

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 21, 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

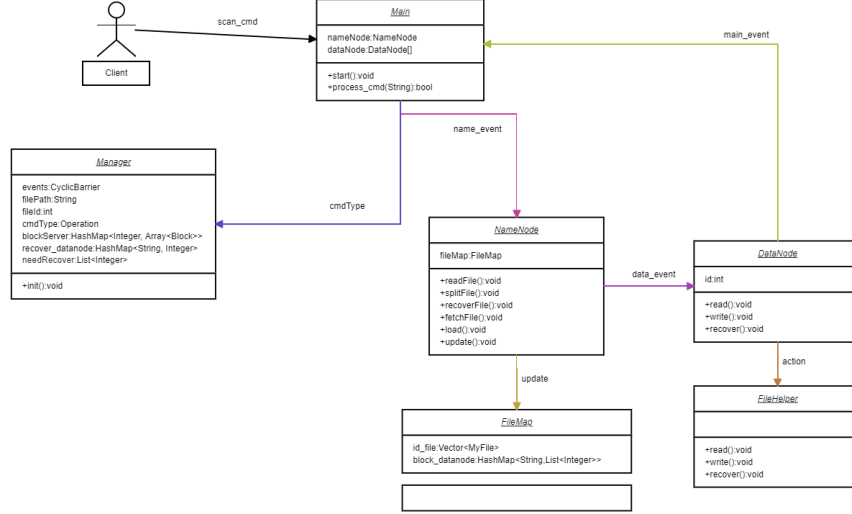


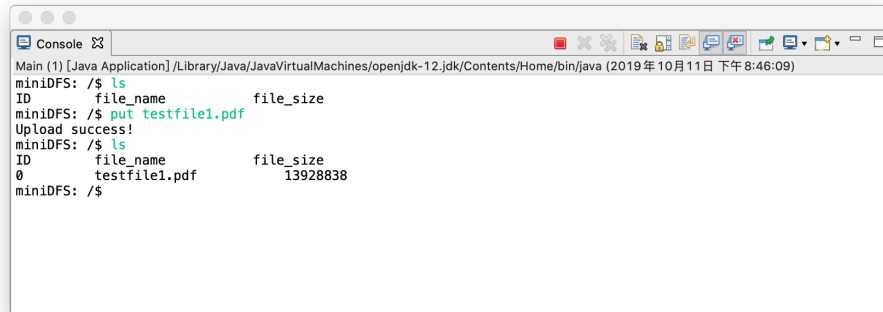
Figure 1: Architecture

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*.



```

Main (1) [Java Application] /Library/Java/JavaVirtualMachines/openjdk-12.jdk/Contents/Home/bin/java (2019年10月11日 下午8:46:09)
miniDFS: /$ ls
miniDFS: /$ put testfile1.pdf
Upload success!
miniDFS: /$ ls
ID      file_name      file_size
0       testfile1.pdf  13928838
miniDFS: /$

```

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.

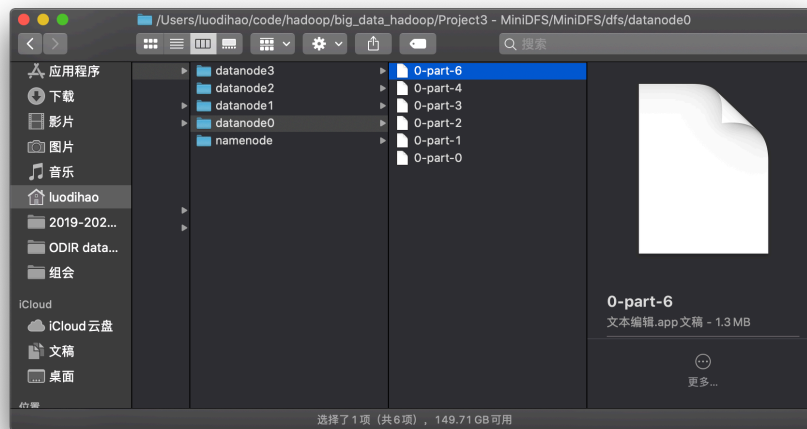
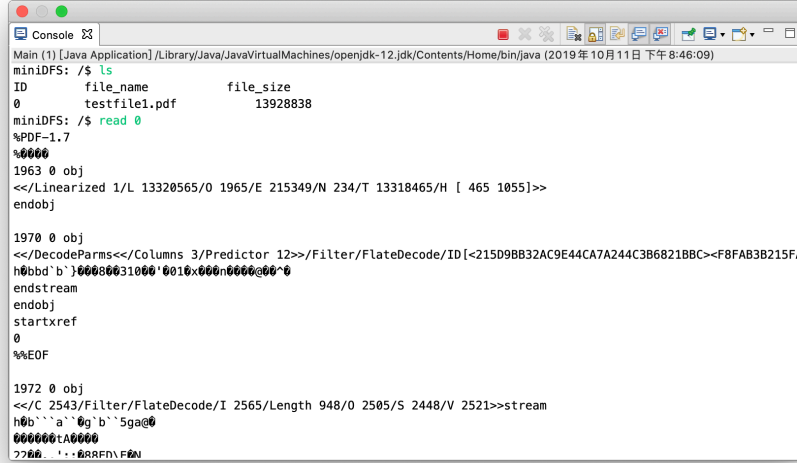


