Parallel Computing Toolbox™ Release Notes

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

For contact information about worldwide offices, see the MathWorks Web site.

Parallel Computing Toolbox™ Release Notes

Contents

Summary by Version 1

Version 5.2 (R2011b) Parallel Computing Toolbox Software 4

Version 5.1 (R2011a) Parallel Computing Toolbox Software 12

Version 5.0 (R2010b) Parallel Computing Toolbox Software 17

Version 4.3 (R2010a) Parallel Computing Toolbox Software 23

Version 4.2 (R2009b) Parallel Computing Toolbox Software 26

Version 4.1 (R2009a) Parallel Computing Toolbox Software 32

Version 4.0 (R2008b) Parallel Computing Toolbox Software 35

Version 3.3 (R2008a) Parallel Computing Toolbox Software 39

Version 3.2 (R2007b) Distributed Computing Toolbox Software 43

Version 3.1 (R2007a) Distributed Computing Toolbox Software 46

Version 3.0 (R2006b) Distributed Computing Toolbox Software 51

Compatibility Summary for Parallel Computing Toolbox Software **56**

Summary by Version

This table provides quick access to what is new in each version. For clarification, see "Using Release Notes" on page 1.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Latest Version V5.2 (R2011b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V5.1 (R2011a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V5.0 (R2010b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V4.3 (R2010a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V4.2 (R2009b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V4.1 (R2009a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V4.0 (R2008b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V3.3 (R2008a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V3.2 (R2007b)	Yes Details	Yes Summary	Bug Reports Includes fixes
V3.1 (R2007a)	Yes Details	Yes Summary	Bug Reports Includes fixes
V3.0 (R2006b)	Yes Details	Yes Summary	Bug Reports Includes fixes

Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features
- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks® products required for this product (for example, MATLAB® or Simulink®). Determine if enhancements, bugs, or compatibility considerations in other products impact you.

If you are upgrading from a software version other than the most recent one, review the current release notes and all interim versions. For example, when you upgrade from V1.0 to V1.2, review the release notes for V1.1 and V1.2.

What Is in the Release Notes

New Features and Changes

- New functionality
- Changes to existing functionality

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product release appear under Bug Reports at the MathWorks Web site. Bug fixes can sometimes result in incompatibilities, so review the fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release time and as more information becomes available. Bug Reports include provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.

Documentation on the MathWorks Web Site

Related documentation is available on `mathworks.com` for the latest release and for previous releases:

- Latest product documentation
- Archived documentation

Version 5.2 (R2011b) Parallel Computing Toolbox Software

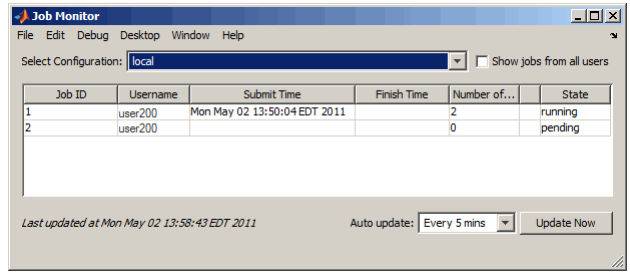
This table summarizes what is new in Version 5.2 (R2011b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

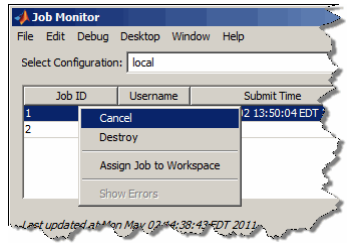
- "New Job Monitor" on page 5
- "Run Scripts as Batch Jobs from the Current Folder Browser" on page 6
- "Number of Local Workers Increased to Twelve" on page 6
- "Enhanced GPU Support" on page 7
- "Enhanced Distributed Array Support" on page 9
- "Conversion of Error and Warning Message Identifiers" on page 9
- "Task Error Properties Updated" on page 11
- "Upgrade Parallel Computing Products Together" on page 11

New Job Monitor

The Job Monitor is a tool that lets you track the jobs you have submitted to a cluster. It displays the jobs for the scheduler determined by your selection of a parallel configuration. Open the Job Monitor from the MATLAB desktop by selecting **Parallel > Job Monitor**.



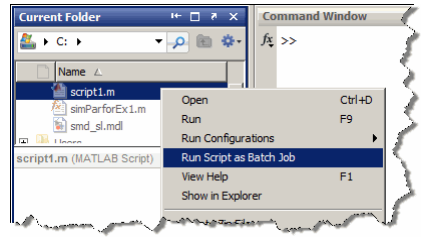
Right-click a job in the list to select a command from the context menu for that job:



For more information about the Job Monitor and its capabilities, see “Job Monitor”.

Run Scripts as Batch Jobs from the Current Folder Browser

From the Current Folder browser, you can run a MATLAB script as a batch job by browsing to the file's folder, right-clicking the file, and selecting **Run Script as Batch Job**. The batch job runs on the cluster identified by the current default parallel configuration. The following figure shows the menu option to run the script from the file `script1.m`:



Number of Local Workers Increased to Twelve

You can now run up to 12 local workers on your MATLAB client machine. If you do not specify the number of local workers in a command or configuration, the default number of local workers is determined by the value of the local scheduler's `ClusterSize` property, which by default equals the number of computational cores on the client machine.

Enhanced GPU Support

Latest NVIDIA CUDA Device Driver

This version of Parallel Computing Toolbox™ GPU functionality supports only the latest NVIDIA CUDA device driver.

Compatibility Considerations. Earlier versions of the toolbox supported earlier versions of CUDA device driver. Always make sure you have the latest CUDA device driver.

Deployment of GPU Applications

MATLAB® Compiler™ generated standalone executables and components now support applications that use the GPU.

Random Number Generation

You can now directly create arrays of random numbers on the GPU using these new static methods for GPUArray objects:

```
parallel.gpu.GPUArray.rand
parallel.gpu.GPUArray.randi
parallel.gpu.GPUArray.randn
parallel.gpu.GPUArray.RandStream
parallel.gpu.GPUArray.rng
```

Also, `arrayfun` called with GPUArray data now supports `rand`, `randi`, and `randn`. For more information about using `arrayfun` to generate random matrices on the GPU, see “Generating Random Numbers on the GPU”.

GPUArray Support

The following functions now support GPUArray data:

chol	isnan	norm
diff	lu	not
eig	max	repmat
find	min	sort
isfinite	mldivide	svd
isinf		

`mldivide` supports complex arrays. It also supports overdetermined matrices (with more rows than columns) when the second input argument is a column vector (has only one column).

`eig` supports only symmetric matrices.

`max` and `min` return only one output argument; they do not return an index vector.

The following functions are not methods of the GPUArray class, but they do work with GPUArray data:

angle	flipdim	mean
beta	fftshift	perms
betaln	ifftshift	squeeze
fliplr	kron	rot90
flipud		

The default display of GPUArray variables now shows the array contents. In previous releases, the display showed some of the object properties, but not the contents. For example, the new enhanced display looks like this:

```
M = gpuArray(magic(3))
M =
     8     1     6
     3     5     7
     4     9     2
```

To see that `M` is a GPUArray, use the `whos` or `class` function.

