mpiFileUtils Documentation

Release 0.9

HPC

Jan 31, 2019

Contents

CHAPTER 1

Overview

mpiFileUtils provides both a library called libmfu and a suite of MPI-based tools to manage large datasets, which may vary from large directory trees to large files. High-performance computing users often generate large datasets with parallel applications that run with many processes (millions in some cases). However those users are then stuck with single-process tools like cp and rm to manage their datasets. This suite provides MPI-based tools to handle typical jobs like copy, remove, and compare for such datasets, providing speedups of up to 50x. It also provides a library that simplifies the creation of new tools or can be used in applications

CHAPTER 2

Utilities

The tools in mpiFileUtils are actually MPI applications. They must be launched as MPI applications, e.g., within a compute allocation on a cluster using mpirun. The tools do not currently checkpoint, so one must be careful that an invocation of the tool has sufficient time to complete before it is killed. Example usage of each tool is provided below.

- dbcast - Broadcast files to compute nodes.
- dchmod - Change owner, group, and permissions on files.
- dcmp - Compare files.
- dcp - Copy files.
- ddup - Find duplicate files.
- dfilemaker - Generate random files.
- dreln - Relink symlinks.
- drm - Remove files.
- dstripe - Restripe files.
- dsync - Synchronize files
- dwalk - List files.

CHAPTER 3

Experimental Utilities

Experimental utilities are under active development. They are not considered to be production worthy, but they are available in the distribution for those interested in developing them further or to provide additional examples. To build the experimental utilities, turn on the CMake option.

```
$ cmake -DENABLE_EXPERIMENTAL=ON ...
```

- dbz2 - Compress a file with bz2.
- dfind - Search for files in parallel.
- dgrep - Run grep on files in parallel.
- dparallel - Perform commands in parallel. experimental/dparallel.1
- dsh - List and remove files with interactive commands.
- dtar - Create file tape archives.

CHAPTER 4

User Guide

4.1 Build

mpiFileUtils depends on several libraries. mpiFileUtils is available in Spack, which simplifies the install to just:

```
$ spack install mpifileutils
```

or to enable all features:

```
$ spack install mpifileutils +lustre +experimental
```

To build from a release tarball, use CMake. Note that this requires the manual installation of the dependencies. Assuming the dependencies have been placed in an *install* directory the build commands are thus:

```
$ git clone https://github.com/hpc/mpifileutils
$ mkdir build install
$ # build DTCMP and other dependencies
$ cd build
$ cmake ../mpifileutils -DWITH_DTCMP_PREFIX=../install -DWITH_LibCircle_PREFIX=../
↪install -DCMAKE_INSTALL_PREFIX=../install
```

One can also use spack to create an environment and view with the provided *spack.yaml* file. First, make sure that you've set up spack in your shell (see these instructions). Next, be sure that your *~/.spack/packages.yaml* is configured to ensure that spack can detect system-provided packages.

From the root directory of mpifileutils, run the command *spack find* to determine which packages spack will install. Next, run *spack concretize* to build have spack perform dependency analysis. Finally, run *spack install* to build the dependencies.

There are two ways to tell CMake about the dependencies. First, you can use *spack load [depname]* to put the installed dependency into your environment paths. Then, at configure time, CMake will automatically detect the location of these dependencies. Thus, the commands to build become:

```
$ git clone https://github.com/hpc/mpifileutils
$ mkdir build install
$ cd mpifileutils
$ spack install
$ spack load dtcmp
$ spack load libcircle
$ spack load libarchive
$ cd ../build
$ cmake ../mpifileutils
```

The other way to use spack is to create a "view" to the installed dependencies. Details on this are coming soon.

4.2 Project Design Principles

The following principles drive design decisions in the project.

4.2.1 Scale

The library and tools should be designed such that running with more processes increases performance, provided there are sufficient data and parallelism available in the underlying file systems. The design of the tool should not impose performance scalability bottlenecks.

4.2.2 Performance

While it is tempting to mimic the interface, behavior, and file formats of familiar tools like cp, rm, and tar, when forced with a choice between compatibility and performance, mpiFileUtils chooses performance. For example, if an archive file format requires serialization that inhibits parallel performance, mpiFileUtils will opt to define a new file format that enables parallelism rather than being constrained to existing formats. Similarly, options in the tool command line interface may have different semantics from familiar tools in cases where performance is improved. Thus, one should be careful to learn the options of each tool.

