The MathWorks™

MATLAB® & SIMULINK®

Speeding Up MATLAB Applications

Cold Spring Harbor Laboratory
6th July, 2009

Heather Wellman Senior Account Manager

Asawari Samant Application Engineer

Further information

- Stay for questions
- Bookmark MATLAB Central as a great resource for shared code
- Trials, onsite demonstrations, technical literature and licensing:

Heather Wellman
Email: heather.wellman@mathworks.com
Direct: (508) 647-7093

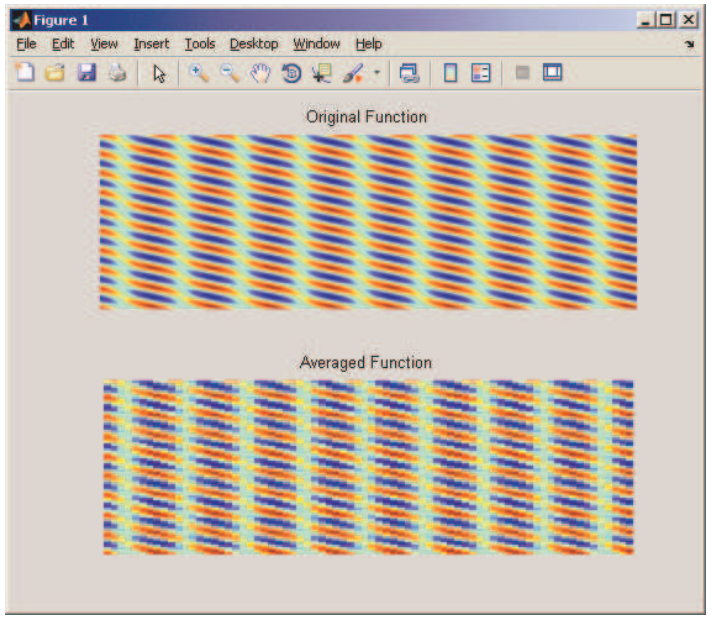
- Company and product information: **www.mathworks.com**

Agenda

- Leveraging the power of vector & matrix operations
- Addressing bottlenecks
- Utilizing additional processing power
- Summary

Example: Block Processing Images

- Evaluate function at grid points
- Reevaluate function over larger blocks
- Compare the results
- Evaluate code performance



Summary of Example

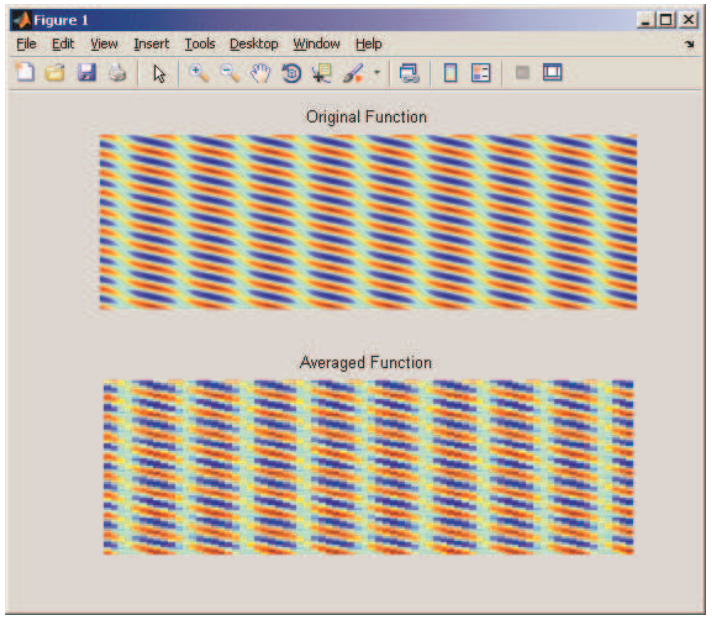
- Used built-in timing functions

```
>> tic
>> toc
```

