Optimising Matlab, GPUs and Raijin
Optimising your Matlab code

1. Register an account on the Mathworks website
   - https://www.mathworks.com/
   - Free access to documentation, instructional videos, examples, benchmarks and discussion groups

2. Read the Matlab documentation
   - Particularly Support » Documentation » Parallel Computing Toolbox » Getting Started

3. Profile your code

4. Check your algorithms

5. Vectorise your code

6. Use GPUs with gpuArray

7. Use parfor, parfeval and spmd

8. Talk to others!
Profile your code

“The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimisation is the root of all evil (or at least most of it) in programming.”


• Where is your code spending its time?
• Use Matlab’s built-in tools
  – profile viewer for a graphical interface
  – tic and toc for basic timing
  – mpiprofile for parallel code
• Focus on areas consuming the majority of time
• Evaluate the effectiveness of your algorithms
• Use built-in functions and toolboxes where possible
  – “Built-in Parallel Computing Support” help topic
Vectorise your code

• Where possible, convert your code from \texttt{for} loops to matrix and vector operations
  – Code looks more like mathematical expressions
  – Code is often shorter
  – Code often runs significantly faster
  – Often a prerequisite for good GPU performance

• Example using vectors:

```python
i = 0;
for t = 0 : 0.01 : 10
  i = i + 1;
  y(i) = sin(t);
end
t = 0 : 0.01 : 10;
y = sin(t);
```

• Example using arrays:

```python
for n = 1 : 10000
  V(n) = 1/12 * pi * (D(n) ^ 2) * H(n));
end
V = 1/12 * pi * (D .^ 2) .* H;
```
Use `parfor`, `parfeval` and friends

- Examine your code for time-consuming `for` loops and replace with `parfor`
  - Each iteration must be *independent* (must not depend on the results) of other iterations
  - Reduction variables (e.g., loop summation) are allowed
  - Converting outer `for` loops work best
    - e.g., Monte Carlo simulations
    - e.g., Parameter sweeps
- Other possibilities: `parfeval`, `distributed`, `datastore`, `mapreduce`, `spmd`
- Can use all cores of a multiprocessor compute node
  - On Raijin: `normal` queue – up to 16 cores; `normalbw` queue – up to 28 cores
- Example:
  ```matlab
  for i = 1 : N
      a(i) = max(abs(eig(rand(A))));
  end
  ```
  ```matlab
  parfor i = 1 : N
      a(i) = max(abs(eig(rand(A))));
  end
  ```
Use GPUs with \texttt{gpuArray}

- Read the “GPU Computing in Matlab” help topic
- Check GPU device capabilities with \texttt{gpuDevice}
- Use \texttt{gpuArray} to create arrays on or copy arrays to the GPU
- Use \texttt{gather} to copy arrays back from the GPU
- Check which inbuilt functions can run on the GPU
  - “Run Built-In Functions on a GPU” help topic
  - Currently 344 intrinsic functions
- Profile using \texttt{tic}, \texttt{toc}, \texttt{gputimeit}
- Consider using single-precision calculations
- Example:

  \begin{verbatim}
  r_cpu = rand(1024);
  r_gpu = rand(1024, 'gpuArray');
  m = eig(r_gpu);
  isOnGPU(m)
  \end{verbatim}

  \% \texttt{r_cpu} array is on the CPU
  \% \texttt{r_gpu} array is on the GPU
  \% Do a calculation on the GPU
  \% \texttt{isOnGPU} returns 1 (true)
Using GPUs on Raijin

- Raijin currently has
  - **gpu** queue:
    - 30 nodes of four Nvidia K80 accelerators (eight GPUs) each,
    - up to 2.91 teraFLOPS double-precision performance per GPU,
    - up to 8.73 teraFLOPS single-precision performance per GPU,
    - 18 SU (72¢ in-kind contribution) per hour per GPU
  - **gpupascal** queue:
    - 2 nodes of four Nvidia P100 GPUs each,
    - up to 5.3 teraFLOPS double-precision performance per GPU,
    - up to 10.6 teraFLOPS single-precision performance per GPU,
    - 24 SU (96¢ in-kind contribution) per hour per GPU
- Develop code on your workstation or desktop computer
  - Can also use Raijin interactively! Use “**ssh -Y**” or **MobaXterm** with inbuilt X server
    ```
    qsub -q gpu -l ngpus=2 -l ncpus=6 -l software=matlab_unsw \ 
    -l walltime=0:30:00 -l mem=32GB -I -X 
    module load matlab
    matlab &
    ```
Why use Raijin? Scale, performance, cost

- Desktop computer with Nvidia GTX760: `paralleldemo_gpu_backslash(0.6)`

- Raijin K80 node (gpu queue), one GPU: `paralleldemo_gpu_backslash(6.0)`
Talk to us!

- **John Zaitseff**  
  *Research Computing Support Engineer*  
  [J.Zaitseff@unsw.edu.au](mailto:J.Zaitseff@unsw.edu.au)

- **Joachim Mai**  
  *HPCD Manager*  
  [Joachim.Mai@unsw.edu.au](mailto:Joachim.Mai@unsw.edu.au)

- **Luc Betbeder-Matibet**  
  *Director, Research Technology Services*  
  [luc@unsw.edu.au](mailto:luc@unsw.edu.au)

- **Research Technology Services**  
  *DVC Research Infrastructure*  
  UNSW Sydney  
  [https://research.unsw.edu.au/research-technology-services](https://research.unsw.edu.au/research-technology-services)