Name: Liang Han
Login Name: lingh2
Student Number: 712397

1.The challenges of multiple clients

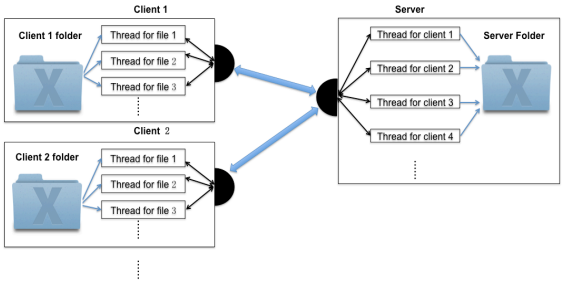

Figure 1.1 Model of multiple clients

It may arise some challenges if the number of clients increases in our design.

One challenge is the scalability problem. The server creates one thread for each of the clients. The threads on the server and clients are persistent until the client is manually shut down. As the increase of the number of clients, the amount of threads on the server will increase. If the number of clients is m and the number of files on each client is n, the cost of resources on the server is O(m*n). As a result, the performance of server may become the bottleneck of the synchronization system.

Another challenge is the concurrency issue. All the clients use the same port to communicate with the server. If there are a great number of clients attempting to connect the server at the same time, the requests are sequenced. Some clients have to wait to connect, which slows down the overall performance of the system.

The last challenge is about security. The program is only designed to create one folder to synchronize the files from different folder. This means the synchronized files from one client may be overridden by the files from another clients if the two clients have files with the same name. What's more, there is no mechanism that records where the files come from. It would be impossible to find the synchronized files if there were further requirements to fetch the synchronized files back to the client.

2. File scaling issues and improvement approaches

2.1 File scaling issues

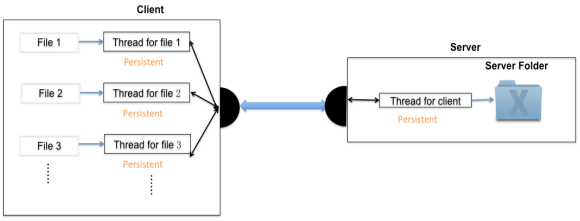

Figure 2.1 Model of file scaling

There are two issues arisen by the increasing of number of files based on our design. One problem of our design will be recourses wasting. One thread is created for each of the file and the file thread is persistent no matter there is a file change or not. The persistent threads will occupy too many physical resources if the number of files increases. The other problem is low efficient. Because the functions of sending instructions from the client are synchronized, some threads may have to wait until other threads release the sending functions.

2.2 Improvement approaches

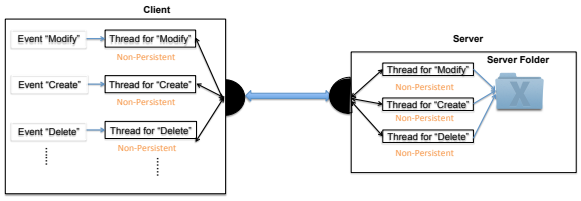

Figure 2.2 Events-Based thread creation pattern optimization

One possible solution to this problem is that the client set up a new thread based on the "events" of the folder instead of the files. The number of events generated by one check of the folder is always no more than the number of files. If the client finds out there is a file change, it establishes a new thread to synchronize the file. What's more, the thread is not persistent. After the synchronization is finished, the thread is closed automatically. This will reduce the cost of resources on the client. So it is with the threads on the server. One server thread is created corresponding to the thread on the client. In this way, the threads on the server will also not be persistent. Compared with our design, it will release the occupation of resources on the client and server, which is more efficient.

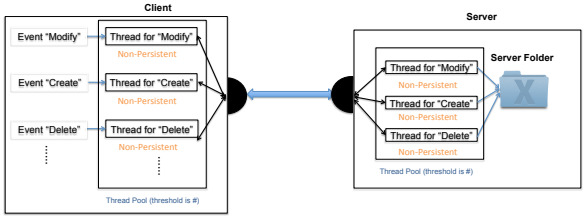

Figure 2.3 Thread Pool optimization

A further optimization is to use “thread pool pattern” especially for the event-based thread creation pattern. In the file-based thread creation pattern, if some of the threads of the files are not used for a certain period, they can be set sleep or destroyed[2]. Another file thread is created to synchronize that file. If the event-based thread creation pattern is applied, a threshold can be set for the maximum amount of threads. Only if the number of events is less than some safe number, the client can create a thread to synchronize the files.

