# Introduction to MDCS

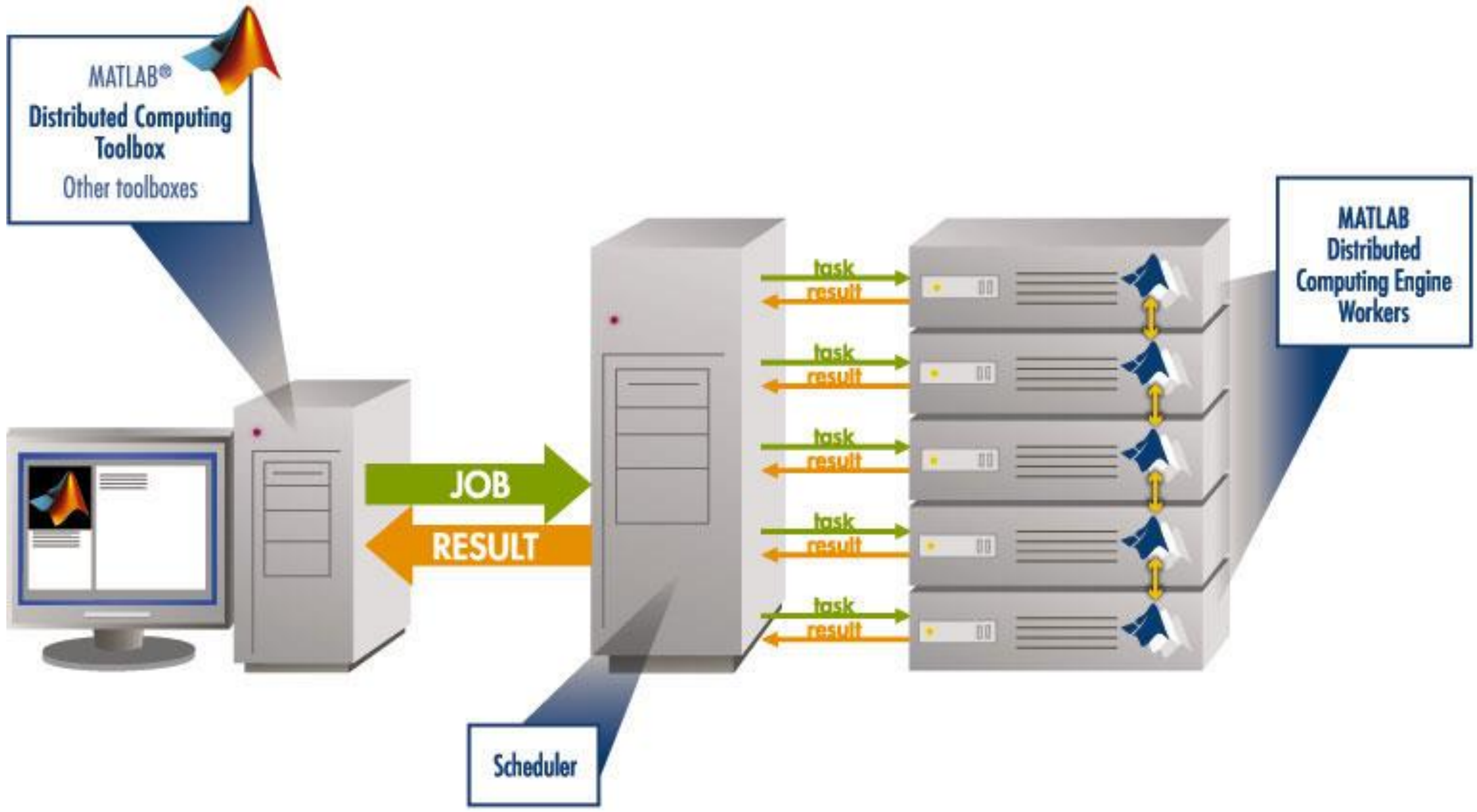
- Matlab Distributed Compute Server
- Preparing Matlab for MDCS
- Example

# What is MDCS

Matlab on your desktop computer:

