Shanghai Jiaotong University

Big Data Processing Technology

Project 3: Mini DFS

Author:
FEI Yixiao
118260910031
LI Yanhao
118260910036
LUO Dihao
118260910039
SHEN Shengyang
118260910042
YAN Shuhan
118260910050

Supervisor:
Chentao WU
Xin XIE

October 22, 2019

上海交大-巴黎高科卓越工程师学院
Ecole d'Ingénieurs SJTU-ParisTech

1 Introduction

Our mini-DFS (Distributed File System) is based on Java and is composed with following files:

- Main
  - Main.java : the client interface
  - Manager.java : we use this class to store and share basic information among nodes
- Node
  - DataNode.java : datanode class to perform save/read/recover functions
  - NameNode.java : namenode class to perform load/ls/split/read/fetch functions
- Tools
  - Block.java : class to store file block's information
  - FileHelper.java : class to read/write/recover file blocks
  - FileMap.java : class to store file mapping information
  - MyFile.java : class to store file information (containing several blocks)
  - Operation.java : class storing different operations including ls,put,fetch,read,quit,recover

1.1 Usage

Users can access Mini-DFS by directly running Main.java.

- ls : show list of files
- put source_file_path : upload local file to mini-DFS
- read file_ID : read the first block of required file
- fetch file_id save_path : download file from mini-DFS to local file system
- recover file_id : try to recover file block if some datanode lost file blocks
- quit : exit Mini-DFS

2 Architecture

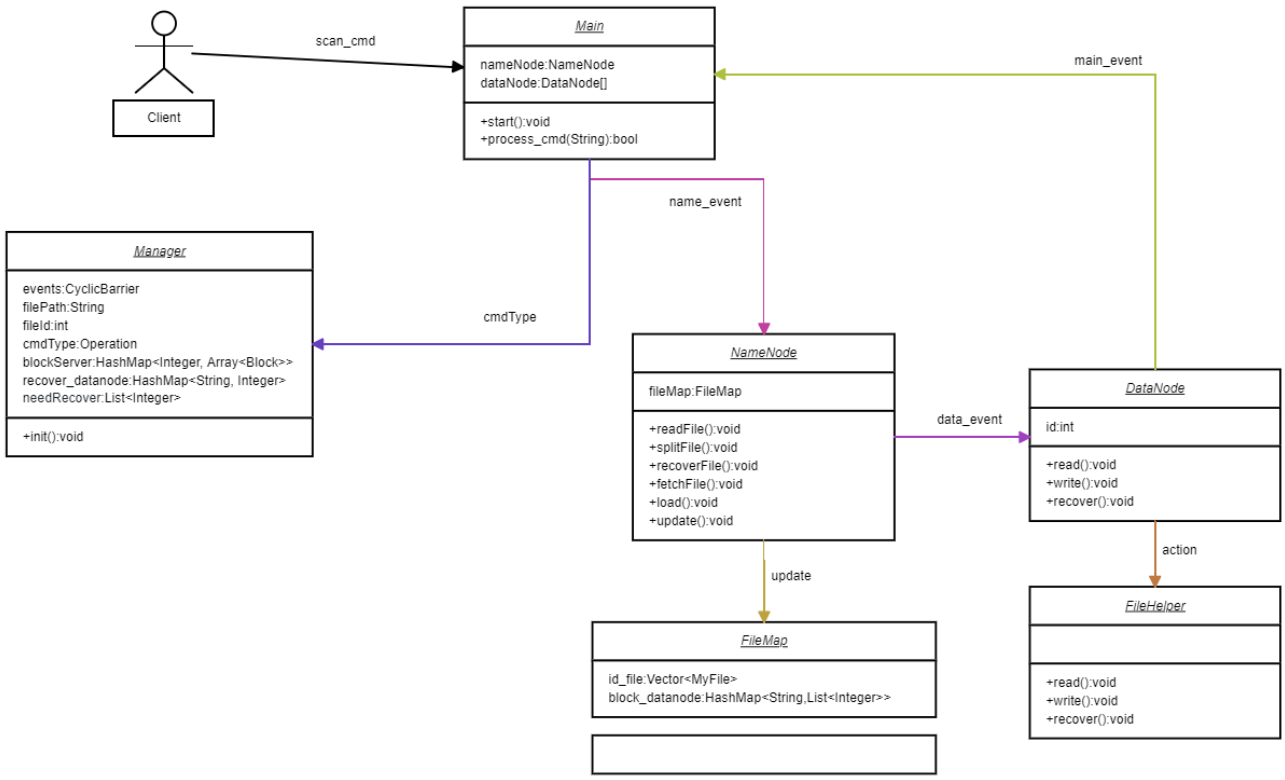

Figure 1: Architecture

As we can see from the image, when a client type some command to the interface *Main*, *Main* share this command with *NameNode* and *DataNode* through *Manager*. At the same time, we use *java.util.concurrent.CyclicBarrier name_event*, *main_event*, *data_event* to send signal from *Main* to *NameNode* and *DataNode*. So, then *Main* uses name_event.await() to notify *NameNode* to do command, *NameNode* uses data_event.await() to notify *DataNode* to do command, finally *DataNode* uses main_event.await() to notify *Main* that all process has been done and it can receive next command from client.