MATLAB Code on the GPU

GPU support is extended to include the following MATLAB code in functions called by `arrayfun` to run on the GPU:

```
rand
randi
randn
xor
```

Also, the handle passed to `arrayfun` can reference a simple function, a subfunction, a nested function, or an anonymous function. The function passed to `arrayfun` can call any number of its subfunctions. The only restriction is that when running on the GPU, nested and anonymous functions do not have access to variables in the parent function workspace. For more information on function structure and relationships, see “Types of Functions”.

Enhanced Distributed Array Support

Newly Supported Functions

The following functions are enhanced to support distributed arrays, supporting all forms of codistributor (1-D and 2DBC):

```
inv
meshgrid
ndgrid
sort
```

The following functions can now directly construct codistributed arrays:

```
codistributed.linspace(m, n, ..., codist)
codistributed.logspace(m, n, ..., codist)
```

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in Parallel Computing Toolbox.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a `try/catch` statement and performs an action based on a specific error identifier.

For example, the `'distcomp:old:ID'` identifier has changed to `'parallel:similar:ID'`. If your code checks for `'distcomp:old:ID'`, you must update it to check for `'parallel:similar:ID'` instead.

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

To determine the identifier for an error, run the following commands just after you see the error:

```
exception = MException.last;
MSGID = exception.identifier;
```

Tip Warning messages indicate a potential issue with your code. While you can turn off a warning, a suggested alternative is to change your code so that it runs warning-free.

Task Error Properties Updated

If there is an error during task execution, the task `Error` property now contains the non-empty `MException` object that is thrown. If there was no error during task execution, the `Error` property is empty.

The identifier and message properties of this object are now the same as the task's `ErrorIdentifier` and `ErrorMessage` properties, respectively. For more information about these properties, see the `Error`, `ErrorIdentifier`, and `ErrorMessage` reference pages.

Compatibility Considerations

In past releases, when there was no error, the `Error` property contained an MException object with empty data fields, generated by `MException('', '')`. Now to determine if a task is error-free, you can query the `Error` property itself to see if it is empty:

```
didTaskError = ~isempty(t.Error)
```

where `t` is the task object you are examining.

Upgrade Parallel Computing Products Together

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

Compatibility Considerations

As with every new release, you must upgrade both the Parallel Computing Toolbox and MATLAB Distributed Computing Server products 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.

Version 5.1 (R2011a) Parallel Computing Toolbox Software

This table summarizes what is new in Version 5.1 (R2011a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

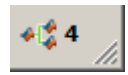
- "Deployment of Local Workers" on page 12
- "New Desktop Indicator for MATLAB Pool Status" on page 12
- "Enhanced GPU Support" on page 13
- "Distributed Array Support" on page 15
- "Enhanced parfor Support" on page 15
- "Enhanced Support for Microsoft Windows HPC Server" on page 15
- "Enhanced Admin Center Support" on page 16
- "New Remote Cluster Access Object" on page 16

Deployment of Local Workers

MATLAB Compiler generated standalone executables and libraries from parallel applications can now launch up to eight local workers without requiring MATLAB Distributed Computing Server software.

New Desktop Indicator for MATLAB Pool Status

When you first open a MATLAB pool from your desktop session, an indicator appears in the lower-right corner of the desktop to show that this desktop session is connected to an open pool. The number indicates how many workers are in the pool.



When you close the pool, the indicator remains displayed and shows a value of 0.

Enhanced GPU Support

Static Methods to Create GPUArray

The following new static methods directly create GPUArray objects:

```
parallel.gpu.GPUArray.linspace
parallel.gpu.GPUArray.logspace
```

GPUArray Support

The following functions are enhanced to support GPUArray data:

```
cat
colon
conv
conv2
cumsum
cumprod
eps
filter
filter2
horzcat
meshgrid
ndgrid
plot
subsasgn
subsindex
subsref
vertcat
```

and all the plotting related functions.

GPUArray Indexing

Because GPUArray now supports `subsasgn` and `subsref`, you can index into a GPUArray for assigning and reading individual elements.

For example, create a GPUArray and assign the value of an element:

```
n = 1000;
D = parallel.gpu.GPUArray.eye(n);
D(1,n) = pi
```

Create a GPUArray and read the value of an element back into the MATLAB workspace:

```
m = 500;
D = parallel.gpu.GPUArray.eye(m);
one = gather(D(m,m))
```

MATLAB Code on the GPU

GPU support is extended to include the following MATLAB code in functions called by `arrayfun` to run on the GPU:

```
&, |, ~, &&, ||,
while, if, else, elseif, for, return, break, continue, eps
```

You can now call `eps` with string inputs, so your MATLAB code running on the GPU can include `eps('single')` and `eps('double')`.

NVIDIA CUDA Driver 3.2 Support

This version of Parallel Computing Toolbox GPU functionality supports only NVIDIA CUDA device driver 3.2.

Compatibility Considerations. Earlier versions of the toolbox supported earlier versions of CUDA device driver. If you have an older driver, you must upgrade to CUDA device driver version 3.2.

Distributed Array Support

Newly Supported Functions

The following functions are enhanced to support distributed arrays, supporting all forms of codistributor (1-D and 2DBC):

```
arrayfun
cat
reshape
```

Enhanced mtimes Support

The `mtimes` function now supports distributed arrays that use a 2-D block-cyclic (2DBC) distribution scheme, and distributed arrays that use 1-D distribution with a distribution dimension greater than 2. Previously, `mtimes` supported only 1-D distribution with a distribution dimension of 1 or 2.

The `mtimes` function now returns a distributed array when only one of its inputs is distributed, similar to its behavior for two distributed inputs.

Compatibility Considerations. In previous releases, `mtimes` returned a replicated array when one input was distributed and the other input was replicated. Now it returns a distributed array.

Enhanced parfor Support

Nested for-Loops Inside parfor

You can now create nested `for`-loops inside a `parfor`-loop, and you can use both the `parfor`-loop and `for`-loop variables directly as indices for the sliced array inside the nested loop. See “Nested Loops”.

Enhanced Support for Microsoft Windows HPC Server

Support for 32-Bit Clients

The parallel computing products now support Microsoft® Windows® HPC Server on 32-bit Windows clients.

Search for Cluster Head Nodes Using Active Directory

The `findResource` function can search Active Directory to identify your cluster head node. For more information, see the `findResource` reference page.

Enhanced Admin Center Support

You can now start and stop mdce services on remote hosts from Admin Center. For more information, see "Starting mdce".

New Remote Cluster Access Object

New functionality is available that lets you mirror job data from a remote cluster to a local data location. This supports the generic scheduler interface when making remote submissions to a scheduler or when using a nonshared file system. For more information, see the `RemoteClusterAccess` object reference page.

