

# **mdtmFTP**

## **Installation & Configuration Manual**

**Version 1.0.2**

**By Fermilab Network Research Group**  
**Jan 2018**

## About

DOE's Advanced Scientific Computing Research (ASCR) office has funded Fermilab and Brookhaven National Laboratory to collaboratively work on the Multicore-Aware Data Transfer Middleware (MDTM) project (<http://mdtm.fnal.gov>). MDTM aims to accelerate data movement toolkits on multicore systems.

The MDTM research team has recently released several MDTM-related software packages – mdtmFTP, mdtmBBCP, mdtmGUI, and MDTM middleware (<http://mdtm.fnal.gov/Releases.html>).

- mdtmFTP and mdtmBBCP are two MDTM-based data transfer tools. Both tools adopt an I/O-centric architecture, and use MDTM middleware to fully utilize the underlying multicore system.
- MDTM middleware will harness multicore parallelism to scale data movement toolkits on Data Transfer Nodes (DTNs). It schedules and assigns system resources based on the needs and requirements of data transfer applications (i.e., data transfer-centric scheduling). It also takes into account other factors, including NUMA topology, I/O locality, and QoS.
- mdtmGUI is a web-based DTN monitoring tool. It is able to provide online & real-time monitoring of DTN system status and configurations, online & real-time monitoring of data transfer status and progress, and online & real-time monitoring of the MDTM-based data transfer tool's status and configuration.

mdtmFTP, MDTM middleware, and mdtmGUI are researched, designed, and developed by Fermilab network research group.

mdtmBBCP is designed, and developed by BNL Computational Science Center.

This document describes the installation and basic use of mdtmFTP.

## Contacts

Wenji Wu ([wenji@fnal.gov](mailto:wenji@fnal.gov))  
Liang Zhang ([liangz@fnal.gov](mailto:liangz@fnal.gov))  
Phil DeMar ([demar@fnal.gov](mailto:demar@fnal.gov))

## Intended Audience

This manual is intended for users and system administrators responsible for installing, running, and managing DTNs.

The manual assumes familiarity with multicore and DTN concepts.

## Acknowledgements

mdtmFTP uses some Globus modules (<http://toolkit.globus.org/toolkit>) for rapid prototyping. We sincerely thank Globus folks at Argonne National Laboratory and University of Chicago.

Here is a list of Globus modules that we use:

- GridFTP protocol module
- Globus xio module
- Globus security module
- Globus user interface

## Section 1. System level requirements

1.1 mdtmFTP has been tested under Linux systems. The system must have a Linux kernel later than 3.12.23.

1.2 System must have installed MDTM middleware and related software packages.

- Download and install *libbitmask-2.0-rev.tar.gz* software package.
  - *download libbitmask-2.0-rev.tar.gz* (<http://mdtm.fnal.gov/Releases.html>)
  - *gunzip libbitmask-2.0-rev.tar.gz*
  - *tar -xvf libbitmask-2.0-rev.tar*
  - *cd libbitmask*
  - *./configure*
  - *make*
  - *sudo make install*
  
- Download and install *libcpuset-1.0-rev.tar.gz* software package.
  - *download libcpuset-1.0-rev.tar.gz* (<http://mdtm.fnal.gov/Releases.html>)
  - *gunzip libcpuset-1.0-rev.tar.gz*
  - *tar -xvf libcpuset-1.0-rev.tar*
  - *cd libcpuset*
  - *./configure*
  - *make*
  - *sudo make install*
  
- Download and install *mdtm-1.0.2.tar.gz* software package
  - *download mdtm-1.0.2.tar.gz* (<http://mdtm.fnal.gov/Releases.html>)
  - *gunzip mdtm-1.0.2.tar.gz*
  - *tar -xvf mdtm-1.0.2.tar*
  - *cd mdtm-1.0.2*
  - *./configure*
  - *make*
  - *sudo make install*

1.3 Download and install mdtmFTP software package

- Download *mdtmftp-1.0.2.tar.gz* (<http://mdtm.fnal.gov/Releases.html>)
- Create a building directory
  - *mkdir xxx/building\_directory*
  - *cd xxx/building\_directory*
- copy the downloaded *mdtmftp-1.0.2.tar.gz* to *xxx/building\_directory*
  - *cd xxx/building\_directory*
- untar the downloaded software package
  - *tar xvfz mdtmftp-1.0.2.tar.gz*
- apply patch
  - *xxx/building\_directory/mdtm-apply-patch*

