MathWorks

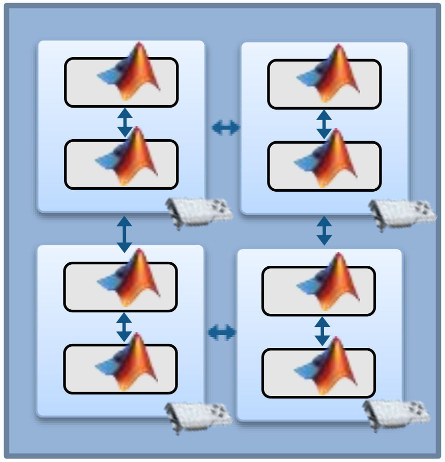
Parallel Computing with MATLAB

Scott Benway
Senior Account Manager

Jiro Doke, Ph.D.
Senior Application Engineer

Acceleration Strategies Applied in MATLAB

Approach	Options
Best coding practices	Preallocation, vectorization, profiling *("Speeding Up MATLAB Applications")*
More hardware	**More Processors, Cores, or GPUs** ***(MATLAB Parallel Computing Tools)***
Integration with other languages	C/C++, Fortran *(MEX, MATLAB Coder)*



Agenda

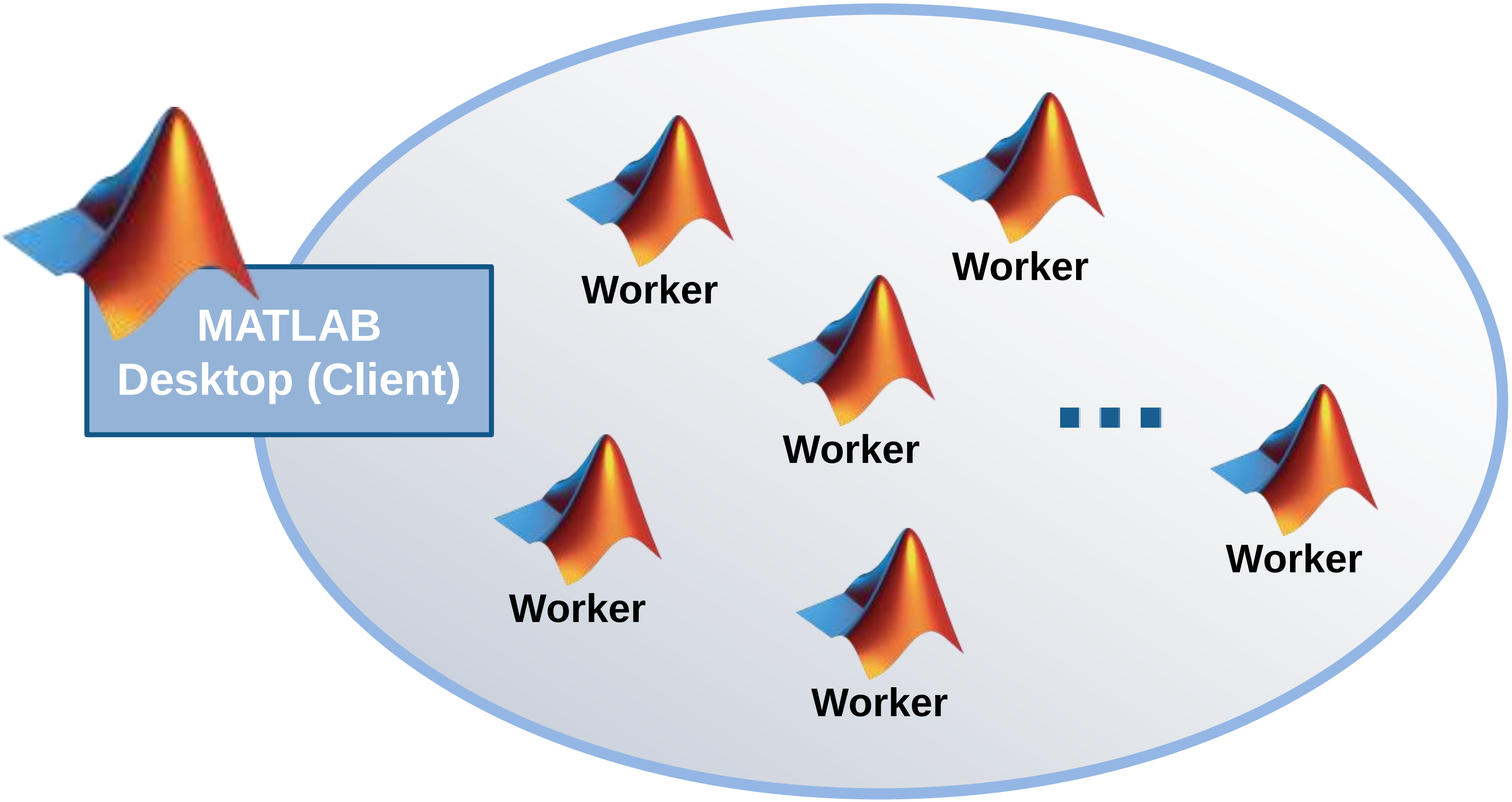
- Introduction to parallel computing tools
- Using multicore/multi-processor computers
- Using graphics processing units (GPUs)
- Scaling up to a cluster

Using More Hardware

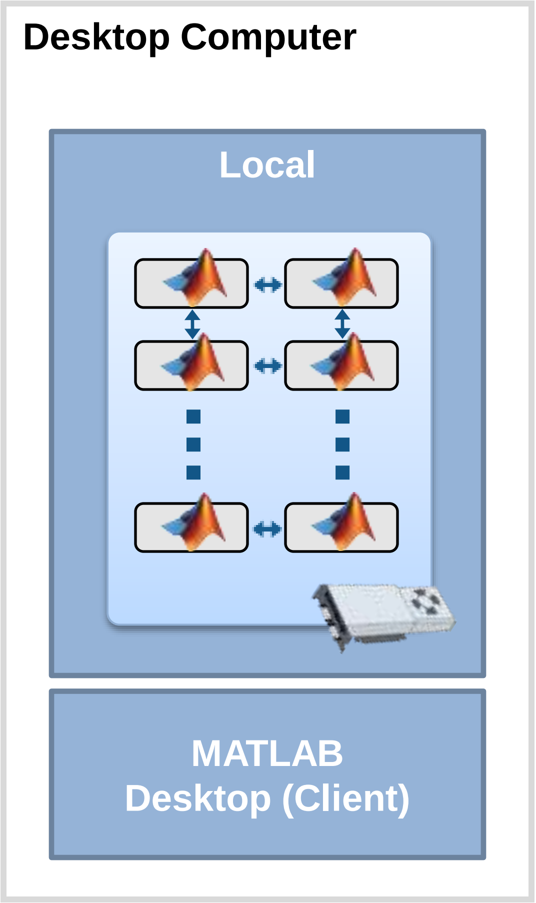
- Built-in multithreading
  - Automatically enabled in MATLAB since R2008a
  - Multiple threads in a single MATLAB computation engine
    www.mathworks.com/discovery/multicore-matlab.html