- Used M-Lint to find suboptimal code

- Preallocated arrays

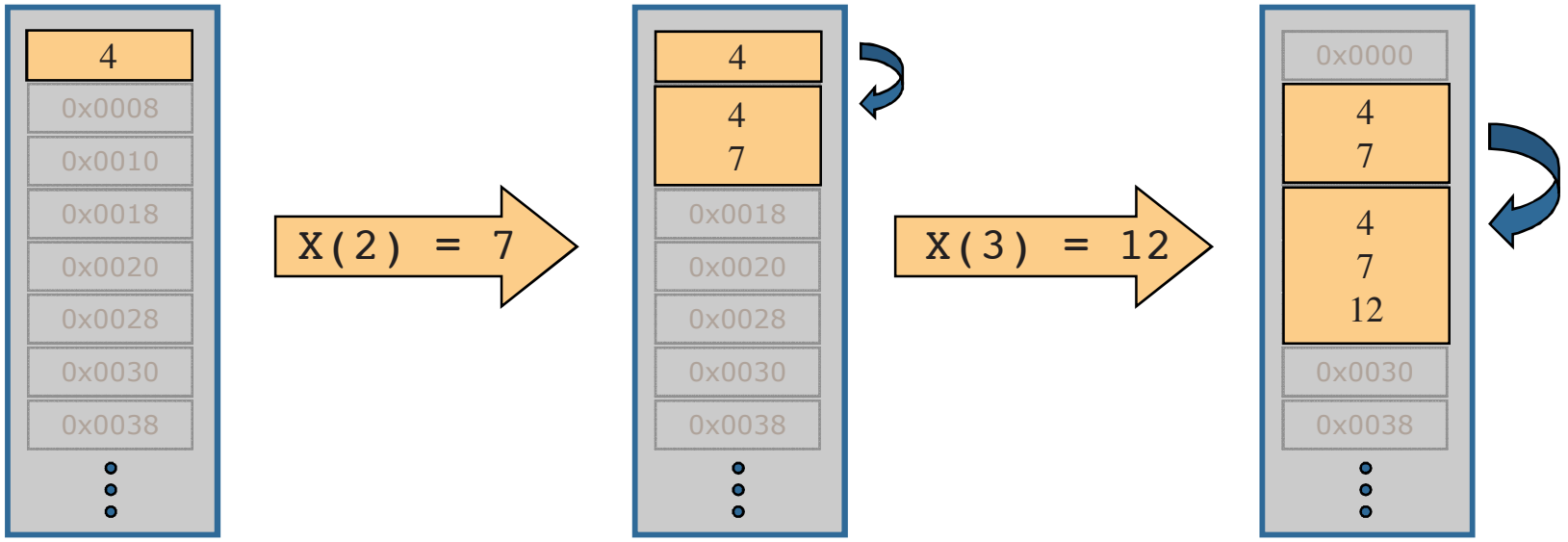
- Vectorized code



Effect of Not Preallocating Memory

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

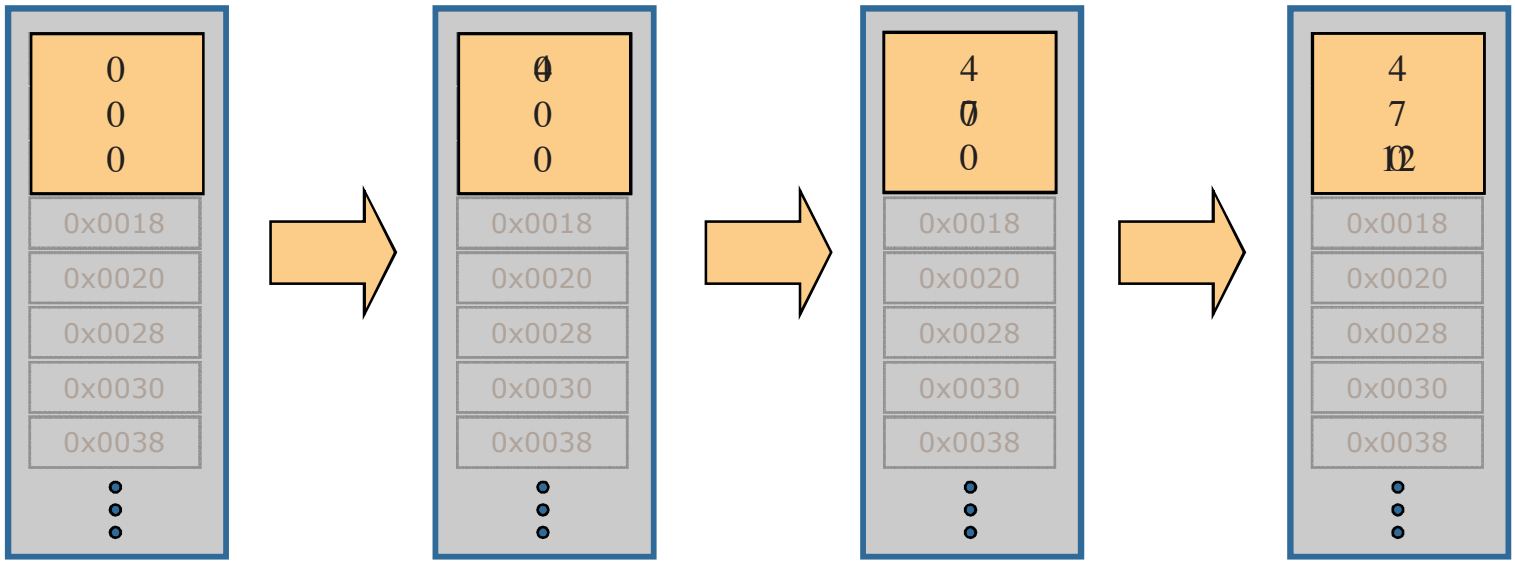
Resizing Arrays is Expensive



Benefit of Preallocation

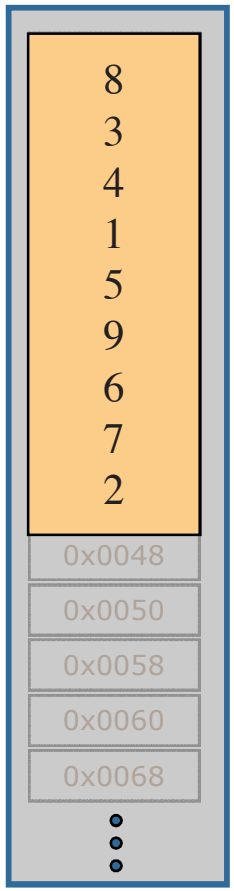
```
>> x = zeros(3,1)
>> x(1) = 4
>> x(2) = 7
>> x(3) = 12
```

