MathWorks®

Workshop: Parallel Computing with MATLAB (Part II)

SNIC

Raymond Norris
Application Engineer, MathWorks
June 7, 2021

© 2021 The MathWorks, Inc.

Agenda

- Part I – Parallel Computing with MATLAB on the Desktop
  - Parallel Computing Toolbox
  - ThinLinc
- Part II – Scaling MATLAB to Aurora
  - MATLAB Parallel Server
  - ThinLinc

https://lunarc-documentation.readthedocs.io/en/latest/MATLAB

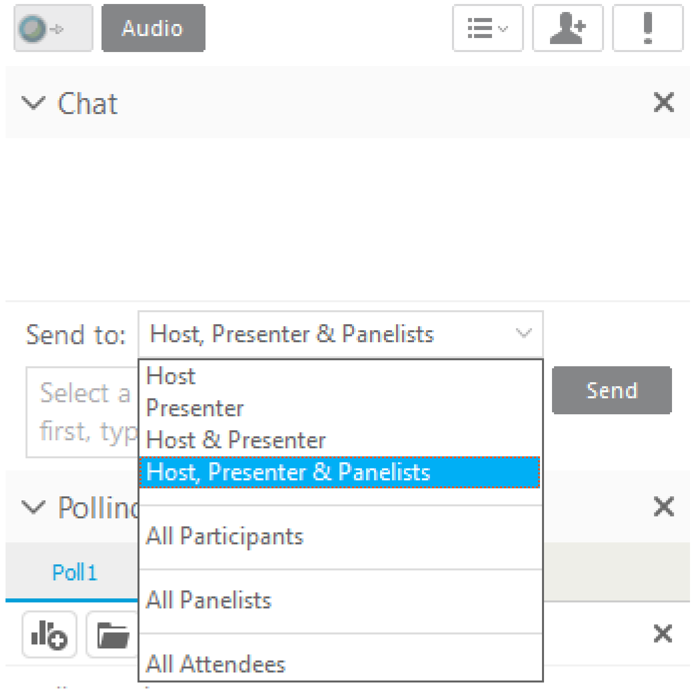
Agenda

- Part I – Parallel Computing with MATLAB on the Desktop
  - Parallel Computing Toolbox
  - ThinLinc
- Part II – Scaling MATLAB to Aurora
  - MATLAB Parallel Server
  - ThinLinc

https://lunarc-documentation.readthedocs.io/en/latest/MATLAB

Chatting

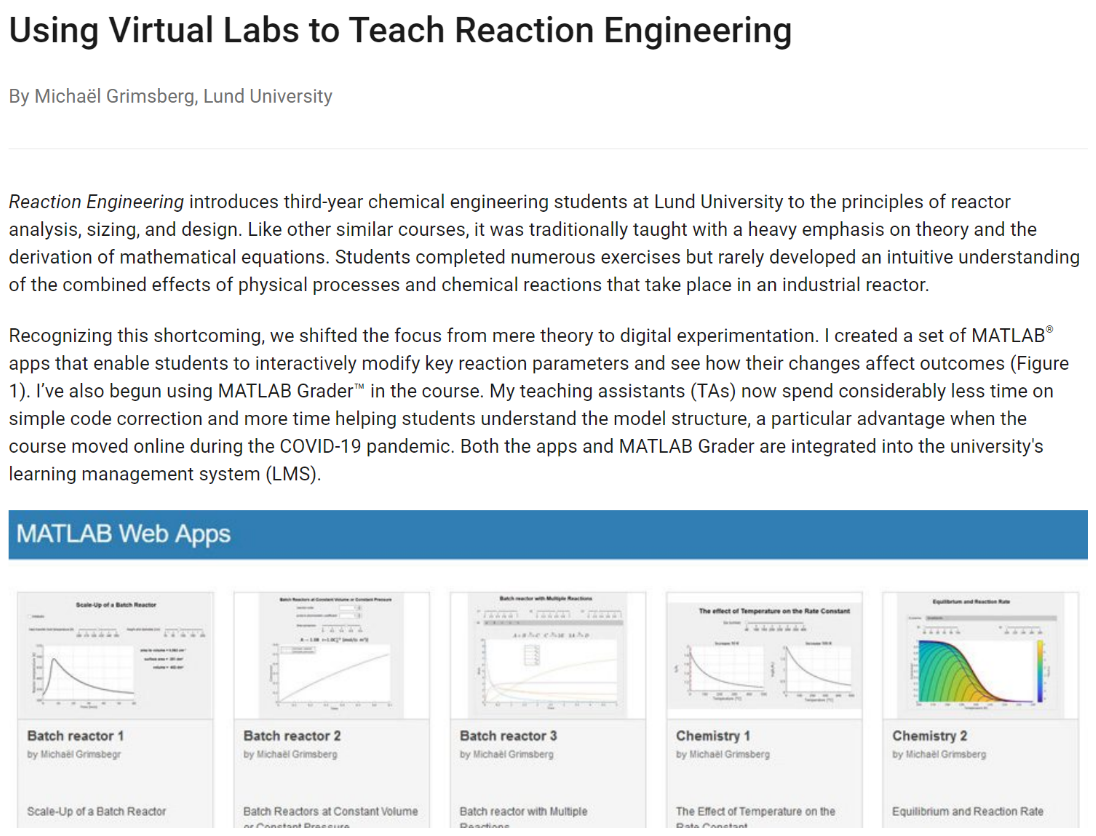
- Send to at least the *Host, Presenter & Panelists*
- Ideally, send to *All Attendees*



Send to: Host, Presenter & Panelists

- Host
- Presenter
- Host & Presenter
- Host, Presenter & Panelists
- All Participants
- All Panelists
- All Attendees

Using Virtual Labs to Teach Reaction Engineering



Using Virtual Labs to Teach Reaction Engineering

By Michaël Grimsberg, Lund University

Reaction Engineering introduces third-year chemical engineering students at Lund University to the principles of reactor analysis, sizing, and design. Like other similar courses, it was traditionally taught with a heavy emphasis on theory and the derivation of mathematical equations. Students completed numerous exercises but rarely developed an intuitive understanding of the combined effects of physical processes and chemical reactions that take place in an industrial reactor.

Recognizing this shortcoming, we shifted the focus from mere theory to digital experimentation. I created a set of MATLAB® apps that enable students to interactively modify key reaction parameters and see how their changes affect outcomes (Figure 1). I've also begun using MATLAB Grader™ in the course. My teaching assistants (TAs) now spend considerably less time on simple code correction and more time helping students understand the model structure, a particular advantage when the course moved online during the COVID-19 pandemic. Both the apps and MATLAB Grader are integrated into the university's learning management system (LMS).

MATLAB Web Apps

Batch reactor 1	Batch reactor 2	Batch reactor 3	Chemistry 1	Chemistry 2
by Michaël Grimsbegr	by Michaël Grimsberg	by Michaël Grimsberg	by Michaël Grimsberg	by Michaël Grimsberg
Scale-Up of a Batch Reactor	Batch Reactors at Constant Volume or Constant Pressure	Batch reactor with Multiple Reactions	The Effect of Temperature on the Rate Constant	Equilibrium and Reaction Rate

Michaël Grimsberg

Department of Chemical Engineering

Lecturer

Directory administrator

Email:
michael.grimsberg@chemeng.lth.se

AFFILIATIONS
Department of Chemical Engineering

MATLAB / MATLAB Web App Server / MATLAB Grader

Lund University Uses Virtual Labs to Teach Reaction Engineering

Challenge

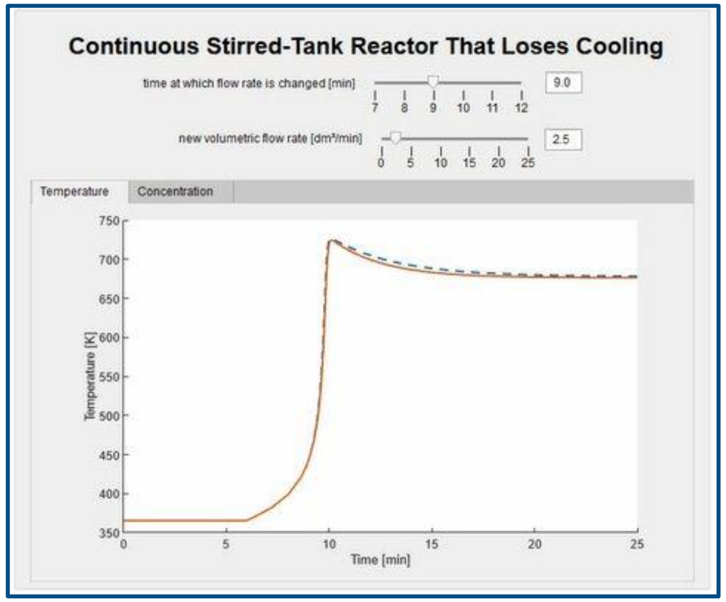
Teach the principles of industrial reactor analysis, sizing, and design by shifting the focus from theory to digital experimentation

Solution

Create MATLAB apps that enable students to interactively modify reaction parameters and see how their changes affect outcomes

Results

- Student understanding of key reactor processes deepened
- Engaging web apps integrated with university LMS
- Virtual lab assignments automatically graded



Continuous Stirred-Tank Reactor That Loses Cooling
time at which flow rate is changed [min]
9.0
new volumetric flow rate [dm³/min]
2.5
Temperature
Concentration
Temperature [K]
Time [min]

MATLAB app for visualizing the temperature of a reactor with a malfunctioning cooling system.

"With MATLAB Grader, we can automatically check the work of all 100 students and provide real-time feedback. TAs can then give their attention to students who need extra help with their code."

- Michaël Grimsberg, Lund University

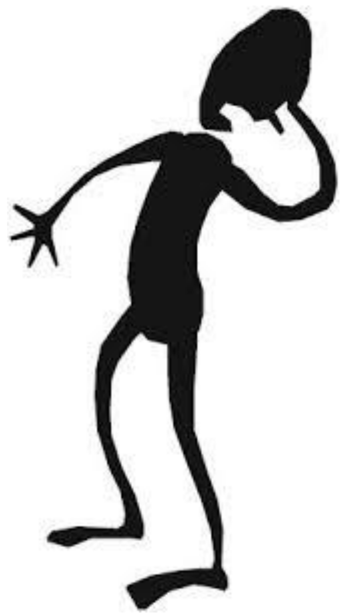
Link to article

Scaling MATLAB to Aurora

- Accessing and running MATLAB on local HPC clusters
- Running parallel and multi-node MATLAB jobs

A note about today's workshop…

- The workflow and examples are about process, not performance



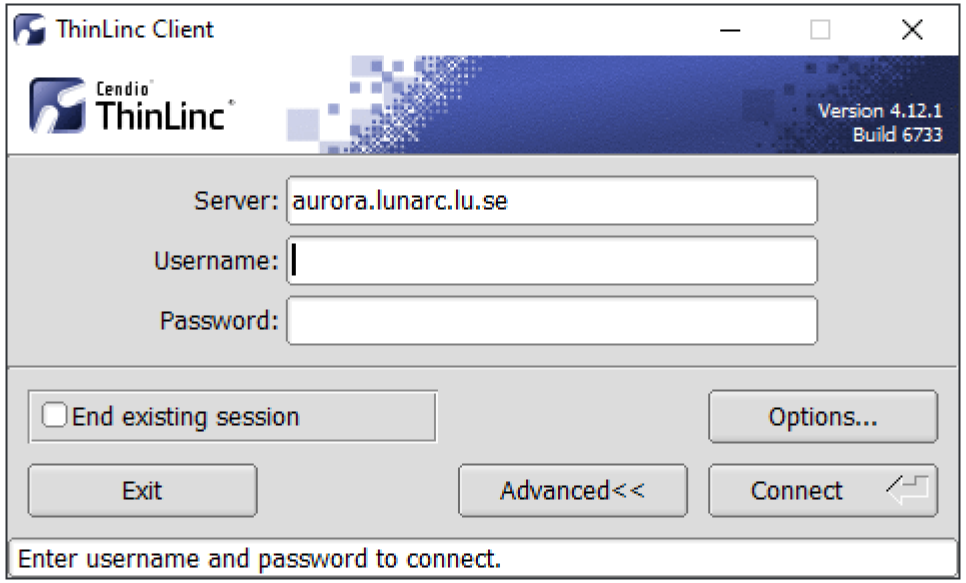
Accessing and running MATLAB on local HPC clusters (1)