4.2.3 Portability

The tools are intended to support common file systems used in HPC centers, like Lustre, GPFS, and NFS. Additionally, methods in the library should be portable and efficient across multiple file systems. Tool and library users can rely on mpiFileUtils to provide portable and performant implementations.

4.2.4 Composability

While the tools do not support chaining with Unix pipes, they do support interoperability through input and output files. One tool may process a dataset and generate an output file that another tool can read as input, e.g., to walk a directory tree with one tool, filter the list of file names with another, and perhaps delete a subset of matching files with a third. Additionally, when logic is deemed to be useful across multiple tools or is anticipated to be useful in future tools or applications, it should be provided in the common library.

4.3 libmfu

Functionality that is common to multiple tools is moved to the common library, libmfu. This goal of this library is to make it easy to develop new tools and to provide consistent behavior across tools in the suite. The library can also be useful to end applications, e.g., to efficiently create or remove a large directory tree in a portable way across different parallel file systems.

4.3.1 libmfu: the mpiFileUtils common library

The mpiFileUtils common library defines data structures and methods on those data structures that makes it easier to develop new tools or for use within HPC applications to provide portable, performant implementations across file systems common in HPC centers.

```
#include "mfu.h"
```

This file includes all other necessary headers.

4.3.2 mfu_flist

The key data structure in libmfu is a distributed file list called mfu_flist. This structure represents a list of files, each with stat-like metadata, that is distributed among a set of MPI ranks.

The library contains functions for creating and operating on these lists. For example, one may create a list by recursively walking an existing directory or by inserting new entries one at a time. Given a list as input, functions exist to create corresponding entries (inodes) on the file system or to delete the list of files. One may filter, sort, and remap entries. One can copy a list of entries from one location to another or compare corresponding entries across two different lists. A file list can be serialized and written to or read from a file.

Each MPI rank "owns" a portion of the list, and there are routines to step through the entries owned by that process. This portion is referred to as the "local" list. Functions exist to get and set properties of the items in the local list, for example to get the path name, type, and size of a file. Functions dealing with the local list can be called by the MPI process independently of other MPI processes.

Other functions operate on the global list in a collective fashion, such as deleting all items in a file list. All processes in the MPI job must invoke these functions simultaneously.

For full details, see mfu_flist.h and refer to its usage in existing tools.

4.3.3 mfu_path

mpiFileUtils represents file paths with the mfu_path structure. Functions are available to manipulate paths to prepend and append entries, to slice paths into pieces, and to compute relative paths.

4.3.4 mfu_param_path

Path names provided by the user on the command line (parameters) are handled through the *mfu_param_path <https://github.com/hpc/mpifileutils/blob/master/src/common/mfu_param_path.h>_* structure. Such paths may have to be checked for existence and to determine their type (file or directory). Additionally, the user may specify many such paths through invocations involving shell wildcards, so functions are available to check long lists of paths in parallel.

4.3.5 mfu_io_and_mfu_util

The mfu_io.h functions provide wrappers for many POSIX-IO functions. This is helpful for checking error codes in a consistent manner and automating retries on failed I/O calls. One should use the wrappers in mfu_io if available, and if not, one should consider adding the missing wrapper.

The mfu_util.h functions provide wrappers for error reporting and memory allocation.

CHAPTER 5

Man Pages

5.1 dbcast

5.1.1 SYNOPSIS

dbcast [OPTION] SRC DEST

5.1.2 DESCRIPTION

Parallel MPI application to recursively broadcast a single file from a global file system to node-local storage, like ramdisk or an SSD.

The file is logically sliced into chunks and collectively copied from a global file system to node-local storage. The source file SRC must be readable by all MPI processes. The destination file DEST should be the full path of the file in node-local storage. If needed, parent directories for the destination file will be created as part of the broadcast.

In the current implementation, dbcast requires at least two MPI processes per compute node, and all compute nodes must run an equal number of MPI processes.

5.1.3 OPTIONS

-s, --size SIZE
: The chunk size in bytes used to segment files during the broadcast. Units like "MB" and "GB" should be immediately follow the number without spaces (ex. 2MB). The default size is 1MB. It is recommended to use the stripe size of a file if this is known.

