**Proposal**

|  |  |
| --- | --- |
| **Name** | **Student ID** |
| **JiaHao Li** | **1652066** |
| **Yicheng Wang** | **1650107** |
| **Tengjiao Zhu** | **1652619** |

**1. Description**

The Distributed Text Editor is a useful tool for empowering multiple users to work on a single document simultaneously, regardless of their location, making it an indispensable asset for teams and individuals working remotely. However, several critical challenges should be addressed like managing concurrent edits by multiple users, maintaining a consistent view in real time of the document for all users, and handling failures, as well as a seamless collaborative editing experience. By minimizing network traffic and reducing latency, a distributed test editor can support real-time collaboration, ensuring that updates made by one user are instantly visible to all others.

**2. Motivation**

Our Distributed Text Editor project aims to enable real-time, seamless co-editing of documents from anywhere, which tackles challenges like concurrency, data consistency, and network issues, fostering collaboration across teams, educational institutions, and open-source projects. By leveraging cloud computing and advanced technology, our project offers improved productivity, wider participation, and dynamic collaboration, addressing critical needs for remote workers and global teams.

**3. Previous Work**

Developing a distributed, fault-tolerant collaborative document editing system requires addressing challenges in document consistency, system scalability, and fault tolerance. Weidner et al.'s work on minimizing interleaving through the Fugue and FugueMax CRDT algorithms provides a theoretical foundation for ensuring document integrity across distributed editing sessions[1]. The Collabs framework offers tools for building custom CRDTs, facilitating efficient synchronization across clients[2]. Tapia et al.'s study on optimizing collaborative editing tools through container orchestration demonstrates a practical approach to scalability and fault tolerance[3]. Kleppmann et al.'s examination of interleaving anomalies in collaborative text editors and the proposed specifications for avoiding such issues further inform the development of robust collaborative features[4]. Additionally, Mathrani and Edwards' insights into knowledge-sharing strategies in distributed collaborative product development highlight the importance of effective communication and collaboration mechanisms[5]. Lastly, Krause-Glau and Hasselbring's work on dynamic software visualization within code editors introduces innovative approaches to enhancing developer collaboration and understanding of code behavior[6]. These references offer a comprehensive approach to overcoming the technical challenges associated with developing a collaborative editing system.

**4. Implement**

Our project boosts productivity by enabling multiple people to simultaneously edit documents like text files, spreadsheets, and presentations. As a distributed system, it ensures concurrency control, data consistency and replication, fault tolerance and recovery, and efficient communication.

**4.1 Pipeline**

There are three types of nodes, which are the client node, the primary server node, and the slave server node. The details of which are as follows.

|  |  |  |
| --- | --- | --- |
| **Node** | **Function** | **Communication Way** |
| Client | create, open, modify, save, delete a files on file servers | TCP/IP |

|  |  |  |
| --- | --- | --- |
| Primary  Server | get commands from clients, and dispatch them to the targeted file servers. | TCP/IP |
| File Server | Perform commands with targeted files on their database | TCP/IP |

Table.1: the roles of nodes

Therefore, the pipeline is depicted as follows.

Firstly, the primary server listens for the connections from clients and file servers on a specific port. Secondly, all the file servers listen for the connections from the primary server to get commands from the primary server and monitor the real-time status of the primary server.

Moreover, once the network is launched successfully, clients can communicate with the file server with the help of the primary server. Specifically, the client sends both commands and data to the primary server. After that, the primary server dispatches the packet to a specific file server with a dispatching algorithm. Once a certain file server gets the packet, it will do the operations on the targeted file in the local database. And then, the file server sends the results to clients via the primary server. Until now, our system has finished the request from a client and sent back operation results to the client.

The client, primary server, and file server will repeat the third part all the time until there are errors. **4.2 Structure**

To get a better understanding of the structure of our system, there is a flowchart depicted below.

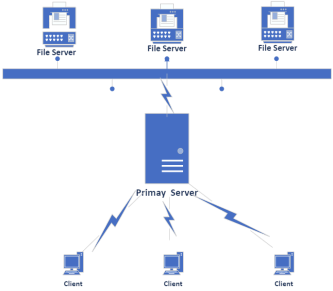


Fig 1: the structure of our system

**4.2 Algorithms to be implemented**

There are at least three types of distributed algorithms to be implemented. Here are details.

1. Distributed shared memory protocols.

As we can see in the Fig2, there is a primary server where there is a copy of the certain file used by some clients in three file servers.

