MathWorks

Optimizing and Accelerating Your MATLAB Code

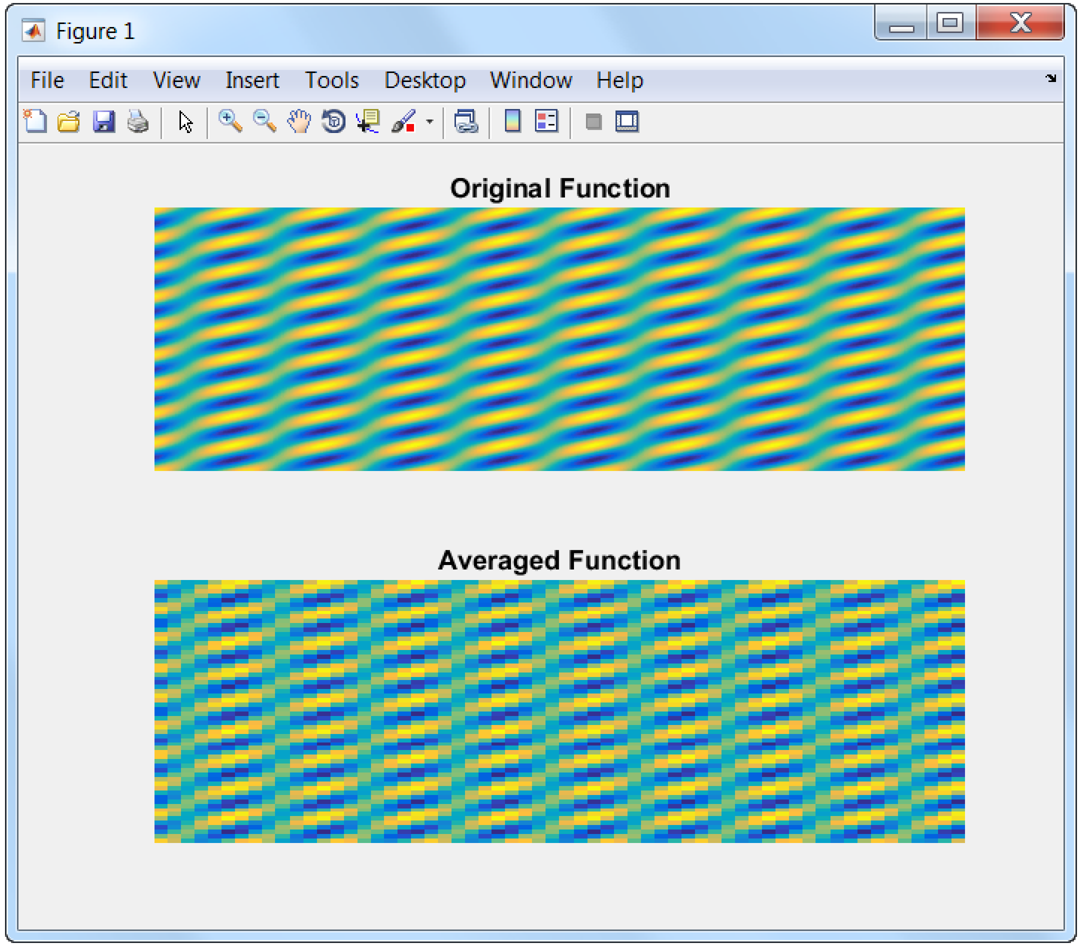
Sofia Mosesson
Senior Application Engineer

Agenda

- Optimizing `for` loops and using vector and matrix operations
- Indexing in different ways
- Finding and addressing bottlenecks
- Generating C code and incorporating it into your application
- Utilizing additional hardware and processing power

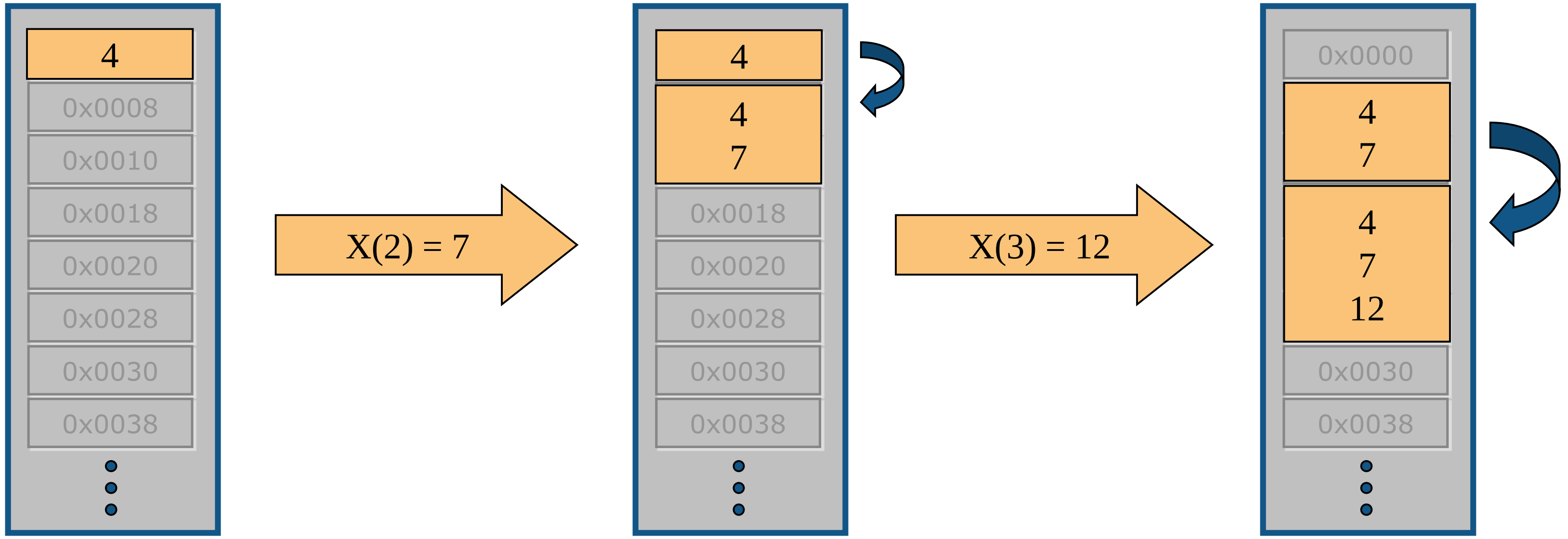
Example: Block Processing Images

- Calculate a function at grid points
- Take the mean of larger blocks
- Analyze and improve performance



