Computer Systems Engineering Lab2

Simple Distributed File System Part 2

Release: Oct 19, 2016

First Checkpoint: **Oct 25, 2016 24:00**

Deadline: **Nov 2, 2016 24:00**

Introduction

In the lab1, you have already finished a part of SDFS. Now, we want to make it an actually C/S architecture program.

Target

You should finish following parts:

RPC

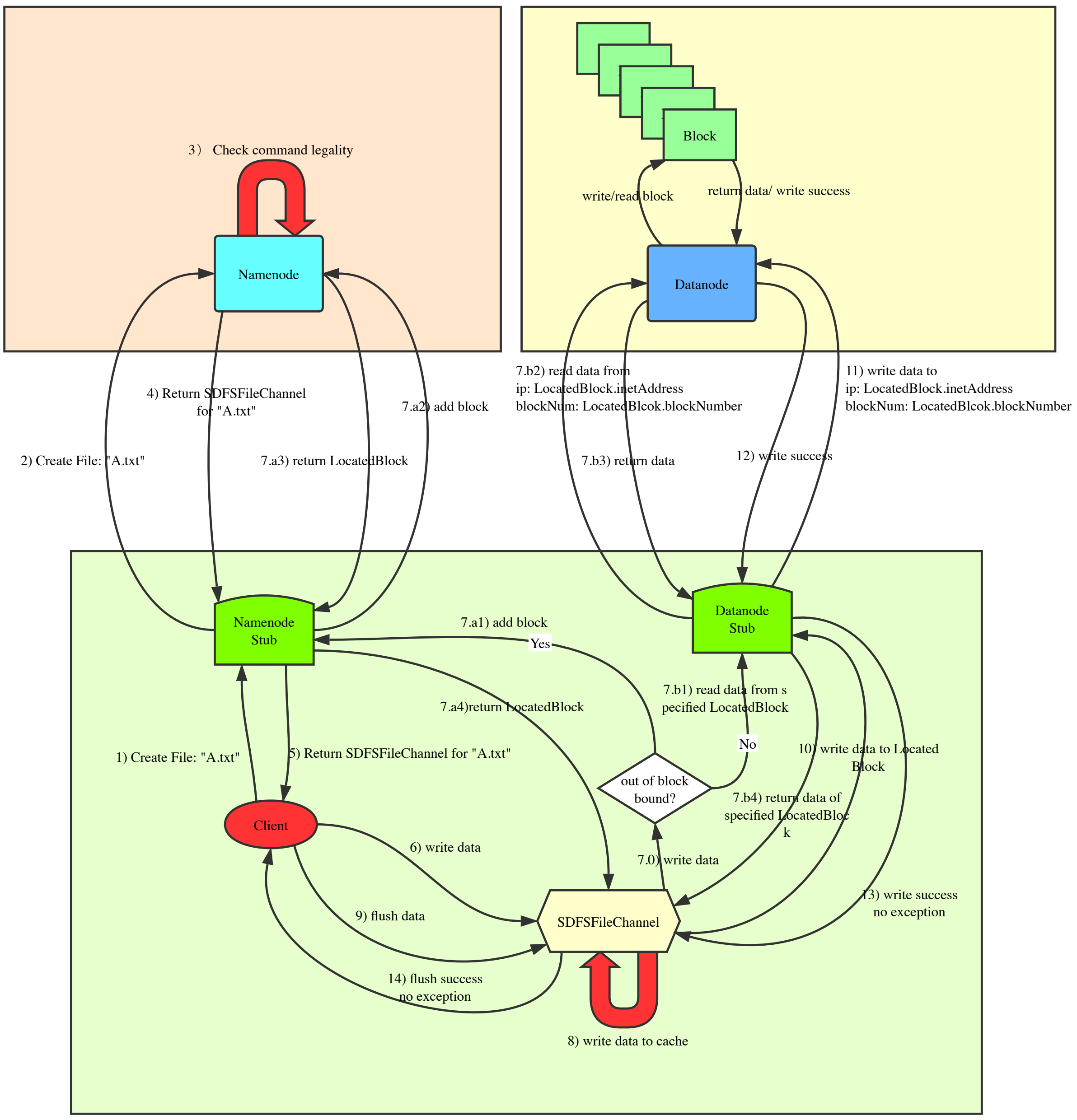
In this lab, you are required to implement a simple RPC framework by your self.

RPC frameworks, including RMI, gRPC, Thrift, etc. are NOT allowed in this lab

You should implement client stub and RPC server by yourself. Any kind of implements without code copy and without any framework are acceptable.

Suggestion: Use Socket, and send/receive custom serializable class to communication between stub and remote server.