3 Example

As you may discover during the experiment, our Mini-DFS accepts only several commands. When the command line does not understand the command, it gives the list of the accepted commands. In addition, each command has its own format of the input. The input number of arguments might be different, so when the format is not correct, it also gives the warning message to help the client enter the correct commands.
Now first, we upload a file to Mini-DFS using *put*.

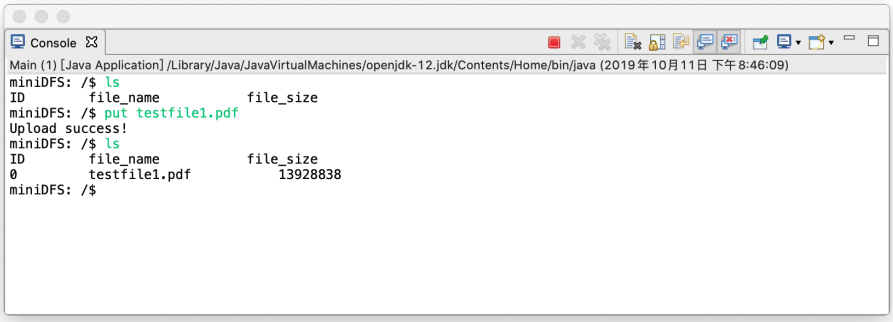

Figure 2: Usage: put

In directory *dfs*, we can see that it creates seven blocks and duplicates them for 3 times distributed uniformly among four datanodes.

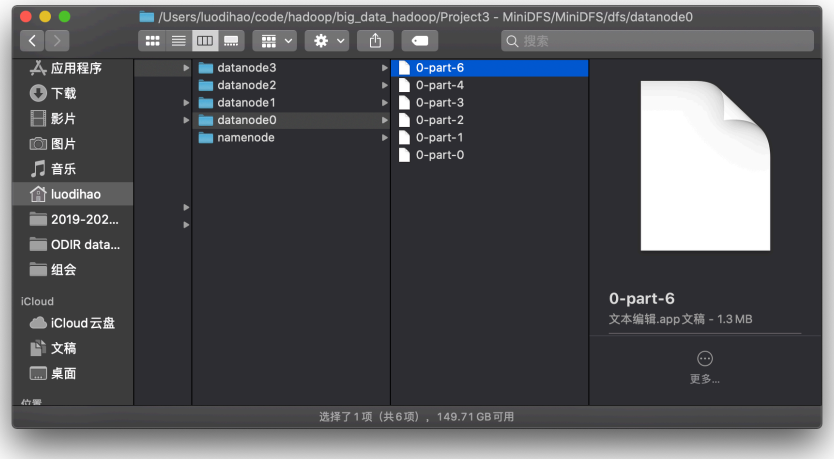

Figure 3: Usage: put

Then, we can try to read the first block of this file. Since it's of format pdf, it seems to be unreadable strings, but it works if this is a text file.

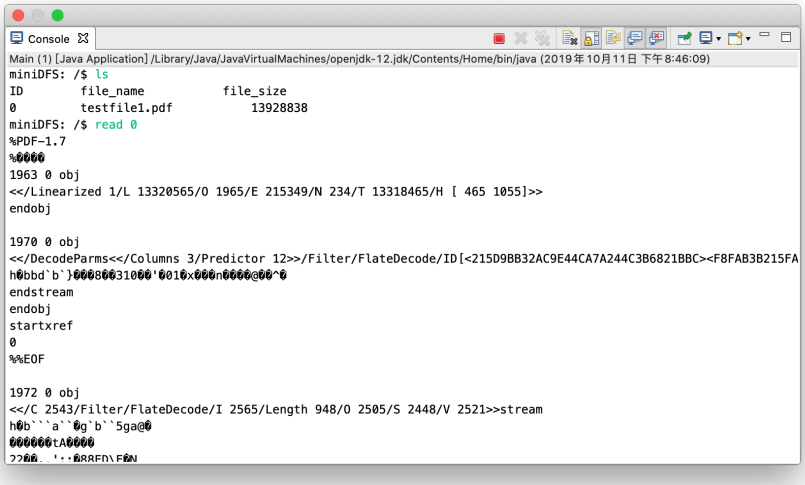

Figure 4: Usage: read

Next, we download this file to local file system: directory *./dfs* . This could be any directory on the client side. We choose *./dfs* just because it is easier to demonstrate the changes on the disk.

