SLURM Job Submission: Advanced Users Module

Overview

The goal of this module is to discuss key features of SLURM job submission including measuring CPU and memory (RAM) usage, submitting multiple jobs in parallel, and best practices for ICS members planning to regularly use blanca-ics compute nodes.

If you have not already done so, please review the core INC data and analysis training module here . We will build on the foundation established in the prior module in this module.

You should already be able to:
- Log into a blanca or summit login node using secure shell (ssh)
- View and edit files either using terminal text editors such as vim, nano, or with GUI applications such as Sublime (requires VNC connection), or by mounting the petalibrary filesystem via SMB or SSHFS
- Submit a compute job via SLURM
- Perform basic functions to check the progress of SLURM job, cancel or pause job.

Today we will complete the basic training with the following:
- Launch a SLURM job array and control the number of parallel jobs running
- Launch multiple SLURM jobs set to run on only a subset of blanca-ics compute nodes
- Check CPU and memory usage for recent job
- Discuss best practices for running jobs on blanca-ics

High Performance Computing (HPC)

An HPC cluster is a collection of many separate servers (computers), called nodes, which are networked together. There are a number of different specifications for each computers (or nodes) that could be specialized for different tasks such as GPU nodes, high speed processor nodes, etc.

High performance computing clusters generally include:

- head / login node - used as the interface between the compute environment and users
- a specialized data transfer node (DTN)
- regular compute nodes (where majority of computations is run)
- GPU nodes (on these nodes computations can be run both on CPU cores and on a Graphical Processing Unit)

All cluster nodes have the same components as a laptop or desktop: CPU cores, memory and disk space (and sometimes GPU cores). The difference between a personal computer and a cluster node is in quantity, quality and power of the components. Importantly, cluster nodes are networked together so you can use resources from multiple nodes in tandem if desired.

Common SLURM Commands and Settings

SLURM is the job scheduler used on the University of Colorado at Boulder HPC cluster. This system is used to run any compute job on the HPC nodes, and manages the resources of all nodes across the cluster. You should be familiar with basic SLURM commands after completing our basic training module here. We will discuss three primary SLURM commands and the most commonly used flags (or options) for each command.

`SBATCH` is used to run a set of instructions on a compute node that has been already stored as a shell script. This is the most commonly used method to run computations on HPC. Here is a list of critical flags or options:

`-q [--qos]` quality of service
`-p [--partition]` cluster partition
`-A [--account]` cluster sub-account information
`-J [--job-name]` jobname for reference job status
`-o [--output]` filepath for output log, use special characters to include job details in filename (`%j` job id)
`-e [--error]` filepath for error log, use special characters to include job details in filename (`%j` job id)
`-c [--cpus-per-task]` number of CPUs or cores requested for each job step
`--mem, --mem-per-cpu` total allocated memory (RAM) or memory per cpu
`-t [--time]` wall time for job completion
`-w [--nodelist]` lets of node names that should be used to run job
`--array` set the job to run as an array

`SINTERACTIVE` is used to launch an interactive session of a compute node. Here you can interactively submit and review the progress of a job running. This is a good option when piloting an analysis or running computation that requires periodic user input. The same set of flags (options) should be used as `SBATCH` to define job configuration.

`SQUEUE` is used to check the progress of jobs currently running on the HPC cluster. You can use this to check the status of jobs you submit as well as what nodes are in use or idle. Critical flags include:

`-q [--qos]` quality of service
`-p [--partition]` cluster partition
`-A [--account]` cluster sub-account information

All Blanca Compute nodes are set up with multiple CPUs, internal memory (RAM), and small disk storage. When submitting a compute job, you want to be sure to request the proper resources to ensure your job completes correctly and in a timely manner. There are a total of 388 cores owned by ICS in which our users have priority. ***Unless otherwise specified***, all computations are run with 2 gigabytes of RAM per CPU and a wall time of 24 hours (**`--mem-per-cpu=2G, --time=24:00:00`**). If users need additional resources you must specify additional resources.

