Parallel and Distributed Computing with MATLAB

Gerardo Hernández

Manager, Application Engineer

Practical Application of Parallel Computing

- Why parallel computing?
  - Need faster insight on more complex problems with larger datasets
  - Computing infrastructure is broadly available (multicore desktops, GPUs, clusters)

- Why parallel computing with MATLAB
  - Leverage computational power of more hardware
  - Accelerate workflows with minimal to no code changes to your original code
  - Focus on your engineering and research, not the computation

Steps for Improving Performance

- First get code working
- Speed up code with core MATLAB
- Include compiled languages and additional hardware

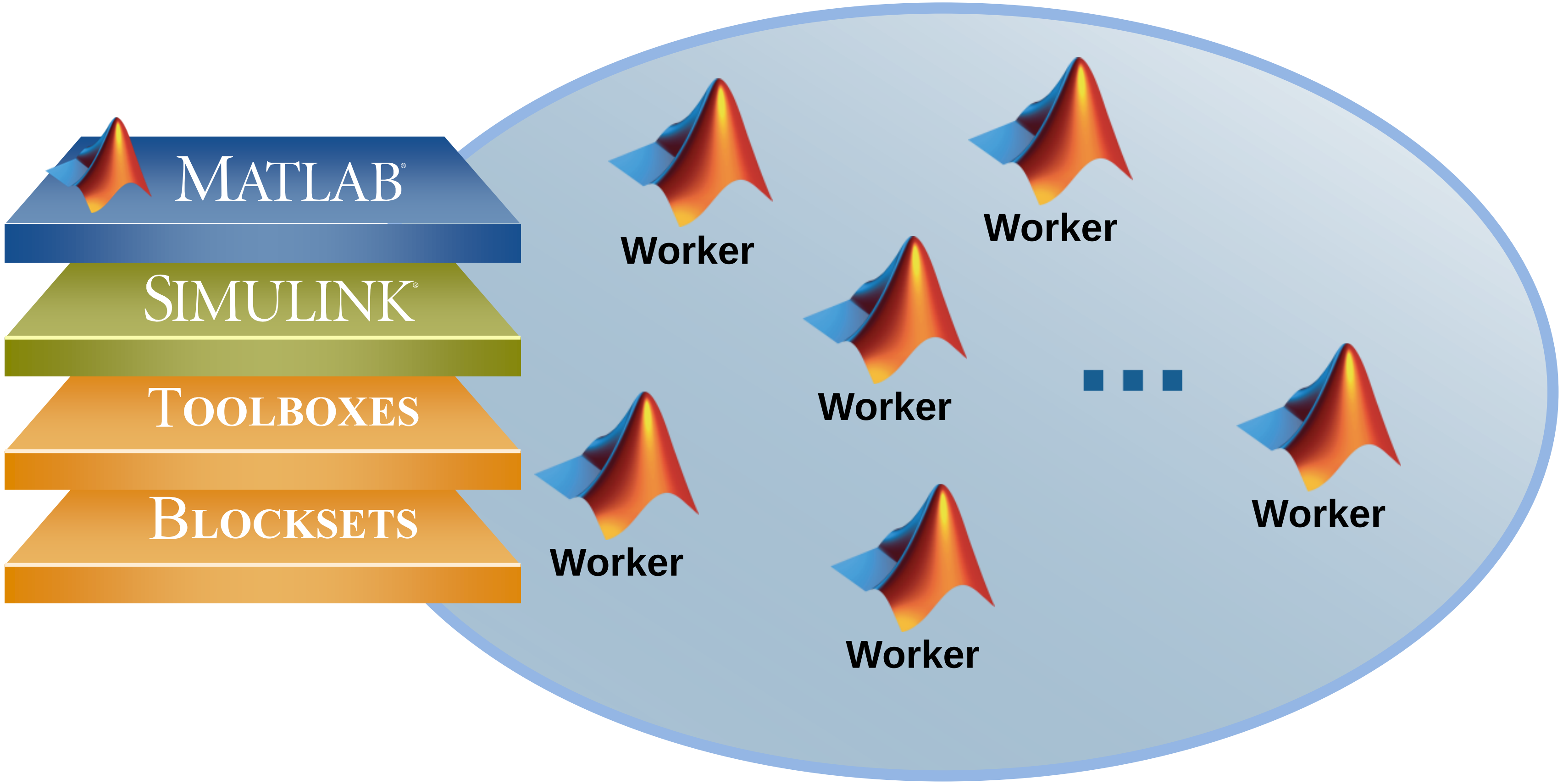
Webinar: Optimizing and Accelerating Your MATLAB Code

Programming Parallel Applications

- Built-in multithreading
  - Automatically enabled in MATLAB since R2008a
  - Multiple threads in a single MATLAB computation engine

- Parallel computing using explicit techniques
  - Multiple computation engines controlled by a single session
  - High-level constructs to let you parallelize MATLAB applications
  - Perform MATLAB computations on GPUs

Parallel Computing

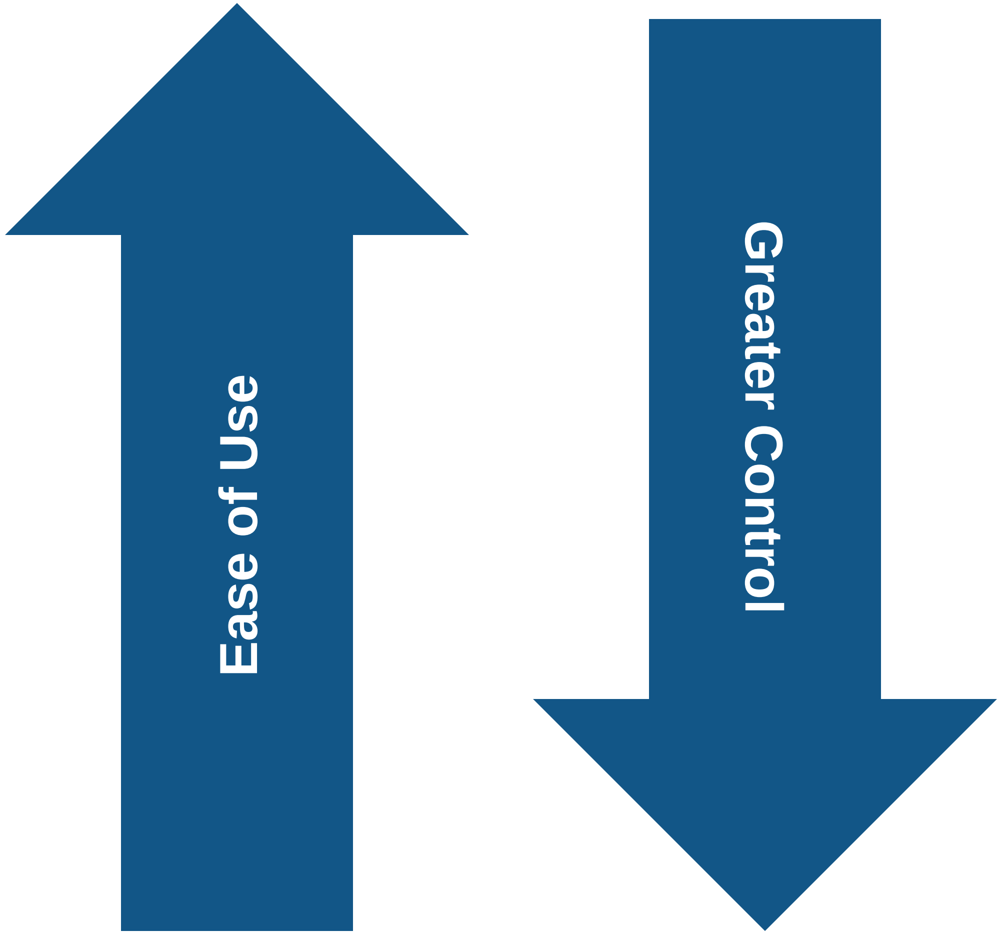


Agenda

- Utilizing multiple cores on a desktop computer
- Scaling up to cluster and cloud resources
- Tackling data-intensive problems on desktops and clusters
- Accelerating applications with NVIDIA GPUs
- Summary and resources

