Job Scheduling Using Slurm

A guide

Contents

Part 1

Why use Slurm?

What is the advantage of Slurm?

- Using Slurm allows you to run programs on the entire CCB cluster, not just the login nodes
- This can be an advantage in three different ways:
  - You can process larger data sets
  - You can run many independent programs at the same time
  - You can get a single program to run faster (sometimes)
- You may only need one of these advantages; at other times, you might benefit from two, or even all three
- Let’s look at these in a bit more detail

Slurm Scenarios

Scenario	Symptoms	How Slurm helps
Process larger data sets	• When you try to process a large data set your program tells you that it ran out of memory • You get an e-mail from us to say that you exceeded your memory limits	• You can run programs on systems with access to more memory (up to 1TB)
Run more programs	• You have lots of independent data to process and can't continue until it's all done • You have multiple independent analyses to process and want to do them all at the same time	• You can run many copies of the same program at the same time • You can run multiple different programs at the same time
Run one program faster	• You have one critical, slow step in your analysis and can't do anything else until it finishes	• Some[1] programs can use more CPUs to do a single job faster

[1] Some programs can't; we'll look into this in more detail later

Reasons not to use Slurm (yet)

- If none of those apply, you may not need Slurm (yet)
  - You don’t have any problems with the size of your data sets
  - You don’t have any need for your programs to run faster
  - You don’t need to run more programs at once
- This is fine! Interactive programs are allowed on cbrglogin3; however:
  - It’s popular and gets a lot of use, so you may find it’s slow
  - It has strict memory limits, which you may eventually go over
- If that happens, consider whether Slurm might indeed help

Part 2

What is Slurm and how does it work?

Slurm in a Nutshell: It runs jobs in a queue

- A **job** is a piece of work you ask Slurm to run on your behalf
  - Process this dataset with STAR
  - Run this pipeline on this input
- Everyone's **jobs** go in a **queue** and wait to run
- Slurm decides which **job** goes where in the **queue** based on the available resources and a fair-share approach
- When a **job** gets to the top of the **queue** it runs on any one of many different servers, called **nodes**
  - You get the resources on the **node** you asked for (no more, no less)
  - It runs exactly what you told it to run
- Having lots of (mostly) identical **nodes** means that lots of **jobs** can all run at the same time
- When a **job** finishes, a new **job** is picked from the **queue** to run

This is the key concept for Slurm - the rest is all how to run your own jobs

Exercises

- As we go forwards, we’ll give you exercises to try
- We recommend you follow along with these to aid your understanding
- Log into the cluster and try the commands
- Compare what you see with the examples

