MathWorks

Optimizing and Accelerating MATLAB Code

Tom McHugh Account Manager
Saket Kharsikar Application Engineer

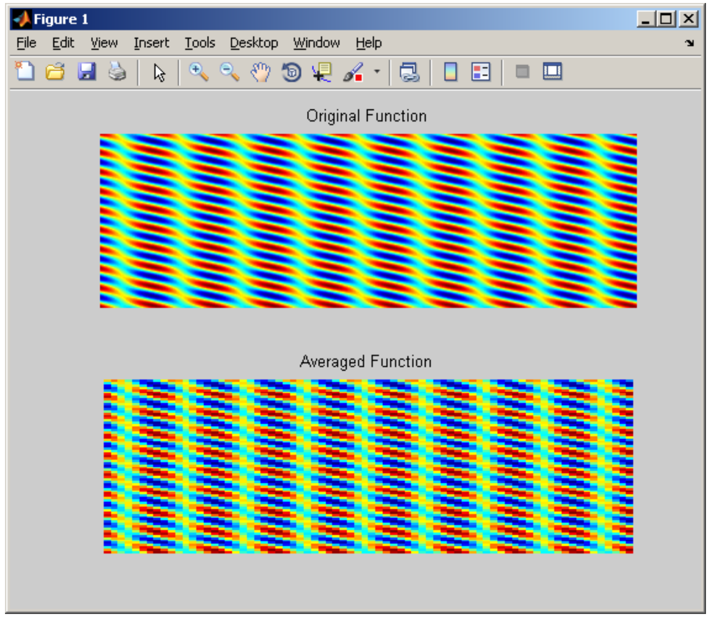
TEXAS A&M UNIVERSITY

Agenda

- Leveraging the power of vector and matrix operations
- Addressing bottlenecks
- Generating and incorporating C code
- 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

