XSEDE Bridges2 User Guide

implementing super-computing on your PC in great detail, Peichen Li 10/02/2021

https://www.psc.edu/resources/bridges-2/user-guide-2/

Topics

- What
- Why
- How
- Main Takeaway

What

- Bridges-2, PSC's newest supercomputer, began production operations in March 2021. It is funded by a $10-million grant from the National Science Foundation.
  - converged HPC + AI + Data
  - custom topology optimized for data-centric HPC, AI and HPDA
  - heterogeneous node types for different aspects of workflows
  - CPUs and AI-targeted GPUS
  - 3 tiers or per-node RAM: 256GB, 512GB, 4TB
  - extremely flexible software environment
  - community data collections & Big Data as a Service

Why

- Fast
  - high-RAM capacity
  - parallel computing (can process thousands of datasets simultaneously)
- Free
  - AWS very costly and slow
  - to apply/renew/upgrade, a research proposal is needed
  - Bridges-2 Regular Memory (PSC) with core-hours
- Solution to empirical research involving Big Data
  - thousands of datasets, with each amount to several gigabytes (e.g. TAQ)
  - with XSEDE Bridges2, data query could only take several mins

How (Our Focus)

- Setup
  - XSEDE and PSC account
  - WinSCP, PuTTy and WSL
- Connecting to Bridges2
  - WSL / Terminal for running programs
  - WinSCP for managing files
- Implementation
  - shell
  - copy files from/to Bridges2
  - launching interactive sessions
  - submitting batch jobs
- Logistics and sample projects

Setup - Account

- XSEDE portal account
  - register your XSEDE account
  - please provide your account name to your PI(s) as to join their project folder
- PSC account
  - to connect to Bridges-2, you have to set PSC password
  - this password will be used * frequently * going forward
  - more on PSC password policies

Setup - Software

- Windows users: either install both PuTTy and WinSCP, or activate windows subsystem for Linux (WSL)
  - I prefer the latter one with a Ubuntu terminal
  - check out manual installation steps for older versions of WSL
- Mac OS / Linux Users: Make sure you know how to open terminal on your computer
  - install WinSCP

Connecting to Bridges2

- PuTTY and WinSCP: hostname bridges2.psc.xsede.org, port 22
  - and then enter your PSC password
- Terminal (WSL Ubuntu): ssh [username]@bridges2.psc.xsede.org
  - when you type your PSC password, your input will not be displayed
  - be confident and go ahead
  - you can try 3 times before server asks you to “calm down”

Implementation - Shell

- Common backbone of all computing
  - on a super-computer: often via a remote connection
- A few key commands
  - cd [dir] change directory (or back to home)
    - cd /ocean/projects/ses190002p
  - ls list files (and / or directories)
    - ls -laht
  - pwd print working directory (“where am I”)
  - cat [filename]: show content in file
  - head –n [filename]: show first n lines

Implementation - Shell

- Use tab to auto-complete to save time
- adding --help to a command (try it, e.g. ls --help)
- of course Google and StackOverflow ...
- Recommended materials Getting started with HPC and The Unix Workbench

Implementation - Transfering Files from/to Bridges2

- WinSCP
  - Login to bridges2.psc.xsede.org with port 22
  - Direct file transfer(~10 MB/s)
- Mac / WSL / Linux
  - Open terminal and used command scp or rsync
- Globus
  - Pro: Very, very fast
  - Con: some complexity in setting up

Implementation - Launching Interactive Sessions

- You can do your production work interactively on Bridges-2, typing commands on the command line, and getting responses back in real time.
  - just like your Python/R console
- **interact -p RM -t 01:00:00**
  - **-p RM**: take control of all 28 cores of one node
  - **-t 1:00:00**: you interact session will be auto-killed after 1 hour (help control consumption of resources)
- **interact -n 20 -t 01:00:00**
  - the usual command I use for all my Python/R scripts
  - if the data gets too large, try **-n 50**

Implementation - Submitting Batch Jobs

- Instead of working interactively on Bridges-2, you can instead run in batch. This means you will
  - create a file called a batch or job script
  - submit that script to a partition (queue) using the **sbatch** command
    - sbatch query_debit-card-spending.batch
  - wait for the job's turn in the queue
  - if you like, check on the job's progress as it waits in the partition and as it is running
    - use **sacct**
  - check the output file for results or any errors when it finishes
    - i.e. record of all program output in slurm-xxx.out

Implementation - Example Batch File (Python)

```
#!/bin/bash
#SBATCH -t 00:30:00
#SBATCH -p RM-shared
#SBATCH -N 1
#SBATCH --ntasks-per-node 20
#SBATCH --array=0-15

set -x
cd /ocean/projects/soc210002p/peichen3/Facteus/code/

# Load software
module load anaconda3

# Run python script with a command line argument
python query_debit-card-spending.py $SLURM_ARRAY_TASK_ID
```

Implementation - Example Batch File (Python)

- This .batch file allocates **20** cores per node, and apply each of **0-15** to **$SLURM_ARRAY_TASK_ID**
- To do parallel computing for all datasets, you probably want to come up with an idea on how to map each of these integer command line arguments to the filename you want
- One general approach is to write a spreadsheet and import this spreadsheet, which reflects mapping relationships, in the Python/R scripts

Implementation - Example Batch File (R)

This .batch file allocates **12** cores per node, and apply each of **0-1000** to **$SLURM_ARRAY_TASK_ID** (parallel computing)

```
#!/bin/bash
#SBATCH -t 00:30:00
#SBATCH -p RM-shared
#SBATCH -N 1
#SBATCH --ntasks-per-node 12
#SBATCH --array=1-1000

set -x
cd /ocean/projects/ses190002p/peichen3/

R --slave < get_taq_5min_79.r --args $SLURM_ARRAY_TASK_ID
```

Logistics

- Administrators pre-installed a lot of things
  - R, Python, Matlab, and their commonly used packages; Stata not available
  - module load [package] is the command to load pre-installed software
    - e.g. for Python, use **module load anaconda3**
- Recall we have two ways of parallel computing:
  - Multi-node parallel - take a whole CS class to learn
    - e.g. the parallel computing packages **parallel** and **foreach** in R
    - for Python, refer to **multiprocessing** and **Dask**
  - Multi-core parallel - how Bridges2 works (i.e. parallel by job management system)

Logistics - Job Management System vs. Parallel by Packages

- **Job Management System**
  - One slurm job, one core
  - Each core get its own partition of memory (128GB/28=4.5GB)
  - Easy interruption recovery
  - Good for independent jobs/data
    - Financial data are usually easy to be split to stock-days
- **Parallel by packages (e.g. R)**
  - One slurm job, multiple core
  - 128GB memory shared for 28 cores
  - Lose progresses if interrupted
  - Good for dependent jobs
    - E.g. cross sectional patterns for stocks need interactions

Main Takeaway

- Takes some effort to learn at first
- Parallel computing is super useful
- Essentially "free" for research
- Unfortunately does not support Stata
- Not stable