Exercise 1 – Checking the queue with **`squeue`**

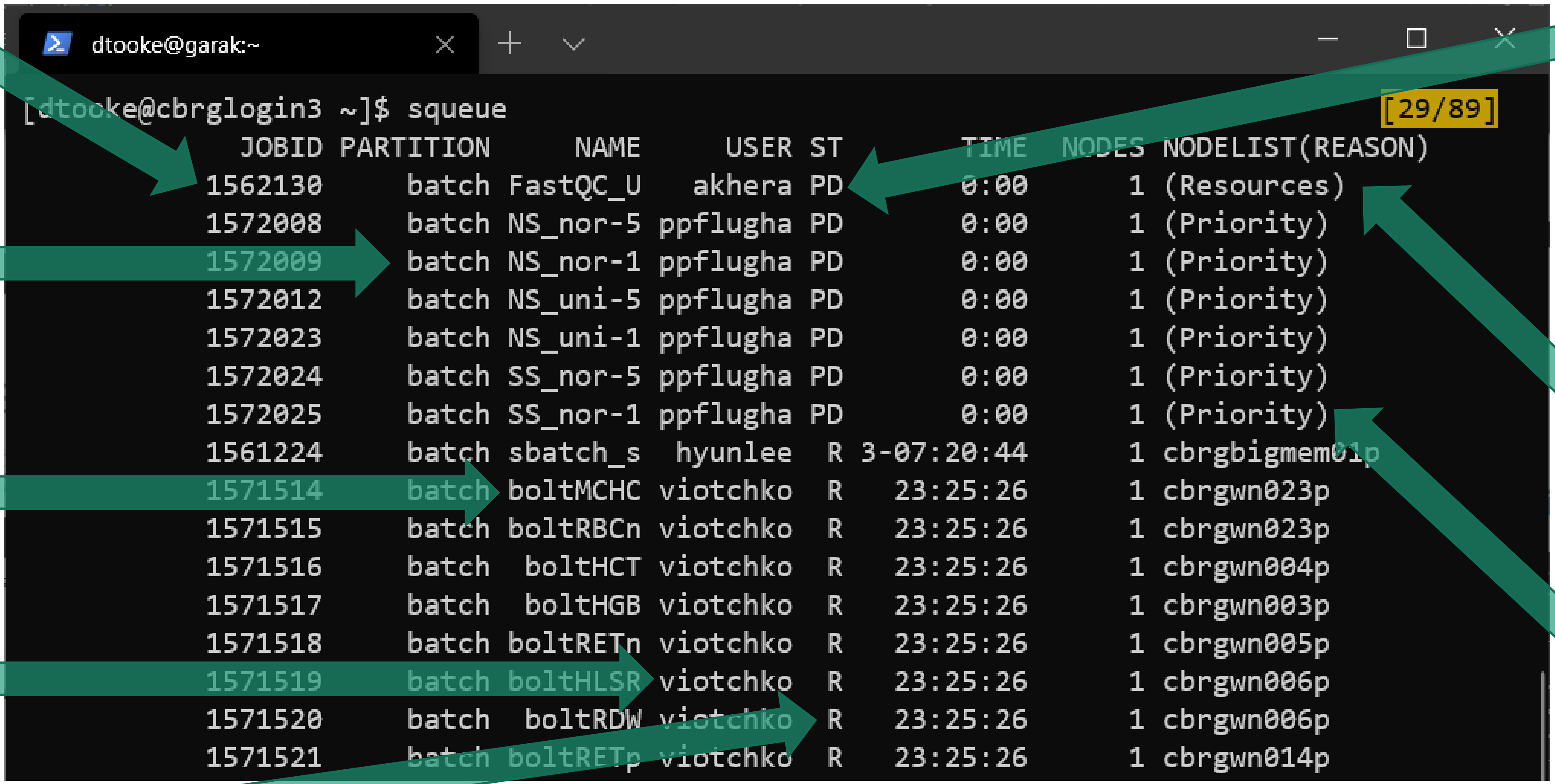
Every job has a unique **JOBID**

PARTITION is an advanced feature - for most jobs it's just **batch**

You can give jobs a **NAME** too

Jobs are owned by **USER**s

These jobs are **R**unning



```
[dtooke@cbrglogin3 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1562130     batch FastQC_U   akhera PD       0:00      1 (Resources)
           1572008     batch NS_nor-5 ppflugha PD       0:00      1 (Priority)
           1572009     batch NS_nor-1 ppflugha PD       0:00      1 (Priority)
           1572012     batch NS_uni-5 ppflugha PD       0:00      1 (Priority)
           1572023     batch NS_uni-1 ppflugha PD       0:00      1 (Priority)
           1572024     batch SS_nor-5 ppflugha PD       0:00      1 (Priority)
           1572025     batch SS_nor-1 ppflugha PD       0:00      1 (Priority)
           1561224     batch sbatch_s  hyunlee  R 3-07:20:44      1 cbrgbigmem01p
           1571514     batch boltMCHC viotchko  R   23:25:26      1 cbrgwn023p
           1571515     batch boltRBCn viotchko  R   23:25:26      1 cbrgwn023p
           1571516     batch  boltHCT viotchko  R   23:25:26      1 cbrgwn004p
           1571517     batch  boltHGB viotchko  R   23:25:26      1 cbrgwn003p
           1571518     batch boltRETn viotchko  R   23:25:26      1 cbrgwn005p
           1571519     batch boltHLSR viotchko  R   23:25:26      1 cbrgwn006p
           1571520     batch  boltRDW viotchko  R   23:25:26      1 cbrgwn006p
           1571521     batch boltRETp viotchko  R   23:25:26      1 cbrgwn014p
```