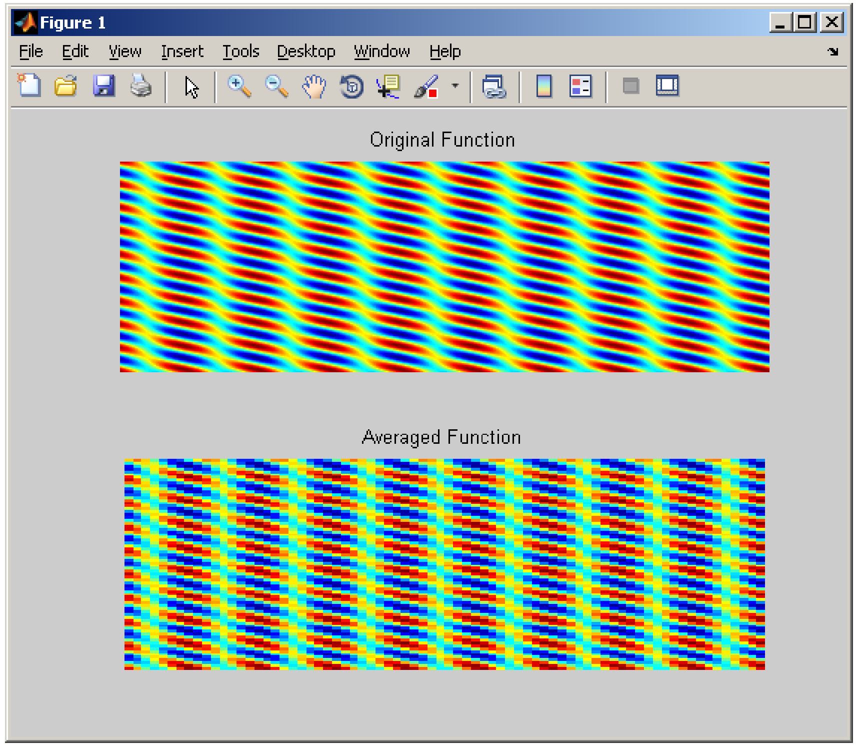
- Used built-in timing functions

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

- Used MATLAB Code Analyzer to find suboptimal code

- Preallocated arrays

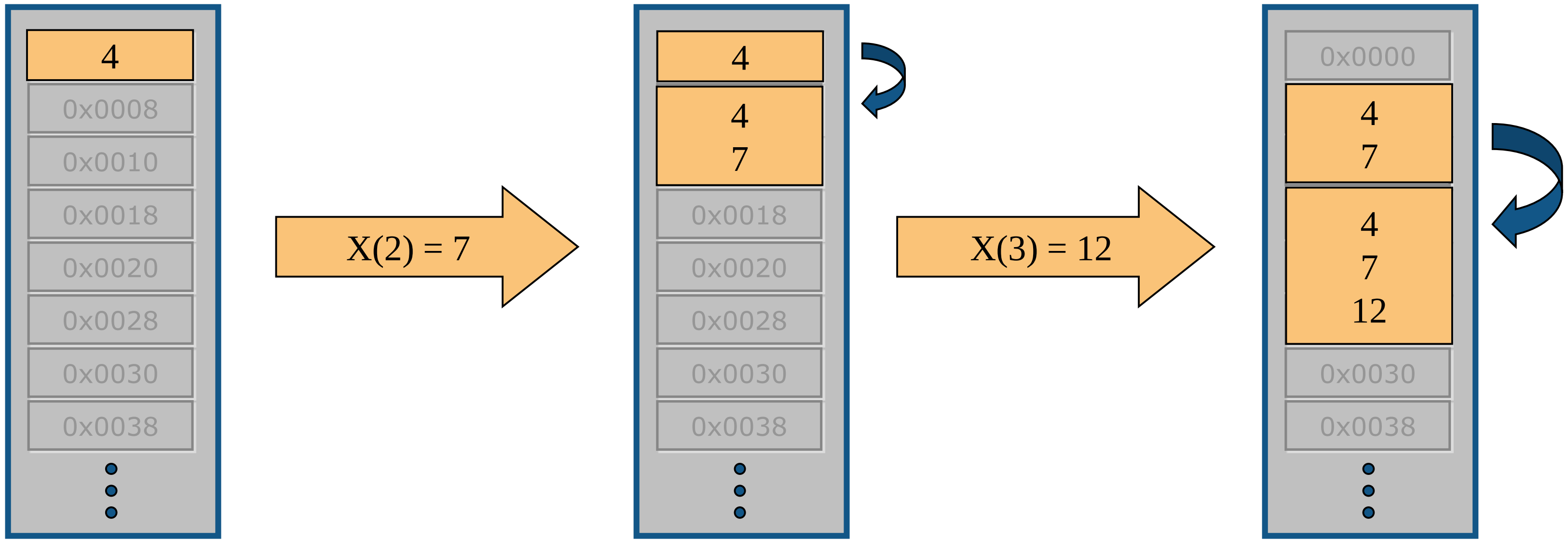
- Vectorized code



Effect of Not Preallocating Memory

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

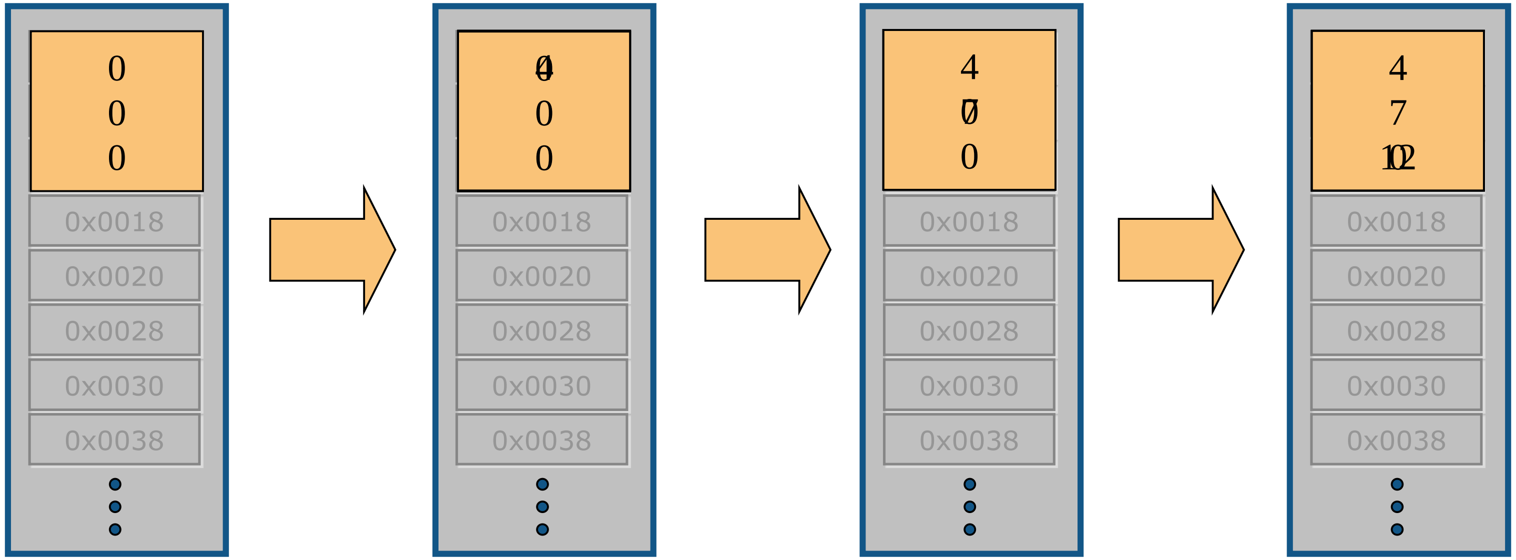
Resizing Arrays is Expensive



Benefit of Preallocation

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

Reduced Memory Operations



MATLAB Underlying Technologies

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

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

BLAS and LAPACK require contiguous arrays

Other Best Practices

- Minimize dynamically changing path

```
>> addpath(…)
>> fullfile(…)
```

instead of `cd(…)`

- Use the functional load syntax

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

instead of `load('myvars.mat')`

- Minimize changing variable class

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

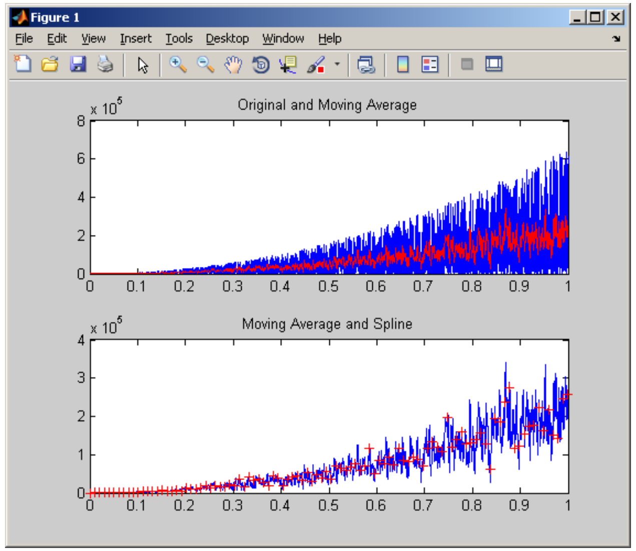
instead of `x = 'hello';`

Agenda

- Leveraging the power of vector and matrix operations
- Addressing bottlenecks
- Generating and incorporating C code
- Utilizing additional processing power
- Summary

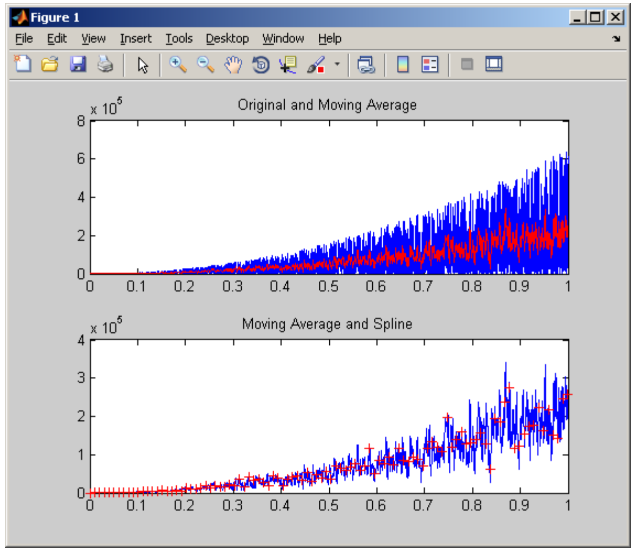
Example: Fitting Data

- 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. MEX, GPUs, FPGAs, clusters, etc.

Steps for Improving Performance

- First focus on getting your code working
- Then speed up the code within core MATLAB
- Consider other languages (i.e. C MEX files) and additional processing power

Agenda

- Leveraging the power of vector and matrix operations
- Addressing bottlenecks
- Generating and incorporating C code
- Utilizing additional processing power
- Summary

Why engineers and scientists translate MATLAB to C today?

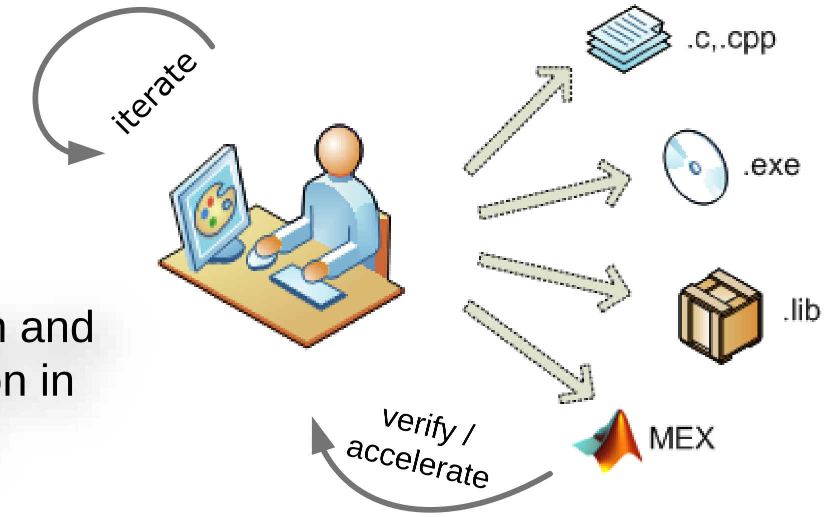
.lib — **Integrate** MATLAB algorithms w/ existing C environment using source code and static/dynamic libraries