- Two options
  - ssh xterm
    - Useful for either low-bandwidth or automation
  - ThinLinc
    - Graphical interface

Accessing and running MATLAB on local HPC clusters (2)

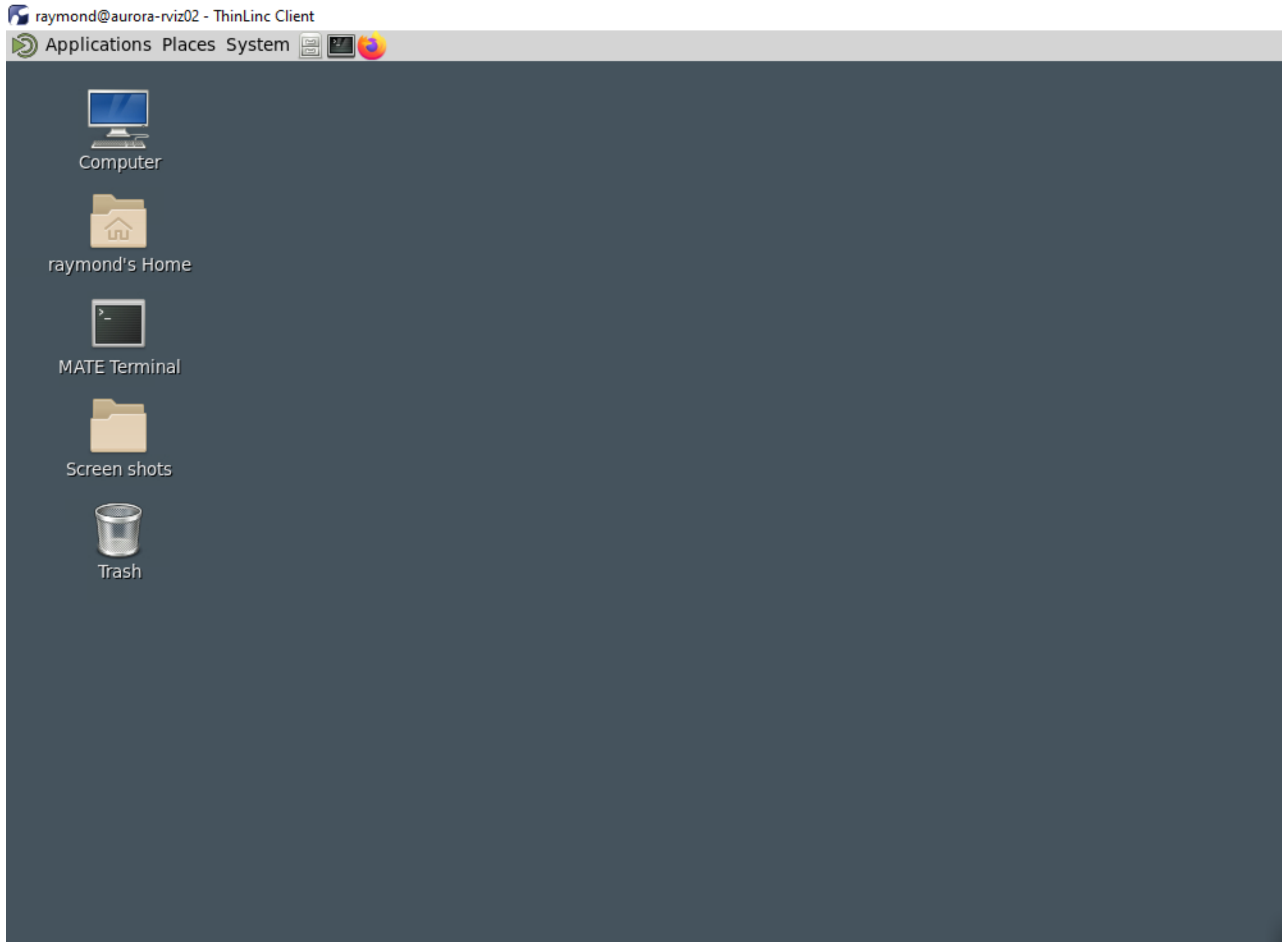
- ThinLinc
  - Linux Remote Desktop Server
  - Easy to use and simple to learn
  - Good way to run a GUI application remotely on the cluster login nodes
- Access
  - Client: https://www.cendio.com/thinlinc/download
- Local resources
  - Overview: https://lunarc-documentation.readthedocs.io/en/latest/using_hpc_desktop

ThinLinc client



Server: aurora.lunarc.lu.se
Username:
Password:

Enter username and password to connect.


raymond@aurora-rviz02 - ThinLinc Client
Applications
Places
System
Computer
raymond's Home
MATE Terminal
Screen shots
Trash

Download workshop files

```
-bash4.2 # Make a local copy of the Workshop files (Part II)
-bash4.2 mkdir -p ~/Documents/MATLAB
-bash4.2 cp -frp /lunarc/nobackup/projects/matlab_mondays/matlab-workshop ~/Documents/MATLAB
```

Ways to run MATLAB

- Interactively
  - with parallel pool, synchronously (`parpool`)
  - with batch jobs, asynchronously (`batch`)
- Noninteractive
  - with Slurm job script (`sbatch`)

MATLAB job submitters

- `parpool`
  - Single session
  - Synchronous execution
  - Seamlessly runs `parfor`, `parfeval`, and `spmd`

https://www.mathworks.com/help/parallel-computing/parpool.html

- `batch`
  - Multiple submissions
  - Non-blocking
  - Calls top-level function or script
  - Requires API to extract results

https://www.mathworks.com/help/parallel-computing/batch.html

Interactively: with parallel pool, synchronously

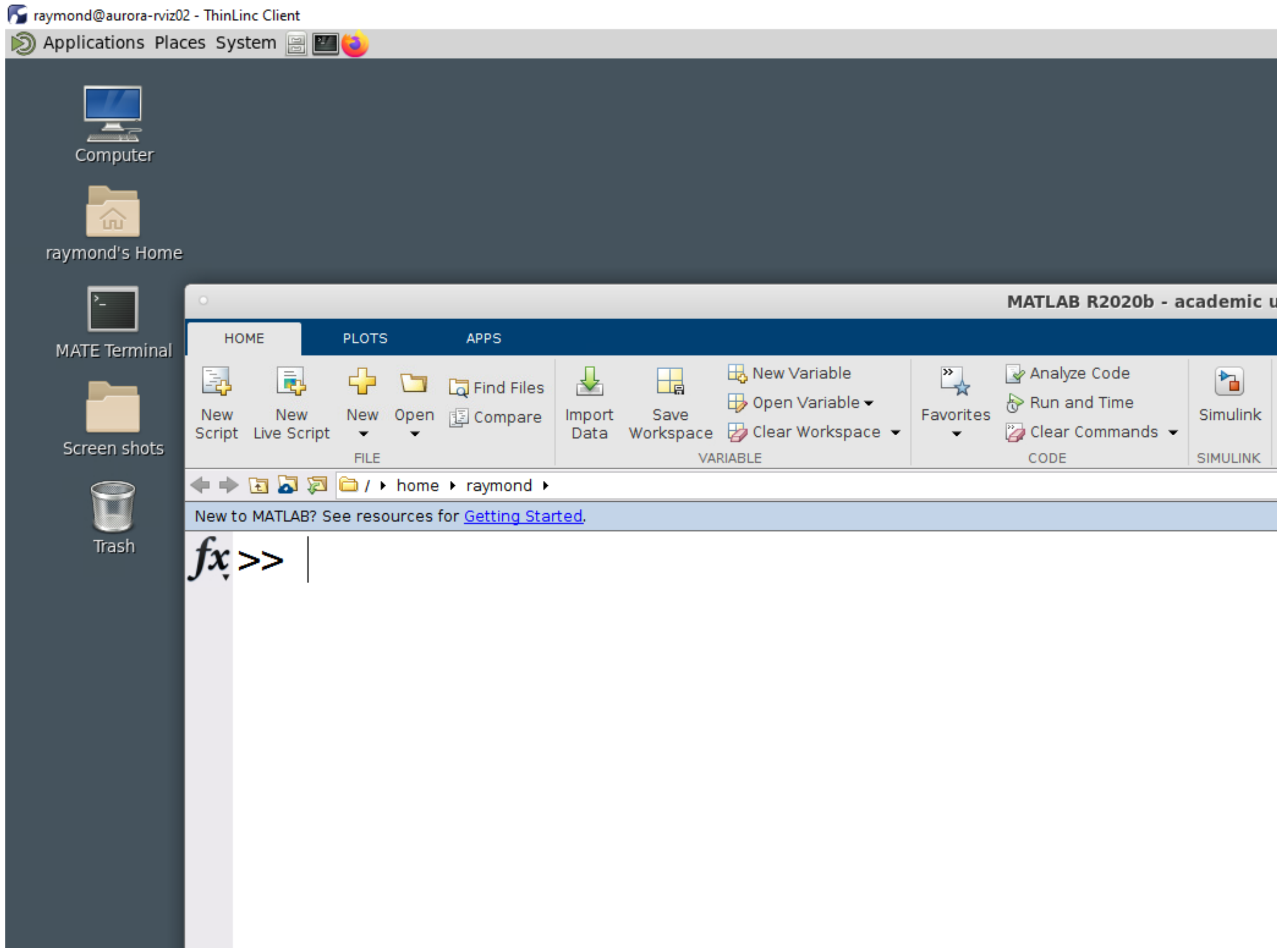
`parpool`

Accessing MATLAB

- Shared resource
  - LUNARC Applications > MATLAB > MATLAB <VERSION>
  - LUNARC Applications > MATLAB > MATLAB <VERSION> - Hardware OpenGL Acceleration
- On-Demand
  - Lunar Applications On-Demand > MATLAB > MATLAB <VERSION>

	Cores	GPU	maxNumCompThreads	Nodes	Scheduled
SR - MATLAB	20	None	1	1	✗
SR – MATLAB w/ OpenGL	20	k80	20	1	✗
OD - MATLAB	16	k20m	16	24	✓

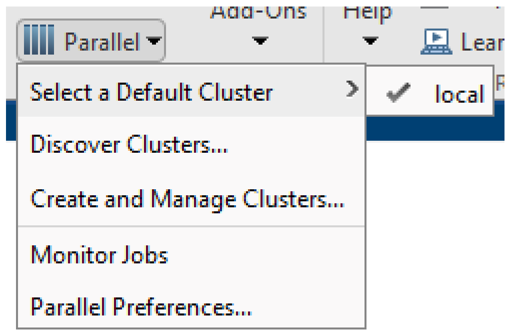
Starting MATLAB



New to MATLAB? See resources for Getting Started.

```
>>
```

Parallel MATLAB – Single Node

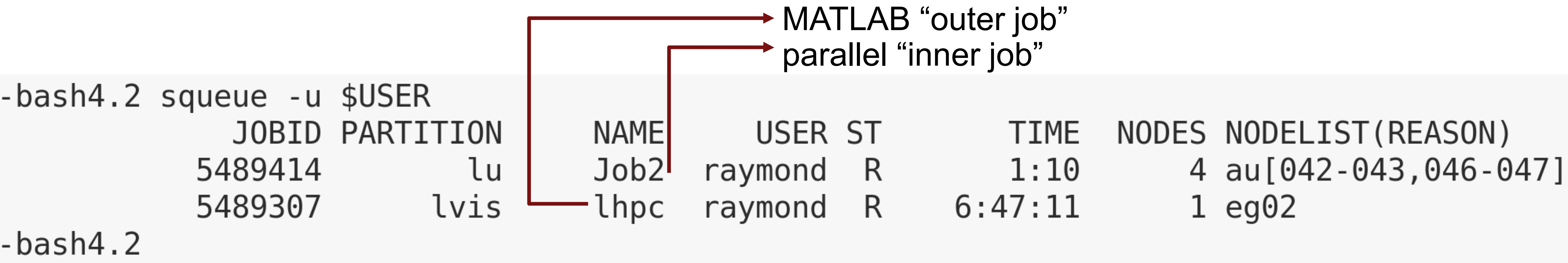


```
>> maxNumCompThreads
ans =
    16
>>
>> p = parpool('local',16);
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 16).
>>
>> tic, parfor idx = 1:320, pause(3), end, toc
Elapsed time is 60.449216 seconds.
>>
```

