

Intro to Parallel Computing with MATLAB

Peter Brady

Housekeeping

- Need a free MathWorks account
 - <https://au.mathworks.com/login>
- Using custom classroom
 - https://www.mathworks.com/licensecenter/classroom/PC_3468800/



MathWorks® Products Solutions Academia Support Community Events

MATLAB & Simulink

Access MATLAB for your Parallel Computing Workshop

MathWorks is pleased to provide a special license to you as a course participant to use for your Parallel Computing Workshop. This is a limited license for the duration of your course and is intended to be used only for course work and not for government, research, commercial, or other organization use.

Course Name:	Parallel Computing with MATLAB at ANU
Organization:	MathWorks Parallel Computing
Ending:	21 Jul 2021

[Access MATLAB Online](#)

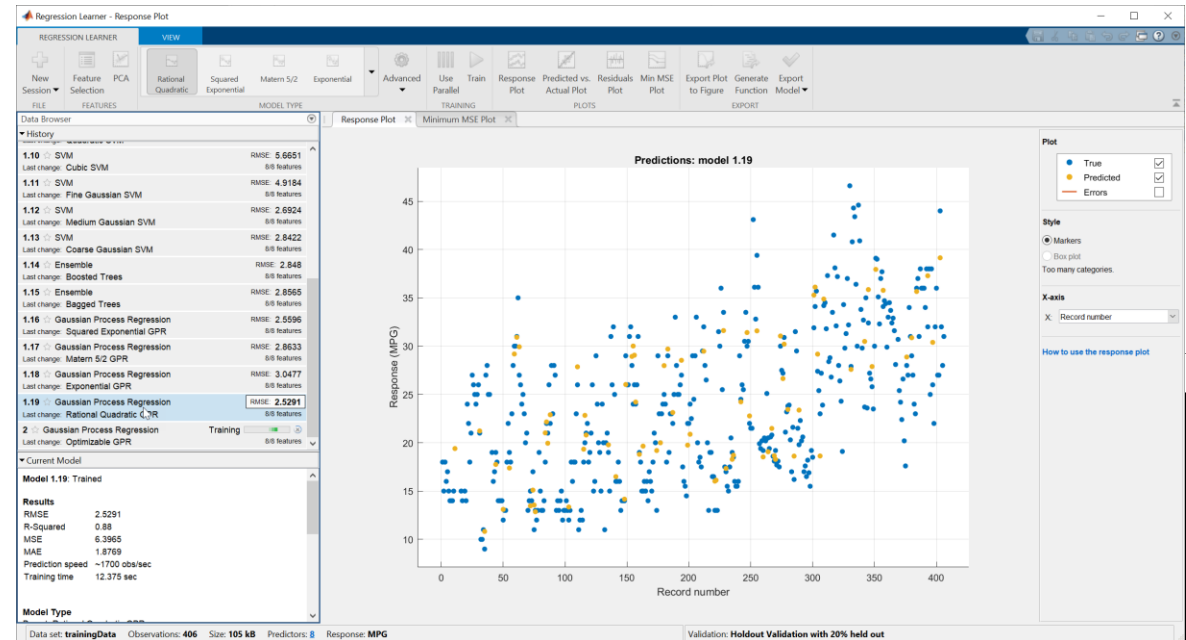
Demo: take app gen code through to NCI

- Process

1. App to train on small subset
2. Generate Code
3. Tweak code
4. Write wrapper
5. Deploy

- Top tips

- a) Double check version: R2020b??
- b) When scaling use hold out validation
- c) Scale out slowly
- d) Write a robust script to check in advance



```

=====
[pb8519@gadi-login-05 carExample]$ ls
25818331.gadi-pbs.log  carFull.e25818331  modelTrainingOutput.mat  trainRegressionModel.m
buildModel_final.m   carFull.o25818331  submit.sh
buildModel_mvp.m     cardata.parquet    submitFull.sh
[pb8519@gadi-login-05 carExample]$ rm *
[pb8519@gadi-login-05 carExample]$ ls
buildModel_final.m  cardata.parquet    submitFull.sh
buildModel_mvp.m    submit.sh           trainRegressionModel.m
[pb8519@gadi-login-05 carExample]$ chmod 755 submit*
[pb8519@gadi-login-05 carExample]$ ls
buildModel_final.m  cardata.parquet    submitFull.sh
buildModel_mvp.m    submit.sh           trainRegressionModel.m
[pb8519@gadi-login-05 carExample]$ vi buildModel_final.m
[pb8519@gadi-login-05 carExample]$ ls
buildModel_final.m  cardata.parquet    submitFull.sh
buildModel_mvp.m    submit.sh           trainRegressionModel.m
[pb8519@gadi-login-05 carExample]$

```

Dr Peter Brady

pbrady@mathworks.com



- BE, PhD, CPEng NER, ACS CP
- Research areas:
 - Fluid and Thermodynamics
 - High performance computing
 - Nanofluidics
 - Visitor UTS FEIT
- Industry Experience
 - Defence sub-contractor in CFD cavitation simulation
 - Civil consultant in surface water, hydrology, hydraulics
 - Specialist HPC systems administrator
- MathWorks Specialties
 - Core maths, statistics and optimization
 - Machine Learning and Deep Learning
 - Predictive Maintenance
 - High performance and scale out
 - Deployment and cloud
- Key Industries and Applications
 - Metals, Minerals and Mining
 - Quantitative Finance and Risk
 - Energy Production
 - Rail and Utilities
 - Civil and Environmental

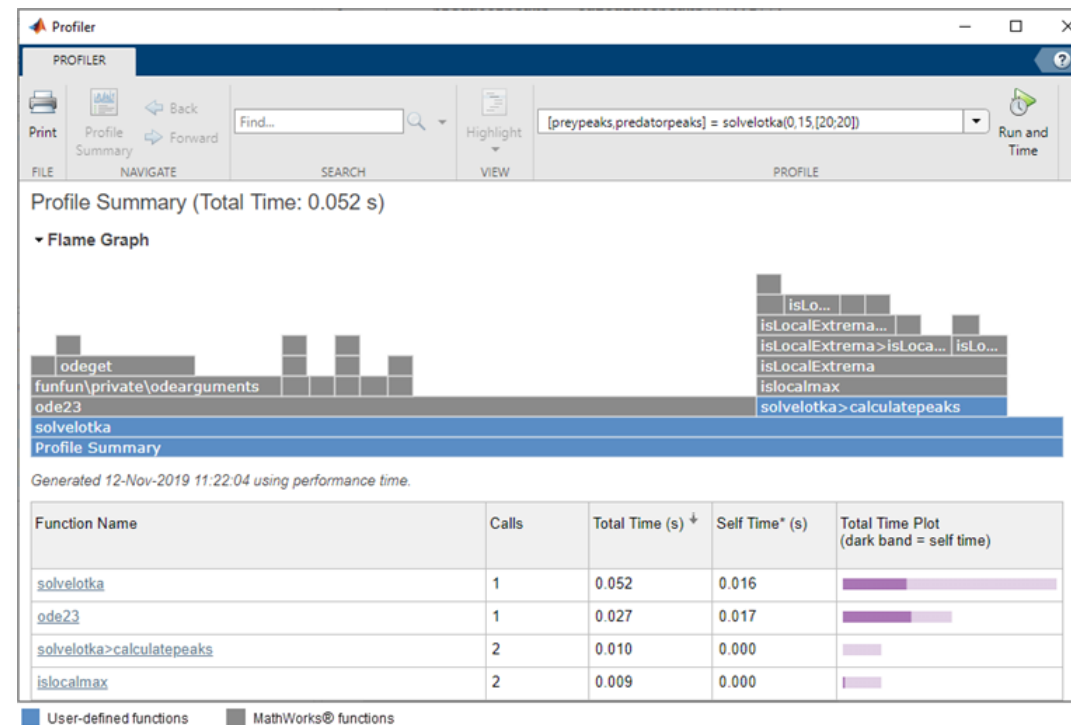
Agenda

1. Get your serial code right first
2. Parallel computing with MATLAB
3. GPU Computing with NVIDIA
4. Parallel computing with Simulink
5. Tackling the challenge of big data

