Introduction to MATLAB in HPC Environment

Sagnik Singha
High Performance Computing Center
(on behalf of the HPCC staff)

Nov 9th, 2022
Outline

• MATLAB in the TTU HPCC environment
• Job submission methods
• Batch Submission
• Graphical Client based Submission
• Containerized MATLAB (MathWorks account)
Logging on to TTU HPCC

https://www.depts.ttu.edu/hpcc/userguides/general_guides/login_general.php

Connecting to RedRaider – On Campus

ssh <eraider>@login.hpcc.ttu.edu
Setup MATLAB environment
module load matlab

Default version loaded (nocona): matlab/R2021b

check if MATLAB is currently loaded in your environment, please type:
module list

```
cpu-26-18:/MATLAB_testing/nocona/matlabPar$ ml matlab
Currently Loaded Modules:
  1) nocona/6.15.4  2) matlab/R2021b
cpu-26-18:/MATLAB_testing/nocona/matlabPar$ ml list
```

```
quannah:$ ml matlab
Currently Loaded Modules:
  1) matlab/R2020a
quannah:$ ml list
```
Job submission methods

• Graphical User Interface –
  ❖ If code is in developmental stages
  ❖ If direct interaction with output necessary
  ❖ Worker node

• BATCH submission –
  ❖ If code is in production stages
  ❖ If multiple runs needed without direct interaction with output
  ❖ Login node
Example file's location:
/lustre/work/examples/nocona/matlab

Copy files to own workspace
`cp -r /lustre/work/examples/nocona/matlab /home/<eraider_ID>/`

Example 1:

Matrix multiplication
Files:
A_matrix.dat
B_matrix.dat
atimesb.txt
matlab.sh
matlab.sh
#!/bin/bash
#SBATCH --job-name=matlab-test
#SBATCH --output=%x.o%j
#SBATCH --error=%x.e%j
#SBATCH --partition nocona
#SBATCH --nodes=1
#SBATCH --ntasks=1
module load matlab
matlab -batch multiplyMatrix

matmulfile.m
load A.dat;
load B.dat;
atimesb = A*B
save -ascii atimesb.txt atimesb
quit
Batch submission and Monitoring

Submission
sbatch matlab.sh

Monitoring
squeue -u <username>

Status 1:
```
cpu-26-18:/MATLAB_testing/nocona/matlab/Example_1$ sbatch matlab.sh
Submitted batch job 7599523
```
```
cpu-26-18:/MATLAB_testing/nocona/matlab/Example_1$ squeue -u ssingha
JOBID PARTITION  NAME    USER ST        TIME NODES NODELIST(REASON)
7599523 nocona matlab-t ssingha PD 0:00 1 (NONE)
```

Status 2:
```
cpu-23-38:/MATLAB_testing/nocona/matlab/Example_1$ squeue -u ssingha
JOBID PARTITION  NAME    USER ST    TIME  NODES NODELIST(REASON)
7628458 nocona INTERACT ssingha R  2:13:20 1 cpu-23-38
7628776 nocona matlab-t ssingha R  0:11  1 cpu-23-38
```

Status 3:
```
cpu-23-38:/MATLAB_testing/nocona/matlab/Example_1$ squeue -u ssingha
JOBID PARTITION  NAME    USER ST    TIME  NODES NODELIST(REASON)
7628458 nocona INTERACT ssingha R  2:13:22 1 cpu-23-38
```

Batch submission - Monitoring

<table>
<thead>
<tr>
<th>Status 2:</th>
</tr>
</thead>
</table>
| ```bash
$ squeue -u <username>
```

<table>
<thead>
<tr>
<th>JobID</th>
<th>Partition</th>
<th>Name</th>
<th>User ST</th>
<th>Time</th>
<th>Nodes</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>7628458</td>
<td>nocona INTERACT</td>
<td>ssingha</td>
<td>R</td>
<td>2:13:20</td>
<td>1 cpu-23-38</td>
<td></td>
</tr>
<tr>
<td>7628776</td>
<td>nocona matlab-t</td>
<td>ssingha</td>
<td>R</td>
<td>0:11</td>
<td>1 cpu-23-38</td>
<td></td>
</tr>
</tbody>
</table>