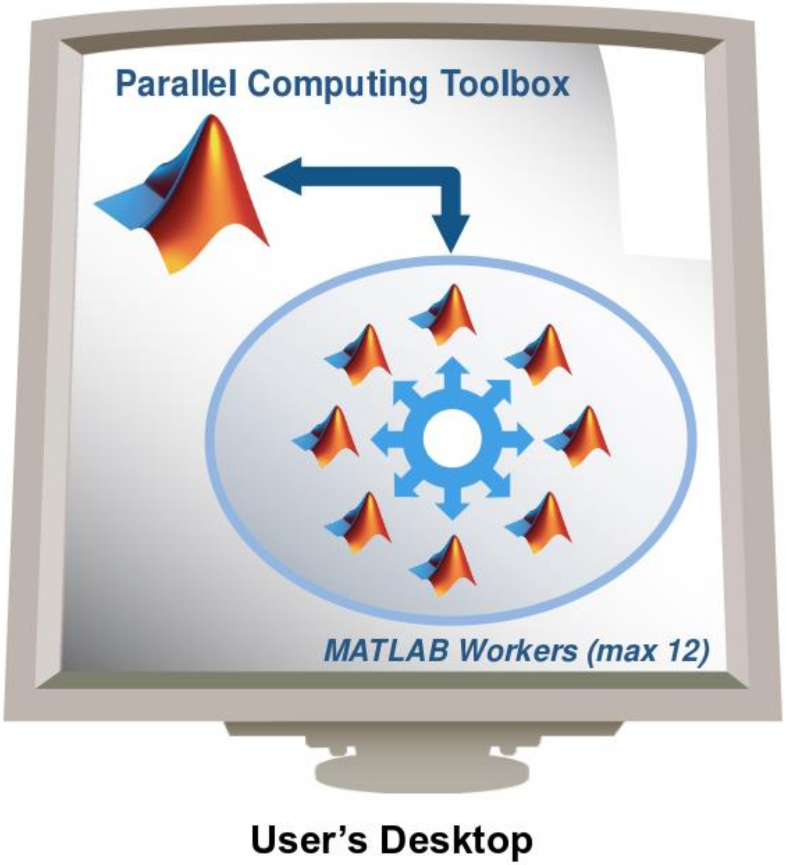
- you are limited by the compute power of your local machine
    - memory
    - CPU speed
- you can only run one job at a time
- your machine may become unusable while your Matlab job is running

# What is MDCS


MATLAB®
Distributed Computing Toolbox
Other toolboxes
JOB
RESULT
task
result
task
result
task
result
task
result
task
result
MATLAB Distributed Computing Engine Workers
Scheduler

