Introduction to MATLAB in HPC Environment

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

November 12th, 2025

Outline

TEXAS TECH UNIVERSITY
Information Technology Division

- MATLAB in the TTU HPCC environment
- Job submission methods
- Batch Submission
- Graphical Client based Submission
- Parallel computing constructs – CPU, GPU
- Containerized MATLAB (MathWorks account)
- National HPC Resources and their allocations

Logging on to TTU HPCC

TEXAS TECH UNIVERSITY
Information Technology Division

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

Connecting to RedRaider – On Campus

ssh <eraider>@login.hpcc.ttu.edu

MATLAB in the TTU HPCC - Environment setup

TEXAS TECH UNIVERSITY
Information Technology Division

Setup MATLAB environment
`module load matlab`

Default version loaded (nocona): `matlab/matlab/R2024a`

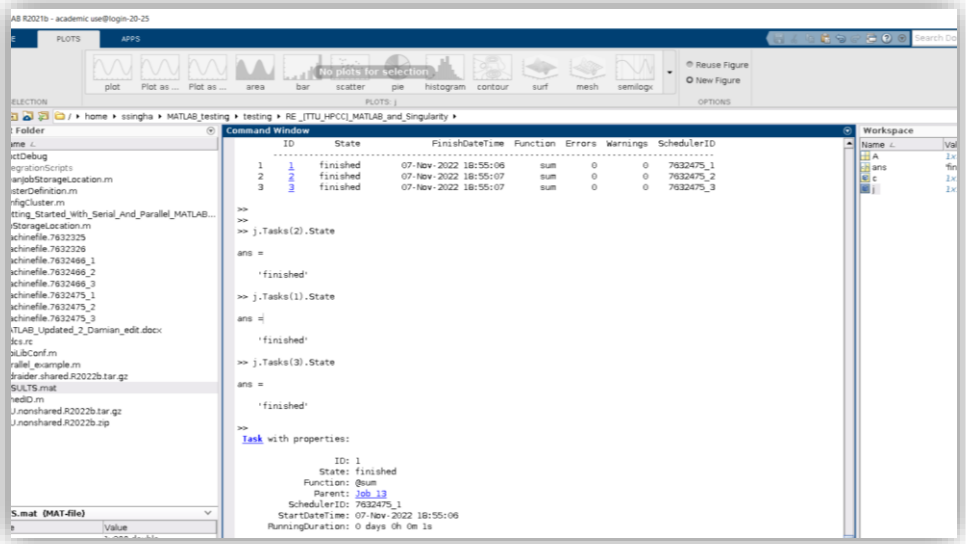
check if MATLAB is currently loaded in your environment, please type:
`module list`

```
cpu-24-34:$ ml matlab
cpu-24-34:$ ml list

Currently Loaded Modules:
  1) nocona/0.15.4   2) matlab/R2024a
```

```
cpu-1-16:/matlab_shortcourse$ ml list

Currently Loaded Modules:
  1) matlab/R2024a
```

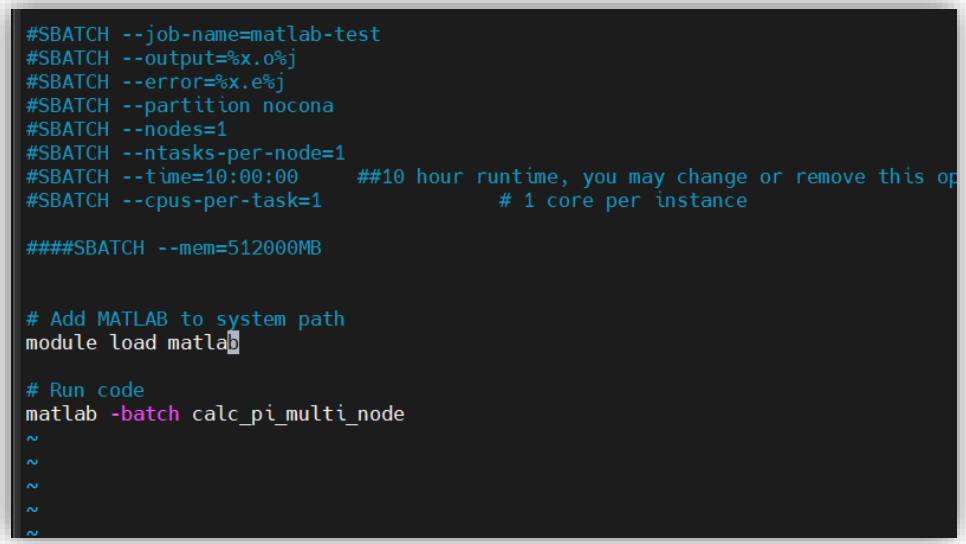
Job submission methods

- Graphical User Interface –
  - Interact directly with the job
  - Prototyping, troubleshooting code - If code is in developmental stages
  - If direct interaction with output necessary
  - Worker node



- BATCH submission –
  - Automate jobs in Linux environment
  - Production, submitting code - If code is in production stages
  - If multiple runs needed without direct interaction with output
  - Login node

https://slurm.schedmd.com/sbatch.html



Batch submission

Example file's location:
`/lustre/work/examples/nocona/matlab/fall_2025`

Copy files to own workspace
`cp -r /lustre/work/examples/nocona/matlab/fall_2025 /home/<eraider_ID>/.`

Example 1:

Matrix multiplication
Files:
`A_matrix.dat`
`B_matrix.dat`
`matmulfile.m`
`matlab.sh`

3 X 3 Matrix

$$\begin{pmatrix} 1 & 23 & 456 \\ 54 & 4 & 412 \\ 0 & 99 & 69 \end{pmatrix}$$

Batch submission

TEXAS TECH UNIVERSITY
Information Technology Division

https://www.depts.ttu.edu/hpcc/userguides/Job_User_Guide.pdf

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;

n=100
A=rand(n)
B=rand(n)

atimesb = A*B
aeletimesb = A.*B

save -ascii atimesb.txt
atimesb
save -ascii aeletimesb.txt
aeletimesb
quit
```

Batch submission and Monitoring

TEXAS TECH UNIVERSITY
Information Technology Division

Submission

```
sbatch matlab.sh
```

Monitoring

```
squeue -u <username>
```

Status 1:

```
cpu-26-10:/MATLAB_testing/nocona/matlab/Example_1$ sbatch matlab.sh
Submitted batch job 7599523
cpu-26-10:/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-23-23
```

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

https://www.depts.ttu.edu/hpcc/userguides/Job_User_Guide.pdf

Batch submission - Monitoring

TEXAS TECH UNIVERSITY
Information Technology Division

Monitoring

```
$squeue -u <username>
```

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-23-23
```

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

```
cpu-23-1:$ top | grep MATLAB
  5318 ssingha   20   0   21.3g 475432 217068 S 133.3   0.1   0:03.12 MATLAB
  4541 ssingha   20   0   21.3g 466260 214360 S 105.6   0.1   0:03.11 MATLAB
  4537 ssingha   20   0   21.3g 482448 221040 S 100.0   0.1   0:03.14 MATLAB
  4544 ssingha   20   0   21.3g 486596 220236 S  94.4   0.1   0:03.15 MATLAB
  4539 ssingha   20   0   21.3g 489100 220840 S  88.9   0.1   0:03.19 MATLAB
  5074 ssingha   20   0   21.3g 480076 216208 S  83.3   0.1   0:03.08 MATLAB
  4534 ssingha   20   0   21.3g 476296 215080 S  77.8   0.1   0:03.11 MATLAB
  4531 ssingha   20   0   21.7g 736408 230976 S  55.6   0.1   0:03.75 MATLAB
  4534 ssingha   20   0   22.1g 951032 262028 S 105.0   0.2   0:06.29 MATLAB
  5074 ssingha   20   0   22.1g 953908 262056 S 103.6   0.2   0:06.22 MATLAB
  4539 ssingha   20   0   22.1g 952092 263620 S 101.0   0.2   0:06.25 MATLAB
  4537 ssingha   20   0   22.0g 936720 262352 S  97.7   0.2   0:06.10 MATLAB
  4544 ssingha   20   0   22.1g 923068 261824 S  97.7   0.2   0:06.11 MATLAB
  4541 ssingha   20   0   22.0g 896352 257788 S  93.4   0.2   0:05.94 MATLAB
  4531 ssingha   20   0   22.2g 976064 272068 S  89.1   0.2   0:06.45 MATLAB
  5318 ssingha   20   0   22.0g 895616 256076 S  87.8   0.2   0:05.78 MATLAB
  3698 ssingha   20   0   23.2g   1.2g 312528 S   0.3   0.2   0:13.07 MATLAB
  5074 ssingha   20   0   22.8g   1.1g 290956 S  99.3   0.2   0:09.23 MATLAB
  4544 ssingha   20   0   22.5g   1.1g 292092 S  99.0   0.2   0:09.11 MATLAB
```

Batch submission - Monitoring

TEXAS TECH UNIVERSITY
Information Technology Division

Monitoring

`squeue -u <username>`

Files

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

Status 1:

```
cpu-26-10:/MATLAB_testing/nocona/matlab/Example_1$ sbatch matlab.sh
Submitted batch job 7599523
cpu-26-10:/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-23-23
```