Version 5.0 (R2010b) Parallel Computing Toolbox Software

This table summarizes what is new in Version 5.0 (R2010b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

- "GPU Computing" on page 17
- "Job Manager Security and Secure Communications" on page 18
- "Generic Scheduler Interface Enhancements" on page 18
- "batch Now Able to Run Functions" on page 19
- "batch and matlabpool Accept Scheduler Object" on page 19
- "Enhanced Functions for Distributed Arrays" on page 20
- "Support for Microsoft Windows HPC Server 2008 R2" on page 21
- "User Permissions for MDCEUSER on Microsoft Windows" on page 21

GPU Computing

This release provides the ability to perform calculations on a graphics processing unit (GPU). Features include the ability to:

- Use a GPU array interface with several MATLAB built-in functions so that they automatically execute with single- or double-precision on the GPU — functions including `mldivide`, `mtimes`, `fft`, etc.
- Create kernels from your MATLAB function files for execution on a GPU
- Create kernels from your CU and PTX files for execution on a GPU
- Transfer data to/from a GPU and represent it in MATLAB with GPUArray objects

- Identify and select which one of multiple GPUs to use for code execution

For more information on all of these capabilities and the requirements to use these features, see “GPU Computing”.

Job Manager Security and Secure Communications

You now have a choice of four security levels when using the job manager as your scheduler. These levels range from no security to user authentication requiring passwords to access jobs on the scheduler.

You also have a choice to use secure communications between the job manager and workers.

For more detailed descriptions of these features and information about setting up job manager security, see “Setting Job Manager Security”.

The default setup uses no security, to match the behavior of past releases.

Generic Scheduler Interface Enhancements

Decode Functions Provided with Product

Generic scheduler interface decode functions for distributed and parallel jobs are now provided with the product. The two decode functions are named:

```
parallel.cluster.generic.distributedDecodeFcn
parallel.cluster.generic.parallelDecodeFcn
```

These functions are included on the workers’ path. If your submit functions make use of the definitions in these decode functions, you do not have to provide your own decode functions. For example, to use the standard decode function for distributed jobs, in your submit function set `MDCE_DECODE_FUNCTION` to `'parallel.cluster.generic.distributedDecodeFcn'`. For information on using the generic scheduler interface with submit and decode functions, see “Use the Generic Scheduler Interface”.

Enhanced Example Scripts

This release provides new sets of example scripts for using the generic scheduler interface. As in previous releases, the currently supported scripts are provided in the folder

```
matlabroot/toolbox/distcomp/examples/integration
```

In this location there is a folder for each type of scheduler:

- `condor` — Condor®
- `lsf` — Platform LSF®
- `pbs` — PBS
- `sge` — Sun™ Grid Engine
- `ssh` — generic UNIX®-based scripts
- `winmpiexec` — mpiexec on Windows

For the updated scheduler folders (`lsf`, `pbs`, `sge`), subfolders within each specify scripts for different cluster configurations: `shared`, `nonshared`, `remoteSubmission`.

For more information on the scripts and their updates, see the `README` file provided in each folder, or see "Supplied Submit and Decode Functions".

Compatibility Considerations. For those schedulers types with updated scripts in this release (`lsf`, `pbs`, `sge`), the old versions of the scripts are provided in the folder *`matlabroot`*`/toolbox/distcomp/examples/integration/old`. These old scripts might be removed in future releases.

batch Now Able to Run Functions

Batch jobs can now run functions as well as scripts. For more information, see the `batch` reference page.

batch and matlabpool Accept Scheduler Object

The `batch` function and the functional form of `matlabpool` now accept a scheduler object as their first input argument to specify which scheduler to

use for allocation of compute resources. For more information, see the `batch` and `matlabpool` reference pages.

Enhanced Functions for Distributed Arrays

qr Supports Distributed Arrays

The `qr` function now supports distributed arrays. For restrictions on this functionality, type

```
help distributed/qr
```

mldivide Enhancements

The `mldivide` function (`\`) now supports rectangular distributed arrays. Formerly, only square matrices were supported as distributed arrays.

When operating on a square distributed array, if the second input argument (or right-hand side of the operator) is replicated, `mldivide` now returns a distributed array.

Compatibility Considerations. In previous releases, `mldivide` returned a replicated array when the second (or right-hand side) input was replicated. Now it returns a distributed array.

chol Supports 'lower' option

The `chol` function now supports the `'lower'` option when operating on distributed arrays. For information on using `chol` with distributed arrays, type

```
help distributed/chol
```

eig and svd Return Distributed Array

When returning only one output matrix, the `eig` and `svd` functions now return a distributed array when the input is distributed. This behavior is now consistent with outputs when requesting more than one matrix, which returned distributed arrays in previous releases.

Compatibility Considerations. In previous releases, `eig` and `svd` returned a replicated array when you requested a single output. Now they return a distributed array if the output is a single matrix. The behavior when requesting more than one output is not changed.

transpose and ctranspose Support 2dbc

In addition to their original support for 1-D distribution schemes, the functions`ctranspose` and `transpose` now support 2-D block-cyclic (`'2dbc'`) distributed arrays.

Inf and NaN Support Multiple Formats

Distributed and codistributed arrays now support `nan`, `NaN`, `inf` and `Inf` for not-a-number and infinity values with the following functions:

Infinity Value	Not-a-Number
`codistributed.Inf`	`codistributed.NaN`
`codistributed.inf`	`codistributed.nan`
`distributed.Inf`	`distributed.NaN`
`distributed.inf`	`distributed.nan`

Support for Microsoft Windows HPC Server 2008 R2

Parallel Computing Toolbox software now supports Microsoft Windows HPC Server 2008 R2. There is no change in interface compared to using HPC Server 2008. Configurations and other toolbox utilities use the same settings to support both HPC Server 2008 and HPC Server 2008 R2.

User Permissions for MDCEUSER on Microsoft Windows

The user identified by the `MDCEUSER` parameter in the `mdce_def` file is now granted all necessary privileges on a Windows system when you install the mdce process. For information about what permissions are granted and how to reset them, see “Setting the User”.

Compatibility Considerations

In past releases, you were required to set the `MDCEUSER` permissions manually. Now this is done automatically when installing the mdce process.

Version 4.3 (R2010a) Parallel Computing Toolbox Software

This table summarizes what is new in Version 4.3 (R2010a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

- "New Save and Load Abilities for Distributed Arrays" on page 23
- "Enhanced Functions for Distributed Arrays" on page 24
- "Importing Configurations Programmatically" on page 24
- "Enhanced 2-D Block-Cyclic Array Distribution" on page 24
- "New Remote Startup of mdce Process" on page 24
- "Obtaining mdce Process Version" on page 24
- "Demo Updates" on page 25
- "taskFinish File for MATLAB Pool" on page 25

New Save and Load Abilities for Distributed Arrays

You now have the ability to save distributed arrays from the client to a single MAT-file. Subsequently, in the client you can load a distributed array from that file and have it automatically distributed to the MATLAB pool workers. The pool size and distribution scheme of the array do not have to be the same when you load the array as they were when you saved it.

You also can now load data directly into distributed arrays, even if the originally saved arrays were not distributed.

For more information, see the `dsave` and `dload` reference pages.

Enhanced Functions for Distributed Arrays

The `svd` function now supports single-precision and complex data in distributed arrays. Other functions enhanced to support single-precision distributed arrays are `chol`, `lu`, `mldivide`, and `eig`.

In addition to their original support for 1-D distribution by columns, the enhanced `tril` and `triu` functions now support arrays with 1-D distribution by rows or when the distribution dimension is greater than 2, and 2-D block-cyclic (`'2dbc'`) distributed arrays.

Importing Configurations Programmatically

A new function allows you to programmatically import parallel configurations. For more information, see the `importParallelConfig` reference page.

