CMSC 621 Final Report

---------KYSS File System

Kejian Hu
Youyong Zou
Shanshan Liu
Shuang Zeng

1. Goal

Our objective is to implement a simple distributed file system based on three tasks: basic bones task, optimistic replication and fileserver discovery.

1.1 Basic Bones Task:

There will be several file servers in the system and normal operations related to file system, such as creating, deleting, opening, closing, reading, writing, will be allowed. And the file server which the client is connected to will process these requests. We call it local file server for this client. For every file, there is a owner file server which save the primary copy.

Other file servers who want to create a replica will ask the owner file server with a lease time. The modification returned to the owner file server within the lease time should be committed by the owner file server. If it is over lease time, the returned file will be dropped. Leasing of files is based on FIFO process and any other lease of the file which has already been leased out will be blocked. Lease time can be extended, owner file server will decide whether leasetime extend success or not.

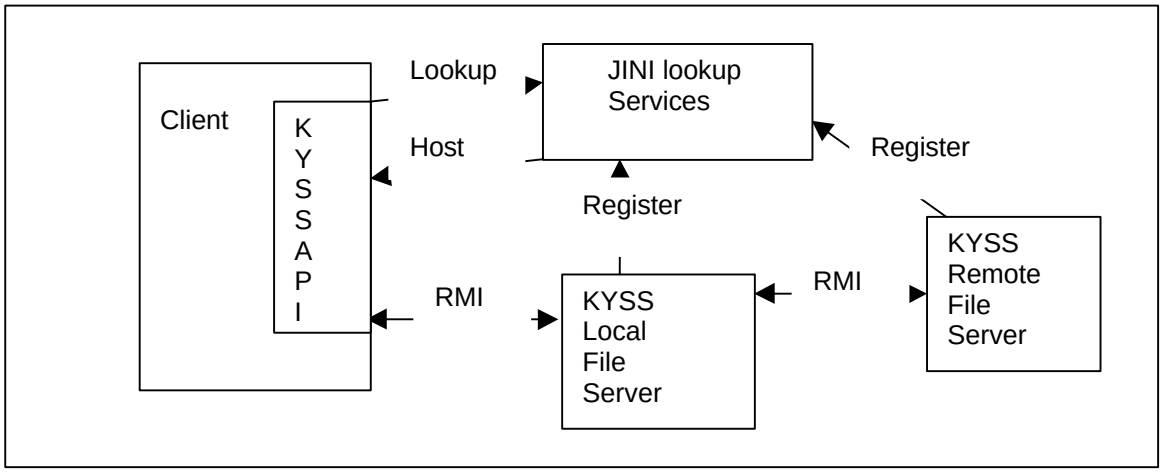
1.2 Optimistic Replication:

The idea here is to extend the bare bone system to permit multiple servers to simultaneously operate on replicas in a manner which will change them (e.g. concurrent writing). When a file server ask the owner file server for a file, owner file system will get the newest file from latest replica, and return it to the file server. When a file server finish operation, it will return the modified file to owner file server, and update all replica.

1.3 Fileserver Discovery:

In bare bone task, every client have a default local file server. Here ,we use jini services to lookup a file server for client. This would be especially useful for mobile system. So the client could move and reconnect at any arbitrary network.

2. General ideas



2.1 RMI

Here we use RMI to communicate among file servers and between file server and client. The RMI remote object in KYSS is UnicastRemoteObject, rather than activatable object, because in KYSS system, each file server has only one registered service, keep the services process running will not influence system performance. The benefit of using RMI is simple, we can run a class from other machine. If performance is the most important, we should use socket directly.