Parallel MATLAB – Multi-node (1)

- In order to run a multi-node MATLAB job, MATLAB will generate and submit a new Slurm job
  - Executed during any “job launcher”
    - `parpool*`, `batch`, `createJob`
  - Run asynchronously while MATLAB session is running, except `parpool`
  - True regardless if we’re running MATLAB desktop or a Slurm job script

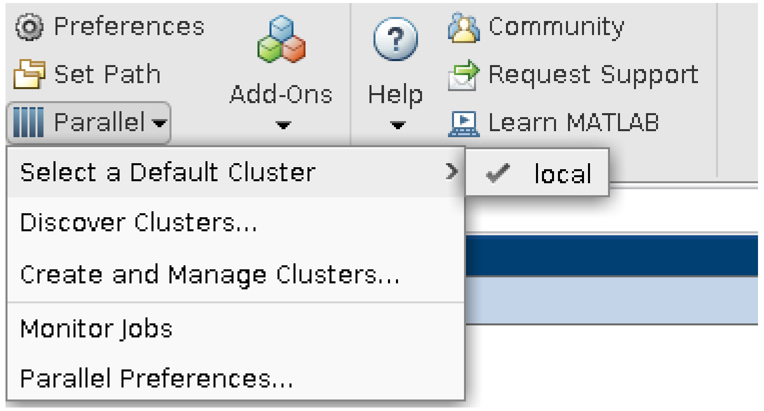
- Need to generate a new profile for Aurora
  - `configCluster`



```
-bash4.2 squeue -u $USER
               JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             5489414        lu     Job2  raymond  R       1:10      4 au[042-043,046-047]
             5489307      lvis     lhpc  raymond  R    6:47:11      1 eg02
-bash4.2
```

local profile

“How does MATLAB know about Aurora?”



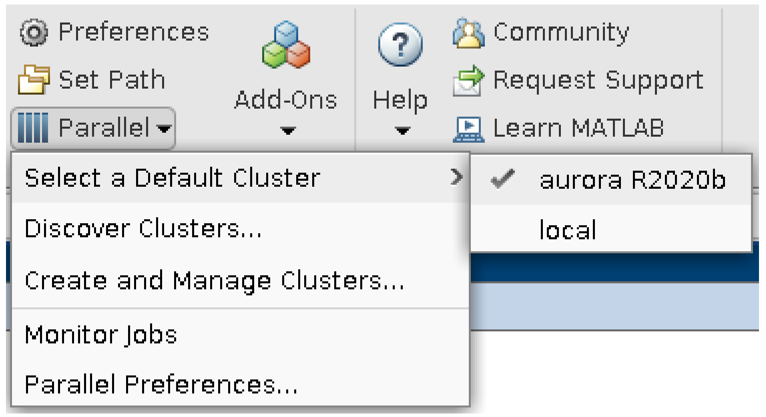
Configure MATLAB to create Aurora profile

```
>> configCluster

        Must set AccountName and WallTime before submitting jobs to AURORA.  E.g.

        >> c = parcluster;
        >> c.AdditionalProperties.AccountName = 'account-name';
        >> % 5 hour walltime
        >> c.AdditionalProperties.WallTime = '05:00:00';
        >> c.saveProfile

>> c = parcluster;
>> c.AdditionalProperties.AccountName = 'lu2021-2-80';
>> c.AdditionalProperties.WallTime = '00:10:00';
>> c.saveProfile
```

Minimum requirements

Call `projinfo` to get listing of accounts

New Aurora profile



Only call `configCluster` once

Workshop: Use reservation just for today

```
>> c.AdditionalProperties.Reservation = 'matlabmonday';
```

Parallel MATLAB – Multi-node (2)

```
>> p = parpool('local',16);
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 16).
>>
>> tic, parfor idx = 1:320, pause(3), end, toc
Elapsed time is 60.449216 seconds.
```

```
>> % Get handle to HPC cluster
>> c = parcluster;
>>
>> % Start multi-node parallel pool
>> p = c.parpool(40);
Starting parallel pool (parpool) using the 'aurora R2020b' profile ...
additionalSubmitArgs =
    '--ntasks=40 --cpus-per-task=1 -N 2 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00
Connected to the parallel pool (number of workers: 40).
>>
>> tic, parfor idx = 1:800, pause(3), end, toc
Elapsed time is 60.413123 seconds.
>>
```

“2.5x more workers, but the same time?”

How big of a Pool? . . .

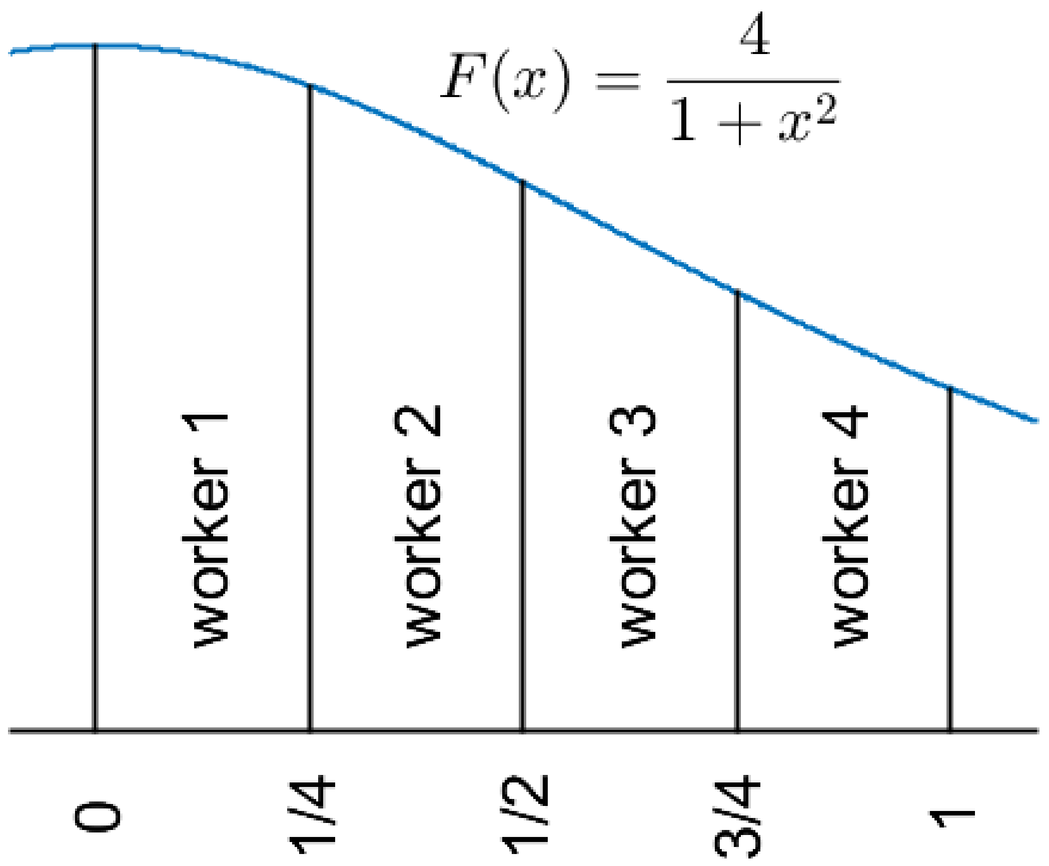
```
>> % Pool of 460 workers across 23 nodes
>> tic, p = c.parpool(23*20); toc
Starting parallel pool (parpool) using the 'aurora R2020b' profile ...

additionalSubmitArgs =

    '--ntasks=460 --cpus-per-task=1 -N 23 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00 --exclusive'

Connected to the parallel pool (number of workers: 460).
Elapsed time is 99.328334 seconds.
>>
>> tic, parfor idx = 1:9200, pause(3), end, toc
Elapsed time is 61.536966 seconds.
>>
>> % Equivalent hours, if run serially
>> 9200 * 3 / 60 / 60

ans =

    7.6667

>>
```

Change directories to workshop

```
>> cd(fullfile(userpath,'matlab-workshop'))
```

Exercise: Calculate π

$$\int_0^1 \frac{4}{1+x^2}\,dx = 4(atan(1) - atan(0)) = \pi$$

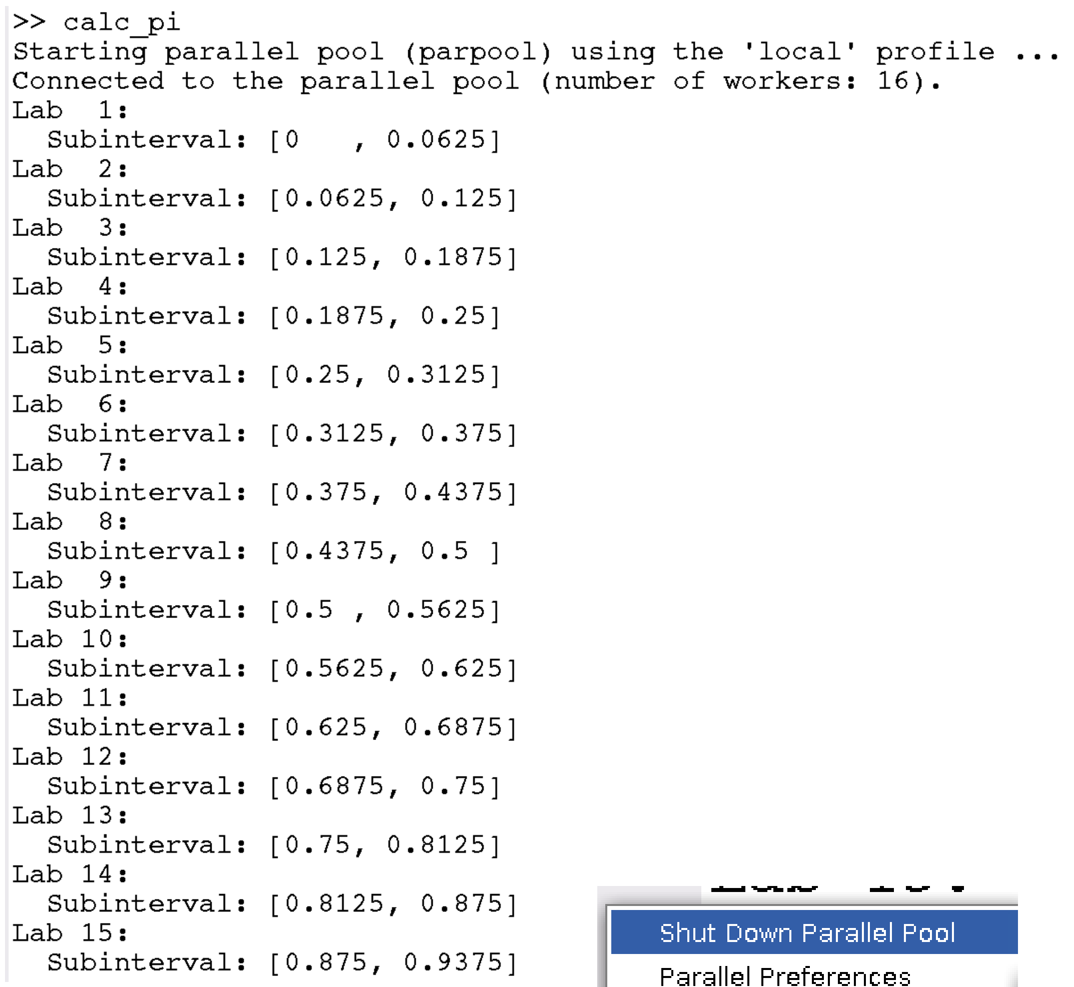


Calculate π

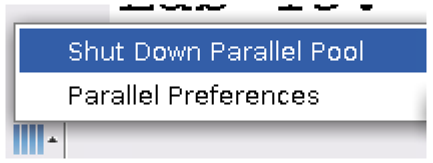
```
function calc_pi

c = parcluster('local');

% Query for available cores (assume either Slurm or PBS)
sz = str2num([getenv('SLURM_CPUS_PER_TASK') getenv('PBS_NP')]); %#ok<ST2NM>
if isempty(sz), sz = maxNumCompThreads; end

c.parpool(sz);

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);
```

