MATLAB Parallel Computing on Andromeda

Initial Setup

1.Load and open the MATLAB module

Note: Normal text denotes user input, inverted colors denote response from software.

```
[johnchris@l001 ~]$ rm -rf .matlab
[johnchris@l001 ~]$ module load matlab/2024a
[johnchris@l001 ~]$ matlab
```

```
MATLAB is selecting SOFTWARE rendering.

                        < M A T L A B (R) >
              Copyright 1984-2024 The MathWorks, Inc.
         R2024a Update 3 (24.1.0.2603908) 64-bit (glnxa64)
                           May 2, 2024
To get started, type doc.
For product information, visit www.mathworks.com.
>>
```

Note: there is a space between rm and -rf, as well as another one between -rf and .matlab

2. Now we are in a MATLAB session. Run the MATLAB command:

Note: Type each individual line, do not not copy paste these commands as a block of code or you may see errors.

```
>> configCluster
```

```
Complete.  Default cluster profile set to "Andromeda".
```

```
>> c = parcluster;
>> c.saveProfile;
```

3. Exit the MATLAB session:

```
>> exit
[johnchris@l001 ~]$
```

Configuring Jobs

1. Create a Slurm file (parallel_matlab.sl) that calls your MATLAB code file which you will construct next.

```
#!/bin/bash
#SBATCH --job-name=sample                #Job name
#SBATCH --time=24:00:00                  #Time limit hrs:min:sec, cannot exceed 120 hours
#SBATCH --ntasks=1                       #Number of tasks
#SBATCH --nodes=1                        #Number of nodes requested
#SBATCH --partition=shared               #Node choice
#SBATCH --cpus-per-task=1                #Number of CPU processors per task
#SBATCH --mem-per-cpu=4G                 #RAM/node, Max (180GB/48 cores, 250GB/64cores)
#SBATCH --mail-type=ALL                  #Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=username@bc.edu #BC email address
module load matlab/2024a
cd /mmfs1/data/johnchris/parallel_matlab   #Directory of your MATLAB code file
matlab -batch parallel_matlab >&parallel_matlab.out
```

Note: Underline denotes input individual users MUST change. Make sure that the time limit is sufficient for the MATLAB job to start and complete.

2. Add the parallel computing command in your MATLAB code file (parallel_matlab.m).

```
c = parcluster;
%Corresponding to the time limit, max is 5 days (120:00:00)
c.AdditionalProperties.WallTime='6:00:00';
%Corresponding to CPU memory, per core
c.AdditionalProperties.MemPerCPU='6G';
%Number of processors requested per node, same as the number in parpool()
c.AdditionalProperties.ProcsPerNode=10;
%Can be different from the node choice
c.AdditionalProperties.Partition='shared';
saveProfile(c)
%Total CPU cores requested for the program, cannot exceed 512 cores
```

```
parpool(10)

%End of the parallel computing command, program codes start here
tic
n = 20000;
A = 500;
a = zeros(1,n);
parfor i=1:n
  if mod(i, 100) == 1
i
  end
  a(i) = max(abs(eig(rand(A))));
end
toc
delete(gcp('nocreate'));
```

Note: Andromeda currently provides many partitions for CPU & GPU nodes. For example for CPU, Andromeda (A1): exclusive and shared, Andromeda2 (A2): long, medium, and short. Submitting a job this way requires you to edit c.AdditionalProperties in order to specify computational resources wanted.

Submitting Jobs

1. Upload the Slurm file (parallel_matlab.sl) and the MATLAB code file (parallel_matlab.m) to the directory specified in your Slurm file (/mmfs1/data/johnchris/parallel_matlab is given in the example, but you must change it to reflect your own file path) on the Andromeda cluster:

Note: It is advised to upload the Slurm file and the MATLAB code file in the same directory.

```
[johnchris@l001 ~/parallel_matlab]$ ls
```

```
parallel_matlab.m    parallel_matlab.sl
```

2. Once logged in to the Andromeda cluster, open the directory you created on the terminal:

```
[johnchris@l001 ~]$ cd parallel_matlab
[johnchris@l001 ~/parallel_matlab]$
```