These jobs are waiting to run (**ST**ate **P**en**D**ing)

The job at the top of the queue can't start until the necessary **Resources** become free on a node

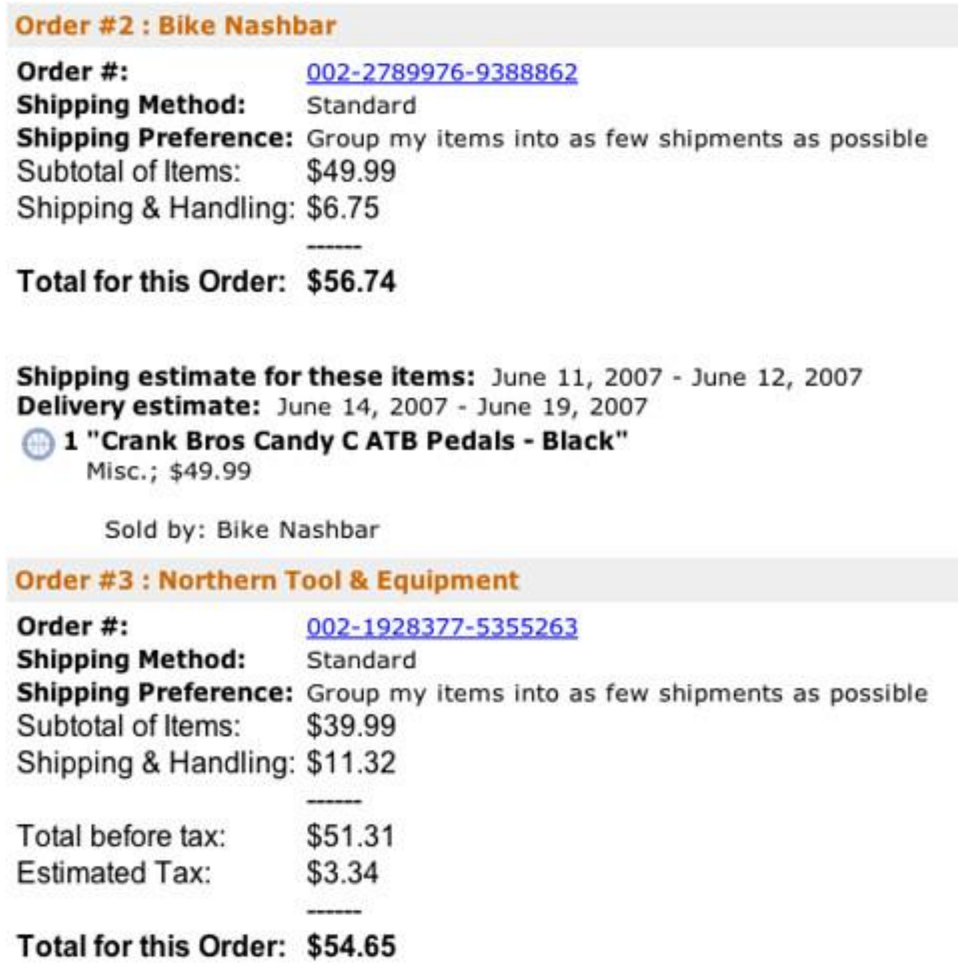
These jobs won't run until they get to the top of the queue and have **Priority**

How long a job has been running is **TIME**d

Jobs run on one (**1**) specific **NODE**

Describing your jobs to Slurm

- Each **job** that you want to run is described in a **job script**
  - Explains what resources (time, CPU, memory, etc.) you think you will need reserving for your use
  - Says what you want to do when it gets to the top of the queue and runs
- The **job script** is a like an online order with two parts:
  - **I want**: to run STAR on this data
  - **Deliver it to**: 8 CPUs and 32GB memory for 18 hours