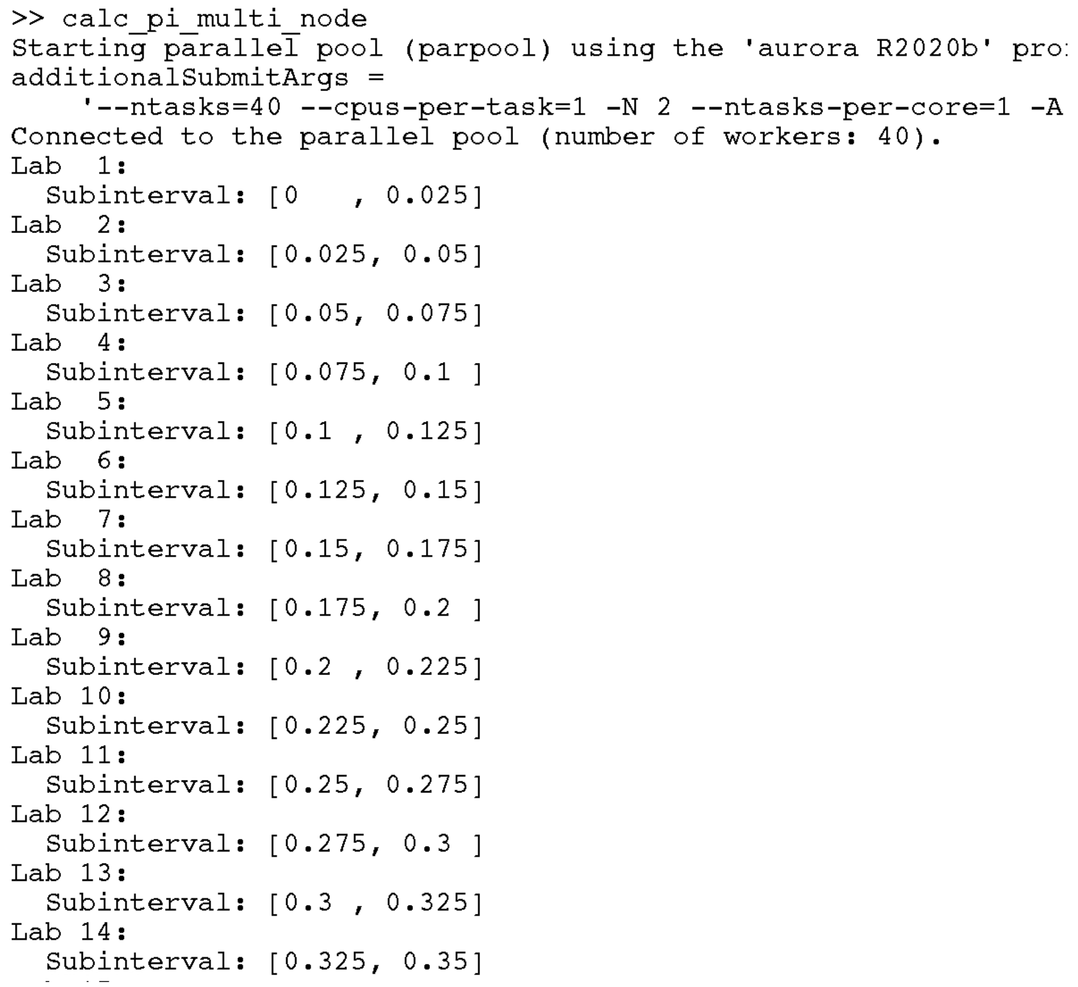
```
function calc_pi_multi_node

c = parcluster;




c.parpool(40);

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);
```

Results



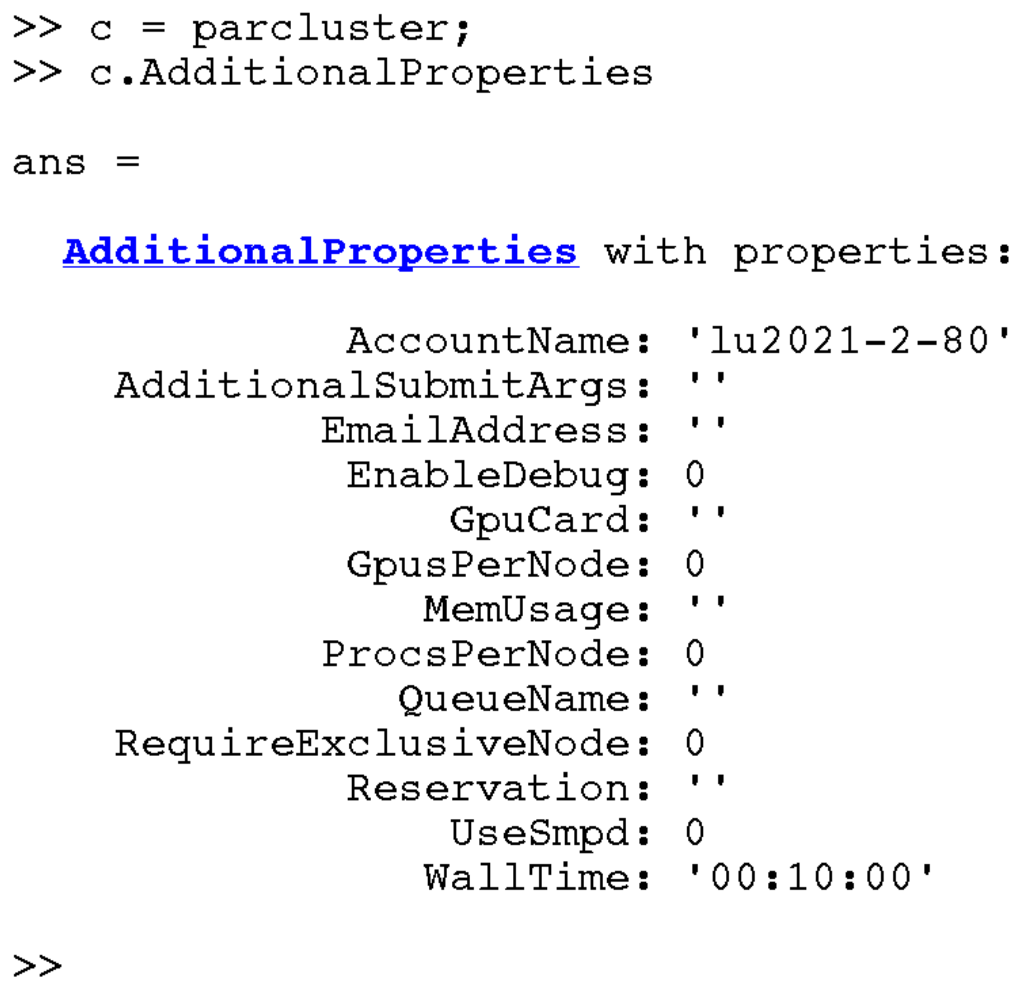
```
>> calc_pi
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 16).
Lab  1:
  Subinterval: [0    , 0.0625]
Lab  2:
  Subinterval: [0.0625, 0.125]
Lab  3:
  Subinterval: [0.125, 0.1875]
Lab  4:
  Subinterval: [0.1875, 0.25]
Lab  5:
  Subinterval: [0.25, 0.3125]
Lab  6:
  Subinterval: [0.3125, 0.375]
Lab  7:
  Subinterval: [0.375, 0.4375]
Lab  8:
  Subinterval: [0.4375, 0.5 ]
Lab  9:
  Subinterval: [0.5 , 0.5625]
Lab 10:
  Subinterval: [0.5625, 0.625]
Lab 11:
  Subinterval: [0.625, 0.6875]
Lab 12:
  Subinterval: [0.6875, 0.75]
Lab 13:
  Subinterval: [0.75, 0.8125]
Lab 14:
  Subinterval: [0.8125, 0.875]
Lab 15:
  Subinterval: [0.875, 0.9375]
```





```
>> calc_pi_multi_node
Starting parallel pool (parpool) using the 'aurora R2020b' pro
additionalSubmitArgs =
    '--ntasks=40 --cpus-per-task=1 -N 2 --ntasks-per-core=1 -A
Connected to the parallel pool (number of workers: 40).
Lab  1:
  Subinterval: [0    , 0.025]
Lab  2:
  Subinterval: [0.025, 0.05]
Lab  3:
  Subinterval: [0.05, 0.075]
Lab  4:
  Subinterval: [0.075, 0.1 ]
Lab  5:
  Subinterval: [0.1 , 0.125]
Lab  6:
  Subinterval: [0.125, 0.15]
Lab  7:
  Subinterval: [0.15, 0.175]
Lab  8:
  Subinterval: [0.175, 0.2 ]
Lab  9:
  Subinterval: [0.2 , 0.225]
Lab 10:
  Subinterval: [0.225, 0.25]
Lab 11:
  Subinterval: [0.25, 0.275]
Lab 12:
  Subinterval: [0.275, 0.3 ]
Lab 13:
  Subinterval: [0.3 , 0.325]
Lab 14:
  Subinterval: [0.325, 0.35]
```

Other settable job properties



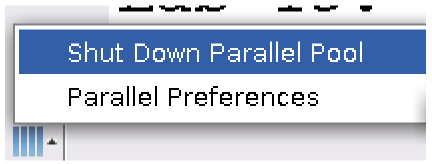
```
>> c = parcluster;
>> c.AdditionalProperties

ans =

  AdditionalProperties with properties:

            AccountName: 'lu2021-2-80'
   AdditionalSubmitArgs: ''
           EmailAddress: ''
            EnableDebug: 0
                GpuCard: ''
            GpusPerNode: 0
               MemUsage: ''
           ProcsPerNode: 0
              QueueName: ''
   RequireExclusiveNode: 0
            Reservation: ''
                UseSmpd: 0
               WallTime: '00:10:00'

>>
```

- AccountName
- EmailAddress
- GpuCard
- GpusPerNode
- MemUsage
- ProcsPerNode
- QueueName
- RequireExclusiveNode
- Reservation
- WallTime

GPUs

Start pool with GPU node

```
>> % Start a parallel pool with a GPU
>> c = parcluster;
>> c.AdditionalProperties.GpusPerNode = 1;
>>
>> p = c.parpool(1);
Starting parallel pool (parpool) using the 'aurora R2020b' profile ...

additionalSubmitArgs =

    '--ntasks=1 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00 --gres=gpu:1 -p gpu'

Connected to the parallel pool (number of workers: 1).
>>
```

