6.824 - Spring 2012

6.824 Lab 8: Project

Due: Friday, May 11th, 5:00pm.

Introduction

As the final lab in 6.824, you will complete a small coding project of your choice related to the class material somehow, submit a short writeup on what you did, and present a demo to the class staff (and possibly the whole class).

You are free to choose any project you like, as long as it has something to do with distributed systems, consists of original work done for the class (*i.e.*, don't simply recycle a past research project and try to put a 6.824 spin on it), and is of an appropriate scope (see below). We suggest that you take your now functional fault-tolerant, distributed file system and extend it in some interesting way.

We expect your project to continue in the tradition of the labs: find some real problem with an existing system (such as YFS), code up a solution to the problem, and present what you did. See the "Examples" section for some possible projects.

Requirements

Your project should meet the following requirements:

- **Collaboration**: Please do your project in a team of two or three people. You will submit one writeup per team. If your team chooses to do a YFS-based project, you will have to choose one member's implementation as the starting codebase. Free free to use Piazza to find a partner.

  Please email 6.824-staff by **5:00 pm on Tuesday, May 1** with the names of your team members and a few sentences describing what you plan to do.

- **Scope**: We expect the programming aspect of your project to be similar in difficulty to one of the 6.824 2-week labs, such as Lab 4 or Lab 6, adjusted to reflect the number of students working on the project. Please email the staff with your project idea if you'd like to know what we think about its suitability. Note that while you will not be required to hand in your code for the project, you will have to meet with the TA and intelligently discuss what you did.

  Again, the project must have some relation to the class material. Any YFS extension of suitable scope is fine, as is any other project related directly to one of the papers

or topics covered in class. If you have a project in mind that is only tangentially related, please check with us first.

- **Write-up**: A brief 1-page writeup on your project is due by the lab deadline stated above. This should explain what problem you solved, how you solved it, and how successful you were in solving it. Please provide enough detail for us to get a sense of how much effort went into your project, but don't write a complete, formal research paper. You may want to also include a measure of the number of lines of code you wrote.

- **TA Demos**: Each team will meet with the TA for five or ten minutes to discuss their project and to demonstrate their code. These meetings will take place on Monday, May 14th, or Wednesday, May 16th. We will send out an announcement about scheduling these meetings when the deadline gets closer.

- **In-Class Demos**: The final lecture of the semester will be dedicated to project demos. There may not be enough time to fit all groups into 80 minutes; we will make a plan when we know how many groups there are.

Examples

Below are some example extensions to YFS that would make suitable projects. Feel free to use one of these if you have no idea of your own.

- **Complete fault-tolerance**: Extend YFS to be tolerant of arbitrary client failures. This will involve some plan of lock expiration (*e.g.*, leases), persistent logging of client operations, and crash recovery, all in the spirit of Frangipani. You may also need to come up with a more robust scheme for RPC sequence numbers, as you don't want a restarted client to re-use the same sequence numbers it was using before it crashed. You should also replicate your extent server using the Paxos-based replicated state machine from Lab 6, making your YFS impervious to failures at all levels.

- **Distributed storage**: At the moment your YFS implementation just uses a single extent server to store its data. It would be much more useful, for load-balancing purposes, if YFS could spread its data across many extent servers. You can implement the Petal design, consistent hashing as in Chord, voting as in FAB, or use a scheme of your own. Show some benefit of this system under a real workload, either in terms of performance or fault tolerance.

- **Peer-to-peer data transfer**: Implement a protocol in which nodes do not have to write dirty extents back to the extent server, but can instead transfer them directly to the next YFS server that wants to access the extent. You should also implement read/write locking, allowing multiple peers to access the extent for reading at the same time. Demonstrate some scenarios where the new scheme outperforms the old scheme.

- **SUNDR**: Implement the strawman proposal of SUNDR (plus whatever other optimizations you have time for) within the context of YFS, and demonstrate that the system provides fork consistency. You will need to modify YFS to keep signed logs of extent accesses, and extend your FUSE interface to have some notion of a user and the user's associated keys.

- **Bandwidth- and disk-space efficient extent service**: Currently, your YFS implementation probably stores whole files as extents in memory, and returns the entire file on each get. The goal in this project would be to reduce the network traffic between the extent server and its clients, as well as reducing the memory and disk usage of the extent server. You should come up with an efficient scheme for laying out extents on disk (for example, use a single file on the native file system to store all of the extents, and do your best to reduce fragmentation of the space within that file); take both large and small files into consideration. You may also want to use compression to limit the size of the data on disk. Furthermore, there are known techniques for how to reduce bandwidth between the server and clients by sending only diffs of data, rather than the full extent (see the LBFS paper, for example); you should implement some similar scheme for YFS. You may also want to consider storing files in blocks, rather than one huge extent, if you think that makes sense. Demonstrate the performance and space benefits of your modifications.

- **Access control**: Implement a robust notion of access control in YFS. As in the SUNDR project, you'll have to extent your FUSE interface to know about users (and groups!), and also make the appropriate extensions to your YFS data structures. You'll also need to add support for any related file system calls (*i.e.*, `chmod`, `chown`, `chgrp`, etc). Once this is in place, you should consider encrypting the files on disk for per-user or per-group data privacy.

- **Disconnected operation**: Since your YFS server already supports the caching of extents, you can imagine a version of YFS that will allow the user to make modifications to cached data, and create new files and directories, even if it cannot currently talk to the lock server or the extent server. To implement this, you'll probably need to implement leases on locks, and come up with some plan for merging divergent extents once the client comes back online. Check out the Coda paper for inspiration on this topic. A cool demo would be to mount YFS on a laptop (where the extent and lock servers are running on a server somewhere), cache some data locally, disconnect the laptop from the network, and continue working on the files. Then once the laptop is back online, mount the directory from a different location and ensure the changes made offline have propagated to the extent server.

Handin procedure

You need to email 6.824-staff by **5:00 pm on Tuesday, May 1** with the names of your team members and a few sentences describing what you plan to do.

E-mail your writeup in PDF format to 6.824-submit@pdos.csail.mit.edu by the deadline at the top of the page. Please name the file `<username1-username2>-project.[pdf,ps]`. You will also need to schedule a meeting time with the TA on May 14th or 16th, and you should be prepared to give an in-class demo of your project on May 17th.

Please post questions or comments on Piazza.
Back to 6.824 home.