*(taken from MathWorks marketing)*

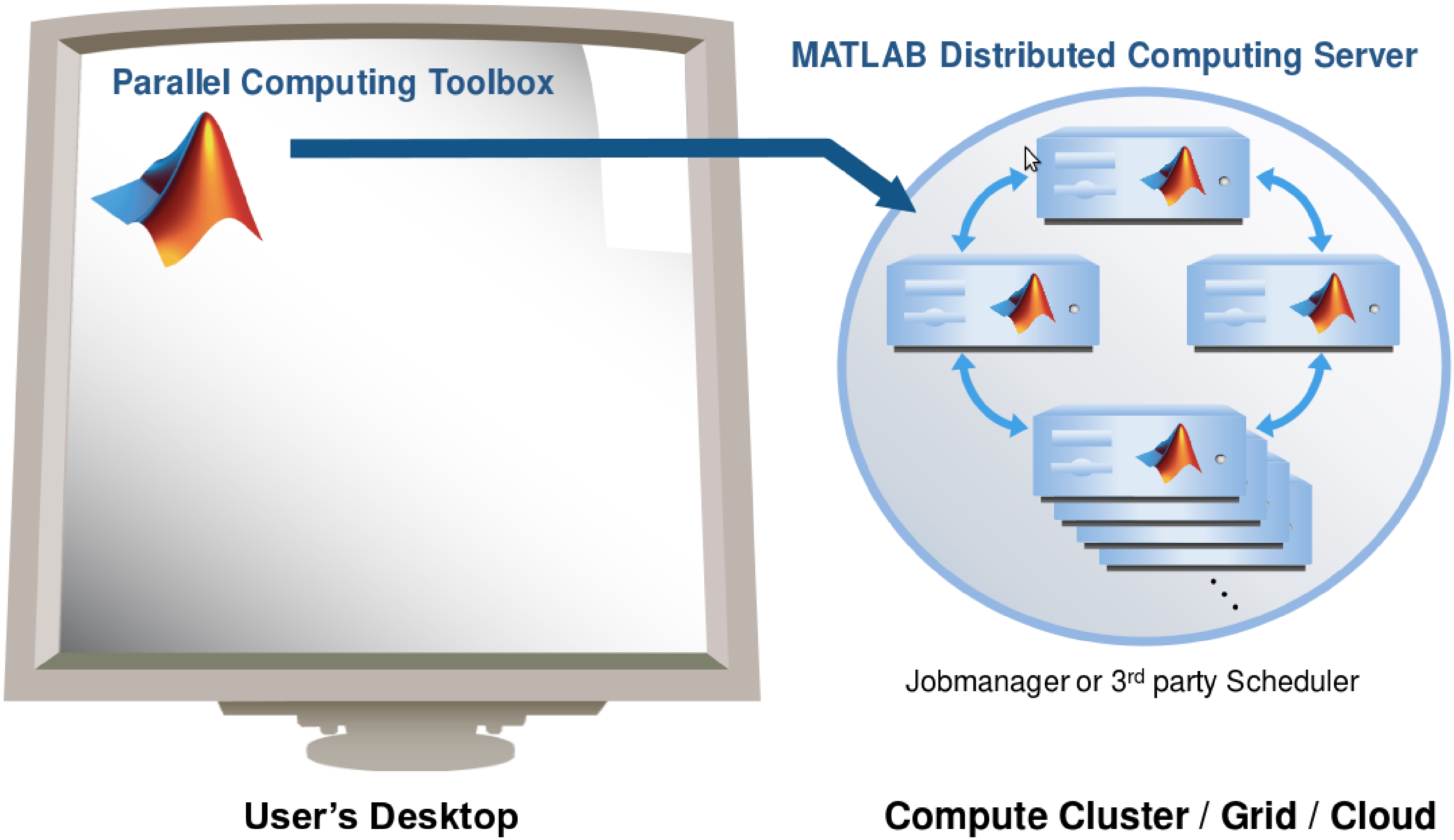
# Parallel Computing with Matlab




- easily experiment with explicit parallelism on multicore machines
- rapidly develop parallel applications on local computer
- take full advantage of desktop power, incl. GPUs
- separate compute cluster not required