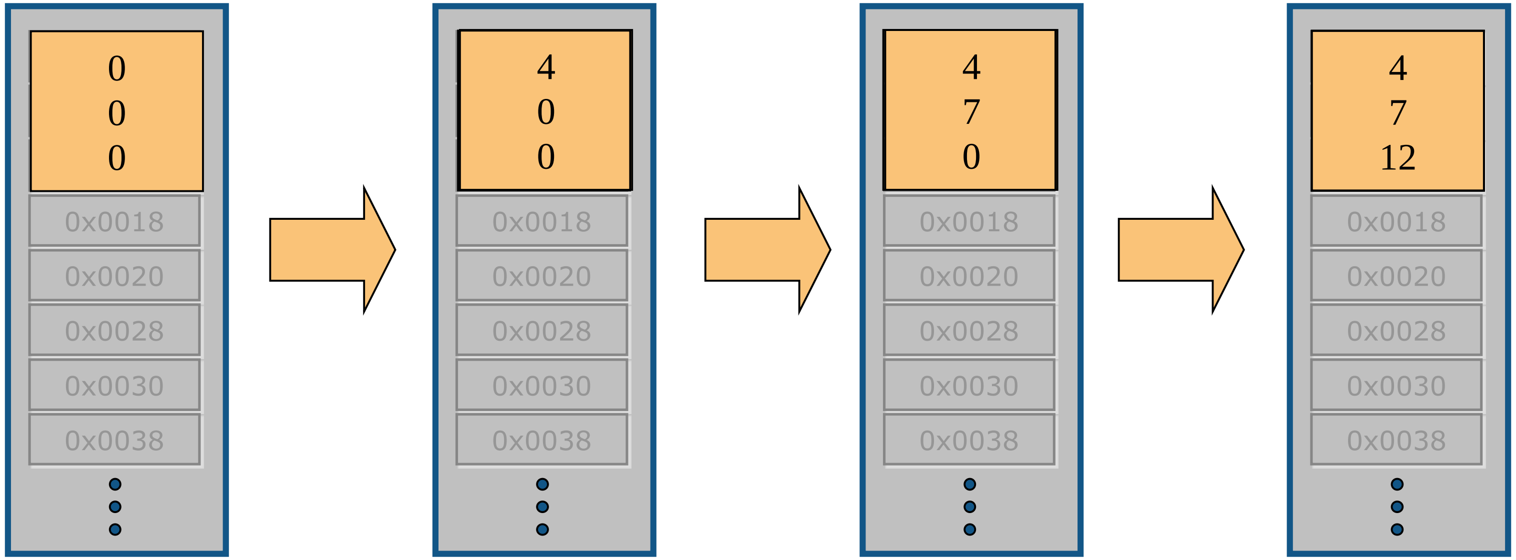
Effect of Not Preallocating Memory

```
x(1) = 4
x(2) = 7
x(3) = 12
```



Benefit of Preallocation

```
x = zeros(3,1)
x(1) = 4
x(2) = 7
x(3) = 12
```



MATLAB Underlying Technologies

- Execution Engine (>=R2015b)
  - All MATLAB code is just-in-time compiled
  - Improves “Nth run” performance
- Commercial Libraries
  - BLAS: Basic Linear Algebra Subroutines
  - LAPACK: Linear Algebra Package
  - IPP: Intel Performance Primitives
  - FFTW: Fastest Fourier Transform in the West

Other Best Practices

- Avoid "`clear all`"
  - Use "`clear`" or "`clearvars`" instead
- Avoid "introspection" functions
  - E.g. "`dbstack`", "`inputname`", "`exist`", "`whos`"

http://www.mathworks.com/help/releases/R2015b/matlab/matlab_prog/techniques-for-improving-performance.html

Agenda

- Optimizing `for` loops and using vector and matrix operations
- Indexing in different ways
- Finding and addressing bottlenecks
- Generating C code and incorporating it into your application
- Utilizing additional hardware and processing power

Exampel: Indexing

- Loop over rows versus columns
- Use `find` command
- Use logical indexing

x

0.123	0.469	0.811
0.532	0.103	0.775
0.320	0.382	0.295

```
y = x;
if x(r,c) < 0.5
     y(r,c) = 0
end
```

y

0.123	0.469	0.811
0.532	0.103	0.775
0.320	0.382	0.295

0	0	0.811
0.532	0	0.775
0	0	0

Matrix processing in memory

What is the fastest way to process MATLAB matrices with `for` loops? (i.e. de-vectorize)

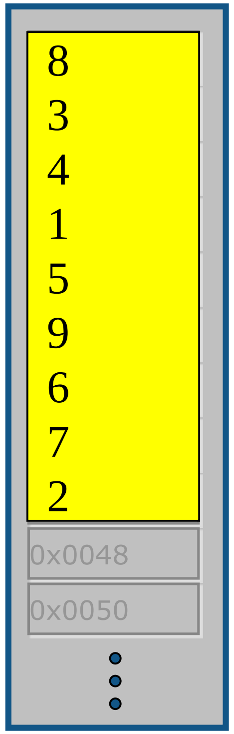
Column-Major Memory Storage

a) Down the columns

b) Along the rows

c) Doesn't matter

```
>> x = magic(3)
x =
        8     1     6
        3     5     7
        4     9     2
```

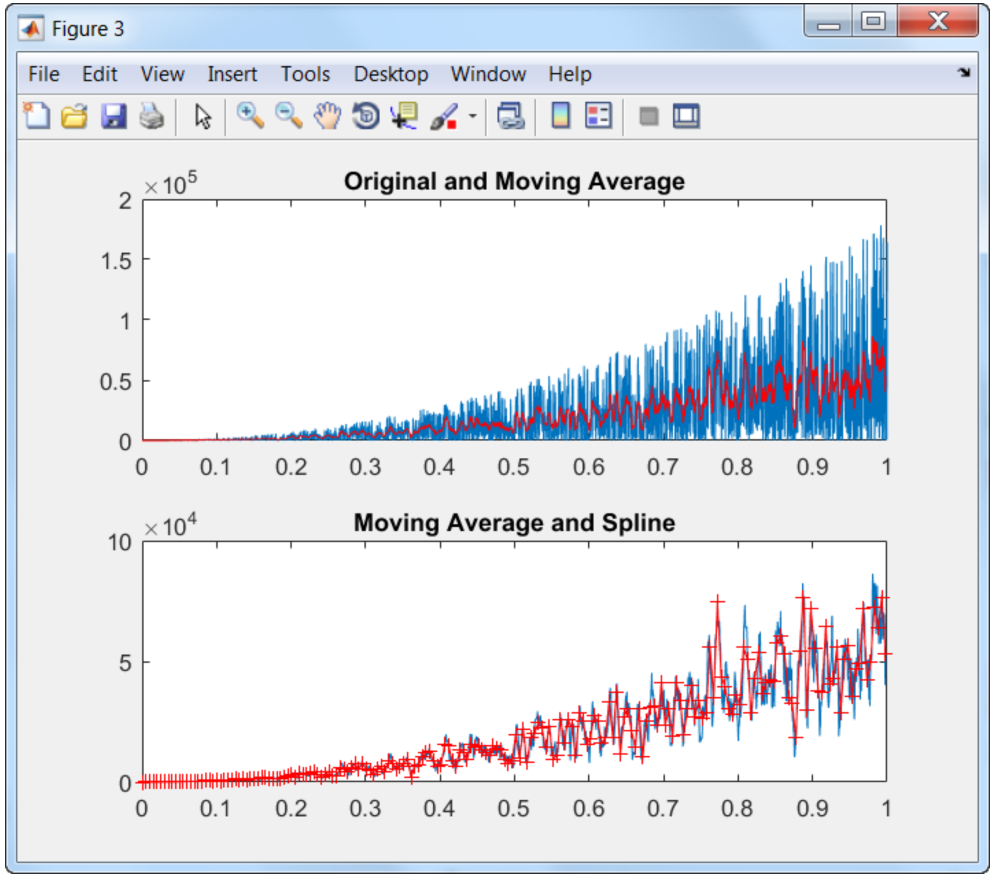


Agenda

- Optimizing `for` loops and using vector and matrix operations
- Indexing in different ways
- $\boxed{\text{Finding and addressing bottlenecks}}$
- Generating C code and incorporating it into your application
- Utilizing additional hardware and processing power

