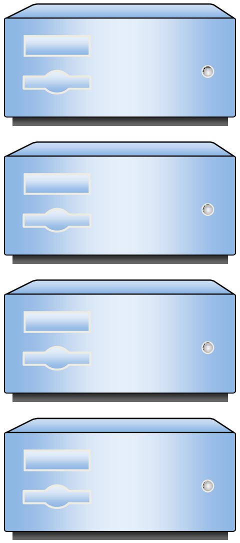
Parallel Computing with MATLAB



Tim Mathieu
Sr. Account Manager

Gerardo Hernandez
Application Engineer

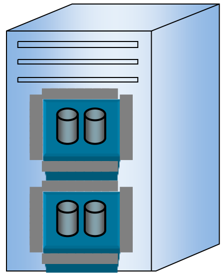
Abhishek Gupta
Application Engineer

Some Questions to Consider

- Do you want to speed up your algorithms?
- Do you have datasets too big to fit on your computer?

If so…

- Do you have a multi-core or multiprocessor desktop machine?
- Do you have access to a computer cluster?



Solving Big Technical Problems

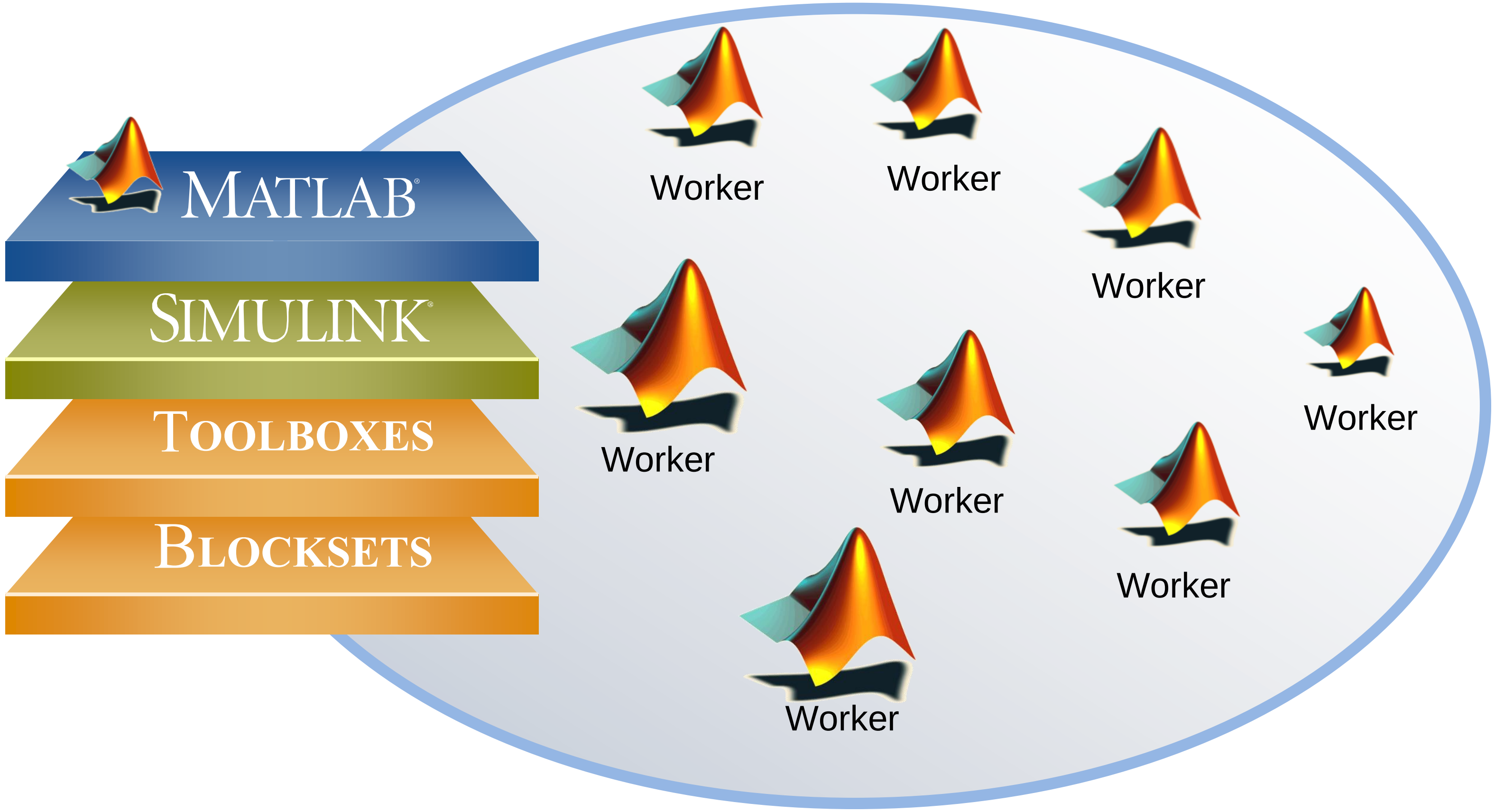
Challenges	You could…		Solutions
Long running Computationally intensive	Wait	⇨	Larger Compute Pool (e.g. More Processors)
Large data set	Reduce size of problem	⇨	Larger Memory Pool (e.g. More Machines)

Utilizing Additional Processing Power

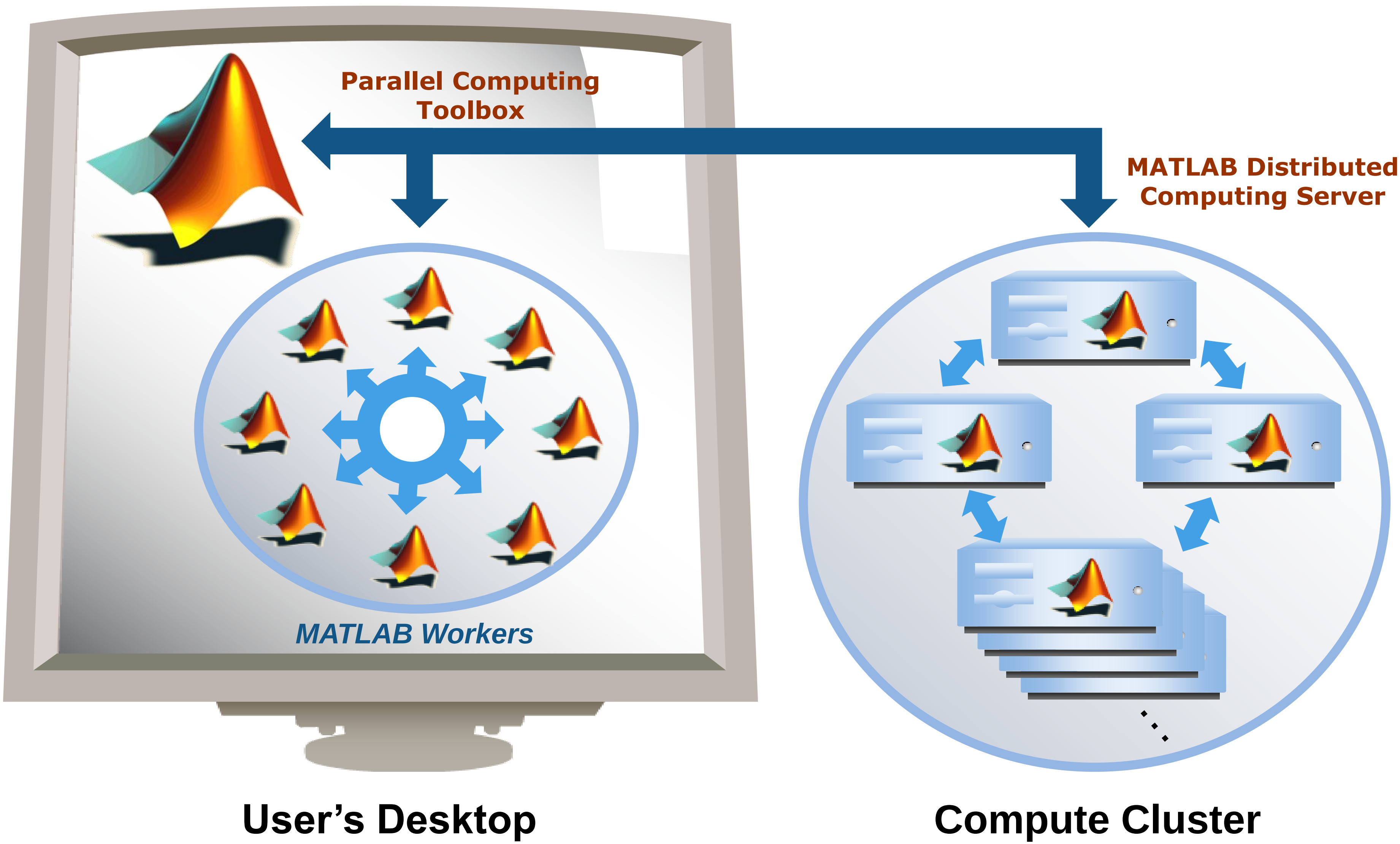
- Built-in multithreading
  - Core MATLAB
  - Introduced in R2007a
  - Utility for specific matrix operations
  - Automatically enabled since R2008a

- Parallel computing tools
  - Parallel Computing Toolbox
  - MATLAB Distributed Computing Server
  - Broad utility controlled by the MATLAB user

Parallel Computing with MATLAB



Parallel Computing with MATLAB



Programming Parallel Applications

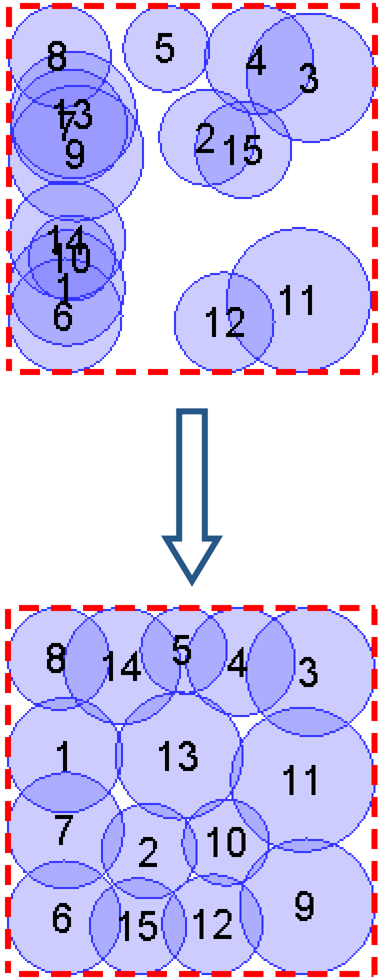
Level of control	*Required effort*
Minimal	***None***
Some	***Straightforward***
Extensive	***Involved***

Programming Parallel Applications

Level of control	*Parallel Options*
Minimal	**Support built into Toolboxes**
Some	**High-Level Programming Constructs: (e.g. parfor, batch, distributed)**
Extensive	**Low-Level Programming Constructs: (e.g. Jobs/Tasks, MPI-based)**

