Distributed Storage Mini Project

This project is focused on understanding, implementing and comparing performance of a distributed storage system using different schemes for redundancy allocation. Use the tools and ideas from the lectures and labs to predict the expected behavior and reason about the observed behavior in your system.

You are expected to work in a team to design and implement a distributed storage system using Python 3. Every team member should participate in all stages of the project: specification, system design, implementation, testing and measurements.

Report: The team should submit a report that briefly describes the system design and answers the Tasks below. The report should be between 5-10 pages.

Exam: Each team member takes the oral exam individually. You will need to defend your team's design decisions and answer questions about implementation details. You will also need to show the results of the measurement tasks and reason about the expected and observed performance.

Groups: Recommended groups of 2 to 4 (max) students. We recommend 2 to 3 students, but can allow 4 in some cases.

Delivery Date: January, 5th 2024

Interface to the Outside World

Your storage system should provide an interface to external clients. It can be either a standard REST API over HTTP that you can test with a tool like Postman (or a simple web application you create), or a ZeroMQ socket API that is called by a native client application. You are encouraged to use the techniques (and code) from the Labs.

There are only two mandatory functions your storage system has to support: **store** and **retrieve** a file.

Pointers:

- Keep the API and system simple to start with
- Consider using Flask for a REST API
- Consider using Protocol Buffers for a ZeroMQ-based interface
- Consider having a lead node that provides the API and coordinates the process

Task 1 - Replication Allocation

Design & Development

1. Implement a strategy that can generate k full replicas of a file and place them in N different nodes. Nodes can be implemented as different processes or containers. Consider splitting the file in four equal-sized fragments. Implement the scheme for a general k and N. The lead node sends the file fragments directly to each of selected nodes for each fragment.

2. Implement a **configurable** selection of the nodes supporting three alternatives: i) **random placement**, ii) min copysets placement, and iii) the buddy approach described in class.

Analysis

3. Calculate the time to generate all replicas and distribute them if the connection bandwidth is R bps (bits per second, same speed among nodes and between the client to any node). Use $k,\ N$ as variables in your analysis as well as files with B bytes.

4. Calculate the total download time of a file (of B bytes) with each scheme. Download starts when the client sends the request, and ends when the last byte of the file has finished transferring. Consider the same connection bandwidth as in Point 3.

Measurements

5. Compare the three selection approaches when ingesting different file sizes. Measure the cases of $k = 3, N = 3, 6, 12, 24$ focusing on full time generate all replicas and store them as well as the total download time. Plot, describe, and explain the results. Discuss the advantages of each approach.

Measurement Configuration

- Metrics:
    - Time from receiving a file in the storage system to successful generation of redundancy
    - Download time for a file, from request to download completion
- File sizes: 100kB, 1MB, 10MB, 100MB
- Measure each file size at least 100 times and report the measured times as a histogram, clearly marking the average and median.

Task 2: Data Loss Performance

Design & Development

1. Implement a control process that allows you to kill off or stop processes/containers representing each node. You should be able to indicate the number of nodes to be stopped, s. These nodes will be considered to be lost.

2. Implement a process/routine that quantifies the fraction of lost files in the system. A file is lost if it cannot be recovered after a loss. Consider and describe different technical approaches, e.g., heart beat monitoring, and implement one.

Analysis

3. Determine for what s values would you expect to see losses as a function of k, N for the different allocation schemes.

Measurements

4. Store 100 files of 100kB each. Use $N = 12$, $k = 3$. Compare the three allocation approaches when removing $s = 2, 3, 4, 6, 8, 10$ nodes. Measure what fraction of files where lost in the system..
5. Repeat the process with $k = 3$, $N = 24, 36$. Provide a figure that summarizes the results from step 4 and 5 for the three values of N. Discuss the results.