Parallel Computing Toolbox

Perform parallel computations on multicore computers, GPUs, and computer clusters

Parallel Computing Toolbox™ lets you solve computationally and data-intensive problems using multicore processors, GPUs, and computer clusters. High-level constructs—parallel for-loops, special array types, and parallelized numerical algorithms—let you parallelize MATLAB® applications without CUDA or MPI programming. You can use the toolbox with Simulink® to run multiple simulations of a model in parallel.

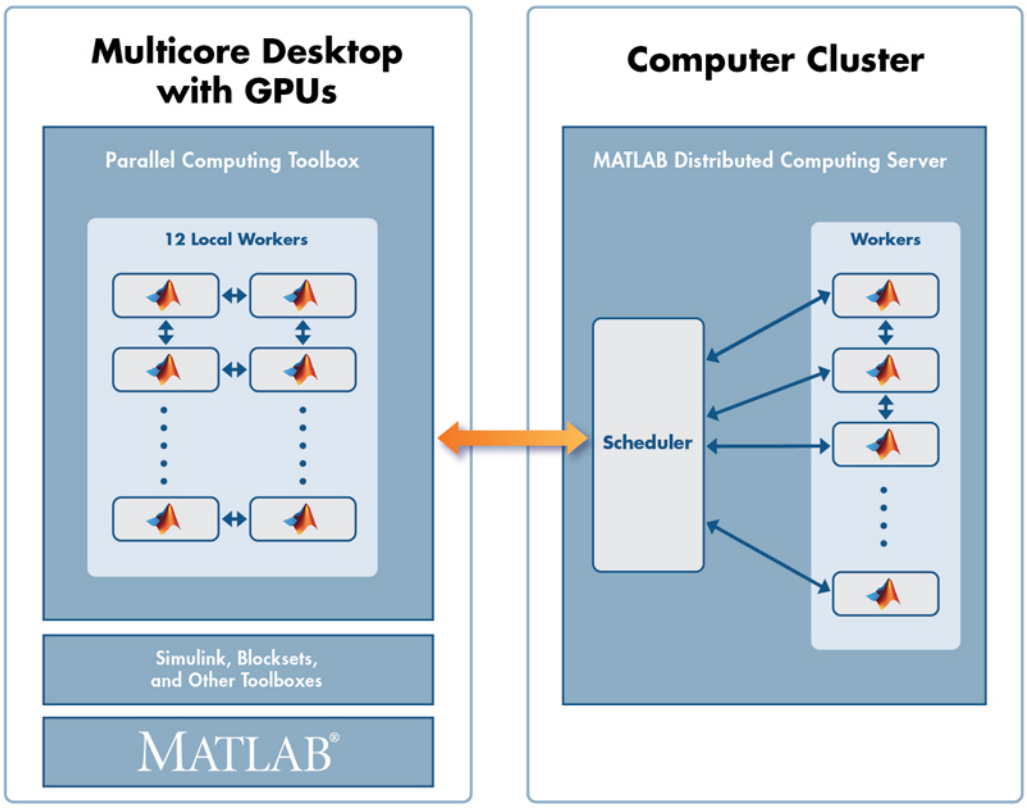
The toolbox lets you use the full processing power of multicore desktops by executing applications on workers (MATLAB computational engines) that run locally. Without changing the code, you can run the same applications on a computer cluster or a grid computing service (using MATLAB Distributed Computing Server™). You can run parallel applications interactively or in batch.

Built-in Parallel Computing Support in MathWorks Products

MATLAB Distributed Computing Server for Amazon EC2 - Early Adopter Program

Key Features

- Parallel `for`-loops (`parfor`) for running task-parallel algorithms on multiple processors
- Support for CUDA-enabled NVIDIA GPUs
- Full use of multicore processors on the desktop via workers that run locally
- Computer cluster and grid support (with MATLAB Distributed Computing Server)
- Interactive and batch execution of parallel applications
- Distributed arrays and single program multiple data (`spmd`) construct for large dataset handling and data-parallel algorithms



Parallel computing with MATLAB. You can use Parallel Computing Toolbox to run applications on a multicore desktop with local workers available in the toolbox, take advantage of GPUs, and scale up to a cluster (with MATLAB Distributed Computing Server).