Reduced Memory Operations



Data Storage of MATLAB Arrays

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

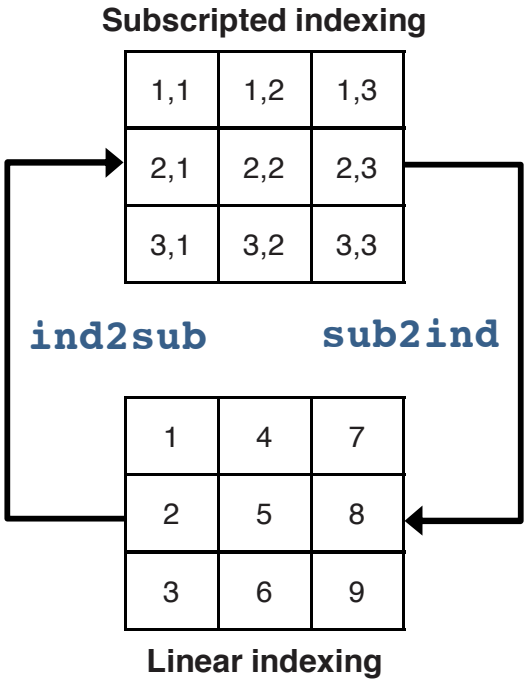


Column-Major Memory Storage

See the June 2007 article in "The MathWorks News and Notes":
http://www.mathworks.com/company/newsletters/news_notes/june07/patterns.html

Indexing into MATLAB Arrays

- Subscripted
  - Access elements by rows and columns
- Linear
  - Access elements with a single number
- Logical
  - Access elements with logical operations or mask



MATLAB Underlying Technologies

- Commercial libraries
  - BLAS: Basic Linear Algebra Subroutines (multithreaded)
  - LAPACK: Linear Algebra Package
  - etc.

BLAS and LAPACK require contiguous arrays

- JIT/Accelerator
  - Improves looping
  - Generates on-the-fly multithreaded code
  - Continually improving

Other Best Practices for Performance

- Minimize dynamically changing path

```
>> addpath(…)
>> fullfile(…)
```

- Use the functional load syntax

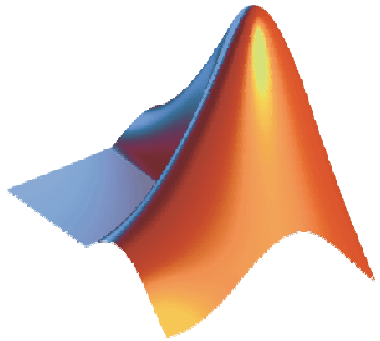
```
>> x = load('myvars.mat')
x =
        a: 5
        b: 'hello'
```

- Minimize changing variable class

```
>> x = 1;
>> x = 'hello';
```

Summary

- Techniques for addressing performance
  - Vectorization
  - Preallocation
- Consider readability and maintainability
  - Looping vs. matrix operations
  - Subscripted vs. linear vs. logical
  - etc.

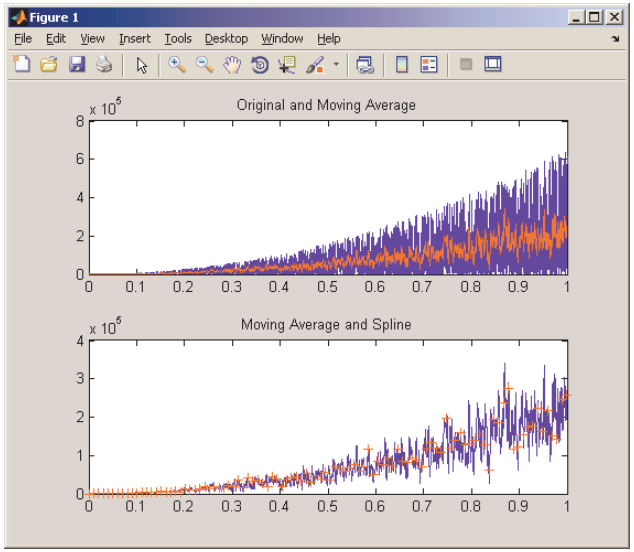


Agenda

- Leveraging the power of vector & matrix operations
-  Addressing bottlenecks
- Utilizing additional processing power
- Summary

