Mumfie DFS

User Guide

Rasmus Holm, carho647@student.liu.se

Last modified: 2012-04-26

Table of content

General

MDFS consists of three main parts.

- ClientAPI
  - Is the software that lets a client access and perform file operations to MDFS
- NameNode
  - Is a server software that keep tracks and stores meta data regarding content in MDFS.
- DataNode
  - Is a server software that provides the storage of files in MDFS, they store raw data on a local file system.

The ClienAPI, NameNode and DataNode are independent softwares that all has to be ran on a separate JVM:s and preferable on different machines. They have no dependencies between each other. However, they all depend on the package mdfs.utils and subpackages to said package as well as org.json.*

Configurations of each software are found in the relative path mdfs/config/config.cfg

Dependencies

- ClientAPI
  - mdfs.utils
- NameNode
  - mdfs.utils
  - mysql-connector-java-5.0.4
- DataNode
  - mdfs.utils
- utils
  - apache-commons-collection-3.2.1
  - apache-commons-configuration-1.8
  - apache-commons-lang-2.6
  - apache-commons-logging-1.1.1

ClientAPI

The ClientAPI consists simply of an api, mdfs.client.api.FileQuery and mdfs.client.api.FileQueryImpl. This are what one use when accessing MDFS and they are self explanatory.

Configuration

The configuration of the client i rather simple. What it must contain is:

- The client's dns-address, this might however just be set to `localhost`
- The dns address and port that the Name Node can be found on.

```
ClientAPI Example configNameNode.address :
[] = optional

[name = client]
[mac = 50:e5:49:39:96:85]

address = localhost

[ip = 127.0.0.1]
[tmpdir = /tmp/]

NameNode.address = namenode.example.com
NameNode.port = 13211
```

Name Node

The Name Node might be a little tricky to configure since it consists of multiple parts. It is a stand alone server that can not be interacted with. The steps to configure are as follows:

- The Name Node needs a MySQL server for permanent storage.
    - Set up a MySQL server, preferable locally but a external slave would be good to ensure data security and remove any single point of failure.
    - Create a user and database for mdfs on.
    - Create correct tables with `MDFS.MySQL.Table.Schema.sql` or appended one
- The Name Node will start and run when a object of mdfs.namenode.io.ConnectionListener is created.
    - However before one start the ConnectionListener it is a good idée to initiate, mdfs.namenode.repositories.MetaDataRepository, UserDataRepository and DataNodeInfoRepository. This because all three repositories sync with the MySQL server before handling any request.
    - NameNode_Bootstrap would be preferred to use.

Configuration

The configuration file at relative path mdfs/config/config.cfg are important that it is correct.

- It has to contain the dns-address, `address`, of the NameNode itself, and this one should be the same as the one that any connection attempts are made too.
- It has to contain the port, `port`, that the name node is listening to
- It has to contain the Replication ratio, `replication.ratio`. This value describes on how many location in MDFS that data should reside. In short, the number of DataNodes that should store each piece of data. 2 means that 2 Data Nodes, if available, will store each file
- It has to contain the value of verbose, `verbose`, which indicate how much information the NameNode should print to terminal when running. Range, 0-5 where 5 is the most information.
- It has contain MySQL information, `MySQL.host,MySQL.user,MySQL.pass, MySQL.db, MySQL.prefix, MySQL.update.rate`, which are connection information to the MySQL database. If a value is not used, leave blank.
- It has to contain at least one set of DataNode information composed of, `datanode.name, datanode.address, datanode.port`. This values must appear in sets in the configuration file. The datanode.name must be uniq and so must the combination of datanode.address and datanode.port be aswell.

```
NameNode Example config:
[] = optional

[name = NameNode]
[mac = 50:e5:49:39:96:85]
address = namenode.example.com
[ip = 127.0.0.1]
port = 13211
[tmpdir = /tmp/]

replication.ratio = 2

verbose = 5

MySQL.host = localhost
MySQL.user = root
MySQL.pass =
MySQL.db = mdfs
MySQL.prefix =
MySQL.update.rate = 2000


datanode.name = datanode1
datanode.address = datanode1.example.com
datanode.port = 13210

[
datanode.name = datanode2
datanode.address = datanode2.example.com
datanode.port = 13210
]
```

Data Node

The DataNode should be fairly straightforward to configure. It is a server software that dose not allow for user interaction. As the NameNode, the DataNode upon creating an object of mdfs.datanode.ConnectionListener, DataNode_Bootstap dose this.