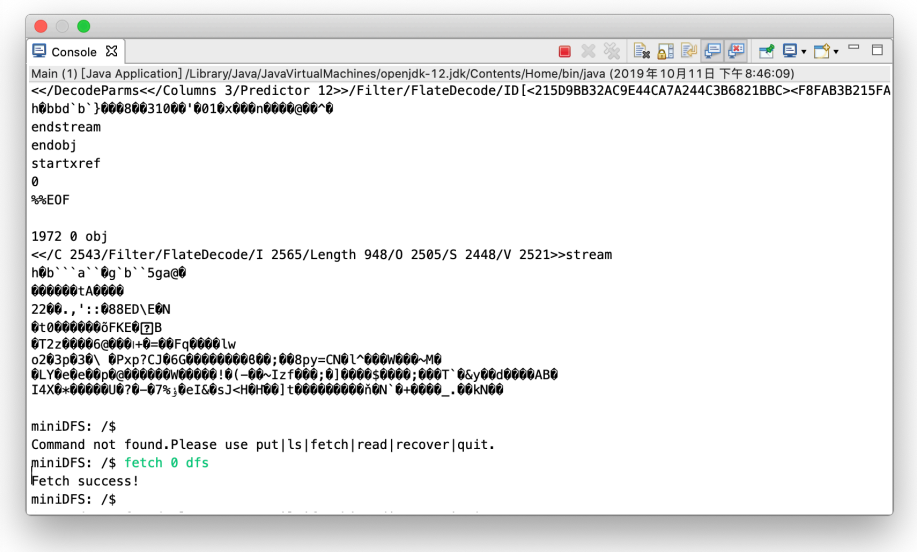

Figure 5: Usage: fetch

Well, it appears in the file system. The file is succefully downloaded from the Mini-DFS. As you could probably see from the picture, this is an e-book about body training.

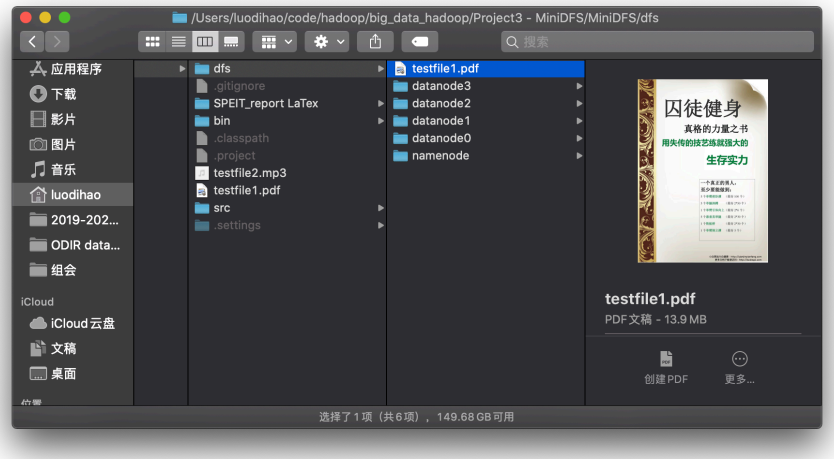

Figure 6: Usage: fetch

Then, we delete *datanode0* and try to recover it. Remember, there should

be at least one replica of the *datanode0* in Mini-DFS.

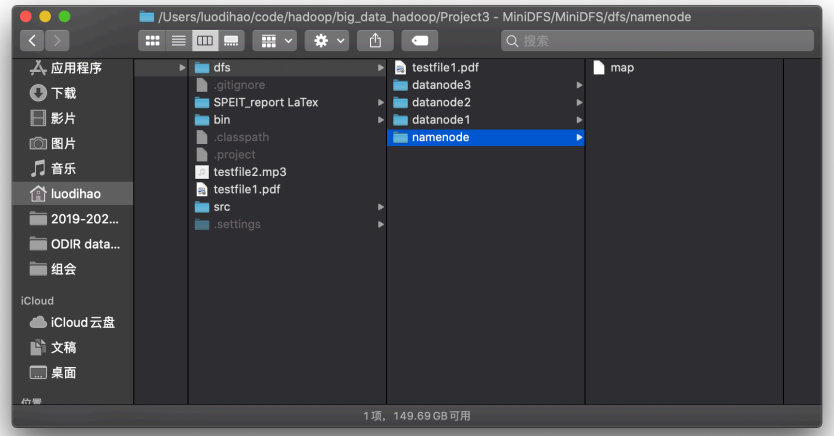

Figure 7: Usage: recover

Using *recover 0*. The command will check the completency of the file 0. Since Mini-DFS has the list of the server which should store the distributed data nodes, it could safely check whether the corresponding data nodes still exist. If this is not the case, it trys to find one replica to recover it.