```bash
$ ssh cpu-23-23
$ top | grep MATLAB
```

<table>
<thead>
<tr>
<th>PID</th>
<th>User</th>
<th>%CPU</th>
<th>%MEM</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>5318</td>
<td>ssingha</td>
<td>21.3g</td>
<td>133.3</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4541</td>
<td>ssingha</td>
<td>21.3g</td>
<td>105.6</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4537</td>
<td>ssingha</td>
<td>21.3g</td>
<td>100.0</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4544</td>
<td>ssingha</td>
<td>21.3g</td>
<td>94.4</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4539</td>
<td>ssingha</td>
<td>21.3g</td>
<td>88.9</td>
<td>MATLAB</td>
</tr>
<tr>
<td>5074</td>
<td>ssingha</td>
<td>21.3g</td>
<td>83.3</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4534</td>
<td>ssingha</td>
<td>21.3g</td>
<td>77.8</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4531</td>
<td>ssingha</td>
<td>21.7g</td>
<td>55.6</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4534</td>
<td>ssingha</td>
<td>22.1g</td>
<td>185.0</td>
<td>MATLAB</td>
</tr>
<tr>
<td>5074</td>
<td>ssingha</td>
<td>22.1g</td>
<td>183.6</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4539</td>
<td>ssingha</td>
<td>22.1g</td>
<td>181.0</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4537</td>
<td>ssingha</td>
<td>22.6g</td>
<td>97.7</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4544</td>
<td>ssingha</td>
<td>22.1g</td>
<td>97.7</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4541</td>
<td>ssingha</td>
<td>22.8g</td>
<td>93.4</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4531</td>
<td>ssingha</td>
<td>22.2g</td>
<td>89.1</td>
<td>MATLAB</td>
</tr>
<tr>
<td>5318</td>
<td>ssingha</td>
<td>22.6g</td>
<td>87.8</td>
<td>MATLAB</td>
</tr>
<tr>
<td>3698</td>
<td>ssingha</td>
<td>23.2g</td>
<td>69.3</td>
<td>MATLAB</td>
</tr>
<tr>
<td>5074</td>
<td>ssingha</td>
<td>22.8g</td>
<td>99.3</td>
<td>MATLAB</td>
</tr>
<tr>
<td>4544</td>
<td>ssingha</td>
<td>22.5g</td>
<td>99.9</td>
<td>MATLAB</td>
</tr>
</tbody>
</table>
Batch submission - Monitoring

**Monitoring**
squeue -u <username>

**Files**
- matlab-test.e<JOB_ID>
- matlab-test.o<JOB_ID>
- atimesb.txt

**Status 1:**
```
cpu-26-18:/MATLAB_testing/nocona/matlab/Example_1$ sbatch matlab.sh
Submitted batch job 7599523
```

```
cpu-26-18:/MATLAB_testing/nocona/matlab/Example_1$ squeue --user ssingha

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7599523</td>
<td>nocona</td>
<td>matlab-t</td>
<td>ssingha</td>
<td>PD</td>
<td>0:00</td>
<td>1</td>
<td>(None)</td>
</tr>
</tbody>
</table>
```

**Status 2:**
```
cpu-23-38:/MATLAB_testing/nocona/matlab/Example_1$ squeue --user ssingha

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7628458</td>
<td>nocona</td>
<td>INTERACT</td>
<td>ssingha</td>
<td>R</td>
<td>2:13:20</td>
<td>1</td>
<td>cpu-23-38</td>
</tr>
<tr>
<td>7628776</td>
<td>nocona</td>
<td>matlab-t</td>
<td>ssingha</td>
<td>R</td>
<td>0:11</td>
<td>1</td>
<td>cpu-23-23</td>
</tr>
</tbody>
</table>
```

**Status 3:**
```
cpu-23-38:/MATLAB_testing/nocona/matlab/Example_1$ squeue --user ssingha
```

