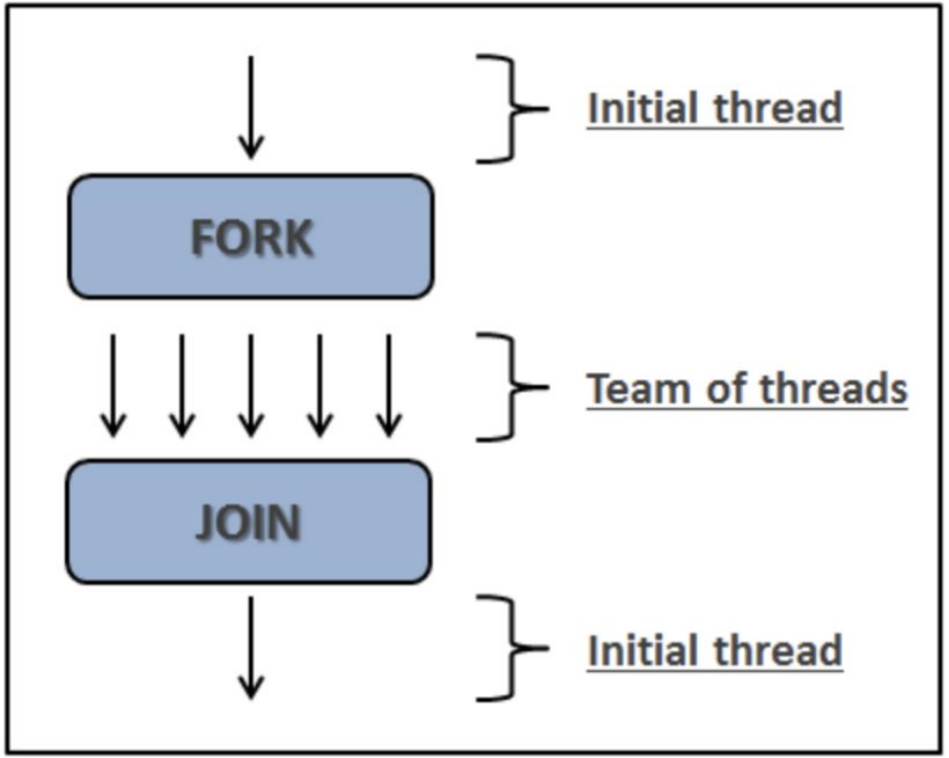
A = 

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



Shared-memory models

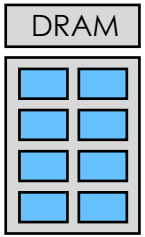
- E.g., OpenMP



Parallel Programming Models

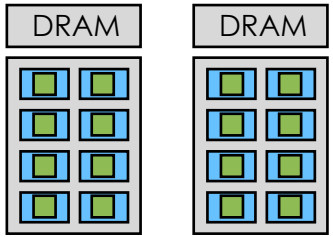
A = 

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



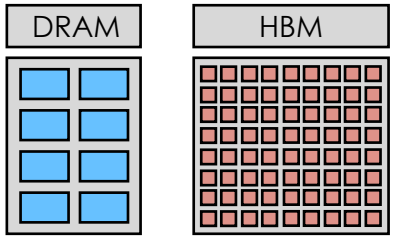
Shared-memory models

- 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)