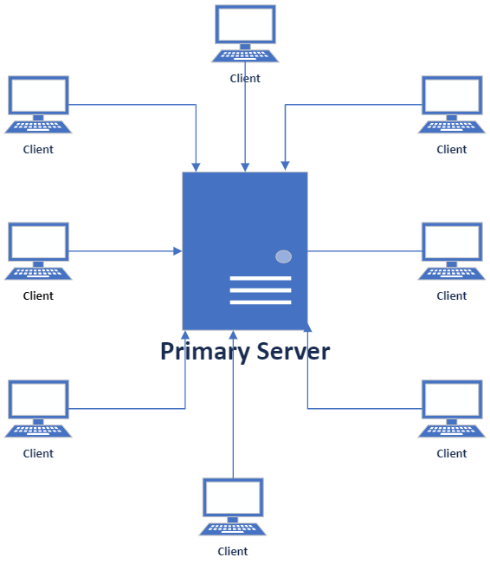


Fig 2. Distributed memory algorithm

For read request from clients, the primary server just send the content of the file to the clients. For write request from clients, the primary server update the file with the new data from the clients. Once the last client closes the file, the primary server will perform migration to store the modified file to the three files.

1. Replication and consistency protocols

There are three file servers for storing all the files created by any of clients. All the files in the three servers are totally same.

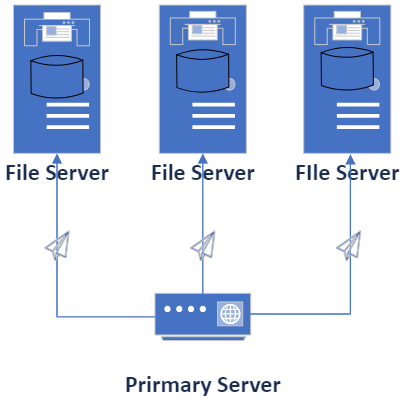


Fig 3. Distributed memory algorithm

The primary server will update the three file servers with the same contents so that the three servers have the same content of same files.

1. Concurrency control.

Operational algorithm is an efficient method for collaborating editor by google. The details are as follows in Fig 4. Both users have the same the file as the primary server. On the user1 side, it inserts a letter into the end of the file. And then, it sends the updated content to server, which update the content with the data from user1 and send the new version of the file to user2. On the user2 side, it also inserts a new letter into the beginning of the file. After that, it also sends the modified data to server. Once receiving the data, the server updates the file with the data from user2.

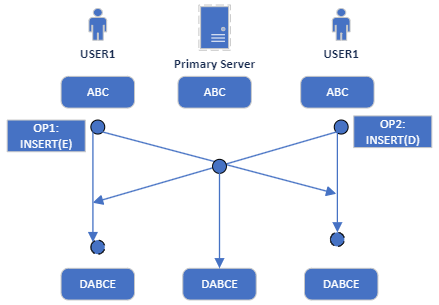


Fig 4. Distributed memory algorithm

For user1, it must update its file with the new data from the primary server, while user2 need to update its own file with the new data from the primary server. At last, the files on both and the primary server have the same data.

**5. Deployment**

Our collaborating editor consists of three parts including clients, primary server, and three file servers. There are many ways of deploying these nodes. In simplicity, here is our method for the deployment of them.

|  |  |  |
| --- | --- | --- |
| **Node** | **Function** | **Location** |
| Clients | Edit files | Users’ personal computer like desktop or laptop. |
| Primary Server | Handle the request from clients and send response to clients. | The primary server is in one EC2 node. |
| File Servers | Store all the files | The three file servers are in three S3 nodes. |

Table. 2 locations

**6. Schedule**

|  |  |  |
| --- | --- | --- |
| **Item** | **Date** | **Milestone** |
| Partner Selection | 01/30/2024 | Form groups. |
| Proposal | 02/08/2024 | Submit a formal proposal for the project and we will use the pre-conceived algorithm when implementing it. |
| Design and  implementation | 03/07/2024 | Combining Concurrency Control, Data Consistency and Replication, Fault Tolerance and Recovery, Efficient Communication and other technologies to complete the function implementation of the system. |
| Deployment and  testing | 03/10/2024 | First test the function implementation of each node in the local environment and fix possible bugs, and then deploy multiple nodes on the cloud platform for real simulation. |
| Code Submission & Presentations & Final Report | 03/14/2024 | Write and submit final reports and presentations. |

Table. 3 schedule

**7. References**

[1] Weidner, Matthew, Joseph Gentle, and Martin Kleppmann. "The Art of the Fugue: Minimizing Interleaving in Collaborative Text Editing." *arXiv preprint arXiv:2305.00583* (2023). [2] Weidner, Matthew, et al. "Collabs: Composable Collaborative Data Structures." *arXiv preprint arXiv:2212.02618* (2022).

[3] Tapia, Freddy, et al. "A Container Orchestration Development that Optimizes the Etherpad Collaborative Editing Tool through a Novel Management System." *Electronics* 9.5 (2020): 828. [4] Kleppmann, Martin, et al. "Interleaving anomalies in collaborative text editors." *Proceedings of the 6th Workshop on Principles and Practice of Consistency for Distributed Data*. 2019.

[5] Mathrani, Sanjay, and Benjamin Edwards. "Knowledge-sharing strategies in distributed collaborative product development." *Journal of Open Innovation: Technology, Market, and Complexity* 6.4 (2020): 194. [6] Krause-Glau, Alexander, and Wilhelm Hasselbring. "Collaborative, Code-Proximal Dynamic Software Visualization within Code Editors." *2023 IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE, 2023.