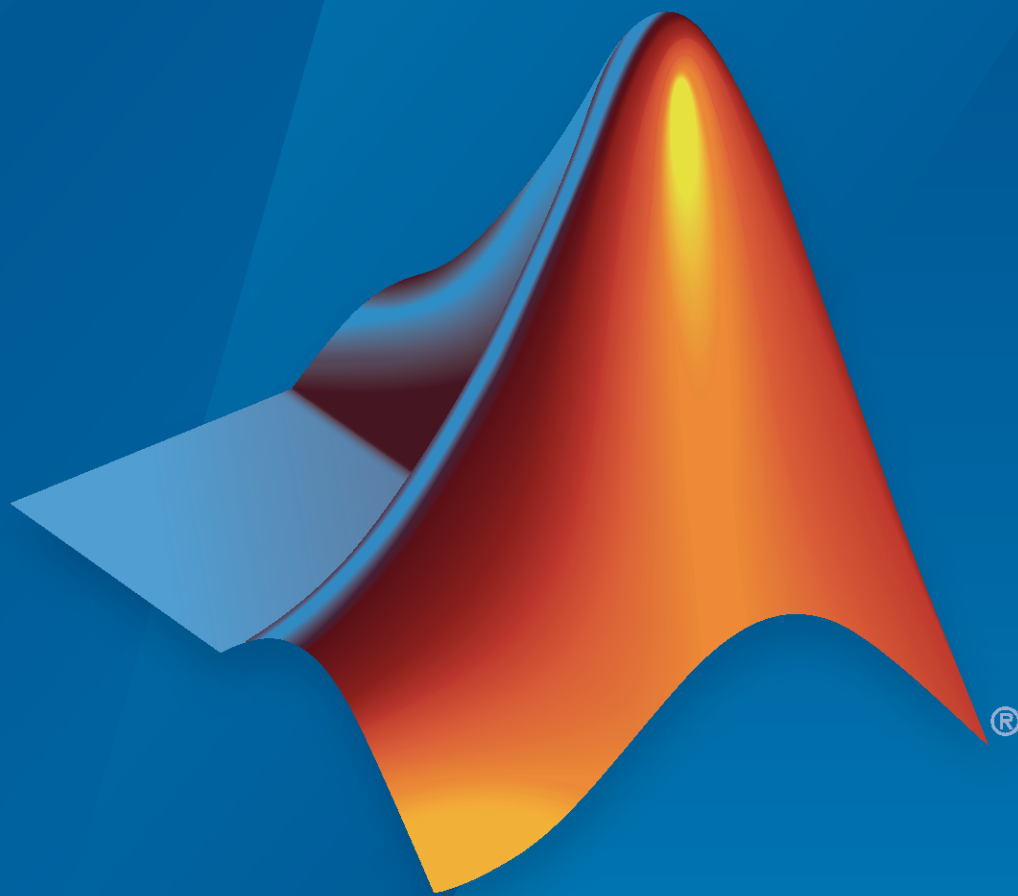


# Parallel Computing Toolbox™ Release Notes



# MATLAB®

## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



The MathWorks, Inc.  
1 Apple Hill Drive  
Natick, MA 01760-2098

### *Parallel Computing Toolbox™ Release Notes*

© COPYRIGHT 2006–2023 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### **Trademarks**

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://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.

### **Patents**

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

<b>PreferredPoolNumWorkers: Specify limited number of workers per cluster profile</b> .....	1-2
<b>Parallel Menu: Select a GPU using the Parallel menu</b> .....	1-2
<b>GPUDevice Object: Changes to device properties</b> .....	1-3
<b>GPU Functionality: Use functions with new and enhanced gpuArray support</b> .....	1-4
<b>PTX Build Option for mexcuda: Compile PTX files using mexcuda</b> .....	1-5
<b>Support for New GPU Architectures: Update to NVIDIA CUDA 11.8</b> .....	1-5
<b>Workflow Examples: New and updated examples and topics</b> .....	1-6
<b>Thread-Based Environment: Use new functionality on thread workers</b> .....	1-6
<b>Tall Arrays: Use functions with new and enhanced tall array support</b> ...	1-6
<b>Distributed and Tall tables and timetables: Perform calculations directly on distributed and tall tables and timetables without extracting their data</b> .....	1-6
<b>Composite arrays: Use gather to convert a Composite array into a local cell array.</b> .....	1-7
<b>Distributed Arrays: Use functions with new and enhanced distributed array support</b> .....	1-8
<b>Discover Clusters Supports Third-Party Schedulers</b> .....	1-8
<b>Third-Party Cluster Schedulers: Access simplified plugin scripts for integration on GitHub</b> .....	1-8
<b>Third-Party Cluster Schedulers: Open ports on workers to listen for connections from client</b> .....	1-9
<b>LDAP Support for MATLAB Job Scheduler Clusters: Authenticate cluster logins against LDAP credentials</b> .....	1-9

<b>Cluster Security: Restrict use of MATLAB Job Scheduler cluster commands</b> .....	<b>1-9</b>
<b>Parallel Computing in MATLAB Online Server Environment: Speed up MATLAB Online workflows</b> .....	<b>1-10</b>
<b>MATLAB Parallel Server in Kubernetes: Configure and run MATLAB Parallel Server workers on Kubernetes clusters</b> .....	<b>1-10</b>
<b>Functionality being removed or changed</b> .....	<b>1-10</b>
Preferred number of workers in a parallel pool option has been removed .....	<b>1-10</b>
rng("default") on MATLAB parallel workers now sets random number generator settings to worker default .....	<b>1-10</b>
parallel.pool.Constant objects no longer automatically transferred to workers .....	<b>1-10</b>
Thread-Based Parallel Pool: startup or matlabrc no longer runs when workers start .....	<b>1-10</b>
pmode has been removed .....	<b>1-11</b>
pload and psave have been removed .....	<b>1-11</b>

## R2022b

<b>Thread-Based Parallel Pool: Specify number of thread workers using parpool</b> .....	<b>2-2</b>
<b>Parallel menu: Select Threads using the Parallel menu</b> .....	<b>2-2</b>
<b>gpuDevice Object: Display values for byte-based memory properties with appropriate units</b> .....	<b>2-2</b>
<b>GPU Workflow Examples: New and updated examples and topics</b> .....	<b>2-3</b>
<b>Cloud GPU Computing: Access Cloud GPUs and Clusters in AWS Using Cloud Center</b> .....	<b>2-3</b>
<b>New Examples: Use ValueStore to monitor batch jobs</b> .....	<b>2-4</b>
<b>Parallel workflow examples: Use clusters for 5G Communications simulations</b> .....	<b>2-4</b>
<b>Data Analysis: Use mapreduce on Spark clusters</b> .....	<b>2-4</b>
<b>Parallel Computing page: Learn about MathWorks solutions that help you take advantage of more hardware resources</b> .....	<b>2-4</b>
<b>Multistep Authentication for Third-Party Schedulers: Connect to remote clients with multiple authentication options</b> .....	<b>2-4</b>
<b>Thread-Based Environment: Use new functionality on thread workers</b> .....	<b>2-5</b>

<b>GPU Functionality: Use functions with new and enhanced gpuArray support</b> .....	2-6
<b>Tall Arrays: Use functions with new and enhanced tall array support</b> ...	2-8
<b>Distributed Arrays: Use functions with new and enhanced distributed array support</b> .....	2-8
<b>MATLAB Job Scheduler: Support for your own Java installation</b> .....	2-9
<b>Element-Wise Operations: Improved performance on GPU</b> .....	2-9
<b>svd Function: Improved performance on GPU</b> .....	2-9
<b>Functionality being removed or changed</b> .....	2-10
labXxx functions are not recommended and are renamed to spmdXxx ...	2-10
gcat, gop, and gplus are not recommended and are renamed to spmdCat, spmdReduce, and spmdPlus .....	2-11
local profile has been renamed to Processes on the Parallel menu .....	2-11
parallel.defaultClusterProfile and parallel.clusterProfiles are renamed to parallel.defaultProfile and parallel.listProfiles .....	2-11
remotecopy has been removed .....	2-12
remotemjs has been removed .....	2-12

## R2022a

<b>ValueStore and FileStore objects: Retrieve data and files on MATLAB clients during job execution</b> .....	3-2
<b>Multifactor Authentication for the Generic Scheduler Interface: Connect to remote clients with RemoteClusterAccess using two or more authentication factors</b> .....	3-2
<b>Cluster resizing: Customize your MATLAB Job Scheduler cluster to resize automatically</b> .....	3-2
<b>Parallel Pools: Check status of pools</b> .....	3-2
<b>Background Pool: Number of thread workers no longer limited to 8 workers</b> .....	3-2
<b>Thread-Based Parallel Pool: See futures in active pool</b> .....	3-3
<b>cancelAll Method: Cancel currently queued and running futures in a parallel pool</b> .....	3-3
<b>GPU Functionality: Use new and enhanced gpuArray functions</b> .....	3-3
<b>Support for NVIDIA CUDA 11.2: Update to CUDA Toolkit 11.2</b> .....	3-4

<b>Tall Arrays: Use new and enhanced tall array functionality</b> .....	<b>3-4</b>
<b>Distributed Arrays: Use new and enhanced distributed array functionality</b> .....	<b>3-4</b>
<b>Parallel Features in Other Products</b> .....	<b>3-5</b>
<b>Solving Linear System: Improved performance when solving linear systems <math>A*X = B</math> with <code>gpuArray</code> for symmetric positive definite matrices <code>A</code></b> .....	<b>3-5</b>
<b>Functionality being removed or changed</b> .....	<b>3-6</b>
<code>distcomp</code> folder removed, now named <code>parallel</code> .....	<b>3-6</b>

## R2021b

<b>Parallel Language in MATLAB: Share parallel code with any MATLAB user</b> .....	<b>4-2</b>
<b>GPU Functionality: Use new and enhanced <code>gpuArray</code> functions</b> .....	<b>4-2</b>
<b>GPU Functionality: Use new and enhanced <code>gpuArray</code> functions in Statistics and Machine Learning Toolbox</b> .....	<b>4-2</b>
<b>GPU Functionality: Use new and enhanced <code>gpuArray</code> functions for working with signals and audio</b> .....	<b>4-3</b>
<b>Memory Usage: Use <code>whos</code> to check memory used by <code>gpuArray</code> and distributed variables</b> .....	<b>4-4</b>
<b>Distributed Arrays: Use new and enhanced distributed array functionality</b> .....	<b>4-4</b>
<b>Thread-Based Environment: Use new and enhanced functionality on threads for working with audio, video, and images</b> .....	<b>4-4</b>
<b>Functionality Being Removed or Changed</b> .....	<b>4-6</b>
<code>parfeval</code> and <code>parfevalOnAll</code> can now run in serial with no pool .....	<b>4-6</b>
<code>mldivide</code> and <code>decomposition</code> now produce the same results for distributed arrays .....	<b>4-6</b>
Default behavior for <code>decomposition</code> has changed for distributed arrays ...	<b>4-6</b>

## R2021a

<b>GPU Functionality: Use new and enhanced <code>gpuArray</code> functions</b> .....	<b>5-2</b>
--	------------

<b>GPU Functionality: Use new and enhanced gpuArray functions in Statistics and Machine Learning Toolbox</b> .....	5-2
<b>GPU Functionality: Use new and enhanced gpuArray functions for working with signals, audio, and wavelets</b> .....	5-2
<b>GPU Functionality: Use new and enhanced gpuArray functions in Image Processing Toolbox</b> .....	5-3
<b>Support for NVIDIA CUDA 11.0: Update to CUDA Toolkit 11.0</b> .....	5-3
<b>Distributed Arrays: Use new and enhanced distributed array functionality</b> .....	5-4
<b>GPU Devices: Count and compare available GPU devices</b> .....	5-4
<b>Thread-Based Parallel Pool: Use new and enhanced functionality on thread workers</b> .....	5-4
<b>parfor Examples: Use a parallel pool to speed up Monte-Carlo code</b> ....	5-4
<b>Upgrade Parallel Computing Products Together</b> .....	5-4
<b>Functionality Being Removed or Changed</b> .....	5-5
Support for cc3.0 and cc3.2 Kepler GPUs is removed .....	5-5
Functions offloaded with batch now evaluate cell array input arguments {C1,...,Cn} as C1,...,Cn .....	5-5

## R2020b

<b>GPU Functionality: Use new and enhanced gpuArray functions</b> .....	6-2
<b>GPU Functionality: Use new and enhanced gpuArray functions in Statistics and Machine Learning Toolbox</b> .....	6-2
<b>GPU Functionality: Use new and enhanced gpuArray functions for working with signals, audio, and wavelets</b> .....	6-2
<b>GPU Communications: Fast data transfer between GPUs in a parallel pool</b> .....	6-3
<b>Support for NVIDIA CUDA 10.2: Update to CUDA Toolkit 10.2</b> .....	6-4
<b>Distributed Arrays: Use new and enhanced distributed array functionality</b> .....	6-4
<b>Tall Arrays: Use new and enhanced tall array functionality in Signal Processing Toolbox</b> .....	6-4

<b>Array Assignment: Assign gpuArray or distributed data directly into existing MATLAB arrays</b> .....	<b>6-4</b>
<b>Distributed Tables: Assign data directly into existing distributed tables</b> .....	<b>6-5</b>
<b>Thread-Based Parallel Pool: Use a gpuArray on thread workers</b> .....	<b>6-5</b>
<b>parfeval Examples: Explore the state of futures and cancel them</b> .....	<b>6-5</b>
<b>HTCondor integration: Plugin script for HTCondor now available as an Add-On</b> .....	<b>6-5</b>
<b>Parallel Workflows: Compare performance of different parallel environments</b> .....	<b>6-5</b>
<b>Query Underlying Data: Query the underlying data type of classes</b> .....	<b>6-6</b>
<b>Query Parallel Functionality: Query if support for Parallel Computing Toolbox functionality is available</b> .....	<b>6-6</b>
<b>Improved Scalability: Use MATLAB Job Scheduler clusters with up to 4000 workers</b> .....	<b>6-6</b>
<b>Reference Architectures: Schedule jobs to run in AWS Batch</b> .....	<b>6-6</b>
<b>Docker Containers: Build a container image with a customized MATLAB installation</b> .....	<b>6-7</b>
<b>labSend, labReceive, labBroadcast and gop Functions: Improved performance of data transfer between GPUs in a parallel pool</b> .....	<b>6-7</b>
<b>Upgrade Parallel Computing Products Together</b> .....	<b>6-8</b>
<b>Functionality Being Removed or Changed</b> .....	<b>6-8</b>
GPU acceleration is not available on macOS .....	<b>6-8</b>
Forward compatibility for GPU devices is disabled by default .....	<b>6-8</b>
classUnderlying and isaUnderlying are not recommended .....	<b>6-8</b>

## R2020a

<b>Parallel profiling: Learn tips and techniques to profile parallel code with new documentation</b> .....	<b>7-2</b>
<b>GPU Functionality: Use new and enhanced gpuArray functions</b> .....	<b>7-2</b>
<b>GPU Functionality: Use new and enhanced gpuArray functions in Statistics and Machine Learning Toolbox</b> .....	<b>7-2</b>
<b>GPU Functionality: New gpuArray support for spectral functions</b> .....	<b>7-2</b>



<b>Job Arrays: Submit job arrays to third-party schedulers with the generic scheduler interface</b> .....	7-2
<b>Distributed Arrays: Use new and enhanced distributed array functionality</b> .....	7-3
<b>New Thread-Based Parallel Pool: Optimized for reduced memory usage, faster scheduling, and less data transfer, for a subset of MATLAB functions</b> .....	7-3
<b>MATLAB Online: Use Parallel Computing Toolbox with Cloud Center clusters in MATLAB Online</b> .....	7-4
<b>Upgrade Parallel Computing Products Together</b> .....	7-4
<b>Functionality Being Removed or Changed</b> .....	7-4
GPU acceleration is not available on macOS 10.14 Mojave and later versions .....	7-4
Support for Kepler and Maxwell GPUs will be removed .....	7-4
pmode will be removed .....	7-5
pload and psave will be removed .....	7-5

## R2019b

<b>GPU Functionality: Use new and enhanced gpuArray functions</b> .....	8-2
<b>Support for NVIDIA CUDA 10.1: Update to CUDA Toolkit 10.1</b> .....	8-2
<b>gpuArray: Load gpuArray data when no GPU is available</b> .....	8-2
<b>mexcuda: Compile mexcuda functions without installing the CUDA toolkit</b> .....	8-2
<b>Distributed Arrays: Use new and enhanced distributed array functionality</b> .....	8-2
<b>Distributed Arrays Examples: Explore multigrid preconditioned iterative solvers with new documentation</b> .....	8-2
<b>Parallel Deep Learning Workflows: Explore deep learning with custom parallel training loops</b> .....	8-3
<b>Batch Jobs Examples: Explore batch workflows with new documentation</b> .....	8-3
<b>parfeval Examples: Explore parfeval workflows with new documentation</b> .....	8-3
<b>Image Acquisition Examples: Perform parallel acquisition and processing of images</b> .....	8-3

<b>Improved Scalability: Use MATLAB Job Scheduler clusters with up to 2000 workers</b> .....	<b>8-3</b>
<b>Third-Party Schedulers: Check the scheduler ID of a MATLAB task</b> .....	<b>8-3</b>
<b>Common Job Schedulers: Improved performance of vectorized task creation</b> .....	<b>8-3</b>
<b>Personalized Clusters: Set Cloud Center clusters for personal use</b> .....	<b>8-4</b>
<b>Upgrade Parallel Computing Products Together</b> .....	<b>8-4</b>
<b>Functionality Being Removed or Changed</b> .....	<b>8-5</b>
Toolbox folder renamed from distcomp to parallel .....	<b>8-5</b>
pmode will be removed .....	<b>8-5</b>
pload and psave will be removed .....	<b>8-5</b>

## R2019a

<b>parfor-Loops: Use parfor without a parallel pool</b> .....	<b>9-2</b>
<b>Parallel Deep Learning Workflows: Explore deep learning with multiple-GPUs</b> .....	<b>9-2</b>
<b>Benchmarks: Use new examples to evaluate the performance of your cluster</b> .....	<b>9-2</b>
<b>parpool Resiliency: Parallel pools are now resilient to dropped network connections</b> .....	<b>9-2</b>
<b>Automatic Cluster Resizing: Resize Cloud Center clusters on Amazon based on usage</b> .....	<b>9-2</b>
<b>Custom Datastore: Read from Hadoop based databases using the custom datastore framework</b> .....	<b>9-2</b>
<b>GPU Functionality: Use new and enhanced gpuArray functions</b> .....	<b>9-3</b>
<b>Support for NVIDIA CUDA 10.0: Update to CUDA Toolkit 10.0</b> .....	<b>9-3</b>
<b>Distributed Arrays: Use new and enhanced distributed array functionality</b> .....	<b>9-3</b>
<b>Renamed Product: MATLAB Distributed Computing Server renamed to MATLAB Parallel Server</b> .....	<b>9-3</b>
<b>Support for MPICH: Update to MPICH Version 3 on Linux for third-party schedulers</b> .....	<b>9-3</b>
<b>Upgrade Parallel Computing Products Together</b> .....	<b>9-4</b>

<b>Functionality Being Removed or Changed</b> .....	<b>9-4</b>
Default random number generator changed for parallel contexts .....	<b>9-4</b>
Toolbox folder renamed from distcomp to parallel .....	<b>9-5</b>

## R2018b

<b>GPU Support: View details of GPU support on over 600 function pages, and browse GPU support for functions by toolbox</b> .....	<b>10-2</b>
<b>GPU Functionality: Use new and enhanced gpuArray functions, such as spline interpolation and vecnorm</b> .....	<b>10-2</b>
<b>Support for NVIDIA CUDA 9.1: Update to CUDA Toolkit 9.1</b> .....	<b>10-2</b>
<b>GPU Optimizations: Enhanced support and optimizations for GPU</b> ....	<b>10-2</b>
<b>GPU Examples: Explore GPU workflows on single and multiple GPUs</b> .....	<b>10-2</b>
<b>Distributed Arrays: Use new and enhanced distributed array functionality, including support for vecnorm and writing to Amazon S3 and Azure</b> .....	<b>10-3</b>
<b>Distributed Support: View details of distributed array support on over 400 function pages, and browse distributed support for functions by toolbox</b> .....	<b>10-3</b>
<b>Parallel Extended Capabilities: View details of automatic parallel support on function pages, and browse support for functions by toolbox</b> ....	<b>10-3</b>
<b>Write Cloud Data: Write distributed arrays and tall arrays to Amazon S3 and Windows Azure storage</b> .....	<b>10-4</b>
<b>Big Data in the Cloud: Explore MATLAB capabilities for big data</b> .....	<b>10-4</b>
<b>Improved Scalability: Use up to 1024 workers per parallel pool</b> .....	<b>10-4</b>
<b>Streamlined Cloud Cluster Setup: Create new Cloud Center clusters directly from the MATLAB desktop</b> .....	<b>10-4</b>
<b>Cluster Workflows: Learn how to scale up from desktop to cluster</b> ....	<b>10-4</b>
<b>Streamlined Installation Documentation: Simplified workflows for integrating MATLAB Distributed Computing Server in your cluster</b> .....	<b>10-4</b>
<b>Generic Scheduler Documentation: Set up Schedulers Using Improved Instructions</b> .....	<b>10-4</b>

<b>Parallel Deep Learning Workflows: Explore deep learning with multiple GPUs locally or in the cloud</b> .....	<b>10-5</b>
<b>Upgrade Parallel Computing Products Together</b> .....	<b>10-5</b>
<b>Functionality Being Removed or Changed</b> .....	<b>10-5</b>
Philox4x32-10 and Threefry4x64-20 random number generators renamed .....	<b>10-5</b>
Default random number generator will change for parallel contexts .....	<b>10-6</b>
Support for MATLAB Parallel Cloud is removed .....	<b>10-6</b>

## R2018a

<b>Parfeval callbacks: New afterAll and afterEach methods for parallel futures</b> .....	<b>11-2</b>
<b>Slurm support: Slurm is a fully supported scheduler</b> .....	<b>11-2</b>
<b>Support for NVIDIA Volta: Update to CUDA 9, support for Volta class GPUs</b> .....	<b>11-2</b>
<b>Improved file mirroring: Performance of file mirroring for generic scheduler integration</b> .....	<b>11-2</b>
<b>Parfor performance improvements: More efficient broadcast variables in parfor for non-local clusters</b> .....	<b>11-2</b>
<b>GPU Array Support: Use enhanced gpuArray functions</b> .....	<b>11-2</b>
<b>Distributed Array Support: Use enhanced distributed array functions</b> .....	<b>11-2</b>
<b>Parameter Sweep Example: Use a DataQueue to monitor results during computations on a parallel pool</b> .....	<b>11-3</b>
<b>Discontinued Support for GPU Devices of Compute Capability less than 3.0</b> .....	<b>11-3</b>
<b>Backwards Compatibility for MATLAB Job Scheduler: Upgrade MATLAB Job Scheduler clusters, and continue to use previous releases of Parallel Computing Toolbox</b> .....	<b>11-3</b>
<b>Upgrade Parallel Computing Products Together</b> .....	<b>11-3</b>
<b>Functionality Being Removed or Changed</b> .....	<b>11-4</b>

<b>Improved Parallel Language Performance: Execute parallel language constructs with reduced overhead</b> .....	12-2
<b>Tall Array Support: Use tall arrays with Windows client access to Linux Spark clusters</b> .....	12-2
<b>Improved Parallel Pool Robustness: Run pools without Message Passing Interface (MPI) by default, making pools resilient to workers crashing</b> .....	12-2
<b>Improved MATLAB Integration with Third-Party Schedulers: Use the Generic Profile Wizard for easier installation and setup of MATLAB Distributed Computing Server</b> .....	12-2
<b>Cloud Storage: Work with data in Microsoft Azure Blob Storage</b> .....	12-2
<b>Copy Client Environment to Workers on Any Cluster: Specify which environment variables on your client machine your workers should automatically inherit</b> .....	12-2
<b>Client Path Sharing: Add user-added-entries on the client's path to the workers' paths</b> .....	12-3
<b>GPU Array Support: Use enhanced gpuArray functions</b> .....	12-3
<b>Distributed Array Support: Use enhanced distributed array functions</b> .....	12-3
<b>Enhanced Support for Microsoft Windows HPC Pack</b> .....	12-3
<b>Discontinued Support for GPU Devices of Compute Capability less than 3.0</b> .....	12-3
<b>Backwards Compatibility for MATLAB Job Scheduler: Upgrade MATLAB Job Scheduler clusters, and continue to use previous releases of Parallel Computing Toolbox</b> .....	12-4
<b>Upgrade Parallel Computing Products Together</b> .....	12-4
<b>Functionality Being Removed or Changed</b> .....	12-4

<b>Access to Intermediate Results and Updates in Parallel Computations: Poll for messages or data from different workers during parallel workflows</b> .....	13-2
--	------

<b>Tall Array Support: Use parallel execution environments for tall timetables and enhanced tall array functions</b> .....	13-2
<b>More Responsive Job Monitor: Automatic updates for new, submitted, or deleted jobs or tasks</b> .....	13-2
<b>Re-create Jobs: Easily rerun all failed or cancelled tasks</b> .....	13-3
<b>GPU Array Support: Use enhanced gpuArray functions</b> .....	13-3
<b>Distributed Array Support: Use enhanced distributed array functions</b> .....	13-3
<b>Simplified Integration for Third-Party Cluster Schedulers: Updates to generic scheduler integration allow folder-based configuration and eliminate the need to specify function handles</b> .....	13-3
<b>MATLAB String Support: Functions in Parallel Computing Toolbox accept MATLAB strings as input</b> .....	13-3
<b>Support for Spark 2.x enabled Hadoop clusters</b> .....	13-4
<b>Upgraded CUDA Toolkit version</b> .....	13-4
<b>Discontinued Support for GPU Devices of Compute Capability less than 3.0</b> .....	13-4
<b>Backwards Compatibility for MATLAB Job Scheduler: Upgrade MATLAB Job Scheduler clusters, and continue to use previous releases of Parallel Computing Toolbox</b> .....	13-4
<b>Upgrade Parallel Computing Products Together</b> .....	13-4
<b>Properties of generic cluster objects have changed</b> .....	13-5
<b>Functionality Being Removed or Changed</b> .....	13-5

## R2016b

<b>Parallel Support for Tall Arrays: Process big data with tall arrays in parallel on your desktop, MATLAB Distributed Computing Server, and Spark clusters</b> .....	14-2
<b>Support for GPU Arrays: Use enhanced gpuArray functions, including new sparse iterative solver bicg</b> .....	14-2
<b>Parallel Menu Enhancement: Use the new menu items in the Parallel Menu to configure and manage cloud based resources</b> .....	14-2

<b>New Data Types in Distributed Arrays: Use enhanced functions for creating distributed arrays of: datetime; duration; calendarDuration; string; categorical; and table</b> .....	14-2
<b>Loading Distributed Arrays: Load distributed arrays in parallel using datastore</b> .....	14-3
<b>Cluster Profile Validation: Choose which validation stages run and the number of MATLAB workers to use</b> .....	14-3
<b>Backwards Compatibility: Upgrade MATLAB Job Scheduler clusters, and continue to use the previous release of Parallel Computing Toolbox</b> .....	14-4
<b>datetime Support for Timestamps: Use built-in datetime objects in MATLAB to access timestamp information for jobs and tasks</b> .....	14-4
<b>Data Transfer Measurement: Use ticBytes and tocBytes to measure the data transfer between MATLAB workers in a parallel pool</b> .....	14-4
<b>Multithreaded Workers: Use multiple computational threads on your MATLAB workers</b> .....	14-4
<b>Support for Distributed Arrays: Use enhanced distributed array functions, including sparse input to iterative solvers (gmres and lsqr)</b> .....	14-5
<b>Increased Data Transfer Limits: Send messages larger than 2 GB using labSend, labSendReceive and labBroadcast</b> .....	14-5
<b>Upgrade Parallel Computing Products Together</b> .....	14-5
<b>Functionality Being Removed or Changed</b> .....	14-5

## R2016a

<b>GPU Support for Sparse Matrices: Use enhanced gpuArray functions for sparse matrices on GPUs</b> .....	15-2
<b>Support for Distributed Arrays: Use enhanced distributed array functions including sparse input to direct (mldivide) and iterative solvers (cgs and pcg)</b> .....	15-2
<b>GPU-Accelerated Deep Learning: Use Neural Network Toolbox to train deep convolutional neural networks with GPU-enabled acceleration for image classification tasks</b> .....	15-2
<b>GPU-enabled MATLAB Functions: Accelerate applications using GPU-enabled MATLAB functions for linear equations, descriptive statistics and set operations</b> .....	15-2

<b>Upgraded CUDA Toolkit version</b> .....	15-3
<b>Parallel-Enabled Gradient Estimation: Accelerate more nonlinear solvers in the Optimization Toolbox with parallel finite difference estimation of gradients and Jacobians</b> .....	15-3
<b>Hadoop Kerberos Support: Improved support for Hadoop in a Kerberos authenticated environment</b> .....	15-3
<b>Transfer unlimited data between client and workers, and attached files up to 4GB in total, in any job using a MATLAB Job Scheduler cluster</b> ..	15-3
<b>matlabpool function removed</b> .....	15-3
<b>Upgrade parallel computing products together</b> .....	15-4

## R2015b

<b>Discontinued support for parallel computing products on 32-bit Windows operating systems</b> .....	16-2
<b>More than 90 GPU-enabled functions in Statistics and Machine Learning Toolbox, including probability distribution, descriptive statistics, and hypothesis testing</b> .....	16-2
<b>Additional GPU-enabled MATLAB functions, including support for sparse matrices</b> .....	16-2
Additional GPU-enabled MATLAB functions .....	16-2
Sparse arrays with GPU-enabled functions .....	16-2
<b>mexcuda function for easier compilation of MEX-files containing CUDA code</b> .....	16-3
<b>Upgraded CUDA Toolkit version</b> .....	16-3
<b>Scheduler integration scripts for SLURM</b> .....	16-3
<b>parallel.pool.Constant function to create constant data on parallel pool workers, accessible within parallel language constructs such as parfor and parfeval</b> .....	16-3
<b>Improved performance of mapreduce on Hadoop 2 clusters</b> .....	16-4
<b>Enhanced and additional MATLAB functions for distributed arrays</b> ...	16-4
<b>Warnings property for tasks</b> .....	16-4
<b>More consistent transparency enforcement</b> .....	16-4
<b>Upgrade parallel computing products together</b> .....	16-4



<b>Support for mapreduce function on any cluster that supports parallel pools</b> .....	17-2
<b>Sparse arrays with GPU-enabled functions</b> .....	17-2
<b>Additional GPU-enabled MATLAB functions</b> .....	17-2
<b>pagefun support for mrdivide and inv functions on GPUs</b> .....	17-3
<b>Enhancements to GPU-enabled linear algebra functions</b> .....	17-3
<b>Parallel data reads from a datastore with MATLAB partition function</b> .....	17-3
<b>Using DNS for cluster discovery</b> .....	17-3
<b>MS-MPI support for local and MATLAB Job Scheduler clusters</b> .....	17-3
<b>Ports and sockets in mdce_def file</b> .....	17-3
<b>Improved profiler accuracy for GPU code</b> .....	17-4
<b>Upgraded CUDA Toolkit version</b> .....	17-4
<b>Discontinued support for GPU devices on 32-bit Windows computers</b> .....	17-4
<b>Discontinued support for parallel computing products on 32-bit Windows computers</b> .....	17-4
<b>matlabpool function removed</b> .....	17-4