Get Your Serial Code Right First

Optimize your code before parallelizing for best performance

- Find bottlenecks by profiling your code



Optimize your code before parallelizing for best performance

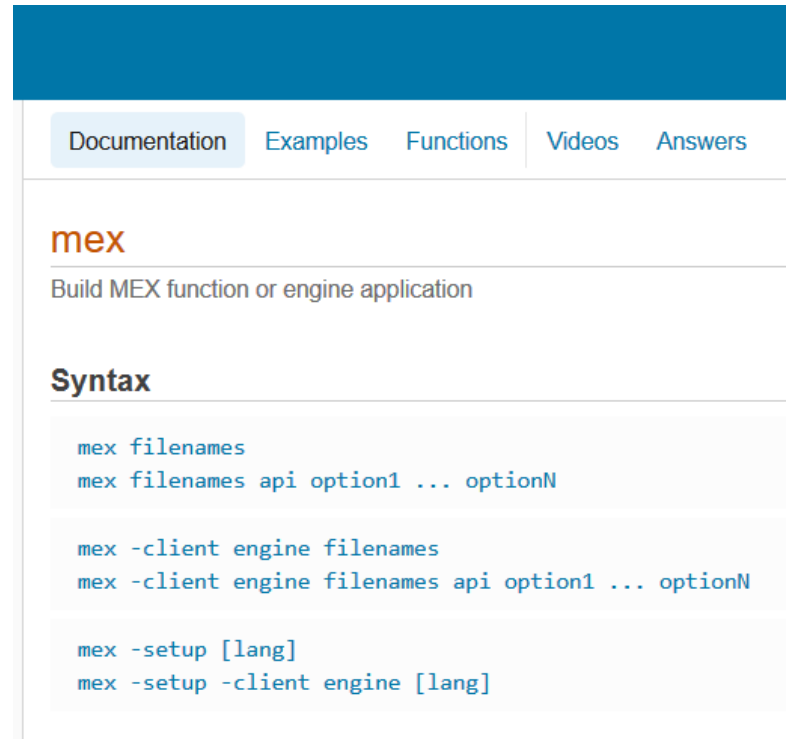
- Implement effective programming techniques

```
59 | % Process each image using preprocessImage
60 | for imgInd = 1:numel(imds.Files)
61 |     fprintf('Processing image %i', imgInd)
62 |     inImageFile = imds.Files{imgInd};
63 |     t(imgInd) = imgDep + imgInd;
64 |     % Output has the same sub-directory structure as input
65 |     % outImageFileWithExtension = strrep(inImageFile, inDir, outDir);
66 |     [~,name,ext] = fileparts(inImageFile);
67 |     outImageFileWithExtension = fullfile(tempdir, [name ext]);
68 |     % Remove the file extension to create the template output file name
69 |     [path, filename,~] = fileparts(outImageFileWithExtension);
70 |     outImageFile = fullfile(path,filename);
- |
```

⚠ Line 63: The variable 't' appears to change size on every loop iteration. Consider preallocating for speed. [Details](#)

Optimize your code before parallelizing for best performance

- (Advanced) Replace code with MEX functions



The screenshot shows the MATLAB documentation page for the `mex` function. The page has a blue header bar and a navigation menu with tabs for Documentation, Examples, Functions, Videos, and Answers. The `mex` title is in orange. Below the title is the description "Build MEX function or engine application". The "Syntax" section is highlighted with a horizontal line and contains three code blocks:

```
mex filenames
mex filenames api option1 ... optionN
```

```
mex -client engine filenames
mex -client engine filenames api option1 ... optionN
```

```
mex -setup [lang]
mex -setup -client engine [lang]
```

My top tips for your serial code

1. Update your product version
2. Read the documentation
3. Use functions
4. Pre-allocate RAM
5. Vectorise
6. Use loops effectively

[Documentation](#)[Examples](#)[Functions](#)[Videos](#)[Answers](#)

Techniques to Improve Performance

To speed up the performance of your code, consider these techniques.

Environment

Be aware of background processes that share computational resources and decrease performance.

Code Structure

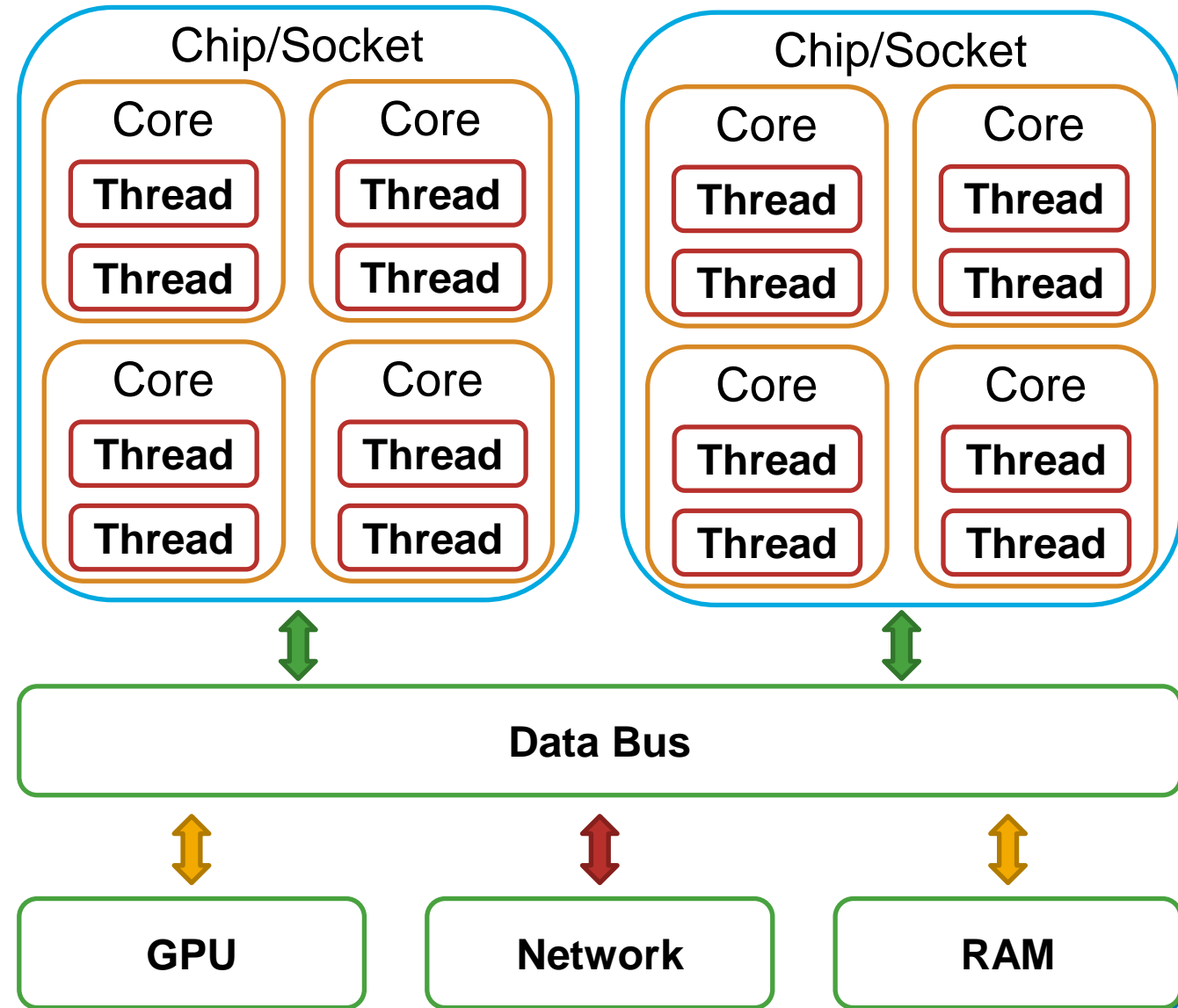
While organizing your code:

- Use functions instead of scripts. Functions are generally faster.
- Prefer local functions over nested functions. Use this practice especially if the function is called frequently.
- Use modular programming. To avoid large files and files with infrequently accessed code.