Best Practices for Blanca-ICS Compute

We are excited to support all INC members in utilizing the ICS services including blanca-ics compute and petalibrary storage. We are always available to answer any questions, but you as the user will be the one making many decisions about when and how to use these resources.

Here are a few rules of thumb to keep in mind:

1. Any large compute job (matlab, python, etc) **SHOULD NOT** be run on the login nodes (blogin-ics2, blogin01). If you are not sure how to run your computation on the compute nodes, please ask!
2. Be fair about the use of blanca-ics (our priority nodes). These resources are shared among many users so please do not overwhelm the queue for days at a time.
   *****Rule of Thumb** Users should allocate no more than 30% (or 128 CPUs) of the available resources at any time (exceptions are short <2 hour jobs)***
3. Consider running your job on other queues (blanca-ccn, blanca-ibg) if there are no resources available on blanca-ics.
4. If you need to reserve large amounts of resources for a quickly approaching deadline, please let Lena Sherbakov or Amy Hegarty know to help facilitate resource sharing.

CPU and Memory Usage (SLURM Jobs)

Critically important for large computations is to estimate the required resources, and allocate those resources appropriately. Using high performance computing, we have the option to run job computations that require massive amounts of resources not available on a local computer. These resources are (unfortunately) not unlimited, and therefore we should be mindful when setting up each computation to ensure efficient allocation of resources.

There are many ways to estimate resources required for your job, and you can perform your own research on this subject starting here. For our purposes, we will discuss only two key steps in job sizing.

1. Check what capabilities exist for your underlying software or tool (multi-core processing?).
2. Test your job either with a small representative sample, or in short segments of the larger analysis design.

After testing your job, how can you tell how many CPUs (or cores) and how much memory is right for your full analysis? ***Measure it!*** Unfortunately, in most cases the best way to figure out how much memory and CPU load is required, you need to just test the job and check how many resources were used.

In Blanca, we can easily check how much active cput time (UserCPU) was used and the maximum amount of memory required (MaxRSS) using the `sacct` command. You may choose to use a common tool developed for INC to check your job performance by loading the SLURM_TOOOLS module provided by ICS.

```
module use /projects/ics/modules
module load SLURM_TOOLS/0.0.1
```

Once you have added the SLURM_TOOLS module, you can simply use the command from the terminal. Here is an example of how use and interpret the job statistics tool.

```
bash $ jobstats -j 11859032
   JobID: ......... 11859032
   JobName: ....... recon-all
   Allocated CPUS: . 2
   Allocated Memory: 16000Mb
   CPU efficiency: .. 38.00% (used .76 cpu)
   Active Memory: .. 5205Mb
   Active CPU Time:  08:18:45
   Total CPU Time: . 21:20:14
```

Here we see that job #11859032 was a freesurfer reconstruction (recon_all) job. This job was run on 2 CPUs (or cores) and was allocated 16000Mb (16Gb) of memory. Both the CPU and Memory usage was well below the allocated.

Where would we go from here? … In the above example we can see that the requested memory (or RAM) was well above what was used for this analysis. We may want to run several more test jobs to confirm what the average performance is for this tool. We can then use this information to set a more reasonable limit on memory for this task, maybe 1Gb.

Submitting Multiple Jobs

In the sections below, we discuss ways to submit multiple jobs to the queue. Each method has its pros/cons that should be considered depending on the use-case. If after reading this document you have doubts about which method to implement for your own project, don't hesitate to ask us!

Job Array

Description: When submitting multiple jobs in a batch, there are a few methods to threshold the job submission to ensure the queue is not overwhelmed by running all of your jobs at the same time. One method, easily implemented if you are calling a script using **`sbatch`** is using a job array.

Use case: You are running identical steps (or interactions) of the same script that need to be run many times. You can set the number of instances to run, and how many jobs to run in parallel. This command can be easily adjusted to iterate through multiple input files / input data.