<b>Parallelization of mapreduce on local workers</b> .....	18-2
<b>Additional GPU-enabled MATLAB functions, including accumarray, histc, cummax, and cummin</b> .....	18-2
<b>pagefun support for mldivide on GPUs</b> .....	18-2
<b>Additional MATLAB functions for distributed arrays, including fft2, fftn, ifft2, ifftn, cummax, cummin, and diff</b> .....	18-2
<b>Data Analysis on Hadoop clusters using mapreduce</b> .....	18-3

<b>Discover Clusters Supports Microsoft Windows HPC Server for Multiple Releases</b> .....	<b>18-3</b>
<b>Upgraded CUDA Toolkit Version</b> .....	<b>18-3</b>
<b>Discontinued Support for GPU Devices of Compute Capability 1.3</b> .....	<b>18-3</b>
<b>Discontinued Support for GPU Devices on 32-Bit Windows Computers</b> .....	<b>18-4</b>

## R2014a

<b>Number of local workers no longer limited to 12</b> .....	<b>19-2</b>
<b>Additional GPU-enabled MATLAB functions: interp3, interpn, besselj, bessely</b> .....	<b>19-2</b>
<b>Additional GPU enabled Image Processing Toolbox functions: bwdist, imreconstruct, iradon, radon</b> .....	<b>19-2</b>
<b>Enhancements to GPU-enabled MATLAB functions: filter (IIR filters); pagefun (additional functions supported); interp1, interp2, conv2, reshape (performance improvements)</b> .....	<b>19-2</b>
<b>Duplication of an existing job, containing some or all of its tasks</b> .....	<b>19-2</b>
<b>More MATLAB functions enhanced for distributed arrays</b> .....	<b>19-3</b>
<b>GPU MEX Support for Updated MEX</b> .....	<b>19-3</b>
<b>Old Programming Interface Removed</b> .....	<b>19-3</b>
<b>matlabpool Function Being Removed</b> .....	<b>19-4</b>
<b>Removed Support for parallel.cluster.Mpiexec</b> .....	<b>19-4</b>

## R2013b

<b>parpool: New command-line interface (replaces matlabpool), desktop indicator, and preferences for easier interaction with a parallel pool of MATLAB workers</b> .....	<b>20-2</b>
Parallel Pool .....	<b>20-2</b>
New Desktop Pool Indicator .....	<b>20-3</b>
New Parallel Preferences .....	<b>20-3</b>

<b>Automatic start of a parallel pool when executing code that uses parfor or spmd</b> .....	<b>20-4</b>
<b>Option to start a parallel pool without using MPI</b> .....	<b>20-4</b>
<b>More GPU-enabled MATLAB functions (e.g., interp2, pagefun) and Image Processing Toolbox functions (e.g., bwmorph, edge, imresize, and medfilt2)</b> .....	<b>20-4</b>
<b>More MATLAB functions enabled for distributed arrays: permute, ipermute, and sortrows</b> .....	<b>20-5</b>
<b>Enhancements to MATLAB functions enabled for GPUs, including ones, zeros</b> .....	<b>20-5</b>
<b>gputimeit Function to Time GPU Computations</b> .....	<b>20-6</b>
<b>New GPU Random Number Generator NormalTransform Option: Box-Muller</b> .....	<b>20-6</b>
<b>Upgraded MPICH2 Version</b> .....	<b>20-6</b>
<b>Discontinued Support for GPU Devices of Compute Capability 1.3</b> ....	<b>20-7</b>
<b>Discontinued Support for parallel.cluster.Mpiexec</b> .....	<b>20-7</b>

## R2013a

<b>GPU-enabled functions in Image Processing Toolbox and Phased Array System Toolbox</b> .....	<b>21-2</b>
<b>More MATLAB functions enabled for use with GPUs, including interp1 and ismember</b> .....	<b>21-2</b>
<b>Enhancements to MATLAB functions enabled for GPUs, including arrayfun, svd, and mldivide ()</b> .....	<b>21-2</b>
<b>Ability to launch CUDA code and manipulate data contained in GPU arrays from MEX-functions</b> .....	<b>21-3</b>
<b>Automatic detection and transfer of files required for execution in both batch and interactive workflows</b> .....	<b>21-3</b>
<b>More MATLAB functions enabled for distributed arrays</b> .....	<b>21-3</b>

<b>More MATLAB functions enabled for GPUs, including convn, cov, and normest</b> .....	22-2
gpuArray Support .....	22-2
MATLAB Code on the GPU .....	22-2
<b>GPU-enabled functions in Neural Network Toolbox, Phased Array System Toolbox, and Signal Processing Toolbox</b> .....	22-2
<b>Performance improvements to GPU-enabled MATLAB functions and random number generation</b> .....	22-2
<b>Automatic detection and selection of specific GPUs on a cluster node when multiple GPUs are available on the node</b> .....	22-3
<b>More MATLAB functions enabled for distributed arrays, including sparse constructor, bsxfun, and repmat</b> .....	22-3
<b>Detection of MATLAB Distributed Computing Server clusters that are available for connection from user desktops through Profile Manager</b> .....	22-3
<b>gpuArray Class Name</b> .....	22-3
<b>Diary Output Now Available During Running Task</b> .....	22-4

<b>New Programming Interface</b> .....	23-2
General Concepts and Phrases .....	23-2
Objects .....	23-2
Functions and Methods .....	23-3
Properties .....	23-4
Getting Help .....	23-4
Enhanced Example Scripts .....	23-6
<b>Cluster Profiles</b> .....	23-7
New Cluster Profile Manager .....	23-7
Programming with Profiles .....	23-7
Profiles in Compiled Applications .....	23-8
<b>Enhanced GPU Support</b> .....	23-8
GPUArray Support .....	23-8
Reset or Deselect GPU Device .....	23-9
Asynchronous GPU Calculations and Wait .....	23-9
Verify GPUArray or CUDAKernel Exists on the Device .....	23-10
MATLAB Code on the GPU .....	23-10
Set CUDA Kernel Constant Memory .....	23-11

Latest NVIDIA CUDA Device Driver .....	23-11
<b>Enhanced Distributed Array Support</b> .....	23-11
Newly Supported Functions .....	23-11
<b>Random Number Generation on Workers</b> .....	23-11

## R2011b

<b>New Job Monitor</b> .....	24-2
<b>Run Scripts as Batch Jobs from the Current Folder Browser</b> .....	24-2
<b>Number of Local Workers Increased to Twelve</b> .....	24-2
<b>Enhanced GPU Support</b> .....	24-3
Latest NVIDIA CUDA Device Driver .....	24-3
Deployment of GPU Applications .....	24-3
Random Number Generation .....	24-3
GPUArray Support .....	24-3
MATLAB Code on the GPU .....	24-4
<b>Enhanced Distributed Array Support</b> .....	24-4
Newly Supported Functions .....	24-4
<b>Conversion of Error and Warning Message Identifiers</b> .....	24-4
<b>Task Error Properties Updated</b> .....	24-5

## R2011a

<b>Deployment of Local Workers</b> .....	25-2
<b>New Desktop Indicator for MATLAB Pool Status</b> .....	25-2
<b>Enhanced GPU Support</b> .....	25-2
Static Methods to Create GPUArray .....	25-2
GPUArray Support .....	25-2
GPUArray Indexing .....	25-2
MATLAB Code on the GPU .....	25-3
NVIDIA CUDA Driver 3.2 Support .....	25-3
<b>Distributed Array Support</b> .....	25-3
Newly Supported Functions .....	25-3
Enhanced mtimes Support .....	25-3

<b>Enhanced parfor Support</b> .....	25-4
Nested for-Loops Inside parfor .....	25-4
<b>Enhanced Support for Microsoft Windows HPC Server</b> .....	25-4
Support for 32-Bit Clients .....	25-4
Search for Cluster Head Nodes Using Active Directory .....	25-4
<b>Enhanced Admin Center Support</b> .....	25-4
<b>New Remote Cluster Access Object</b> .....	25-4

## R2010b

<b>GPU Computing</b> .....	26-2
<b>Job Manager Security and Secure Communications</b> .....	26-2
<b>Generic Scheduler Interface Enhancements</b> .....	26-2
Decode Functions Provided with Product .....	26-2
Enhanced Example Scripts .....	26-2
<b>batch Now Able to Run Functions</b> .....	26-3
<b>batch and matlabpool Accept Scheduler Object</b> .....	26-3
<b>Enhanced Functions for Distributed Arrays</b> .....	26-3
qr Supports Distributed Arrays .....	26-3
mldivide Enhancements .....	26-3
chol Supports 'lower' option .....	26-4
eig and svd Return Distributed Array .....	26-4
transpose and ctranspose Support 2dbc .....	26-4
Inf and NaN Support Multiple Formats .....	26-4
<b>Support for Microsoft Windows HPC Server 2008 R2</b> .....	26-4
<b>User Permissions for MDCEUSER on Microsoft Windows</b> .....	26-4

# R2023a

---

**Version: 7.8**

**New Features**

**Compatibility Considerations**

## PreferredPoolNumWorkers: Specify limited number of workers per cluster profile

The global preference for the default number of workers in a pool is replaced by a per-profile property. You can configure the new `PreferredPoolNumWorkers` cluster object property in the cluster profile manager or at the command line. The `PreferredPoolNumWorkers` property default is the number of workers available to the cluster (`NumWorkers`) for personal cloud cluster profiles or the minimum of `NumWorkers` and 32 for other nonlocal profiles.

Use this new property to create pools with a reasonable number of workers when you call `parpool` without requesting a specific number of workers. You can also use this property to control how many workers you start on clusters to avoid taking up too many resources.

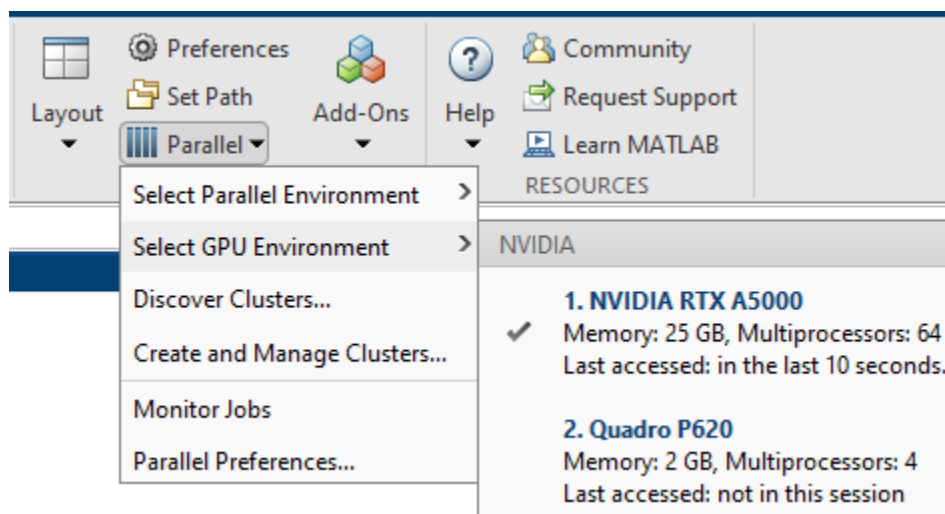
For local profiles, the default parallel pool size is `NumWorkers`. For all profiles, you can continue to request a specific number of workers when you call `parpool` to start a parallel pool outside the main body of code. If you specify a pool size at the command line, you override your preferences. For more details, see “Pool Size and Cluster Selection”.

## Compatibility Considerations

The **Preferred number of workers in a parallel pool** option in **Parallel Preferences** has been removed. Use the `PreferredPoolNumWorkers` or `NumWorkers` cluster object properties instead. For more information, see *Functionality being removed or changed* on page 1-10.

## Parallel Menu: Select a GPU using the Parallel menu

You can now select which GPU device to use for computation from the MATLAB® desktop. On the **Home** tab, in the **Environment** area, select **Parallel > Select GPU Environment**. For each available GPU, the menu displays the total memory, the multiprocessor count, and an indication of when the device was last used. To inspect more properties of your GPU devices, use the `gpuDevice` function.





---

## GPUDevice Object: Changes to device properties

gpuDevice objects that the gpuDevice function returns now have these additional properties:

- GraphicsDriverVersion — Graphics driver version currently in use.
- DriverModel — Operating model of the graphics driver.

On Windows® operating systems, the model options are:

- 'WDDM' — Display model.
- 'TCC' — Compute model.

On other operating systems, DriverModel is 'N/A'.

- CachePolicy — Policy that determines how much GPU memory the GPU can cache to accelerate computation. You can set the CachePolicy property to "balanced", "minimum", or "maximum".

For example, select a GPU using the gpuDevice function to inspect the new properties.

```
D = gpuDevice
```

```
D =
```

```
CUDADevice with properties:
```

```
                Name: 'NVIDIA RTX A5000'  
                Index: 1  
    ComputeCapability: '8.6'  
        SupportsDouble: 1  
GraphicsDriverVersion: '511.79'  
        DriverModel: 'TCC'  
        ToolkitVersion: 11.2000  
    MaxThreadsPerBlock: 1024  
    MaxShmemPerBlock: 49152 (49.15 KB)  
    MaxThreadBlockSize: [1024 1024 64]  
        MaxGridSize: [2.1475e+09 65535 65535]  
        SIMDWidth: 32  
        TotalMemory: 25553076224 (25.55 GB)  
    AvailableMemory: 25145376768 (25.15 GB)  
        CachePolicy: 'balanced'  
    MultiprocessorCount: 64  
        ClockRateKHz: 1695000  
        ComputeMode: 'Default'  
    GPUOverlapsTransfers: 1  
    KernelExecutionTimeout: 0  
    CanMapHostMemory: 1  
    DeviceSupported: 1  
    DeviceAvailable: 1  
    DeviceSelected: 1
```

Change the caching policy to allow the GPU to cache the maximum amount of memory for accelerating computation.

```
D.CachePolicy = "maximum";
```

The `DriverVersion` property no longer displays by default, but you can still query the property using dot notation. There are no plans to remove the `DriverVersion` property.

## GPU Functionality: Use functions with new and enhanced `gpuArray` support

These MATLAB functions have new and enhanced `gpuArray` support:

- `cumtrapz`
- `eig` — You can now provide two input matrices, compute three outputs (eigenvalues, left eigenvectors, and right eigenvectors), specify the output format of the eigenvalues as a matrix or a vector, and provide input matrices containing NaN or Inf values.
- `fillmissing`
- `interp1` — You can now use the "next" and "previous" interpolation methods.
- `lscov`
- `mustBeFloat`, `mustBeGreaterThan`, `mustBeGreaterThanOrEqual`, `mustBeInRange`, `mustBeLessThan`, `mustBeLessThanOrEqual`, `mustBeNegative`, `mustBeNonempty`, `mustBeNonmissing`, `mustBeNonnegative`, `mustBeNonpositive`, `mustBeNonsparse`, `mustBeNumeric`, `mustBeNumericOrLogical`, `mustBePositive`, `mustBeReal`, `mustBeScalarOrEmpty`, `mustBeVector`
- `tensorprod`

For more information, see “Run MATLAB Functions on a GPU”.

These Statistics and Machine Learning Toolbox™ functions have new and enhanced `gpuArray` support:

- `regress`
- `fitrsvm` — You can now specify `gpuArray` inputs.
  - Most object functions of `RegressionSVM`, `CompactRegressionSVM`, and `RegressionPartitionedSVM` run on a GPU when you provide GPU array input arguments. These object functions do not support GPU array input arguments.

Object Function	Model Type
<code>incrementalLearner</code>	<code>RegressionSVM</code> and <code>CompactRegressionSVM</code>
<code>lime</code>	<code>RegressionSVM</code> and <code>CompactRegressionSVM</code>
<code>shapley</code>	<code>RegressionSVM</code> and <code>CompactRegressionSVM</code>
<code>update</code>	<code>CompactRegressionSVM</code>

- You can now use the `gather` function to create a `RegressionSVM`, `CompactRegressionSVM`, or `RegressionPartitionedSVM` object from an equivalent object fitted with GPU array data. The properties of the new object are stored in host memory.

For a list of all Statistics and Machine Learning Toolbox functions with GPU functionality, see Functions with `gpuArray` support.

---

These Signal Processing Toolbox™ functions have new and enhanced gpuArray support:

- bandpower
- medfreq
- obw
- powerbw
- rssq
- signalFrequencyFeatureExtractor
- signalTimeFeatureExtractor
- sinad
- snr
- thd

For a list of all Signal Processing Toolbox functions with GPU functionality, see Functions with gpuArray support.

These Wavelet Toolbox™ functions have new and enhanced gpuArray support:

- modwpt
- wentropy

For a list of all Wavelet Toolbox functions with GPU functionality, see Functions with gpuArray support.

These Communications Toolbox™ functions have new gpuArray support:

- ofdmmod
- ofdmmod

For a list of all Communications Toolbox functions with GPU functionality, see Functions with gpuArray support.

## **PTX Build Option for mexcuda: Compile PTX files using mexcuda**

You can now compile parallel thread execution (PTX) files using the mexcuda function by specifying the -ptx build option. The function compiles the PTX file and gives it the same name as the input CUDA® C++ (CU) source file.

You no longer need to install the CUDA Toolkit to compile a PTX file from a CU file.

For an updated overview of the workflow for creating and executing your own custom CUDAKernel object, see “Run CUDA or PTX Code on GPU”.

## **Support for New GPU Architectures: Update to NVIDIA CUDA 11.8**

Parallel Computing Toolbox software now uses CUDA version 11.8, which supports NVIDIA® GPUs with compute capability up to 9.x, including Hopper architecture GPUs. For more information, see “GPU Computing Requirements”.

## Workflow Examples: New and updated examples and topics

These new and updated examples show how to accelerate your code using a GPU:

- “Accelerate Linear Model Fitting on GPU” (Statistics and Machine Learning Toolbox)
- “Improve Performance of Small Matrix Problems on the GPU Using pagefun”

This updated example shows how to monitor the training of deep neural networks in a batch job using a `ValueStore` object:

- “Send Deep Learning Batch Job to Cluster” (Deep Learning Toolbox)

This new topic helps you choose the right parallel language and workflow for your needs:

- “Parallel Language Decision Tables”

## Thread-Based Environment: Use new functionality on thread workers

These MATLAB functions and classes now have thread-based support:

- `readtable`
- `ToFile`
- `ToUniqueFile`

For more information, see [Run MATLAB Functions in Thread-Based Environment](#).

## Tall Arrays: Use functions with new and enhanced tall array support

These functions have new and enhanced tall array support:

- `mape`
- `rmse`

For more information, see [“Tall Arrays”](#).

## Distributed and Tall tables and timetables: Perform calculations directly on distributed and tall tables and timetables without extracting their data

You can now perform calculations directly on distributed and tall tables and timetables without extracting their data. In previous releases, all calculations require you to extract data from your distributed and tall tables and timetables by indexing into them.

For example, to scale a distributed table that contains numeric data, multiply it by a scale factor.

```
table = array2table(rand(2),VariableNames={"V1","V2"});
distTable = distributed(table)
```

```
distTable =
    2×2 distributed table
      V1      V2
  _____  _____
```

```

    0.69483    0.95022
    0.3171    0.034446

distTable = distTable .* 10

distTable =
    2x2 distributed table
      V1      V2
    _____  _____
    6.9483    9.5022
    3.171    0.34446

```

These MATLAB functions have new support for performing calculations directly on distributed and tall tables and timetables.

- **Arithmetic functions** - ceil, fix, floor, ldivide, minus, mod, plus, power, prod, rdivide, rem, round, sum, times
- **Relational functions** - eq, ge, gt, le, lt, ne
- **Boolean functions** - and, not, or, xor
- **Trigonometric functions** - acos, acosd, acosh, acot, acotd, acoth, acsc, acscd, acsch, asech, asecd, asech, asin, asind, asinh, atan2, atan2d, atand, atanh, cos, cosd, cosh, cospi, cot, cotd, cscd, csch, sec, secd, sech, sin, sind, sinh, sinpi, tan, tand, tanh
- **Exponential and logarithmic functions** - exp, expm1, log, log10, log1p, log2, nextpow2, nthroot, pow2, reallog, realpow, realsqrt, sqrt
- **Complex number functions** - abs
- **Statistics functions** - max, mean, min

For more information about performing calculations directly on tables and timetables, see “Direct Calculations on Tables and Timetables”.

## Composite arrays: Use gather to convert a Composite array into a local cell array.

You can now use the gather function to convert a Composite array containing data on parallel workers into a cell array in the local workspace.

For example, gather data after the execution of an spmd statement.

```

parpool("Threads",3)
spmd, c = ones(spmdIndex); end
data = gather(c)

data =

    1x3 cell array

    {[1]}    {2x2 double}    {3x3 double}

```

## Compatibility Considerations

In previous releases, when you use `gather` on a `Composite` array, MATLAB returns the same `Composite` array as the output. If you have code that relies on this behaviour, update your code to avoid compatibility issues.

## Distributed Arrays: Use functions with new and enhanced distributed array support

These functions have new and enhanced distributed array support:

- `cumtrapz`
- `fillmissing`
- `kron`
- `lscov`
- `movmad`, `movmax`, `movmean`, `movmedian`, `movmin`, `movprod`, `movstd`, `movsum`, `movvar` — You can now use the `SamplePoints` name-value argument.
- `mpower` — You can now use a square matrix for the base or exponent.
- `mustBeFinite`, `mustBeFloat`, `mustBeGreaterThan`, `mustBeGreaterThanOrEqual`, `mustBeInRange`, `mustBeInteger`, `mustBeLessThan`, `mustBeLessThanOrEqual`, `mustBeNegative`, `mustBeNonempty`, `mustBeNonmissing`, `mustBeNonNan`, `mustBeNonnegative`, `mustBeNonpositive`, `mustBeNonsparse`, `mustBeNonzero`, `mustBeNumeric`, `mustBeNumericOrLogical`, `mustBePositive`, `mustBeReal`, `mustBeScalarOrEmpty`, `mustBeVector`
- `null`
- `tensorprod`

For more information, see “Run MATLAB Functions with Distributed Arrays”.

## Discover Clusters Supports Third-Party Schedulers

The Discover Clusters dialog can now locate third-party scheduler clusters for different releases of parallel computing products. For information about cluster discovery, see “Discover Clusters”.

## Third-Party Cluster Schedulers: Access simplified plugin scripts for integration on GitHub

When you integrate MATLAB with your scheduler using MATLAB Parallel Server™, you can now use a single set of scripts and specify the values of one of these `AdditionalProperties` properties to describe your network configuration.

Network Configuration	Previous Submission Mode	New Properties to Set
The MATLAB client does not share the file system with the cluster nodes.	Shared	<ul style="list-style-type: none"> <li>• <code>HasSharedFilesystem = true</code></li> </ul>

Network Configuration	Previous Submission Mode	New Properties to Set
The MATLAB client is unable to directly submit jobs to the third-party scheduler.	Remote	<ul style="list-style-type: none"> <li>• HasSharedFilesystem = true</li> <li>• ClusterHost = HeadnodeHostname</li> </ul>
The MATLAB client does not share the file system with the cluster nodes and is unable to directly submit jobs to the third-party scheduler.	Nonshared	<ul style="list-style-type: none"> <li>• HasSharedFilesystem = false</li> <li>• ClusterHost = HeadnodeHostname</li> </ul>

In previous releases, you have to choose a set of scripts for shared, nonshared, and remote submission modes

The simplified plugin scripts for these third-party schedulers are now available in these GitHub® repositories:

- SLURM
- LSF
- Grid Engine family
- PBS family
- HTCondor

You can still download the plugin scripts from MATLAB Central™ File Exchange.

## Third-Party Cluster Schedulers: Open ports on workers to listen for connections from client

You can now open listening ports on MATLAB Parallel Server workers running on third-party scheduler clusters to enable your MATLAB client to connect to the workers. Use this functionality to run interactive parallel pools on third-party scheduler clusters without the need to modify your network systems such as firewalls. For more details, see `pctconfig`.

## LDAP Support for MATLAB Job Scheduler Clusters: Authenticate cluster logins against LDAP credentials

Validate and control user access to a MATLAB Job Scheduler cluster by using a Lightweight Directory Access Protocol (LDAP) server to authenticate user logins. For more information, see “Configure LDAP Server Authentication for MATLAB Job Scheduler” (MATLAB Parallel Server).

## Cluster Security: Restrict use of MATLAB Job Scheduler cluster commands

You can now restrict the use of cluster changing commands to specific users by verifying commands sent to MATLAB Job Scheduler clusters. In previous releases, any user can execute commands that can change the state of cluster. For more information, see “Set Cluster Command Verification” (MATLAB Parallel Server).

## Parallel Computing in MATLAB Online Server Environment: Speed up MATLAB Online workflows

You can now integrate Parallel Computing Toolbox software and MATLAB Parallel Server with your MATLAB Online Server™ instance and use parallel computing to speed up MATLAB Online™ workflows. For more details about how to configure and use parallel computing features in your MATLAB Online Server instance, see [Integrate MATLAB Online Server with Parallel Computing Toolbox and MATLAB Parallel Server](#) and [Use Parallel Computing in MATLAB Online Hosted by Your Organization](#).

## MATLAB Parallel Server in Kubernetes: Configure and run MATLAB Parallel Server workers on Kubernetes clusters

You can now integrate MATLAB Parallel Server with your Kubernetes® cluster and interface Parallel Computing Toolbox software on your computer to the Kubernetes cluster. For more details, see the [Parallel Computing Toolbox for MATLAB Parallel Server with Kubernetes plugin script on GitHub](#) or [MATLAB Central File Exchange](#).

## Functionality being removed or changed

### Preferred number of workers in a parallel pool option has been removed

*Behavior change*

The **Preferred number of workers in a parallel pool** option in **Parallel Preferences** has been removed. This change gives you more control over how many workers to start on nonlocal clusters.

For MATLAB Job Scheduler, third party schedulers, and cloud clusters, you can use the `PreferredPoolNumWorkers` property of your cluster profiles to specify the preferred number of workers to start in a parallel pool. For the local `Processes` and `Threads` profiles, use the `NumWorkers` property to specify the default number of workers to start in a parallel pool.

### rng("default") on MATLAB parallel workers now sets random number generator settings to worker default

*Behavior change*

When you call the `rng` function with the `"default"` argument on MATLAB parallel workers, MATLAB resets the random number generator settings to the worker default values. The default corresponds to the Threefry generator with 20 rounds and a seed of 0

In previous releases, when you run `rng("default")` on parallel workers, MATLAB changes the worker random number generator settings to the client default values. The default corresponds to the Mersenne Twister generator with a seed of 0.

### parallel.pool.Constant objects no longer automatically transferred to workers

*Behavior change*

MATLAB no longer automatically transfers the `parallel.pool.Constant` object from your current MATLAB session to workers in a parallel pool. MATLAB sends the `Constant` object to workers only if the object is required to execute your code.

### Thread-Based Parallel Pool: startup or matlabrc no longer runs when workers start

*Behavior change*



---

When you start a thread-based parallel pool, the thread-based workers no longer run the `startup` or `matlabrc` file. All startup options on the client are automatically mirrored to the thread-based workers.

**`pmode` has been removed**

*Errors*

The `pmode` function has been removed. To execute commands interactively on multiple workers, use `spm` instead.

**`pload` and `psave` have been removed**

*Errors*

The `pload` and `psave` functions have been removed. To save and load data on workers, in the form of composite arrays or distributed arrays, use `dsave` and `dload` instead.



# R2022b

---

**Version: 7.7**

**New Features**

**Compatibility Considerations**

## Thread-Based Parallel Pool: Specify number of thread workers using `parpool`

You can now specify the pool size of a thread-based parallel pool by using `parpool`.

For example, start a pool of four thread workers.

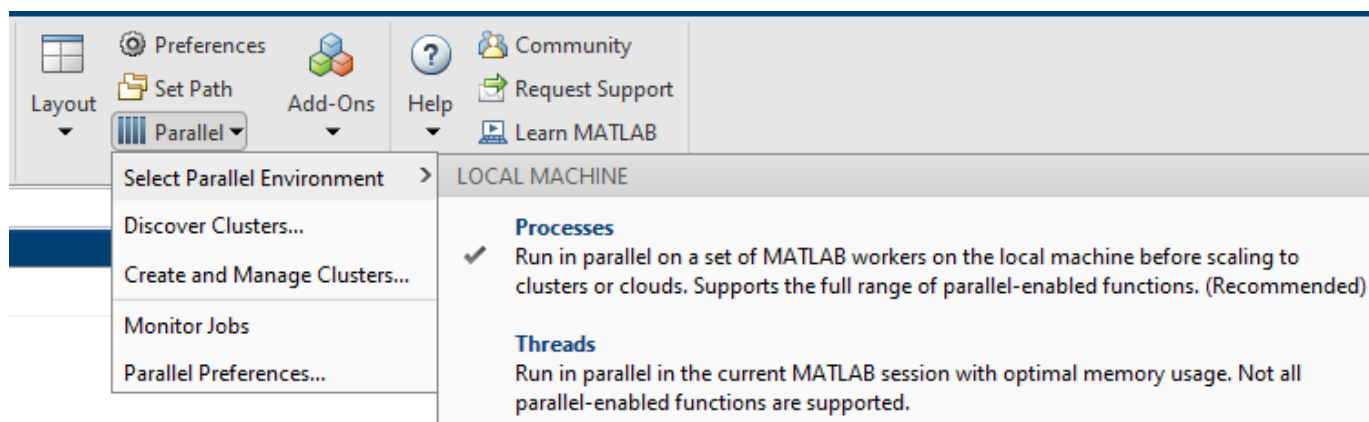
```
parpool("Threads",4);
```

You can also set `Threads` as the default parallel environment and start a parallel pool with the largest possible number of workers.

```
parallel.defaultProfile("Threads");
parpool([1 50]);
```

## Parallel menu: Select Threads using the Parallel menu

You can now select the `Threads` parallel environment option from the MATLAB desktop **Parallel** menu.



- You can set the default parallel environment on your local machine from the MATLAB desktop **Home** tab, in the **Environment** area, by selecting one of these options:
  - Parallel > Select Parallel Environment > Processes**
  - Parallel > Select Parallel Environment > Threads**

You can also change the default profile in **Parallel Preferences**. For more information, see [Specify Your Parallel Preferences](#).

## Compatibility Considerations

The `local` profile is no longer recommended. For more information, see [Functionality being removed or changed on page 2-11](#).

## gpuDevice Object: Display values for byte-based memory properties with appropriate units

The `gpuDevice` object now displays the `TotalMemory`, `AvailableMemory`, and `MaxShmemPerBlock` properties with appropriately scaled units. The object displays these properties

---

as bytes (B), kilobytes (KB), megabytes (MB), or gigabytes (GB). Accessing any of these properties using dot notation still returns a value in bytes.

For example, create a `gpuDevice` object and access its `TotalMemory` property.

```
g = gpuDevice
```

```
g =
```

```
  CUDADevice with properties:
```

```
          Name: 'NVIDIA RTX A5000'  
          Index: 1  
  ComputeCapability: '8.6'  
    SupportsDouble: 1  
      DriverVersion: 11.6000  
      ToolkitVersion: 11.2000  
  MaxThreadsPerBlock: 1024  
  MaxShmemPerBlock: 49152 (49.15 KB)  
  MaxThreadBlockSize: [1024 1024 64]  
      MaxGridSize: [2.1475e+09 65535 65535]  
      SIMDWidth: 32  
      TotalMemory: 25553076224 (25.55 GB)  
      AvailableMemory: 25153765376 (25.15 GB)  
  MultiprocessorCount: 64  
      ClockRateKHz: 1695000  
      ComputeMode: 'Default'  
  GPUOverlapsTransfers: 1  
  KernelExecutionTimeout: 0  
  CanMapHostMemory: 1  
  DeviceSupported: 1  
  DeviceAvailable: 1  
  DeviceSelected: 1
```

```
g.TotalMemory
```

```
ans =
```

```
 2.5553e+10
```

## GPU Workflow Examples: New and updated examples and topics

These new and updated examples and topics show how to accelerate your code using a GPU:

- Run MATLAB using GPUs in the Cloud
- Measure and Improve GPU Performance
- Improve Performance Using a GPU and Vectorized Calculations
- Improve Performance of Element-Wise MATLAB Functions on the GPU Using `arrayfun`
- Using GPU `arrayfun` for Monte-Carlo Simulations

## Cloud GPU Computing: Access Cloud GPUs and Clusters in AWS Using Cloud Center

If you do not have a GPU available, you can speed up your MATLAB code with one or more high-performance NVIDIA GPUs in the cloud. Working in the cloud requires some initial setup, but using

cloud resources can significantly accelerate your code without requiring you to buy and set up your own local GPUs. The simplest way to access a high-performance cloud GPU is to use Cloud Center.