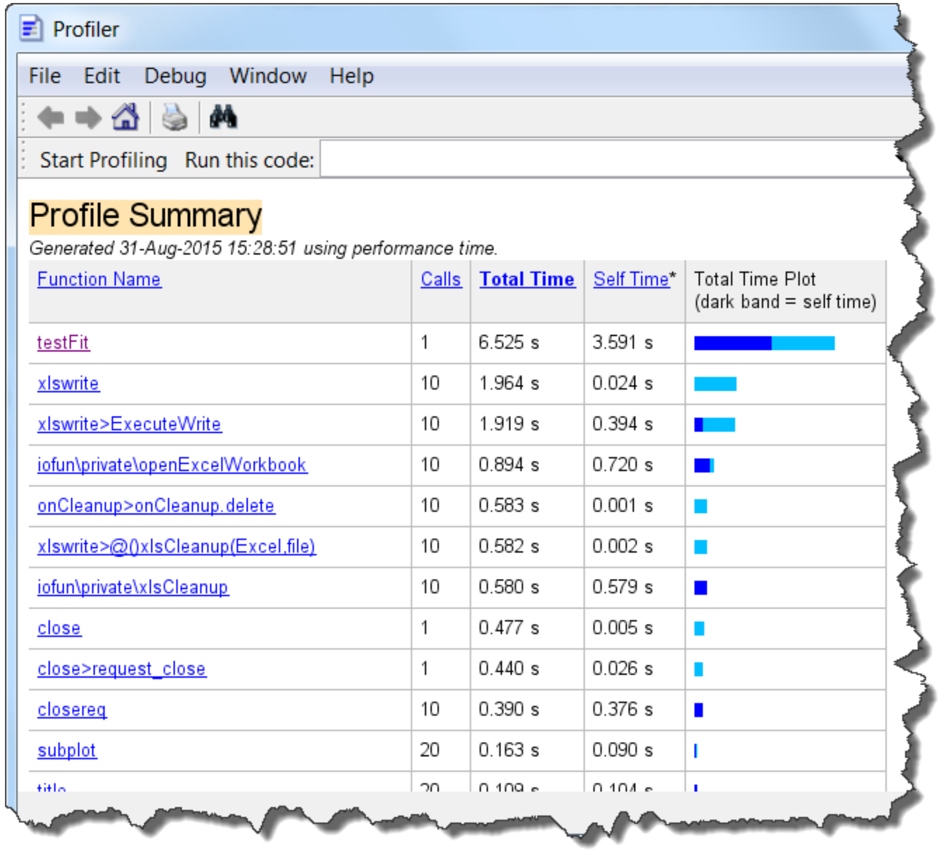
Example: Addressing Bottlenecks

- Run and time program
- Identify bottlenecks
- Improve run time



Profiler

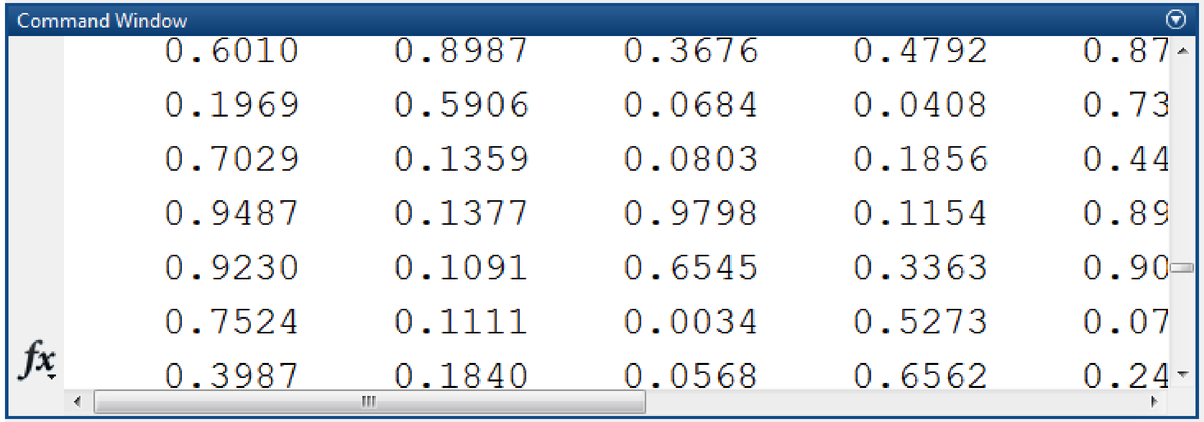
- Total number of function calls
- Time per function call
- Self time in a function call
- Statement coverage of code



Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
testFit	1	6.525 s	3.591 s	
xlswrite	10	1.964 s	0.024 s	
xlswrite>ExecuteWrite	10	1.919 s	0.394 s	
iofun\private\openExcelWorkbook	10	0.894 s	0.720 s	
onCleanup>onCleanup.delete	10	0.583 s	0.001 s	
xlswrite>@()xlsCleanup(Excel,file)	10	0.582 s	0.002 s	
iofun\private\xlsCleanup	10	0.580 s	0.579 s	
close	1	0.477 s	0.005 s	
close>request_close	1	0.440 s	0.026 s	
closereq	10	0.390 s	0.376 s	
subplot	20	0.163 s	0.090 s	

Best Practices

- Minimize file I/O
- Reuse existing graphics components
- Avoid printing to Command Window



Agenda

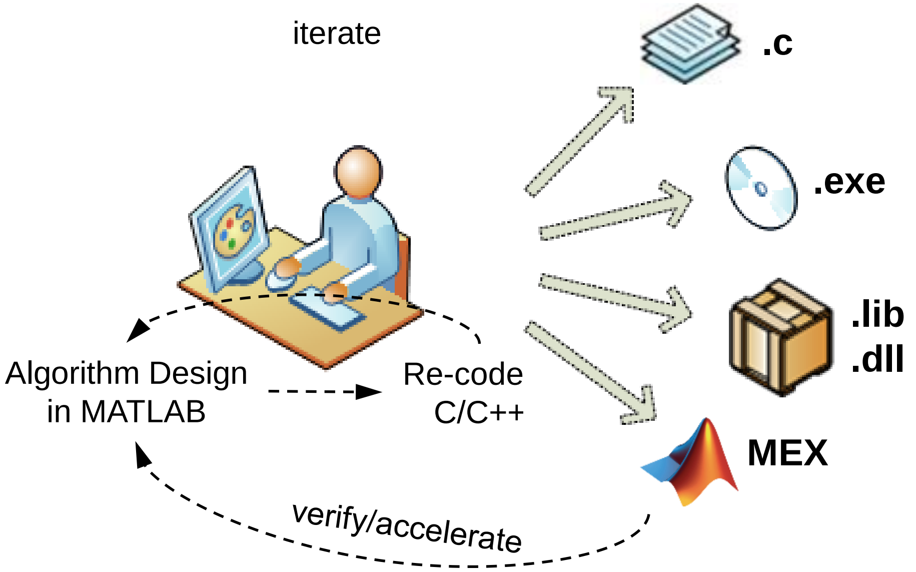
- Optimizing `for` loops and using vector and matrix operations
- Indexing in different ways
- Finding and addressing bottlenecks
- Generating C code and incorporating it into your application
- Utilizing additional hardware and processing power

Why Engineers Translate MATLAB to C

- Implement C code on processors or hand off to software engineers .c
- Accelerate MATLAB algorithms MEX

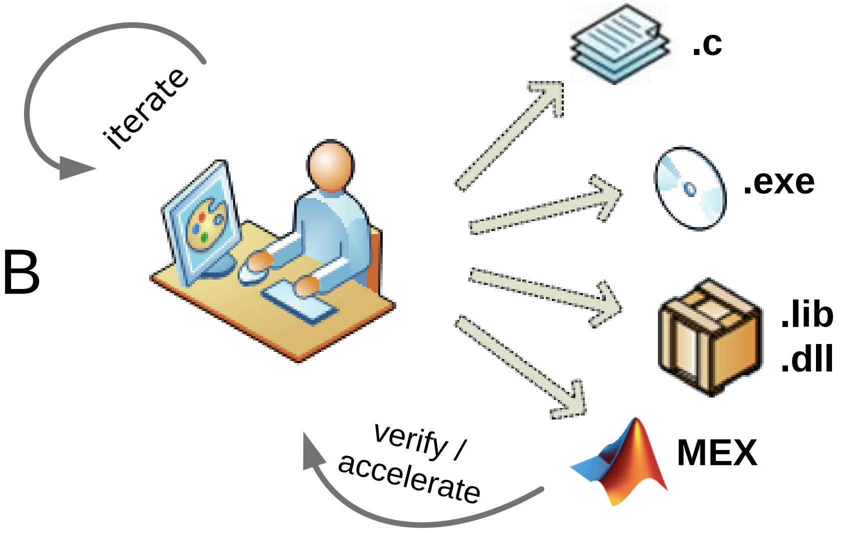
Challenges with Manual Translation of MATLAB to C