Know the jargon before you start

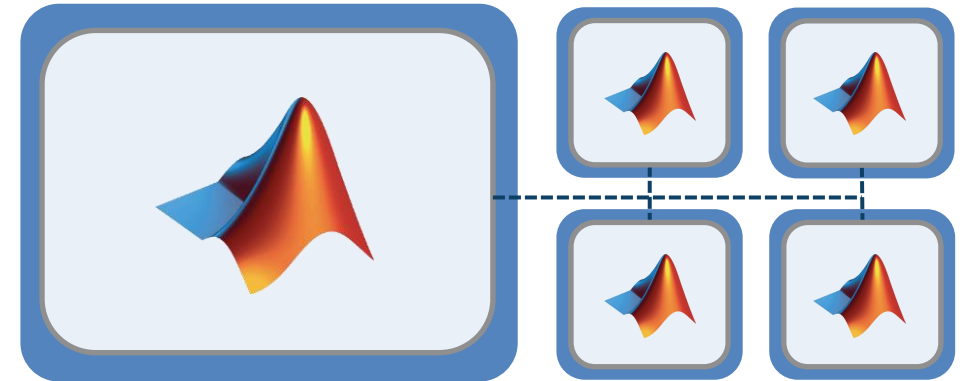
Terms and Jargon: Hardware

- Chips sold by:
 - Cores
 - Threads – usually 2 x cores
- Installed in Sockets
- One FPU per core**
- Memory
 - In chip is very fast
 - In GPU is very fast
 - In RAM is fast
 - Between these is *slowwww*
- AWS and Azure sell
 - vCPUs == threads

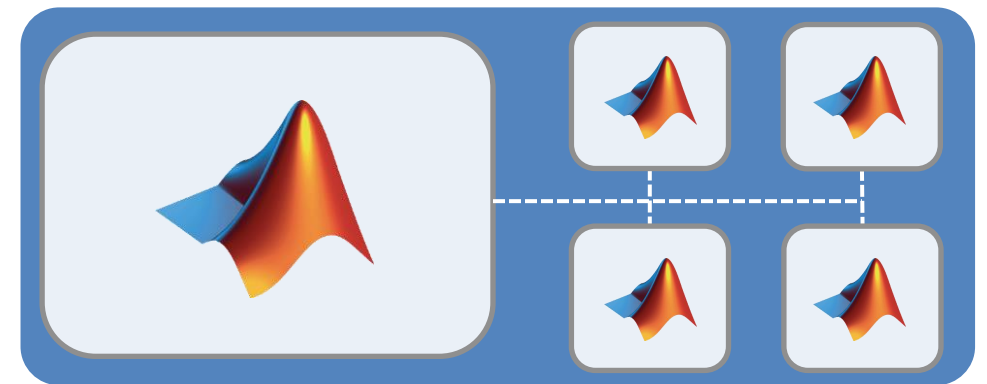


Terms and Jargon: MATLAB and Simulink

- Workers do the compute
- Types of workers:
 - Local
 - Threads



```
>> parpool("local")
```



```
parpool("threads")
```

[Choose between thread-based and process-based environments](#)

Parallel With MATLAB

Accelerating MATLAB and Simulink Applications



Ease of Use

Parallel-enabled toolboxes

```
('UseParallel', true)
```

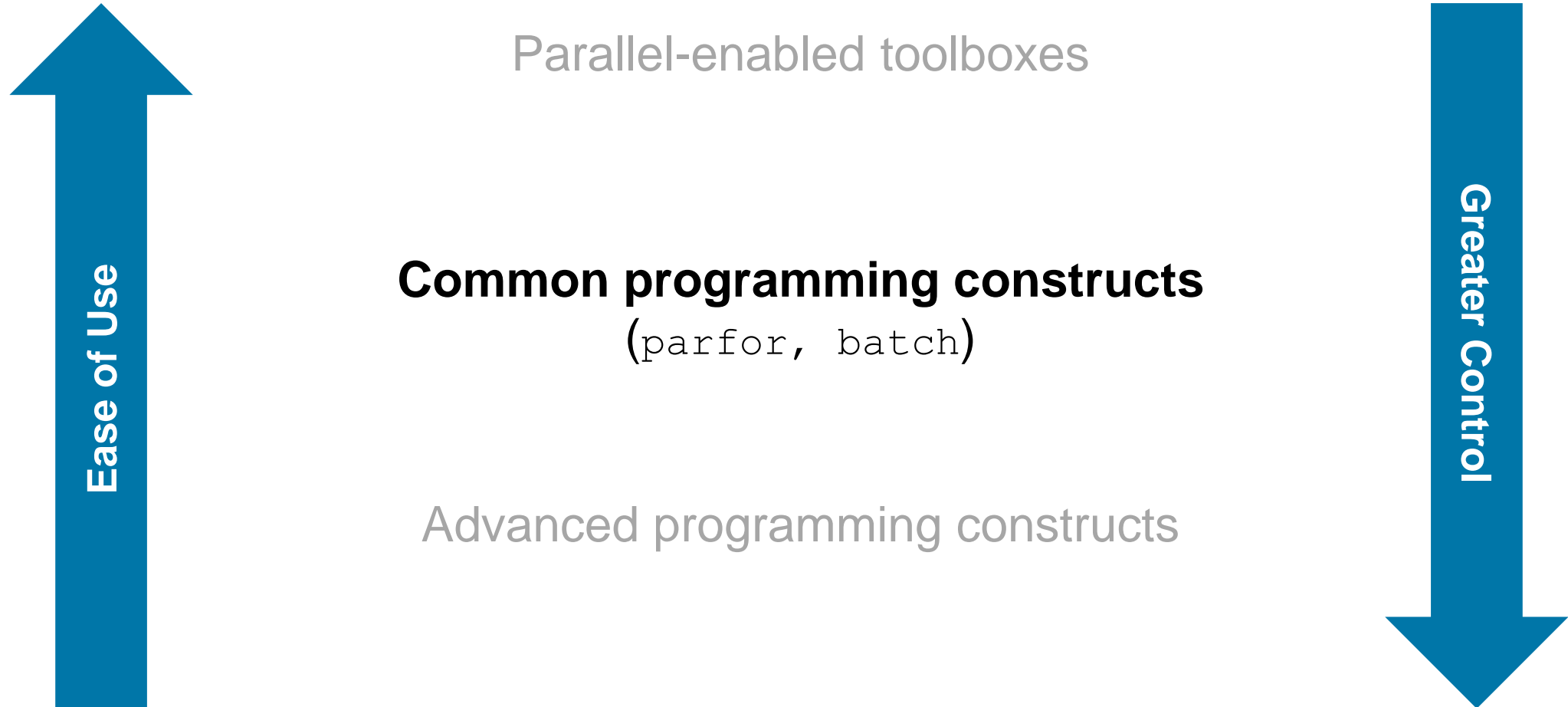
Common programming constructs

Advanced programming constructs



Greater Control

Accelerating MATLAB and Simulink Applications



Key commands that we are going to use

- parfor
- parfeval
- batch

The screenshot displays the MATLAB R2021a Live Editor environment. The main window shows a document titled "parpool Introduction Exercise" with the following content:

parpool Introduction Exercise
The purpose of this exercise is to learn what is a parallel pool and the different ways to use it in MATLAB. To get started, run the following command

```
1 doc parpool
```

Open and close a parallel pool programmatically
Open a parallel pool of 2 local workers through the command line.

```
2 % Enter code here
```

What happens if you try to run the above command twice?

Open and close a parallel pool interactively
Hint: Look at the bottom left corner of the MATLAB desktop.

Query the number of workers in the parallel pool
Hint: Use `gpc` to query the state of the parallel pool.

```
3 % Enter code here
```

Shutdown the parallel pool

```
4 % Enter code here
```

Copyright 2021 The MathWorks, Inc.

The Command Window at the bottom is empty, showing the prompt `fx >>`. The workspace on the right is also empty.

Accelerating MATLAB and Simulink Applications



Parallel-enabled toolboxes

Common programming constructs

Advanced programming constructs
(`spmd`, `createJob`, `labSend`, ..)