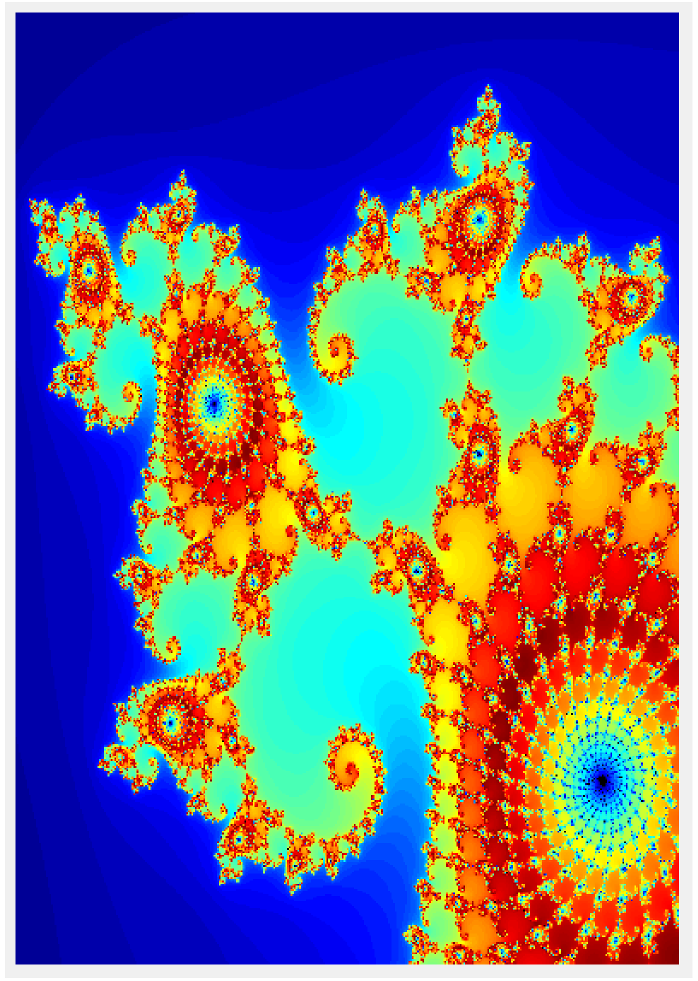


Tesla K20/40/80

```
>> spmd, gpuDevice, end
Lab 1:
  ans =
    CUDADevice with properties:

                      Name: 'Tesla K80'
                     Index: 1
         ComputeCapability: '3.7'
            SupportsDouble: 1
             DriverVersion: 11.2000
            ToolkitVersion: 10.2000
        MaxThreadsPerBlock: 1024
          MaxShmemPerBlock: 49152
        MaxThreadBlockSize: [1024 1024 64]
               MaxGridSize: [2.1475e+09 65535 65535]
                 SIMDWidth: 32
               TotalMemory: 1.1997e+10
           AvailableMemory: 1.1601e+10
       MultiprocessorCount: 13
              ClockRateKHz: 823500
               ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 0
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1
>>
```

Example: mandelbrot (1)

```
function [x,y,count,t] = calc_mandelbrot(type)

maxIterations = 1000;
gridSize = 4000;
xlim = [-0.748766713922161, -0.748766707771757];
ylim = [ 0.123640844894862,  0.123640851045266];

t0 = tic;
if strcmp(type,'gpuArray')
    x = gpuArray.linspace(xlim(1),xlim(2),gridSize);
    y = gpuArray.linspace(ylim(1),ylim(2),gridSize);
else
    x = linspace(xlim(1),xlim(2),gridSize);
    y = linspace(ylim(1),ylim(2),gridSize);
end

[xGrid,yGrid] = meshgrid(x,y);
z0 = complex(xGrid,yGrid);
count = ones(size(z0),type);

z = z0;
for n = 0:maxIterations
    z = z.*z + z0;
    inside = abs(z) <= 2;
    count = count + inside;
end
count = log(count);
t = toc(t0);

end
```



Example: mandelbrot (2)

```
function mandelbrot_example

% Run on CPU
[~, ~, ~, cpu_t] = calc_mandelbrot('double');

% Run on GPU
[~, ~, ~, gpu_t] = calc_mandelbrot('gpuArray');

fprintf('CPU time: %0.2f\n',cpu_t)
fprintf('GPU time: %0.2f\n',gpu_t)

end
```

```
>> spmd, mandelbrot_example, end
Lab 1:
  CPU time: 524.85
  GPU time: 12.08
>>
```

Example: FFT (1)

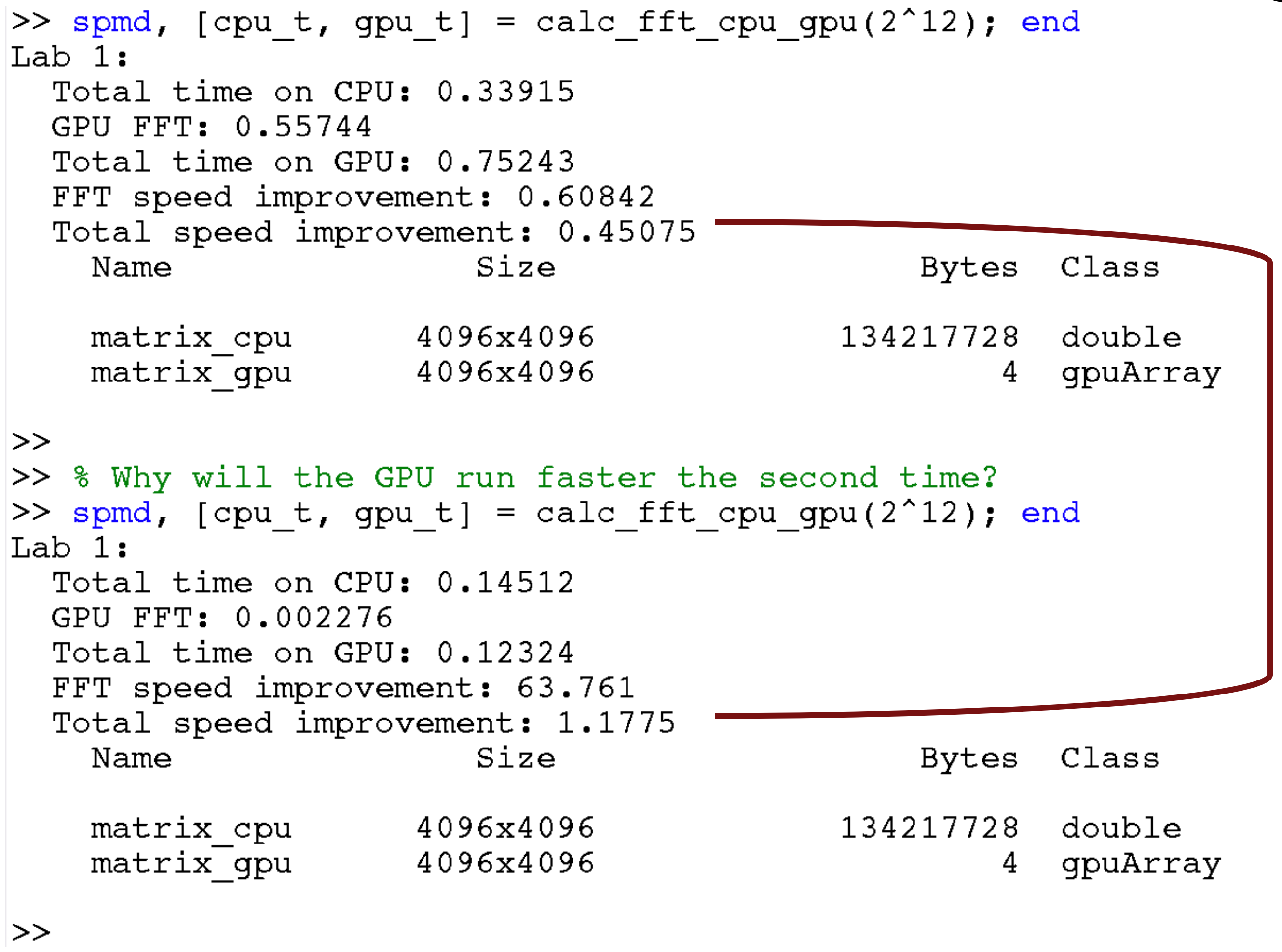
```
function [time_cpu, time_gpu] = calc_fft_cpu_gpu(N)

matrix_cpu = rand(N);

tic
out_cpu = fft(matrix_cpu);
time_cpu = toc;
disp(['Total time on CPU: ' num2str(time_cpu)])

t0 = tic;
% Transfer matrix to GPU device
matrix_gpu = gpuArray(matrix_cpu);

t1 = tic;
out_gpu = fft(matrix_gpu);
time_gfft = toc(t1);

% Gather back from GPU to CPU
gather_gpu = gather(out_gpu);

% Wait for transfer to complete
wait(gpuDevice)
time_gpu = toc(t0);

disp(['GPU FFT: ' num2str(time_gfft)])
disp(['Total time on GPU: ' num2str(time_gpu)])

disp(['FFT speed improvement: ' num2str(time_cpu/time_gfft)])
disp(['Total speed improvement: ' num2str(time_cpu/time_gpu)])

whos matrix_cpu matrix_gpu

end
```

Example: FFT (2)

```
>> % 128 (MB)
>> sz = 2^12*2^12*8/1024^2
sz =
   128
>>
>> % GPU memory (GB)
>> spmd, d = gpuDevice; d.AvailableMemory/1024^2, end
Lab 1:
  ans =
          11208.94
>>
```

Example: FFT (3)

Why did the GPU code run faster the 2nd time?



```
>> spmd, [cpu_t, gpu_t] = calc_fft_cpu_gpu(2^12); end
Lab 1:
  Total time on CPU: 0.33915
  GPU FFT: 0.55744
  Total time on GPU: 0.75243
  FFT speed improvement: 0.60842
  Total speed improvement: 0.45075
    Name                   Size                     Bytes  Class

    matrix_cpu          4096x4096               134217728  double
    matrix_gpu          4096x4096                       4  gpuArray

>>
>> % Why will the GPU run faster the second time?
>> spmd, [cpu_t, gpu_t] = calc_fft_cpu_gpu(2^12); end
Lab 1:
  Total time on CPU: 0.14512
  GPU FFT: 0.002276
  Total time on GPU: 0.12324
  FFT speed improvement: 63.761
  Total speed improvement: 1.1775
    Name                   Size                     Bytes  Class

    matrix_cpu          4096x4096               134217728  double
    matrix_gpu          4096x4096                       4  gpuArray

>>
```

Turnoff GPU requests when you don’t need them anymore

```
>> c.AdditionalProperties.GpusPerNode = 0;
```

Interactively: with batch job, asynchronously

`batch`

Exercise: “Hello, World!”

```
>> % Submit job to Aurora to find out where MATLAB is running
>> c = parcluster;
>> j = c.batch(@pwd,1,{});
additionalSubmitArgs =
    '--ntasks=1 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00
>>
>> % Check the state of the job
>> j.State
ans =
    'finished'
>>
>> % Fetch the results
>> j.fetchOutputs{:}
ans =
    '/home/raymond/Documents/MATLAB/matlab-workshop'
>>
```

Set the `batch` *CurrentFolder* argument to change default location

Exercise: Calculate π

```
>> % Submit calc_pi job
>> c = parcluster;
>>
>> % Request 16 workers
>> j = c.batch(@calc_pi,0,{}, 'Pool',16);
additionalSubmitArgs =
    '--ntasks=17 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00
>>
```

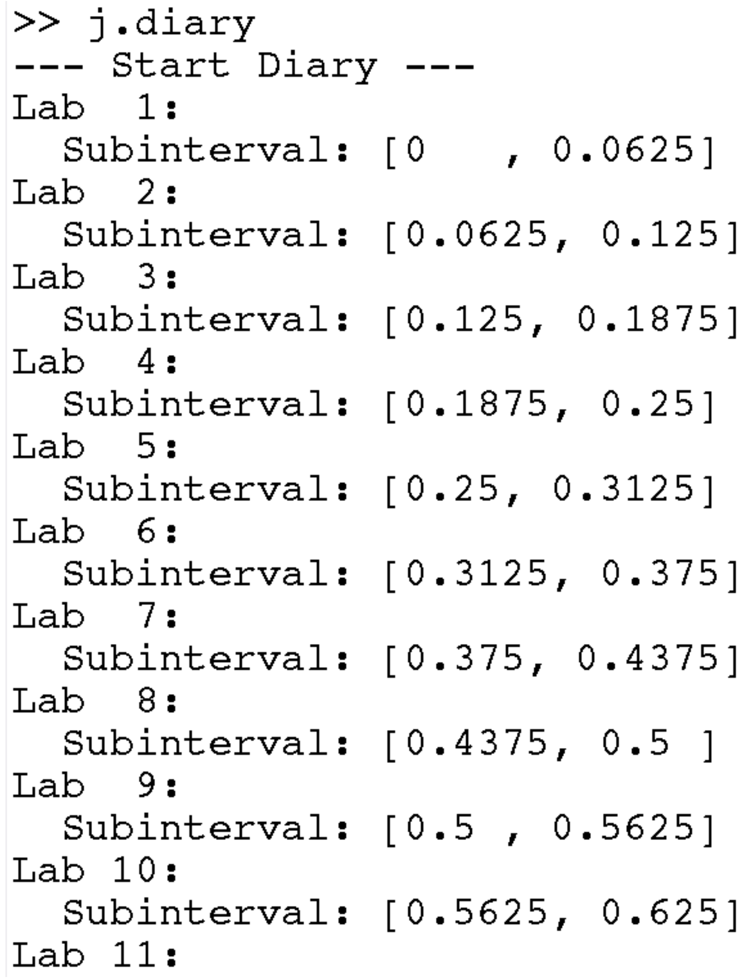
“If my Pool is size 16, why am I requesting 17 tasks?”