.exe — **Prototype** MATLAB algorithms on desktops as standalone executables

MEX — **Accelerate** user-written MATLAB algorithms

.c,.cpp — **Implement** C code on processors or hand-off to software engineers

Automatic Translation of MATLAB to C



Algorithm Design and Code Generation in MATLAB

With MATLAB Coder, design engineers can

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

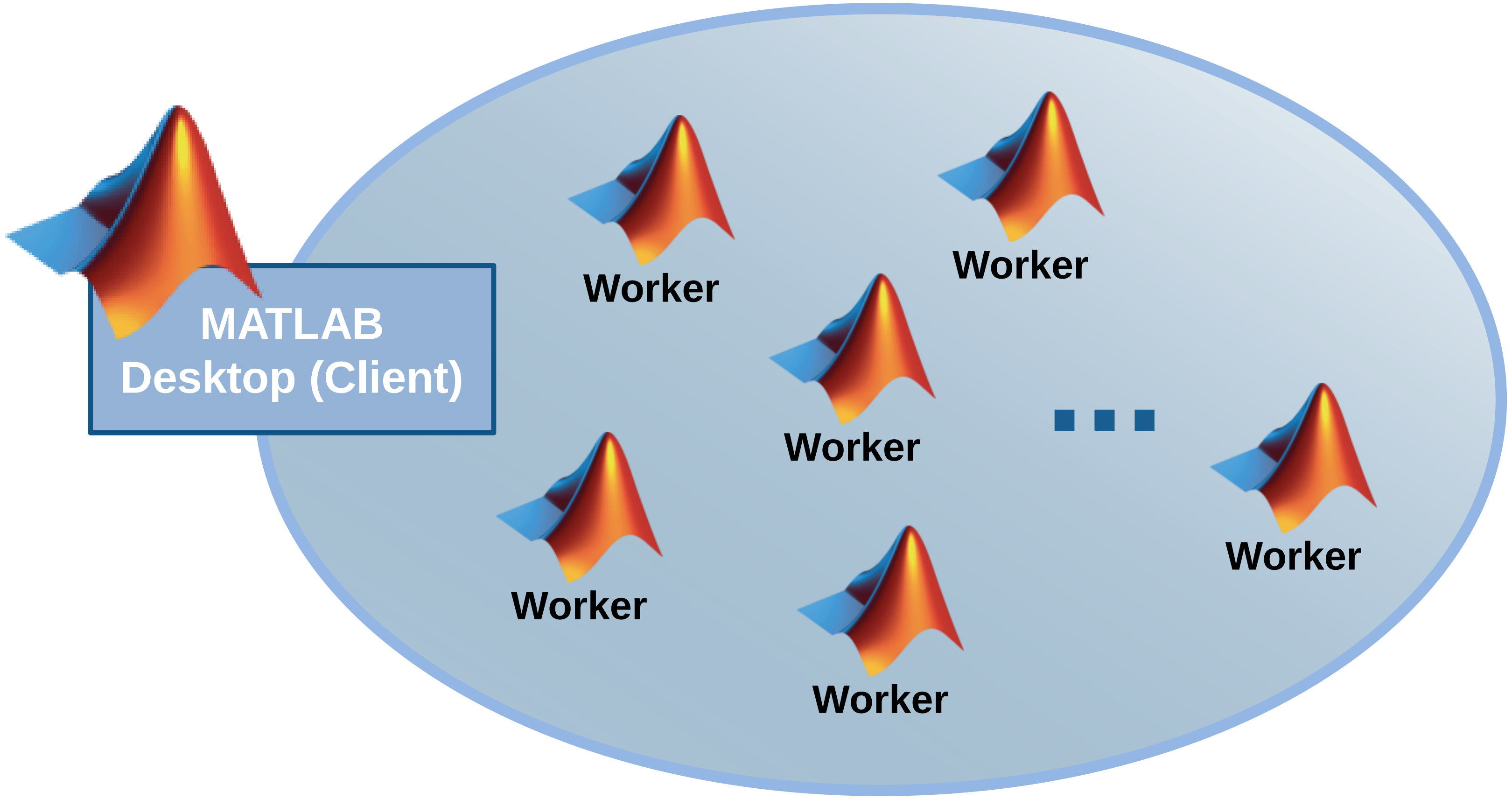
Acceleration using MEX

- Speed-up factor will vary
- When you **may** see a speedup
  - Often for Communications and Signal Processing
  - Always for Fixed-point
  - Likely for loops with states or when vectorization isn’t possible
- When you **may not** see a speedup
  - MATLAB implicitly multithreads computation
  - Built-functions call IPP or BLAS libraries

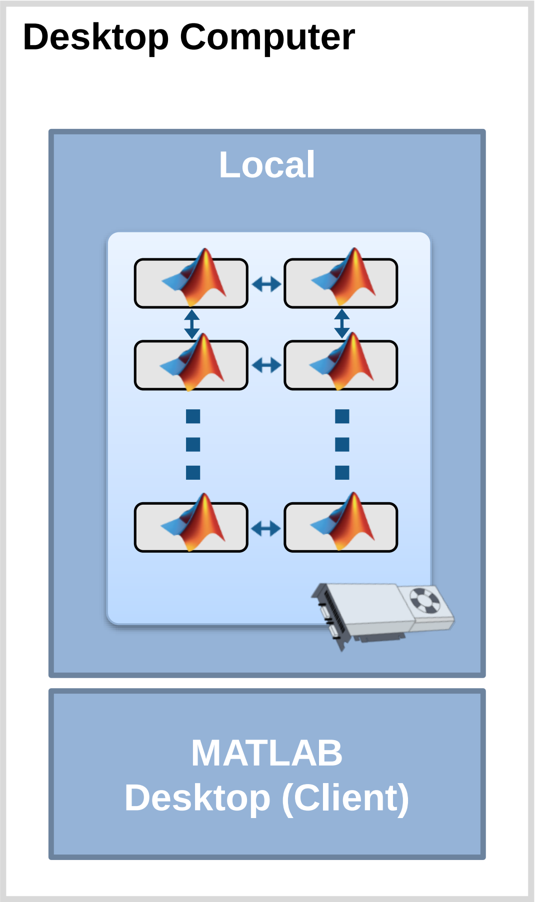
Agenda

- Leveraging the power of vector and matrix operations
- Addressing bottlenecks
- Generating and incorporating C code
- Utilizing additional processing power
- Summary

