Directions for CS455: Introduction to High Performance Computing

Using ACER Lakeshore Resource

This guide explains compiling and running programs on the `lakeshore.acer.uic.edu` resource.

Connect to `lakeshore.acer.uic.edu`

To connect to the lakeshore cluster, you must be on the UIC network or connected to it by VPN, then you may access using SSH[1]:

```
ssh netid@lakeshore.acer.uic.edu
# Example: ssh ashov@lakeshore.acer.uic.edu
# Provide UIC password
# Use 1. Duo Push or 2. SMS passcodes
```

Upon successful login, you will be in your home directory at `/home/<netid>` on `login` node, which is meant for file management, code editing, and job submission. Computationally intensive programs should be executed on `compute` nodes to ensure optimal performance and prevent system slowdowns. There are two ways to access compute nodes: `interactive` mode for real-time execution and `batch` job submission for scheduled runs using job scripts.

OpenMP program on `lakeshore.acer.uic.edu`

- Create a sample OpenMP program called `ompRanks.cc`:

```
#include <iostream>
#include <omp.h>

int main() {
    // Parallel region with OpenMP
    #pragma omp parallel
    {
        int threadId = omp_get_thread_num();
        int numThreads = omp_get_num_threads();
        // Print thread information
        std::cout << "Hello from thread " << threadId
                  << " out of " << numThreads << std::endl;
    }
    return 0;
}
```

- Create a Makefile:

```
CXX = g++
CXXFLAGS = -std=c++17 -Wall -fopenmp

# Source files and targets
SOURCES = $(wildcard *.cc)
TARGETS = $(SOURCES:.cc=)

all: $(TARGETS)

%: %.cc
	$(CXX) $(CXXFLAGS) -o $@ $<

clean:
	rm -f $(TARGETS)
```

[1] Simplify login by adding your SSHKEY to `lakeshore.acer.uic.edu`

Running OpenMP programs in interactive mode

To request one interactive node with four ranks for 15 minutes on `batch` partition:

```
salloc --job-name "InteractiveJob" --time 00:15:00 -p batch --nodes=1 --ntasks=4
```

Managing modules

- List currently loaded modules:

```
module list
```

- Search for modules:

```
module avail mpi
```

- Load `OpenMPI` module:

```
module load OpenMPI
```

Compiling and running an OpenMP program in interactive node

- Compile the program:

```
make
```

- Execute the program:

```
./ompRanks
```

- To set the number of OpenMP threads:

```
export OMP_NUM_THREADS=4
```

- Build and run again to observe the effect of setting thread count.

Running OpenMP programs using job submission

- Create a job script `openmp-job.sh` to request 1 compute node with 8 threads for a maximum runtime of 5 minutes. The script compiles the OpenMP program using `make`, sets the number of OpenMP threads to 8, and executes the program.

```
#!/bin/bash

#SBATCH --partition=batch
#SBATCH --job-name=openmpjob
#SBATCH --nodes=1
#SBATCH --tasks-per-node=8
#SBATCH --time=00:05:00
#SBATCH --output=openmpjob.output
#SBATCH --error=openmpjob.error

make
export OMP_NUM_THREADS=8
./ompRanks
```

- Submit the job:

```
sbatch openmp-job.sh
```

- Check job in queue:

```
squeue -u $USER
```

- View output:

```
cat openmpjob.output
```

MPI program on Lakeshore

- Create a sample MPI program named `sample.cc`:

```
#include <mpi.h>
#include <cstdio>

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    printf("Hello from rank %d of %d\n", rank, size);
    MPI_Finalize();
    return 0;
}
```

- Create a `Makefile`:

```
CXX = mpicxx
CXXFLAGS = -std=c++17 -Wall

# Source files and targets
SOURCES = $(wildcard *.cc)
TARGETS = $(SOURCES:.cpp=)

all: $(TARGETS)

%: %.cc
        $(CXX) $(CXXFLAGS) -o $@ $<

clean:
        rm -f $(TARGETS)
```

Running MPI programs in interactive mode

To request one node with four ranks for 15 minutes on `batch` partition:

```
salloc --job-name "InteractiveJob" --time 00:15:00 -p batch --nodes=1 --ntasks=4
```

Managing modules

- List currently loaded modules:

```
module list
```

- Search for modules:

```
module avail mpi
```

- Load `OpenMPI` module:

```
module load OpenMPI
```

Compiling and running MPI program in interactive node

- Compile the program:

```
make
```

- Execute the program with 4 MPI ranks:

```
mpirun -np 4 sample
```

Running MPI programs using job submission

- Create a job script `mpi-job.sh` to request 4 compute nodes, each with 8 threads, for a total of 32 threads over a 5 minutes runtime. The script loads the required OpenMPI module, compiles the MPI program using `make`, and executes it across the allocated resources.

```
#!/bin/bash

#SBATCH --partition=batch
#SBATCH --job-name=mpijob
#SBATCH --nodes=4
#SBATCH --tasks-per-node=8
#SBATCH --time=00:05:00
#SBATCH --output=mpijob.output
#SBATCH --error=mpijob.error

module load OpenMPI
make
mpirun -np 32 sample
```

- Submit the job:

```
sbatch mpi-job.sh
```

- Check job in queue:

```
squeue -u $USER
```

- View output:

```
cat mpijob.output
```

Additional Resources

1. **Documentation on submitting job on Lakeshore:** Submitting a Job on the Cluster.
2. **Getting Started on Lakeshore:** List of docs to start using Lakeshore.
3. **Documentation related to batch job:** List of docs on job submission.