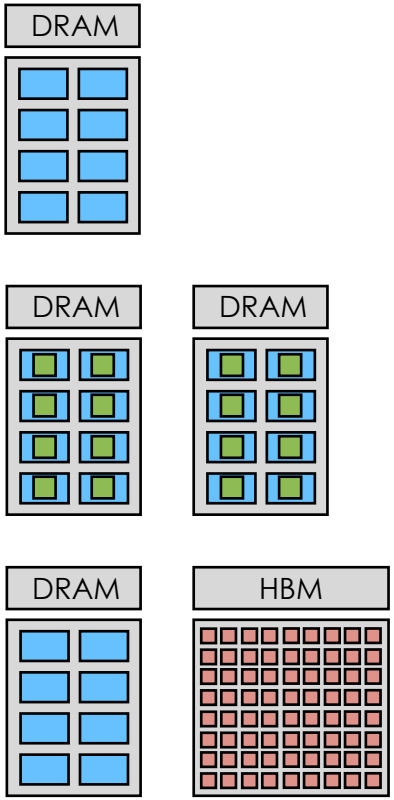
GPU

- 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]
}
```



. . .

MPI_Init()

#sets up communication

Code that you want to run on many processors

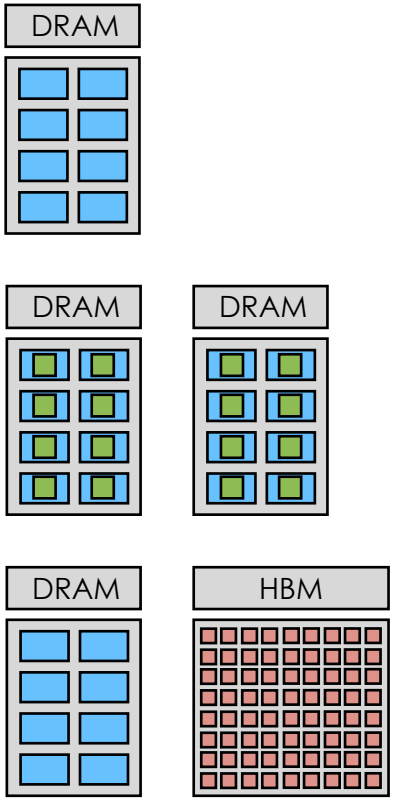
MPI_Finalize

. . .

Parallel Programming Models

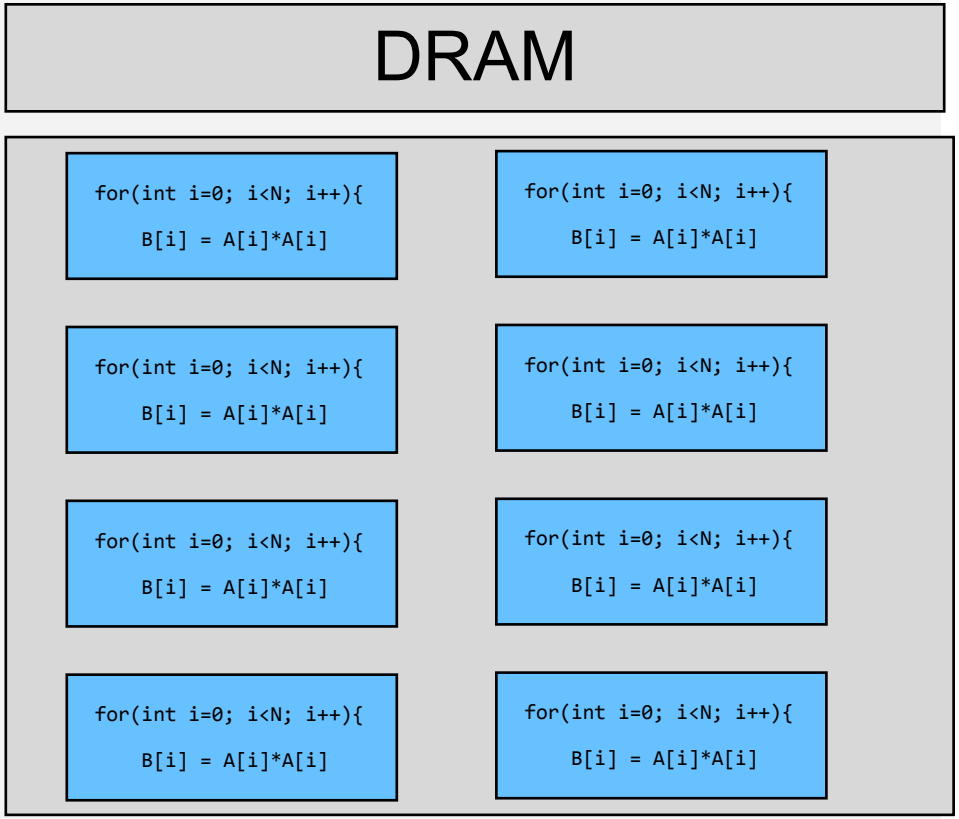
A =

```
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.