Going Beyond Serial MATLAB Applications

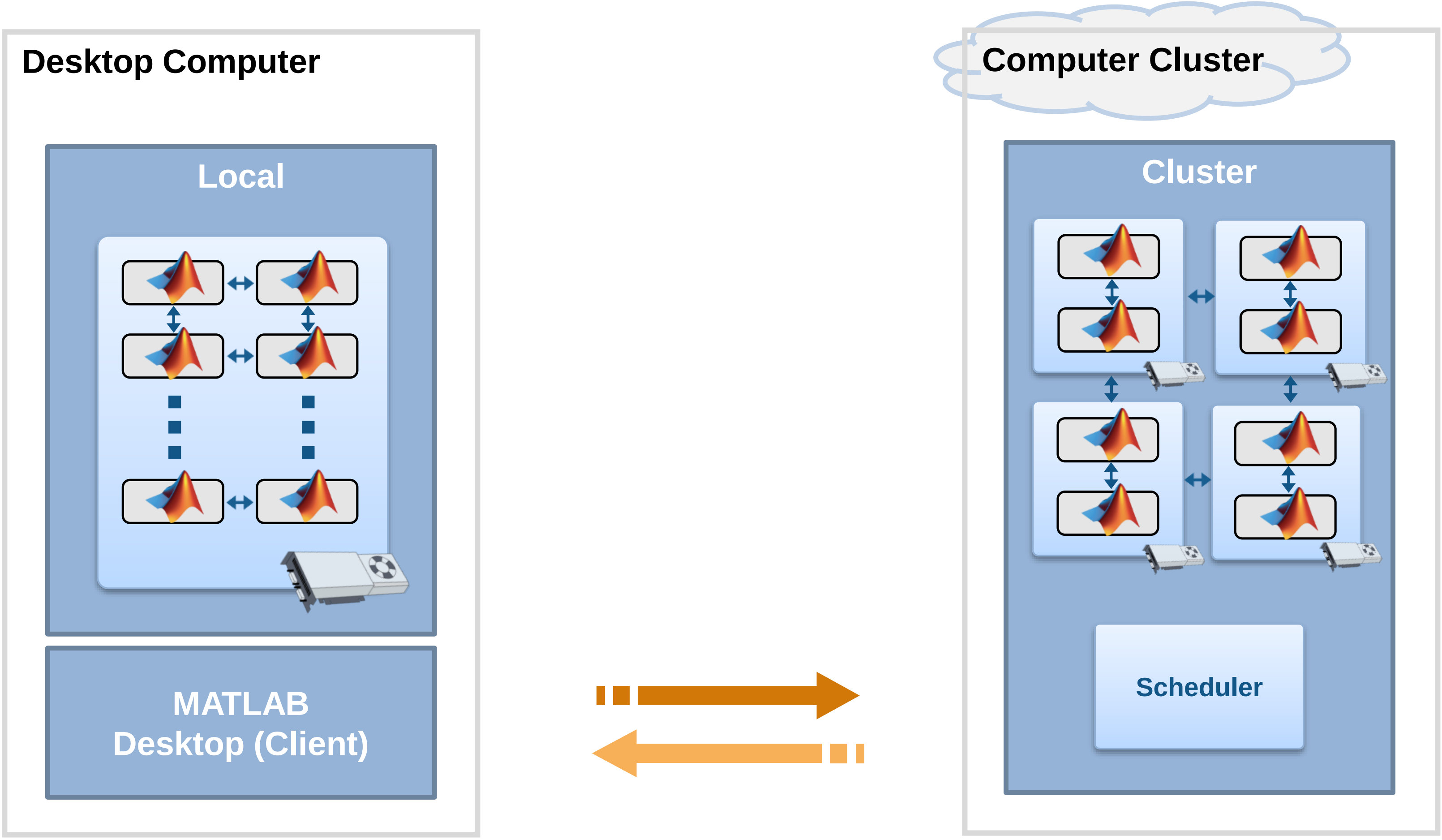


Parallel Computing Toolbox for the Desktop



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

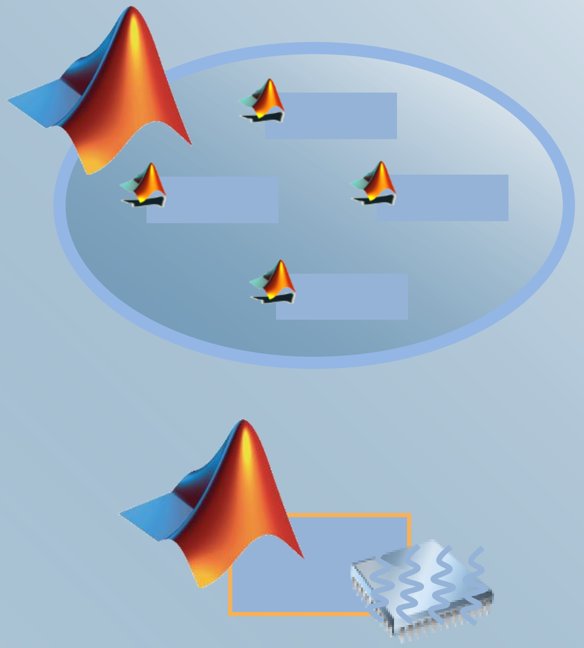
Scale Up to Clusters and Clouds



Parallel Computing enables you to …

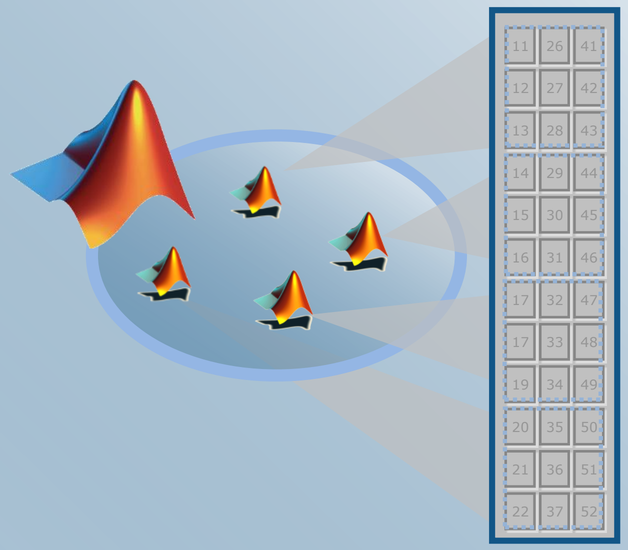
Larger Compute Pool

Speed up Computations

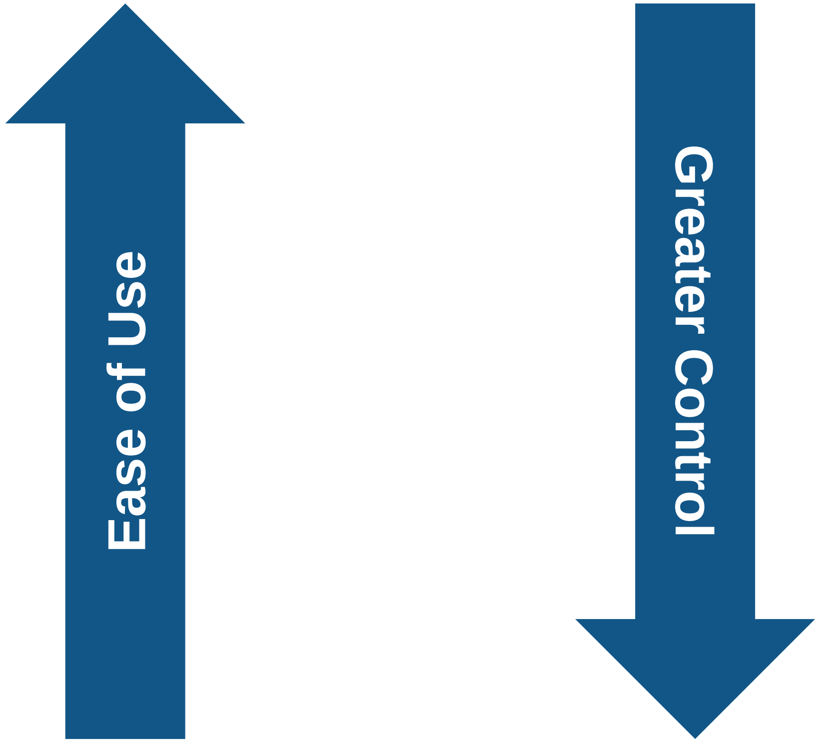


Larger Memory Pool

Work with Large Data



Programming Parallel Applications



Programming Parallel Applications (CPU)

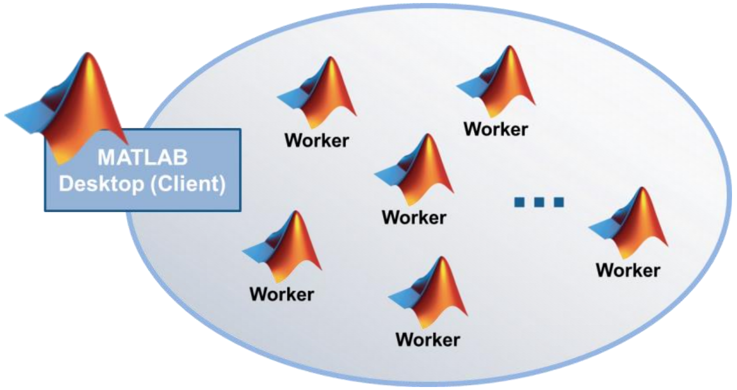
Ease of Use

- Built-in support with toolboxes

Greater Control

Tools Providing Parallel Computing Support

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



http://www.mathworks.com/products/parallel-computing/builtin-parallel-support.html

