Puddlestore

Advanced Implementation

Matt Kohn & Kei Kinoshita

Senior Capstone Abstract

"OceanStore is a utility infrastructure designed to span the globe and provide continuous access to persistent information. Since this infrastructure is comprised of untrusted servers, data is protected through redundancy and cryptographic techniques. To improve performance, data is allowed to be cached anywhere, anytime. Additionally, monitoring of usage patterns allows adaptation to regional outages and denial of service attacks; monitoring also enhances performance through pro-active movement of data." This is the description for OceanStore on the CS Berkeley website. For our senior capstone, we implemented two A-Level features together on the Introduction to Distributed Systems final assignment, Puddlestore, based on this distributed file system.

Narrowly defined, a file system is the storage utility, usually provided by the operating system, where applications can read and write persistent data (information that is still available after the program has ended). Usually, persistent storage utilities are backed by a disk. In Puddlestore, all data resides in memory and is distributed among the nodes of the system. We built a functioning version of this distributed file system that supported reading and writing to memory, construction of files and directories, opening files, and listing the contents of directories. Puddlestore utilizes the Tapestry protocol for its distributed hash table that stores the components of the files and directories, as well as the Raft paradigm for passive replication of data for distributed consensus.

For the advanced implementation, we also completed log compaction for the Raft component, incorporated ZooKeeper as a membership service for the Tapestry and Raft nodes, and actualized salting and hotspot caching for the Tapestry component. Log compaction reduces the memory used by the Raft nodes to store logs by creating a snapshot of consistent log entries and storing that instead. Salting hashes an object at several Tapestry nodes, ensuring reliability in

the case of a node failure for any specific node (so there are multiple copies stored at multiple nodes). Hotspot caching stores objects at specific nodes that request those objects frequently, improving performance speeds for certain lookups.