```
cpu-23-38:/MATLAB_testing/nocona/matlab/Example_1$ squeue --user ssingha

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7628458</td>
<td>nocona</td>
<td>INTERACT</td>
<td>ssingha</td>
<td>R</td>
<td>2:13:22</td>
<td>1</td>
<td>cpu-23-38</td>
</tr>
</tbody>
</table>
```
```
Example 2:

Matrix multiplication
Files:
example_2.m
matlab.sh

Files
• matlab-test.e<JOB_ID>
• matlab-test.o<JOB_ID>
• func_plot.png
Batch submission -
Parallel implementation, Maximum Eigenvalue of a matrix

Serial version

Code:
%Start timing
tic

%Calculates spectral radius of each matrix and
displays results
n = 10;
A = 500;
a = zeros(n);
for i = 1:n*n
    a(i) = max(abs(eig(rand(A))));
end
disp(a)

%End timing
time = toc;

%Display the time of computation
fprintf('The parallel method is executed in %5.2f
seconds. \n', time);

Parallel version

Uniform distribution of random numbers
Serial version

Code:
%Start timing
 tic
%Calculates spectral radius of each matrix and displays results
 n = 10;
 A = 500;
 a = zeros(n);
 for i = 1:n*n
     a(i) = max(abs(eig(rand(A))));
 end
 disp(a)
%End timing
 time = toc;
%Display the time of computation
 fprintf('The parallel method is executed in %5.2f seconds. \n', time);

Parallel version

Code:

 c = parcluster('local'); % local is a cluster profile name
 sz = str2num([getenv('SLURM_CPUS_PER_TASK')]);
 if isempty(sz), sz = maxNumCompThreads; end
 if isempty(gcp('nocreate')), c.parpool(sz); end

%Start timing
 tic
%Calculates spectral radius of each matrix and displays results
 n = 10;
 A = 500;
 a = zeros(n);
 parfor i = 1:n*n
     a(i) = max(abs(eig(rand(A))));
 end
 disp(a)
%End timing
 time = toc;
%Display the time of computation
 fprintf('The parallel method is executed in %5.2f seconds. \n', time);
Parallel implementation – Components

P1-
- Creates a cluster object representing the cluster identified by the cluster profile name *local*

Code:
```matlab
% Create a cluster object representing the cluster identified by the cluster profile name 'local'
c = parcluster('local');

% Get the number of CPUs per task from the environment variable 'SLURM_CPUS_PER_TASK'
sz = str2num(getenv('SLURM_CPUS_PER_TASK'));

% If the number of CPUs is empty, use the maximum number of threads
if isempty(sz), sz = maxNumCompThreads; end

% Check if the parallel pool is not created
if isempty(gcp('nocreate')), c.parpool(sz); end

% Start timing
tic

% Calculate the spectral radius of each matrix and display the results
n = 10;
A = 500;
a = zeros(n);
parfor i = 1:n*n
    a(i) = max(abs(eig(rand(A))));
end
disp(a)

% End timing
time = toc;

% Display the time of computation
fprintf('The parallel method is executed in %5.2f seconds.
', time);
```
Parallel implementation – Components

P1-
• Picks up --cpus-per-task from batch submit script
• Assigns to the variable sz
• String to number check to ensure correct format

Parallel version

Code:

c = parcluster('local'); % local is a cluster profile name
sz = str2num([getenv('SLURM_CPUS_PER_TASK')]);
if isempty(sz), sz = maxNumCompThreads; end
if isempty(gcp('nocreate')), c.parpool(sz); end