- Separate functional and implementation specifications
  - Leads to multiple implementations which are inconsistent
  - Hard to modify requirements during development
  - Difficult to keep MATLAB code and C code in sync
- Manual coding errors
- Time consuming and expensive process



Automatic Translation of MATLAB to C

- Maintain one design in MATLAB
- Design faster and get to C quickly
- Test more systematically and frequently
- Spend more time improving algorithms in MATLAB



Acceleration Using MEX

- Speedup factor will vary
- When you **may** see a speedup:
  - Often for communications or signal processing
  - Likely for loops with states or when vectorization is not possible
  - Always for fixed point
- When you **may not** see a speedup:
  - MATLAB implicitly multithreads computation
  - Built in functions that call BLAS or IPP

Supported Language Features and Functions

- New functions and features are supported each release

Matrices and Arrays	Data Types	Programming Constructs	Functions

• Matrix operations
• N-dimensional arrays
• Subscripting
• Frames
• Persistent variables
• Global variables
	• Complex numbers
	• Integer math
	• Double/single-precision
	• Fixed-point arithmetic
	• Characters
	• Structures
	• Cell arrays
	• Numeric class
	• Variable-sized data
	• MATLAB Class
	• System objects
		• Arithmetic, relational, and logical operators
		• Program control (`if`, `for`, `while`, `switch`)
			• MATLAB functions and subfunctions
			• Variable-length argument lists
			• Function handles
			
			Supported algorithms
			• More than 1100 MATLAB operators (R2015b), functions, and System objects for:
			  • Communications
			  • Computer vision
			  • Image processing
			  • Phased Array signal processing
			  • Robotics System Toolbox
			  • Signal processing
			  • Statistic & Machine Learning Toolbox

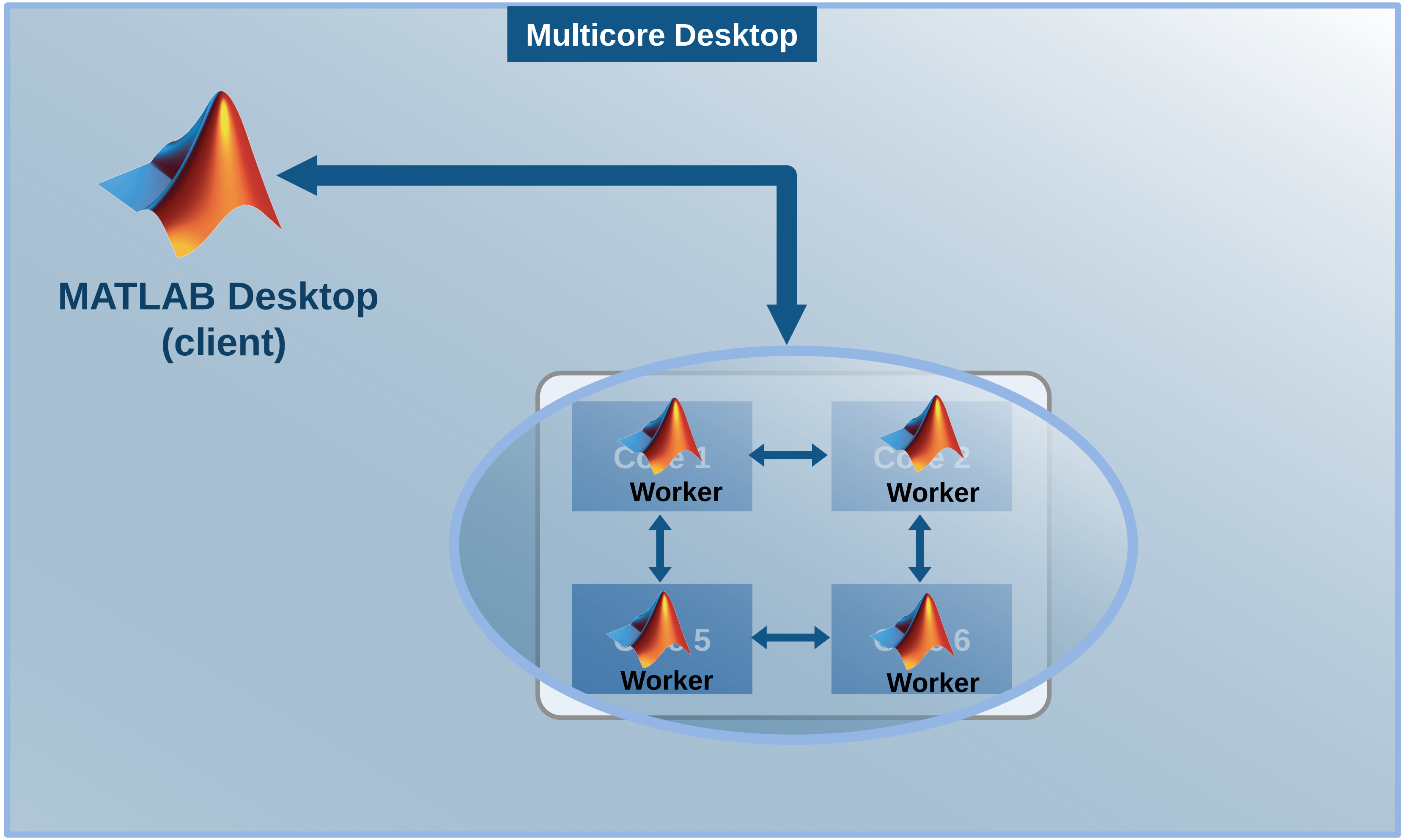
http://www.mathworks.com/help/coder/language-supported-for-code-generation.html

Agenda

- Optimizing `for` loops and using vector and matrix operations
- Indexing in different ways
- Finding and addressing bottlenecks
- Generating C code and incorporating it into your application
- Utilizing additional hardware and processing power