Directly leverage functions in Parallel Computing Toolbox

www.mathworks.com/builtin-parallel-support

Programming Parallel Applications (CPU)

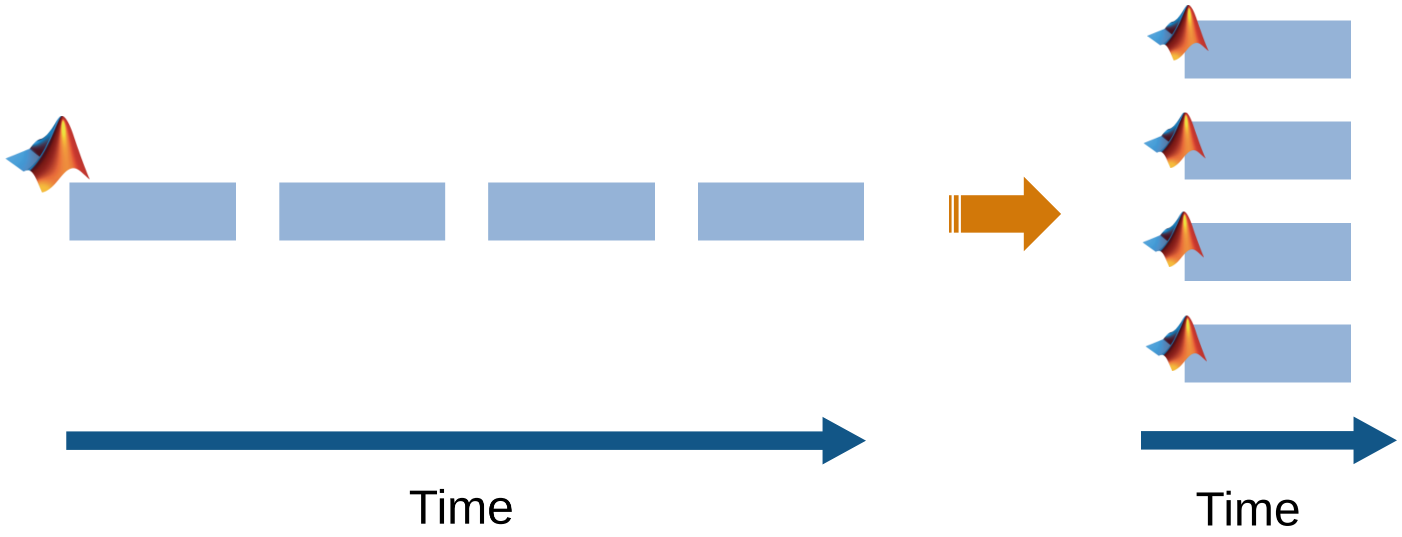
Ease of Use

- 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



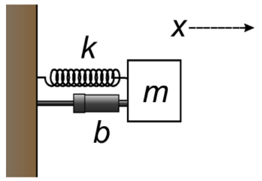
blogs.mathworks.com/loren/2009/10/02/using-parfor-loops-getting-up-and-running/

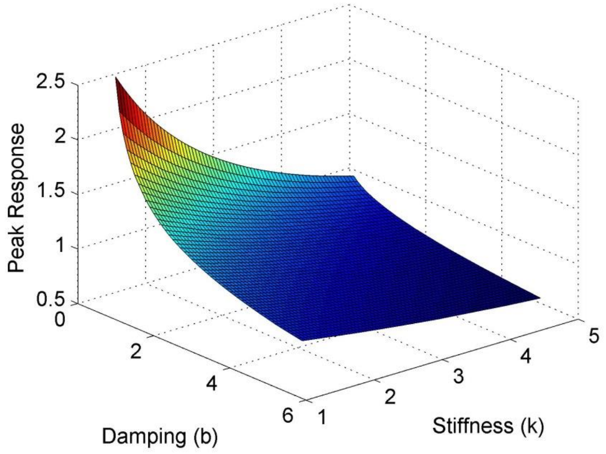
Example: Parameter Sweep of ODEs

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

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





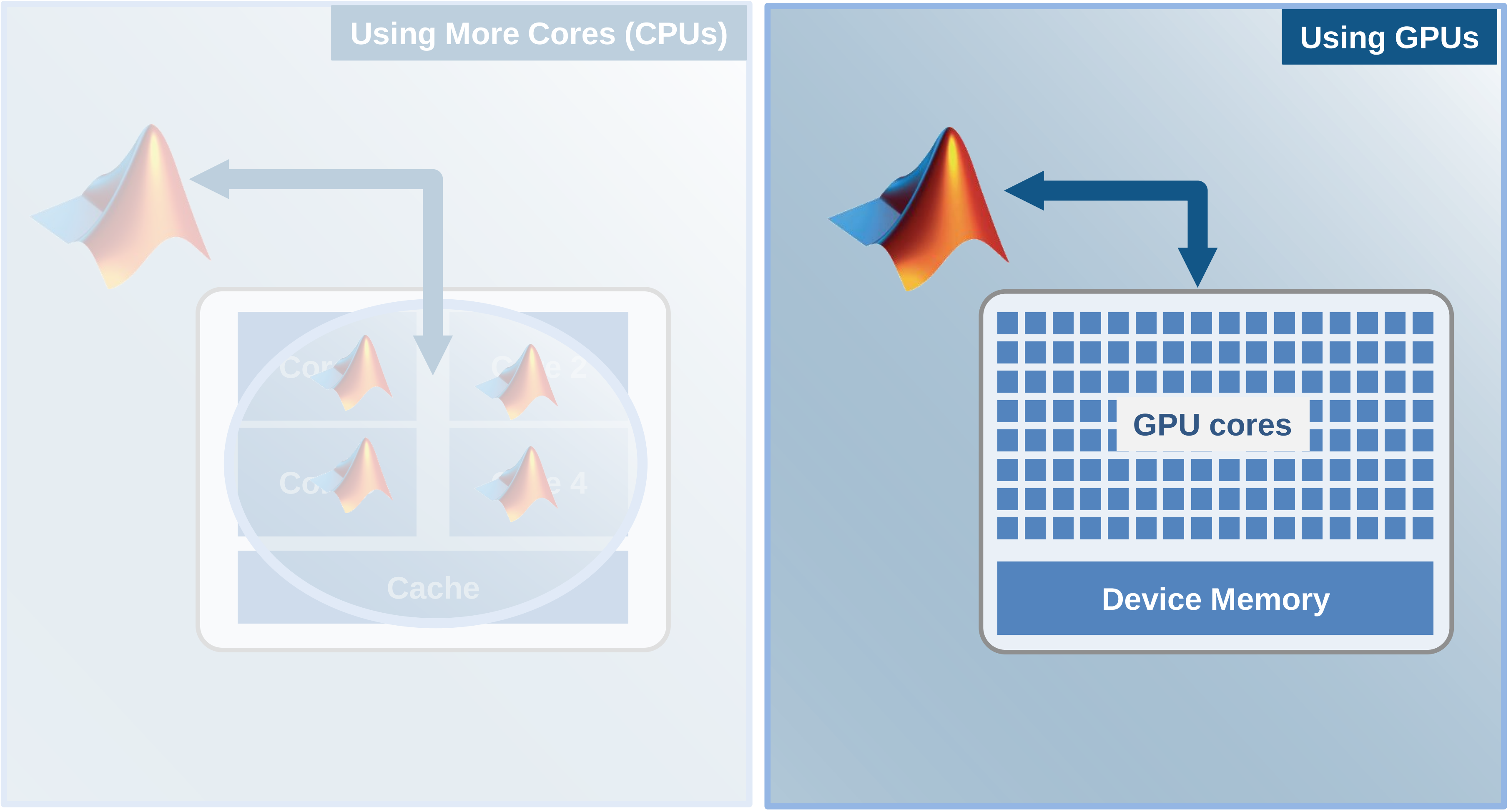
Programming Parallel Applications (CPU)