2.2 Cache

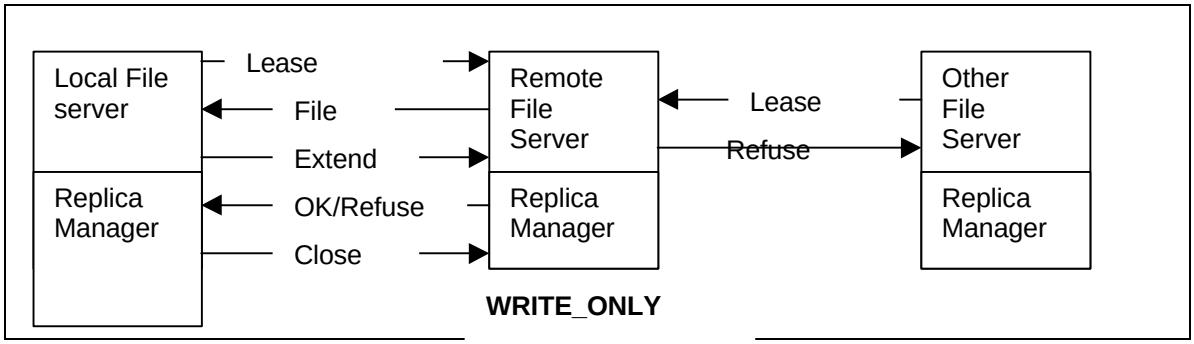
Cache is saved in DISK, rather than save in memory.
When client want to operate a file, if the file is not in local file server, the local file server will contact remote file server which own this file and create a replica locally. All operations will proceed only on local replica.

We use following cache consistency strategy in KYSS:
Server-Initiated: when a replica finished update operation, it will return newest file to owner file server. The owner file server will inform the file servers which is writing the same file that the replica is invalid, and upload the newest file.
Client-Initiated: when a file server want to operate a file and find the file already in its cache, it will check the cached file's timestamp with owner file server's timestamp, if it is the same, operate on local replica, else update replica file.

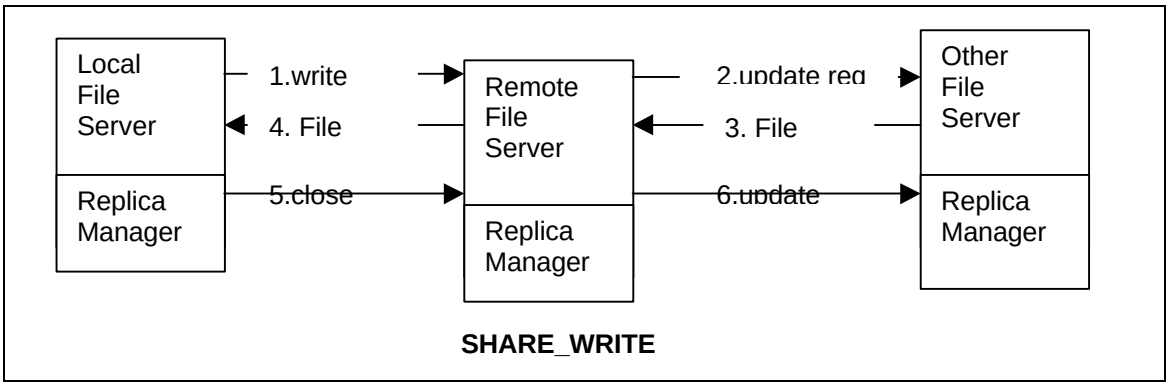
2.3 policy

Writing policy: two kinds of write mode:WRITE_ONLY and SHARE_WR.
WRITE_ONLY (Bare bones Task) is exclusive write operation: it asks the remote owner file server to lock the file with a lease time. If success, If the file is in cache, it will compare the cache file 's timestamp with remote file server's timestamp, if not the same or not in cache, create a replica in local. If within lease period, the lease file server return the file to remote file server, the update is valid. Else, it is invalid. The lease can be renewed, if the renew time is in lease period, the lease

is extended, else remote file server will refuse. In this mode, replica manager will save only one record for every file.



SHARE_WRITE (Optimistic Replication): when local file server want to share write a file with other file servers, it send SHARE_WRITE mode write request to remote owner file server, the remote owner file server will ask the other file server which has the newest replica for the file to update the file, save the file , then send the newest file back to file server. If there is no share write record in database and If the file is in cache, it will compare the cache file ‘s timestamp with remote file server’s timestamp, if not the same or not in cache, create a replica in local. From now on, the operation will be on local replica file, when finished operation, it will return the file to remote owner file server. Remote file server will update all the replica to newest file.



Reading policy: If the file is in cache, it will compare the cache file ‘s timestamp with remote file server’s timestamp, if not the same or not in cache, create a replica in local.

The exclusive relation:

	READ_ONLY	WRITE_ONLY	SHARE_WRITE
READ_ONLY	Y	Y	Y
WRITE_ONLY	Y	N	N
SHARE_WRITE	Y	N	Y

2.4Disconnect operation:
Every file server save a replica in local for future use after accessing remote file. When operating a file, local file server will try to compare the cache file' timestamp with owner file server. If the remote server is unreachable, file server will operate on local replica file, and update the modify to remote file server when the remote file server is reachable.