3. Submit your job (the Slurm .sl file) onto Andromeda:

```
[johnchris@l001 ~/parallel_matlab]$ sbatch parallel_matlab.sl
```

```
Submitted batch job 2146192
```

4. You can check on the status of your jobs using the following code. "R" indicates that the job is running.

```
[johnchris@l001 ~/parallel_matlab]$ squeue | grep username
```

```
2146826    shared     Job1    johnchris    R       0:11      1 c005
2146825 exclusive   sample johnchris    R       0:23      1 c028
```

Note: Job 2146825 (sample) is what the Slurm file calls directly. Job 2146826 (Job1) is the parallel pool that the code in parallel_matlab.m calls within MATLAB.

5. Once your job is done, a file will appear (parallel_matlab.out), as named in the Slurm file, with the output of your MATLAB session.

```
[johnchris@l001 ~/parallel_matlab]$ ls
```

```
parallel_matlab.m    parallel_matlab.sl   parallel_matlab.out
```

Running Interactively

If you do not wish to use a Slurm file, you may instead write code directly in the terminal after entering an interactive session by using the command 'srun'. This will take you to an available compute node on Andromeda. Do not run MATLAB on the login node (node you are on when you enter the cluster

```
[johnchris@l001 ~]$ srun --job-name=sample --nodes=1 --ntasks=1 --time=24:00:00 --
mem=20G --pty bash -l
[johnchris@c116 ~]$ module load matlab/2024a
[johnchris@c116 ~]$ matlab
```

Note: You get assigned to a node on the cluster (eg. c116 in this case). Then, you get the following message and the MATLAB window opens interactively on the terminal.

```
MATLAB is selecting SOFTWARE rendering.

                                        < M A T L A B (R) >
                              Copyright 1984-2023 The MathWorks, Inc.
                              R2024a (24.1.0.2603908) 64-bit (glnxa64)
                                              May 2, 2024
To get started, type doc.
For product information, visit www.mathworks.com.
```

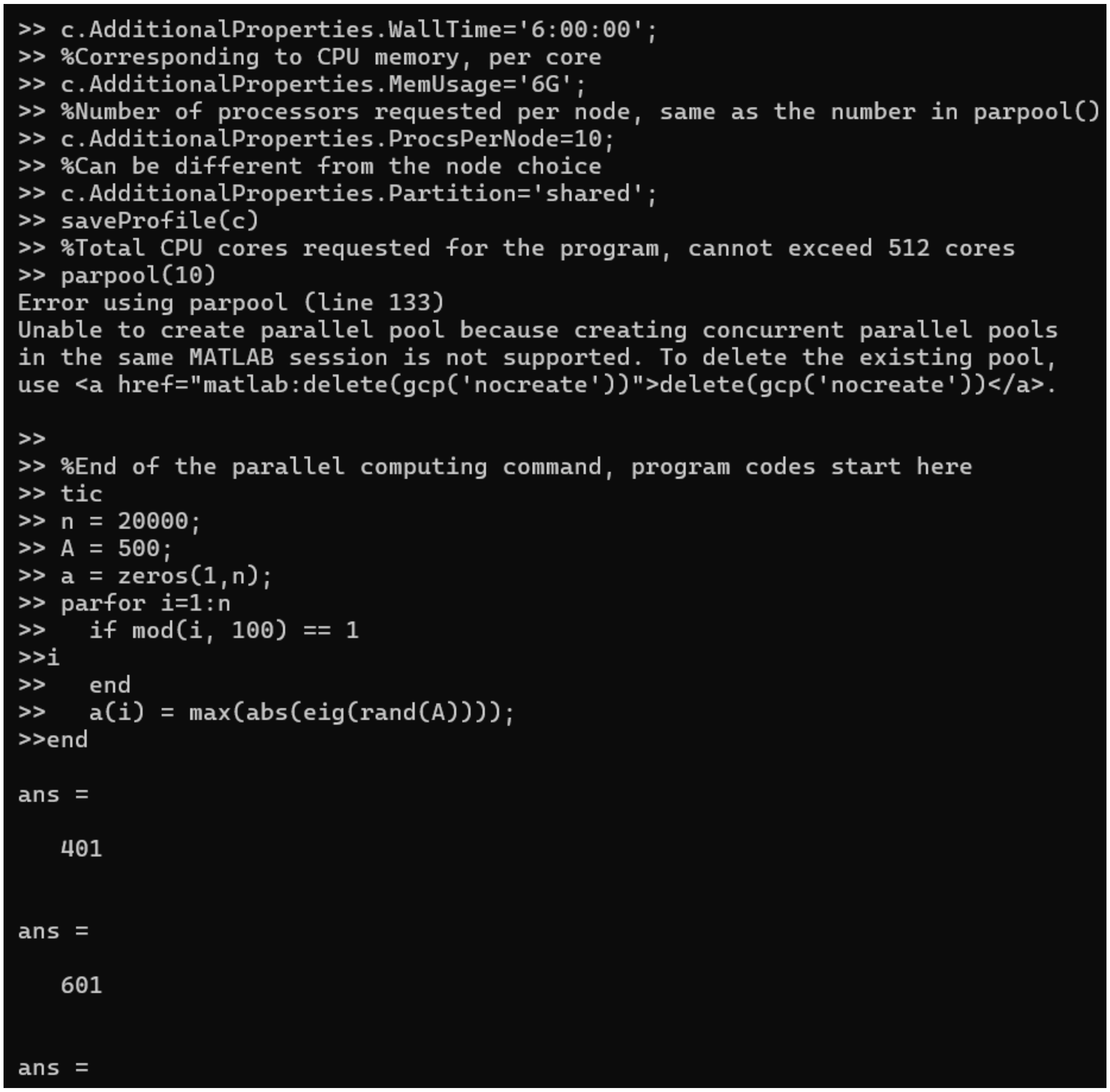
You can now enter the code in MATLAB interactively as below:

