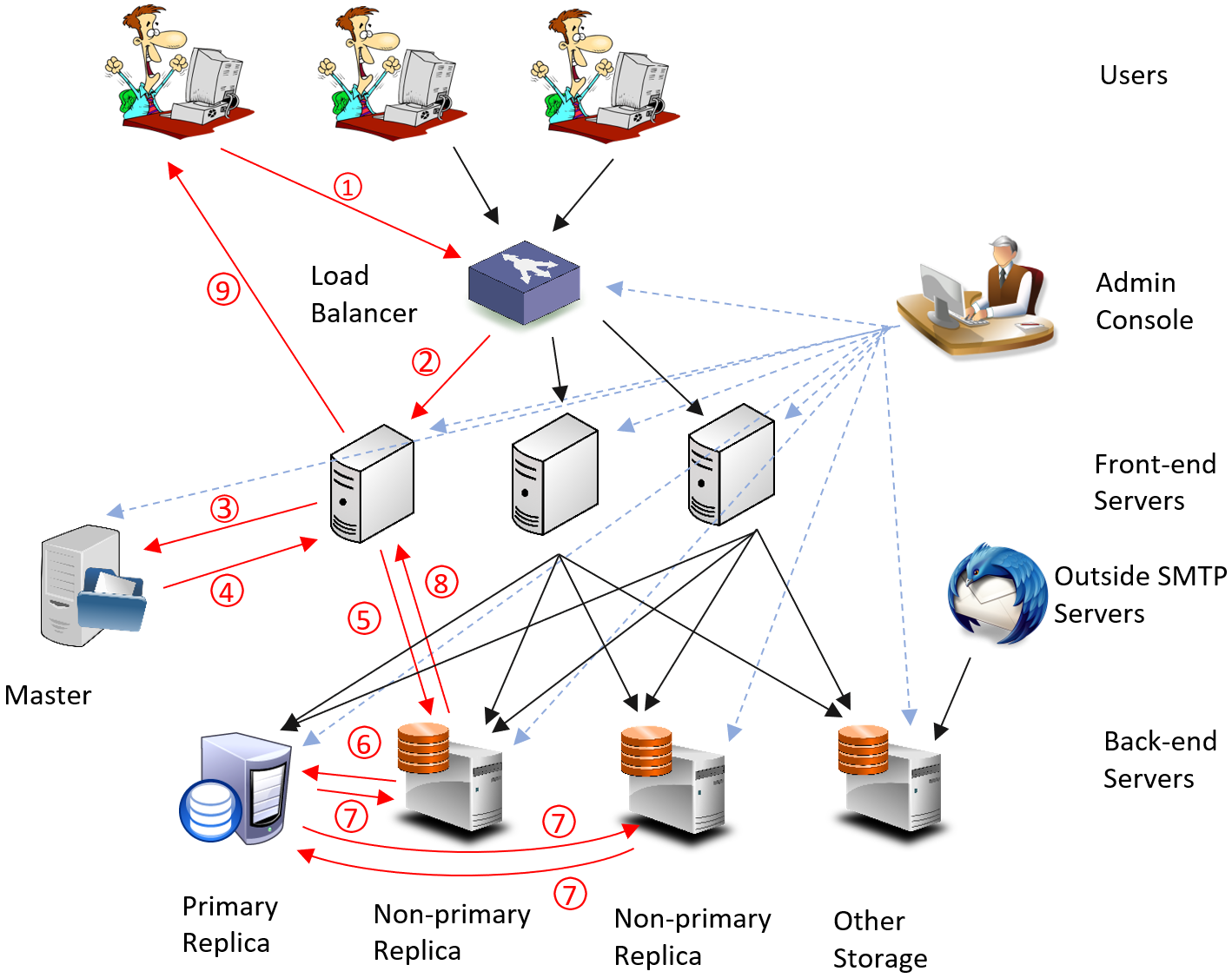
**CIS505 Final Project Proposal**

Wen Zhong, Yilong Ju, Haoran Shao, Gongyao Chen, Qianyi Guo

**Design Descriptions:**

The system is constructed with two ends, front-end servers and back-end servers that deal with human-computer interaction and storage process, and also a load balancer. When a user accesses the front-end server from web page, a master node will direct the front-end server to the back-end storage server where the specific tablet of key-value is stored. Caching should be used in the front-end server so that it will contact master node just once to get directed to the back-end server if it works well. Each time the user does a write operation, the back-end servers with the specific key-value stored should make replication to avoid data loss.