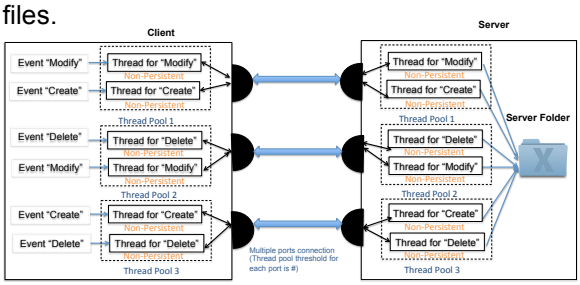

Figure 2.4 Multiple Posts optimization

The last optimization is to use multiple ports (Figure 2.4). In the event-based thread creation pattern, the number of threads for each port has a limit. To avoid the congestion caused by the limit of threads pool of the port, a practical method to use more ports to communicate between the client and server.

3. Discussion about two commercial file synchronization systems

3.1 EverNote

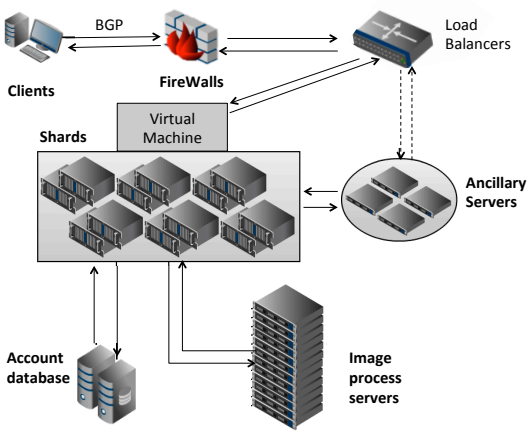

Figure 3.1 EverNote Architectural Model[3]

One good example, which is similar to our design, is EverNote. This is a notes-synchronization system. There are many good solutions to the challenges of distributed system based on the architectural models.

The users of EverNote produces up to 150 million HTTPS requests per day, with peak traffic around 250 Mbps.[3] For this scalability challenge, EverNote uses “Border Gateway Protocol” to direct traffic through fully independent network feeds from NTT and Level 3. What’s more the firewalls are connected to the balance loaders. It dealt with fluctuation of the number of visiting of users to ensure the Qos.

Secondly, EverNote saved the user files on a farm of servers “shards”. To make it easier to manage, Evernote develops a Virtual Machine that runs the core application stack, made up of Debian, Java MySQL and so on, to manage the local file data. This visual can be regarded as a middleware, which help handle the programming languages heterogeneity problem. No matter the terminal is a mobile or a web, the application on the terminal can easily use the Virtual Machine to get the users data via the interfaces without knowing the inner programming languages in the system.

Thirdly, the user accounts servers, image processing servers and the ancillary servers are separated to perform a single function and EnverNote also has a secondary data centre to backup the whole copy of data. All these approaches are to handle of failure of the system.

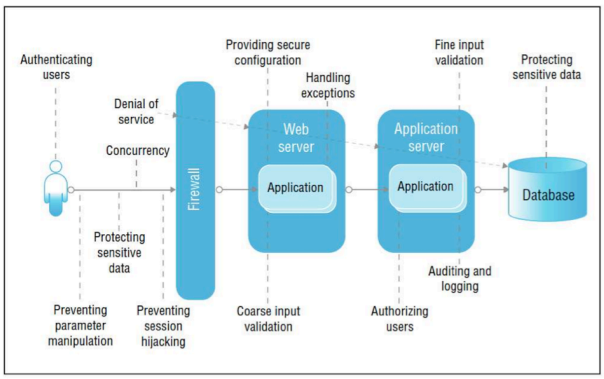

Figure 3.2 EverNote Security Model[4]

Finally, based on the Security Model (Figure 3.2), it can be seen that Evernote deals with the security issues on different layers of the system. The firewall protects the database from the parameter manipulation and the session hijacking. What’s more, the concurrency issue is handled by the queuing module of the firewall. Inside the system, web server provides the secure configuration and input validation. The application server records the log and authorize the users. The three tiers play joint functions to ensure the security of system.

3.2 Google File System (Google Drive)

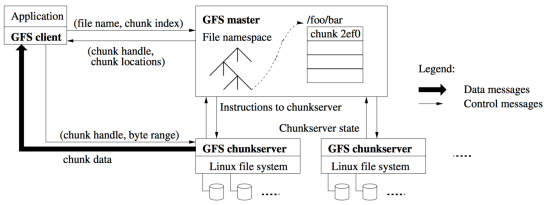

Figure 3.3 GFS Architectural Model[1]

Compared with the EverNote, the Google File system is more complicated, because it need to deal with the files more than documents and the size is files is larger. The architectural model and corresponding solutions to challenges are different.