Programming Parallel Applications

- Built in support
  - `..., 'UseParallel', true)`



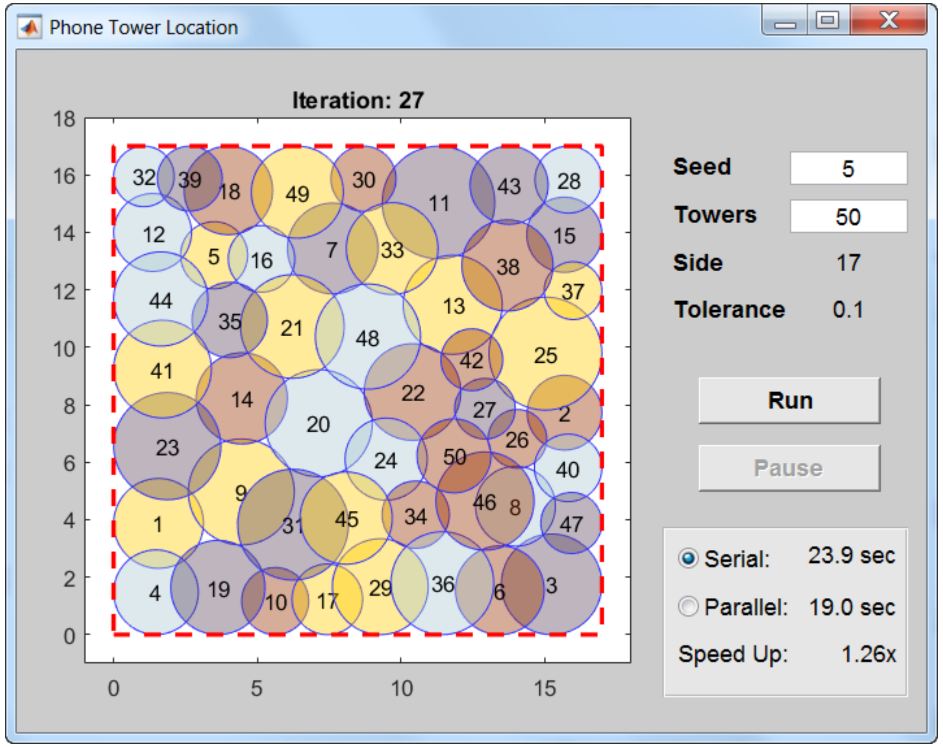
Ease of Use

Greater Control

Demo: Cell Phone Tower Optimization

Using Parallel-Enabled Functions

- Parallel-enabled functions in Optimization Toolbox
- Set flags to run optimization in parallel
- Use pool of MATLAB workers to enable parallelism



Seed 5

Towers 50

Side 17

Tolerance 0.1

● Serial: 23.9 sec

○ Parallel: 19.0 sec

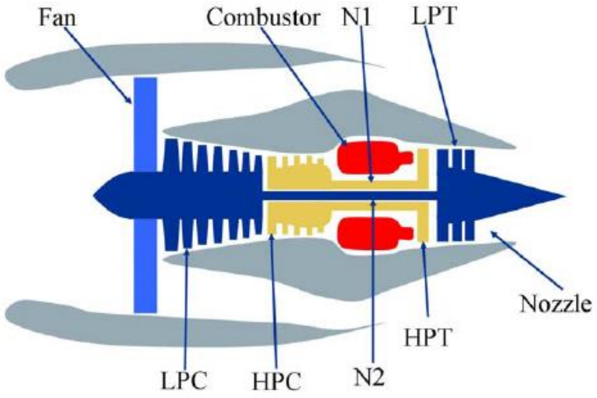
Speed Up: 1.26x

Predictive Maintenance of Turbofan Engine

Sensor data from 100 engines of the same model

Scenario: No data from failures

- Performing scheduled maintenance
- No failures have occurred
- Maintenance crews tell us most engines could run for longer
- Can we be smarter about how to schedule maintenance **without** knowing what failure looks like?



Data provided by NASA PCoE
http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/

Parallel-enabled Toolboxes (MATLAB® Product Family)

Enable parallel computing support by setting a flag or preference

Image Processing

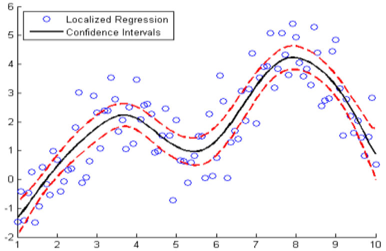
Batch Image Processor, Block Processing, GPU-enabled functions

Original Image of Peppers

Recolored Image of Peppers

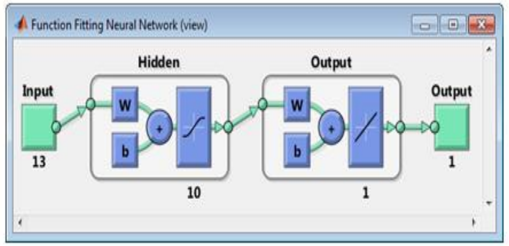
Statistics and Machine Learning

Resampling Methods, k-Means clustering, GPU-enabled functions



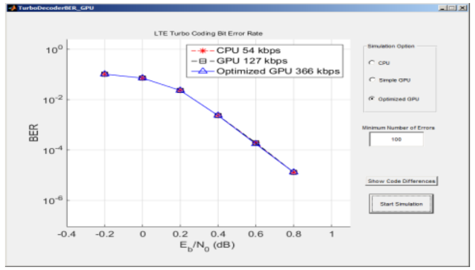
Deep Learning

Deep Learning, Neural Network training and simulation



Signal Processing and Communications

GPU-enabled FFT filtering, cross correlation, BER

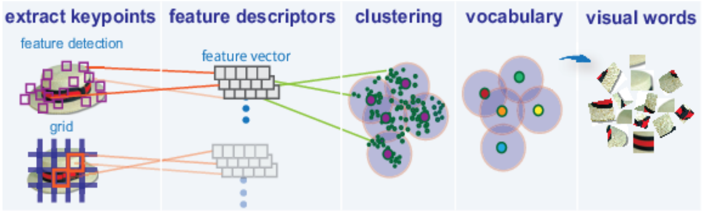


Simulation Option
○ CPU
○ Simple GPU
● Optimized GPU

Minimum Number of Errors: 100

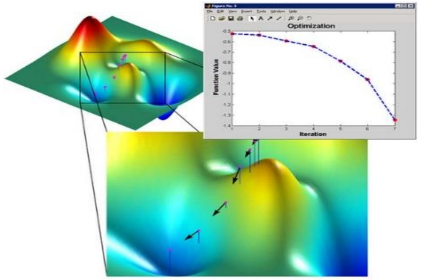
Computer Vision

Parallel-enabled functions in bag-of-words workflow



Optimization

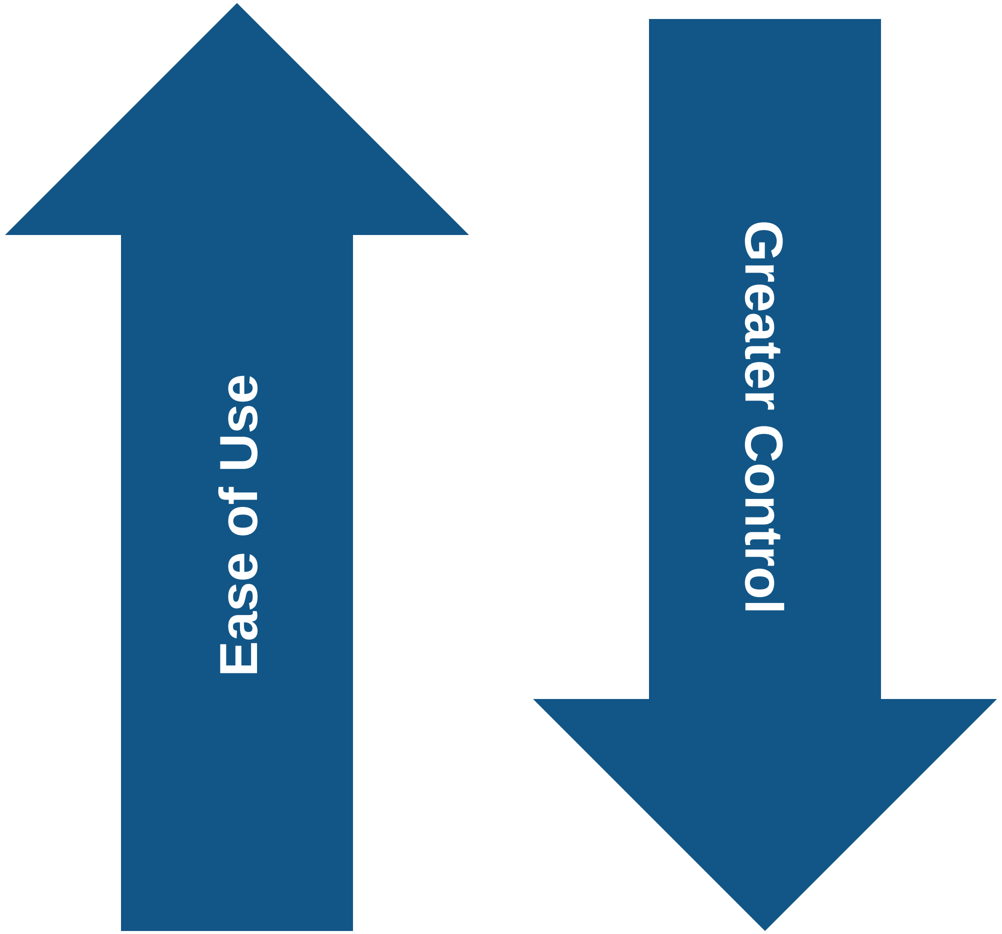
Parallel estimation of gradients



[Other parallel-enabled Toolboxes]