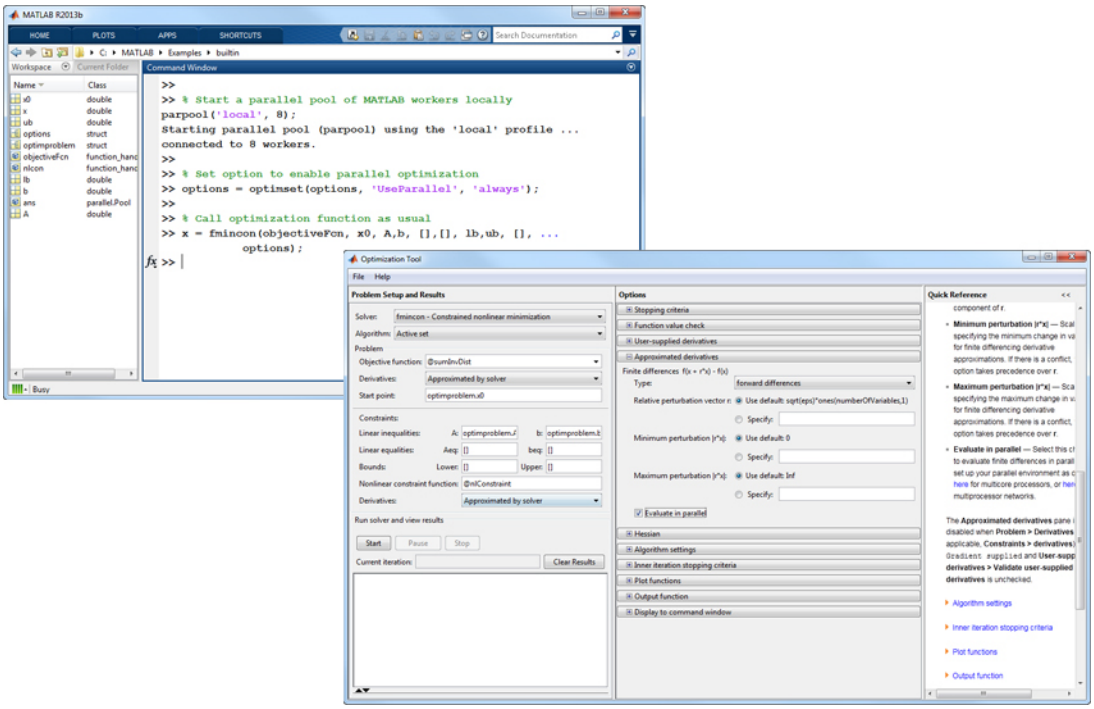
Programming Parallel Applications

Parallel Computing Toolbox provides several high-level programming constructs that let you convert your applications to take advantage of computers equipped with multicore processors and GPUs. Constructs such as parallel `for`-loops `(parfor)` and special array types for distributed processing and for GPU computing simplify parallel code development by abstracting away the complexity of managing computations and data between your MATLAB session and the computing resource you are using.

You can run the same application on a variety of computing resources without reprogramming it. The parallel constructs function in the same way, regardless of the resource on which your application runs—a multicore desktop (using the toolbox) or on a larger resource such as a computer cluster (using toolbox with MATLAB Distributed Computing Server).

Using Built-In Parallel Algorithms in Other MathWorks Products

Key functions in several MathWorks products have built-in parallel algorithms. In the presence of Parallel Computing Toolbox, these functions can distribute computations across available parallel computing resources, allowing you to speed up not just your MATLAB and Simulink based analysis or simulation tasks but also code generation for large Simulink models. You do not have to write any parallel code to take advantage of these functions.