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

Going Beyond Serial MATLAB Applications

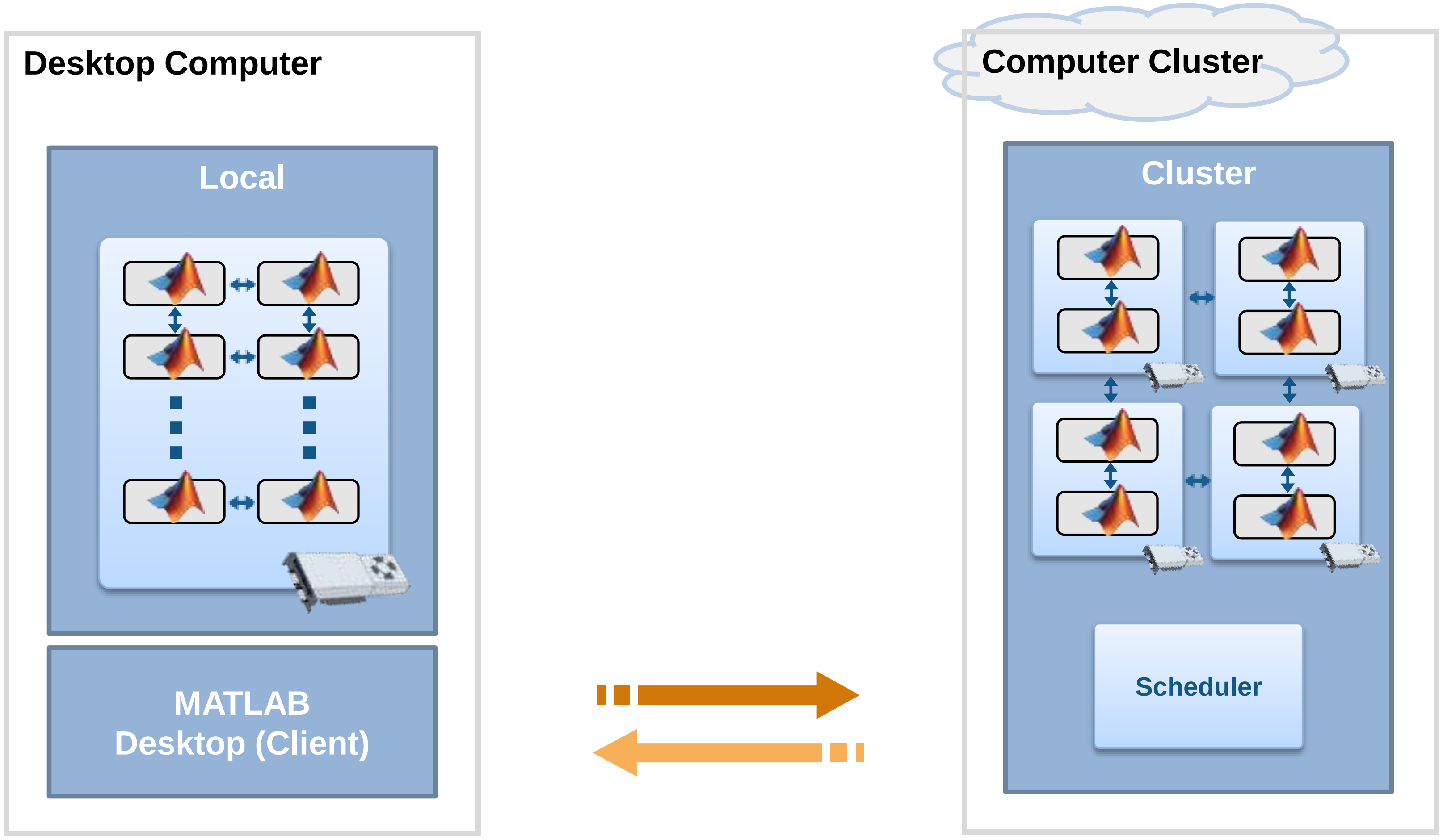


Parallel Computing Toolbox for the Desktop



- Speed up parallel applications
- Take advantage of GPUs
- Prototype code for your cluster

Scale Up to Clusters and Clouds



Agenda

- Introduction to parallel computing tools
- Using multicore/multi-processor computers
- Using graphics processing units (GPUs)
- Scaling up to a cluster

Programming Parallel Applications (CPU)

Ease of Use

- Built-in support with Toolboxes

Greater Control

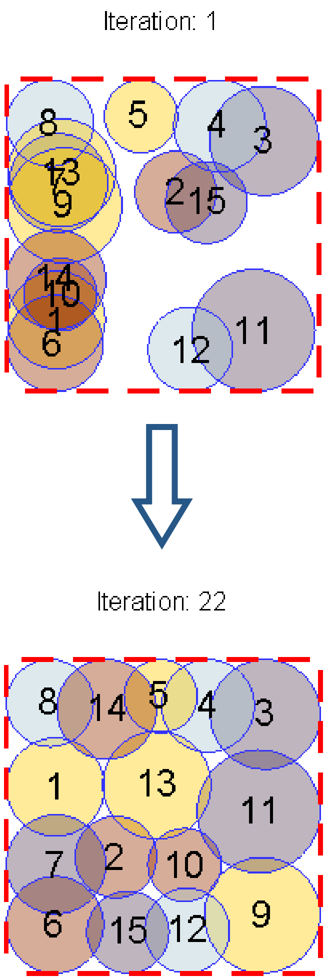
Example: Optimizing Cell Tower Position

Built-in parallel support

- With Parallel Computing Toolbox use built-in parallel algorithms in Optimization Toolbox

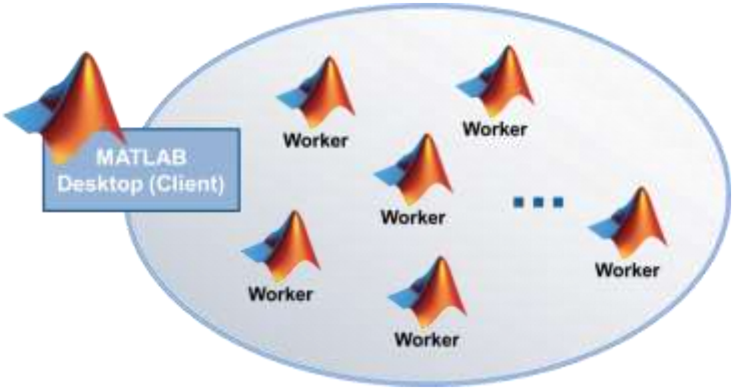
- Run optimization in parallel

- Use pool of MATLAB workers



Tools Providing Parallel Computing Support

- Optimization Toolbox, Global Optimization Toolbox
- Statistics Toolbox
- Signal Processing Toolbox
- Neural Network Toolbox
- Image Processing Toolbox
- …



Directly leverage functions in Parallel Computing Toolbox

www.mathworks.com/builtin-parallel-support

Programming Parallel Applications (CPU)

