Distributed Storage Mini Project

This project is focused on understanding, implementing and comparing performance of a distributed storage system using different schemes for data redundancy: replication and erasure coding. 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 and run it on the provided Raspberry Pi Stack. Every team member should participate in all stages of the project: specification, system design, implementation, testing and measurements. The storage system has to run on the Raspberry Pi Stack mini datacenter, using all four devices.

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.
When the last team member completed the exam, the Raspberry Pi Stack must be returned to Prof. Daniel Lucani Rötter or one of the teaching assistants. Please make sure you do not leave any files on the stack that contain personal information.

Measurement Configuration

Use the following measurement configurations for each Task below.

- Metrics to measure:
  - Time from receiving a file in the storage system to successful generation of redundancy (all replicas or coded fragments are stored on the Pi's)
  - Processing time of encoding/decoding in the case of erasure coding
  - Apparent access time measured at the client, from requesting a file to receiving the last byte.
- File sizes: 10kB, 100kB, 1MB, and 10MB
- Measure each file size at least 100 times, remove the outliers (top and bottom 5%) and report the measured times as a histogram, clearly marking the average and median.

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

Design & Development

1. Implement a strategy that can generate k full replicas of a file and place them in different nodes (Raspberry Pi devices), with a general k, from the lead node, i.e., the lead sends the file directly to each of the other nodes. Selection of the nodes is random.

2. Implement a strategy that can generate k full replicas of a file and places them in different nodes (Raspberry Pi devices), with a general k, also using random placement strategy but where the lead node **delegates** to other nodes the generation of the next replica (similar to what Hadoop HDFS does).

Analysis

3. Calculate the time to generate all replicas of both schemes if the connection bandwidth is R bps (bits per second, same speed among nodes and between the client to any node).

4. Calculate the time for the lead node (the one originally receiving the file) to finish its replication task in both strategies. In which strategy is the lead node freed earlier to take in another request?

5. Calculate the total download time of a file of B bytes with each strategy. Download starts when the client sends the request, and ends when the last byte of the file has finished transferring.

Measurements

6. Compare the two approaches when ingesting different file sizes. Measure the cases of $k = 2, 3, 4$ and the time metrics: time to generate full replicas, time for the lead node to

complete its tasks, and download time. Explain the results and discuss the advantages of each approach.

Task 2: Erasure Coded Storage

Design & Development

1. Develop a coded strategy that tolerates $l = 1, 2$ node losses in order to later compare to the case of $k = 2, 3$ replicas above, respectively. Consider that the lead node performs the encoding/decoding of the data, for simplicity. Suggestion: consider splitting the file in two equal-sized fragments in both cases
2. Consider the case where the lead node randomly selects another node to carry out the encoding/decoding of the file.

Analysis

3. Calculate the time to generate redundancy of both schemes, if the connection from one node to another is R bps and a maximum outgoing/incoming data rate of R bps. Consider also that encoding/decoding can be carried out at R_{enc} and R_{dec} bps, respectively.

   Compare the case of $l = 1$ and $l = 2$ and meditate on the behaviour when more than 4 nodes are available.
4. Calculate the time for the lead node (the one originally receiving the file) from finishing its task in both strategies. When is the lead node freed earlier to take in another request?
5. Calculate the total download time of a file of B bytes with each strategy. Download starts when the client sends the request, and ends when the last byte of the file has finished transferring.

Measurements

6. Compare the two coded approaches when ingesting different file sizes and compare to the analysis. Consider the cases of $l = 1, 2$ and the two time metrics: time to generate full redundancy, time for the lead node to complete its tasks, encoding/decoding time, and overall access time at time of retrieval of data. Explain the results and discuss the advantages of each approach.

Task 3 - Cross comparison

1. Compare the measurements $l = 1$ of the coded case to the case of $k = 2$ replicas and explain the results. Consider the different variations of each coded scheme and reason about the implications in larger systems.

2. Compare the measurements $l = 2$ of the coded case to the case of $k = 3$ replicas and explain the results. Consider the different variations of each coded scheme and reason about the implications in larger systems.