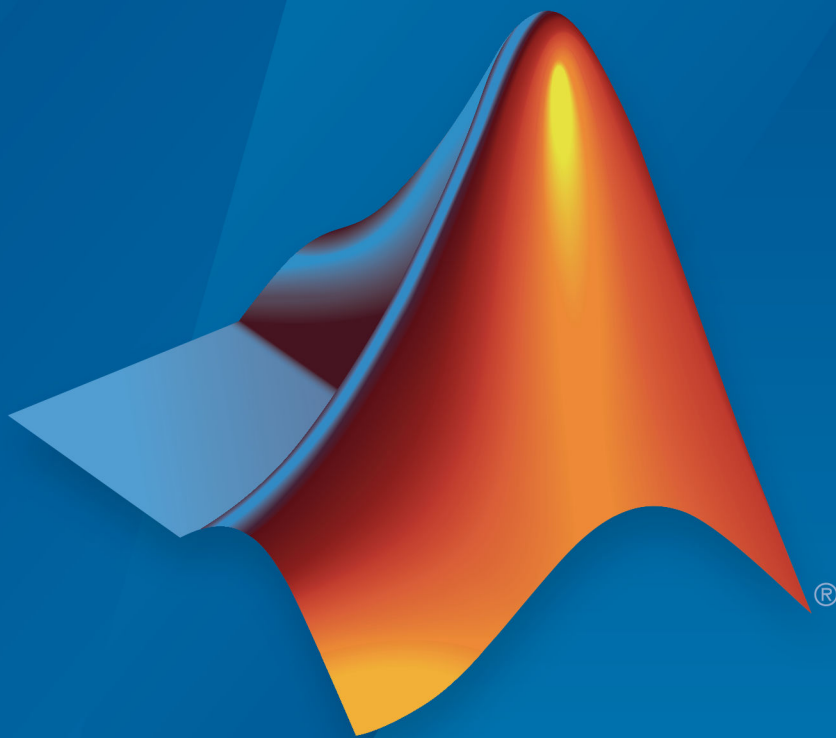


MATLAB® Distributed Computing Server™ Release Notes



MATLAB®

How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



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

MATLAB® Distributed Computing Server™ Release Notes

© COPYRIGHT 2012–2018 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 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 for more information.

Parfeval callbacks: New afterAll and afterEach methods for parallel futures	1-2
Slurm support: Slurm is a fully supported scheduler	1-2
Support for NVIDIA Volta: Update to CUDA 9, support for Volta class GPUs	1-2
Improved file mirroring: Performance of file mirroring for generic scheduler integration	1-2
Parfor performance improvements: More efficient broadcast variables in parfor for non-local clusters	1-2
GPU Array Support: Use enhanced gpuArray functions	1-3
Distributed Array Support: Use enhanced distributed array functions	1-3
Parameter Sweep Example: Use a DataQueue to monitor results during computations on a parallel pool	1-3
Discontinued Support for GPU Devices of Compute Capability less than 3.0	1-3
Backwards Compatibility for MATLAB Job Scheduler: Upgrade MATLAB Job Scheduler (MJS) clusters, and continue to use previous releases of Parallel Computing Toolbox	1-4
Upgrade Parallel Computing Products Together	1-4
Functionality Being Removed or Changed	1-5

