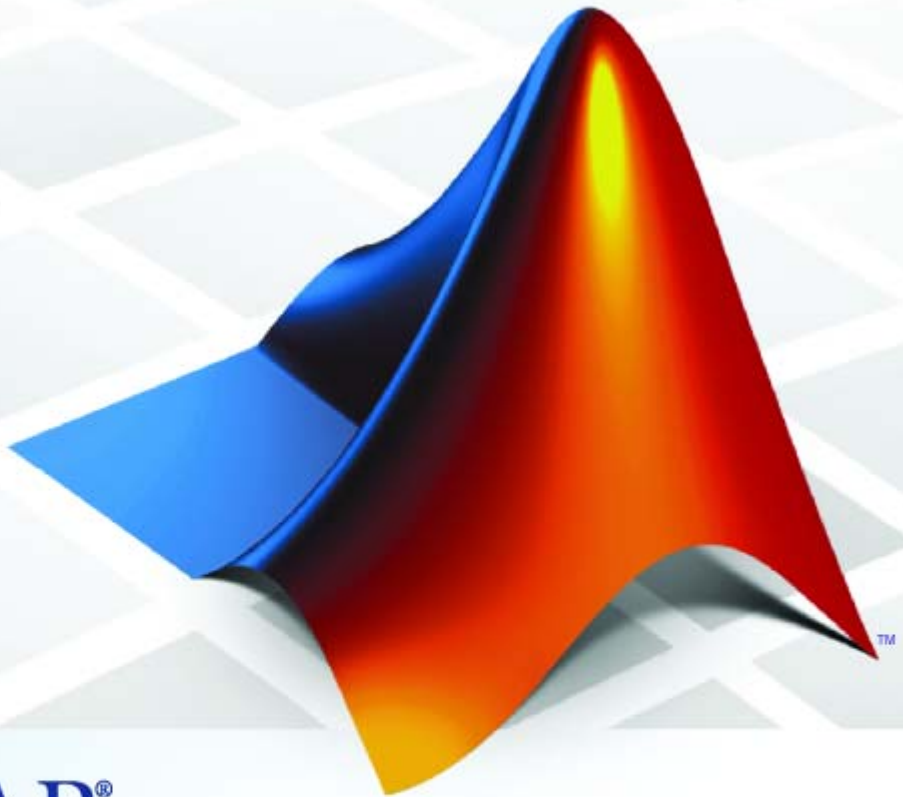


MATLAB[®] Distributed Computing Server[™] 4

System Administrator's Guide



MATLAB[®]

How to Contact The 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.

MATLAB® Distributed Computing Server™ System Administrator's Guide

© COPYRIGHT 2005–2009 by The MathWorks, Inc.

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

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

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

November 2005 Online only
December 2005 Online only
March 2006 Online only
September 2006 Online only
March 2007 Online only
September 2007 Online only
March 2008 Online only
October 2008 Online only
March 2009 Online only
September 2009 Online only

New for Version 2.0 (Release 14SP3+)
Revised for Version 2.0 (Release 14SP3+)
Revised for Version 2.0.1 (Release 2006a)
Revised for Version 3.0 (Release 2006b)
Revised for Version 3.1 (Release 2007a)
Revised for Version 3.2 (Release 2007b)
Revised for Version 3.3 (Release 2008a)
Revised for Version 4.0 (Release 2008b)
Revised for Version 4.1 (Release 2009a)
Revised for Version 4.2 (Release 2009b)

Introduction

1

Product Overview	1-2
Overview	1-2
Determining Product Installation and Versions	1-3
Toolbox and Server Components	1-4
Job Managers, Workers, and Clients	1-4
Third-Party Schedulers	1-6
Components on Mixed Platforms or Heterogeneous Clusters	1-7
mdce Service	1-7
Using Parallel Computing Toolbox Software	1-8

Network Administration

2

Preparing for Parallel Computing	2-2
Before You Start	2-2
Planning Your Network Layout	2-2
Network Requirements	2-3
Fully Qualified Domain Names	2-3
Security Considerations	2-4
Installing and Configuring	2-5
Using a Different MPI Build on UNIX Operating Systems	2-6
Building MPI	2-6
Using Your MPI Build	2-6

Shutting Down a Job Manager Configuration	2-9
UNIX and Macintosh Operating Systems	2-9
Microsoft Windows Operating Systems	2-11
Customizing Server Services	2-13
Defining the Script Defaults	2-13
Overriding the Script Defaults	2-15
Accessing Service Record Files	2-17
Locating Log Files	2-17
Locating Checkpoint Directories	2-18
Troubleshooting	2-19
License Errors	2-19
Verifying Multicast Communications	2-21
Memory Errors on UNIX Operating Systems	2-22
Running Server Processes from a Windows Network	
Installation	2-22
Required Ports	2-23
Ephemeral TCP Ports with Job Manager	2-24

Admin Center

3

Starting Admin Center	3-2
Setting Up Resources	3-3
Adding Hosts	3-3
Starting a Job Manager	3-4
Starting Workers	3-5
Stopping, Destroying, Resuming, Restarting Processes ...	3-7
Moving a Worker	3-8
Updating the Display	3-8
Testing Connectivity	3-9
Saving and Loading Sessions	3-13

Preparing for User Configurations 3-14

Control Script Reference

4

mdce Process Control 4-2

Job Manager Control 4-2

Worker Control 4-2

Control Scripts — Alphabetical List

5

Glossary

Index

Introduction

This chapter provides an introduction to the concepts and terms of Parallel Computing Toolbox™ software and MATLAB® Distributed Computing Server™ software.

- “Product Overview” on page 1-2
- “Toolbox and Server Components” on page 1-4
- “Using Parallel Computing Toolbox Software” on page 1-8

Product Overview

In this section...
“Overview” on page 1-2
“Determining Product Installation and Versions” on page 1-3

Overview

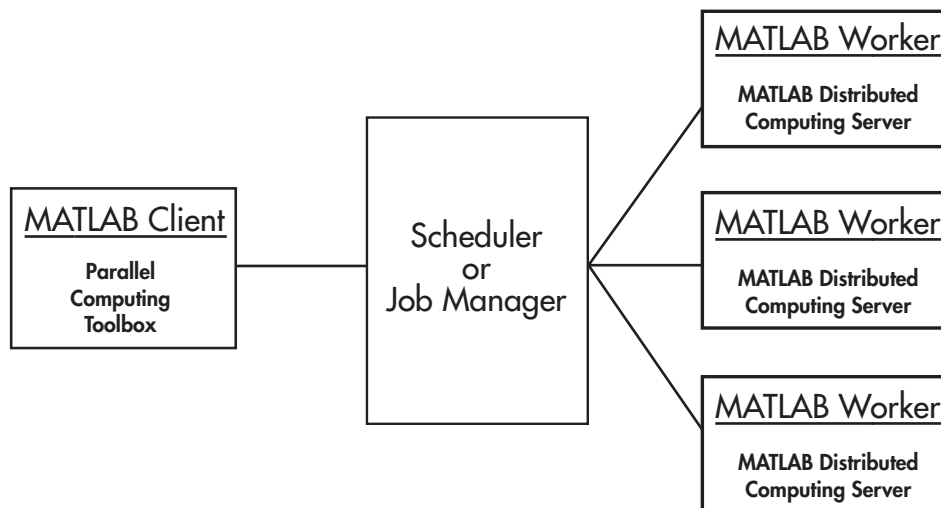
Parallel Computing Toolbox and MATLAB Distributed Computing Server software let you solve computationally and data-intensive problems using MATLAB® and Simulink® on multicore and multiprocessor computers. Parallel processing constructs such as parallel for-loops and code blocks, distributed arrays, parallel numerical algorithms, and message-passing functions let you implement task-parallel and data-parallel algorithms at a high level in MATLAB without programming for specific hardware and network architectures.

A *job* is some large operation that you need to perform in your MATLAB session. A job is broken down into segments called *tasks*. You decide how best to divide your job into tasks. You could divide your job into identical tasks, but tasks do not have to be identical.

The MATLAB session in which the job and its tasks are defined is called the *client* session. Often, this is on the machine where you program MATLAB. The client uses Parallel Computing Toolbox software to perform the definition of jobs and tasks. The MATLAB Distributed Computing Server product performs the execution of your job by evaluating each of its tasks and returning the result to your client session.

The *job manager* is the part of the server software that coordinates the execution of jobs and the evaluation of their tasks. The job manager distributes the tasks for evaluation to the server’s individual MATLAB sessions called *workers*. Use of the MathWorks™ job manager is optional; the distribution of tasks to workers can also be performed by a third-party scheduler, such as Window HPC Server (including CCS), a Platform LSF® scheduler, or a PBS Pro® scheduler.

See the “Glossary” on page Glossary-1 for definitions of the parallel computing terms used in this manual.



Basic Parallel Computing Configuration

Determining Product Installation and Versions

To determine if Parallel Computing Toolbox software is installed on your system, type this command at the MATLAB prompt:

```
ver
```

When you enter this command, MATLAB displays information about the version of MATLAB you are running, including a list of all toolboxes installed on your system and their version numbers.

You can run the `ver` command as part of a task in a distributed or parallel application to determine what version of MATLAB Distributed Computing Server software is installed on a worker machine. Note that the toolbox and server software must be the same version.