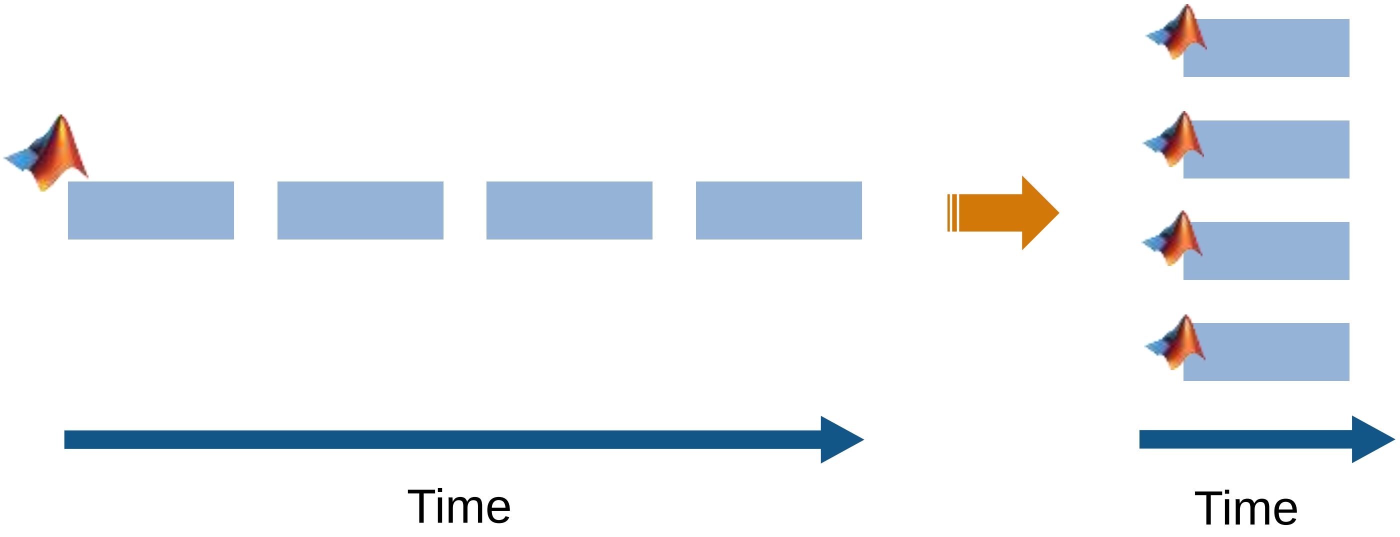
Ease of Use

- Built-in support with Toolboxes
- Simple programming constructs:
  `parfor, batch, distributed`

Greater Control

Independent Tasks or Iterations

- Ideal problem for parallel computing
- No dependencies or communications between tasks
- Examples: parameter sweeps, Monte Carlo simulations



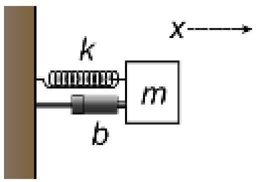
blogs.mathworks.com/loren/2009/10/02/using-parfor-loops-getting-up-and-running/

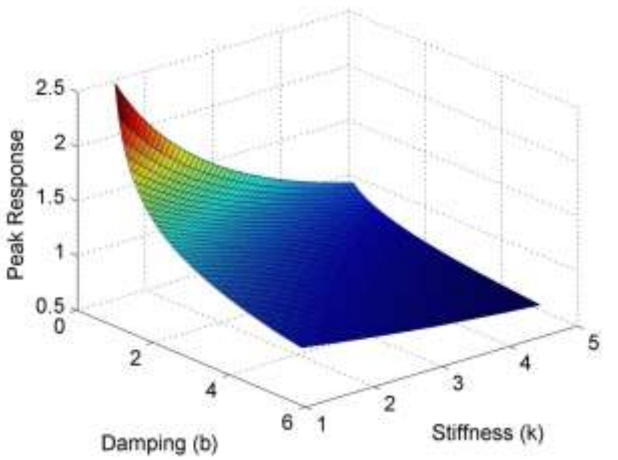
Example: Parameter Sweep of ODEs

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

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





Programming Parallel Applications (CPU)

Ease of Use

- Built-in support with Toolboxes
- Simple programming constructs: `parfor, batch, distributed`
- Advanced programming constructs: `createJob, labSend, spmd`

Greater Control

Agenda

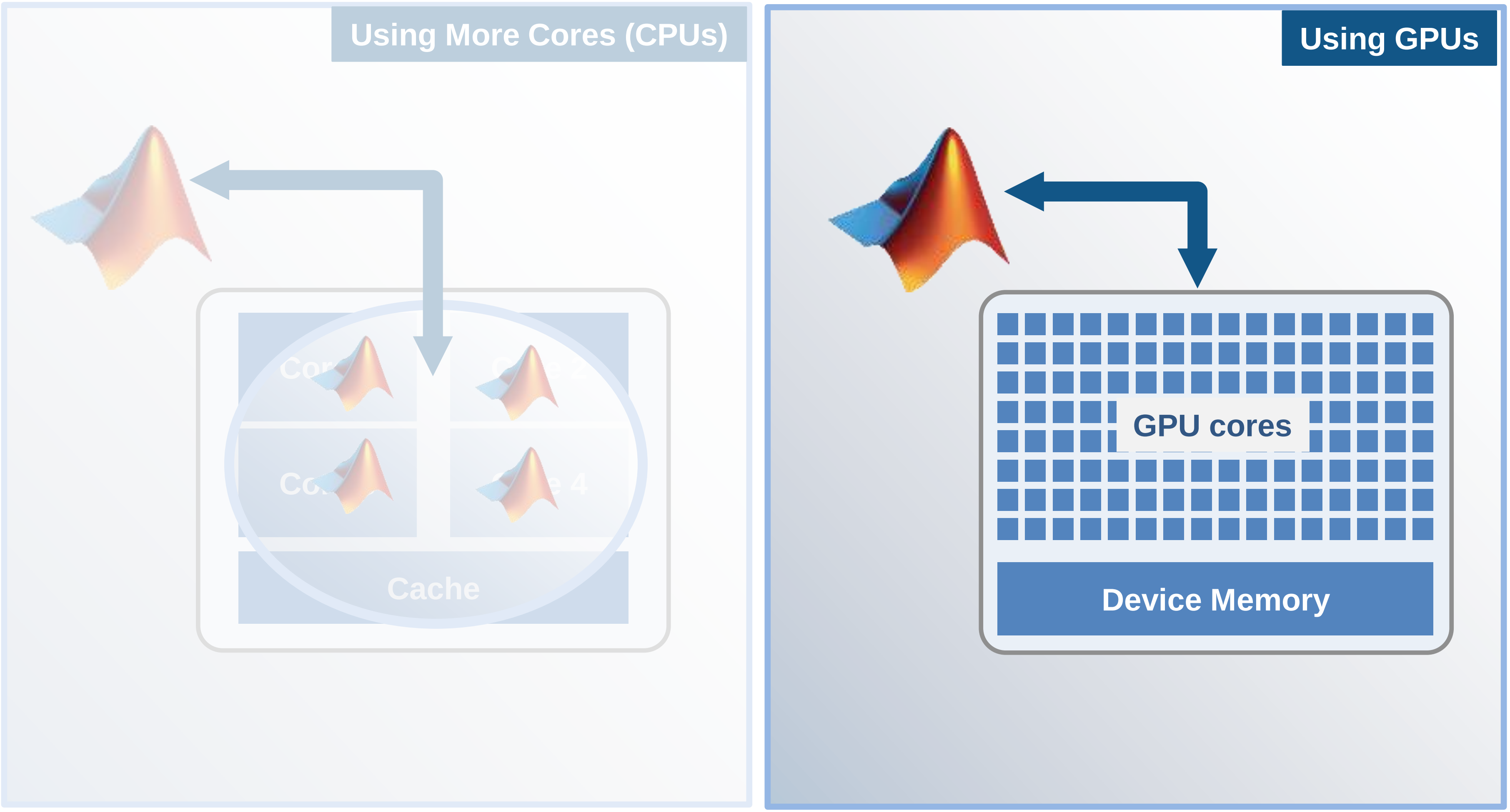
- Introduction to parallel computing tools
- Using multicore/multi-processor computers
- Using graphics processing units (GPUs)
- Scaling up to a cluster

What is a Graphics Processing Unit (GPU)

- Originally for graphics acceleration, now also used for scientific calculations
- Massively parallel array of integer and floating point processors
  - Typically hundreds of processors per card
  - GPU cores complement CPU cores
- Dedicated high-speed memory