Note: Type each individual line, do not not copy paste these commands as a block of code or you may see errors.

```
>> c = parcluster;
>> c.AdditionalProperties.MemPerCPU='6G';
>> c.AdditionalProperties.WallTime = '24:00:00';
>> saveProfile(c)
>> parpool(10)
```

```
Starting parallel pool (parpool) using the 'Andromeda' profile ...

additionalSubmitArgs =

    '--ntasks=10 --cpus-per-task=1 --ntasks-per-node=10 --ntasks-per-core=1 --mem-per-
cpu=4gb -p shared -t 24:00:00'

Connected to parallel pool with 10 workers.

ans =

 ClusterPool with properties:

            Connected: true
           NumWorkers: 10
                 Busy: false
              Cluster: Andromeda (Generic Cluster)
        AttachedFiles: {}
    AutoAddClientPath: true
            FileStore: [1x1 parallel.FileStore]
           ValueStore: [1x1 parallel.ValueStore]
          IdleTimeout: 30 minutes (30 minutes remaining)
          SpmdEnabled: true
 EnvironmentVariables: {}
```

>> [insert your desired MATLAB code]



```
>> c.AdditionalProperties.WallTime='6:00:00';
>> %Corresponding to CPU memory, per core
>> c.AdditionalProperties.MemUsage='6G';
>> %Number of processors requested per node, same as the number in parpool()
>> c.AdditionalProperties.ProcsPerNode=10;
>> %Can be different from the node choice
>> c.AdditionalProperties.Partition='shared';
>> saveProfile(c)
>> %Total CPU cores requested for the program, cannot exceed 512 cores
>> parpool(10)
Error using parpool (line 133)
Unable to create parallel pool because creating concurrent parallel pools
in the same MATLAB session is not supported. To delete the existing pool,
use <a href="matlab:delete(gcp('nocreate'))">delete(gcp('nocreate'))</a>.

>>
>> %End of the parallel computing command, program codes start here
>> tic
>> n = 20000;
>> A = 500;
>> a = zeros(1,n);
>> parfor i=1:n
>>   if mod(i, 100) == 1
>>i
>>   end
>>   a(i) = max(abs(eig(rand(A))));
>>end

ans =

   401


ans =

   601


ans =
```

Screenshot: Example of MATLAB code and interactive output from the run.

```
>> delete(gcp('nocreate'));
Parallel pool using the 'Andromeda' profile is shutting down.
```

```
>> exit
[johnchris@c116 ~]$
```

Batch Job

You can also use the ‘batch’ command to submit asynchronous jobs to the cluster. The batch command will return a job object which is used to access the output of the submitted job.

