EEL-4636/EEL-5737
Principles of Computer System Design
Final Project - Assigned: 10/23/2015
Due date: 12/7/2015 5pm

This is the final project for this class. You may work individually, or in a group of 2-3 students for this project. (See note below for difference in project scope for EEL-4636)

You will take the FUSE filesystem that has been developed over the semester and extend it to support advanced features – server replication for performance and fault tolerance. The design assumptions and requirements are as follows:

- Assume that there is only a single client, a single file meta-data server, and multiple replica servers for file data (Nreplicas)
- Each file data replica server stores a copy of each file block
- The system supports a quorum approach – parameters determine the minimum number of servers that need to reply to a read request (Qr, configurable between 1 and Nreplicas) and a write request (Qw=Nreplicas) (see textbook section 10.3, in particular 10.3.5 for a discussion on quorum replication)
- The system should tolerate the following failure modes:
    - Server crash, restart – where one or more servers may fail, and a replacement server may start from a blank slate (i.e. without any file data)
    - Data corruption in a single server (EEL-5737 students only) – where data contents in a server has been be corrupted. You can assume that data corruption errors are only tolerated if Qr >= 3
- The repair mechanism should be done at the server side and transparent to the client
- The system may block on a write request if there are unavailable servers, but should be able to respond to a read request as long as there are at least Qr servers available

You will build upon and extend the code used in the solution given for homework #2, and simpleht.py. We will provide instructions on how to name and configure the servers and client in a uniform fashion (and sample test scripts), to facilitate automated grading. These will be described in Sakai at a later date.

While the use case is a system where replicas run on different physical computers, you can run client and servers processes in a single (virtual) machine for all your development and testing. However, your client and servers should communicate only through network protocols, such as XML-RPC (not through shared memory, or the file system).

Your primary goal is to achieve functional correctness – while performance enhancements are welcome, your project will be graded on the basis of functionality, not performance.

1. Final submission. This includes your software and a 10 to 20-page document describing your system, modeled after a typical technical report. Due **12/7/2015 via Sakai submission**. Requirements are as follows:
   a. Code – Good use of modularization, limiting complexity, and readability; examples of these can be shown by looking at the source code given as solution to homework #2.
   b. Documentation (submitted as a PDF):
      i. Background – Describe your problem, why it is interesting – cite papers, web sites, or other documentation. Remember your solution does not need to be novel.
      ii. Implementation – Your implementation documentation should make it so that your code does not need to be read to appreciate your work. It should also not be so overloaded with information that reading the source code would be easier than reading the documentation itself.
      iii. Testing mechanisms – When implementing new projects, tests must be established that focus on ensuring that no bugs were created. Identify potential problem areas and describe the tests used to verify that those problems do not exist. The purpose of a testing mechanism is to verify that there are no functional regressions. Example: test.py.
      iv. Evaluation mechanisms – What are the benefits of your work? Design a scheme that evaluates your work with the previous approach. In some cases, this may have overlap with the testing mechanism. The point of this portion, though, is to show the enhancement, whereas a testing mechanism is used to verify there are no regressions. Example: time for very_long_io to execute in hw3.
      v. Evaluation – Present your evaluation, showing graphs and describing the interesting results from your work
      vi. Potential issues – List any potential issues that exist with the addition of your new feature as well as any components that were left undone

Submit your final work via Sakai as follows:

- Submit only PDF and .py files for your documentation and code
- Zip all your files
- If submitting as a group, only one person needs to submit
- Non-conformant submissions will lose significant points