Enhanced 2-D Block-Cyclic Array Distribution

2-D block-cyclic (`'2dbc'`) array distribution now supports column orientation of the lab grid. The `codistributor2dbc` function now accepts the value `'col'` for its orientation argument, which is reflected in the codistributor object's `Orientation` property. For information on `'2dbc'` distribution and using lab grids, see "2-Dimensional Distribution".

New Remote Startup of mdce Process

New command-line functionality allows you to remotely start up MATLAB Distributed Computing Server processes on cluster machines from the desktop computer. For more information, see the `remotemdce` reference page.

Obtaining mdce Process Version

An enhancement to the `mdce` command lets you get the command version by executing

```
mdce -version
```

For more information on this command, see the `mdce` reference page.

Demo Updates

Product demos are available in the `Demos` node under `Parallel Computing Toolbox` in the help broswer.

Benchmarking A\b

This new demo benchmarks Parallel Computing Toolbox performance with the `mldivide` function.

BER Performance of Equalizer Types

Demos of BER Performance of Several Equalizer Types have been removed from Parallel Computing Toolbox, because Communications Toolbox now incorporates support for Parallel Computing Toolbox. See in the Communications Toolbox release notes.

taskFinish File for MATLAB Pool

The `taskFinish` file (`taskFinish.m`) for a MATLAB pool now executes when the pool closes.

Compatibility Considerations

In previous versions of the software, the `taskFinish` file executed when a MATLAB pool opened. Beginning with this release, it runs when the pool closes.

Version 4.2 (R2009b) Parallel Computing Toolbox Software

This table summarizes what is new in Version 4.2 (R2009b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

- "New Distributed Arrays" on page 26
- "Renamed codistributor Functions" on page 27
- "Enhancements to Admin Center" on page 29
- "Adding or Updating File Dependencies in an Open MATLAB Pool" on page 29
- "Updated globalIndices Function" on page 29
- "Support for Job Templates and Description Files with HPC Server 2008" on page 30
- "HPC Challenge Benchmarks" on page 30
- "pctconfig Enhanced to Support Range of Ports" on page 30
- "Random Number Generator on Client Versus Workers" on page 31

New Distributed Arrays

A new form of distributed arrays provides direct access from the client to data stored on the workers in a MATLAB pool. Distributed arrays have the same appearance and rules of indexing as regular arrays.

You can distribute an existing array from the client workspace with the command

```
D = distributed(X)
```

where `X` is an array in the client, and `D` is a distributed array with its data on the workers in the MATLAB pool. Distributing an array is performed outside an `spmd` statement, but a MATLAB pool must be open.

Codistributed arrays that you create on the workers within `spmd` statements are accessible on the client as distributed arrays.

The following new functions and methods support distributed arrays.

Function Name	Description
`distributed`	Distribute existing array from client workspace to workers
`distributed.rand`, `distributed.ones`, etc.	Create distributed array consistent with indicated method, constructing on workers only
`gather`	Transfer data from MATLAB pool workers to client
`isdistributed`	True for distributed array
`C(x,y)`	Indexing into distributed array `C` on client to access data stored as codistributed arrays on workers

Renamed codistributor Functions

As part of the general enhancements for distributed arrays, several changes to the codistributed interface appear in this release.

Compatibility Considerations

The following table summarizes the changes in function names relating to codistributed arrays.

Old Function Name	New Function Name
`codcolon`	`codistributed.colon`
`codistributed(..., 'convert')`	`codistributed(...)`

Old Function Name	New Function Name
`codistributed(...)` without `'convert'` option	`codistributed.build`
`codistributed(L, D)` using distribution scheme of `D` to define that of `L`	`codistributed.build(L, getCodistributor(D))`
`codistributor('1d', ...)`	Still available, but can also use `codistributor1d`
`codistributor('2d', ...)`	`codistributor('2dbc', ...)` or `codistributor2dbc`
`codistributor(arrayname)`	`getCodistributor`
`defaultLabGrid`	`codistributor2dbc.defaultLabGrid`
`defaultPartition`	`codistributor1d.defaultPartition`
`isa(X, 'codistributed')`	Still available, but can also use `iscodistributed(X)`
`localPart`	`getLocalPart`
`redistribute(D)` using default distribution scheme	`redistribute(D, codistributor())`
`redistribute(D1, D2)` using distribution scheme of `D2` to define that of `D1`	`redistribute(D1, getCodistributor(D2))`

Some object methods are now properties:

Old Method Name	New Property Name
`blockSize(codistObj)`	`codistObj.BlockSize`
`defaultBlockSize`	`codistributor2dbc.defaultBlockSize`
`distributionDimension(codistObj)`	`codistObj.Dimension`
`distributionPartition(codistObj)`	`codistObj.Partition`
`labGrid(codistObj)`	`codistObj.LabGrid`

Enhancements to Admin Center

Admin Center has several small enhancements, including more conveniently located menu choices, modified dialog boxes, properties dialog boxes for listed items, etc.

Adding or Updating File Dependencies in an Open MATLAB Pool

Enhancements to the `matlabpool` command let you add or update file dependencies in a running MATLAB pool. The new forms of the command are

```
matlabpool('addfiledependencies', filedepCell)
matlabpool updatefiledependencies
```

where `filedepCell` is a cell array of strings, identical in form to those you use when adding file dependencies to a job or when you open a MATLAB pool. The `updatefiledependencies` option replicates any file dependency changes to all the labs in the pool.

Updated globalIndices Function

The `globalIndices` function now requires that you specify the dimension of distribution as its second argument. Because this argument is required, it must precede the optional argument specifying the lab.

Compatibility Considerations

In previous toolbox versions, the `globalIndices` function accepted the lab argument before the dimension argument, and both were optional. Now the dimension argument is required, and it must precede the optional lab argument.

Support for Job Templates and Description Files with HPC Server 2008

Using job templates and job description files with Windows HPC Server 2008 lets you specify nodes and other scheduler properties for evaluating your jobs. To support these features, the ccsscheduler object has new properties:

- `ClusterVersion` — A string set to `'CCS'` or `'HPCServer2008'`
- `JobTemplate` — A string set to the name of the job template to use for all jobs
- `JobDescriptionFile` — A string set to the name of the XML file defining a base state for job creation

Compatibility Considerations

CCS is now just one of multiple versions of HPC Server. While `'ccs'` is still acceptable as a type of scheduler for the `findResource` function, you can also use `'hpcserver'` for this purpose. In the Configurations Manager, the new scheduler type is available by selecting **File > New > hpcserver (ccs)**.

HPC Challenge Benchmarks

Several new MATLAB files are available to demonstrate HPC Challenge benchmark performance. You can find the files in the folder *`matlabroot`*`/toolbox/distcomp/examples/benchmark/hpcchallenge`. Each file is self-documented with explanatory comments.

pctconfig Enhanced to Support Range of Ports

The `pctconfig` function now lets you specify a range of ports for the Parallel Computing Toolbox client session to use. This range also includes ports used for a pmode session.

Compatibility Considerations

You now specify the range of `pctconfig` ports with the `'portrange'` property; you no longer use the `'port'` property. As any client pmode session uses those ports of the `'portrange'` setting, you no longer use the `'pmodeport'` property.