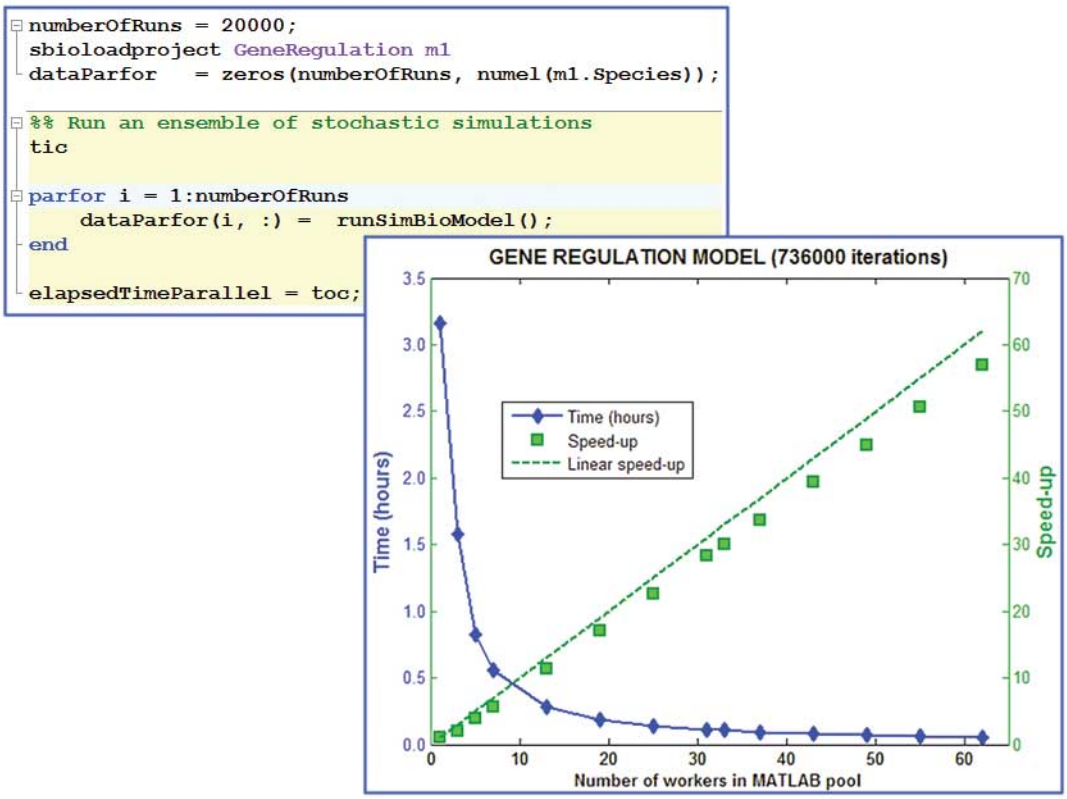


Using built-in parallel algorithms in MathWorks products. Built-in parallel algorithms can speed up MATLAB and Simulink computations as well as code generation from Simulink models.

Speeding Up Task-Parallel Applications

You can speed up some applications by organizing them into independent *tasks* (units of work) and executing multiple tasks concurrently. This class of task-parallel applications includes simulations for design optimization, BER testing, Monte Carlo simulations, and repetitive analysis on a large number of data files.

The toolbox offers `parfor`, a parallel `for`-loop construct that can automatically distribute independent tasks to multiple MATLAB *workers* (MATLAB computational engines running independently of your desktop MATLAB session). This construct automatically detects the presence of workers and reverts to serial behavior if none are present. You can also set up task execution using other methods, such as manipulating `task` objects in the toolbox.

