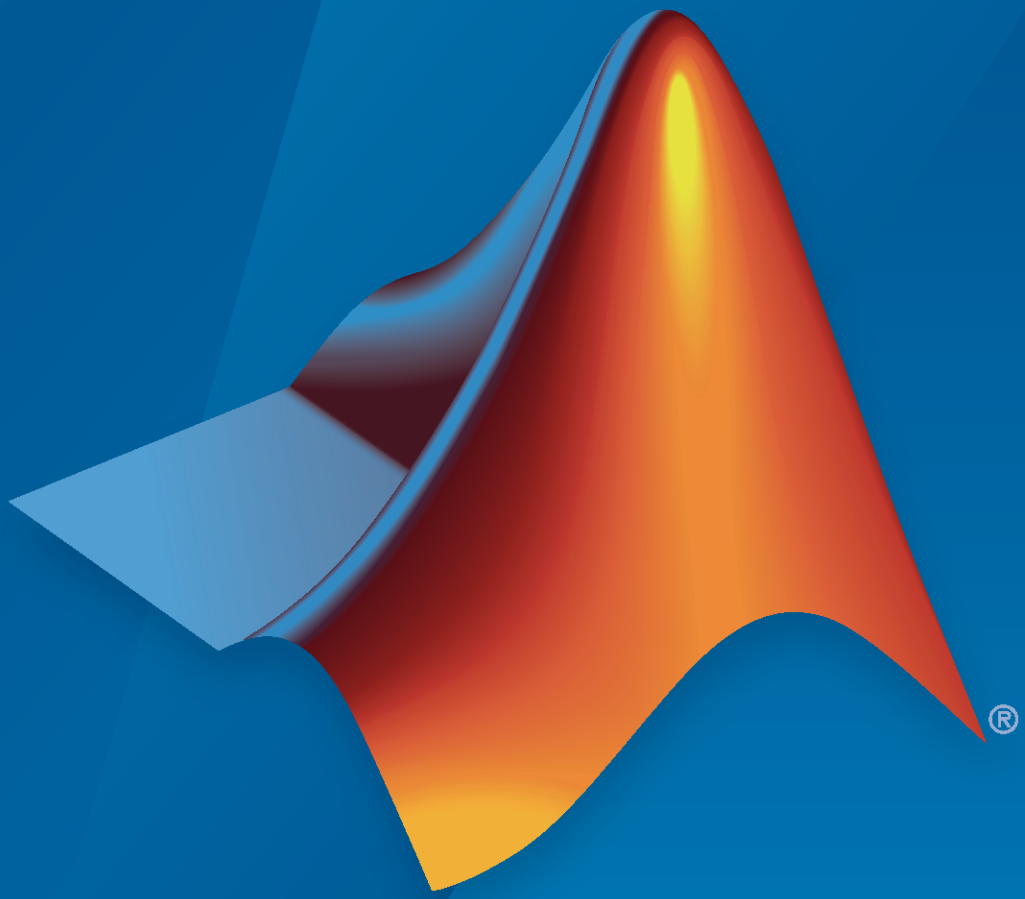
MATLAB® Parallel Server™ Release Notes



MATLAB®

MathWorks®

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.
1 Apple Hill Drive
Natick, MA 01760-2098

MATLAB® Parallel Server™ Release Notes

Contents

R2021b

R2021a

R2020b

R2020a

R2019b

R2019a

R2018b

R2018a

R2017b

R2017a

R2016b

R2016a

R2015b

R2015a

R2014b

R2014a

R2013b

R2013a

R2012b

R2021b

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” (MATLAB) and “Write Portable Parallel Code” (Parallel Computing Toolbox).

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” (Parallel Computing Toolbox).

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

- `betafit`
- `fitcecoc`
- `fitcensemble`
- `fitctree`
- `fitdist`
- `fitrtree`
- `gevfit`

- `gpfit`
- `ksdensity`
- `mle`
- `mvksdensity`
- `nbinfit`

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

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.

  - `downsample`
  - `filtfilt`
  - `pspectrum`
  - `resample`
  - `shiftdata`
  - `unshiftdata`
  - `upfirdn`
  - `upsample`
  - `xspectrogram`
- The following functions have new `gpuArray` support in Audio Toolbox™:

  - `classifySound`
  - `crepePostprocess`
  - `crepePreprocess`
  - `detectSpeech`
  - `harmonicRatio`
  - `openl3Features`
  - `openl3Preprocess`
  - `pitchnn`
  - `vggishFeatures`
  - `vggishPreprocess`
  - `yamnetPreprocess`

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

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” (Parallel Computing Toolbox).