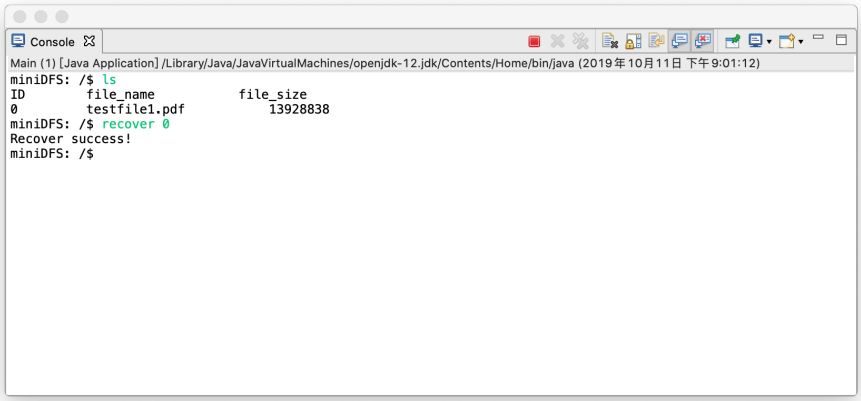

Figure 8: Usage: recover

It appears again in directory *./dfs/* . The recovery will fail if there is no at least one replica, which makes sense.

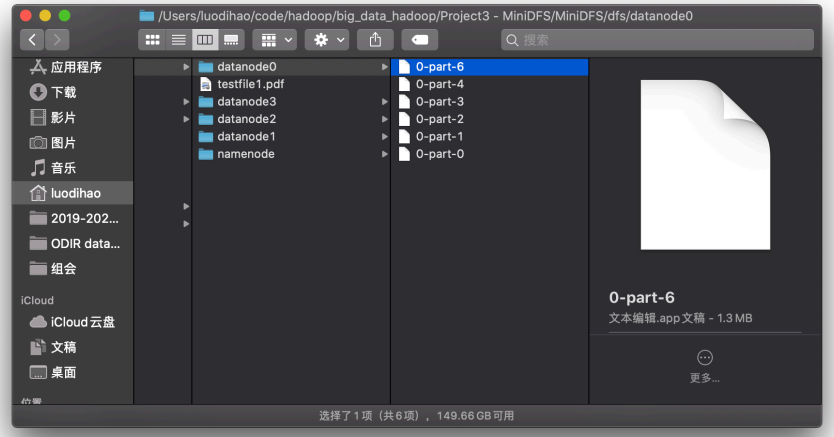

Figure 9: Usage: recover

Finally, we can safely exit Mini-DFS and this completes our examples of the experiment.

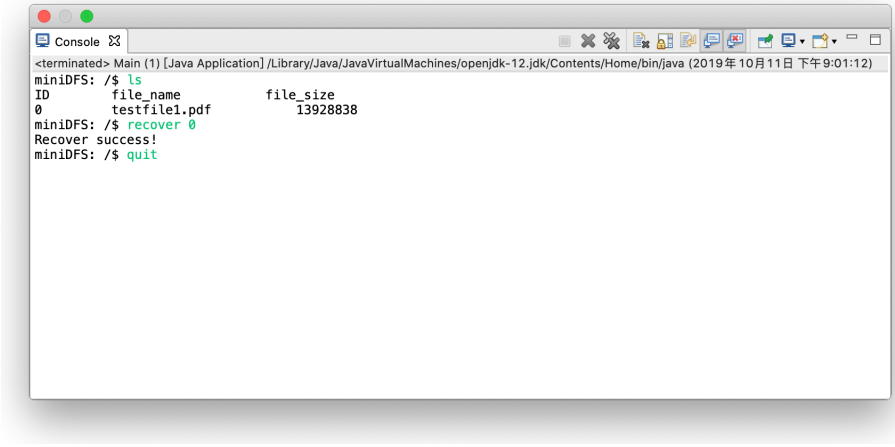

Figure 10: Usage: quit