HPC at IU – Hands On

Abhinav Thota
Scientific Applications and Performance Tuning (SciAPT)
Research Technologies
Indiana University
How do I get access?

• I am sure you all already did this…
• Create an account:
  – [https://itaccounts.iu.edu/](https://itaccounts.iu.edu/)
  – Graduate students with hourly positions may not see BRII listed, please contact us for an account!
• Login domain names:
  • bigred2.uits.iu.edu
  • karst.uits.iu.edu
  • mason.indiana.edu
  – IU network ID and pass phrase
  – Read the message of the day (MOTD)

[https://kb.iu.edu/d/achr](https://kb.iu.edu/d/achr)
Let’s get our fingers dirty

- We will do the following on Karst today:
  - Copy something from your desktop to Karst
  - Change your environment
  - Submit a job to the queue to run something
  - Get an interactive node and launch a GUI application

- This covers the most common activities on HPC systems
SSH and SCP

• Sounds great, where do I login? Not in a browser 😞
• SSH is used to login to a remote machine
• SCP is used to copy files to and from a remote machine
• Example: (in a terminal, different on Windows)
  ssh username@karst.uits.iu.edu

  scp local-file username@karst.uits.iu.edu:/path/to/remote/dir

https://kb.iu.edu/d/aelc
https://kb.iu.edu/d/agy
You need clients, at least on Windows

- For SSH: Putty on Windows (http://www.putty.org/)
- For SCP: Winscp for windows (http://winscp.net/eng/index.php)
  - Filezilla is cross platform (https://filezilla-project.org/)
- On Mac and Linux: you can use the terminal for SSH
  - Use the terminal for SCP as well
  - Or use Filezilla for GUI interface
Environment – in your login session

• Home directory:
  – /N/u/<username>/Karst

• Scratch space for temporary files:
  – /N/dc2/scratch/<username>

https://kb.iu.edu/d/avmj
Task 1: Login and transfer files to Karst

- ssh username@karst.uits.iu.edu
- A. Copy something from your desktop/laptop to Karst
  - Use scp in the terminal or use an scp client
- B. Download something from the web onto Karst:
  - wget lets you download from the web
How to find and use software

- module command for adding software to your environment
  - module avail
  - module list
  - module add <key>
  - module rm <key>

- Permanently configure your environment via the .modules file
- Modules arranged in groups, development, applications, life sciences, etc.

https://kb.iu.edu/d/bcwy
What does “module load” do?

- Makes the software ready/available for use
  - Makes the software findable
- Does not start or run the application
- There isn’t anything like this on Windows

https://kb.iu.edu/d/bcwy
Task 2: Play with modules

- Check “module list”
- Check “echo $LD_LIBRARY_PATH”
- Add intel to your programming environment
  - module add intel/15.0.1
  - Check “module list”
  - Check “echo $LD_LIBRARY_PATH”
- Remove intel from your environment
  - module rm intel
  - Check “module list”
  - Check “echo $LD_LIBRARY_PATH”
Contd…

• Load the gnu openmpi module
  • `module load openmpi/gnu/1.8.4`
  • Check “module list”
  • Loads the libraries needed to run parallel applications

• Make this change permanent
  • Edit .modules in your $HOME
  • Common editors: vi, nano, emacs
  • Output of “cat .modules” should look like this:
    – “module load openmpi/gnu/1.8.4”
    – “cat” prints contents of a file
Sample PBS script

• At the most basic level, on Karst:

```bash
#!/bin/bash
#PBS -l nodes=2:ppn=16,walltime=30:00
cd /path/to/working/dir
mpirun -np 32 -machinefile $PBS_NODEFILE ~/bin/binaryname
```

• mpirun – the command used to run applications in parallel
• np – number of processors; machinefile – list of processors
• Put these lines in a text file and run “qsub script.sh”

https://kb.iu.edu/d/avmy
Queue commands

- Some other useful commands to see what’s going on with the queues:
  - `qstat -Q` #shows available queues
  - `qstat -u username` #shows the status of your queued jobs
  - `showq -i` #shows what’s going to run next
- Use “#PBS -q” to specify a particular queue in the PBS script
  - Need to do this on Big Red II, to specify the cpu or gpu queue (default is cpu)
  - Not always required on Karst, the default queue routes the jobs
    - But needed if you want to use the debug or interactive queues

[Link: https://kb.iu.edu/d/avmy]
On Karst or Mason (non-Cray machines)

- There is a login node(s)
- There are compute nodes
- You build, install and setup your job environment on the login node
- Run the job on the compute node
- Using mpirun if parallel or just the binary if it is serial
  - That is:
  - mpirun –np 32 app_name
  - Or just:
    - ./app_name
On Big Red II and other Cray’s

- Login, aprun and compute nodes

https://kb.iu.edu/d/bdkt
ESM – Extreme Scalability Mode

• No compromise *scalability*
• Low-Noise Kernel for scalability
• Native Comm. & Optimized MPI
• Application-specific performance tuning and scaling

CCM – Cluster Compatibility Mode

• No compromise *compatibility*
• Fully standard x86/Linux
• Standardized Communication Layer
• Out-of-the-box ISV Installation
• ISV applications simply install and run

Launch everything with aprun or ccmrun, whether it is parallel or serial
Contd.