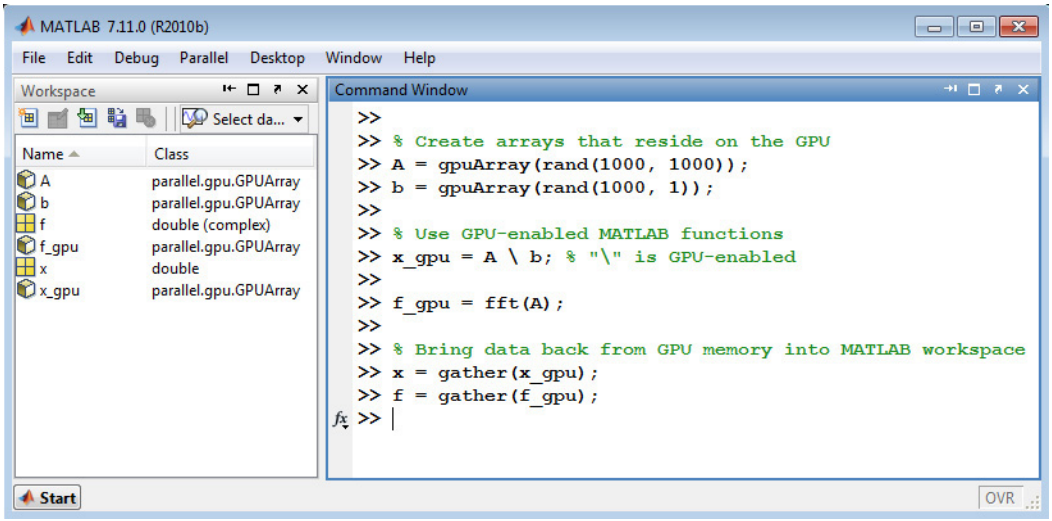


Using parallel `for`-loops for a task-parallel application. You can use parallel `for`-loops in MATLAB scripts and functions and execute them both interactively and offline.

Speeding Up MATLAB Computations with GPUs

Parallel Computing Toolbox provides GPUArray, a special array type with several associated functions that lets you perform computations on CUDA-enabled NVIDIA GPUs directly from MATLAB. Functions include `fft`, element-wise operations, and several linear algebra operations such as `lu` and `mldivide`, also known as the backslash operator (\). The toolbox also provides a mechanism that lets you use your existing CUDA-based GPU kernels directly from MATLAB.

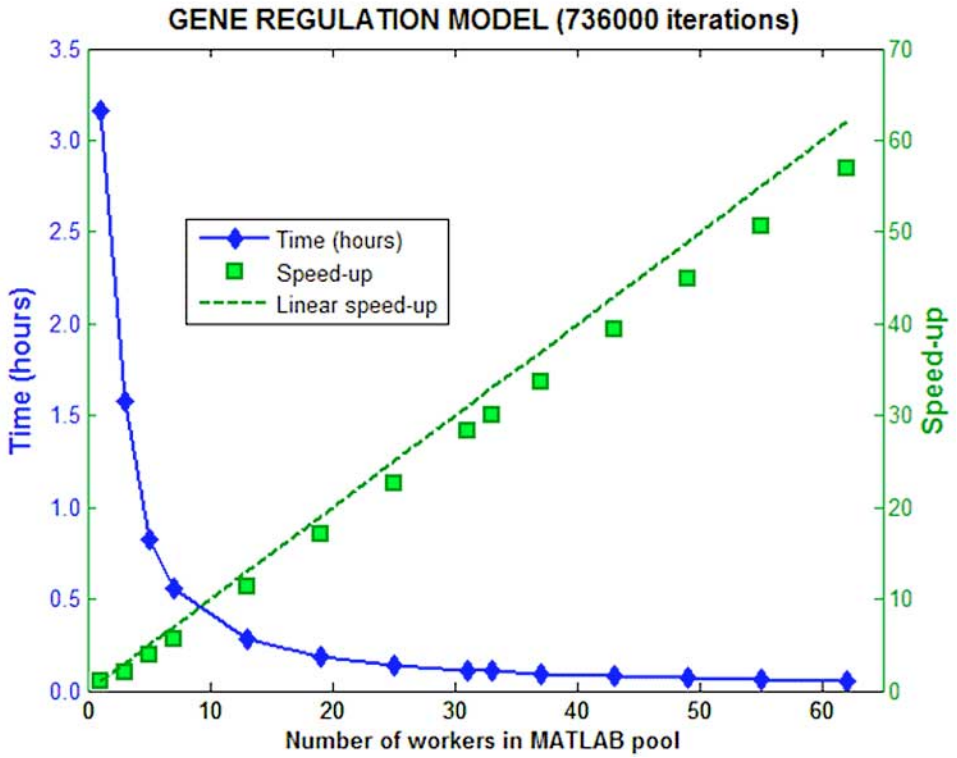
Learn more about GPU computing with MATLAB.



