Working with HPC and HTC Apps

Abhinav Thota
Research Technologies
Indiana University

RESEARCH TECHNOLOGIES
INDIANA UNIVERSITY
University Information Technology Services

PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY

Outline

- What are HPC apps?
  - Working with typical HPC apps
  - Compilers
    - Optimizations and libraries
  - Installation
  - Modules
- What are HTC apps?
  - Dealing with HTC apps
  - Tools available
  - Benefits and things to keep in mind

What's so hard about working with HPC apps?

- It's not, if :
  - you developed the application yourself
  - were involved in the development
  - had training on how to use the application
  - there is really good documentation on the web
- Many of the popular Linux applications are easy to figure out because there is a lot of documentation on the web
- But HPC is a small world and even the popular HPC applications are not as popular as your average windows or mac application, unfortunately

Most apps are 3rd party

- What are 3rd party apps?
  - The software that you or us did not develop
  - The software that did not come with the OS or from Cray
- Why do we care about them?
  - Most of the applications that people run and use are 3rd party
  - Most people don't develop their own software
  - Could be applications, libraries, compilers, etc.
- How do we deal with them?
  - Will talk about installation, usage, etc.
  - How to get support?
- Discuss examples of CPU and GPU apps
- What do we have at IU?

Examples of software at IU

- NAMD, AMBER, GROMACS, LAMMPS
- WRF, SAS, MATLAB, R
- Compilers
    - GNU, PGI, Intel, Cray
- Not just applications, but tools like debuggers, tracers and libraries such as Boost, NetCDF, Vampir
- OpenMPI, mpich, etc.
- We might have expertise in installing these applications, but we are just installing software developed by someone else most of the time
- We are also not root on the machine
- We are not domain science experts!
- Software request form: http://rt.uits.iu.edu/systems/SciAPT/software-request-form.php

Installing software in a Linux Environment

- Who here has installed anything in a Linux/Unix environment?
  - Installing on HPC machines is not much different than installing on your Linux desktop
    - But, you are not root
    - Can't install RPMs, can't install in /usr/bin or /usr/local
    - Must specify non-default location
- We do system wide installs in /N/soft
  - /N/soft/rhel6 – for Quarry and Mason, which run RHEL 6
  - /N/soft/cle4 – for Big Red 2, which runs Cray Linux Environment 4.2

Common procedures

- Compiling is installing
  - gcc main.c gives you a.out
    - a.out is the binary that you just installed
    - More complex applications involve more steps
- Common procedures are:
    - configure, make, make install is the most common
    - cmake
    - Bjam
    - rpms
    - Binaries

Choosing a compiler

- We have GNU, Intel, PGI and Cray
- What should you pick?
    - No straight answer
    - Depends on the code and CPU make
    - GNU compilers are the most forgiving, but not generally known for performance
    - Most applications build with Intel and PGI and generally Intel and PGI perform well
- Configure – options, install location, libraries etc

What is high throughput computing or HTC?

- Large number of independent serial or parallel jobs
- Generally in the hundreds or thousands
- Parameter sweeps, multiple branches of the same problem, etc.
- Traditionally run through high throughput grids
    - Examples include Open Science Grid (OSG), Condor, Boinc, Folding@home
- You can get started on any of these grids
    - Volunteer computing
    - If you are interested, can contribute spare desktop or Xbox/PS3 cycles