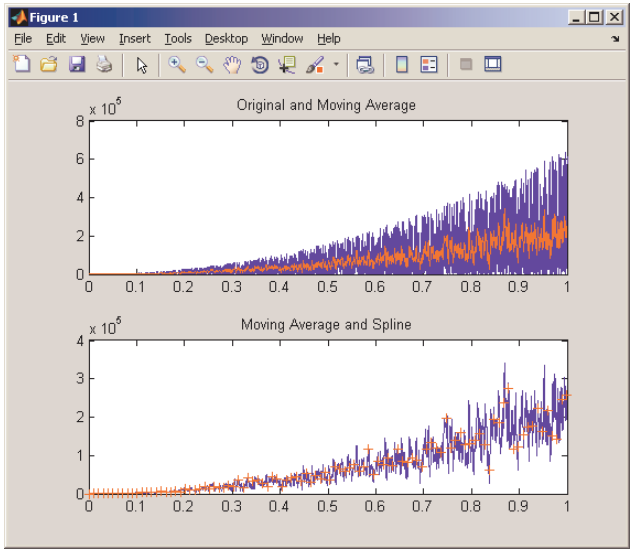
Example: Fitting Data

- Load data from multiple files
- Extract a specific test
- Fit a spline to the data
- Write results to Microsoft Excel



Summary of Example

- Used profiler to analyze code
- Targeted significant bottlenecks
- Reduced file I/O
- Reused figure



Interpreting Profiler Results

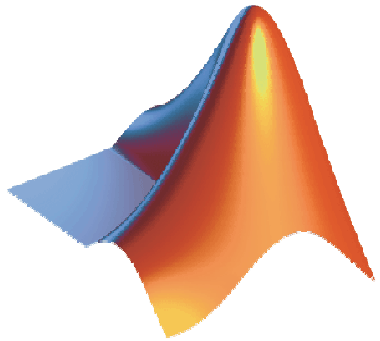
- Focus on top bottleneck
  - Total number of function calls
  - Time per function call
- Functions
  - All function calls have overhead
  - MATLAB functions often take vectors or matrices as inputs
  - Find the right function – performance may vary
    - Search MATLAB functions (e.g., `textscan` vs. `textread`)
    - Write a custom function (specific/dedicated functions may be faster)
    - Many shipping functions have viewable source code

Classes of Bottlenecks

- File I/O
  - Disk is slow compared to RAM
  - When possible, use `load` and `save` commands
- Displaying output
  - Creating new figures is expensive
  - Writing to command window is slow
- Computationally intensive
  - Use what you've learned today
  - Trade-off modularization, readability and performance
  - Integrate other languages or additional hardware
    - e.g. `emlmex`, MEX, GGPUs, FPGAs, clusters, etc.

Steps for Improving Performance

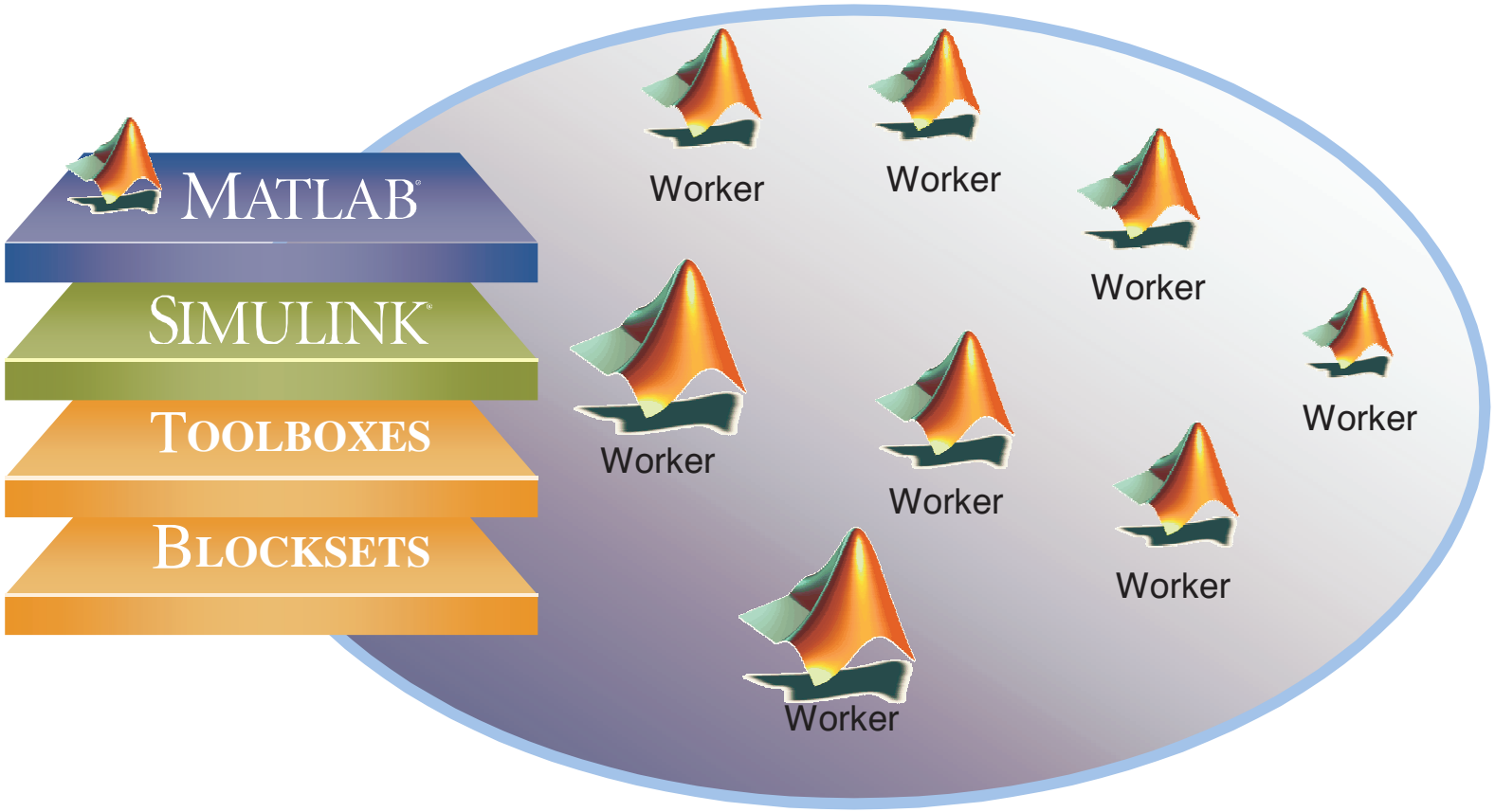
- First focus on getting your code working
- Then speed up the code within core MATLAB
- Consider additional processing power