Programming Parallel Applications

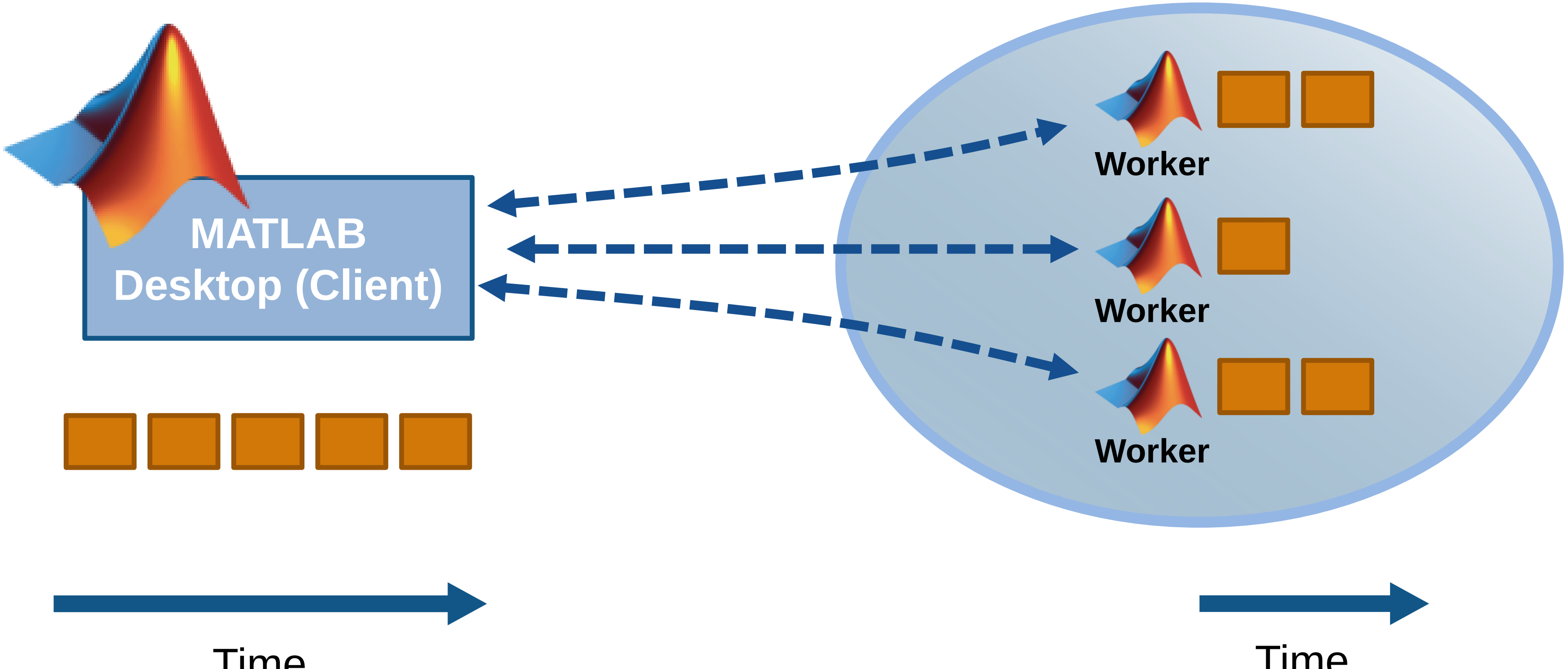
- Built in support
  - `..., 'UseParallel', true)`
- Simple programming constructs
  - `parfor, batch`


Ease of Use

Greater Control

Embarrassingly Parallel: Independent Tasks or Iterations

- No dependencies or communications between tasks
- Examples: parameter sweeps, Monte Carlo simulations

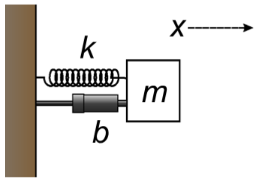


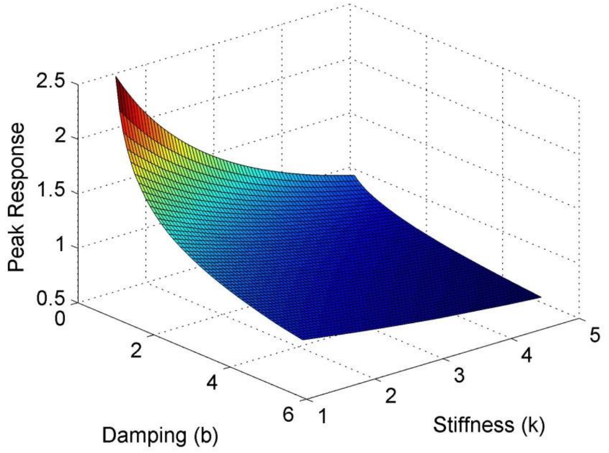
Example: Parameter Sweep of ODEs

Parallel for-loops

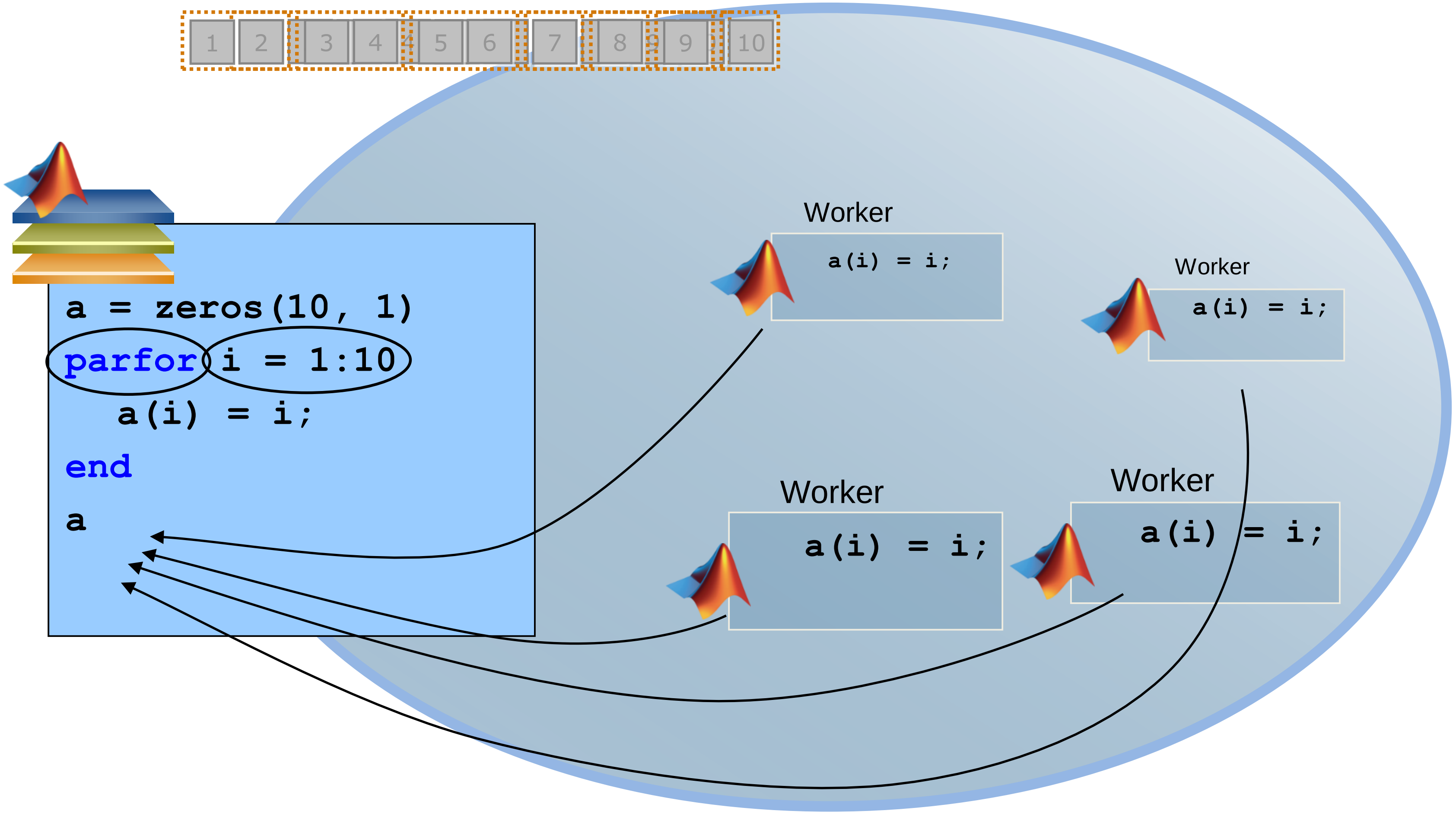
- Parameter sweep of ODE system
  - Damped spring oscillator
  - Sweep through different values of damping and stiffness
  - Record peak value for each simulation
- Convert `for` to `parfor`
- Use pool of MATLAB workers

$$\overbrace{m}^{5}\ddot{x} + \underbrace{b}_{1,2,\ldots}\dot{x} + \underbrace{k}_{1,2,\ldots}x = 0$$





