MATLAB Distributed Computing Server (MDCS) Training

Artemis HPC Integration and Parallel Computing with MATLAB

Dr Hayim Dar
hayim.dar@sydney.edu.au

Dr Nathaniel Butterworth
nathaniel.butterworth@sydney.edu.au

Sydney Informatics Hub
A Core Research Facility
Introduction

- MATLAB has the functionality to use HPC from within the MATLAB GUI.
- We have installed Matlab Distributed Computing Server (MDCS) on Artemis
- This lets you submit jobs to Artemis from your own computer running Matlab R2017a
- Up to 12 CPUs and 2 GPUs to run your job interactively.
- Check out the website: https://sydneyuni.atlassian.net/wiki/spaces/RC/pages/228458497/Submitting+jobs+to+Artemis+from+Matlab
Informatics and Research Computing Expertise

Provides research computing expertise, training, and support

- Data science, analyses and support (bioinformatics, modelling, simulation, visualisation, machine learning, statistics)

- Training and workshops
  - High Performance Computing (HPC)
  - Programming (R, Python, Matlab, Scripting, GPU)
  - Code management (Git)
  - Bioinformatics (RNA-Seq, Genomics)

- Research Computing Support
  - Artemis HPC
  - Argus Virtual Research Desktop
  - Bioinformatics software support (CLC Genomics Workbench, Ingenuity Pathways Analysis)

- Events and Competitions
  - HPC Publication Incentive – High quality papers that acknowledge SIH and/or HPC/VRD
  - Artemis HPC Symposium
Complete the Survey:  
https://redcap.sydney.edu.au/surveys/?s=RAF898M7XL

W: https://informatics.sydney.edu.au  
E: sih.info@sydney.edu.au
Target users

- Power-users of Matlab
- Or users that don’t have experience using HPC but are comfortable with Matlab
- Want more interactivity with their Matlab code
- Need parallel processing their local machine can not provide
Prerequisites

- An Artemis account – received via an RDMP: https://rdmp.sydney.edu.au/


- And integration scripts: https://github.sydney.edu.au/skol2049/Matlab-Artemis-Integration-Scripts.git
Hello World

- Let start with a simple Matlab function.
- Open up the Matlab function, `prime.m`

- Run the function with the terminal command:

  \[
  [\text{noPrimes},\text{primeTime}] = \text{prime}(1,10000)
  \]

  - This should return the number of primes between 1 and 10,000
  - And the time for the function to run (about < 1 second for this number range)

  \[
  \begin{align*}
  \text{noPrimes} &= 1229 \\
  \text{primeTime} &= 0.8236
  \end{align*}
  \]
Hello Par-World

- Let’s now delegate the main for-loop to multiple processors.

- We will make use of Matlab’s Parallel Computing Toolbox command ‘parfor’ in place of ‘for’.

- Most modern computers have some kind of multi-processing power. Your local machine may be capable of running this function in parallel (if not we will soon show you how to do this with Artemis).
Hello Par-World

- Open up the Matlab function `prime_par.m`
- Run the function with the terminal command:

  ```matlab
  [noPrimes,primeTime] = prime_par(1,10000)
  ```

- The First time this runs you will get the message:

  Starting parallel pool (parpool) using the 'local' profile ...connected to 4 workers.

- Run it again (now the pool has started) and the speed improvement is apparent:

  noPrimes = 1229
  primeTime = 0.4380
Hello Par-Optimised-World

- We can further optimise the function now that it is parallelised, which can speed things up.
Hello Par-Optimised-World

- Open up the Matlab function `prime_ppar.m`
- Run the function with the terminal command:
  
  \[ \text{[noPrimes,primeTime,attrib] = prime_ppar(1,10000)} \]

- Once again, we get an improvement with speed:

  \[ \text{noPrimes = 1229} \]
  \[ \text{primeTime = 0.2734} \]

- This function makes manually breaks up the input domain and divides it amongst the workers, which perform the same algorithm over their chunk of the data.
Hello BIG Par-Artemis World

- If the workloads become too much for our local machines you can put Artemis to work for us.

- A ‘cluster profile’ contains all the necessary information to connect your machine to USYD’s HPC Artemis


- We will come back to this script after we set up our client…
Import the Artemis cluster profile


- After extraction, you should have an Artemis.settings file and a directory called artemis-integration-scripts.
Import the Artemis cluster profile

- Open Matlab R2017a on your computer.