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

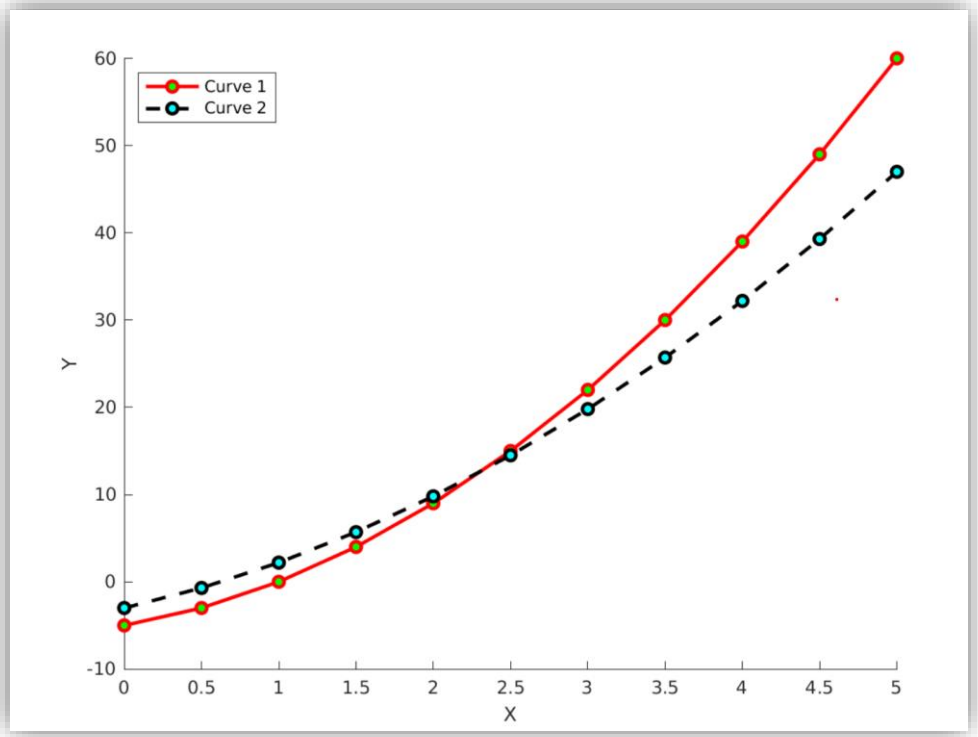
Batch submission

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

TEXAS TECH UNIVERSITY
Information Technology Division

Link: /matlab/matlabPar

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);
```

Uniform distribution of random numbers

Parallel version

Batch submission -

Parallel implementation, Maximum Eigenvalue of a matrix

TEXAS TECH UNIVERSITY
Information Technology Division

Link: /matlab/matlabPar

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);
```

P 1

P 2

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

TEXAS TECH UNIVERSITY
Information Technology Division

Link: /matlab/matlabPar

P1-

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

P 1

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

TEXAS TECH UNIVERSITY
Information Technology Division

Link: /matlab/matlabPar

P1-

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

P 1

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

TEXAS TECH UNIVERSITY
Information Technology Division

Link: /matlab/matlabPar

P1-

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

P 1

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

TEXAS TECH UNIVERSITY
Information Technology Division

Link: /matlab/matlabPar

P1-

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

Takes some extra time to initialize the local cluster

P 1

Parallel version

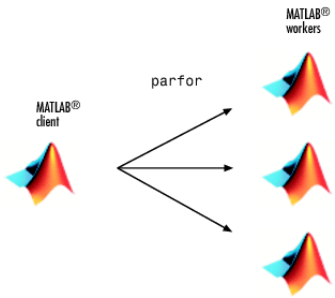
Code:

```
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

Link: /matlab/matlabPar

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



P 2

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- Benefits

TEXAS TECH UNIVERSITY
Information Technology Division

1st Run

Serial version	Parallel version
1 core	**10 cores**
1304.93 seconds(~21.75 mins)	133.62 seconds (~2.3 mins)

Next Run

Serial version	Parallel version
1 core	**10 cores**
10.31 seconds	1.89 seconds

Parallel implementation-
Multi core parallel implementation

Nocona – 128 cores/node

https://www.depts.ttu.edu/hpcc/operations/equipment.php

Partition:	Nocona	Quanah / XLQuanah	Matador	Toreador	Ivy
Type	CPU	CPU	GPU	GPU	Auxiliary CPU*
Total Nodes	240	467 / 16	20	11	50 / 2
Theoretical Max	983 TFLOPS	565 TFLOPS /19 TFLOPS	280 TFLOPS	287 TFLOPS	40 TFLOPS
Benchmarked	804 TFLOPS	485 TFLOPS / (N/A)	226 TFLOPS		N/A
OS	CentOS 8.1	CentOS 7.4 /CentOS 8.1	CentOS 8.1	CentOS 8.1	Rocky Linux 8.5 /CentOS 8.1
Manufacturer	Dell	Dell	Dell	Dell	Dell
Node Model	PowerEdge C6525	PowerEdge C6320	PowerEdge R740	Poweredge R7525	PowerEdge C6220 II
Cooling	Liquid Cooled	Air Cooled	Air Cooled	Air Cooled	Air Cooled
Processor Make and Model	AMD EPYC™ 7702	Intel Xeon E5-2695 v4	Intel Xeon Gold 6242	AMD EPYC™ 7262	Intel Xeon E5-2670v2
Family	Rome	Broadwell	Cascade Lake	Rome	Ivy Bridge
Cores/Processor	64	18	20	8	10
Cores/Node	128	36	40 cpu + 1280 tensor + 10,240 CUDA	16 cpu + 1,296 tensor + 20,736 CUDA	20
Total Cores In Partition	30,720	16,812 / 576	800 cpu + 25,600 tensor +	528 cpu + 14,256 tensor+	1,000 / 40

MATLAB Graphical Interface

Interactive session

$ interactive -p nocona -c 8

$ interactive -h

Number of cores

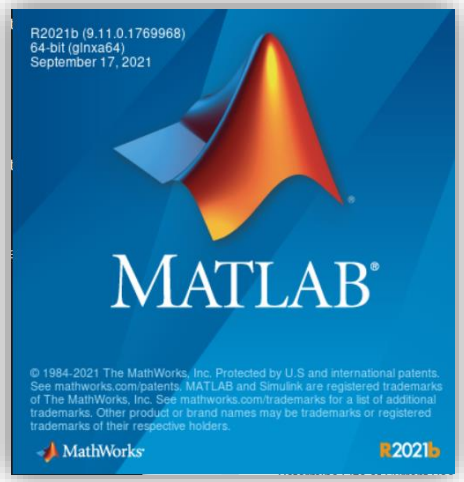
Setting up environment

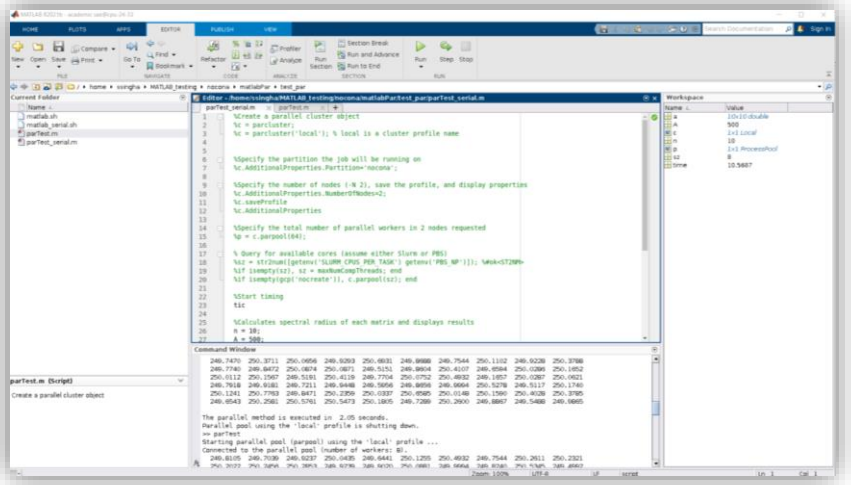
$module load matlab
$ml list

Launching Client

$matlab

```
cpu-23-14:/matlab_shortcourse$ ml matlab
cpu-23-14:/matlab_shortcourse$ ml list

Currently Loaded Modules:
  1) matlab/R2024a
```





Parallel implementation – Maximum Eigenvalue of a matrix

TEXAS TECH UNIVERSITY
Information Technology Division

Link: /matlab/matlabPar/parTest_interactive

Serial version

parTest_serial.m

Parallel version

parTest.m

Parallel implementation – Maximum Eigenvalue of a matrix

TEXAS TECH UNIVERSITY
Information Technology Division

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

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);
p.delete
```

Parallel implementation – Maximum Eigenvalue of a matrix

TEXAS TECH UNIVERSITY
Information Technology Division

Parallel version

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*n=10, 10 workers available, MATLAB assigns it to each core

- shuts down the parallel pool associated with the object *p*
- destroys the communicating job that comprises the pool