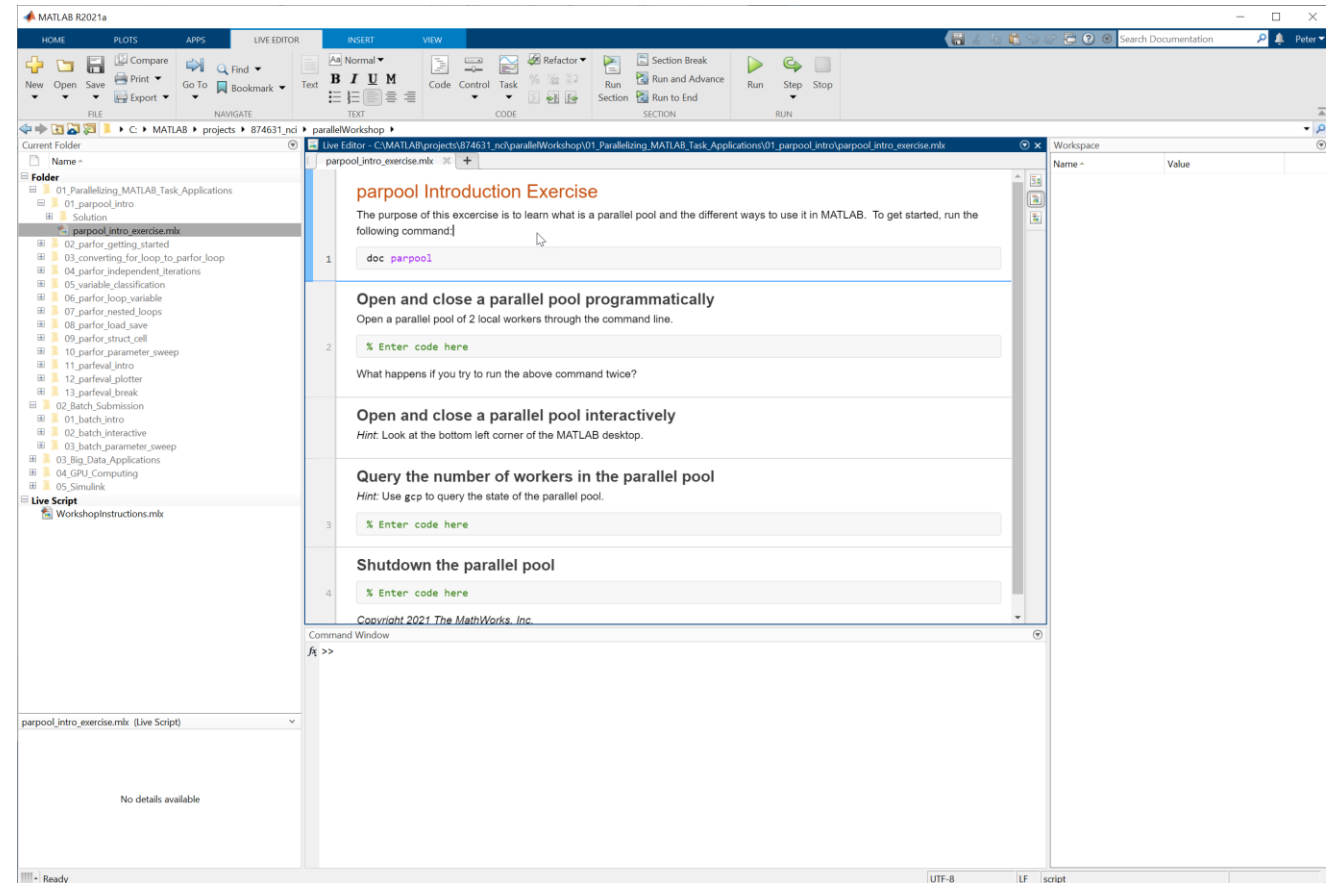


Progression

- First sorry, this is how I normally run this not sure what is wrong today
- Option 1: Pull into MATLAB online or desktop (should work on both)
 - `downloadedFile = websave('ExerciseFiles.zip', 'https://auaeg.s3.ap-southeast-2.amazonaws.com/parallelWorkshop.zip')`
 - `unzip(downloadFile)`
 - `cd parallelWorkshop`
- Option 2: can pull from MATLAB drive
 - <https://drive.matlab.com/sharing/90faf62b-c2a8-4c68-8022-2a4769a57410>
- Option 3: you can download the exercise files in a browser for later
 - <https://auaeg.s3.ap-southeast-2.amazonaws.com/parallelWorkshop.zip>

Key commands that we are going to use

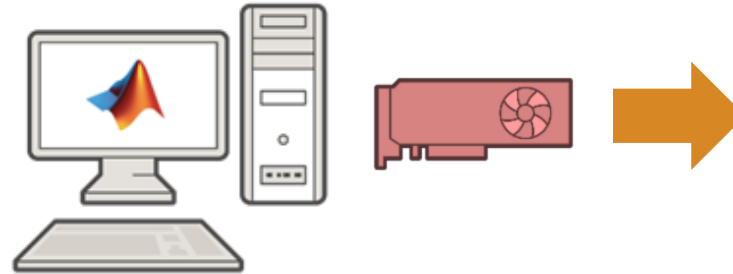
- `spmdd`
- `createJob`
- `labSend`
- `arrayfun`
- `CUDAKernel`



GPU Acceleration

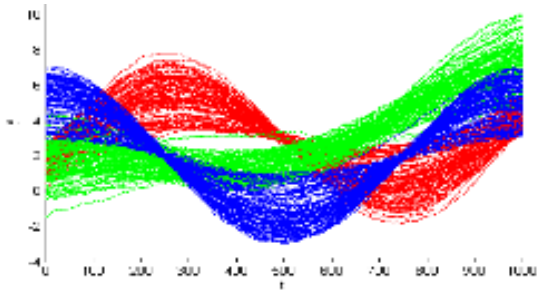
GPU Computing Paradigm

NVIDIA CUDA-enabled GPUs

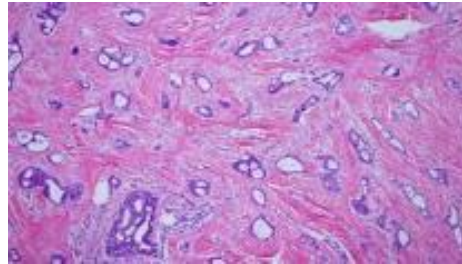


Parallel Computing Toolbox

Speeding up MATLAB applications with GPUs



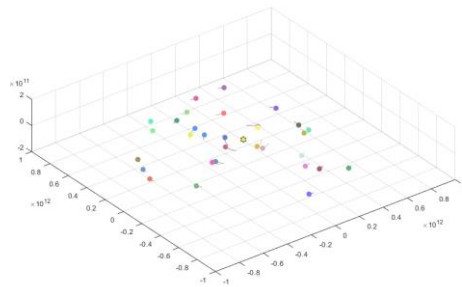
10x speedup
K-means clustering algorithm