Parallel Computing Paradigm

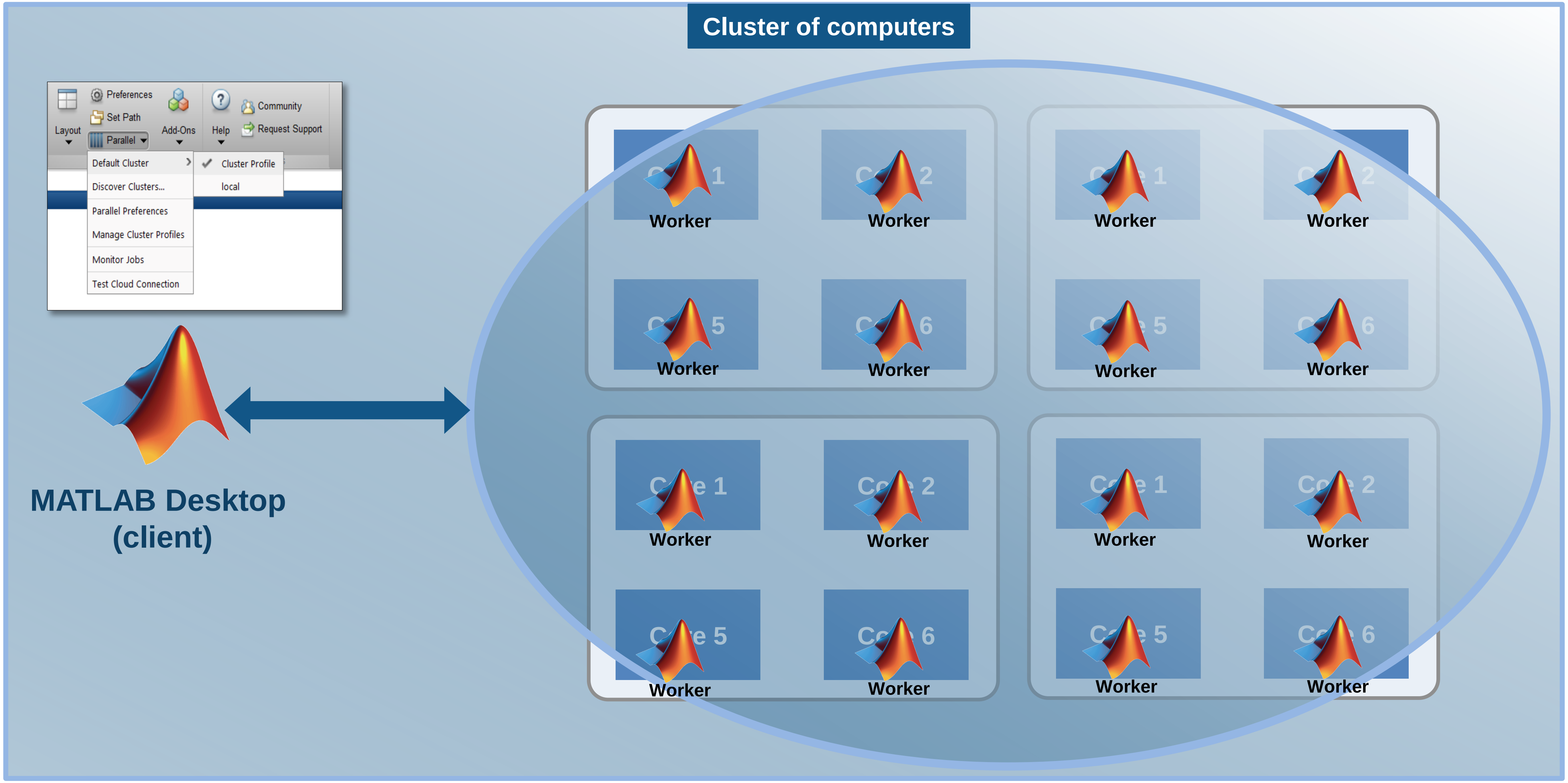
Multicore Desktops



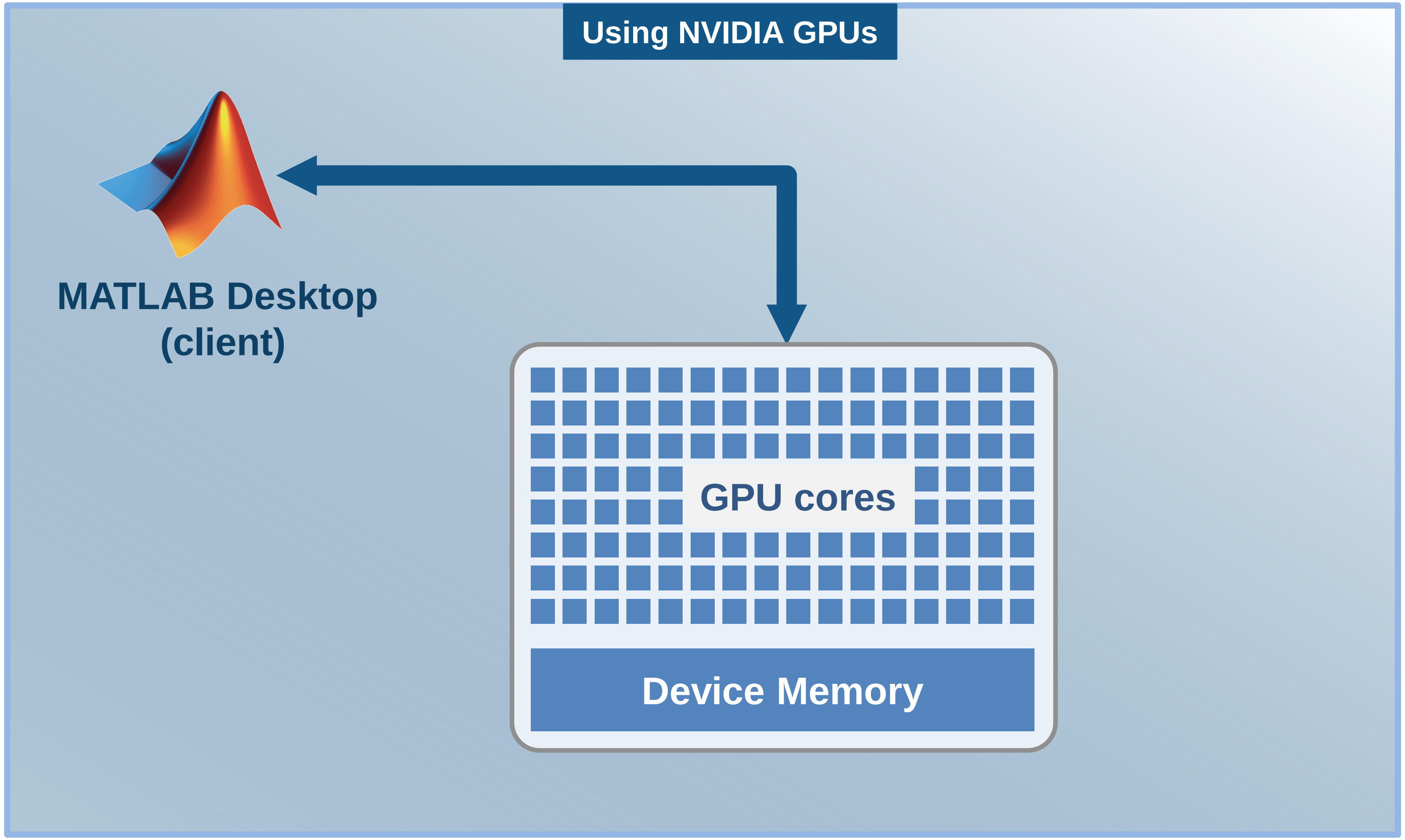
[MATLAB multicore]