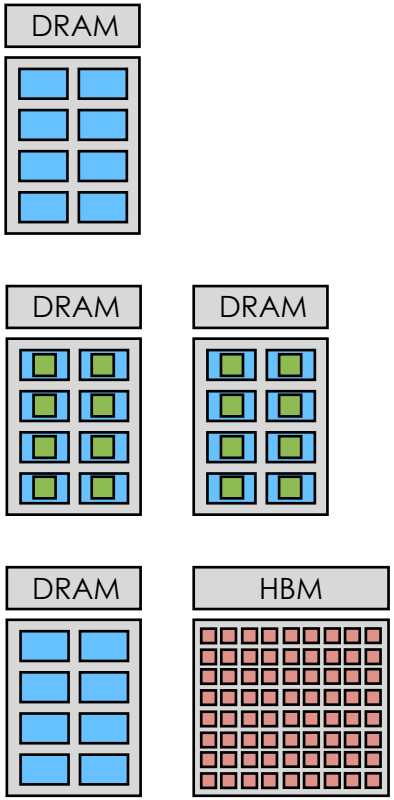
The programmer needs to supply the logic for how to distribute the work.



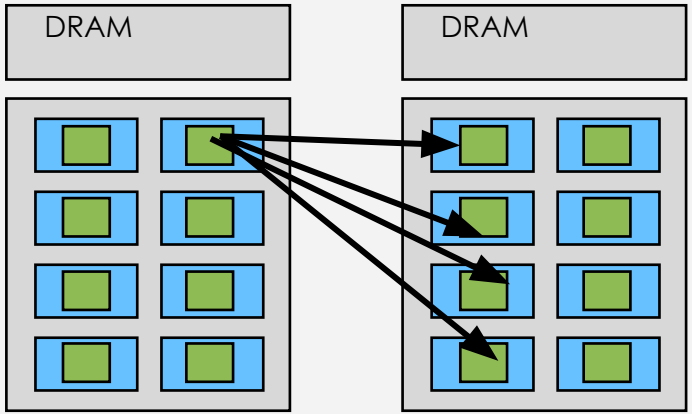
Parallel Programming Models

A = 

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



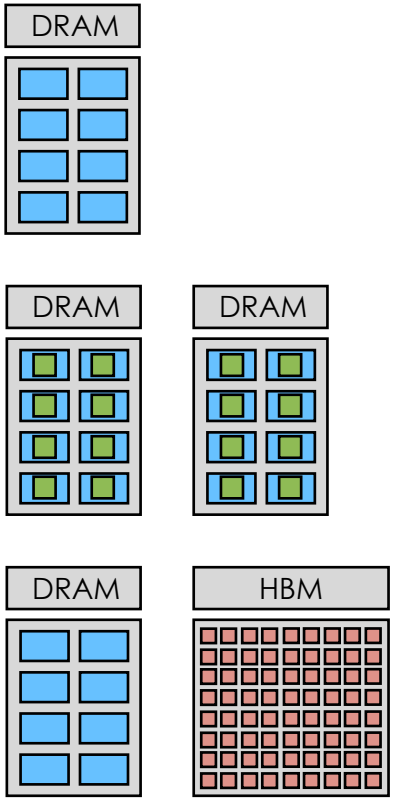
Processor can communicate collectively, for example, by broadcasting data from one processor to many.



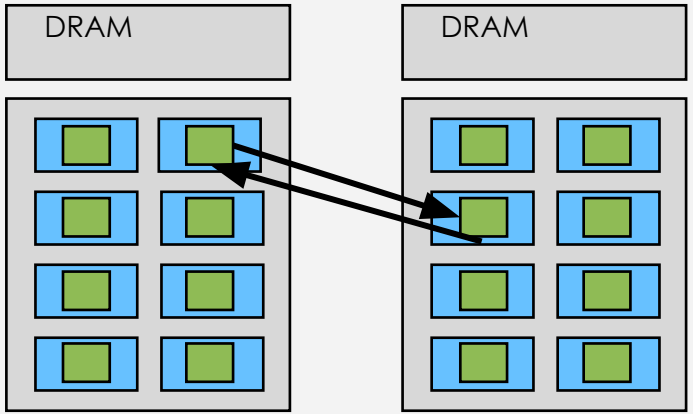
Parallel Programming Models

A =

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



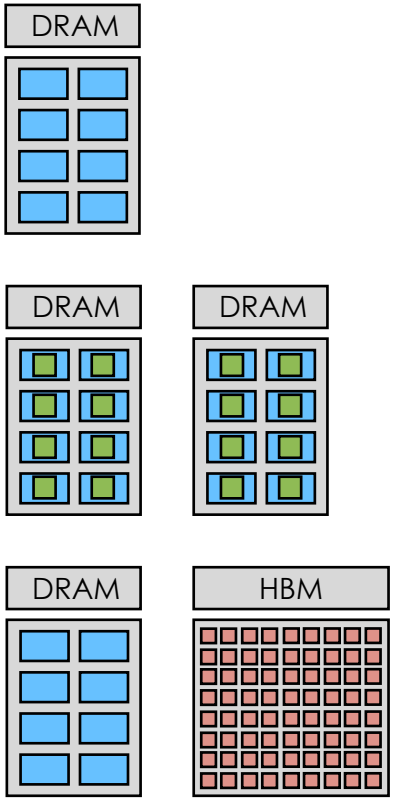
Communication of data can flow from from one process to another in a point to point communication.