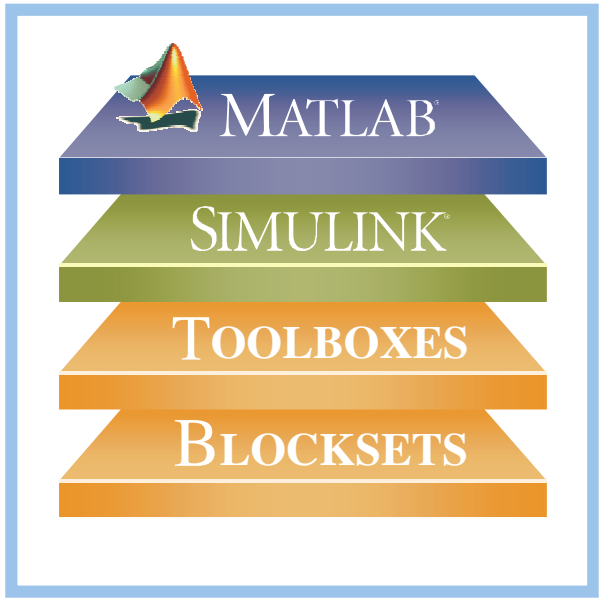
Agenda

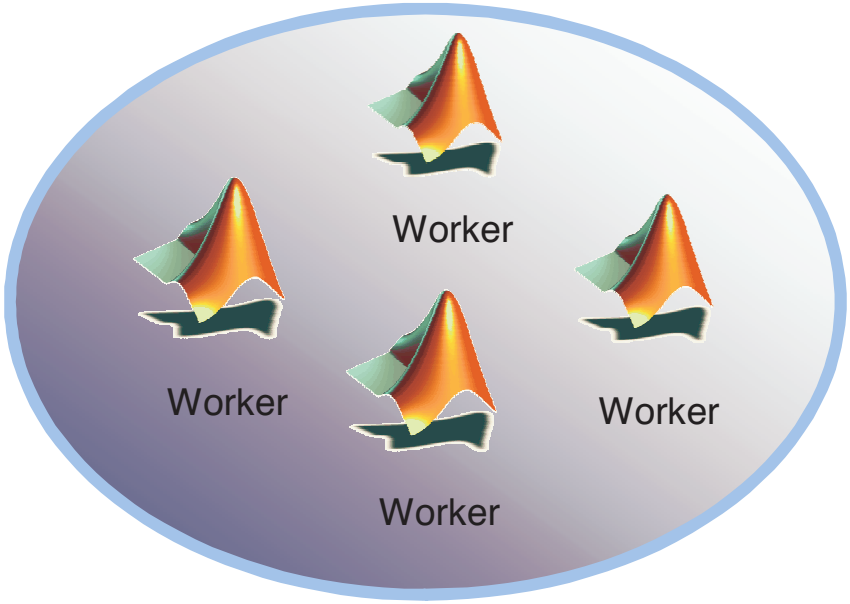
- Leveraging the power of vector & matrix operations
- Addressing bottlenecks
-  Utilizing additional processing power
- Summary

