

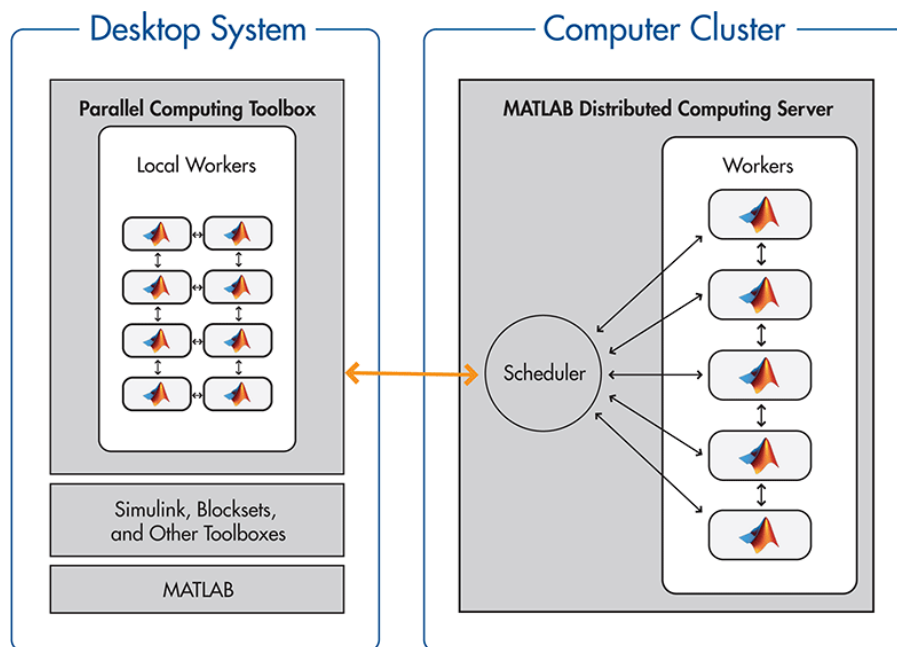
Parallel Computing Toolbox 4.3

Perform parallel computations on multicore computers and computer clusters

Parallel Computing Toolbox™ lets you solve computationally and data-intensive problems using MATLAB® and Simulink® on multicore and multiprocessor computers. Parallel processing constructs such as parallel for-loops, distributed arrays, parallel numerical algorithms, and message-passing functions let you implement task- and data-parallel algorithms in MATLAB at a high level without programming for specific hardware and network architectures. As a result, [converting serial MATLAB applications to parallel MATLAB applications](#) requires few code modifications and no programming in a low-level language. You can run your applications interactively or offline, in batch environments.

Key Features

- Support for data-parallel and task-parallel application development
- Ability to annotate code segments with `parfor` (parallel for-loops) and `spmd` (single program multiple data) for implementing task- and data-parallel algorithms
- High-level constructs such as distributed arrays, parallel algorithms, and message-passing functions for processing large data sets on multiple processors
- Ability to run eight workers locally on a multicore desktop
- Integration with MATLAB Distributed Computing Server for cluster- and grid-based applications that use any scheduler or any number of workers
- Interactive and batch execution modes



Developing parallel applications with Parallel Computing Toolbox. The toolbox enables application prototyping on the desktop with up to eight local workers (left), and, with MATLAB Distributed Computing Server (right), applications can be scaled to multiple computers on a cluster or a grid or cloud computing service.

You can use the toolbox to execute applications on a single multicore or multiprocessor desktop. Without changing the code, you can run the same application on a computer cluster or a grid or cloud computing service (using [MATLAB Distributed Computing Server™](#)). Parallel MATLAB applications can be distributed as executables or shared libraries (built using MATLAB Compiler™) that can access MATLAB Distributed Computing Server.

Programming Parallel Applications in MATLAB

Parallel Computing Toolbox provides several high-level programming constructs that let you convert your serial MATLAB code to run in parallel on several *workers* (MATLAB computational engines that run independently of MATLAB clients).

Constructs such as parallel for-loops (`parfor`) and distributed arrays simplify parallel code development by abstracting away the complexity of managing coordination and distribution of computations and data between a MATLAB client and workers, as well as between workers. These constructs function even in the absence of workers, letting you maintain a single version of your code for both serial and parallel execution.