-h, --help
: Print the command usage, and the list of options available.

5.1.4 EXAMPLES

1. To broadcast a file to /ssd on each node:

```
mpirun -np 128 dbcast /global/path/to/filenane /ssd/filename
```

2. Same thing, but slicing at 10MB chunks:

```
mpirun -np 128 dbcast -s 10MB /global/path/to/filenane /ssd/filename
```

3. To read the current striping parameters of a file on Lustre:

```
lfs getstripe /global/path/to/filename
```

5.1.5 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.2 dchmod

5.2.1 SYNOPSIS

dchmod [OPTION] PATH ...

5.2.2 DESCRIPTION

Parallel MPI application to recursively change permissions and/or group from a top level directory.

dchmod provides functionality similar to *chmod(1)*, *chown(1)*, and *chgrp(1)*. Like *chmod(1)*, the tool supports the use of octal or symbolic mode to change the permissions.

5.2.3 OPTIONS

-i, --input FILE
Read source list from FILE. FILE must be generated by another tool from the mpiFileUtils suite.

-u, --owner USER
Change owner to specified USER name.

-g, --group GROUP
Change group to specified GROUP name.

-m, --mode MODE
The mode to apply to each item. MODE may be octal or symbolic syntax similar to *chmod(1)*. In symbolic notation, "ugoa" are supported as are "rwxX". As with chmod, if no leading letter "ugoa" is provided, mode bits are combined with umask to determine the actual mode.

--exclude REGEX
Do not modify items whose full path matches REGEX, processed by *regexec(3)*.

--match REGEX
Only modify items whose full path matches REGEX, processed by *regexec(3)*.

--name
Change –exclude and –match to apply to item name rather than its full path.

-v, --verbose
Run in verbose mode. Prints a list of statistics including the number of files walked, the number of levels there are in the directory tree, and the number of files the command operated on, and the files/sec rate for each of those.

-h, --help
Print the command usage, and the list of options available.

5.2.4 EXAMPLES

1. Use octal mode to change permissions:

```
mpirun -np 128 dchmod --mode 755 /directory
```

2. Set group and mode in a single command using symbolic mode:

```
mpirun -np 128 dchmod --group mygroup --mode u+r,g+rw /directory
```

3. Set owner and group, leaving permissions the same:

```
mpirun -np 128 dchmod --owner user1 --group mygroup /directory
```

4. Change permissions to u+rw on all items EXCEPT those whose name match regex:

```
mpirun -np 128 dchmod --name --exclude 'afilename' --mode u+rw /directory
```

Note: You can use –match to change file permissions on all of the files/directories that match the regex.

5.2.5 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.3 dcmp

5.3.1 SYNOPSIS

dcmp [OPTION] SRC DEST

5.3.2 DESCRIPTION

Parallel MPI application to compare two files or to recursively compare files with same relative paths within two different directories.

dcmp provides functionality similar to a recursive *cmp(1)*. It reports how many files in two different directories are the same or different.

dcmp can be configured to compare a number of different file properties.

5.3.3 OPTIONS

-o, --output `EXPR:FILE`
Writes list of files matching expression EXPR to specified FILE. The expression consists of a set of fields and states described below. More than one -o option is allowed in a single invocation, in which case, each option should provide a different output file name.

-t, --text
Change –output to write files in text format rather than binary.

-b, --base
Enable base checks and normal stdout results when –output is used.

-v, --verbose
Run in verbose mode. Prints a list of statistics/timing data for the command. Files walked, started, completed, seconds, files, bytes read, byte rate, and file rate.

-l, --lite
lite mode does a comparison of file modification time and size. If modification time and size are the same, then the contents are assumed to be the same. Similarly, if the modification time or size is different, then the contents are assumed to be different. The lite mode does no comparison of data/content in the file.

-h, --help
Print the command usage, and the list of options available.

5.3.4 EXPRESSIONS

An expression is made up of one or more conditions, where each condition specifies a field and a state. A single condition consists of a field name, an '=' sign, and a state name.

Valid fields are listed below, along with the property of the entry that is checked.