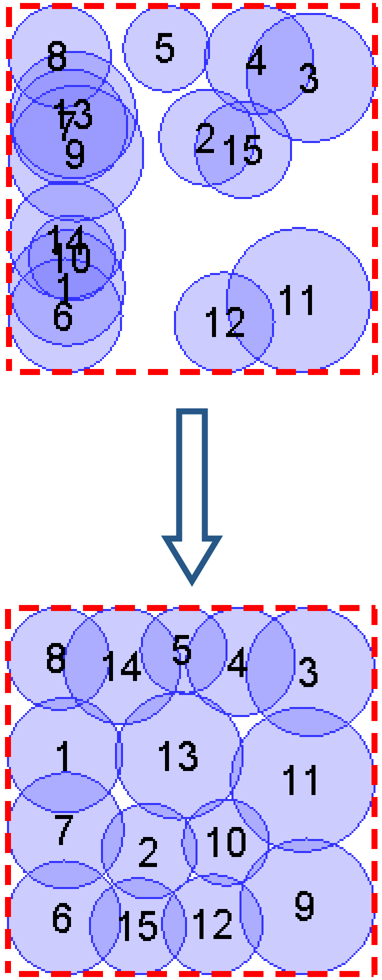
Example: Optimizing Tower Placement

- Determine location of cell towers
- Maximize coverage
- Minimize overlap



Summary of Example

- Enabled built-in support for Parallel Computing Toolbox in Optimization Toolbox
- Used a pool of MATLAB workers
- Optimized in parallel using `fmincon`



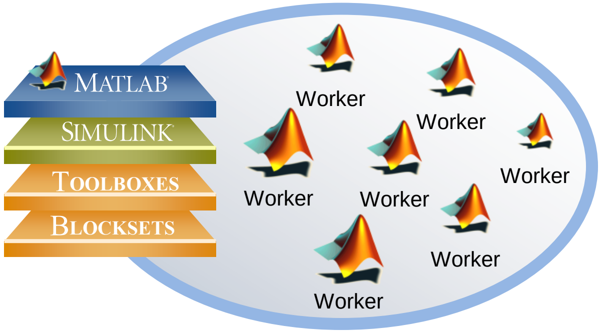
Parallel Support in Optimization Toolbox

- Functions:
  - **`fmincon`**
    - Finds a constrained minimum of a function of several variables
  - **`fminimax`**
    - Finds a minimax solution of a function of several variables
  - **`fgoalattain`**
    - Solves the multiobjective goal attainment optimization problem

- Functions can take finite differences in parallel in order to speed the estimation of gradients

Tools with Built-in Support

- Optimization Toolbox
- Global Optimization Toolbox
- Statistics Toolbox
- SystemTest
- Simulink Design Optimization
- Bioinformatics Toolbox
- Model-Based Calibration Toolbox
- …



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

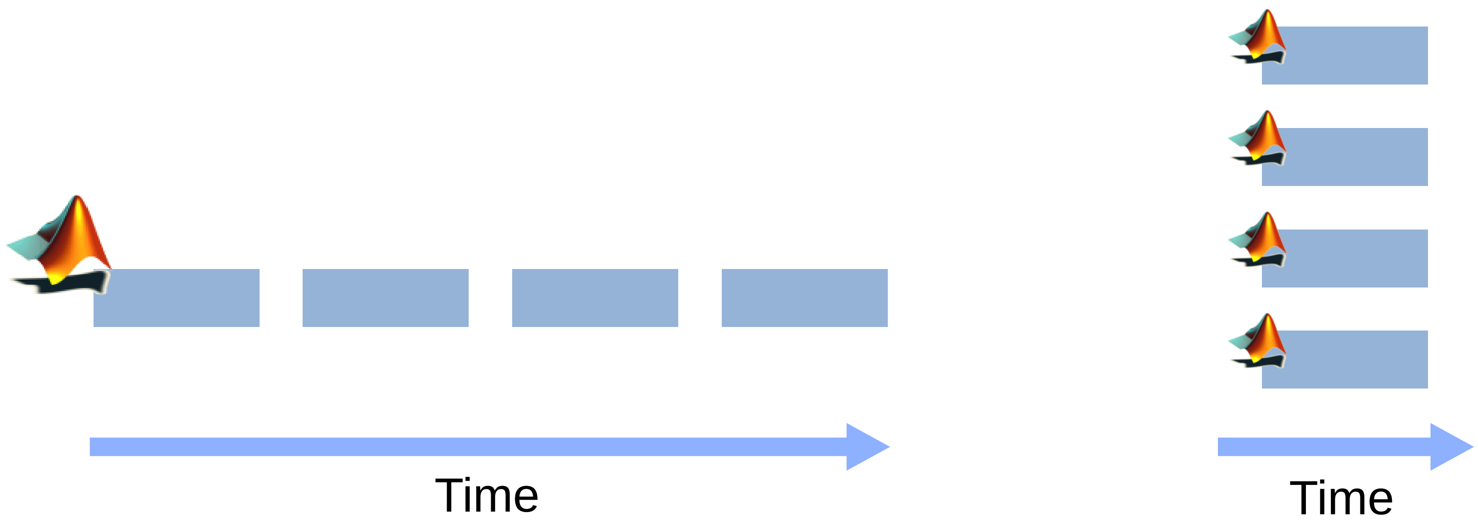
Directly leverage functions in Parallel Computing Toolbox

Programming Parallel Applications

Level of control	*Parallel Options*
Minimal	**Support built into Toolboxes**
Some	**High-Level Programming Constructs: (e.g. parfor, batch, distributed)**
Extensive	**Low-Level Programming Constructs: (e.g. Jobs/Tasks, MPI-based)**

Running Independent Tasks or Iterations

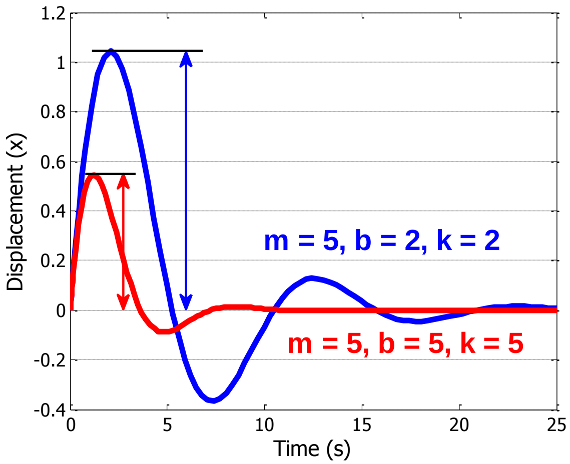
- Ideal problem for parallel computing
- No dependencies or communications between tasks
- Examples include parameter sweeps and Monte Carlo simulations



Example: Parameter Sweep of ODEs

- Solve a 2nd order ODE

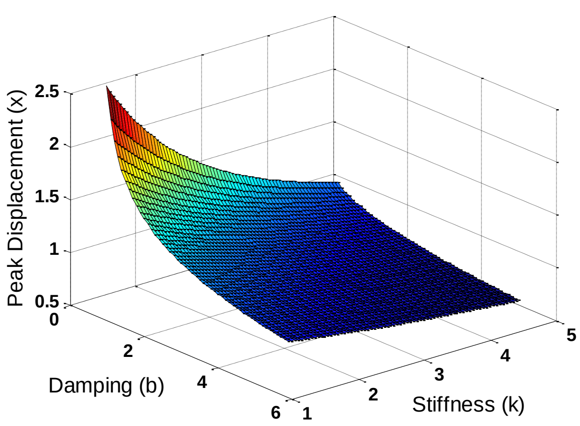
$$\overbrace{m}^{5}\ddot{x} + \underbrace{b}_{1,2,\ldots}\dot{x} + \underbrace{k}_{1,2,\ldots}x = 0$$



- Simulate with different values for ***b*** and ***k***

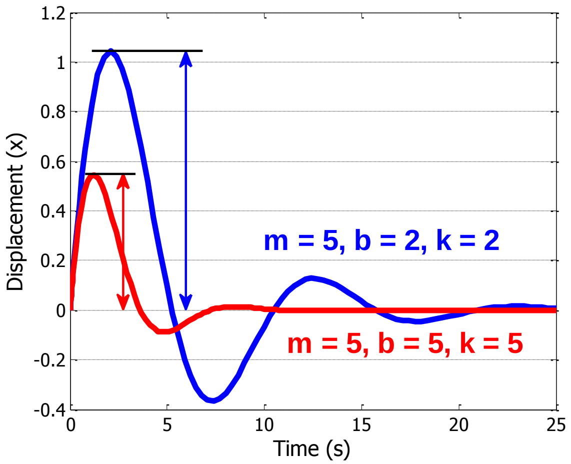
- Record peak value for each run

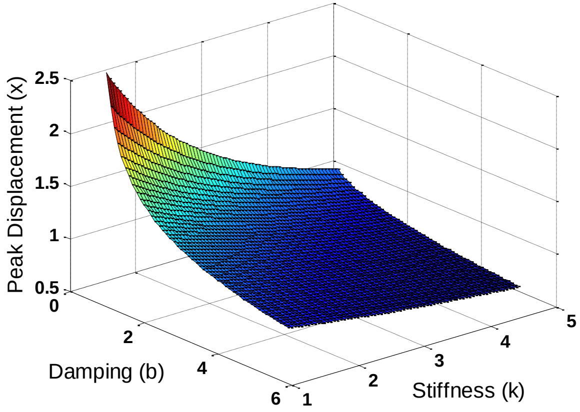
- Plot results



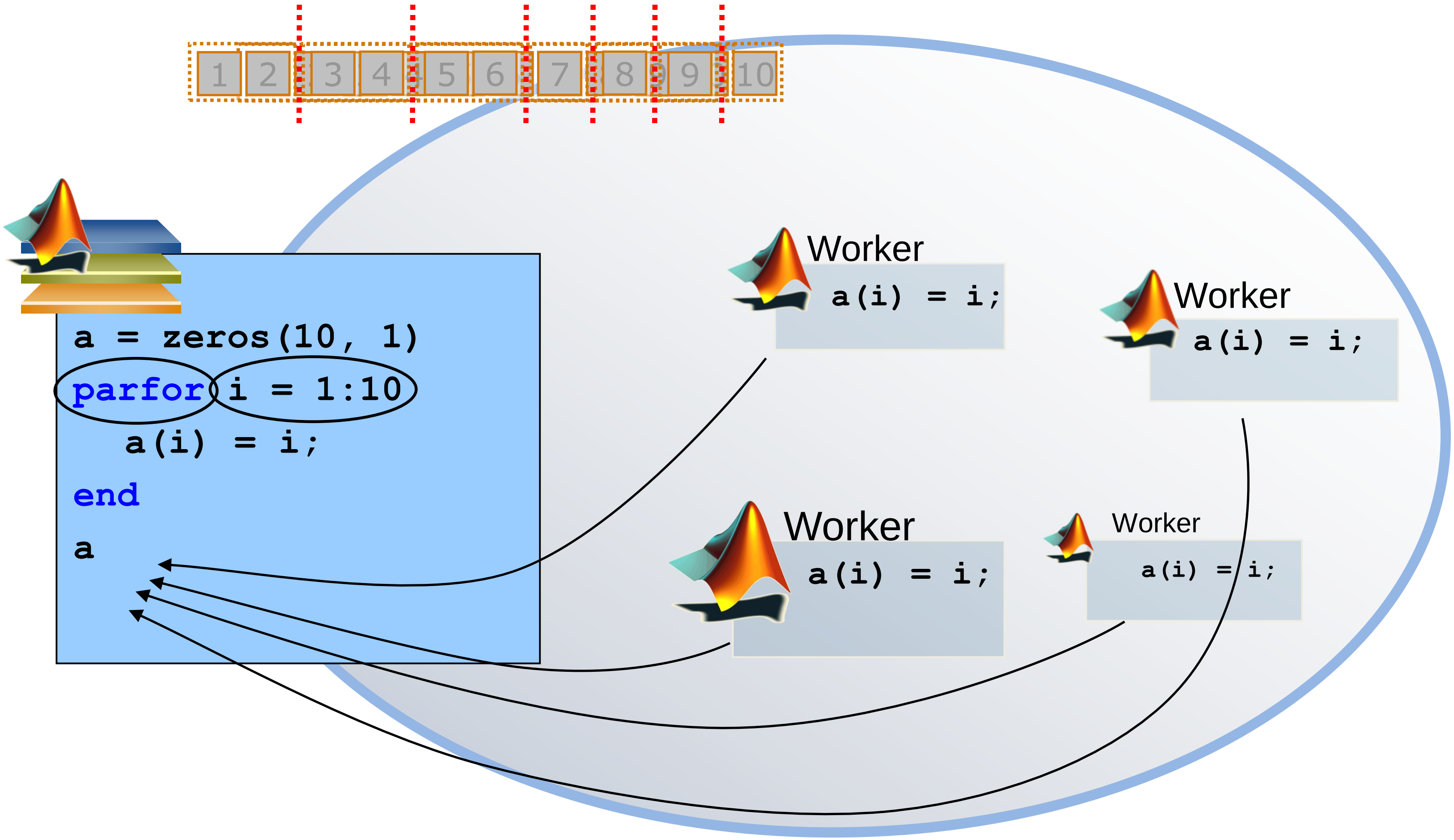
Summary of Example

- Mixed task-parallel and serial code in the same function
- Ran loops on a pool of MATLAB resources
- Used Code Analyzer to help in converting existing `for`-loop into `parfor`-loop