Parallel Computing Paradigm

Cluster Hardware

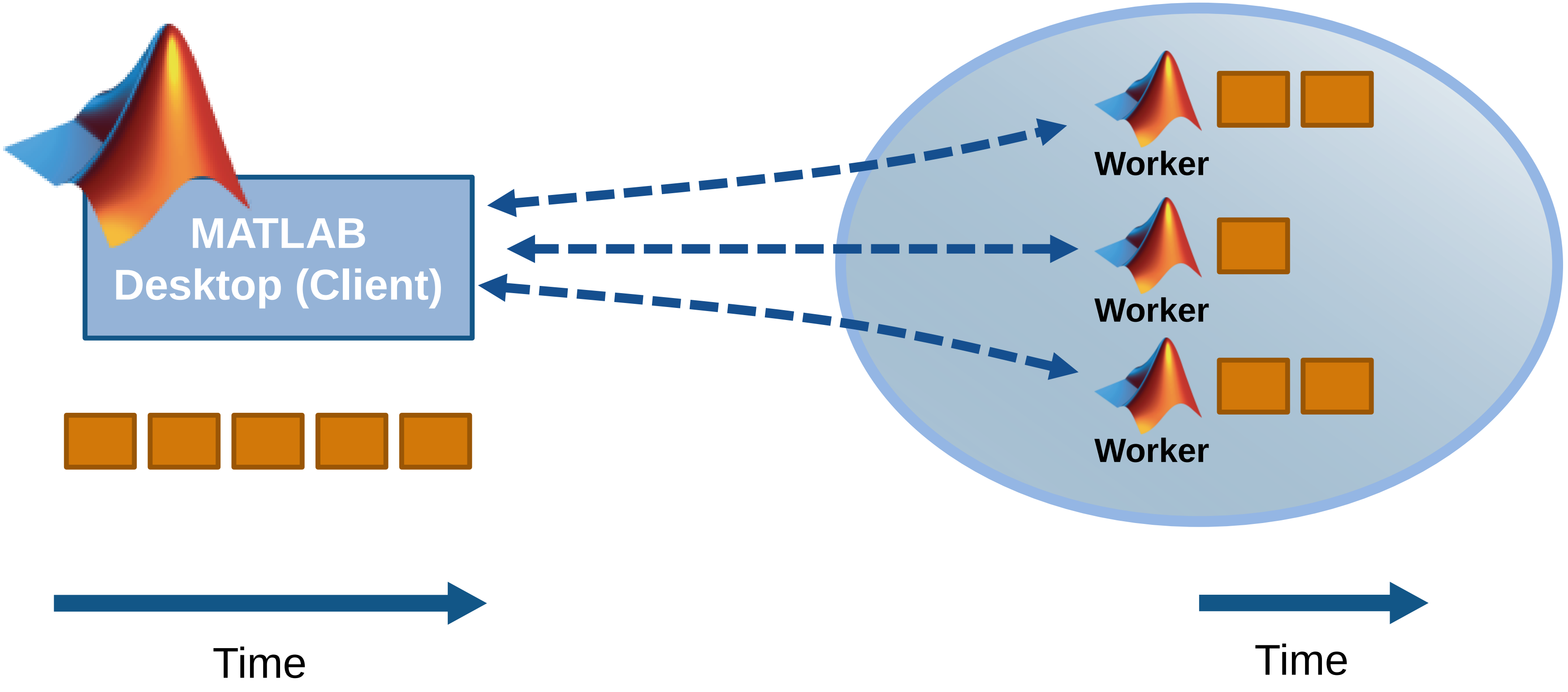


Parallel Computing Paradigm
NVIDIA GPUs



Explicit Parallelism: Independent Tasks or Iterations
Parallel for loops

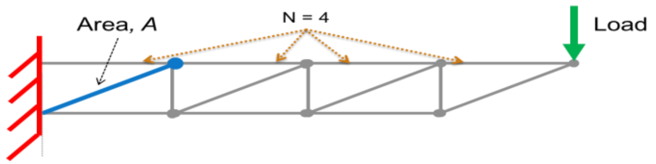
- Ideal problem for parallel computing
- No dependencies or communications between tasks
- Examples: parameter sweeps, Monte Carlo simulations

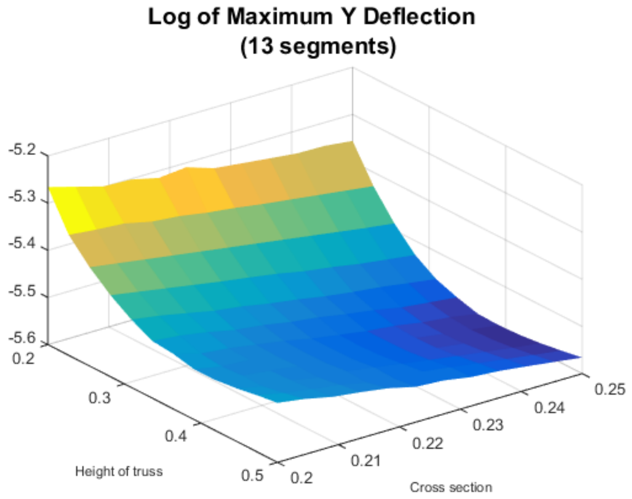


Example: Parameter Sweep
Parallel for-loops

$$M\ddot{x} + C\dot{x} + Kx = F$$

- Deflection of customizable truss
  - Initial dynamic load, damping
  - Parameters:
    - Height of truss
    - Cross sectional area of truss elements