%Start timing
tic

%Calculates spectral radius of each matrix and displays results
n = 10;
A = 500;
a = zeros(n);
parfor i = 1:n*n
    a(i) = max(abs(eig(rand(A))));
end
disp(a)

%End timing
time = toc;

%Display the time of computation
fprintf('The parallel method is executed in %5.2f seconds.
\n', time);
Parallel implementation – Components

P1-
• If --cpus-per-task not defined, maximum number of cores are allotted

Parallel version

Code:
c = parcluster('local'); % local is a cluster profile name
sz = str2num([getenv('SLURM_CPUS_PER_TASK')]);
if isempty(sz), sz = maxNumCompThreads; end
if isempty(gcp('nocreate')), c.parpool(sz); end

%Start timing
tic

%Calculates spectral radius of each matrix and displays results
n = 10;
A = 500;
a = zeros(n);
parfor i = 1:n*n
    a(i) = max(abs(eig(rand(A))));
end
disp(a)

%End timing
time = toc;

%Display the time of computation
fprintf('The parallel method is executed in %5.2f seconds. \n', time);
P1-

- If a parallel pool is not created, creates one and takes the size from --cpus-per-task

**Parallel version**

```matlab
Code:
c = parcluster('local'); % local is a cluster profile name
sz = str2num([getenv('SLURM_CPUS_PER_TASK')]);
if isempty(sz), sz = maxNumCompThreads; end
if isempty(gcp('nocreate')), c.parpool(sz); end

%Start timing
tic

%Calculates spectral radius of each matrix and displays results
n = 10;
A = 500;
a = zeros(n);
parfor i = 1:n*n
    a(i) = max(abs(eig(rand(A))));
end
disp(a)

%End timing
time = toc;

%Display the time of computation
fprintf('The parallel method is executed in %5.2f seconds.\n', time);
```
Parallel implementation – Components

P2- `parfor` splits the execution of for-loop iterations over the workers in a parallel pool

Parallel version

Code:
```matlab
% Query for available cores (assume either Slurm or PBS)
sz = str2num(getenv('SLURM_CPUS_PER_TASK'));
if isempty(sz), sz = maxNumCompThreads; end
if isempty(gcp('nocreate')), c.parpool(sz); end

c = parcluster('local'); % local is a cluster profile name
% Start timing
tic

% Calculates spectral radius of each matrix and displays results
n = 10;
A = 500;
a = zeros(n);
parfor i = 1:n*n
    a(i) = max(abs(eig(rand(A))));
end
disp(a)
% End timing
time = toc;

fprintf('The parallel method is executed in %5.2f seconds.\n', time);
```

P2
## Parallel implementation

### Benefits

<table>
<thead>
<tr>
<th></th>
<th>Serial version</th>
<th></th>
<th>Parallel version</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st Run</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 core</td>
<td></td>
<td></td>
<td>10 cores</td>
<td></td>
</tr>
<tr>
<td>1304.93 seconds (~21.75 mins)</td>
<td></td>
<td>133.62 seconds (~2.3 mins)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Next Run</th>
<th></th>
<th>Parallel version</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 core</td>
<td></td>
<td></td>
<td>10 cores</td>
<td></td>
</tr>
<tr>
<td>10.31 seconds</td>
<td></td>
<td></td>
<td>1.89 seconds</td>
<td></td>
</tr>
</tbody>
</table>
### Parallel implementation - Multi core parallel implementation

Nocona – 128 cores/node

[https://www.depts.ttu.edu/hpcc/operations/equipment.php](https://www.depts.ttu.edu/hpcc/operations/equipment.php)