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);
p.delete
```

MATLAB Graphical Interface – Parallel implementation via parpool

TEXAS TECH UNIVERSITY
Information Technology Division

Client outputs

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

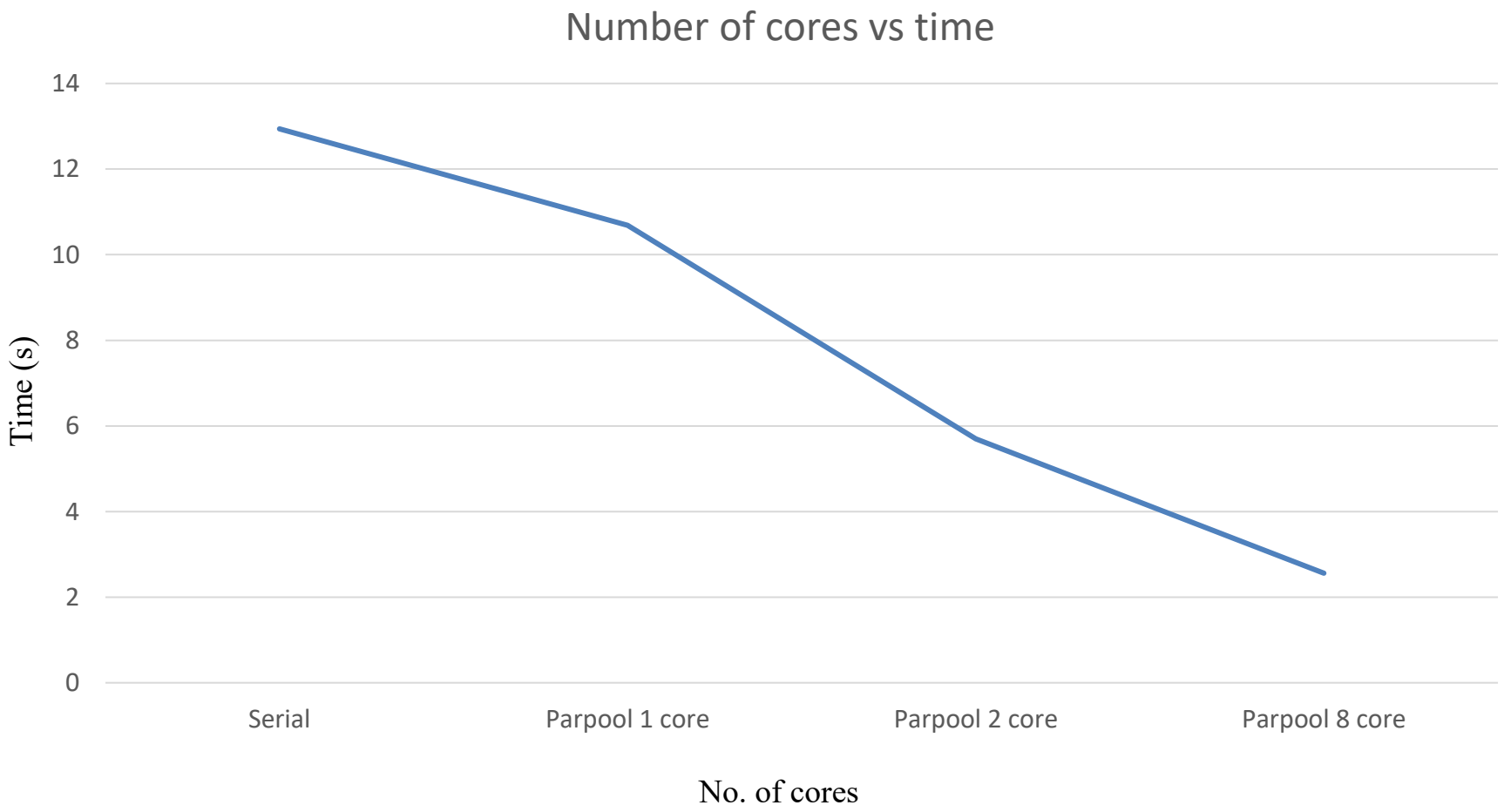
```
>> parTest
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 1).
…
The parallel method is executed in 10.69 seconds.
Parallel pool using the 'local' profile is shutting down.
```

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

```
>>… Connected to the parallel pool (number of workers: 8).
The parallel method is executed in 2.56 seconds …
```

MATLAB Graphical Interface – Parallel implementation via parpool

TEXAS TECH UNIVERSITY
Information Technology Division



Parallel constructs in MATLAB

parfor

- Do not allow communication between workers
- Require synchronization

spmd

- To enable you require fine-grained worker-to-worker communication and collaboration between workers
- Require synchronization

parfeval

- Do not allow communication between workers
- Jobs can be run asynchronously

Options for parallel computing in MATLAB

Serial version

Code:

```
%Start timing
tic

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

%End timing
time = toc;


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

Parfor version

Code:

```
%Create a parallel cluster object
% local is a cluster profile name
c = parcluster('local');
p = c.parpool(4);

%Start timing
tic

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

%End timing
time = toc

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

%Delete parallel workers
p.delete;
delete(p);
```

Options for parallel computing in MATLAB

Parfeval version

Code:

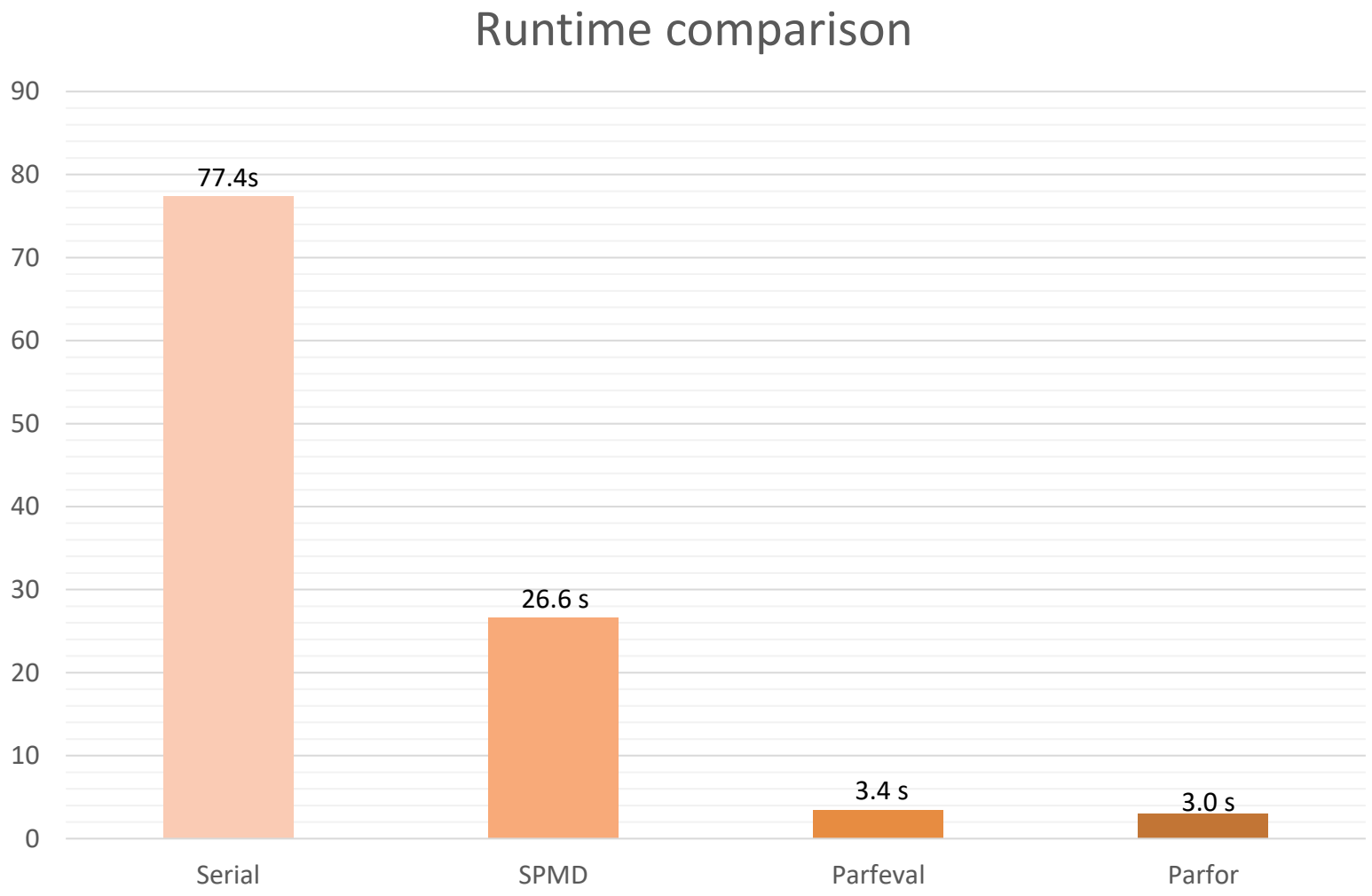
```
c = parcluster('local');
p = c.parpool(8);

tic
n = 500;
size = 500;
a = zeros(n); % Initialize as a column vector for easier indexing
feig = @(size) max(abs(eig(rand(size)))); % Corrected anonymous function
input

futures = cell(n); % Cell array to store Future objects

for i = 1:n
    futures{i} = parfeval(feig, 1, i); % Store the Future object
end

% Retrieve the results from the Future objects
for i = 1:n
    a(i) = fetchOutputs(futures{i});
end

toc

%Delete parallel workers
p.delete;
delete(p);
```

Spmd version

Code:

```
c = parcluster('local');
p = c.parpool(10);

tic

spmd (10)
    n = 500;
    A = 500;
    a = zeros(n);
    for i = 1:n
        a(i) = max(abs(eig(rand(i))));
    end
end

toc

p.delete;
delete(p);
```

Options for parallel computing in MATLAB



MATLAB in the TTU HPCC - GPU Environment setup

TEXAS TECH UNIVERSITY
Information Technology Division

Interactive session

$ interactive -p matador -c 6

$ interactive -h

Setting up environment

$module load matlab
$ml list

Launching Client

$matlab

Setup MATLAB environment
`module load matlab`

Version to load:
`matlab/matlab/R2023b`

Check loaded modules:
`module list`

GPU Array

TEXAS TECH UNIVERSITY
Information Technology Division

Serial version

Code:

```
%Start timing
tic

%Calculates spectral radius of each matrix and
displays results
n = 500;
A = 500;
a = zeros(n);
for i = 1: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);
```