Toolbox and Server Components

In this section...
“Job Managers, Workers, and Clients” on page 1-4
“Third-Party Schedulers” on page 1-6
“Components on Mixed Platforms or Heterogeneous Clusters” on page 1-7
“mdce Service” on page 1-7

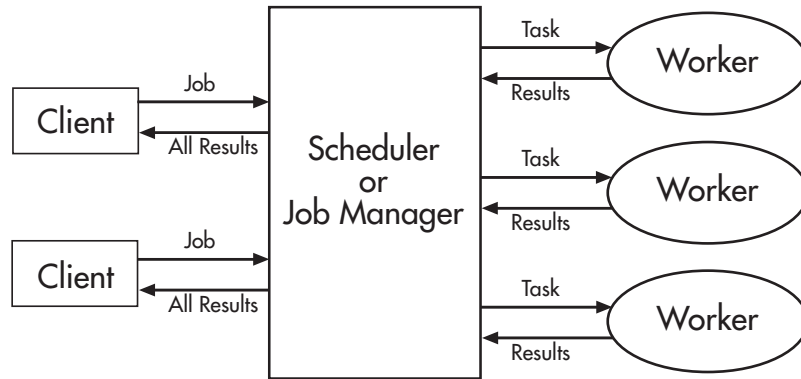
Job Managers, Workers, and Clients

The optional job manager can run on any machine on the network. The job manager runs jobs in the order in which they are submitted, unless any jobs in its queue are promoted, demoted, canceled, or destroyed.

Each worker receives a task of the running job from the job manager, executes the task, returns the result to the job manager, and then receives another task. When all tasks for a running job have been assigned to workers, the job manager starts running the next job with the next available worker.

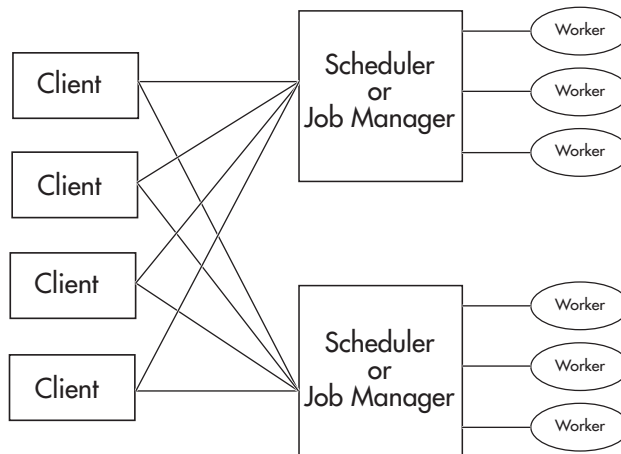
A MATLAB Distributed Computing Server network configuration usually includes many workers that can all execute tasks simultaneously, speeding up execution of large MATLAB jobs. It is generally not important which worker executes a specific task. Each worker evaluates tasks one at a time, returning the results to the job manager. The job manager then returns the results of all the tasks in the job to the client session.

Note For testing your application locally or other purposes, you can configure a single computer as client, worker, and job manager. You can also have more than one worker session or more than one job manager session on a machine.



Interactions of Parallel Computing Sessions

A large network might include several job managers as well as several client sessions. Any client session can create, run, and access jobs on any job manager, but a worker session is registered with and dedicated to only one job manager at a time. The following figure shows a configuration with multiple job managers.



Configuration with Multiple Clients and Job Managers

Third-Party Schedulers

As an alternative to using the MathWorks job manager, you can use a third-party scheduler. This could be a Microsoft® Windows HPC Server (including CCS), Platform LSF scheduler, PBS Pro scheduler, TORQUE scheduler, mpiexec, or a generic scheduler.

Choosing Between a Scheduler and Job Manager

You should consider the following when deciding to use a scheduler or the MathWorks job manager for distributing your tasks:

- Does your cluster already have a scheduler?

If you already have a scheduler, you may be required to use it as a means of controlling access to the cluster. Your existing scheduler might be just as easy to use as a job manager, so there might be no need for the extra administration involved.

- Is the handling of parallel computing jobs the only cluster scheduling management you need?

The MathWorks job manager is designed specifically for MathWorks parallel computing applications. If other scheduling tasks are not needed, a third-party scheduler might not offer any advantages.

- Is there a file sharing configuration on your cluster already?

The MathWorks job manager can handle all file and data sharing necessary for your parallel computing applications. This might be helpful in configurations where shared access is limited.

- Are you interested in batch or interactive processing?

When you use a job manager, worker processes usually remain running at all times, dedicated to their job manager. With a third-party scheduler, workers are run as applications that are started for the evaluation of tasks, and stopped when their tasks are complete. If tasks are small or take little time, starting a worker for each one might involve too much overhead time.

- Are there security concerns?

Your scheduler may be configured to accommodate your particular security requirements.

- How many nodes are on your cluster?

If you have a large cluster, you probably already have a scheduler. Consult your MathWorks representative if you have questions about cluster size and the job manager.

- Who administers your cluster?

The person administering your cluster might have a preference for how jobs are scheduled.

Components on Mixed Platforms or Heterogeneous Clusters

Parallel Computing Toolbox software and MATLAB Distributed Computing Server software are supported on Windows®, UNIX® (including Linux®), and Macintosh® operating systems. Mixed platforms are supported, so that the clients, job managers, and workers do not have to be on the same platform. The cluster can also be comprised of both 32-bit and 64-bit machines, so long as your data does not exceed the limitations posed by the 32-bit systems.

For a complete listing of all network requirements, including those for heterogeneous environments, see the System Requirements page for MATLAB Distributed Computing Server software at

<http://www.mathworks.com/products/distriben/requirements.html>

In a mixed platform environment, be sure to follow the proper installation instructions for each local machine on which you are installing the software.

mdce Service

If you are using the MathWorks job manager, every machine that hosts a worker or job manager session must also run the mdce service.

The mdce service recovers worker and job manager sessions when their host machines crash. If a worker or job manager machine crashes, when mdce starts up again (usually configured to start at machine boot time), it automatically restarts the job manager and worker sessions to resume their sessions from before the system crash.

Using Parallel Computing Toolbox Software

A typical Parallel Computing Toolbox client session includes the following steps:

- 1 Find a Job Manager (or scheduler)** — Your network may have one or more job managers available (but usually only one scheduler). The function you use to find a job manager or scheduler creates an object in your current MATLAB session to represent the job manager or scheduler that will run your job.
- 2 Create a Job** — You create a job to hold a collection of tasks. The job exists on the job manager (or scheduler's data location), but a job object in the local MATLAB session represents that job.
- 3 Create Tasks** — You create tasks to add to the job. Each task of a job can be represented by a task object in your local MATLAB session.
- 4 Submit a Job to the Job Queue for Execution** — When your job has all its tasks defined, you submit it to the queue in the job manager or scheduler. The job manager or scheduler distributes your job's tasks to the worker sessions for evaluation. When all of the workers are completed with the job's tasks, the job moves to the finished state.
- 5 Retrieve the Job's Results** — The resulting data from the evaluation of the job is available as a property value of each task object.
- 6 Destroy the Job** — When the job is complete and all its results are gathered, you can destroy the job to free memory resources.

Network Administration

This chapter provides information useful for network administration of Parallel Computing Toolbox software and MATLAB Distributed Computing Server software.

- “Preparing for Parallel Computing” on page 2-2
- “Installing and Configuring” on page 2-5
- “Using a Different MPI Build on UNIX Operating Systems” on page 2-6
- “Shutting Down a Job Manager Configuration” on page 2-9
- “Customizing Server Services” on page 2-13
- “Accessing Service Record Files” on page 2-17
- “Troubleshooting” on page 2-19

Preparing for Parallel Computing

In this section...
“Before You Start” on page 2-2
“Planning Your Network Layout” on page 2-2
“Network Requirements” on page 2-3
“Fully Qualified Domain Names” on page 2-3
“Security Considerations” on page 2-4

This section discusses the requirements and configurations for your network to support parallel computing.

Before You Start

Before attempting to install Parallel Computing Toolbox software and MATLAB Distributed Computing Server software, read Chapter 1, “Introduction” to familiarize yourself with the concepts and vocabulary of the products.

Planning Your Network Layout

Generally, it is easy to decide which machines will run worker processes and which will run client processes. Worker sessions usually run on the cluster of machines dedicated to that purpose. The MATLAB client session usually runs where MATLAB programs are run, often on a user’s desktop.

The job manager process should run on a stable machine, with adequate resources to manage the number of tasks and amount of data expected in your parallel computing applications.

The following table shows what products and processes are needed for each of these roles in the parallel computing configuration.

Session	Product	Processes
Client	Parallel Computing Toolbox	MATLAB with toolbox
Worker	MATLAB Distributed Computing Server	worker; mdce service (if using a job manager)
Job manager	MATLAB Distributed Computing Server	mdce service; job manager

The server software includes the mdce service or daemon. The mdce service is separate from the worker and job manager processes, and it must be running on all machines that run job manager sessions or workers that are registered with a job manager. (The mdce service is not used with third-party schedulers.)

You can install both toolbox and server software on the same machine, so that one machine can run both client and server sessions.

Network Requirements

To view the network requirements for MATLAB Distributed Computing Server software, visit the product requirements page on the MathWorks Web site at

<http://www.mathworks.com/products/distriben/requirements.html>

Fully Qualified Domain Names

MATLAB Distributed Computing Server software and Parallel Computing Toolbox software support both short hostnames and fully qualified domain names. The default usage is short hostnames. If your network requires fully qualified hostnames, you can use the `mdce_def` file to identify the worker nodes by their full names. See “Customizing Server Services” on page 2-13. To set the hostname used for a MATLAB client session, see the `pctconfig` reference page.

