Distributed System Report

Architecture

The design of the system is based on the Google File System. The system consists of a single Master and multiple Slaves and Clients can access the system to write and read data. The Master maintains the files’ Metadata. The Metadata of a file includes: encoded parts of the file and Slaves storing these parts, and the original file’s size. Clients interact with the Master for this metadata and communicate with Slaves for the actual data. Slaves are responsible for storing data objects sent to them and returning these objects when requested. Each Slave periodically sends its state to the Master through Heartbeat messages. The stored data object corresponds to a file and the encoding/decoding operations are executed on this object.



Figure 1 - The system architecture

Operations

1. Write operation

A write operation is executed in 3 steps. First, the Client makes a request to the Master to notify about this operation. The original file’s size is also included in this request for the decoding step in the future. The Master sends back the list of live Slaves where data will be written to. In the case of Hierarchical code, the number of Slaves in the list is 7. Each of these Slaves is associated to an encoded part which will be stored on that Slave. The Master itself records these pieces of information as the file’s Metadata before sending them. Then, the Client encodes the file into several parts using erasure codes. After that, the Client contacts to each of Slaves on the list and transfers the corresponding file name, part index and encoded data. The Slave stores this data as a file in its local file system with the name combined from original file name and part index (for example: a.txt.O1, a.txt.O1O2, etc.).

However, there may be failures in the communication with the Slaves. These failures may be caused by a communication problem or the fact that the Slave already died and the Master has not known about this (because of the delay in the maintaining process). We assume that there is only second type of failure in our experiments. In that case, the Client should ignore these errors because sooner or later the Master will know about this dead Slave. The Client does not have to acknowledge the Master about the failed Slave, which keeps the system still simple and efficient in failure handling.

1. Read operation

Similarly, a read operation includes 3 phases. At the beginning, the Client requests to the Master for the Metadata of the file. The Master returns all the live Slaves containing encoded parts as well as the original file’s size. Based on this list, the Client determines which Slaves it should contact to retrieve data and reconstruct the file. The Client can communicate to each Slave sequentially or concurrently for a better performance. Similar to the write operation, there may be a chance of failures because of 2 above reasons. In both cases, the Client has to recalculate to decide one or more other parts needed to recover data; therefore, more communications are required. With these collected pieces of information, the decoding process is executed by the Client. If the original file cannot be reconstructed in some fault scenarios, the read operation is unsuccessful.