Fetch the results

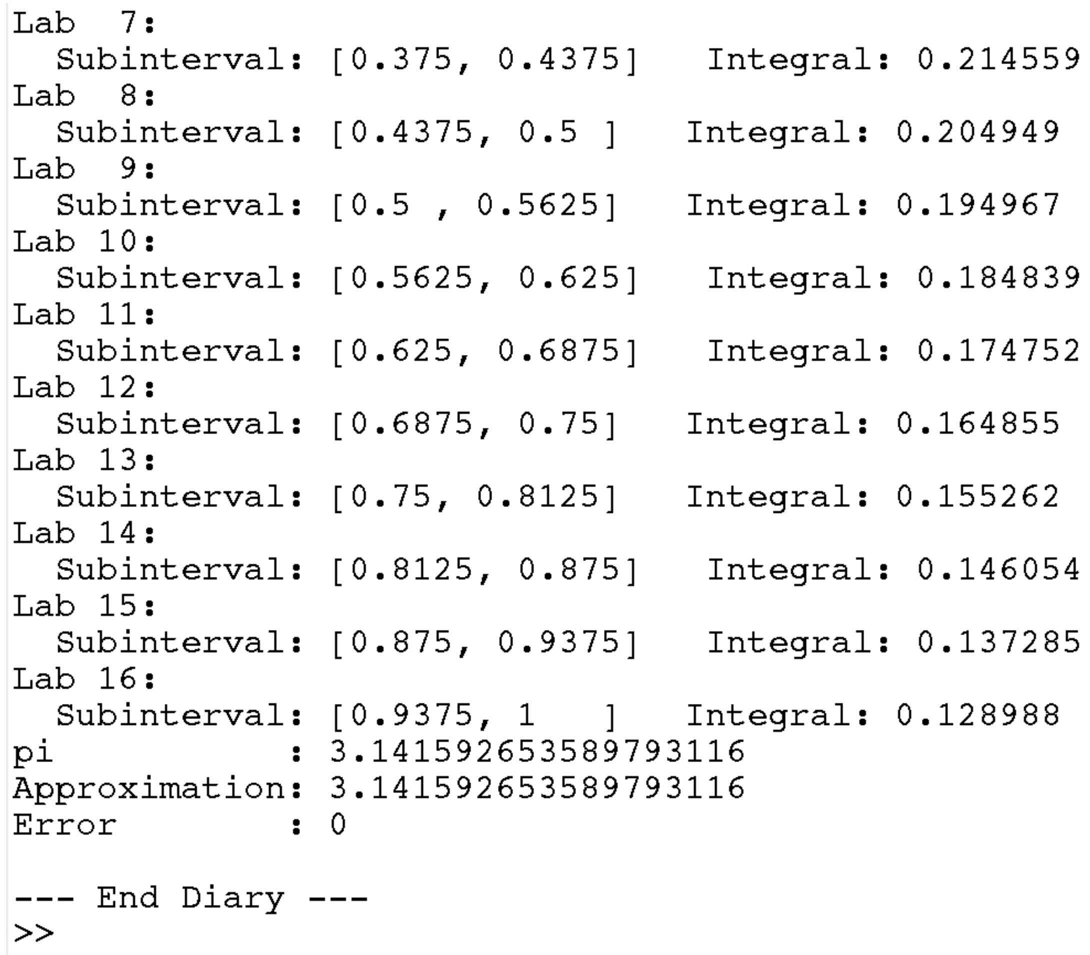
```
>> % Submit calc_pi job
>> c = parcluster;
>>
>> % Request 16 workers
>> j = c.batch(@calc_pi,0,{}, 'Pool',16);
additionalSubmitArgs =
    '--ntasks=17 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00
>>
>> % Check the state of the job
>> j.State
ans =
    'finished'
>>
>> % Fetch the results
>> j.fetchOutputs{:}
>>
```

“Where’s the output?”

Fetch the diary



```
>> j.diary
--- Start Diary ---
Lab  1:
  Subinterval: [0    , 0.0625]
Lab  2:
  Subinterval: [0.0625, 0.125]
Lab  3:
  Subinterval: [0.125, 0.1875]
Lab  4:
  Subinterval: [0.1875, 0.25]
Lab  5:
  Subinterval: [0.25, 0.3125]
Lab  6:
  Subinterval: [0.3125, 0.375]
Lab  7:
  Subinterval: [0.375, 0.4375]
Lab  8:
  Subinterval: [0.4375, 0.5 ]
Lab  9:
  Subinterval: [0.5 , 0.5625]
Lab 10:
  Subinterval: [0.5625, 0.625]
Lab 11:
```



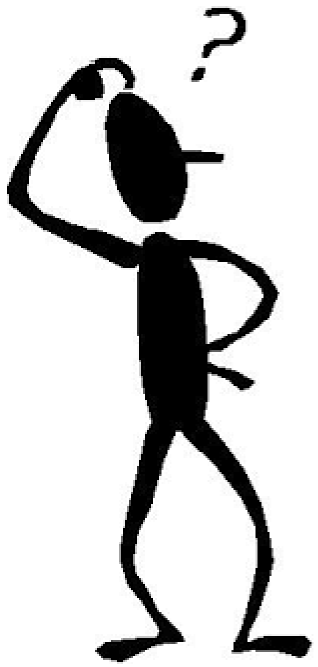
```
Lab  7:
  Subinterval: [0.375, 0.4375]   Integral: 0.214559
Lab  8:
  Subinterval: [0.4375, 0.5 ]   Integral: 0.204949
Lab  9:
  Subinterval: [0.5 , 0.5625]   Integral: 0.194967
Lab 10:
  Subinterval: [0.5625, 0.625]   Integral: 0.184839
Lab 11:
  Subinterval: [0.625, 0.6875]   Integral: 0.174752
Lab 12:
  Subinterval: [0.6875, 0.75]   Integral: 0.164855
Lab 13:
  Subinterval: [0.75, 0.8125]   Integral: 0.155262
Lab 14:
  Subinterval: [0.8125, 0.875]   Integral: 0.146054
Lab 15:
  Subinterval: [0.875, 0.9375]   Integral: 0.137285
Lab 16:
  Subinterval: [0.9375, 1    ]   Integral: 0.128988
pi           : 3.141592653589793116
Approximation: 3.141592653589793116
Error        : 0

--- End Diary ---
>>
```

What gets “returned”?

- Function output
- Diary
- Saved files

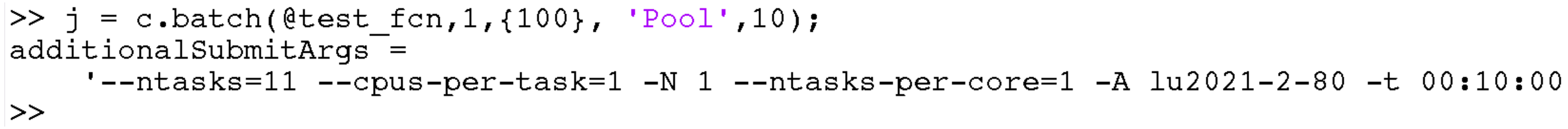
Example

“What size Pool am I running?



```
function [t, A] = test_fcn(sims)

disp('Start sim')

A = nan(sims,1);
t0 = tic;
parfor idx = 1:sims
    A(idx) = idx;
    pause(0.5)
    idx
end
t = toc(t0);

disp('Finished')

save RESULTS A
```

Job submission



```
>> j = c.batch(@test_fcn,1,{100}, 'Pool',10);
additionalSubmitArgs =
    '--ntasks=11 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00
>>
```

Fetch the results

```
function [t, A] = test_fcn(sims)
```

```
c.batch(@test_fcn,1,{100},
```

```
>> % Fetch the results
>> j.fetchOutputs{:}
ans =
    6.0654
>>
```

“Where’s A?”

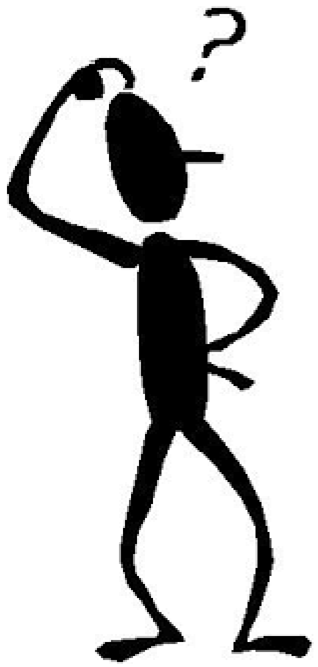
Fetch the diary

```
% View the diary
j.diary
--- Start Diary ---
Start sim

ans =

     2


ans =

     4

 ...

ans =

   100


ans =

    98

Finished

--- End Diary ---
```

```
function [t, A] = test_fcn(sims)

disp('Start sim')

A = nan(sims,1);
t0 = tic;
parfor idx = 1:sims
    A(idx) = idx;
    pause(0.5)
    idx
end
t = toc(t0);

disp('Finished')

save RESULTS A
```

Save files

"Where does **RESULTS** get written to?"



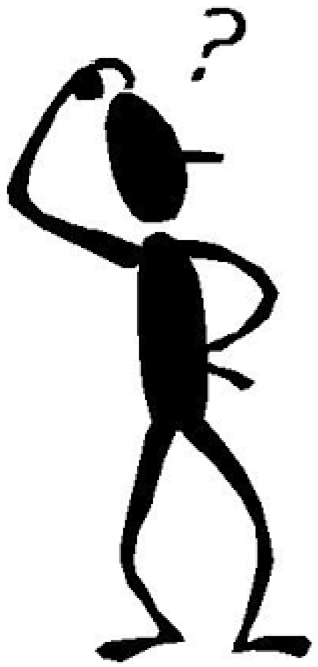
```
function [t, A] = test_fcn(sims)

disp('Start sim')

A = nan(sims,1);
t0 = tic;
parfor idx = 1:sims
    A(idx) = idx;
    pause(0.5)
    idx
end
t = toc(t0);

disp('Finished')

save RESULTS A
```

Submitting scripts, instead of functions

```
>> z = 10;
>>
>> % Submit a script (instead of a function)
>> j = c.batch('x = 3; y = 4, z');

additionalSubmitArgs =

    '--ntasks=1 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00'

>>
```

Loading variables to local workspace

"If we cleared z, then why does who display it?"