* Parallel Computing Toolbox requires NVIDIA GPUs with Compute Capability 1.3 or higher, including NVIDIA Tesla 20-series products. See a complete listing at www.nvidia.com/object/cuda_gpus.html

Performance Gain with More Hardware



Programming Parallel Applications (GPU)

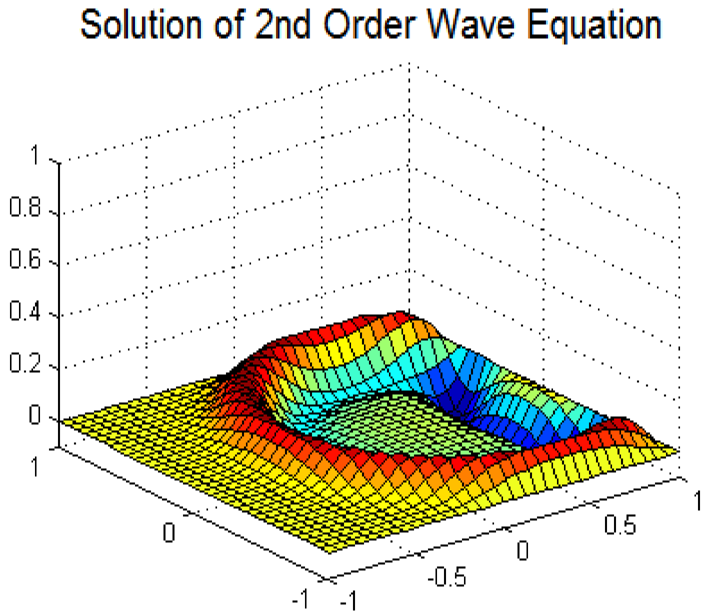
Ease of Use

- Built-in support with Toolboxes
- Simple programming constructs:
  `gpuArray, gather`
- Advanced programming constructs:
  `arrayfun, bsxfun, spmd`
- Interface for experts:
  `CUDAKernel, MEX support`

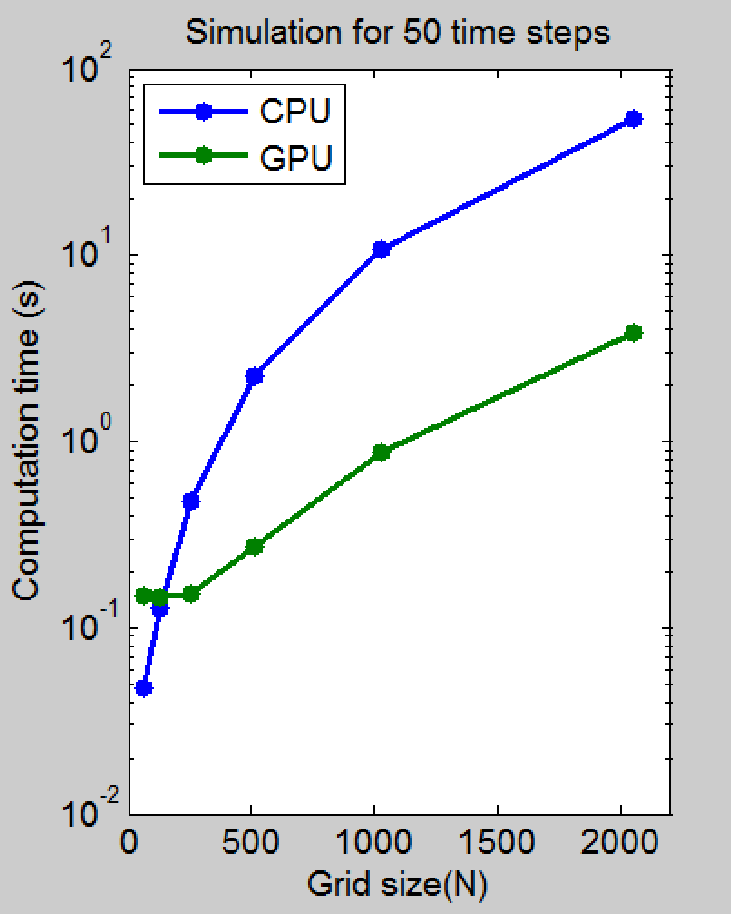
Greater Control

www.mathworks.com/gpu

Example: Solving 2D Wave Equation
GPU Computing



$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$



Intel Xeon Processor W3690 (3.47GHz),
NVIDIA Tesla K20 GPU

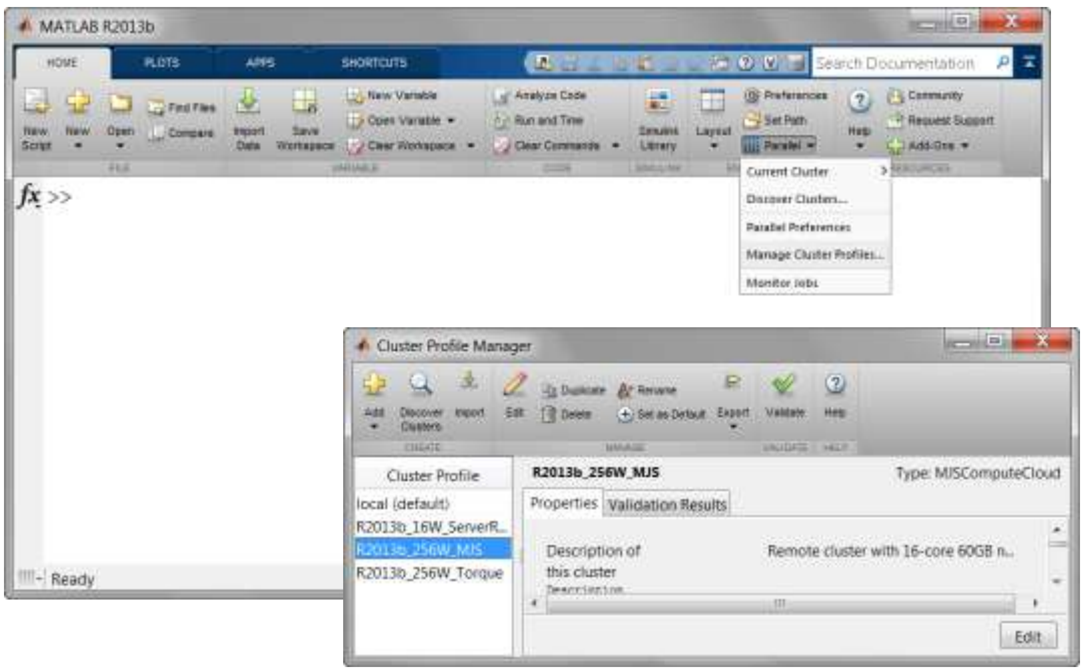
Agenda

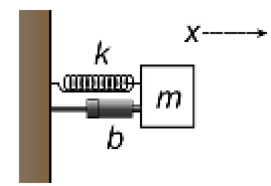
- Introduction to parallel computing tools
- Using multicore/multi-processor computers
- Using graphics processing units (GPUs)
- Scaling up to a cluster