Mechanics of parfor Loops



```
a = zeros(10, 1)
parfor i = 1:10
   a(i) = i;
end
a
```

Tips for Leveraging PARFOR

- Consider creating smaller arrays on each worker versus one large array prior to the parfor loop

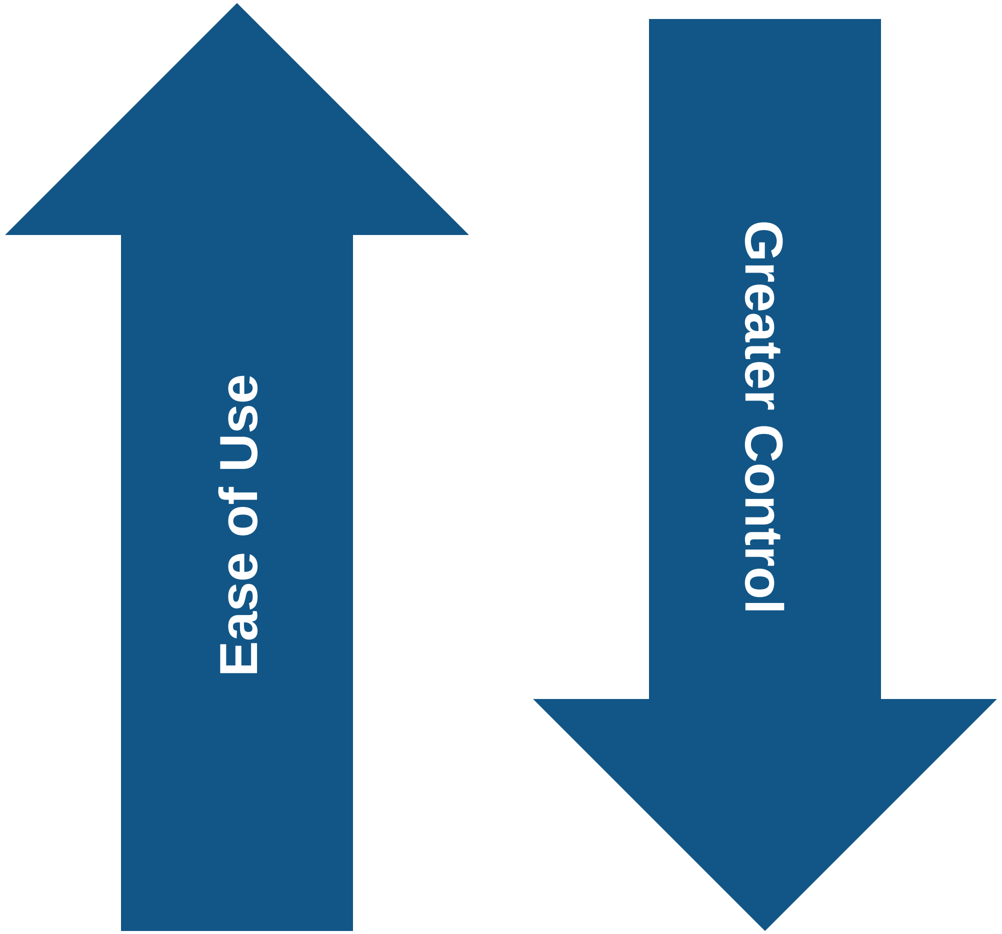
- Take advantage of `parallel.pool.Constant` to establish variables on pool workers prior to the loop

- Encapsulate blocks as functions when needed

Understanding parfor

Programming Parallel Applications

- Built in support
  - `..., 'UseParallel', true)`
- Simple programming constructs
  - `parfor, batch`
- Full control of parallelization
  - `spmd, parfeval`

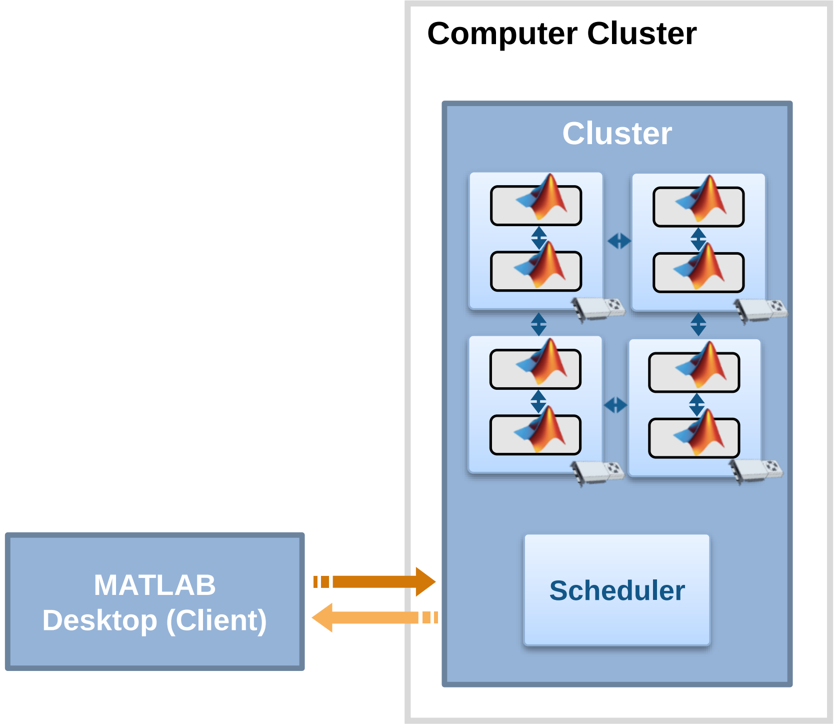


Agenda

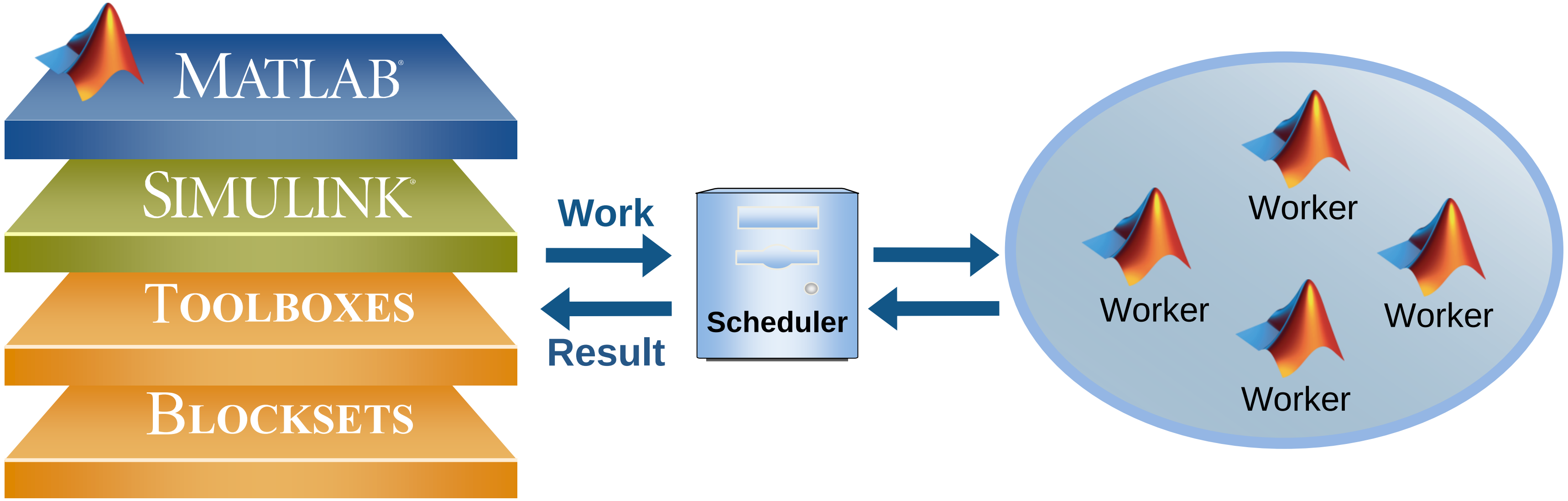
- Utilizing multiple cores on a desktop computer
- Scaling up to cluster and cloud resources
- Tackling data-intensive problems on desktops and clusters
- Accelerating applications with NVIDIA GPUs
- Summary and resources

Take Advantage of Cluster Hardware

- Offload computation:
  - Free up desktop
  - Access better computers
- Scale speed-up:
  - Use more cores
  - Go from hours to minutes
- Scale memory:
  - Utilize tall arrays and distributed arrays
  - Solve larger problems without re-coding algorithms