Security Considerations

The parallel computing products do not provide any security measures. Therefore, be aware of the following security considerations:

- MATLAB workers run as whatever user the administrator starts the node's mdce service under. By default, the mdce service starts as root on UNIX operating systems, and as LocalSystem on Microsoft Windows operating systems. Because MATLAB provides system calls, users can submit jobs that execute shell commands.
- The mdce service does not enforce any access control or authentication. Anyone with local or remote access to the mdce services can start and stop their workers and job managers, and query for their status.
- The job manager does not restrict access to the cluster, nor to job and task data. Using a third-party scheduler instead of the MathWorks job manager could allow you to take advantage of the security measures it provides.
- The parallel computing processes must all be on the same side of a firewall, or you must take measures to enable them to communicate with each other through the firewall. Workers running tasks of the same parallel job cannot be firewalled off from each other, because their MPI-based communication will not work.
- If certain ports are restricted, you can specify the ports used for parallel computing. See "Defining the Script Defaults" on page 2-13.
- If your network supports multicast, the parallel computing processes accommodate multicast. However, because multicast is disabled on many networks for security reasons, you might require unicast communication between parallel computing processes. Most examples of parallel computing scripts and functions in this documentation show unicast usage.
- If your organization is a member of the Internet Multicast Backbone (MBone), make sure that your parallel computing cluster is isolated from MBone access if you are using multicast for parallel computing. This is generally the default condition. If you have any questions about MBone membership, contact your network administrator.

Installing and Configuring

To find the most up-to-date instructions for installing and configuring the current or past versions of the parallel computing products, visit the MathWorks Web site at

http://www.mathworks.com/support/product/DM/installation/ver_current/

Using a Different MPI Build on UNIX Operating Systems

In this section...

“Building MPI” on page 2-6

“Using Your MPI Build” on page 2-6

Building MPI

To use an MPI build that differs from the one provided with Parallel Computing Toolbox, this stage outlines the steps for creating an MPI build. If you already have an alternative MPI build, proceed to “Using Your MPI Build” on page 2-6.

- 1 Unpack the MPI sources into the target file system on your machine. For example, suppose you have downloaded `mpich2-distro.tgz` and want to unpack it into `/opt` for building:

```
# cd /opt
# mkdir mpich2 && cd mpich2
# tar zxvf path/to/mpich2-distro.tgz
# cd mpich2-1.0.8
```

- 2 Build your MPI using the `enable-sharedlibs` option (this is vital, as you must build a shared library MPI, binary compatible with `MPICH2-1.0.8` for R2009b and later). For example, the following commands build an MPI with the `nemesis` channel device and the `gforker` launcher.

```
# ./configure -prefix=/opt/mpich2/mpich2-1.0.8 \
--enable-sharedlibs=gcc \
--with-device=ch3:nemesis \
--with-pm=gforker 2>&1 | tee log
# make 2>&1 | tee -a log
# make install 2>&1 | tee -a log
```

Using Your MPI Build

When your MPI build is ready, this stage highlights the steps to use it. To get the Parallel Computing Toolbox `mpiexec` scheduler working with a different MPI build, follow these steps. Most of these steps are also needed if you want to use a different MPI build with third party-schedulers (LSF, generic).

- 1 Test your build by running the `mpiexec` executable. The build should be ready to test if its `bin/mpiexec` and `lib/libmpich.so` are available in the MPI installation location.

Following the example in “Building MPI” on page 2-6, `/opt/mpich2/mpich2-1.0.8/bin/mpiexec` and `/opt/mpich2/mpich2-1.0.8/lib/libmpich.so` are ready to use, so you can test the build with:

```
$ /opt/mpich2/mpich2-1.0.8/bin/mpiexec -n 4 hostname
```

- 2 Create an `mpiLibConf` function to direct Parallel Computing Toolbox to use your new MPI. Write your `mpiLibConf.m` to return the appropriate information for your build. For example:

```
function [primary, extras] = mpiLibConf
primary = '/opt/mpich2/mpich2-1.0.8/lib/libmpich.so';
extras = {};
```

The primary path *must* be valid *on the cluster*; and your `mpiLibConf.m` file must be higher on the cluster workers’ path than `matlabroot/toolbox/distcomp/mpi`. (Sending `mpiLibConf.m` as a file dependency for this purpose does not work. You can get the `mpiLibConf.m` function on the worker path by either moving the file into a directory on the path, or by having the scheduler use `cd` in its command so that it starts the MATLAB worker from within the directory that contains the function.)

- 3 Determine necessary daemons and command-line options.
 - Determine all necessary daemons (often something like `mpdboot` or `smpd`). The `gforker` build example in this section uses an MPI that needs no services or daemons running on the cluster, but it can use only the local machine.
 - Determine the correct command-line options to pass to `mpiexec`.
- 4 Use one of the following options to set up your scheduler to use your new MPI build:
 - For the simplest case of the `mpiexec` scheduler, set up a configuration to use the `mpiexec` executable from your new MPI build. It is crucial that you use matching `mpiexec`, MPI library, and any daemons (if

any), together. Set the configuration's `MpiexecFileName` property to `/opt/mpich2/mpich2-1.0.8/bin/mpexec`.

- If you are using a generic scheduler or LSF, modify your parallel wrapper script to pick up the correct `mpexec`. Additionally, there may be a stage in the wrapper script where the MPI daemons are launched.

The parallel submission wrapper script must:

- Determine which nodes are allocated by the scheduler.
- Start required daemon processes. For example, for the MPD process manager this means calling "`mpdboot -f <nodefile>`".
- Define which `mpexec` executable to use for starting workers.
- Stop the daemon processes. For example, for the MPD process manager this means calling "`mpdallexit`".

For examples of parallel wrapper scripts, see `matlabroot/toolbox/distcomp/examples/integration/`; specifically for an example of Sun Grid Engine, look in the folder `sg` for `sgParallelWrapper.sh`. Adopt and modify the appropriate script for your particular cluster usage.

Shutting Down a Job Manager Configuration

In this section...

“UNIX and Macintosh Operating Systems” on page 2-9

“Microsoft Windows Operating Systems” on page 2-11

If you are done using the job manager and its workers, you might want to shut down the server software processes so that they are not consuming network resources. You do not need to be at the computer running the processes that you are shutting down. You can run these commands from any machine with network access to the processes. The following sections explain shutting down the processes for different platforms.

UNIX and Macintosh Operating Systems

Enter the commands of this section at the prompt in a UNIX shell.

Stopping the Job Manager and Workers

- 1 To shut down the job manager, enter the commands

```
cd matlabroot/toolbox/distcomp/bin
```

(Enter the following command on a single line.)

```
stopjobmanager -remotehost <job manager hostname> -name  
<MyJobManager> -v
```

If you have more than one job manager running, stop each of them individually by host and name.

For a list of all options to the script, type

```
stopjobmanager -help
```

- 2 For each MATLAB worker you want to shut down, enter the commands

```
cd matlabroot/toolbox/distcomp/bin  
stopworker -remotehost <worker hostname> -v
```

If you have more than one worker session running, you can stop each of them individually by host and name.

```
stopworker -name worker1 -remotehost <worker hostname>
stopworker -name worker2 -remotehost <worker hostname>
```

For a list of all options to the script, type

```
stopworker -help
```

Stopping and Uninstalling the mdce Daemon

Normally, you configure the mdce daemon to start at system boot time and continue running until the machine shuts down. However, if you plan to uninstall the MATLAB Distributed Computing Server product from a machine, you might want to uninstall the mdce daemon also, because you no longer need it.

Note You must have root privileges to stop or uninstall the mdce daemon.

1 Use the following command to stop the mdce daemon:

```
/etc/init.d/mdce stop
```

2 Remove the installed link to prevent the daemon from starting up again at system reboot:

```
cd /etc/init.d/
rm mdce
```

Stopping the Daemon Manually. If you used the alternative manual startup of the mdce daemon, use the following commands to stop it manually:

```
cd matlabroot/toolbox/distcomp/bin
mdce stop
```

Microsoft Windows Operating Systems

Stopping the Job Manager and Workers

Enter the commands of this section at the prompt in a DOS command window.

- 1 To shut down the job manager, enter the commands

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

(Enter the following command on a single line.)

```
stopjobmanager -remotehost <job manager hostname> -name  
<MyJobManager> -v
```

If you have more than one job manager running, stop each of them individually by host and name.

For a list of all options to the script, type

```
stopjobmanager -help
```

- 2 For each MATLAB worker you want to shut down, enter the commands

```
cd matlabroot\toolbox\distcomp\bin  
stopworker -remotehost <worker hostname> -name <worker name> -v
```

If you have more than one worker session running, you can stop each of them individually by host and name.

```
stopworker -remotehost <worker hostname> -name <worker1 name>  
stopworker -remotehost <worker hostname> -name <worker2 name>
```

For a list of all options to the script, type

```
stopworker -help
```

Stopping and Uninstalling the mdce Service

Normally, you configure the mdce service to start at system boot time and continue running until the machine shuts down. If you need to stop the mdce

service while leaving the machine on, enter the following commands at a DOS command prompt:

```
cd matlabroot\toolbox\distcomp\bin  
mdce stop
```

If you plan to uninstall the MATLAB Distributed Computing Server product from a machine, you might want to uninstall the mdce service also, because you no longer need it.

You do not need to stop the service before uninstalling it.

To uninstall the mdce service, enter the following commands at a DOS command prompt:

```
cd matlabroot\toolbox\distcomp\bin  
mdce uninstall
```

Customizing Server Services

In this section...
“Defining the Script Defaults” on page 2-13
“Overriding the Script Defaults” on page 2-15

The MATLAB Distributed Computing Server scripts run using several default parameters. You can customize the scripts, as described in this section.

Defining the Script Defaults

The scripts for the server services require values for several parameters. These parameters set the process name, the user name, log file location, ports, etc. Some of these can be set using flags on the command lines, but the full set of user-configurable parameters are in the `mdce_def` file.

Note The startup script flags take precedence over the settings in the `mdce_def` file.

The default parameters used by the server service scripts are defined in the file:

- `matlabroot\toolbox\distcomp\bin\mdce_def.bat` (on Microsoft Windows operating systems)
- `matlabroot/toolbox/distcomp/bin/mdce_def.sh` (on UNIX or Macintosh operating systems)

To set the default parameters, edit this file before installing or starting the `mdce` service.