2.5 Scalability
To improve scalability, we don't save read_only operation in file server 's replica manager. That is, in server initialized cache consistency , read_only maybe not using newest file.
If there are many processes operating the same file in same machine, we will not upload the file to remote file server until all the process has finished operation.

3. Name Resolution
3.1 two-Level
File name is composed of two part : /Directory/Filename.
file server will read a configure file when started, its format is:
/abc : fileserver1
/bcd : fileserver2
/aaa : fileserver1

When mapping a file name into a file server, file server will search the configure file , find the matching Directory name, get its fileserver name.

Every file server will include a complete configure file, include all directory include in the configure file. If the directory belong to local host, it will contain files. If the directory belong to remote file server, it will contain replicas.

3.2 Multi-Level
Two level is very simple, but every file server have to maintain all the directory configure and have to contain all directory of others file server, it doesn't work well in scalability.

We can think about a more complex name resolution.
File name is composed of many parts: /Directory1/Directory2/Filename

file server will read a configure file when started, its format is:
/abc : fileserver1
/bcd/aaa : fileserver2
/abc/a : fileserver3
* : fileserver4

Try to matching the longest string in configure file.

Central mode: we may have one file server act as name server, contain all the directory mapping information. We put *:central fileserver in all client's configure file.
Distributed mode: matching file server either contain the file, or can give a name resolution for this file.

3.3 class definition of two-level name resolution
In KYSS ,we use two-level name resolution, and extend it to multi-level.
The config file is like :
/tmp/mpeg : sunserver1.cs.umbc.edu
/tmp/gif/abc : grad-sun-04.cs.umbc.edu
/tmp/gif : grad-sun-05.cs.umbc.edu

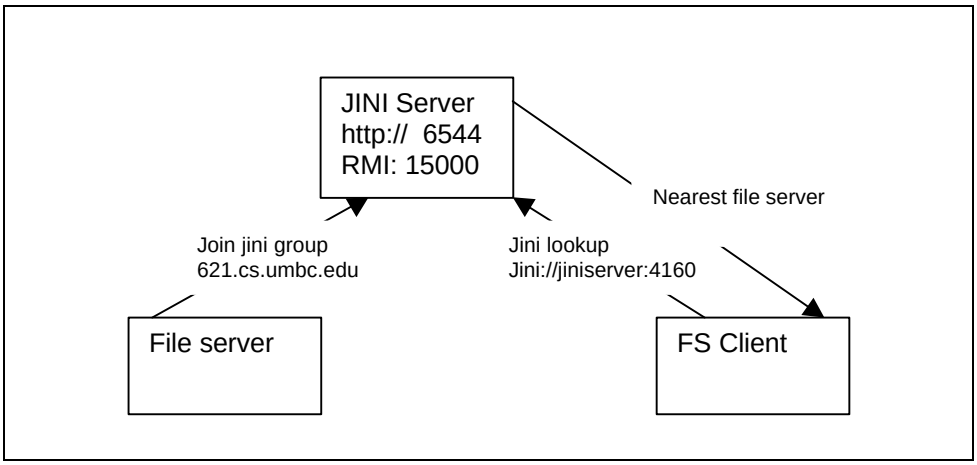
CLASS KYSS_name
{
KYSS_name(): read in name configure file kyss.conf
String Map(filename): return a hostname
}
Such as: Map(/tmp/gif/abc.txt) will return grad-sun-05.cs.umbc.edu
Map(/tmp/gif/abc/abc.txt) will return grad-sun-04.cs.umbc.edu

4. Jini: client use jini lookup services to locate a local fileserver for use.
Every KYSS file server will register a service in jini lookup server when started. Client call method jini_lookup() to get a nearest host, and use this as local file server.

What is the nearest file server?
Solution1:
first get local hostname, then visit jini lookup services, find a nearest host (compare IP address) assume the difference between IP address is the same as difference between hosts.
For example: as to 159.226.41.101
159.226.41.102 > 159.226.41.17 > 159.226.33.110 > 10.226.41.101
Solution 2:
First get local hostname, test the network communication speed, find the fastest host (send a ping to file servers, check the response time).

In KYSS, we use solution 1.

How jini works?