You can run an application that uses up to eight workers on your multicore desktop. Using MATLAB Distributed Computing Server, you can run the same application without modification on large resources such as computer clusters or grid and cloud computing services.

Taking Advantage of Built-in Parallel Computing Support in MathWorks Products

Key functions in a growing number of products in the MATLAB and Simulink product families, such as Statistics Toolbox™ and Simulink Design Optimization™, provide [built-in support for parallel computing](#). In the presence of Parallel Computing Toolbox, these functions exploit parallel computing resources by distributing computations across available workers, enabling you to take advantage of parallel computing resources without additional programming effort.

Implementing Task-Parallel Algorithms

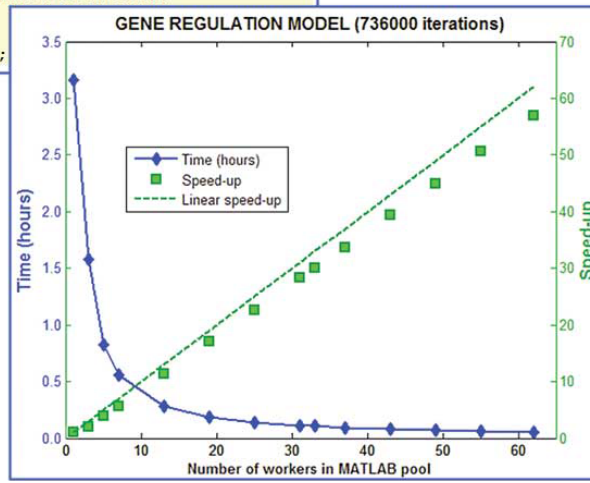
You can parallelize Monte Carlo simulations and other coarse-grained or embarrassingly parallel problems by organizing them into independent *tasks* (units of work). Parallel for-loops in the toolbox offer one way to distribute tasks across multiple MATLAB workers. Using these loops, you can automatically distribute independent loop iterations to multiple MATLAB workers. The `parfor` construct manages data and code transfer between the MATLAB client session and the workers. It automatically detects the presence of workers and reverts to serial behavior if none are present.

```

numberOfRuns = 20000;
sbioloadproject GeneRegulation m1
dataParfor = zeros(numberOfRuns, numel(m1.Species));

%% Run an ensemble of stochastic simulations
tic
parfor i = 1:numberOfRuns
    dataParfor(i, :) = runSimBioModel();
end
elapsedTimeParallel = toc;

```



Using parallel for-loops for a task-parallel application. You can use parallel for-loops in MATLAB scripts and functions and execute them both interactively and offline.

You can also explicitly program tasks either as MATLAB functions or as MATLAB scripts. You execute the tasks, when specified as functions, by manipulating task and job objects in the toolbox or, when specified as scripts, by using the batch function.

Implementing Data-Parallel Algorithms

For MATLAB algorithms that require large data set processing, Parallel Computing Toolbox provides distributed arrays and parallel functions that can operate on these arrays. Distributed arrays let you process significantly larger data sets than you could in a single MATLAB session by distributing data across multiple workers and using parallel functions to perform operations on the distributed data. The toolbox also provides functions for moving distributed data to and from MAT-files.

You can work with distributed arrays interactively from your MATLAB session as you would with MATLAB arrays. Working with distributed arrays and their associated parallel functions does not require knowledge of low-level programming tools, such as MPI. The toolbox provides more than 150 parallel functions for operating on distributed arrays, including linear algebra routines based on ScaLAPACK. You can use the familiar operators for arrays in MATLAB to perform indexing, matrix multiplication, and transforms directly on distributed arrays.

```

MATLAB 7.9.0 (R2009b)
File Edit Debug Parallel Desktop Window Help
Workspace
Select data to plot
Name Class
A distributed
S distributed
U distributed
V distributed
b distributed
x distributed
A_part double
e double
Command Window
>> % Start a pool of MATLAB workers
>> matlabpool
--Connected to a matlabpool session with 8 labs.
>>
>> % Create distributed arrays
>> A = distributed.randn(10000, 10000);
>> b = distributed.rand(10000, 1);
>>
>> % Solve Ax = b using Parallel "\"
>> x = A \ b;
>>
>> % Distributed arrays ops are automatically parallel
>> [U,S,V] = svd(A, 'econ');
>> e = norm(A * x - b);
>>
>> % Remote data can be brought back to client MATLAB
>> A_part = gather(A(1:100));
>>
>> % Some visualization functions also supported
>> plot(A(1:500)); % Don't plot all of really large A!
fx>>
Start OVR

```

Programming with distributed arrays. Distributed arrays and parallel algorithms let you create data-parallel MATLAB programs with minimal changes to your code and without programming in MPI.