Improved Parallel Language Performance: Execute parallel language constructs with reduced overhead	2-2
Tall Array Support: Use tall arrays with Windows client access to Linux Spark clusters	2-2
Improved Parallel Pool Robustness: Run pools without Message Passing Interface (MPI) by default, making pools resilient to workers crashing	2-2
Improved MATLAB Integration with Third-Party Schedulers: Use the Generic Profile Wizard for easier installation and setup of MATLAB Distributed Computing Server	2-2
Cloud Storage: Work with data in Windows Azure Blob Storage	2-3
Copy Client Environment to Workers on Any Cluster: Specify which environment variables on your client machine your workers should automatically inherit	2-3
Client Path Sharing: Add user-added-entries on the client's path to the workers' paths	2-3
GPU Array Support: Use enhanced gpuArray functions	2-3
Distributed Array Support: Use enhanced distributed array functions	2-4
Enhanced Support for Microsoft Windows HPC Pack	2-4
Discontinued Support for GPU Devices of Compute Capability less than 3.0	2-4
Backwards Compatibility for MATLAB Job Scheduler: Upgrade MATLAB Job Scheduler (MJS) clusters, and continue to use previous releases of Parallel Computing Toolbox	2-4
Upgrade Parallel Computing Products Together	2-5

Functionality Being Removed or Changed	2-5
---	------------

R2017a

Backwards Compatibility for MATLAB Job Scheduler: Upgrade MATLAB Job Scheduler (MJS) clusters, and continue to use previous releases of Parallel Computing Toolbox	3-2
Simplified Integration for Third-Party Cluster Schedulers: Updates to generic scheduler integration allow folder-based configuration and eliminate the need to specify function handles	3-2
Ability to Re-create Parallel Jobs: Easily rerun all failed or cancelled tasks for a job	3-2
More Responsive Job Monitor: Automatic updates for new, submitted, or deleted jobs or tasks	3-2
Access to Intermediate Results and Updates in Parallel Computations: Poll for messages or data from different workers during parallel workflows	3-3
Distributed Array Support: Use enhanced distributed array functions	3-3
Support for Spark 2.x enabled Hadoop clusters	3-3
Discontinued Support for GPU Devices of Compute Capability less than 3.0	3-3
Upgrade parallel computing products together	3-4
Properties of generic cluster objects have changed	3-4
Functionality Being Removed or Changed	3-5

Backwards Compatibility: Upgrade MATLAB Job Scheduler (MJS) clusters, and continue to use the previous release of Parallel Computing Toolbox	4-2
Cluster Profile Validation: Choose which validation stages run and the number of MATLAB workers to use	4-2
Parallel Support for Tall Arrays: Process big data with tall arrays in parallel on your desktop, MATLAB Distributed Computing Server, and Spark clusters	4-2
Parallel Menu Enhancement: Use the new menu items in the Parallel Menu to configure and manage cloud based resources	4-3
New Data Types in Distributed Arrays: Use enhanced functions for creating distributed arrays of: datetime; duration; calendarDuration; string; categorical; and table	4-3
Loading Distributed Arrays: Load distributed arrays in parallel using datastore	4-4
datetime Support for Timestamps: Use built-in datetime objects in MATLAB to access timestamp information for jobs and tasks	4-4
Data Transfer Measurement: Use ticBytes and tocBytes to measure the data transfer between MATLAB workers in a parallel pool	4-5
Multithreaded Workers: Use multiple computational threads on your MATLAB workers	4-5
MathWorks Hosted License Manager: Use new option in MATLAB Distributed Computing Server script to ensure workers use MathWorks Hosted License Manager	4-6

JSON support for nodestatus: View the output of the nodestatus script in JavaScript Object Notation (JSON) format	4-6
Upgrade Parallel Computing Products Together	4-6
Functionality Being Removed or Changed	4-7

R2016a

Support for Distributed Arrays: Use enhanced distributed array functions including sparse input to direct (mldivide) and iterative solvers (cgs and pcg)	5-2
Hadoop Kerberos Support: Improved support for Hadoop in a Kerberos authenticated environment	5-2
Transfer unlimited data between client and workers, and attached files up to 4GB in total, in any job using a MATLAB Job Scheduler cluster	5-2
Third Party Scheduler Integration: Obtain integration scripts for Third Party Schedulers (IBM Platform LSF, Grid Engine, PBS and SLURM) from MATLAB Central File Exchange instead of Parallel Computing Toolbox	5-2
Upgrade parallel computing products together	5-3

