Design document

Name: Ning Zhang
CWID: A20336916
Department of Computer Science, Illinois Institute of Technology

The Decentralized File Sharing System is developed by JAVA and the remote communication is based on sockets and multithread. The whole system has three important classes which are respectively indexserver, fileserver and client. And the fileserver and client are in the same process which is the peer. The function of the indexserver is keeping the filename and file location by storing these into the hash table and dealing with the requests from the client like registering several files and searching file location. And the function of the fileserver is keeping the files and dealing with the requests from the client like obtaining and downloading some file. The function of the client is sending requests like registering several files and searching file location to the indexserver and sending obtain requests to fileserver for downloading the file.

The difference of Decentralized File Sharing System from Centralized File Sharing System in PA1 is that it has multiple indexservers and the hash table is distributed among these index servers. The structure of this Decentralized File Sharing System is shown in Fig1.

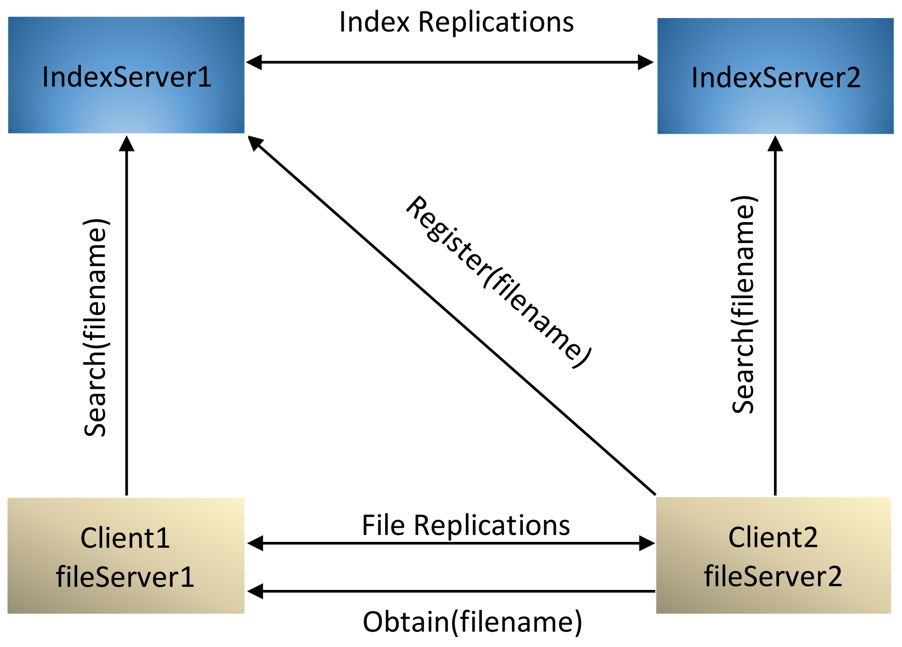

Fig1. The Structure of Decentralized File Sharing System

Through this Fig1, we can see there are replication operations among fileservers and among indexservers. We design a principle of implementing the replication. As the Fig2 shows, fileServer1 does the file replication of fileServer3, and fileServer2 does the file replication of fileServer1 , and fileServer3 does the file replication of fileServer2. This is like a cycle.

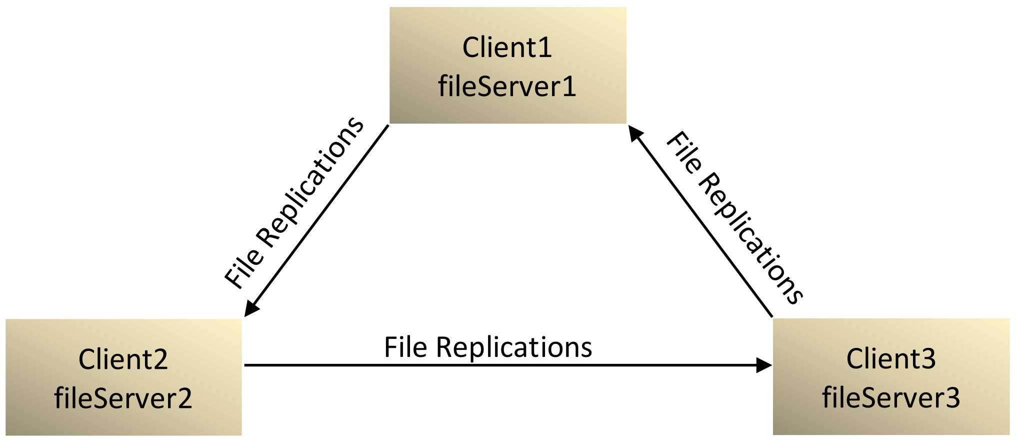

Fig2. Replication Principle

Metadata structure

The metadata is used to keep track of file locations. Specifically, it is a map of files to different peer servers because multiple peers can own a file simultaneously. Therefore, we use a hashmap to store the information. The key is the file name and the value is a hashset that contains file locations, i.e. peer server IDs. The hashmap is the global view of the filesystem in this system. Any update should be reflected accurately and timely to this hashmap.