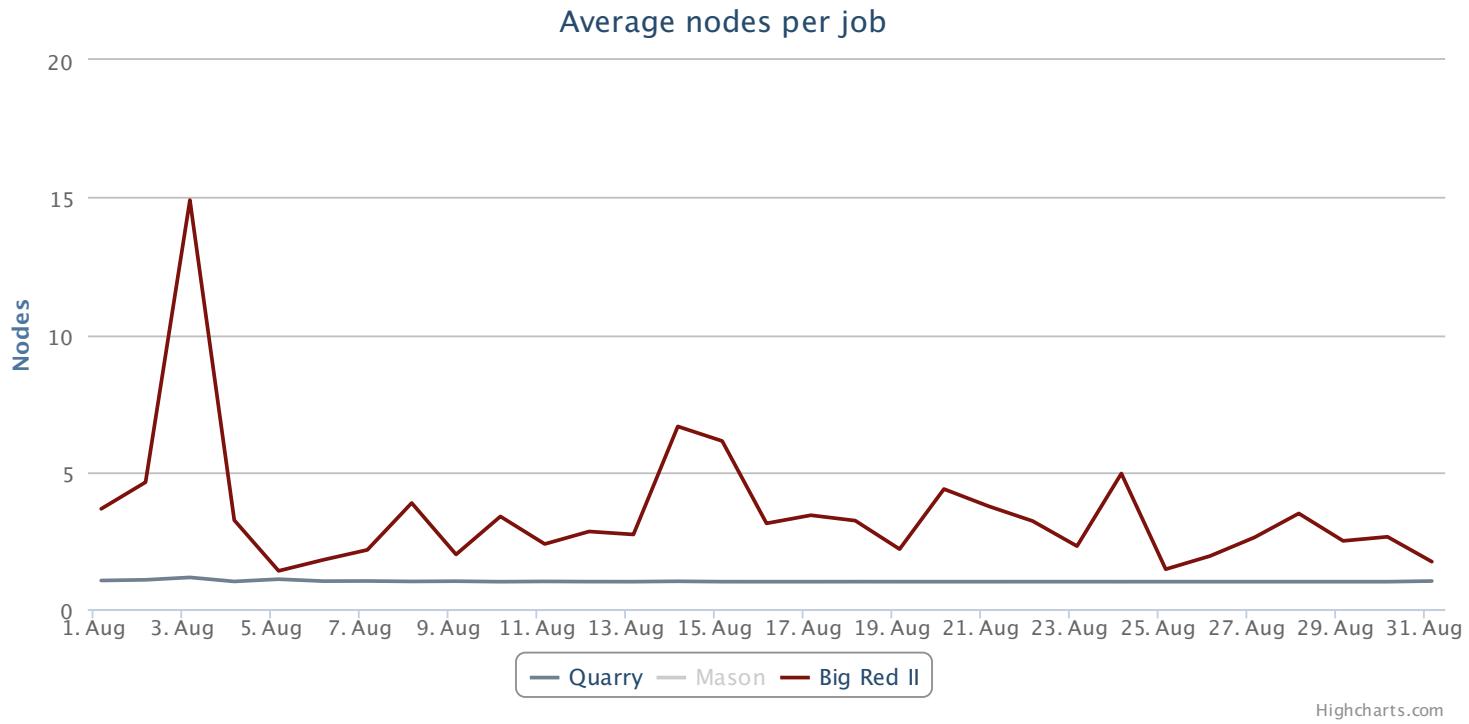
What's out there?

- Boinc is a project run by UC Berkeley and Folding@home is run by Stanford
- More than half a million people contribute to these projects
    - But they don't allocate resources
- OSG or Open Science grid open to everyone
    - Doesn't really allocate resources
        - Best effort basis
        - Different priorities for those who contribute resources and for those who just receive them
        - Can get started directly through OSG or via XSEDE

Where do you do this at IU?

- Quarry is best suited for this kind of workload
- Your jobs can share a node
    - If you request just one core on a node, the rest of the cores are used by your other single core jobs
- Just keep submitting single core jobs
- But Quarry has only so many cores
    - 218 nodes with a total of 1744 cores
        - 8 or 12 cores, 16 or 24 GB of memory
- You may on the other hand need more than what Quarry can offer you
- Big Red II is the other option
    - BR 2 is not designed for this purpose though
    - But there are ways to make this work

Average nodes requested per job on BR 2 & Quarry



What does the graph say?