Order #2 : Bike Nashbar

Order #:	002-2789976-9388862
Shipping Method:	Standard
Shipping Preference:	Group my items into as few shipments as possible
Subtotal of Items:	$49.99
Shipping & Handling:	$6.75

Total for this Order:	**$56.74**

Shipping estimate for these items: June 11, 2007 - June 12, 2007
Delivery estimate: June 14, 2007 - June 19, 2007
1 "Crank Bros Candy C ATB Pedals - Black"
Misc.; $49.99

Sold by: Bike Nashbar

Order #3 : Northern Tool & Equipment

Order #:	002-1928377-5355263
Shipping Method:	Standard
Shipping Preference:	Group my items into as few shipments as possible
Subtotal of Items:	$39.99
Shipping & Handling:	$11.32

Total before tax:	$51.31
Estimated Tax:	$3.34

Total for this Order:	**$54.65**

An example job script

```
#!/bin/sh

#SBATCH --time=00:30:00
#SBATCH --ntasks=4
#SBATCH --mem=10G

module load rna-star

STAR --genomeDir
/databank/igenomes/Drosophila_melanogaster/
UCSC/dm3/Sequence/STAR/ --outSAMtype BAM
SortedByCoordinate --readFilesIn
C1_R1_1.fq.gz C1_R1_2.fq.gz --
readFilesCommand zcat --outSAMattributes
All --outFileNamePrefix C1_R1_ --
limitBAMsortRAM 7000000000 --runThreadN 4
```

- The time we want, here 30 minutes
- The number of CPU cores / threads that we want
- The memory we want, here 10 GB

- The commands we want to run, and;
- The options we want to give to the program

These options need to match `--ntasks` and `--mem` or it won't work properly

!

Exercise 2 – Running a job

- `cd` into your personal `/t1-data` directory
- `cp /t1-data/user/course/slurm/star.sh .` (don't miss the last " . ")
- `cat star.sh`
- `sbatch star.sh`
- `watch squeue --me`
- Wait for it to end (`CTRL+C` to exit `watch`)
- `ls -ltrh`
- `cat` the output and take a look (but don't worry too much, we're going to come back to this later)

Part 3

How do I know what resources I’ll need?

Estimating your resources

- This is the most important (and the most tricky) part, because of two critical factors:
  - **The more resources you ask for, the lower in the queue you'll end up**
  - **If you ask for too little memory or time, your job will die before it finishes**
- Thus, it's really important to get it right
- Let's look at the 3 critical resources: time, CPU cores, and memory

How much time do I need?

- This varies depending on the program and the data
- If you don’t say how much you want, you get the default (currently 7 days)
- **If you really don’t know, just leave it at the default**
- In a moment, we’ll look at a way to understand this better

Asking for less time will mean your job goes higher in the queue, but it will die if it overruns

How many CPU cores do I need?

- The answer is typically one of two options:
  - The number the documentation recommends; or
  - One
- There's a limit to the number of CPUs a program can use – adding more gives **no benefit**
- It's really important that you:
  - Set the CPUs in the Slurm options
  - Set the same number in the program options
- **If you really don't know, set it at 1**
- In a moment, we'll look at a way to understand this better

Using more CPUs **doesn't** mean your job will go faster

How much memory do I need?

- This is the most important to get right
- Again, it depends on the program and the data
- For this, you really will need to look at it in more detail using the technique we'll explore next
- **If you really don't know, set it at something moderate, like 20G**

Asking for less memory will mean your job goes higher in the queue, but it will die if it uses a lot more

Job profiling

- When you were looking at your STAR output you may have noticed extra information and some graphs at the end – this is job profiling information
- Job profiling is a where a small helper program tracks how much time, CPU and memory your job uses and provides statistics
- Using these, you can submit a single job with your best guesses, look at the output, and then have a reasonable idea of what to use for subsequent jobs of a similar type
- Let’s have a closer look using a run of our STAR example