*(taken from MathWorks marketing)*

# Parallel Computing with Matlab



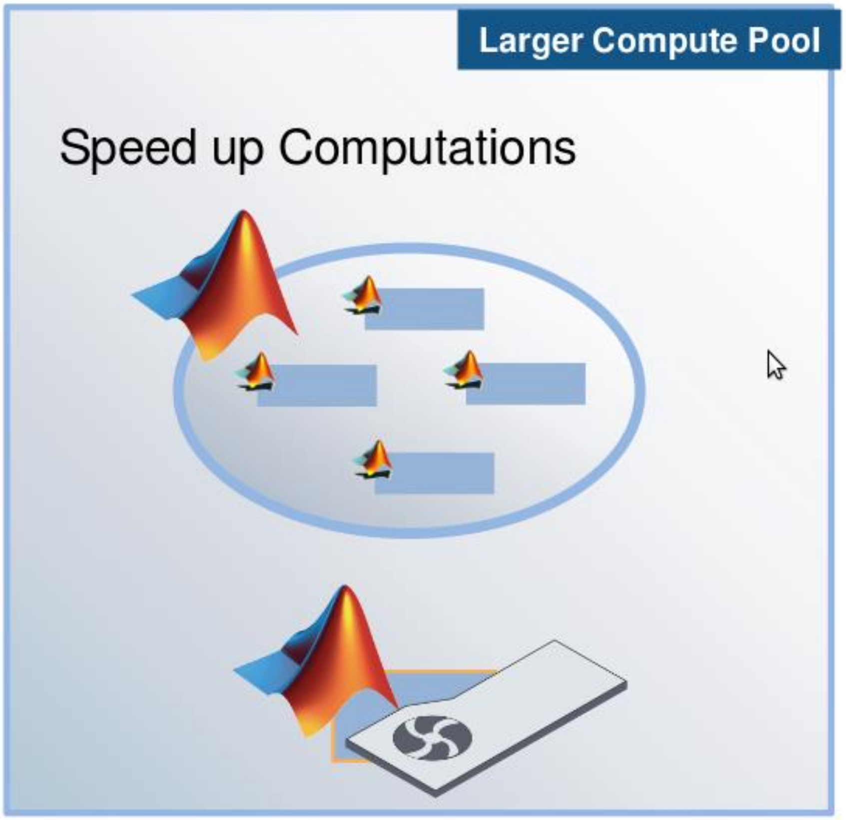
# What is MDCS

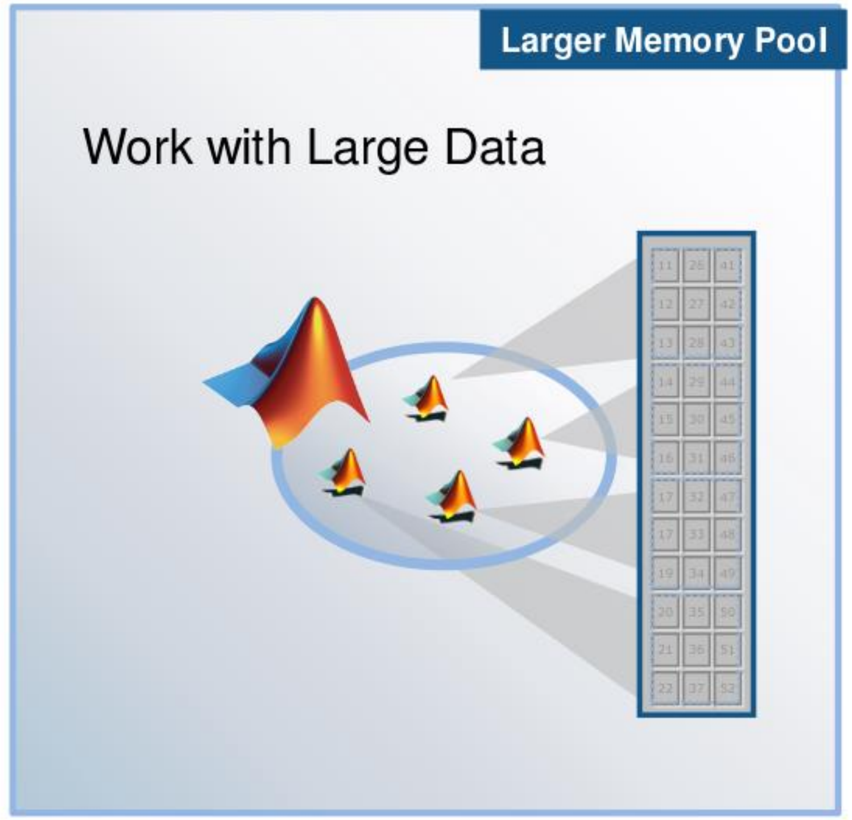
- MDCS allows you to off-load Matlab programs to a compute server
- simplified workflow
  - you can develop and test your application locally before submitting jobs, also in parallel
  - results are automatically returned to your local machine for post-processing
- the Parallel Computing Toolbox provides utilities for parallelization
  - task-parallel
  - data-parallel

# Why to use MDCS on the Cluster?

- with MDCS come 224 worker licenses
  - these are in addition to the normal Matlab licenses (200)
  - you can use also any of the toolboxes (50)
  - allows the control over used licenses and prevents failed jobs
  - for fair sharing not more than 36 MDCS licenses should be used

