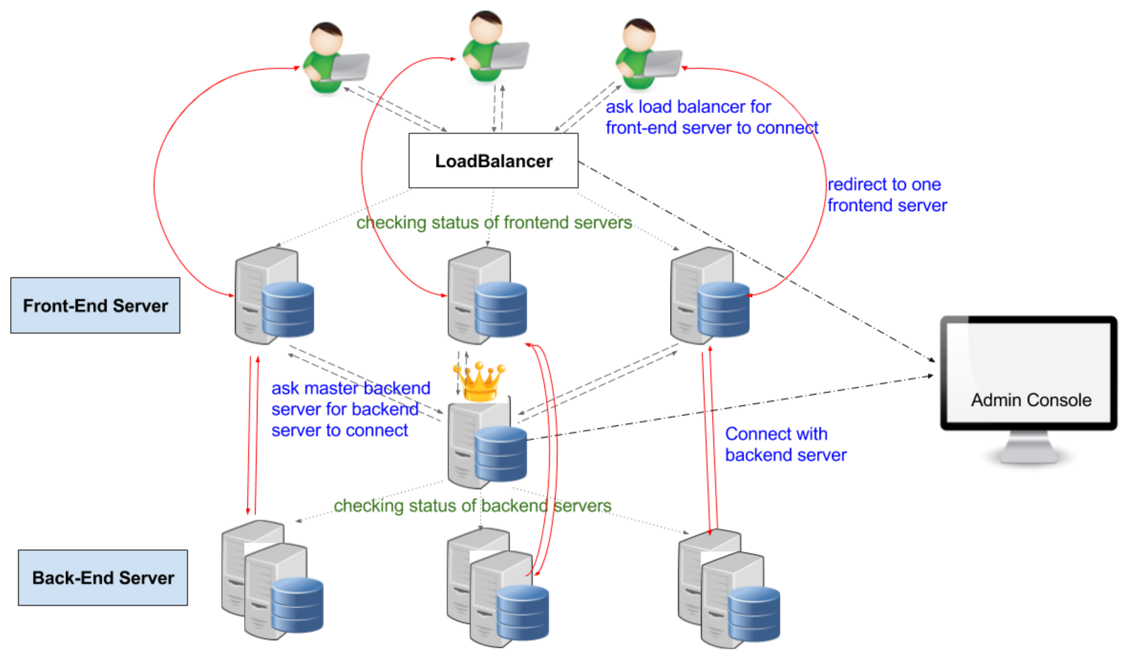
505 Final Project Report

Team Member: Xiaofan Dou, Qingxiao Dong, Man Hu, Zhi Xu

System Design


ask load balancer for front-end server to connect
LoadBalancer
redirect to one frontend server
checking status of frontend servers
Front-End Server
ask master backend server for backend server to connect
Admin Console
Connect with backend server
checking status of backend servers
Back-End Server

Features

Frontend

- Load Balancer
- Apply round-robin to distribute users to connect with different alive front-end servers
- Check the running status of all front-end servers and provide the information to Admin Console when requested

- User Account

When a user first logs in, system would check whether Get(username, “pwd”) is same as the inputted password. If it’s same, it would generate a sessionid based on current time and username and a random number. And then store (username, “sessionid”, sessionid) into the backend. And then, Set the cookie to: sessionid=xxx and user=yyy back to the user. And then, send back the dashboard page back to user.

If the user needs to create a new accout, it stores (username, “pwd”, password), (username, “files”, “”), (username, “emails”, “”) and the generated sessionid into the system. And then, set the cookie with sesssionid and username and send back the dashboard page back to user.

Every time when a user want to access emails or files resources, system would check whether the sessionid stored in cookie with corresponding username matches the backend stored ones. If matched, it would send back the resources. Otherwise, it would send back the login page to let user login.

- Webmail Service

System stores email-index and email header pairs in the (user, “emails”) cell. For example, it stores 1\nemail1, time1\n2\nemail2, time2\n5\nemail5, time5\n in the “emails” col of the user row. Once the user requrest the email list, system would generate a list of emails based on these. If the user wants a specific email, for example, email 5, it would retrieve the stored value (username, “email-5”) from the backend and display the stored content in this retrieved value.

