mdtmGUI
Installation & Configuration Manual
- Docker Release

Version 1.0.1

By Fermilab Network Research Group
Oct 2016

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 released several MDTM-related software packages – mdtmFTP, mdtmBBCP, mdtmGUI, and MDTM middleware (http://mdtm.fnal.gov/Releases.html).

- mdtmFTP and mdtmBBCP. 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. The middleware will hardness 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. 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 mdtmGUI.

mdtmGUI is developed by Lauri Loebel Carpenter (lauri@fnal.gov).

The MDTM project PI is Dr. Wenji Wu (wenji@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.

System level requirements

1) System must have installed Docker (version 1.10 +). The Docker project website is available at http://www.docker.com. For some Linux distributions, you can install Docker packages through *yum* or *apt-get*.

2) Download and install mdtmGUI Docker package

    - The mdtmGUI Docker repository: https://hub.docker.com/r/wenji/mdtm

    - Download mdtmGUI container:
        - *"docker pull docker.io/wenji/mdtm:mdtmgui"* (for normal users)
        - *"docker pull docker.io/wenji/mdtmgui-ESnet"* (for ESnet users)

    - Run "*docker images*" to check the container that you have pulled

Running the mdtmGUI container

1) On a system, start mdtmGUI by running

   "*docker run –net=host –ti docker.io/wenji/mdtmgui*".

   This command will start the mdtmGUI container interactively. You can login the container to launch applications, or edit/configure files.

   Within the container, mdtmGUI is located at folder *"/home/mdtmgui"*

   To automatically start mdtmGUI, the following lines of commands have been added to *"/etc/bash.bashrc"* (in the mdtmGUI container). Therefore, once the container is started, mdtmGUI will be automatically launched.

   ```
   /usr/bin/monitorix -c /etc/monitorix/monitorix.conf -p /var/run/monitorix.pid;
   cd /home/mdtmgui;
   forever start app.js --port 8080;
   ```

   In the container, you can stop mdtmGUI by running

   ```
   forever stop 0
   ```

   Or relaunch mdtmGUI by running

   ```
   cd /home/mdtmgui;
   forever start app.js –port 8080;
   ```

   Note: 8080 is mdtmGUI listening port, which is configurable

2) On another system, run http://mdtmgui-server-ip:8080 to check if mdtmGUI is launched successfully. A successfully launched mdtmGUI is shown as in Figure 1. Note: please let mdtmGUI run for several minutes to collect data.

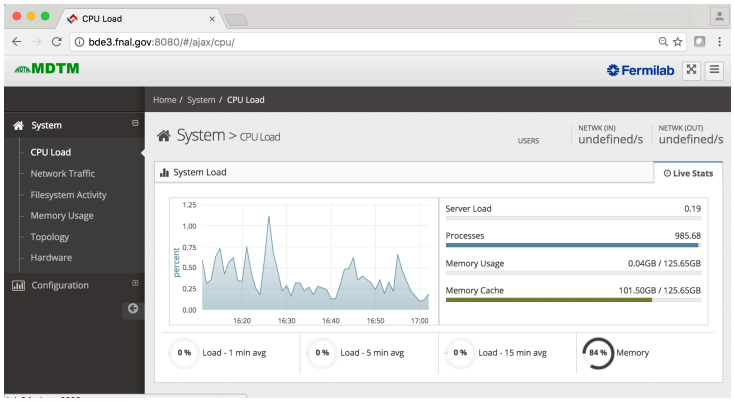

Figure 1 a successfully launched mdtmGUI

If not, please check whether your system firewall, network firewall, or mdtmGUI port number is correctly configured.

3) If mdtmGUI is successfully launched, congratulation! However, the configuration task is only partially completed. You will notice that mdtmGUI's "Network Traffic", and "Filesystem Activity" pages are data empty (Figure 2 and 3) because no network interface cards and file systems have been configured for it. Because each system typically has system-specific hardware configurations, a mdtmGUI system requires customized configurations.