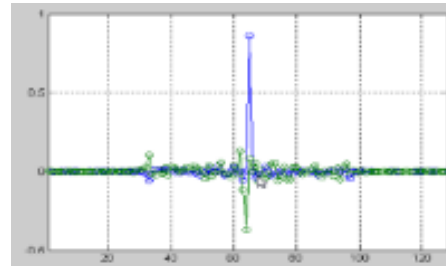
14x speedup
template matching routine



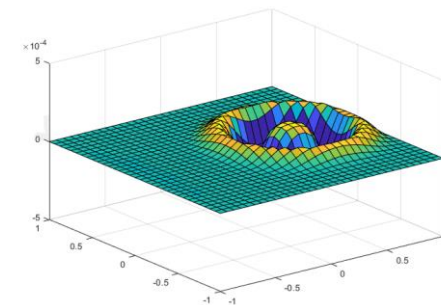
12x speedup
using Black-Scholes model



44x speedup
simulating the movement of celestial objects



4x speedup
adaptive filtering routine

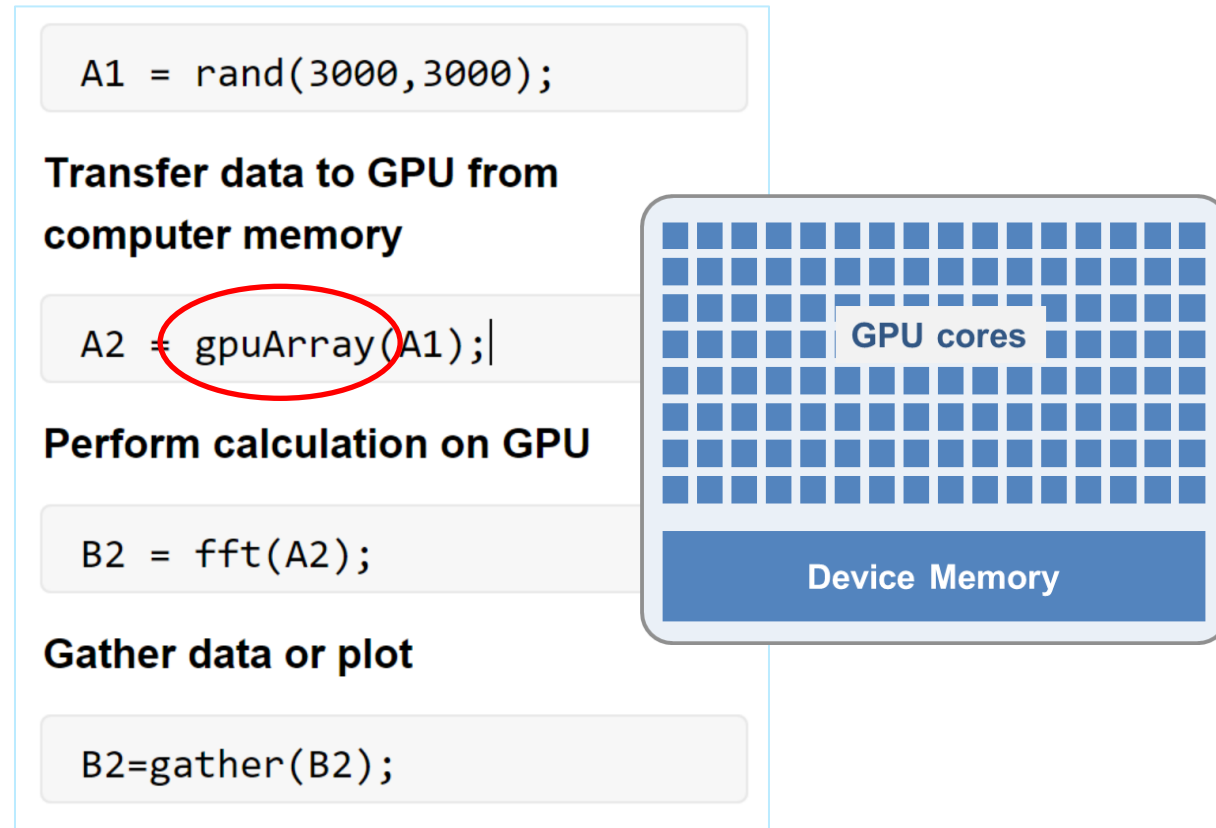


77x speedup
wave equation solving

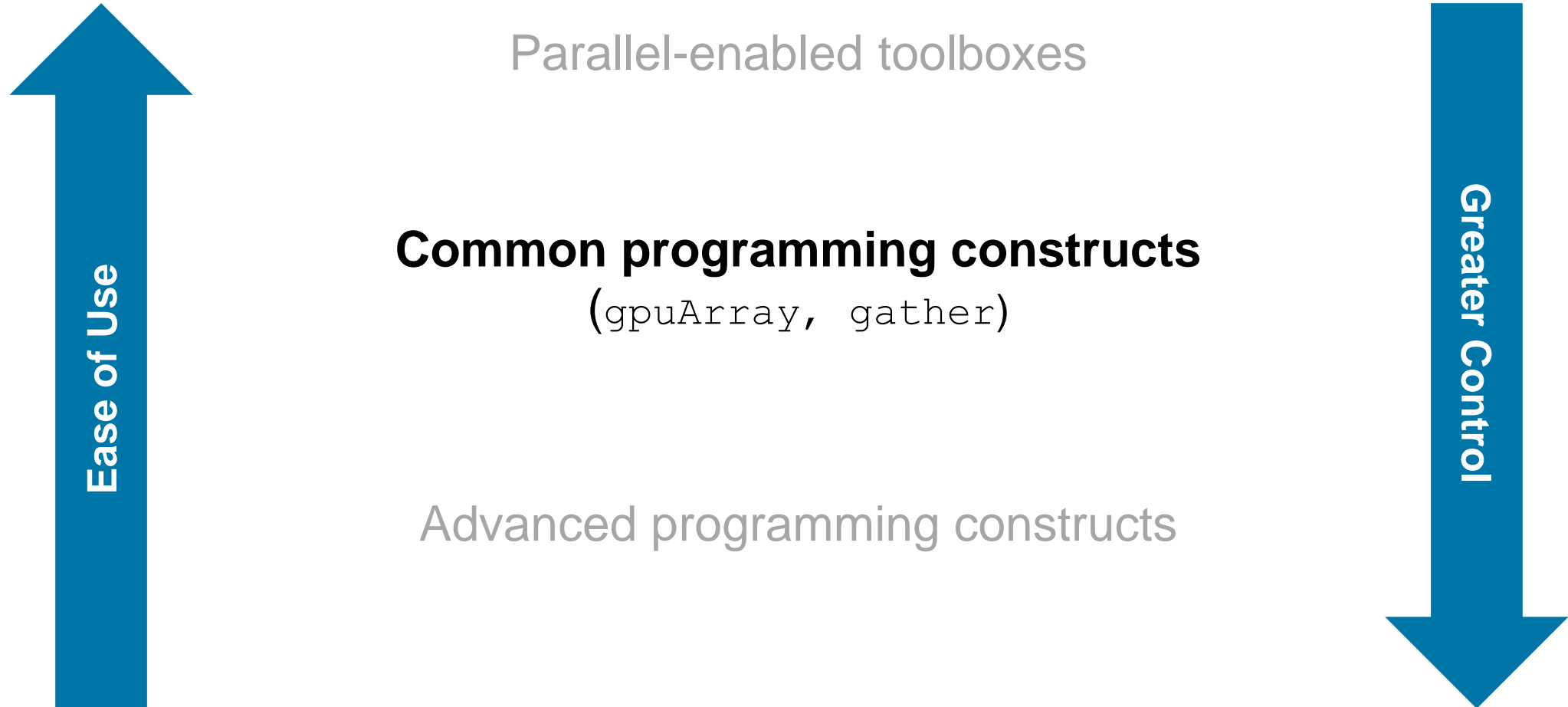
NVIDIA Titan V GPU, Intel® Core™ i7-8700T Processor (12MB Cache, 2.40GHz)

Speed-up using NVIDIA GPUs

- Ideal Problems
 - Massively Parallel and/or Vectorized operations
 - Computationally Intensive
- 500+ GPU-enabled MATLAB functions
- Simple programming constructs
 - `gpuArray`, `gather`



Programming with GPUs



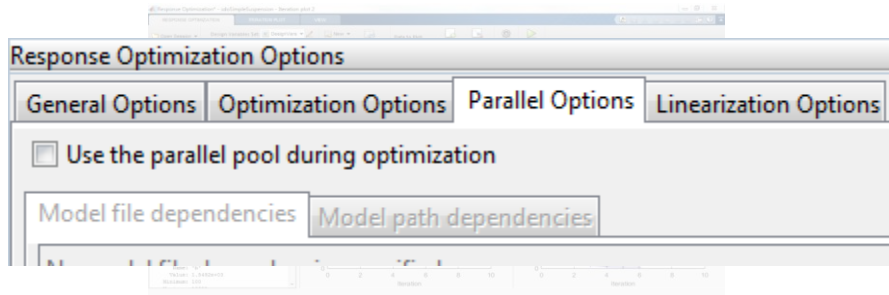
Accelerating Simulink

Enable parallel computing support by setting a flag or preference

Automatic parallel support (*Simulink*)