From April 2022, you can use Cloud Center to start MATLAB in Amazon® Web Services (AWS®). You can start a single machine, with MATLAB installed, that you can access from a web browser or a remote desktop application. To get started, see [Get Started with Cloud Center and Start MATLAB on Amazon Web Services \(AWS\) Using Cloud Center](#). To learn more about other cloud options for GPU access, see [Run MATLAB using GPUs in the Cloud](#).

You can continue to use Cloud Center to start and manage MATLAB Parallel Server clusters that you can access from any MATLAB client. MATLAB Parallel Server clusters provide more hardware, including GPU and CPU clusters.

## **New Examples: Use ValueStore to monitor batch jobs**

These new examples show how to retrieve data and monitor batch jobs with a ValueStore object:

- [Monitor Batch Jobs with ValueStore](#) shows how to use ValueStore to update a waitbar.
- [Monitor Monte Carlo Batch Jobs with ValueStore](#) shows how to use ValueStore to access and monitor data from a batch Monte Carlo simulation.

## **Parallel workflow examples: Use clusters for 5G Communications simulations**

This new Communications Toolbox example shows how to accelerate simulations using clusters for 5G LDPC Block Error Rate:

- [5G LDPC Block Error Rate Simulation Using the Cloud or a Cluster \(Communications Toolbox\)](#)

## **Data Analysis: Use mapreduce on Spark clusters**

Parallel Computing Toolbox and MATLAB Parallel Server support the use of Spark™ clusters for the execution environment of mapreduce applications. For more information, see:

- [Configure a Spark Cluster \(MATLAB Parallel Server\)](#)
- [Use Tall Arrays on a Spark Cluster](#)

## **Parallel Computing page: Learn about MathWorks solutions that help you take advantage of more hardware resources**

Use the new Parallel Computing page to discover solutions that use parallel, GPU, and cloud computing.

## **Multistep Authentication for Third-Party Schedulers: Connect to remote clients with multiple authentication options**

You can now use a RemoteClusterAccess object to perform multistep authentication with any number of authentication options. To connect to a cluster with multiple authentication requirements, specify the AuthenticationMode property of the RemoteClusterAccess object as a string array

---

or cell array containing a combination of "Agent", "IdentityFile", "Multifactor", and "Password".

For example, create a `RemoteClusterAccess` object to connect to a cluster that requires a password and identity file.

```
RemoteClusterAccess(AuthenticationMode={"IdentityFile", "Password"});
```

The sample plugin scripts for third-party schedulers now accept the cell array `{"IdentityFile", "Password"}` as the value of the `AuthenticationMode` property of an `AdditionalProperties` object, which sets the same `AuthenticationMode` property value in the `RemoteClusterAccess` object.

## Thread-Based Environment: Use new functionality on thread workers

These Parallel Computing Toolbox functions now have new thread-based support:

- `spmdBarrier`
- `spmdBroadcast`
- `spmdIndex`
- `spmdProbe`
- `spmdReceive`
- `spmdSend`
- `spmdSendReceive`
- `spmdSize`
- `spmdCat`
- `spmdReduce`
- `spmdPlus`

These Image Processing Toolbox™ functions now have thread-based support.

<code>adapthisteq</code> (Image Processing Toolbox)	<code>bfscore</code> (Image Processing Toolbox)	<code>bwconncomp</code> (Image Processing Toolbox)	<code>bwdist</code> (Image Processing Toolbox)
<code>bwferet</code> (Image Processing Toolbox)	<code>bwlabel</code> (Image Processing Toolbox)	<code>bwlabeln</code> (Image Processing Toolbox)	<code>bwlookup</code> (Image Processing Toolbox)
<code>bwmorph3</code> (Image Processing Toolbox)	<code>bwpack</code> (Image Processing Toolbox)	<code>bwskel</code> (Image Processing Toolbox)	<code>bwtraceboundary</code> (Image Processing Toolbox)
<code>bwunpack</code> (Image Processing Toolbox)	<code>demosaic</code> (Image Processing Toolbox)	<code>edge</code> (Image Processing Toolbox)	<code>edge3</code> (Image Processing Toolbox)
<code>entropyfilt</code> (Image Processing Toolbox)	<code>exrHalfAsSingle</code> (Image Processing Toolbox)	<code>exrinfo</code> (Image Processing Toolbox)	<code>exrread</code> (Image Processing Toolbox)
<code>exrwrite</code> (Image Processing Toolbox)	<code>fibermetric</code> (Image Processing Toolbox)	<code>grabcut</code> (Image Processing Toolbox)	<code>histeq</code> (Image Processing Toolbox)

<code>hough</code> (Image Processing Toolbox)	<code>imabsdiff</code> (Image Processing Toolbox)	<code>imapplymatrix</code> (Image Processing Toolbox)	<code>imbothat</code> (Image Processing Toolbox)
<code>imboxfilt</code> (Image Processing Toolbox)	<code>imboxfilt3</code> (Image Processing Toolbox)	<code>imclose</code> (Image Processing Toolbox)	<code>imdiffusefilt</code> (Image Processing Toolbox)
<code>imdilate</code> (Image Processing Toolbox)	<code>imerode</code> (Image Processing Toolbox)	<code>imfill</code> (Image Processing Toolbox)	<code>imfilter</code> (Image Processing Toolbox)
<code>imgaussfilt</code> (Image Processing Toolbox)	<code>imgaussfilt3</code> (Image Processing Toolbox)	<code>imgradient</code> (Image Processing Toolbox)	<code>imgradient3</code> (Image Processing Toolbox)
<code>imhist</code> (Image Processing Toolbox)	<code>imlincomb</code> (Image Processing Toolbox)	<code>imnlmfilt</code> (Image Processing Toolbox)	<code>imopen</code> (Image Processing Toolbox)
<code>imreconstruct</code> (Image Processing Toolbox)	<code>imregionalmax</code> (Image Processing Toolbox)	<code>imregionalmin</code> (Image Processing Toolbox)	<code>imsegfmm</code> (Image Processing Toolbox)
<code>imtophat</code> (Image Processing Toolbox)	<code>inpaintCoherent</code> (Image Processing Toolbox)	<code>inpaintExemplar</code> (Image Processing Toolbox)	<code>integralImage</code> (Image Processing Toolbox)
<code>integralImage3</code> (Image Processing Toolbox)	<code>intlut</code> (Image Processing Toolbox)	<code>iradon</code> (Image Processing Toolbox)	<code>isexr</code> (Image Processing Toolbox)
<code>label2idx</code> (Image Processing Toolbox)	<code>medfilt2</code> (Image Processing Toolbox)	<code>medfilt3</code> (Image Processing Toolbox)	<code>modfilt</code> (Image Processing Toolbox)
<code>multisim</code> (Image Processing Toolbox)	<code>multisim3</code> (Image Processing Toolbox)	<code>nitfread</code> (Image Processing Toolbox)	<code>obliquesslice</code> (Image Processing Toolbox)
<code>ordfilt2</code> (Image Processing Toolbox)	<code>planar2raw</code> (Image Processing Toolbox)	<code>poly2mask</code> (Image Processing Toolbox)	<code>qtdecomp</code> (Image Processing Toolbox)
<code>radon</code> (Image Processing Toolbox)	<code>raw2planar</code> (Image Processing Toolbox)	<code>raw2rgb</code> (Image Processing Toolbox)	<code>rawread</code> (Image Processing Toolbox)
<code>regionprops</code> (Image Processing Toolbox)	<code>regionprops3</code> (Image Processing Toolbox)	<code>rgb2lab</code> (Image Processing Toolbox)	<code>rgb2lightness</code> (Image Processing Toolbox)
<code>superpixels</code> (Image Processing Toolbox)	<code>superpixels3</code> (Image Processing Toolbox)	<code>watershed</code> (Image Processing Toolbox)	

For more information, see [Run MATLAB Functions in Thread-Based Environment](#).

## GPU Functionality: Use functions with new and enhanced gpuArray support

These MATLAB functions have new and enhanced `gpuArray` support:

- `allfinite`
- `anymissing`
- `anynan`



- 
- `ind2rgb`
  - `islocalmax`
  - `islocalmin`
  - `linsolve`
  - `mape`
  - `pagefun` - Support for `norm`
  - `pagemldivide`
  - `pagemrdivide`
  - `pagenorm`
  - `rmoutliers` - Support for defining outlier locations and returning the outlier indicator, thresholds, and center value outputs
  - `rmse`

For more information, see [Run MATLAB Functions on a GPU](#).

These Statistics and Machine Learning Toolbox functions have enhanced `gpuArray` support:

- `fitcecoc` (Statistics and Machine Learning Toolbox) - Support for categorical predictors for classification tree learners
- `fitcensemble` (Statistics and Machine Learning Toolbox) - Support for categorical predictors for classification tree learners
- `fitcsvm` (Statistics and Machine Learning Toolbox)
  - Support for hyperparameter optimization
  - When `fitcsvm` fits the SVM model on a GPU, some of the properties of the `ClassificationSVM` (Statistics and Machine Learning Toolbox) model are stored in GPU memory
  - Use `gather` to create a `ClassificationSVM` or `CompactClassificationSVM` object with properties stored in the local workspace from an equivalent object with properties stored in GPU memory
- `fitctree` (Statistics and Machine Learning Toolbox) - Support for categorical predictors
- `fitensemble` (Statistics and Machine Learning Toolbox) - Support for categorical predictors
- `fitrensemble` (Statistics and Machine Learning Toolbox) - Support for categorical predictors
- `fitrtree` (Statistics and Machine Learning Toolbox) - Support for categorical predictors
- `pca` (Statistics and Machine Learning Toolbox) - Improved performance when you use the SVD algorithm on page 2-9
- Object functions of the `CompactClassificationSVM` (Statistics and Machine Learning Toolbox) model execute on a GPU if the model is fitted with GPU arrays
- Object functions of the `ClassificationSVM` (Statistics and Machine Learning Toolbox) model execute on a GPU if the model is fitted with GPU arrays

For a list of all Statistics and Machine Learning Toolbox functions with GPU functionality, see [Functions with `gpuArray` support \(Statistics and Machine Learning Toolbox\)](#).

These Signal Processing Toolbox functions have new and enhanced `gpuArray` support:

- `instbw` (Signal Processing Toolbox)
- `instfreq` (Signal Processing Toolbox)
- `meanfreq` (Signal Processing Toolbox)
- `p octave` (Signal Processing Toolbox)
- `sosfilt` (Signal Processing Toolbox) - Support for FIR second-order subsections of the input filter

For a list of all Signal Processing Toolbox functions with GPU functionality, see [Functions with `gpuArray` support](#) (Signal Processing Toolbox).

This Audio Toolbox™ function has enhanced `gpuArray` support :

- `audioFeatureExtractor` (Audio Toolbox) - Code acceleration support for `zerocrossrate`

For a list of all Audio Toolbox functions with GPU functionality, see [Functions with `gpuArray` support](#) (Audio Toolbox).

This Image Processing Toolbox function has new `gpuArray` support:

- `gray2ind` (Image Processing Toolbox)

For a list of all Image Processing Toolbox functions with GPU functionality, see [Functions with `gpuArray` support](#) (Image Processing Toolbox).

## **Tall Arrays: Use functions with new and enhanced tall array support**

These functions have new and enhanced tall array support:

- `allfinite`
- `anymissing`
- `anynan`
- `rmoutliers` - Support for defining outlier locations and returning the outlier indicator, thresholds, and center value outputs
- `round` - Support for specifying a direction for rounding ties

For more information, see [Tall Arrays](#).

## **Distributed Arrays: Use functions with new and enhanced distributed array support**

These functions have new and enhanced distributed array support:

- `allfinite`
- `anymissing`
- `anynan`
- `ind2rgb`
- `mape`
- `pageinv`
- `pagemldivide`

- 
- `pagemrdivide`
  - `pagenorm`
  - `rank`
  - `rms`
  - `rmse`
  - `round` - Support for specifying a direction for breaking ties

For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## MATLAB Job Scheduler: Support for your own Java installation

You can now use your own Java® 8 installation with MATLAB Job Scheduler (MJS). To use your own Java 8 installation with MJS, specify a path to a Java Runtime Environment (JRE) installation for the `MJS_Java` variable in your `mjs_def` file. For more information on where to find the `mjs_def` file, see [Customize Startup Parameters \(MATLAB Parallel Server\)](#).

## Element-Wise Operations: Improved performance on GPU

Element-wise operations on large numbers of `gpuArray` objects show improved performance. Improvements are greater when you operate on large numbers of `gpuArray` objects. For example, adding one to every element of a cell array of the `gpuArray` data in this function is about 43.8x faster than in the previous release:

```
function timeElementWiseOps

% Prepare a cell array of gpuArray data
x = cell(5000,1);
x = cellfun(@(z) rand(10,"gpuArray"),x,"UniformOutput",false);

% Make all arrays different to one another
x = cellfun(@(z) z+rand(1,"gpuArray"),x,"UniformOutput",false);

% Time adding one to each element
gputimeit(@( ) cellfun(@(z) z+1,x,"UniformOutput",false))

end
```

The approximate execution times are:

**R2022a:** 5.70 seconds

**R2022b:** 0.13 seconds

The code was timed on a Windows 10, Intel® Xeon® W-2133 @ 3.60 GHz test system with an NVIDIA RTX A5000 GPU by calling the `timeElementWiseOps` function.

## svd Function: Improved performance on GPU

The `svd` function shows improved performance when called with a `gpuArray` input matrix that is short and wide ( $n > 3m$ ) or tall and narrow ( $m > 6n$ ). For example, performing a singular value decomposition of a matrix in this function is about 1.4x faster than in the previous release:

```

function timeSVD
% Prepare input matrix
A = rand(1000,10000,"gpuArray");
% Time SVD
gputimeit(@() svd(A),3)
end

```

The approximate execution times are:

**R2022a:** 2.64 seconds

**R2022b:** 1.88 seconds

The code was timed on a Windows 10, Intel Xeon W-2133 @ 3.60 GHz test system with an NVIDIA RTX A5000 GPU by calling the `timeSVD` function.

## Functionality being removed or changed

### labXxx functions are not recommended and are renamed to spmdXxx

*Still runs*

To align these function names with their intended use within `spmd` blocks, these `spmd` block code execution and communication functions have been renamed:

- `labBarrier`
- `labBroadcast`
- `labindex`
- `labProbe`
- `labReceive`
- `labSend`
- `labSendReceive`
- `numlabs`

These functions are no longer recommended, but they will continue to work. This table shows the recommended replacement functions.

Functionality	Recommended Replacement	Compatibility Considerations
<code>labBarrier</code>	<code>spmdBarrier</code>	Replace all instances of <code>labBarrier</code> with <code>spmdBarrier</code>
<code>labBroadcast</code>	<code>spmdBroadcast</code>	Replace all instances of <code>labBroadcast</code> with <code>spmdBroadcast</code>
<code>labindex</code>	<code>spmdIndex</code>	Replace all instances of <code>labindex</code> with <code>spmdIndex</code>

Functionality	Recommended Replacement	Compatibility Considerations
labProbe	spmdProbe	Replace all instances of labProbe with spmdProbe
labReceive	spmdReceive	Replace all instances of labReceive with spmdReceive
labSend	spmdSend	Replace all instances of labSend with spmdSend
labSendReceive	spmdSendReceive	Replace all instances of labSendReceive with spmdSendReceive
numlabs	spmdSize	Replace all instances of numlabs with spmdSize

The labXxx functions will not be removed.

**gcat, gop, and gplus are not recommended and are renamed to spmdCat, spmdReduce, and spmdPlus**

*Still runs*

For performing SPMD operations on all the workers in an spmd block, gcat, gop, and gplus are no longer recommended. Use spmdCat, spmdReduce, and spmdPlus as direct replacements. The gcat, gop, and gplus functions will not be removed.

**local profile has been renamed to Processes on the Parallel menu**

*Still runs*

For a process-based parallel environment on a local machine, the local profile is no longer recommended. Use Processes instead.

- To start a parallel pool of process workers, use this code.

```
parpool("Processes")
```

In previous releases, you used this code which is no longer recommended.

```
parpool("local")
```

The local profile option has been removed from the **Parallel** menu but will continue to work when you use it programmatically.

**parallel.defaultClusterProfile and parallel.clusterProfiles are renamed to parallel.defaultProfile and parallel.listProfiles**

*Still runs*

The parallel.defaultClusterProfile and parallel.clusterProfiles functions are renamed. parallel.defaultClusterProfile is renamed to parallel.defaultProfile. parallel.clusterProfiles is renamed to parallel.listProfiles. The parallel.defaultClusterProfile and parallel.clusterProfiles will not be removed.

To update your code, replace all instances of parallel.defaultClusterProfile with parallel.defaultProfile and parallel.clusterProfiles with parallel.listProfiles.

**remotecopy has been removed**

*Errors*

Starting in R2022b, `remotecopy` (MATLAB Parallel Server) has been removed. To copy files to and from a remote host, use `scp` or `sftp` instead.

- Previously you used `remotecopy` and `-protocol scp` to copy files to and from a remote host using the secure copy protocol (SCP).

This table shows how to use `scp` instead.

Errors	Recommended
<code>remotecopy -remotehost host1 -local /my/file/path -to -remotehost host1:/remote/file/path</code>	<code>scp /my/file/path host1:/remote/file/path</code>
<code>remotecopy -remotehost host1,host2 -local /my/file/path -to -remotehost host1:/remote/file/path -protocol scp</code>	<code>scp /my/file/path host1:/remote/file/path scp /my/file/path host2:/remote/file/path</code>
<code>remotecopy -remotehost host1 -local /my/file/path -to -remotehost host1:/remote/file/path -protocol scp</code>	<code>scp host1:/remote/file/path /my/file/path</code>

- Previously you used `remotecopy` and `-protocol sftp` to copy files to and from a remote host using the secure file transfer protocol (SFTP).

This table shows how to use `sftp` instead.

Errors	Recommended
<code>remotecopy -remotehost host1 -local /my/file/path -to -remotehost host1:/remote/file/path -protocol sftp</code>	<code>sftp /my/file/path host1:/remote/file/path</code>
<code>remotecopy -remotehost host1,host2 -local /my/file/path -to -remotehost host1:/remote/file/path -protocol sftp</code>	<code>sftp /my/file/path host1:/remote/file/path sftp /my/file/path host2:/remote/file/path</code>
<code>remotecopy -remotehost host1 -local /my/file/path -to -remotehost host1:/remote/file/path -protocol sftp</code>	<code>sftp host1:/remote/file/path /my/file/path</code>

**remotemjs has been removed**

*Errors*

Starting in R2022b, `remotemjs` (MATLAB Parallel Server) has been removed. Use `ssh` instead.

Previously you used `remotemjs` to run MATLAB Job Scheduler commands on a remote host using `-protocol ssh` or `-protocol winsc`.

This table shows how to use `ssh` instead.

Errors	Recommended
<code>remotemjs &lt;mjs options&gt; -matlabroot &lt;installfoldername&gt; -remotehost &lt;remotehost&gt;</code>	<code>ssh &lt;remotehost&gt; &lt;installfoldername&gt;/toolbox/parallel/bin/mj</code>
<code>remotemjs &lt;mjs options&gt; -matlabroot &lt;installfoldername&gt; -remotehost &lt;remotehost&gt; -protocol ssh</code>	<code>ssh host2 &lt;installfoldername&gt;/toolbox/parallel/bin/mj</code>

# R2022a

---

**Version: 7.6**

**New Features**

**Compatibility Considerations**

## **ValueStore and FileStore objects: Retrieve data and files on MATLAB clients during job execution**

ValueStore and FileStore objects now allow you to store data and files from MATLAB workers that can be retrieved by MATLAB clients during the execution of a job (even while the job is still running). These objects are not held in system memory, so they can be used to store large results. A FileStore object provides a universal location for workers to copy files. MATLAB clients can then access these files regardless of the specific cluster environment where the worker is running.

The ValueStore and FileStore objects are automatically created when you create a job on a cluster, a parallel pool of process workers on your local machine, or a parallel pool of workers on a cluster of machines. To access the ValueStore and FileStore objects on a worker, use the `getCurrentValueStore` and `getCurrentFileStore` functions, respectively.

## **Multifactor Authentication for the Generic Scheduler Interface: Connect to remote clients with RemoteClusterAccess using two or more authentication factors**

RemoteClusterAccess objects now allow you to perform multifactor authentication, including two-factor authentication. To do so, specify 'Multifactor' for the AuthenticationMode name-value argument. For details, see the RemoteClusterAccess reference page. The sample plugin scripts for third-party schedulers now accept 'Multifactor' as the value for the AuthenticationMode property of AdditionalProperties, which sets that value on RemoteClusterAccess.

## **Cluster resizing: Customize your MATLAB Job Scheduler cluster to resize automatically**

You can customize your MATLAB Job Scheduler (MJS) cluster to resize automatically. After you set up auto-resizing, also called auto-scaling, the cluster can automatically change the number of workers with the amount of work submitted. The cluster grows (scales up) when there is more work to do and shrinks (scales down) when there is less work to do. This allows you to use your compute resources more efficiently and can result in cost savings. To learn more, see Set up MATLAB Job Scheduler Cluster for Auto-Resizing (MATLAB Parallel Server).

## **Parallel Pools: Check status of pools**

Starting in R2022a, you can query if any type of pool is currently running by using the Busy property of the pool. This property indicates whether the parallel pool is busy, specified as true or false. The pool is busy if there is outstanding work for the pool to complete.

## **Background Pool: Number of thread workers no longer limited to 8 workers**

Before R2022a, the NumWorkers property was capped at a maximum of 8 workers when Parallel Computing Toolbox was installed, and 1 worker when not installed. This limit is no longer in place when the Parallel Computing Toolbox is installed. When Parallel Computing Toolbox is not installed, the background pool remains limited to 1 worker.



---

## Thread-Based Parallel Pool: See futures in active pool

Starting in R2022a, you can query all queued and running futures on a parallel pool by using the `FevalQueue` property of the pool. To create futures, use `parfeval` and `parfevalOnAll`. For more information on futures, see `Future`.

## cancelAll Method: Cancel currently queued and running futures in a parallel pool

`cancelAll` cancels all futures currently queued or running in a parallel pool. Queued or running futures are listed in the `FevalQueue` property.

For example, you can use `cancelAll` to stop all `Futures` in an `FevalQueue`.

```
pool = parpool;  
cancelAll(pool.FevalQueue);
```

## GPU Functionality: Use new and enhanced gpuArray functions

New GPU support in MATLAB:

- `besseli`, `besselk`
- `besselj`, `bessely` — Support for `scale` input argument
- `deconv`
- `eps`, `flintmax`, `intmax`, `intmin`, `realmax`, `realmin` — Support for "like" syntax to return scalars from prototype object
- `filter` — Support for all syntaxes and any vector sizes of the numerator and denominator coefficients of a rational transfer function in the input arguments
- `pagefun` — Support for `conv2`, `qr`, and `svd`
- `pageinv`
- `pagesvd`
- `svd`

For more information, see `Run MATLAB Functions on a GPU`.

New GPU support in Statistics and Machine Learning Toolbox:

- `fitensemble` (Statistics and Machine Learning Toolbox)
- `fitrensemble` (Statistics and Machine Learning Toolbox)
- New support for probability functions: `gev*`, `gp*`, `nbin*`

For a list of all Statistics and Machine Learning Toolbox functions with GPU functionality, see `Functions with gpuArray support (Statistics and Machine Learning Toolbox)`.

The following functions have new and enhanced `gpuArray` support in Signal Processing Toolbox:

- `findpeaks` (Signal Processing Toolbox)
- `zerocrossrate` (Signal Processing Toolbox)
- `pkurtosis` (Signal Processing Toolbox)

- `pentropy` (Signal Processing Toolbox)
- `levinson` (Signal Processing Toolbox)
- `hilbert` (Signal Processing Toolbox)
- `pow2db` (Signal Processing Toolbox)
- `db2pow` (Signal Processing Toolbox)

For a list of all Signal Processing Toolbox functions with GPU functionality, see [Functions with `gpuArray` support](#) (Signal Processing Toolbox).

The following functions have new `gpuArray` support in Audio Toolbox:

- `pitch` (Audio Toolbox)
- `audioDataAugmenter` (Audio Toolbox)
- `audioFeatureExtractor` (Audio Toolbox) - improved support

For a list of all Audio Toolbox functions with GPU functionality, see [Functions with `gpuArray` support](#) (Audio Toolbox).

The following functions have new `gpuArray` and `dlArray` support in Wavelet Toolbox:

- `imodwt` (Wavelet Toolbox)
- `modwt` (Wavelet Toolbox)
- `modwtmra` (Wavelet Toolbox)

For a list of all Wavelet Toolbox functions with GPU functionality, see [Functions with `gpuArray` support](#) (Wavelet Toolbox).

## Support for NVIDIA CUDA 11.2: Update to CUDA Toolkit 11.2

The parallel computing products now use CUDA toolkit version 11.2. To generate CUDA kernel objects from CU code or compile CUDA compatible source code, libraries, and executables using GPU Coder™, you must use toolkit version 11.2. For more information, see [Run CUDA or PTX Code on GPU](#).

## Tall Arrays: Use new and enhanced tall array functionality

- `eps`, `flintmax`, `intmax`, `intmin`, `realmax`, `realmin` — Support for "like" syntax to return scalars from prototype object

For more information, see [Tall Arrays](#).

## Distributed Arrays: Use new and enhanced distributed array functionality

- `deconv`
- `eps`, `flintmax`, `intmax`, `intmin`, `realmax`, `realmin` — Support for "like" syntax to return scalars from prototype object
- `interp3`, `interpn`

- 
- `movmad`, `movmax`, `movmean`, `movmedian`, `movmin`, `movprod`, `movstd`, `movsum`, `movvar`

For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## Parallel Features in Other Products

Parallel features added in other products:

- **Experiment Manager**: Offload deep learning experiments as batch jobs in a cluster

Starting in R2022a, **Experiment Manager** (Deep Learning Toolbox) supports offloading experiments as batch jobs in a cluster. You can configure the cluster to run multiple trials at the same time or to run a single trial at a time on multiple parallel workers. While the experiment is running in the cluster, you can run other experiments, close the app and continue using MATLAB, or close your MATLAB session. For more information, see [Offload Experiments as Batch Jobs to Cluster](#) (Deep Learning Toolbox).

- **Parallel Simulations**: Perform parameter sweeps using Parameter Combinations

In R2022a, you can use Parameter Combinations in the Multiple Simulations panel of the Simulink® Editor for workflows with multiple simulations, such as Monte-Carlo simulations and parameter sweeps. Parameter Combinations allows you to create sequential and exhaustive combinations of parameters, specify value ranges, and run simulations with these combinations. The Multiple Simulations panel was introduced in R2021b. For more information, see [Multiple Simulations Panel: Simulate for Different Values of Stiffness for a Vehicle Dynamics System](#) (Simulink).

- **Machine Learning Apps**: Train draft models in parallel or train in the background to keep the app responsive

In **Classification Learner** (Statistics and Machine Learning Toolbox) and **Regression Learner** (Statistics and Machine Learning Toolbox), you can train multiple draft models in parallel. You can use the **Use Parallel** or **Use Background** buttons while training models.

## Solving Linear System: Improved performance when solving linear systems $A \cdot X = B$ with `gpuArray` for symmetric positive definite matrices **A**

Solving a linear system of the form  $A \cdot X = B$  with `gpuArray` by executing `A \ B` shows improved performance when **A** is a symmetric positive definite matrix.

For example, this code solves  $A \cdot X = B$  for a 10,000-by-10,000 symmetric positive definite matrix **A** and a 10,000-by-1 column vector **B**. The code is about 2.4x faster than in the previous release.

```
function timingTest
rng default
R = rand(10000, "gpuArray");
A = R'*R;
B = ones(10000, 1, "gpuArray");
X = A \ B;
end
```

The approximate execution times are:

**R2021b:** 1.03 s

**R2022a:** 0.43 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1640 v3 @ 3.50 GHz with an NVIDIA Titan V GPU test system using the `gputimeit` function:

```
gputimeit(@timingTest)
```

## **Functionality being removed or changed**

### **distcomp folder removed, now named parallel**

The `distcomp` folder has been removed.

In R2019b, the Parallel Computing Toolbox folder was renamed. Since then, the name of the toolbox folder has been `parallel`. If you need to reference the location of the toolbox, update your references to use `toolbox/parallel` instead of `toolbox/distcomp`.

# R2021b

---

**Version: 7.5**

**New Features**

**Compatibility Considerations**

## Parallel Language in MATLAB: Share parallel code with any MATLAB user

From R2021b, you can use more parallel language features in serial without Parallel Computing Toolbox. To scale up and speed up computations that use these features, use parallel pools and Parallel Computing Toolbox.

Additionally, you can now share parallel code with other MATLAB users who do not have Parallel Computing Toolbox.

The following features are now available in MATLAB:

- `parfeval` and related functionality such as `afterEach` and `afterAll`
- `parallel.pool.DataQueue`, `parallel.pool.PollableDataQueue`, and related functionality such as `afterEach`

For more information, see [Background Processing](#) and [Write Portable Parallel Code](#).

## GPU Functionality: Use new and enhanced `gpuArray` functions

- `makima`
- `movmad`
- `movmax`
- `movmedian`
- `movmin`
- `movprod`
- `pchip`
- `ppval`
- `filter` — Support for more than 10 elements for numerator coefficients of rational transfer function input argument.
- `pagefun` — Support for `conv`

For more information, see [Run MATLAB Functions on a GPU](#).

## GPU Functionality: Use new and enhanced `gpuArray` functions in Statistics and Machine Learning Toolbox

- `betafit` (Statistics and Machine Learning Toolbox)
- `fitcecoc` (Statistics and Machine Learning Toolbox)
- `fitcensemble` (Statistics and Machine Learning Toolbox)
- `fitctree` (Statistics and Machine Learning Toolbox)
- `fitdist` (Statistics and Machine Learning Toolbox)
- `fitrtree` (Statistics and Machine Learning Toolbox)
- `gevfit` (Statistics and Machine Learning Toolbox)
- `gpfit` (Statistics and Machine Learning Toolbox)

- 
- `ksdensity` (Statistics and Machine Learning Toolbox)
  - `mle` (Statistics and Machine Learning Toolbox)
  - `mvksdensity` (Statistics and Machine Learning Toolbox)
  - `nbinfit` (Statistics and Machine Learning Toolbox)

For a list of all Statistics and Machine Learning Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Statistics and Machine Learning Toolbox\)](#).