Field	Property of entry
EXIST	whether entry exists
TYPE	type of entry, e.g., regular file, directory, symlink
SIZE	size of entry in bytes, if a regular file
UID	user id of entry
GID	group id of entry
ATIME	time of last access
MTIME	time of last modification
CTIME	time of last status change
PERM	permission bits of entry
ACL	ACLs associated with entry, if any
CONTENT	file contents of entry, byte-for-byte comparision, if a regular file

Valid conditions for the EXIST field are:

Condition	Meaning
EXIST=SRC_ONLY	entry exists only in source path
EXIST=DST_ONLY	entry exists only in destination path
EXIST=DIFFER	entry exists in either source or destination, but not both
EXIST=COMMON	entry exists in both source and destination

All other fields may only specify the DIFFER and COMMON states.

Conditions can be joined together with AND (@) and OR (,) operators without spaces to build complex expressions. For example, the following expression reports entries that exist in both source and destination paths, but are of different types:

```
EXIST=COMMON@TYPE=DIFFER
```

The AND operator binds with higher precedence than the OR operator. For example, the following expression matches on entries which either (exist in both source and destination and whose types differ) or (only exist in the source):

```
EXIST=COMMON@TYPE=DIFFER,EXIST=SRC_ONLY
```

Some conditions imply others. For example, for CONTENT to be considered the same, the entry must exist in both source and destination, the types must match, the sizes must match, and finally the contents must match:

```
SIZE=COMMON    => EXISTS=COMMON@TYPE=COMMON@SIZE=COMMON
CONTENT=COMMON => EXISTS=COMMON@TYPE=COMMON@SIZE=COMMON@CONTENT=COMMON
```

A successful check on any other field also implies that EXIST=COMMON.

When used with the -o option, one must also specify a file name at the end of the expression, separated with a ':'. The list of any entries that match the expression are written to the named file. For example, to list any entries matching the above expression to a file named outfile1, one should use the following option:

```
-o EXIST=COMMON@TYPE=DIFFER:outfile1
```

If the –base option is given or when no output option is specified, the following expressions are checked and numeric results are reported to stdout:

```
EXIST=COMMON
EXIST=DIFFER
EXIST=COMMON@TYPE=COMMON
EXIST=COMMON@TYPE=DIFFER
EXIST=COMMON@CONTENT=COMMON
EXIST=COMMON@CONTENT=DIFFER
```

5.3.5 EXAMPLES

1. Compare two files in different directories:

```
mpirun -np 128 dcmp /src1/file1 /src2/file2
```

2. Compare two directories with verbose output. The verbose output prints timing and number of bytes read:

```
mpirun -np 128 dcmp -v /src1 /src2
```

3. Write list of entries to outfile1 that are only in src1 or whose names exist in both src1 and src2 but whose types differ:

```
mpirun -np 128 dcmp -o EXIST=COMMON@TYPE=DIFFER,EXIST=SRC_ONLY:outfile1 /src1
/src2
```

4. Same as above but also write list of entries to outfile2 that exist in either src1 or src2 but not both:

```
mpirun -np 128 dcmp -o EXIST=COMMON@TYPE=DIFFER,EXIST=SRC_ONLY:outfile1 -o
EXIST=DIFFER:outfile2 /src1 /src2
```

5.3.6 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.4 dcp

5.4.1 SYNOPSIS

dcp [OPTION] SRC DEST

5.4.2 DESCRIPTION

Parallel MPI application to recursively copy files and directories.

dcp is a file copy tool in the spirit of *cp(1)* that evenly distributes work across a large cluster without any centralized state. It is designed for copying files that are located on a distributed parallel file system.

5.4.3 OPTIONS

-i, --input FILE
Read source list from FILE. FILE must be generated by another tool from the mpiFileUtils suite.

-p, --preserve
Preserve permissions, group, timestamps, and extended attributes.

-s, --synchronous
Use synchronous read/write calls (open files with 0_DIRECT)

-S, --sparse
Create sparse files when possible (non-functioning).

-v, --verbose
Run in verbose mode.

-h, --help
Print a brief message listing the *dcp(1)* options and usage.

5.4.4 RESTRICTIONS

If a long-running copy is interrupted, one should delete the partial copy and run dcp again from the beginning. One may use drm to quickly remove a partial copy of a large directory tree.