The toolbox also provides the `spmd` (single program multiple data) construct, which you can use to designate sections of your code to run concurrently across workers participating in a parallel computation. During program execution, `spmd` automatically transfers data and code used within its body to the workers and, once the execution is complete, brings results back to the MATLAB client session.

For explicit, fine-grained control over your parallelization scheme, which also requires explicitly managing interworker coordination, Parallel Computing Toolbox functions provide access to message-passing routines based on the MPI standard (MPICH2), including functions for send, receive, broadcast, barrier, and probe operations.

Working in an Interactive Parallel Environment

Parallel Computing Toolbox extends the MATLAB interactive environment, letting you prototype and develop task- and data- parallel applications in your familiar environment. The `matlabpool` command allocates a set of dedicated computational resources for you, connecting your MATLAB session to a pool of MATLAB workers that can run either locally on your desktop or on a computer cluster. The command sets up an interactive parallel execution environment in which you can execute your parallel MATLAB code from the command prompt and retrieve results immediately as computations finish.

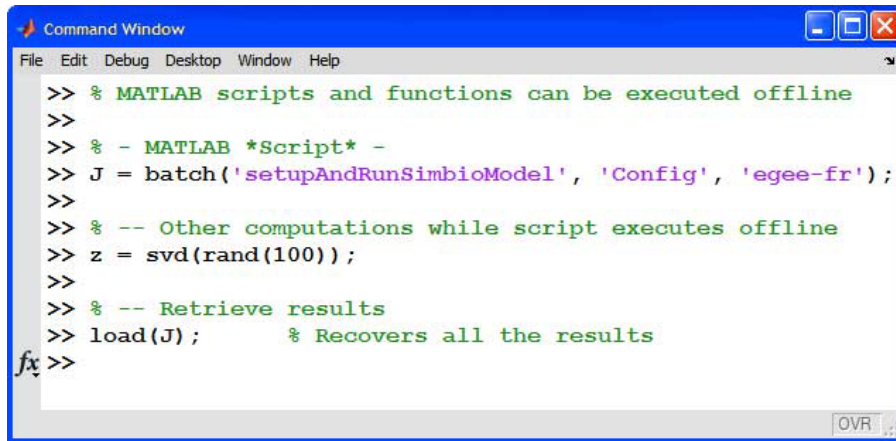
In this environment, parallel for-loops and distributed arrays automatically detect the presence of workers and distribute computations and data between your MATLAB session and the workers. Commands can be collected into MATLAB functions or scripts and submitted for offline execution.

Working in Batch Environments

Parallel Computing Toolbox lets you use batch environments for offline execution, enabling you to free your MATLAB client session for other activities while you execute large MATLAB and Simulink applications. You can also shut down your MATLAB client session while your application executes and retrieve results later.

The `batch` function lets you execute MATLAB scripts offline. It provides a mechanism for data transfer between MATLAB client and worker workspaces, which frees you from explicitly managing data in multiple workspaces.

The toolbox also provides job and task objects. These objects provide a lower-level but more general mechanism to execute parallel MATLAB applications in batch environments. Both the `batch` function and the job and task objects can be used to offload the execution of serial MATLAB programs from a desktop MATLAB session to a worker.