## **GPU Functionality: Use new and enhanced gpuArray functions for working with signals and audio**

- The following functions have new and enhanced `gpuArray` support in Signal Processing Toolbox: For a list of all Signal Processing Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Signal Processing Toolbox\)](#).
  - `downsample` (Signal Processing Toolbox)
  - `filtfilt` (Signal Processing Toolbox)
  - `pspectrum` (Signal Processing Toolbox)
  - `resample` (Signal Processing Toolbox)
  - `shiftdata` (Signal Processing Toolbox)
  - `unshiftdata` (Signal Processing Toolbox)
  - `upfirdn` (Signal Processing Toolbox)
  - `upsample` (Signal Processing Toolbox)
  - `xspectrogram` (Signal Processing Toolbox)
- The following functions have new `gpuArray` support in Audio Toolbox:
  - `classifySound` (Audio Toolbox)
  - `crepePostprocess` (Audio Toolbox)
  - `crepePreprocess` (Audio Toolbox)
  - `detectSpeech` (Audio Toolbox)
  - `harmonicRatio` (Audio Toolbox)
  - `openl3Features` (Audio Toolbox)
  - `openl3Preprocess` (Audio Toolbox)
  - `pitchnn` (Audio Toolbox)
  - `vggishFeatures` (Audio Toolbox)
  - `vggishPreprocess` (Audio Toolbox)
  - `yamnetPreprocess` (Audio Toolbox)

For a list of all Audio Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Audio Toolbox\)](#).

## Memory Usage: Use whos to check memory used by gpuArray and distributed variables

You can now use the `whos` function to check the amount of memory used by `gpuArray` and distributed variables.

Previously, the `Bytes` variable of the output of `whos` showed the number of bytes of the pointer to the variable in the local memory of the host machine. Now, the `Bytes` variable displays the amount of GPU or distributed memory allocated to that variable. For `gpuArray` variables, `whos` displays the amount of GPU memory used by that variable. For distributed variables, `whos` displays the total memory used by that variable across all workers in the pool.

## Distributed Arrays: Use new and enhanced distributed array functionality

- `interp2`
- `linsolve`
- `makima`
- `orth`
- `pagesvd`
- `pchip`
- `subspace`
- `svdsketch`
- `decomposition` - Support for new decompositions
  - For dense matrices, new support for banded decomposition and permuted triangular decomposition
  - For sparse matrices, new support for LDL decomposition and QR decomposition
- `pagefun` - Support for `conv`, `conv2`, and `svd`

For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## Thread-Based Environment: Use new and enhanced functionality on threads for working with audio, video, and images

- The following MATLAB functions have new and enhanced thread support:
  - `imresize`
  - `audioread`
  - `audiowrite`
  - `imwrite`
  - `VideoReader`
  - `VideoWriter`
  - `imread` - Support for JPEG 2000 (J2C, J2K, JP2, JPF, JPX) format images on threads
- The following functions and objects have new thread support in Image Processing Toolbox:



- 
- `imcrop` (Image Processing Toolbox)
  - `imcrop3` (Image Processing Toolbox)
  - `imresize3` (Image Processing Toolbox)
  - `imrotate` (Image Processing Toolbox)
  - `imrotate3` (Image Processing Toolbox)
  - `imtranslate` (Image Processing Toolbox)
  - `impyramid` (Image Processing Toolbox)
  - `imwarp` (Image Processing Toolbox)
  - `affineOutputView` (Image Processing Toolbox)
  - `findbounds` (Image Processing Toolbox)
  - `fliptform` (Image Processing Toolbox)
  - `makesampler` (Image Processing Toolbox)
  - `maketform` (Image Processing Toolbox)
  - `tformarray` (Image Processing Toolbox)
  - `tformfwd` (Image Processing Toolbox)
  - `tforminv` (Image Processing Toolbox)
  - `imregconfig` (Image Processing Toolbox)
  - `imregcorr` (Image Processing Toolbox)
  - `imregdemons` (Image Processing Toolbox)
  - `imregmtb` (Image Processing Toolbox)
  - `normxcorr2` (Image Processing Toolbox)
  - `MattesMutualInformation` (Image Processing Toolbox)
  - `MeanSquares` (Image Processing Toolbox)
  - `RegularStepGradientDescent` (Image Processing Toolbox)
  - `OnePlusOneEvolutionary` (Image Processing Toolbox)
  - `cpcorr` (Image Processing Toolbox)
  - `cpstruct2pairs` (Image Processing Toolbox)
  - `fitgeotrans` (Image Processing Toolbox)
  - `imref2d` (Image Processing Toolbox)
  - `imref3d` (Image Processing Toolbox)
  - `affine2d` (Image Processing Toolbox)
  - `affine3d` (Image Processing Toolbox)
  - `projective2d` (Image Processing Toolbox)
  - `gray2ind` (Image Processing Toolbox)
  - `ind2gray` (Image Processing Toolbox)
  - `mat2gray` (Image Processing Toolbox)
  - `label2rgb` (Image Processing Toolbox)
  - `imsplit` (Image Processing Toolbox)
  - `adaptthresh` (Image Processing Toolbox)

- `otsuthresh` (Image Processing Toolbox)
- `imquantize` (Image Processing Toolbox)
- `grayslice` (Image Processing Toolbox)
- `im2int16` (Image Processing Toolbox)
- `im2single` (Image Processing Toolbox)
- `im2uint16` (Image Processing Toolbox)
- `im2uint8` (Image Processing Toolbox)

For more information, see [Run MATLAB Functions in Thread-Based Environment](#).

## Functionality Being Removed or Changed

### **parfeval and parfevalOnAll can now run in serial with no pool**

#### *Behavior change*

Starting in R2021b, you can now run `parfeval` and `parfevalOnAll` in serial with no pool. This behavior allows you to share parallel code that you write with users who do not have Parallel Computing Toolbox.

When you use the syntaxes `parfeval(fcn,n,X1,...,Xm)` or `parfevalOnAll(fcn,n,X1,...,Xm)`, MATLAB tries to use an open parallel pool if you have Parallel Computing Toolbox. If a parallel pool is not open, MATLAB will create one if automatic pool creation is enabled.

If parallel pool creation is disabled or if you do not have Parallel Computing Toolbox, the function is evaluated in serial. In previous releases, MATLAB threw an error instead.

### **mldivide and decomposition now produce the same results for distributed arrays**

#### *Behavior change*

Starting in R2021b, `mldivide` and `decomposition` now produce the same results when you run the following code using distributed arrays `A` and `b`.

```
X = A \ b;  
X = decomposition(A) \ b;
```

Previously, you sometimes saw different results when you used `decomposition` before `mldivide` for a non-square distributed matrix `A`.

### **Default behavior for decomposition has changed for distributed arrays**

#### *Behavior change*

Starting in R2021b, the algorithm for `decomposition` with type set to 'auto' (default) has changed for distributed array input. You see this behavior change when you use `decomposition` to decompose a distributed matrix that is Hermitian, banded, or permuted triangular.

When you use `decomposition(A)` or `decomposition(A, 'auto')` with a distributed matrix `A`, the type of decomposition selected by MATLAB is limited to the supported decomposition types for distributed arrays.

- For a distributed dense matrix `A`, in the syntax `decomposition(A, type)` the decomposition types 'ldl', 'cod', and 'hessenberg' are not supported.

- 
- For a distributed sparse matrix  $A$ , in the syntax `decomposition(A, type)` the decomposition types 'chol', 'cod', and 'hessenberg' are not supported.



# R2021a

---

**Version: 7.4**

**New Features**

**Compatibility Considerations**

## **GPU Functionality: Use new and enhanced gpuArray functions**

- `pagectranspose`
- `pagetranspose`
- `spline`
- `imresize` — Support for all non-sparse numeric or logical images, except categorical or indexed images. Support for 'nearest' and 'bilinear' interpolation methods. Support for 'box', 'triangle', 'cubic', 'lanczos2', 'lanczos3', and custom interpolation kernels. Support for 'AntiAliasing' name-value argument.
- `issorted` — Support for `dim` and `direction` input arguments and `ComparisonMethod` name-value option.

For more information, see [Run MATLAB Functions on a GPU](#).

## **GPU Functionality: Use new and enhanced gpuArray functions in Statistics and Machine Learning Toolbox**

- `fitcknn` (Statistics and Machine Learning Toolbox)
- `gamfit` (Statistics and Machine Learning Toolbox)
- `pca` (Statistics and Machine Learning Toolbox)
- `randsample` (Statistics and Machine Learning Toolbox)

For a list of all Statistics and Machine Learning Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Statistics and Machine Learning Toolbox\)](#).

## **GPU Functionality: Use new and enhanced gpuArray functions for working with signals, audio, and wavelets**

- The following functions have new and enhanced `gpuArray` support in Signal Processing Toolbox:
  - `binmask2sigroi` (Signal Processing Toolbox)
  - `bitrevorder` (Signal Processing Toolbox)
  - `buffer` (Signal Processing Toolbox)
  - `digitrevorder` (Signal Processing Toolbox)
  - `extendsigroi` (Signal Processing Toolbox)
  - `mergesigroi` (Signal Processing Toolbox)
  - `removesigroi` (Signal Processing Toolbox)
  - `shortensigroi` (Signal Processing Toolbox)
  - `sigroi2binmask` (Signal Processing Toolbox)
  - `sosfilt` (Signal Processing Toolbox)
  - `stftmag2sig` (Signal Processing Toolbox)

For a list of all Signal Processing Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Signal Processing Toolbox\)](#).

- The following functions have new `gpuArray` support in Wavelet Toolbox:

- 
- `appcoef` (Wavelet Toolbox)
  - `appcoef2` (Wavelet Toolbox)
  - `detcoef` (Wavelet Toolbox)
  - `detcoef2` (Wavelet Toolbox)
  - `haart` (Wavelet Toolbox)
  - `haart2` (Wavelet Toolbox)
  - `idwt` (Wavelet Toolbox)
  - `idwt2` (Wavelet Toolbox)
  - `ihaart2` (Wavelet Toolbox)
  - `ihaart` (Wavelet Toolbox)
  - `waveletScattering` (Wavelet Toolbox)
  - `waverec` (Wavelet Toolbox)
  - `waverec2` (Wavelet Toolbox)

For a list of all Wavelet Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Wavelet Toolbox\)](#).

- The following function has new `gpuArray` support in Audio Toolbox:
  - `audioFeatureExtractor` (Audio Toolbox)

For a list of all Audio Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Audio Toolbox\)](#).

## **GPU Functionality: Use new and enhanced `gpuArray` functions in Image Processing Toolbox**

- `imwarp` (Image Processing Toolbox)
- `imcrop` (Image Processing Toolbox)
- `multisim` (Image Processing Toolbox)
- `multisim3` (Image Processing Toolbox)
- `psnr` (Image Processing Toolbox)
- `wiener2` (Image Processing Toolbox)

For a list of all Image Processing Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Image Processing Toolbox\)](#).

## **Support for NVIDIA CUDA 11.0: Update to CUDA Toolkit 11.0**

The parallel computing products are now using CUDA toolkit version 11.0. To generate CUDA kernel objects from CU code or compile CUDA compatible source code, libraries, and executables using GPU Coder, you must use toolkit version 11.0. For more information, see [GPU Support by Release](#).

## **Distributed Arrays: Use new and enhanced distributed array functionality**

- `colon`
- `ichol`
- `interp1`
- `pagectranspose`
- `pagetimes`
- `pagetranspose`

For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## **GPU Devices: Count and compare available GPU devices**

Use new functionality to count and compare GPU devices in your local system

- `gpuDeviceCount` — Count all detected devices, all supported devices, or only devices available in your current session.
- `gpuDeviceTable` — Compare the properties of all local GPU devices detected in your system.

## **Thread-Based Parallel Pool: Use new and enhanced functionality on thread workers**

- `imageDatastore`
- `imread`
- `parallel.pool.DataQueue`
- `parallel.pool.PollableDataQueue`
- Use thread workers in standalone applications created with MATLAB Compiler™ and web apps hosted on MATLAB Web App Server™.

For more information, see [Run MATLAB Functions on Thread Workers](#).

## **parfor Examples: Use a parallel pool to speed up Monte-Carlo code**

Use this new example to learn how to speed up your Monte-Carlo code using a `parfor`-loop. For more information, see [Use parfor to Speed Up Monte-Carlo Code](#).

## **Upgrade Parallel Computing Products Together**

As with every new release, if you upgrade Parallel Computing Toolbox, you must upgrade MATLAB Parallel Server at the same time so that they interact properly with each other.

## **Compatibility Considerations**

If you are using MATLAB Job Scheduler, the backward compatibility feature allows you to connect to multiple versions of MATLAB Parallel Server in your cluster. For more information, see [Run Multiple MATLAB Parallel Server Versions \(MATLAB Parallel Server\)](#).



---

Jobs created in one version of Parallel Computing Toolbox software do not run in a different version of MATLAB Parallel Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` might not be compatible between different versions of MATLAB Parallel Server. You must specify a different `JobStorageLocation` for each parallel computing product, and each version on your cluster must have its own `JobStorageLocation`.

## Functionality Being Removed or Changed

### Support for cc3.0 and cc3.2 Kepler GPUs is removed

#### *Errors*

Starting in R2021a, support for Kepler GPU architectures with compute capability 3.0 and 3.2 is removed. Using a GPU with MATLAB requires a GPU device with compute capability 3.5 or greater. Using a GPU device with compute capability 6.0 or greater is recommended.

For more information about supported GPU devices, see [GPU Support by Release](#).

### Functions offloaded with `batch` now evaluate cell array input arguments `{C1,...,Cn}` as `C1,...,Cn`

#### *Behavior change*

Starting in R2021a, a function `fcn` offloaded with `batch` evaluates cell array input arguments `{C1,...,Cn}` as `fcn(C1,...,Cn)`. In previous releases `{C1,...,Cn}` threw an error and `{{C1,...,Cn}}` was evaluated as `fcn(C1,...,Cn)`.

Starting in R2021a, use the following code to offload `fcn({a,b},{c,d})` on the cluster `myCluster` with one output.

```
batch(myCluster,@fcn,1,{{a,b},{c,d}});
```

In previous releases, you used the following code instead.

```
batch(myCluster,@fcn,1,{{{a,b},{c,d}}});
```



# R2020b

---

**Version: 7.3**

**New Features**

**Compatibility Considerations**

## GPU Functionality: Use new and enhanced gpuArray functions

- `isgpuarray`
- `isoutlier`
- `pagetimes`
- `rmissing`
- `rmoutliers`
- Message functions `assert`, `error`, and `warning` (these functions gather `gpuArray` data and run on the CPU)
- `arrayfun` — Support for `cast` using 'like' syntax
- `qr` — Support for one- and three-argument syntaxes

For more information, see [Run MATLAB Functions on a GPU](#).

## GPU Functionality: Use new and enhanced gpuArray functions in Statistics and Machine Learning Toolbox

- `fitlm` (Statistics and Machine Learning Toolbox)
- `fitglm` (Statistics and Machine Learning Toolbox)
- `glmfit` (Statistics and Machine Learning Toolbox)
- `glmval` (Statistics and Machine Learning Toolbox)
- `grp2idx` (Statistics and Machine Learning Toolbox)

For a list of all Statistics and Machine Learning Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Statistics and Machine Learning Toolbox\)](#).

## GPU Functionality: Use new and enhanced gpuArray functions for working with signals, audio, and wavelets

- The following functions have new and enhanced `gpuArray` support in Signal Processing Toolbox:
  - `cpsd` (Signal Processing Toolbox)
  - `fsst` (Signal Processing Toolbox)
  - `goertzel` (Signal Processing Toolbox)
  - `istft` (Signal Processing Toolbox)
  - `mscohere` (Signal Processing Toolbox)
  - `periodogram` (Signal Processing Toolbox)
  - `pwelch` (Signal Processing Toolbox)
  - `spectrogram` (Signal Processing Toolbox) — Support for nonuniformly spaced frequencies

For a list of all Signal Processing Toolbox functions with GPU functionality, see [Functions with gpuArray support \(Signal Processing Toolbox\)](#).

- The following functions have new `gpuArray` support in Wavelet Toolbox:
  - `dwt` (Wavelet Toolbox)

- 
- `dwt2` (Wavelet Toolbox)
  - `dyadup` (Wavelet Toolbox)
  - `dyaddown` (Wavelet Toolbox)
  - `timeSpectrum`
  - `scaleSpectrum`
  - `wavedec` (Wavelet Toolbox)
  - `wavedec2` (Wavelet Toolbox)
  - `wcoherence` (Wavelet Toolbox)
  - `wextend` (Wavelet Toolbox)
  - `wkeep` (Wavelet Toolbox)

For a list of all Wavelet Toolbox functions with GPU functionality, see [Functions with gpuArray support](#) (Wavelet Toolbox).

- The following functions have new `gpuArray` support in Audio Toolbox:
  - `audioDelta` (Audio Toolbox)
  - `cepstralCoefficients` (Audio Toolbox)
  - `shiftPitch` (Audio Toolbox)
  - `spectralCentroid` (Audio Toolbox)
  - `spectralCrest` (Audio Toolbox)
  - `spectralDecrease` (Audio Toolbox)
  - `spectralEntropy` (Audio Toolbox)
  - `spectralFlatness` (Audio Toolbox)
  - `spectralFlux` (Audio Toolbox)
  - `spectralKurtosis` (Audio Toolbox)
  - `spectralRolloffPoint` (Audio Toolbox)
  - `spectralSkewness` (Audio Toolbox)
  - `spectralSlope` (Audio Toolbox)
  - `spectralSpread` (Audio Toolbox)
  - `stretchAudio` (Audio Toolbox)

For a list of all Audio Toolbox functions with GPU functionality, see [Functions with gpuArray support](#) (Audio Toolbox).

## **GPU Communications: Fast data transfer between GPUs in a parallel pool**

Data transfer between GPUs in a parallel pool now uses fast peer-to-peer communication, including NVLink, if available. Fast data transfer takes place when you send `gpuArray` data using the following functions:

- `labSend`
- `labReceive`

- `labBroadcast`
- `gop`

## Support for NVIDIA CUDA 10.2: Update to CUDA Toolkit 10.2

The parallel computing products are now using CUDA toolkit version 10.2. To compile CUDA code for `CUDAKernel` or CUDA MEX-files, you must use toolkit version 10.2. For more information, see [GPU Support by Release](#).

## Distributed Arrays: Use new and enhanced distributed array functionality

- `filter`
- `mpower`
- `pagefun`
- `rmissing`

For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## Tall Arrays: Use new and enhanced tall array functionality in Signal Processing Toolbox

- `pwelch` (Signal Processing Toolbox)

For a list of all Signal Processing Toolbox functions with support for tall arrays, see [Functions with tall support](#) (Signal Processing Toolbox).

## Array Assignment: Assign `gpuArray` or distributed data directly into existing MATLAB arrays

You can now assign values stored in a `gpuArray`, distributed array, or codistributed array into a MATLAB array directly, without first having to gather the data. In assignment statements such as `A(1:k) = C`, where `A` has a built-in data type such as `double`, MATLAB attempts to convert `C` to the same data type as `A`. That conversion behavior has changed.

For example, the following code now runs.

```
x = rand(3,3);  
y = rand(1,'gpuArray');  
x(2,2) = y;
```

## Compatibility Considerations

Some assignment statements that used to throw an error now execute. If your code relied on the errors that MATLAB threw for those conversions, such as within a `try/catch` block, then your code might no longer catch those errors.

---

## Distributed Tables: Assign data directly into existing distributed tables

You can now assign values directly into a distributed table. In assignment statements such as `distTable.Score(1:k) = C`, where `distTable.Score` has a built-in data type such as `double`, MATLAB attempts to convert `C` to the same data type as `distTable.Score`. That conversion behavior has changed.

For example, the following code now runs.

```
table = array2table(rand(2), 'VariableNames', {'ID', 'Score'});
distTable = distributed(table);
distTable.Score(1:2) = [1 2];
```

## Compatibility Considerations

Some assignment statements that used to throw an error now execute. If your code relied on the errors that MATLAB threw when assigning values into a distributed table, such as within a `try/catch` block, then your code might no longer catch those errors.

## Thread-Based Parallel Pool: Use a gpuArray on thread workers

You can now use a `gpuArray` on thread workers. Any functionality with both GPU and `ThreadPool` support now supports GPUs in a `ThreadPool`. For more information, see [Check Support for Thread-Based Environment](#).

## parfeval Examples: Explore the state of futures and cancel them

Use this new example to learn how to query the state of `parfeval` futures and cancel them. For more information, see [Query and Cancel parfeval Futures](#).

## HTCondor integration: Plugin script for HTCondor now available as an Add-On

Integrate the third party scheduler HTCondor and MATLAB via the generic scheduler interface using an easy-to-configure plugin script. To download the plugin script, use the [Add-On Explorer](#). Alternatively, you can download the plugin from the [File Exchange](#). For more information, see [Configure Using the Generic Scheduler Interface \(MATLAB Parallel Server\)](#).

## Parallel Workflows: Compare performance of different parallel environments

Use new examples to determine the best parallel programming environment to use in your workflows:

- [Compare Performance of Multithreading and ProcessPool](#)
- [Compare Performance of parfor, parfeval, and spmd](#)

## Query Underlying Data: Query the underlying data type of classes

You can now use the following functions to query the underlying data type of data stored in `gpuArray` objects, distributed arrays, `dlarray` objects, and more:

- `underlyingType`
- `isUnderlyingType`
- `mustBeUnderlyingType`

The `class` function is useful to determine the class of a variable. However, some classes in MATLAB can contain underlying data that has a different type compared to what `class` returns. Example classes include `gpuArray`, `dlarray`, and distributed arrays. The `underlyingType`, `isUnderlyingType`, and `mustBeUnderlyingType` functions now provide a simple way to query the underlying data types of those classes.

For most classes, `class(X)` and `underlyingType(X)` return the same answer. However, for classes that can contain underlying data of a different type, `class(X)` returns the name of the class (such as `gpuArray`), and `underlyingType(X)` returns the underlying MATLAB data type that determines how the array `X` behaves (such as `double`).

## Query Parallel Functionality: Query if support for Parallel Computing Toolbox functionality is available

You can now query if support for GPU and parallel pool functionality is available in your MATLAB installation using the following functions:

- `canUseGPU`
- `canUseParallelPool`

Use these functions to check supported functionality and avoid executing code that relies on specific hardware constraints.

## Improved Scalability: Use MATLAB Job Scheduler clusters with up to 4000 workers

MATLAB Parallel Server with the MATLAB Job Scheduler now supports clusters up to 4000 workers. Support for large parallel pools remains at 1024 workers.

When you scale above 1000 workers, you must increase the heap memory available to the job manager. For more information, see [Customize Startup Parameters \(MATLAB Parallel Server\)](#).

## Reference Architectures: Schedule jobs to run in AWS Batch

Use the MATLAB Parallel Server with AWS Batch reference architecture to build your own MATLAB Parallel Server solution for batch processing jobs using AWS Batch. AWS Batch dynamically provisions and manages Amazon EC2<sup>®</sup> instances based on the volume and resource requirements of the submitted jobs. AWS Batch supports both On-Demand and Spot Amazon EC2 instances.

To connect to the cluster from your MATLAB client, install the Parallel Computing Toolbox plugin for MATLAB Parallel Server with AWS Batch.



---

## Docker Containers: Build a container image with a customized MATLAB installation

Use the reference Dockerfile to build a Docker container image that runs a custom MATLAB installation. You can choose the toolboxes you require in your custom installation. Containers are a scalable and reproducible method to deploy MATLAB in cloud and server environments. To access the Dockerfile and create a custom MATLAB container image, see [Create a MATLAB Container Image](#).

## labSend, labReceive, labBroadcast and gop Functions: Improved performance of data transfer between GPUs in a parallel pool

The `labSend`, `labReceive`, `labBroadcast` and `gop` functions show improved performance when transferring data in a parallel pool between MATLAB workers with GPUs. Data transfer between GPUs in a parallel pool now uses fast peer-to-peer communication, including NVLink, if available. For example, using the `labSend` and `labReceive` functions to transfer 72 MB data between two GPUs is approximate 5.1x faster than in the previous release.

```
% Set up parallel pool
parpool('local',2);

% Clear all GPUs
spmd
    gpuDevice([]);
end

% Set sender/receiver worker index
senderIndex = 1;
receiverIndex = 2;

% Create data
spmd
    devices = gpuDevice();
    if labindex == senderIndex
        gpuData = ones(3000,3000,'gpuArray');
    end
    wait(devices);
end

% Measure time to transfer data 20 times
tic;
for j=1:20
    spmd
        if labindex == senderIndex
            labSend(gpuData,receiverIndex);
        elseif labindex == receiverIndex
            newGpuData = labReceive(senderIndex);
        end
        wait(devices);
    end
end
toc

% Delete parallel pool
delete(gcf('nocreate'))
```

The approximate execution times are:

**R2020a:** 9.50 seconds

**R2020b:** 1.85 seconds

The code was timed on a Windows 10, Intel Xeon E5-2623 v4 @ 2.60 GHz test system with four NVIDIA Titan V 12GB GPUs by running the above script.

## Upgrade Parallel Computing Products Together

As with every new release, if you upgrade Parallel Computing Toolbox, you must upgrade MATLAB Parallel Server at the same time so that they interact properly with each other.

## Compatibility Considerations

If you are using MATLAB Job Scheduler, the backward compatibility feature allows you to connect to multiple versions of MATLAB Parallel Server in your cluster. For more information, see [Run Multiple MATLAB Parallel Server Versions \(MATLAB Parallel Server\)](#).

Jobs created in one version of Parallel Computing Toolbox software do not run in a different version of MATLAB Parallel Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` might not be compatible between different versions of MATLAB Parallel Server. You must specify a different `JobStorageLocation` for each parallel computing product, and each version on your cluster must have its own `JobStorageLocation`.

## Functionality Being Removed or Changed

### GPU acceleration is not available on macOS

*Errors*

Starting in R2020b, GPU acceleration, including `gpuArray` and CUDA functionality, is not supported on macOS platforms. Using `gpuArray` or CUDA functionality on macOS platforms results in an error.

For more information on platforms supported by MATLAB, see [Platform Road Map for MATLAB and Simulink](#). For more information on GPU support for macOS, see [Can I use MATLAB with an NVIDIA GPU on macOS 10.14 Mojave?](#)

### Forward compatibility for GPU devices is disabled by default

*Behavior change*

Starting in R2020b, forward compatibility for GPU devices is disabled by default. In previous releases, forward compatibility for GPU devices is enabled and cannot be disabled.

When forward compatibility is disabled, you cannot perform computations using a GPU device with an architecture that was released after the version of MATLAB you are using was built.

To reproduce the behaviour of earlier MATLAB versions, enable forward compatibility using the `parallel.gpu.enableCUDAForwardCompatibility` function or by setting the environment variable `MW_CUDA_FORWARD_COMPATIBILITY` to 1.

For more information, see [Forward Compatibility for GPU Devices](#).

### `classUnderlying` and `isaUnderlying` are not recommended

*Still runs*

`classUnderlying` and `isaUnderlying` are not recommended. Use `underlyingType` and `isUnderlyingType` instead.

# R2020a

---

**Version: 7.2**

**New Features**

**Compatibility Considerations**

## **Parallel profiling: Learn tips and techniques to profile parallel code with new documentation**

Use this new example to learn how to profile your parallel code and find out what functions parallel pool workers are spending most of the time on. For more information, see [Profile Parallel Code](#).

## **GPU Functionality: Use new and enhanced gpuArray functions**

- `mldivide`: Support for batch operations on rectangular matrices.
- `tril`: Support for batch operations.
- `triu`: Support for batch operations.
- `qmr`
- `tfqmr`

For more information, see [Run MATLAB Functions on a GPU](#).

## **GPU Functionality: Use new and enhanced gpuArray functions in Statistics and Machine Learning Toolbox**

Over 140 functions in Statistics and Machine Learning Toolbox have support for `gpuArrays`. Functions with new `gpuArray` support include:

- `corr`
- `random`

For a full list of functions with new and enhanced GPU functionality, see *Release Notes for Statistics and Machine Learning Toolbox* (Statistics and Machine Learning Toolbox).

## **GPU Functionality: New gpuArray support for spectral functions**

Several spectral functions in Audio Toolbox, Signal Processing Toolbox, and Wavelet Toolbox now support `gpuArrays`. The following functions now have `gpuArray` support:

- `mfcc` (Audio Toolbox)
- `melSpectrogram` (Audio Toolbox)
- `czt` (Signal Processing Toolbox)
- `spectrogram` (Signal Processing Toolbox)
- `stft` (Signal Processing Toolbox)
- `wvd` (Signal Processing Toolbox), `wvd` (Wavelet Toolbox)
- `cwt` (Wavelet Toolbox)
- `cwtfilterbank` and `wt` object function (Wavelet Toolbox)

## **Job Arrays: Submit job arrays to third-party schedulers with the generic scheduler interface**

Now MATLAB leverages job arrays by default when you submit jobs to Slurm, LSF®, PBS Pro®, and Grid Engine using the generic scheduler interface. For more information on integrating a third-party

---

scheduler with MATLAB and the generic scheduler interface, see [Integrate MATLAB with Third-Party Schedulers \(MATLAB Parallel Server\)](#) and [Configure Using the Generic Scheduler Interface \(MATLAB Parallel Server\)](#).

## **Distributed Arrays: Use new and enhanced distributed array functionality**

- `matches`
- `ilu`
- `del2`
- `inpolygon`
- `polyfit`
- `polyval`
- `renamevars`
- `write`: support for new name-value pairs `'VariableEncoding'` and `'Version'` for writing Parquet files.
- `subsasgn`: support for indexed deletion.

For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## **New Thread-Based Parallel Pool: Optimized for reduced memory usage, faster scheduling, and less data transfer, for a subset of MATLAB functions**

You can now run parallel language features on a thread-based parallel pool. Thread-based environments are optimized for reduced memory usage, faster scheduling, and less data transfer. These environments are an alternative to the existing family of process-based environments.

Thread workers support a subset of the MATLAB functions available for process workers. If you are interested in a function that is not supported, let the MathWorks Technical Support team know.

The features that thread workers support are `parpool`, `parfor`, `parfeval`, `tall`, and `parallel.pool.Constant`. Features that are not supported include `spmd`, `distributed`, and `parallel.pool.DataQueue`.

In general, many core features of MATLAB are supported, including:

- Language fundamentals
- Mathematics
- Core data types (double, single, logical, integer types, char, cell arrays, string, table, timetable, categorical, datetime, and duration)
- Control flow and logic (for example, `if`, `for` loops, and `while` loops)
- Scripts, functions and classes
- Custom classes

In general, features that modify or access things outside of the thread worker are not supported, including:

- Data import and export
- Graphics
- External languages

For more information, see [Choose Between Thread-Based and Process-Based Environments](#).

## **MATLAB Online: Use Parallel Computing Toolbox with Cloud Center clusters in MATLAB Online**

You can now use Parallel Computing Toolbox in MATLAB Online with Cloud Center clusters. Access MATLAB Online from here: <https://matlab.mathworks.com>. To learn more about the specific capabilities and the differences from MATLAB Desktop, see [Use Parallel Computing Toolbox with Cloud Center clusters in MATLAB Online](#).

## **Upgrade Parallel Computing Products Together**

As with every new release, if you upgrade Parallel Computing Toolbox you must upgrade MATLAB Parallel Server at the same time so that they interact properly with each other.

## **Compatibility Considerations**

If you are using MATLAB Job Scheduler, the backward compatibility feature allows you to connect to multiple versions of MATLAB Parallel Server in your cluster. For more information, see [Run Multiple MATLAB Parallel Server Versions \(MATLAB Parallel Server\)](#).

Jobs created in one version of Parallel Computing Toolbox software do not run in a different version of MATLAB Parallel Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` (formerly `DataLocation`) might not be compatible between different versions of MATLAB Parallel Server. Therefore, `JobStorageLocation` should not be shared by parallel computing products running different versions, and each version on your cluster should have its own `JobStorageLocation`.