The Mechanics of `parfor` Loops



Pool of MATLAB Workers

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`

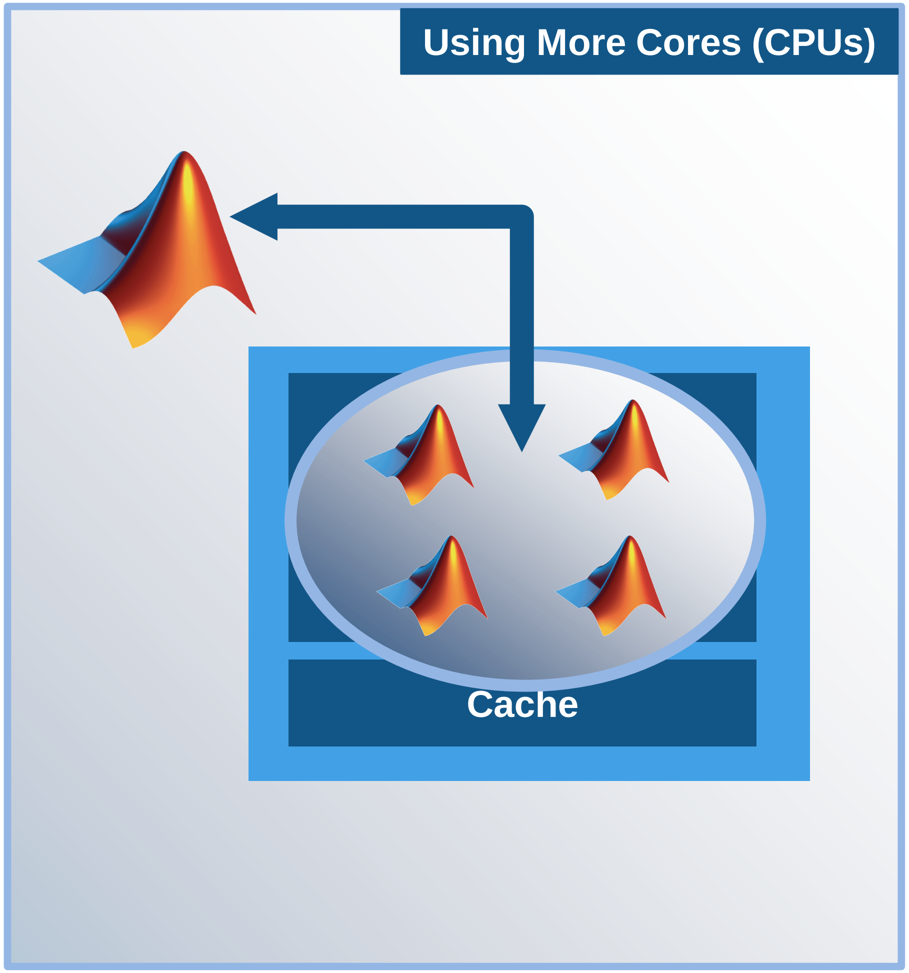
- Use Code Analyzer to diagnose `parfor` issues

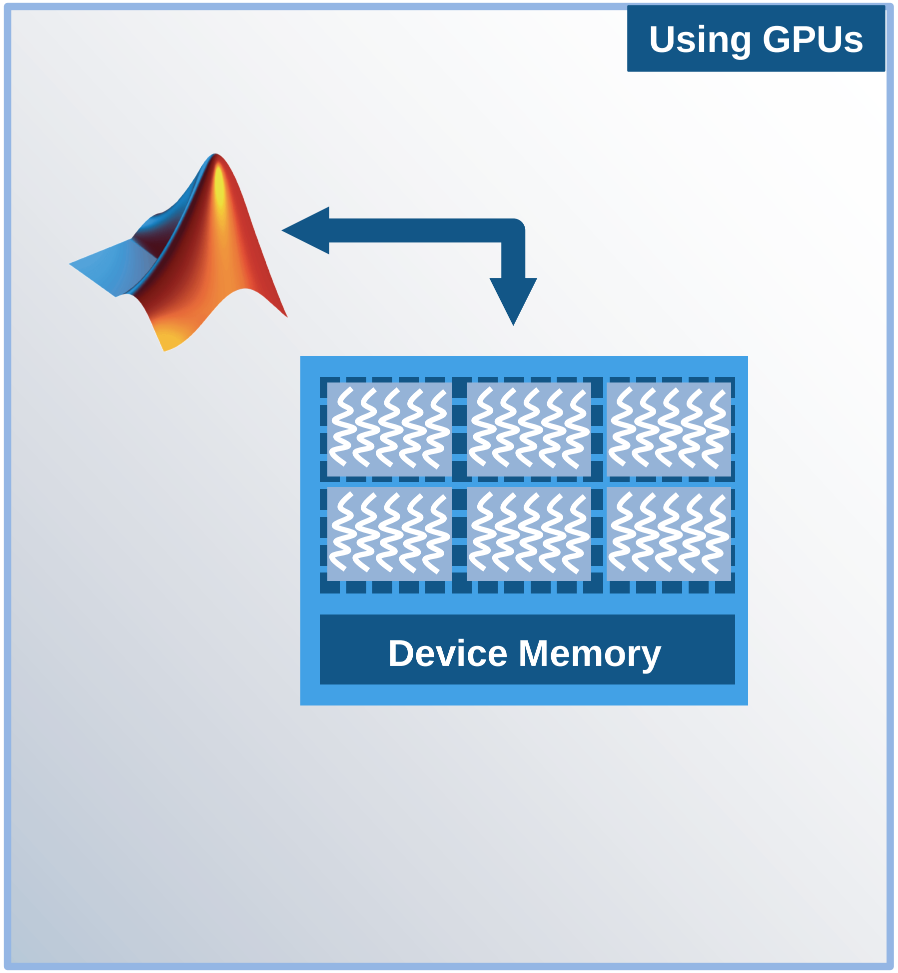
- If your `for` loop cannot be converted to a `parfor`, consider wrapping a subset of the body to a function

- Read the section in the documentation on classification of variables

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

Performance Gain with More Hardware





What is a Graphics Processing Unit (GPU)

New: GPU functionality in R2010b

- 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 greater, including NVIDIA Tesla 10-series and 20-series products. See http://www.nvidia.com/object/cuda_gpus.html for a complete listing

Summary of Options for Targeting GPUs

Ease of Use

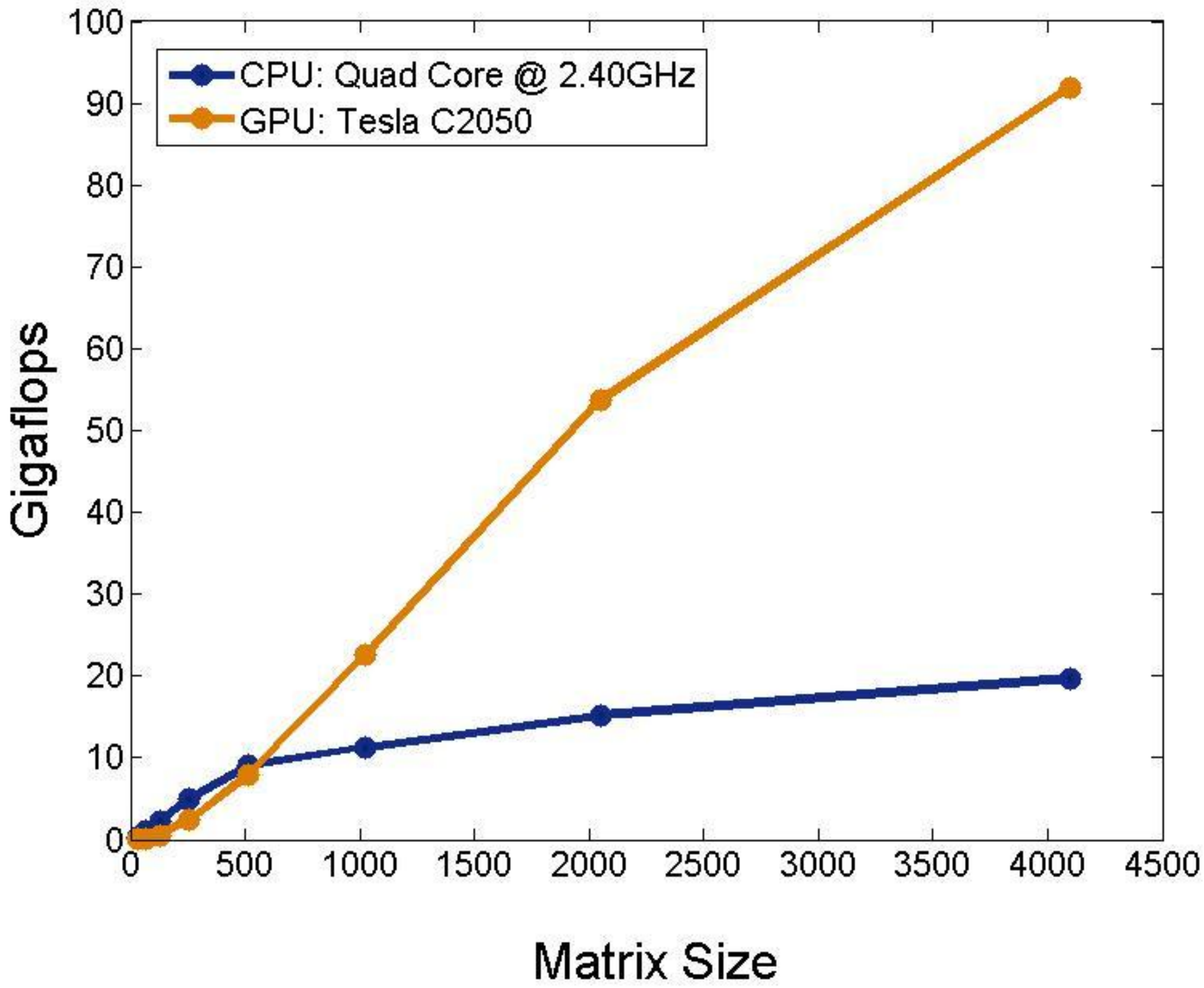
1) Use GPU array interface with MATLAB built-in functions