The `mdce_def` file is self-documented, and includes explanations of all its parameters.

Note If you want to run more than one job manager on the same machine, they must all have unique names. Specify the names using flags with the startup commands.

Setting the User

By default, the job manager and worker services run as the user who starts them. You can run the services as a different user with the following settings in the `mdce_def` file.

Parameter	Description
MDCEUSER	Set this parameter to run the mdce services as a user different from the user who starts the service. On a UNIX operating system, set the value before starting the service; on a Windows operating system, set it before installing the service.
MDCEPASS	On a Windows operating system, set this parameter to specify the password for the user identified in the MDCEUSER parameter; otherwise, the system prompts you for the password when the service is installed.

On UNIX operating systems, MDCEUSER requires that the current machine has the `sudo` utility installed, and that the current user be allowed to use `sudo` to execute commands as the user identified by MDCEUSER. For further information, refer to your system documentation on the `sudo` and `sudoers` utilities (for example, `man sudo` and `man sudoers`).

On Windows operating systems, when executing the `mdce start` script, the user defined by MDCEUSER must be listed among those who can log on as a service. To see the list of valid users, select the Windows menu **Start > Settings > Control Panel**. Double-click Administrative Tools, then Local Security Policy. In the tree, select User Rights Assignment, then in the right pane, double-click Log on as a service. This dialog box must list the user defined for MDCEUSER in your `mdce_def.bat` file. If not, you can add the user to this dialog box according to the instructions in the `mdce_def.bat` file, or when running `mdce start`, you can use another `mdce_def.bat` file that specifies a listed user.

Overriding the Script Defaults

Specifying an Alternative Defaults File

The default parameters used by the mdce service, job managers, and workers are defined in the file:

- `matlabroot\toolbox\distcomp\bin\mdce_def.bat` (on Windows operating systems)
- `matlabroot/toolbox/distcomp/bin/mdce_def.sh` (on UNIX or Macintosh operating systems)

Before installing and starting the mdce service, you can edit this file to set the default parameters with values you require.

Alternatively, you can make a copy of this file, modify the copy, and specify that this copy be used for the default parameters.

On UNIX or Macintosh operating systems, enter the command

```
mdce start -mdcedef my_mdce_def.sh
```

On Windows operating systems, enter the command

```
mdce install -mdcedef my_mdce_def.bat  
mdce start -mdcedef my_mdce_def.bat
```

If you specify a new `mdce_def` file instead of the default file for the service on one computer, the new file is not automatically used by the mdce service on other computers. If you want to use the same alternative file for all your mdce services, you must specify it for each mdce service you install or start.

For more information, see “Defining the Script Defaults” on page 2-13.

Note The startup script flags take precedence over the settings in the `mdce_def` file.

Starting in a Clean State

When a job manager or worker starts up, it normally resumes its session from the past. This way, a job queue is not destroyed or lost if the job manager machine crashes or if the job manager is inadvertently shut down. To start up a job manager or worker from a clean state, with all history deleted, use the `-clean` flag on the `start` command:

```
startjobmanager -clean -name MyJobManager  
startworker -clean -jobmanager MyJobManager
```


Accessing Service Record Files

In this section...

“Locating Log Files” on page 2-17

“Locating Checkpoint Directories” on page 2-18

The MATLAB Distributed Computing Server services generate various record files in the normal course of their operations. The mdce service, job manager, and worker sessions all generate such files. This section describes the types of information stored by the services.

Locating Log Files

Log files for each service contain entries for the service’s operations. These might be of particular interest to the network administrator in cases when problems arise.

Operating System	File Location
Windows	<p>The default location of the log files is <TEMP>\MDCE\Log, where <TEMP> is the value of the system TEMP variable. For example, if TEMP is set to C:\TEMP, the log files are placed in C:\TEMP\MDCE\Log.</p> <p>You can set alternative locations for the log files by modifying the LOGBASE setting in the mdce_def.bat file before starting the mdce service.</p>
UNIX and Macintosh	<p>The default location of the log files is /var/log/mdce/.</p> <p>You can set alternative locations for the log files by modifying the LOGBASE setting in the mdce_def.sh file before starting the mdce service.</p>

Locating Checkpoint Directories

Checkpoint directories contain information related to persistence data, which the server services use to create continuity from one instance of a session to another. For example, if you stop and restart a job manager, the new session continues the old session, using all the same data.

A primary feature offered by the checkpoint directories is in crash recovery. This allows server services to automatically resume their sessions after a system goes down and comes back up, minimizing the loss of data. However, if a MATLAB worker goes down during the evaluation of a task, that task is neither reevaluated nor reassigned to another worker. In this case, a finished job may not have a complete set of output data, because data from any unfinished tasks might be missing.

Note If a job manager crashes and restarts, its workers can take up to 2 minutes to reregister with it.

Platform	File Location
Windows	<p>The default location of the checkpoint directories is <TEMP>\MDCE\Checkpoint, where <TEMP> is the value of the system TEMP variable. For example, if TEMP is set to C:\TEMP, the checkpoint directories are placed in C:\TEMP\MDCE\Checkpoint.</p> <p>You can set alternative locations for the checkpoint directories by modifying the CHECKPOINTBASE setting in the mdce_def.bat file before starting the mdce service.</p>
UNIX and Macintosh	<p>The checkpoint directories are placed by default in /var/lib/mdce/.</p> <p>You can set alternative locations for the checkpoint directories by modifying the CHECKPOINTBASE setting in the mdce_def.sh file before starting the mdce service.</p>

Troubleshooting

In this section...

“License Errors” on page 2-19

“Verifying Multicast Communications” on page 2-21

“Memory Errors on UNIX Operating Systems” on page 2-22

“Running Server Processes from a Windows Network Installation” on page 2-22

“Required Ports” on page 2-23

“Ephemeral TCP Ports with Job Manager” on page 2-24

This section offers advice on solving problems you might encounter with MATLAB Distributed Computing Server software.

License Errors

When starting a MATLAB worker, a licensing problem might result in the message

```
License checkout failed. No such FEATURE exists.  
License Manager Error -5
```

There are many reasons why you might receive this error:

- This message usually indicates that you are trying to use a product for which you are not licensed. Look at your `license.dat` file located within your MATLAB installation to see if you are licensed to use this product.
- If you are licensed for this product, this error may be the result of having extra carriage returns or tabs in your license file. To avoid this, ensure that each line begins with either `#`, `SERVER`, `DAEMON`, or `INCREMENT`.

After fixing your `license.dat` file, restart your license manager and MATLAB should work properly.

- This error may also be the result of an incorrect system date. If your system date is before the date that your license was made, you will get this error.

- If you receive this error when starting a worker with MATLAB Distributed Computing Server software:
 - You may be calling the `startworker` command from an installation that does not have access to a worker license. For example, starting a worker from a client installation of the Parallel Computing Toolbox product causes the following error:

```
The mdce service on the host hostname
returned the following error:
```

```
Problem starting the MATLAB worker.
```

```
The cause of this problem is:
```

```
=====
Most likely, the MATLAB worker failed to start due to a
licensing problem, or MATLAB crashed during startup. Check
the worker log file
/tmp/mdce_user/node_node_worker_05-11-01_16-52-03_953.log
for more detailed information. The mdce log file
/tmp/mdce_user/mdce-service.log
may also contain some additional information.
=====
```

In the worker log files, you see the following information:

```
License checkout failed.
License Manager Error -15
MATLAB is unable to connect to the license server.
Check that the license manager has been started, and that the
MATLAB client machine can communicate with the license server.
```

```
Troubleshoot this issue by visiting:
http://www.mathworks.com/support/lme/R2009a/15
```

```
Diagnostic Information:
Feature: MATLAB_Distrib_Comp_Engine
License path: /apps/matlab/etc/license.dat
FLEXnet Licensing error: -15,570. System Error: 115
```

- If you installed only the Parallel Computing Toolbox product, and you are attempting to run a worker on the same machine, you will receive this error because the MATLAB Distributed Computing Server product is not installed, and therefore the worker cannot obtain a license.

Verifying Multicast Communications

Note Although Version 4 of the parallel computing products continues to support multicast communications between its processes, multicast is not recommended and might not be supported in future releases.

Multicast, unlike TCP/IP or UDP, is a subscription-based protocol where a number of machines on a network indicate to the network their interest in particular packets originating somewhere on that network. By contrast, both UDP and TCP packets are always bound for a single machine, usually indicated by its IP address.

The main tools for investigating this type of packet are:

- `tcpdump` for UNIX operating systems
- `wincap` and `ethereal` for Microsoft Windows operating systems
- A Java™ class included with Version 3 of the parallel computing products.

The Java class is called `com.mathworks.toolbox.distcomp.test.MulticastTester`. Both its static `main` method and its constructor take two input arguments: the multicast group to join and the port number to use.

This Java class has a number of simple methods to attempt to join a specified multicast group. Once the class has successfully joined the group, it has methods to send messages to the group, listen for messages from the group, and display what it receives. The class can be used both inside MATLAB and from a call to Java software.

Inside MATLAB, use the class as follows:

```
m = com.mathworks.toolbox.distcomp.test.MulticastTester('239.1.1.1', 9999);  
m.startSendingThread;
```

```
m.startListeningThread;  
0 : host1name : 0  
1 : host2name : 0
```

From a shell prompt, type (assuming that java is on your path)

```
java -cp distcomp.jar com.mathworks.toolbox.distcomp.test.MulticastTester  
0 : host1name : 0  
1 : host2name : 0
```

Memory Errors on UNIX Operating Systems

If the number of threads created by the server services on a machine running a UNIX operating system exceeds the limitation set by the `maxproc` value, the services fail and generate an out-of-memory error. Check your `maxproc` value on a UNIX operating system with the `limit` command. (Different versions of UNIX software might have different names for this property instead of `maxproc`, such as `descriptors` on Solaris™ operating systems.)