- Looks like a lot of single node jobs on both BR 2 and Quarry
- It doesn't matter all that much on Quarry as the node are shared
- But how many of these jobs are using all the cores on a single node of BR 2 that they are requesting?
    - Probably not a lot
    - The immediate question is, what can we do to maximize utilization?
    - We can obviously ask serial users to stop using BR 2 and move to Quarry
    - But there are only so many cores available on Quarry
    - The better solution, that does not inconvenience anyone is to offer a tool which can help them do this BR 2

What's so important about using all the cores on a node?

- Intra-node parallelism
- Quarry has 8 or 12 cores per node
    - All the nodes are shared among the jobs of a single user , so cores are not wasted
- Nodes on BR 2 are not shared
    - 32 or 16 cores per node
    - Whether you use 1 core or 32 cores, you own the node
    - You will be wasting 95% of the node if you use only a single core
    - This is a big problem!
- Please be a good citizen and don't do this!

On Big Red II

- Big Red II has more than 21,000 cores available
    - So that is attractive
- If you have a large bunch of serial jobs that you need to run, use all the cores on a node
- Let's see how to do that
- Contact us if you are unsure how to do this!

The kinds of jobs that fit

- Let's once again go over the kind of jobs that we are talking about
- Prerequisites:
  - Have a bunch of independent serial or parallel jobs
  - Independent meaning jobs that can be run without waiting for results from other jobs
  - Ideally, all these jobs essentially take the same amount of time to complete, within a few minutes of each other
    - So that we don't waste resources waiting on a few jobs
  - Low memory requirements, ~2 GB per job on BR 2
    - 32 cores and 64 GB of memory per node.. Each job can get up to 2 GB

Some of the applications that need this

- Parameter sweeps
- Many instances of the same task (ensemble applications)
- Image processing
- Geography apps

Tools available

- Bash scripting on Quarry or BR 2
  - This can be easily accomplished with some creative scripting within your PBS job script
- PCP on BR 2
  - We have a tool called PCP (parallel command processor) that takes care of the scripting (some if it) for you
- BigJob/Radical Pilot for more complex workflows
  - We also have Bigjob, comes with the python module on BR 2 and Quarry
  - If you have more complex workflows, where your jobs need to interact with each other, wait on output from one job to use it as input for a different job
  - Do load balancing between multiple jobs, etc.

Bash Scripting

- You basically put all the commands you want to run in a file
- And run that file, as simple as this:
  - aprun -n 1 ./test_32.sh
  - Where test_32.sh is:

```
#!/bin/bash
./binary > test_32.out1 &
./binary > test_32.out2 &
./binary > test_32.out3 &
.
.
.
./binary > test_32.out32 &
wait
```

A couple of notes

- You probably need a script to generate the script!
- Not at all difficult, can practically copy and paste code from the web
- The “&” backgrounds each of the serial jobs
- The “wait” at the end makes the script wait till all of them return
    - Making sure the script does not prematurely return, ending the job
- You will probably be passing options and flags to the binary that are different for every serial job

A couple of benefits

- Simplify job management?
  - Instead of submitting tens or hundreds of jobs to the scheduler, you can now bundle them and run with a single submission
  - Potentially makes job management and tracking easier
- Not really an issue at IU, but many places prioritize and give discounts and bonuses to people running large jobs
  - At the same time penalize people running small jobs
  - It is easier for the admins and scheduler to optimally schedule fewer large jobs than lots of small jobs
  - This happens at the large XSEDE and national centers
    - The machines are purchased to do big science
    - Just because you have lots of small jobs should not disqualify you
    - But if it does, you have a workaround!

PCP – Parallel Command Processor

- This is a nice little tool that let's you run multiple serial jobs on a single node
- Not all that different from bash scripting when it comes down to it
- You still have to create a script file containing all the commands that you want to run
- This works across nodes
- You don't have to background all the command and wait for them to return
- Going to look something like this:
    - aprun –n 32 pcp script.sh
    - Where script.sh will look the same as the previous script, just without the &'s and wait at the end
- Will not work with applications that need CCM

https://kb.iu.edu/d/bdka

Questions?