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

- `adaptthresh`
- `otsuthresh`
- `imquantize`
- `grayslice`
- `im2int16`
- `im2single`
- `im2uint16`
- `im2uint8`

For more information, see “Run MATLAB Functions in Thread-Based Environment” (MATLAB).

Reference Architectures: Deploy and Run MATLAB Parallel Server from Azure Marketplace

Quickly and easily deploy MATLAB Parallel Server™ in the cloud using bring-your-own-license (BYOL) software plans developed by MathWorks® for Azure® Marketplace.

The MATLAB Parallel Server BYOL reference architecture for Azure Marketplace now supports R2021a. The MATLAB Parallel Server BYOL reference architecture was released in February 2021 with support for R2020b.

For more information, see Run MATLAB Parallel Server from Azure Marketplace.

Reference Architectures: Deploy and Run Network License Manager from Azure Marketplace

Quickly and easily deploy a network license manager in the cloud using software plans developed by MathWorks for Azure Marketplace.

The network license manager reference architecture for Azure Marketplace now supports R2021a. The network license manager for Azure Marketplace was released in February 2021 with support for R2020b.

For more information, see Run Network License Manager from Azure Marketplace.

Functionality Being Removed or Changed

Support for macOS will be removed
Still runs

Support for macOS will be removed in R2022a. To run the MATLAB Parallel Server cluster, use a Windows® or Linux® operating system instead.

You can still connect to a MATLAB Parallel Server cluster when you use Parallel Computing Toolbox running on a macOS client.

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 nonsquare 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.

remotecopy will be removed
Warns

`remotecopy` will be removed in a future release.

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

  As a best practice use `scp` instead.

Not Recommended	Recommended
`remotecopy -remotehost host1 -local /my/file/path -remote /remote/file/path -protocol scp`	`scp /my/file/path host1:/remote/file/path`
`remotecopy -remotehost host1,host2 -local /my/file/path -remote /remote/file/path -protocol scp`	`scp /my/file/path host1:/remote/file/path` `scp /my/file/path host2:/remote/file/path`

Not Recommended	Recommended
`remotecopy -remotehost host1 -local /my/file/path -remote /remote/file/path -protocol sftp`	`sftp /my/file/path host1:/remote/file/path`

- Previously you used `remotecopy` and `-protocol sftp` to copy files to and from a remote host using the `sftp` protocol.

  As a best practice use `sftp` instead.

Not Recommended	Recommended
`remotecopy -remotehost host1 -local /my/file/path -remote /remote/file/path -protocol sftp`	`sftp /my/file/path host1:/remote/file/path`
`remotecopy -remotehost host1,host2 -local /my/file/path -remote /remote/file/path -protocol sftp`	`sftp /my/file/path host1:/remote/file/path` `sftp /my/file/path host2:/remote/file/path`
`remotecopy -remotehost host1 -local /my/file/path -remote /remote/file/path -protocol sftp`	`sftp /my/file/path host1:/remote/file/path`

remotemjs will be removed
Warns

`remotemjs` will be removed in a future release.

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

As a best practice use `ssh` instead.

Not Recommended	Recommended
`remotemjs <mjs options> -matlabroot <installfoldername> -remotehost host1`	`ssh host1 <installfoldername>/toolbox/parallel/bin/mj`
`remotemjs <mjs options> -matlabroot <installfoldername> -remotehost host1,host2`	`ssh host1 <installfoldername>/toolbox/parallel/bin/mj` `ssh host2 <installfoldername>/toolbox/parallel/bin/mj`

R2021a

Version: 7.4

New Features

Compatibility Considerations

GPU Functionality: Use new and enhanced gpuArray functions

- `pagectranspose` (MATLAB)
- `pagetranspose` (MATLAB)
- `spline` (MATLAB)
- `imresize` (MATLAB) — 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` (MATLAB) — Support for `dim` and `direction` input arguments and `ComparisonMethod` name-value option.

For more information, see Run MATLAB Functions on a GPU (Parallel Computing Toolbox).

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

- `fitcensemble` (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)
- `multissim` (Image Processing Toolbox)
- `multissim3` (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 (Parallel Computing Toolbox).