Example: Here is a simple example of how to set up a shell script that will be launched in SLURM as a job array with 100 iterations run in groups of 10. The body of the code here could be replaced by any custom code for your own use. In addition, we show how to change the number of jobs run in parallel using **`scontrol`**.

```
bash $ cat script.sh
   #!/bin/bash
   #
   #SBATCH --job-name=example
   #SBATCH --qos=preemptable
   #SBATCH --partition=blanca-ics
   #SBATCH --account=blanca-ics-XXXX
   #SBATCH --array=1-100%10
   #
```

```
    # --------- Your Code Here --------- #
    #
    # Set up job array index - calls subject number from list of inputs
    # **Option 1: Use counter as the job identifier...
    NUM=$SLURM_ARRAY_TASK_ID
    SUBJECT=$NUM

    # **Option 2: User counter to extract information from an
    #   accompanying file (e.g. subject list)
    NUM=$SLURM_ARRAY_TASK_ID
    SUBJECT=$(sed -n "$NUM"p $subjlist)
    echo $SUBJECT

    # **Option 3: User counter to extract files or subdirectories
    NUM=$SLURM_ARRAY_TASK_ID
    SUBJECT="$(cut -d' ' -f${NUM} <<<`ls $dir`)"
    echo $SUBJECT
--------------------------------------------------------
bash $ sbatch script.sh

bash $ # How to control the number of array jobs running during job
bash $ scontrol update JobID=<jobid> ArrayTaskThrottle=<NewArrayLimit>
```

Set Job Nodes

Description: Specifying a subset of the total possible blanca-ics nodes provides the user with the opportunity to launch multiple jobs but ensures those jobs do not allocate all available nodes (in blanca-ics). This is an alternative solution to using a job-array if your script is not well designed to be launched with a single iterator.

Use case #1: Launch many jobs at one time without overwhelming the blanca-ics queue and limiting other users from submitting jobs. This alternative should be used if your script is not well designed for using a job array.

Example: Here is a simple example of how to set up a shell script that will be launched with SLURM on a specified node list. The **`sbatch`** command is called from the terminal and will be called for each 'subject' in our analysis. The body of the code here could be replaced by any custom code for your own use.

```
bash $ cat script.sh
    #!/bin/bash
    #
    #SBATCH --job-name=example
    #SBATCH --qos=preemptable
    #SBATCH --partition=blanca-ics
    #SBATCH --account=blanca-ics-XXXX
    #SBATCH --nodelist=bnode0101,bnode0102,bnode0103
    #
    # --------- Your Code Here --------- #
    #
    # provide a subject id as input to the script
    SUBJECT=$1
    echo $SUBJECT

--------------------------------------------------------
bash $ subjects={001 004 005 007 009}
bash $ for i in $subjects; do
         sbatch script.sh $i
       done
```

Use case #2: Another case where specifying a node list can be useful is when you want your job to run, or avoid running, on specific hardware setups. For example, if I wanted run my job only on nodes with faster processing speeds, I could select to exclude the bnodes0101-bnode0105.

From the example above, Here's How…

Replace: `#SBATCH --nodelist=bnode0101,bnode0102,bnode0103`

With: `#SBATCH --exclude=bnode01[01-05]`

Use case #3: Another case where specifying a node can be useful is when you want to test the reproducibility of your job running on the same node multiple times. In this case we would want to specify a single node to run our job from.

From the example above, Here's How…

Replace: `#SBATCH --nodelist=bnode0101,bnode0102,bnode0103`

With: `#SBATCH --nodelist=bnode0101`

Computation Jobs With Other Softwares

Many of the examples here are written with the use of bash scripting. This is only one example of how you can run computations of HPC cluster. You will use similar syntax, and the same #SBATCH flags if you want to use python or any other scripting language. One other amazing resource to consider is CURC jupyterhub. Jupyterhub is an interactive jupyter notebook that is run on the summit or blanca computing core and has petalibary mounted. You may generate bash, python or R scripts that can be run interactively in your jupyter notebooks. For more information go to research computing documentation here.

Questions ???

If you have any questions, please feel free to reach out to Lena Sherbakov (Lena.Sherbakov@colorado.edu) or Amy Hegarty (Amy.Hegarty@colorado.edu).