Parallel Programming Models

A = 

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



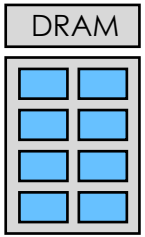
There are 4 things to know about MPI

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 your 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 a MPI region *as is*, it will multiply *all* of the elements of those two vectors in each process. If you want a 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

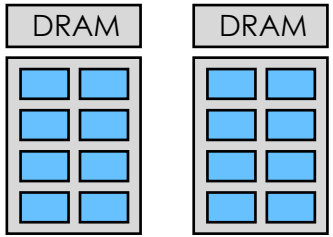
A = 

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



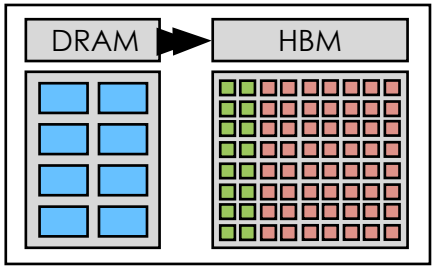
Shared-memory models

- 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)



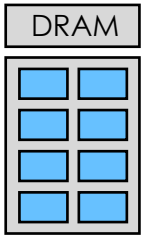
GPU

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

Parallel Programming Models

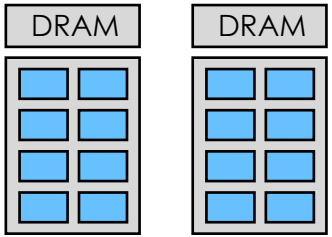
A = 

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



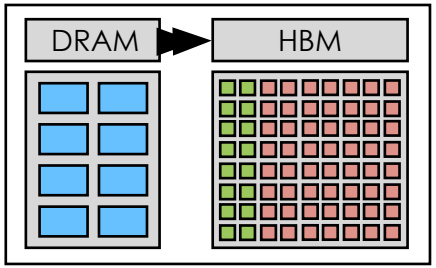
Shared-memory models

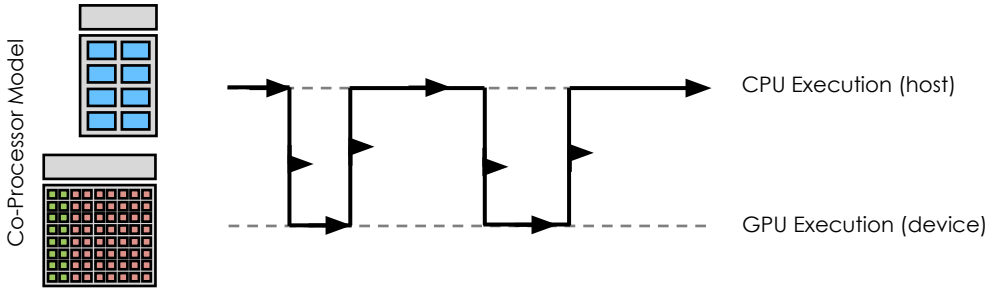
- E.g., OpenMP
  - All process threads can access same memory (single-node)