2) Execute custom functions on elements of the GPU array

3) Invoke your CUDA kernels directly from MATLAB

Greater Control

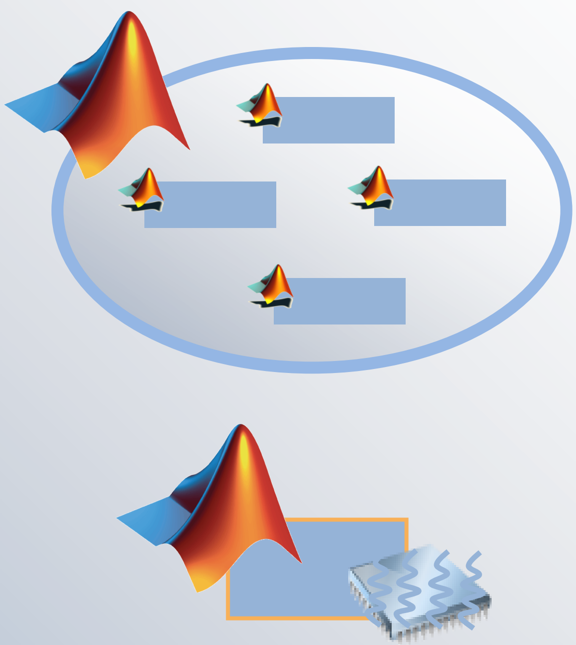
Performance: A\b with Double Precision



Parallel Computing enables you to …

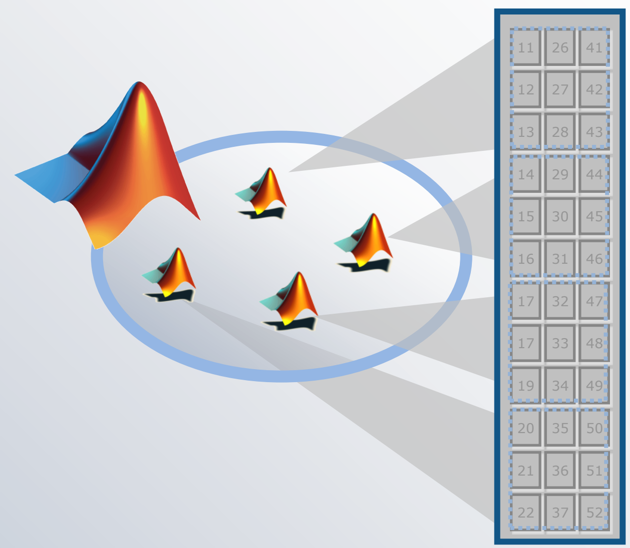
Larger Compute Pool

Speed up Computations



Larger Memory Pool

Work with Large Data

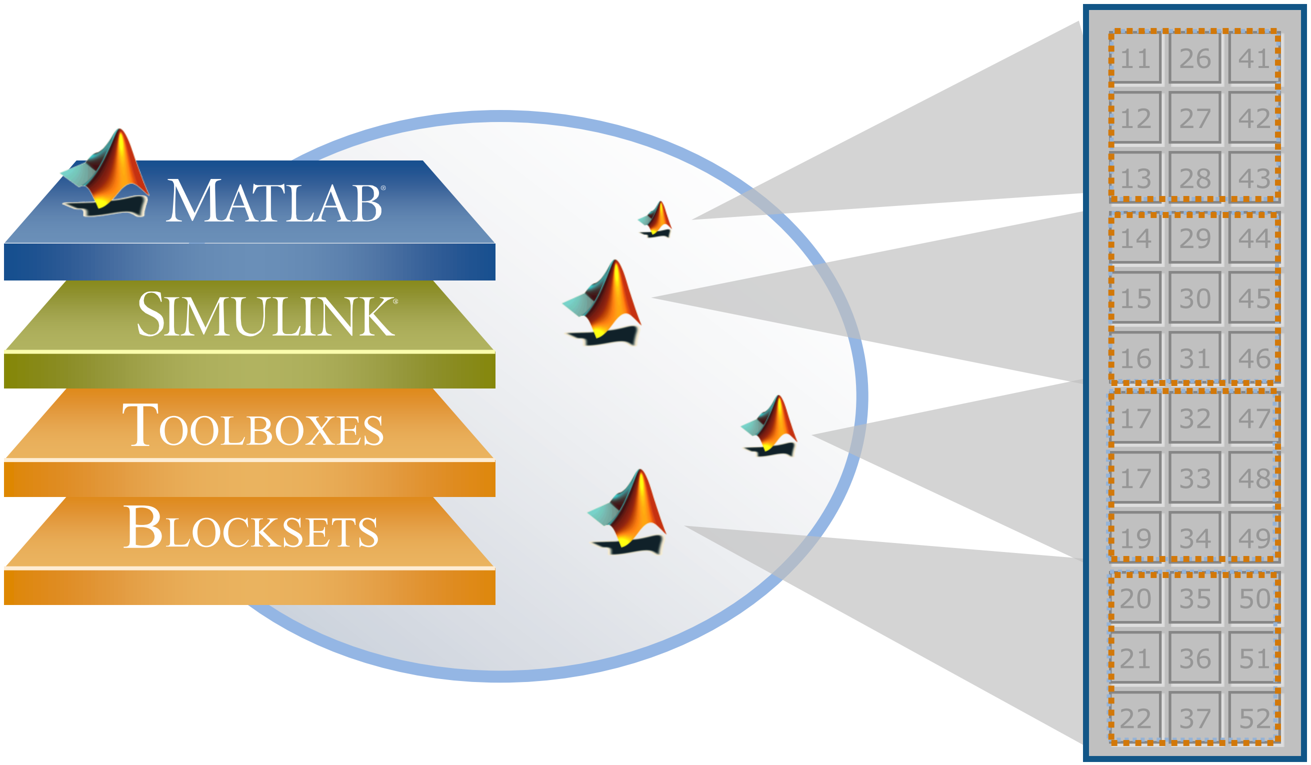


Limited Process Memory

- 32-bit platforms
  - Windows 2000 and XP (by default): 2 GB
  - Linux/UNIX/MAC system configurable: 3-4 GB
  - Windows XP with /3gb boot.ini switch: 3 GB

- 64-bit platforms
  - Linux/UNIX/MAC: 8 TB
  - Windows XP Professional x64: 8TB

Client-side Distributed Arrays



Remotely Manipulate Array
from Desktop

Distributed Array
Lives on the Cluster

Enhanced MATLAB Functions That Operate on Distributed Arrays

Type of Function	Function Names
Data functions	cumprod, cumsum, fft, max, min, prod, sum
Data type functions	cast, cell2mat, cell2struct, celldisp, cellfun, char, double, fieldnames, int16, int32, int64, int8, logical, num2cell, rmfield, single, struct2cell, swapbytes, typecast, uint16, uint32, uint64, uint8
Elementary and trigonometric functions	abs, acos, acosd, acosh, acot, acotd, acoth, acsc, acscd, acsch, angle, asec, asecd, asech, asin, asind, asinh, atan, atan2, atand, atanh, ceil, complex, conj, cos, cosd, cosh, cot, cotd, coth, csc, cscd, csch, exp, expm1, fix, floor, hypot, imag, isreal, log, log10, log1p, log2, mod, nextpow2, nthroot, pow2, real, reallog, realpow, realsqrt, rem, round, sec, secd, sech, sign, sin, sind, sinh, sqrt, tan, tand, tanh
Elementary matrices	cat, diag, eps, find, isempty, isequal, isequalwithequalnans, isfinite, isinf, isnan, length, ndims, size, tril, triu
Matrix functions	chol, eig, lu, norm, normest, svd
Array operations	all, and, any, bitand, bitor, bitxor, ctranspose, end, eq, ge, gt, horzcat, ldivide, le, lt, minus, mldivide, mrdivide, mtimes, ne, not, or, plus, power, rdivide, subsasgn, subsindex, subsref, times, transpose, uminus, uplus, vertcat, xor
Sparse matrix functions	full, issparse, nnz, nonzeros, nzmax, sparse, spfun, spones
Special functions	dot

`spmd` blocks

```
spmd
    % single program across workers
end
```

- Mix parallel and serial code in the same function
- Run on a pool of MATLAB resources
- Single Program runs simultaneously across workers
  - Distributed arrays, message-passing
- Multiple Data spread across multiple workers
  - Data stays on workers

Programming Parallel Applications

Level of control	*Parallel Options*
Minimal	Support built into Toolboxes
Some	High-Level Programming Constructs: (e.g. parfor, batch, distributed)
Extensive	**Low-Level Programming Constructs: (e.g. Jobs/Tasks, MPI-based)**

MPI-Based Functions in Parallel Computing Toolbox™

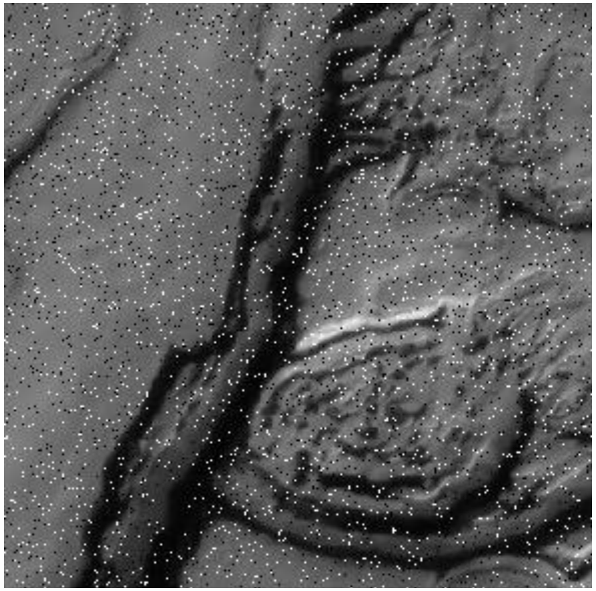
Use when a high degree of control over parallel algorithm is required

- High-level abstractions of MPI functions
  - `labSendReceive`, `labBroadcast`, and others
  - Send, receive, and broadcast any data type in MATLAB
- Automatic bookkeeping
  - Setup: communication, ranks, etc.
  - Error detection: deadlocks and miscommunications
- Pluggable
  - Use any MPI implementation that is *binary*-compatible with MPICH2

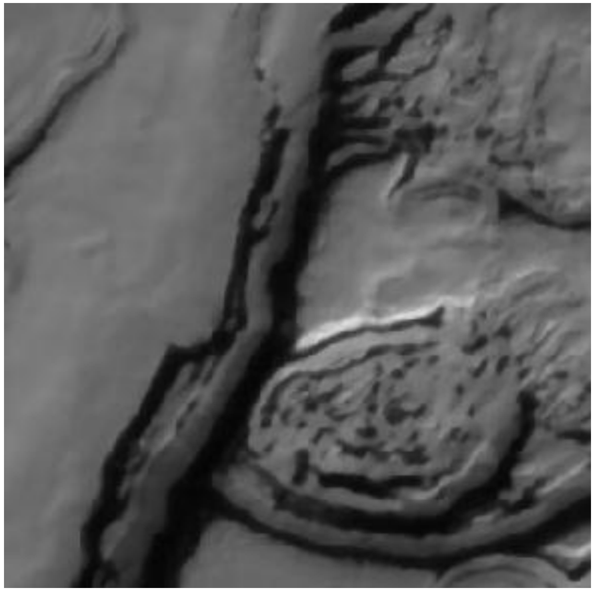
Example: Image De-Noising (Large Image Processing)

Use median filtering to reduce "salt & pepper" noise.

