SAHAKARA: A Scalable and Secure Cooperative-Caching File System

Siddhartha Annapureddy, Michael J. Freedman, David Mazières – NYU

Goal: A file server that can handle thousands of clients

Serverless File Systems

- ✓ Massive scalability via replication
- ✓ Cross-FS cache sharing possible
- ✗ No accountability for files
- ✗ Significant administrative complexity
- ✗ Elaborate authorization mechanisms
- ✗ Consistency issues

Traditional File Servers

- ✗ Cannot scale to large client populations
- ✗ Cross-FS cache sharing not possible
- ✓ Accountability for files
- ✓ Ease of administration
- ✓ Simple authorization schemes
- ✓ Simple consistency mechanism

Insight: Obtain the “best of both worlds” of traditional servers and serverless systems

Potential Applications

- Easily install files on all PlanetLab nodes
- Transparent mirroring of data, *e.g.* openbsd-current
- Building load-balanced server farms

With Sahakara, these tasks are as simple and as user-friendly as mounting a remote file system

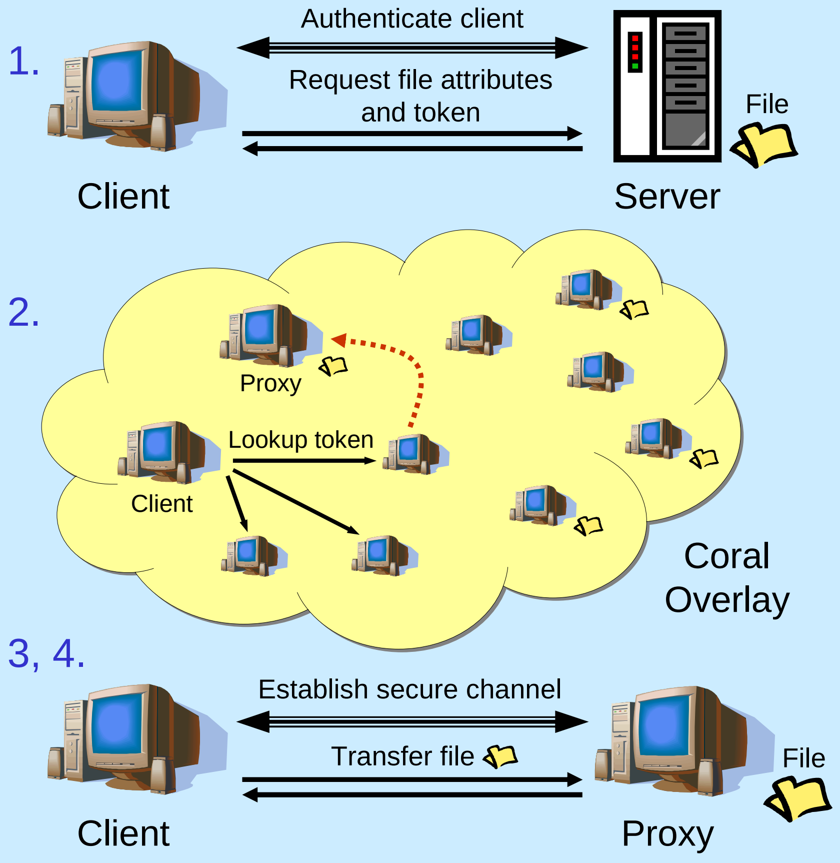
Key Features

- A server-based file system
- Servers have self-certifying names
- Replication via client caches
- Whole-file caching by clients
- Writes synchronize at server
- Simple lease mechanism
- Clients discover nearby proxies
- Clients use secure channels

How Sahakara Works

To read a file, a client:

1. Fetches the file attribute information and file token from an SFS-like file server
2. Queries the Coral indexing infrastructure using the file token to obtain a list of nearby proxies
3. Selects a proxy and establishes a secure channel using the token as a symmetric key
4. Obtains the file and verifies its integrity using the token
5. Caches the file locally and announces itself in Coral as a proxy for the file


1.
Authenticate client
Request file attributes and token
File
Client
Server
2.
Proxy
Lookup token
Client
Coral Overlay
3, 4.
Establish secure channel
Transfer file
File
Client
Proxy