Time profiling

```
+--------------------------------------------------------------+
|   This is the CCB job profiler for your job. For help with   |
|   the results, please contact genmail@molbiol.ox.ac.uk       |
+--------------------------------------------------------------+
REQUESTED    : 00:30:00
JOB RUN TIME : Days 0, Hours 0, Minutes 2, Seconds 11
UTILISATION  : 0.01273148%
+--------------------------------------------------------------+
|                  NOTICE - TIME OVERREQUEST                   |
|   Elapsed job time is less than 10% the time that the job  |
|   requested. Over requesting time will mean that your        |
|   jobs take longer to get to the top of the queue.           |
+--------------------------------------------------------------+
```

- I asked for 30 minutes in my job script
- Of this, I only used 2 minutes and 11 seconds and so got an information notice
- This is useful, because knowing this I could manually set it to something shorter using `--time` and my job would go higher in the queue.

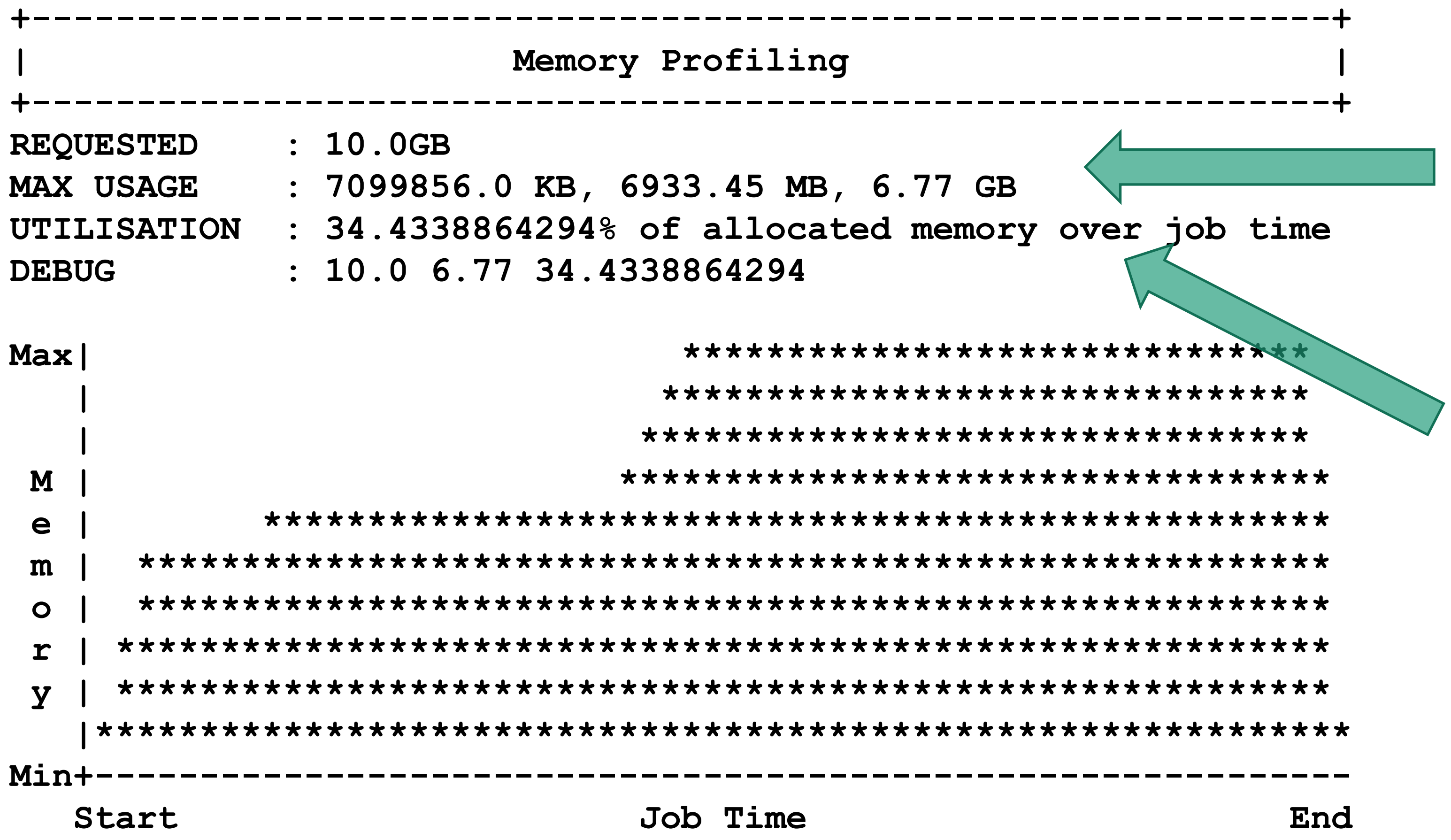
CPU profiling

```
+---------------------------------------------------------------+
|                         CPU Profiling                         |
+---------------------------------------------------------------+
REQUESTED     : 4 CPU cores
MAX USAGE     : 3.269 CPU cores
UTILISATION   : 62.3870689655% of allocated CPU over job time
DEBUG         : 4 3.269 62.3870689655

Max|           ***********
   |         ****************
   |       **********************
 C |      *****************************
 P |     **************************************
 U |    ******************************************************
   |    *******************************************************
 % |   *********************************************************
   |  **********************************************************
   |*************************************************************
Min+-------------------------------------------------------------
   Start                      Job Time                       End
```