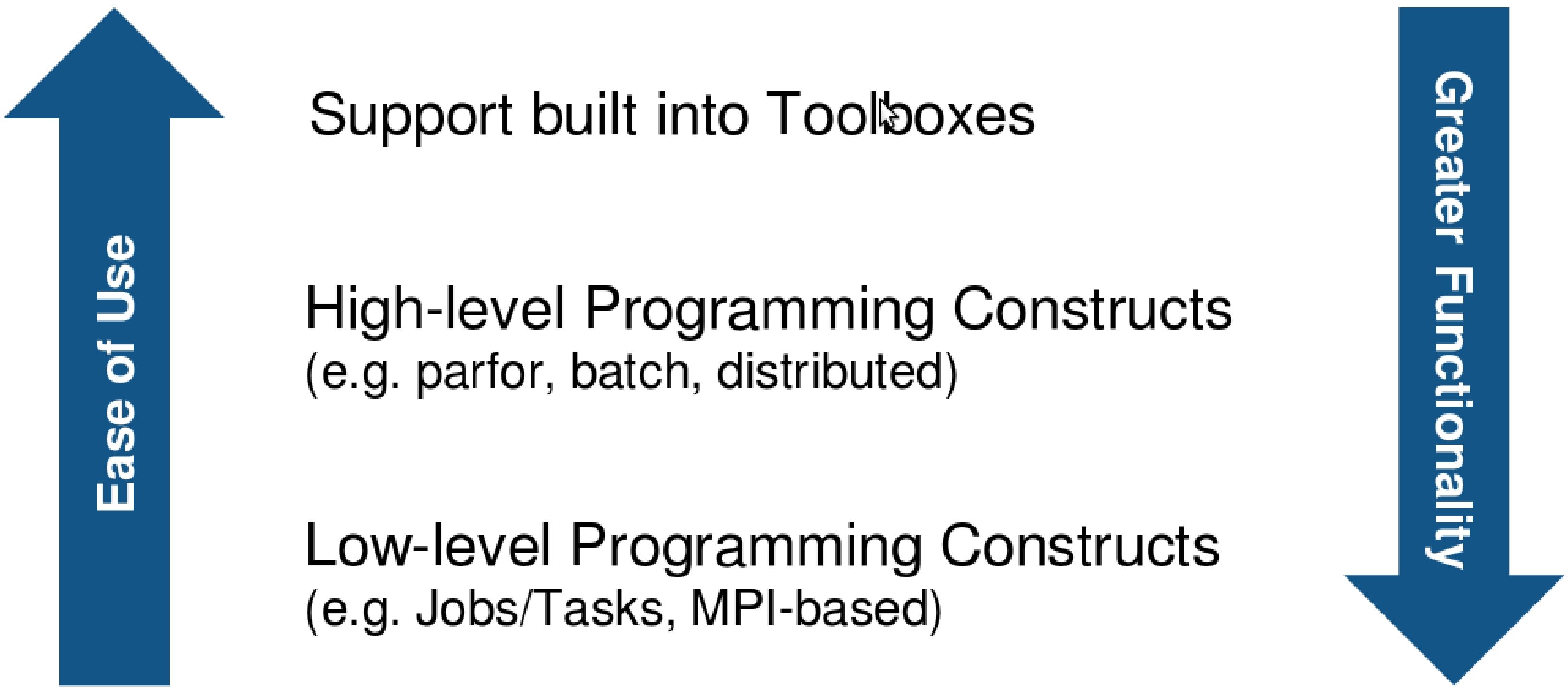
# Parallel Computing with Matlab



# Parallel Computing with Matlab

## Three levels of Integration:



# Parallel Computing Support in Toolboxes

- Optimization Toolbox
- Global Optimization Toolbox
- Statistics Toolbox
- Simulink Design Optimization
- Bioinformatics Toolbox
- Communications Toolbox
- Model-Based Calibration Toolbox
- ... and more