1. Configure the MATLAB code (parabatch_matlab.m) with desired parallel computing process as a function:

```
function t = parabatch_matlab()
t0 = tic;
n = 20000;
A = 500;
a = zeros(1,n);
parfor i = 1:n
if mod(i, 100) == 1
i
end
a(i) = max(abs(eig(rand(A))));
end
t = toc(t0);
end
```

2. Open the specified directory on Andromeda:

```
[johnchris@l001 ~]$ cd parallel_matlab
[johnchris@l001 ~/parallel_matlab]$
```

3. Use ‘srun’ to go to a Compute node in Andromeda. Then start a MATLAB session, and open a parcluster:

```
[johnchris@l001 ~/parallel_matlab]$ srun --job-name=sample --nodes=1 --ntasks=1 --
time=24:00:00 --mem=20G --pty bash -l
```

```
[johnchris@c001 parallel_matlab]$ module load matlab/2024a
[johnchris@c001 parallel_matlab]$ matlab
```

Note: You get assigned to a node on the cluster (eg. c001 in this case). Then, you get the following message and the MATLAB window opens interactively on the terminal.

```
MATLAB is selecting SOFTWARE rendering.

                          < M A T L A B (R) >
                Copyright 1984-2024 The MathWorks, Inc.
           R2024a Update 3 (24.1.0.2603908) 64-bit (glnxa64)
                              May 2, 2024


To get started, type doc.
For product information, visit www.mathworks.com.
```

Note: Type each individual line, do not not copy paste these commands as a block of code or you may see errors.

```
>> c = parcluster;
>> c.AdditionalProperties.MemPerCPU='6G';
>> c.AdditionalProperties.WallTime = '24:00:00';
>> saveProfile(c)
>> parpool(10)
```

```
Starting parallel pool (parpool) using the 'Andromeda' profile ...

additionalSubmitArgs =

    '--ntasks=10 --cpus-per-task=1 --ntasks-per-node=10 --ntasks-per-core=1 --mem-per-
cpu=4gb -p shared -t 24:00:00'

Connected to parallel pool with 10 workers.

ans =

 ClusterPool with properties:

            Connected: true
           NumWorkers: 10
```

```
               Busy: false
            Cluster: Andromeda (Generic Cluster)
      AttachedFiles: {}
  AutoAddClientPath: true
          FileStore: [1x1 parallel.FileStore]
         ValueStore: [1x1 parallel.ValueStore]
        IdleTimeout: 30 minutes (30 minutes remaining)
        SpmdEnabled: true
EnvironmentVariables: {}
```

Note: Please ensure to specify an appropriate wall time for your job as the MATLAB parallel process may encounter errors if wall time is not specified in the above code.

4. Now call your function file using the batch command. The number after 'Pool' specifies the number of workers to be used (change this according to your needs). The batch command returns a job object, j.

```
>>  j = c.batch(@parabatch_matlab,1,{},'Pool',10,'Currentfolder','.','AutoAddClientPath',false)
```

```
additionalSubmitArgs =

    '--ntasks=11 --cpus-per-task=1 --ntasks-per-node=10 --ntasks-per-core=1 --mem-per-cpu=4gb -p shared -t 24:00:00'


j =

 Job

    Properties:

                ID: 2
              Type: pool
          Username: johnchris
             State: queued
    SubmitDateTime: 17-Oct-2024 18:30:12
     StartDateTime:
   RunningDuration: 0 days 0h 0m 0s
   NumWorkersRange: [11 11]
        NumThreads: 1
       SpmdEnabled: true
     AutoAttachFiles: true
 Auto Attached Files: /mmfs1/data/johnchris/parallel_matlab//parabatch_matlab.m
       AttachedFiles: {}
   AutoAddClientPath: false
     AdditionalPaths: {}
           FileStore: [1x1 parallel.FileStore]
          ValueStore: [1x1 parallel.ValueStore]
EnvironmentVariables: {}

   Associated Tasks:

      Number Pending: 11
      Number Running: 0
     Number Finished: 0
   Task ID of Errors: []
 Task ID of Warnings: []
  Task Scheduler IDs: 2691675
```

Note: Be aware that in addition to the number of workers that you ask for in your 'Pool' argument, you will also receive an additional "Orchestrator" worker that is required for job execution. Therefore, a request of 10 pool workers will result in 11 total tasks from the scheduler.