- install
  - *cd "xxx/building\_directory/mdtmftp/"*
  - *./configure --prefix=mdtmftp\_install\_directory*
  - *make mdtmftp\_server mdtmftp\_client*
- set LD\_LIBRARY\_PATH variable to find libmdtm.so.0
  - *setenv LD\_LIBRARY\_PATH /usr/local/lib (for csh)*
  - *or export LD\_LIBRARY\_PATH=/usr/local/lib (for bash)*
- set C\_INCLUDE\_PATH variable
  - *setenv C\_INCLUDE\_PATH /usr/local/include (for csh)*
  - *or export C\_INCLUDE\_PATH=/usr/local/include (for bash)*

*mdtmftp* consists of two pieces -- *mdtm-ftp-client* and *mdtm-ftp-server*. After installation, *mdtm-ftp-client* goes to directory – "*mdtmftp\_install\_directory/bin.*" And *mdtm-ftp-server* goes to directory – "*mdtmftp\_install\_directory/sbin.*"

Note: when installing the above software packages, you may get dependency problems. Please install the related software packages accordingly.

## **Section 2. Configuring Security for mdtmFTP**

mdtmFTP uses Globus security model. mdtmFTP's security configuration is the same as GridFTP. Please refer to the following Globus document for mdtmFTP's security configuration.

GT 5.2.5 GridFTP: System Administrator's Guide

- Chapter 2. Configuring GridFTP
  - Section 4. Configuration Security for GridFTP

Note: When creating password file, do not create user root.

## Section 3. Configuring and Running mdtmFTP server

### 3.1 Running mdtmFTP server with root or non-root privilege

For the purpose of performing permission checks, traditional UNIX implementations distinguish two categories of processes: *privileged* processes (whose effective user ID is 0, referred to as superuser or root), and *unprivileged* processes (whose effective UID is nonzero). Privileged processes bypass all kernel permission checks, while unprivileged processes are subject to full permission checking based on the process's credentials (usually: effective UID, effective GID, and supplementary group list).

Starting with kernel 2.2, Linux divides the privileges traditionally associated with superuser into distinct units, known as *capabilities*, which can be independently enabled and disabled. Capabilities are a per-thread attribute.

Running mdtmFTP server requires four capabilities:

- CAP\_SYS\_NICE, to bind threads to cores.
- CAP\_IPC\_LOCK, to lock memory.
- CAP\_SYS\_RESOURCE, to increase the capacity of a pipe for mdtmFTP splice.
- CAP\_SYS\_ADMIN, to increase the number of open files for mdtmFTP splice.

Therefore, when system admin runs mdtmFTP server with root privilege, no additional actions are needed.

However, if mdtmFTP server is run as a normal application, please add capabilities to *mdtm-ftp-server*

- “CAP\_SYS\_NICE”
- “CAP\_IPC\_LOCK”
- “CAP\_SYS\_RESOURCE”
- “CAP\_SYS\_ADMIN”

Assume mdtm-ftp-server has been installed to directory “/xxx/mdtmftp\_directory”, please run the following command:

```
“setcap cap_sys_nice,cap_ipc_lock,cap_sys_admin,cap_sys_resource+ep  
/xxx/mdtmftp_directory/sbin/mdtm-ftp-server”
```

## 3.2 mdtmFTP server configuration files

Running mdtmFTP server requires properly configuring two files – mdtmconfig.xml and server.conf.

### 3.2.1 mdtmconfig.xml

mdtmconfig.xml configures a mdtmFTP server's MDTM-related parameters. This file can exist in two places:

1. /etc/mdtm/mdtmconfig.xml: it is installed with MDTM middleware and contains template values. System administrators need to modify it to fit the system in use.
2. mdtmconfig.xml in the current working directory: specific to the current application.

The mdtmconfig.xml in the working directory overrides /etc/mdtm/mdtmconfig.xml.

mdtmconfig.xml consists of four sections: Topology, Online, Thread, and File section:

- Topology section. The syntax is defined as:

```
<Topology>
  <Device type="Device_Type" numa="Numa_ID">Device_Name</device>
  ...
</Topology>
```

**Device\_Type** refers to MDTM device type. MDTM defines three types of devices: *network*, *block*, and *virtual*.

- *Network* refers to a network I/O device.
- *Block* refers to a storage I/O device.
- *Virtual* refers to a virtual device, which is defined particularly for mdtmFTP server.

