6.824 - Spring 2015

6.824 Project

Proposals due:	Weds Apr 1 11:59pm
Code and write-up due:	Fri May 8 11:59pm
Presentations:	Thu May 14 in class

Introduction

This project is optional. You should do Lab 5, or do this project, but not both.

If you choose to do a project, you get to choose what to build, subject to our approval. You must form a group of 2 or 3 6.824 students to collaborate on the project. You'll turn in your code and a short write-up describing the design and implementation of your project, and make a short in-class presentation about your work. We will post your write-up and code on the web site after the end of the semester, unless you explicitly talk to us about why you want to keep yours confidential.

Your project should be something interesting and challenging that's closely related to 6.824 core topics, such as fault tolerance. Below you'll find some half-baked ideas that we think could turn into interesting projects, but we haven't given them too much thought.

Deliverables

There are four concrete steps to the final project, as follows:

1. Form a group and decide on the project you would like to work on. Feel free to use Piazza to find group members and discuss ideas. Course staff will be happy to discuss project ideas via e-mail or in person.
2. Flesh out the exact problem you will be addressing and how you will go about solving it. By the proposal deadline, you must submit a proposal (less than a page) describing: your **group members** list, **the problem** you want to address, **how you plan to address it**, and what are you proposing to **specifically design and implement**. Submit your proposal to https://6824.scripts.mit.edu:444/submit/handin.py/. We'll tell you whether we approve, or not, and give you feedback.
3. Execute your project: design and build something neat!
4. Write a document describing the design and implementation of your project, and turn it in along with your project's code by the final deadline. The document should be about 3 pages of text that helps us understand what problem you

solved, and what your code does. The code and writeups will be posted online after the end of the semester.

5. Prepare a short in-class presentation about the work that you have done for your final project. We will provide a projector that you can use to demonstrate your project. Depending on the number of project groups, we may have to limit the total number of presentations, so some groups might not end up presenting.

Half-baked project ideas

Here's a list of ideas to get you started thinking -- but you should feel free to propose your own ideas.

- Build a distributed, decentralized, fault-tolerant reddit.
- Make the state synchronization protocol (DDP) in Meteor more efficient (e.g., send fewer bytes between server and client) and more fault-tolerant (e.g., a client should be able to tolerate server failures, as long as enough servers remain live).
- Build a fault-tolerant file service; on the client side, you could use FUSE to run your own client code, or you could have clients talk NFS to your server, as in Harp.
- Build a better fault-tolerant peer-to-peer tracker for BitTorrent.
- Build a system for making Node.js applications fault-tolerant, perhaps using some form of replicated execution.
- Improve the Roost Javascript Zephyr client by replicating the backend to make it fault-tolerant.
- Use Mylar to build a secure webmail system with end-to-end encryption, contact lists, usable public key infrastructure, etc.
- Add cross-shard atomic transactions to Lab 4, using two-phase commit and/or snapshots.
- Build a system with asynchronous replication (like Dynamo or Ficus or Bayou). Perhaps add stronger consistency (as in COPS or Walter or Lynx).
- Build a file synchronizer (like Unison or Tra).
- Build a distributed shared memory (DSM) system, so that you can run multi-threaded shared memory parallel programs on a cluster of machines, using paging to give the appearance of real shared memory. When a thread tries to access a page that's on another machine, the page fault will give the DSM system a chance to fetch the page over the network from whatever machine currently stores.
- Build a distributed RAID in the style of FAB. Maybe you can get standard operating systems to talk to you network virtual disk using iSCSI or Linux's NBD (network block device).
- Build a coherent caching system for use by web sites (a bit like memcached), perhaps along the lines of TxCache.
- Build a distributed cooperative web cache, perhaps along the lines of Firecoral or Maygh.

- Build a collaborative editor like EtherPad.