Distributed Systems Project

Max Finney

January 28th, 2016

High-Level Overview

I implemented a basic distributed file system using Ruby on Rails. I managed to implement a

- Proxy for the client to use
- A distributed system that replicates and caches
- A directory server that manages the distributed systems
- A locking server

An example scenario of the system is a user via the proxy, wants to read a file on the system. The proxy gets the current logical clock value of the file from the lock server. It then downloads the file. A user can then write to the file. The proxy will update the distributed system. If a conflict is found using the logical clocks (the users clock is behind the systems clock), then the user is prompted if they wish to continue writing the file to the server. The proxy when writing also trys to acquire the lock for the file, after the lock is acquired the file is written to and the lock is released. This is used in order to make sure that there is a delay between different proxies editing the same file.
The distributed server replicates any files on it to other servers. Ensuring that everyone is up to date. Files are also cached when used.

Problems

Locking server isn't fully distributed and could be easily bypassed since it does not interact directly wiht directory or distributed file servers.
Proxy interacts directly with the distributed nodes, instead of a directory. This means that they system is not completely transparent.
The cacheing is currently unlimited, this would need to be restricted in the real world. This could be done by creating a que set so that the longer a object is unused the more likely it'll be removed from the cache.
If either the directory or the lock server went down, the system would crash.