

SHANGHAI JIAOTONG UNIVERSITY

BIG DATA PROCESSING TECHNOLOGY

Project 3: Mini DFS

Author:

LUO Dihao

118260910039

YAN Shuhan

118260910050

SHEN Shengyang

118260910042

Supervisor:

Patrick BERBON

October 11, 2019



1 Introduction

Our mini-DFS (Distributed File System) is based on Java and is composed by 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

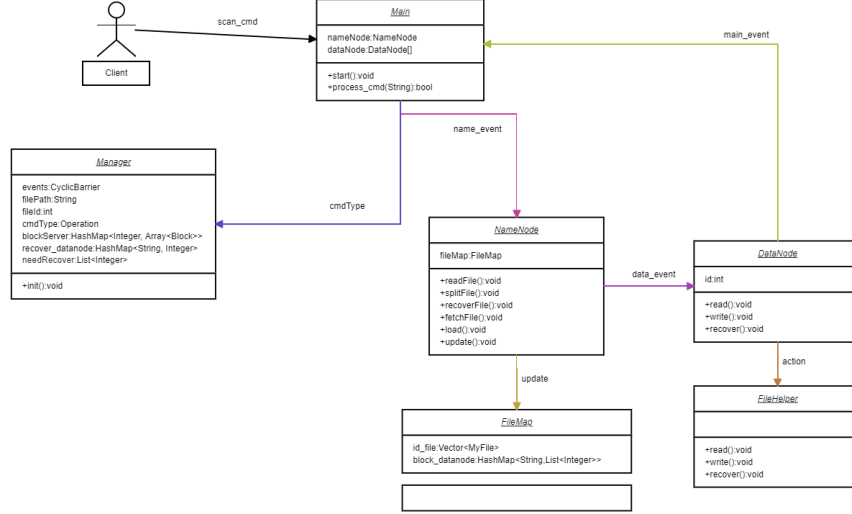


Figure 1: Architecture

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

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 it creates seven blocks distributed uniformly among four datanodes.

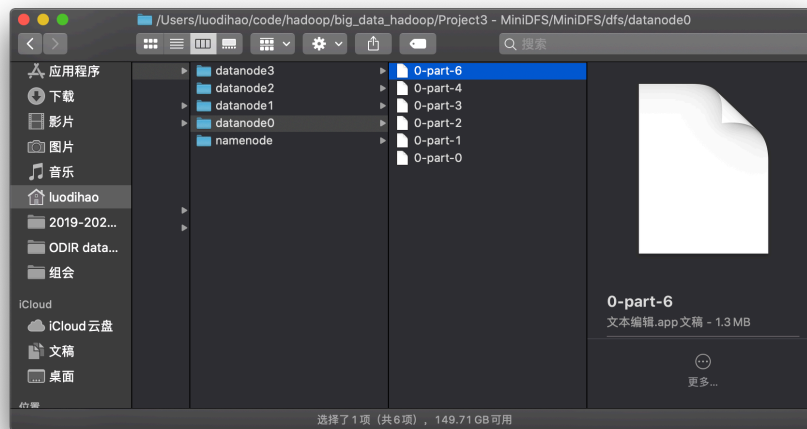


Figure 3: Usage: put

Then, we read the first block of this file. Since it's of format pdf, it contains mostly unreadable strings.

```

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[<215D9BB32AC9E44CA7A244C3B6821BBC><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.``.88RFN\F0N

```

Figure 4: Usage: read

Next, we download this file to local file system: directory `./dfs`

```

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[<215D9BB32AC9E44CA7A244C3B6821BBC><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.,':088ED\E0N
0t00000000FKE0[0]B
0T2z0000060000+0=00Fq0000lw
o203p030\0Pxp?CJ06G0000000000;008py=CN0l~000n000~M0
0LYe0e00p00000000000!0(-00~Izf000;0)000050000;000T`0Sy00d0000AB0
I4X0*00000U070-07%;0eIS0sJ<H0H00)t000000000n0N`0+000_.00kN00

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

```

Figure 5: Usage: fetch

It appears in the file system.

