spmd stuff

AJR from miscellaneous searches

September 18, 2020

`isempty(gcp('nocreate'))` may be equivalent to (spmd or ~parpool):

- `==true`
  - if no parallel pool, or
  - if there is a parallel pool and code is inside a `spmd`
- `==false`
  - if there is a parallel pool, and code is not inside a `spmd`

Interestingly, `numlabs` function returns the total number of workers operating when inside `spmd`. Outside `spmd` or inside `parfor`-loop it always returns one.

Global and Persistent Variables "The body of an spmd statement cannot contain global or persistent variable declarations. The reason is that these variables are not synchronized between workers. You can use global or persistent variables within functions, but their value is only visible to the worker that creates them. Instead of global variables, it is a better practice to use function arguments to share values."

Walter Roberson on 23 Jan 2018 "global variables are not copied to workers. The workers do know the variable as global—but each worker has its own version (because they are different processes); and the values will not be copied back. You can deliberately send copies of the globals you know about

to workers using `parfevalOnAll` to run a function that does the appropriate 'global' call and assigns the given value."

Harsha Medikonda on 17 Aug 2015 "You can declare persistent variables in a function that is called by the spmd block. Please make sure that you are attaching the file `'fun1.m'` using `addAttachedFiles` so that it is accessible by all the workers. For Example:

```
poolobj = gcp;
addAttachedFiles(poolobj,{'fun1.m'})
spmd
  M=fun1(labindex);
end
```