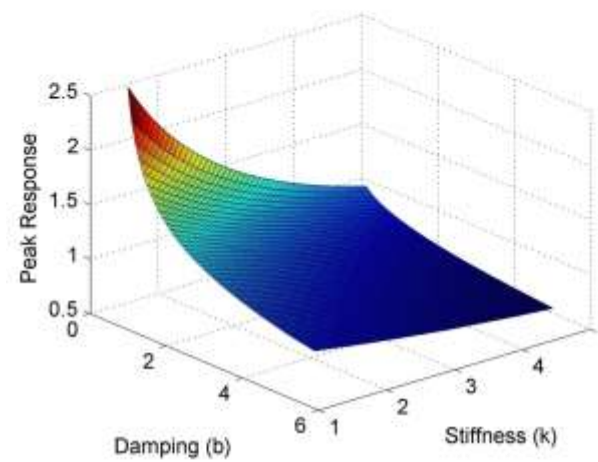
Example: Migrate from Desktop to Cloud

- Change hardware without changing algorithmic code

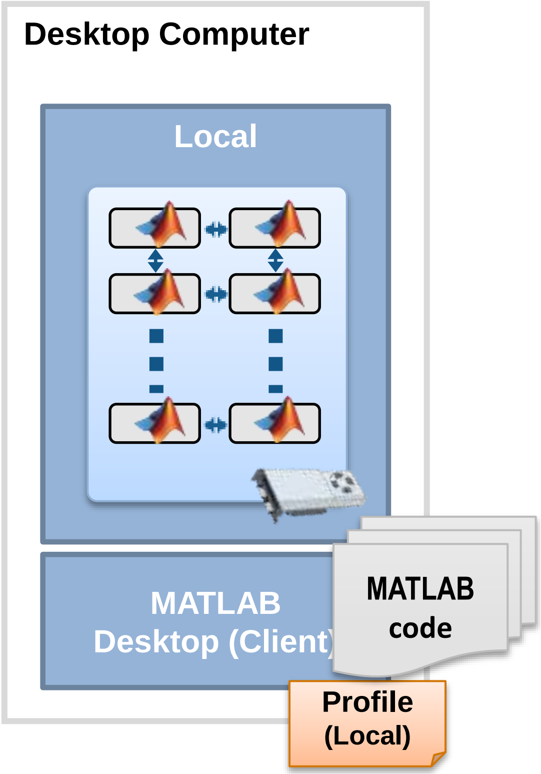
$$\overbrace{m}^{5}\ddot{x} + \underbrace{b}_{1,2,\ldots}\dot{x} + \underbrace{k}_{1,2,\ldots}x = 0$$