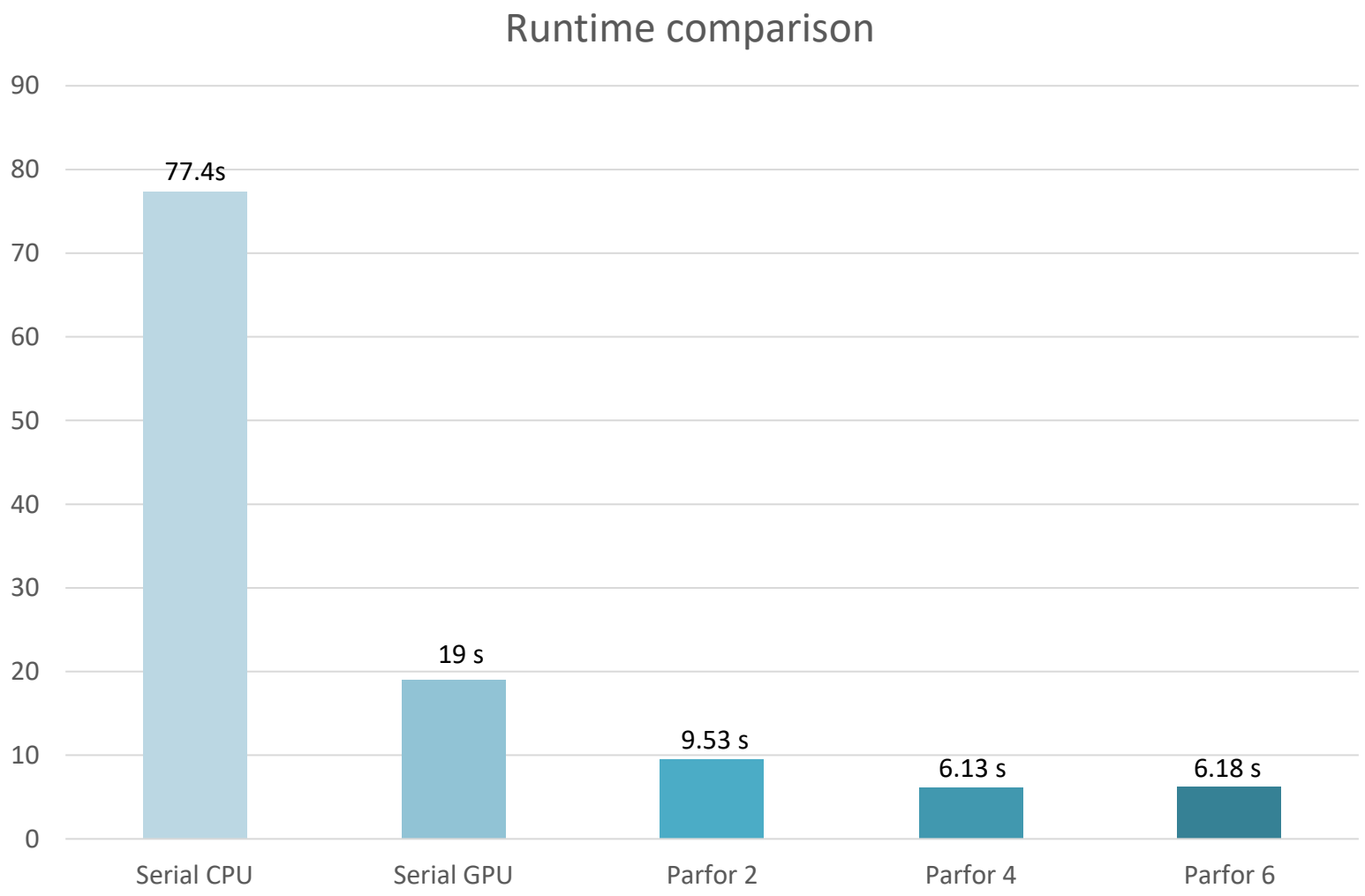
GPU Array

Parallel version

Code:

```
c = parcluster('local');
p = c.parpool(2);

%Start timing
tic
n = 500;
A = 500;
%a = zeros(n);
a = zeros(n,"gpuArray");
parfor i = 1:n
    a(i) = max(abs(eig(rand(i))));
End

%End timing
time = toc;

%Delete parallel workers
p.delete;
delete(p);
```

GPU Array


Runtime comparison
90
80
70
60
50
40
30
20
10
0
77.4s
19 s
9.53 s
6.13 s
6.18 s
Serial CPU
Serial GPU
Parfor 2
Parfor 4
Parfor 6

Break

Reconvene at 1 pm

Batch submission -

Prime Factorization – complexity ~ magnitude of number

TEXAS TECH UNIVERSITY
Information Technology Division

Serial version

Code:

```
primeNumbers = primes(uint64(2^21));
compositeNumbers =
primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

%Start timing
tic

%Calculates the prime factors of composite numbers
for idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
end
%End timing
time = toc;


%Display the time of computation
fprintf('The parallel method is executed in seconds=')
disp(time)
```

Returns and array of prime numbers

Batch submission -
Prime Factorization

TEXAS TECH UNIVERSITY
Information Technology Division

Serial version

Code:

```
primeNumbers = primes(uint64(2^21));
compositeNumbers =
primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

%Start timing
tic

%Calculates the prime factors of composite numbers
for idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
end
%End timing
time = toc;


%Display the time of computation
fprintf('The parallel method is executed in seconds=')
disp(time)
```

Returns no. of elements

Batch submission -
Prime Factorization

TEXAS TECH UNIVERSITY
Information Technology Division

Serial version

Code:

```
primeNumbers = primes(uint64(2^21));
compositeNumbers =
primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

%Start timing
tic

%Calculates the prime factors of composite numbers
for idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
end
%End timing
time = toc;


%Display the time of computation
fprintf('The parallel method is executed in seconds=')
disp(time)
```

Returns row matrix of random permutation

Batch submission -
Prime Factorization

TEXAS TECH UNIVERSITY
Information Technology Division

Serial version

Code:

```
primeNumbers = primes(uint64(2^21));
compositeNumbers =
primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

%Start timing
tic

%Calculates the prime factors of composite numbers
for idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
end
%End timing
time = toc;


%Display the time of computation
fprintf('The parallel method is executed in seconds=')
disp(time)
```

Returns a null matrix

Batch submission -
Prime Factorization

TEXAS TECH UNIVERSITY
Information Technology Division

Serial version

Code:

```
primeNumbers = primes(uint64(2^21));
compositeNumbers =
primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

%Start timing
tic

%Calculates the prime factors of composite numbers
for idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
end
%End timing
time = toc;


%Display the time of computation
fprintf('The parallel method is executed in seconds=')
disp(time)
```

Returns prime factors

Batch submission -
Prime Factorization

TEXAS TECH UNIVERSITY
Information Technology Division

Serial version

Code:

```
primeNumbers = primes(uint64(2^21));
compositeNumbers =
primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

%Start timing
tic

%Calculates the prime factors of composite numbers
for idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
end
%End timing
time = toc;


%Display the time of computation
fprintf('The parallel method is executed in seconds=')
disp(time)
```

Returns and array of prime numbers

Returns no. of elements

Returns row matrix of random permutation

Returns a null matrix

Returns prime factors

Batch submission -

Prime Factorization

Editor - /home/ssingha/MATLAB_User_Training_Spring_2023/testing/matlab/Example_4/example_4.m

example_4.m

```
1  primeNumbers = primes(uint64(2^21));
2  compositeNumbers = primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
3  factors = zeros(numel(primeNumbers),2);
4
5  tic;
6  for idx = 1:numel(compositeNumbers)
7      factors(idx,:) = factor(compositeNumbers(idx));
8  end
9  toc
10
11
12
```

Command Window

```
>>
>>
>> example_4
Elapsed time is 724.906721 seconds.
>>
>>
>>
>>
>>
>>
>>
```

Time of computation (serial)

Batch submission -
Prime Factorization

TEXAS TECH UNIVERSITY
Information Technology Division

Serial version

Code:

```
primeNumbers = primes(uint64(2^21));
compositeNumbers =
primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

%Start timing
tic

%Calculates the prime factors of composite numbers
for idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
end
%End timing
time = toc;


%Display the time of computation
fprintf('The parallel method is executed in seconds=')
disp(time)
```

Parallel version

Code:

```
%Create a parallel cluster object
c = parcluster('local'); % local is a cluster profile name

p = c.parpool(10);

primeNumbers = primes(uint64(2^11));
compositeNumbers =
primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

%Start timing
tic;

%Calculates the prime factors of composite numbers
parfor idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
End
%End timing
time=toc;

%Display the time of computation
fprintf('The parallel method is executed in seconds=')
disp(time)
delete(gcp('nocreate'))
```

Batch submission -
Benefits

TEXAS TECH UNIVERSITY
Information Technology Division

Editor - /home/ssingha/MATLAB_User_Training_Spring_2023/testing/matlab/Example_4/example_4.m

example_4.m

```
primeNumbers = primes(uint64(2^21));
compositeNumbers = primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

tic;
for idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
end
toc
```

Command Window

```
>>
>>
>> example_4
Elapsed time is 724.906721 seconds.
>>
>>
>>
>>
>>
>>
```

Time of computation (serial)

Editor - /home/ssingha/MATLAB_User_Training_Spring_2023/matlab/Example_4/example_4_parallel.m

example_4_parallel.m

```
%Create a parallel cluster object
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

p = c.parpool(10);

primeNumbers = primes(uint64(2^11));
compositeNumbers = primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
factors = zeros(numel(primeNumbers),2);

tic;
parfor idx = 1:numel(compositeNumbers)
    factors(idx,:) = factor(compositeNumbers(idx));
end
time=toc;

%Display the time of computation
fprintf('The parallel method is executed in seconds=')
disp(time)
```