Running Server Processes from a Windows Network Installation

Many networks are configured not to allow `LocalSystem` to have access to UNC or mapped network shares. In this case, run the `mdce` process under a different user with rights to log on as a service. See “Setting the User” on page 2-14.

Required Ports

Using a Job Manager

BASE_PORT. The `mdce_def` file specifies and describes the ports required by the job manager and all workers. See the following file in the MATLAB installation used for each cluster process:

- `matlabroot/toolbox/distcomp/bin/mdce_def.sh` (on UNIX operating systems)
- `matlabroot\toolbox\distcomp\bin\mdce_def.bat` (on Windows operating systems)

Parallel Jobs. On worker machines running a UNIX operating system, the number of ports required by MPICH for the running of parallel jobs ranges from `BASE_PORT + 1000` to `BASE_PORT + 2000`.

Using a Third-Party Scheduler

Before the worker processes start, you can control the range of ports used by the workers for parallel jobs by defining the environment variable `MPICH_PORT_RANGE` with the value `minport:maxport`.

Client Ports

With the `pctconfig` function, you specify the ports used by the client. If the default ports cannot be used, this function allows you to configure ports separately for communication with the job manager and communication with `pmode` or a MATLAB pool.

Ephemeral TCP Ports with Job Manager

If you use the job manager on a cluster of nodes running Windows operating systems, you must make sure that a large number of ephemeral TCP ports are available on the job manager machine. By default, the maximum valid ephemeral TCP port number on a Windows operating system is 5000, but transfers of large data sets might fail if this setting is not increased. In particular, if your cluster has 32 or more workers, you should increase the maximum valid ephemeral TCP port number using the following procedure:

- 1 Start the Registry Editor.
- 2 Locate the following subkey in the registry, and click **Parameters**:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
```
- 3 On the Registry Editor window, select **Edit > New > DWORD Value**.
- 4 In the list of entries on the right, change the new value name to **MaxUserPort** and press **Enter**.
- 5 Right-click on the **MaxUserPort** entry name and select **Modify**.
- 6 In the Edit DWORD Value dialog, enter **65534** in the **Value data** field. Select **Decimal** for the **Base** value. Click **OK**.

This parameter controls the maximum port number that is used when a program requests any available user port from the system. Typically, ephemeral (short-lived) ports are allocated between the values of 1024 and 5000 inclusive. This action allows allocation for port numbers up to 65534.

- 7 Quit the Registry Editor.
- 8 Reboot your machine.

Admin Center

- “Starting Admin Center” on page 3-2
- “Setting Up Resources” on page 3-3
- “Testing Connectivity” on page 3-9
- “Saving and Loading Sessions” on page 3-13
- “Preparing for User Configurations” on page 3-14

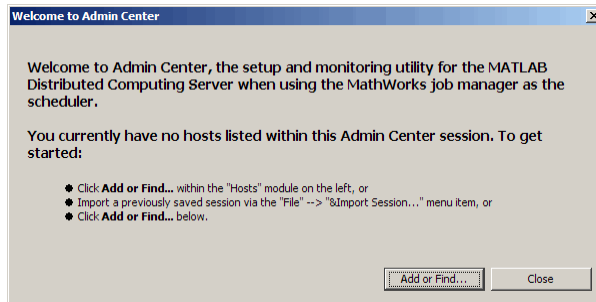
Starting Admin Center

Admin Center is a graphical user interface that lets you control and verify MATLAB Distributed Computing Server resources if you are using a job manager as your scheduler.

You must start Admin Center outside a MATLAB session by executing the following:

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

The first time you start Admin Center, you see a welcome dialog box.



A new session has no hosts listed, so the usual first step is to identify the hosts you want to include in your listing. To do this, click **Add or Find**. Further information continues in the next section.

If you start Admin Center again on the same host, your previous session for that machine is loaded; and unless the update rate is set to *never*, Admin Center performs an update immediately for the listed hosts and processes. To clear this information and start a new session, select the pull-down **File > New Session**.

Setting Up Resources

In this section...

- “Adding Hosts” on page 3-3
- “Starting a Job Manager” on page 3-4
- “Starting Workers” on page 3-5
- “Stopping, Destroying, Resuming, Restarting Processes” on page 3-7
- “Moving a Worker” on page 3-8
- “Updating the Display” on page 3-8

Adding Hosts

To specify the hosts you want displayed in Admin Center, click **Add or Find** in the Welcome dialog box, or if this is not a new session, click **Add or Find** in the Hosts module.

In the Add or Find Hosts dialog box, identify the hosts you want to add to the listing by one of the following methods:

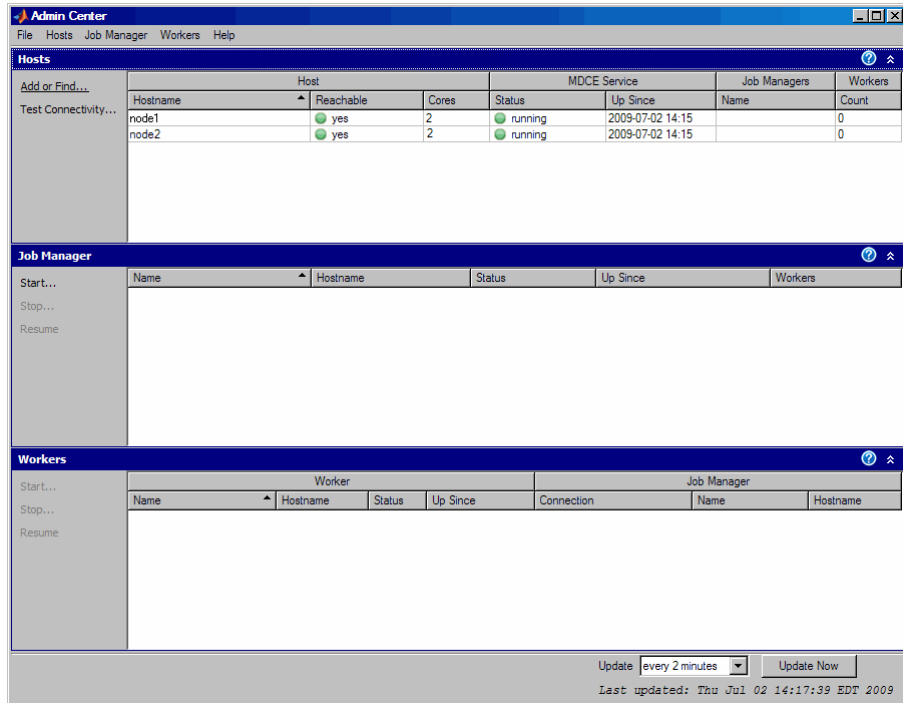
- Select **Enter Hostnames** and provide short host names, fully qualified domain names, or individual IP addresses for the hosts, or
- Select **Enter IP Range** and provide the range of IP addresses for your hosts.

Note While you can add any hosts to Admin Center, a host must be running the mdce service if a job manager or worker is to run on that host. See the installation instructions available at:

http://www.mathworks.com/support/product/DM/installation/ver_current/

If one of the hosts you have specified is running a job manager, Admin Center will automatically find and list all the hosts running workers registered with that job manager. Similarly, if you specify a host that is running a worker,

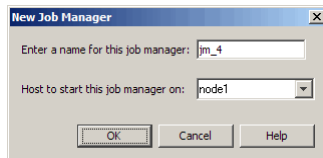
Admin Center will find and list the host running that worker's job manager, and also all hosts running other workers under that job manager.



Starting a Job Manager

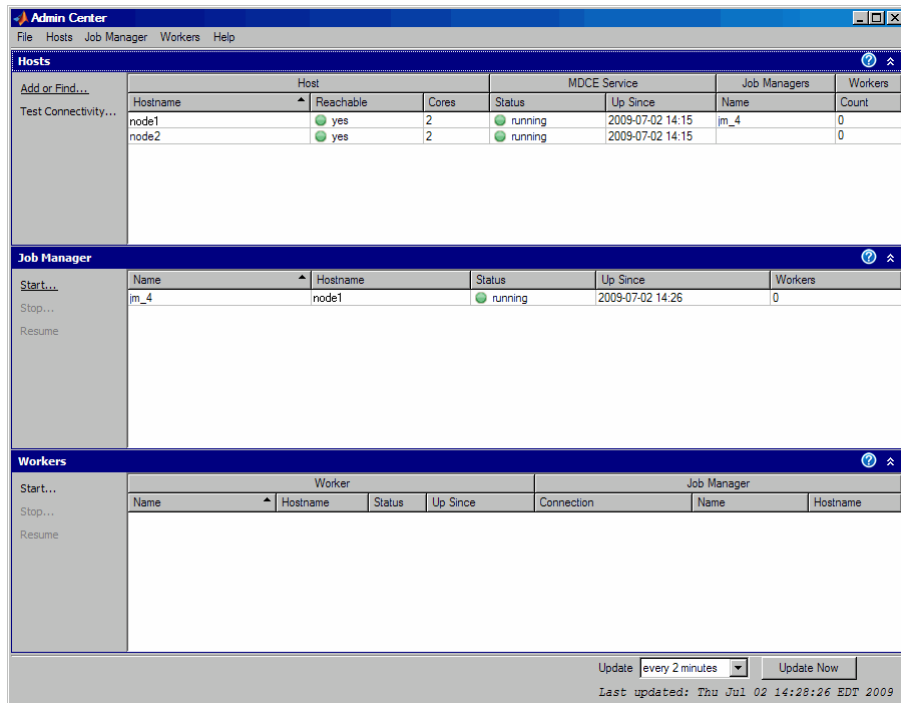
To start a job manager, click **Start** in the Job Manager module.

In the New Job Manager dialog box, provide a name for the job manager, and select a host to run it on.



Alternative methods for starting a job manager include selecting the pull-down **Job Manager > Start**, or right-clicking a listed host and selecting, **Start Job Manager**.

With a job manager running on your cluster, Admin Center might look like the following figure, with the job manager listed in the Job Manager module, as well as being listed by name in the Hosts module in the line for the host on which it is running.

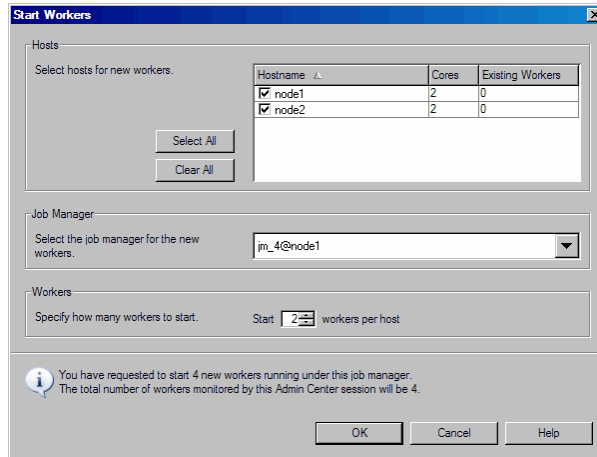


Starting Workers

To start MATLAB workers, click **Start** in the Workers module.

In the Start Workers dialog box, specify the numbers of workers to start on each host, and select the hosts to run them. From the list, select the job manager for these workers. Click **OK** to start the workers. Admin center

automatically provides names for the workers, based on the hosts running them.



Alternative methods for starting workers include selecting the pull-down **Workers > Start**, or right-clicking a listed host or job manager and selecting, **Start Workers**.

With workers running on your cluster, Admin Center might look like the following figure, which shows the workers listed in the Workers module. Also, the number of workers running under the job manager is listed in the Job Manager module, and the number of workers for each job manager is listed in the Hosts module.

The screenshot shows the Admin Center interface with three main sections: Hosts, Job Manager, and Workers. Each section has a table of data and a sidebar with actions like 'Start...', 'Stop...', and 'Resume'.

Hosts Section:

Host		MDCE Service			Job Managers		Workers
Hostname	Reachable	Cores	Status	Up Since	Name	Count	
node1	yes	2	running	2009-07-02 14:15	jm_4	2	
node2	yes	2	running	2009-07-02 15:07			

Job Manager Section:

Name	Hostname	Status	Up Since	Workers
jm_4	node1	running	2009-07-02 14:26	4

Workers Section:

Worker				Job Manager		
Name	Hostname	Status	Up Since	Connection	Name	Hostname
node1_worker01	node1	idle	2009-07-02 15:16	connected	jm_4	node1
node1_worker02	node1	idle	2009-07-02 15:16	connected	jm_4	node1
node2_worker01	node2	idle	2009-07-02 15:15	connected	jm_4	node1
node2_worker02	node2	idle	2009-07-02 15:15	connected	jm_4	node1

At the bottom right, there is an 'Update' button set to 'every 2 minutes' and an 'Update Now' button. The status bar shows 'Last updated: Thu Jul 02 15:26:55 EDT 2009'.

To get more information on any host, job manager, or worker listed in Admin Center, right-click its name in the display and select **Properties**. Alternatively, you can find the **Properties** option under the **Hosts**, **Job Manager**, and **Workers** drop-down menus.

Stopping, Destroying, Resuming, Restarting Processes

You can **Stop** or **Destroy** job managers and workers. The primary difference is that stopping a process shuts it down but retains its data; destroying a process shuts it down and clears its data. Use **Resume** to have a process continue with its existing data. When you use **Restart**, a dialog box requires you to confirm your intention of starting a new process while keeping or discarding data.

Moving a Worker

To move a worker from one host to another, you must completely shut it down, then start a new worker on the desired host:

- 1 Right-click the worker in the Workers module list.
- 2 Select **Destroy**. This shuts down the worker process and removes all its data.
- 3 If the old worker host is not running any other MDCS processes (mdce service, job manager, or workers), you might want to remove it from the Admin Center listing.
- 4 If necessary, add the new host to the Admin Center host listing.
- 5 In the Workers module, click **Start**. Select the desired host in the Start Workers dialog box, along with the appropriate number and job manager name.

Use a similar process to move a job manager from one host to another. Note, however, that all workers registered with the job manager must be destroyed and started again, registering them with the new instance of the job manager.

Updating the Display

Admin Center updates its data automatically at regular intervals. To set the update rate, select an option from the **Update** list. Click **Update Now** to immediately update the display data.

Testing Connectivity

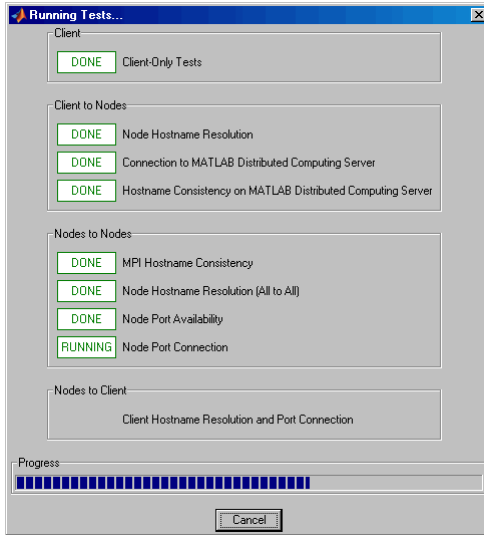
Admin Center lets you test communications between your job manager node, worker nodes, and the node where Admin Center is running.

The tests are divided into four categories:

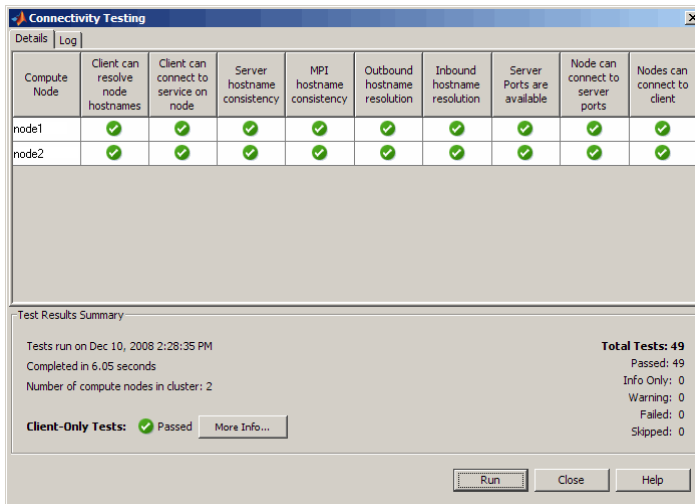
- **Client** — Verifies that the node running Admin Center is properly configured so that further cluster testing can proceed.
- **Client to Nodes** — Verifies that the node running Admin Center can identify and communicate with the other nodes in the cluster.
- **Nodes to Nodes** — Verifies that the other nodes in the cluster can identify each other, and that each node allows its mdce service to communicate with the mdce service on the other cluster nodes.
- **Nodes to Client** — Verifies that other cluster nodes can identify and communicate with the node running Admin Center.

First click **Test Connectivity** to open the Connectivity Testing dialog box. By default, the dialog box displays the results of the last test. To run new tests and update the display, click **Run**.






During test execution, Admin Center displays this progress dialog box.



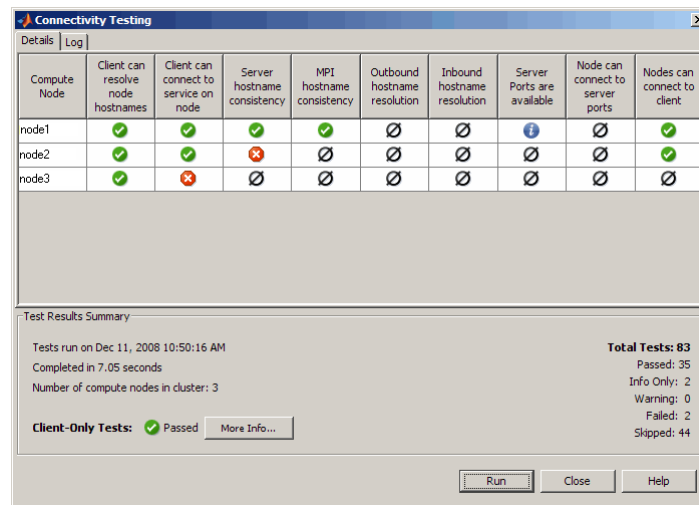
When the tests are complete, the Running Tests dialog box automatically closes, and Admin Center displays the test results in the Connectivity Testing dialog box.



The possible test result symbols are described in the following table.

Test Result	Description
	Test passed.
	Test passed, extra information is available.
	Test passed, but generated a warning.
	Test failed.
	Test was skipped, possibly because prerequisite tests did not pass.

Test that include failures or other results might look like the following figure.



Double-click any of the symbols in the test results to drill down for more detail. Use the **Log** tab to see the raw data from the tests.

The results of the tests that run on only the client are displayed in the lower-left corner of the dialog box. To drill into client-only test results, click **More Info**.

Saving and Loading Sessions

By default, Admin Center saves the cluster definition, process status, and test results, so the next time the same user runs Admin Center on the same machine, that information is available and displayed by default. You can export session data so that a different user or a different host can access it, by selecting the pull-down **File > Export**. Browse to the location where you want to store the session data and provide a name for the file. Admin Center applies the extension `.mdcs` to the file name.

You can import that saved session data into a subsequent session of Admin Center by selecting the pull-down **File > Import**. The imported data includes cluster definition and test results.

When identifying the file for importing in the Import Session dialog box, there is a **Disable updates** check box. Checking this box lets you import a session that does not automatically update, so that you can statically examine a cluster setup for evaluation or diagnostic purposes. Otherwise, unless the update rate is set to `never`, Admin Center performs an update immediately after starting or loading a session.

Preparing for User Configurations

Admin Center does not create user configurations, but the information displayed in Admin Center is of vital importance when you create your parallel configuration — information such as job manager name, job manager host, and number of workers. For more information about creating and using configurations, see “Programming with User Configurations” in the Parallel Computing Toolbox documentation.

Control Script Reference

mdce Process Control (p. 4-2)

Job Manager Control (p. 4-2)

Worker Control (p. 4-2)

Control mdce service

Control job manager

Control MATLAB workers

mdce Process Control

mdce

Install, start, stop, or uninstall mdce service

nodestatus

Status of mdce processes running on node

Job Manager Control

startjobmanager

Start job manager process

stopjobmanager

Stop job manager process

Worker Control

startworker

Start MATLAB worker session

stopworker

Stop MATLAB worker session

Control Scripts — Alphabetical List

mdce

Purpose Install, start, stop, or uninstall mdce service

Syntax