5. ‘j.State’ tells you whether the job is queued (waiting to start), running, or finished.

```
>> j.State
```

```
ans =
'running'
```

6. Use ‘j.fetchOutputs’ to retrieve function output arguments. Use ‘j.fetchOutputs{:}’ to display all contents in it. In this case, t = 198.7795 indicates that the program took 198.7795 seconds to run. If calling a batch with a script, use load instead. Data that has been written to files on the cluster needs to be retrieved directly from the file system, such as via FTP.

```
>> j.fetchOutputs
```

```
ans =

  1x1 cell array
```

```
{[226.6763]}
```

Note: outputs can only be fetched if the job is in state ‘finished’.

7. You can view a list of your past and current jobs, as well as their IDs, using the ‘c.Jobs’ Command.

```
>> c.Jobs
```

```
ans =
     ID         Type       State           FinishDateTime  Username  Tasks
  —-------------------------------------------------------------------------------------------
  1  1         pool        running                           johnchris      6
  2  3         pool        finished                         johnchris     11
  3  4         pool        running                           johnchris    10
  4  5         pool        running                           johnchris    11
```

8. If a serial job produces an error, call the ‘getDebugLog’ method to view the error log file. When submitting independent jobs, with multiple tasks, specify the task number. You can also analyze the job’s log file output when debugging.

```
>> c.getDebugLog(j.Tasks(3))
```

For Pool jobs, only specify the job object.

```
>> c.getDebugLog(j)
>> j.getTaskSchedulerIDs{:}
```

```
ans =
'2183065'
```

LOG FILE OUTPUT:

```
The scheduler has allocated the following nodes to this job:
c110
```

```
"/mmfs1/public/matlab/2024a/bin/mw_mpiexec" -bind-to core:1 -l -n 11
"/mmfs1/public/matlab/2024a/bin/worker" -parallel
[0] Parallel pool is shutting down.
Exiting with code: 0
```

SPMD Example

Single Program Multiple Data (SPMD) is an advanced construct that enables communication amongst different workers (processors) throughout the computation, and customization of tasks across workers. Under SMPD, each worker has a unique index, spmdIndex. In the example below, workers 1-10 are assigned to only take the loops with the same digit as their spmdIndex

1. Create a Slurm file (spmd_test.sl) that calls your MATLAB code file (spmd_matlab.m).

```
#!/bin/bash
#SBATCH --job-name=spmd
#SBATCH --time=12:00:00
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --partition=shared
#SBATCH --mem-per-cpu=4G
#SBATCH --cpus-per-task=1
#SBATCH --mail-type=ALL
#SBATCH --mail-user=username@bc.edu

module load matlab/2024a
cd /mmfs1/data/johnchris/parallel_matlab
matlab -batch spmd_matlab >&spmd_matlab.out
```

Note: Underline denotes input individual users MUST change.

2. Configure the MATLAB code (spmd_matlab.m) with desired parallel computing process:

```
c = parcluster;
c.AdditionalProperties.WallTime='6:00:00';
c.AdditionalProperties.MemPerCPU='6G';
c.AdditionalProperties.ProcsPerNode=10;
c.AdditionalProperties.Partition='shared';
saveProfile(c)
parpool(10)
```

```
tic
spmd(10)
n = 20000;
A = 500;
a = zeros(1,n);
for i = 1:n
   if mod(i, 10) == mod(labindex,10)
   a(i) = max(abs(eig(rand(A))));
   end
end
end
toc
delete(gcp('nocreate'));
```

3. The other steps are the same as Step 1 to Step 5 in the “Submitting Jobs” section.