- In the Matlab R2017a client, click the Home tab, select Parallel, then select Manage Cluster Profiles.
Import the Artemis cluster profile

- In the Cluster Profile Manager, click the Import button and select the Artemis.settings file.
Artemis cluster profile configuration

- If you successfully imported the Artemis cluster profile, you should see a new cluster profile called *Artemis* in the Cluster Profile manager.
- The imported Artemis cluster profile needs additional configuration before use.
- Select the imported Artemis cluster profile in the Cluster profile manager window.
- Click the *Edit* button, either in the lower-right hand corner, or in the menu bar at the top of the window.
Artemis cluster profile configuration

- In the Scheduler Integration section, set the Folder containing scheduler integration scripts option to the path to the artemis-integration-scripts folder you extracted earlier.
- In the additional properties for integration scripts section:
  - Set Project to your abbreviated project name as specified in your RDMP.
  - Set RemoteJobStorageLocation to a directory on Artemis where you would like your job’s files to be stored. (e.g. `/project/Training, /home/<unikey>`)
Artemis cluster profile configuration:

- Adjust the *Walltime* option to a value appropriate for your job. Walltime is the maximum amount of time your job will run on Artemis. If your job exceeds this walltime, it will be terminated and you will lose all unsaved progress. It is best to overestimate the walltime if you are unsure of how much you need.

- *Mem*: the amount of RAM (memory) your job needs. It is best to allocate a few gigabytes more than you expect to use.

- *Ngpus*: the number of GPUs required. The maximum you can request is two. Only use this option if you know your job can use GPUs.
Queue: the Artemis job queue your job will be placed in. If you are unsure of what these are, leave it blank. See the Artemis user guide for information about Artemis queues.

- **Important note:** Matlab cannot submit jobs to the scavenger queue. Jobs submitted to scavenger will be rejected with an error.

- **AdditionalSubmitArgs:** Additional PBS options for your submitted job.
Artemis cluster profile configuration

- In the Files and Folders section, in the AttachedFiles option, you may specify a directory or multiple directories on your local computer that will be copied to Artemis before your job starts. You may leave this blank if not required.

- In the AdditionalPaths box in the Files and Folders section, you may specify a folder or multiple folders on Artemis that contains input files needed for your job that you copied to Artemis in advance. You may leave this blank if not required.

- When you have finished editing the Artemis cluster profile, click “Done” in the bottom right-hand corner of the window.
Validate the cluster profile

- Select the Artemis Cluster Profile in the Cluster Profile manager, then Click the “Validate” icon in the menu bar at the top of the Cluster Profile Manager.

- The validation window will appear. Before the validation tests can run, you must Authenticate to Artemis.
Validate the cluster profile

- First, enter your UniKey in the “Enter the username for mdcshpc.sydney.edu.au” box, then click OK.
Validate the cluster profile

- Next, when it asks if you wish to use an identity file to login to mdcs.hpc.sydney.edu.au, answer No.
Validate the cluster profile

- Finally, enter your password in the next dialog box, then click OK.
Validate the cluster profile

- If the test was successful, all tests should pass except the "Parallel pool test (parpool)".
Let's submit a job!

- Now open up the Matlab script *primeBatch.m*

- This script uses our parallel function *prime_par.m* but sends the work to the remote HPC.

- The Matlab command ‘`batch`’ is where we give these instructions.

- Run the script to send the job to Artemis. It may take a while, so let's look at the code while we wait.
Hello BIG Par-Artemis World

- If you have not done so, click Run on `primeBatch.m`
- You will have to enter your unikey and password and select ‘No’ when prompted for an identity file. The same as when you were setting up the profile.
  - You will have to do this every time you submit a Matlab job to Artemis through your local client.
- Uncomment and increase the number of the upper bound to compare the timing of each method.
Hello BIG Par-Artemis World

```
job = batch(...
  @prime_par,... %Function Name to send to HPC
  2,... %Number of variables returned by fcn
  {lower,upper},... %Inputs to use in function
  'Profile','Artemis',... %Name of the Cluster Profile to use
  'Pool',11); %Number of cpus to use in pool (max of 11
    % - actually 11 compute cores +1 job
    %management core = 12) This max is enforced
    %by the current way the MDCS is set up.

wait(job) %This prevents us from using the terminal while we wait for
  %results from the batch job. Remove this to continue working.

%Get the results from the returned job
jobOut = fetchOutputs(job);
noPrimes4 = jobOut{1};
t4 = jobOut{2};
```
Hello BIG Par-Artemis World

```matlab
job = batch(...
@prime_par,... %Function Name to send to HPC
2,... %Number of variables returned by fcn
{lower,upper},... %Inputs to use in function
'Profile','Artemis',... %Name of the Cluster Profile to use
'Pool',11); %Number of cpus to use in pool (max of 11 
 % - actually 11 compute cores +1 job%
 %management core = 12) This max is enforced 
 %by the current way the MDCS is set up.

wait(job) %This prevents us from using the terminal while we wait for 
 %results from the batch job. Remove this to continue working.

%Get the results from the returned job
jobOut = fetchOutputs(job);
noPrimes4 = jobOut{1};
t4 = jobOut{2};
```
Hello BIG Par-Artemis World

```matlab
%wait(job)  %This prevents us from using the terminal while we wait for  
%results from the batch job. Remove this to continue working.

disp('hello world')
four = 2+2;
%blah blah... Some other computations that may be good to do locally.

%NOW because the next step relies on our Artemis calculation we should wait.
wait(job)

%Get the results from the returned job
jobOut = fetchOutputs(job);
noPrimes4 = jobOut{1};
t4 = jobOut{2};
```
Monitor the Job