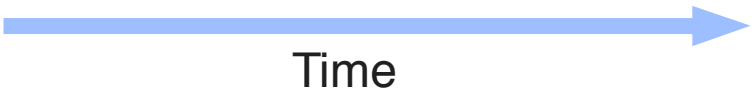
Going Beyond Serial MATLAB Applications

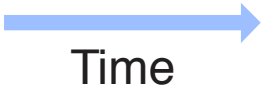


Task Parallel Applications









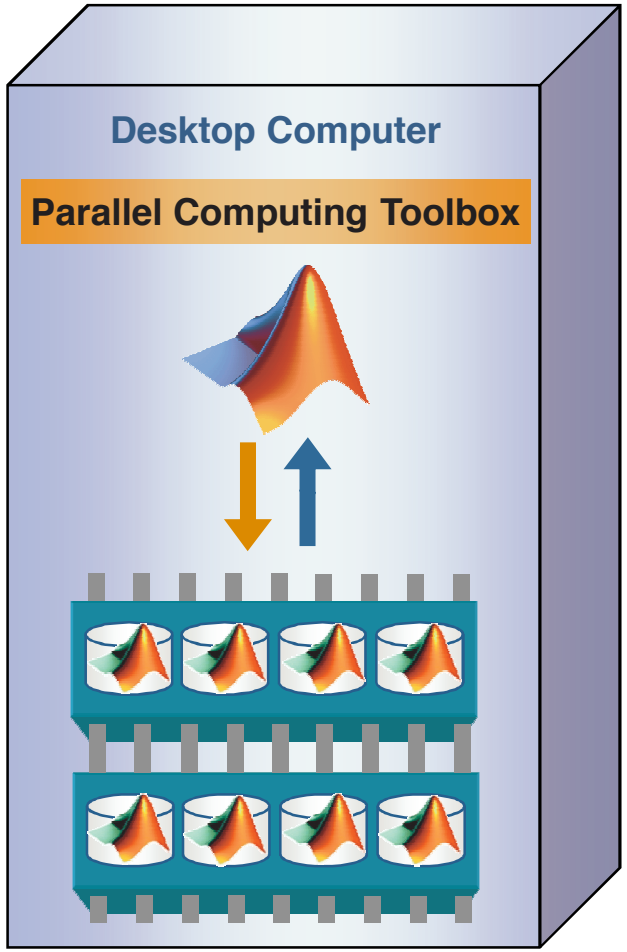
Converting `for` to `parfor`

- Requirements for `parfor` loops
  - Task independent
  - Order independent
- Constraints on the loop body
  - Cannot "introduce" variables (e.g. `eval`, `load`, `global`, etc.)
  - Cannot contain `break` or `return` statements
  - Cannot contain another `parfor` loop

Advice for Converting `for` to `parfor`

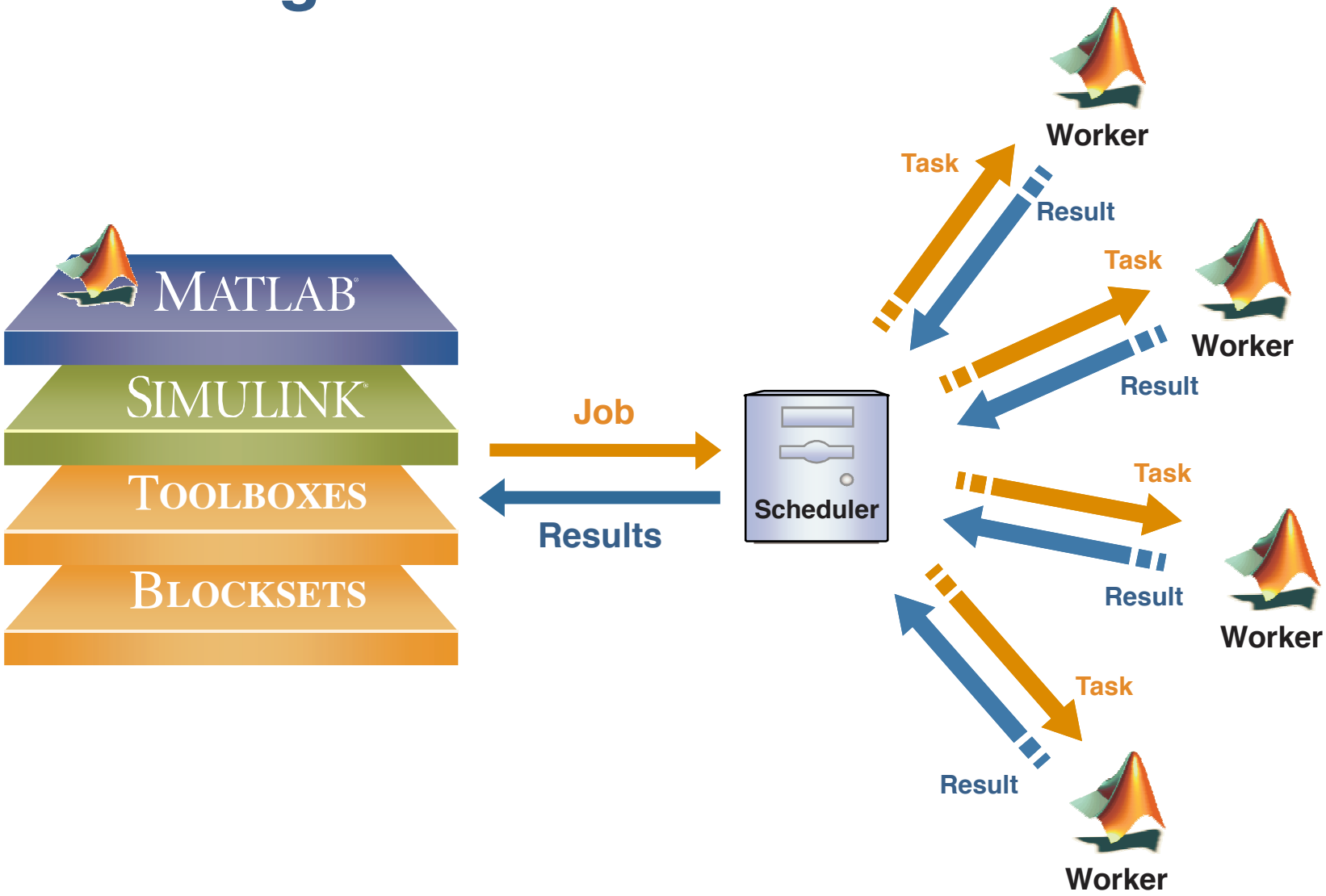
- Use M-Lint to diagnose `parfor` issues
- If your `for` loop cannot be converted to a `parfor`, consider wrapping a subset of the body to a function
- Read the section in the documentation on classification of variables