R2015b

Discontinued support for parallel computing products on 32-bit Windows operating systems	6-2
Scheduler integration scripts for SLURM	6-2

Improved performance of mapreduce on Hadoop 2 clusters	6-2
parallel.pool.Constant function to create constant data on parallel pool workers, accessible within parallel language constructs such as parfor and parfeval	6-3
Upgrade parallel computing products together	6-3

R2015a

Support for mapreduce function on any cluster that supports parallel pools	7-2
Using DNS for cluster discovery	7-2
MS-MPI support for MJS clusters	7-2
Ports and sockets in mdce_def file	7-2
Discontinued support for GPU devices on 32-bit Windows computers	7-3
Discontinued support for parallel computing products on 32-bit Windows computers	7-3

R2014b

Data Analysis on Hadoop clusters using mapreduce	8-2
Additional MATLAB functions for distributed arrays, including fft2, fftn, ifft2, ifftn, cummax, cummin, and diff	8-2

R2014a

Duplication of an existing job, containing some or all of its tasks	9-2
More MATLAB functions enhanced for distributed arrays . . .	9-2
Old Programming Interface Removed	9-3
matlabpool Function Being Removed	9-3

R2013b

parpool: New command-line interface (replaces matlabpool), desktop indicator, and preferences for easier interaction with a parallel pool of MATLAB workers	10-2
Automatic start of a parallel pool when executing code that uses parfor or spmd	10-2
Option to start a parallel pool without using MPI	10-3
More MATLAB functions enabled for distributed arrays: permute, ipermute, and sortrows	10-3
Upgraded MPICH2 Version	10-4
Discontinued Support for parallel.cluster.Mpiexec	10-4

R2013a

Automatic detection and transfer of files required for execution in both batch and interactive workflows	11-2
---	-------------

Automatic detection and selection of specific GPUs on a cluster node when multiple GPUs are available on the node	12-2
Detection of MATLAB Distributed Computing Server clusters that are available for connection from user desktops through Profile Manager	12-2

R2018a

Version: 6.12

New Features

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`” (Parallel Computing Toolbox).

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” (Parallel Computing Toolbox).

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” (Parallel Computing Toolbox).

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” (Parallel Computing Toolbox).

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” (Parallel Computing Toolbox).

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 (MJS) clusters, and continue to use previous releases of Parallel Computing Toolbox

You can upgrade your MATLAB Job Scheduler (MJS) 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 (MJS) 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 (MJS) 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” (Parallel Computing Toolbox).
<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.

For more detail on Hadoop clusters, see [Configure a Hadoop Cluster](#).

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](#).

Cloud Storage: Work with data in Windows Azure Blob Storage

You can now use Windows Azure® Blob Storage to access data in the cloud using `datastore`. For more information on this topic, see [Read Remote Data](#).

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` using `parpool`, `batch`, `createJob`, or in the Cluster Profile Manager. `EnvironmentVariables` is also a common property of the `parallel.Job` 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](#).

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 (MJS) clusters, and continue to use previous releases of Parallel Computing Toolbox

You can upgrade your MATLAB Job Scheduler (MJS) 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](#).

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 (MJS) 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](#).

R2017a

Version: 6.10

New Features

Bug Fixes

Compatibility Considerations

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

You can upgrade your MATLAB Job Scheduler (MJS) 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”.

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”.

Ability to Re-create Parallel Jobs: Easily rerun all failed or cancelled tasks for a job

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.

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.

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 send and poll for messages or data from different workers using a `DataQueue`.

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](#).

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.

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.

Upgrade parallel computing products together

This version of MATLAB Distributed Computing Server software is accompanied by a corresponding new version of Parallel Computing Toolbox 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" (Parallel Computing Toolbox)
- "Integration Scripts for Generic Schedulers" (Parallel Computing Toolbox)

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](#).

R2016b

Version: 6.9

New Features

Bug Fixes

Compatibility Considerations

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

You can upgrade your MATLAB Job Scheduler (MJS) 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](#).