A graph of the system architecture with workflow is shown below. The red arrows show the procedure of a write request with: 1,send write request 2,assign a front-end server 3,request the back-end server 4,assign the back-end server 5,request write 6,send request to primary 7,replicate 8,response to front-end server 9,response to user



**Summary of the Design Decisions:**

1. Key-value store:

The backend system stores all the data is a distributed key-value store. The full solution of this component has the following four requirements:

***Distributed:***Divide up the key-value pairs into small “tablets” between the storage nodes by defining ranges of row keys as users and assign each range of column to storage nodes as types of data that is stored.

* Design and build a master node that keeps the mapping from ranges to storage nodes, gives clients the mapping when getting request and the client would contact with the assigned nodes directly. The master node should keep the metadata of all back-end servers to get the correct back-end storage server to front-end server. Some fault-tolerance algorithms may be used in the future.

***Replicate:***Each value should be stored on more than one back-end storage node to realize distributed partition.

* Use Primary-based protocols as replication protocol and implemented by remote writes. Once a user makes a write operation, the updated data should be replicated to all the servers with the key of the data. The primary replica could be selected by some Leader Selection algorithm and all non-primary replica should forward the write request to the primary and make replication with response.

***Consistency:***Use lock schema to prevent inconsistencies if multiple clients are issuing PUTs and GETs in parallel. Our team choose to use Client-Centric Consistency model.

***Fault tolerance:*** Keep all the tablets in memory and keep one “append-only” log on disk in which all operations are recorded per tablet. The size of the log is limited with writing a checkpoint to the disk.

2. Frontend server

Implement a multithreaded frontend server together with load balancer to deal with user requests. Caching is used after directed to the back-end data storage for future use.

3. User accounts

1. Support multiple user accounts.
2. After a user logged in, show the user’s own inbox and file folders
3. Allow users to sign up for a new account, as well as change their passwords.
4. All user data are stored on key-value store

4. Webmail service

1. Add a GUI for users to view emails in the mailbox, compose new emails and delete emails.
2. Mailbox data are stored in key-value store instead of mbox files. The keys are passed from the frontend server, like username. And certain operations will be performed, like reading values (view mailbox), editing values (compose/delete emails).
3. Adapt the SMTP server in HW2 first. At the beginning, we will restrict email transmissions to users within the system. And later in our full solution, we will add support to accept emails from an outside SMTP client, using the DNS to look up the MX records for the recipient’s domain, and connect to one of the servers that are specified in these records.
4. Some anti-spam techniques will be implemented to our SMTP server.
5. Email service will support address books and sending/receiving attachments. Some enhancement on GUI will also be considered if we have time.

5. Storage service

The storage service stores files as key-value pairs. It also stores the folder structure in the key-value storage. In the key-value storage, the row key is the full path of file/folder, and there is a type column indicating whether this row represents a file or a folder. For the content column, its value is the binary content for files and a serialized list of folders and files in it. Finally, the service provides a web interface for the user to view, upload, download files and manage folders.

6. Admin console

The admin page shows the nodes in the system and their status. It should also show the raw data in the key-value storage in a table format to the user.

**Division of labor:**

Frontend server / User account: Wen Zhong

Load balance / Webmail service: Yilong Ju

Key-value store: Haoran Shao, Gongyao Chen

Storage / admin: Qianyi Guo

**Collaboration plan**

Regular meeting once a week: 3pm - 4:30pm on Monday

Other meetings depending on the project progress.

**Set of milestones for the project**

Nov 9th - Nov 22nd: Finish minimum solutions of all components and integration.

Nov 23rd - Dec 7th: Finish full solutions of all components and debug.

Dec 8th - Dec 14th: Finish components integration, fault-tolerance and whole system test and improvement.

Dec 14th - Demo Day: Project report or extra credits if possible.