Figure 3: Usage: put

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.



```

Main (1) [Java Application] /Library/Java/JavaVirtualMachines/openjdk-12.jdk/Contents/Home/bin/java (2019年10月11日 下午8:46:09)
miniDFS: /$ ls
ID      file_name      file_size
0       testfile1.pdf     13928838
miniDFS: /$ read 0
%PDF-1.7
%0000
1963 0 obj
<</Linearized 1/L 13320565/O 1965/E 215349/N 234/T 13318465/H [ 465 1055]>>
endobj

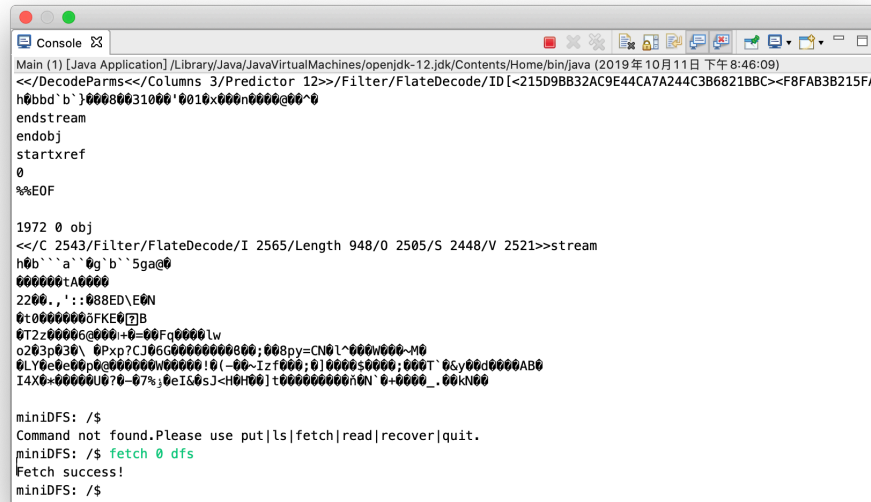
1970 0 obj
<</DecodeParms<</Columns 3/Predictor 12>>/Filter/FlateDecode/ID[<215D9BB32AC9E44CA7A244C3B6821B8C><F8FAB3B215FA
h0bbd`b`>00000031000'0010x000n00000000
endstream
endobj
startxref
0
%%EOF

1972 0 obj
<</C 2543/Filter/FlateDecode/I 2565/Length 948/O 2505/S 2448/V 2521>>stream
h0b` ``a` `0g`b` `5ga@0
000000tA0000
??00. .':08RFD\ F0N

```

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.



```

Main (1) [Java Application] /Library/Java/JavaVirtualMachines/openjdk-12.jdk/Contents/Home/bin/java (2019年10月11日 下午8:46:09)
<</DecodeParms<</Columns 3/Predictor 12>>/Filter/FlateDecode/ID[<215D9BB32AC9E44CA7A244C3B6821B8C><F8FAB3B215FA
h0bbd`b`>00000031000'0010x000n00000000
endstream
endobj
startxref
0
%%EOF

1972 0 obj
<</C 2543/Filter/FlateDecode/I 2565/Length 948/O 2505/S 2448/V 2521>>stream
h0b` ``a` `0g`b` `5ga@0
000000tA0000
2200. .':08ED\ E0N
0t0000000fKE0[7]B
0T2z000060000+0=00Fq0000lw
o203p030\ 0Pxp?CJ06G0000000000;008py=CN0l^000n000~M0
0LY0e0e00p0000000n00000!0(-00~Izf000;0)00005000;000T`00y00d0000AB0
I4X0+00000070-07%;0eIC0sJ<H0:00)t000000000n0N`0+0000_.00kN00

miniDFS: /$
Command not found.Please use put|ls|fetch|read|recover|quit.
miniDFS: /$ fetch 0 dfs
Fetch success!
miniDFS: /$

```

Figure 5: Usage: fetch

Well, it appears in the file system. The file is successfully downloaded from

the Mini-DFS. As you could probably see from the picture, this is an e-book about body training.

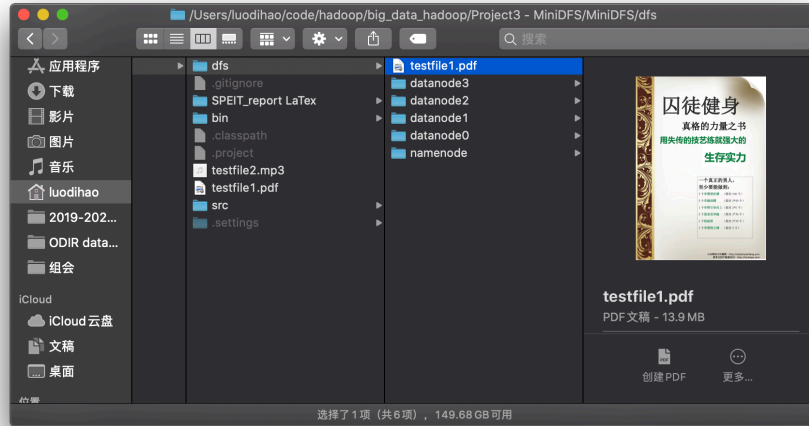


Figure 6: Usage: fetch

Then, we delete *datanode0* and try to recover it. Remember, there should be at least one replica of the *datanode0* in Mini-DFS.