Firstly, a GFS cluster consists of a single master and multiple chunkservers and is accessed by multiple clients.[1] Large size file is divided into fixed-size chunk to store on the chunkservers. The Master maintains all file system metadata, including access control information, the mapping from files to chunks, and the current locations of chunks.[1] GFS uses the MetaServer and ChunkServer to handle the scalability issue.

Secondly, GFS has its own “GFS client code” which can be run on each kind of applications and communicate with the master server on behalf of the application. Different from the Virtual Machine used in the EverNote, GFS relies on Mobile Code to deal with heterogeneity problem.

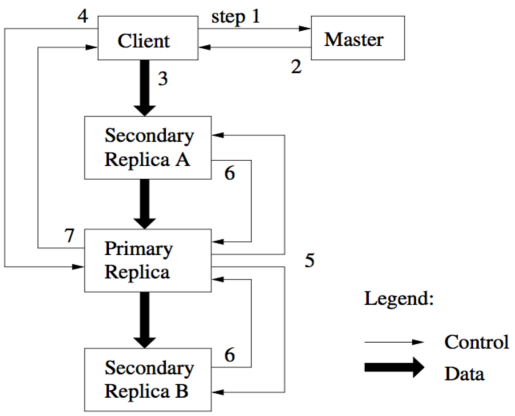

Figure 3.4 GFS Reliability Model

Thirdly, according to the Reliability Model, GFS establishes secondary replication server to back up files more than one replication, which greatly increases the reliably and avoid the failure.

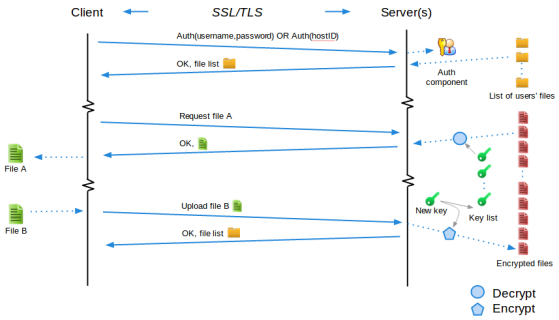

Figure 3.5 GFS Security Model[5]

Lastly, different from the EverNote three security tiers model, the security mechanism of GFS relies on the secure transmission layer and the encryption storage. The clients use the SSL/TLS layer to communicate with the server. This process is request and reply process. The client firstly sends a request to access to his own file folder. The username, passwords or hostID are used to verify the identification of users. Only if the users pass the verification, it can upload and download the files. If the client upload a file, the sever will generates two keys. One is used to encrypt the files to store one the sever. The other one is stored in the key list used to decrypt the stored files. The client sends a request to fetch file, the server will send back the files encrypted by the corresponding key in the key lists. This mechanism ensures the security of the file system.

3.3 Critical Comparison of the two systems

EverNote and Google File System have different approaches to deal the scalability issue, as a result of different types of files to be synchronized. Both of them can achieve the synchronizing function. But EverNote isolates the servers with certain function (account database, image processing servers and so on). This approach will increase the service efficiency and quality to a great extent. GFS does not take this approach but separate its chunk servers. EverNote does better in the scalability problem.

In security, both of the two systems use the SSL to transmit to ensure the channel security. However, GFS does better in the encryption of storage of files. This approach will make sure that the files are safe when the servers are attacked.

In terms of heterogeneity, the two systems take different approaches based on their business scenario. The clients of EverNote are all developed by itself no matter web or app. So the Virtual Machine can be open to the inner developers without considering about safety and

it is easy to realize the advanced functions. Different from EverNote, GFS is open for the public developers besides its own application. So GFS uses the Mobile Code and regulated API to run on other systems. It can not only handle the heterogeneity problem but also improve the safety of the file system.

References:

[1]Ghemawat S, Gobioff H, Leung S T. The Google file system[C]//ACM SIGOPS operating systems review. ACM, 2003, 37(5): 29-43.

[2]http://en.wikipedia.org/wiki/Thread_pool_pattern

[3]https://blog.evernote.com/tech/2011/05/17/architectural-digest/

[4]http://www.kandasoft.com/softwaredevelopment/software-security-services/

[5]http://softwarerecs.stackexchange.com/questions/7677/alternatives-to-google-drive-with-emphasis-on-privacy-and-user-experience