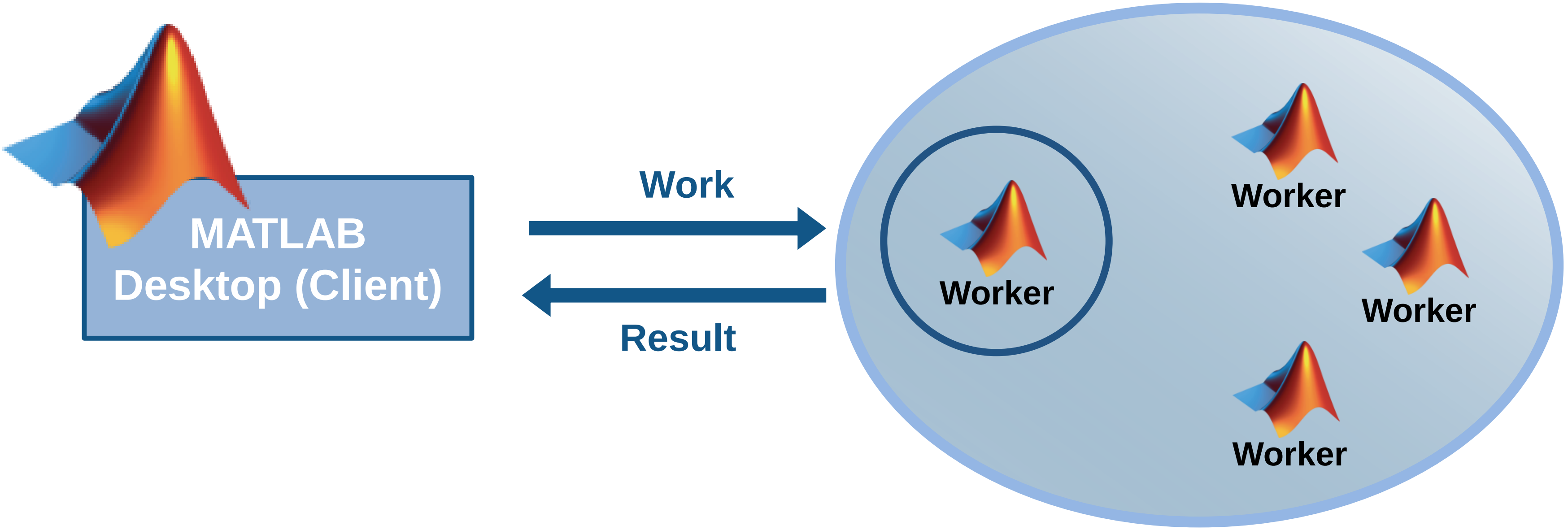


Offloading Computations



Offloading Serial Computations

- `job = batch(...);`



Example: Parameter Sweep of ODEs

Offload and Scale Processing

- Offload processing to workers:

```
batch
```

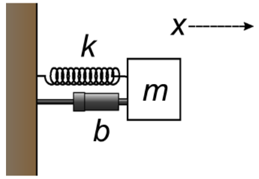
- Scale offloaded processing:

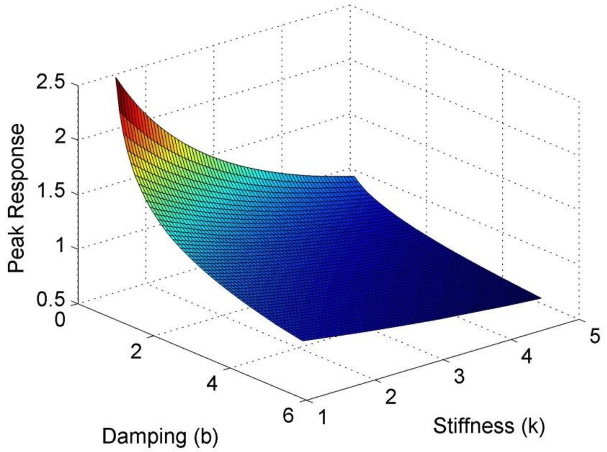
```
batch(…,'Pool',…)
```

- Retrieve results from job:

```
fetchOutputs
```

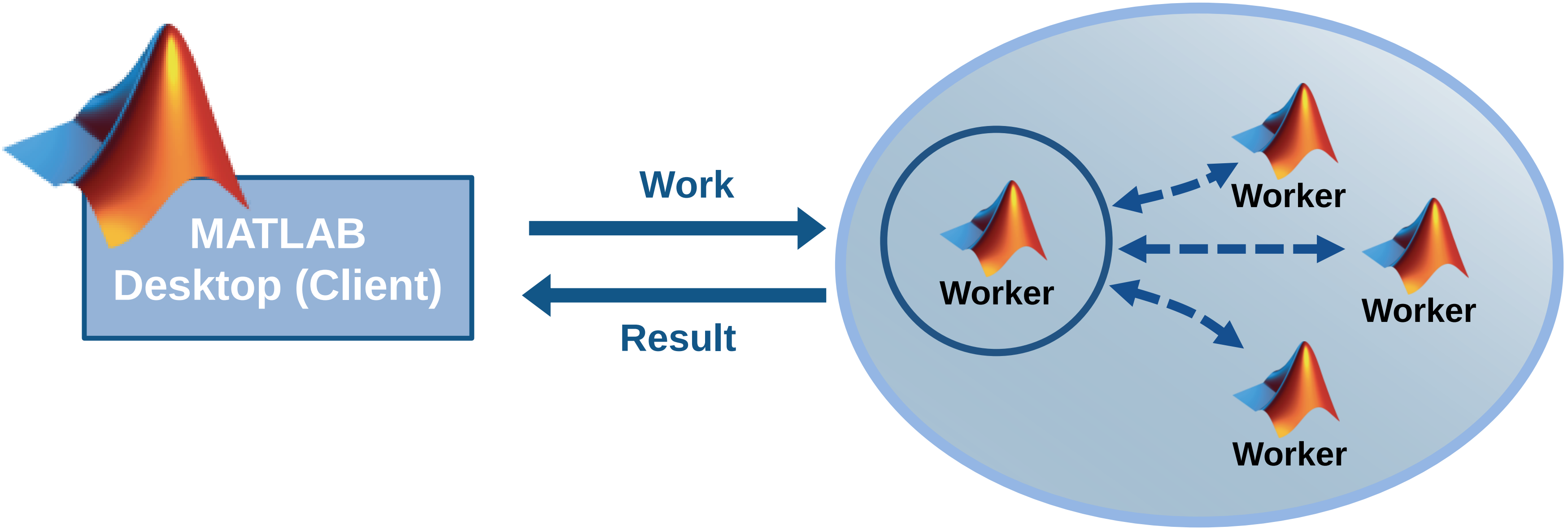
$$\overbrace{m}^{5}\ddot{x}+\underbrace{b}_{1,2,\dots}\dot{x}+\underbrace{k}_{1,2,\dots}x=0$$





Offloading and Scaling Computations

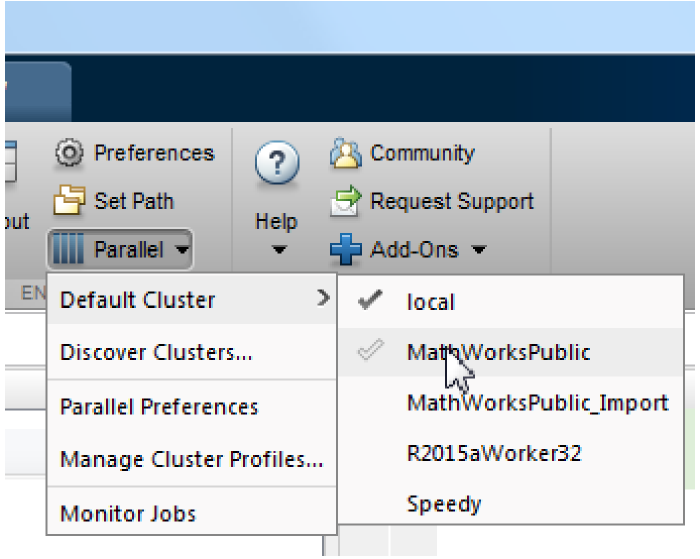
- `job = batch(... , 'Pool', n);`



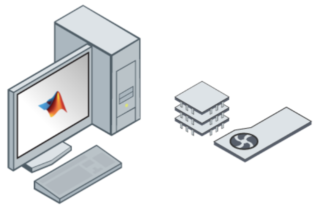
Migrate to Cluster / Cloud

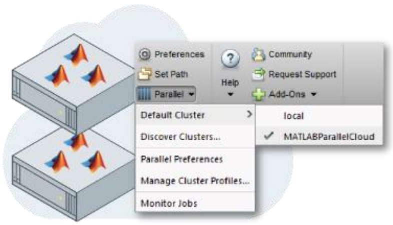
- Use MATLAB Distributed Computing Server

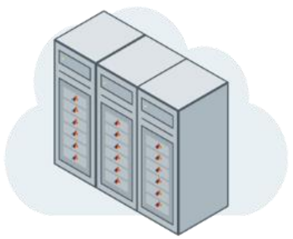
- Change hardware without changing algorithm