Ease of Use

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

Greater Control

Performance Gain with More Hardware



Programming Parallel Applications (GPU)

Ease of Use

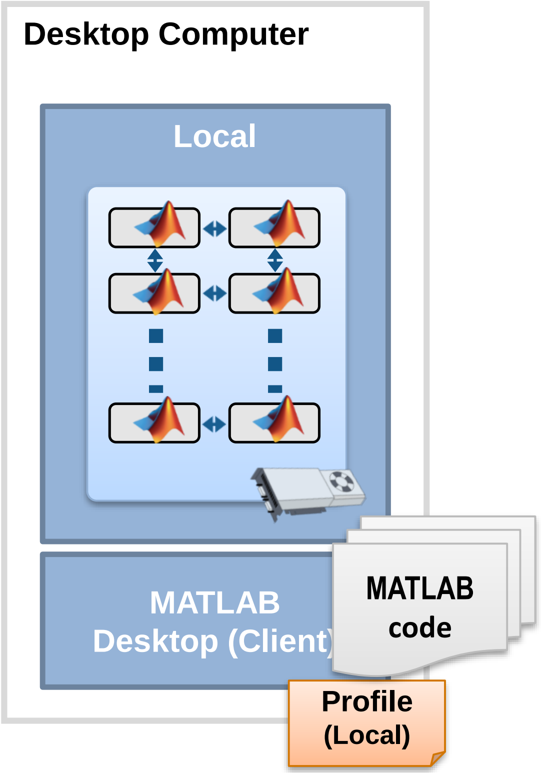
- Built-in support with toolboxes
- Simple programming constructs:
  `gpuArray, gather`
- Advanced programming constructs:
  `arrayfun, spmd`
- Interface for experts:
  `CUDAKernel, MEX support`

