Contents

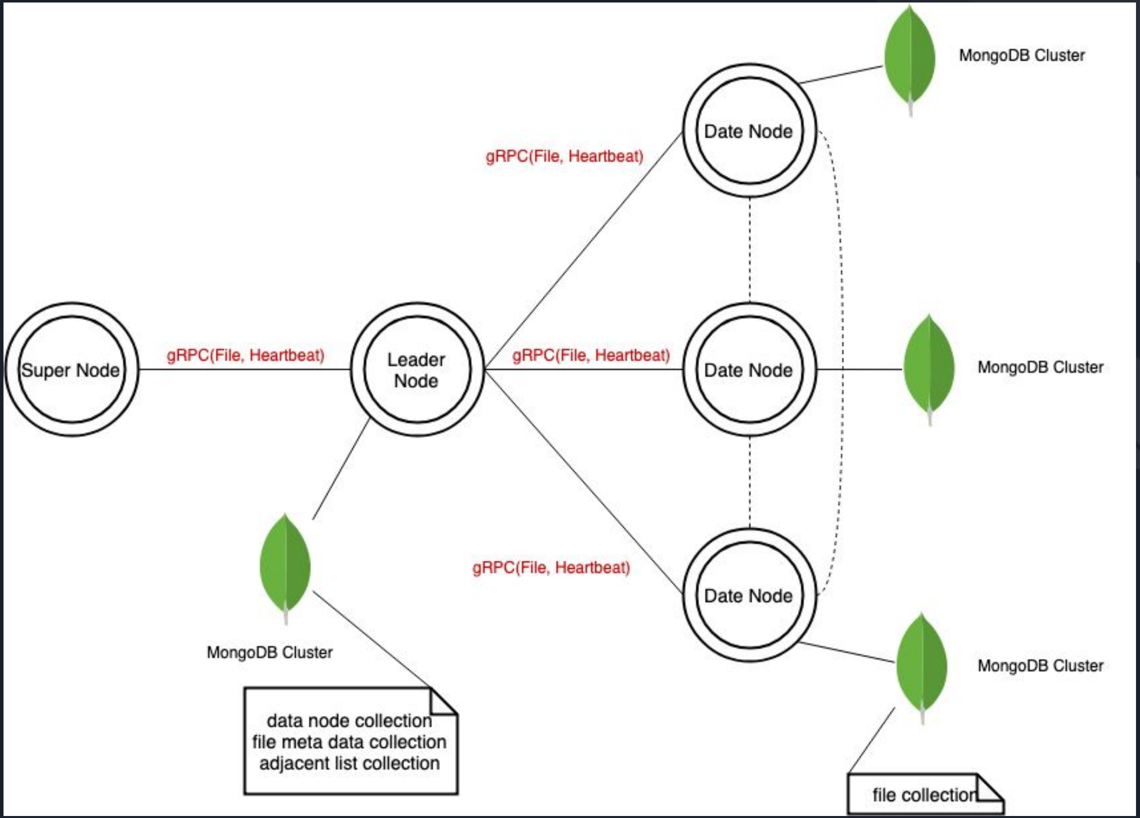
- Objective
- System architecture
- File Uploading
- File Downloading
- File Deletion
- File Searching
- Task Dispatching
- Fault Tolerance
- Summary

Objective

Build a distributed file storage system which can handle

- File uploading, downloading, deletion, and searching
- Concurrent requests
- Task dispatching
- Fault tolerance

System Architecture



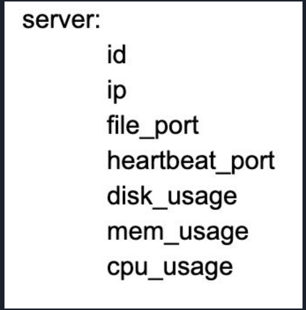
File Storage && Database Schema

File Storage

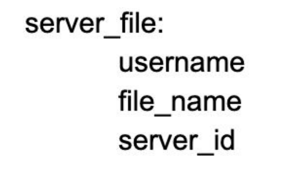
- Use Linux file system to store files
- Use MongoDB to store file metadata

Database Schema

❖ Leader Node Collections

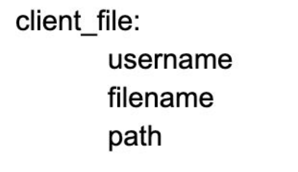


```
server:
        id
        ip
        file_port
        heartbeat_port
        disk_usage
        mem_usage
        cpu_usage
```



```
server_file:
        username
        file_name
        server_id
```

❖ Data Node Collections



```
client_file:
        username
        filename
        path
```