Random Number Generator on Client Versus Workers

The random number generator of the MATLAB workers now use a slightly different seed from previous releases, so that all the MATLAB workers and the client have separate random number streams.

Compatibility Considerations

In past releases, while all the workers running a job had separate random number streams, the client had the same stream as one of the workers. Now the workers all have unique random number streams different from that of the client.

Version 4.1 (R2009a) Parallel Computing Toolbox Software

This table summarizes what is new in Version 4.1 (R2009a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

- "Number of Local Workers Increased to Eight" on page 32
- "Admin Center Allows Controlling of Cluster Resources" on page 33
- "Support for Microsoft Windows HPC Server 2008 (CCS v2)" on page 33
- "New Benchmarking Demos" on page 33
- "Pre-R2008b Distributed Array Syntax Now Generates Error" on page 34
- "LSF Support on Mac OS X 10.5.x" on page 34

Number of Local Workers Increased to Eight

You can now run up to eight local workers on your MATLAB client machine. If you do not specify the number of local workers in a command or configuration, the default number of local workers is determined by the value of the local scheduler's `ClusterSize` property, which by default is equal to the number of computational cores on the client machine.

Compatibility Considerations

In previous versions, the default number of local workers was four, regardless of the number of cores. If you want to run more local workers than cores (for example, four workers with only one or two cores), you must set the value of `ClusterSize` equal to or greater than the number of workers you need. Then you can specify the increased number of workers in the appropriate command

or configuration, or let your `ClusterSize` setting control the default number of workers.

Admin Center Allows Controlling of Cluster Resources

When using the MathWorks job manager, the Admin Center GUI now allows you to start, stop, and otherwise control job managers and MATLAB workers on your cluster nodes. For more information about Admin Center, see "Admin Center" in the MATLAB Distributed Computing Server documentation.

Compatibility Considerations

You can no longer start Admin Center from the MATLAB Desktop **Parallel** pull-down menu. You must start Admin Center from outside MATLAB by executing the following:

- *`matlabroot`*`/toolbox/distcomp/bin/admincenter` (on UNIX operating systems)
- *`matlabroot`*`\toolbox\distcomp\bin\admincenter.bat` (on Microsoft Windows operating systems)

Support for Microsoft Windows HPC Server 2008 (CCS v2)

The parallel computing products now support Microsoft Windows HPC Server 2008 (CCS v2), including service-oriented architecture (SOA) job submissions. There is no change to the programming interface for `ccs` options, other than the addition of a new CCS scheduler object property, `UseSOAJobSubmission`. For implications to the installation of the MATLAB Distributed Computing Server, see the online installation instructions at `http://www.mathworks.com/distconfig`.

New Benchmarking Demos

New benchmarking demos for Parallel Computing Toolbox can help you understand and evaluate performance of the parallel computing products. You can access these demos in the Help Browser under the `Parallel Computing Toolbox` node: expand the nodes for `Demos` then `Benchmarks`.

Pre-R2008b Distributed Array Syntax Now Generates Error

In R2008b, distributed array syntax was updated for codistributed arrays. In that release, the old form of the syntax still worked, but generated a warning. Now in R2009a, the old forms of the syntax no longer work and generate an error. For a summary of the syntax updates, see "Changed Function Names for Codistributed Arrays" on page 38.

LSF Support on Mac OS X 10.5.x

For availability of Platform LSF support on Macintosh® OS X 10.5.x, contact Platform Computing Corporation via their Web site at `http://www.platform.com/Products/platform-lsf/technical-information`. If Platform Computing does not support LSF on Mac OS X 10.5.x, then Parallel Computing Toolbox and MATLAB Distributed Computing Server cannot support this combination.

Version 4.0 (R2008b) Parallel Computing Toolbox Software

This table summarizes what is new in Version 4.0 (R2008b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

- "MATLAB® Compiler Product Support for Parallel Computing Toolbox Applications" on page 35
- "spmd Construct" on page 36
- "Composite Objects" on page 37
- "Configuration Validation" on page 37
- "Rerunning Failed Tasks" on page 37
- "Enhanced Job Control with Generic Scheduler Interface" on page 37
- "Changed Function Names for Codistributed Arrays" on page 38
- "Determining if a MATLAB Pool is Open" on page 38

MATLAB Compiler Product Support for Parallel Computing Toolbox Applications

This release offers the ability to convert Parallel Computing Toolbox applications, using MATLAB Compiler, into executables and shared libraries that can access MATLAB Distributed Computing Server. For information on this update to MATLAB Compiler, see "Applications Created with Parallel Computing Toolbox Can Be Compiled".

Limitations

- MATLAB Compiler does not support configurations that use the local scheduler or local workers (i.e., workers that run locally on the desktop machine running the MATLAB client session).
- Compiled Parallel Computing Toolbox applications do not support Simulink software. For a list of other unsupported products, see the Web page http://www.mathworks.com/products/ineligible_programs/.
- When workers are running a task from compiled code, they can execute only compiled code and toolbox code. They cannot execute functions contained in the current directory. Batch and MATLAB pool jobs attempt to change the worker working directory to the client working directory. When noncompiled files in the current directory conflict with compiled versions (for example, files with different extensions), an error is thrown.

spmd Construct

A new single program multiple data (spmd) language construct allows enhanced interleaving of serial and parallel programming, with interlab communication.

The general form of an `spmd` statement is:

```
spmd
    <statements>
end
```

The block of code represented by `<statements>` executes in parallel on workers in the MATLAB pool. Data on the labs is available for access from the client via Composite objects. For more information, see the `spmd` reference page and “Distributing Arrays and Running SPMD”.

Compatibility Considerations

Because `spmd` is a new keyword, it will conflict with any user-defined functions or variables of the same name. If you have any code with functions or variables named `spmd`, you must rename them.

Composite Objects

Composite objects provide direct access from the client (desktop) program to data that is stored on labs in the MATLAB pool. The data of variables assigned inside an spmd block is available via Composites in the client. When a MATLAB pool is open, you can also create Composites directly from the client using the `Composite` function. See also "Distributing Arrays and Running SPMD".

Configuration Validation

The Configurations Manager is enhanced with the capability for validating configurations. Open the Configurations Manager on the MATLAB Desktop by clicking **Parallel > Manage Configurations**. For more information, see "Validate Configurations".

Rerunning Failed Tasks

When using a job manager, if a task does not complete due to certain system failures, it can attempt to rerun up to a specified number of times. New properties of a task object to control reruns and access information about rerun attempts are:

- `MaximumNumberOfRetries`
- `AttemptedNumberOfRetries`
- `FailedAttemptInformation`

Enhanced Job Control with Generic Scheduler Interface

The generic scheduler interface now allows you to cancel and destroy jobs and tasks and to investigate the state of a job. The following new properties of the generic scheduler object facilitate these features:

- `GetJobStateFcn`
- `DestroyJobFcn`
- `DestroyTaskFcn`
- `CancelJobFcn`

- `CancelTaskFcn`

New toolbox functions to accommodate this ability are:

- `getJobSchedulerData`
- `setJobSchedulerData`

For more information on this new functionality, see “Manage Jobs with Generic Scheduler”.