- No notice here
- At the peak of my usage graph, I used just over 3 of my 4 CPU cores
- However, because I didn’t use them all the time, I only used 62% of the total CPU time I was given
- I also get a graph of rough CPU use over time, which is useful when you get more experienced

Memory profiling



```
+--------------------------------------------------------------+
|                       Memory Profiling                       |
+--------------------------------------------------------------+
REQUESTED    : 10.0GB
MAX USAGE    : 7099856.0 KB, 6933.45 MB, 6.77 GB
UTILISATION  : 34.4338864294% of allocated memory over job time
DEBUG        : 10.0 6.77 34.4338864294

Max|                            ******************************
   |                           *******************************
   |                          ********************************
 M |                         **********************************
 e |        ***************************************************
 m |  *********************************************************
 o |  *********************************************************
 r | **********************************************************
 y | **********************************************************
   |************************************************************
Min+------------------------------------------------------------
   Start                      Job Time                       End
```

- I told STAR to use 7GB on the command line and it used roughly that
- Because of the way STAR works, it used more memory in the second half of the job, so overall I only used 34% of all the memory I could have

Exercise 3 – Changing resource parameters

- Change the job to use double the CPUs (remember to set this in both places) and try it again. What happens to the output? How much faster does it run?
- Try deliberately changing the number of CPUs requested from Slurm to half the number that is set in the STAR options, so that it doesn’t get enough. What happens?
- Run the job but only request 2G memory from Slurm (i.e. not enough). What happens?
- Run the job but only request 1 minute of time from Slurm. What happens?

Part 4

Top tips

You can’t go wrong with the default 7 days job time

- The worst thing that can possibly happen is that your job needs longer than that and dies
- If it looks like that’s going to happen just contact us and we can manually extend it for you
- Asking for less time will tend to put you higher in the queue when the cluster is busy, but that’s all.

Advanced tip

Getting your time close to what you use does make it more likely that your job will run when the cluster is busy; try to get to within an order of magnitude (a week, a day, an hour, etc.)