To ensure the copy is successful, one should run dcmp after dcp completes to verify the copy, especially if dcp was not run with the -s option.

5.4.5 EXAMPLES

1. To copy dir1 as dir2:

```
mpirun -np 128 dcp /source/dir1 /dest/dir2
```

2. To copy contents of dir1 into dir2:

```
mkdir /dest/dir2 mpirun -np 128 dcp /source/dir1/\* /dest/dir2
```

3. To copy while preserving permissions, group, timestamps, and attributes:

```
mpirun -np 128 dcp -p /source/dir1/ /dest/dir2
```

5.4.6 KNOWN BUGS

Using the -S option for sparse files does not work yet at LLNL. If you try to use it then dcp will default to a normal copy.

The maximum supported file name length for any file transferred is approximately 4068 characters. This may be less than the number of characters that your operating system supports.

5.4.7 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.5 ddup

5.5.1 SYNOPSIS

ddup [OPTION] PATH

5.5.2 DESCRIPTION

Parallel MPI application to report files under a directory tree having identical content.

ddup reports path names to files having identical content (duplicate files). A top-level directory is specified, and the path name to any file that is a duplicate of another anywhere under that same directory tree is reported. The path to each file is reported, along with a final hash representing its content. Multiple sets of duplicate files can be matched using this final reported hash.

5.5.3 OPTIONS

-d, --debug LEVEL
: Set verbosity level. LEVEL can be one of: fatal, err, warn, info, dbg.

-h, --help
: Print the command usage, and the list of options available.

5.5.4 EXAMPLES

1. To report any duplicate files under a directory tree:

```
mpirun -np 128 ddup /path/to/haystack
```

5.5.5 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.6 dfilemaker

5.6.1 SYNOPSIS

dfilemaker <nitems> <nlevels> <maxflen>

5.6.2 DESCRIPTION

dfilemaker creates a random directory tree with files having random data that is useful for testing.

Files and directories are created in the current working directory where the tool is executed. dfilemaker takes three positional parameters:

nitems
Total number of items to create.

nlevels
Maximum depth to create in directory level (relative to current path).

maxflen
Maximum number of bytes to write to a file.

The following options are planned in future releases, but they are not yet implemented.

5.6.3 OPTIONS

-d, --depth=*min*-*max*
Specify the depth of the file system tree to generate. The depth will be selected at random within the bounds of min and max. The default depth is set to 10 min, 20 max.

-f, --fill=*type*
Specify the fill pattern of the file. Current options available are: `random`, `true`, `false`, and `alternate`. `random` will fill the file using *urandom(4)*. `true` will fill the file with a 0xFF pattern. `false` will fill the file with a 0x00 pattern. `alternate` will fill the file with a 0xAA pattern. The default fill is `random`.

-r, --ratio=*min*-*max*
Specify the ratio of files to directories as a percentage. The ratio will be chosen at random within the bounds of min and max. The default ratio is 5% min to 20% max.

-i, --seed=*integer*
Specify the seed to use for random number generation. This can be used to create reproducible test runs. The default is to generate a random seed.

-s, --size=*min*-*max*
Specify the file sizes to generate. The file size will be chosen at random random within the bounds of min and max. The default file size is set from 1MB to 5MB.

-w, --width=*min*-*max*
Specify the width of the file system tree to generate. The width will be selected at random within the bounds of min and max. The width of the tree is determined by counting directories. The default width is set to 10 min, 20 max.

-h, --help
Print a brief message listing the *dfilemaker(1)* options and usage.

-v, --version
Print version information and exit.

5.6.4 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.7 dfind

5.7.1 SYNOPSIS

dfind [OPTION] [EXPRESSION] PATH ...

5.7.2 DESCRIPTION

Parallel MPI application to filter a list of files according to an expression.

dfind provides functionality similar to *find(1)*.

The file list can be obtained by either walking one or more paths provided on the command line or through an input list.

The filtered list can be written to an output file.

5.7.3 OPTIONS

-i, --input FILE
Read source list from FILE. FILE must be generated by another tool from the mpiFileUtils suite.

-o, --output FILE
Write the processed list to a file.

-v, --verbose
Run in verbose mode.

-h, --help
Print a brief message listing the *dfind(1)* options and usage.

