Week 8: Introduction to Stampede2 Ctd.

Computational Tools and Techniques in STEM

Apr 2-4, 2019

Learning Goals

- **L1**: Formal introduction to unix shell.
- **L2**: Changing .bashrc.
- **L3**: Xsede account charging.
- **L4**: Batch jobs.
- **L5**: Batch submission script.
- **L6**: Monitoring queue.
- **L7**: Setting up Globus account.
- **L8**: Using command line arguments.
- **L9**: Writing output.
- **L10**: Trapezoid Rule
- **L11**: Exercise.

Unix Shell

Motivation: Graphical user interfaces (GUI) are easy to learn and use for simple tasks. But when the tasks are complex, we need a command line interface (CLI). A shell provides a simple language and a CLI with which you can use run commands.

What is a unix shell?

A Shell is program that reads commands and run programs. Every bash command corresponds to a program stored on the computer and bash knows where to look for that program in your computer.

Are there different kinds of shell?

Yes! But the most popular one is Bash (Bourne Again SHell). It is the default shell on most unix systems.

Advantages of Using Shell

- Automation of routine tasks.
- Combining commands into single command (pipelining).
- Portable across any Unix-like operating system.
- Can handle large volume of data.
- High action-to-keystroke ratio.

Disadvantage: Can be hard for beginners and the commands can be cryptic.

PATH

To show the path:

```
$echo $PATH
```

The $ sign before the variables is used to evaluate and output the value of the variable. By default, an initial set of path names are always assigned to PATH. But these can be removed and added from PATH either temporarily or permanently.

To add a path in your home dir permanently, open .bashrc and add the following lines:

```
# this adds anaconda to the path
export PATH="/home/prao/software/install/anaconda/
    bin:$PATH"
```

If you use this command in the terminal window, this path will be added to the PATH variable until the session expires.

As a precaution, to confirm with the user before deleting any files, add this to your .bashrc file:

```
alias rm='rm -i'
```

Then source the .bashrc file for the change to take effect:

```
$source ~/.bashrc
```

To add a module permanently to the user's environment (say python3), create a file .modules in your home dir and then add the following line in that file:

```
module load python3
```

```
module load python3
```

Then source the file from the command line for the changes to take effect:

```
source ~/.modules
```

Xsede Account Charge

- The account is charged based on node hours.
- One service unit (SU) represents a single compute node used for one hour (a node-hour).

SUs billed (node-hours) = (# nodes) x (job duration in wall clock hours) x (charge rate per node-hour)

Production Queue and Job Submission

Queue Name	Node Type	Max Nodes per Job (assoc'd cores)*	Max Duration	Max Jobs in Queue*	Charge Rate (per node-hour)
development	KNL cache-quadrant	16 nodes (1,088 cores)*	2 hrs	1*	0.8 Service Unit (SU)
normal	KNL cache-quadrant	256 nodes (17,408 cores)*	48 hrs	50*	0.8 SU
large**	KNL cache-quadrant	2048 nodes (139,264 cores)*	48 hrs	5*	0.8 SU
long	KNL cache-quadrant	32 nodes (2,176 cores)*	120 hrs	2*	0.8 SU
flat-quadrant	KNL flat-quadrant	32 nodes (2,176 cores)*	48 hrs	5*	0.8 SU
skx-dev	SKX	4 nodes (192 cores)*	2 hrs	1*	1 SU
skx-normal	SKX	128 nodes (6,144 cores)*	48 hrs	25*	1 SU
skx-large**	SKX	868 nodes (41,664 cores)*	48 hrs	3*	1 SU

Figure: `https://portal.tacc.utexas.edu/user-guides/stampede2#table5`

Batch Jobs

Batch jobs do not start immediately.

They wait in the queue until the requested resources become available.

Stampede2 uses SLURM for the workload management.

For this, we create a script containing SLURM directives that specifies:

1. computing resources requested
2. job to run

When the job gets accepted, it will run in the environment which includes your currently loaded modules and the current working directory.

Slurm directives begin with “#SBATCH”, followed by an option.

Batch Submission Script

```
#SBATCH -J myjob      # Job name
#SBATCH -o myjob.o%   # name of std output file
#SBATCH -e myjob.e%   # name of stderr error file
#SBATCH -p normal     # Queue name
#SBATCH -N 1          # Total # of nodes (must be 1 for serial)
#SBATCH -n 1          # Total # of mpi tasks (should be 1 for
    serial)
#SBATCH -t 00:01:00   # Run time (hh:mm:ss)
#SBATCH --mail-user=myname@myschool.edu
#SBATCH -A TG-DMS190011 # Allocation name (in case more than
    one)

# Launch serial code
python hello_world.py
```

Submit the batch job from the login nodes using:

```
$sbatch myjobscript
```

Monitoring Jobs and Queues

To show all jobs in all queues:

```
$squeue
```

To show all jobs owned by a specific user:

```
$squeue -u <username>
```

The column labeled “ST” displays each job’s status: “PD” means pending, “R” means running, “CG” means completing.

To cancel a job:

```
$scancel <jobid>
```

To get more info on squeue command:

```
$man squeue
```

Globus File Tranfer

Go to the page and follow instuctions to setup a globus account:
https://portal.xsede.org/data-management

Using command line arguments

Command line argument: Arguments that you pass in your program from the command line at run time.

```
import sys

# total number of arguments passed
total = len(sys.argv)

# sys.argv is a list of all the arguments
# first argument is always the name of the python script
print str(sys.argv[0])

# all the command line arguments can be printed using:
print str(sys.argv)
```

Writing Output

To remind ourselves how to write files:

```
import numpy as np
fileout="full/path/to/file"
# save the result into a text file
np.savetxt(fileOut, [sum_f])
```

Exercise: Trapezoid Rule

On your local desktop, write a program to calculate the numerical integration of the function using the trapezoid rule.

$$f(x) = x^3$$

Basic strategy:

1. Divide the interval of integration into subintervals.
2. Apply the midpoint rule into each of the subintervals.
3. Add the result from each of the subintervals to obtain the final value of the integration.
4. For reference:
   `https://en.wikipedia.org/wiki/Trapezoidal_rule`

Exercise: Trapezoid Rule

- Add the lower and upper bounds of the interval of integration as command line args.
- Add the number of intervals in the program as command line arg.
- Print all the command line args on screen.
- Dump the final result to a file in your $WORK on stampede2.
- Put the command line args as the first line in the output file.
- Compare the numerical result to the analytical result and if not within tolerance, increase the number of steps adaptively within the program.
- Move the files to Xsede.
- Run it as an interactive job.
- Submit this script as a batch job.