Distributed-memory models

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

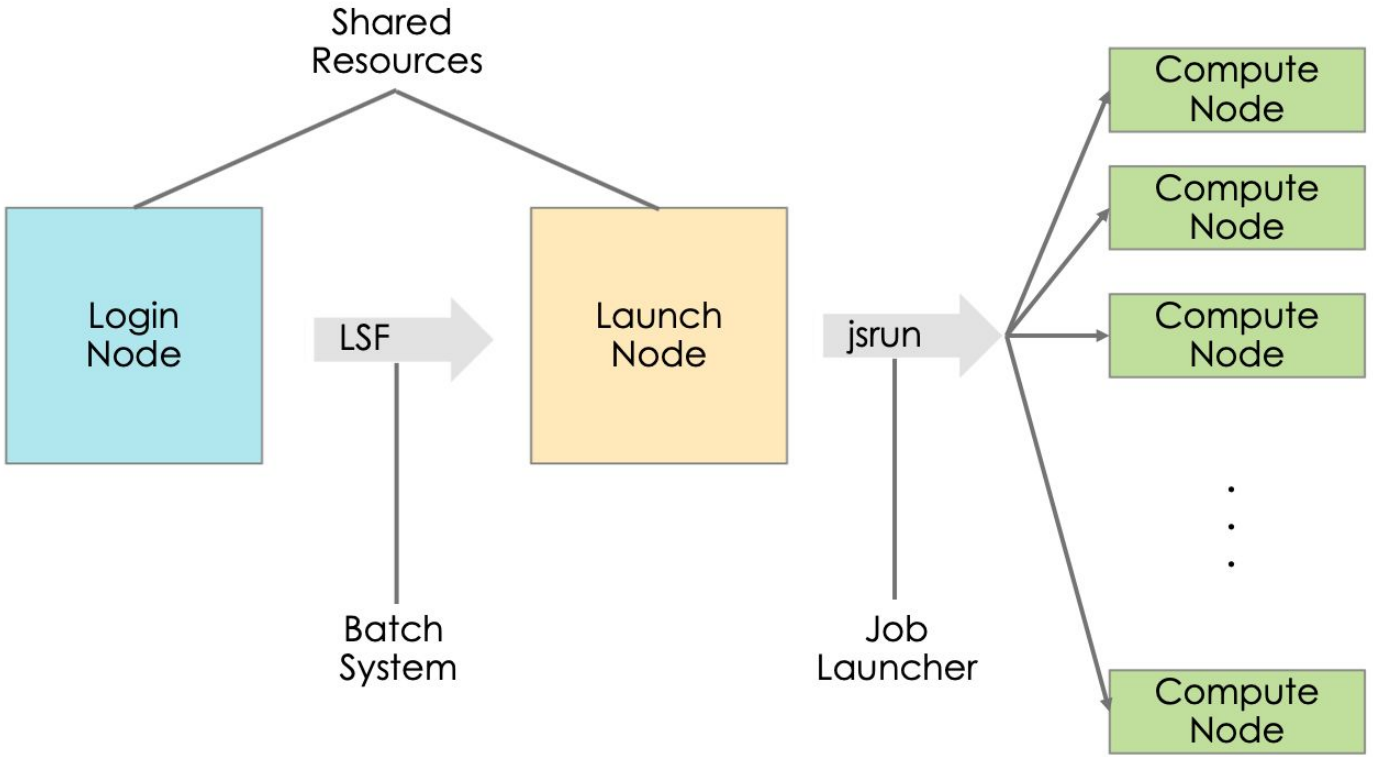




OAK RIDGE National Laboratory

Resource Scheduler
&
Parallel Job Launcher

Login, Launch, and Compute Nodes



Parallel Job Execution

Batch Scheduling System

IBM Load Sharing Facility (LSF)

- Allocates resources
- Batch scheduler
- Similar functionality to PBS/MOAB or Slurm
- Allocates entire nodes (@OLCF)

Parallel Job Launcher

jsrun

- Developed by IBM for the Oak Ridge and Livermore CORAL systems
- Similar functionality to aprun, mpirun, & srun

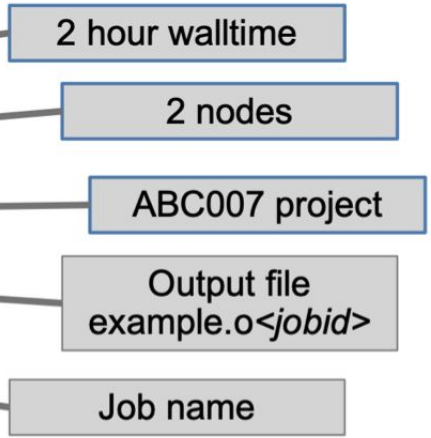
LSF Example Batch Script (non-interactive)

Batch script example

```
#!/bin/bash

#BSUB -W 2:00
#BSUB -nnodes 2
#BSUB -P abc007
#BSUB -o example.o%J
#BSUB -J example

jsrun -n2 -r1 -a1 -c1 hostname
```



Batch submission

```
summit-login1> bsub example.lsf
Job <29209> is submitted to default queue <batch>.
summit-login1>
```