Distributed Arrays: Use new and enhanced distributed array functionality

- `colon` (MATLAB)
- `ichol` (MATLAB)
- `interp1` (MATLAB)
- `pagectranspose` (MATLAB)
- `pagemtimes` (MATLAB)
- `pagetranspose` (MATLAB)

For more information, see Run MATLAB Functions with Distributed Arrays (Parallel Computing Toolbox).

GPU Devices: Count and compare available GPU devices

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

- `gpuDeviceCount` (Parallel Computing Toolbox) — Count all detected devices, all supported devices, or only devices available in your current session.
- `gpuDeviceTable` (Parallel Computing Toolbox) — 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` (MATLAB)
- `imread` (MATLAB)
- `parallel.pool.DataQueue` (Parallel Computing Toolbox)
- `parallel.pool.PollableDataQueue` (Parallel Computing Toolbox)
- 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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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.

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 Kepler GPUs (CC 3.0 and 3.2) 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 (Parallel Computing Toolbox).

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

For more information, see Run MATLAB Functions on a GPU (Parallel Computing Toolbox).

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` (Parallel Computing Toolbox)
- `labReceive` (Parallel Computing Toolbox)
- `labBroadcast` (Parallel Computing Toolbox)

- `gop` (Parallel Computing Toolbox)

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 (Parallel Computing Toolbox).

Distributed Arrays: Use new and enhanced distributed array functionality

- `filter` (MATLAB)
- `mpower` (MATLAB)
- `pagefun` (Parallel Computing Toolbox)
- `rmmissing` (MATLAB)

For more information, see Run MATLAB Functions with Distributed Arrays (Parallel Computing Toolbox).

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 be 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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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.

Parallel Workflows: Compare performance of different parallel environments

Learn how to choose when to use parallel pools, and when to use different parallel programming constructs. Use new examples to improve the performance of your parallel workflows:

- Compare Performance of Multithreading and `ProcessPool` (Parallel Computing Toolbox)
- Compare Performance of `parfor`, `parfeval`, and `spmd` (Parallel Computing Toolbox)

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` (MATLAB) 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` (MATLAB)
- `canUseParallelPool` (MATLAB)

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.

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® 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(gcp('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` (Parallel Computing Toolbox) function or by setting the environment variable `MW_CUDA_FORWARD_COMPATIBILITY` to `1`.

For more information, see Forward Compatibility for GPU Devices (Parallel Computing Toolbox).

classUnderlying and isaUnderlying are not recommended

Still runs

`classUnderlying` (Parallel Computing Toolbox) and `isaUnderlying` (Parallel Computing Toolbox) 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.

AdditionalProperties Documentation: Learn how to customize the behavior of the sample plugin scripts

When you use the generic scheduler interface, you can modify the behavior of the plugin scripts by setting additional properties. For example, you can specify additional scheduler arguments for job submission. Follow these new instructions to learn more about the different additional properties that are available for the sample plugin scripts and how to set them: Customize Behavior of Sample Plugin Scripts.

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 and Configure Using the Generic Scheduler Interface.

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 (Parallel Computing Toolbox).

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`
- `melSpectrogram`
- `czt`
- `spectrogram`
- `stft`
- `wvd`, `wvd`
- `cwt`
- `cwtfilterbank` and `wt` object function

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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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.

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

`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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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. gpuArrays loaded without a GPU are limited and you cannot use them for computations. To use a gpuArray loaded without a GPU, retrieve the contents with `gather`.

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 (Parallel Computing Toolbox).

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 Computing Toolbox).

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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox)
- Run Batch Job and Access Files from Workers (Parallel Computing Toolbox)

parfeval Examples: Explore parfeval workflows with new documentation

Use these new examples to learn more about `parfeval` workflows:

- Plot During Parameter Sweep with `parfeval` (Parallel Computing Toolbox)
- Perform Webcam Image Acquisition in Parallel with Postprocessing (Parallel Computing Toolbox)
- Perform Image Acquisition and Parallel Image Processing (Parallel Computing Toolbox)

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 (Parallel Computing Toolbox)
- Perform Image Acquisition and Parallel Image Processing (Parallel Computing Toolbox)

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 (Parallel Computing Toolbox). 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.

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.

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

Updated license model: Scale beyond 200 workers without checking out more licenses

If you have `200` workers checked out and are not using on-demand licensing, you can continue to scale further without checking out more workers. To learn more, see License Model.

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

MATLAB Distributed Computing Server now has the name MATLAB Parallel Server.

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`.

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.

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 (Parallel Computing Toolbox)
- Benchmark Your Cluster with the HPC Challenge (Parallel Computing Toolbox)

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.

