

Distributed File System

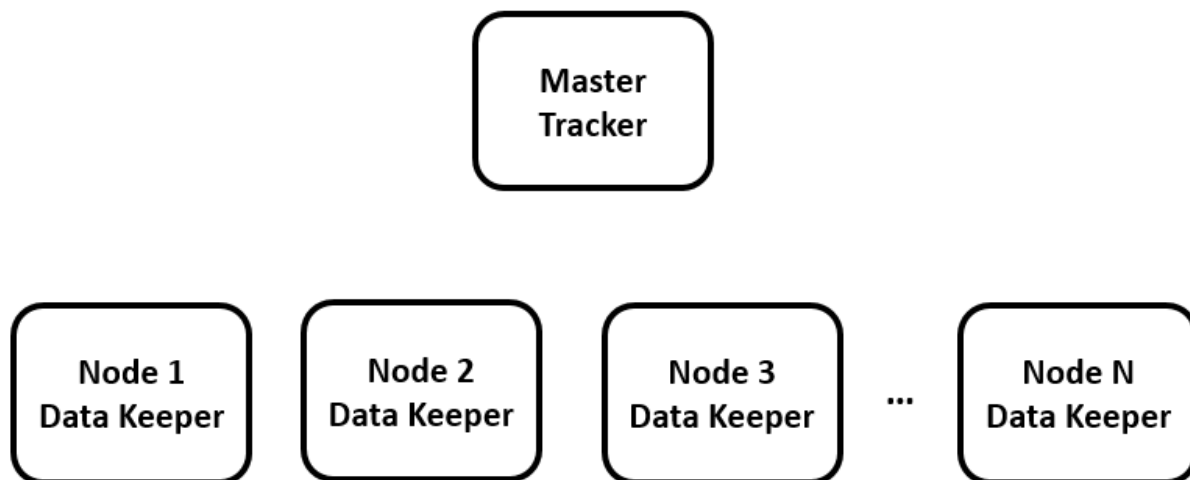
With the ever-growing technological expansion of the world, distributed systems are becoming more and more widespread. They are a vast and complex field of study in computer science. A distributed system in its most simplest definition is a group of computers working together to appear as a single computer to the end-user.

These machines have a shared state, operate concurrently and can fail independently without affecting the whole system's uptime.

In this assignment, you will build a simple distributed file system that supports reading and writing mp4 files while keeping files replicated for fault tolerance.

Architecture

DFS is a centralized distributed system having 2 types of machine nodes. First, the **Master Tracker** node. This node has a look-up table. The look-up table columns are (*file name, Data Keeper node, file path on that data node, is data node alive*). Second, the **Data Keeper** nodes which are the actual nodes that have the data files. Both the **Master Tracker** and **Data Keeper** nodes should be multi-threaded to handle multiple requests simultaneously. (Check <https://go.dev/tour/concurrency/11> for more about concurrency in Go)



Communication

All communication between **Master Tracker**, **Data Keepers** and **Clients** should be over **gRPC**. File transfers should be over **tcp**.

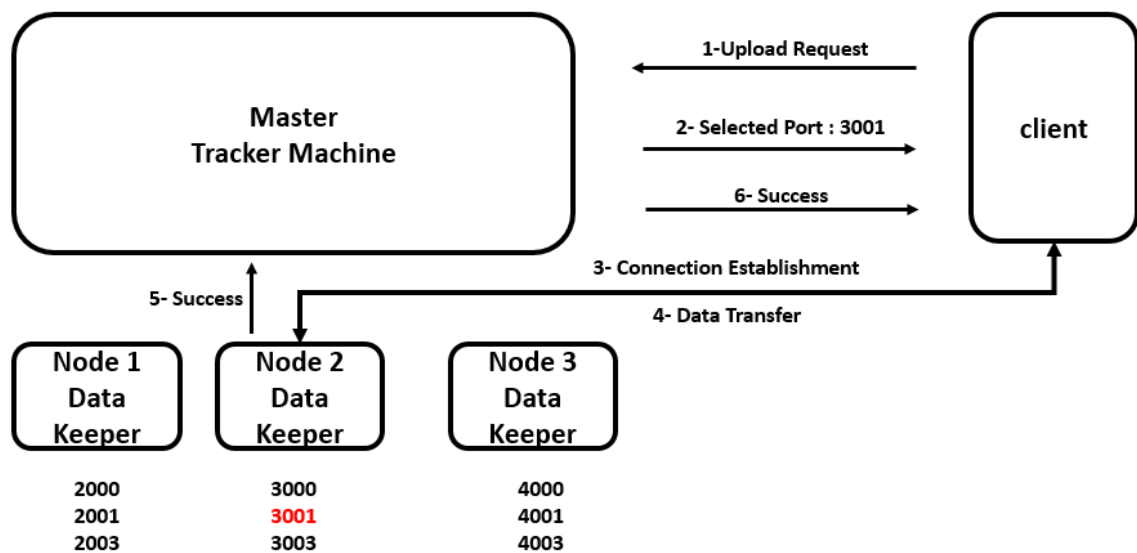
Heartbeats

Every 1 sec, each data keeper node sends a keepalive ping to the master tracker node. The master tracker node then updates the look-up table mentioned above. If one of the data keeper nodes is down it will change the corresponding cell in the 'is data node alive' column.

Uploading A file

For a client to upload a file to a cluster, the following protocol **MUST** be followed:

1. A client process **MUST** communicate with the master tracker node.
2. The master tracker **responds with a port number** of one of the data keeper nodes.
3. The client then constructs a communication with this port, and **transfers the file to it**.
4. When the transferring procedure is finished, the data keeper node will then **notify the master tracker**.
5. The master tracker then adds the file record to the main look-up table.
6. The master will **notify the client with a successful message**.
7. The master chooses 2 **other nodes to replicate the file transferred**.



Replication

A separate thread on the **Master Tracker** should awake every ($n= 10$) seconds and check for replication according to the below algorithm. Each file should exist on at least 3 alive data nodes.

initialization;

for k: distinct file instances **do**

 source_machine = getSourceMachine(file[k]);

while getInstanceCount(file[k]) is less than 3 **then**

 destination_machine = selectMachineToCopyTo();

 notifyMachineDataTransfer(source_machine, destination_machine, file[k]);

end

end

getSourceMachine is a function that takes a file record, then gets the source machine and the file path on that machine.

selectMachineToCopyTo returns a valid IP and a valid port of a machine to copy a file instance to.

notifyMachineDataTransfer This function must notify both source and destination machine to start copying the file.

Downloading A file

A client can download his mp4 files. The following protocol MUST be followed:

1. Client **request from the Master Tracker** to download a certain file name.
2. Master Tracker **responds with a list of machines IPs and ports to download a file** from.
3. Client **MUST request from every port uniformly**. (Parallel download is considered a bonus)

Rules

- You are required to build the **Master Tracker, Data Keeper and Client**.
- Team consists of up to 4 members.
- Submission **Due 11:59 PM, Saturday 16/03/2024** on <https://forms.gle/SM5dZP4SEBR9i2fn9>
- Discussion will be on Sunday 17 Mar. during tutorial time.
- Plagiarism will result in zero grade.