5.7.4 EXPRESSIONS

Numeric arguments can be specified as:

+N	more than N
-N	less than N
N	exactly N

--amin N
File was last accessed N minutes ago.

--anewer FILE
File was last accessed more recently than FILE was modified.

--atime N
File was last accessed N days ago.

--cmin N
File's status was last changed N minutes ago.

--cnewer FILE
File's status was last changed more recently than FILE was modified.

--ctime N
File's status was last changed N days ago.

--gid N
File's numeric group ID is N.

--group NAME
File belongs to group NAME.

--mmin N
File's data was last modified N minutes ago.

--name PATTERN
Base of file name matches shell pattern PATTERN.

--path PATTERN
Full path to file matches shell pattern PATTERN.

--regex REGEX
Full path to file matches POSIX regular expression REGEX. Regular expressions processed by *regexec(3)*.

--newer FILE
File was modified more recently than FILE.

--mtime N
File's data was last modified N days ago.

--size N
File size is N bytes. Units can be used like 'KB', 'MB', 'GB'.

--type C
File is of type C:

d	directory
f	regular file
l	symbolic link

--uid N
File's numeric user ID is N.

--user NAME
File is owned by user NAME.

5.7.5 ACTIONS

--print
Print file name to stdout.

--exec CMD **;**
Execute command CMD on file. All following arguments are taken as arguments to the command until ';' is encountered. The string '{}' is replaced by the current file name.

5.7.6 EXAMPLES

1. Print all files owner by user1 under given path:

```
mpirun -np 128 dfind -v --user user1 --print /path/to/target
```

2. To find all files less than 1GB and write them to a file:

```
mpirun -np 128 dfind -v -o outfile --size -1GB /path/to/target
```

3. Filter list in infile to find all regular files not changed in the past 180 days and write new list to outfile:

```
mpirun -np 128 dfind -v -i infile -o outfile --type f --mtime +180
```

5.7.7 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.8 dreln

5.8.1 SYNOPSIS

dreln [OPTION] OLDPATH NEWPATH PATH ...

5.8.2 DESCRIPTION

Parallel MPI application to recursively update symlinks within a directory.

dreln walks the specified PATH and updates any symlink whose target includes an absolute path to OLDPATH and replaces that symlink with a new link whose target points to NEWPATH instead.

This is useful to update symlinks after migrating a large directory from one file system to another, whose links specify absolute paths to the original file system.

5.8.3 OPTIONS

-i, --input FILE
: Read source list from FILE. FILE must be generated by another tool from the mpiFileUtils suite.

-p, --preserve
: Preserve existing modification times on links.

-r, --relative
: Replace links using target paths that are relative to NEWPATH.

-v, --verbose
: Run in verbose mode.

-h, --help
: Print a brief message listing the *drm(1)* options and usage.

5.8.4 EXAMPLES

1. To update all links under /walk/path whose targets point to /orig/path and replace them with targets that point to /new/path:

```
mpirun -np 128 dreln -v /orig/path /new/path /walk/path
```

2. Same as above, but replace each link target with a relative path from the link to its new target under /new/path:

```
mpirun -np 128 dreln -v --relative /orig/path /new/path /walk/path
```

3. One can preserve existing modification times on links:

```
mpirun -np 128 dreln -v --preserve /orig/path /new/path /walk/path
```

4. One can specifiy multiple paths to walk:

```
mpirun -np 128 dreln -v /orig/path /new/path /walk/path1 /walk/path2
```

5.8.5 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.9 drm

5.9.1 SYNOPSIS

drm [OPTION] PATH...

5.9.2 DESCRIPTION

Parallel MPI application to recursively delete a directory and its contents.

drm is a tool for removing files recursively in parallel. drm behaves like *rm -rf*, but it is faster.

Note: DO NOT USE SHELL REGEX!!! The –match and –exclude options use POSIX regex syntax. Because of this make sure that the shell does not try to interpret your regex before it gets passed to the program. You can generally use quotes around your regex to prevent the shell from expanding. An example of this using the –match option with –dryrun would be:

```
mpirun -np 128 drm --dryrun -v --name --match 'file_.*' /path/to/dir/*
```

5.9.3 OPTIONS

-i, --input FILE
: Read source list from FILE. FILE must be generated by another tool from the mpiFileUtils suite.