  www.mathworks.com/help/distcomp/run-cuda-or-ptx-code-on-gpu

  www.mathworks.com/help/distcomp/run-mex-functions-containing-cuda-code

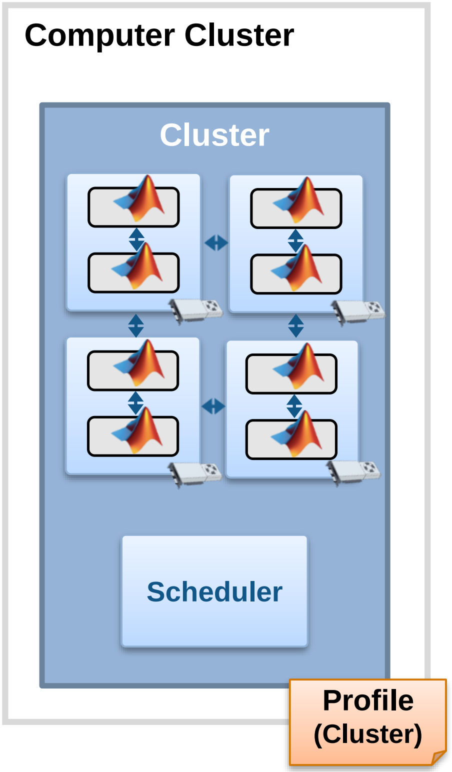
Greater Control

Use MATLAB Distributed Computing Server



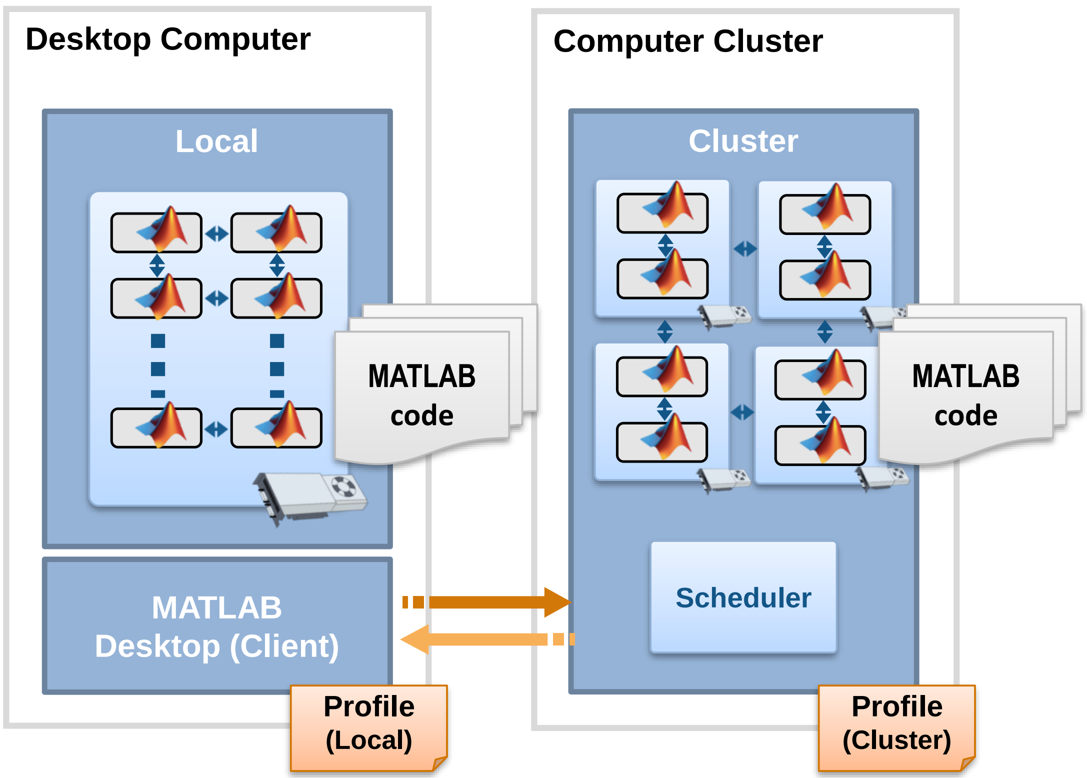
1. Prototype code

Use MATLAB Distributed Computing Server



1. Prototype code
2. Get access to an enabled cluster

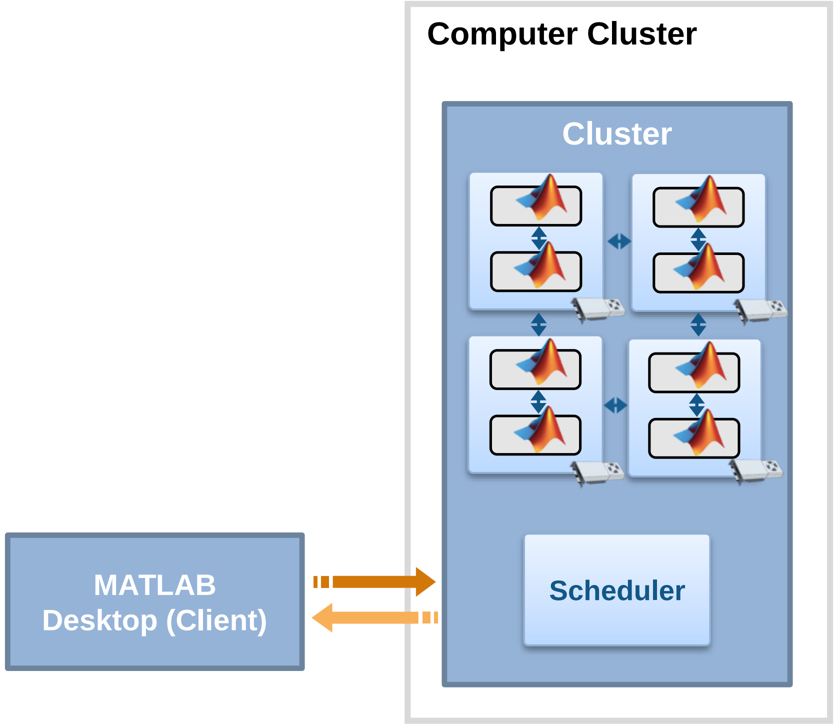
Use MATLAB Distributed Computing Server



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

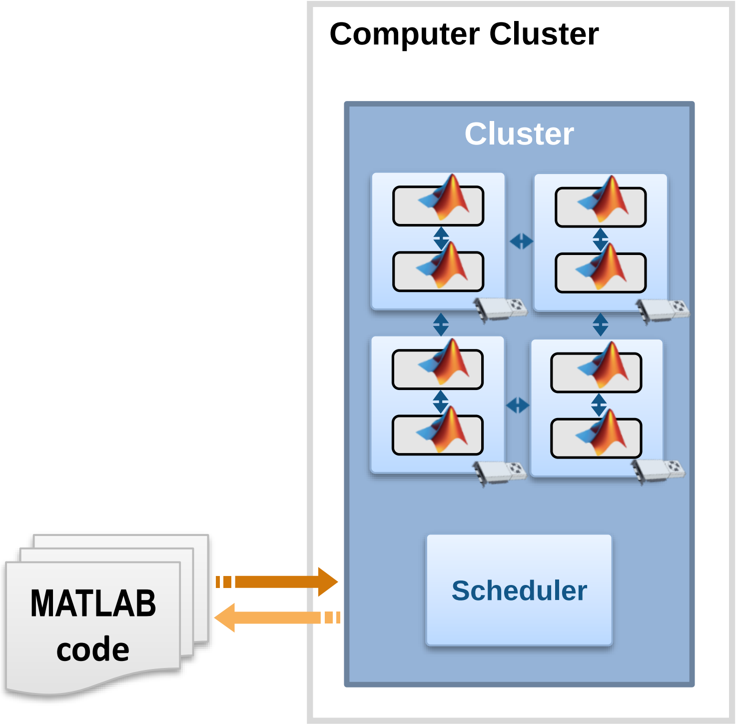
Take Advantage of Cluster Hardware

- 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

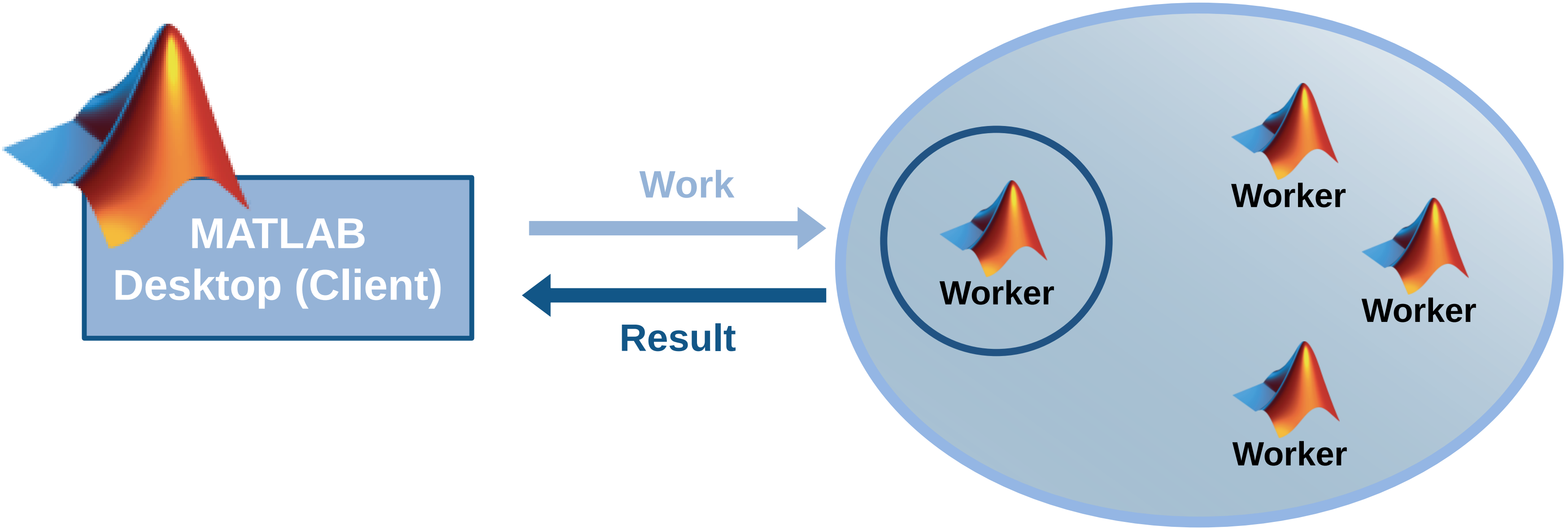


Offloading Computations

- 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

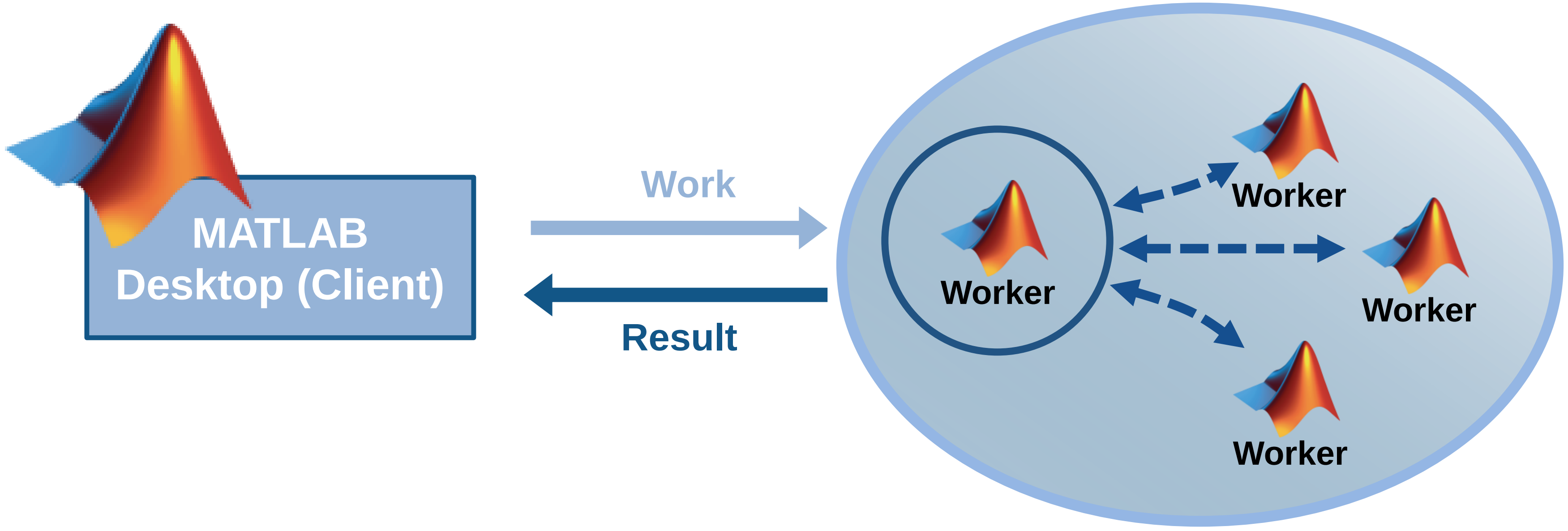


Offload Computations with `batch`



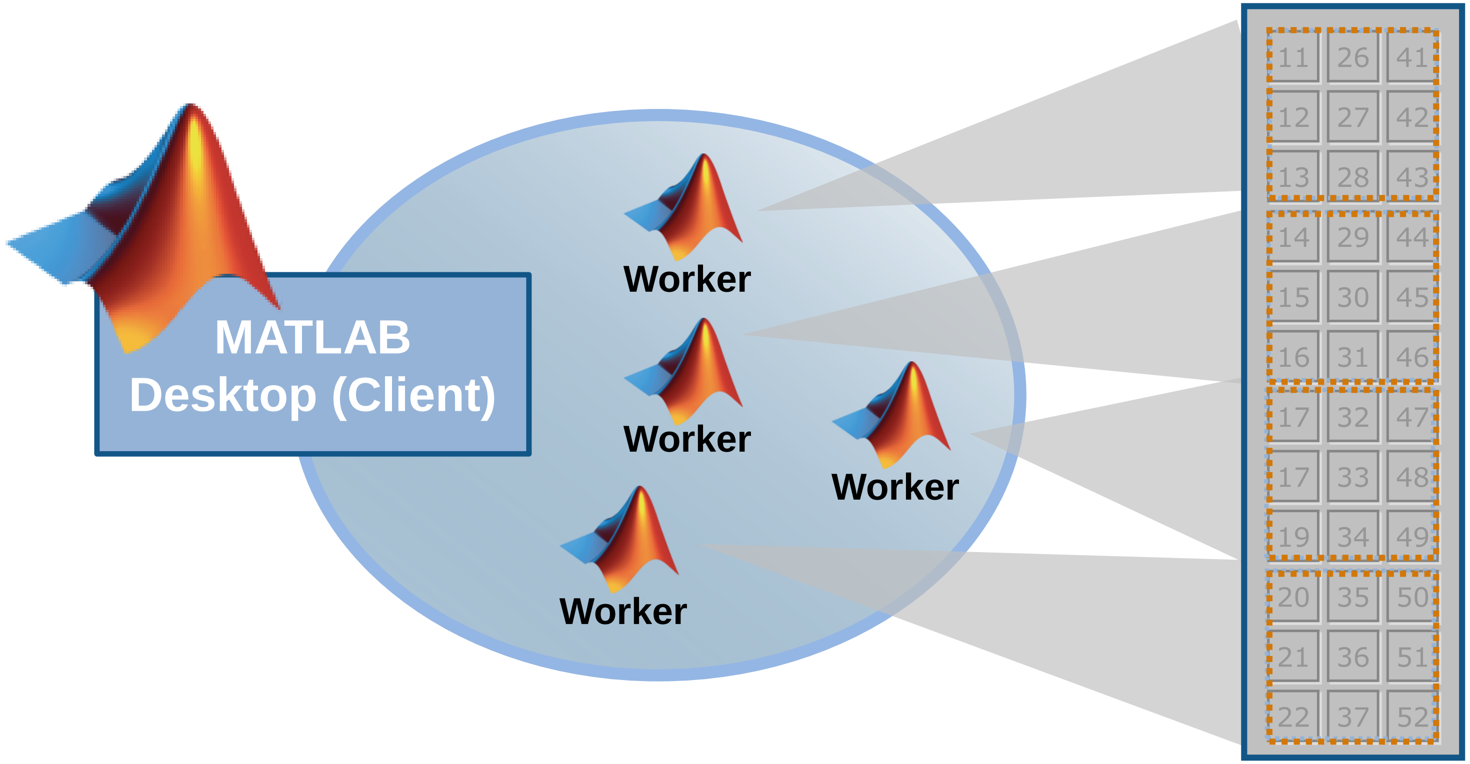
`batch(…)`

Offload and Scale Computations with `batch`



`batch(…,'Pool',…)`

Distributing Large Data



Remotely Manipulate Array
from Client

Distributed Array
Lives on the Workers

Distributed Arrays and SPMD

- Distributed arrays
  - Hold data remotely on workers running on a cluster
  - Manipulate directly from client MATLAB (desktop)
  - Use MATLAB functions directly on distributed arrays
    - www.mathworks.com/help/distcomp/using-matlab-functions-on-codistributed-arrays
- `spmd`
  - Execute blocks of code on workers
  - Explicitly communicate between workers with message passing