Scale your applications beyond the desktop







Option	**Parallel Computing Toolbox**	**MATLAB Parallel Cloud**	**MATLAB Distributed Computing Server *for Amazon EC2***	**MATLAB Distributed Computing Server *for Custom Cloud***	**MATLAB Distributed Computing Server**
Description	Explicit desktop scaling	Single-user, basic scaling to cloud	Scale to EC2 with some customization	Scale to custom cloud	Scale to clusters
Maximum workers	No limit	16	256	No limit	No limit
Hardware	Desktop	MathWorks Compute Cloud	Amazon EC2	Amazon EC2, Microsoft Azure, Others	Any
Availability	Worldwide	United States and Canada	United States, Canada and other select countries in Europe	Worldwide	Worldwide

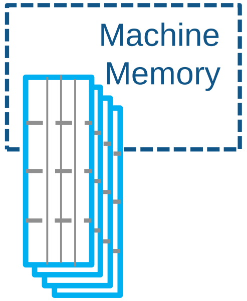
Learn More: [Parallel Computing on the Cloud]

Agenda

- Utilizing multiple cores on a desktop computer
- Scaling up to cluster and cloud resources
- Tackling data-intensive problems on desktops and clusters
- Accelerating applications with NVIDIA GPUs
- Summary and resources

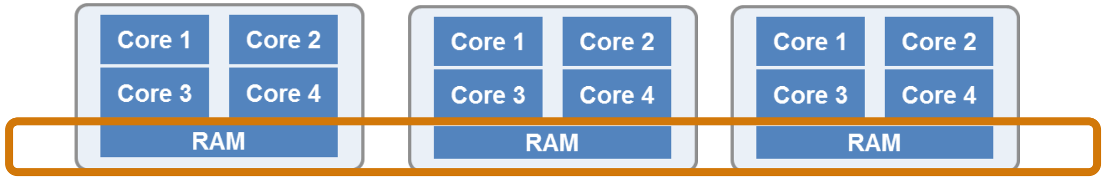
Tall and Distributed Data

- Tall Data
  - Columnar data that does not fit in memory of a desktop or cluster



- Common Actions
  - Data manipulation, math, statistics
  - Summary visualizations
  - Machine learning

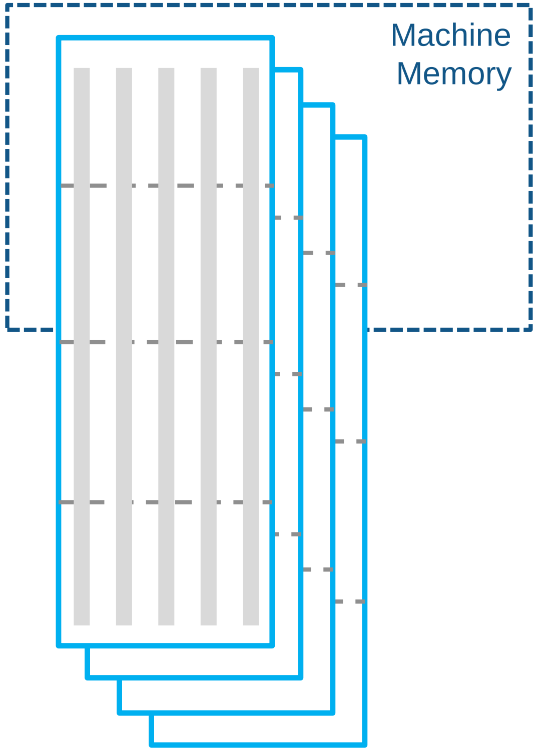
- Distributed Data
  - Large matrices using the combined memory of a cluster



- Common Actions
  - Matrix Manipulation
  - Linear Algebra and Signal Processing

Tall Arrays

- New data type in MATLAB R2016b
- Applicable when:
  - Data is **columnar** – with **many** rows
  - Overall data size is **too big to fit into memory**
  - Operations are mathematical/statistical in nature
- Statistical and machine learning applications
  - Hundreds of functions supported in MATLAB and Statistics and Machine Learning Toolbox



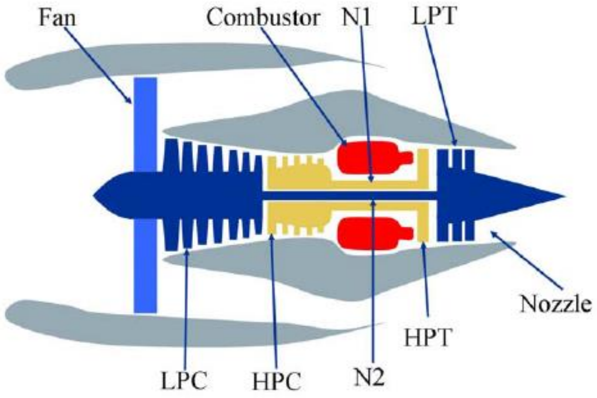
Tall Data

Predictive Maintenance of Turbofan Engine

Sensor data from 100 engines of the same model

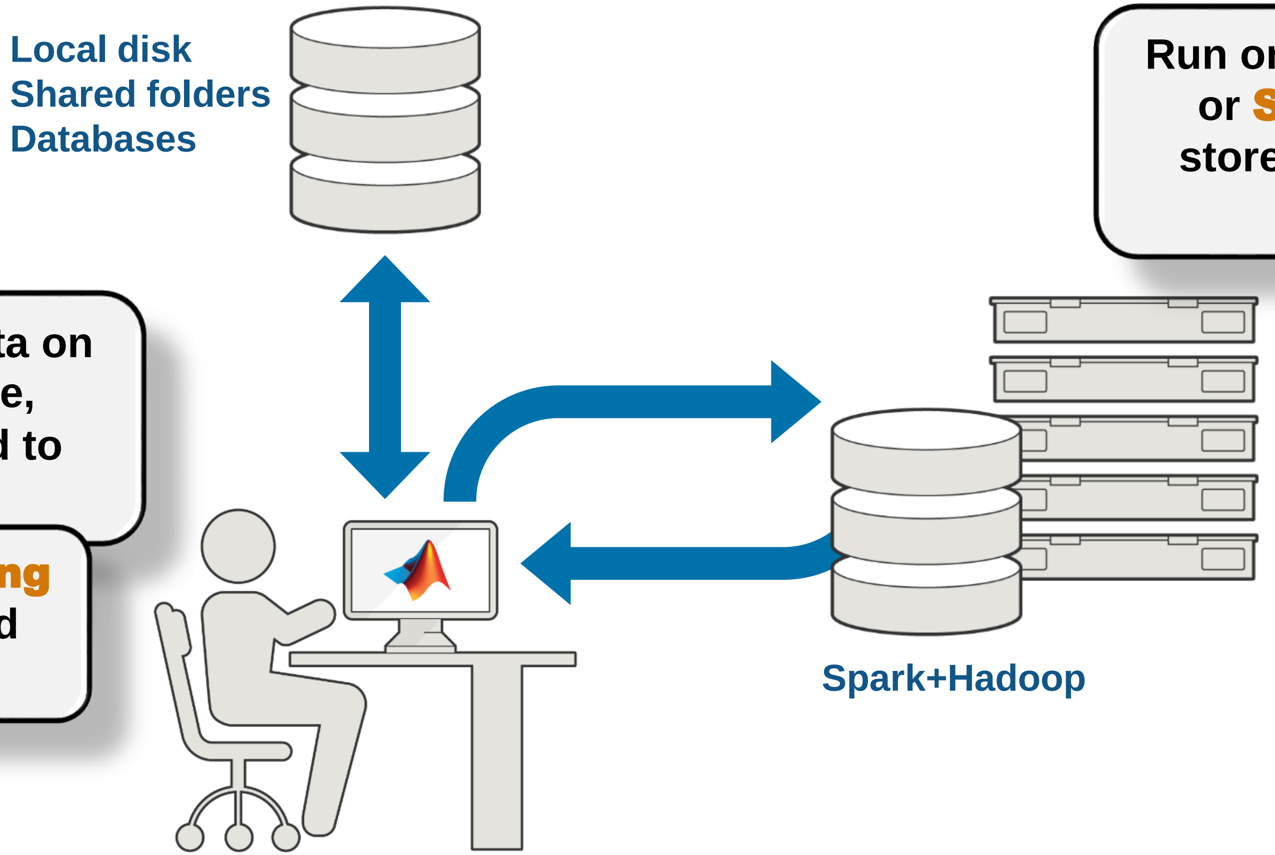
Scenario: No data from failures

- Performing scheduled maintenance
- No failures have occurred
- Maintenance crews tell us most engines could run for longer
- Can we be smarter about how to schedule maintenance **without** knowing what failure looks like?