Noisy Image

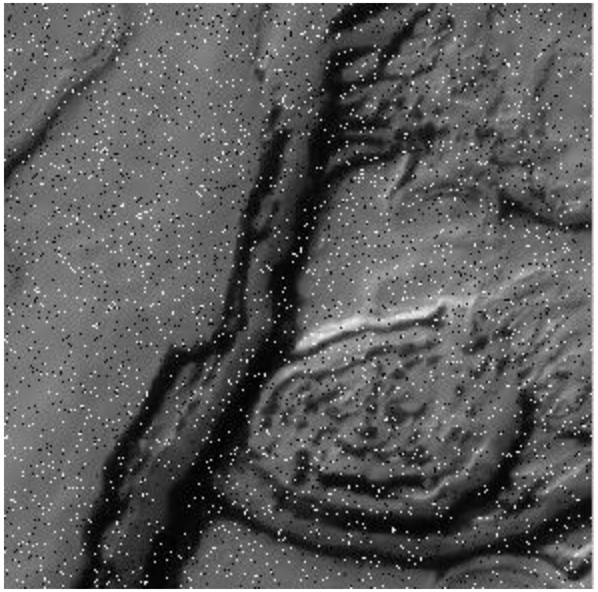


Filtered Image

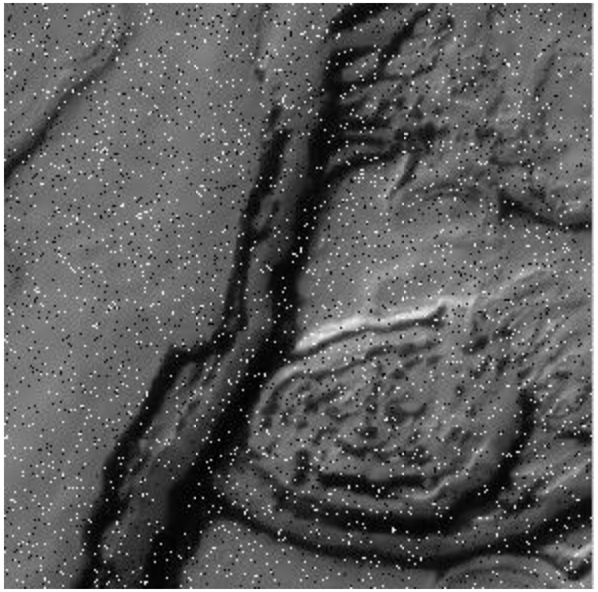


http://hirise.lpl.arizona.edu/
From - NASA/JPL/University of Arizona

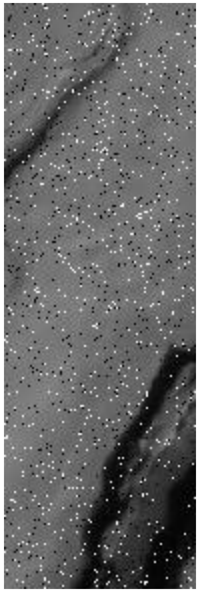
Noisy Image – too large for a desktop

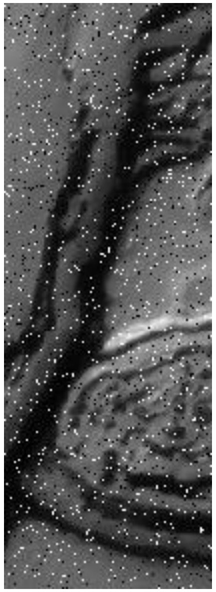


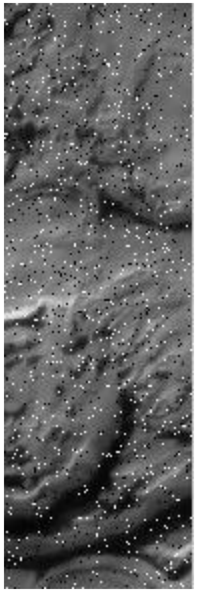
Distribute Data



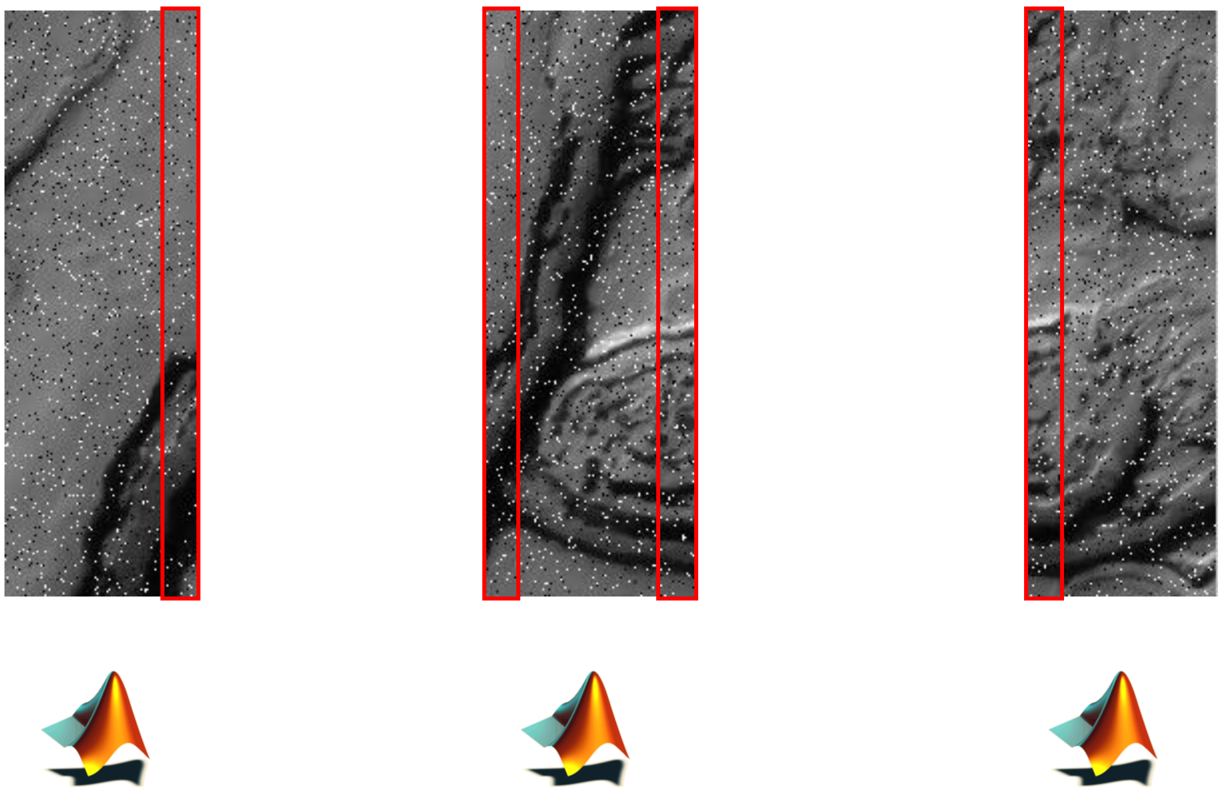
Distribute Data



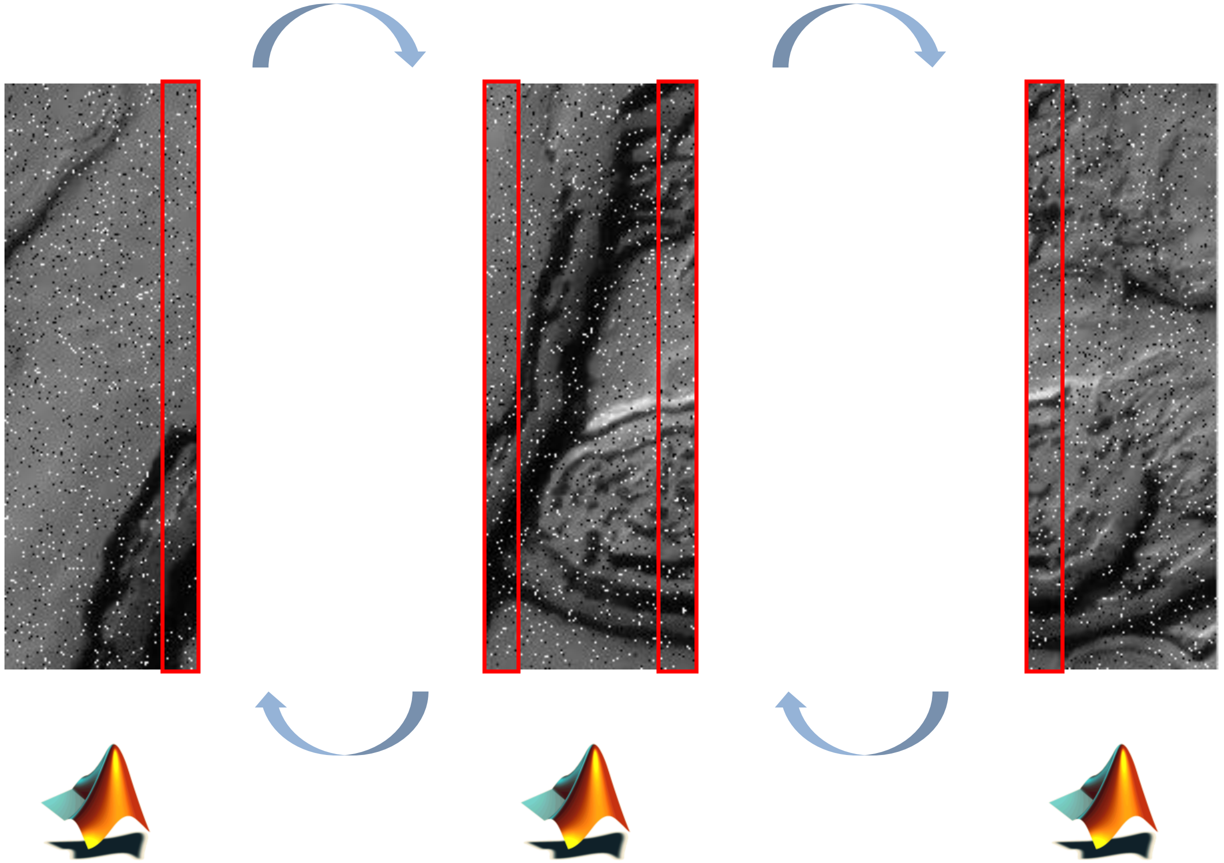




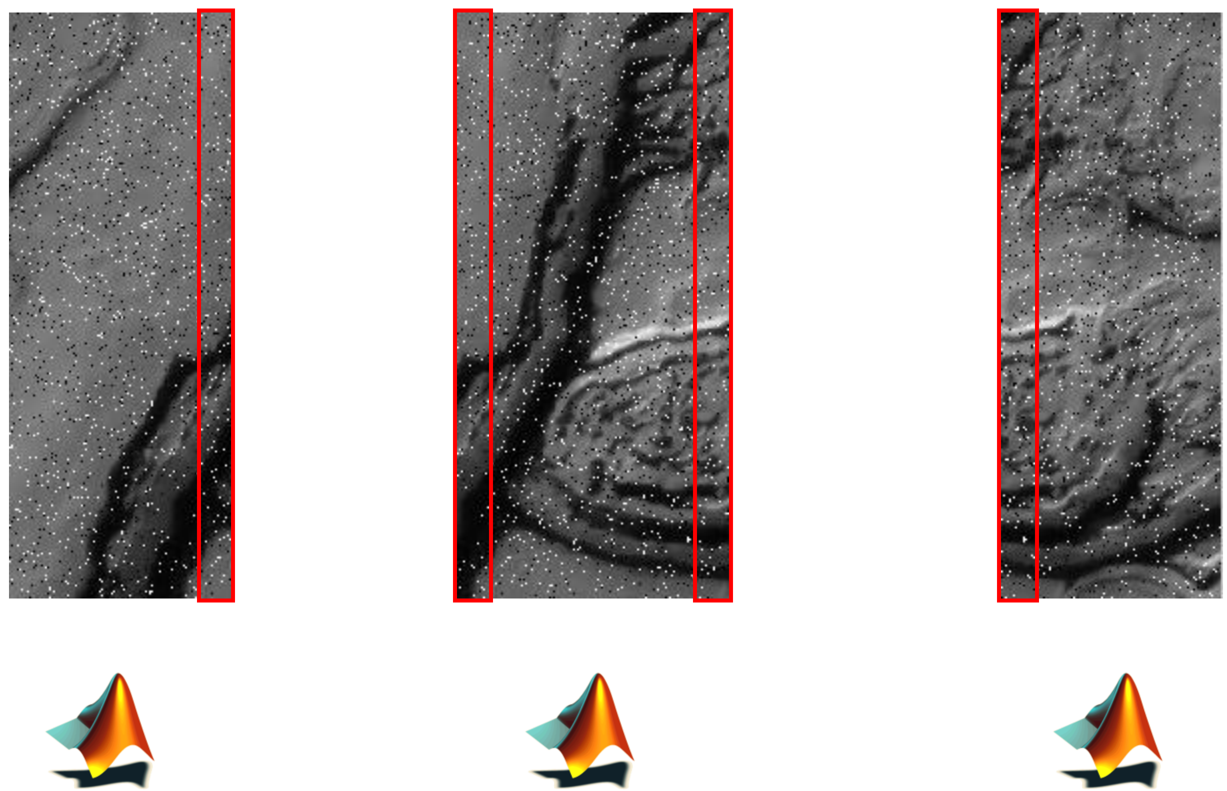
Pass Overlap Data



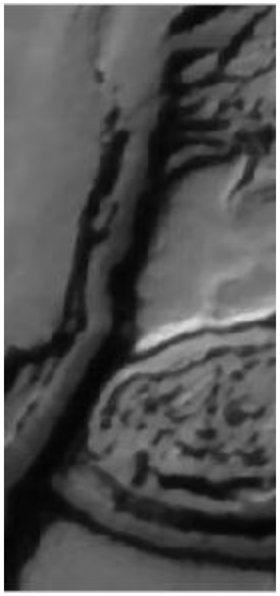
Pass Overlap Data

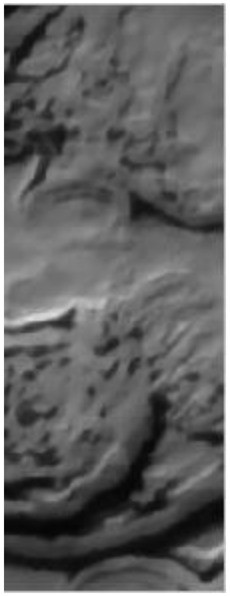