Common `bsub` Options

Option	Example Usage	Description
-W	#BSUB –W 1:00	Requested Walltime [hours:]minutes
-nnodes	#BSUB –nnodes 1024	Number of nodes (CORAL systems)
-P	#BSUB –P ABC123	Project to which the job should be charged
-J	#BSUB –J MyJobName	Name of the job. If not specified, will be set to 'Not_Specified'
-o	#BSUB –o jobout.%J	File into which job STDOUT should be directed (%J will be replaced with the job ID number) If not specified will be set to 'JobName.%J'
-e	#BSUB –e joberr.%J	File into which job STDERR should be directed
-w	#BSUB –w ended(1234)	Place dependency on previously submitted jobID 1234
-N -B	#BSUB –N #BSUB -B	Send job report via email once job completes (N) or begins (B)
-alloc_flags	#BSUB –alloc_flags gpumps #BSUB –alloc_flags smt1	Used to request GPU Multi-Process Service (MPS) and to set SMT (Simultaneous Multithreading) levels. Setting gpumps enables NVIDIA's Multi-Process Service, which allows multiple MPI ranks to simultaneously access a GPU.

See **`man bsub`** for full options list

Common LSF Commands

Function	LSF Command
Submit a batch script	bsub
Monitor Queue	bjobs / jobstat
Investigate Job	bhist
Alter Queued Job	bmod
Remove Queued Job	bkill
Hold Queued Job	bstop
Release Held Job	bresume