```
mdce install
mdce uninstall
mdce start
mdce stop
mdce console
mdce restart
mdce ... -mdcedef <mdce_defaults_file>
mdce ... -clean
mdce status
```

Description The mdce service ensures that all other processes are running and that it is possible to communicate with them. Once the mdce service is running, you can use the `nodestatus` command to obtain information about the mdce service and all the processes it maintains.

`mdce install` installs the mdce service in the Microsoft Windows Service Control Manager. This causes the service to automatically start when the Windows operating system boots up. The service must be installed before it is started.

`mdce uninstall` uninstalls the mdce service from the Windows Service Control Manager. Note that if you wish to install mdce service as a different user, you must first uninstall the service and then reinstall as the new user.

`mdce start` starts the mdce service. This creates the required logging and checkpointing directories, and then starts the service as specified in the mdce defaults file.

`mdce stop` stops running the mdce service. This automatically stops all job managers and workers on the computer, but leaves their checkpoint information intact so that they will start again when the mdce service is started again.

`mdce console` starts the mdce service as a process in the current terminal or command window rather than as a service running in the background.

`mdce restart` performs the equivalent of `mdce stop` followed by `mdce start`. This command is available only on UNIX and Macintosh operating systems.

`mdce ... -mdcedef <mdce_defaults_file>` uses the specified alternative `mdce` defaults file instead of the one found in *matlabroot/toolbox/distcomp/bin*.

`mdce ... -clean` performs a complete cleanup of all service checkpoint and log files before installing or starting the service, or after stopping or uninstalling it. This deletes all information about any job managers or workers this service has ever maintained.

`mdce status` reports the status of the `mdce` service, indicating whether it is running and with what PID. Use `nodestatus` to obtain more detailed information about the `mdce` service. The `mdce status` command is available only on UNIX and Macintosh operating systems.

See Also

`nodestatus`, `startjobmanager`, `startworker`, `stopjobmanager`, `stopworker`

nodestatus

Purpose Status of mdce processes running on node

Syntax
nodestatus
nodestatus *-flags*

Description nodestatus displays the status of the mdce service and the processes which it maintains. The mdce service must already be running on the specified computer.

nodestatus *-flags* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
-remotehost <hostname>	Displays the status of the mdce service and the processes it maintains on the specified host. The default value is the local host.
-infolevel <level>	Specifies how much status information to report, using a level of 1-3. 1 means only the basic information, 3 means all information available. The default value is 1.
-baseport <port_number>	Specifies the base port that the mdce service on the remote host is using. You need to specify this only if the value of <code>BASE_PORT</code> in the local <code>mdce_def</code> file does not match the base port being used by the mdce service on the remote host.
-v	Verbose mode displays the progress of the command execution.

Examples

Display basic information about the mdce processes on the local host.

```
nodestatus
```

Display detailed information about the status of the mdce processes on host node27.

```
nodestatus -remotehost node27 -infolevel 2
```

See Also

mdce, startjobmanager, startworker, stopjobmanager, stopworker

startjobmanager

Purpose Start job manager process

Syntax startjobmanager
startjobmanager *-flags*

Description startjobmanager starts a job manager process and the associated job manager lookup process under the mdce service, which maintains them after that. The job manager handles the storage of jobs and the distribution of tasks contained in jobs to MATLAB workers that are registered with it. The mdce service must already be running on the specified computer.

startjobmanager *-flags* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
-name <job_manager_name>	Specifies the name of the job manager. This identifies the job manager to MATLAB worker sessions and MATLAB clients. The default is the value of the DEFAULT_JOB_MANAGER_NAME parameter in the mdce_def file.
-remotehost <hostname>	Specifies the name of the host where you want to start the job manager and the job manager lookup process. If omitted, they are started on the local host.
-clean	Deletes all checkpoint information stored on disk from previous instances of this job manager before starting. This cleans the job manager so that it initializes with no jobs or tasks.

Flag	Operation
-multicast	Overrides the use of unicast to contact the job manager lookup process. It is recommended that you not use -multicast unless you are certain that multicast works on your network. This overrides the setting of JOB_MANAGER_HOST in the mdce_def file on the remote host, which would have the job manager use unicast. If this flag is omitted and JOB_MANAGER_HOST is empty, the job manager uses unicast to contact the job manager lookup process running on the same host.
-baseport <port_number>	Specifies the base port that the mdce service on the remote host is using. You need to specify this only if the value of BASE_PORT in the local mdce_def file does not match the base port being used by the mdce service on the remote host.
-v	Verbose mode displays the progress of the command execution.

Examples

Start the job manager MyJobManager on the local host.

```
startjobmanager -name MyJobManager
```

Start the job manager MyJobManager on the host JMHost.

```
startjobmanager -name MyJobManager -remotehost JMHost
```

See Also

mdce, nodestatus, startworker, stopjobmanager, stopworker

startworker

Purpose Start MATLAB worker session

Syntax startworker
startworker *-flags*

Description startworker starts a MATLAB worker process under the mdce service, which maintains it after that. The worker registers with the specified job manager, from which it will get tasks for evaluation. The mdce service must already be running on the specified computer.

startworker *-flags* accepts the following input flags. Multiple flags can be used together on the same command, except where noted.

Flag	Operation
-name <worker_name>	Specifies the name of the MATLAB worker. The default is the value of the DEFAULT_WORKER_NAME parameter in the mdce_def file.
-remotehost <hostname>	Specifies the name of the computer where you want to start the MATLAB worker. If omitted, the worker is started on the local computer.
-jobmanager <job_manager_name>	Specifies the name of the job manager this MATLAB worker will receive tasks from. The default is the value of the DEFAULT_JOB_MANAGER_NAME parameter in the mdce_def file.

Flag	Operation
-jobmanagerhost <job_manager_hostname>	<p>Specifies the host on which the job manager is running. The worker uses unicast to contact the job manager lookup process on that host to register with the job manager.</p> <p>This overrides the setting of <code>JOB_MANAGER_HOST</code> in the <code>mdce_def</code> file on the worker computer, which would also have the worker use unicast.</p> <p>Cannot be used together with <code>-multicast</code>.</p>
-multicast	<p>If you are certain that multicast works on your network, you can force the worker to use multicast to locate the job manager lookup process by specifying <code>-multicast</code>. Note: If you are using this flag to change the settings of and restart a stopped worker, then you should also use the <code>-clean</code> flag.</p> <p>Cannot be used together with <code>-jobmanagerhost</code>.</p>
-clean	<p>Deletes all checkpoint information associated with this worker name before starting.</p>
-baseport <port_number>	<p>Specifies the base port that the <code>mdce</code> service on the remote host is using. You only need to specify this if the value of <code>BASE_PORT</code> in the local <code>mdce_def</code> file does not match the base port being used by the <code>mdce</code> service on the remote host.</p>
-v	<p>Verbose mode displays the progress of the command execution.</p>

startworker

Examples

Start a worker on the local host, using the default worker name, registering with the job manager MyJobManager on the host JMHost.

```
startworker -jobmanager MyJobManager -jobmanagerhost JMHost
```

Start a worker on the host WorkerHost, using the default worker name, and registering with the job manager MyJobManager on the host JMHost. (The following command should be entered on a single line.)

```
startworker -jobmanager MyJobManager -jobmanagerhost JMHost  
-remotehost WorkerHost
```

Start two workers, named worker1 and worker2, on the host WorkerHost, registering with the job manager MyJobManager that is running on the host JMHost. Note that to start two workers on the same computer, you must give them different names. (Each of the two commands below should be entered on a single line.)

```
startworker -name worker1 -remotehost WorkerHost  
-jobmanager MyJobManager -jobmanagerhost JMHost  
startworker -name worker2 -remotehost WorkerHost  
-jobmanager MyJobManager -jobmanagerhost JMHost
```

See Also

mdce, nodestatus, startjobmanager, stopjobmanager, stopworker

Purpose Stop job manager process

Syntax
stopjobmanager
stopjobmanager *-flags*

Description stopjobmanager stops a job manager that is running under the mdce service.

stopjobmanager *-flags* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
-name <job_manager_name>	Specifies the name of the job manager to stop. The default is the value of DEFAULT_JOB_MANAGER_NAME parameter the mdce_def file.
-remotehost <hostname>	Specifies the name of the host where you want to stop the job manager and the associated job manager lookup process. The default value is the local host.
-clean	Deletes all checkpoint information stored on disk for the current instance of this job manager after stopping it. This cleans the job manager of all its job and task data.

stopjobmanager

Flag	Operation
-baseport <port_number>	Specifies the base port that the mdce service on the remote host is using. You need to specify this only if the value of <code>BASE_PORT</code> in the local <code>mdce_def</code> file does not match the base port being used by the mdce service on the remote host.
-v	Verbose mode displays the progress of the command execution.

Examples

Stop the job manager MyJobManager on the local host.