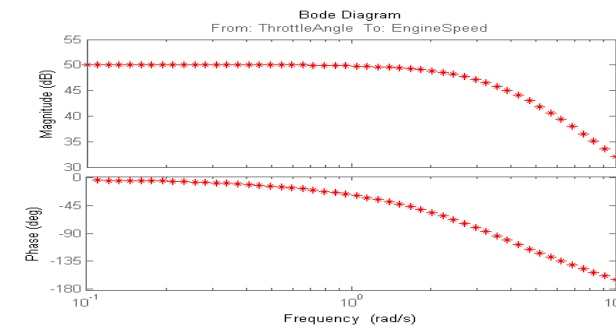
Simulink Design Optimization

Response optimization, sensitivity analysis, parameter estimation



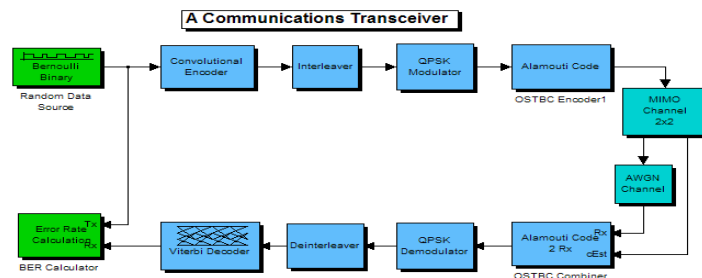
Simulink Control Design

Frequency response estimation



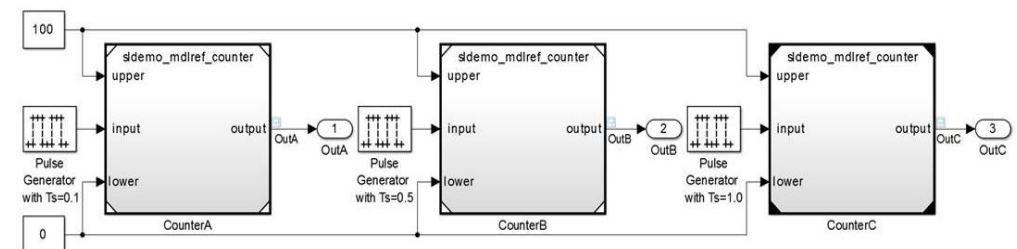
Communication Systems Toolbox

GPU-based System objects for Simulation Acceleration



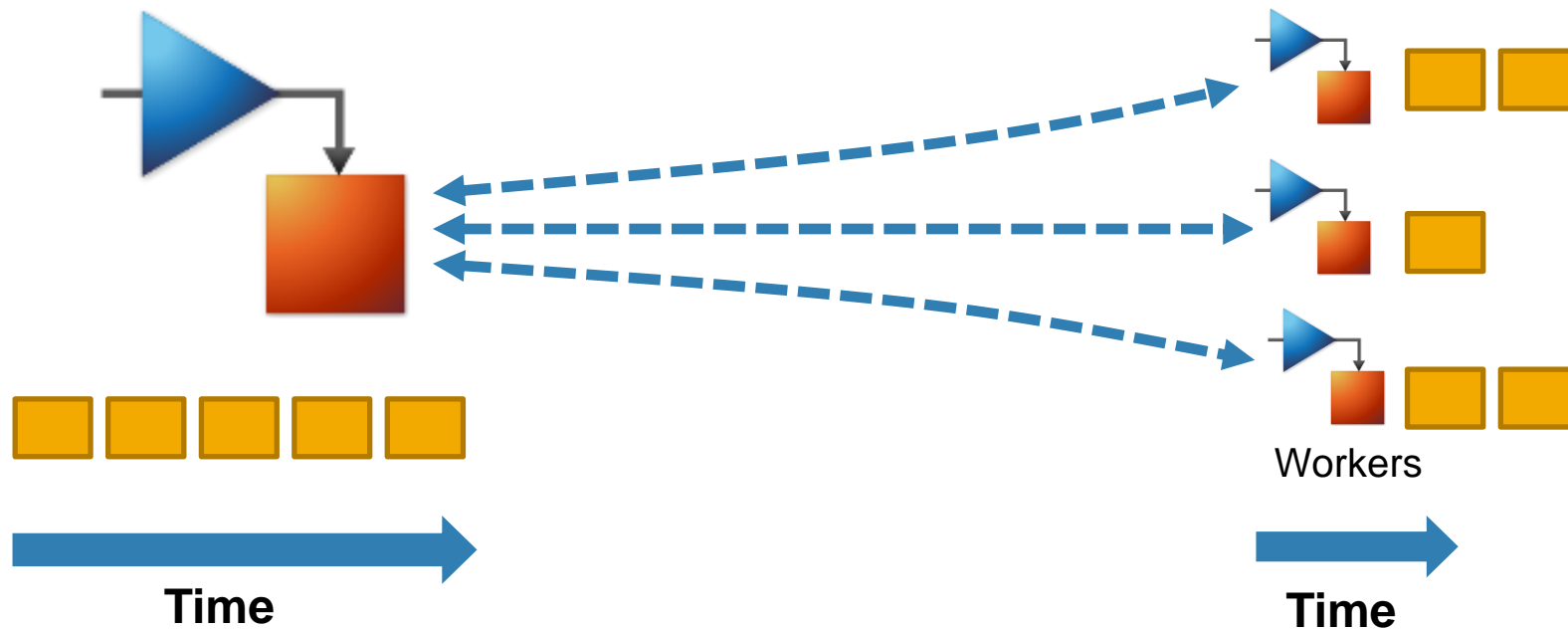
Simulink/Embedded Coder

Generating and building code



[Additional automatic parallel support](#)

Run multiple Simulink simulations in parallel with `parsim`

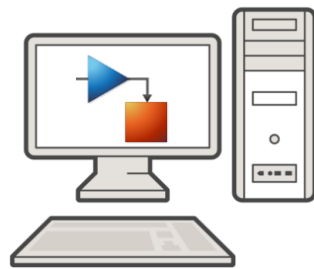


- Run independent Simulink simulations in parallel using the `parsim` function

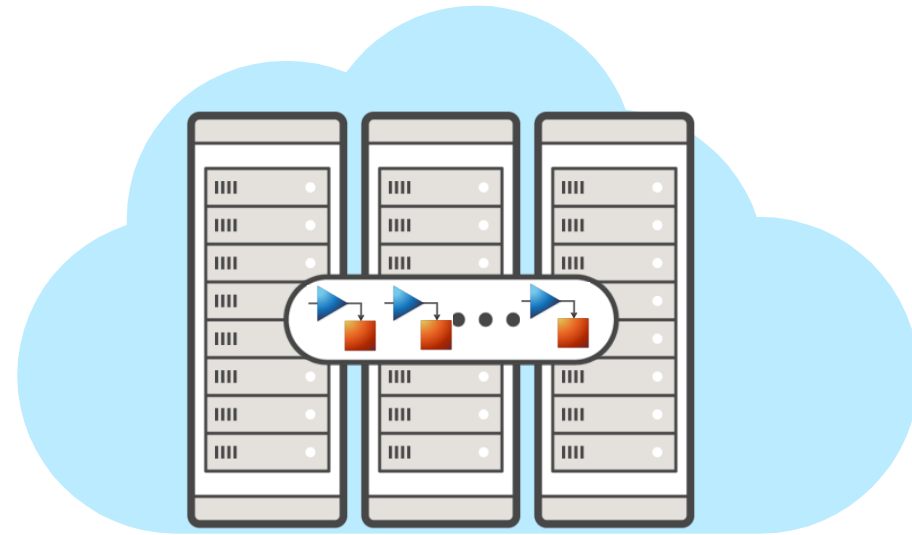
```
for i = 10000:-1:1
    in(i) = Simulink.SimulationInput(my_model);
    in(i) = in(i).setVariable(my_var, i);
end
out = parsim(in);
```

Benefits of using `parsim`

- Run multiple simulations on your machine or clouds and clusters
- Transfer base workspace variables to workers
- Automatically transfer all files to workers
- Automatically return file logging data
- Automatically manage build folders
- Display progress
- Manage errors



