Computer Systems Engineering Lab1

Simple Distributed File System Part 1

Release: September 21, 2016

First Phase Deadline: **Sept 29, 2016 24:00**

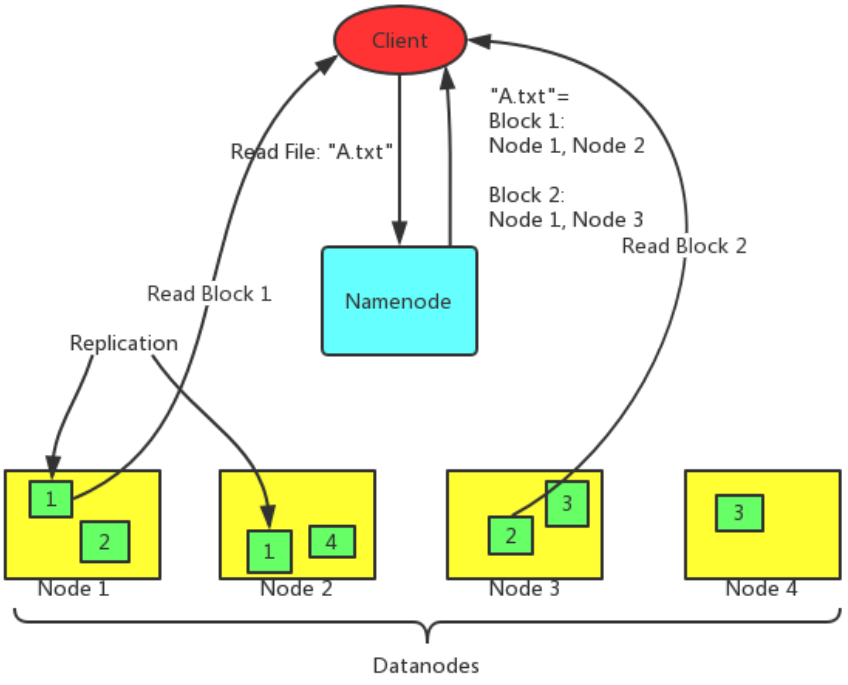
Final Deadline: **October 11, 2016 24:00**

TA: 叶政擎13302010074@fudan.edu.cn 傅鹏程13302010036@fudan.edu.cn

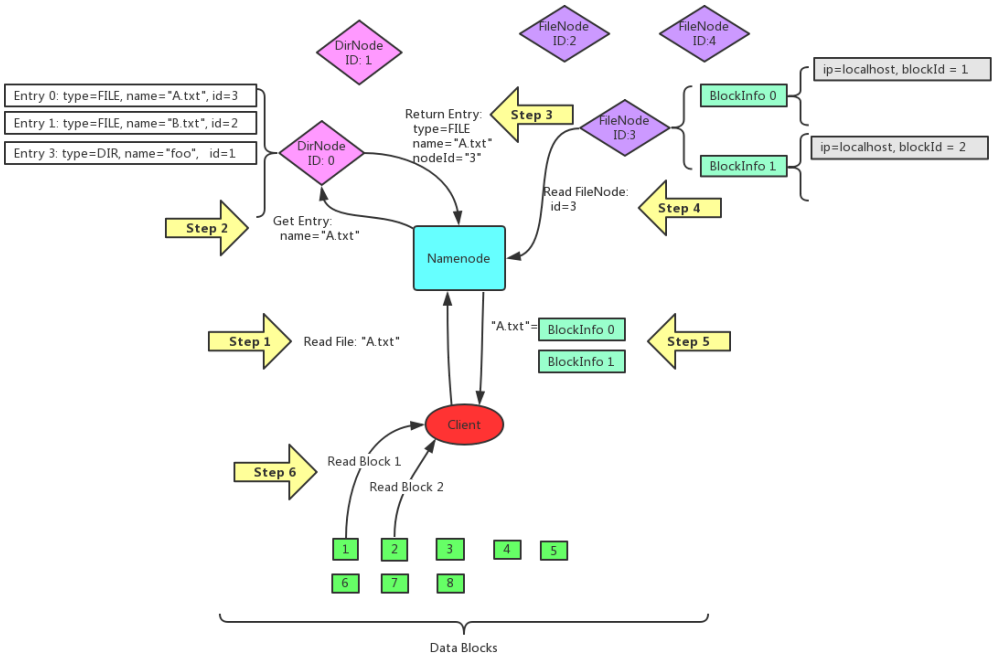
If you have any question, don't hesitate to contact us.

Introduction

> In this semester, you are required to build a simple distributed file system. The following picture show the basic concept of our Simple Distributed File System(SDFS):



> In this part, you are required to build a application layer file system running on a single machine. The following picture show how it work step by step:



> This program is a simplified version of HDFS. Please read about HDFS design before you start the program.

Architecture

For forward compatibility, you program should be divided into client, namenode and datanode.

In the future, they will be put on different machine.

For now, they will be compiled into one program.

Client

Client is the consumer of SDFS. Your client should support following function:

1. open

> Open a file in SDFS. Return a input stream to read data.

