Tips to use MATLAB parallel computing toolbox

Kai Du

21/12/2020

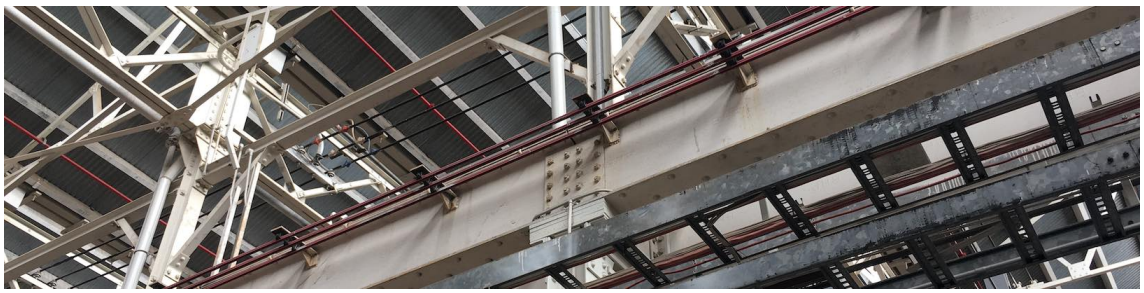

Figure 1: Structure

> Parallel computing toolbox enables researchers to harness a multicore computer, GPU, cluster and even cloud to solve computationally and data-intensive problems.

However, there are some options to use it (e.g. *parfeval*, *parfor* and even *spmd*), such as

n = 100; parfor (or parfeval) i = 1:n output(i)=Function(input(i)) end

As MATLAB suggested, *parfor* and *parfeval* usually perform better than *spmd* for a set of tasks under these conditions:

- The computational time taken per task is not deterministic.
- The computational time taken per task is not uniform.
- The data returned from each task is small.

Use *parfeval* when:

- You want to run computations in the background.
- Each task is dependent on other tasks.

In this post, I share my experience to use these three commands. The general rule is the tradeoff between the time to deploy the codes in each worker and the time to run the codes in each worker. It might be more reasonable to fully use all workers *first* and *then* keep them running as long as possible. Thus, I prefer to use *parfor*, rather than *parfeval* if the number of tasks is less than the number of workers. When the number of tasks is more than (or equals to) the number of workers, *parfeval* works faster than *parfor*. Second, *parfor* only works when the tasks are independent but *parfeval* works for the 'dependent' tasks.

Finally, it does not mean we have to use both of *parfor* and *parfeval* when the time to finish each task is very small. In fact, it might not a bad idea to use *for* for some very small jobs and then use *parfor* or *parfeval* for its 'parent' loops.