GPU computing with MATLAB. Using GPUArrays and GPU-enabled MATLAB functions help speed up MATLAB operations without low-level CUDA programming.

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

Parallel Computing Toolbox provides the ability to run MATLAB workers locally on your multicore desktop to execute your parallel applications allowing you to fully use the computational power of your desktop. Using the toolbox in conjunction with MATLAB Distributed Computing Server, you can run your applications on large scale computing resources such as computer clusters or grid and cloud computing resources



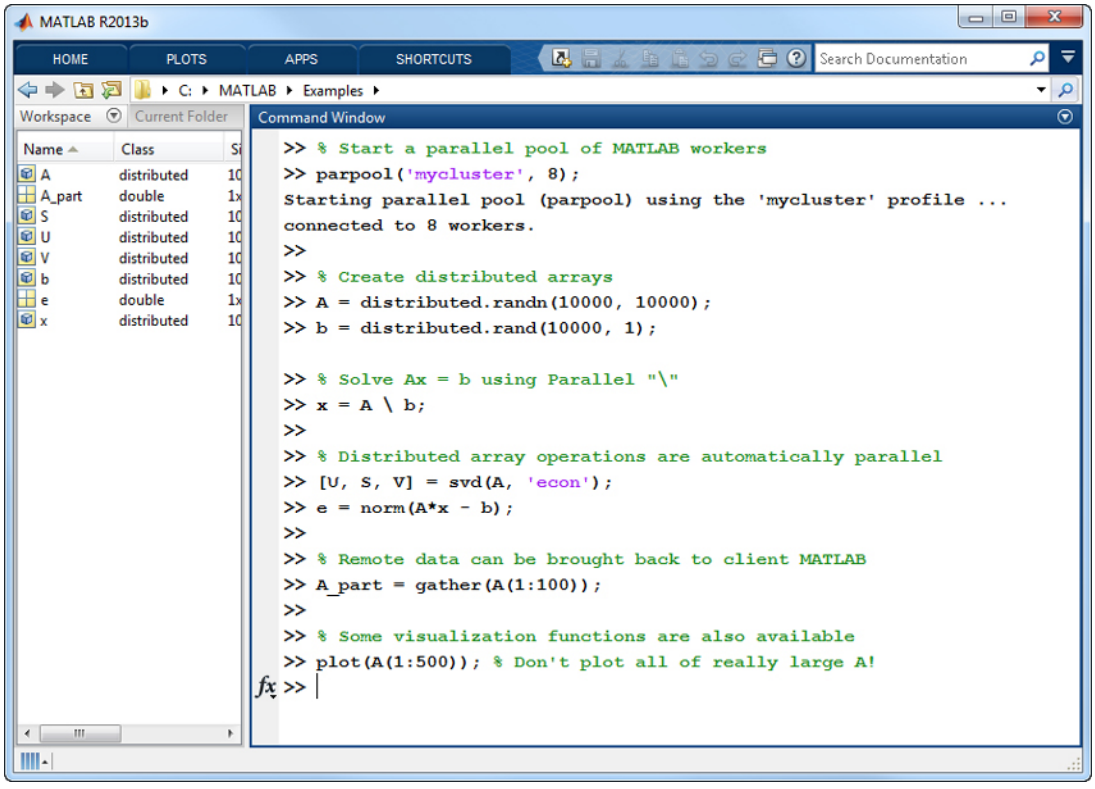
Running a gene regulation model on a cluster using MATLAB Distributed Computing Server. The server enables applications developed using Parallel Computing Toolbox to harness computer clusters for large problems.

Implementing Data-Parallel Applications using the Toolbox and MATLAB Distributed Computing Server

Distributed arrays in Parallel Computing Toolbox are special arrays that hold several times the amount of data that your desktop computer's memory (RAM) can hold. Distributed arrays apportion the data across several MATLAB worker processes running on a computer cluster (using MATLAB Distributed Computing Server). As a result, with distributed arrays you can overcome the memory limits of your desktop computer and solve problems that require manipulating very large matrices.