- Jobs submitted to Artemis from Matlab can be monitored using the Job Monitor in Matlab R2017a. To open the Job Monitor, click the Home tab, click the Parallel button, then click Monitor Jobs.
Monitor the Job

- The Job Monitor window should open inside Matlab. Next, select the Artemis profile in the Select profile tab.
  
- To interact with the jobs, you can right-click them and select one of:
  
- **Cancel**: terminates the job or removes it from the queue.
  
- **Delete**: terminates the job, removes it from the queue and deletes it from the Job Monitor. Any data associated with the job will still remain on Artemis.
  
- **Show details/errors/warnings**: Shows details/errors/warnings for the job.
  
- **Fetch outputs or load variables**: Downloads all data stored in Matlab variables for that job and loads them into your local workspace.
Finally, let’s race them!

- Open up the Matlab script `primeProcess.m`

- This script runs all of our prime counting functions, locally and remotely.

- Run the script to see the results for each method.

- You will have to input your username for each job sent to Artemis. And select ‘no’ for identity file. Your password only has to be entered once.
More Examples: Attaching files to your world

- Run the script `batchTomo.m`
- This example sends some data stored in a file along with the function, `readTomoKmean.m`, to Artemis.

```matlab
%the Name of the file we are going to read and process
filename = 'ggge1202-sup-0002-ds01.txt';

%Send the function which reads the file to Artemis
job = batch(...
    @readTomoKmean,... %Function Name sent to Artemis
    5,...            %Number of variables returned by fcn to local
    {filename},,... %Input Parameters for function
    'AttachedFiles','ggge1202-sup-0002-ds01.txt',... %Files sent to Artemis
    'Profile','Artemis',... %Name of the Cluster Profile to use
    'Pool',2);        %How many CPUs to use (N-1)
```
More Examples: GPU Mandelbrot

- Run the script `gpu_demo_Mandelbrot.m` on Artemis using a batch command
- First, modify the cluster profile to use GPUs

```matlab
myCluster = parcluster('Artemis');
myCluster.AdditionalProperties.Queue = 'defaultQ';
myCluster.AdditionalProperties.Ngpus = '1';
myCluster.AdditionalProperties.Walltime = '00:5:00';

project_folder = myCluster.AdditionalProperties.RemoteJobStorageLocation;
```

- Then, submit the batch job!

```matlab
JOB = batch(...
    'gpu_demo_Mandelbrot', ...
    'Profile', 'Artemis', ...
    'CurrentFolder', project_folder);
```

- `qstat -T ...
- While we wait, let’s open the script and have a look what it does.

Optional! or use ‘ . ’
More Examples: GPU Mandelbrot

- Run the script `gpu_demo_Mandelbrot.m`
- Open it and have a look what it does!

```matlab
% The Mandelbrot Set in MATLAB
% The Mandelbrot set consists of complex numbers z0 for which the
% iterative series
%  z(0) = 0
%  { z(n+1) = z(n)^2 + z0
% remains bounded. This will be true (e.g.) if the limit of the magnitude
%  || z(n->Inf) || <= 2.

% Initialise the domain (a 2D grid in complex plane)
x = linspace( xlim(1), xlim(2), gridSize );
y = linspace( ylim(1), ylim(2), gridSize );
[xGrid,yGrid] = meshgrid( x, y );

z0 = xGrid + 1i*yGrid;
count = ones( size(z0) );
```

Based on the Mathworks example `paraleldemo_gpu_mandelbrot`
More Examples: GPU Mandelbrot