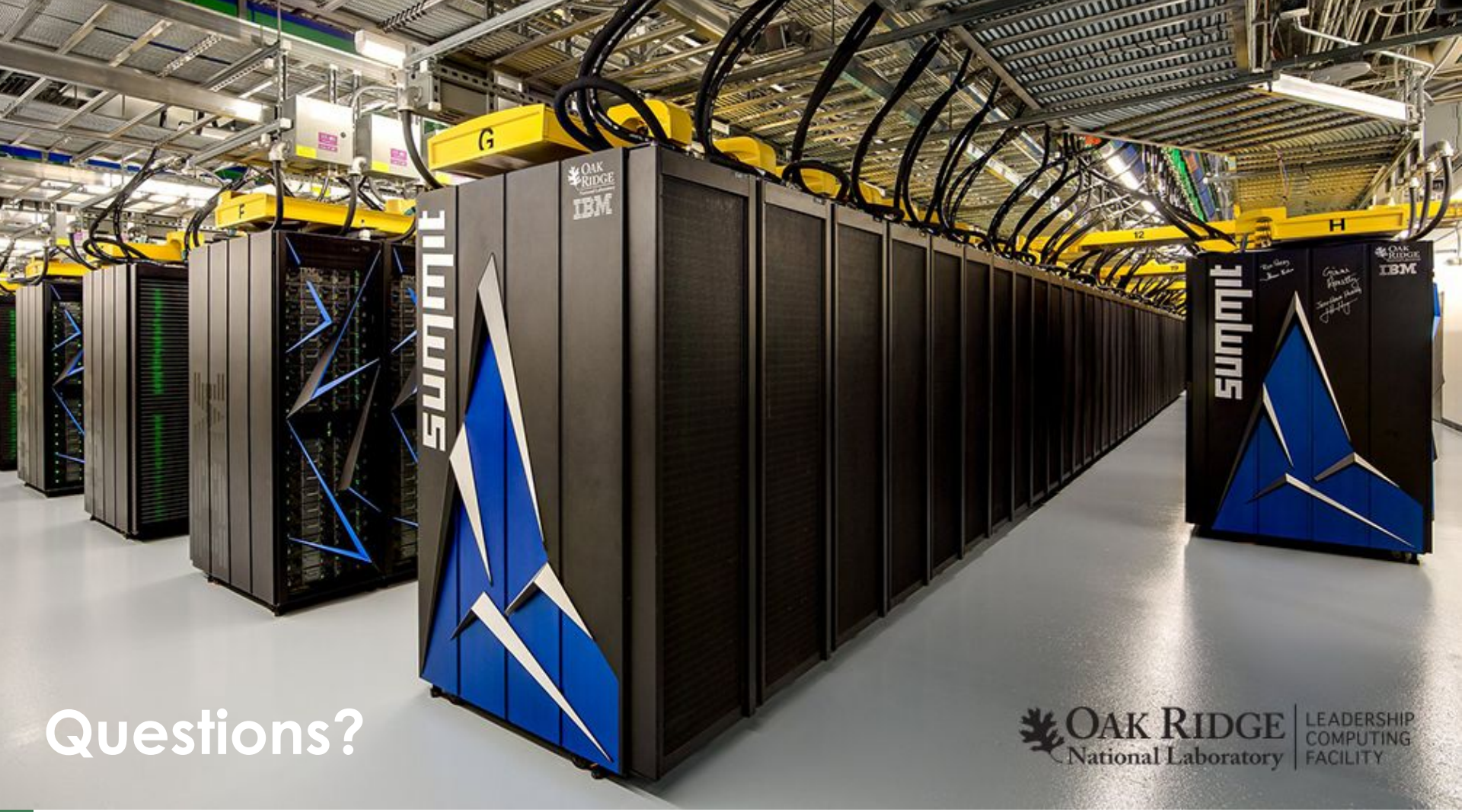
See manual pages for more info

jsrun – Basic Options

```
jsrun [-n #resource sets] [CPU cores, GPUs, tasks in each resource set] program
[program args]
```

jsrun Flags		Description	Default Value
Long	Short		
--nrs	-n	Number of RS	All available physical cores
--tasks_per_rs	-a	Number of MPI tasks (ranks) per RS	N/A (total set instead [-p])
--cpu_per_rs	-c	Number of CPUs (physical cores) per RS	1
--gpu_per_rs	-g	Number of GPUs per RS	0
--bind	-b	Number of physical cores allocated per task	packed:1
--rs_per_host	-r	Number of RS per host (node)	N/A
--latency_priority	-l	Controls layout priorities	gpu-cpu,cpu-mem,cpu-cpu
--launch_distribution	-d	Order of tasks started on multiple RS	packed

See `man jsrun` for full list of options



Questions?

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Jupyter or Mac terminal

MAC

- Applications> Utilities > Terminal

```
ssh username@login1.ascent.olcf.ornl.gov
```

- Enter your password when prompted.

Jupyter

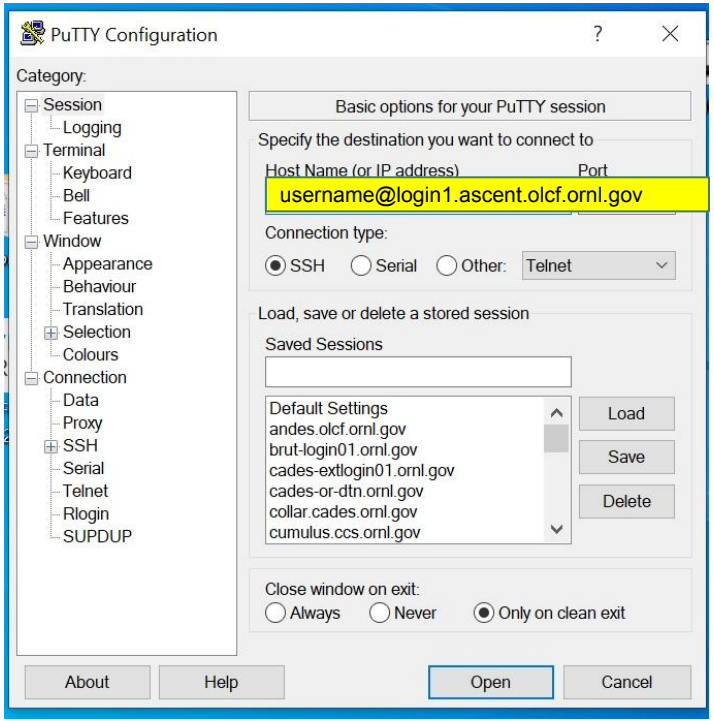
- Open browser
- Go to https://jupyter-open.olcf.ornl.gov/
- Login with your XCAMS Username and password.
- Choose Crash Course training lab
- Click On “Terminal” icon
  - Then type

```
ssh username@login1.ascent.olcf.ornl.gov
```

- Enter your password when prompted.

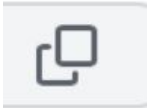
Windows Putty

- Click Open Putty
- Enter username@login1.ascent.olcf.ornl.gov
- Click Open
- Enter your password when prompted.