Changed Function Names for Codistributed Arrays

What was known in previous releases as distributed arrays are henceforth called *codistributed* arrays. Some functions related to constructing and accessing codistributed arrays have changed names in this release.

Compatibility Considerations

The following table summarizes the changes in function names relating to codistributed arrays. The first three functions behave exactly the same with no change in operation, arguments, etc. The `isa` function takes the argument `'codistributed'` in addition to the array in question.

Old Function Name	New Function Name
`distributed`	`codistributed`
`distributor`	`codistributor`
`dcolon`	`codcolon`
`isdistributed`	`isa(X, 'codistributed')`

Determining if a MATLAB Pool is Open

The function `matlabpool` now allows you to discover if a pool of workers is already open. The form of the command is:

```
matlabpool size
```

For more information about this option and others, see the `matlabpool` reference page.

Version 3.3 (R2008a) Parallel Computing Toolbox Software

This table summarizes what is new in Version 3.3 (R2008a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

- "Renamed Functions for Product Name Changes" on page 39
- "New batch Function" on page 40
- "Enhanced Job Creation Functions" on page 40
- "Increased Data Size Transfers" on page 40
- "Changed Function Names for Distributed Arrays" on page 40
- "Support for PBS Pro and TORQUE Schedulers" on page 41
- "findResource Now Sets Properties According to Configuration" on page 41
- "parfor Syntax Has Single Usage" on page 42
- "dfeval Now Destroys Its Job When Finished" on page 42

Renamed Functions for Product Name Changes

As of result of the product name changes, some function names are changing in this release.

Compatibility Considerations

Two function names are changed to correspond to the new product names:

- `dctconfig` has been renamed `pctconfig`.
- `dctRunOnAll` has been renamed `pctRunOnAll`.

New batch Function

The new `batch` function allows you to offload work from the client to one or more workers. The batch submission can run scripts that can include jobs that distribute work to other workers. For more information, see the `batch` reference page, and "Getting Started" in the Parallel Computing Toolbox User's Guide.

New Matlabpool Job

The batch functionality is implemented using the new MATLABpool job feature. A MATLAB pool job uses one worker to distribute a job to other workers, thereby freeing the client from the burden of tracking and job's progress and manipulating data. For more information, see the `createMatlabPoolJob` reference page.

Enhanced Job Creation Functions

The `createJob` and `createParallelJob` functions have been enhanced to run without requiring a scheduler object as an argument. This is also true for the new `createMatlabPoolJob` function. When a scheduler is not specified, the function uses the scheduler identified in the applicable parallel configuration. For details, see the reference page for each function.

Increased Data Size Transfers

The default size limitation on data transfers between clients and workers has been significantly increased. In previous releases the default limitation imposed by the JVM memory allocation was approximately 50 MB. The new higher limits are approximately 600 MB on 32-bit systems and 2 GB on 64-bit systems. See "Object Data Size Limitations".

Changed Function Names for Distributed Arrays

Several functions related to distributed arrays have changed names in this release.

Compatibility Considerations

The following table summarizes the changes in function names relating to distributed arrays.

Old Function Name	New Function Name
darray	distributed, distributor
distribute	distributed
dcolonpartition	defaultPartition
distribdim	distributionDimension
isdarray	isdistributed
labgrid	labGrid
local	localPart
partition	distributionPartition
localspan	globalIndices

Support for PBS Pro and TORQUE Schedulers

Parallel Computing Toolbox software now fully supports PBS Pro® and TORQUE schedulers. These schedulers are integrated into parallel configurations and scheduler-related functions like `findResource`.

Note If you do not have a shared file system between client and cluster machines, or if you cannot submit jobs directly to the scheduler from the client machine, any use of third-party schedulers for parallel jobs (including `pmode`, `matlabpool`, and `parfor`) requires that you use the generic scheduler interface.

---

findResource Now Sets Properties According to Configuration

The `findResouce` function now sets the properties on the object it creates according to the configuration identified in the function call.

Compatibility Considerations

In past releases, `findResource` could use a configuration to identify a scheduler, but did not apply the configuration settings to the scheduler object

properties. If your code uses separate statements to find an object then set properties, this still works, but is not necessary any more.

parfor Syntax Has Single Usage

The `parfor` statement is now recognized only for parallel `for`-loops, not for loops over a distributed range in parallel jobs.

Compatibility Considerations

In R2007b, the pre-existing form of `parfor` was replaced by `for i = (drange)`, but both forms of syntax were recognized in that release. Now `parfor` has only one context, so `parfor` statements used in parallel jobs in code for versions prior to R2007a must be modified to use `for (drange)`.

Limitations

P-Code Scripts. You can call P-code script files from within a `parfor`-loop, but P-code script cannot contain a `parfor`-loop.

sim Inside parfor-Loops. Running simulations in a `parfor`-loop with the `sim` command at the top level of the loop is not allowed in this release. A `sim` command visible in a `parfor`-loop generates an error, although you can call `sim` inside a function that is called from the loop. Be sure that the various labs running simulations do not have the same working directory, as interference can occur with the simulation data.

dfeval Now Destroys Its Job When Finished

When finished performing its distributed evaluation, the `dfeval` function now destroys the job it created.

Compatibility Considerations

If you have any scripts that rely on a job and its data still existing after the completion of `dfeval`, or that destroy the job after `dfeval`, these scripts will no longer work.

Version 3.2 (R2007b) Distributed Computing Toolbox Software

This table summarizes what is new in Version 3.2 (R2007b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

- "New Parallel for-Loops (parfor-Loops)" on page 43
- "Configurations Manager and Dialogs" on page 44
- "Default Configuration" on page 45
- "Parallel Profiler" on page 45
- "MDCE Script for Red Hat Removed" on page 45

New Parallel for-Loops (parfor-Loops)

New parallel `for`-loop (`parfor`-loop) functionality automatically executes a loop body in parallel on dynamically allocated cluster resources, allowing interleaved serial and parallel code. For details of new parfor functionality, see "Parallel for-Loops (parfor)" in the Distributed Computing Toolbox™ documentation.

Limitations

P-Code Scripts. You can call P-code script files from within a `parfor`-loop, but P-code script cannot contain a `parfor`-loop.

Compatibility Considerations

In past releases, `parfor` was a different function. The new parfor uses parentheses in defining its range to distinguish it from the old `parfor`.

New `parfor`:

```
parfor (ii = 1:N); <body of code>; end;
```

Old `parfor`:

```
parfor ii = 1:N; <body of code>; end;
```

For this release, the old form of `parfor` without parentheses is still supported, although it generates a warning. You can read more about the new form of this existing functionality in "Using a for-Loop Over a Distributed Range (for-drange)". You should update your existing `parfor` code to use the new form of `for`-loops over a distributed range (`for-drange`), thus,

```
for ii = drange(1:N); <body of code>; end;
```

Configurations Manager and Dialogs

This release introduces a new graphical user interface for creating and modifying user configurations, and for designating the default configuration used by some toolbox functions. For details about the configurations manager, see "Parallel Configurations for Cluster Access" in the Distributed Computing Toolbox documentation.

Compatibility Considerations

This new feature has no impact on how configurations are used in a program, only on how configurations are created and shared among users. In previous versions of the product, you modified your configurations by editing the file *`matlabroot`*`/toolbox/distcomp/user/distcompUserConfig.m`. Now the configuration data is stored as part of your MATLAB software preferences.

