Design Document for CS2510 Course Project 2: Fault-Tolerant Distributed Storage System

Authors: meiqi.guo@pitt.edu; yud42@pitt.edu
Reviewers: lol16@pitt.edu

Overview

This is a design document for the second project of CS2510. The document provides a quick overview and prospect of our design for the Fault-Tolerant Distributed Storage System. We plan to use replicas for both directory server and storage node to provide the fault-tolerant. Our project is programmed in Python.

Proposed Design

In our design we will implement replication to handle directory node failure and event triggering comprehension to handle data node failure. In this session the assumed architecture is shown first, then we will go over the fault tolerant design in different parts separately.

Network Architecture

The architecture and basic communication of our design is shown in Figure 1.
There are 2 directory servers and 3 storage nodes in our network. Directory server A is the primary server and Directory server B is a replica backup server. Similarly, Storage node A is the primary node and B, C are replicas. There could be several simultaneous clients.

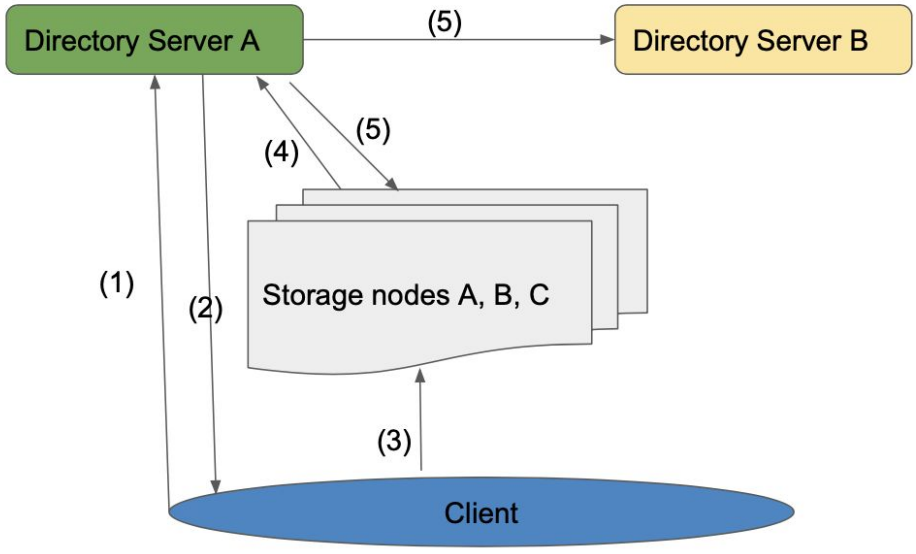

Figure 1: The architecture of our design

The basic process for our distributed storage system is described as following:

- Step 1: A client connects to the primary directory server and request for storage node locations or a complete file list.

- Step 2: The primary directory server A sends the location of the primary storage node A or a complete file list to the client.
- Step 3: The client add/request file to/from Storage node A.
- Step 4: Storage node A notifies the directory server A that a new file should be added to the storage system if Step 3 is adding file to the storage system.
- Step 5: Directory server A does two things at the same time if there is a new file added to the storage system: 1) Directory server A will trigger add file requests same as a client to other storage nodes (for example, B, C) to propagate the new file; 2) Directory server A will send updated file list to the Directory server B for consistency objective.

Fault tolerance in directory nodes

In our design the potential failure in directory nodes is prevented by replicated directory nodes. There are two directory nodes within the system, a primary node and a backup node.

The primary node will operate as follows: Firstly, it will handle all the communication with clients and storage nodes if it is alive. Secondary, it will send updates to backup directory node to keep the backup one up-to-date at best. And each client and storage node will have a list of directory nodes, once the primary node is down, it will go for the backup one and change its role to primary.

The backup node, on the other hand, will be promoted to primary one once it receives information from clients or storage nodes. Before acting as a primary node, it needs to do the following: Firstly, it will ask all storage nodes about their file list to make sure its file list is up-to-date (in case the primary directory server failed before sending the update to the backup directory server). Then it will start the next backup directory node and send all information to it.

By using backup directory node, we can make sure the reliability of services in directory nodes.

Fault tolerance in storage nodes

The failure in storage node will be found by clients and then handled by directory nodes. The stage can be partitioned into finding and handling.

During the finding stage, the clients will trigger a fault event once it finds that the storage node which it supposed to communicate is down. The error detected event will triggered by a socket error exception caused during building connections to the node. It will send an error message to the directory node to indicate the potential failure. The directory server will remove the dead storage node and launch a new storage node with following synchronizations. And once there is no reply, the system will assume that the storage server is actually down.

The failure handling stage will be launched if failure found in a storage server, the directory server will do the following: Firstly it will open up another a new storage node, and ask living storage nodes to send files to this new one. Once the file list and data are synchronized

successfully in the new node, the directory node will add this new one into the storage nodes list.

Similarly, for a directory server failure, the detected event will be triggered if a socket error exception is met by client during connecting. Then the client will change its directory server to the backup one, send a directory server down error, then build link with the new primary directory server (previous backup one) to continue operations.

On the backup directory server side, once a directory server error is met it will be promoted to the primary directory server. And as a primary server, it will launch its own backup directory as the new backup one.

Evaluation Metric

- Number of messages exchanged
- Number of bytes transferred
- Compute the average response time per client for getting file list (from directory server and storage node), readFile and addFile request by measuring the response time seen by a client.

Extensions

In our original design, when the primary directory node is up, the backup directory node will not communicate with clients and storage nodes. However, it can also participate in normal communications to balance loads on primary directory node. And this could be a potential improvement to our design.