First, we need start a jini server in one machine:

- ✧ Start jini download server: java -jar $jiniDir/tools.jar -port $httpPort -dir $classDir -verbose , we use port 6544.
- ✧ Start rmi: rmid –port 15000
- ✧ Start reggie: java -Djava.rmi.activation.port=$rmidPort -jar $jiniDir/reggie.jar \ http://$httpServer.cs.umbc.edu:$httpPort/reggie-dl.jar $classDir/policy \ $jiniLog/reggie 621.cs.umbc.edu

Then, every file server register a jini service in jini server when start.

Jini_lookup() is used by client to locate a nearest file server for use.

```
public class KYSS_API {
   public String jini_lookup(String localHostIP) {
         System.setSecurityManager (new RMISecurityManager ());
          LookupLocator lookup = new LookupLocator (jiniServer);
         ServiceRegistrar registrar = lookup.getRegistrar ();
         Class[] types = {KYSS_IP.class};
         ServiceTemplate template = new ServiceTemplate (null, types,null);
         ServiceMatches matches = registrar.lookup (template, 10);
         //computer IP value of localHostIP, compute IP value of matches IP,return
the one with least difference.
        }
}
```

KYSS_jini is user by file server to register a services in jini server.

```
public class KYSS_jini {
       System.setSecurityManager (new RMISecurityManager());
                LookupDiscovery ld = new
          LookupDiscovery(LookupDiscovery.NO_GROUPS);
              ld.addDiscoveryListener(new KYSS_jini());
              String[] groups = new String[1];
              groups[0] = new String("621.cs.umbc.edu");
              ld.setGroups(groups);
```

```
   }
}
```

Class KYSS_IP: include two methods, getIP() and getHostName(), jini register put this class as remote object, client download this object and use those methods remotely.

5. Client: demo the function of KYSS

run java KYSS_client

```
KYSS>help
    list: list the available fileserver
    connect <fileserver>: connect to a fileserver
    cat <filename>: print this file to screen
    touch <filename>: create the specified file
    rm <filename>: delete this file;
    append <filename> <string>: append a string into a file by WRITE_ONLY
    shappend <filename> <string>: append a string into a file by SHARE_WRITE
    help: show this message
    quit: quit KYSS;
KYSS>
```

list: list the available fileserver, connect to jini server, download all matching object, run object.getHostName() printout file server's name. Use this command to check all available file server in system.

connect <fileserver>: connect to a fileserver, you can connect to any file server you want by use this command. After the connect, this file server is used as local file server. So, there is three ways to connect to a file server: use connect command; use jini lookup find a nearest file server; default use localhost as local file server.

Cat: print the specified file in screen. Client will connect to local file server, local file server goto remote file server, get the newest file.

```
FileInputStream is = kyss.open(filename, READ_ONLY,0);
Byte[] file_content = kyss.read(is);
System.out.println( file_content);
```

Append: append a string into a file in WRITE_ONLY mode, specify a lease time. You must return the leased file in time, or the modify is invalid. You can check the return boolean value of open operation (true if open success) and decide what to do next, either keeping wait and try again, or exit.

```
Boolean abc = kyss.open(filename, "WRITE_ONLY",30);
kyss.write(filename, string, "WRITE_ONLY");
Kyss.close(filename,"WRITE_ONLY");
```

Sh_Append: append a string into a file in SHARE_WRITE mode, specify a lease time. You cannot WRITE_ONLY and SHARE_WRITE a file at the same time.
Boolean abc = kyss.open(filename, "SHARE_WRITE",0);
kyss.write(filename, string, "SHARE_WRITE");
Kyss.close(filename,"SHARE_WRITE");

Rm: delete a file in local file server, it can be a primary copy or replica.
Kyss.delete(filename);

Touch: create a blank file in local file server, if the file doesn't belong to this file server (name resolution) , touch operation failed. You should use connect command to connect to the file server which the file belong to, then touch this file.
Kyss.touch(filename);

6. CLASS KYSS_API : interface between KYSS and client.
KYSS_FS is a object get from rmi server.
obj = (KYSS_fs)Naming.lookup("//"+fshostname+":6543" + "/KYSS_FS");