-l, --lite
: Walk file system without stat.

--exclude REGEX
: Do not remove items whose full path matches REGEX, processed by *regexec(3)*.

--match REGEX
Only remove items whose full path matches REGEX, processed by *regexec(3)*.

--name
Change –exclude and match to apply to item name rather than its full path.

--dryrun
Print a list of files that **would** be deleted without deleting them. This is useful to check list of items satisfying –exclude or –match options before actually deleting anything.

--aggressive
This option will delete files during the walk phase, and then delete directories by level after the walk in drm. You cannot use this option with –dryrun.

-T, --traceless
Delete child items without updating the mtime on their parent directory.

-v, --verbose
Run in verbose mode.

-h, --help
Print a brief message listing the *drm(1)* options and usage.

5.9.4 EXAMPLES

1. To delete a directory and its contents:

```
mpirun -np 128 drm -v /dir/to/delete
```

2. Delete all items (files and directories) ending with .core from directory tree:

```
mpirun -np 128 drm --match '.core$' /dir/to/delete/from
```

3. List items that would be deleted without removing them:

```
mpirun -np 128 drm --dryrun --match '.core$' /dir/to/delete/from
```

4. Delete all items named foo:

```
mpirun -np 128 drm --name --match '^foo$' /dir/to/delete/from
```

5.9.5 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.10 dstripe

5.10.1 SYNOPSIS

dstripe [OPTION] PATH...

5.10.2 DESCRIPTION

Parallel MPI application to restripe files.

This tool is in active development. It currently only works on Lustre.

dstripe enables one to restripe file(s) across the underlying storage devices. One must specify a list of paths. All files in those paths can be restriped. By default, stripe size is 1MB and stripe count is -1 allowing dstripe to use all available stripes.

5.10.3 OPTIONS

-c, --count STRIPE_COUNT
: The number of stripes to use during file restriping. If STRIPE_COUNT is -1, then all available stripes are used. If STRIPE_COUNT is 0, the lustre file system default is used. The default stripe count is -1.

-s, --size STRIPE_SIZE
: The stripe size to use during file restriping. Units like "MB" and "GB" can immediately follow the number without spaces (ex. 2MB). The default stripe size is 1MB.

-m, --minsize SIZE
: The minimum size a file must be to be a candidate for restriping. Files smaller than SIZE will not be restriped. Units like "MB" and "GB" can immediately follow the number without spaces (ex. 2MB). The default minimum file size is 0MB.

-r, --report
: Display the file size, stripe count, and stripe size of all files found in PATH. No restriping is performed when using this option.

-v, --verbose
: Run in verbose mode.

-h, --help
: Print the command usage, and the list of options available.

5.10.4 EXAMPLES

1. To stripe a file on all storage devices using a 1MB stripe size:

```
mpirun -np 128 dstripe -s 1MB /path/to/file
```

2. To stripe a file across 20 storage devices with a 1GB stripe size:

```
mpirun -np 128 dstripe -c 20 -s 1GB /path/to/file
```

3. To restripe all files in /path/to/files/ that are at least 1GB in size:

```
mpirun -np 128 dstripe -m 1GB /path/to/files/
```

4. To restripe all files in /path/to/files/ across 10 storage devices with 2MB stripe size:

```
mpirun -np 128 dstripe -c 10 -s 2MB /path/to/files/
```

5. To display the current stripe count and stripe size of all files in /path/to/files/:

```
mpirun -np 128 dstripe -r /path/to/files/
```

5.10.5 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.11 dsync

5.11.1 SYNOPSIS

dsync [OPTION] SRC DEST

5.11.2 DESCRIPTION

Parallel MPI application to synchronize two files or two directory trees.

dsync makes DEST match SRC, adding missing entries from DEST, and updating existing entries in DEST as necessary so that SRC and DEST have identical content, ownership, timestamps, and permissions.

5.11.3 OPTIONS

--dryrun
Show differences without changing anything.

-b, --batch-files N
Batch files into groups of up to size N during copy operation.

-c, --contents
Compare files byte-by-byte rather than checking size and mtime to determine whether file contents are different.

-D, --delete
Delete extraneous files from destination.