File Uploading

1. Leader node receives request
2. Leader node performs task dispatching based on the heartbeat packets
3. Data node receives the file and store it in the file system
4. Data node update its file location collection
5. Data node notifies Leader node the storage is finished
6. Leader node update its file metadata collection

File Downloading

1. Leader node receives request
2. Leader node locate the file based on metadata
3. Data node sends the file to Leader node
4. Data node updates its file location collection
5. Leader node sends the file to client
6. Leader node updates its meta data

File Deletion

1. Leader node receives request
2. Leader node checks where the file is located based on the file metadata
3. Leader node ask the specific data node to delete file
4. Data node delete the file and update its file location collection
5. Data node notifies the Leader node the deletion is done
6. Leader node update its file metadata collection

File Searching

1. Leader node receives the request
2. Leader node checks its meta data to see if file exists in the system

Task Dispatching

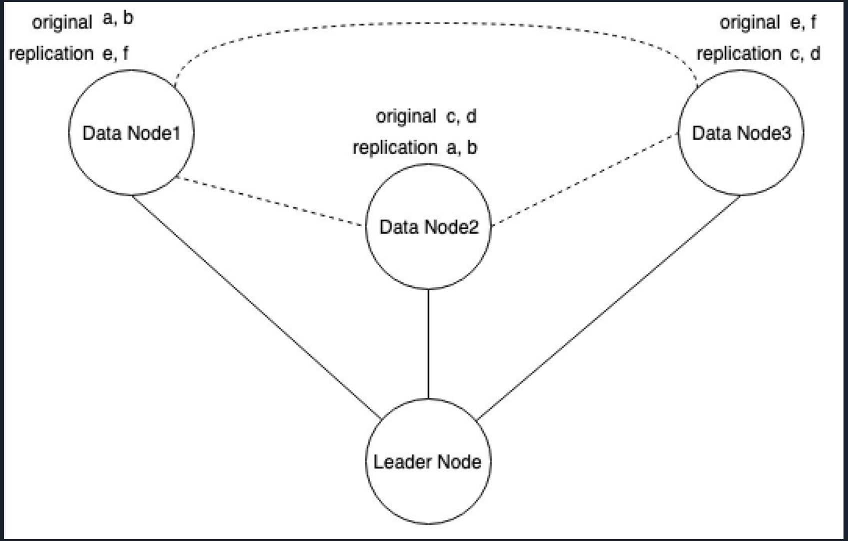
1. Heartbeat packet sent from Data node to Leader node contains its current CPU, Memory, and Disk Utilization information.

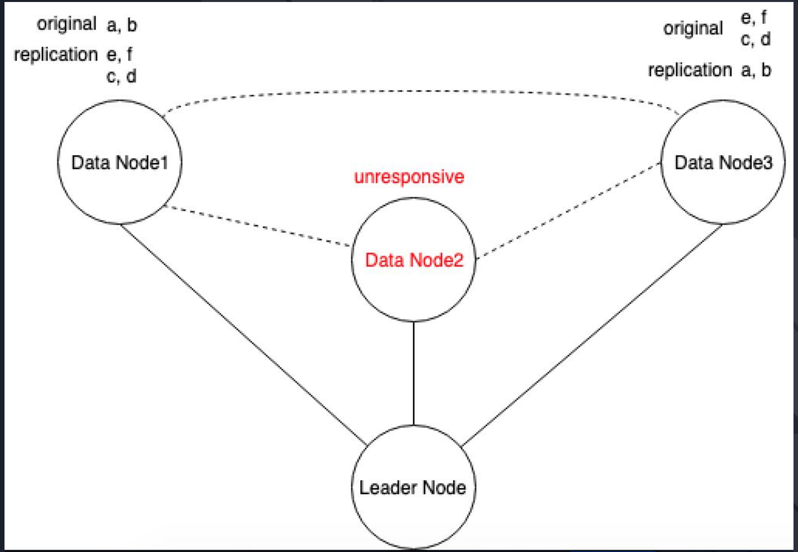
2. Every Data node sends a heartbeat packet to Leader node in every five seconds.

3. Leader node select the Data node with the lowest CPU, Memory and Disk Utilization as the new data node to store the file.

Fault Tolerance(Focus)

Algorithm 1 - Ring backup





Fault Tolerance(Focus)

Algorithm 2 (implemented)

- Every time a new file comes, the Leader node selects two Data nodes with the lowest utilization rates and store this file on both of them.

- When one of these two node fails, the Leader will be notified and select another node with the lowest utilization rate and make another replication on it.

Summary

- Our file storage system supports file uploading, downloading, deletion, and searching.
- Our file storage system can handle concurrent requests.
- Our file storage system dispatch task based on heartbeat packet.
- Our file storage system supports fault tolerance.
- The technologies used includes Java, Python, Protobuf, gRPC, MongoDB, etc.