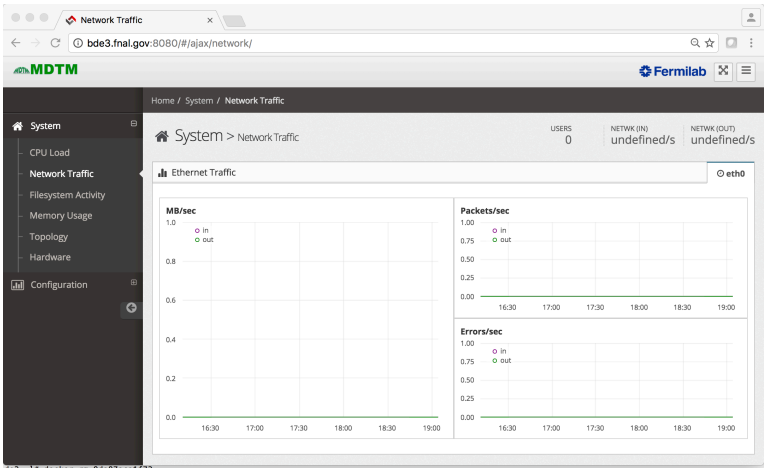

Figure 2 Empty Network Traffic Page

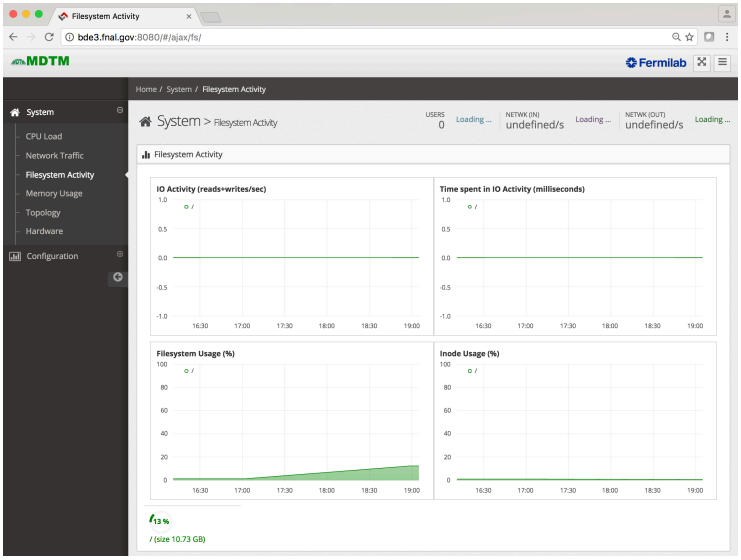

Figure 3 Empty Filesystem Activity Page

The system-specific configurations for mdtmGUI represent in two aspects: NICs or file systems. Please follow the following instructions:

a) *NIC configuration*. The mdtmGUI container runs in "host" network mode. It uses the host's network stack, and sees all NICs.
   - Login to the mdtmGUI container.
   - "vi /etc/monitorix/monitorix.conf."
   - Edit the "NET graph" section, add the NICs that you want to monitor. Please refer to http://www.monitorix.org/documetation.html on how to edit monitorix.conf.

b) File system configuration.
   - Login to the mdtmGUI container
   - Mount particular storage/file systems to the mdtmGUI container. You can put the mount scripts in "/etc/bash.bashrc" to automate the process. Therefore, when the container is launched, file systems will be automatically mounted.
   - "vi /etc/monitoirx/monitorix.conf"
   - Edit the "FS graph" section, add the file systems that you want to monitor. Please refer to http://www.monitorix.org/documentation.html on how to edit monitorix.conf

c) In the container, restart the Monitorix by running *"/etc/init.d/monitorix restart"*

d) In the container, restart mdtmGUI

e) On another system, run http://mdtmgui-server-ip:8080 to check if mdtmGUI is launched successfully. Make sure that the "Network Traffic", and "Filesystem Activity" pages are not empty. If yes, congratulation! You are almost done. If not, go back to check whether file systems, or monitoirx is correctly mounted, or configured.

4) You have made system-specific configurations for mdtmGUI. You want to save the changes.
   - Exit the mdtmGUI container
   - In the host system, run "docker ps –a" to find *container_id* for the container that you just exit.
   - Save the container changes by running "docker commit *container id xxx:yyy*" Note: xxx is the repository name, yyy is the tag name for the customized mdtmGUI container that is specific to your DTN system.

5) On the mdtmGUI server, start the mdtmGUI by running

   "docker run –t --net=host xxx:yyy &".

   This command will start the customized mdtmGUI container in the background.

Running modes

mdtmGUI is designed and developed for MDTM-enabled DTNs. However, it can also be deployed to monitor and manage any networked computer systems.

An MDTM-enabled DTN is a DTN that runs an MDTM-based data transfer tool (e.g., mdtmFTP), along with MDTM middleware.

For an MDTM-enabled DTN, mdtmGUI provides a full set of functions:

- Online & real-time monitoring of system status and configurations
- Online & real-time monitoring of data transfer status and progress.

When mdtmGUI is deployed to monitor a Non-MDTM-enabled computer system, it provides a reduced set of functions:

- Online and real-time monitoring of system status and configurations.

mdtmGUI CODE ORGANIZATION

- mdtmConfig.js

  This is the main configuration file, needs to be edited and then the server restarted. Lots of things are configured in here -- the colors to use, the number of "cores" to show on each plot on the "System> CPU Load" page, the paths to the CSV files that are written by the MDTM software for data transfer statistics, etc. The default mdtmConfig.js should be fairly self-explanatory, with full examples indicating required syntax, etc.

  When mdtmGUI runs, the “Configuration -> User Config" and “Configuration->Full Configuration” tabs show the contents of the configuration file.

  When mdtmGUI is deployed to monitor and manage non-mdtm-enabled computer systems, all MDTM features should be disabled by setting “$MDTM_MONITORING_ENABLED” to “false”.

- api/controllers/
  - MainController: handle the "index.ejs" front-page web requests
  - AjaxController: handle the main "nav panel" web requests
  - RrdController: handle the ajax callbacks related to obtaining monitorix “rrd fetch” data
  - SnapshotController: handle "the rest" of the ajax callbacks, which generally get a “snapshot" of "how some command output looks right now”

- api/services/
  - mdtmConfigService: called during startup, this builds a $CONFIG data structure of All Known Static Data (including the $USER_CONFIG information from the mdtmConfig.js file, and parsing the output of several other commands to obtain static information such as hardware, memory, colors to use, etc.).

  xxx.js: naming convention should make it fairly clear what each of the files in this area do.

- assets/
  - Javascript source code/scripts, css style sheets, images, jQuery packages, etc.

- views/
  - main/index.index.ejs - the "front page".
  - ajax/*.ejs - the "container" template for individual pages.
  - ajax/templates/*.ejs - the individual templates sucked into each "container".

Feedbacks

If you encounter any problems when installing, configuring, and running mdtmGUI, please send emails to Dr. Wenji Wu (wenji@fnal.gov).