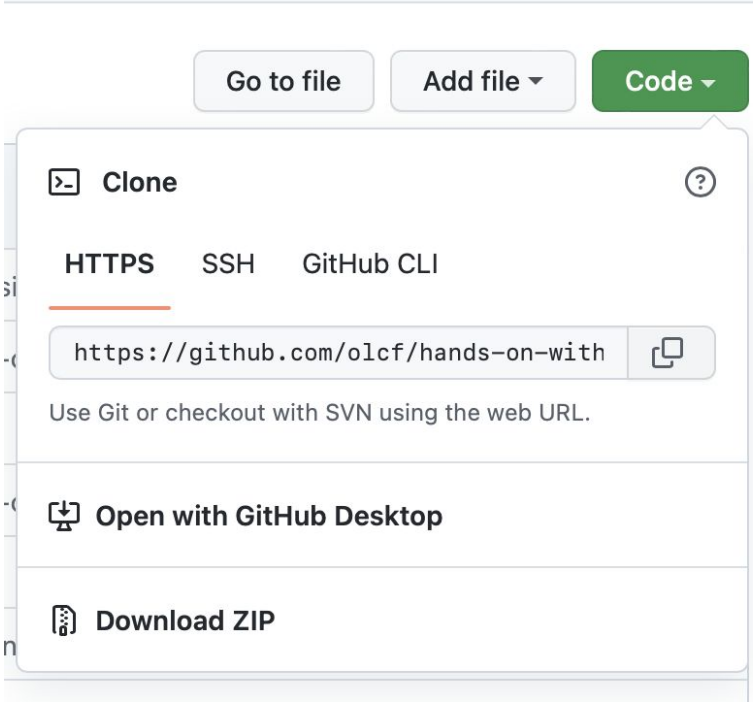


How to get the repo

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

Click on the  to copy the address.

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




```
[csep101@login2.summit ~]$ git clone https://github.com/suzannepk/hands-on-with-summit.git
Cloning into 'hands-on-with-summit'...
remote: Enumerating objects: 1805, done.
remote: Counting objects: 100% (1805/1805), done.
remote: Compressing objects: 100% (706/706), done.
Receiving objects:  68% (1230/1805), 24.97 MiB | 4.58 MiB/s
```

Certificate

Do 7 challenges to get a certificate as outlined below.
We are looking for evidence that you tried and learned - you don't need perfect solutions for all 7 challenges, but you do need to submit a file for all but challenge 1. The kind of files you need are listed next to ea

Do these 6 first challenges first

1. Access_Ascent_and_Clone_Repo (no file needed)

1.5. Basic_Unix_Vim (Only do this one if you have no _____experience with Unix or a text editor)

2. Basic_Workflow (`add_vec_cpu.JOBID`)
3. OpenMP_Basics (any one of the three job output files)
4. MPI_Basics (any one of the three job output files)
5. Find_the_Compiler_Flag (`add_vec_acc.JOBID`)
6. GPU_Data_Transfers (`add_vec_cuda.%J`)

After completing those challenges, choose at least one of the following:

- Password_in_a_Haystack (make a .txt file containing passwords)
- Python_Conda_Basics (`environment.yml`)
- jsrun_Job_Launcher (make a .txt file containing jsrun commands for ex. 4,5,6,7)
- Parallel_Scaling_Performance (job output)
- Jobs_in_Time_Window (.txt file with answers)
- OpenMP_Offload (any job output)
- Python_Parallel_HDF5 (any job output)
- Python_Cupy_Basics (any job output)
- Score-P_and_Vampir_Basics (traces.otf2 from any example)
- Python_Pytorch_Basics (any job output)
- GPU_Matrix_Multiply (any job output)
- GPU_Profiling (`profiling_output_optimized.<jobid>`)

Certificate Continued

When you complete each challenge, copy your output files to
/gpfs/wolf/world-shared/trn016/<your-user-name>

Example 1 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***
Copy file to /gpfs/wolf/world-shared/trn016/<your-user-name>

```
$ cp password_in_haystack.txt /gpfs/wolf/world-shared/trn016/<your-user-name>
```

Example 2 For a challenge with jobfiles
For example, OpenMP_basics has a job output file called vec_add.<JOBID

```
$cd OpenMP_Basics/vector_addition
$ ls
```

Look for files of the form `vec_add.<JOBID>`

```
$ cp vec_add.<JOBID> /gpfs/wolf/world-shared/trn016/<your-user-name>
```

Certificate Continued

You have the rest of the time in this session to get started.
You have until January 31 to finish.

Tips and Tricks:

To get back to your home dir from anywhere on the filesystems:

cd ~ or cd $Home