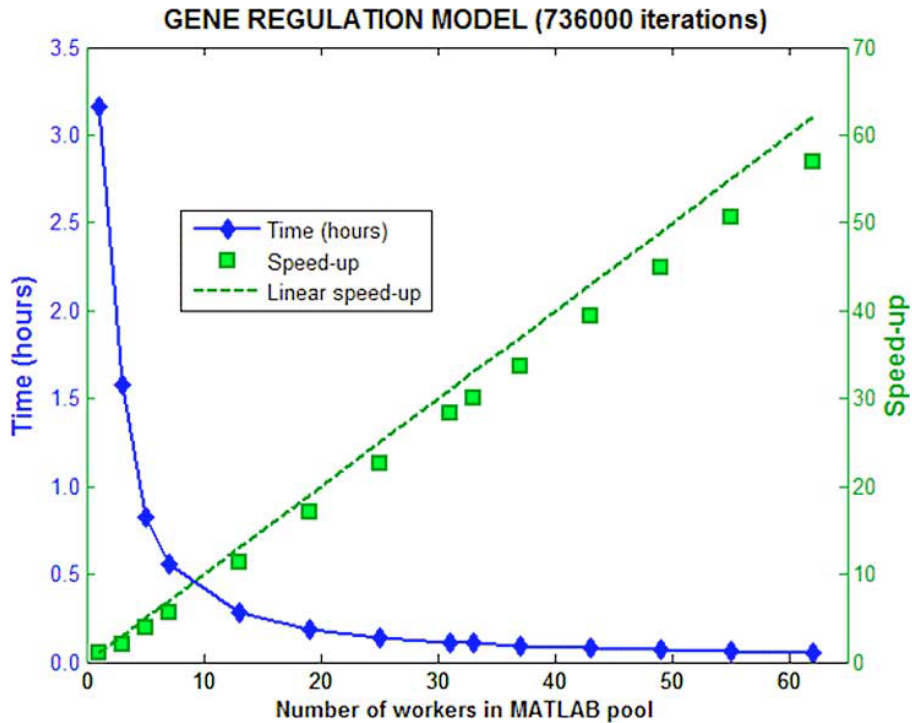
```
Command Window
File Edit Debug Desktop Window Help
>> % MATLAB scripts and functions can be executed offline
>>
>> % - MATLAB *Script* -
>> J = batch('setupAndRunSimbioModel', 'Config', 'egee-fr');
>>
>> % -- Other computations while script executes offline
>> z = svd(rand(100));
>>
>> % -- Retrieve results
>> load(J); % Recovers all the results
fx >>
```

Executing MATLAB scripts offline. With the Configurations Manager, you can direct program execution to your workstation using eight local workers or to a cluster using any scheduler and any number of workers.

Scaling Up to Clusters, Grids, and Clouds Using MATLAB Distributed Computing Server

Parallel Computing Toolbox provides the ability to use up to eight local workers on a multicore or multiprocessor computer using a single toolbox license. When used with MATLAB Distributed Computing Server, you can run an application using any number of workers on large-scale resources such as computer clusters or grid and cloud computing services. The server also supports both interactive and batch workflows.

Using MATLAB Compiler, you can build MATLAB applications that use toolbox functions into standalone executables or shared software components and distribute them royalty-free. These executables and libraries can connect to MATLAB Distributed Computing Server workers and perform MATLAB computations on large computing resources.



Running a gene regulation model on a cluster using MATLAB Distributed Computing Server. The server enables applications developed using Parallel Computing Toolbox to harness computer clusters for large problems. Compiled parallel MATLAB applications can also use the server.

Integrating Products into Your Computing Environment

MATLAB Distributed Computing Server provides a job manager and directly supports several third-party schedulers such as Platform LSF®, Microsoft® Windows® HPC Server, Altair PBS Pro®, and TORQUE. Using the Configurations Manager in the toolbox, you can maintain named settings such as scheduler type, path settings, and cluster usage policies. Switching between clusters or schedulers typically requires changing the configuration name only.

MATLAB Distributed Computing Server dynamically enables the required product licenses when an application runs on the cluster, based on the user's profile. As a result, administrators need to manage only the server license on the cluster rather than separate toolbox and blockset licenses for every cluster user.

Resources

Product Details, Demos, and System Requirements

www.mathworks.com/products/parallel-computing

Trial Software

www.mathworks.com/trialrequest

Sales

www.mathworks.com/contactsales

Technical Support

www.mathworks.com/support

Online User Community

www.mathworks.com/matlabcentral

Training Services

www.mathworks.com/training

Third-Party Products and Services

www.mathworks.com/connections

Worldwide Contacts

www.mathworks.com/contact

© 2010 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.