<table>
<thead>
<tr>
<th>Partition:</th>
<th>Nocona</th>
<th>Quanah / XL</th>
<th>Matador</th>
<th>Toreador</th>
<th>Ivy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>CPU</td>
<td>CPU</td>
<td>GPU</td>
<td>GPU</td>
<td>Aux CPU*</td>
</tr>
<tr>
<td>Total Nodes</td>
<td>240</td>
<td>467 / 16</td>
<td>20</td>
<td>11</td>
<td>50 / 2</td>
</tr>
<tr>
<td>Theoretical Max</td>
<td>983 TFLOPS / 565 TFLOPS</td>
<td>280 TFLOPS / 287 TFLOPS</td>
<td>40 TFLOPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benchmarked</td>
<td>804 TFLOPS / (N/A)</td>
<td>226 TFLOPS / N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>CentOS 8.1</td>
<td>CentOS 7.4 / CentOS 8.1</td>
<td>CentOS 8.1</td>
<td>CentOS 8.1</td>
<td>Rocky Linux 8.5 / CentOS 8.1</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>Dell</td>
<td>Dell</td>
<td>Dell</td>
<td>Dell</td>
<td>Dell</td>
</tr>
<tr>
<td>Node Model</td>
<td>PowerEdge C6220</td>
<td>PowerEdge C6220</td>
<td>Poweredge R740</td>
<td>Poweredge R7525</td>
<td>Poweredge C6220 II</td>
</tr>
<tr>
<td>Cooling</td>
<td>Liquid Cooled</td>
<td>Air Cooled</td>
<td>Air Cooled</td>
<td>Air Cooled</td>
<td>Air Cooled</td>
</tr>
<tr>
<td>Processor Make and Model</td>
<td>AMD EPYC™ 7702</td>
<td>Intel Xeon ES-2695 v4</td>
<td>Intel Xeon Gold 6242</td>
<td>AMD EPYC™ 7262</td>
<td>Intel Xeon ES-2670v2</td>
</tr>
<tr>
<td>Family</td>
<td>Rome</td>
<td>Broadwell</td>
<td>Cascade Lake</td>
<td>Rome</td>
<td>Ivy Bridge</td>
</tr>
<tr>
<td>Cores/Processor</td>
<td>64</td>
<td>36</td>
<td>20</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Cores/Node</td>
<td>128</td>
<td>36</td>
<td>40 cpu + 1280 tensor + 10,240 CUDA</td>
<td>16 cpu + 1,296 tensor + 20,736 CUDA</td>
<td>20</td>
</tr>
<tr>
<td>Total Cores in Partition</td>
<td>30,720</td>
<td>16,812</td>
<td>800 cpu + 25,600 tensor</td>
<td>528 cpu + 14,256 tensor</td>
<td>1,000</td>
</tr>
<tr>
<td></td>
<td>576</td>
<td>76</td>
<td>528 cpu + 14,256 tensor</td>
<td>528 cpu + 14,256 tensor</td>
<td>40</td>
</tr>
</tbody>
</table>
MATLAB Graphical Interface

Interactive session
$ interactive -p nocona -c 8
$ interactive -h

Setting up environment
$module load matlab
$ml list

Launching Client
$matlab

Number of cores
## Parallel implementation – Maximum Eigenvalue of a matrix

<table>
<thead>
<tr>
<th>Serial version</th>
<th>Parallel version</th>
</tr>
</thead>
<tbody>
<tr>
<td>parTest_serial.m</td>
<td>parTest.m</td>
</tr>
</tbody>
</table>

**Link**: /matlab/matlabPar/parTest_interactive
Parallel implementation – Maximum Eigenvalue of a matrix

**Serial version**

```
Code:
%Start timing
tic

%Calculates spectral radius of each matrix and displays results
n = 10;
A = 500;
a = zeros(n);
for i = 1:n*n
    a(i) = max(abs(eig(rand(A))));
end
disp(a)

%End timing
time = toc;

%Display the time of computation
fprintf('The serial method is executed in %5.2f seconds. \n', time);
```

**Parallel version**

```
Code:
c = parcluster('local'); % local is a cluster profile name
p = c.parpool(10);

%Start timing
tic

%Calculates spectral radius of each matrix and displays results
n = 10;
A = 500;
a = zeros(n);
parfor i = 1:n*n
    a(i) = max(abs(eig(rand(A))));
end
disp(a)

%End timing
time = toc;

%Display the time of computation
fprintf('The parallel method is executed in %5.2f seconds. \n', time);
```
Parallel implementation –
Maximum Eigenvalue of a matrix.