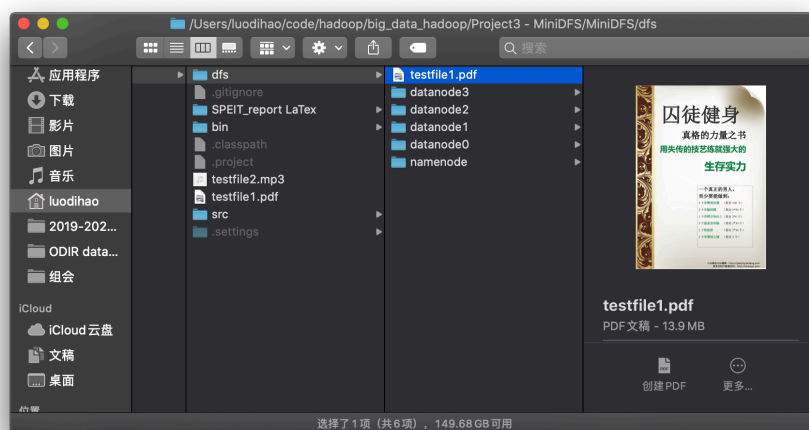


Figure 6: Usage: fetch

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

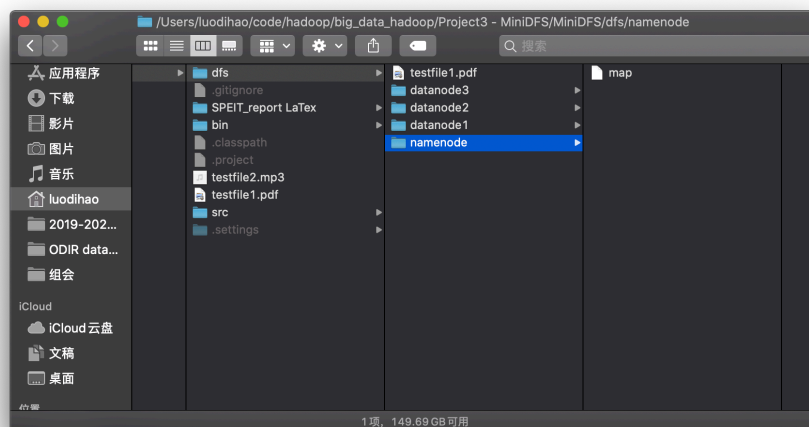
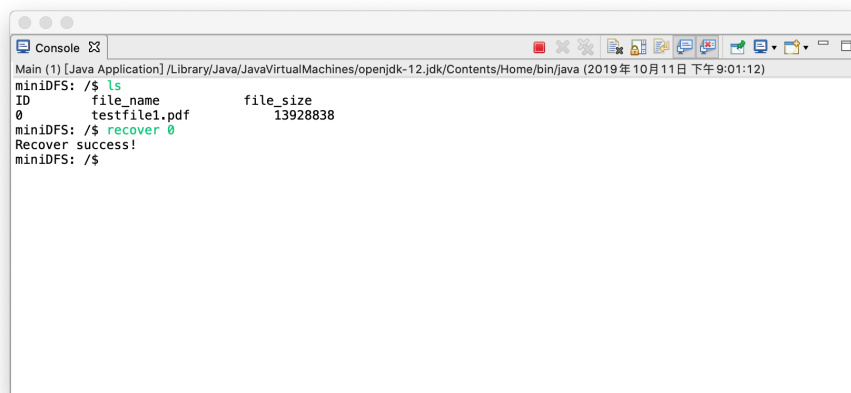


Figure 7: Usage: recover

Using *recover 0*.



```
Console
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/`

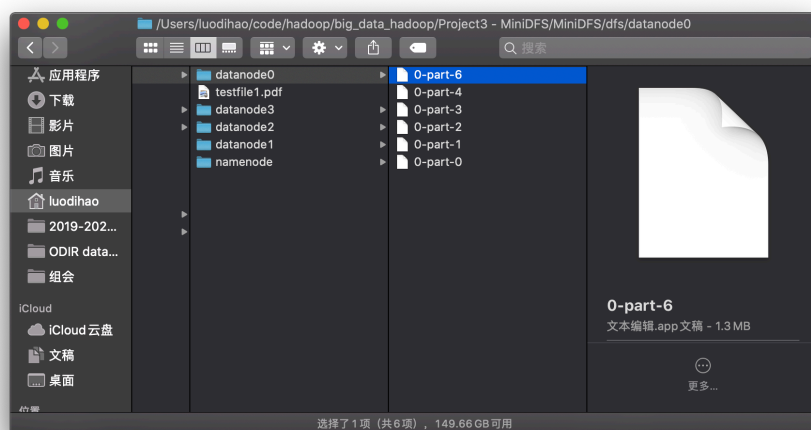
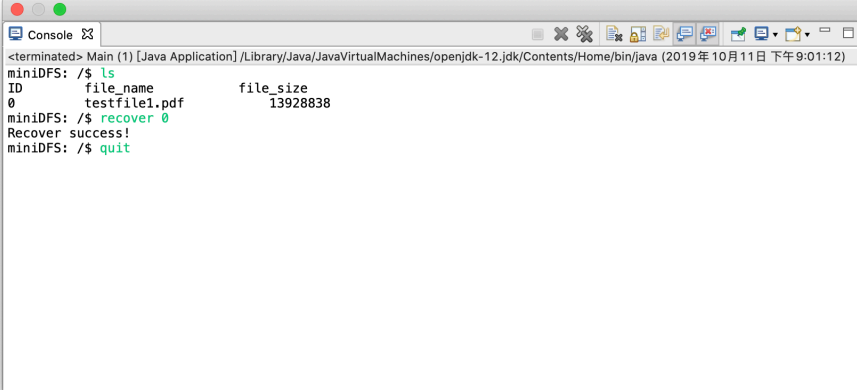


Figure 9: Usage: recover

Finally, we exit Mini-DFS.



```
<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)
minIDFS: /$ ls
ID      file_name      file_size
0       testfile1.pdf   13928838
minIDFS: /$ recover 0
Recover success!
minIDFS: /$ quit
```

Figure 10: Usage: quit