Pass Overlap Data

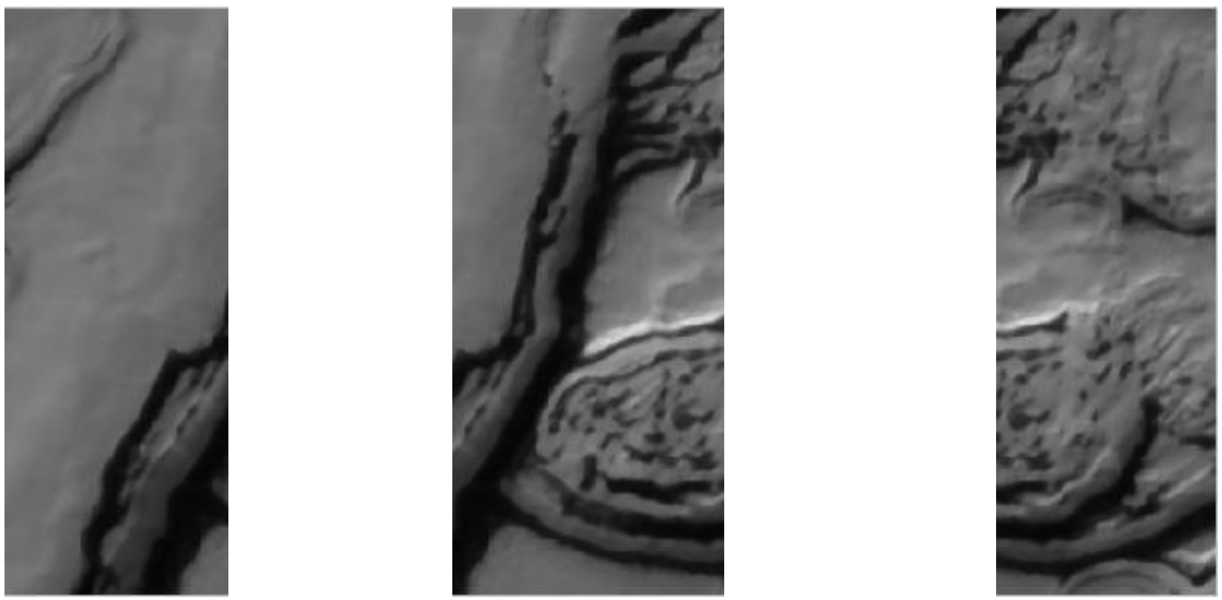


Apply Median Filter

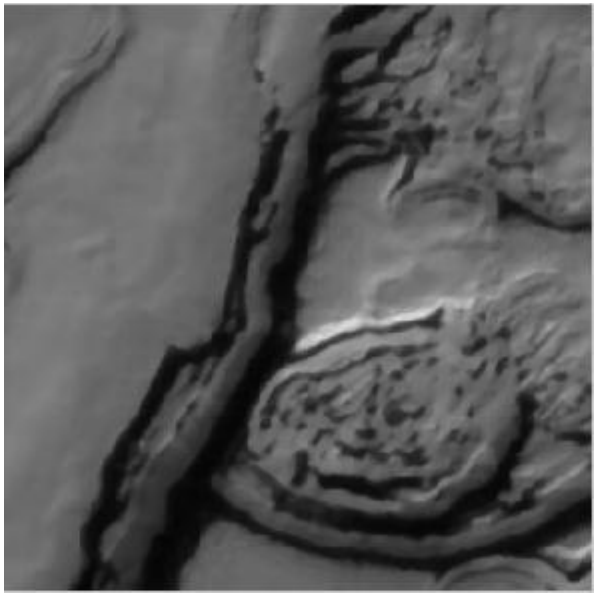




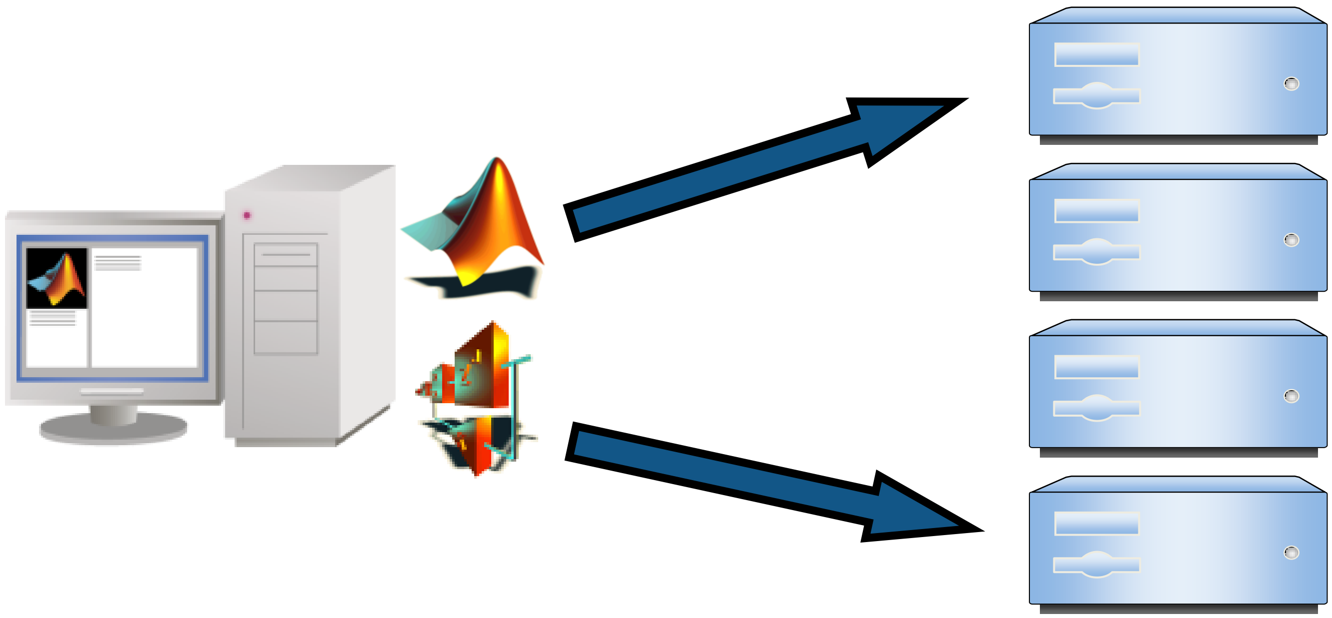
Combine as Distributed Data



Combine as Distributed Data



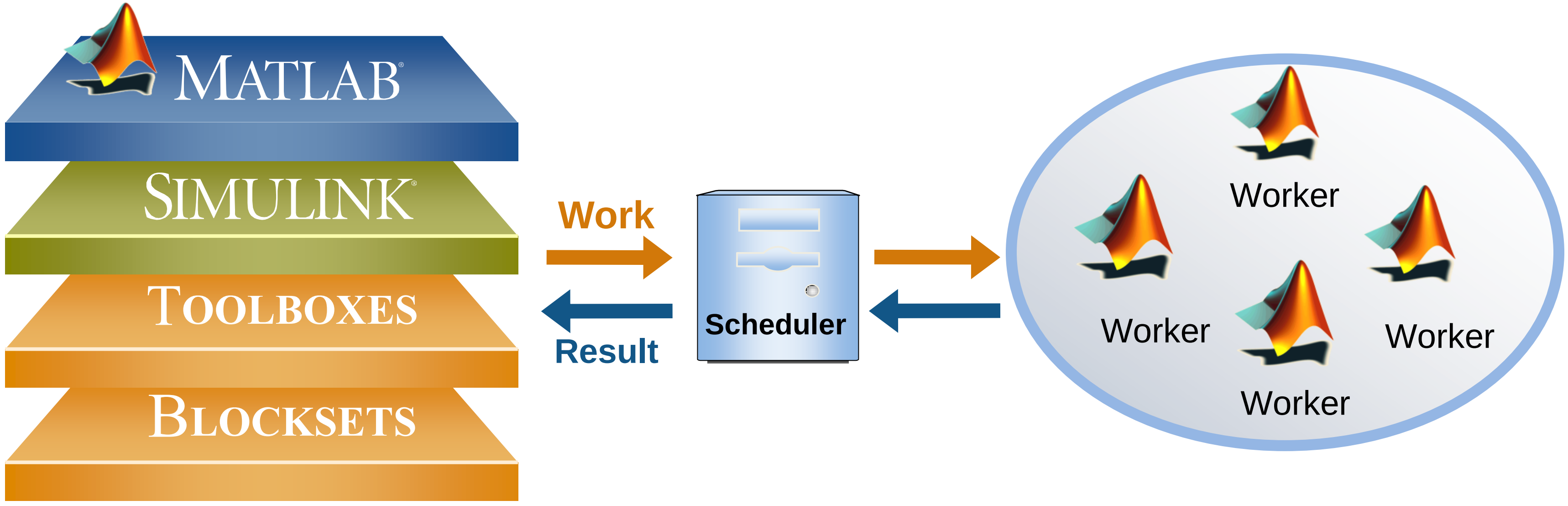
Scheduling Applications



Interactive to Scheduling

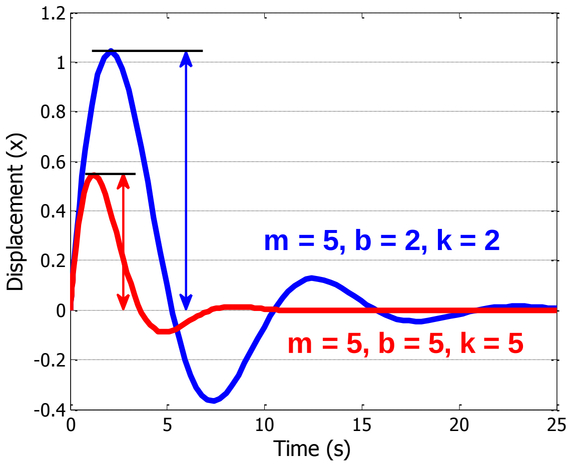
- Interactive
  - Great for prototyping
  - Immediate access to MATLAB workers
- Scheduling
  - Offloads work to other MATLAB workers (local or on a cluster)
  - Access to more computing resources for improved performance
  - Frees up local MATLAB session