Desktop

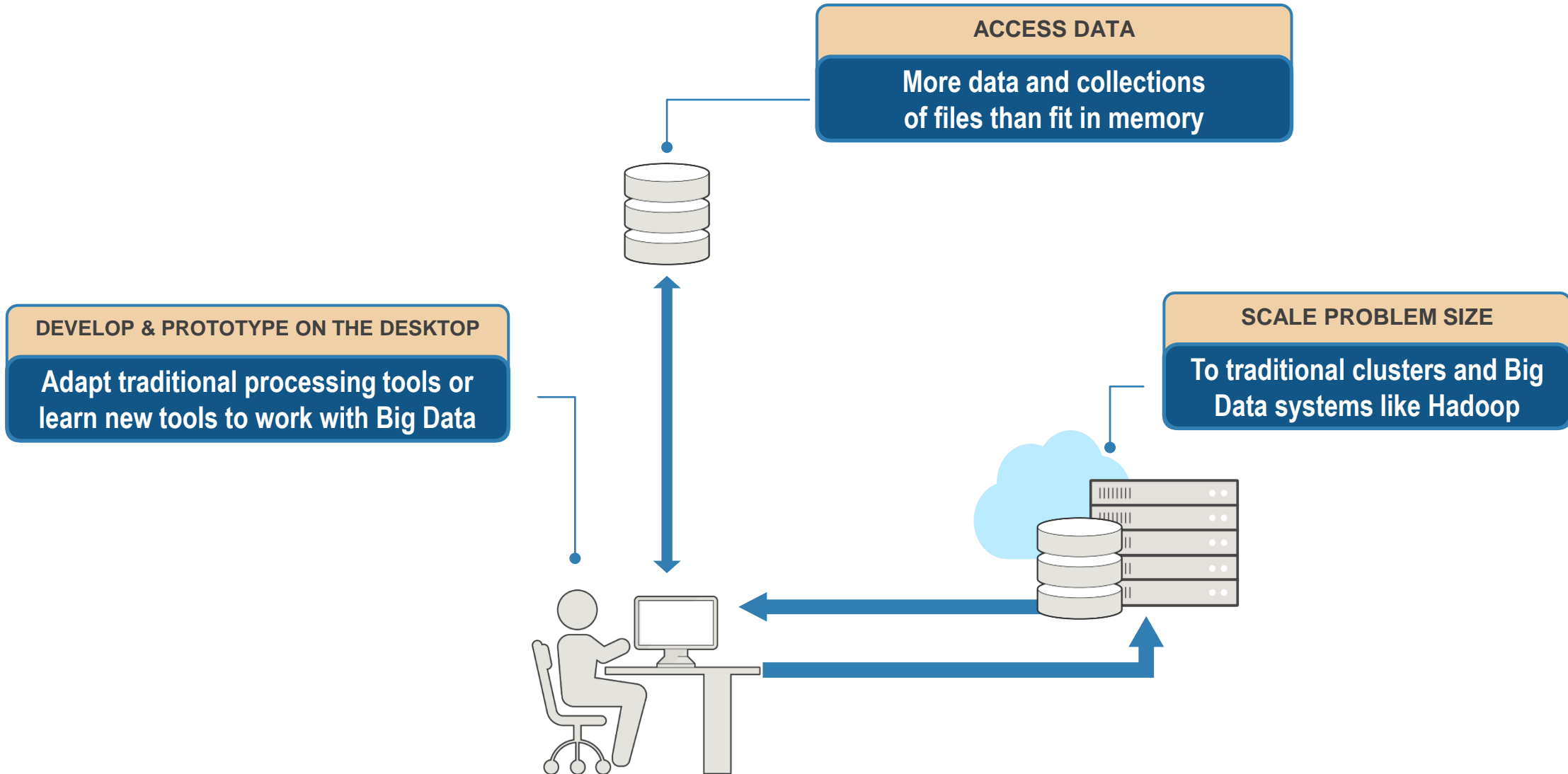


Multicore

Cluster

Big Data

Big data workflows



Access data with datastore

- **For:**
 - Handling collections of files or large files
- **Provides:**
 - Preview and configure I/O properties
 - Read data into memory (*all at once, or incrementally*)
 - Transform data one file at a time for data engineering workflows
 - Combine with tall arrays to analyze the entire out-of-memory dataset with few code changes

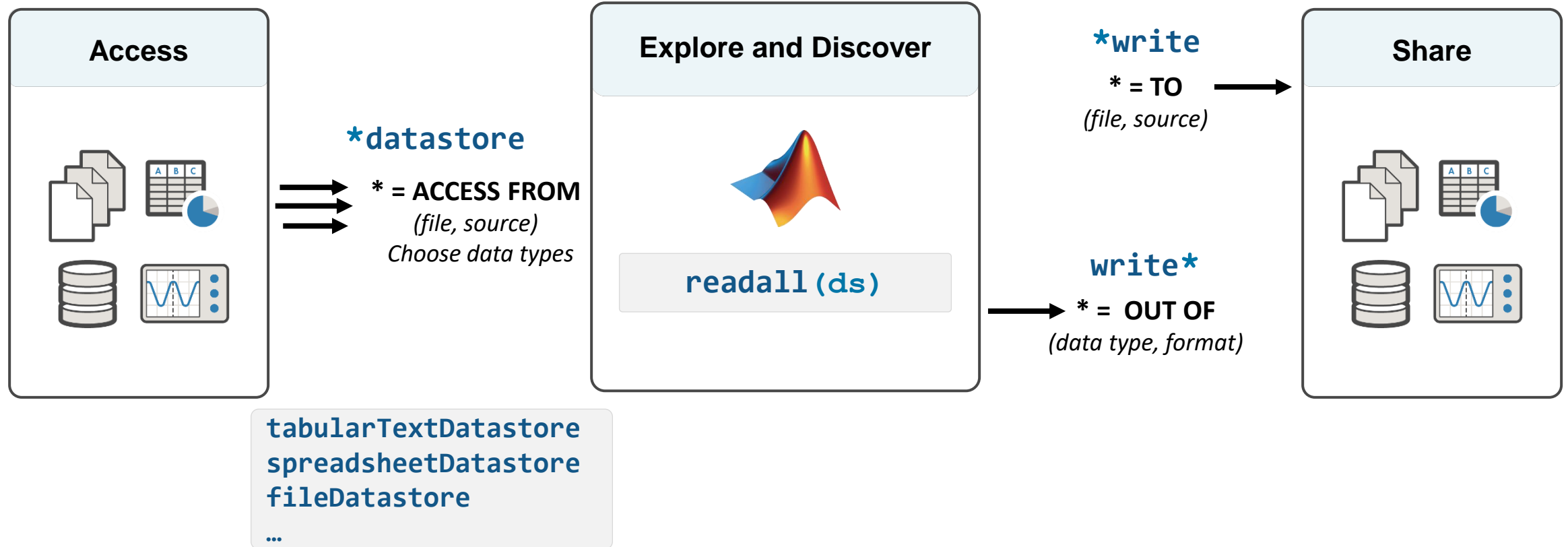
```
ds = datastore("data_flat/*.csv")

    Files: {
        'C:\Marketing\WhatsNew\datastores\data_1'
        'C:\Marketing\WhatsNew\datastores\data_2'
        'C:\Marketing\WhatsNew\datastores\data_3'
        ... and 86 more
    }
    FileEncoding: 'UTF-8'
AlternateFileSystemRoots: {}
    PreserveVariableNames: false
    ReadVariableNames: true
    VariableNames: {'TimeStamp', 'TimeZone', 'Name' ... and 86 more}
    DatetimeLocale: en_US

Text Format Properties:
    NumHeaderLines: 0
    Delimiter: ','
    RowDelimiter: '\r\n'
    TreatAsMissing: ''
    MissingValue: NaN

Advanced Text Format Properties:
    TextscanFormats: {'%{MM/dd/yyyy HH:mm:ss}D', '%q', '%q'}
```

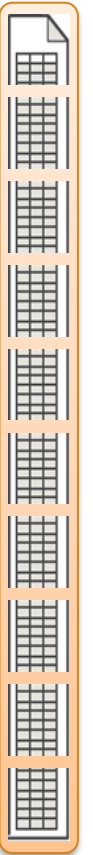

Use datastores for reading collections of files into memory



[Select Datastore for File Format or Application](#)

tall arrays

- New data type designed for data that doesn't fit into memory
- Lots of observations (hence “tall”)
- Looks like a normal MATLAB array
 - Supports numeric types, tables, datetimes, strings, etc.
 - Supports several hundred functions for basic math, stats, indexing, etc.
 - Statistics and Machine Learning Toolbox support (clustering, classification, etc.)