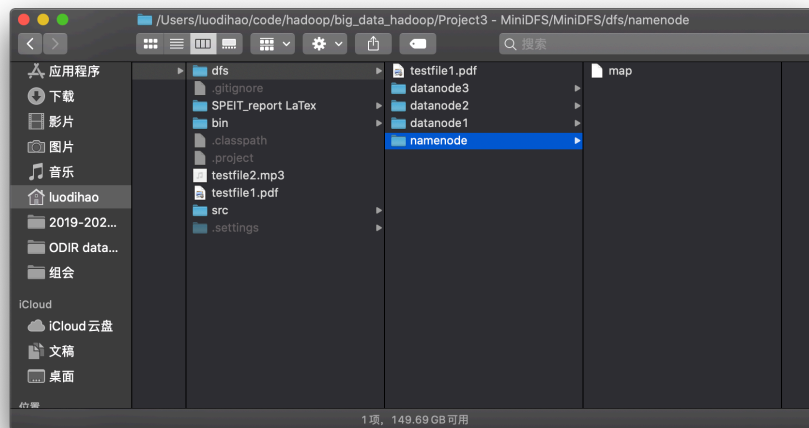
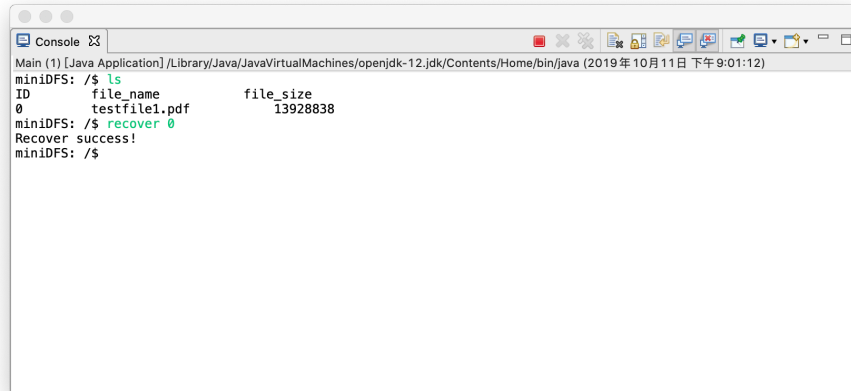


Figure 7: Usage: recover

Using *recover 0*. The command will check the competency 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 tries to find one replica to recover it.



```

Main (1) [Java Application] /Library/Java/JavaVirtualMachines/openjdk-12.jdk/Contents/Home/bin/java (2019年10月11日 下午9:01:12)
miniDFS: /$ ls
ID      file_name      file_size
0       testfile1.pdf  13928838
miniDFS: /$ recover 0
Recover success!
miniDFS: /$

```

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.

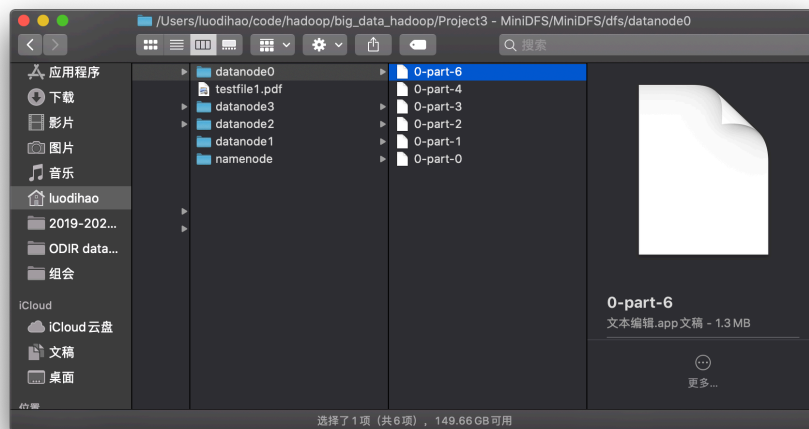


Figure 9: Usage: recover

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



```
<terminated> Main (1) [Java Application] /Library/Java/JavaVirtualMachines/openjdk-12.jdk/Contents/Home/bin/java (2019年10月11日 下午9:01:12)
min1DFS: /$ ls
ID      file_name      file_size
0       testfile1.pdf   13928838
min1DFS: /$ recover 0
Recover success!
min1DFS: /$ quit
```

Figure 10: Usage: quit