-v, --verbose
Run in verbose mode. Prints a list of statistics/timing data for the command. Files walked, started, completed, seconds, files, bytes read, byte rate, and file rate.

-h, --help
Print the command usage, and the list of options available.

5.11.4 EXAMPLES

1. Synchronize dir2 to match dir1:

```
mpirun -np 128 dsync /path/to/dir1 /path/to/dir2
```

5.11.5 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.12 dwalk

5.12.1 SYNOPSIS

dwalk [OPTION] PATH ...

5.12.2 DESCRIPTION

Parallel MPI application to recursively walk and list contents in a directory.

dwalk provides functionality similar to *ls(1)* and *du(1)*. Like *du(1)*, the tool reports a summary of the total number of files and bytes. Like *ls(1)*, the tool sorts and prints information about individual files.

The output can be sorted on different fields (e.g, name, user, group, size, etc). A histogram of file sizes can be computed listing the number of files that fall into user-defined bins.

5.12.3 OPTIONS

-i, --input FILE
Read source list from FILE. FILE must be generated by another tool from the mpiFileUtils suite.

-o, --output FILE
Write the processed list to a file.

-l, --lite
Walk file system without stat.

-s, --sort FIELD
Sort output by comma-delimited fields (see below).

-d, --distribution size:SEPARATORS
Print the distribution of file sizes. For example, specifying size:0,80,100 will report the number of files that have size 0 bytes, between 1-80 bytes, between 81-99 bytes, and 100 bytes or greater.

-p, --print
Print files to the screen.

-v, --verbose
Run in verbose mode.

-h, --help
Print usage.

5.12.4 SORT FIELDS

By default, the list of files dwalk captures is not sorted. To sort the list, one or more fields can be specified in a comma-delimited list:

name,user,group,uid,gid,atime,mtime,ctime,size

A field name can be preceded with '-' to sort by that field in reverse order.

A lexicographic sort is executed if more than one field is given.

5.12.5 EXAMPLES

1. To print summary information for a directory:

```
mpirun -np 128 dwalk -v /dir/to/walk
```

2. To print a list of files, sorted by file size, then by file name:

```
mpirun -np 128 dwalk -print -sort size,name /dir/to/walk
```

3. To save the list of files:

```
mpirun -np 128 dwalk -output out.dwalk /dir/to/walk
```

4. Print the file distribution for specified histogram based on the size field from the top level directory.

```
mpirun -np 128 dwalk -v -print -d size:0,20,1G src/
```

5.12.6 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.13 dbz2

5.13.1 SYNOPSIS

dbz2 [OPTIONS] [-z|-d] FILE

5.13.2 DESCRIPTION

Parallel MPI application to compress or decompress a file.

5.13.3 OPTIONS

-d, --decompress
Decompress the file

-z, --compress
Compress the file

-k, --keep
Keep the input file (optional).

-f, --overwrite
Overwrite the output file, if it exists (optional).

-b, --block SIZE
Set the compression block size, from 1 to 9. Where 1=100kB ... and 9=900kB. Default is 9 (optional).

-m, --memory SIZE
Limit the memory that can be used by a process, in bytes (optional).

-v, --verbose
Verbose output (optional).

--debug
Show debug output (optional).

5.14 dgrep

5.14.1 SYNOPSIS

dgrep ...

5.14.2 DESCRIPTION

5.14.3 OPTIONS

-h, --help
Print a brief message listing the *dgrep(1)* options and usage.

-v, --version
Print version information and exit.

Known bugs

5.14.4 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.15 dparallel

5.15.1 SYNOPSIS

dparallel ...

5.15.2 DESCRIPTION

5.15.3 OPTIONS

-h, --help
Print a brief message listing the *dparallel(1)* options and usage.

-v, --version
Print version information and exit.

Known bugs

5.15.4 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

5.16 dtar

5.16.1 SYNOPSIS

dtar ...

5.16.2 DESCRIPTION

5.16.3 OPTIONS

-h, --help
Print a brief message listing the *dtar(1)* options and usage.

-v, --version
Print version information and exit.

Known bugs

5.16.4 SEE ALSO

The mpiFileUtils source code and all documentation may be downloaded from <https://github.com/hpc/mpifileutils>

CHAPTER 6

Indices and tables

- genindex
- search

Index

Symbols

C

M

N