Running parallel R jobs on Taito

Seija Sirkiä

19 lokakuuta 2016

Taito supercluster

What is a supercluster?

`https://research.csc.fi/taito-supercluster`

- Many computers pretending to be just one computer
- What, in fact, is a computer?
  - `https://en.wikipedia.org/wiki/Computer`
  - `https://en.wikipedia.org/wiki/Central_processing_unit`
  - The processor on your laptop probably has more than one core

It's just many cores, ok?

CPU, processor, node, core... the terminology is a mess but Taito has *many*, which ever you're counting: 985 nodes each with 16 or 24 cores, totalling at 17 704 cores.
(But Taito is not a supercomputer. Sisu is another kind of beast.)

Running R on Taito

The disappointment

- In many ways Taito is just like any other Linux computer. You can run your familiar R code in the familiar way. No changes needed.
  - Ok ok, batch jobs might not be familiar (more on them later) but the code still could be exactly the same
  - On the other hand, on Taito-shell things can look very familiar, even more so with NoMachine client and RStudio
- But without any changes your **R code will not automatically run faster on Taito** than on your laptop...
  - Although it might run faster (*)
  - It might even run slower

The reason

. . . because without any changes your R code will run on **one core** (*)

- The many cores won't do anything just by existing unless you explicitely put them to work! (*)
- Having "more memory" also won't help just by being there
- But simply running R on *somebody else's computer* might be enough to make you happy. Besides, you can run many R jobs on Taito at the same time! Take note: array jobs.

((*) Ask me later about Intel MKL)

Running R interactively on Taito-shell

Read the friendly manual

`http://classicprogrammerpaintings.com/image/146552180542`
Short version:

1. Connect to Taito-shell (via SSH or NoMachine)
2. Load the R module
3. Start R or RStudio
4. Do what you usually do

More details in the CSC R documentation at `https://research.csc.fi/-/r` and in the Taito-shell guide at `https://research.csc.fi/taito-shell-user-guide`

Running R as a batch job on Taito

Seriously, read the manual

This is explained in Taito user guide
`https://research.csc.fi/taito-batch-jobs` and again further at `https://research.csc.fi/-/r`
But here's the short version:

1. In addition to your R code, you also write a batch job file
2. You send the batch job file to SLURM
3. You wait. SLURM will decide when it's your turn and runs your R code for you then
4. When your job has finished, you get to look at the result files you made it create

Running parallel R batch jobs

The batch job file is where things start to look like parallel computing. It is where you tell SLURM that *you want to use more than one core*.
And then you *change something in your R code* so that it actually gives something to do to those extra cores. If you don't, you are not getting any benefits and are also wasting resources.
You might also want to look at the High Performance Computing taskview on CRAN.

Rmpi, snow and foreach

Rmpi

MPI or Message Passing Interface is something that actually allows all those multiple cores to work together. Rmpi is an R package that works as an interface between MPI and R. Both snow and foreach (through doMPI) depend on it.
But if you don't already have some code or a very relevant example that explicitly uses Rmpi functions then **do not touch Rmpi** yourself.
(Even if you actually know MPI, I would still advice against it. Consider the package pbdMPI instead.)

snow

(snow = simple network of workstations)
Snow provides parallel versions of apply-like functions. This means, roughly, that you can run a piece of code e.g. on different parts of your data, and they will be executed in parallel on different cores.
Also, there are packages that will take advantage of multiple cores via a snow cluster handle.
For both cases, if you find an example using a snow-cluster, it should be applicable on Taito simply by using RMPISNOW in your batch script, and getMPIcluster() in your R code.

foreach

In a similar fashion as snow, foreach provides a way to run for-loops in parallel. It is relatively easy to understand and comes with very thorough tutorials.
Any example code using foreach and its %dopar% syntax is applicable on Taito, as long as you pair it with the package doMPI, as explained on CSC's R documentation page.

Doing it well

Assumption

You have a lot of work to do. Assigning one worker to do it takes a long time.
If you can divide the workload in to 5 chunks, and assign 5 workers one to each chunk, the work will get done in one 1/5 of the time, right?
Not necessarily.

Reality

Dividing the work introduces overhead: now there are extra things that need to be done, in addition to the actual work, that didn't need to be done at all in the one worker case
In practice, only part of the work can be divided up anyway.
And this doesn't even begin to explain what should be divided and in to how many parts.

So, what should you do?

Get really intimate with your code and method. Which part is the heavy one? Learn how to use System.time. Can the method be applied on pieces of data independently? How does the needed computing time behave as the amount of data increases? Has someone already created a function to apply the method in parallel? Find out what others have done in similar cases. Ask for help. Try with a small example. Just don't blindly use up a lot of resources and hope for the best.
And if it feels that all this takes too much effort... then just do it serially.