## **Functionality Being Removed or Changed**

### **GPU acceleration is not available on macOS 10.14 Mojave and later versions**

*Errors*

GPU acceleration, including `gpuArray` and CUDA functionality, is not supported on macOS 10.14 Mojave and later versions. Using `gpuArray` or CUDA functionality on macOS 10.13 High Sierra and earlier versions results in a warning.

For more information, see [Can I use MATLAB with an NVIDIA GPU on macOS 10.14 Mojave?](#).

### **Support for Kepler and Maxwell GPUs will be removed**

*Warns*

In a future release, support for Kepler and Maxwell GPU architectures will be removed. At that time, using a GPU with MATLAB will require a GPU device with compute capability 6.0 or greater. Pascal, Volta, and Turing architectures are supported GPU architectures with compute capability 6.0 or greater.

---

In R2020a, Kepler and Maxwell GPUs are still supported. MATLAB generates a warning the first time you use a Kepler or Maxwell GPU.

For more information about supported GPU devices, see GPU Support by Release.

**pmode will be removed**

*Warns*

In a future release, the `pmode` function will be removed. To execute commands interactively on multiple workers, use `spmd` instead.

**pload and psave will be removed**

*Warns*

In a future release, the `psave` and `pload` functions will be removed. To save and load data on the workers, in the form of Composite arrays or distributed arrays, use `dsave` and `dload` instead.





# R2019b

---

**Version: 7.1**

**New Features**

**Compatibility Considerations**

## **GPU Functionality: Use new and enhanced gpuArray functions**

- `min`, `max`: Support for the 'linear' option.
- `diag`, `trace`: Support for sparse gpuArrays.
- `times`, `.*`: Support for sparse gpuArrays.

For more information, see [Run MATLAB Functions on a GPU](#).

## **Support for NVIDIA CUDA 10.1: Update to CUDA Toolkit 10.1**

The parallel computing products are now using CUDA Toolkit version 10.1. To compile CUDA code for `CUDAKernel` or CUDA MEX-files, you must use toolkit version 10.1. For more information, see [GPU Support by Release](#).

## **gpuArray: Load gpuArray data when no GPU is available**

You can now load MAT files containing gpuArray data as in-memory arrays when a GPU is not available. A gpuArray object loaded without a GPU is limited and you cannot use it for computations. To use a gpuArray loaded without a GPU, retrieve the contents with `gather`.

## **mexcuda: Compile mexcuda functions without installing the CUDA toolkit**

You can now use `mexcuda` to compile CUDA code without installing the CUDA toolkit. If the CUDA toolkit is not detected or is not a supported version, MATLAB compiles the CUDA code using the NVIDIA `nvcc` compiler installed with MATLAB. To check which compiler `mexcuda` is using, use the `-v` flag for verbose output in the `mexcuda` command.

## **Distributed Arrays: Use new and enhanced distributed array functionality**

- `decomposition`
- `min`, `max`: Support for the 'linear' option.
- `mink`, `maxk`, `topkrows`
- `issorted`, `issortedrows`
- `distributed`: Support for creating a distributed array from a Composite array.
- Support for accessing whole rows or columns in 2-D block-cyclic distributed arrays.

For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## **Distributed Arrays Examples: Explore multigrid preconditioned iterative solvers with new documentation**

Use this new example to learn how to use a multigrid preconditioned iterative solver with distributed arrays. For more information, see [Solve Differential Equation Using Multigrid Preconditioner on Distributed Discretization](#).

---

## **Parallel Deep Learning Workflows: Explore deep learning with custom parallel training loops**

Use this new example to learn how to set up custom training loops to train deep learning networks in parallel. For more information, see [Train Network in Parallel with Custom Training Loop](#).

## **Batch Jobs Examples: Explore batch workflows with new documentation**

Use these new examples to learn about batch jobs and options:

- [Run Script as Batch Job](#)
- [Run Batch Job and Access Files from Workers](#)

## **parfeval Examples: Explore parfeval workflows with new documentation**

Use these new examples to learn more about parfeval workflows:

- [Plot During Parameter Sweep with parfeval](#)
- [Perform Webcam Image Acquisition in Parallel with Postprocessing](#)
- [Perform Image Acquisition and Parallel Image Processing](#)

## **Image Acquisition Examples: Perform parallel acquisition and processing of images**

Use these new examples to learn how to implement these parallel workflows:

- [Perform Webcam Image Acquisition in Parallel with Postprocessing](#)
- [Perform Image Acquisition and Parallel Image Processing](#)

## **Improved Scalability: Use MATLAB Job Scheduler clusters with up to 2000 workers**

MATLAB Parallel Server with the MATLAB Job Scheduler now supports clusters up to 2000 workers. Support for large parallel pools remains at 1024 workers.

## **Third-Party Schedulers: Check the scheduler ID of a MATLAB task**

If you submit a task to a third-party scheduler, then you can check its ID on the scheduler by looking at the `SchedulerID` property of the task object. For more information, see [CJS Tasks](#). To get the scheduler IDs of all tasks in a job, use `getTaskSchedulerIDs`.

## **Common Job Schedulers: Improved performance of vectorized task creation**

Local and third-party schedulers benefit from improved task creation rates and submission times when you use `createTask` for vectorized task creation or `parfor` without a parallel pool.

For example, the following functions show about a 75x speed-up for task creation and a 6x speed-up for job submission.

```
function taskCreationTimingTest
    c = parcluster('local');
    j = createJob(c);
    numTasks = 10000;
    allTaskArgs = cell(1,numTasks);
    for ii = 1:numTasks
        ithTaskArgs = {ii}; % Arguments to the ith task
        allTaskArgs{ii} = ithTaskArgs;
    end
    tic;
    createTask(j,@rand,1,allTaskArgs);
    toc
end

function jobSubmissionTimingTest
    c = parcluster('local');
    j = createJob(c);
    numTasks = 10000;
    allTaskArgs = cell(1,numTasks);
    for ii = 1:numTasks
        ithTaskArgs = {ii}; % Arguments to the ith task
        allTaskArgs{ii} = ithTaskArgs;
    end
    createTask(j,@rand,1,allTaskArgs);
    tic;
    submit(j);
    toc
end
```

The approximate times are:

- R2019a: 296 s for task creation and 150 s for job submission.
- R2019b: 3.9 s for task creation and 24 s for job submission.

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system using the `timeit` function:

```
timeit(@taskCreationTimingTest)
timeit(@jobSubmissionTimingTest)
```

## Personalized Clusters: Set Cloud Center clusters for personal use

Now Cloud Center clusters can have two levels of authorization: for personal use or sharable. For more information, see [Create Cloud Cluster](#).

## Upgrade Parallel Computing Products Together

As with every new release, if you upgrade Parallel Computing Toolbox you must upgrade MATLAB Parallel Server at the same time so that they interact properly with each other.

---

## Compatibility Considerations

If you are using MATLAB Job Scheduler, the backward compatibility feature allows you to connect to multiple versions of MATLAB Parallel Server in your cluster. For more information, see [Run Multiple MATLAB Parallel Server Versions \(MATLAB Parallel Server\)](#).

Jobs created in one version of Parallel Computing Toolbox software do not run in a different version of MATLAB Parallel Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` (formerly `DataLocation`) might not be compatible between different versions of MATLAB Parallel Server. Therefore, `JobStorageLocation` should not be shared by parallel computing products running different versions, and each version on your cluster should have its own `JobStorageLocation`.

## Functionality Being Removed or Changed

### **Toolbox folder renamed from `distcomp` to `parallel`**

*Behavior change*

In R2019b, the Parallel Computing Toolbox folder has been renamed. The new name for the toolbox folder is `parallel`. If you need to reference the location of the toolbox, update your references to use `toolbox/parallel` instead of `toolbox/distcomp`.

### **`pmode` will be removed**

*Still runs*

In a future release, the `pmode` function will be removed. To execute commands interactively on multiple workers, use `spmd` instead.

### **`pload` and `psave` will be removed**

*Still runs*

In a future release, the `psave` and `pload` functions will be removed. To save and load data on the workers, in the form of Composite arrays or distributed arrays, use `dsave` and `dload` instead.



# R2019a

---

**Version: 7.0**

**New Features**

**Compatibility Considerations**

## **parfor-Loops: Use parfor without a parallel pool**

`parfor` can now run without a parallel pool of workers. If you choose this approach, `parfor` uses the available cluster resources as needed. You can also control options, such as the worker load, with `parforOptions`.

## **Parallel Deep Learning Workflows: Explore deep learning with multiple-GPUs**

Use these new examples to learn how to use multiple GPUs to train a single or several deep learning networks in parallel.

- Train Network Using Automatic Multi-GPU Support
- Run Multiple Deep Learning Experiments

## **Benchmarks: Use new examples to evaluate the performance of your cluster**

Use the computational tests in these benchmarks to measure the performance of your compute cluster.

- Benchmark Cluster Workers
- Benchmark Your Cluster with the HPC Challenge

## **parpool Resiliency: Parallel pools are now resilient to dropped network connections**

Parallel pools are now robust to network failures, and can recover from connectivity issues. For example, if your network connection drops, MATLAB attempts to reconnect to the workers in the pool. There is no loss of data if the disconnection happens during data transfer.

## **Automatic Cluster Resizing: Resize Cloud Center clusters on Amazon based on usage**

You can now create cloud clusters that resize automatically based on usage. These clusters grow or shrink to allocate the optimal number of workers for your submitted tasks. For more information, see the MathWorks Cloud Center documentation.

## **Custom Datastore: Read from Hadoop based databases using the custom datastore framework**

Author a custom database datastore and access data from a database of your choice using these extensions:

- `matlab.io.datastore.HadoopLocationBased` - Use this mixin to specify the location of your data in Hadoop®.
- `matlab.io.datastore.HadoopInput` - Use this helper class when a Hadoop java class is available for a database library.



---

Perform big data analysis on your custom datastore with tall arrays or mapreduce. These custom datastores can leverage the location of the data to make computations more efficient.

For more information on the custom datastore framework, see [Develop Custom Datastore \(MATLAB\)](#).

## **GPU Functionality: Use new and enhanced gpuArray functions**

- `sinpi`
- `cospi`
- `normalize`
- `validateattributes`

For more information, see [Run MATLAB Functions on a GPU](#).

## **Support for NVIDIA CUDA 10.0: Update to CUDA Toolkit 10.0**

The parallel computing products are now using CUDA Toolkit version 10.0. To compile CUDA code for `CUDAKernel` or `CUDA MEX`-files, you must use toolkit version 10.0. For more information, see [GPU Support by Release](#).

## **Distributed Arrays: Use new and enhanced distributed array functionality**

- `sinpi`
- `cospi`
- `normalize`
- `validateattributes`
- `write`: Support for writing to Parquet files
- `eig`: Support for non-symmetric matrices
- `sprandsym`
- Distributed iterative solvers, such as `pcg` or `gmres`, now allow arbitrary matrices as preconditioners

For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## **Renamed Product: MATLAB Distributed Computing Server renamed to MATLAB Parallel Server**

MATLAB Distributed Computing Server™ now has the name MATLAB Parallel Server. To learn more about MATLAB Parallel Server, see [MATLAB Parallel Server](#).

## **Support for MPICH: Update to MPICH Version 3 on Linux for third-party schedulers**

The parallel computing products are now shipping MPICH version 3.2.1 on Linux for third-party schedulers. Communicating jobs for third-party schedulers now use the Hydra process manager.

## Compatibility Considerations

If you use your own MPI builds, you might need to create new builds compatible with this latest version. For instructions, see [Use Different MPI Builds on UNIX Systems \(MATLAB Parallel Server\)](#).

## Upgrade Parallel Computing Products Together

As with every new release, if you upgrade Parallel Computing Toolbox you must upgrade MATLAB Parallel Server at the same time so that they interact properly with each other.

## Compatibility Considerations

If you are using MATLAB Job Scheduler, the backward compatibility feature allows you to connect to multiple versions of MATLAB Parallel Server in your cluster. For more information, see [Run Multiple MATLAB Parallel Server Versions \(MATLAB Parallel Server\)](#).

Jobs created in one version of Parallel Computing Toolbox software do not run in a different version of MATLAB Parallel Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` (formerly `DataLocation`) might not be compatible between different versions of MATLAB Parallel Server. Therefore, `JobStorageLocation` should not be shared by parallel computing products running different versions, and each version on your cluster should have its own `JobStorageLocation`.

## Functionality Being Removed or Changed

### Default random number generator changed for parallel contexts

*Behavior change*

Starting in R2019a, the default random number generator for parallel computations is changed to `Threefry`. This change applies to calculations on parallel workers, GPU arrays, distributed arrays, and tall arrays. This generator offers performance enhancements for parallel calculations over the previous default. In releases up to and including R2018b, the default random number generator for parallel computations is `CombRecursive`.

With a different default generator, MATLAB generates different random number sequences by default in the context of parallel computations. However, the statistics of these calculations remain unaffected. Therefore, you might want to update any code that relies on the specific random numbers being generated, but most calculations on the random numbers are unaffected.

To set the generator to the settings used by default in R2018b and earlier on parallel workers, GPU arrays, and tall arrays, use the following commands.

Calculations on parallel workers and distributed arrays:

```
spmd
    stream = RandStream.create("CombRecursive", "NumStreams", 2^32,...
        "StreamIndices", 2*labindex);
    RandStream.setGlobalStream(stream);
end
```

Calculations on GPU arrays:

```
gpurng(0, "CombRecursive")
```

---

Calculations on tall arrays:

```
tallrng(0, "CombRecursive")
```

**Toolbox folder renamed from distcomp to parallel**

*Behavior change in future release*

In R2019b, the Parallel Computing Toolbox toolbox folder will be renamed. The new name for the toolbox folder is `parallel`. If you need to reference the location of the toolbox, update your references to use `toolbox/parallel` instead of `toolbox/distcomp`.



# R2018b

---

**Version: 6.13**

**New Features**

**Compatibility Considerations**

## **GPU Support: View details of GPU support on over 600 function pages, and browse GPU support for functions by toolbox**

Consult additional GPU usage information on function pages of GPU-enabled functions in MATLAB and other toolboxes. You can find this information in the Extended Capabilities section at the end of the function page. Also, browse GPU function lists filtered by product. For more information, see [Run MATLAB Functions on a GPU](#).

## **GPU Functionality: Use new and enhanced gpuArray functions, such as spline interpolation and vecnorm**

- `interp1`: support for 'spline' interpolation method
- `vecnorm`
- `norm` for sparse gpuArrays: support for all vector norms; and 1, Inf, and frobenius matrix norms
- Reduction functions (`min`, `max`, `sum`,...): support for reducing multiple dimensions simultaneously
- `nthroot`: support for logical inputs
- `sign`: support for logical inputs

For more information, see [Run MATLAB Functions on a GPU](#).

## **Support for NVIDIA CUDA 9.1: Update to CUDA Toolkit 9.1**

The parallel computing products are now using CUDA Toolkit version 9.1. To compile CUDA code for CUDAKernel or CUDA MEX-files, you must use toolkit version 9.1. For more information, see [GPU Support by Release](#).

## **GPU Optimizations: Enhanced support and optimizations for GPU**

- `gpuDevice` and `gpuDeviceCount` now run faster if you do not have a GPU, because they do not load the GPU libraries. You can use these functions to check if a setup supports GPUs or not.
- GPU memory management is now better optimized. Benefit from improved performance in memory intensive GPU applications, such as deep learning.

## **GPU Examples: Explore GPU workflows on single and multiple GPUs**

New and updated GPU documentation now includes:

- [Use MATLAB Functions with a GPU](#)
- [Identify and Select a GPU Device](#)
- [Sharpen an Image Using the GPU](#)
- [Run MATLAB Functions on Multiple GPUs](#)
- [Use Multiple GPUs in a Parallel Pool](#)
- [gpuArray](#)
- [Establish Arrays on a GPU](#)
- [GPU Support by Release](#)

---

## **Distributed Arrays: Use new and enhanced distributed array functionality, including support for vecnorm and writing to Amazon S3 and Azure**

- `vecnorm`
- `write`: support for writing to XLS, CSV, and TXT formats
- `write`: extended support for writing SEQ and MAT formats to all supported file systems
- `write`: support for writing to S3 and WASBS filesystems
- Reduction functions (`min`, `max`, `sum`,...): support for reducing multiple dimensions simultaneously
- `nthroot`: support for logical inputs
- `sign`: support for logical inputs
- Distributed timetable support
- `mldivide`: more robust sparse distributed system solve

For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## **Distributed Support: View details of distributed array support on over 400 function pages, and browse distributed support for functions by toolbox**

Consult additional distributed array usage information on function pages of distributed-enabled functions in MATLAB and other toolboxes. You can find this information in the Extended Capabilities section at the end of the function page. Also, browse distributed array function lists filtered by product. For more information, see [Run MATLAB Functions with Distributed Arrays](#).

## **Parallel Extended Capabilities: View details of automatic parallel support on function pages, and browse support for functions by toolbox**

Many functions in MATLAB, Simulink, and toolboxes help you take advantage of parallel computing resources without requiring any extra coding. You can enable this support by simply setting a flag or preference.

Now you can consult automatic parallel usage information on function pages and browse supported function lists filtered by product.

Similarly, you can view lists of functions with extended capabilities for GPU arrays and Distributed arrays enabled by Parallel Computing Toolbox.

For details, see

- [Run MATLAB Functions with Automatic Parallel Support](#)
- [Run MATLAB Functions on a GPU](#)
- [Run MATLAB Functions with Distributed Arrays](#)

## **Write Cloud Data: Write distributed arrays and tall arrays to Amazon S3 and Windows Azure storage**

If you have access to cloud storage in Amazon S3™ or Windows Azure Blob Storage, you can upload your distributed and tall data using the `write` function. When you write data from a cluster in the same cloud service, the data transfer is more efficient. To learn more about your options for accessing cloud storage, see [Work with Remote Data \(MATLAB\)](#).

## **Big Data in the Cloud: Explore MATLAB capabilities for big data**

A new example shows how to access a publicly available dataset in the cloud and work with it using datastores, tall arrays and Parallel Computing Toolbox. This example shows you step-by-step how to use MATLAB to analyse huge datasets. See [Process Big Data in the Cloud](#).

## **Improved Scalability: Use up to 1024 workers per parallel pool**

MATLAB Distributed Computing Server can now support clusters with up to 1024 workers. Use large parallel pools of workers for big scaling problems.

## **Streamlined Cloud Cluster Setup: Create new Cloud Center clusters directly from the MATLAB desktop**

You can now create a MATLAB Distributed Computing Server enabled cluster on Amazon EC2 directly from MATLAB. On the **Home** tab, in the **Environment** area, select **Parallel > Create and Manage Clusters**, and choose **Create Cloud Cluster** to set up a new cloud cluster. For more information, see [Create Cloud Cluster](#).

## **Cluster Workflows: Learn how to scale up from desktop to cluster**

Use this new example to learn how to prototype interactively your parallel code on your local machine and scale up to a cluster. For more information, see [Scale up from Desktop to Cluster](#).

## **Streamlined Installation Documentation: Simplified workflows for integrating MATLAB Distributed Computing Server in your cluster**

Follow simplified workflows to integrate your third-party scheduler or MATLAB Job Scheduler with MATLAB Distributed Computing Server in your cluster. For details, see [Getting Started with MATLAB Distributed Computing Server \(MATLAB Distributed Computing Server\)](#).

## **Generic Scheduler Documentation: Set up Schedulers Using Improved Instructions**

Follow improved instructions to integrate a third-party scheduler using the generic scheduler interface. For more information, see [Configure Using the Generic Scheduler Interface \(MATLAB Distributed Computing Server\)](#).



---

## Parallel Deep Learning Workflows: Explore deep learning with multiple GPUs locally or in the cloud

Use examples to explore options for scaling up deep learning training. You can use multiple GPUs locally or in the cloud without changing your code. Use parallel computing to train multiple networks locally or on cloud clusters, and use datastores to access cloud data. New examples include:

- Train Network in the Cloud Using Built-in Parallel Support
- Use `parfor` to Train Multiple Deep Learning Networks
- Use `parfeval` to Train Multiple Deep Learning Networks
- Upload Deep Learning Data to the Cloud
- Send Deep Learning Batch Job To Cluster

To learn about options, see [Scale Up Deep Learning in Parallel and in the Cloud \(Deep Learning Toolbox\)](#).

## Upgrade Parallel Computing Products Together

As with every new release, if you upgrade Parallel Computing Toolbox you must upgrade MATLAB Distributed Computing Server at the same time so that they interact properly with each other.

## Compatibility Considerations

If you are using MATLAB Job Scheduler, the backward compatibility feature allows you to connect to multiple versions of MATLAB Distributed Computing Server in your cluster. For more information, see [Run Multiple MATLAB Distributed Computing Server Versions \(MATLAB Distributed Computing Server\)](#).

Jobs created in one version of Parallel Computing Toolbox software do not run in a different version of MATLAB Distributed Computing Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` (formerly `DataLocation`) might not be compatible between different versions of MATLAB Distributed Computing Server. Therefore, `JobStorageLocation` should not be shared by parallel computing products running different versions, and each version on your cluster should have its own `JobStorageLocation`.

## Functionality Being Removed or Changed

### **Philox4x32-10 and Threefry4x64-20 random number generators renamed**

*Still runs*

The Random123 random number generators `Philox` and `Threefry` are renamed to match the names used in MATLAB. `Philox4x32-10` is renamed to `Philox4x32_10` or `Philox`. `Threefry4x64-20` is renamed to `Threefry4x64_20` or `Threefry`.

- Replace all instances of `Philox4x32-10` with `Philox4x32_10` or `Philox`
- Replace all instances of `Threefry4x64-20` with `Threefry4x64_20` or `Threefry`

The names `Philox4x32-10` and `Threefry4x64-20` continue to work but are not recommended. There are no plans to remove support for these names at this time.

## **Default random number generator will change for parallel contexts**

*Behavior change in future release*

In a future release, the default random number generator for parallel computations will change to `Threefry`. This change will apply to calculations on parallel workers, GPU arrays, distributed arrays, and tall arrays. This generator will offer performance enhancements for parallel calculations over the current default. In releases up to R2018b, the default random number generator for parallel computations is `CombRecursive`.

With a different default generator, MATLAB will generate different random numbers sequences by default in the context of parallel computations. However, statistics of these calculations will remain unaffected. Therefore, you might want to update any code that relies on the specific random numbers being generated, but most calculations on the random numbers should be unaffected.

To set the generator to the settings used by default in R2018b and earlier on parallel workers, GPU arrays, and tall arrays, use the following commands.

Calculations on parallel workers and distributed arrays:

```
spm�
    stream = RandStream.create("CombRecursive", "NumStreams", 2^32,...
        "StreamIndices", 2*labindex);
    RandStream.setGlobalStream(stream);
end
```

Calculations on GPU arrays:

```
gpurng(0,"CombRecursive")
```

Calculations on tall arrays:

```
tallrng(0,"CombRecursive")
```

## **Support for MATLAB Parallel Cloud is removed**

Starting in R2018b, support for MATLAB Parallel Cloud clusters is removed. For easy-to-use scaling, you can use MathWorks Cloud Center to run MATLAB on preconfigured, customizable cloud clusters powered by Amazon EC2. To find out more about MathWorks Cloud Center, see [Parallel Computing on the Cloud with MATLAB and MathWorks Cloud Center](#).

# R2018a

---

**Version: 6.12**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## **Parfeval callbacks: New afterAll and afterEach methods for parallel futures**

You can use the new methods `afterAll` and `afterEach` to specify functions to automatically execute when parallel futures complete. You create parallel futures using the `parfeval` function. A useful application for `afterEach` and `afterAll` is to update user interfaces such as plots and apps during parallel computations using `parfeval`. For an example, see [Update a User Interface Asynchronously Using `afterEach` and `afterAll`](#).

## **Slurm support: Slurm is a fully supported scheduler**

Slurm is now a fully supported scheduler. You can create Slurm profiles directly in the Profile Manager without needing templates and scripts from File Exchange. For more information, see [Configure for Slurm, PBS Pro, Platform LSF, TORQUE](#).

## **Support for NVIDIA Volta: Update to CUDA 9, support for Volta class GPUs**

The parallel computing products are now using CUDA Toolkit version 9.0. To compile CUDA code for `CUDAkernel` or CUDA MEX-files, you must use toolkit version 9.0.

This update improves support for Volta class GPUs.

## **Improved file mirroring: Performance of file mirroring for generic scheduler integration**

File mirroring for the generic scheduler integration has been improved through the use of zip archives.

## **Parfor performance improvements: More efficient broadcast variables in parfor for non-local clusters**

If you use broadcast variables in `parfor` loops with non-local clusters, now performance is improved. Broadcast variables in `parfor` loops are sent only once to the cluster, instead of once per worker.

## **GPU Array Support: Use enhanced gpuArray functions**

You can now use the following enhanced `gpuArray` functions:

- `rescale`
- new syntaxes of `sort` and `sortrow`

For more information, see [Run Built-In Functions on a GPU](#).

## **Distributed Array Support: Use enhanced distributed array functions**

You can now use the following enhanced distributed array functions:

- 
- `rescale`
  - new syntaxes of `sort` and `sortrow`
  - `bicgstabl`

For more information, see [Using MATLAB Functions on Distributed Arrays](#).

## **Parameter Sweep Example: Use a DataQueue to monitor results during computations on a parallel pool**

The new example *Plot during Parameter Sweep with parfor* shows how to perform a parameter sweep in parallel and plot progress during parallel computations. For details, see [Plot during Parameter Sweep with parfor](#).

## **Discontinued Support for GPU Devices of Compute Capability less than 3.0**

In R2018a, support for GPU devices of compute capability less than 3.0 is removed. A minimum compute capability of 3.0 is required for GPU computing in MATLAB. For more information on using GPUs, see [GPU Computing in MATLAB](#).

## **Compatibility Considerations**

In R2018a, any use of `gpuDevice` to select a GPU with compute capability less than 3.0 generates an error. Support is removed for devices with compute capability less than 3.0.

## **Backwards Compatibility for MATLAB Job Scheduler: Upgrade MATLAB Job Scheduler clusters, and continue to use previous releases of Parallel Computing Toolbox**

You can upgrade your MATLAB Job Scheduler clusters to the latest release and still connect to it using previous releases (back to R2016a) of Parallel Computing Toolbox on your MATLAB desktop client. For the supported older releases, you must maintain an installation of the same release of MATLAB Distributed Computing Server for each release of MATLAB you want to use. The latest MATLAB Job Scheduler routes your jobs accordingly. You can configure MATLAB Job Scheduler with the location of these installations in the `mdce_def` file. For more information, see [Install Products and Choose Cluster Configuration](#).

## **Upgrade Parallel Computing Products Together**

As with every new release, if you upgrade Parallel Computing Toolbox you must upgrade MATLAB Distributed Computing Server at the same time so that they interact properly with each other. Note that you do not need to do this if you are taking advantage of MATLAB Job Scheduler's backwards compatibility.

The R2018a version of Parallel Computing Toolbox software is accompanied by a corresponding new version of MATLAB Distributed Computing Server software.

## Compatibility Considerations

As with every new release, if you are using both parallel computing products, you must upgrade Parallel Computing Toolbox and MATLAB Distributed Computing Server together. These products must be the same version to interact properly with each other. This requirement applies only if you are not taking advantage of MATLAB Job Scheduler backwards compatibility.

Jobs created in one version of Parallel Computing Toolbox software do not run in a different version of MATLAB Distributed Computing Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` (formerly `DataLocation`) might not be compatible between different versions of MATLAB Distributed Computing Server. Therefore, `JobStorageLocation` should not be shared by parallel computing products running different versions, and each version on your cluster should have its own `JobStorageLocation`.

## Functionality Being Removed or Changed

Functionality	Result	Use Instead	Compatibility Considerations
Support for GPU devices of compute capability less than 3.0 is removed.	Errors	Use GPUs with a minimum compute capability of 3.0.	Support is removed for GPU devices of compute capability 2.x. Use GPUs with a minimum compute capability 3.0. Use the function <code>gpuDevice</code> to query the compute capabilities of your GPUs. For more information, see GPU Computing in MATLAB.
<code>parallel.gpu.rng</code> is renamed to <code>gpurng</code>	Still runs	<code>gpurng</code>	Replace all instances of <code>parallel.gpu.rng</code> with <code>gpurng</code>

# R2017b

---

**Version: 6.11**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## **Improved Parallel Language Performance: Execute parallel language constructs with reduced overhead**

All parallel language constructs, including `parfor`, run with reduced overhead, particularly constructs with short duration. The scheduling of `parfor`-loop intervals has been changed to utilize the cluster hardware more efficiently.

## **Tall Array Support: Use tall arrays with Windows client access to Linux Spark clusters**

You can use tall arrays on Spark enabled Hadoop clusters supporting all architectures for the client, while supporting Linux and Mac architectures for the cluster. This includes cross-platform support.

## **Improved Parallel Pool Robustness: Run pools without Message Passing Interface (MPI) by default, making pools resilient to workers crashing**

You can now run parallel pools without MPI by default, improving the resilience of your parallel pool to workers crashing.

## **Improved MATLAB Integration with Third-Party Schedulers: Use the Generic Profile Wizard for easier installation and setup of MATLAB Distributed Computing Server**

You can now use the new workflow to automatically create generic profiles using the support packages for third-party schedulers. For more details, see [Distribute a Generic Cluster Profile and Integration Scripts \(MATLAB Distributed Computing Server\)](#).

## **Cloud Storage: Work with data in Microsoft Azure Blob Storage**

You can now use Microsoft® Azure® Blob Storage to access data in the cloud using `datastore`. For more information on this topic, see [Distribute a Generic Cluster Profile and Integration Scripts \(MATLAB Distributed Computing Server\)](#).

## **Copy Client Environment to Workers on Any Cluster: Specify which environment variables on your client machine your workers should automatically inherit**

You can now mirror any of your client environment variables to the workers on a cluster. For example, you can include any environment variables required by third-party software. Or you can include cloud service provider credentials in your local environment to give worker processes access to your data stored in the cloud.

You can now set `EnvironmentVariables` in `parpool`, `batch`, `createJob` and in the Cluster Profile Manager. `EnvironmentVariables` is also a common property of the `parallelJob` class.



---

## **Client Path Sharing: Add user-added-entries on the client's path to the workers' paths**

You can now automatically add user-added-entries on the client's path to the workers' paths for batch and independent jobs. This functionality can be enabled or disabled by specifying the `AutoAddClientPath` option to the `parpool`, `batch` and `createJob` commands. `AutoAddClientPath` is also a common property of the `parallel.Job` class.

## **GPU Array Support: Use enhanced gpuArray functions**

You can now use the following enhanced `gpuArray` functions:

- `bounds` for computing `max` and `min` simultaneously
- Sparse iterative solvers `cgs` and `lsqr`
- `spdiags`

For more information on this topic, see [Run Built-In Functions on a GPU](#).

## **Distributed Array Support: Use enhanced distributed array functions**

You can now use the following enhanced distributed array functions:

- Sparse iterative solvers `minres`, `symmlq` and `bicgstab`
- `bounds` for computing `max` and `min` simultaneously
- `eigs`
- `spdiags`

For more information, see [Using MATLAB Functions on Distributed Arrays](#).

## **Enhanced Support for Microsoft Windows HPC Pack**

The parallel computing products now support Microsoft Windows HPC Pack 2016. For details, see [Configure for HPC Pack \(MATLAB Distributed Computing Server\)](#).

## **Discontinued Support for GPU Devices of Compute Capability less than 3.0**

In a future release, support for GPU devices of compute capability less than 3.0 will be removed. At that time, a minimum compute capability of 3.0 will be required.

## **Compatibility Considerations**

In R2017b, any use of `gpuDevice` to select a GPU with compute capability less than 3.0, generates a warning. The device is still supported in this release, but in a future release support will be completely removed for devices with compute capability less than 3.0.