Run 8 Local Workers on Desktop

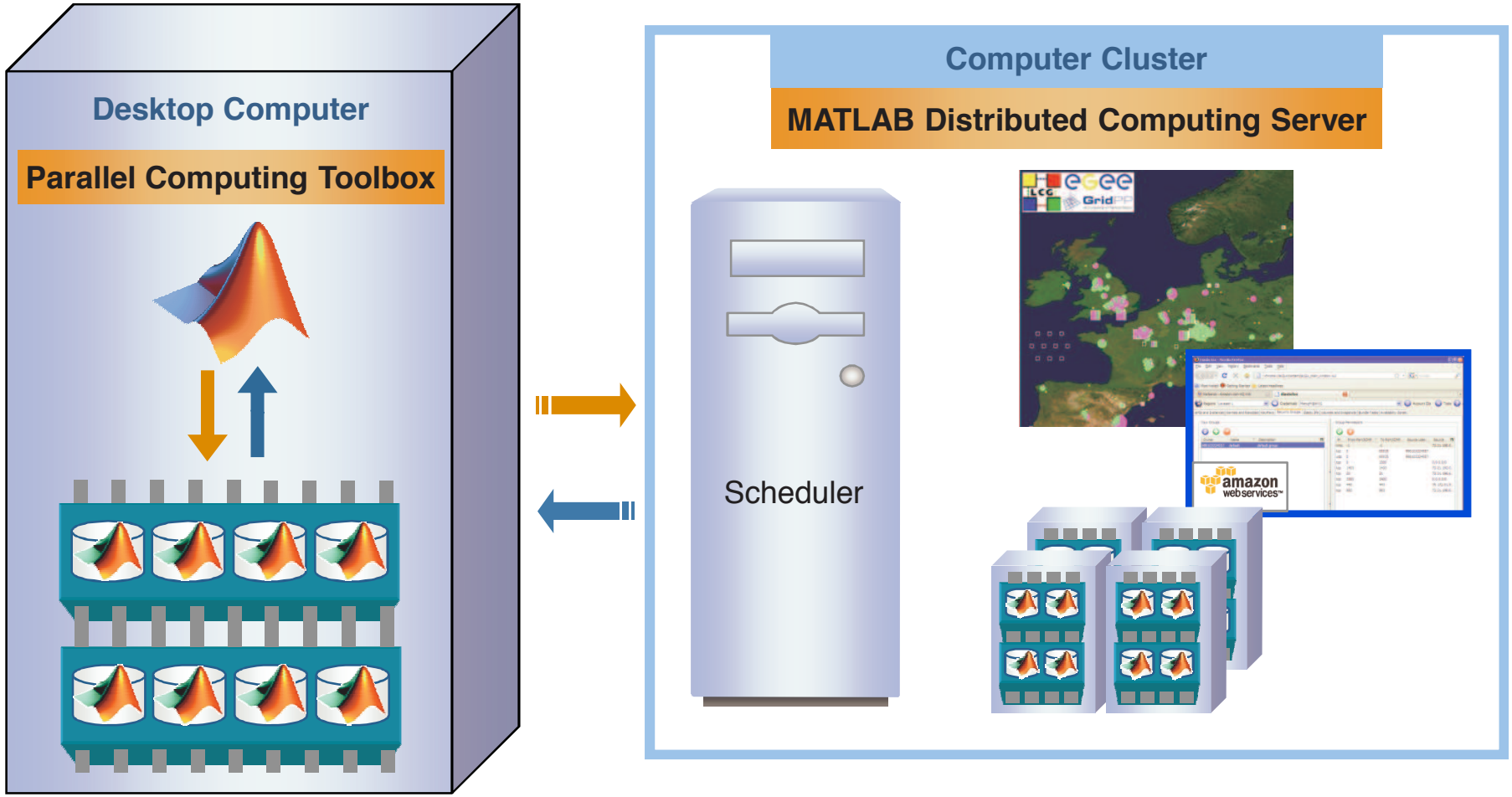


- Rapidly develop parallel applications on local computer
- Take full advantage of desktop power
- Separate computer cluster not required