**Numa\_ID** sets which NUMA node a device belongs to (i.e., NUMA location).

**Device\_Name** specifies a device name.

MDTM middleware is typically able to detect physical I/O devices and their locations (i.e., which NUMA node that a I/O device belongs to) on a NUMA system. However, there are two cases that MDTM middleware cannot detect physical I/O devices or their locations correctly: (a) in a fully virtualized environment, where information on physical I/O devices is not exposed to MDTM middleware. And (b) Some vendors' I/O devices may not comply to OS rules to expose device information properly. Under these conditions, system admin should manually configure I/O devices and their NUMA locations.

*Virtual* device is defined particularly for mdtmFTP server to monitor data transfer status. mdtmFTP server spawns a dedicated management thread to collect and record



data transfer statistics. The management thread is associated with a virtual device, which will be pinned to a specified NUMA node.

- Online section. The syntax is defined as:

```
<Online>
  <Device>device_name</Device>
...
</Online>
```

This section specifies the I/O devices that are assigned for data transfer.

For example, assume a DTN has the following I/O devices:

- Ethernet NIC devices
  - eth0 – configured for management access
  - eth1 – configured for WAN data transfer
- Block I/O devices
  - /dev/sda – system disk
  - /dev/sdb – data repository for WAN data transfer

In this case, the online section would be defined as

```
<Online>
  <Device>eth1</Device>
  <Device>sdb</Device>
</Online>
```

- Thread section. The syntax is defined as:

```
<Threads threads="Default_Num">
  <Device type="Device_Type" threads="Num">Device_Name</Device>
...
</Threads>
```

This section defines the number of threads that needs to be allocated for an I/O device. The number of threads allocated for an I/O device should be proportional to the device's I/O bandwidth. The rule of thumb is that a thread can handle an I/O rate of 10Gbps. For example, four threads should be allocated for a 40GE NIC while one thread be allocated for a 10GE NIC.

**Default\_Num** sets the default number of threads allocated for each I/O device.

If a different number of threads should be allocated for a particular I/O device, a separate entry for the device should to be specified here.

A virtual device should be allocated with 1 thread.

- File section. The syntax is defined as:

```
<File segment="File_Size_Threshold">
</File>
```

MDTM splits a large file into segments, which are spread to different threads for disk and network operations to increase performance.

**File\_Size\_Threshold** sets file size threshold that a file should be split into segments

Here is a sample *mdtmconfig.xml* file for mdtmFTP server:

```
-----
<?xml version="1.0" standalone="no" ?>
<Topology>
  <Device type="Virtual" numa="1">man</Device>
  <Device type="Network" numa="0">eth40.4020</Device>
</Topology>
<Online>
  <Device>eth40.4020</Device>
  <Device>sda</Device>
  <Device>man</Device>
</Online>
<Threads threads="1">
  <Device type="Network" threads="2">eth40.4020</Device>
  <Device type="Block" threads="2">sda</Device>
  <Device type="Virtual" threads="1">man</Device>
</Threads>
<File segment="2G">
</File>
-----
```

### 3.2.2 *server.conf*

*server.conf* configures a *mdtmFTP* server's operation parameters. Note: the file name can be changed.

- ***blocksize*** sets the block size for disk I/O operations. The block size should be 4K or multiple of 4k (e.g. 4M).
- ***direct*** is a flag to enable or disable direct I/O
- ***splice*** is a flag to enable or disable zero-copy by using `splice()`
- ***monitor*** is a flag to enable or disable MDTM monitoring

Here is a sample *server.conf* file:

```
-----  
blocksize 4194304  
direct 1  
splice 1  
monitor 0  
-----
```

### 3.2.3 Running *mdtmFTP* server

*mdtmFTP* server command syntax

```
mdtm-ftp-server -data-interface <ip_address> -password-file <passwd_file> -p <port_num> -c <server.conf>
```

command line options:

|  |   |
|--|---|
| <i>-data-interface &lt;ip_address&gt;</i>      | Specifies a server's IP interface that is used for data transfer                                    |
| <i>-password-file &lt;passwd_file_name&gt;</i> | Specifies a password file to authenticate users. You can use Globus tools to create a password file |
| <i>-p &lt;port_num&gt;</i>                     | Specifies the port that <i>mdtmFTP</i> server listens on  |
| <i>-c &lt;server.conf&gt;</i>                  | Specifies a CONFIG file to set data transfer parameters   |

## Section 4. Configuring and Running mdtmFTP client

### 4.1 Running mdtmFTP client with root or non-root privilege

mdtmFTP client can be launched with either root or non-root privilege.