2. create

> Create a file in SDFS. Return a output stream to write data.

3. mkdir

> Create a directory.

NameNode

The responsibility for namenode is to store the directory information and tell client which blocks should be used to put or get the file contents.

In this lab, each block should be 64K. If a file is less than 64K, it should occupy one block(not necessary as large as 64K) to store the file. If a file is 65K, it should occupy 2 blocks. etc.

In namenode, you should implement following functions:

1. open

> Open a file in SDFS. Return file metadata to the client.

2. create

> Create a file in SDFS. Return file metadata to the client
>
> FileNode represent a file in SDFS. It contains blockInfos. Each blockInfo store the replication info for each block of the file. BlockInfo will store the info which datanode and which blockId currently store the data.
>
> e.g. From the design pictures, A.txt occupies two blocks, thus it contains two blockInfo. From blockInfo 1, we know that the first block of A.txt will store in localhost, blockId is 1. In the future, it may contain more than one duplicate of this block. So we may continue add LocatedBlock(10.132.141.33,2) to blockInfo 0.
>
> For FileNode, you should serialize FileNode and write them as an individual file.

3. mkdir

> Create a directory in namenode. Directory information should be stored with the namenode server. Each directory should be store in an individual file. You should use class DirNode to represent the node of the directory.

> For DirNode, each entry represent its subdirectory or file. The first byte is the entry type (file=0, dir=1). Then the following 3 bytes is file name length. The following is filename. The last 4 bytes store its node id. Also the node id should generate increasingly.
>
> The store file name should be [NodeID].node.
> e.g. 0.node 1.node
> You could use File.exist() to judge a block is free or not.
>
> The root DirNode id should always be 0. And it should be automatic created if it is not exist.

4. close

> Close a opened file. When file is close, all metadata should be write to disk.

5. addBlock

> Require a new block for the file. Return a available datanode and block information to the client. Currently the datanode info should be ignored. And the block number should be generated increasingly, which start from zero and then find the smallest free block. e.g. 0 1 2.

DataNode

The datanode should accept read/write request from client.

Each block should be store on a individual file on working directory.

The filename should be naming as [block number].block

> e.g. 1.block 2.block

Remember that each block except the last block of a file should be exactly 64K.

Currently it should impletement following functions:

1. read

> Read data from file and return to client.

2. write

> Write data to a file.

Combine three part

You should also finish some support code in order to make the program work. You could create some main function or test case to ensure your code will work. But do not include the test code when you submit your lab.

Shell or Unit Test (Optional)

You could build a jar that supported command line or use unit test to ensure your code are correctly work.
If you are convinced that you code would be 100% sure correct, those steps could be omit.

> In following command, the file on SDFS should be sdfs://[ip]:[port]/[dir]/.../[filename]
> For now, port could be omitted.

The command are supported to accept following arguments:

1. java -jar SDFS.jar put
   > copy a local file to sdfs
2. java -jar SDFS.jar get
   > copy a file on sdfs to local
3. java -jar SDFS.jar mkdir
   > mkdir on sdfs

First Phase Document

Your first phase document of this project should include following part:

1. The architecture of your design.
2. Describe the execution flow of open, create, mkdir step by step in ISimpleDistributedFileSystem.
3. Describe the execution flow of read, close step by step in SDFSInputStream.
4. Describe the execution flow of write, close step by step in SDFSOutputStream.
5. Describe the difficulty you met or you will meet.
6. Describe the bottleneck in this design.

Save it as [YourStudentID]_Document-1.pdf
Please submit [YourStudentID]_Document-1.pdf to ftp://10.132.141.33/classes/14/161 计算机系统工程 冯红伟/WORK_UPLOAD/Lab1/Phase1 before **Sept 29, 2016 24:00**

Final Document

Your document of this project should include following part:

1. Describe what you have done.
2. Describe the problem you met during developing and how you solve it.
3. Extra work you have done. That will be the bonus.

Save this document as [YourStudentID]_Document-2.pdf

Submit file

> [YourStudentID] //the parent directory
> |-[YourStudentID]_Document-1.pdf
> |-[YourStudentID]_Document-2.pdf
> |-src //the directory contains all your source code
> |-[YourStudentID].sdfs-1.jar //The compile version of your code. If your code could not be compile due to lack of main, create a empty one.

Please archive all your file to a zip file [YourStudentID].zip

Please submit [YourStudentID].zip to ftp://10.132.141.33/classes/14/161 计算机系统工程 冯红伟/WORK_UPLOAD/Lab1/Phase2 before **October 9, 2016 24:00**

Make sure all your file are correctly placed. The score will be judged by an automatic test running in a docker machine with java 8.

Grade policy

Document: 35%, Code design 25%, A correct program 30%, Bonus 10%
Any cheat behaviors(e.g. Copy code/design or code are too similar to any others/Internet) will get zero in all grading part

You should submit all the code to ftp on time. After deadline, each day the total grade will be cut down 10%. e.g. if you summit on Oct 10, 2016, and if your lab should be graded 95, then you will only get 95*0.9=85.5 and if you submit on Oct 11, 2016, you will only get 95*0.8=76. Hence, start early.