The new configurations manager cannot directly import old-style configurations that were defined in the `distcompUserConfig.m` file. However, a utility called `importDistcompUserConfig`, available on the MATLAB Central Web site, allows you to convert and import your existing configurations into the new configurations manager.

Visit `http://www.mathworks.com/matlabcentral` and search for `importDistcompUserConfig`.

Default Configuration

This version of the toolbox enables you to select a user configuration to use as the default. Thus, commands such as `pmode` and `matlabpool` will use the default configuration without your having to specify it each time you run the command. You can set the default configuration using the configurations graphical interface, or programmatically with the `defaultParallelConfig` function.

Parallel Profiler

A new parallel profiler graphical user interface generates reports on lab computation and communication times during execution of parallel jobs. For details about this new feature, see "Profiling Parallel Code".

MDCE Script for Red Hat Removed

The MDCE script `rh_mdce`, specific to Red Hat Linux®, has been removed from *`matlabroot`*`/toolbox/distcomp/util/bin`.

Compatibility Considerations

If you make use of this script, you must replace it with its more generic equivalent,

`matlabroot``/toolbox/distcomp/bin/mdce`.

Version 3.1 (R2007a) Distributed Computing Toolbox Software

This table summarizes what is new in Version 3.1 (R2007a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

- "Local Scheduler and Workers" on page 46
- "New pmode Interface" on page 47
- "New Default Scheduler for pmode" on page 47
- "Vectorized Task Creation" on page 47
- "Additional Submit and Decode Scripts" on page 48
- "Jobs Property of Job Manager Sorts Jobs by ID" on page 48
- "New Object Display Format" on page 48
- "Enhanced MATLAB Functions" on page 49
- "darray Function Replaces distributor Function" on page 49
- "rand Seeding Unique for Each Task or Lab" on page 50
- "Single-Threaded Computations on Workers" on page 50

Local Scheduler and Workers

A local scheduler allows you to schedule jobs and run up to four workers or labs on a single MATLAB client machine without requiring engine licenses. These workers/labs can run distributed jobs or parallel jobs, including pmode sessions, for all products for which the MATLAB client is licensed. This local scheduler and its workers do not require a job manager or third-party scheduler.

New pmode Interface

The interactive parallel mode (pmode) has a new interface. The pmode command input and displays of the lab outputs are provided in a user interface that you can separate from the MATLAB client Command Window.

Compatibility Considerations

In previous versions of Distributed Computing Toolbox, the pmode interface used the MATLAB Command Window, with the pmode input using a different prompt. The output from the labs was intermingled with the MATLAB client output.

New Default Scheduler for pmode

If you start pmode without specifying a configuration,

```
pmode start
```

pmode automatically starts a parallel job using the local scheduler with labs running on the client machine. For more information about running pmode, see "Interactive Parallel Computation with pmode" in the Distributed Computing Toolbox documentation.

Compatibility Considerations

In the previous version of the toolbox, when pmode was started without specifying a configuration, it searched the network for the first available job manager to use as a scheduler.

Vectorized Task Creation

The `createTask` function can now create a vector of tasks in a single call when you provide a cell array of cell arrays for input arguments. For full details, see the `createTask` reference page.

Compatibility Considerations

In previous versions of the distributed computing products, if your task function had an input argument that was a cell array of cell arrays, your code will need to be modified to run the same way in this release.

For example, your old code may have been written as follows so that the function `myfun` gets four cell array input arguments:

```
createTask(j, @myfun, 1, {{C1} {C2} {C3} {C4}})
```

In this new version, the same code will produce four tasks. To get the old functionality, you must wrap the four cell arrays in another cell array, so that `createTask` knows to create only one task.

```
createTask(j, @myfun, 1, { {{C1} {C2} {C3} {C4}} })
```

Additional Submit and Decode Scripts

There are several submit and decode functions provided with the toolbox for your use with the generic scheduler interface. These files are in the directory

```
matlabroot/toolbox/distcomp/examples/integration
```

This version of the toolbox includes new subdirectories for Platform LSF and PBS, to support network configurations in which the client and worker computers do not share a file system. For more information, see "Supplied Submit and Decode Functions" in the Distributed Computing Toolbox documentation.

Jobs Property of Job Manager Sorts Jobs by ID

The `Jobs` property of a job manager object now contains the jobs in the order in which they were created, as indicated by the `ID` property of each job. Similarly, the `findJob` function returns jobs sequenced by their `ID`, unless otherwise specified. This change makes job manager behavior consistent with the behavior of third-party schedulers.

Compatibility Considerations

In previous versions of the distributed computing products, when using a job manager, jobs were arranged in the `Jobs` property or by `findJob` according to the status of the job.

New Object Display Format

When you create distributed computing objects (scheduler, job, or task) without a semicolon at the end of the command, the object information is

displayed in a new format. This new format is also shown when you use the `display` function to view an object or simply type the object name at the command line.

Compatibility Considerations

With this enhancement, the output format shown when creating an object has changed.

Enhanced MATLAB Functions

Several MATLAB functions have been enhanced to work on distributed arrays:

- `cat`
- `find`
- `horzcat`
- `subsindex`
- `vertcat`

For a complete list of MATLAB functions that are enhanced to work on distributed arrays, see "Using MATLAB Functions on Codistributed Arrays" in the Distributed Computing Toolbox documentation.

darray Function Replaces distributor Function

The function `darray` now defines how an array is distributed among the labs in a parallel job.

Compatibility Considerations

In the previous version of the toolbox, the `distributor` function was used to define how an array was distributed. In many cases, you can replace a call to `distributor` with a call to `darray`. For example, if you used `distributor` without arguments as an input to an array constructor,

```
rand(m, n, distributor());
```

you can update the code to read,

```
rand(m, n, darray());
```

rand Seeding Unique for Each Task or Lab

The random generator seed is now initialized based on the task ID for distributed jobs, or the `labindex` for parallel jobs (including pmode). This ensures that the set of random numbers generated for each task or lab within a job is unique, even when you have more than 82 tasks or labs.

Compatibility Considerations

In the previous version of the distributed computing products, the `rand` function could by default generate the same set of numbers for some tasks or labs when these exceeded 82 for a job.

Single-Threaded Computations on Workers

Despite the ability in MATLAB software to perform multithreaded computations on multiple-CPU machines, the workers and labs running distributed and parallel jobs perform only single-threaded computations, so that multiprocessor cluster machines can better accommodate multiple workers or labs.

Version 3.0 (R2006b) Distributed Computing Toolbox Software

This table summarizes what is new in Version 3.0 (R2006b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as **Compatibility Considerations**, below. See also Summary.	Bug Reports Includes fixes

New features and changes introduced in this version are

- "Support for Windows Compute Cluster Server (CCS)" on page 51
- "Windows 64 Support" on page 52
- "Parallel Job Enhancements" on page 52
- "Distributed Arrays" on page 52
- "Interactive Parallel Mode (pmode)" on page 53
- "Moved MDCE Control Scripts" on page 53
- "rand Seeding Unique for Each Task or Lab" on page 54
- "Task ID Property Now Same as labindex" on page 55

Support for Windows Compute Cluster Server (CCS)

Distributed Computing Toolbox software and MATLAB® Distributed Computing Engine™ software now let you program jobs and run them on a Windows Compute Cluster Server. For information about programming in the toolbox to use Windows Compute Cluster Server (CCS) as your scheduler, see the `findResource` reference page, and see also "Find a Windows HPC Server Scheduler".