Data provided by NASA PCoE
http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/

Execution Environments for Tall Arrays



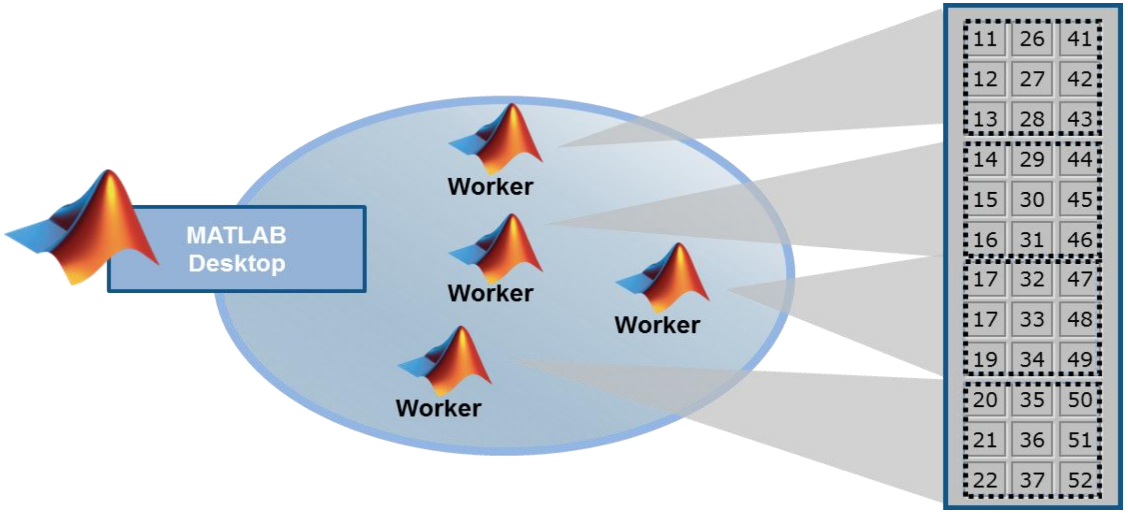
Process out-of-memory data on your **Desktop** to explore, analyze, gain insights and to develop analytics

Use **Parallel Computing Toolbox** for increased performance

Run on **Compute Clusters**, or **Spark** if your data is stored in **HDFS,** for large scale analysis

Distributed Arrays

- Distributed Arrays hold data remotely on workers running on a cluster
- Manipulate directly from client MATLAB (desktop)
- 200+ MATLAB functions overloaded for distributed arrays



11	26	41
12	27	42
13	28	43
14	29	44
15	30	45
16	31	46
17	32	47
17	33	48
19	34	49
20	35	50
21	36	51
22	37	52

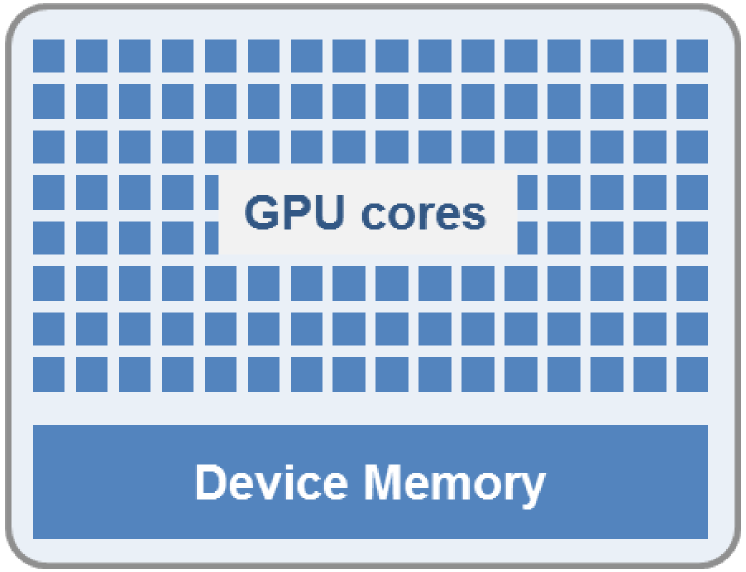
Supported functions for Distributed Arrays

Agenda

- Utilizing multiple cores on a desktop computer
- Scaling up to cluster and cloud resources
- Tackling data-intensive problems on desktops and clusters
- Accelerating applications with NVIDIA GPUs
- Summary and resources

Graphics Processing Units (GPUs)

- For graphics acceleration and scientific computing
- Many parallel processors
- Dedicated high speed memory



GPU Requirements

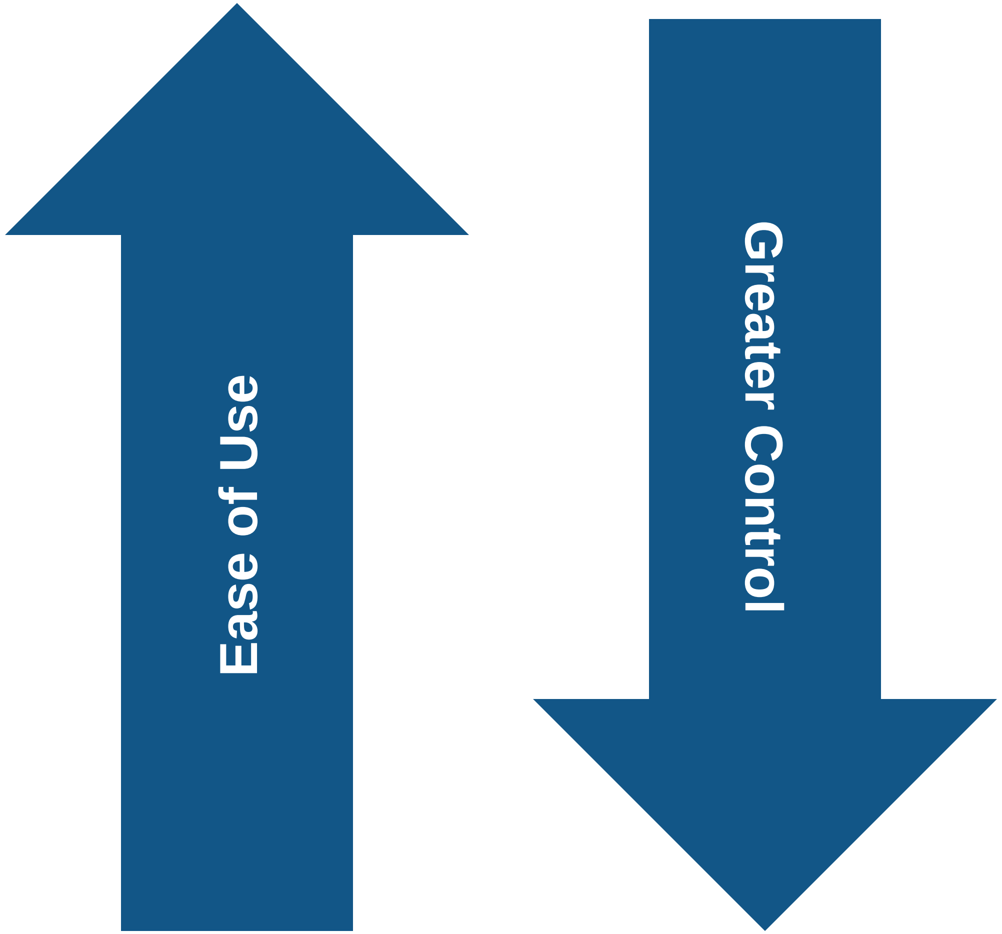
- Parallel Computing Toolbox requires NVIDIA GPUs
- www.nvidia.com/object/cuda_gpus.html

MATLAB Release	Required Compute Capability
MATLAB R2018a and later releases	3.0 or greater
MATLAB R2014b – MATLAB R2017b	2.0 or greater
MATLAB R2014a and earlier releases	1.3 or greater