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

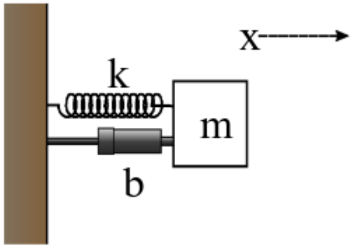
# Using MDCS on FLOW/HERO

- before you can use MDCS a few preparations are needed (only needed to be done once)
  - Matlab needs to be installed (see local web page) on your local machine, only versions R2010b, R2011a, R2011b are licensed for MDCS
  - your local machine must be able to login to FLOW/HERO via ssh
    - Linux/Mac have ssh per default, for Windows you can use PuTTY
    - if you are not in the university network you also need to connect to a VPN (see HPC-Wiki for details)
  - a number of files (from a zipped archive from the HPC-Wiki) have to copied to your local Matlab directory (depending on the setup of your local machine, your system admin has to help you)
  - a parallel configuration has to be setup with Matlab

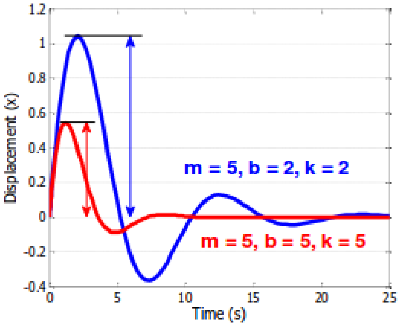
# Using MDCS on FLOW/HERO

- once you have completed the setup you can submit jobs to the cluster
  - example parameter sweep for 2nd-order ODE (taken from the HPC-Wiki)
  - dampened oscillator

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




  - simulate with different values for *b* and *k*
  - record peak value for each run



# 2nd-order ODE for example

odesystem.m

```
function dy = odesystem(t, y, m, b, k)
% 2nd-order ODE
%
%   m*X'' + b*X' + k*X = 0
%
% --> system of 1st-order ODEs
%
%   y  = X'
%   y' = -1/m * (k*y + b*y')
% Copyright 2009 The MathWorks, Inc.

dy(1) = y(2);
dy(2) = -1/m * (k * y(1) + b * y(2));

dy = dy(:); % convert to column vector
```