- Storage Service

System stored all the folder and files name seperated by line feed. For example, folder-f1/aaa\nfolder-f1/bbb\nfile-f1/a.png\n in the folder-f1 col. It represents there are two folders, aaa and bbb and one file a.png in the f1 folder. The root path is “files”. So when user first request all the files he wants, it would return value stored in (username, “files”). Folders and files are separated by prefix. “folder-” prefix names a folder and “file-” prefix names a file. A file slot stores all the file data by hex string. If a user request a file, server would process these hex string and converted it into bytes and send back to the user. When a user wants to upload a file, server would convert it to hexstring and stored it into the the storage server.

When a user wants to rename a file, he retrieves data of the previous file, and then transfer the stored data to the newly named slot and then delete the previous file. And then, retrieve the parent folder and rename the file.

When a user wants to delete a file or folder, he delete the file or folder name in the parent folder and then delete the data stored in the file or folder. If it’s a folder, it would need to delete all the children folder and files by DFS.

When a user wants to move a file, server would delete it from the parent folder and then add it into the destination folder. And then, server should also rename the file prefix as it indicates the file location.

- Admin Console
  - Show the current status of nodes (frontend servers and backend servers) in the system
  - Display raw data in the alive backend servers

Backend

- Key-value Store

The key-value store is implemented in Bigtable+GFS manner. The backend was implemented in master-slave mode.

- **File System:** Similar to GFS, we set a 5KB chunk(file) as the basic unit. The size of a single chunk will grow as data stored until it reaches the limit, which is flexible and currently is set 5kB. Then system will generate a new chunk to for storage. The metadata (key-file mappings) will be stored in a file and loaded into cache in system initialization.

- **LFU Cache:** LFU cache is applied on top of the file system for better performance. The node will read the whole file containing the queried key into memory for more efficient read/write and evict the least frequently used file if the cache is oversized. Meanwhile, we set checkpoints (periodically write snapshots) and in each node, we maintain an append-only full and a temporary log, in which the node will log every write operation for recovering from crash.

- **Fault-tolerance/Replications:** In addition to logging and replay, we use replications for fault-tolerance. Each key-value pair will be stored into several replicas (now the number is set to 2) storage servers using consistent hashing. Whenever the primary update an entry, it will force the other replicas to update it too. If the primary goes down, the replica will be used. Once the primary goes back, it will get freshed data from its replica.

● RPC

We used gRPC for remote procedure calls. The communications between servers is realized using RPC. For example, frontend servers could ask master server for a storage server for a specific key. Then the frontend server will send request (PUT/GET/…) directly to the storage servers.

Extra Credit

● Dynamic Membership

Dynamic membership was implemented in backend by applying consistent hashing in master, where the storage servers can join/leave the system at run time:

1. When a new storage server wants to join the group, it will firstly contact master with its ip. The master will do consistent hashing and respond with the ip of the node(the node clockwisely next to the new node) from which the new node should ask for data. Then data that is in the space of the new node will be migrated from the old node via grpc method calls. And replica will be updated in the corresponding nodes accordingly.
2. If a storage node leave the group, master would be able to detect it such that next time new requests come from the frontend servers, master will respond with the addresses of the nodes that are currently up in the system instead the old nodes.

<u>Discussion</u>

Design Decisions

- Use prefixes to distinguish files and folders as request of them expect different behavior.
- Using full path of files and folders when storing them to avoid duplicate file problem.
- Using email index to distinguish different emails to avoid duplicate email headers.
- Storing different files into different location to avoid retrieving useless data when user requests a list of emails.

Major Chanllenges:

1. For backend:

a. The design decisions in the first beginning.

b. The dynamic membership implementation in master side.

<u>Division of Work</u>

Zh Xui: Frontend servers, Webmail Service, Storage Service, Frontend HTML, Mail Forwarding
Man Hu: Load balancer, Admin console, Logging, UI(using Bootstrap)
Qingxiao Dong: File system, GRPC (storage client and server framework), UI(using Bootstrap)
Xiaofan Dou: LFU Cache, Master(with Consistent hashing for dynamic membership), UI(using Bootstrap)