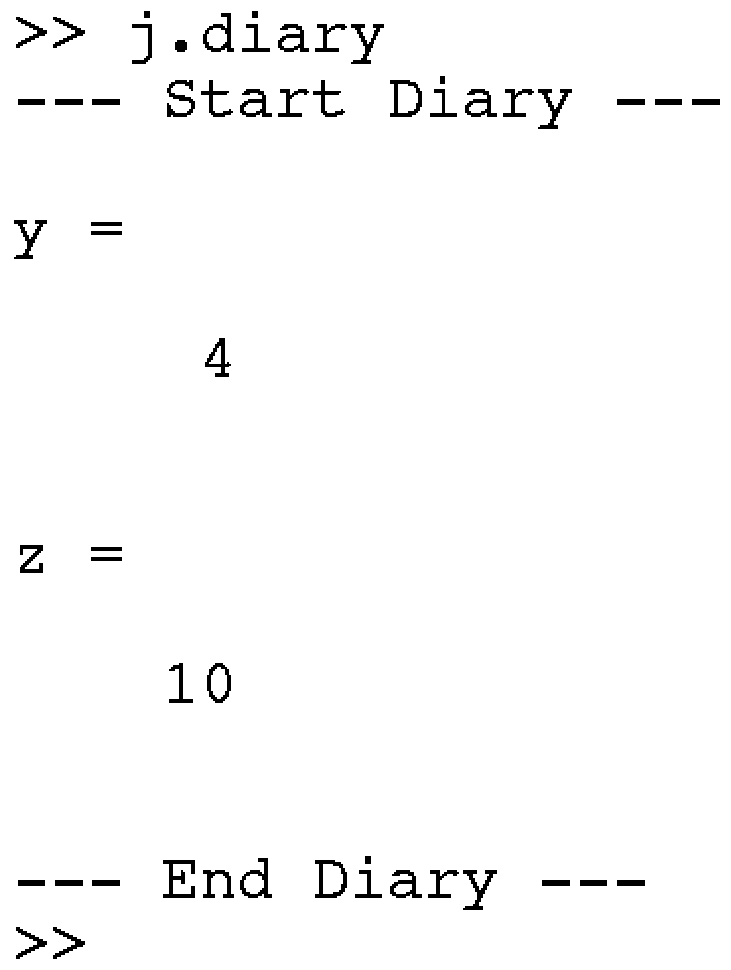


```
>> clear z
>> who

Your variables are:

c   j

>> % Check the State of the job
>> j.State

ans =

    'finished'

>> % Load variables from the job
>> j.load
>> who

Your variables are:

c   j   x   y   z

>>
```

"I'll pass all of the variables in my local workspace to all of the workers. Then I'll receive everything the workers generate and load it back to my local workspace."

```
>> z = 10;
>>
>> % Submit a
>> j = c.batch
```

Getting the diary of scripts

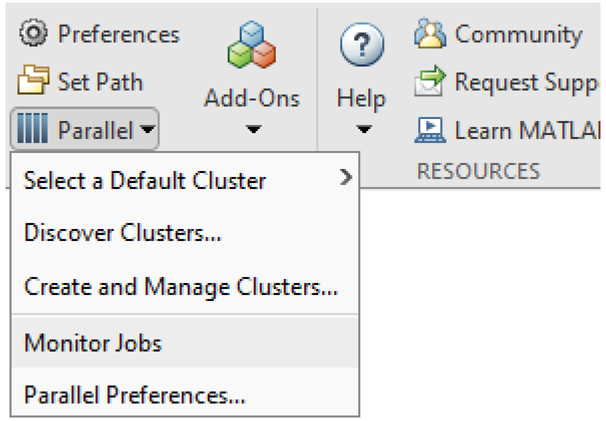


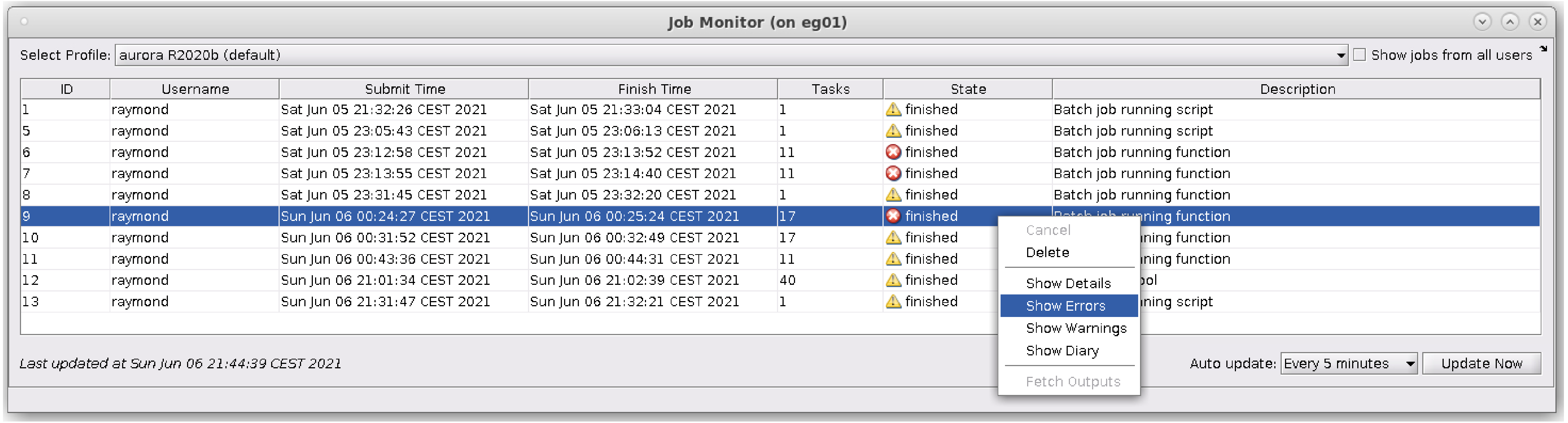
```
>> j.diary
--- Start Diary ---

y =

     4


z =

    10


--- End Diary ---
>>
```

When has my job run and finished?

```
>> % Get email notification when job has finished
>> c.AdditionalProperties.EmailAddress = 'userid@lu.se';
>>
>> j = c.batch(@test_fcn,1,{100}, 'Pool',10);
additionalSubmitArgs =
    '--ntasks=11 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00 --mail-type=ALL
>>
```

Retrieving past jobs





Select Profile: aurora R2020b (default)   ☐ Show jobs from all users

ID	Username	Submit Time	Finish Time	Tasks	State	Description
1	raymond	Sat Jun 05 21:32:26 CEST 2021	Sat Jun 05 21:33:04 CEST 2021	1	⚠ finished	Batch job running script
5	raymond	Sat Jun 05 23:05:43 CEST 2021	Sat Jun 05 23:06:13 CEST 2021	1	⚠ finished	Batch job running script
6	raymond	Sat Jun 05 23:12:58 CEST 2021	Sat Jun 05 23:13:52 CEST 2021	11	✖ finished	Batch job running function
7	raymond	Sat Jun 05 23:13:55 CEST 2021	Sat Jun 05 23:14:40 CEST 2021	11	✖ finished	Batch job running function
8	raymond	Sat Jun 05 23:31:45 CEST 2021	Sat Jun 05 23:32:20 CEST 2021	1	⚠ finished	Batch job running function
9	raymond	Sun Jun 06 00:24:27 CEST 2021	Sun Jun 06 00:25:24 CEST 2021	17	✖ finished	Batch job running function
10	raymond	Sun Jun 06 00:31:52 CEST 2021	Sun Jun 06 00:32:49 CEST 2021	17	⚠ finished	ning function
11	raymond	Sun Jun 06 00:43:36 CEST 2021	Sun Jun 06 00:44:31 CEST 2021	11	⚠ finished	ning function
12	raymond	Sun Jun 06 21:01:34 CEST 2021	Sun Jun 06 21:02:39 CEST 2021	40	⚠ finished	ol
13	raymond	Sun Jun 06 21:31:47 CEST 2021	Sun Jun 06 21:32:21 CEST 2021	1	⚠ finished	ning script

Last updated at Sun Jun 06 21:44:39 CEST 2021

Auto update: Every 5 minutes

Keep cluster files minimal: delete jobs

- As a good practice, delete jobs you no longer need

```
>> % Finished with the job, delete it to cleanup list of jobs
>> j.delete
>>
```

Noninteractively: with Slurm job script

`sbatch`

Single-node job & multi-node job scripts

If we're running a multi-node job, why did we only request 1 core?

Why are we asking for more walltime for a multi-node job?

```
#!/bin/sh

# #SBATCH -A
#SBATCH -n 1                            # 1 instance of MATLAB
#SBATCH --cpus-per-task=8               # 8 cores per instance
#SBATCH --mem-per-cpu=4gb               # 4 GB RAM per core
#SBATCH --time=00:10:00                 # 10 minutes

# Add MATLAB to system path
module load matlab

# Run code
matlab -batch calc_pi
```

matlab-single-node.slurm

```
#!/bin/sh

# #SBATCH -A
#SBATCH -n 1                            # 1 instance of MATLAB
#SBATCH --cpus-per-task=1               # 1 core per instance
#SBATCH --mem-per-cpu=4gb               # 4 GB RAM per core
#SBATCH --time=00:20:00                 # 20 minutes

# Add MATLAB to system path
module load matlab

# Run code
matlab -batch calc_pi_multi_node
```

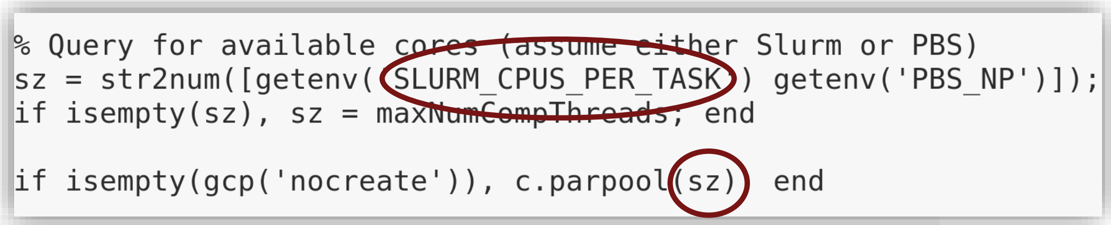
matlab-multi-node.slurm

Job submission

```
-bash4.2 sbatch -A lu2021-2-80 matlab-single-node.slurm
Submitted batch job 5489731
```

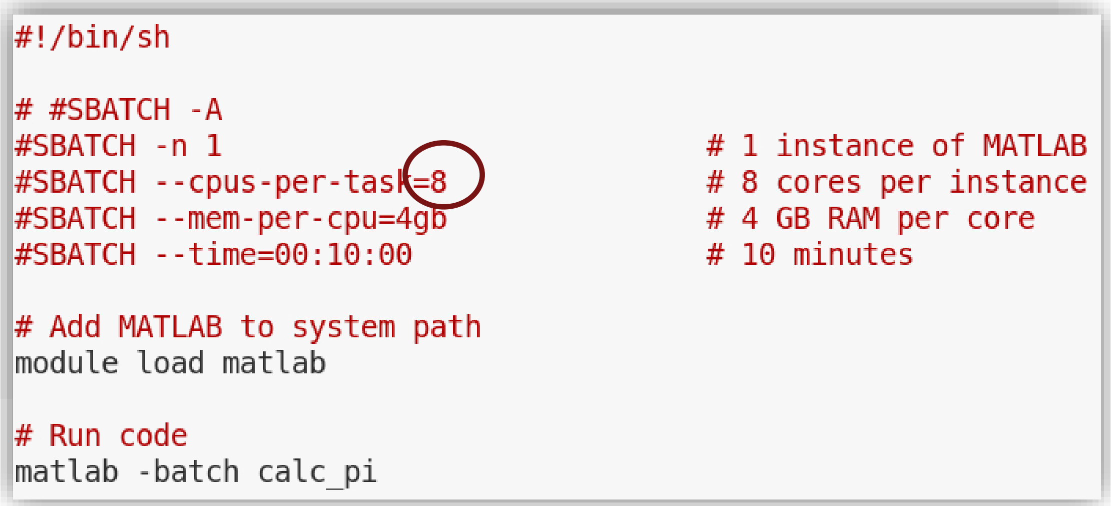
```
-bash4.2 sbatch -A lu2021-2-80 matlab-multi-node.slurm
Submitted batch job 5489738
```