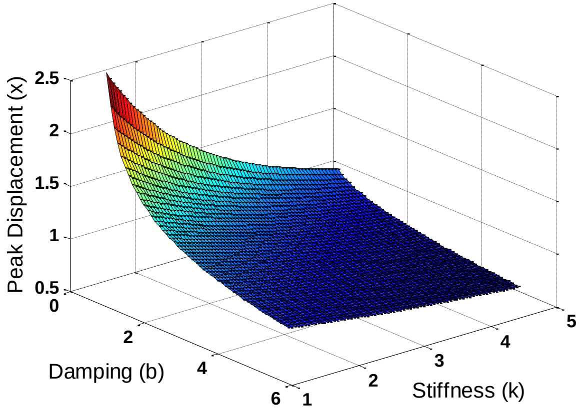
Scheduling Work



Example: Schedule Processing

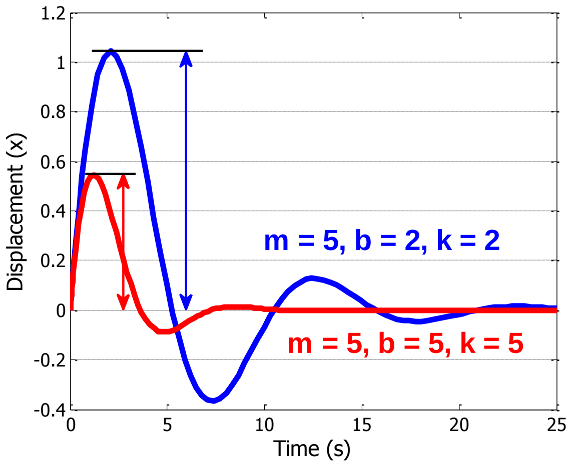
- Offload parameter sweep to local workers
- Get peak value results when processing is complete
- Plot results in local MATLAB

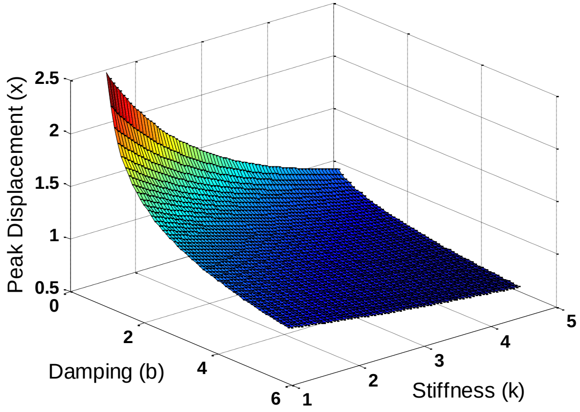




Summary of Example

- Used `batch` for off-loading work
- Used `matlabpool` option to off-load and run in parallel
- Used `load` to retrieve worker's workspace

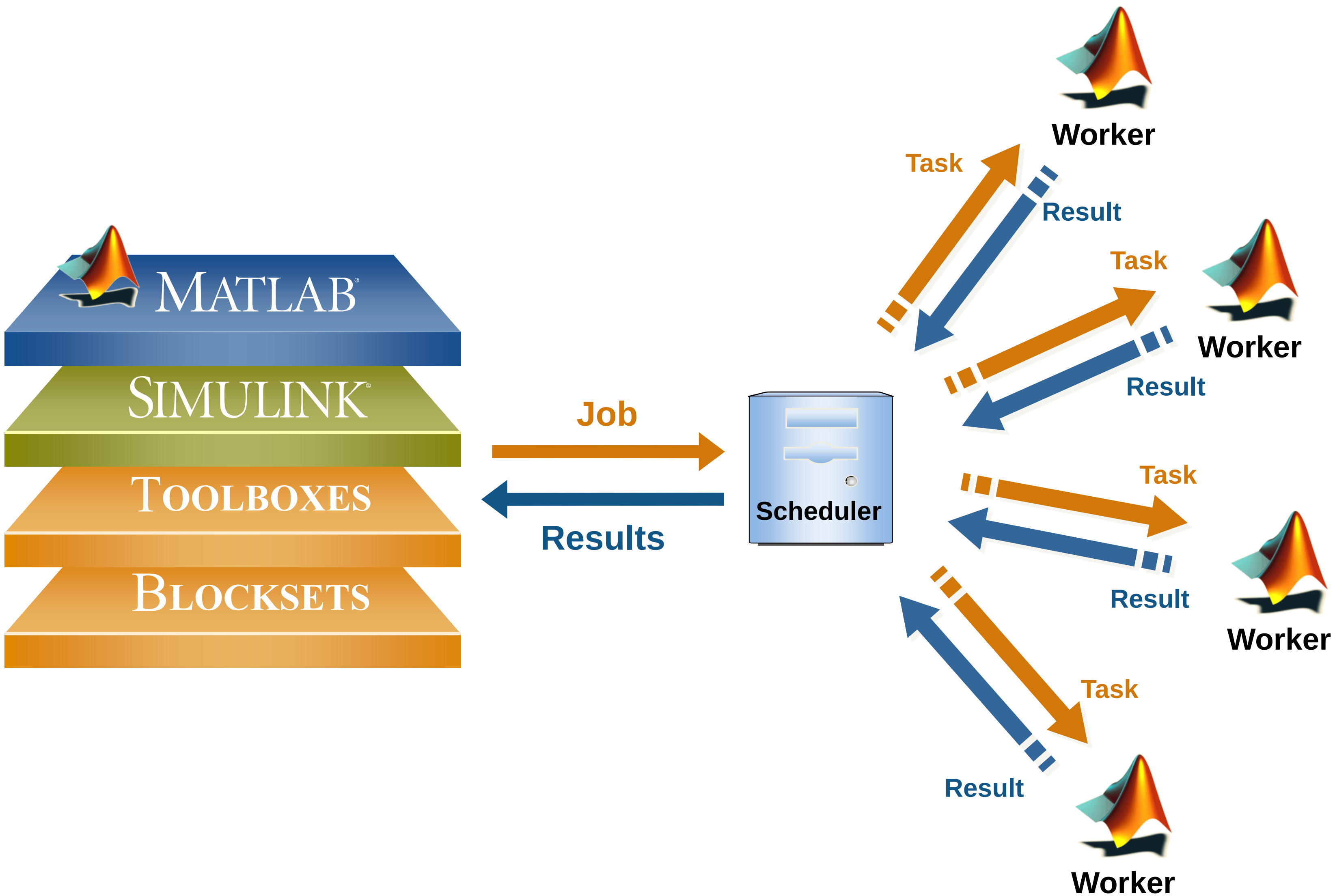




Scheduling Workflows

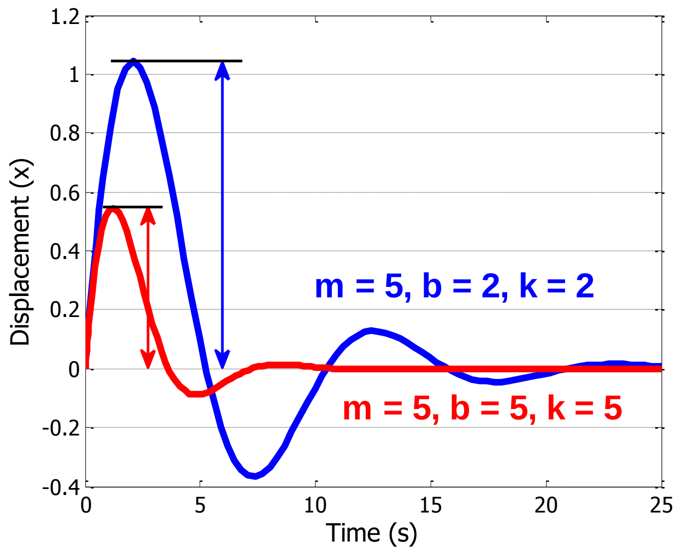
- `parfor`
  - Multiple independent iterations
  - Easy to combine serial and parallel code
  - Workflow
    - Interactive using `matlabpool`
    - Scheduled using `batch`
- `jobs/tasks`
  - Series of independent tasks; not necessarily iterations
  - Workflow ➔ Always scheduled

Scheduling Jobs and Tasks



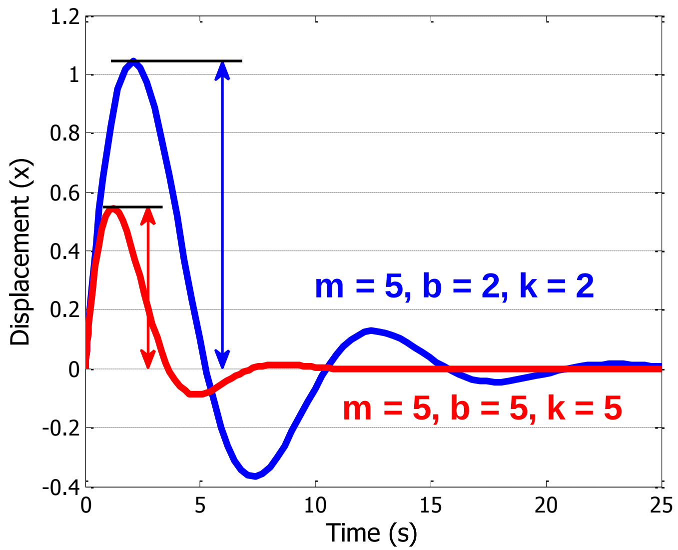
Example: Scheduling Independent Simulations

- Offload three independent approaches to solving our previous ODE example
- Retrieve simulated displacement as a function of time for each simulation
- Plot comparison of results in local MATLAB



Summary of Example

- Used `findResource` to find scheduler
- Used `createJob` and `createTask` to set up the problem
- Used `submit` to off-load and run in parallel
- Used `getAllOutputArguments` to retrieve all task outputs