Windows 64 Support

The distributed computing products now support Windows 64 (Win64) for both MATLAB client and MATLAB worker machines.

Parallel Job Enhancements

Parallel Jobs Support Any Scheduler

Support for parallel jobs now extends to any type of scheduler. In previous releases, only the MathWorks job manager and mpiexec scheduler object supported parallel jobs. You can now run parallel jobs on clusters scheduled by a job manager, Windows Compute Cluster Server (CCS), Platform LSF, mpiexec, or using the generic scheduler interface. For programming information, see "Program Parallel Jobs".

New labSendReceive Function

The `labSendReceive` function is introduced in this release. This function performs the same things as both `labSend` and `labReceive`, but greatly reduces the risk of deadlock, because the send and receive happen simultaneously rather than by separate statements. For more information, see the `labSendReceive` reference page.

Improved Error Detection

This release offers improved error detection for miscommunication between labs running parallel jobs. Most notable among the improvements are error detection of mismatched `labSend` and `labReceive` statements.

Distributed Arrays

Distributed arrays are partitioned into segments, with each segment residing in the workspace of a different lab, so that each lab has its own array segment to work with. Reducing the size of the array that each lab has to store and process means a more efficient use of memory and faster processing, especially for large data sets. For more information, see "Working with Codistributed Arrays".

There are many new and enhanced MATLAB functions to work with distributed arrays in parallel jobs. For a listing of these functions and their reference pages, see "Job Management".

parfor: Parallel for-Loops

Parallel for-loops let you run a for-loop across your labs simultaneously. For more information, see "Using a for-Loop Over a Distributed Range (for-drange)" or the `parfor` reference page.

Interactive Parallel Mode (pmode)

The interactive parallel mode (pmode) lets you work interactively with a parallel job running simultaneously on a number of labs. Commands you type at the pmode command line are executed on all labs at the same time. Each lab executes the commands in its own workspace on its own local variables or segments of distributed arrays. For more information, see "Run Parallel Jobs Interactively Using pmode".

Moved MDCE Control Scripts

To provide greater consistency across all platforms, the MDCE control scripts for Windows have moved and those for UNIX and Macintosh have new names.

Compatibility Considerations

Windows Utilities Moved. In previous versions of the distributed computing products, the MDCE utilities for Windows computers were located in

```
matlabroot\toolbox\distcomp\bin\win32
```

The utilities are now located in

```
matlabroot\toolbox\distcomp\bin
```

The files that have moved are

```
nodestatus
mdce
startjobmanager
```

```
stopjobmanager
startworker
stopworker
mdce_def.bat
```

UNIX and Macintosh Utilities Renamed. In previous versions of the distributed computing products, the MDCE utilities for UNIX and Macintosh computers were called by

```
nodestatus.sh
startjobmanager.sh
stopjobmanager.sh
startworker.sh
stopworker.sh
```

You can now call these with the following commands:

```
nodestatus
startjobmanager
stopjobmanager
startworker
stopworker
```

Note For UNIX and Macintosh, `mdce` and `mdce_def.sh` have not been moved or renamed.

---

rand Seeding Unique for Each Task or Lab

The random generator seed is now initialized based on the task ID for distributed jobs, or the `labindex` for parallel jobs (including pmode). This ensures that the random numbers generated for each task or lab are unique within a job.

Compatibility Considerations

In previous versions of the distributed computing products, the `rand` function would by default generate the same set of numbers on each worker.

Task ID Property Now Same as labindex

Although you create only one task for a parallel job, the system copies this task for each worker that runs the job. For example, if a parallel job runs on four workers (labs), the `Tasks` property of the job contains four task objects. The first task in the job's `Tasks` property corresponds to the task run by the lab whose `labindex` is `1`, and so on, so that the `ID` property for the task object and `labindex` for the lab that ran that task have the same value. Therefore, the sequence of results returned by the `getAllOutputArguments` function corresponds to the value of `labindex` and to the order of tasks in the job's `Tasks` property.

Compatibility Considerations

In past releases, there was no correlation between `labindex` and the task `ID` property.

Compatibility Summary for Parallel Computing Toolbox Software

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions. Details are provided with the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version V5.2 (R2011b)	See the **Compatibility Considerations** subheading for this change: • "Latest NVIDIA CUDA Device Driver" on page 7 • "Conversion of Error and Warning Message Identifiers" on page 9 • "Task Error Properties Updated" on page 11 • "Upgrade Parallel Computing Products Together" on page 11
V5.1 (R2011a)	See the **Compatibility Considerations** subheading for this change: • "NVIDIA CUDA Driver 3.2 Support" on page 14 • "Enhanced mtimes Support" on page 15

Version (Release)	New Features and Changes with Version Compatibility Impact
V5.0 (R2010b)	See the **Compatibility Considerations** subheading for this change: • "Generic Scheduler Interface Enhancements" on page 18 • "mldivide Enhancements" on page 20 • "eig and svd Return Distributed Array" on page 20 • "User Permissions for MDCEUSER on Microsoft Windows" on page 21
V4.3 (R2010a)	See the **Compatibility Considerations** subheading for this change: • "taskFinish File for MATLAB Pool" on page 25
V4.2 (R2009b)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Renamed codistributor Functions" on page 27 • "Updated globalIndices Function" on page 29 • "Support for Job Templates and Description Files with HPC Server 2008" on page 30 • "pctconfig Enhanced to Support Range of Ports" on page 30 • "Random Number Generator on Client Versus Workers" on page 31

Version (Release)	New Features and Changes with Version Compatibility Impact
V4.1 (R2009a)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Number of Local Workers Increased to Eight" on page 32 • "Admin Center Allows Controlling of Cluster Resources" on page 33
V4.0 (R2008b)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "spmd Construct" on page 36 • "Changed Function Names for Codistributed Arrays" on page 38
V3.3 (R2008a)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Renamed Functions for Product Name Changes" on page 39 • "Changed Function Names for Distributed Arrays" on page 40 • "findResource Now Sets Properties According to Configuration" on page 41 • "parfor Syntax Has Single Usage" on page 42 • "dfeval Now Destroys Its Job When Finished" on page 42

Version (Release)	New Features and Changes with Version Compatibility Impact
V3.2 (R2007b)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "New Parallel for-Loops (parfor-Loops)" on page 43 • "Configurations Manager and Dialogs" on page 44 • "MDCE Script for Red Hat Removed" on page 45
V3.1 (R2007a)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "New pmode Interface" on page 47 • "New Default Scheduler for pmode" on page 47 • "Vectorized Task Creation" on page 47 • "Jobs Property of Job Manager Sorts Jobs by ID" on page 48 • "New Object Display Format" on page 48 • "darray Function Replaces distributor Function" on page 49 • "rand Seeding Unique for Each Task or Lab" on page 50
V3.0 (R2006b)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Moved MDCE Control Scripts" on page 53 • "rand Seeding Unique for Each Task or Lab" on page 54