With over 150 functions available for working with distributed arrays, you can interact with these arrays as you would with MATLAB arrays and manipulate data available remotely on workers without low-level MPI programming. Available functions include linear algebra routines based on ScaLAPACK, such as `mldivide`, also known as the backslash operator (\), `lu` and `chol`, and functions for moving distributed data to and from MAT-files.

For fine-grained control over your parallelization scheme, the toolbox provides the single program multiple data (`spmd`) construct and several message-passing routines based on an MPI standard library (MPICH2). The `spmd` construct lets you 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. Message-passing functions for send, receive, broadcast, barrier, and probe operations are available.



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.

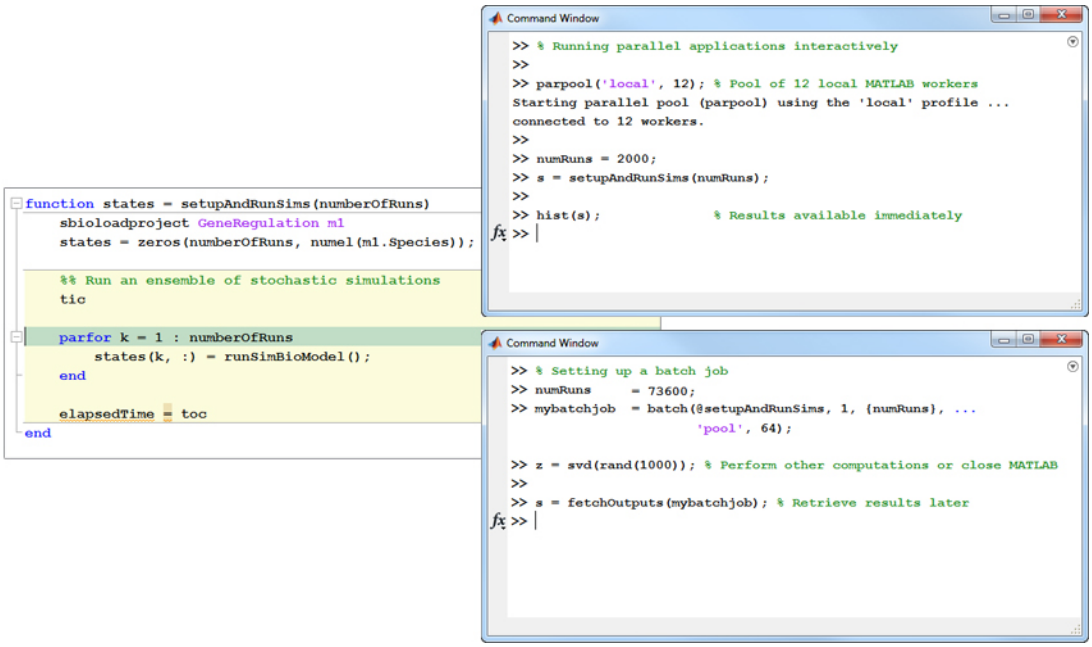
Running Parallel Applications Interactively and as Batch Jobs

You can execute parallel applications interactively and in batch using Parallel Computing Toolbox. Using the `parpool` command, you can connect your MATLAB session to a pool of MATLAB workers that can run either

locally on your desktop (using the toolbox) or on a computer cluster (using MATLAB Distributed Computing Server) to setup a dedicated interactive parallel execution environment. You can execute parallel applications from the MATLAB prompt on these workers and retrieve results immediately as computations finish, just as you would in any MATLAB session.

Running applications interactively is suitable when execution time is relatively short. When your applications need to run for a long time, you can use the toolbox to set them up to run as batch jobs. This enables you to free your MATLAB session for other activities while you execute large MATLAB and Simulink applications.

While your application executes in batch, you can shut down your MATLAB session and retrieve results later. The toolbox provides several mechanisms to manage offline execution of parallel programs, such as the `batch` function and `job` and `task` objects. Both the `batch` function and the `job` and `task` objects can be used to offload the execution of serial MATLAB and Simulink applications from a desktop MATLAB session.



Running parallel applications interactively and as batch jobs. You can run applications on your workstation using twelve workers available with the toolbox, or on a computer cluster using more workers available with MATLAB Distributed Computing Server.

Resources

Product Details, Examples, 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

© 2014 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.