Command Window

```
>>
>>
>>
>>
>>
>> example_4_parallel
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 10).
The parallel method is executed in seconds=    0.3251

Parallel pool using the 'local' profile is shutting down.
>>
>>
```

Time of computation (parallel)

Parallel implementation-
Multi core parallel implementation

TEXAS TECH UNIVERSITY
Information Technology Division

Nocona – 128 cores/node

https://www.depts.ttu.edu/hpcc/operations/equipment.php

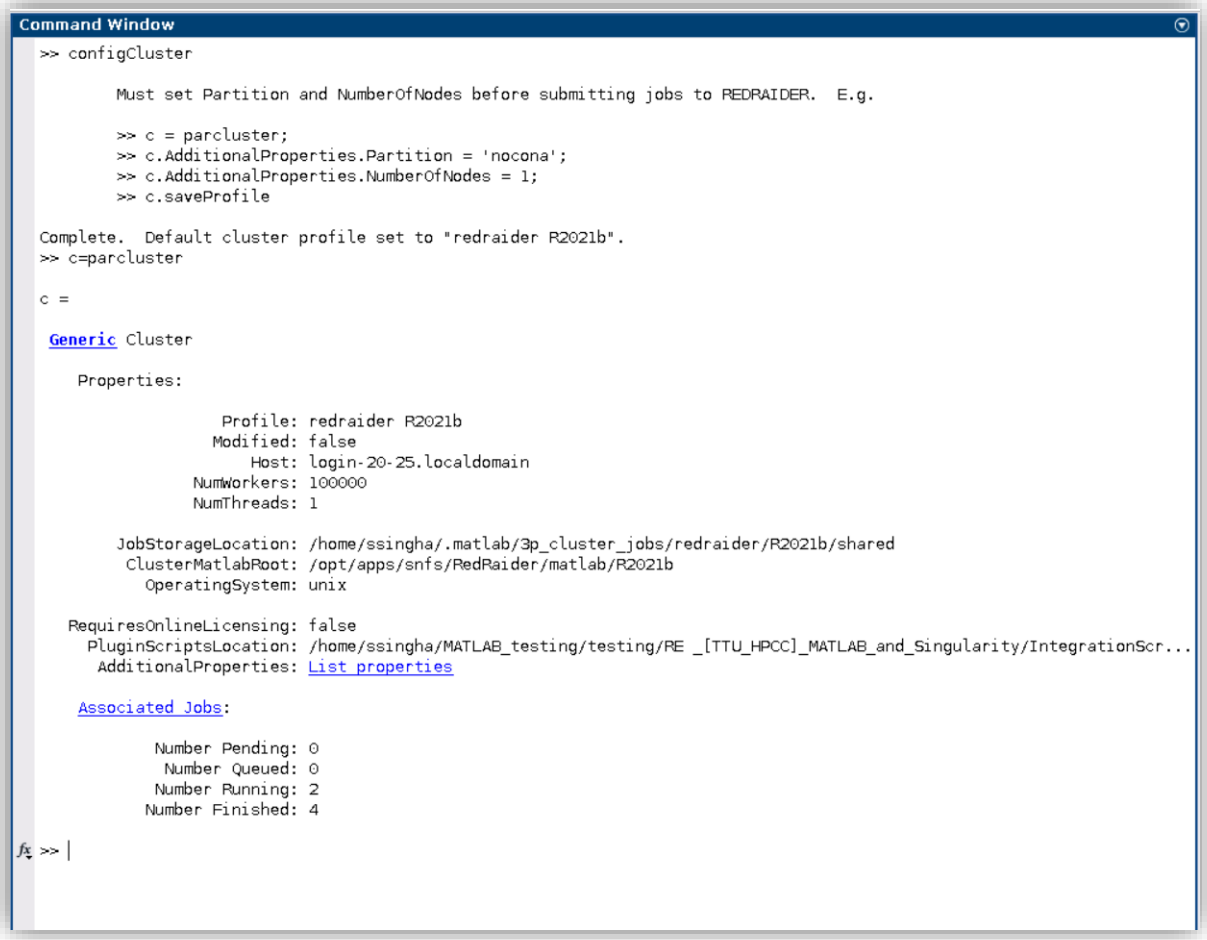
Partition:	Nocona	Quanah / XLQuanah	Matador	Toreador	Ivy
Type	CPU	CPU	GPU	GPU	Auxiliary CPU*
Total Nodes	240	467 / 16	20	11	50 / 2
Theoretical Max	983 TFLOPS	565 TFLOPS /19 TFLOPS	280 TFLOPS	287 TFLOPS	40 TFLOPS
Benchmarked	804 TFLOPS	485 TFLOPS / (N/A)	226 TFLOPS		N/A
OS	CentOS 8.1	CentOS 7.4 /CentOS 8.1	CentOS 8.1	CentOS 8.1	Rocky Linux 8.5 /CentOS 8.1
Manufacturer	Dell	Dell	Dell	Dell	Dell
Node Model	PowerEdge C6525	PowerEdge C6320	PowerEdge R740	Poweredge R7525	PowerEdge C6220 II
Cooling	Liquid Cooled	Air Cooled	Air Cooled	Air Cooled	Air Cooled
Processor Make and Model	AMD EPYC™ 7702	Intel Xeon E5-2695 v4	Intel Xeon Gold 6242	AMD EPYC™ 7262	Intel Xeon E5-2670v2
Family	Rome	Broadwell	Cascade Lake	Rome	Ivy Bridge
Cores/Processor	64	18	20	8	10
Cores/Node	128	36	40 cpu + 1280 tensor + 10,240 CUDA	16 cpu + 1,296 tensor + 20,736 CUDA	20
Total Cores In Partition	30,720	16,812 / 576	800 cpu + 25,600 tensor +	528 cpu + 14,256 tensor+	1,000 / 40

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

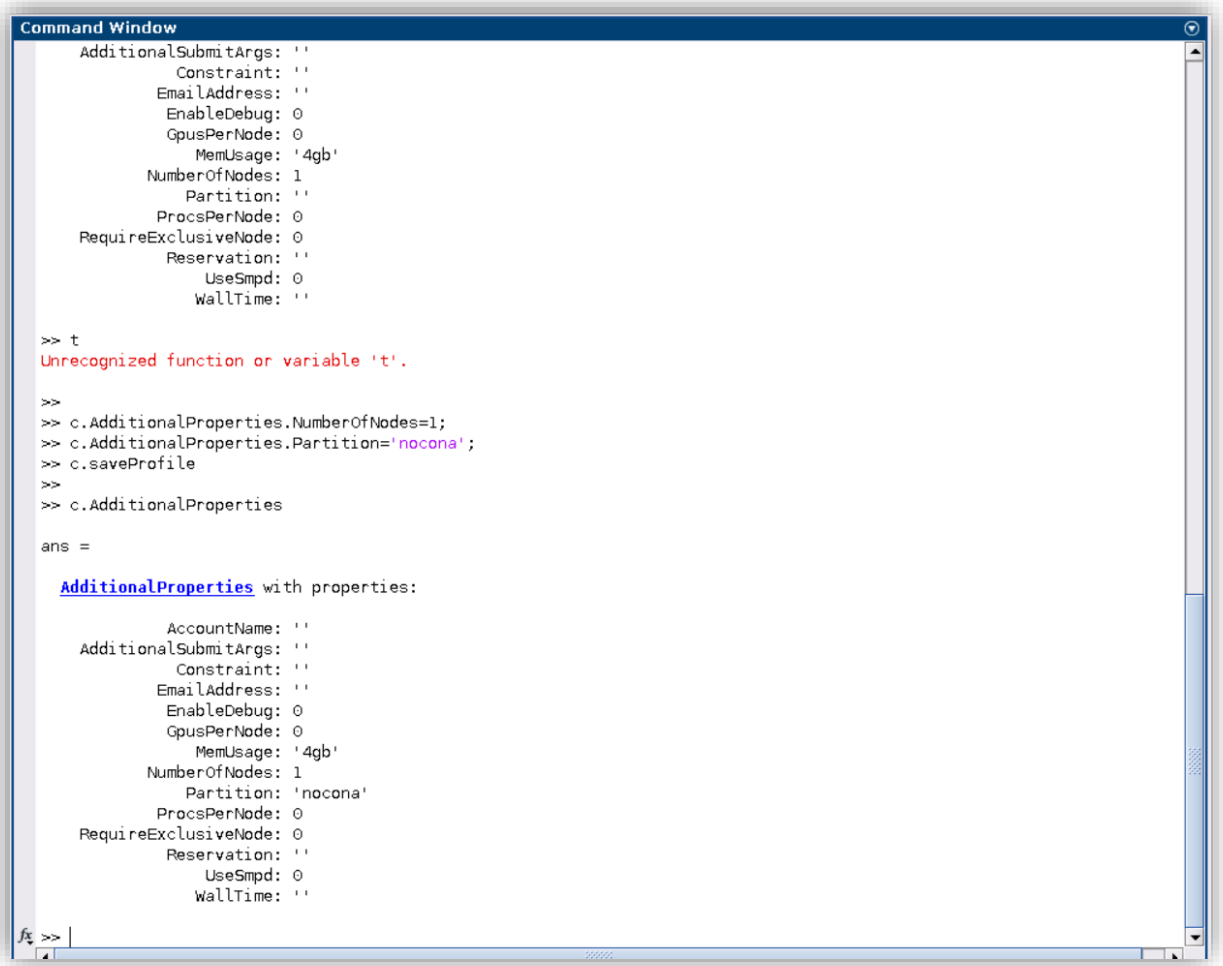


Job submission – Asynchronous Batch GUI

TEXAS TECH UNIVERSITY
Information Technology Division

Profile Properties

- Slurm properties that can be modified
- *AdditionalSubmitArgs* – Manually allows user to pass arguments, flags say constrains of the cluster
- c.AdditionalProperties.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