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 (Parallel Computing Toolbox)

- Run Multiple Deep Learning Experiments (Parallel Computing Toolbox)

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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

Reference Architectures: Support for network license manager

Use reference architectures to serve a network license manager for your MATLAB Parallel Server cloud clusters. Use preconfigured server settings for Amazon® AWS and Microsoft® Azure.

- MATLAB Parallel Server on Amazon Web Services.
- License Manager for MATLAB on Amazon Web Services.
- MATLAB Parallel Server on Microsoft Azure.
- License Manager for MATLAB on Microsoft Azure.

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.

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.

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")
```

The remotecopy command will no longer support the rcp protocol in a future release
Warns

The `rcp` protocol is insecure. The `remotecopy` function will no longer support `rcp` as an option to the `-protocol` flag in a future release, and warns starting in `R2018b`. Instead, specify any other of the supported protocols, such as `scp`. For a complete list, see remotecopy.

The remotemjs command will no longer support the rsh protocol in a future release
Warns

The `rsh` protocol is insecure. The `remotemjs` function will no longer support `rsh` as an option to the `-protocol` flag in a future release, and warns starting in `R2018b`. Instead, specify any other of the supported protocols, such as `ssh`. For a complete list, see remotemjs.

The mdce command will be removed in a future release
Warns

`mdce` will be removed in a future release. Use `mjs` instead.

The defaults file is now `mjs_def`. To specify an alternative file, you can use the `-mjsdef` flag.

For more information, see mjs.

The remotemdce command will be removed in a future release
Warns

`remotemdce` will be removed in a future release. Use `remotemjs` instead. For more information, see remotemjs.

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

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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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

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 file systems
- 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 (Parallel Computing Toolbox).

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 Computing Toolbox).

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 (Parallel Computing Toolbox)
- Run MATLAB Functions on a GPU (Parallel Computing Toolbox)
- Run MATLAB Functions with Distributed Arrays (Parallel Computing Toolbox)

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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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.

Reference Architectures: Create customized MATLAB Distributed Computing Server clusters in the cloud using Amazon Web Services (AWS) or Microsoft Azure

Use reference architectures to build your own MATLAB Distributed Computing Server cluster solution for the cloud. Use preconfigured cluster settings for Amazon AWS and Microsoft Azure. For more information on launching and configuring clusters, see

- MATLAB Parallel Server on Amazon Web Services.
- MATLAB Parallel Server on Microsoft Azure.

For more information, see MATLAB in the Cloud (https://www.mathworks.com/cloud.html).

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 (Parallel Computing Toolbox).

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.

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.

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 (Parallel Computing Toolbox)
- Use parfor to Train Multiple Deep Learning Networks (Parallel Computing Toolbox)
- Use `parfeval` to Train Multiple Deep Learning Networks (Parallel Computing Toolbox)
- Upload Deep Learning Data to the Cloud (Parallel Computing Toolbox)
- Send Deep Learning Batch Job To Cluster (Parallel Computing Toolbox)

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.

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`.

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`.

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 `gpuDevice` to query the compute capabilities of your GPUs. For more information, see GPU Computing in MATLAB.
`parallel.gpu.rng` is renamed to `gpurng`	Still runs	`gpurng`	Replace all instances of `parallel.gpu.rng` with `gpurng`

R2017b

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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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 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.

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.

R2017a

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.

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 (Parallel Computing Toolbox).

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 (Parallel Computing Toolbox).

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
- 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.

R2016b

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.

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 `table`s:

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 `-usemhlm` 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

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

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

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 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 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.

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

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
besseli
besselj
besselk
bessely
beta
betainc
betaincinv
betaln
cart2pol
cart2sph
compan
corrcoef
cov
cummax
cummin
diff
erf
erfc
erfcinv
erfcx
erfinv
fft2
fftn
gamma
gammainc
gammaincinv
gammaln
hankel
hsv2rgb
ifft
ifft2
ifftn
isfloat
isinteger
islogical
isnumeric
median
mode
pol2cart
psi
rgb2hsv
sph2cart
std
toeplitz
trapz
unwrap
vander
var
```

For a list of MATLAB functions that support distributed arrays, see MATLAB Functions on Distributed and Codistributed Arrays.

R2014a

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

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 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 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:

`ipermute`	`zeros`
`permute`	`ones`
`sortrows`	`nan`
	`inf`
`cast`	`true`
	`false`

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.