MATLAB Parallel Computing Toolbox on Casper and Cheyenne

The MATLAB Parallel Computing Toolbox (PCT) allows you to run MATLAB code in parallel across multiple workers, which are analogous to MPI tasks or OpenMP threads. Additionally, NCAR has a license for the MATLAB Parallel Server (MPS) – formerly the MATLAB Distributed Computing Server – which allows you to run a MATLAB script using workers from multiple compute nodes via a batch job you submit with PBS.

Using MPS is critical for workflows that result in MATLAB processing running on multiple nodes. Here's why:

Any MATLAB component – MATLAB itself or one of the many toolboxes – follows the same rules regarding license use. When you load MATLAB or activate a toolbox, you check out a single license. If you launch the same product *on the same node*, no additional licenses are used. However, if you run the same product *on a different node*, you consume another license. These rules mean that running multiple batch jobs, with each executing a MATLAB script, consumes many licenses if they are scheduled on different compute nodes.

The MPS alleviates this issue. If you start a PBS job via the MPS, you consume only a single MATLAB license (and single licenses of any other toolboxes loaded including the PCT itself) no matter how many nodes you request. Instead, MATLAB will use MPS worker licenses for each worker requested. **Therefore, if you have a workflow that will result in MATLAB processes running on multiple nodes, always look to use the MATLAB Parallel Server.**

In summary, here are basic guidelines for when you should use each product:

- **MATLAB only** - when you need only a single worker/task (e.g., running a script with only serial computation)
- **PCT (local)** - when you want to run a script or function using a parallel pool of workers on a single node
- **MPS (distributed PCT)** - when you need to run a script/function across multiple nodes of workers OR you want to run multiple scripts on multiple nodes (similar to a command-file PBS job)

Page contents

- Running a simple parallel code on one node using the toolbox
- MPS cluster profiles
- Using the MATLAB parallel server (MPS) to span multiple nodes
- Sample PCT scripts

Running a simple parallel code on one node using the toolbox

Example 1 (func_local): In this example, a simple code uses multiple workers on a single node to compute a sum in parallel. Here is the sample code:

```
parallel_sum.m:
function s = parallel_sum(N)
  s = 0;
  parfor i = 1:N
    s = s + i;
  end

  fprintf('Sum of numbers from 1 to %d is %d.n', N, s);
end
```

This function is executed by MATLAB code that reads in a few parameters from your user environment. For single-node parallel jobs, this example uses the “local” cluster profile that is available by default when using the toolbox.

Do not run the local toolbox profile on login nodes; excessive CPU and/or memory use will result in your script being terminated. Instead, use a PBS batch job as in the following example to run a single-node cluster on a compute node. This PBS job also specifies the number of workers, which corresponds to the number of CPUs requested in the job script.

```
submit_local.pbs:
#!/bin/bash
#PBS -N matlab_pct
#PBS -A <PROJECT>
#PBS -l walltime=05:00
#PBS -q casper
#PBS -j oe
#PBS -o local.log
#PBS -l select=1:ncpus=4:mpiprocs=4:mem=10GB

# The following script is not distributed; it uses threads
# and so this PBS job should only ever request a single node

module load matlab

# Derive the number of workers to use in the toolbox run script
export NUMWORKERS=$(wc -l $PBS_NODEFILE | cut -d' ' -f1)

SECONDS=0
matlab -nodesktop -nosplash << EOF
% Start local cluster and submit job with custom number of workers
c = parcluster('local')
j = c.batch(@parallel_sum, 1, {100}, 'pool', $((NUMWORKERS - 1)));

% Wait for the job to finish, then get output
wait(j);
diary(j);
exit;
EOF
echo "Time elapsed = $SECONDS s"
```

MPS cluster profiles

When using the PCT, you are expected to create and use *cluster profiles* that manage either node-local tasks or batch-scheduler tasks. While there is a preconfigured profile for single-node use (local), you will need to do some setup before you can use the MPS. CISL provides a distributed cluster profile for PBS on both Casper and Cheyenne for all versions of MATLAB starting with R2020a.

You can import an existing cluster profile using the wizard in the graphical interface, or you can do it programmatically as follows. If you use our sample distributed script provided in the following section, we include the MPS cluster profile setup for you, so you can skip the commands in this section.

At the MATLAB command line, enter the following line to import the MPS profile:

```
ncar_mps = parallel.importProfile('/glade/u/apps/opt/matlab/parallel/ncar_mps.mlsettings');
```

You need to import the profile only once; MATLAB will remember it in future sessions. If you anticipate using the parallel server profile frequently, you may want to make it your default parallel profile as shown here:

```
parallel.defaultClusterProfile(ncar-mps);
```

Using the MATLAB parallel server (MPS) to span multiple nodes

The configuration above will limit your job to the number of CPUs on a single node; on Casper and Cheyenne this means 36 workers, or 72 if you use hyperthreads. However, you can use the parallel server to span multiple nodes. When using MPS, MATLAB itself will submit a job to the batch scheduler and use an internal MPI library to enable communication between remote workers.

Example 2 (func_mps): Here again, use a MATLAB script to set up your parallel cluster as in this example, which embeds MATLAB code into a driver script:

```
submit_server.sh:
#!/bin/bash

# This script doesn't need to run on a batch node... we can simply submit
# the parallel job by running this script on the login node

module rm ncarenv
module load matlab

mkdir -p output

# Job parameters
MPSNODES=2
MPSTASKS=4
MPSACCOUNT=<PROJECT>
MPSQUEUE=casper@casper-pbs
MPSWALLTIME=300
SECONDS=0

matlab -nodesktop -nosplash << EOF
% Add cluster profile if not already present
if ~any(strcmp(parallel.clusterProfiles, 'ncar_mps'))
    ncar_mps = parallel.importProfile('/glade/u/apps/opt/matlab/parallel/ncar_mps.mlsettings');
end

% Start PBS cluster and submit job with custom number of workers
c = parcluster('ncar_mps');

% Matlab workers will equal nodes * tasks-per-node - 1
jNodes = '$MPSNODES';
jTasks = '$MPSTASKS';
jWorkers = str2num(jNodes) * str2num(jTasks) - 1;

c.ClusterMatlabRoot = getenv('NCAR_ROOT_MATLAB');
c.ResourceTemplate = append('-l select=', jNodes, ':ncpus=', jTasks, ':mpiprocs=', jTasks);
c.SubmitArguments = append('-A $MPSACCOUNT -q $MPSQUEUE -l walltime=$MPSWALLTIME');
c.JobStorageLocation = append(getenv('PWD'), '/output');

% Output cluster settings
c

% Submit job to batch scheduler (PBS)
j = batch(c, @parallel_sum, 1, {100}, 'pool', jWorkers);

% Wait for job to finish and get output
wait(j);
diary(j);
exit;
EOF

echo "Time elapsed = $SECONDS s"
```

Sample PCT scripts

Including the scripts shown above, there are four sets of example scripts that you can use, modify, and extend to fit your purposes. All four examples can be copied from /glade/u/apps/opt/matlab/parallel/examples.

- **func_local** - Run a specified function using a pool of local (single-node) workers. This is Example 1 above.
- **func_mps** - Run a specified function using a pool of workers distributed across multiple nodes. This is Example 2 above.
- **multi_script_mps** - Run a specified collection of MATLAB functions in separate script files using a pool of workers. Configured for MPS use but can be modified to use the local profile.
- **spmd_mps** - Run a single MATLAB function in a specified script using many input parameters.

The last two examples are functionally similar to a command-file PBS job, but with the licensing benefits of using MPS.