### 4.2 mdtmFTP client configuration file

Running mdtmFTP client requires properly configuring a file -- mdtmconfig.xml.

#### 4.2.1 *mdtmconfig.xml*

mdtmconfig.xml configures a mdtmFTP client's MDTM-related parameters. This file must be put in mdtmFTP client's working directory.

mdtmFTP client's configuration is similar to that of mdtmFTP server, except that mdtmFTP client does not need to configure a virtual device.

Here is a sample mdtmconfig.xml file:

```
-----  
<?xml version="1.0" standalone="no" ?>  
<Topology>  
  <Device type="Network" numa="1">eth40.4012</Device>  
</Topology>  
<Online>  
  <Device>eth40.4012</Device>  
  <Device>sda</Device>  
</Online>  
<Threads threads="1">  
  <Device type="Network" threads="2">eth40.4012</Device>  
  <Device type="Block" threads="2">sda</Device>  
</Threads>  
<File segment="10G">  
</File>  
-----
```

### 4.3 running *mdtmFTP* client

mdtmFTP client command syntax

```
mdtm-ftp-client -p <parallelism> [-splice] Source_URL Destination_URL
```

command line options:

|                  |  |
|------------------|--|
| -p <parallelism> | Specifies the number of parallel data streams that should be use |
| -Source_URL      | Specifies the URL of data source                                 |
| -Destination_URL | Specifies the URL of data destination                            |
| -splice          | Enable zero-copy by using splice()                               |

## Section 5. Data Transfer Examples

### 5.1 Client – Server data transfer

#### Step 1: Launch the server on DTN A

```
[rootww@mdtm-nersc-tbn-2-189 mdtm-test]# /home/wenji/mdtmftp/sbin/mdtm-ftp-server -data-interface 10.40.130.189 -password-file pwfile -p 5001 -c server.conf
```

#### Step 2: Launch the client on DTN B

- Single file data transfer: transfer a single file from DTN A to DTN B

```
/home/wenji/mdtmftp/bin/mdtm-ftp-client -p 8  
ftp://wenji:123456@10.40.130.189:5001/storage/data1/testfiles/100G/file1  
file:///storage/data1/tmp/
```

- Single file data transfer: transfer a single file from DTN B to DTN A

```
/home/wenji/mdtmftp/bin/mdtm-ftp-client -p 8  
file:///storage/data1/tmp/file1  
ftp://wenji:123456@10.40.130.189:5001/storage/data1/tmp/
```

- Folder data transfer: transfer a Linux folder from DTN A to DTN B

```
/home/wenji/mdtmftp/bin/mdtm-ftp-client -p 8  
ftp://wenji:123456@10.40.130.189:5001/storage/data1/linux-3.18.21/  
file:///storage/data1/tmp/
```

- Folder data transfer: transfer a Linux folder from DTN B to DTN A

```
/home/wenji/mdtmftp/bin/mdtm-ftp-client -p 8  
file:///storage/data1/tmp/linux-3.18.21/  
ftp://wenji:123456@10.40.130.189:5001/storage/data1/tmp/
```

## 5.2 Third party data transfer between two remote DTNs

### Step 1: Launch mdtmFTP server on remote DTN A

```
docker -v /storage_x:/storage_y --net=host xxx:yyy /bin/bash -c "cd /home/mdtmftp_server;  
./mdtm-ftp-server -data-interface 131.225.2.29 -password-file passfile -p 5001 -c server.conf" &
```

### Step 2: Launch mdtmFTP server on remote DTN B

```
docker -v /storage_x:/storage_y --net=host xxx:yyy /bin/bash -c "cd /home/mdtmftp_server;  
./mdtm-ftp-server -data-interface 131.225.2.31 -password-file passfile -p 5001 -c server.conf" &
```

### Step 3: Launch the client on local DTN C

Assuming the mdtmFTP client runs in a container.

- Single file data transfer: transfer a single file from DTN A to DTN B

```
/home/mdtmftp_client/mdtm-ftp-client -p 8 -vb  
ftp://wenji:123456@131.225.2.29:5001/storage/data1/testfiles/100G/file1  
ftp://wenji:123456@131.225.2.31:5001/storage/data1/tmp/
```

- Folder data transfer: transfer a Linux folder from DTN A to DTN B

```
/home/mdtmftp_client/mdtm-ftp-client -p 8 -vb  
ftp://wenji:123456@131.225.2.29:5001/storage/data1/linux-3.18.21/  
ftp://wenji:123456@131.225.2.31:5001/storage/data1/tmp/
```