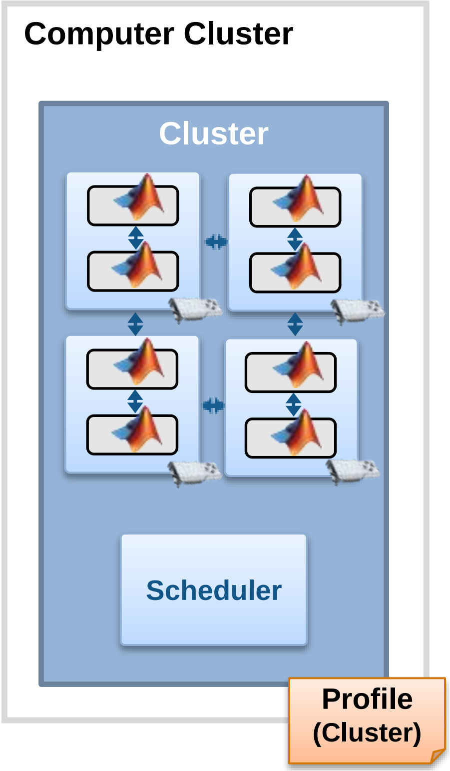


Use MATLAB Distributed Computing Server



1. Prototype code

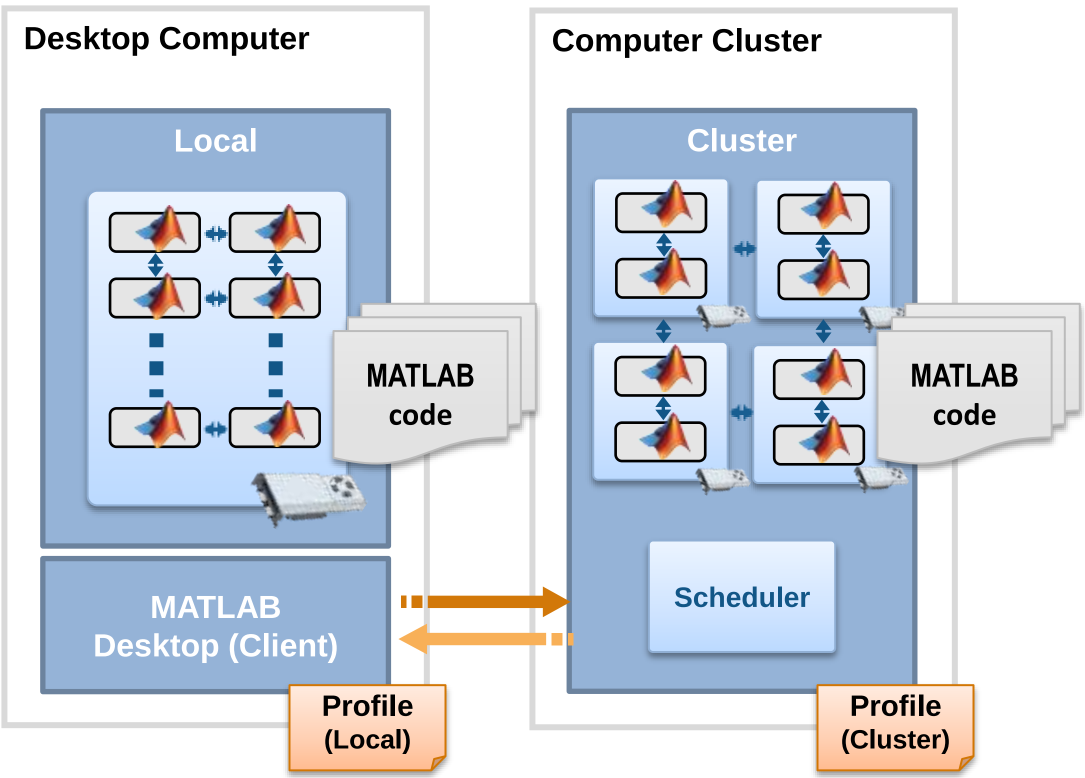
Use MATLAB Distributed Computing Server



1. Prototype code

2. Get access to an enabled cluster

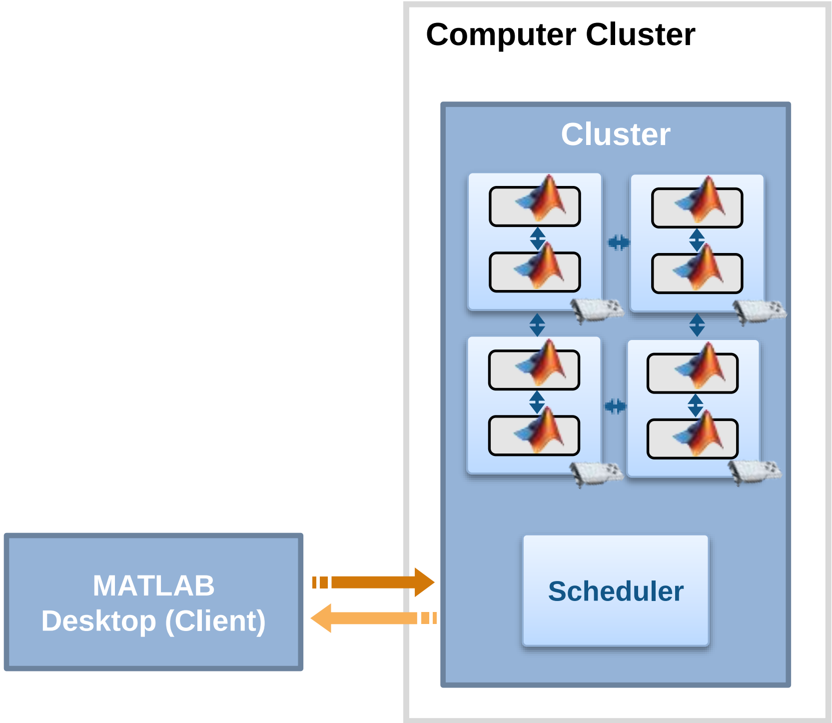
Use MATLAB Distributed Computing Server



1. Prototype code
2. Get access to an enabled cluster
3. Switch cluster profile to run on cluster resources

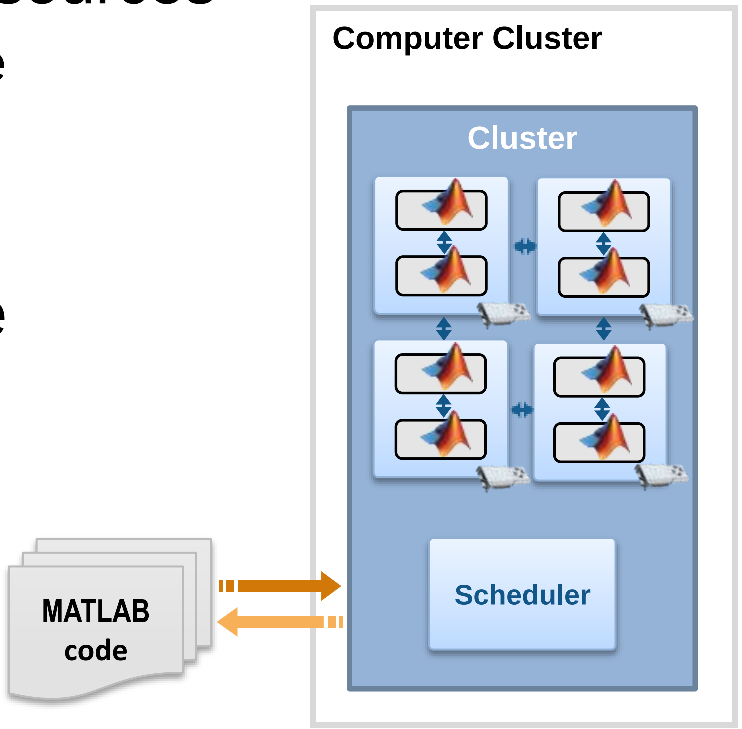
Take Advantage of Cluster Hardware

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

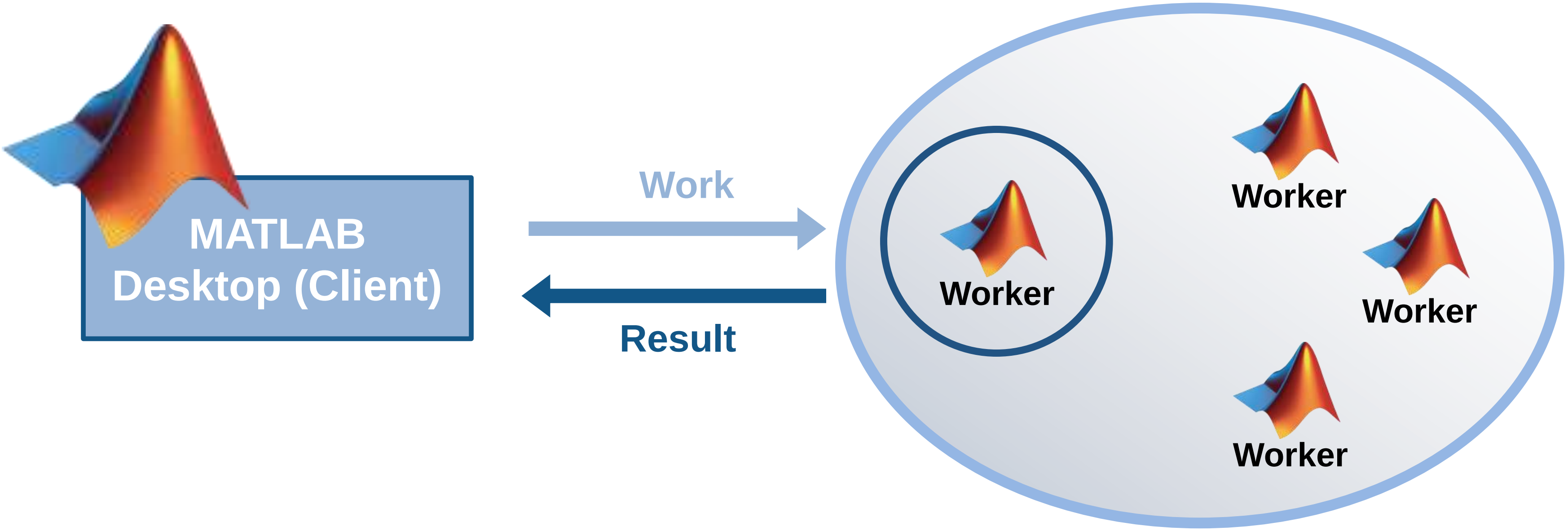


Offloading Computations

- Send desktop code to cluster resources
  - No parallelism required within code
  - Submit directly from MATLAB
- Leverage supplied infrastructure
  - File transfer / path augmentation
  - Job monitoring
  - Simplified retrieval of results
- Scale offloaded computations