Scheduling Jobs and Tasks

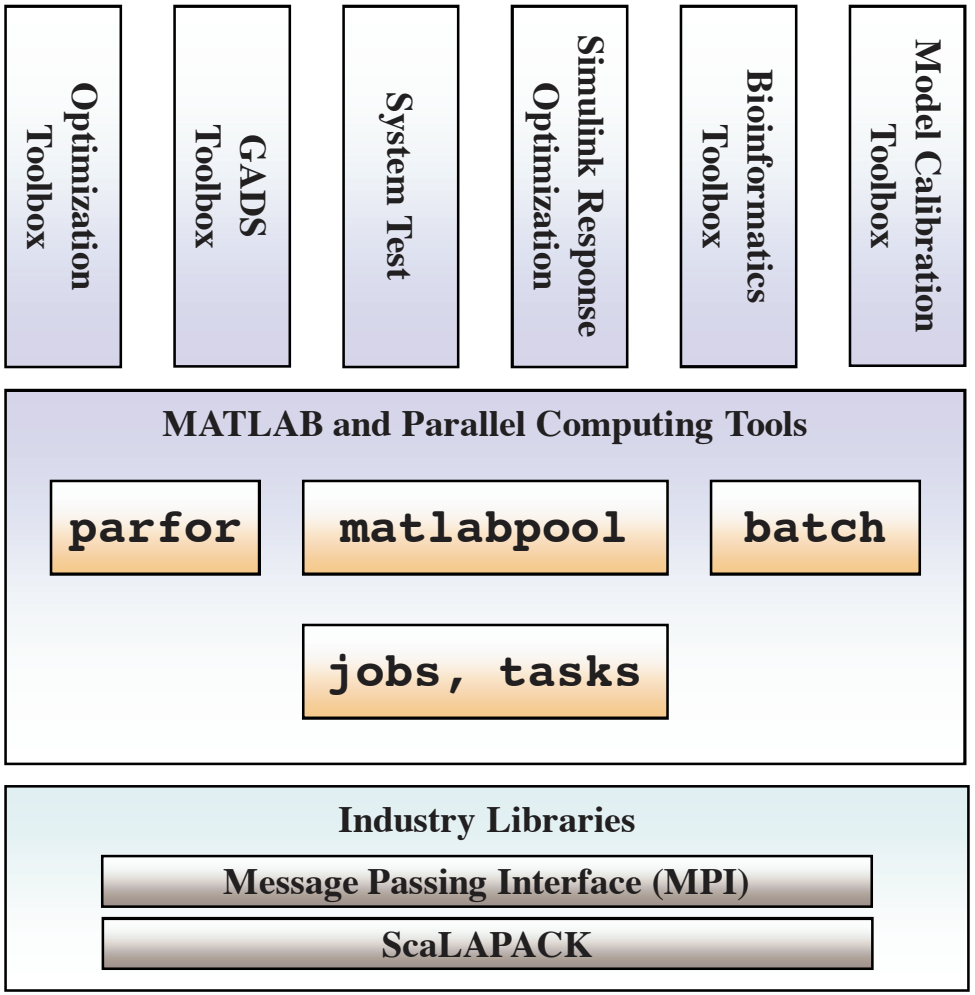


Scale Up to Clusters, Grids and Clouds



Parallel Computing with MATLAB

- Built in parallel functionality within specific toolboxes (also requires Parallel Computing Toolbox)
- High level parallel flow control for users
- Low level parallel flow control
- Built on industry standard libraries

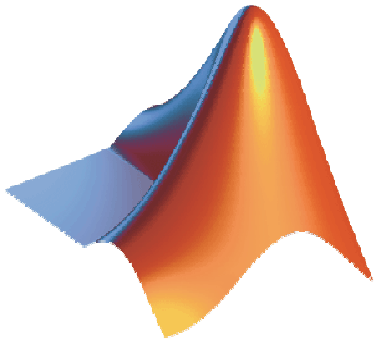


Agenda

- Leveraging the power of vector & matrix operations
- Addressing bottlenecks
- Utilizing additional processing power

Summary

- Consider performance benefit of vector and matrix operations in MATLAB
- Analyze your code for bottlenecks and address most critical items
- Leverage parallel computing tools to take advantage of additional computing resources



Sample of Other Performance Resources

- MATLAB documentation

  MATLAB → Programming Fundamentals → Performance

- Memory Management Guide

  www.mathworks.com/support/tech-notes/1100/1106.html?BB=1

- The Art of MATLAB, Loren Shure's blog

  blogs.mathworks.com/loren/

- MATLAB Central's Usenet portal

  www.mathworks.com/matlabcentral/newsreader.html

The MathWorks™

MATLAB® & SIMULINK®