Hands On Supercomputing

1. Interactive Supercomputing
   - working on a fast workstation
   - running scripts with mpi4py
   - running programs with MPI.jl
   - running compiled programs with mpicc
2. Using a Real Supercomputer
   - advanced cyberinfrastructure for education and research
   - getting started with message passing
   - computing on the cluster with submit scripts

MCS 572 Lecture 8
Introduction to Supercomputing
Jan Verschelde, 13 September 2024

Hands On Supercomputing

1. Interactive Supercomputing
   - working on a fast workstation
   - running scripts with mpi4py
   - running programs with MPI.jl
   - running compiled programs with mpicc
2. Using a Real Supercomputer
   - advanced cyberinfrastructure for education and research
   - getting started with message passing
   - computing on the cluster with submit scripts

working on a fast workstation

Microway numbersmasher Xeon + Tesla GPU server (2016):

- two 22-core Intel Xeon E5-2699v4 Broadwell at 2.20GHz,
- 256GB of internal memory at 2400MHz,
- 2 NVIDIA Tesla P100 16GB Pascal GPU accelerators, 4.7 TFLOPS (FP64) peak performance.

Login with `ssh` at a Terminal or PowerShell window:

```
$ ssh pascal.math.uic.edu -l netid
```

You will be prompted for your password.
If off campus, then you must be connected via VPN and have 2-factor authentication set up.

To change your password, type `passwd` at the command prompt.

transferring files

To transfer files from/to your computer to/from `pascal`, use `scp`.

```
$ scp netid@pascal.math.uic.edu:~/test.py .
```

copies `test.py` in your home folder at `pascal`
to your current computer, in the current folder, the dot.

The secure copy `scp` is normally installed with `ssh`,
available on linux, macos x, and windows.

To download from the course web site, use `wget`.

about pascal

To get the information about the processors, type

```
$ more /proc/cpuinfo
```

For information about the random access memory, type

```
$ more /proc/meminfo
```

For information about the GPUs, do

```
$ cd /usr/local/cuda/samples/1_Utilities
$ cd deviceQuery
$ ./deviceQuery
```

Typing `cd ~` brings you back to your home folder.

a newer workstation

Microway 2U Xeon + NVIDIA GPU server (2024):

- two 24-core Intel Xeon 5318Y Ice Lake-SP, up to 3.40GHz,
- 256GB of internal memory at 3200MHz,
- NVIDIA "Ampere" A100 80GB GPU accelerator,
  8.6 TFLOPS (FP64) peak performance.
  TensorCore performance: up to 19.5 TFLOPS (FP64).

Login with `ssh` at a Terminal or PowerShell window:

```
$ ssh netid@ampere.math.uic.edu
```

You will be prompted for your password.
You must be connected via VPN, even on wireless on campus.

The `ampere` has more available disk space than `pascal`.

software installation

On `ampere`, OpenMPI is installed.

- For `mpi4py`, install `miniconda3` on your account.

  And then:

  ```
  conda install -c conda-forge openmpi mpi4py
  ```

  Use `pip` to install `numpy`, `scipy`, `matplotlib`.

- For `MPI.jl`, first install `julia`.

  In your `.julia` folder locate `mpiexecjl` and adjust the path.

Hands On Supercomputing

1. Interactive Supercomputing
   - working on a fast workstation
   - running scripts with mpi4py
   - running programs with MPI.jl
   - running compiled programs with mpicc

2. Using a Real Supercomputer
   - advanced cyberinfrastructure for education and research
   - getting started with message passing
   - computing on the cluster with submit scripts

running scripts with mpi4py

With `wget` you can download scripts from the class web site.

In the `wget` below, replace `X.py` by `mpi4py_hello_world3.py`. The 2nd `wget` is from the backup site.

```
$ wget http://www.math.uic.edu/~jan/mcs572/X.py
$ wget https://janv.people.uic.edu/mcs572/X.py
$ ls
mpi4py_hello_world3.py
$ mpiexec -n 4 python3 mpi4py_hello_world3.py
Hello from 0 of 4 on pascal.math.uic.edu.
Hello from 1 of 4 on pascal.math.uic.edu.
Hello from 2 of 4 on pascal.math.uic.edu.
Hello from 3 of 4 on pascal.math.uic.edu.
$
```

Hands On Supercomputing

1. Interactive Supercomputing
   - working on a fast workstation
   - running scripts with mpi4py
   - running programs with MPI.jl
   - running compiled programs with mpicc

2. Using a Real Supercomputer
   - advanced cyberinfrastructure for education and research
   - getting started with message passing
   - computing on the cluster with submit scripts

the julia environment on pascal

In your `.bashrc` file, in your home directory
add the following two lines:

```
PATH="/home/microway/Downloads/julia-1.8.5/bin:$PATH"
export PATH
```

Then do `source ~/.bashrc` at the prompt. Type `julia`.
In the `julia` prompt, do `import Pkg; Pkg.add("MPI")`
to add the wrappers `MPI.jl` to your environment.

To use `mpiexecjl`, add the following line to your `.bashrc`:

```
export PATH=$PATH:/home/Y/.julia/packages/MPI/APiiL/bin
```

where the `Y` has to be replaced by your login name.
The `APiiL` will be different on another version of Julia.

running programs with MPI.jl

With `wget` you can download programs from the class web site.

Alternatively, use `scp` (secure copy).