## Backwards Compatibility for MATLAB Job Scheduler: Upgrade MATLAB Job Scheduler clusters, and continue to use previous releases of Parallel Computing Toolbox

You can upgrade your MATLAB Job Scheduler clusters to R2017b and still connect to it using the R2017a, R2016b, and R2016a releases of Parallel Computing Toolbox on your MATLAB desktop client. For releases prior to R2016b, you must maintain an installation of the same release of MATLAB Distributed Computing Server for each release of MATLAB you want to use. The latest MATLAB Job Scheduler routes your jobs accordingly. You can configure MATLAB Job Scheduler with the location of these installations in the `mdce_def` file. For more information, see [Install Products and Choose Cluster Configuration \(MATLAB Distributed Computing Server\)](#).

### Upgrade Parallel Computing Products Together

The R2017b version of Parallel Computing Toolbox software is accompanied by a corresponding new version of MATLAB Distributed Computing Server software.

### Compatibility Considerations

As with every new release, if you are using both parallel computing products, you must upgrade Parallel Computing Toolbox and MATLAB Distributed Computing Server together. These products must be the same version to interact properly with each other. This requirement applies only if you are not taking advantage of MATLAB Job Scheduler backwards compatibility.

Jobs created in one version of Parallel Computing Toolbox software do not run in a different version of MATLAB Distributed Computing Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` (formerly `DataLocation`) might not be compatible between different versions of MATLAB Distributed Computing Server. Therefore, `JobStorageLocation` should not be shared by parallel computing products running different versions, and each version on your cluster should have its own `JobStorageLocation`.

### Functionality Being Removed or Changed

Functionality	Result	Use Instead	Compatibility Considerations
Support for running MATLAB MapReduce on Hadoop 1.x clusters has been removed.	Errors	Use clusters that have Hadoop 2.x or higher installed to run MATLAB MapReduce.	Migrate MATLAB MapReduce code that runs on Hadoop 1.x to Hadoop 2.x.

For more information, see [Configure a Hadoop Cluster \(MATLAB Distributed Computing Server\)](#).

# R2017a

---

**Version: 6.10**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Access to Intermediate Results and Updates in Parallel Computations: Poll for messages or data from different workers during parallel workflows

You can now transfer intermediate results when you carry out `parfor`, `spmd`, or `parfeval` calculations. Use the `send` and `poll` methods together to transfer and retrieve messages or data from different workers, using a pollable `DataQueue`.

## Tall Array Support: Use parallel execution environments for tall timetables and enhanced tall array functions

You can now use the following enhanced tall array functions:

- `timetable` supports:

<code>head</code>	<code>join</code>	<code>stack</code>	<code>unique</code>
<code>height</code>	<code>ndims</code>	<code>standardizeMissing</code>	<code>varfun</code>
<code>isempty</code>	<code>numel</code>	<code>table2array</code>	<code>width</code>
<code>innerjoin</code>	<code>size</code>	<code>table2cell</code>	
<code>ismember</code>	<code>sortrows</code>	<code>tail</code>	
<code>ismissing</code>	<code>splitapply</code>	<code>topkrows</code>	

- `array2table`
- `bsxfun`
- `conv`
- `cumsum`, `cumprod`, `cummax`, `cummin`
- `diff`
- `ismember`
- `issorted`, `issortedrows`
- `movmad`, `movmax`, `movmean`, `movmedian`, `movmin`, `movprod`, `movstd`, `movsum`, `movvar`
- `pie` for categoricals
- `repmat`, `repelem`
- `sort`, `sortrows`
- `table2timetable`, `timetable2table`
- `varfun`
- Improved indexing, allowing sorted (ascending or descending) indices in the first dimension

For more information, see [Functions That Support Tall Arrays \(A-Z\)](#).

## More Responsive Job Monitor: Automatic updates for new, submitted, or deleted jobs or tasks

You can now use the Job Monitor to get an up-to-date view of your cluster. Verify your changes without manually querying the cluster or forcing an update to the Job Monitor user interface.

---

Examine your latest changes and execute quickly without interrupting your workflow. For more details, see Job Monitor.

## **Re-create Jobs: Easily rerun all failed or cancelled tasks**

If your job failed or was cancelled, you can now use the `recreate` method to return a new job object. Call `submit` on the new job object to easily rerun all failed or cancelled tasks.

## **GPU Array Support: Use enhanced `gpuArray` functions**

You can now use the following enhanced `gpuArray` functions:

- `head`, `tail`
- Cubic support for `interp1` and `interp2`
- `movmean`, `movstd`, `movsum`, `movvar`
- `svds`
- `tril` and `triu` for sparse arrays

For more information on this topic, see [Run Built-In Functions on a GPU](#).

## **Distributed Array Support: Use enhanced distributed array functions**

You can now use the following enhanced distributed array functions:

- `issymmetric`, `ishermitian`
- `qmr`
- `svds`
- `tfqmr`
- `write` for storing a snapshot of a distributed array in a format suitable for datastore
- `mldivide`, providing improved performance for triangular and diagonal systems

For more information, see [Using MATLAB Functions on Distributed Arrays](#).

## **Simplified Integration for Third-Party Cluster Schedulers: Updates to generic scheduler integration allow folder-based configuration and eliminate the need to specify function handles**

You no longer need to specify function handles in your generic scheduler profile. Instead, specify only one folder name and some `AdditionalProperties`, which are similar to name-value pairs. For more information, see [Configure for a Generic Scheduler](#).

## **MATLAB String Support: Functions in Parallel Computing Toolbox accept MATLAB strings as input**

You can now use MATLAB strings for functions in Parallel Computing Toolbox. For information on MATLAB strings, see [Create String Arrays \(MATLAB\)](#).

## Support for Spark 2.x enabled Hadoop clusters

Tall array integration on a Spark enabled Hadoop cluster, running MATLAB Distributed Computing Server, now supports both versions 1.x and 2.x.

## Upgraded CUDA Toolkit version

The parallel computing products are now using CUDA Toolkit version 8.0. To compile CUDA code for `CUDAkernel` or CUDA MEX-files, you must use toolkit version 8.0.

## Discontinued Support for GPU Devices of Compute Capability less than 3.0

In a future release, support for GPU devices of compute capability less than 3.0 will be removed. At that time, a minimum compute capability of 3.0 will be required.

## Compatibility Considerations

In R2017a, any use of `gpuDevice` to select a GPU with compute capability less than 3, generates a warning. The device is still supported in this release, but in a future release support will be completely removed for devices with compute capability less than 3.

## Backwards Compatibility for MATLAB Job Scheduler: Upgrade MATLAB Job Scheduler clusters, and continue to use previous releases of Parallel Computing Toolbox

You can upgrade your MATLAB Job Scheduler clusters to R2017a and still use the R2016b and R2016a releases of Parallel Computing Toolbox on your MATLAB desktop client to connect to it. This situation only applies to the R2016a release onward. You must maintain an installation of the same release of MATLAB Distributed Computing Server for each release of MATLAB you want to use. The latest MATLAB Job Scheduler routes your jobs accordingly. You can configure MATLAB Job Scheduler with the location of these installations in the `mdce_def` file. For more information, see [Install Products and Choose Cluster Configuration](#).

## Upgrade Parallel Computing Products Together

This version of Parallel Computing Toolbox software is accompanied by a corresponding new version of MATLAB Distributed Computing Server software.

## Compatibility Considerations

As with every new release, if you are using both parallel computing products, you must upgrade Parallel Computing Toolbox and MATLAB Distributed Computing Server together. These products must be the same version to interact properly with each other.

Jobs created in one version of Parallel Computing Toolbox software do not run in a different version of MATLAB Distributed Computing Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` (formerly `DataLocation`) might not be compatible between different versions of MATLAB Distributed

---

Computing Server. Therefore, `JobStorageLocation` should not be shared by parallel computing products running different versions, and each version on your cluster should have its own `JobStorageLocation`.

## Properties of generic cluster objects have changed

The properties of new generic cluster objects have changed.

## Compatibility Considerations

From R2017a onwards, you can no longer use the following old properties:

- `CancelJobFcn`
- `CancelTaskFcn`
- `CommunicatingSubmitFcn`
- `DeleteJobFcn`
- `DeleteTaskFcn`
- `GetJobStateFcn`
- `IndependentSubmitFcn`

Instead, set the new `IntegrationScriptsLocation` property to the folder containing your cluster's integration scripts. You must now use integration scripts with the default function name and signature. Specify any additional input arguments to these scripts using the new `AdditionalProperties` property. For more details, see

- [Configure for a Generic Scheduler](#)
- [Program Communicating Jobs for a Generic Scheduler](#)
- [Program Independent Jobs for a Generic Scheduler](#)

## Functionality Being Removed or Changed

Functionality	Result	Use Instead	Compatibility Considerations
Support for running MATLAB MapReduce on Hadoop 1.x clusters will be removed in a future release.	Warns	Use clusters that have Hadoop 2.x or higher installed to run MATLAB MapReduce.	Migrate MATLAB MapReduce code that runs on Hadoop 1.x to Hadoop 2.x.

For more information, see “[Configure a Hadoop Cluster](#)” (MATLAB Parallel Server).





# R2016b

---

**Version: 6.9**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Parallel Support for Tall Arrays: Process big data with tall arrays in parallel on your desktop, MATLAB Distributed Computing Server, and Spark clusters

Speed up your tall array workflows with Parallel Computing Toolbox. You can use Parallel Computing Toolbox to evaluate tall array expressions in parallel using a parallel pool on your desktop. You can also use Parallel Computing Toolbox to scale up tall-array processing by connecting to a parallel pool running on a MATLAB Distributed Computing Server cluster, or to a Spark enabled Hadoop cluster running MATLAB Distributed Computing Server. For more information, see

- Big Data Workflow Using Tall Arrays and Datastores
- Use Tall Arrays on a Parallel Pool
- Use Tall Arrays on a Spark Enabled Hadoop Cluster

## Support for GPU Arrays: Use enhanced gpuArray functions, including new sparse iterative solver bicg

- New sparse iterative solver `bicg`
- Support for tolerance set functions: `ismembertol`, `uniquetol`
- Create sparse arrays using `sprandsym`

For more information on this topic, see [Run Built-In Functions on a GPU](#).

## Parallel Menu Enhancement: Use the new menu items in the Parallel Menu to configure and manage cloud based resources

Open the Cloud Center web application, and view MATLAB Distributed Computing Server license usage. For more information, see [Use Parallel Menu and Cluster Profiles](#).

## New Data Types in Distributed Arrays: Use enhanced functions for creating distributed arrays of: `datetime`; `duration`; `calendarDuration`; `string`; `categorical`; and `table`

Distributed `calendarDuration` Arrays:

<code>calendarDuration</code>	<code>calquarters</code>	<code>cellstr</code>	<code>time</code>
<code>caldays</code>	<code>calweeks</code>	<code>datevec</code>	
<code>calmonths</code>	<code>calyears</code>	<code>iscalendarduration</code>	

Distributed `categorical` Arrays:

<code>categorical</code>	<code>iscategorical</code>	<code>isundefined</code>	<code>setcats</code>
<code>addcats</code>	<code>iscategory</code>	<code>removecats</code>	
<code>categories</code>	<code>isordinal</code>	<code>renamecats</code>	
<code>countcats</code>	<code>isprotected</code>	<code>reordercats</code>	

Distributed `datetime` Arrays:

---

datetime	exceltime	isweekend	string
between	hms	juliandate	timeofday
cellstr	hour	minute	tzoffset
datenum	isbetween	month	week
dateshift	isdatetime	posixtime	year
datevec	isdst	quarter	ymd
day	isnat	second	yyyymmdd

Distributed duration Arrays:

duration	datevec	hours	minutes
cellstr	days	isduration	seconds
datenum	hms	milliseconds	years

Distributed string Arrays:

string	erase	insertBefore	replaceBetween
cellstr	eraseBetween	ismissing	reverse
compose	extractAfter	isstring	startsWith
contains	extractBefore	lower	strip
count	extractBetween	pad	strlength
endsWith	insertAfter	replace	upper

Distributed tables:

table	istable	table2cell
head	standardizeMissing	tail
ismissing	table2array	

For more information, see [Using MATLAB Functions on Distributed Arrays](#).

## Loading Distributed Arrays: Load distributed arrays in parallel using datastore

Create distributed arrays more easily using `datastore`, and eliminate the need to create distributed arrays with `codistributed`. For more information, see [Load Distributed Arrays in Parallel Using datastore](#).

## Cluster Profile Validation: Choose which validation stages run and the number of MATLAB workers to use

In previous releases, validating your cluster profile ran all validation stages and used a fixed number of workers determined from your profile. You can now choose to run a subset of the validation stages and specify the number of workers to use when validating your profile. For more information on the detailed validation steps in your cluster profile, see [Validate Cluster Profiles](#).

## **Backwards Compatibility: Upgrade MATLAB Job Scheduler clusters, and continue to use the previous release of Parallel Computing Toolbox**

You can upgrade your MATLAB Job Scheduler clusters to R2016b and still use the R2016a release of Parallel Computing Toolbox on your MATLAB desktop client to connect to it. This situation only applies to the R2016a release onward. You must maintain an installation of the same release of MATLAB Distributed Computing Server for each release of MATLAB you want to use. The latest MATLAB Job Scheduler will route your jobs accordingly. You can configure MATLAB Job Scheduler with the location of these installations in the `mdce_def` file. For more information, see [Install Products and Choose Cluster Configuration](#).

## **datetime Support for Timestamps: Use built-in datetime objects in MATLAB to access timestamp information for jobs and tasks**

You can now use the following properties for MATLAB jobs:

CreateDateTime  
SubmitDateTime  
StartDateTime  
FinishDateTime

You can use the following properties for MATLAB tasks:

CreateDateTime  
StartDateTime  
FinishDateTime

For more information, see `parallel.Job` and `parallel.Task`

## **Data Transfer Measurement: Use ticBytes and tocBytes to measure the data transfer between MATLAB workers in a parallel pool**

You can now measure how much data needs to be passed around to carry out `parfor`, `spmd` or `parfeval`. Use `ticBytes` and `tocBytes` to optimize your code and pass around less data.

## **Multithreaded Workers: Use multiple computational threads on your MATLAB workers**

MATLAB workers used to run in single-threaded mode. Now you can control the number of computational threads so that workers can run in multithreaded mode and use all the cores on your cluster. This enables you to increase the number of computational threads, `NumThreads`, on each worker, without increasing the number of workers, `NumWorkers`. If you have more cores available, increase `NumThreads` to take full advantage of the built-in parallelism provided by the multithreaded nature of many of the underlying MATLAB libraries. For more information, see [Create and Modify Cluster Profiles](#).

---

## Support for Distributed Arrays: Use enhanced distributed array functions, including sparse input to iterative solvers (gmres and lsqr)

- Sparse support for gmres and lsqr
- Support for isdiag, istril, istriu, isbanded, bandwidth, and mrdivide for both sparse and dense arrays
- N-dimensional convolutions with convn
- find now supports the 2-D block-cyclic (2dbc) distribution scheme

For more information, see Using MATLAB Functions on Distributed Arrays.

## Increased Data Transfer Limits: Send messages larger than 2 GB using labSend, labSendReceive and labBroadcast

In previous releases, it was not possible to use labSend, labSendReceive, or labBroadcast to send messages larger than 2GB when the message was not a real dense numeric array (such as a complex array, a sparse array, a cell array or a struct). In R2016b, this limitation has been removed.

## Upgrade Parallel Computing Products Together

This version of Parallel Computing Toolbox software is accompanied by a corresponding new version of MATLAB Distributed Computing Server software.

## Compatibility Considerations

If you are running MATLAB Job Scheduler on your cluster, then you can upgrade MATLAB® Distributed Computing Server without upgrading Parallel Computing Toolbox. However, you cannot upgrade Parallel Computing Toolbox without upgrading MATLAB Distributed Computing Server.

If you are not running MATLAB Job Scheduler, then you must upgrade Parallel Computing Toolbox and MATLAB Distributed Computing Server together. Jobs created in one version of Parallel Computing Toolbox software will not run in a different version of MATLAB Distributed Computing Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by JobStorageLocation (formerly DataLocation) might not be compatible between different versions of MATLAB Distributed Computing Server. Therefore, JobStorageLocation should not be shared by parallel computing products running different versions, and each version on your cluster should have its own JobStorageLocation.

## Functionality Being Removed or Changed

Functionality	Result	Use Instead	Compatibility Considerations
Support for running MATLAB MapReduce on Hadoop 1.x clusters will be removed in a future release.	Still runs	Use clusters that have Hadoop 2.x or higher installed to run MATLAB MapReduce.	Migrate MATLAB MapReduce code that runs on Hadoop 1.x to Hadoop 2.x

For more information, see [Configure a Hadoop Cluster](#).

# R2016a

---

**Version: 6.8**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## **GPU Support for Sparse Matrices: Use enhanced gpuArray functions for sparse matrices on GPUs**

- Support for 5-argument form of sparse constructor
- Support for sparse inputs to bicgstab and pcg

You can create a sparse gpuArray either by calling sparse with a gpuArray input, or by calling gpuArray with a sparse input.

For more information on this topic, see Sparse Arrays on a GPU.

## **Support for Distributed Arrays: Use enhanced distributed array functions including sparse input to direct (mldivide) and iterative solvers (cgs and pcg)**

The following functions are new in supporting distributed arrays:

```
cgs
conv
conv2
expint
ischar
pcg
superiorfloat
```

For more details, see MATLAB Functions on Distributed and Codistributed Arrays.

In addition, the following features are new in providing enhanced functionality for distributed arrays:

- Sparse input to mldivide (direct system solve)
- Sparse input to cgs and pcg (iterative system solve)
- Single argument syntax for sprand and sprandn

## **GPU-Accelerated Deep Learning: Use Neural Network Toolbox to train deep convolutional neural networks with GPU-enabled acceleration for image classification tasks**

For specific information, see the Neural Network Toolbox™ release notes. To access this functionality, Parallel Computing Toolbox is required.

## **GPU-enabled MATLAB Functions: Accelerate applications using GPU-enabled MATLAB functions for linear equations, descriptive statistics and set operations**

The following functions are new in their support for gpuArrays:



---

```
bicg
detrnd
discretize
expint
pcg
spconvert
sprand
sprandn
```

New support is available for the 'rows' option in the following Set Operations:

```
intersect
ismember
setdiff
setxor
union
unique
```

For more details, see Run Built-In Functions on a GPU.

## **Upgraded CUDA Toolkit version**

The parallel computing products are now using CUDA Toolkit version 7.5. To compile CUDA code for CUDAKernel or CUDA MEX-files, you must use toolkit version 7.5.

## **Parallel-Enabled Gradient Estimation: Accelerate more nonlinear solvers in the Optimization Toolbox with parallel finite difference estimation of gradients and Jacobians**

## **Hadoop Kerberos Support: Improved support for Hadoop in a Kerberos authenticated environment**

In R2016a, enhanced support for the recommended setup for the Cloudera distribution of Hadoop has been provided.

## **Transfer unlimited data between client and workers, and attached files up to 4GB in total, in any job using a MATLAB Job Scheduler cluster**

In previous releases, the data transfer limit for a MATLAB Job Scheduler cluster was 2GB. In R2016a, this limit has been removed.

## **matlabpool function removed**

matlabpool function has been removed. Use parpool instead.

## Compatibility Considerations

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
matlabpool	Errors	parpool	<p>In R2016a, matlabpool generates the following error:</p> <pre>Undefined function or variable 'matlabpool'.</pre> <p>In R2015a and R2015b, matlabpool generates the following error:</p> <pre>Error using matlabpool. matlabpool: Use parpool instead.</pre>

### Upgrade parallel computing products together

This version of Parallel Computing Toolbox software is accompanied by a corresponding new version of MATLAB Distributed Computing Server software.

### Compatibility Considerations

As with every new release, if you are using both parallel computing products, you must upgrade Parallel Computing Toolbox and MATLAB Distributed Computing Server together. These products must be the same version to interact properly with each other.

Jobs created in one version of Parallel Computing Toolbox software will not run in a different version of MATLAB Distributed Computing Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` (formerly `DataLocation`) might not be compatible between different versions of MATLAB Distributed Computing Server. Therefore, `JobStorageLocation` should not be shared by parallel computing products running different versions, and each version on your cluster should have its own `JobStorageLocation`.

# R2015b

---

**Version: 6.7**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Discontinued support for parallel computing products on 32-bit Windows operating systems

This release of MATLAB products no longer supports 32-bit Parallel Computing Toolbox and MATLAB Distributed Computing Server on Windows operating systems.

## Compatibility Considerations

You can no longer install the parallel computing products on 32-bit Windows operating systems. If you must use Windows operating systems for the parallel computing products, upgrade to 64-bit MATLAB products on a 64-bit operating system.

## More than 90 GPU-enabled functions in Statistics and Machine Learning Toolbox, including probability distribution, descriptive statistics, and hypothesis testing

Statistics and Machine Learning Toolbox offers enhanced functionality for some of its functions to perform computations on a GPU. For specific information, see the Statistics and Machine Learning Toolbox release notes. Parallel Computing Toolbox is required to access this functionality.

## Additional GPU-enabled MATLAB functions, including support for sparse matrices

- “Additional GPU-enabled MATLAB functions” on page 16-2
- “Sparse arrays with GPU-enabled functions” on page 16-2

### Additional GPU-enabled MATLAB functions

The following functions are new in their support for `gpuArrays`:

<code>assert</code>	<code>isbanded</code>	<code>setxor</code>
<code>bandwidth</code>	<code>rad2deg</code>	<code>sortrows</code>
<code>deg2rad</code>	<code>randperm</code>	<code>union</code>
<code>expm</code>	<code>rectint</code>	<code>unique</code>
<code>gmres</code>	<code>repelem</code>	
<code>intersect</code>	<code>setdiff</code>	

In addition to the functions above, the following functions are enhanced in their `gpuArray` support:

- `arrayfun` now supports code that includes the functions `isfloat`, `isinteger`, `islogical`, `isnumeric`, `isreal`, and `issparse`.
- `pagefun` support for `@mldivide` and `@mrdivide` is no longer limited to 32-by-32 pages.
- `cov`, `max`, `mean`, `median`, `min`, `std`, `sum`, and `var` now support the `'omitnan'` option.
- `accumarray` now supports the logical sparse argument, for generating a sparse `gpuArray` output.

For a list of MATLAB functions that support `gpuArray`, see [Run Built-In Functions on a GPU](#).

### Sparse arrays with GPU-enabled functions

The following functions are new in their support for sparse `gpuArrays`:

---

abs	isequal	round
angle	isequaln	sign
ceil	log1p	spfun
deg2rad	minus	sqrt
expm1	nextpow2	sum
fix	plus	uminus
floor	rad2deg	uplus
gmres	realsqrt	

You can create a sparse `gpuArray` either by calling `sparse` with a `gpuArray` input, or by calling `gpuArray` with a sparse input.

For more information on this topic, see [Sparse Arrays on a GPU](#).

## **mexcuda function for easier compilation of MEX-files containing CUDA code**

A new `mexcuda` function allows you to compile MEX-functions for GPU computation.

For more information, see the `mexcuda` function reference page. See also [Run CUDA or PTX Code on GPU](#).

## **Upgraded CUDA Toolkit version**

The parallel computing products are now using CUDA Toolkit version 7.0. To compile CUDA code for `CUDAKernel` or `CUDA MEX-files`, you must use toolkit version 7.0.

## **Scheduler integration scripts for SLURM**

This release offers a new set of scripts containing `submit` and `decode` functions to support Simple Linux<sup>®</sup> Utility for Resource Management (SLURM), using the generic scheduler interface. The pertinent code files are in the folder:

```
matlabroot/toolbox/distcomp/examples/integration/slurm
```

where *matlabroot* is your installation location.

The `slurm` folder contains a `README` file of instructions, and folders for `shared`, `nonshared`, and `remoteSubmission` network configurations.

For more information, view the files in the appropriate folders. See also [Program Independent Jobs for a Generic Scheduler](#) and [Program Communicating Jobs for a Generic Scheduler](#).

## **parallel.pool.Constant function to create constant data on parallel pool workers, accessible within parallel language constructs such as parfor and parfeval**

A new `parallel.pool.Constant` function allows you to define a constant whose value can be accessed by multiple `parfor`-loops or other parallel language constructs (e.g., `spmd` or `parfeval`) without the need to transfer the data multiple times.

For more information and examples, see `parallel.pool.Constant`.

## Improved performance of mapreduce on Hadoop 2 clusters

The performance of `mapreduce` running on a Hadoop 2.x cluster with MATLAB Distributed Computing Server is improved in this release for large input data.

## Enhanced and additional MATLAB functions for distributed arrays

The following functions are new in supporting distributed arrays, with all forms of codistributor (1-D and 2DBC):

<code>bicg</code>	<code>rad2deg</code>
<code>deg2rad</code>	<code>rectint</code>
<code>polyarea</code>	<code>repelem</code>
<code>polyint</code>	

In addition to the new functions above, the following functions are enhanced in their distributed array support:

- `cov`, `max`, `mean`, `median`, `min`, `std`, `sum`, and `var` now support the `'omitnan'` option.

For a list of MATLAB functions that support distributed arrays, see [MATLAB Functions on Distributed and Codistributed Arrays](#).

## Warnings property for tasks

There is a new `Warnings` property for `parallel.Task` objects. This property contains warning information that was issued during execution of the task. The data type is a structure array with the fields `message`, `identifier`, and `stack`. This information is part of the job display in the Command Window and Job Monitor.

## More consistent transparency enforcement

For more consistent behavior and results, transparency violations in `parfor`-loops and `spmd` blocks are now being enforced more strictly.

## Compatibility Considerations

It is possible that `parfor` or `spmd` code that did not error in previous releases will now generate a transparency violation error. For more information, see [Transparency](#).

Anonymous functions that were saved in a previous release and that reference nested functions will not execute in this release. They will load, but when executed will issue an “Undefined function handle” error.

## Upgrade parallel computing products together

This version of Parallel Computing Toolbox software is accompanied by a corresponding new version of MATLAB Distributed Computing Server software.

---

## Compatibility Considerations

As with every new release, if you are using both parallel computing products, you must upgrade Parallel Computing Toolbox and MATLAB Distributed Computing Server together. These products must be the same version to interact properly with each other.

Jobs created in one version of Parallel Computing Toolbox software will not run in a different version of MATLAB Distributed Computing Server software, and might not be readable in different versions of the toolbox software. The job data stored in the folder identified by `JobStorageLocation` (formerly `DataLocation`) might not be compatible between different versions of MATLAB Distributed Computing Server. Therefore, `JobStorageLocation` should not be shared by parallel computing products running different versions, and each version on your cluster should have its own `JobStorageLocation`.





# R2015a

---

**Version: 6.6**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Support for mapreduce function on any cluster that supports parallel pools

You can now run parallel mapreduce on any cluster that supports a parallel pool. For more information, see [Run mapreduce on a Parallel Pool](#).

## Sparse arrays with GPU-enabled functions

This release supports sparse arrays on a GPU. You can create a sparse `gpuArray` either by calling `sparse` with a `gpuArray` input, or by calling `gpuArray` with a sparse input. The following functions support sparse `gpuArrays`.

<code>classUnderlying</code>	<code>isfloat</code>	<code>nnz</code>
<code>conj</code>	<code>isinteger</code>	<code>numel</code>
<code>ctranspose</code>	<code>islogical</code>	<code>nzmax</code>
<code>end</code>	<code>isnumeric</code>	<code>real</code>
<code>find</code>	<code>isreal</code>	<code>size</code>
<code>full</code>	<code>issparse</code>	<code>sparse</code>
<code>gpuArray.speye</code>	<code>length</code>	<code>spones</code>
<code>imag</code>	<code>mtimes</code>	<code>transpose</code>
<code>isaUnderlying</code>	<code>ndims</code>	
<code>isempty</code>	<code>nonzeros</code>	

Note the following for some of these functions:

- `gpuArray.speye` is a static constructor method.
- `sparse` supports only single-argument syntax.
- `mtimes` does not support the case of full-matrix times a sparse-matrix.

For more information on this topic, see [Sparse Arrays on a GPU](#).

A new C function, `mxGPUisSparse`, is available for the MEX interface, to query whether a `gpuArray` is sparse or not. However, even though the MEX interface can query properties of a sparse `gpuArray`, its functions cannot access sparse `gpuArray` elements.

## Additional GPU-enabled MATLAB functions

The following functions are new in their support for `gpuArrays`:

<code>cdf2rdf</code>	<code>istril</code>	<code>polyarea</code>
<code>gpuArray.freqspace</code>	<code>istriu</code>	<code>polyder</code>
<code>histcounts</code>	<code>legendre</code>	<code>polyfit</code>
<code>idivide</code>	<code>nonzeros</code>	<code>polyint</code>
<code>inpolygon</code>	<code>nthroot</code>	<code>polyval</code>
<code>isdiag</code>	<code>pinv</code>	<code>polyvalm</code>
<code>ishermitian</code>	<code>planerot</code>	
<code>issymmetric</code>	<code>poly</code>	

Note the following for some of these functions:

- `gpuArray.freqspace` is a static constructor method.

For a list of MATLAB functions that support `gpuArray`, see [Run Built-In Functions on a GPU](#).

---

## pagefun support for mrdivide and inv functions on GPUs

For `gpuArray` inputs, `pagefun` is enhanced to support:

- `@inv`
- `@mrdivide` (for square matrix divisors of sizes up to 32-by-32)

## Enhancements to GPU-enabled linear algebra functions

Many of the linear algebra functions that support `gpuArrays` are enhanced for improved performance. Among those functions that can exhibit improved performance are `svd`, `null`, `eig` (for nonsymmetric input), and `mtimes` (for inner products).

## Parallel data reads from a datastore with MATLAB partition function

The `partition` function can perform a parallel read and partition of a Datastore. For more information, see `partition` and `numpartitions`. See also `Partition a Datastore in Parallel`.

## Using DNS for cluster discovery

In addition to multicast, the `discover` cluster functionality of Parallel Computing Toolbox can now use DNS to locate MATLAB Job Scheduler clusters. For information about cluster discovery, see `Discover Clusters`. For information about configuring and verifying the required DNS SRV record on your network, see `DNS SRV Record`.

## MS-MPI support for local and MATLAB Job Scheduler clusters

On 64-bit Windows platforms, Microsoft MPI (MS-MPI) is now the default MPI implementation for local clusters on the client machine.

For MATLAB Job Scheduler clusters on Windows platforms, you can use MS-MPI by specifying the `-useMSMPI` flag with the `startjobmanager` command.

## Ports and sockets in mdce\_def file

The following parameters are new to the `mdce_def` file for controlling the behavior of MATLAB Job Scheduler clusters.

- `ALL_SERVER_SOCKETS_IN_CLUSTER` — This parameter controls whether all client connections are outbound, or if inbound connections are also allowed.
- `JOBMANAGER_PEERSESSION_MIN_PORT`, `JOBMANAGER_PEERSESSION_MAX_PORT` — These parameters set the range of ports to use when `ALL_SERVER_SOCKETS_IN_CLUSTER = true`.
- `WORKER_PARALLELPOOL_MIN_PORT`, `WORKER_PARALLELPOOL_MAX_PORT` — These parameters set the range of ports to use on worker machines for parallel pools.

For more information and default settings for these parameters, see the appropriate `mdce_def` file for your platform:

- `matlabroot\toolbox\distcomp\bin\mdce_def.bat` (Windows)

- `matlabroot/toolbox/distcomp/bin/mdce_def.sh` (UNIX)

## Compatibility Considerations

By default in this release, `ALL_SERVER_SOCKETS_IN_CLUSTER` is `true`, which makes all connections outbound from the client. For pre-R2015a behavior, set its value to `false`, which also initiates a set of inbound connections to the client from the MATLAB Job Scheduler and workers.