Object representing cluster identified by the cluster profile name *local*

Starts a parallel pool of 10 workers using the cluster object *c*

- Splits the execution of for-loop iterations over the workers in a parallel pool
- Say $n^2=10$, 10 workers available, MATLAB assigns it to each core

**Parallel version**

**Code:**

```matlab
% Object representing cluster identified by the cluster profile name 'local'
c = parcluster('local');
p = c.parpool(10);
%Start timing
tic
%Calculates spectral radius of each matrix and displays results
n = 10;
A = 500;
a = zeros(n);
parfor i = 1:n*n
    a(i) = max(abs(eig(rand(A))));
end
disp(a)
%End timing
time = toc;
%Display the time of computation
fprintf('The parallel method is executed in %5.2f seconds. \n', time);
```
MATLAB Graphical Interface – Parallel implementation via parpool

Client outputs

```matlab
>> parTest_serial
...
The parallel method is executed in **12.94** seconds.
```

```matlab
>> parTest
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 1).
...
The parallel method is executed in **10.69** seconds.
Parallel pool using the 'local' profile is shutting down.
```

```matlab
>>... Connected to the parallel pool (number of workers: 2).
The parallel method is executed in **5.70** seconds ...
```

```matlab
>>... Connected to the parallel pool (number of workers: 8).
The parallel method is executed in **2.56** seconds ...
```
Parallel implementation -
Multi core parallel implementation

Nocona – 128 cores/node

https://www.depts.ttu.edu/hpcc/operations/equipment.php
Job submission – Asynchronous Batch GUI

- Run once
- Re-runs deletes older profile, creates a new profile
- All saved data deleted

- `>> c=parcluster;`
  - Default cluster profile is set as `redraider`
  - Always makes a new secondary job submission to Slurm

- `>> c=parcluster('local');`
  - Opens cluster profile to utilize present allocation
  - Uses just resources on present node, no submission to scheduler
Profile Properties

- Slurm properties that can be modified
- `AdditionalSubmitArgs` – Manually allows user to pass arguments, flags say constrains of the cluster
- `c.Additional.Properties.NumberofNodes = 1;`
- `c.AdditionalProperties.Partitions='nocona';`
- `c.saveProfile`
- Creates a submission script and performs the submission for the user
Job submission – Asynchronous Batch GUI

Job object name | Function name | Number of outputs
---|---|---

```matlab
>> j=batch(c,@pwd,1,{});
```

**Inputs to function**

```matlab
>> j.State – returns status of job
```

```matlab
>> j.fetchOutputs – returns output of job
```

```matlab
>> j.diary – returns command line outputs
```
Serial Example

function [t, A] = parallel_example(iter)
if nargin==0
    iter = 8;
end

disp('Start sim')
t0 = tic;
for idx = 1:iter
    A(idx) = idx;
    pause(2)
    idx
end
t = toc(t0);
disp('Sim completed')
save RESULTS A
end

>> j=batch(c,@serial_example,1,{8});

>> j.State

>> j.fetchOutputs

ans =
1×1 cell array
{{[16.0098]}}
Parallel Example

function [t, A] = parallel_example(iter)
    if nargin==0
        iter = 8;
    end
    disp('Start sim')
    t0 = tic;
    parfor idx = 1:iter
        A(idx) = idx;
        pause(2)
        idx
    end
    t = toc(t0);
    disp('Sim completed')
    save RESULTS A
    end

>> j=batch(c,@serial_example,1,{8});

{[16.0098]}

>>j=batch(c,@parallel_example,1,{8},'pool',2);

{[8.3045]}

>> =batch(c,@parallel_example,1,{8},'pool',4);