  - Mix parallel and serial code in same program

TAMU HPRC MATLAB Resources

Why use MATLAB on HPRC clusters?

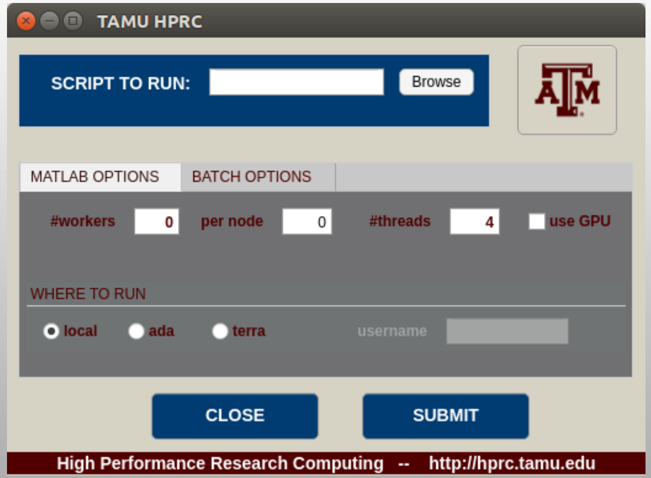
- Long running Matlab scripts
- Large memory requirements
  - At least 64GB per node, up to 2TB
  - Distribute data over multiple nodes
- Utilizing Matlab parallel toolbox
  - Start up to 28 Matlab workers per node
  - Start Matlab workers on multiple nodes
- Utilizing Matlab GPU capabilities
  - 48 nodes with dual K80 gpus on terra
  - 30 nodes with dual K40 gpus on ada

Who can use HPRC resources?

- All A&M students/staff/faculty
- Apply for account at: hprc.tamu.edu/accounts/apply/

What HPRC offers

- Latest versions of Matlab
- Matlab Distributed Computing Server (MDCS) license
  - Currently 96 tokens
  - Distribute workers over nodes
- Assistance parallelizing code
- Consulting
- Framework to run parallel code
- HPRC Matlab App
  - Submit Matlab jobs from your own desktop/laptop



Agenda

- Leveraging the power of vector and matrix operations
- Addressing bottlenecks
- Generating and incorporating C code
- Utilizing additional processing power
- Summary

Key Takeaways

- Consider performance benefit of vector and matrix operations in MATLAB
- Analyze your code for bottlenecks and address most critical items
- Leverage MATLAB Coder to speed up applications through generated C/C++ code
- Leverage parallel computing toolsto take advantage of additional computing resources

MathWorks®

Accelerating the pace of engineering and science