Configuration

The configuration file at relative path mdfs/config/config.cfg are important that it is correct.

- It has to contain the dns-address of the datanode, `address`, which should be the same as the NameNode and other DataNode has as address to said DataNode.
- It has to contain the port, `port`, that the datanode should be listening to. This as well as the address has to be the same one that appears in relation to said DataNode on the NameNode or other DataNodes.
    - the combination of address and port has to be unique in in relation to other DataNodes.
- It has to contain the value of verbose, `verbose`, which indicate how much information the DataNode should print to terminal when running. Range, 0-5 where 5 is the most information.
- It has to contain at least one DataNode storage path, `DataNode.storage.path`, but can contain as many one want. Data stored on the Data Node will be divided on this paths.
    - No paths should be added nor should the order in which they appear in the config file change after the first file has been written to the DataNode.
- It has to contain the NameNode dns address and port, `NameNode.address`, `NameNode.port` , and they should be the same as the are configured on the Name Node.
- The DataNode should have a reference to it neighboring DataNodes if any, `datanode.name,datanode.address, datanode.port`. This nodes are where data will be replicated to and the values should be the same as the ones that occur on each node.

```
DataNode Example config:
[] = optional

[name = DataNode1]
[mac = 50:e5:49:39:96:85]
address = datanode1.example.com
port = 13210
[ip = 127.0.0.1]
[tmpdir = /tmp/]

verbose = 5

DataNode.storage.path = /tmp/test0
[DataNode.storage.path = /tmp/test1]

NameNode.address = namenode.example.com
NameNode.port = 13211

[
datanode.name = DataNode2
datanode.address = datanode2.example.com
datanode.port = 13210
]
```

MySQL MDFS Table Schema

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

CREATE TABLE IF NOT EXISTS `data-node` (
  `name` varchar(256) NOT NULL,
  `address` varchar(256) NOT NULL,
  `port` int(11) NOT NULL,
  PRIMARY KEY (`name`),
  UNIQUE KEY `address` (`address`,`port`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Table structure for table `meta-data`
--
CREATE TABLE IF NOT EXISTS `meta-data` (
  `filePath` varchar(256) NOT NULL,
  `size` bigint(20) DEFAULT NULL,
  `fileType` varchar(128) DEFAULT NULL,
  `storageName` varchar(256) DEFAULT NULL,
  `permission` int(11) DEFAULT NULL,
  `owner` varchar(256) DEFAULT NULL,
  `group` varchar(256) DEFAULT NULL,
  `created` timestamp NULL DEFAULT NULL,
  `lastEdited` timestamp NULL DEFAULT NULL,
  `lastTouched` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`filePath`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Table structure for table `meta-data_data-node`
--
CREATE TABLE IF NOT EXISTS `meta-data_data-node` (
  `Meta_Data_filePath` varchar(256) NOT NULL,
  `Data_Node_Name` varchar(256) NOT NULL,
  PRIMARY KEY (`Meta_Data_filePath`,`Data_Node_Name`),
  KEY `Data_Node_Name` (`Data_Node_Name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Table structure for table `user-data`
--
CREATE TABLE IF NOT EXISTS `user-data` (
  `name` varchar(256) NOT NULL,
  `pwdHash` varchar(256) DEFAULT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Constraints for table `meta-data_data-node`
--
ALTER TABLE `meta-data_data-node`
  ADD CONSTRAINT `meta@002ddata_data@002dnode_ibfk_2` FOREIGN KEY (`Data_Node_Name`) REFERENCES
`data-node` (`name`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `meta@002ddata_data@002dnode_ibfk_1` FOREIGN KEY (`Meta_Data_filePath`)
REFERENCES `meta-data` (`filePath`) ON DELETE CASCADE ON UPDATE CASCADE;
```