RPC example for write to a file


3） Check command legality
Namenode
Block
write/read block
return data/ write success
Datanode
4) Return SDFSFileChannel for "A.txt"
7.a2) add block
2) Create File: "A.txt"
7.a3) return LocatedBlock
7.b2) read data from ip: LocatedBlock.inetAddress blockNum: LocatedBlcok.blockNumber
11) write data to ip: LocatedBlock.inetAddress blockNum: LocatedBlcok.blockNumber
7.b3) return data
12) write success
Namenode Stub
Datanode Stub
7.a1) add block
Yes
7.a4)return LocatedBlock
7.b1) read data from specified LocatedBlock
No
1) Create File: "A.txt"
5) Return SDFSFileChannel for "A.txt"
out of block bound?
10) write data to Located Block
7.b4) return data of specified LocatedBlock
Client
6) write data
7.0) write data
9) flush data
SDFSFileChannel
13) write success no exception
14) flush success no exception
8) write data to cache

> client->create
> NameNodeStub->create
> NameNodeServer->onRpcReceive
> NameNode->create return SDFSFileChannel with readwrite permission
> NameNodeServer return SDFSFileChannel
> NameNodeStub return SDFSFileChannel
>
> SDFSFileChannel->write block not enough
> NameNodeStub->addBlock
> NameNodeServer->onRpcReceive
> NameNode->addBlock return LocatedBlock
> NameNodeServer return LocatedBlock
> NameNodeStub return LocatedBlock
> SDFSFileChannel addBlock to cache
> SDFSFileChannel continue to write data on local cache

> SDFSFileChannel->close
> SDFSFileChannel->flush
> DataNodeStub->write
> DataNodeServer->onRpcReceive
> DataNode->write to disk and return result
> DataNodeServer return result
> DataNodeStub return result
> SDFSFileChannel close successful

Cache

You are required to implement read/write cache for data block on the datanode.

All file metadata should be cache on client when client open a file.

And it should contain at most *fileDataBlockCacheSize* of data block cache when read of write of a file. It should implement LRU algorithm when replace the cache.

Multi client read/write problem (put off to future lab due to its difficulties)

For now you don't need to multi client read/write problem. It will make yourself convenient if you consider its forward compatibility.

Following is original requirement:

> You should consider how to solve read write conflict between multi client. My suggestion is use uuid to tag each file access. When client open a file, the name node server should return a uuid to identification this access. And each access could be readonly access or readwrite access. So there will be following situations:
>
> 1. Multi read access to the same file: It should always success.
> 2. Multi write access to the same file: Only the first write access success. The following access will be denied.
> 3. Multi read access and one write access: Before write access close, all read access, including read request after write access begin, should refer to the version of file before modified. You should implement copy-on-write technique to ensure both read and write access will work correctly. When write access close, file metadata should be stored, but any read access should also refer the version of file before modified. Only when all read access of old version of file

> close could the name node remove the old file metadata from its memory.
>
> Also, you could have your own design to solve multi client read/write problem.

Other tips

1. Since you are familiar with inode, from this lab, you are allowed to store the DirNode directly to the disk. That is to say, you could serialise the entire file tree to disk. It depends on you to use the old design or the new design.
2. Some class are resigned in order to support copy-on-write technique.
3. Overwrite and append as well as truncate of existing file are allowed now. If a block occupy less then BLOCK_SIZE in dataNode except the last block, then it should fill zero to rest of this block. It could be written to disk or reflected just in the memory.

Document

Basically, your document should include following:

1. your design of RPC
2. your design of Cache
3. ~~your Read/Write problem solution~~
4. Describe the problem you met during developing and how you solve it.
5. Describe the change of the architecture of yours(if possible)
6. Extra work you have done.

First Checkpoint Submission

You should finish RPC part and summit your code.

Your namenode.jar and datanode.jar should be executable. It doesn't matter client.jar could be executable or not.

Submit file tree:

> [YourStudentID] //the parent directory
> |-[YourStudentID]_Document-1.pdf
> |-src //the directory contains all your source code
> |-[YourStudentID].sdfs-2.1.client.jar |-[YourStudentID].sdfs-2.1.namenode.jar |-[YourStudentID].sdfs-2.1.datanode.jar

The document should contains the work you have done at that point.

Please archive it as [YourStudentID]-1.zip or [YourStudentID]-1.tar.gz

Final Submission

You should finish all part and summit your code.

Submit file tree:

> [YourStudentID] //the parent directory
> |-[YourStudentID]_Document-1.pdf
> |-[YourStudentID]_Document-2.pdf
> |-src //the directory contains all your source code
> |-[YourStudentID].sdfs-2.2.client.jar |-[YourStudentID].sdfs-2.2.namenode.jar |-[YourStudentID].sdfs-2.2.datanode.jar

Please archive it as [YourStudentID]-2.zip or [YourStudentID]-2.tar.gz

Grade policy

Document: 40, Code design 30, A correct program 30, Bonus 20
Any cheat behaviours(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%. Hence, start early.