## Improved profiler accuracy for GPU code

The MATLAB profiler now reports more accurate timings for code running on a GPU. For related information, see [Measure Performance on the GPU](#).

## Upgraded CUDA Toolkit version

The parallel computing products are now using CUDA Toolkit version 6.5. To compile CUDA code for CUDAKernel or CUDA MEX files, you must use toolkit version 6.5.

## Discontinued support for GPU devices on 32-bit Windows computers

This release no longer supports GPU devices on 32-bit Windows machines.

## Compatibility Considerations

GPU devices on 32-bit Windows machines are not supported in this release. Instead, use GPU devices on 64-bit machines.

## Discontinued support for parallel computing products on 32-bit Windows computers

In a future release, support will be removed for Parallel Computing Toolbox and MATLAB Distributed Computing Server on 32-bit Windows machines.

## Compatibility Considerations

Parallel Computing Toolbox and MATLAB Distributed Computing Server are still supported on 32-bit Windows machines in this release, but parallel language commands can generate a warning. In a future release, support will be completely removed for these computers, at which time it will not be possible to install the parallel computing products on them.

## matlabpool function removed

The `matlabpool` function has been removed.

## Compatibility Considerations

Calling `matlabpool` now generates an error. You should instead use `parpool` to create a parallel pool.

# R2014b

---

**Version: 6.5**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Parallelization of mapreduce on local workers

If you have Parallel Computing Toolbox installed, and your default cluster profile specifies a local cluster, then execution of mapreduce opens a parallel pool and distributes tasks to the pool workers.

---

**Note** If your default cluster profile specifies some other cluster, the mapreduce function does not use a parallel pool.

---

For more information, see Run mapreduce on a Local Cluster.

## Additional GPU-enabled MATLAB functions, including accumarray, histc, cummax, and cummin

The following functions are new in their support for gpuArrays:

accumarray	cummax	psi
acosd	cummin	rgb2hsv
acotd	del2	roots
acscd	factorial	secd
asecd	gammainc	sind
asind	gammaincinv	sph2cart
atan2d	gradient	subspace
atand	hankel	superiorfloat
betainc	histc	swapbytes
betaincinv	hsv2rgb	tand
cart2pol	isaUnderlying	toeplitz
cart2sph	median	trapz
compan	mode	typecast
corrcoef	nextpow2	unwrap
cosd	null	vander
cotd	orth	
cscd	pol2cart	

Note the following for some of these functions:

- The first input argument to gammainc cannot contain any negative elements.

For a list of MATLAB functions that support gpuArray, see Run Built-In Functions on a GPU.

## pagefun support for mldivide on GPUs

For gpuArray inputs, pagefun is enhanced to support @mldivide for square matrix divisors of sizes up to 32-by-32.

## Additional MATLAB functions for distributed arrays, including fft2, fftn, ifft2, ifftn, cummax, cummin, and diff

The following functions now support distributed arrays with all forms of codistributor (1-D and 2DBC), or are enhanced in their support for this release:

---

besselh	erf	isinteger
besseli	erfc	islogical
besselj	erfcinv	isnumeric
besselk	erfcx	median
bessely	erfinv	mode
beta	fft2	pol2cart
betainc	fftn	psi
betaincinv	gamma	rgb2hsv
betaln	gammainc	sph2cart
cart2pol	gammaincinv	std
cart2sph	gammaln	toeplitz
compan	hankel	trapz
corrcoef	hsv2rgb	unwrap
cov	ifft	vander
cummax	ifft2	var
cummin	ifftn	
diff	isfloat	

Note the following for some of these functions:

- `isfloat`, `isinteger`, `islogical`, and `isnumeric` now return results based on `classUnderlying` of the distributed array.

For a list of MATLAB functions that support distributed arrays, see [MATLAB Functions on Distributed and Codistributed Arrays](#).

## Data Analysis on Hadoop clusters using mapreduce

Parallel Computing Toolbox and MATLAB Distributed Computing Server support the use of Hadoop clusters for the execution environment of mapreduce applications. For more information, see:

- [Configure a Hadoop Cluster](#)
- [Run mapreduce on a Hadoop Cluster](#)

## Discover Clusters Supports Microsoft Windows HPC Server for Multiple Releases

The Discover Clusters dialog can now list Microsoft Windows HPC Server clusters for different releases of parallel computing products. For more information about cluster discovery, see [Discover Clusters](#).

For this functionality, the HPC Server client utilities must be installed on the client machine from which you are discovering. See [Configure Client Computer for HPC Server](#).

## Upgraded CUDA Toolkit Version

The parallel computing products are now using CUDA Toolkit version 6.0. To compile CUDA code for CUDA Kernel or CUDA MEX files, you must use toolkit version 6.0 or earlier.

## Discontinued Support for GPU Devices of Compute Capability 1.3

This release no longer supports GPU devices of compute capability 1.3.

## **Compatibility Considerations**

This release supports only GPU devices of compute capability 2.0 or greater.

## **Discontinued Support for GPU Devices on 32-Bit Windows Computers**

In a future release, support for GPU devices on 32-bit Windows machines will be removed.

## **Compatibility Considerations**

GPU devices on 32-bit Windows machines are still supported in this release, but in a future release support will be completely removed for these devices.



# R2014a

---

**Version: 6.4**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Number of local workers no longer limited to 12

You can now run a local cluster of more than 12 workers on your client machine. Unless you adjust the cluster profile, the default maximum size for a local cluster is the same as the number of computational cores on the machine.

## Additional GPU-enabled MATLAB functions: `interp3`, `interpN`, `besselj`, `bessely`

The following functions are new in their support for `gpuArrays`:

```
besselj  
bessely  
interp3  
interpN
```

For a list of MATLAB functions that support `gpuArray`, see [Run Built-In Functions on a GPU](#).

## Additional GPU enabled Image Processing Toolbox functions: `bwdist`, `imreconstruct`, `iradon`, `radon`

Image Processing Toolbox offers enhanced functionality for some of its functions to perform computations on a GPU. For specific information, see the [Image Processing Toolbox release notes](#). Parallel Computing Toolbox is required to access this functionality.

## Enhancements to GPU-enabled MATLAB functions: `filter` (IIR filters); `pagefun` (additional functions supported); `interp1`, `interp2`, `conv2`, `reshape` (performance improvements)

The following functions are enhanced in their support for `gpuArray` data:

```
conv2  
filter  
interp1  
interp2  
pagefun  
  
rand  
randi  
randn  
reshape
```

Note the following enhancements for some of these functions:

- `filter` now supports IIR filtering.
- `pagefun` is enhanced to support most element-wise `gpuArray` functions. Also, these functions are supported: `@ctranspose`, `@fliplr`, `@flipud`, `@mtimes`, `@rot90`, `@transpose`.
- `rand( ___, 'like', P)` returns a `gpuArray` of random values of the same underlying class as the `gpuArray P`. This enhancement also applies to `randi`, `randn`.

For a list of MATLAB functions that support `gpuArray`, see [Run Built-In Functions on a GPU](#).

## Duplication of an existing job, containing some or all of its tasks

You can now duplicate job objects, allowing you to resubmit jobs that had finished or failed.

---

The syntax to duplicate a job is

```
newjob = recreate(oldjob)
```

where `oldjob` is an existing job object. The `newjob` object has all the same tasks and settable properties as `oldjob`, but receives a new ID. The old job can be in any state; the new job state is `pending`.

You can also specify which tasks from an existing independent job to include in the new job, based on the task IDs. For example:

```
newjob = recreate(oldjob, 'TaskID', [33:48]);
```

For more information, see the `recreate` reference page.

## More MATLAB functions enhanced for distributed arrays

The following functions now support distributed arrays with all forms of codistributor (1-D and 2DBC), or are enhanced in their support for this release:

```
eye  
ifft  
rand  
randi  
randn
```

Note the following enhancements for some of these functions:

- `ifft` and `randi` are new in support of distributed and codistributed arrays.
- `rand(____, 'like', D)` returns a distributed or codistributed array of random values of the same underlying class as the distributed or codistributed array `D`. This enhancement also applies to `randi`, `randn`, and `eye`.

For a list of MATLAB functions that support distributed arrays, see [MATLAB Functions on Distributed and Codistributed Arrays](#).

## GPU MEX Support for Updated MEX

The GPU MEX support for CUDA code now incorporates the latest MATLAB MEX functionality. See [Streamlined MEX compiler setup and improved troubleshooting](#).

## Compatibility Considerations

The latest MEX does not support the `mexopts` files shipped in previous releases. Updated `.xml` files are provided instead. To use the updated MEX functionality and to avoid a warning, replace the `mexopts` file you used in past releases with the appropriate new `.xml` file as described in [Set Up for MEX-File Compilation](#).

## Old Programming Interface Removed

The programming interface characterized by distributed jobs and parallel jobs has been removed. This old interface used functions such as `findResource`, `createParallelJob`, `getAllOutputArguments`, `dfeval`, etc.

## Compatibility Considerations

The functions of the old programming interface now generate errors. You must migrate your code to the interface described in the R2012a release topic “New Programming Interface” on page 23-2.

### **matlabpool Function Being Removed**

The `matlabpool` function is being removed.

## Compatibility Considerations

Calling `matlabpool` continues to work in this release, but now generates a warning. You should instead use `parpool` to create a parallel pool.

### **Removed Support for `parallel.cluster.Mpiexec`**

Support for clusters of type `parallel.cluster.Mpiexec` has been removed.

## Compatibility Considerations

Any attempt to use `parallel.cluster.Mpiexec` clusters now generates an error. As an alternative, consider using the generic scheduler interface, `parallel.cluster.Generic`.

# R2013b

---

**Version: 6.3**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## parpool: New command-line interface (replaces matlabpool), desktop indicator, and preferences for easier interaction with a parallel pool of MATLAB workers

- “Parallel Pool” on page 20-2
- “New Desktop Pool Indicator” on page 20-3
- “New Parallel Preferences” on page 20-3

### Parallel Pool

#### Replaces MATLAB Pool

Parallel pool syntax replaces MATLAB pool syntax for executing parallel language constructs such as `parfor`, `spmd`, `Composite`, and `distributed`. The pool is represented in MATLAB by a `parallel.Pool` object.

The general workflow for these parallel constructs remains the same: When the pool is available, `parfor`, `spmd`, etc., run the same as before. For example:

MATLAB Pool	Parallel Pool
<pre>matlabpool open 4 parfor ii=1:n     X(ii) = myFun(ii); end matlabpool close</pre>	<pre>p = parpool(4) parfor ii=1:n     X(ii) = myFun(ii); end delete(p)</pre>

#### Functions for Pool Control

The following functions provide command-line interface for controlling a parallel pool. See the reference pages for more information about each.

Function	Description
<code>parpool</code>	Start a parallel pool
<code>gcp</code>	Get current parallel pool or start pool
<code>delete</code>	Shut down and delete the parallel pool
<code>addAttachedFiles</code>	Attach files to the parallel pool
<code>listAutoAttachedFiles</code>	List files automatically attached to the parallel pool
<code>updateAttachedFiles</code>	Send updates to files already attached to the parallel pool

#### Asynchronous Function Evaluation on Parallel Pool

You can evaluate functions asynchronously on one or all workers of a parallel pool. Use `parfeval` to evaluate a function on only one worker, or use `parfevalOnAll` to evaluate a function on all workers in the pool.

`parfeval` or `parfevalOnAll` returns an object called a *future*, from which you can get the outputs of the asynchronous function evaluation. You can create an array of futures by calling `parfeval` in a `for`-loop, setting unique parameters for each call in the array.

The following table lists the functions for submitting asynchronous evaluations to a parallel pool, retrieving results, and controlling future objects. See the reference page of each for more details.

Function	Description
<code>parfeval</code>	Evaluate function asynchronously on worker in parallel pool
<code>parfevalOnAll</code>	Evaluate function asynchronously on all workers in parallel pool
<code>fetchOutputs</code>	Retrieve all output arguments from future
<code>fetchNext</code>	Retrieve next available unread future outputs
<code>cancel</code>	Cancel queued or running future
<code>wait</code>	Wait for future to complete

For more information on parallel future objects, including their methods and properties, see the `parallel.Future` reference page.

## Compatibility Considerations

This release continues to support MATLAB pool language usage, but this support might discontinue in future releases. You should update your code as soon as possible to use parallel pool syntax instead.

### New Desktop Pool Indicator

A new icon at the lower-left corner of the desktop indicates the current pool status.



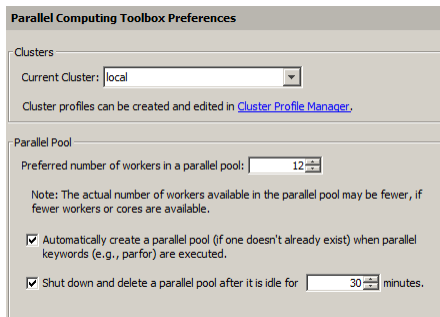
Icon color and tool tips let you know if the pool is busy or ready, how large it is, and when it might time out. You can click the icon to start a pool, stop a pool, or access your parallel preferences.

### New Parallel Preferences

Your MATLAB preferences now include a group of settings for parallel preferences. These settings control general behavior of clusters and parallel pools for your MATLAB session.

You can access your parallel preferences in these ways:

- In the **Environment** section of the **Home** tab, click **Parallel > Parallel Preferences**
- Click the desktop pool indicator icon, and select **Parallel preferences**.
- Type preferences at the command line, and click the Parallel Computing Toolbox node.



Settings in your parallel preferences control the default cluster choice, and the preferred size, automatic opening, and timeout conditions for parallel pools. For more information about these settings, see Parallel Preferences.

## Automatic start of a parallel pool when executing code that uses `parfor` or `spmd`

You can set your parallel preferences so that a parallel pool automatically starts whenever you execute a language construct that runs on a pool, such as `parfor`, `spmd`, `Composite`, `distributed`, `parfeval`, and `parfevalOnAll`.

## Compatibility Considerations

The default preference setting is to automatically start a pool when a parallel language construct requires it. If you want to make sure a pool does not start automatically, you must change your parallel preference setting. You can also work around this by making sure to explicitly start a parallel pool with `parpool` before encountering any code that needs a pool.

By default, a parallel pool will shut down *if idle for 30 minutes*. To prevent this, change the setting in your parallel preferences; or the pool indicator tool tip warns of an impending timeout and provides a link to extend it.

## Option to start a parallel pool without using MPI

You now have the option to start a parallel pool on a local or MATLAB Job Scheduler cluster so that the pool does not support running SPMD constructs. This allows the parallel pool to keep running even if one or more workers aborts during `parfor` execution. You explicitly disable SPMD support when starting the parallel pool by setting its `'SpmdEnabled'` property `false` in the call to the `parpool` function. For example:

```
p = parpool('SpmdEnabled',false);
```

## Compatibility Considerations

Running any code (including MathWorks toolbox code) that uses SPMD constructs, on a parallel pool that was created without SPMD support, will generate errors.

## More GPU-enabled MATLAB functions (e.g., `interp2`, `pagefun`) and Image Processing Toolbox functions (e.g., `bwmorph`, `edge`, `imresize`, and `medfilt2`)

Image Processing Toolbox offers enhanced functionality for some of its functions to perform computations on a GPU. For specific information, see the Image Processing Toolbox release notes. Parallel Computing Toolbox is required to access this functionality.

The following functions are new or enhanced in Parallel Computing Toolbox to support `gpuArrays`:

```
flip  
interp2  
pagefun
```



---

Note the following for some of these functions:

- `pagefun` allows you to iterate over the pages of a `gpuArray`, applying `@mtimes` to each page.  
For more information, see the `pagefun` reference page, or type `help pagefun`.

For complete lists of functions that support `gpuArray`, see [Run Built-In Functions on a GPU](#).

## More MATLAB functions enabled for distributed arrays: `permute`, `ipermute`, and `sortrows`

The following functions now support distributed arrays with all forms of codistributor (1-D and 2DBC), or are enhanced in their support for this release:

<code>ipermute</code>	<code>zeros</code>
<code>permute</code>	<code>ones</code>
<code>sortrows</code>	<code>nan</code>
	<code>inf</code>
<code>cast</code>	<code>true</code>
	<code>false</code>

Note the following enhancements for some of these functions:

- `ipermute`, `permute`, and `sortrows` support distributed arrays for the first time in this release.
- `cast` supports the `'like'` option for distributed arrays, applying the underlying class of one array to another.
- `Z = zeros(___, 'like', P)` returns a distributed array of zeros of the same complexity as distributed array `P`, and same underlying class as `P` if class is not specified in the function call. The same behavior applies to the other similar constructors in the right-hand column of this table.

For more information on any of these functions, type `help distributed.functionname`. For example:

```
help distributed.ipermute
```

For complete lists of MATLAB functions that support distributed arrays, see [MATLAB Functions on Distributed and Codistributed Arrays](#).

## Enhancements to MATLAB functions enabled for GPUs, including `ones`, `zeros`

The following functions are enhanced in their support for `gpuArray` data:

<code>zeros</code>	<code>eye</code>
<code>ones</code>	<code>true</code>
<code>nan</code>	<code>false</code>
<code>inf</code>	
	<code>cast</code>

Note the following enhancements for these functions:

- `Z = zeros(___, 'like', P)` returns a `gpuArray` of zeros of the same complexity as `gpuArray P`, and same underlying class as `P` if class is not specified. The same behavior applies to the other constructor functions listed in this table.

- `cast` also supports the `'like'` option for `gpuArray` input, applying the underlying class of one array to another.

For more information on any of these functions, type `help gpuArray.functionname`. For example:

```
help gpuArray.cast
```

For complete lists of MATLAB functions that support `gpuArray`, see [Run Built-In Functions on a GPU](#).

## gputimeit Function to Time GPU Computations

`gputimeit` is a new function to measure the time to run a function on a GPU. It is similar to the MATLAB function `timeit`, but ensures accurate time measurement on the GPU. For more information and examples, see the `gputimeit` reference page.

## New GPU Random Number Generator NormalTransform Option: Box-Muller

When generating random numbers on a GPU, there is a new option for `'NormalTransform'` called `'BoxMuller'`. The Box-Muller transform allows faster generation of normally distributed random numbers on the GPU.

This new option is the default `'NormalTransform'` setting when using the `Philox4x32-10` or `Threefry4x64-20` generator. The following commands, therefore, use `'BoxMuller'` for `'NormalTransform'`:

```
parallel.gpu.rng(0, 'Philox4x32-10')
parallel.gpu.rng(0, 'Threefry4x64-20')
```

---

**Note** The `'BoxMuller'` option is not supported for the `CombRecursive (mrg32k3a)` generator

---

## Compatibility Considerations

In previous releases, the default `'NormalTransform'` setting when using the `Philox4x32-10` or `Threefry4x64-20` generator on a GPU was `'Inversion'`. If you used either of these generators with the default `'NormalTransform'` and you want to continue with the same behavior, you must explicitly set the `'NormalTransform'` with either of these commands:

```
stream = parallel.gpu.RandStream('Philox4x32-10', 'NormalTransform', 'Inversion')
parallel.gpu.RandStream.setGlobalStream(stream)

stream = parallel.gpu.RandStream('Threefry4x64-20', 'NormalTransform', 'Inversion')
parallel.gpu.RandStream.setGlobalStream(stream)
```

## Upgraded MPICH2 Version

The parallel computing products are now shipping MPICH2 version 1.4.1p1 on all platforms.

---

## Compatibility Considerations

If you use your own MPI builds, you might need to create new builds compatible with this latest version, as described in [Use Different MPI Builds on UNIX Systems](#).

## Discontinued Support for GPU Devices of Compute Capability 1.3

In a future release, support for GPU devices of compute capability 1.3 will be removed. At that time, a minimum compute capability of 2.0 will be required.

## Compatibility Considerations

In R2013b, any use of `gpuDevice` to select a GPU with compute capability 1.3, generates a warning. The device is still supported in this release, but in a future release support will be completely removed for these 1.3 devices.

## Discontinued Support for `parallel.cluster.Mpiexec`

Support for clusters of type `parallel.cluster.Mpiexec` is being discontinued.

## Compatibility Considerations

In R2013b, any use of `parallel.cluster.Mpiexec` clusters generates a warning. In a future release, support will be completely removed.



# R2013a

---

**Version: 6.2**

**New Features**

**Bug Fixes**

## GPU-enabled functions in Image Processing Toolbox and Phased Array System Toolbox

More toolboxes offer enhanced functionality for some of their functions to perform computations on a GPU. For specific information about these other toolboxes, see their respective release notes. Parallel Computing Toolbox is required to access this functionality.

### More MATLAB functions enabled for use with GPUs, including `interp1` and `ismember`

The following functions are enhanced to support `gpuArray` data:

<code>interp1</code>	<code>ismember</code>
<code>isfloat</code>	<code>isnumeric</code>
<code>isinteger</code>	

Note the following for some of these functions:

- `interp1` supports only the linear and nearest interpolation methods.
- `isfloat`, `isinteger`, and `isnumeric` now return results based on `classUnderlying` of the `gpuArray`.
- `ismember` does not support the 'rows' or 'legacy' option for `gpuArray` input.

For complete lists of functions that support `gpuArray`, see [Built-In Functions That Support `gpuArray`](#).

### Enhancements to MATLAB functions enabled for GPUs, including `arrayfun`, `svd`, and `mldivide` (`\`)

The following functions are enhanced in their support for `gpuArray` data:

<code>arrayfun</code>	<code>mrdivide</code>
<code>bsxfun</code>	<code>svd</code>
<code>mldivide</code>	

Note the following enhancements for some of these functions:

- `arrayfun` and `bsxfun` support indexing and accessing variables of outer functions from within nested functions.
- `arrayfun` supports singleton expansion of all arguments for all operations. For more information, see the `arrayfun` reference page.
- `mldivide` and `mrdivide` support all rectangular arrays.
- `svd` can perform economy factorizations.

For complete lists of MATLAB functions that support `gpuArray`, see [Built-In Functions That Support `gpuArray`](#).

---

## Ability to launch CUDA code and manipulate data contained in GPU arrays from MEX-functions

You can now compile MEX-files that contain CUDA code, to create functions that support `gpuArray` input and output. This functionality is supported only on 64-bit platforms (win64, glnxa64, maci64). For more information and examples, see [Execute MEX-Functions Containing CUDA Code](#).

For a list of C functions supporting this capability, see the group of [C Functions in GPU Computing](#).

## Automatic detection and transfer of files required for execution in both batch and interactive workflows

Parallel Computing Toolbox can now automatically attach files to a job so that workers have the necessary code files for evaluating tasks. When you set a job object's `AutoAttachFiles` to true, an analysis determines what files on the client machine are necessary for the evaluation of your job, and those files are automatically attached to the job and sent to the worker machines.

You can set the `AutoAttachFiles` property in the Cluster Profile Manager, or at the command-line. To get a listing of the files that are automatically attached, use the `listAutoAttachedFiles` method on a job or task object.

For more information, see [Pass Data to and from Worker Sessions](#).

If the `AutoAttachFiles` property in the cluster profile for the MATLAB pool is set to true, MATLAB performs an analysis on `spmd` blocks and `parfor`-loops to determine what code files are necessary for their execution, then automatically attaches those files to the MATLAB pool job so that the code is available to the workers.

When you use the MATLAB editor to update files on the client that are attached to a `matlabpool`, those updates are automatically propagated to the workers in the pool.

## More MATLAB functions enabled for distributed arrays

The following functions now support distributed arrays with all forms of codistributor (1-D and 2DBC), or are enhanced in their support for this release:

<code>cumprod</code>	<code>qr</code>
<code>cumsum</code>	<code>prod</code>
<code>eig</code>	

Note the following enhancements for some of these functions:

- `eig` supports generalized eigenvalues for symmetric matrices.
- `qr` supports column pivoting.
- `cumprod`, `cumsum`, and `prod` now support all integer data types; and `prod` accepts the optional 'native' or 'double' argument.





# R2012b

---

**Version: 6.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## More MATLAB functions enabled for GPUs, including convn, cov, and normest

- “gpuArray Support” on page 22-2
- “MATLAB Code on the GPU” on page 22-2

### gpuArray Support

The following functions are enhanced to support gpuArray data, or are expanded in their support:

bitget	cov	normest
bitset	issparse	pow2
cond	mpower	var
convn	nnz	

The following functions are not methods of the gpuArray class, but they now work with gpuArray data:

blkdiag	ismatrix	isvector
cross	isrow	std
iscolumn		

For complete lists of functions that support gpuArray, see Built-In Functions That Support gpuArray.

### MATLAB Code on the GPU

bsxfun now supports the same subset of the language on a GPU that arrayfun does.

GPU support is extended to include the following MATLAB code in functions called by arrayfun and bsxfun to run on the GPU:

```
bitget  
bitset  
pow2
```

## GPU-enabled functions in Neural Network Toolbox, Phased Array System Toolbox, and Signal Processing Toolbox

A number of other toolboxes now support enhanced functionality for some of their functions to perform computations on a GPU. For specific information about these other toolboxes, see their respective release notes. Parallel Computing Toolbox is required to access this functionality.

## Performance improvements to GPU-enabled MATLAB functions and random number generation

The performance of some MATLAB functions and random number generation on GPU devices is improved in this release.

You now have a choice of three random generators on the GPU: the combined multiplicative recursive MRG32K3A, the Philox4x32-10, and the Threefry4x64-20. For information on these generators and how to select them, see Control the Random Stream for gpuArray. For information about generating

---

random numbers on a GPU, and a comparison between GPU and CPU generation, see Control Random Number Streams.

## Automatic detection and selection of specific GPUs on a cluster node when multiple GPUs are available on the node

When multiple workers run on a single compute node with multiple GPU devices, the devices are automatically divided up among the workers. If there are more workers than GPU devices on the node, multiple workers share the same GPU device. If you put a GPU device in 'exclusive' mode, only one worker uses that device. As in previous releases, you can change the device used by any particular worker with the `gpuDevice` function.

## More MATLAB functions enabled for distributed arrays, including sparse constructor, bsxfun, and repmat

The following functions now support distributed arrays with all forms of codistributor (1-D and 2DBC), or are enhanced in their support for this release:

<code>atan2d</code>	<code>mrdivide</code>	<code>struct2cell</code>
<code>bsxfun</code>	<code>repmat</code>	<code>typecast</code>
<code>cell2struct</code>	<code>sparse</code>	

Note the following enhancements for some of these functions:

- `cell2struct`, `struct2cell`, and `typecast` now support 2DBC in addition to 1-D distribution.
- `mrdivide` is now fully supported, and is no longer limited to accepting only scalars for its second argument.
- `sparse` is now fully supported for all distribution types.

This release also offers improved performance of `fft` functions for long vectors as distributed arrays.

## Detection of MATLAB Distributed Computing Server clusters that are available for connection from user desktops through Profile Manager

You can let MATLAB discover clusters for you. Use either of the following techniques to discover those clusters which are available for you to use:

- On the **Home** tab in the **Environment** section, click **Parallel > Discover Clusters**.
- In the Cluster Profile Manager, click **Discover Clusters**.

For more information, see Discover Clusters.

## gpuArray Class Name

The object type formerly known as a `GPUArray` has been renamed to `gpuArray`. The corresponding class name has been changed from `parallel.gpu.GPUArray` to the shorter `gpuArray`. The name of the function `gpuArray` remains unchanged.

## Compatibility Considerations

You cannot load `gpuArray` objects from files that were saved in previous versions.

Code that uses the old class name must be updated to use the shorter new name. For example, the functions for directly generating `gpuArrays` on the GPU:

Previous version form	New version form
<code>parallel.gpu.GPUArray.rand</code> <code>parallel.gpu.GPUArray.ones</code>	<code>gpuArray.rand</code> <code>gpuArray.ones</code>
etc.	etc.

## Diary Output Now Available During Running Task

Diary output from tasks (including those of batch jobs) can now be obtained while the task is still running. The diary is text output that would normally be sent to the Command Window. Now this text is appended to the task's `Diary` property as the text is generated, rather than waiting until the task is complete. You can read this property at any time. Diary information is accumulated only if the job's `CaptureDiary` property value is `true`. (**Note:** This feature is not yet available for SOA jobs on HPC Server clusters.)

# R2012a

---

**Version: 6.0**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## New Programming Interface

This release provides a new programming interface for accessing clusters, jobs, and tasks.

### General Concepts and Phrases

This table maps some of the concepts and phrases from the old interface to the new.

Previous Interface	New Interface in R2012a
MathWorks job manager	MATLAB Job Scheduler
Third-party or local scheduler	Common job scheduler (CJS)
Configuration	Profile
Scheduler	Cluster

The following code examples compare programming the old and new interfaces, showing some of the most common commands and properties. Note that most differences involve creating a cluster object instead of a scheduler object, and some of the property and method names on the job. After the example are tables listing some details of these new objects.

Previous Interface	New Interface
<pre>sched = findResource('scheduler',...     'Configuration','local'); j = createJob(sched,...     'PathDependencies',{'/share/app/'},...     'FileDependencies',{'funa.m','funb.m'}); createTask(j,@myfun,1,{3,4}); submit(j); waitForState(j); results = j.getAllOutputArguments;</pre>	<pre>clust = parcluster('local'); j = createJob(clust,...     'AdditionalPaths',{'/share/app/'}),...     'AttachedFiles',{'funa.m','funb.m'}); createTask(j,@myfun,1,{3,4}); submit(j); wait(j); results = j.fetchOutputs;</pre>

### Objects

These tables compare objects in the previous and new interfaces.

Previous Scheduler Objects	New Cluster Objects
ccsscheduler	parallel.cluster.HPCServer
genericscheduler	parallel.cluster.Generic
jobmanager	parallel.cluster.MJS
localscheduler	parallel.cluster.Local
lsfscheduler	parallel.cluster.LSF
mpiexec	parallel.cluster.Mpiexec
pbsproscheduler	parallel.cluster.PBSPro
torquescheduler	parallel.cluster.Torque

For information on each of the cluster objects, see the `parallel.Cluster` reference page.

Previous Job Objects	New Job Objects
job (distributed job)	parallel.job.MJSIndependentJob

Previous Job Objects	New Job Objects
matlabpooljob	parallel.job.MJSCommunicatingJob where ('Type' = 'Pool')
paralleljob	parallel.job.MJSCommunicatingJob where ('Type' = 'SPMD')
simplejob	parallel.job.CJSIndependentJob
simplematlabpooljob	parallel.job.CJSCommunicatingJob ( 'Type' = 'Pool' )
simpleparalleljob	parallel.job.CJSCommunicatingJob ( 'Type' = 'SPMD' )

For information on each of the job objects, see the `parallel.Job` reference page.

Previous Task Objects	New Task Objects
task	parallel.task.MJSTask
simpletask	parallel.task.CJSTask

For information on each of the task objects, see the `parallel.Task` reference page.

Previous Worker Object	New Worker Objects
worker	parallel.cluster.MJSWorker, parallel.cluster.CJSWorker

For information on each of the worker objects, see the `parallel.Worker` reference page.

## Functions and Methods

This table compares some functions and methods of the old interface to those of the new. Many functions do not have a name change in the new interface, and are not listed here. Not all functions are available for all cluster types.

Previous Name	New Name
findResource	parcluster
createJob	createJob (no change)
createParallelJob	createCommunicatingJob (where 'Type' = 'SPMD')
createMatlabPoolJob	createCommunicatingJob (where 'Type' = 'Pool')
destroy	delete
clearLocalPassword	logout
createTask	createTask (no change)
getAllOutputArguments	fetchOutputs
getJobSchedulerData	getJobClusterData
setJobSchedulerData	setJobClusterData

Previous Name	New Name
getCurrentJobmanager	getCurrentCluster
getFileDependencyDir	getAttachedFilesFolder

## Properties

In addition to a few new properties on the objects in the new interface, some other properties have new names when using the new interface, according to the following tables.