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](#).

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](#)

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:

calendarDuration	calquarters	cellstr	time
caldays	calweeks	datevec	
calmonths	calyears	iscalendarduration	

Distributed categorical Arrays:

categorical	iscategorical	isundefined	setcats
addcats	iscategory	removecats	
categories	isordinal	renamecats	
countcats	isprotected	reordercats	

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](#).

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`.

MathWorks Hosted License Manager: Use new option in MATLAB Distributed Computing Server script to ensure workers use MathWorks Hosted License Manager

You can now use a new `-usemhl` option in your `mdce` script to ensure workers use MathWorks Hosted License Manager. For more information, see `mdce`.

JSON support for nodestatus: View the output of the nodestatus script in JavaScript Object Notation (JSON) format

You can now use a new `-json` option in your `nodestatus` script to view your output in JavaScript Object Notation (JSON) format. For more information, see `nodestatus`.

Upgrade Parallel Computing Products Together

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

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 (Parallel Computing Toolbox).

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`

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.

Third Party Scheduler Integration: Obtain integration scripts for Third Party Schedulers (IBM Platform LSF, Grid Engine,

PBS and SLURM) from MATLAB Central File Exchange instead of Parallel Computing Toolbox

Obtain integration scripts for Third Party Schedulers (IBM® Platform LSF, Grid Engine family, PBS family and SLURM) from MATLAB Central File Exchange instead of Parallel Computing Toolbox.

- IBM Platform LSF
- Grid Engine family
- PBS family
- SLURM

Compatibility Considerations

In previous releases, integration scripts for Third Party Schedulers were available from Parallel Computing Toolbox. In R2016a, these integration scripts are obtained from MATLAB Central File Exchange and are no longer available from Parallel Computing Toolbox.

Upgrade parallel computing products together

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

Scheduler integration scripts for SLURM

This release offers a new set of scripts containing submit and decode functions to support Simple Linux® 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.

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.

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`.

Upgrade parallel computing products together

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

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 (MJS) 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 MJS 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 (MJS) 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 (MJS) 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_PARALLELPPOOL_MIN_PORT`, `WORKER_PARALLELPPOOL_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 MJS and workers.

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.

R2014b

Version: 6.5

New Features

Bug Fixes

Data Analysis on Hadoop clusters using mapreduce

MATLAB Distributed Computing Server supports 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

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	

For a list of MATLAB functions that support distributed arrays, see [MATLAB Functions on Distributed and Codistributed Arrays](#).

R2014a

Version: 6.4

New Features

Bug Fixes

Compatibility Considerations

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  
randi  
rand  
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](#).

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 Parallel Computing Toolbox release topic [New Programming Interface](#).

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.

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

This release introduces a number of enhancements for interacting with parallel pool resources. For more detailed descriptions of these enhancements, see [parpool: New command-line interface \(replaces matlabpool\), desktop indicator, and preferences for easier interaction with a parallel pool of MATLAB workers](#) in the Parallel Computing Toolbox release notes.

- 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.
- 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.
- 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 the MATLAB toolstrip, from the parallel pool status icon, or by typing preferences at the command line.

For more information, see [Parallel Preferences](#).

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.

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 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 MJS 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 'SpmEnabled' property `false` in the call to the `parpool` function. For example:

```
p = parpool('SpmEnabled', 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 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>

For more information about these functions and distributed arrays, see More MATLAB functions enabled for distributed arrays: `permute`, `ipermute`, and `sortrows` in the Parallel Computing Toolbox release notes.

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 `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 might be completely removed.

R2013a

Version: 6.2

New Features

Bug Fixes

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.

For more information, see [Automatic detection and transfer of files required for execution in both batch and interactive workflows](#) in the Parallel Computing Toolbox release notes.

R2012b

Version: 6.1

New Features

Bug Fixes

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.

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.