- String jini_lookup(client_IP): Return a nearest file server name. In jini approach, we use jini lookup services to find a nearest file server(find jini group 621.cs.umbc.edu), in non-jini approach, to make it simple, we assume local host is the nearest file server. We map a IP address into a IP value, such as IP=a.b.c.d, IP value= a * Math.pow(255, 3) + b* Math.pow(255, 2) + c* Math.pow(255, 1) + d. We compute the IP value of all file server, and return the file server which have the smallest difference with client IP value.
- Boolean open(filename,MODE,leasetime): call object KYSS_FS.open, return true if success, else false.
- Close(filename,mode): call KYSS_FS.close(filename,mode) to close this file.
- Byte[] Read(filename):call KYSS_FS.read(filename) to read this file, return file content.
- Write(filename,string,mode): call KYSS_FS.write(filename, string ,mode) to append the string to file.
- Delete(filename): call KYSS_FS.delete(filename) to delete this file.
- Fcconnect(fileserver): connect to a specified file server.
- Extend(filename,leasetime) : extend the WRITE_ONLY operation lease time, call KYSS_FS.extend(filename,lease_time)
- Touch(filename): create a blank file, call KYSS_FS.touch(filename);

How to use KYSS_API in client:

- Read: if (open(filename,"READ_ONLY",0))
  Byte[] abc = read(filename);
- Write only: if (open(filename,"WRITE_ONLY",leasetime))
  {
  write(filename,string,"WRITE_ONLY");

```
            // after operation
            close(filename,"WRITE_ONLY");
        }
```

- SHARE Write : if (open(filename,"SHARE_WRITE",0))

```
        {
            write(filename,string,"SHARE_WRITE");
            // after operation
            close(filename,"SHARE_WRITE");
        }
```

7. CLASS KYSS_Replica

KYSS file server will manage a replica database, which contain all file operation records. We only add the record which will influence the file into database, that is write operation. The key for this database is filename and replica_server.

Record format is:

replica_server leasetime replica_time filename

example:

grad-sun-05.cs.umbc.edu 0 945021922095 /tmp/mp3/mp3.txt
grad-sun-04.cs.umbc.edu 0 945021922096 /tmp/mp3/mp3.txt

- add(filename, replica_server, lease): add a record into replica database. Replica time is systime.
- Delete(filename, replica_server): delete a record from replica database
- Boolean Valid(filename, replica_server): if the lease is valid, return true. Else if invalid return false. If there is no such record, also return true. Check the replica_time+lease with systime, if replica_time+lease>systime, return false, else true.
- String [] Gethost(filename): return the hosts that operating this file, this is useful if you want to update all replicas when you get a modified file.
- String Getlatest(filename): get the latest host which operating this file, this is useful if you want to get the newest file content to a client, even if the file server who has the newest file is still not finished operating of the file.
- Boolean last(filename): check if there is someone who come from the same machine is also operate this file. For scalable reason, if there is others from the same machine is also operate this file, you neednot update the new file to all replica evrrytime you finished operation. You can wait until all processese have finished operate on this file, then update all replicas on other file server, only one time. This will not influence others file server get the latest file content, because other file server can call getLatest(filename) to get newest file content.

8. CLASS KYSS_FS : this is the kernel of KYSS file system.

KYSS_fs(): register a KYSS_FS services in registry of localhost,port 6543.

- Write(filename,string,mode) : if the mode is WRITE_ONLY: because the file has been locked, write into file in local file server directly. If the mode is

SHARE_WRITE, write into a cache file in local file server. After the operation, the cache will be merge into current file content.