In the `wget` below, replace `X.jl` by `mpi_hello_world.jl`. The 2nd `wget` is from the backup site.

```
$ wget http://www.math.uic.edu/~jan/mcs572/X.jl
$ wget https://janv.people.uic.edu/mcs572/X.jl
$ ls
mpi_hello_world.jl
$ mpiexecjl -n 4 julia mpi_hello_world.jl
Hello from 0 of 4.
Hello from 1 of 4.
Hello from 2 of 4.
Hello from 3 of 4.
$
```

Hands On Supercomputing

1. Interactive Supercomputing
   - working on a fast workstation
   - running scripts with mpi4py
   - running programs with MPI.jl
   - running compiled programs with mpicc

2. Using a Real Supercomputer
   - advanced cyberinfrastructure for education and research
   - getting started with message passing
   - computing on the cluster with submit scripts

running compiled programs with mpicc

With `wget` you can download programs from the class web site.

Alternatively, use `scp` (secure copy).

In the `wget` below, replace `X.c` by `mpi_hello_world.c`. The 2nd `wget` is from the backup site.

```
$ wget http://www.math.uic.edu/~jan/mcs572/X.c
$ wget https://janv.people.uic.edu/~jan/mcs572/X.c
$ more mpi_hello_world.c
$ mpicc -o hello mpi_hello_world.c
$ mpiexec -n 4 hello
Hello world from processor 2 out of 4.
Hello world from processor 3 out of 4.
Hello world from processor 1 out of 4.
Hello world from processor 0 out of 4.
$
```

redirecting the output and time

To redirect the output, use the `>` after the command, for example:

```
$ mpiexec ... > output
```

where `output` is the name of the output file.

To compute the wall clock time, use `time`, for example:

```
$ time mpiexec ...
```

which prints first the wall clock time.

Note that the wall clock time will fluctuate,
depending on other processes that are running on the computer.
Type `w` to see the load average.

Hands On Supercomputing

1. Interactive Supercomputing
   - working on a fast workstation
   - running scripts with mpi4py
   - running programs with MPI.jl
   - running compiled programs with mpicc

2. Using a Real Supercomputer
   - advanced cyberinfrastructure for education and research
   - getting started with message passing
   - computing on the cluster with submit scripts

getting access

- Access has been granted through `https://access-ci.org`.
- HPC workflow:
  1. request access to the cluster
  2. login to the cluster
  3. user work spaces and directories
  4. requesting and running software
  5. submitting jobs
  6. monitoring a job

Hands On Supercomputing

1. Interactive Supercomputing
   - working on a fast workstation
   - running scripts with mpi4py
   - running programs with MPI.jl
   - running compiled programs with mpicc

2. Using a Real Supercomputer
   - advanced cyberinfrastructure for education and research
   - getting started with message passing
   - computing on the cluster with submit scripts

using mpi4py

Login to an interactive node:

```
ssh netid@login-1.extreme.acer.uic.edu
```

or use `login-2`, `login-3`.

At the command prompt, type the following:

```
module load Anaconda3
```

```
source activate /classes/mcs572/common/conda/env/mpi4py
```

Then, to check, type `import mpi4py` in a `python3` prompt.

Replace the `x.py` by `mpi4py_hello_world3.py` below:

```
wget https://janv.people.uic.edu/mcs572/x.py
```

```
mpiexec -n 3 mpi4py_hello_world3.py
```

using `mpicc`

The `module avail` lists all available modules on the cluster.

```
module load OpenMPI
```

then type `which mpicc` to see if we can compile.

Replace the `x.c` by `mpi_hello_world.c` below:

```
wget https://janv.people.uic.edu/mcs572/x.c

mpicc -o hello mpi_hello_world.c
mpiexec -n 3 hello
```

the julia environment on extreme

In your `.bashrc` file, in your home directory
add the following two lines:

```
PATH="/classes/mcs572/common/julia-1.8.5/bin:$PATH"
export PATH
```

Then do `source ~/.bashrc` at the prompt. Type `julia`.
In the `julia` prompt, do `import Pkg; Pkg.add("MPI")`
to add the wrappers `MPI.jl` to your environment.

To use `mpiexecjl`, add the following line to your `.bashrc`:

```
export PATH=$PATH:/home/Y/.julia/packages/MPI/APiiL/bin
```

where the `Y` has to be replaced by your login name.

After downloading `mpi_hello_world.jl` from the class web site, do

```
mpiexecjl -n 3 julia mpi_hello_world.jl
```

Hands On Supercomputing

1. Interactive Supercomputing
   - working on a fast workstation
   - running scripts with mpi4py
   - running programs with MPI.jl
   - running compiled programs with mpicc

2. Using a Real Supercomputer
   - advanced cyberinfrastructure for education and research
   - getting started with message passing
   - computing on the cluster with submit scripts

a sample submit script

Earlier, we compiled `mpi_hello_world.c` into `hello`.

In the script below, replace `netid` with your netid:

```
#!/bin/bash
#PBS -l mem=1gb
#PBS -l walltime=00:05:00
#PBS -l nodes=8:ppn=1
#PBS -q edu_shared
#PBS -j oe
#PBS -m abe
#PBS -M netid@uic.edu
#PBS -N hello
#PBS -d /home/netid

module load OpenMPI
mpirun -machinefile $PBS_NODEFILE -np $PBS_NP ./hello
```

requesting resources

- How much memory we intend to use:

```
#PBS -l mem=1gb
```

- How long the job should run:

```
#PBS -l walltime=00:05:00
```

- How many nodes and how many cores per node:

```
#PBS -l nodes=8:ppn=1
```

- To which queue to submit:

```
#PBS -q edu_shared
```

the submit script continued

- Manipulate output and error in one single file:

```
#PBS -j oe
```

- Request email at begin, ends, or aborts.

```
#PBS -m abe
```

- Specify email address, name of job, and home directory:

```
#PBS -M netid@uic.edu
#PBS -N hello
#PBS -d /home/netid
```

Then submit, requesting 8 nodes as

```
qsub -l nodes=8 testhello.pbs
```

Commands `showq` and `checkjob` allow to monitor.