Previous Scheduler Properties	New Cluster Properties
DataLocation	JobStorageLocation
Configuration	Profile
HostAddress	AllHostAddresses
Computer	ComputerType
ClusterOsType	OperatingSystem
IsUsingSecureCommunication	HasSecureCommunication
SchedulerHostname, MasterName, Hostname	Host
ParallelSubmissionWrapperScript	CommunicatingJobWrapper
ClusterSize	NumWorkers
NumberOfBusyWorkers	NumBusyWorkers
NumberOfIdleWorkers	NumIdleWorkers
SubmitFcn	IndependentSubmitFcn
ParallelSubmitFcn	CommunicatingSubmitFcn
DestroyJobFcn	DeleteJobFcn
DestroyTaskFcn	DeleteTaskFcn

Previous Job Properties	New Job Properties
FileDependencies	AttachedFiles
PathDependencies	AdditionalPaths
MinimumNumberOfWorkers, MaximumNumberOfWorkers	NumWorkersRange

Previous Task Properties	New Task Properties
MaximumNumberOfRetries	MaximumRetries

## Getting Help

In addition to these changes, there are some new properties and methods, while some old properties are not used in the new interface. For a list of the methods and properties available for clusters, jobs, and tasks, use the following commands for help on each class type:

```
help parallel.Cluster
help parallel.Job
```



---

```
help parallel.Task
```

```
help parallel.job.CJSIndependentJob
help parallel.job.CJSCommunicatingJob
help parallel.task.CJSTask
```

```
help parallel.job.MJSIndependentJob
help parallel.job.MJSCommunicatingJob
help parallel.task.MJSTask
```

There might be slight changes in the supported format for properties whose names are still the same. To get help on an individual method or property, the general form of the command is:

```
help parallel.obj-type.method-or-property-name
```

You might need to specify the subclass in some cases, for example, where different cluster types use properties differently. The following examples display the help for specified methods and properties of various object types:

```
help parallel.cluster.LSF.JobStorageLocation
help parallel.Job.fetchOutputs
help parallel.job.MJSIndependentJob.FinishedFcn
help parallel.Task.delete
```

## Compatibility Considerations

### Jobs

*This release still supports the old form of interface, however, the old interface and the new interface are not compatible in the same job. For example, a job manager scheduler object that you create with `findResource` requires that you use only the old interface methods and properties, and cannot be used for creating a communicating job (`createCommunicatingJob`). A cluster that was defined with `parcluster` must use the new interface, and cannot be used to create a parallel job (`createParallelJob`).*

Graphical interfaces provide access only to the new interface. The Configurations Manager is no longer available and is replaced by the Cluster Profile Manager. Actions that use profiles automatically convert and upgrade your configurations to profiles. Therefore, if you already have a group of configurations, the first time you open the Cluster Profile Manager, it should already be populated with your converted profiles. Furthermore, you can specify cluster profiles when using the old interface in places where configurations are expected.

One job manager (or MATLAB Job Scheduler) can accommodate jobs of both the old and new interface at the same time.

### Creating and Finding Jobs and Tasks

In the old interface, to create or find jobs or tasks, you could specify property name and value pairs in structures or cell arrays, and then provide the structure or cell array as an input argument to the function you used. In the new interface, property names and values must be pairs of separate arguments, with the property name as a string expression and its value of the appropriate type. This applies to the functions `createJob`, `createCommunicatingJob`, `createTask`, `findJob`, and `findTask`.

### Batch

Jobs created by the `batch` command use the new interface, even if you specify configurations or properties using the old interface. For example, the following code generates two identical jobs of the new interface, even though job `j2` is defined with an old interface property. The script `randScript` contains just the one line of code, `R = rand(3)`, and the default profile is `local`.

```
j1 = batch('randScript','AdditionalPaths','c:\temp');
j1.wait;
R1 = j1.load('R');
```

or

```
j2 = batch('randScript','PathDependencies','c:\temp');
j2.wait;
R2 = j2.load('R');
```

```
whos
  Name      Size      Bytes  Class
  R1        1x1        248    struct
  R2        1x1        248    struct
  j1        1x1        112    parallel.job.CJSIndependentJob
  j2        1x1        112    parallel.job.CJSIndependentJob
```

### Communicating Job Wrapper

In the old interface, for a parallel job in an LSF, TORQUE, or PBS Pro scheduler, you would call the scheduler's `setupForParallelExecution` method with the necessary arguments so that the toolbox could automatically set the object's `ClusterOSType` and `ParallelSubmissionWrapperScript` properties, thus determining which wrapper was used. In the new interface, with a communicating job you only have to set the LSF, Torque, or PBSPro cluster object's `OperatingSystem` and `CommunicatingJobWrapper` properties, from which the toolbox calculates which wrapper to use. For more information about these properties and their possible values, in MATLAB type

```
help parallel.cluster.LSF.CommunicatingJobWrapper
```

You can change the LSF to PBSPro or Torque in this command, as appropriate.

### Enhanced Example Scripts

An updated set of example scripts is available in the product for using a generic scheduler with the new programming interface. These currently supported scripts are provided in the folder:

```
matlabroot/toolbox/distcomp/examples/integration
```

For more information on these scripts and their updates, see the README file provided in each subfolder, or see `Supplied Submit and Decode Functions`.

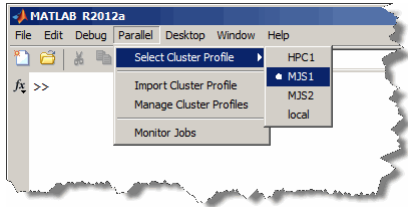
### Compatibility Considerations

Scripts that use the old programming interface are provided in the folder `matlabroot/toolbox/distcomp/examples/integration/old`. The scripts that resided in this folder in previous releases are no longer available. The scripts currently in this folder might be removed in future releases.

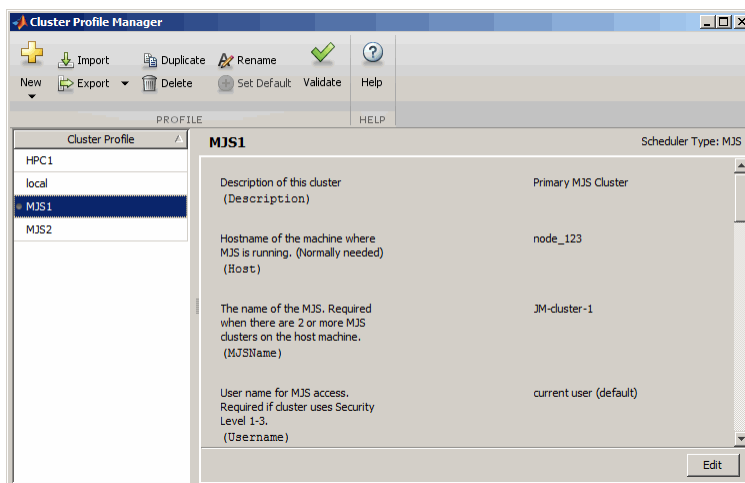
## Cluster Profiles

### New Cluster Profile Manager

Cluster profiles replace parallel configurations for defining settings for clusters and their jobs. You can access profiles and the Cluster Profile Manager from the desktop **Parallel** menu. From this menu you can select or import profiles. To choose a profile as the default, select the desktop menu **Parallel** > **Select Cluster Profile**. The current default profile is indicated with a bold dot.



The Cluster Profile Manager lets you create, edit, validate, import, and export profiles, among other actions. To open the Cluster Profile Manager, select **Parallel** > **Manage Cluster Profiles**.



For more information about cluster profiles and the Cluster Profile Manager, see Cluster Profiles.

### Programming with Profiles

These commands provide access to profiles and the ability to create cluster objects.

Function	Description
<code>p = parallel.clusterProfiles</code>	List of your profiles
<code>parallel.defaultClusterProfile</code>	Specifies which profile to use by default
<code>c = parcluster('clustername')</code>	Creates cluster object, c, for accessing parallel compute resources
<code>c.saveProfile</code>	Saves changes of cluster property values to its current profile

Function	Description
<code>c.saveAsProfile</code>	Saves cluster properties to a profile of the specified name, and sets that as the current profile for this cluster
<code>parallel.importProfile</code>	Import profiles from specified <code>.settings</code> file
<code>parallel.exportProfile</code>	Export profiles to specified <code>.settings</code> file

### Profiles in Compiled Applications

Because compiled applications include the current profiles of the user who compiles, in most cases the application has the profiles it needs. When other profiles are needed, a compiled application can also import profiles that have been previously exported to a `.settings` file. The new `ParallelProfile` key supports exported parallel configuration `.mat` files and exported cluster profile `.settings` files; but this might change in a future release. For more information, see [Export Profiles for MATLAB Compiler](#).

### Compatibility Considerations

In past releases, when a compiled application imported a parallel configuration, that configuration would overwrite a configuration of the same if it existed. In this release, imported profiles are renamed if profiles already exist with the same name; so the same profile might have a different name in the compiled application than it does in the exported profile file.

### Enhanced GPU Support

#### GPUArray Support

The following functions are added to those that support GPUArray data, or are enhanced in their support for this release:

<code>beta</code>	<code>ifft</code>	<code>mldivide</code>
<code>betaln</code>	<code>ifft2</code>	<code>min</code>
<code>bsxfun</code>	<code>ifftn</code>	<code>mrdivide</code>
<code>circshift</code>	<code>ind2sub</code>	<code>norm</code>
<code>det</code>	<code>int2str</code>	<code>num2str</code>
<code>eig</code>	<code>inv</code>	<code>permute</code>
<code>fft</code>	<code>ipermute</code>	<code>qr</code>
<code>fft2</code>	<code>isequaln</code>	<code>shiftdim</code>
<code>fftn</code>	<code>issorted</code>	<code>sprintf</code>
<code>fprintf</code>	<code>mat2str</code>	<code>sub2ind</code>
<code>full</code>	<code>max</code>	

Note the following enhancements and restrictions to some of these functions:

- GPUArray usage now supports all data types supported by MATLAB, except `int64` and `uint64`.
- For the list of functions that `bsxfun` supports, see the [bsxfun reference page](#).
- The full range of syntax is now supported for `fft`, `fft2`, `fftn`, `ifft`, `ifft2`, and `ifftn`.
- `eig` now supports all matrices, symmetric or not.
- `issorted` supports only vectors, not matrices.
- `max` and `min` can now return two output arguments, including an index vector.

- `mldivide` supports complex arrays. It also supports overdetermined matrices (with more rows than columns), with no constraints on the second input.
- `mrdivide` supports underdetermined matrices (more columns than rows) as the second input argument.
- `norm` now supports the form `norm(X,2)`, where `X` is a matrix.

The following functions are not methods of the `GPUArray` class, but they do work with `GPUArray` data:

<code>angle</code>	<code>ifftshift</code>	<code>rank</code>
<code>fliplr</code>	<code>kron</code>	<code>squeeze</code>
<code>flipud</code>	<code>mean</code>	<code>rot90</code>
<code>flipdim</code>	<code>perms</code>	<code>trace</code>
<code>fftshift</code>		

### Reset or Deselect GPU Device

Resetting a GPU device clears your `GPUArray` and `CUDAKernel` data from the device. There are two ways to reset a GPU device, while still keeping it as the currently selected device. You can use the `reset` function, or you can use `gpuDevice(idx)` with the current device's index for `idx`. For example, use `reset`:

```
idx = 1
g = gpuDevice(idx)
reset(g)
```

Alternatively, call `gpuDevice` again with the same index argument:

```
idx = 1
g = gpuDevice(idx)
.
.
.
g = gpuDevice(idx) % Resets GPU device, clears data
```

To deselect the current device, use `gpuDevice([ ])` with an empty argument (as opposed to no argument). This clears the GPU of all arrays and kernels, and invalidates variables in the workspace that point to such data.

### Asynchronous GPU Calculations and Wait

All GPU calculations now run asynchronously with MATLAB. That is, when you initiate a calculation on the GPU, MATLAB continues execution while the GPU runs its calculations at the same time. The `wait` command now accommodates a GPU device, so that you can synchronize MATLAB and the GPU. The form of the command is

```
wait(gpudev)
```

where `gpudev` is the object representing the GPU device to wait for. At this command, MATLAB waits until all current calculations complete on the specified device.

## Compatibility Considerations

In previous releases, MATLAB and the GPU were synchronous, so that any calls to the GPU had to complete before MATLAB proceeded to the next command. This is no longer the case. Now MATLAB continues while the GPU is running. The `wait` command lets you time GPU code execution.

### Verify GPUArray or CUDAKernel Exists on the Device

The new function `existsOnGPU` lets you verify that a `GPUArray` or `CUDAKernel` exists on the current GPU device, and that its data is accessible from MATLAB. It is possible to reset the GPU device, so that a `GPUArray` or `CUDAKernel` object variable still exists in your MATLAB workspace even though it is no longer available on the GPU. For example, you can reset a GPU device using the command `gpuDevice(index)` or `reset(dev)`:

```
index = 1;
g = gpuDevice(index);
R = parallel.gpu.GPUArray.rand(4,4)
    0.5465    0.3000    0.4067    0.6110
    0.9024    0.8965    0.6635    0.7709
    0.8632    0.7481    0.9901    0.0420
    0.2307    0.7008    0.7516    0.5059

existsOnGPU(R)
    1

reset(g); % Resets GPU device
existsOnGPU(R)
    0

R % View GPUArray contents
Data no longer exists on the GPU.
```

Any attempt to use the data from `R` generates an error.

### MATLAB Code on the GPU

GPU support is extended to include the following MATLAB code in functions called by `arrayfun` to run on the GPU:

<code>and</code>	<code>intmin</code>	<code>power</code>
<code>beta</code>	<code>ldivide</code>	<code>rdivide</code>
<code>beta1n</code>	<code>le</code>	<code>realmax</code>
<code>eq</code>	<code>lt</code>	<code>realmin</code>
<code>ge</code>	<code>minus</code>	<code>times</code>
<code>gt</code>	<code>ne</code>	<code>uint8</code>
<code>int8</code>	<code>not</code>	<code>uint16</code>
<code>int16</code>	<code>or</code>	
<code>intmax</code>	<code>plus</code>	

Note the following enhancements and restrictions to some of these functions:

- `plus`, `minus`, `ldivide`, `rdivide`, `power`, `times`, and other arithmetic, comparison or logical operator functions were previously supported only when called with their operator symbol. Now they are supported in their functional form, so can be used as direct argument inputs to `arrayfun` and `bsxfun`.

- 
- `arrayfun` and `bsxfun` on the GPU now support the integer data types `int8`, `uint8`, `int16`, and `uint16`.

The code in your function can now call any functions defined in your function file or on the search path. You are no longer restricted to calling only those supported functions listed in the table of Supported MATLAB Code.

### **Set CUDA Kernel Constant Memory**

The new `setConstantMemory` method on the `CUDAKernel` object lets you set kernel constant memory from MATLAB. For more information, see the `setConstantMemory` reference page.

### **Latest NVIDIA CUDA Device Driver**

This version of Parallel Computing Toolbox GPU functionality supports only the latest NVIDIA CUDA device driver.

## **Compatibility Considerations**

Earlier versions of the toolbox supported earlier versions of CUDA device driver. Always make sure you have the latest CUDA device driver.

## **Enhanced Distributed Array Support**

### **Newly Supported Functions**

The following functions now support distributed arrays with all forms of codistributor (1-D and 2DBC), or are enhanced in their support for this release:

`isequaln`

### **Random Number Generation on Workers**

MATLAB worker sessions now generate random number values using the combined multiplicative recursive generator (`mrg32k3a`) by default.

## **Compatibility Considerations**

In past releases, MATLAB workers used the same default generator as a MATLAB client session. This is no longer the case.





# R2011b

---

**Version: 5.2**

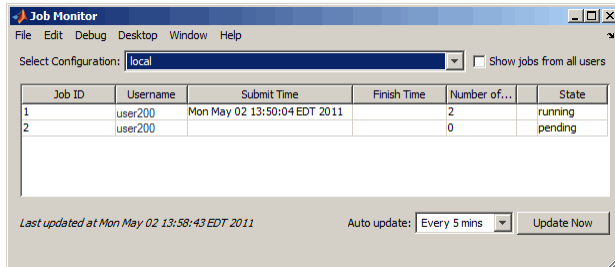
**New Features**

**Bug Fixes**

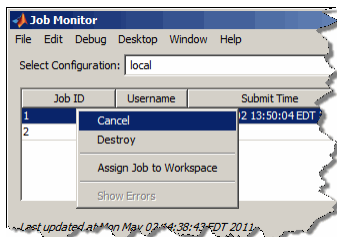
**Compatibility Considerations**

## New Job Monitor

The Job Monitor is a tool that lets you track the jobs you have submitted to a cluster. It displays the jobs for the scheduler determined by your selection of a parallel configuration. Open the Job Monitor from the MATLAB desktop by selecting **Parallel > Job Monitor**.



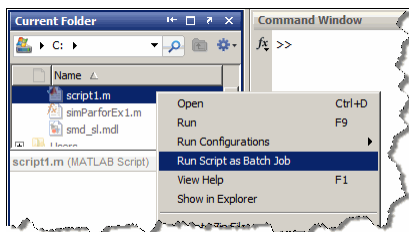
Right-click a job in the list to select a command from the context menu for that job:



For more information about the Job Monitor and its capabilities, see [Job Monitor](#).

## Run Scripts as Batch Jobs from the Current Folder Browser

From the Current Folder browser, you can run a MATLAB script as a batch job by browsing to the file's folder, right-clicking the file, and selecting **Run Script as Batch Job**. The batch job runs on the cluster identified by the current default parallel configuration. The following figure shows the menu option to run the script from the file `script1.m`:



## Number of Local Workers Increased to Twelve

You can now run up to 12 local workers on your MATLAB client machine. If you do not specify the number of local workers in a command or configuration, the default number of local workers is determined by the value of the local scheduler's `ClusterSize` property, which by default equals the number of computational cores on the client machine.

---

## Enhanced GPU Support

### Latest NVIDIA CUDA Device Driver

This version of Parallel Computing Toolbox GPU functionality supports only the latest NVIDIA CUDA device driver.

### Compatibility Considerations

Earlier versions of the toolbox supported earlier versions of CUDA device driver. Always make sure you have the latest CUDA device driver.

### Deployment of GPU Applications

MATLAB Compiler generated standalone executables and components now support applications that use the GPU.

### Random Number Generation

You can now directly create arrays of random numbers on the GPU using these new static methods for GPUArray objects:

```
parallel.gpu.GPUArray.rand
parallel.gpu.GPUArray.randi
parallel.gpu.GPUArray.randn
```

The following functions set the GPU random number generator seed and stream:

```
parallel.gpu.rng
parallel.gpu.RandStream
```

Also, `arrayfun` called with GPUArray data now supports `rand`, `randi`, and `randn`. For more information about using `arrayfun` to generate random matrices on the GPU, see [Generating Random Numbers on the GPU](#).

### GPUArray Support

The following functions now support GPUArray data:

<code>chol</code>	<code>isnan</code>	<code>norm</code>
<code>diff</code>	<code>lu</code>	<code>not</code>
<code>eig</code>	<code>max</code>	<code>repmat</code>
<code>find</code>	<code>min</code>	<code>sort</code>
<code>isfinite</code>	<code>mldivide</code>	<code>svd</code>
<code>isinf</code>		

`mldivide` supports complex arrays. It also supports overdetermined matrices (with more rows than columns) when the second input argument is a column vector (has only one column).

`eig` supports only symmetric matrices.

`max` and `min` return only one output argument; they do not return an index vector.

The following functions are not methods of the GPUArray class, but they do work with GPUArray data:

angle	flipdim	mean
beta	fftshift	perms
betaln	ifftshift	squeeze
fliplr	kron	rot90
flipud		

The default display of GPUArray variables now shows the array contents. In previous releases, the display showed some of the object properties, but not the contents. For example, the new enhanced display looks like this:

```
M = gpuArray(magic(3))
M =
     8     1     6
     3     5     7
     4     9     2
```

To see that M is a GPUArray, use the `whos` or `class` function.

### **MATLAB Code on the GPU**

GPU support is extended to include the following MATLAB code in functions called by `arrayfun` to run on the GPU:

```
rand
randi
randn
xor
```

Also, the handle passed to `arrayfun` can reference a simple function, a subfunction, a nested function, or an anonymous function. The function passed to `arrayfun` can call any number of its subfunctions. The only restriction is that when running on the GPU, nested and anonymous functions do not have access to variables in the parent function workspace. For more information on function structure and relationships, see [Types of Functions](#).

## **Enhanced Distributed Array Support**

### **Newly Supported Functions**

The following functions are enhanced to support distributed arrays, supporting all forms of codistributor (1-D and 2DBC):

```
inv
meshgrid
ndgrid
sort
```

The following functions can now directly construct codistributed arrays:

```
codistributed.linspace(m, n, ..., codist)
codistributed.logspace(m, n, ..., codist)
```

## **Conversion of Error and Warning Message Identifiers**

For R2011b, error and warning message identifiers have changed in Parallel Computing Toolbox.

---

## Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a `try/catch` statement and performs an action based on a specific error identifier.

For example, the `'distcomp:old:ID'` identifier has changed to `'parallel:similar:ID'`. If your code checks for `'distcomp:old:ID'`, you must update it to check for `'parallel:similar:ID'` instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

To determine the identifier for an error, run the following commands just after you see the error:

```
exception = MException.last;  
MSGID = exception.identifier;
```

---

**Tip** Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so that it runs warning-free.

---

## Task Error Properties Updated

If there is an error during task execution, the `task Error` property now contains the non-empty `MException` object that is thrown. If there was no error during task execution, the `Error` property is empty.

The identifier and message properties of this object are now the same as the task's `ErrorIdentifier` and `ErrorMessage` properties, respectively. For more information about these properties, see the `Error`, `ErrorIdentifier`, and `ErrorMessage` reference pages.

## Compatibility Considerations

In past releases, when there was no error, the `Error` property contained an `MException` object with empty data fields, generated by `MException(' ', '')`. Now to determine if a task is error-free, you can query the `Error` property itself to see if it is empty:

```
didTaskError = ~isempty(t.Error)
```

where `t` is the task object you are examining.



# R2011a

---

**Version: 5.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Deployment of Local Workers

MATLAB Compiler generated standalone executables and libraries from parallel applications can now launch up to eight local workers without requiring MATLAB Distributed Computing Server software.

## New Desktop Indicator for MATLAB Pool Status

When you first open a MATLAB pool from your desktop session, an indicator appears in the lower-right corner of the desktop to show that this desktop session is connected to an open pool. The number indicates how many workers are in the pool.



When you close the pool, the indicator remains displayed and shows a value of 0.

## Enhanced GPU Support

### Static Methods to Create GPUArray

The following new static methods directly create GPUArray objects:

```
parallel.gpu.GPUArray.linspace  
parallel.gpu.GPUArray.logspace
```

### GPUArray Support

The following functions are enhanced to support GPUArray data:

```
cat  
colon  
conv  
conv2  
cumsum  
cumprod  
eps  
filter  
filter2  
horzcat  
meshgrid  
ndgrid  
plot  
subsasgn  
subsindex  
subsref  
vertcat
```

and all the plotting related functions.

### GPUArray Indexing

Because GPUArray now supports `subsasgn` and `subsref`, you can index into a GPUArray for assigning and reading individual elements.

For example, create a GPUArray and assign the value of an element:



---

```
n = 1000;
D = parallel.gpu.GPUArray.eye(n);
D(1,n) = pi
```

Create a GPUArray and read the value of an element back into the MATLAB workspace:

```
m = 500;
D = parallel.gpu.GPUArray.eye(m);
one = gather(D(m,m))
```

### **MATLAB Code on the GPU**

GPU support is extended to include the following MATLAB code in functions called by `arrayfun` to run on the GPU:

```
&, |, ~, &&, ||,
while, if, else, elseif, for, return, break, continue, eps
```

You can now call `eps` with string inputs, so your MATLAB code running on the GPU can include `eps('single')` and `eps('double')`.

### **NVIDIA CUDA Driver 3.2 Support**

This version of Parallel Computing Toolbox GPU functionality supports only NVIDIA CUDA device driver 3.2.

## **Compatibility Considerations**

Earlier versions of the toolbox supported earlier versions of CUDA device driver. If you have an older driver, you must upgrade to CUDA device driver version 3.2.

## **Distributed Array Support**

### **Newly Supported Functions**

The following functions are enhanced to support distributed arrays, supporting all forms of codistributor (1-D and 2DBC):

```
arrayfun
cat
reshape
```

### **Enhanced `mtimes` Support**

The `mtimes` function now supports distributed arrays that use a 2-D block-cyclic (2DBC) distribution scheme, and distributed arrays that use 1-D distribution with a distribution dimension greater than 2. Previously, `mtimes` supported only 1-D distribution with a distribution dimension of 1 or 2.

The `mtimes` function now returns a distributed array when only one of its inputs is distributed, similar to its behavior for two distributed inputs.

## Compatibility Considerations

In previous releases, `mtimes` returned a replicated array when one input was distributed and the other input was replicated. Now it returns a distributed array.

## Enhanced `parfor` Support

### Nested for-Loops Inside `parfor`

You can now create nested for-loops inside a `parfor`-loop, and you can use both the `parfor`-loop and for-loop variables directly as indices for the sliced array inside the nested loop. See [Nested Loops](#).

## Enhanced Support for Microsoft Windows HPC Server

### Support for 32-Bit Clients

The parallel computing products now support Microsoft Windows HPC Server on 32-bit Windows clients.

### Search for Cluster Head Nodes Using Active Directory

The `findResource` function can search Active Directory to identify your cluster head node. For more information, see the [findResource](#) reference page.

## Enhanced Admin Center Support

You can now start and stop `mdce` services on remote hosts from Admin Center. For more information, see [Start mdce Service](#).

## New Remote Cluster Access Object

New functionality is available that lets you mirror job data from a remote cluster to a local data location. This supports the generic scheduler interface when making remote submissions to a scheduler or when using a nonshared file system. For more information, see the [RemoteClusterAccess](#) object reference page.

# R2010b

---

**Version: 5.0**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## GPU Computing

This release provides the ability to perform calculations on a graphics processing unit (GPU). Features include the ability to:

- Use a GPU array interface with several MATLAB built-in functions so that they automatically execute with single- or double-precision on the GPU — functions including `mldivide`, `mtimes`, `fft`, etc.
- Create kernels from your MATLAB function files for execution on a GPU
- Create kernels from your CU and PTX files for execution on a GPU
- Transfer data to/from a GPU and represent it in MATLAB with GPUArray objects
- Identify and select which one of multiple GPUs to use for code execution

For more information on all of these capabilities and the requirements to use these features, see GPU Computing.

## Job Manager Security and Secure Communications

You now have a choice of four security levels when using the job manager as your scheduler. These levels range from no security to user authentication requiring passwords to access jobs on the scheduler.

You also have a choice to use secure communications between the job manager and workers.

For more detailed descriptions of these features and information about setting up job manager security, see Set MJS Cluster Security.

The default setup uses no security, to match the behavior of past releases.

## Generic Scheduler Interface Enhancements

### Decode Functions Provided with Product

Generic scheduler interface decode functions for distributed and parallel jobs are now provided with the product. The two decode functions are named:

```
parallel.cluster.generic.distributedDecodeFcn  
parallel.cluster.generic.parallelDecodeFcn
```

These functions are included on the workers' path. If your submit functions make use of the definitions in these decode functions, you do not have to provide your own decode functions. For example, to use the standard decode function for distributed jobs, in your submit function set `MDCE_DECODE_FUNCTION` to `'parallel.cluster.generic.distributedDecodeFcn'`. For information on using the generic scheduler interface with submit and decode functions, see Use the Generic Scheduler Interface.

### Enhanced Example Scripts

This release provides new sets of example scripts for using the generic scheduler interface. As in previous releases, the currently supported scripts are provided in the folder

```
matlabroot/toolbox/distcomp/examples/integration
```

---

In this location there is a folder for each type of scheduler:

- `lsf` — Platform LSF®
- `pbs` — PBS
- `sgc` — Sun® Grid Engine
- `ssh` — generic UNIX-based scripts
- `winmpiexec` — mpiexec on Windows

For the updated scheduler folders (`lsf`, `pbs`, `sgc`), subfolders within each specify scripts for different cluster configurations: `shared`, `nonshared`, `remoteSubmission`.

For more information on the scripts and their updates, see the README file provided in each folder, or see Supplied Submit and Decode Functions.

## Compatibility Considerations

For those schedulers types with updated scripts in this release (`lsf`, `pbs`, `sgc`), the old versions of the scripts are provided in the folder `matlabroot/toolbox/distcomp/examples/integration/old`. These old scripts might be removed in future releases.

## batch Now Able to Run Functions

Batch jobs can now run functions as well as scripts. For more information, see the `batch` reference page.

## batch and matlabpool Accept Scheduler Object

The `batch` function and the functional form of `matlabpool` now accept a scheduler object as their first input argument to specify which scheduler to use for allocation of compute resources. For more information, see the `batch` and `matlabpool` reference pages.

## Enhanced Functions for Distributed Arrays

### qr Supports Distributed Arrays

The `qr` function now supports distributed arrays. For restrictions on this functionality, type

```
help distributed/qr
```

### mldivide Enhancements

The `mldivide` function (`\`) now supports rectangular distributed arrays. Formerly, only square matrices were supported as distributed arrays.

When operating on a square distributed array, if the second input argument (or right-hand side of the operator) is replicated, `mldivide` now returns a distributed array.

## Compatibility Considerations

In previous releases, `mldivide` returned a replicated array when the second (or right-hand side) input was replicated. Now it returns a distributed array.

### **chol Supports 'lower' option**

The `chol` function now supports the 'lower' option when operating on distributed arrays. For information on using `chol` with distributed arrays, type

```
help distributed/chol
```

### **eig and svd Return Distributed Array**

When returning only one output matrix, the `eig` and `svd` functions now return a distributed array when the input is distributed. This behavior is now consistent with outputs when requesting more than one matrix, which returned distributed arrays in previous releases.

## **Compatibility Considerations**

In previous releases, `eig` and `svd` returned a replicated array when you requested a single output. Now they return a distributed array if the output is a single matrix. The behavior when requesting more than one output is not changed.

### **transpose and ctranspose Support 2dbc**

In addition to their original support for 1-D distribution schemes, the functions `ctranspose` and `transpose` now support 2-D block-cyclic ('2dbc') distributed arrays.

### **Inf and NaN Support Multiple Formats**

Distributed and codistributed arrays now support `nan`, `NaN`, `inf` and `Inf` for not-a-number and infinity values with the following functions:

<b>Infinity Value</b>	<b>Not-a-Number</b>
<code>codistributed.Inf</code>	<code>codistributed.NaN</code>
<code>codistributed.inf</code>	<code>codistributed.nan</code>
<code>distributed.Inf</code>	<code>distributed.NaN</code>
<code>distributed.inf</code>	<code>distributed.nan</code>

## **Support for Microsoft Windows HPC Server 2008 R2**

Parallel Computing Toolbox software now supports Microsoft Windows HPC Server 2008 R2. There is no change in interface compared to using HPC Server 2008. Configurations and other toolbox utilities use the same settings to support both HPC Server 2008 and HPC Server 2008 R2.

## **User Permissions for MDCEUSER on Microsoft Windows**

The user identified by the `MDCEUSER` parameter in the `mdce_def` file is now granted all necessary privileges on a Windows system when you install the `mdce` process. For information about what permissions are granted and how to reset them, see [Set the User](#).

---

## **Compatibility Considerations**

In past releases, you were required to set the MDCEUSER permissions manually. Now this is done automatically when installing the mdce process.