- Merge(filename,cache): append the cache into local file server's filename. We use a modified solution of SPRITE system. No one can really write the file when share write,every file server only write into local's cache, when finished operation, submit the modify to owner file server which keep the primary copy, owner file server will merge the cache into file. SPRITE disable all replicas when share write. In KYSS, replica is still valid, and be updated to newest if there is any modify, so file servers can make use of local replica, have better performance and scalable.
- Long gettimestamp(filename): return the last modify time of the file. In KYSS, we change replica file's last modify time the same the timestamp as primary copy's , whether the replica is the newest is decided by timestamp.
- Log_last(filename,fileserver): check replica manager whether there is other processes from the same machine is also operate this file.
- Log_delete(filename,fileserver): delete a log in replica manager
- Log_add(filename,fileserver,leasetime): add a log into replica manager
- Extend(filename,leasetime): extend the lease time of write_only operation. Here we assume: extend is permit if the process is still in lease time, and can extend any times.
- Log_getlatest(filename): get the latest file server which operate this file.
- Update(filename,filecontent, timestamp): filecontent is the newest file content. Get all file servers which operate this file, update the replica to newest filecontent(call save()), set last modify time to timestamp.
- Save(filename,filecontent,timestamp): save the filecontent into filename, set last modify to timestamp.
- Read(filename): read in the local file server, if the owner file server is unavailable, it will use local replica directly.
- Delete(filename): delete a file in local file server, it can be a primary file or replica.
- Deletecache(filename): delete the cache file, only when there is no process in the same machine share write this file.
- Readcache(filename): read in the cache file, when a file server want the newest file, it will go to getLatest() file server which has the newest copy of this file, use readcache() to get newest cache, merge into file.
- Touch(filename): create a blank file in local file server, if the file belong to other file server, print error message.
- Close(filename,mode): finished the operation of a file.
  - ✧ READ_ONLY: do nothing
  - ✧ WRITE_ONLY: update the file content to owner file server, if the lease is valid, save in owner file server. Delete replica log.
  - ✧ SHARE_WRITE: upload the cache to owner file server, merge into file, update the file servers's replica (Here, the file server is the file servers which share write this file currently)
- open (filename,mode,leasetime): open a file for operation.
  - ✧ READ_ONLY: get the newest file content from owner file server.

- ✧ WRITE_ONLY: try to lock the file, if there is no one write_only or share_write this file, lock is success. Get the newest file content from owner file server, write a log in replica database.
- ✧ SHARE_WRITE: first try to share lock this file. if there is no one write_only this file, lock is success. Get the newest file from owner file server, get the newest file cache from latest file server which share write this file, merge them into a really newest file. Write a log in replica database.

9. test

we have done some test on KYSS. Because we have to run jini in sun solaris, we test following machine:

sunserver1.cs.umbc.edu
grad-sun-01.cs.umbc.edu
grad-sun-02.cs.umbc.edu
grad-sun-04.cs.umbc.edu
grad-sun-05.cs.umbc.edu

grad-sun-04.cs.umbc.edu: is jini server and file server.
grad-sun-05.cs.umbc.edu and grad-sun-02.cs.umbc.edu is file server.
grad-sun-04.cs.umbc.edu ,grad-sun-01.cs.umbc.edu and sunserver1 is client.

We get following result
Localhost is to use grad-sun-04.cs.umbc.edu operate grad-sun-04.cs.umbc.edu's file.
Localhost is to use client sunserver1.cs.umbc.edu operate owner file server grad-sun-04.cs.umbc.edu's file, through local file server grad-sun-02.cs.umbc.edu

READ_ONLY:	localhost: 0.2 second
	Remote file: 0.3 second
WRITE_ONLY:	localhost: 0.16 second
	Remote file: 0.4 second
SHARE_WRITE:	localhost: 0.25 second
	Remote file: 0.9 second

From the result, we know , SHARE_write is the slowest, because share write has to get newest cache, merge, and update all replicas. But share write support parallel operation, if there are many clients operate the same file , SHARE_WRITE will has the best average performance.

10. demo

- ✓ we write some demo program to demo the function of KYSS.
- ✓ Demo1.java: demo how the write_only work, write only a file with a lease time, return them in time.
- ✓ Demo2.java: demo how the write_only work, write only a file with a lease time, return them after lease time, file will not be accepted by owner file server.

- ✓ Demo3.java: demo how the extend lease time works, write only a file with a lease time, extend it to a long time, then return them in time.
- ✓ Demo4.java: demo how the extend lease time fail, if the process submit lease extend request when lease has already expired, extend request is rejected.
- ✓ Demo5.java: demo how the jini works, find a nearest file server
- ✓ Demo6.java: demo how the share write works. Run this program from many client at the same time, share write will put every update into owner file server's file.

11. limitation and future works.
    - ✓ Delete operation: when you delete a file in KYSS, you only delete it from local file server, either primary copy or replica, future, it should delete all replica when you want to delete a file.
    - ✓ Replica log: if there is many processes from the same machine, KYSS cannot make any difference between them, in the future, we should add process id into replica log.
    - ✓ File permission: in KYSS, the primary file and replica have the same last modified time, in future, they should have the same read/write permission and other file attribute.
    - ✓ Current merge only support append operation.
    - ✓ Lease time extend algorithm is too simple, if a process keeping extend lease time, other process will never get change to lock this file.