- Run the script `gpu_demo_Mandelbrot.m`
- Open it and have a look what it does!

```matlab
% The Mandelbrot Set in MATLAB
% The Mandelbrot set consists of complex numbers z_0 for which
% the iterative series
% \[ z(0) = 0 \]
% \[ z(n+1) = z(n)^2 + z_0 \]
% remains bounded. This will be true if the limit of the magnitude
% \( || z(n) || \to \infty \) is less than or equal to 2.
% || z(n->\infty) || <= 2.

% Initialise the domain (a 2D grid in complex plane)
x = linspace( xlim(1), xlim(2), gridSize );
y = linspace( ylim(1), ylim(2), gridSize );
[xGrid,yGrid] = meshgrid( x, y );

z0 = xGrid + 1i*yGrid;
count = ones( size(z0) );
```
More Examples: GPU Mandelbrot

- Run the script `gpu_demo_Mandelbrot.m`
- `>>` main routine:

```matlab
% Iterate!
% We will count the iteration after which it becomes evident that a series is
% unbounded.
z = z0; % Starting from iteration 1 ( z(0)=0 => z(1)=z0 )
for n = 1:maxIterations
    z = z.*z + z0;
    inside = abs( z )<=2;
    count = count + inside;
end
count = log( count ); % Use log to compress the range, for better plotting

% Show
cpuTime = toc( t );
```
More Examples: GPU Mandelbrot

- Run the script `gpu_demo_Mandelbrot.m`
- `>> GPU version 1: initialisation`

```matlab
% Initialise the domain (a 2D grid in complex plane)
% This time we initialise ON the GPU with 'gpuArray' <-- Matlab will
% then do all computations on this data on the GPU
x = gpuArray.linspace( xlim(1), xlim(2), gridSize );
y = gpuArray.linspace( ylim(1), ylim(2), gridSize );
[xGrid,yGrid] = meshgrid( x, y );

z0 = complex( xGrid, yGrid );
count = ones( size(z0), 'gpuArray' );
```
More Examples: GPU Mandelbrot

- Run the script `gpu_demo_Mandelbrot.m`
- `>> GPU version 1: main routine... (almost) same as before!`

```matlab
% Iterate!
% We will count the iteration after which it becomes evident that a series is
% unbounded.
z = z0; % Starting from iteration 1 ( z(0)=0 => z(1)=z0 )
for n = 1:maxIterations
    z = z.*z + z0;
    inside = abs( z )<=2;
    count = count + inside;
end
count = log( count ); % Use log to compress the range, for better plotting

% Show
count = gather( count ); % Fetch the data back from the GPU
naiveGPUTime = toc( t );
```
More Examples: GPU Mandelbrot

- Run the script `gpu_demo_Mandelbrot.m`
- `>>` GPU version 2: using built-in `arrayfun` to optimise
  - First, wrap algorithm into a function

```matlab
% GPU Mandelbrot evaluation function
function count = GPU_mandelbrot(x0,y0,maxIterations)
    z0 = complex(x0,y0);
    z  = z0;
    count = 1;
    while (count <= maxIterations) && (abs(z) <= 2)
        count = count + 1;
        z = z*z + z0;
    end
    count = log(count);  % Use log to compress the range
end
```

- NB. functions declared in scripts must come at the end!
More Examples: GPU Mandelbrot

- Run the script `gpu_demo_Mandelbrot.m`
- `>>` GPU version 2: using built-in `arrayfun` to optimise
  - Then, call using `arrayfun`:

```matlab
...
y = gpuArray.linspace( ylim(1), ylim(2), gridSize );
[xGrid,yGrid] = meshgrid( x, y );

% Calculate
count = arrayfun( @GPU_mandelbrot, ...
    xGrid, yGrid, maxIterations );

% Show
count = gather( count ); % Fetch the data back from the GPU
...```

- Function handle `handle`
- Input arguments

The University of Sydney
More Examples: GPU Mandelbrot

- Run the script `gpu_demo_Mandelbrot.m`
Additional commands

- Cleaning up local
  - These files are also copied to your Matlab working directory, including the Artemis log files if the job completes:
  - E.g.

  Job3.log
  Job3.log_usage
  Task1.diary.txt

Artemis log files ( JobN.log etc )

Matlab command window output from job; will generally be named ‘Task1.diary.txt’
Additional commands

- Cleaning up Artemis
  - Matlab will leave a lot of intermediate files in your Artemis directory. You may remove them.

```matlab
myCluster = parcluster('Artemis');
delete(myCluster.Jobs)

% deletes all jobs and residual data stored on Artemis.
```
Known Errors

- Error using parallel.Job/fetchOutputs (line 1094) Task with ID 1 returned 0 outputs but 5 were expected. Error in batchTomo (line 21) jobOut = fetchOutputs(job);

- Possibly due to not allocating enough memory. Or running the job on 3 separate nodes.
Thank you

- Good stuff!
- For additional help check the user guides on rc.sydney.edu.au
- This is a new service, if you come across things that seem broken – they might be!
Some notes on running via batch
Running via `batch`

- The Matlab `batch` function allows execution of code in the ‘background’, provided there are some workers to perform it.