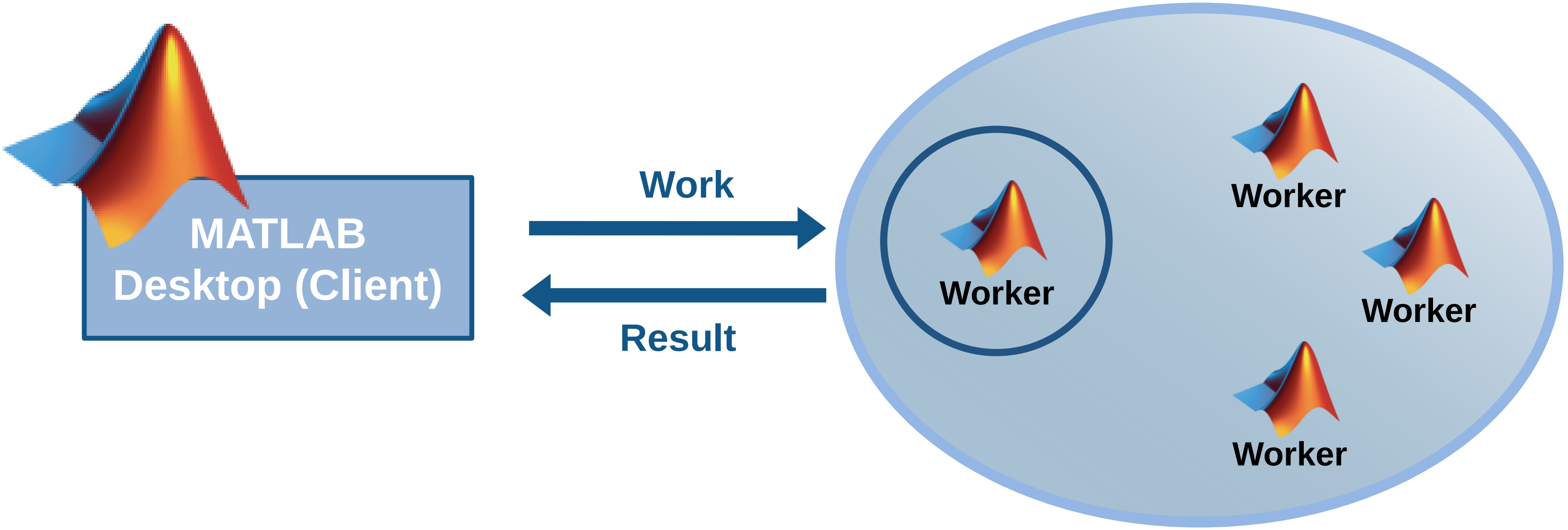


- Convert for to parfor

- Use pool of MATLAB workers

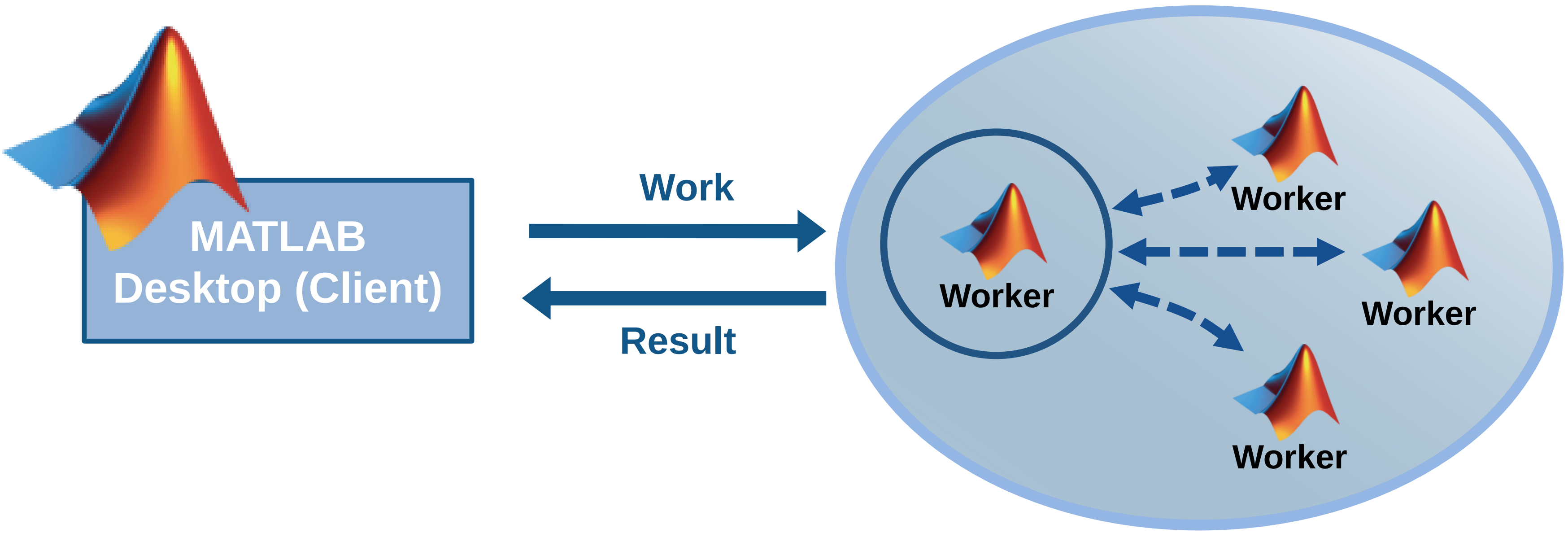
Offloading Serial Computations

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



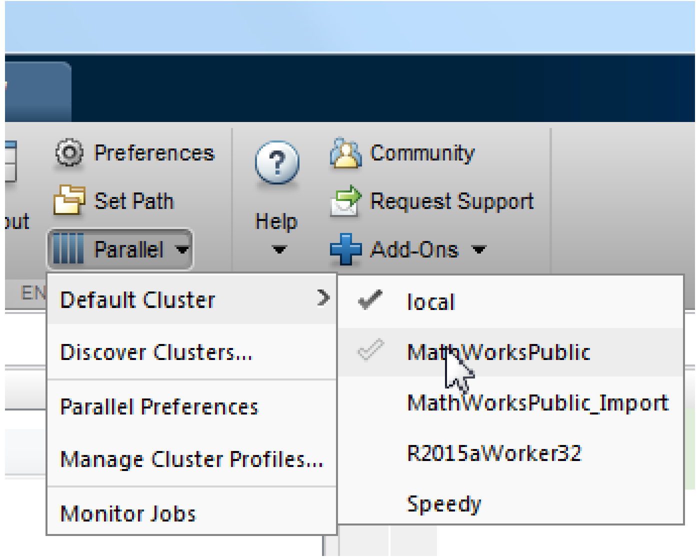
Offloading and Scaling Computations

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



Migrate to Cluster / Cloud

- Use MATLAB Distributed Computing Server
- Change hardware without changing algorithm