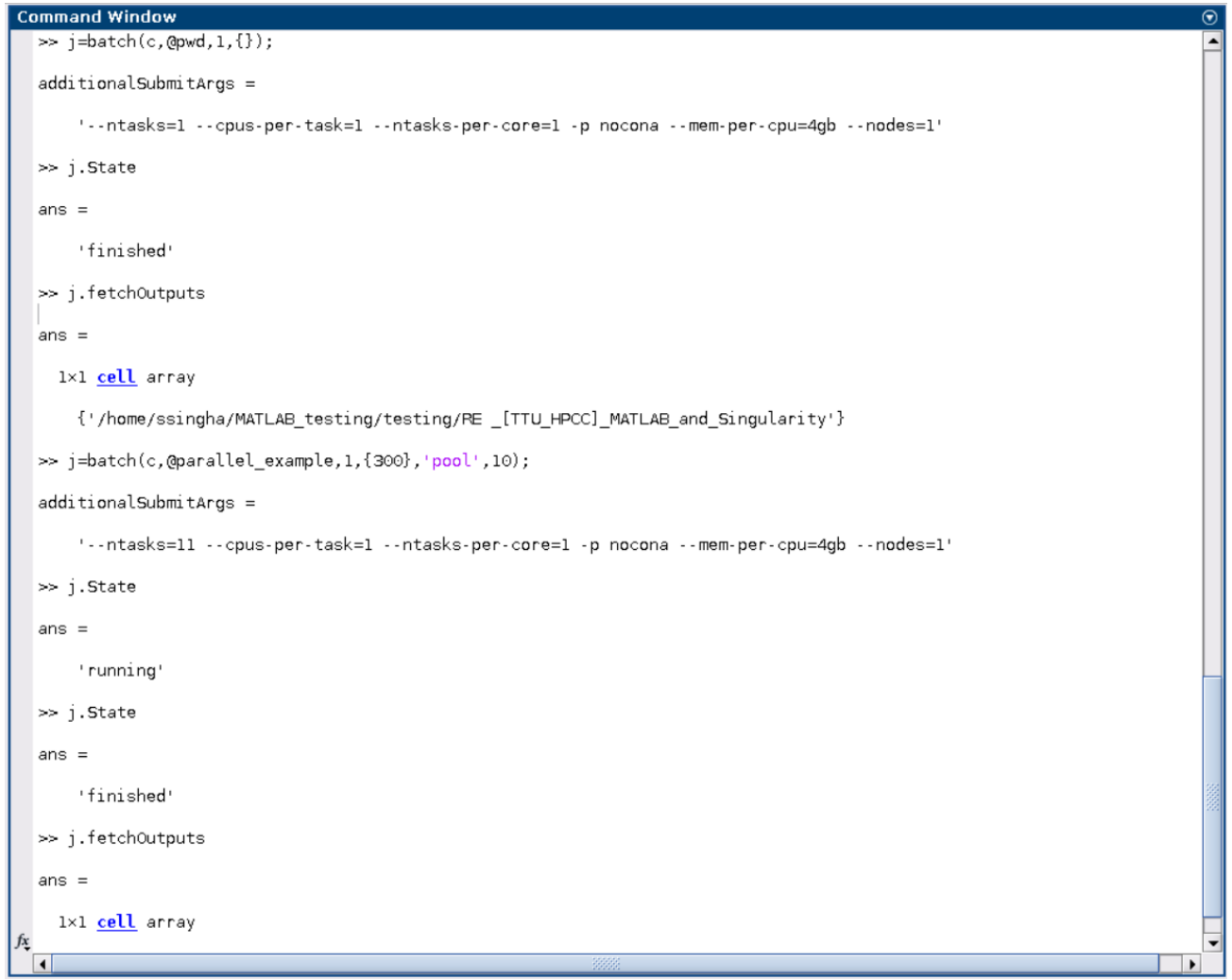
```
>>j=batch(c,@pwd,1,{});
```

Inputs to function

>>j.State – returns status of job

>>j.fetchOutputs – returns output of job

>>j.diary – returns command line outputs



Job submission – Asynchronous Batch GUI

TEXAS TECH UNIVERSITY
Information Technology Division

Serial Example

```
function [t, A] = serial_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]}
```

Job submission – Asynchronous Batch GUI

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
```

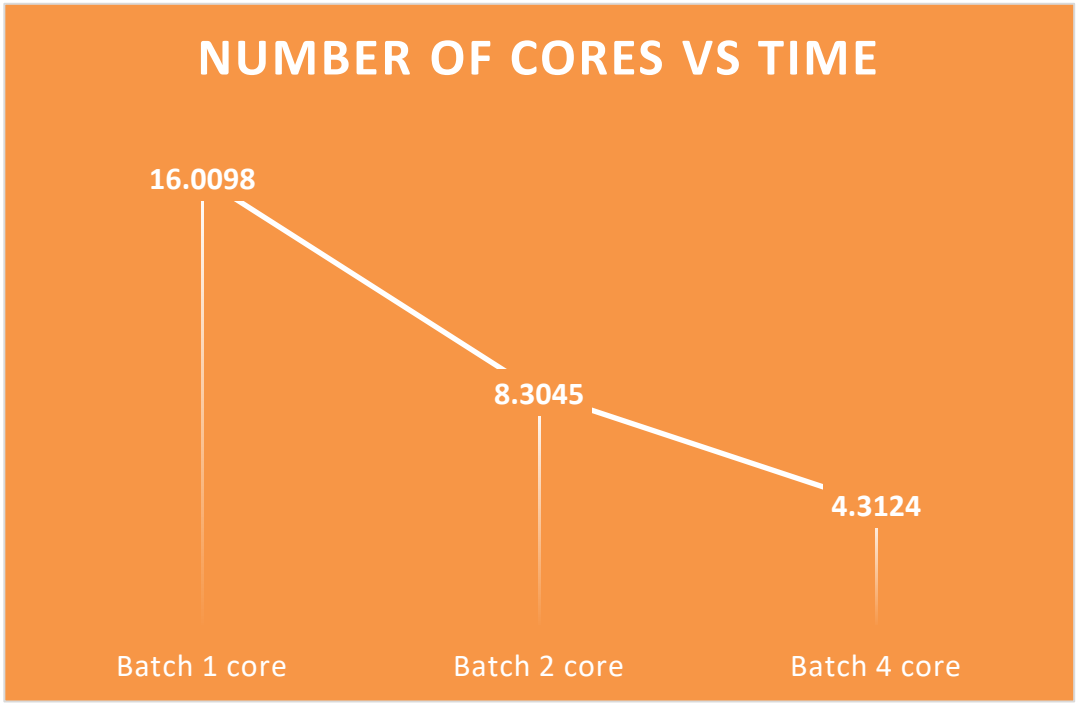
```
>> j=batch(c,@serial_example,1,{8});

{[16.0098]}

>>j=batch(c,@parallel_example,1,{8},'pool',2);
{[8.3045]}

>> =batch(c,@parallel_example,1,{8},'pool',4);
{[4.3124]}
```

Job submission – Asynchronous Batch GUI

TEXAS TECH UNIVERSITY
Information Technology Division

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
```



Job submission – Asynchronous Batch GUI

Job Arrays

```
>> 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);
```

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

additionalSubmitArgs =

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

>> j.Tasks

ans =

 3x1 Task array:

        ID       State              FinishDateTime  Function  Errors  Warnings  SchedulerID
    ------------------------------------------------------------------------------------------
    1    1      running                                 sum       0         0    7632475_1
    2    2      pending                                 sum       0         0    7632475_2
    3    3      pending                                 sum       0         0    7632475_3

>>
>>
>>
```

Job submission – Asynchronous Batch GUI

Job Arrays

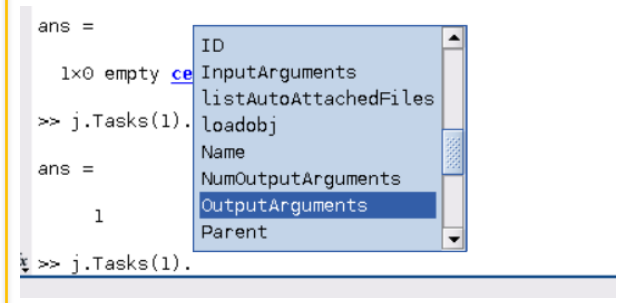
>> j.Tasks

```
>> 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
```

>> j.Tasks(2).cancel
>> j.Tasks(1).State

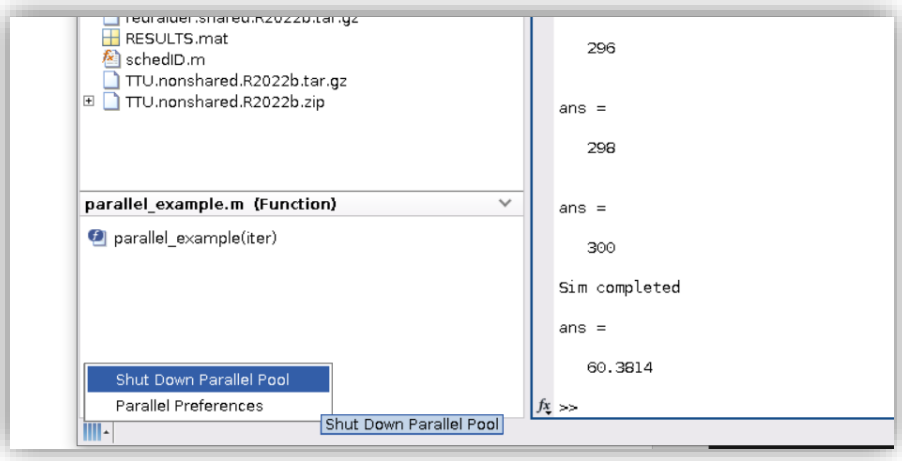
```
>> j.Tasks(2).cancel
>> j.Tasks

ans =

 3x1 Task array:

        ID       State              FinishDateTime  Function  Errors  Warnings
      ------------------------------------------------------------------------
    1    1      pending                                 sum       0         0
    2    2      finished       09-Nov-2022 00:11:14     sum       1         0
    3    3      pending                                 sum       0         0
```

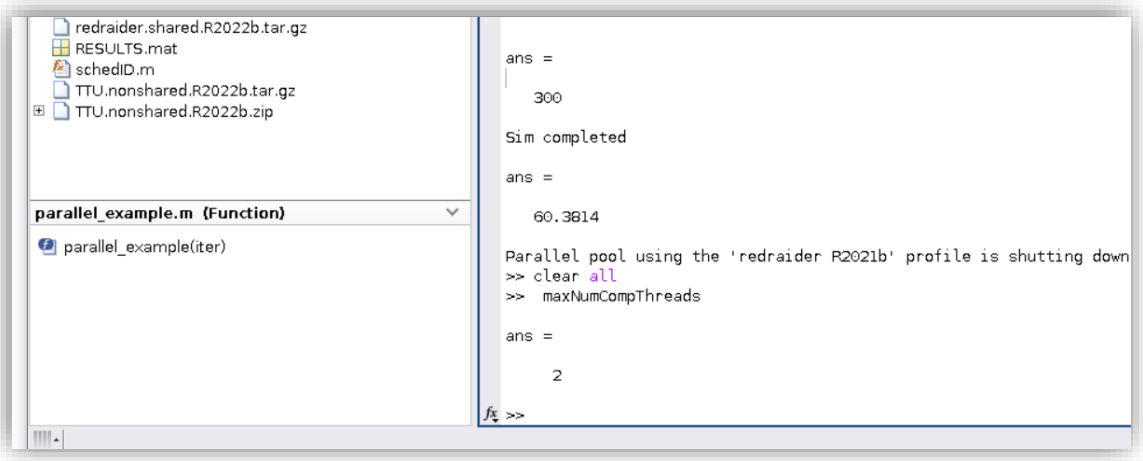
Monitoring job parameters



Job submission – Interactive pool submission

Shutting down Parallel pool



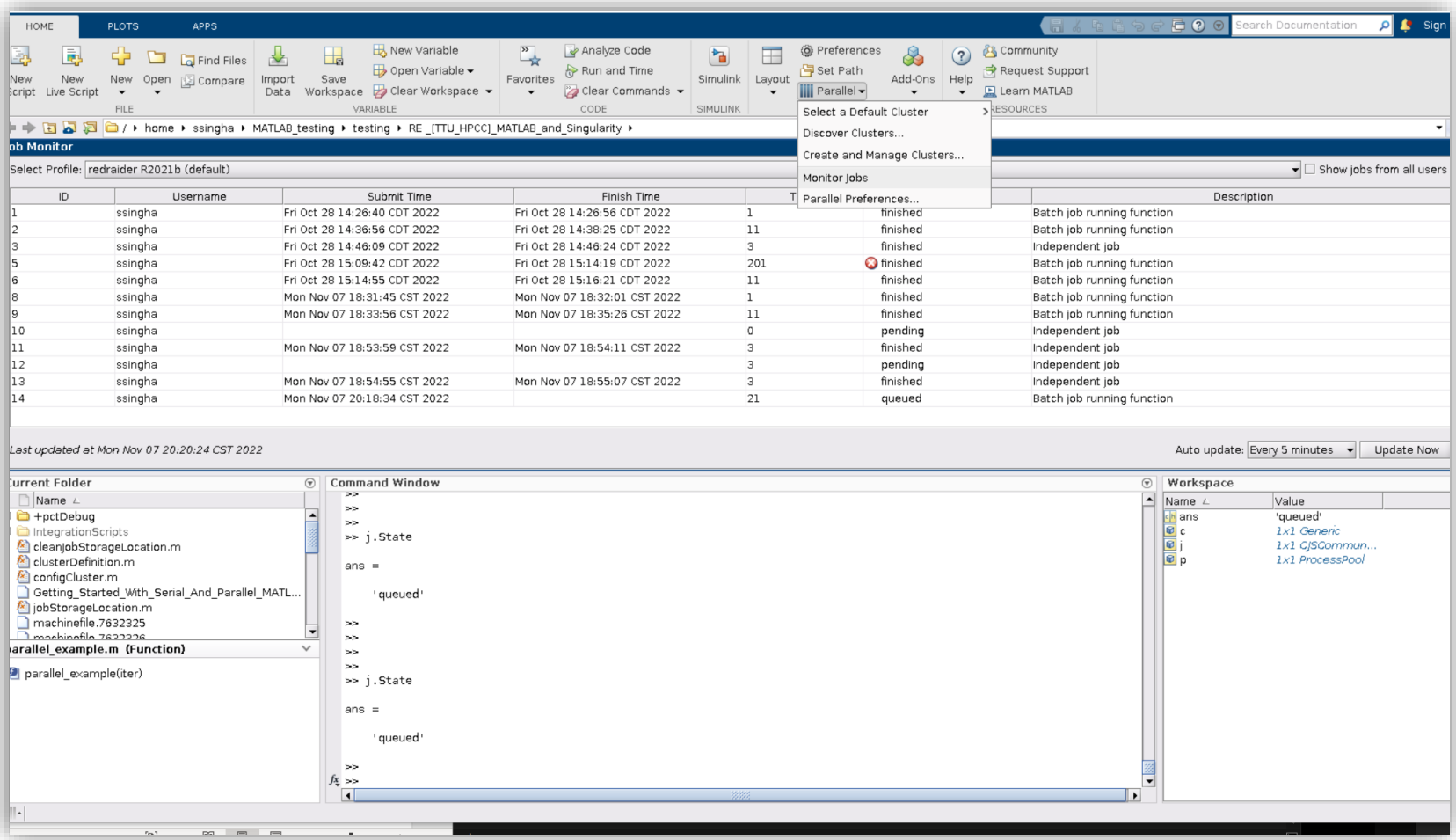


>>clear all

Job submission – Interactive pool submission

TEXAS TECH UNIVERSITY
Information Technology Division

Monitoring

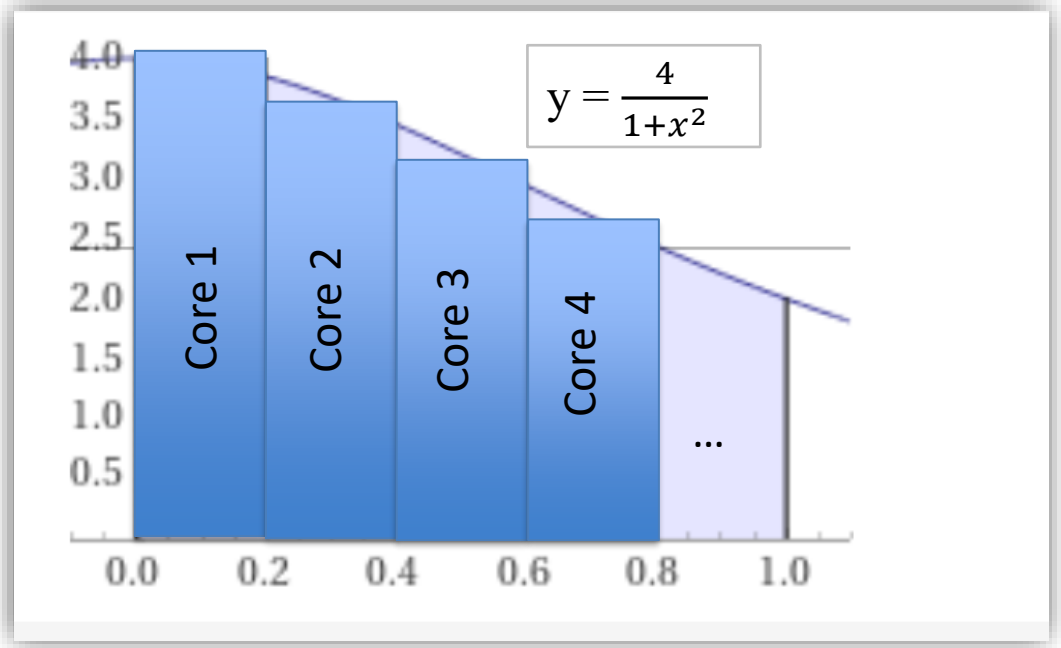


Parallel job implementation example

Calculation of π

(based on Leibniz formula)

$$\int_0^1 \frac{4}{1+x^2}\,dx = 4\arctan = \pi$$



Plotted in Wolfram Alpha

Parallel job implementation example

TEXAS TECH UNIVERSITY
Information Technology Division

Single node parallel

```
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           : %.18f\n', pi)
fprintf('Approximation: %.18f\n', approx1)
fprintf('Error        : %g\n',    abs(pi - approx1))

function y = quadpi(x)
%QUADPI Return data to approximate pi.