Single node job



```
% Query for available cores (assume either Slurm or PBS)
sz = str2num([getenv('SLURM_CPUS_PER_TASK') getenv('PBS_NP')]);
if isempty(sz), sz = maxNumCompThreads; end

if isempty(gcp('nocreate')), c.parpool(sz), end
```

```
-bash4.2 squeue -j 5489731
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           5489731        lu matlab-s  raymond  R       0:17      1 au015
-bash4.2
-bash4.2 squeue -j 5489731
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
-bash4.2
-bash4.2 head -10 slurm-5489731.out
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 8).
Lab 1:
  Subinterval: [0    , 0.125]
Lab 2:
  Subinterval: [0.125, 0.25]
Lab 3:
  Subinterval: [0.25, 0.375]
Lab 4:
  Subinterval: [0.375, 0.5 ]
-bash4.2
```



```
#!/bin/sh

# #SBATCH -A
#SBATCH -n 1                                # 1 instance of MATLAB
#SBATCH --cpus-per-task=8                   # 8 cores per instance
#SBATCH --mem-per-cpu=4gb                   # 4 GB RAM per core
#SBATCH --time=00:10:00                     # 10 minutes

# Add MATLAB to system path
module load matlab

# Run code
matlab -batch calc_pi
```

Multi-node job

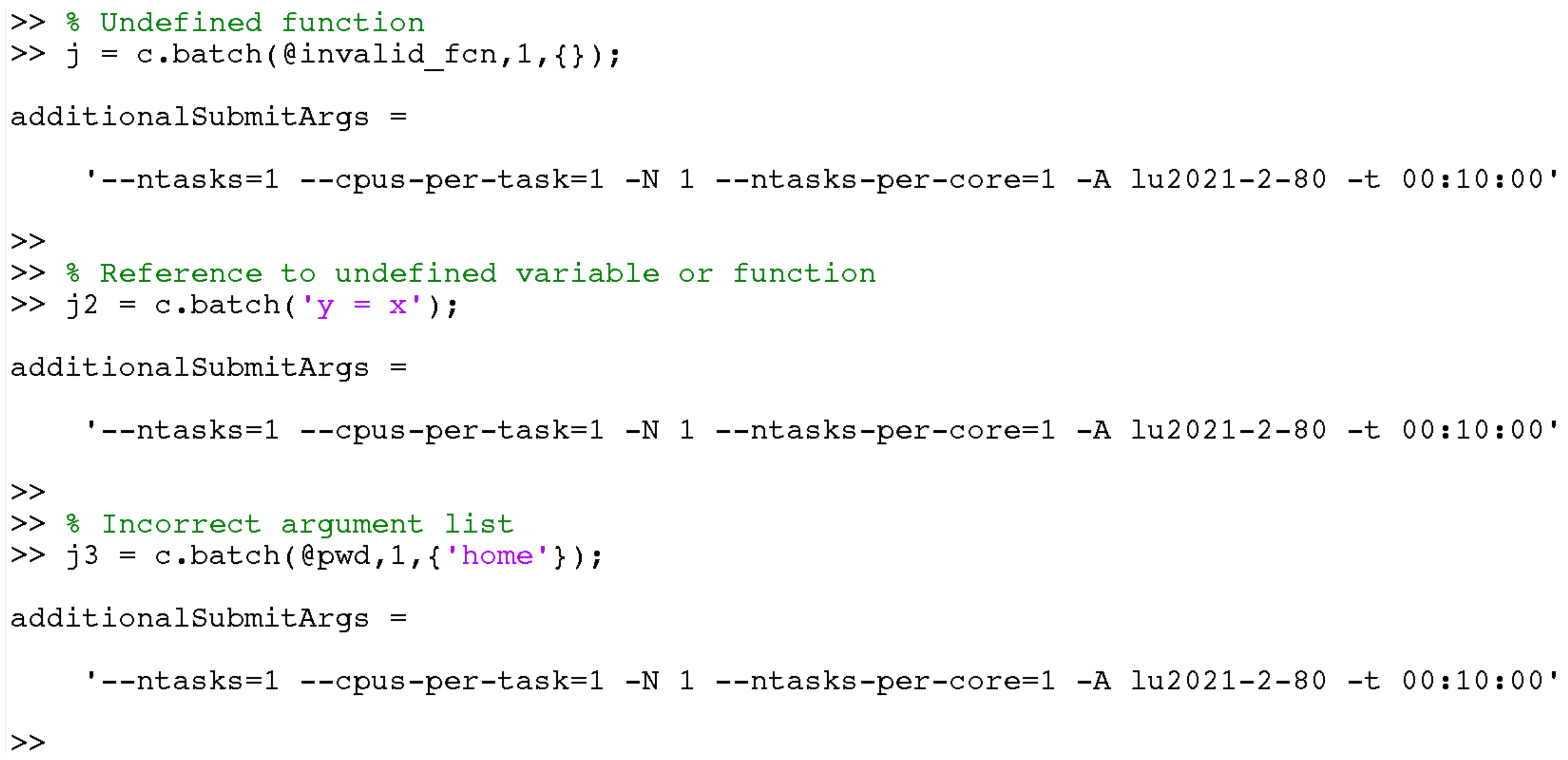
```
-bash4.2 head -10 slurm-5489738.out
Starting parallel pool (parpool) using the 'aurora R2020b' profile ...

additionalSubmitArgs =

    '--ntasks=40 --cpus-per-task=1 -N 2 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00 --exclusive'

Connected to the parallel pool (number of workers: 40).
Lab  2:
  Subinterval: [0.025, 0.05]
Lab  3:
-bash4.2
```

Debugging and Troubleshooting

Scheduler ID

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

additionalSubmitArgs =

    '--ntasks=1 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00'

>>
>> % Job ID vs Scheduler ID
>> j.ID

ans =

    14

>>
>> j.getTaskSchedulerIDs{1}

ans =

    '5489841'

>>
```

Examples



```
>> % Undefined function
>> j = c.batch(@invalid_fcn,1,{});

additionalSubmitArgs =

    '--ntasks=1 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00'

>>
>> % Reference to undefined variable or function
>> j2 = c.batch('y = x');

additionalSubmitArgs =

    '--ntasks=1 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00'

>>
>> % Incorrect argument list
>> j3 = c.batch(@pwd,1,{'home'});

additionalSubmitArgs =

    '--ntasks=1 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00'

>>
```

Errored jobs (1)

```
>> % Undefined function
>> j.State

ans =

    'finished'

>>
>> j.fetchOutputs{:}
Error using parallel.Job/fetchOutputs (line 1264)
An error occurred during execution of Task with ID 1.

Caused by:
    Unrecognized function or variable 'invalid_fcn'.

>>
```

Errored jobs (2)

```
>> % Reference to undefined variable or function
>> j2.State

ans =

    'finished'

>>
>> j2.fetchOutputs{:}
Error using parallel.Job/fetchOutputs (line 1264)
An error occurred during execution of Task with ID 1.

Caused by:
    Unrecognized function or variable 'x'.

>>
```

Errored jobs (3)

```
>> % Incorrect argument list
>> j3.State

ans =

    'finished'

>>
>> j3.fetchOutputs{:}
Error using parallel.Job/fetchOutputs (line 1264)
An error occurred during execution of Task with ID 1.

Caused by:
    Too many input arguments.

>>
```

Logfile: Single core job

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

additionalSubmitArgs =

    '--ntasks=1 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00'

>>
>> % Retrieve log file for single core job
>> c.getDebugLog(j.Tasks(1))
LOG FILE OUTPUT:
Executing: /sw/pkg/matlab/R2020b/bin/worker
2021-06-07 06:12:21 | About to exit MATLAB normally
2021-06-07 06:12:21 | About to exit with code: 0
Exiting with code: 0

>>
```

Logfile: Multi-core job

```
>> j = c.batch(@pwd,1,{}, 'Pool',2);

additionalSubmitArgs =

    '--ntasks=3 --cpus-per-task=1 -N 1 --ntasks-per-core=1 -A lu2021-2-80 -t 00:10:00'

>>
>> % Retrieve log file for multi-core job
>> c.getDebugLog(j)
LOG FILE OUTPUT:
The scheduler has allocated the following nodes to this job:
au113

The following have been reloaded with a version change:
  1) GCCcore/6.3.0 => GCCcore/8.2.0     2) binutils/2.27 => binutils/2.31.1

"mpiexec.hydra" -l -n 3 "/sw/pkg/matlab/R2020b/bin/worker" -parallel
[0] Sending a stop signal to all the labs...
[0] 2021-06-07 06:09:40 | About to exit MATLAB normally
[0] 2021-06-07 06:09:40 | About to exit with code: 0
Exiting with code: 0

>>
```

Designing Robust Code

From Coding to Cluster (1)

```
% Notes - From Coding to Cluster
% 1. Using a script, not a function
% 2. Paths are hardcoded
% 3. File separator is hard coded
% 4. Assumes TIF file exists
% 5. TIF files must be on the MATLAB path
% 6. Assumes output folder already exists where ever MATLAB is running
% 7. Results MAT-File will be overwritten next time it's run
% 8. Changes MATLAB working directory

filelist = dir('tif\*.tif');
fileNames = {filelist.name}';

segmentedCellSequence = batchProcessFiles(@detectCells,fileNames);
cd output
save SCS segmentedCellSequence
```

```
function [ofile, segmentedCellSequence] = process_files_v2(rootd,outd)
if nargin==0
    rootd = fullfile(pwd,'tif');
    outd = fullfile(pwd,'output');
end

filelist = dir(fullfile(rootd,'*.tif'));
if isempty(filelist)
    error('Failed to find image files: %s',rootd)
end
fileNames = {filelist.name}';

addpath(rootd)
segmentedCellSequence = batchProcessFiles(@detectCells,fileNames);

% Ensure output directory exists
if exist(outd,'dir')==false
    [FAILED,emsg,eid] = mkdir(outd);
    if FAILED==true
        error(eid,emsg)
    end
end

% Add timestamp for file uniqueness
ts = strrep(strrep(datestr(now),' ','_'),':','-');

% Save dir
old_dir = pwd;
c = onCleanup(@()cd(old_dir));
cd(outd)
ofile = ['SCS_' ts];
save(ofile,'segmentedCellSequence')
```

Run it locally

```
>> % Start local parallel pool
>> parpool(4);
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).
>>
>> % Call the function locally
>> ofile = process_files_v2

ofile =

    'S:\sandbox\Workshops\Parallel-Computing-Workshop\matlab-workshop-files'

>>
```

Run it on the cluster

```
>> % Submit job to cluster
>> c = parcluster;
>> j = c.batch(@process_files_v2, 1, {'/work/raymond/proj-tiffs','/home/raymond/output-results'},'Pool',3);
>>
>> % Wait for job to finish
>> j.wait
>>
>> % Fetch the results
>> ofile = j.fetchOutputs{:}

ofile =

    '/home/raymond/output-results/SCS_27-Apr-2021_16-54-28'

>>
```

From Coding to Cluster (2)

```
% Notes - From Coding to Cluster
% 1. Using a script, not a function
%         return status or output directory
% 2. Paths are hardcoded
%         pass in root directory
% 3. File separator is hard coded
%         use fullfile
% 4. Assumes TIF file exists
%         check results when touching the file system
% 5. TIF files must be on the MATLAB path
%         add tif folder to the MATLAB path
% 6. Assumes output folder already exists where ever MATLAB is running
%         supply output directory to write to.  check if folder exists; if
%         not, create it
% 7. Results MAT-File will be overwritten next time it's run
%         add timestamp to filename
% 8. Changes MATLAB working directory
%         Track old directory, change back before leaving
```