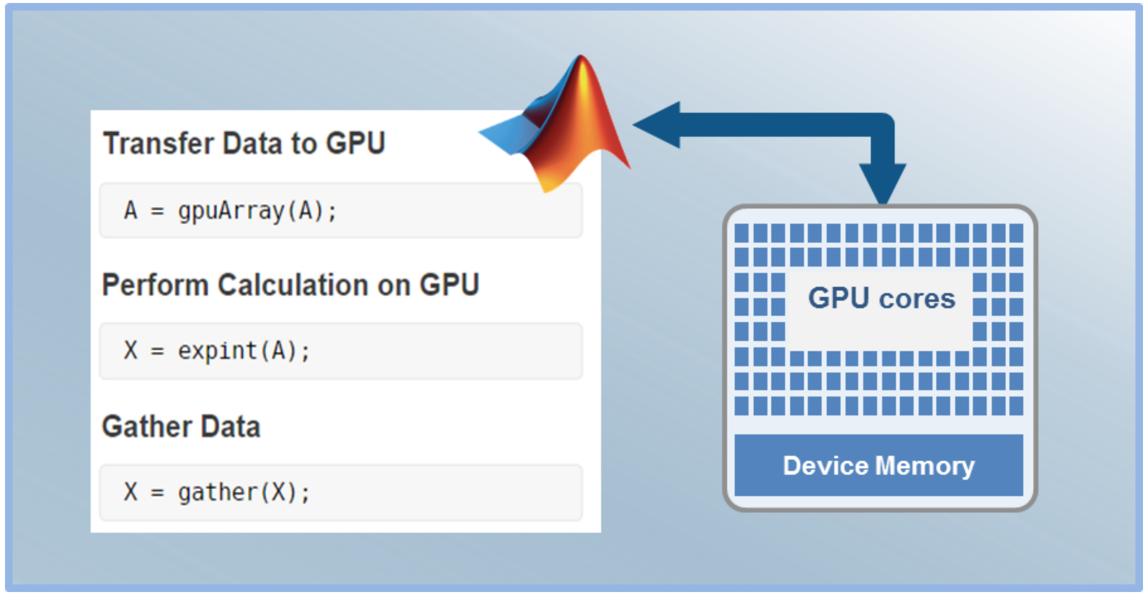
Speed-up using NVIDIA GPUs

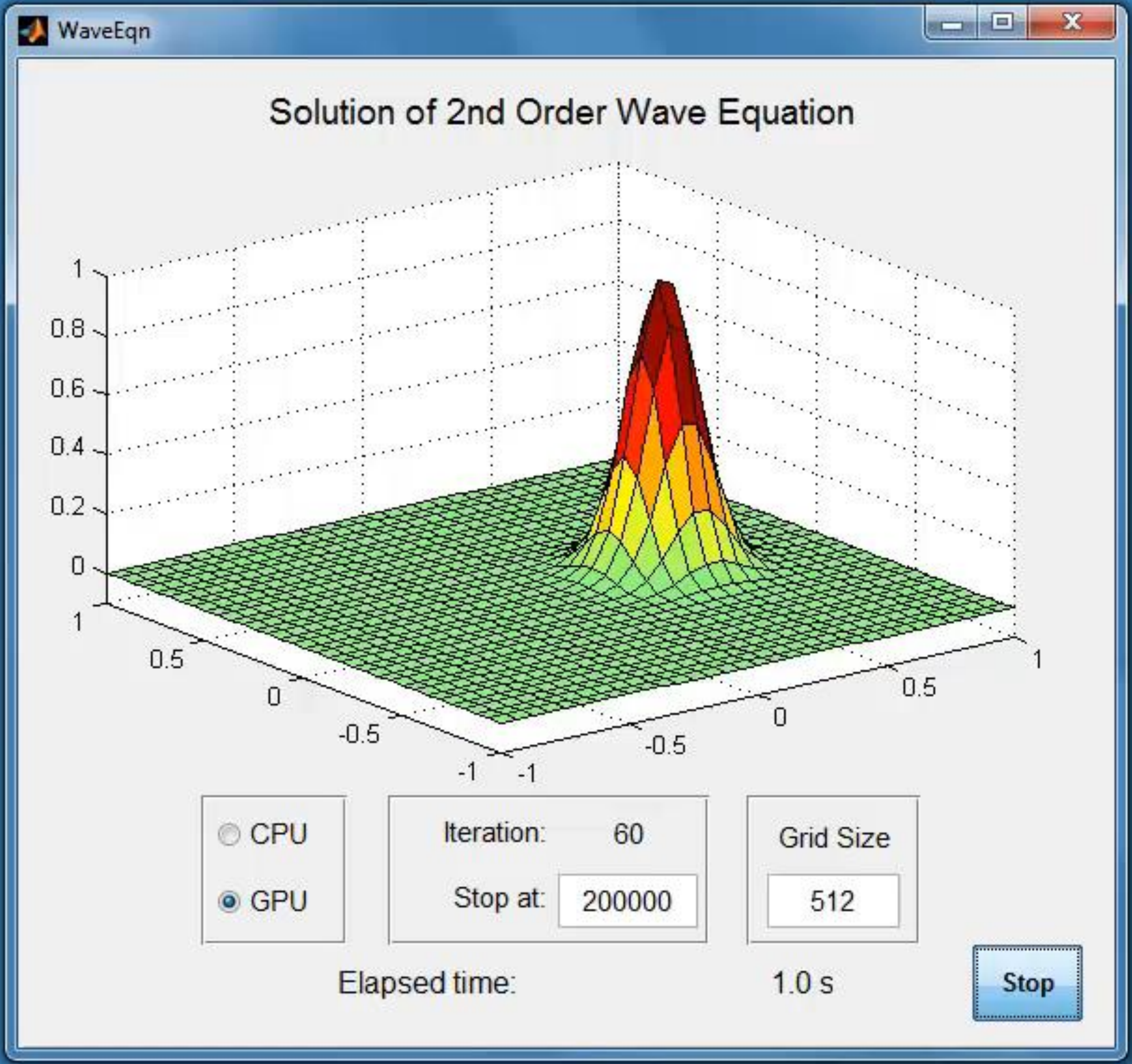
- Ideal Problems
  - Massively Parallel and/or Vectorized operations
  - Computationally Intensive
  - Algorithm consists of supported functions

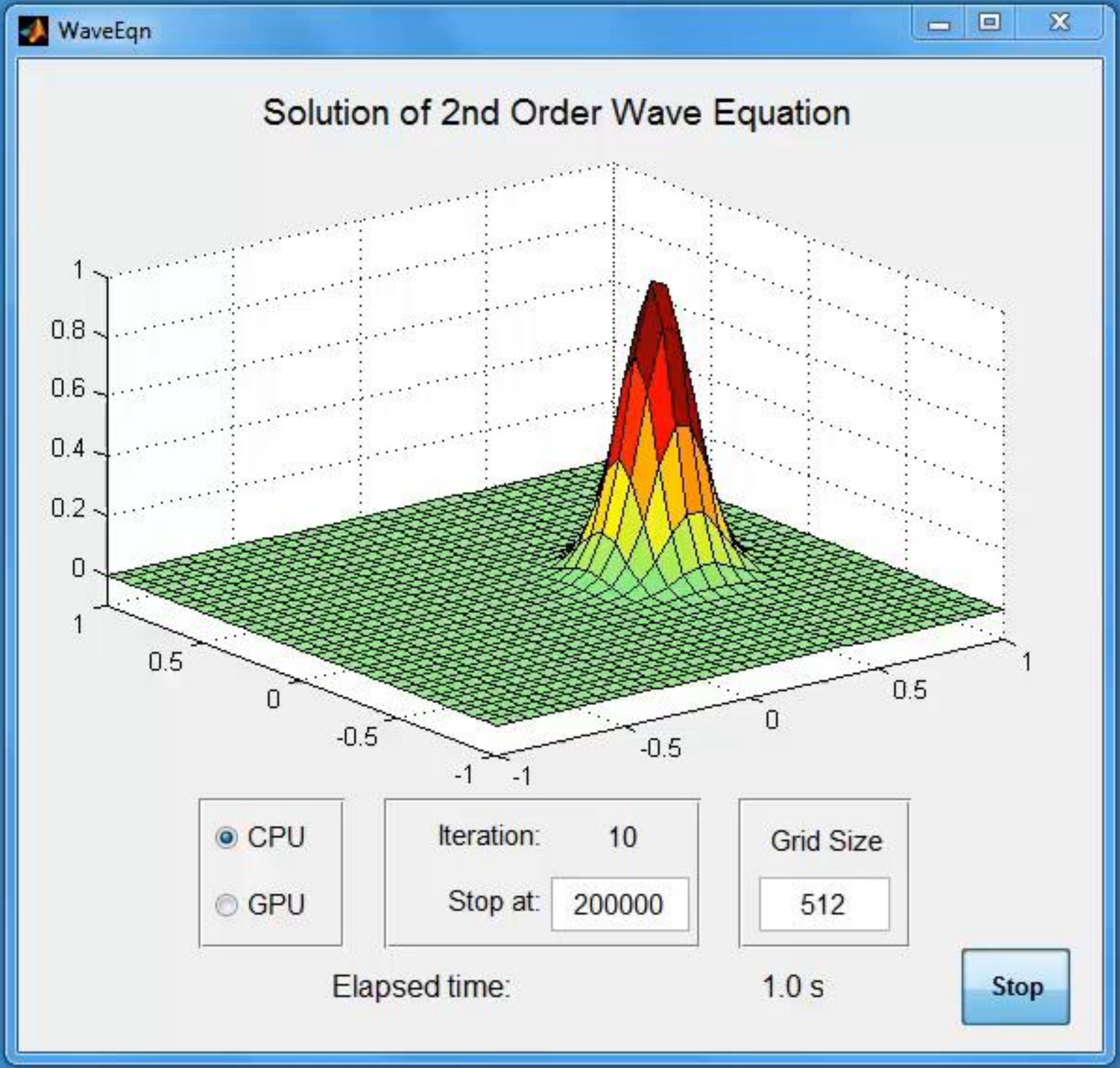
- 300+ GPU-enabled MATLAB functions

- Additional GPU-enabled Toolboxes
  - Neural Networks
  - Image Processing
  - Communications
  - Signal Processing

  ..... Learn More




WaveEqn
Solution of 2nd Order Wave Equation
CPU
GPU
Iteration: 60
Stop at: 200000
Grid Size
512
Elapsed time: 1.0 s
Stop


WaveEqn
Solution of 2nd Order Wave Equation
CPU
GPU
Iteration: 10
Stop at: 200000
Grid Size
512
Elapsed time: 1.0 s
Stop

Agenda

- Optimizing `for` loops and using vector and matrix operations
- Indexing in different ways
- Finding and addressing bottlenecks
- Generating C code and incorporating it into your application
- Utilizing additional hardware and processing power

Key Takeaways

- Consider the performance benefits of vector and matrix operations
- Analyze your code for bottlenecks to address the critical areas
- Leverage MATLAB Coder to speed up functions with generated C code
- Leverage parallel computing tools to take advantage of additional hardware

Some Other Valuable Resources

- MATLAB Documentation
  - MATLAB → Advanced Software Development → Performance and Memory
- Accelerating MATLAB algorithms and applications
  - http://www.mathworks.com/company/newsletters/articles/accelerating-matlab-algorithms-and-applications.html
- Loren Shure's Blog: "The Art of MATLAB"
  - http://blogs.mathworks.com/loren/
- MATLAB Question and Answers Site: MATLAB Answers
  - http://www.mathworks.com/matlabcentral/answers/