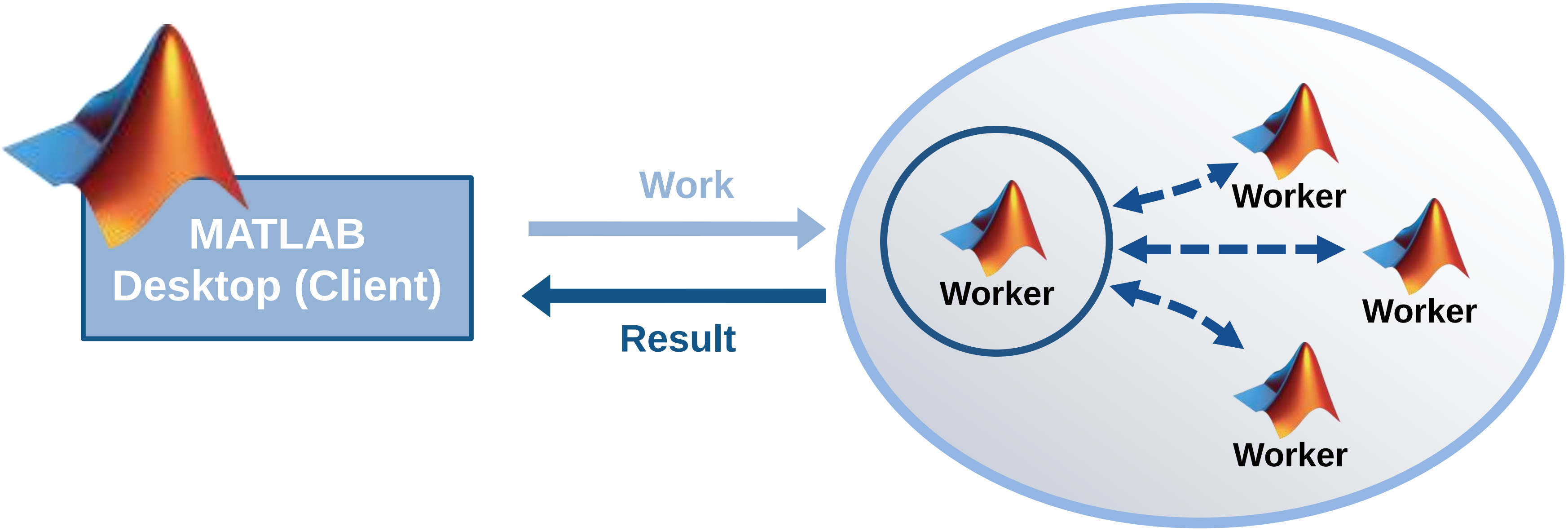


Offload Computations with `batch`



`batch(...)`

Offload and Scale Computations with `batch`

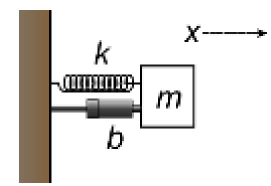


`batch(…,'Pool',…)`

Example: Parameter Sweep of ODEs

Offload and Scale Processing

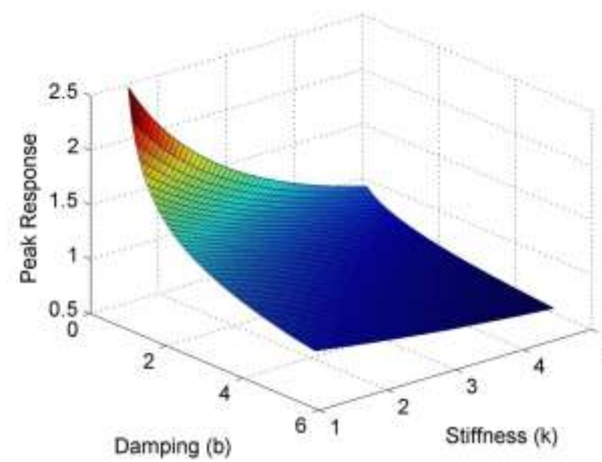
$$\overbrace{m}^{5}\ddot{x} + \underbrace{b}_{1,2,\ldots}\dot{x} + \underbrace{k}_{1,2,\ldots}x = 0$$



- Offload processing to workers:
  `batch`

- Scale offloaded processing:
  `batch(…,'Pool',…)`

- Retrieve results from job:
  `fetchOutputs`

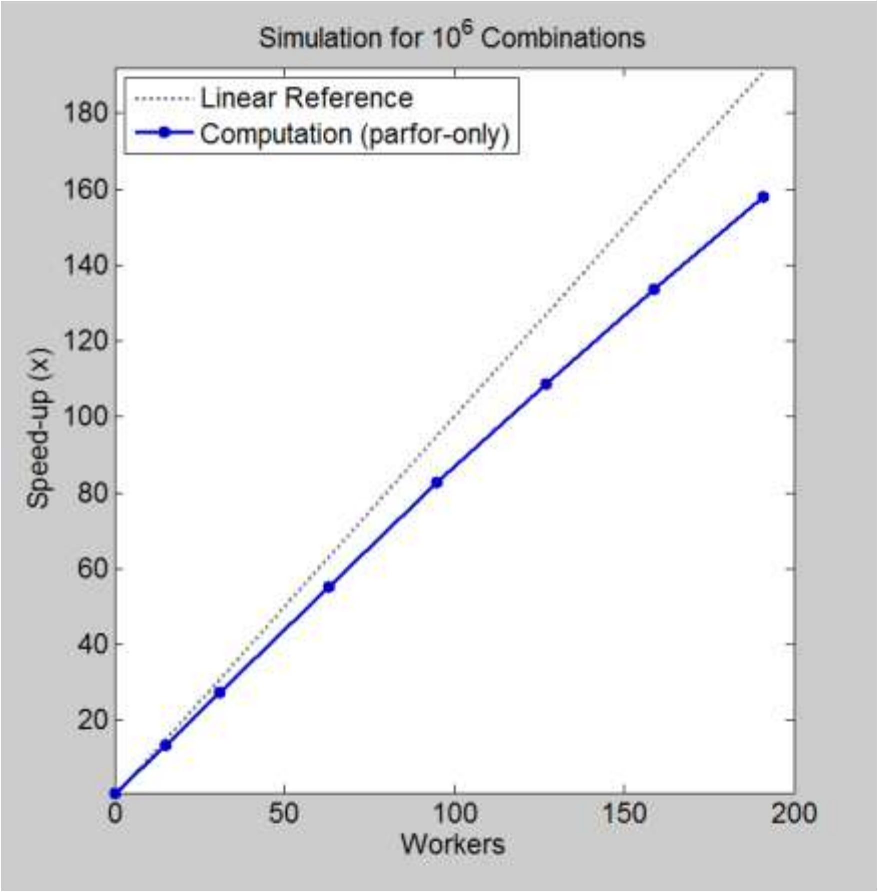


Benchmark: Parameter Sweep of ODEs

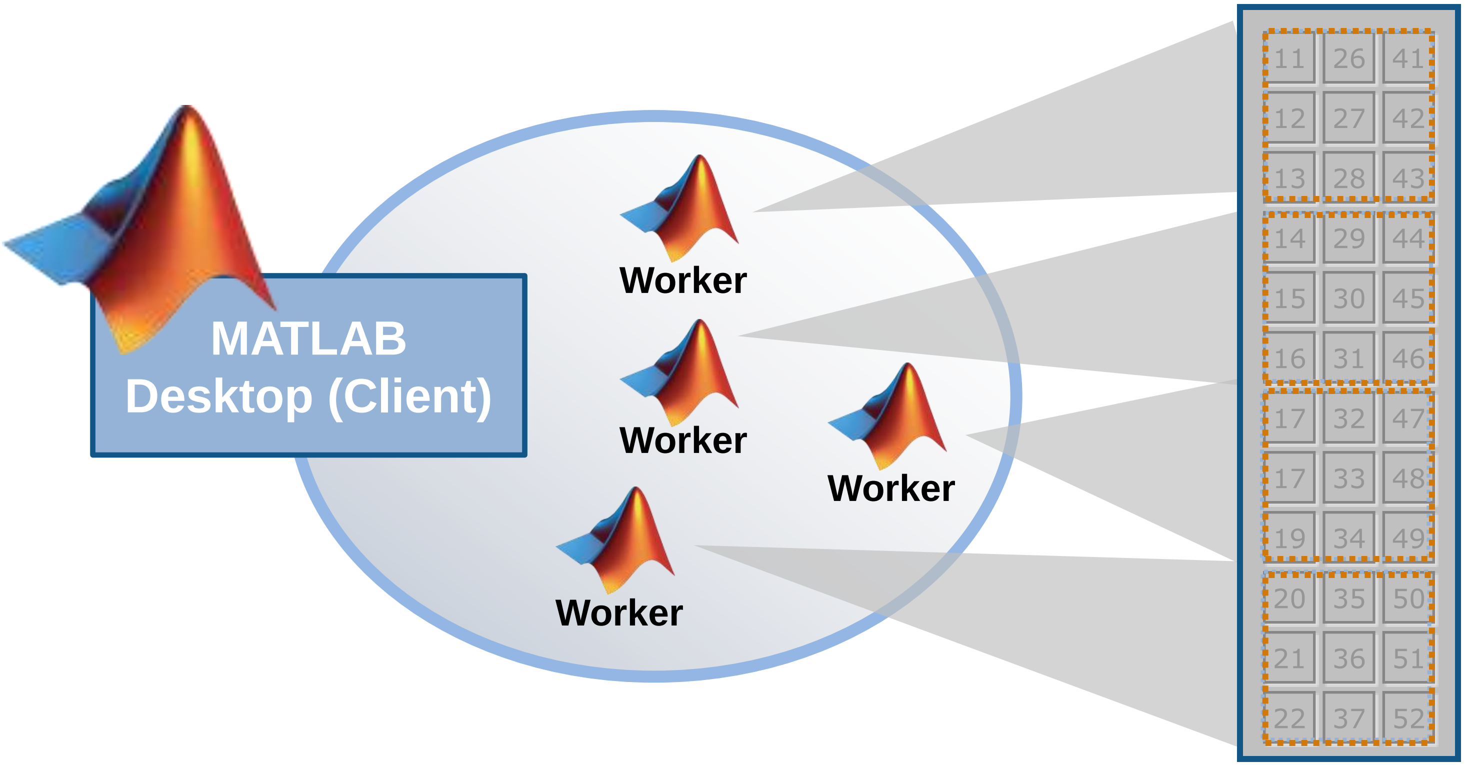
Scaling case study for a fixed problem size with a cluster

