SLURM TOP TIPS

TIP 1 - 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, but if it looks like that’s going to happen just contact us and we can manually extend it for you. Asking for less will tend to put you higher in the queue when the cluster is busy, but that’s all.

Due to the low impact of asking for lots of time, we’ve tweaked the profiling to only let you know if you used less than 10% of what you asked for.

Advanced tip: Getting your time estimate in the ballpark of what you’re going to use really does make it more likely that your job will be chosen to run when the cluster is busy. Try to get to within an order of magnitude of the expected runtime (a week, a day, an hour, etc.)

TIP 2 - IT’S OFTEN QUICKER TO USE FEWER CPUS (THROUGHPUT IS KING)

This is a bit counter intuitive but stick with it.

Essentially, it’s often only worth adding more CPUs to your job if the profiling output looks like this:

```
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 well running job. It’s using all the CPU cores it’s asked for and it’s using them all the time.

If, however, your CPU graph has significant gaps, you’re typically better off asking for fewer CPUs and running more jobs at the same time. Take the job on the next page for example:

```
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
```

Looks OK at first glance, yes? Asked for 8 cores, maxed out at just under that. Great!

Well, not really. The number under the max usage is the utilisation. That's a single number that shows what percentage of the total CPU time which was available actually got used. In this case it's less than 50% and you can see why - the main issue is that the job only used all those CPU cores for the first 30-40% of the time. For the rest, it barely used 2.

Let's do some maths. 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, so 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, so we can potentially run 4 jobs at the same time for every one of these. Which means that if we need to run a few, we've just doubled the throughput because in the same amount of time we can run 4 of the new small jobs for every 2 of the old large ones we had.

The other reason to do this is that jobs which have an erratic CPU profile tend to take away resources from other people whose jobs could have used them. Doing things this way is more considerate.

Advanced tip: If your CPU profile looks like the graph on this page, 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. That way, you can run each job with just the right amount of resources and add extra CPUs only where it adds value. This can further multiply your throughput.

Tip 3 - To estimate your memory, profile a single job

Of the three key resources your job needs - time, CPUs and memory - memory is certainly the hardest to get right. There are three reasons for this:

- Memory is (relatively) scarce compared to time and CPU, so it's unfriendly to use too much; but…
- If you ask for too little your job will die, and you'll have to start all over again!
- Memory requirements often vary depending on the data you input to your job

The easiest way to get it right is to profile a single job. This way, you can get real-world data to inform subsequent runs. There are three key ways to do this. 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 simply because we're doing this with no idea of what's going to happen and 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: As we said above, 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. Conversely, avoid asking for a whole node's worth of memory (250G) - or anything more than that which would require using our only large memory node - if you possibly can.

Tip 4 - Add 25% - 50% of memory headroom (maximum)

Given how tricky it is to get the memory requirements right, here's a second memory related tip.

Once you've profiled a job and got a figure for how much memory it used, add a small amount of headroom for your next jobs. From what we've seen so far, 25% - 50% is likely to be sufficient. Remember! If you add a lot of memory that you don't use, your throughput will drop, and you could waste resources that someone else could have used. You can always increase it again if it turns out to be too little, and you can always re-run jobs which ran out without having to re-run jobs which didn't.

Conversely, if after running more jobs 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. You'll also have a better chance of getting jobs to run when the cluster is busy.