```
stopjobmanager -name MyJobManager
```

Stop the job manager MyJobManager on the host JMHost.

```
stopjobmanager -name MyJobManager -remotehost JMHost
```

See Also

mdce, nodestatus, startjobmanager, startworker, stopworker

Purpose Stop MATLAB worker session

Syntax
 stopworker
 stopworker *-flags*

Description stopworker stops a MATLAB worker process that is running under the mdce service.

stopworker *-flags* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
-name <worker_name>	Specifies the name of the MATLAB worker to stop. The default is the value of the DEFAULT_WORKER_NAME parameter in the mdce_def file.
-remotehost <hostname>	Specifies the name of the host where you want to stop the MATLAB worker. The default value is the local host.
-clean	Deletes all checkpoint information associated with this worker name after stopping it.

stopworker

Flag	Operation
-baseport <port_number>	Specifies the base port that the mdce service on the remote host is using. You need to specify this only if the value of <code>BASE_PORT</code> in the local <code>mdce_def</code> file does not match the base port being used by the mdce service on the remote host.
-v	Verbose mode displays the progress of the command execution.

Examples

Stop the worker with the default name on the local host.

```
stopworker
```

Stop the worker with the default name, running on the computer WorkerHost.

```
stopworker -remotehost WorkerHost
```

Stop the workers named worker1 and worker2, running on the computer WorkerHost.

```
stopworker -name worker1 -remotehost WorkerHost  
stopworker -name worker2 -remotehost WorkerHost
```

See Also

mdce, nodestatus, startjobmanager, startworker, stopjobmanager

CHECKPOINTBASE

The name of the parameter in the `mdce_def` file that defines the location of the job manager and worker checkpoint directories.

checkpoint directory

Location where job manager checkpoint information and worker checkpoint information is stored.

client

The MATLAB session that defines and submits the job. This is the MATLAB session in which the programmer usually develops and prototypes applications. Also known as the MATLAB client.

client computer

The computer running the MATLAB client.

cluster

A collection of computers that are connected via a network and intended for a common purpose.

coarse-grained application

An application for which run time is significantly greater than the communication time needed to start and stop the program. Coarse-grained distributed applications are also called embarrassingly parallel applications.

codistributed array

An array partitioned into segments, with each segment residing in the workspace of a different lab.

Composite

An object in a MATLAB client session that provides access to data values stored on the labs in a MATLAB pool, such as the values of variables that are assigned inside an `spmd` statement.

computer

A system with one or more processors.

distributed application

The same application that runs independently on several nodes, possibly with different input parameters. There is no communication, shared data, or synchronization points between the nodes. Distributed applications can be either coarse-grained or fine-grained.

distributed computing

Computing with distributed applications, running the application on several nodes simultaneously.

distributed computing demos

Demonstration programs that use Parallel Computing Toolbox software, as opposed to sequential demos.

DNS

Domain Name System. A system that translates Internet domain names into IP addresses.

dynamic licensing

The ability of a MATLAB worker or lab to employ all the functionality you are licensed for in the MATLAB client, while checking out only a server product license. When a job is created in the MATLAB client with Parallel Computing Toolbox software, the products for which the client is licensed will be available for all workers or labs that evaluate tasks for that job. This allows you to run any code on the cluster for which you are licensed on your MATLAB client, without requiring extra licenses for the worker beyond that for the MATLAB Distributed Computing Server product. For a list of products that are not eligible for use with Parallel Computing Toolbox software, see http://www.mathworks.com/products/ineligible_programs/.

fine-grained application

An application for which run time is significantly less than the communication time needed to start and stop the program. Compare to coarse-grained applications.

head node

Usually, the node of the cluster designated for running the job manager and license manager. It is often useful to run all the nonworker-related processes on a single machine.

heterogeneous cluster

A cluster that is not homogeneous.

homogeneous cluster

A cluster of identical machines, in terms of both hardware and software.

job

The complete large-scale operation to perform in MATLAB, composed of a set of tasks.

job manager

The MathWorks process that queues jobs and assigns tasks to workers. A third-party process that performs this function is called a scheduler. The general term “scheduler” can also refer to a job manager.

job manager checkpoint information

Snapshot of information necessary for the job manager to recover from a system crash or reboot.

job manager database

The database that the job manager uses to store the information about its jobs and tasks.

job manager lookup process

The process that allows clients, workers, and job managers to find each other. It starts automatically when the job manager starts.

lab

When workers start, they work independently by default. They can then connect to each other and work together as peers, and are then referred to as labs.

LOGDIR

The name of the parameter in the `mdce_def` file that defines the directory where logs are stored.

MathWorks job manager

See job manager.

MATLAB client

See client.

MATLAB pool

A collection of labs that are reserved by the client for execution of parfor-loops or spmd statements. See also lab.

MATLAB worker

See worker.

mdce

The service that has to run on all machines before they can run a job manager or worker. This is the server foundation process, making sure that the job manager and worker processes that it controls are always running.

Note that the program and service name is all lowercase letters.

mdce_def file

The file that defines all the defaults for the mdce processes by allowing you to set preferences or definitions in the form of parameter values.

MPI

Message Passing Interface, the means by which labs communicate with each other while running tasks in the same job.

node

A computer that is part of a cluster.

parallel application

The same application that runs on several labs simultaneously, with communication, shared data, or synchronization points between the labs.

private array

An array which resides in the workspaces of one or more, but perhaps not all labs. There might or might not be a relationship between the values of these arrays among the labs.

random port

A random unprivileged TCP port, i.e., a random TCP port above 1024.

register a worker

The action that happens when both worker and job manager are started and the worker contacts job manager.

replicated array

An array which resides in the workspaces of all labs, and whose size and content are identical on all labs.

scheduler

The process, either third-party or the MathWorks job manager, that queues jobs and assigns tasks to workers.

spmd (single program multiple data)

A block of code that executes simultaneously on multiple labs in a MATLAB pool. Each lab can operate on a different data set or different portion of distributed data, and can communicate with other participating labs while performing the parallel computations.

task

One segment of a job to be evaluated by a worker.

variant array

An array which resides in the workspaces of all labs, but whose content differs on these labs.

worker

The MATLAB process that performs the task computations. Also known as the MATLAB worker or worker process.

worker checkpoint information

Files required by the worker during the execution of tasks.

A

- administration
 - network 2-1

C

- checkpoint directory
 - definition Glossary-1
 - locating 2-18
- CHECKPOINTBASE
 - definition Glossary-1
- clean state
 - starting services 2-16
- client
 - definition Glossary-1
 - process 1-4
- client computer
 - definition Glossary-1
- cluster
 - definition Glossary-1
- coarse-grained application
 - definition Glossary-1
- Composite
 - definition Glossary-1
- computer
 - definition Glossary-1
- configuring MATLAB® Distributed Computing Server™ 2-5
- control scripts
 - customizing 2-13
 - defaults 2-13
 - mdce 5-2
 - nodestatus 5-4
 - startjobmanager 5-6
 - startworker 5-8
 - stopjobmanager 5-11
 - stopworker 5-13

D

- distributed application
 - definition Glossary-2
- distributed computing
 - definition Glossary-2
- distributed computing demos
 - definition Glossary-2
- DNS
 - definition Glossary-2
- dynamic licensing
 - definition Glossary-2

F

- fine-grained application
 - definition Glossary-2

H

- head node
 - definition Glossary-2
- heterogeneous cluster
 - definition Glossary-3
 - support 1-7
- homogeneous cluster
 - definition Glossary-3

I

- installing MATLAB® Distributed Computing Server™ 2-5

J

- job
 - definition Glossary-3
- job manager
 - checkpoint information
 - definition Glossary-3
 - database
 - definition Glossary-3

- definition Glossary-3
- logs 2-17
- lookup process
 - definition Glossary-3
- multiple on one machine 2-14
- process 1-4
- stopping
 - on UNIX or Macintosh 2-9
 - on Windows 2-11
- versus third-party scheduler 1-6

L

- lab
 - definition Glossary-3
- log files
 - locating 2-17
- LOGDIR
 - definition Glossary-3

M

- MathWorks job manager. *See* job manager
- MATLAB client
 - definition Glossary-4
- MATLAB pool
 - definition Glossary-4
- MATLAB worker
 - definition Glossary-4
- mdce (service)
 - definition Glossary-4
- mdce control script 5-2
- mdce_def file
 - definition Glossary-4
- MPI
 - definition Glossary-4

N

- network
 - administration 2-1

- layout 2-2
- preparation 2-2
- requirements 2-3
- security 2-4

- node
 - definition Glossary-4
- nodestatus control script 5-4

P

- parallel application
 - definition Glossary-4
- parallel computing products
 - server 1-4
 - toolbox 1-4
 - version 1-3
- Parallel Computing Toolbox
 - using 1-8
- platforms
 - supported 1-7

R

- random port
 - definition Glossary-5
- register a worker
 - definition Glossary-5
- requirements 2-3

S

- scheduler
 - definition Glossary-5
 - third-party 1-6
- security 2-4
- smpd
 - definition Glossary-5
- startjobmanager control script 5-6
- startworker control script 5-8
- stopjobmanager control script 5-11
- stopworker control script 5-13

T

task

definition Glossary-5

third-party scheduler 1-6

versus job manager 1-6

troubleshooting

license errors 2-19

memory errors 2-22

verifying multicast 2-21

Windows network installation 2-22

U

user

setting 2-14

W

worker

definition Glossary-5

process 1-4

worker checkpoint information

definition Glossary-5

workers

logs 2-17

stopping

on UNIX or Macintosh 2-9

on Windows 2-11