- mpirun becomes aprun
- Flags and options are slightly different, but accomplish the same tasks
- Think of the aprun nodes as management nodes for the compute nodes
Interactive jobs

- For graphical applications – you can launch GUIs
- For testing
- `qsub -l`
  - Followed by options to specify walltime, queue, etc.
  - `qsub -I -l walltime=00:30:00 -q interactive`
- Straightforward on Karst
- Will land on an aprun node on Big Red II
  - module load ccm
  - Need to ccmlogin for the compute node

https://kb.iu.edu/d/bdsi
## Compilers

<table>
<thead>
<tr>
<th></th>
<th>Big Red II</th>
<th>Karst</th>
<th>Mason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray c</td>
<td>module load PrgEnv-cr</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Cray c++</td>
<td>cc</td>
<td>not available</td>
<td>not available</td>
</tr>
<tr>
<td>Cray fortran</td>
<td>Ftn</td>
<td>not available</td>
<td>not available</td>
</tr>
<tr>
<td>GNU c</td>
<td>module load PrgEnv-gnu</td>
<td>module load gcc</td>
<td>module load gcc</td>
</tr>
<tr>
<td>GNU c++</td>
<td>cc</td>
<td>gcc</td>
<td>gcc</td>
</tr>
<tr>
<td>GNU fortran</td>
<td>ftn</td>
<td>g++</td>
<td>g++</td>
</tr>
<tr>
<td>GNU gfortran</td>
<td></td>
<td>gfortran</td>
<td></td>
</tr>
<tr>
<td>Intel c</td>
<td>module load PrgEnv-intel</td>
<td>module load intel</td>
<td>module load intel</td>
</tr>
<tr>
<td>Intel c++</td>
<td>cc</td>
<td>icc</td>
<td>icc</td>
</tr>
<tr>
<td>Intel fortran</td>
<td>ftn</td>
<td>icpc</td>
<td>icpc</td>
</tr>
<tr>
<td>Intel ifort</td>
<td></td>
<td>ifort</td>
<td></td>
</tr>
<tr>
<td>PGI c</td>
<td>module load PrgEnv-pgi</td>
<td>module load pgi</td>
<td>module load pgi</td>
</tr>
<tr>
<td>PGI c++</td>
<td>cc</td>
<td>pgcc</td>
<td>pgcc</td>
</tr>
<tr>
<td>PGI CC</td>
<td>CC</td>
<td>pgCC</td>
<td>pgCC</td>
</tr>
<tr>
<td>PGI ftn</td>
<td></td>
<td>pgf77 or pgfortran (Fortran 90/95)</td>
<td>pgf77 or pgfortran (Fortran 90/95)</td>
</tr>
</tbody>
</table>

Java and MPI wrapper compilers (Open MPI and MPICH) are also available.

For more information, see "What compilers are available on the IU research systems?" [kb.iu.edu/d/abby](https://kb.iu.edu/d/abby)
Task 3: Submit a job to the queue

- Download an example program and job submission script:
  - cd /N/dc2/scratch/username
  - tar xvf job-submission.tar
- Change directory to job-submission
  - cd job-submission
- Compile the mpi_hello.c program
  - mpicc mpi_hello.c
    - Will create a binary “a.out”
    - Run “ls” to verify
    - “mpicc” is the MPI wrapper used to compiler MPI programs
• Edit the pbs.sh file to set the working directory and binary name
  • It will not work out of the box
  • You can check the current working directory by running “pwd”
  • Add “cd /N/dc2/scratch/username” to the script
  • Change “binary_name to a.out”
• Submit the job to the queue
  • “qsub pbs.sh”
• Change the PBS script to now submit the job to run across 2 nodes
  • #PBS -l nodes=2:ppn=16
  • ”qsub pbs.sh”
• Check output in the working directory or in your HOME directory if you did not specify this in your PBS script
Launching GUI apps

- For X forwarding in SSH to work you must be running an X server program
- Most Linux systems have X server installed by default
- On Windows, can use Xming: http://sourceforge.net/projects/xming/
- On Mac, can use Xquartz: http://xquartz.macosforge.org/landing/
- Start these apps first before you launch SSH app
- Enable X11 forwarding by checking a box in your SSH app
Task 4: X forwarding exercise

1. On Karst, login with X forwarding enabled
   1. Launch xterm
      1. ssh –X username@karst.uits.iu.edu

2. On Karst, start an interactive job with X forwarding enabled
   1. Launch xterm from the compute node
      1. qsub –l –X –q interactive –l nodes=1:ppn=1

Hint: Find step by step instructions on the KB pages