% Derivative of 4*atan(x)
y = 4./(1 + x.^2);
```

Multi node parallel

```
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           : %.18f\n', pi)
fprintf('Approximation: %.18f\n', approx1)
fprintf('Error        : %g\n',    abs(pi - approx1))


function y = quadpi(x)
%QUADPI Return data to approximate pi.

% Derivative of 4*atan(x)
y = 4./(1 + x.^2);
```

Parallel job implementation example

Accuracy of numerical calculation

```
>> 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
```

Lower

```
>> 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: [0.05, 0.075]

. . .

Worker 18:
  Subinterval: [0.425, 0.45]   Integral: 0.0839331
Worker 20:
  Subinterval: [0.525, 0.55]   Integral: 0.0775848
pi            : 3.141592653589793116
Approximation: 3.141592653589793116
Error         : 0
```

Lower

Containerized MATLAB

TEXAS TECH UNIVERSITY
Information Technology Division

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 1.4.3-1.el9
4. Pull a sample singularity container to get started
   $ singularity pull shub://singularityhub/hello-world

```
cpu-25-27:/singularity_testing/nocona$ singularity pull shub://singularityhub/hello-world
INFO:    Downloading shub image
59.8MiB / 59.8MiB [==========================================================================================================================================] 100 % 44.5 MiB/s 0s
cpu-25-27:/singularity_testing/nocona$ ls
hello-world_latest.sif
```

Containerized MATLAB

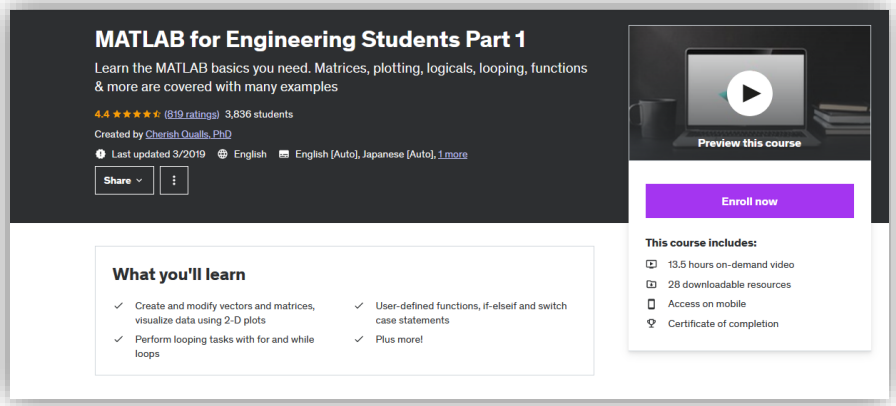
TEXAS TECH UNIVERSITY
Information Technology Division

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

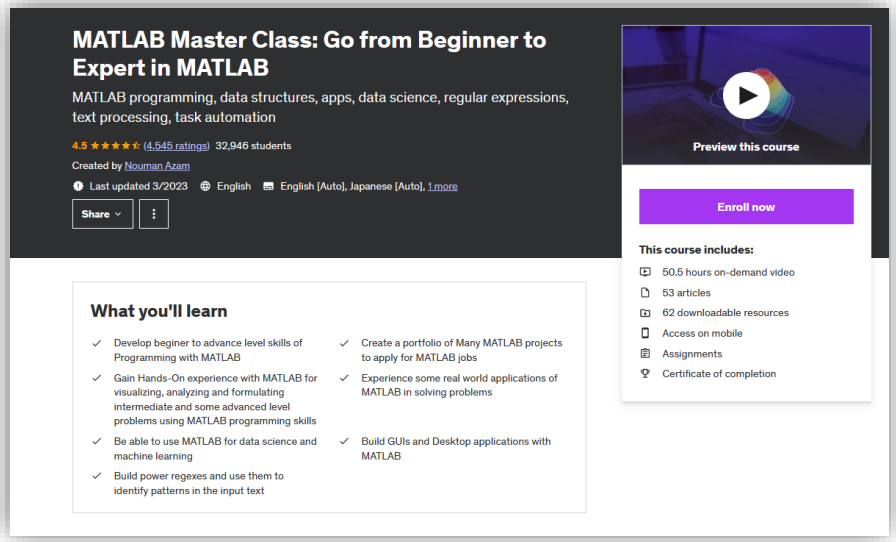
Additional Resources

TEXAS TECH UNIVERSITY
Information Technology Division

- Udemy –
  - https://ttu.udemy.com/course/matlab-the-need-to-know-basics/



  - https://ttu.udemy.com/course/matlab-essentials-for-engineering-and-science-students/



National Testbed Allocations - ACCESS

ACCESS Resources

CPU Computing

Anvil (Purdue)—1,000 AMD Milan nodes, 128 cores per node, large memory nodes available
Bridges-2 (PSC)—504 AMD Rome nodes, 128 cores per node, large memory nodes available; extreme memory (4 TB) nodes allocated separately
Delta (UIUC)—124 AMD Milan nodes, 128 cores per node
Expanse (SDSC)—728 AMD Rome nodes, 128 cores & 1 TB NVMe per node
KyRIC (U Kentucky)—Large-memory (3 TB, 6 TB) nodes, 300TB storage
Stampede3 (TACC) Intel Skylake, Ice Lake, Sapphire Rapids w/HBM H100 nodes

GPU Computing

Anvil GPU (Purdue)—16 nodes, 4 NVIDIA A100 GPUs each
Bridges-2 GPU (PSC) —10 nodes w/ 8x NVIDIA H100 GPUs, 24 nodes w/ 8x V100-32GB GPUs, 9 nodes w/ 8x NVIDIA V100 16GB GPUs
DeltaAI(UIUC) –152 nodes w/ 4x NVIDIA GH200-96GB
Delta GPU (NCSA)—4 node configs: 100 nodes w/ 4x A100s; 100 w/ 4x A40 GPUs; 5 w/ 8x A100s; 1 w/ 8x AMD MI100 GPUs
Expanse GPU (SDSC)—52 nodes, 4 NVIDIA V100 GPUs each

National Testbed Allocations - ACCESS

Allocation Types

- **Code characteristics**: Core-hours, Node-hours, GPU-hours
- *Credits*
- Types

➢ **Explore** 400, 000 credits
  - Getting started, evaluating resources, dissertations, small-scale activities.
  - Grad students with a letter from their PI.

➢ **Discover** 1.5 million credits
  - 1-page document with science and computational description,
  - In house review 1 day-1 week
  - Grad students with a letter from their PI.

➢ **Accelerate** 3 million credits
  - 3 pg. doc science and computational methods, research plan, scaling, research team
  - External reviewers few days - few weeks

➢ **Maximize** 10 million credits
  - 10-page proposal, larger panel of external reviewers twice a year, 100 proposals /year of this scale
  - Detailed doc of code runs, scaling, assure reviewers of funding for using resources!

Note: They try to help. Emphasis on folks who have done good scaling runs, have run on ACCESS machines at a smaller scale and now are sure that they require larger resource pool.

National Testbed Allocations - DOE

TEXAS TECH UNIVERSITY
Information Technology Division

DOE Resources

CPU and GPU Computing

Argonne LCF

- **Polaris** - 560 - 1 x AMD Milan CPU + 4x NVIDIA A100 GPU
  560 CPUs + 2,240 GPUs
- **Aurora** - 10,624 - 2 x Intel Xeon SPR +6 x Intel PVC GPU
  21,248 CPUs + 63,744 GPUs

Oak Ridge LCF

- **Frontier** - 9,408 - 1 x AMD EYPC CPU + 4 x AMD MI250x GPU
  9,408 CPUs + 37,362 GPUs

NERSC

- **Perlmutter** -
  - 3,072 CPU only nodes - 2 AMD Milan per node
  - 1,792 GPU Accelerated Nodes with 4 NVIDIA A100 + 1 AMD Milan per node.

  6144 CPUs and 1792 CPUs + 7,168 GPUs

National Testbed Allocations - DOE

Allocation Types

- **Code characteristics**: Core-hours, Node-hours, GPU-hours
- ALCF (and OLCF)
  - **Director's Discretionary** - Relatively new to High-Performance Computing (HPC).
  - **INCITE** –Large-scale, computationally intensive projects that address "*grand challenges*" in science and engineering. **Cycle**: Mid-April and mid to late June. **Review**: Two-part peer review by international experts and a computational-readiness review. **Time**: Awarded for one to three years.
  - **ALCC** –High-risk, high-payoff scientific campaigns in areas directly related to the DOE mission e.g., Biophysics, Energy Efficiency in Aerospace and Combustion technologies, Fusion Energy, Geosciences, High Energy Physics, Materials Sciences, Nuclear Engineering, and Nuclear Physics. Current awards. **Cycle**: July 1 – June 30th of the subsequent year. **Review**: Like INCITE. **Time**: Possibility of renewal for up to 2 years.

- NERSC
  - **Education, Exploratory**: Small allocations (~250 CPU Node Hours) to support research projects to explore using HPC and for educational efforts to train the next-generation workforce. For first-time PIs to try out NERSC resources. Decision within 4 weeks of applying (pending DOE approval for Exploratory) valid till 18 months from award.

National Testbed Allocations - NAIRR

National Artificial Intelligence Research Resource Pilot

- **Aim**: Advancing AI methods that enable scientific discovery, accelerating through AI enabled automation, new experimental methods, integrating simulations and AI, developing open-source AI tools, models, datasets, and methods, training AI-savvy workforce.
- Start-Up projects are awarded for three months, May 16, 2025 through the end of the NAIRR Pilot program.
- Resources
- Instructions for writing a proposal

Thoughts ?

Feedback

- Course assessment survey.
- Your feedback is incredibly valuable, it helps us enhance your learning experience.
- Your insights guide us in making our training course even better.