{[4.3124]}
Parallel Example

function [t, A] = parallel_example(iter)
    if nargin==0
        iter = 8;
    end
    disp('Start sim')
    t0 = tic;
    parfor idx = 1:iter
        A(idx) = idx;
        pause(2)
    end
    t = toc(t0);
    disp('Sim completed')
    save RESULTS A
end
Job submission – Asynchronous Batch GUI

Job Arrays

```matlab
>> j = batch(c, @sum, 1, {[1 1]});

>> j = createJob(c);
createTask(j, @sum, 1, {[1 1]});
createTask(j, @sum, 1, {[1 1]});
createTask(j, @sum, 1, {[1 1]});
submit(j);
```

```
additionalSubmitArgs =
    '-n tasks=1 --cpus-per-task=1 --ntasks-per-core=1 -p nocona --mem-per-cpu=4gb --nodes=1'
```

```matlab
>> j.Tasks
ans =
3x1 Task array:
    ID   State            FinishDateTime Function  Errors  Warnings  SchedulerID
    1    running         sum     0 0          7632475_1
    2    pending        sum     0 0          7632475_2
    3    pending        sum     0 0          7632475_3
```
Job submission – Asynchronous Batch GUI

Job Arrays

```
>> j.Tasks
ans =
3x1 Task array:
      ID  State FinishDateTime Function Errors Warnings
      1   1 pending          sum   0   0
      2   2 pending          sum   0   0
      3   3 pending          sum   0   0
```

Monitoring job parameters

```
>> j.Tasks(2).cancel
>> j.Tasks
ans =
3x1 Task array:
      ID  State FinishDateTime Function Errors Warnings
      1   1 pending          sum   0   0
      2   2 finished 06-Nov-2022 00:11:14 sum   1   0
      3   3 pending          sum   0   0
```
Job submission – Interactive pool submission

Opening an interactive parpool (local profile)

>> c=parcluster('local');
>> p=c.parpool(n)

n is the number of cores across the cluster requested

Running a parpool job (local profile)

>> parallel_example
Another parallel construct like `parfor`
Job submission – Interactive pool submission

Shutting down Parallel pool

>> clear all
Job submission – Interactive pool submission

Monitoring
Parallel job implementation example

Calculation of $\pi$
(based on Leibniz formula)

$$\int_{0}^{1} \frac{4}{1 + x^2} dx = 4 \arctan(1) = \pi$$
## Single node parallel

```matlab
function calc_pi
    c = parcluster('local');
    sz = str2num([getenv('SLURM_CPUS_PER_TASK')]); %#ok<ST2NM>
    sz = str2num([getenv('SLURM_CPUS_PER_TASK')]); %#ok<ST2NM>
    if isempty(sz), sz = maxNumCompThreads; end
    if isempty(gcp('nocreate')), c.parpool(sz); end
    spmd
        a = (labindex - 1)/numlabs;
        b = labindex/numlabs;
        fprintf('Subinterval: [%-4g, %-4g]\n', a, b)
        myIntegral = integral(@quadpi, a, b);
        fprintf('Subinterval: [%-4g, %-4g] Integral: %4g\n', a, b, myIntegral)
        piApprox = gplus(myIntegral);
    end
    approx1 = piApprox{1};  % 1st element holds value on worker 1
    fprintf('pi Approximation: %4g\n', approx1)
end
```

## Multi node parallel

```matlab
function calc_pi_multi_node
    c = parcluster;
    % Required fields
    c.AdditionalProperties.WallTime = '00:20:00';
    c.AdditionalProperties.Partition = 'nocona';
    c.AdditionalProperties.NumberOfNodes = 1;
    if isempty(gcp('nocreate')), c.parpool(20); end
    spmd
        a = (labindex - 1)/numlabs;
        b = labindex/numlabs;
        fprintf('Subinterval: [%-4g, %-4g]\n', a, b)
        myIntegral = integral(@quadpi, a, b);
        fprintf('Subinterval: [%-4g, %-4g] Integral: %4g\n', a, b, myIntegral)
        piApprox = gplus(myIntegral);
    end
    approx1 = piApprox{1};  % 1st element holds value on worker 1
    fprintf('pi Approximation: %4g\n', approx1)
end
```