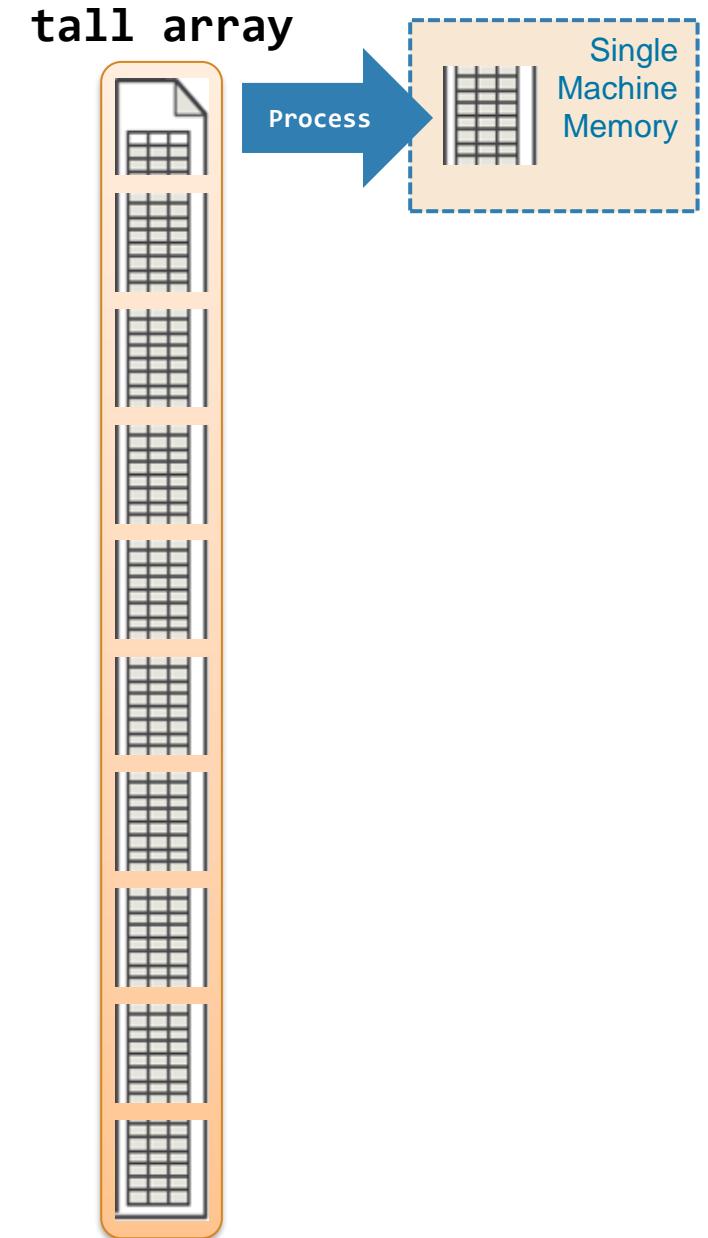
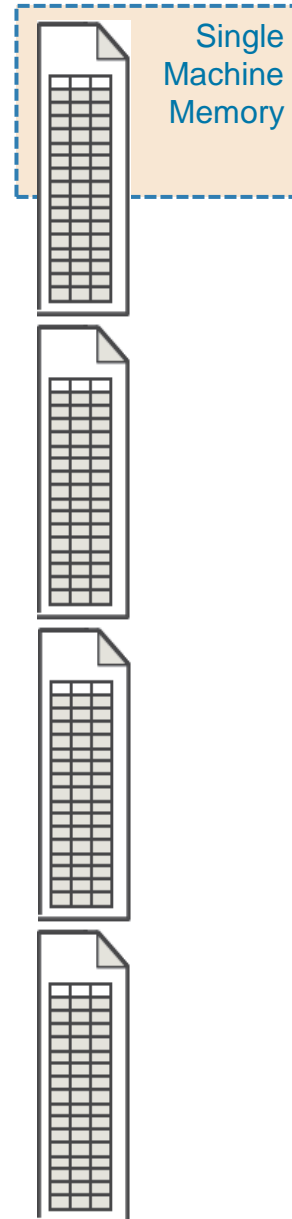


tall arrays

- Automatically breaks data up into small “chunks” that fit in memory
- Tall arrays scan through the dataset one “chunk” at a time

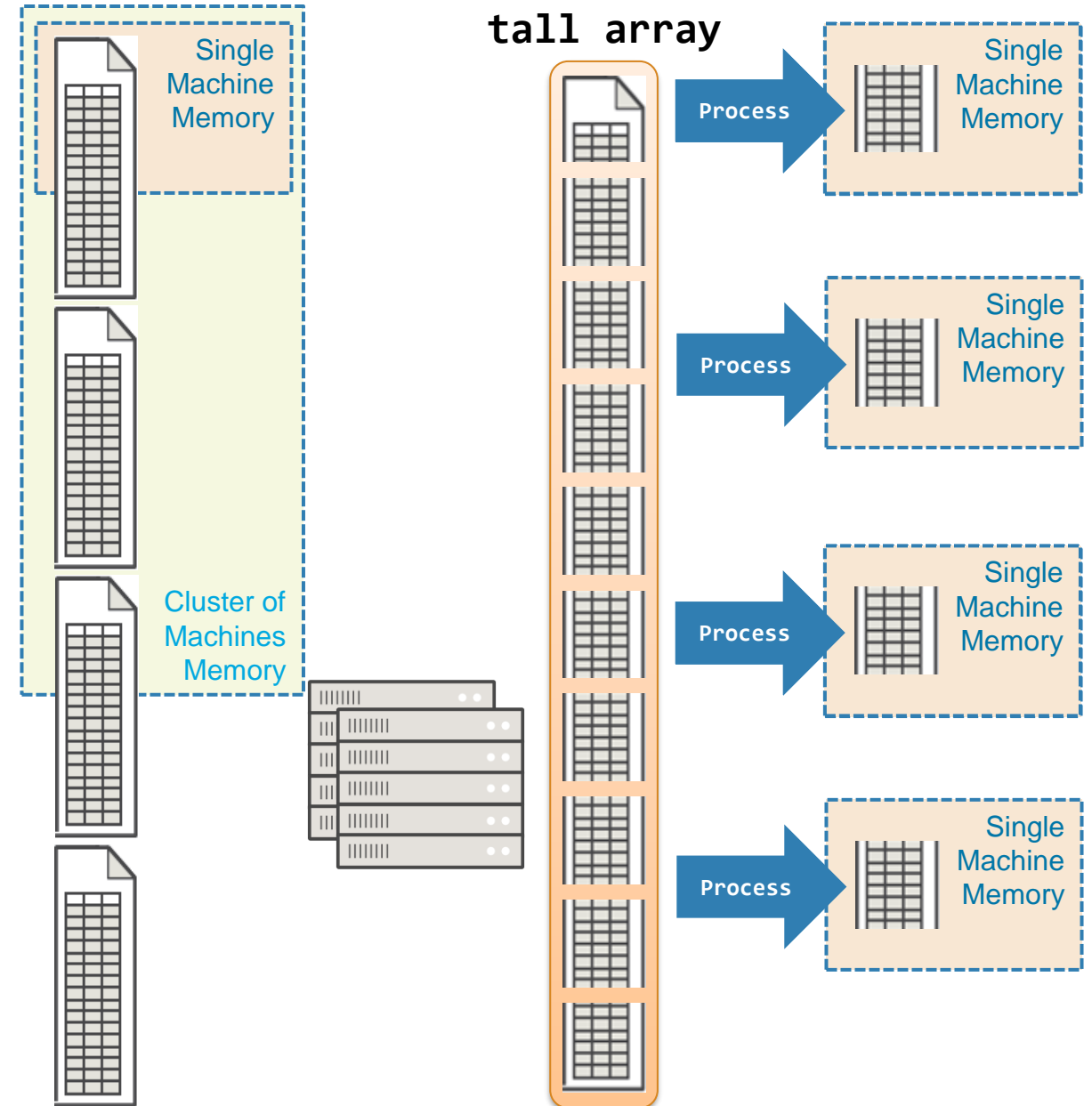
```
tt = tall(ds)
mDep = mean(tt.DepDelay, 'omitnan')
mDep = gather(mDep)
```

- Processing code for tall arrays is the same as ordinary arrays



tall arrays

- With Parallel Computing Toolbox, process several “chunks” at once
- Can scale up to clusters with MATLAB Parallel Server



Summary

- Squeeze your serial code first
- Use the appropriate technology:
 - Cluster
 - Threads
 - GPU
- With appropriate tools:
 - parfor
 - parfeval
 - Batch
- Test as you go, scaling slowly