Programming with GPUs

- Built in toolbox support

- Simple programming constructs
  - `gpuArray, gather`



Ease of Use

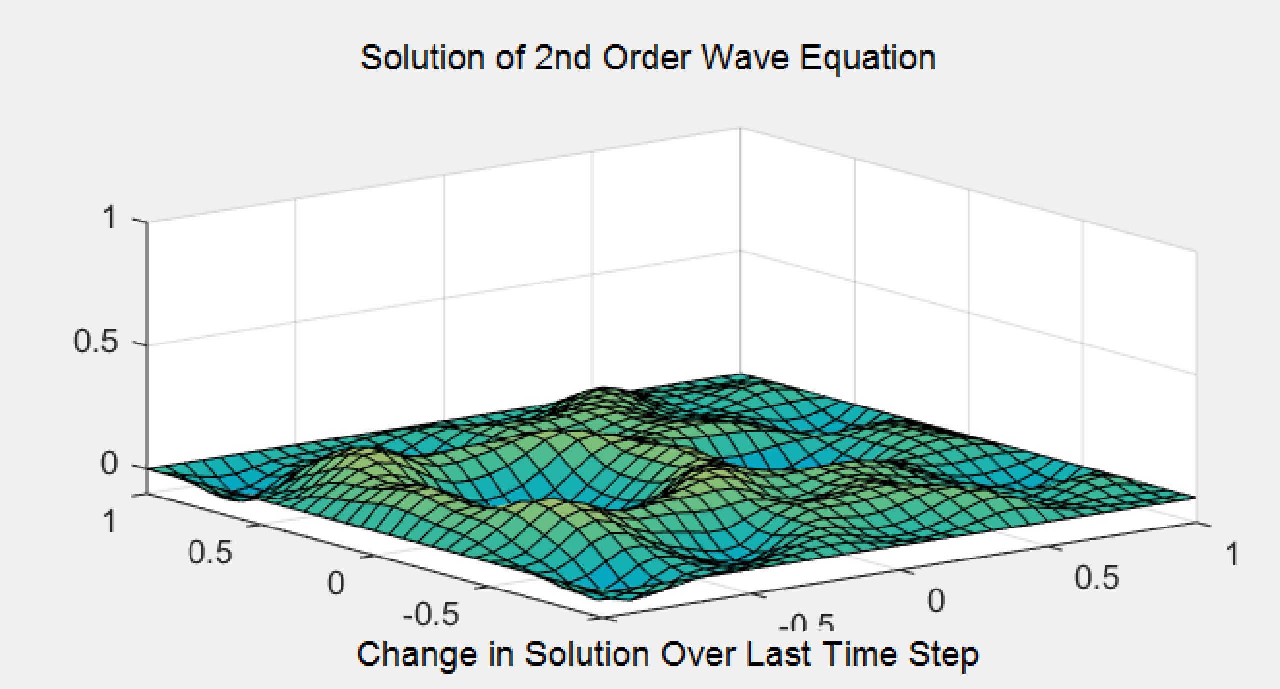
Greater Control

Demo: Wave Equation

Accelerating scientific computing in MATLAB with GPUs

- **Objective:** Solve 2nd order wave equation with spectral methods

- **Approach:**
  - Develop code for CPU
  - Modify the code to use GPU computing using `gpuArray`
  - Compare performance of the code using CPU and GPU



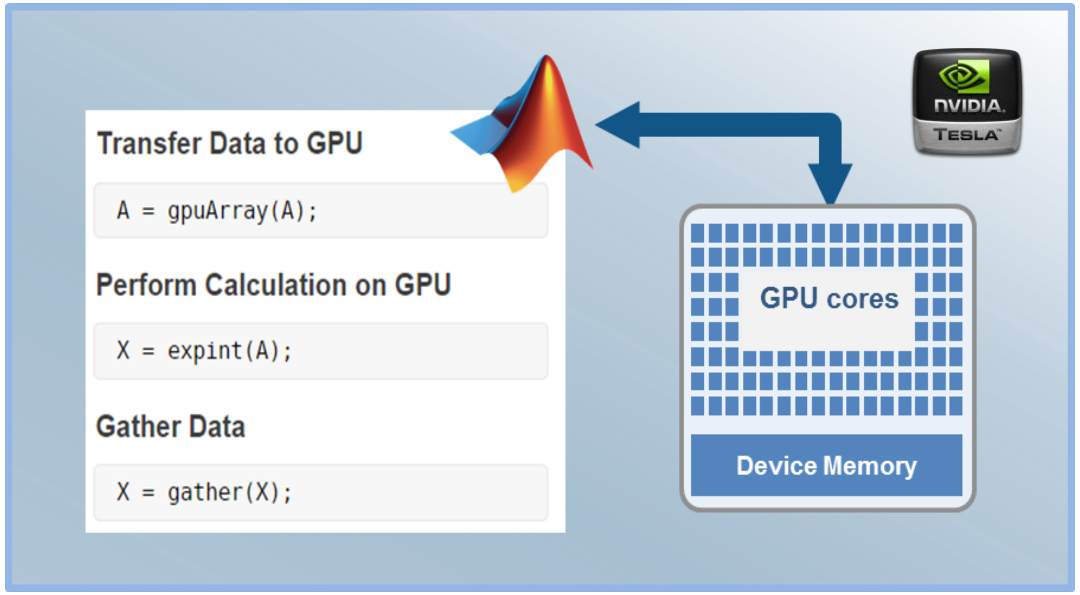
Speed-up MATLAB code with NVIDIA GPUs

- Ideal Problems
  - Massively Parallel and/or Vectorized operations
  - Computationally Intensive

- 300+ GPU-enabled MATLAB functions
  - Enable existing MATLAB code to run on GPUs
  - Support for sparse matrices on GPUs

- Additional GPU-enabled Toolboxes
  - Deep Learning
  - Image Processing
  - Signal Processing

  ..... Learn More



Transfer Data to GPU

```
A = gpuArray(A);
```

Perform Calculation on GPU

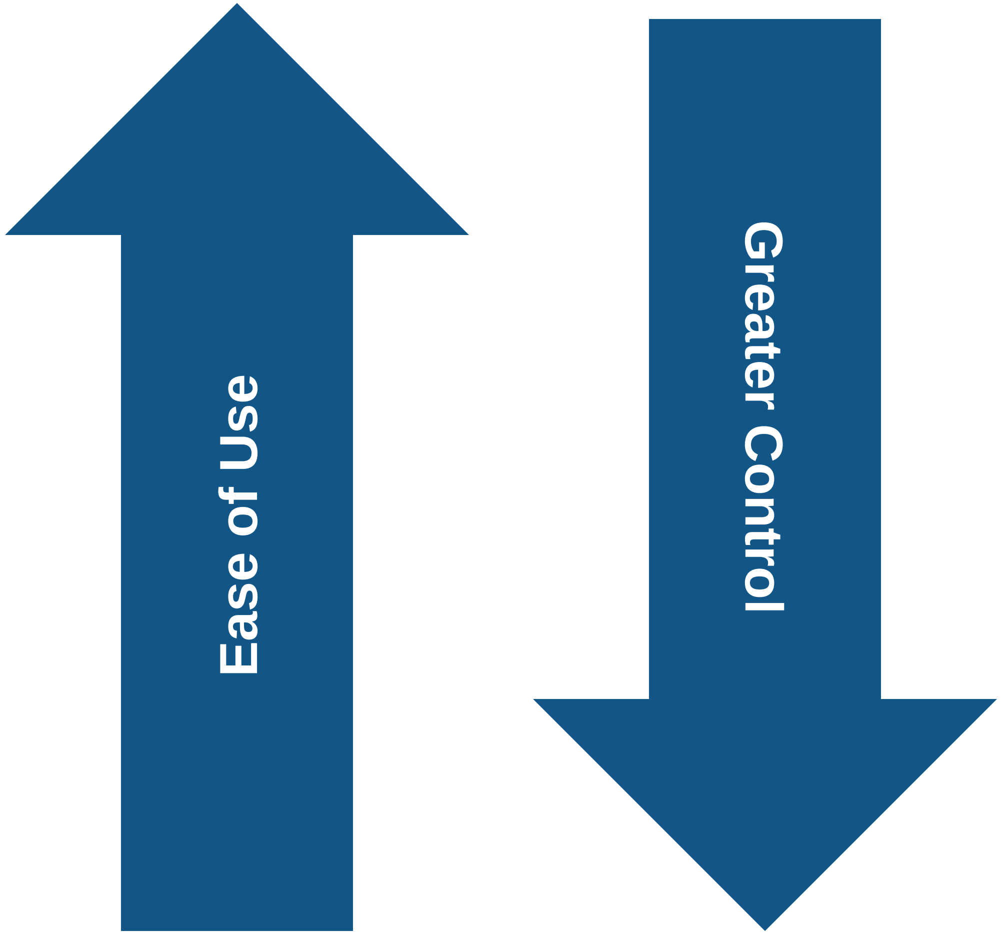
```
X = expint(A);
```

Gather Data

```
X = gather(X);
```

Programming with GPUs

- Built in toolbox support
- Simple programming constructs
  - `gpuArray, gather`
- Advanced programming constructs
  - `spmd, arrayfun`
- Interface for experts
  - `CUDAKernel, mex`



Ease of Use

Greater Control

Agenda

- Utilizing multiple cores on a desktop computer
- Scaling up to cluster and cloud resources
- Tackling data-intensive problems on desktops and clusters
- Accelerating applications with NVIDIA GPUs
- Summary and resources

Summary

- Easily develop parallel MATLAB applications without being a parallel programming expert

- Speed up the execution of your MATLAB applications using additional hardware

- Develop parallel applications on your desktop and easily scale to a cluster when needed

Some Other Valuable Resources

- MATLAB Documentation
  - MATLAB → Advanced Software Development → Performance and Memory
  - Parallel Computing Toolbox
- Parallel and GPU Computing Tutorials
  - https://www.mathworks.com/videos/series/parallel-and-gpu-computing-tutorials-97719.html
- Parallel Computing on the Cloud with MATLAB
  - http://www.mathworks.com/products/parallel-computing/parallel-computing-on-the-cloud/