Workers	Computation (minutes)	Speed-up
1	173	1
16	13	13
32	6.4	27
64	3.2	55
96	2.1	83
128	1.6	109
160	1.3	134
192	1.1	158

Processor: Intel Xeon E5-2670
16 cores per node



Distributing Large Data



Remotely Manipulate Array
from Client

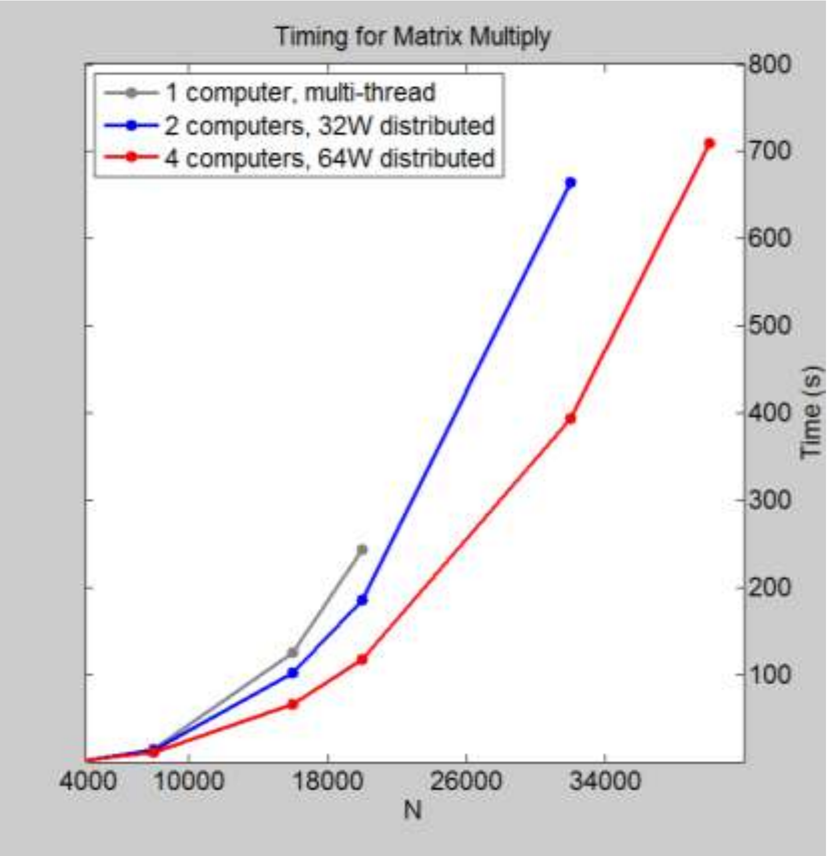
Distributed Array
Lives on the Workers

Investigation: Distributed Calculations

Effect of number of computers on execution time

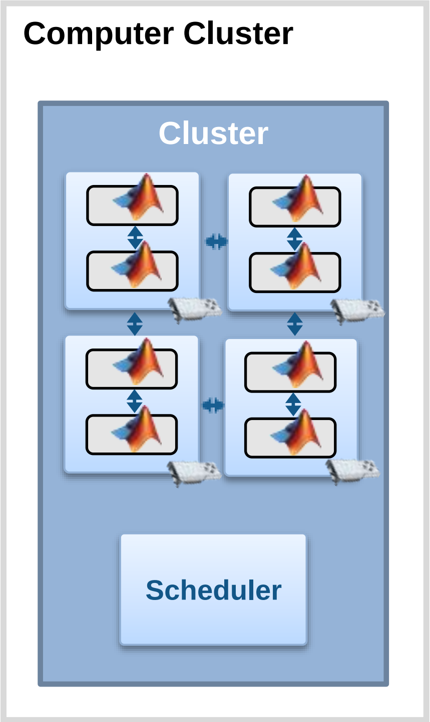
N	Time (s)		
	1 node, multi-threaded	Distributed	
		2 nodes, 32W	4 nodes, 64W
4000	2	3	3
8000	16	14	12
16000	126	102	67
20000	244	187	118
32000	-	664	394
40000	-	-	710

Processor: Intel Xeon E5-2670
16 cores, 60 GB RAM per compute node
10 Gigabit Ethernet



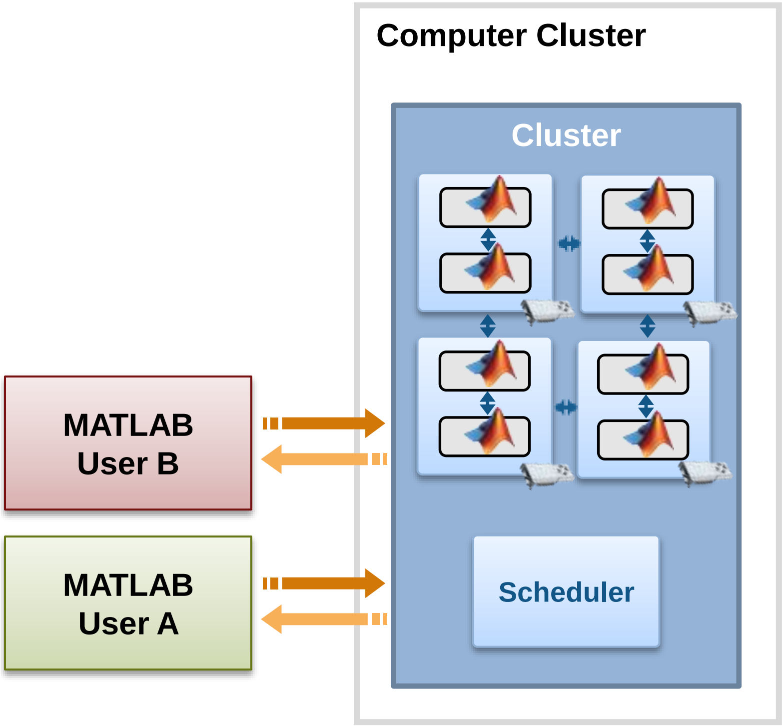
MATLAB Distributed Computing Server

- Extension of desktop parallel computing
- Pre-built framework and infrastructure
- Simplified license and maintenance



Dynamic Licensing Model

- Users have access to their licensed products
- Server does not check out any licenses on the client
- User can exit MATLAB once the job is queued



Job Schedulers

Ease of Use

- MathWorks Job Scheduler
- Direct support for specific schedulers
  (Platform LSF, Microsoft HPCS, PBS)
- Open API to support other schedulers

Greater Control

www.mathworks.com/products/distriben/supported

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

For more information

Visit

http://www.mathworks.com/products/parallel-computing