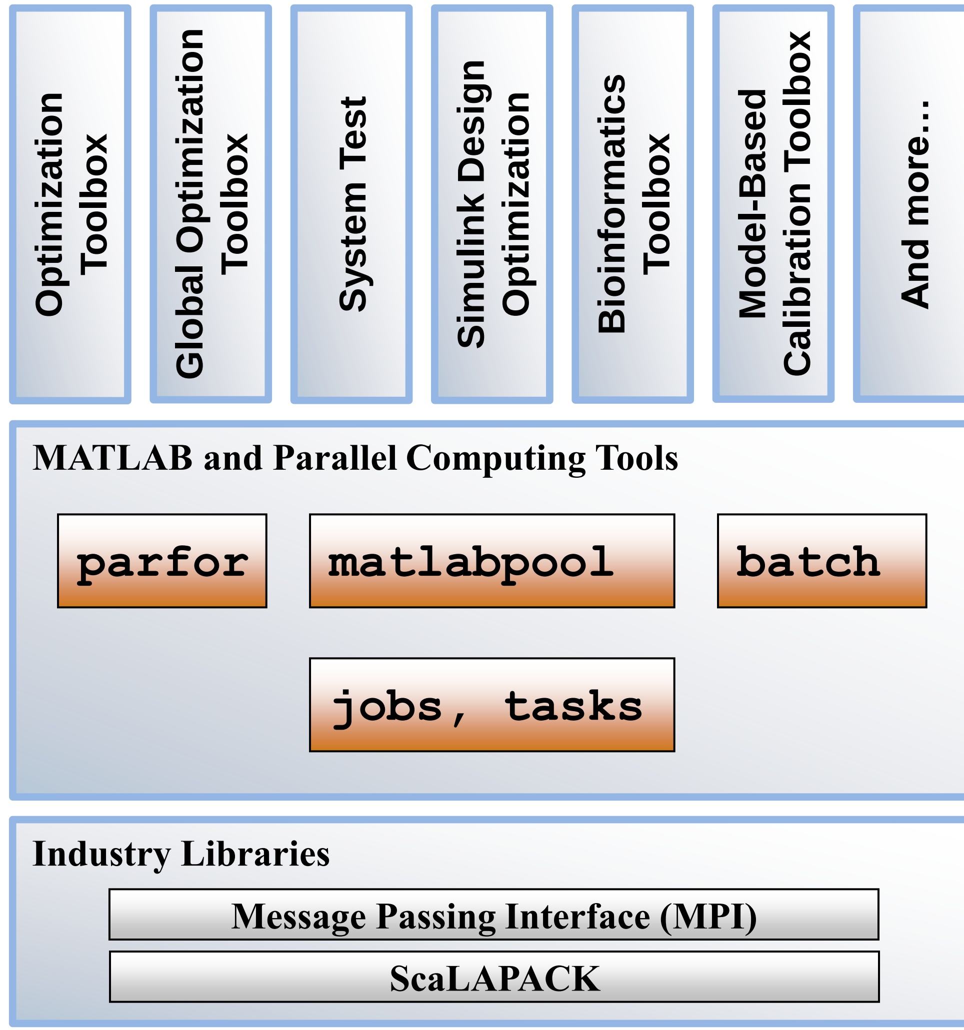


Factors to Consider for Scheduling

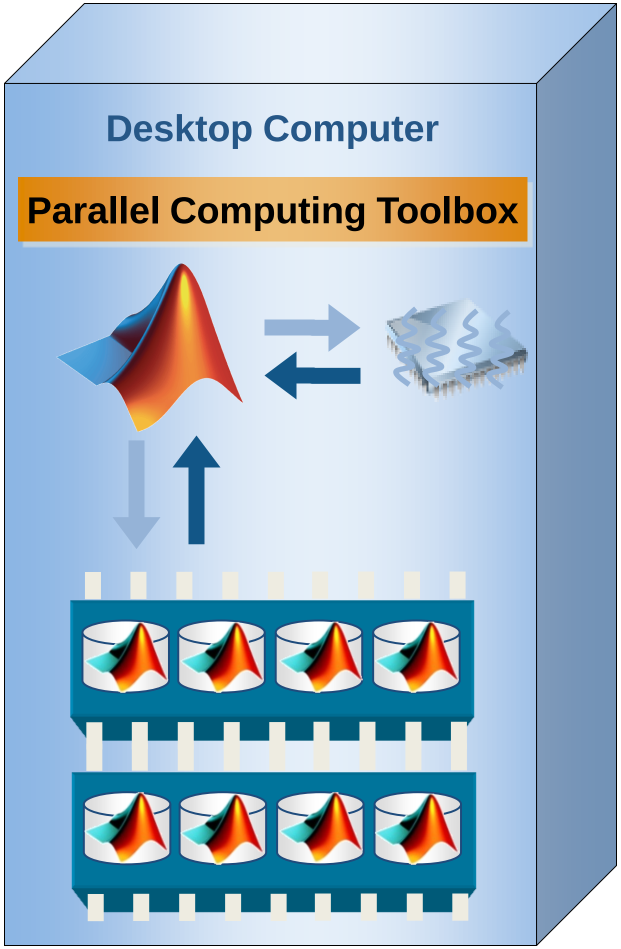
- There is always an overhead to distribution
  - Combine small repetitive function calls
- Share code and data with workers efficiently
  - Set job properties (`FileDependencies, PathDependencies`)
- Minimize I/O
  - Enable `Workspace` option for `batch`
- Capture command window output
  - Enable `CaptureDiary` option for `batch`

Parallel Computing with MATLAB

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

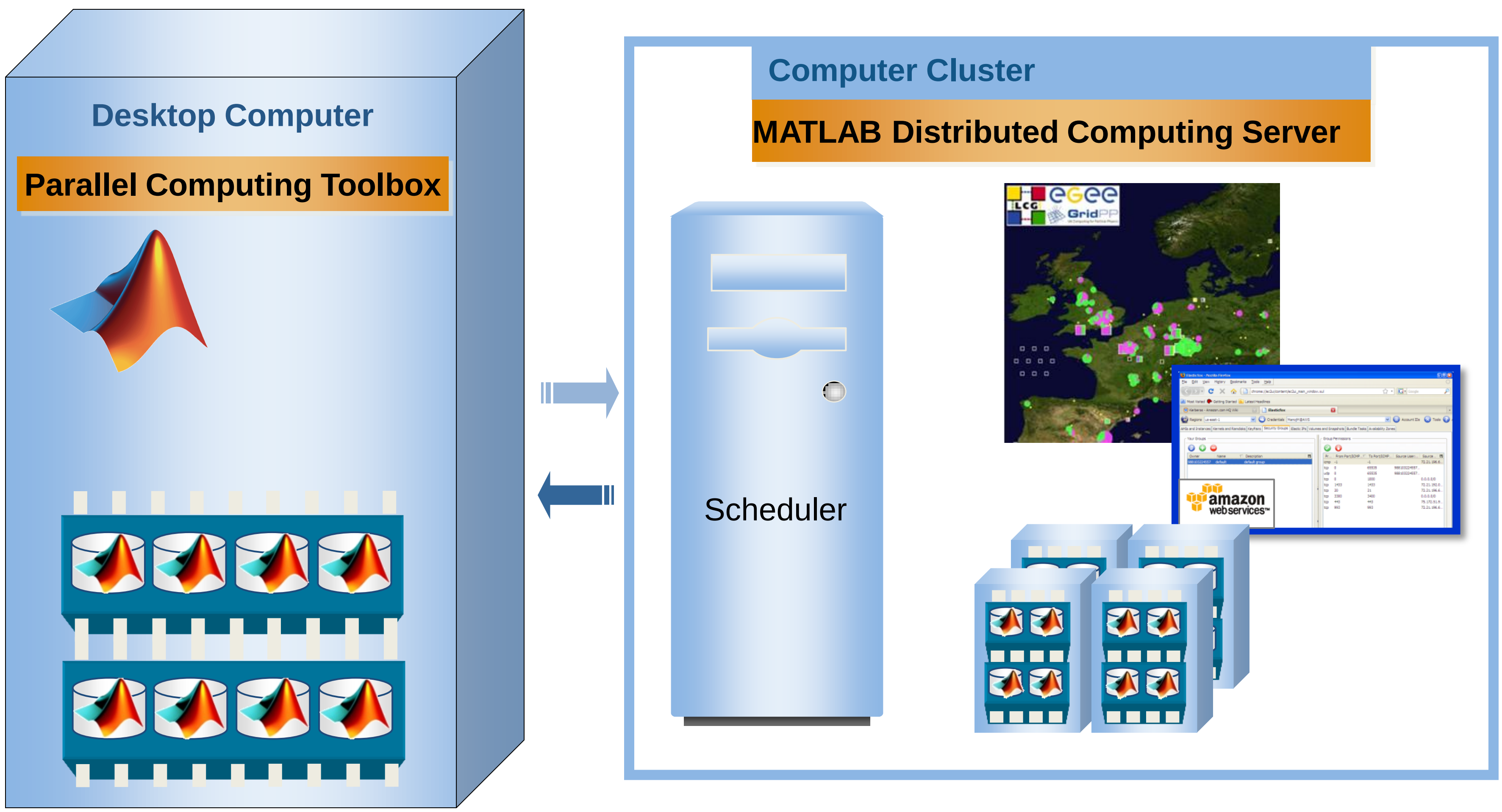


Parallel Computing on the Desktop



- Rapidly develop parallel applications on local computer
- Take full advantage of desktop power by using CPUs and GPUs
- Separate computer cluster not required

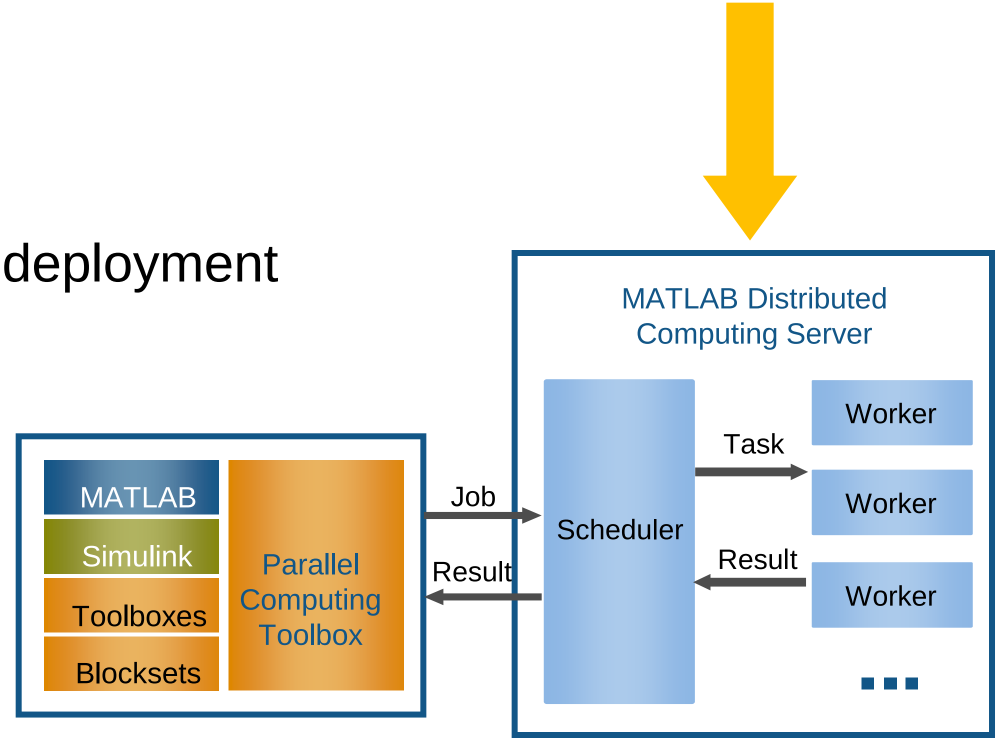
Scale Up to Clusters, Grids and Clouds



Licensing: MATLAB® Distributed Computing Server™

- One key required per worker:
  - Packs of 8, 16, 32, 64, 128, etc.
  - *Worker* is a MATLAB® session, not a processor

- All-product install
  - No code generation or deployment products



Support for Schedulers

Direct Support

Open API for others

MathWorks Contact Information

For pricing, licensing, trials and general questions:

Tim Mathieu
Sr. Account Manager
Education Sales Department
Email: Tim.Mathieu@mathworks.com
Phone: 508.647.7016

Customer Service: service@mathworks.com
508.647.7000

Technical Support: support@mathworks.com
508.647.7000