Automatic registration & metadata update

In the context of this system, automatic means any change in the local file system should be reported to the index server accurately and immediately. Upon initialization, this system peers will automatically send out a hashmap of its local filesystem to the index server. The server merges all the initialization messages into a centered and global hashmap which contains all the information of this system system. this system is design to utilize VFS' nice features to achieve the automatic mechanism. A virtual file system (VFS) or virtual filesystem switch is an abstraction layer on top of a more concrete file system. The purpose of a VFS is to allow client applications to access different types of concrete file systems in a uniform way. In this system, we leverage VFS' automatic file system monitoring function and event triggering mechanism. Each peer has its own implementation of VFS' monitor. The monitor will trigger a corresponding method call in this system and send out the most updated local filesystem hashmap to the index server, therefore keeping the global metadata updated and accurate. This process is invisible to all clients.

Seletion of IndexServers

Every server keeps a hash table to store the key and value. Before every client send a command, the system will use MD5 algorithm and mod operation to decide which sever the client will send this command to. Firstly, the system will use MD5 algorithm to transfer the string key into the 16-byte hex. Then using 16-byte hex to mod the number of servers for getting which server the client will send the command to as Fig 3.

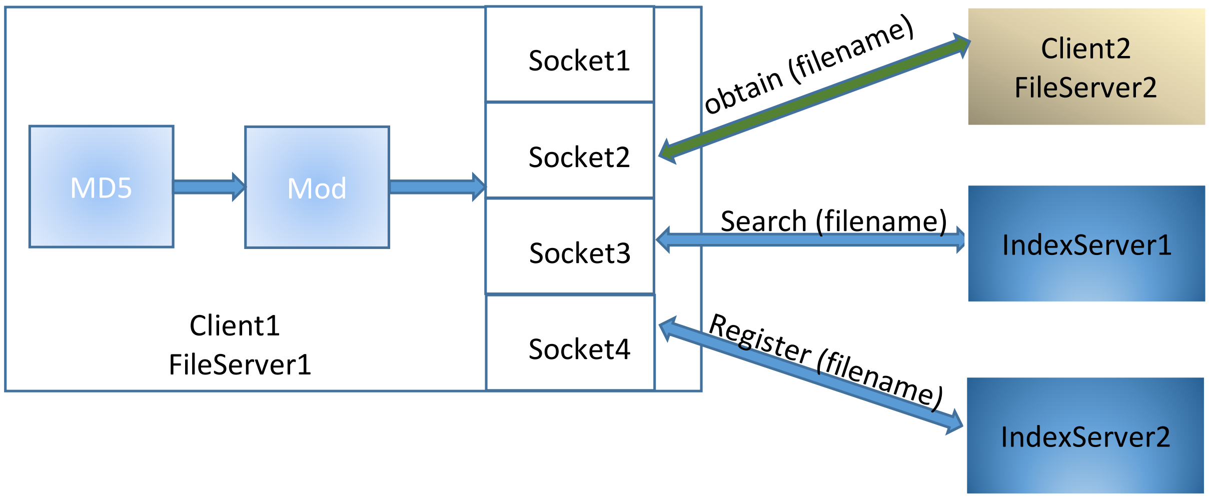

Fig3. The implementation of this DistributedHashTable system

Socket Cache

As the Fig 3 shows, the Client1 selects the IndexServer2 to send the Register command. And in order to improve the communication efficiency, we use a hash table to store the sockets connected to the servers as the socket cache. By this method, we reduce the cost of establishing and releasing the socket connection.

Synchronization of Threads

Because every server has a hash table and this hash table may be simultaneously written and read by more than two clients, we must handle this synchronization of these client threads. So we use the concurrenthashtable in JAVA to implement the hash table for handling the synchronization of these client threads.

MD5 algorithm

The MD5 message-digest algorithm is a widely used cryptographic hash function producing a 128-bit (16-byte)hash value, typically expressed in text format as a 32 digit hexadecimal number. MD5 has been utilized in a wide variety of cryptographic applications, and is also commonly used to verify data integrity. So we use this algorithm and mod operation to assign the keys on average to the servers.

Running in the cloud

This system is designed to function in the cloud environment in that each peer can run independently on different machines. As discussed previously, the communication is entirely based on sockets/TCP/IP. Specifically, we assign each peer a PID based on ip address to ensure its uniqueness and this PID is used for internal system communication. The basic functions including Register, Search and Obtain which are verified in the system.