It’s often quicker to use fewer CPUs (throughput is king)

```
REQUESTED : 8 CPU cores
MAX USAGE : 7.893 CPU cores
UTILISATION : 96.9486702128% of allocated CPU over job time
DEBUG : 8 7.893 96.9486702128

Max|************************************************************
   |************************************************************
   |************************************************************
 C |************************************************************
 P |************************************************************
 U |************************************************************
   |************************************************************
 % |************************************************************
   |************************************************************
   |************************************************************
Min+------------------------------------------------------------
   Start                      Job Time                      End
```

- This is a bit counter intuitive
- Essentially, it’s often only worth adding more CPUs to your job if the profiling output looks like this
- This is a well running job - it’s using all the CPU cores it’s asked for and it’s using them all the time

It’s often quicker to use fewer CPUs (throughput is king)

```
REQUESTED : 8 CPU cores
MAX USAGE : 7.666 CPU cores
UTILISATION : 46.9523343251% of allocated CPU over job time
DEBUG : 8 7.666 46.9523343251

Max|*********************
   |*********************
   |*********************
 C |**********************               *
 P |**********************               *
 U |**********************   **          *
   |**********************   ****        *
 % |**********************   *******    **    **       **
   |**********************  **************    ******************
   |************************************************************
Min+------------------------------------------------------------
   Start                      Job Time                       End
```

- This job looks OK at first glance, but…
- The number under the max usage is the utilisation - the percentage of the total available CPU time which was actually used
- In this case it’s less than 50%
- The job only used all those CPU cores for the first 30-40% of the time
- For the rest, it barely used 2

It’s often quicker to use fewer CPUs (throughput is king)

- Let’s say that the first part of the job was 1/3rd the total running time
- If we only ask for 2 CPUs then that part’s probably going to take 4 times as long
- That would add an extra 3 x 1/3rd time, for a total of double the runtime
- However, we’re asking for 4 times fewer CPUs
- Over multiple jobs we’ve doubled the throughput - 4 of the new jobs can run for every 2 of the old ones

Advanced tip

If your CPU profile looks like this example, check whether you’re running more than one command per job; if you are, consider splitting each job into multiple jobs - one for each command being run – so that you can run each job with just the right amount of resources

To estimate your memory, profile a single job

- Memory is the hardest resource to get right
- There are three key ways to estimate it; from most- to least-preferred they are:
  - Submit one job with a moderate memory requirement - if it doesn't have enough it'll run out and die, and you can try again with more
  - Submit one job with a large memory requirement but a short time limit, say an hour
  - Submit one job with a large memory requirement and let it finish
- The reason that the last is the least preferred is that we could end up wasting a lot of what we've asked for
- Please **do not** submit a significant number of jobs which ask for a lot of memory when you really don't know how much they're going to use

Advanced tip

Memory is relatively scarce compared to time and CPU; if you can run the same job for longer with less memory, or more jobs with each using less memory by chopping your data into smaller chunks, this may be worth doing

Add 25% - 50% of memory headroom (maximum)

- Once you have a figure for how much memory to use, add a small amount of headroom
- 25% - 50% is likely to be sufficient
- Don’t overdo it!
  - If it isn’t used, your throughput will drop
  - You could waste resources
  - You can always increase it again if it turns out to be too little
  - You can always re-run jobs which ran out without having to re-run jobs which didn’t
- Conversely, if you find that the first one was using more than typical, reduce it again

Advanced tip

The memory graph can identify jobs which use the majority of their memory for only a small percentage of the time; if you can split a single job into one memory intensive job and a second, lower memory job, you can increase your throughput

Exercise 4 – Optimising your jobs

- Take a look at the profiling data from your own recent jobs
- How much of the time you asked for are you using? Could you safely ask for less (think “week, day, hour”)
- How much of your available CPU are your jobs using? Could you realistically ask for less? Could you split a single job into a separate CPU intensive job and a second job with less CPUs?
- How much of your requested memory are your jobs using? How much does it vary between data sets? Could you improve this? Could you have more, smaller jobs?

Part 5

Summary

Summary

- Using Slurm can allow you to do orders-of-magnitude more science in the same amount of time
- Getting your job resources correct is not trivial, and takes practice
- Use the profiling data – it's your number one resource for understanding your own workflow
- Getting your science done speedily is more about throughput than running any one job as fast as possible
- If you have any questions or would like extra help please contact us

genmail@molbiol.ox.ac.uk