# Parameter Sweep: serial Matlab code

paramSweep_batch.m

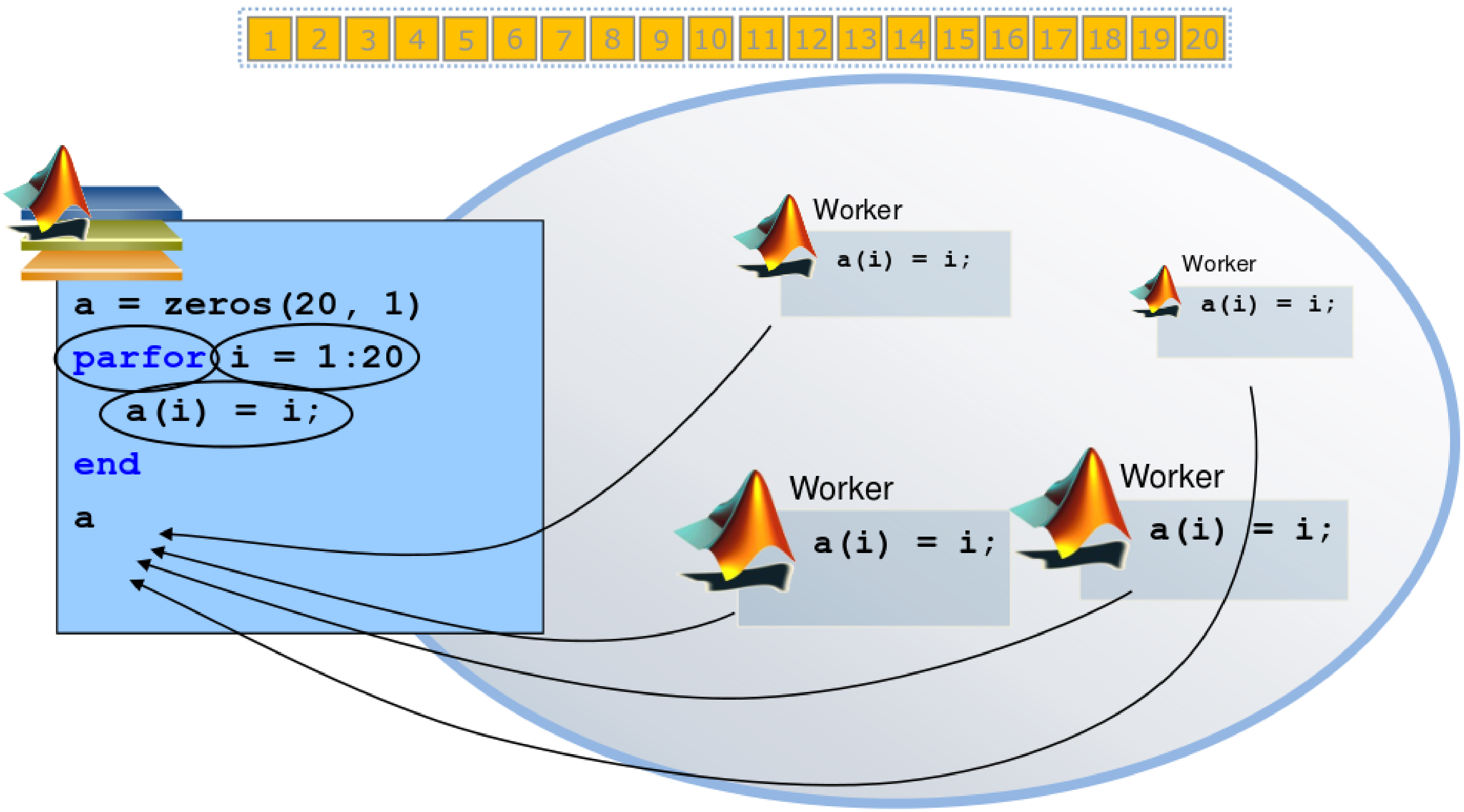
```
%% Initialize Problem
m      =         5;  % mass
bVals = 0.1:.1:15;  % damping values (step .1)
kVals = 1.5:.1:15;  % stiffness values (step .1) damping
[kGrid, bGrid] = meshgrid(bVals, kVals);
peakVals = nan(size(kGrid));

%% Parameter Sweep
tic;

for idx = 1:numel(kGrid)
  % Solve ODE
  [T,Y] = ode45(@(t,y) odesystem(t, y, m, bGrid(idx), kGrid(idx)), ...
    [0, 25], ...  % simulate for 25 seconds
    [0, 1]);      % initial conditions

  % Determine peak value
  peakVals(idx) = max(Y(:,1));
end

t1 = toc;
```

# Parameter Sweep: parallel Matlab code

paramSweep_batch.m

```
%% Initialize Problem
m     =         5;  % mass
bVals = 0.1:.1:15;  % damping values (step .1)
kVals = 1.5:.1:15;  % stiffness values (step .1) damping
[kGrid, bGrid] = meshgrid(bVals, kVals);
peakVals = nan(size(kGrid));

%% Parameter Sweep
tic;

parfor idx = 1:numel(kGrid)
  % Solve ODE
  [T,Y] = ode45(@(t,y) odesystem(t, y, m, bGrid(idx), kGrid(idx)), ...
    [0, 25], ...  % simulate for 25 seconds
    [0, 1]);      % initial conditions

  % Determine peak value
  peakVals(idx) = max(Y(:,1));
end

t1 = toc;
```

# 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`**)
  - cannot contain **`break`** or **`return`** statements
  - cannot contain another **`parfor`** loop

# Variable Classification

- all variables referenced at the top level of the parfor must be resolved and classified

| Classification | Description |
|---|---|
| Loop | serves as a loop index for arrays |
| sliced | an array whose segments are operated on by different iterations |
| broadcast | a variable defined before the loop whose value is used inside the loop, but never assigned in the loop |
| reduction | accumulates a value across iterations of the loop, regardless of iteration order |
| temporary | variable created inside the loop but unlike sliced or reduction variables, not available outside the loop |

# parfor Considerations

- parfor often only involves minimal code changes
- if a for loop cannot be converted to parfor, consider wrapping a subset of loop body in a function
  - e.g. load works not in parfor, however it does work in function that is called inside a parfor loop
- more information http://blogs.mathworks.com/loren/2009/10/02/using-parfor-loops-getting-up-and-running/
- there is a Code-Analyzer to diagnose parfor issues