Name: Manik Behl

Student id: 17313056

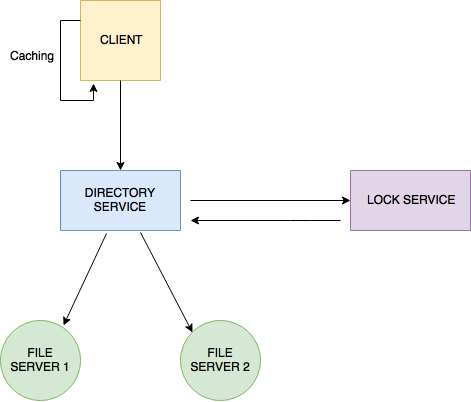
Email id: behlm@tcd.ie

Dependencies: Python 2.7

**Running the Project**

1. Start.sh file is used to run the program
2. ConfigM.py file contains all the server port numbers.
3. Database.py file is using for the caching server as it will only checks for the file is present in the database or not

**Distributed Systems**

****

**Introduction:**

A distributed System is one in which self-sufficient, independent – frequently, self-governing or heterogeneous – spatially-isolated segments must utilize a typical interconnect to exchange data and facilitate activities and enable the entire to appear to the client as one single rational framework. At the point when a client gets to a record on the server, the server sends the client a duplicate of the document, which is reserved on the client's PC while the information is being prepared and is then come back to the server. All together for discrete and autonomous procedures to understand the correspondence that they are participating in, at that point they should take after an arrangement of conventions of engagement and trade.

**Sockets:**

Low level, coherent level IPC.

In unix, they can be seen as information/yield streams.

Customer side attachments are regularly fleeting, e.g. in a web program one is made to send a demand and get a reaction, and after that disposed of:

⁃ Create an attachment for the given transport (e.g. TCP) and web convention (e.g. IPv4).

⁃ Connect to the server utilizing the port connected to the convention (e.g. HTTP is 80).

⁃ Then the customer can send the demand. Through a similar attachment a reaction is sat tight for and after that got in pieces.

⁃ Socket is disposed of.

⁃ Gopher is a great illustration.

Server-Side Sockets:

⁃ Acts more like a dispatcher.

⁃ It tunes in for associations on the server attachment and ports them.

⁃ When it gets another association it gets another attachment accordingly.

Server-Side Socket set up:

⁃ Create a server attachment for the given transport (e.g. TCP) and web convention (e.g. IPv4).

⁃ Set the attachment choices.

⁃ Bind the server attachment to a host address and a port.

⁃ Start tuning in to associations on the server attachment and set the maximum number that could be left holding up.

I have implemented :

1. TCP server

2. Directory Service

3. Lock service

4. Replication

5. Caching

Brief description of the services implemented is below.

**Directory service:** Directory Server gives a focal vault to putting away and overseeing data. Any sort of data can be put away, from character profiles and access benefits to data about application and system assets, printers, arrange gadgets and made parts. Data put away in Directory Server can be utilized for the confirmation and approval of clients to empower secure access to big business and Internet administrations and applications. Index Server is extensible, can be incorporated with existing frameworks, and empowers the combination of representative, client, provider, and accomplice data. Directory server provides mapping by considering all the standard naming transparency while providing the service.

It circles through all record servers and returns/refreshes/adds documents to each of the record servers which have been indicated by the customer.

Steps included in the directory service implementation:

1. overriding request processing function

2. requesting file details from directory

3. adding the folder to directory listing if writing

4. checking if folder exists in directory listing

5. if not writing then assigning folder to random server

6. returning the directory id and location

7. checking if the file is in directory

8. returning the file not found if file\_id key not in files directory.

**Lock Server** : Lock server more works like a semaphores for each file that is accessed by the user. When a client needs to access any file or anything from a server it sends a request to the server and ask if that file is available, if the file is available then the server gives the client the permission to work on that file. But during that situation if any other client comes asking for the same file then it will get the message “the file is locked” as Lock server provides the mutual exclusion. So, when a file is being used by the client, it is locked till the time client finishes the work on the client. After finishing the work the file is unlocked by the client so that other clients can take mutual exclusion.

Steps included in the implementation:

1.overriding the request processing function

2.lock request

3. acquiring locks binary semaphore/mutex

4.returning failure if file is locked and lock owner is different client

5. otherwise file is acquired by the client by locking file for them and returning success

6. unlock request after work has been finished

7. unlock binary semaphore/mutex

8.unlocking and return success if file is locked and owned by client

9.otherwise returning failure if file is not in array

10.otherwise returning file locked by another client

**Replication:** Replication is the strategy for making a few duplicates or copies of a particular asset which must be utilized by the client over and over. The client can pick the duplicate from the closest server and work on it. Replication is done keeping in mind the end goal to diminish the heap on the server, get to idleness and the system .blockage.

**Caching Service:** Caching in the distributed file systems are employed to avoid the network accessesat different clients and also to compensate the differential speed between main memory and disks at file servers. It also contributes to the scalability and reliability of the distributed systems as data can be remotely cached on the client node, an effective caching strategy will seek to maintain consistent copies of the file server data. There are three Locations where caching solution implemented:

·         Maintaining a copy in memory on client

·         Maintaining a copy in disk on client

·         Maintaining in memory of server

Steps followed for the implementation:-

1.importing thread

2. Setting the socket.

3. Waiting for connection.

4. Accepting connection

5. Starting thread for connection.

6. Parse to return file object

7. Returning file TODO

8. Closing the connection.

9. New socket.

10. Getting a response of a file name and path

11. saving the file into my database

12. saving to cached files list.