### Quadpi function

```matlab
function y = quadpi(x)
    %QUADPI Return data to approximate pi.
    % Derivative of 4*atan(x)
    y = 4./(1 + x.^2);
end
```
Parallel job implementation example

Accuracy of numerical calculation

```plaintext
>> calc_pi
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).
Worker 1:
  Subinterval: [0  , 0.25]
  Subinterval: [0   , 0.25]  Integral: 0.979915
Worker 2:
  Subinterval: [0.25, 0.5 ]
  Subinterval: [0.25, 0.5 ]  Integral: 0.874676
Worker 3:
  Subinterval: [0.5 , 0.75]
  Subinterval: [0.5 , 0.75]  Integral: 0.719414
Worker 4:
  Subinterval: [0.75, 1 ]
  Subinterval: [0.75, 1 ]  Integral: 0.567588
pi   : 3.141592653589793116
Approximation: 3.141592653589793560
Error  : 4.44089e-16
```

```plaintext
>> calc_pi_multi_node
Starting parallel pool (parpool) using the 'teton R2022a' profile ...
additionalSubmitArgs =
  '--ntasks=20 --cpus-per-task=1 --ntasks-per-core=1
Connected to the parallel pool (number of workers: 20).
Worker 1:
  Subinterval: [0  , 0.025]
Worker 2:
  Subinterval: [0.025, 0.05]
Worker 3:
  Subinterval: [0.025, 0.05]
Worker 4:
  Subinterval: [0.025, 0.05]
Worker 5:
  Subinterval: [0.025, 0.05]
Worker 6:
  Subinterval: [0.025, 0.05]
Worker 7:
  Subinterval: [0.025, 0.05]
Worker 8:
  Subinterval: [0.025, 0.05]
Worker 9:
  Subinterval: [0.025, 0.05]
Worker 10:
  Subinterval: [0.025, 0.05]
Worker 11:
  Subinterval: [0.025, 0.05]
Worker 12:
  Subinterval: [0.025, 0.05]
Worker 13:
  Subinterval: [0.025, 0.05]
Worker 14:
  Subinterval: [0.025, 0.05]
Worker 15:
  Subinterval: [0.025, 0.05]
Worker 16:
  Subinterval: [0.025, 0.05]
Worker 17:
  Subinterval: [0.025, 0.05]
Worker 18:
  Subinterval: [0.425, 0.45]  Integral: 0.0839331
Worker 19:
  Subinterval: [0.425, 0.45]
Worker 20:
  Subinterval: [0.525, 0.55]  Integral: 0.0775848
pi   : 3.141592653589793116
Approximation: 3.141592653589793116
Error  : 0
```
Containerized MATLAB

1. Login into login.hpcc.ttu.edu using your eRaider account and password.
2. Get an interactive session on a partition (say nocona)
   $ interactive -p nocona
3. Check singularity installation and version
   $ singularity --version
   singularity version 3.7.3-1.el8
4. Pull a sample singularity container to get started
   $ singularity pull shub://singularityhub/hello-world
1. In the event of errors, re-runs execute
   $ singularity cache list
   to list container files, present in the account memory
   $ singularity cache clean
   to remove such files before beginning a fresh pull
2. Execute the singularity image file (*.sif)
   $ ./hello-world_latest.sif
   Tacotacotaco
3. Alternate ways to explore yourself:
   Directly run a singularity container
   $ singularity run hello-world_latest.sif
4. Pull latest MATLAB docker container:
   $ singularity pull docker://mathworks/matlab:r2022b
Thoughts?

Feedback