```matlab
job = batch('script','Profile','local','Pool',4)
```

```matlab
job = batch(@func,2,{5,1.4,0}'Profile','local','Pool',4)
```

- Script to run
- ..or function handle
- Cluster profile to use
- Number of workers on which to run in parallel
- Number of output arguments
- Input arguments (as cell arrays)
Running via **batch**

- The **Matlab batch** function allows execution of code in the ‘background’, provided there are some workers to perform it.

- To use batch, ensure you have a *cluster profile* set.
Running via batch

- The Matlab batch function allows execution of code in the ‘background’, provided there are some workers to perform it.

- To use batch, ensure you have a cluster profile set.

Click ‘Edit’ to change Number of workers to 5 (= 1 client + 4 compute)
Running via `batch`

- The Matlab `batch` function allows execution of code in the ‘background’, provided there are some workers to perform it.

- To use `batch`, ensure you have a `cluster profile` set.

- *And close your current `parpool`, or else the batch executions will never run!*

  ```matlab
  delete(myPool)
  or
  delete(gcp('nocreate'))
  ```
Running via `batch`

```plaintext
job1 = batch(@prime_par, 3, {1, 1e4}, 'Profile', 'local', 'Pool', 4)
wait(job1);
data1 = fetchOutputs(job1)
diary(job1)
```

- Wait for job to complete (optional)
- Retrieve data from job object
- Replay any output from job (optional)
Running via batch

- Get info on the local cluster with `parcluster`:

  ```matlab
  myClust = parcluster('local')
  
  Local Cluster
  Properties:
    Profile: local
    Modified: false
    Host: vlan1765-j12-10-65-21-2
    NumWorkers: 5
    NumThreads: 1

  JobStorageLocation: /Users/hayim/Library/Application Support/MathWorks/MATLAB/local_cluster_jobs/R2017a
  RequiresMathWorksHostedLicensing: false

  Associated Jobs:
    Number Pending: 0
    Number Queued: 0
    Number Running: 3
    Number Finished: 46

  49x1 Job array:
   ID  Type State FinishDateTime  Username  Tasks
    1  pool finished 14-Nov-2017 18:01:11 hayim 5
    2  pool finished 14-Nov-2017 18:01:11 hayim 5
    3  pool finished 14-Nov-2017 18:09:12 hayim 5

due to use of 'pool' command; numWorkwers+1
Running via batch

- Get info on the local cluster with `parcluster`
- Can click on a job ID:
- And/or on tasks links

---

**Job**

Properties:

- ID: 52
- Type: pool
- Username: hayim
- State: finished
- SubmitDateTime: 16-Nov-2017 11:14:07
- StartDateTime: 16-Nov-2017 11:14:20
- Running Duration: 0 days 0h 0m 7s
- NumWorkersRange: [5 5]
- NumThreads: 1

AutoAttachFiles: true
Auto Attached Files: List files
AttachedFiles: {}
AdditionalPaths: {}

**Associated Tasks**:

- Number Pending: 0
- Number Running: 0
- Number Finished: 5
- Task ID of Errors: []
- Task ID of Warnings: []
Running via batch

- Get info on the local cluster with `parcluster`
- Can click on a job ID:
- And/or on tasks links

### Job Properties:

- ID: 52
- Type: pool
- Username: hayim
- State: finished
- SubmitDateTime: 16-Nov-2017 11:14:07
- StartDateTime: 16-Nov-2017 11:14:20
- Running Duration: 0 days 0h 0m 7s
- NumWorkersRange: [5 5]
- NumThreads: 1

### 5x1 Task array:

<table>
<thead>
<tr>
<th>ID</th>
<th>State</th>
<th>FinishDateTime</th>
<th>Function</th>
<th>Errors</th>
<th>Warnings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>finished</td>
<td>16-Nov-2017 11:14:25</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>finished</td>
<td>16-Nov-2017 11:14:27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>finished</td>
<td>16-Nov-2017 11:14:27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>finished</td>
<td>16-Nov-2017 11:14:27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>finished</td>
<td>16-Nov-2017 11:14:27</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Running via **batch**

- Get info on the local cluster with Job Monitor
Running via `batch`

- Get info on the local cluster with **Job Monitor**

![Job Monitor screenshot](image-url)
Running via **batch**

- Get info on the local cluster with **Job Monitor**