EEL-4736/EEL-5737 Principles of Computer Systems Design

Homework #3 Assigned: 9/28/2015

Part1 due 10/5/2015 Part-2 due 10/19/2015

This assignment is divided into two parts, Part-1 will test your understanding of the concepts covered in class by working out problems and Part-2 will involve extending the implementation in homework 2 to support persistent storage and implementing a caching layer in the File-System.

Part-1 (Due- 10/5) To be done individually

Textbook exercise 5.2

Textbook exercise 5.6

Textbook problem set 8

Textbook problem set 9

Submission

submit a pdf file named homework3part1.pdf with solution to the above problems.

Part-2 (Due- 10/19) Can work in a team of two students

In homework 2 you implemented a hierarchical FS where storage was hosted on a dictionary exposed via the simpleht server. One of the limitations of that implementation was its inability to provide persistent storage i.e. a restart of the server would have wiped out all the contents stored on it, as the dictionary used was a in-memory data-structure of the simpleht process.

In this assignment you are going to replace the simpleht service by a "NoSQL" database that provides persistent storage. We suggest that you use MongoDB, given it has open source python client packages that you could reuse in your code. After you have made the replacement, test your FS functionality to ensure nothing is broken. Using persistent storage comes at the cost of performance, i.e. the time taken to perform file operations is bound to increase compared to when you were using in-memory data-structures. This cost could be alleviated to some extent by implementing a caching layer in the FS, analogous to caches in the processors. One way of implementing a cache is to use memcached (an open source software to provide the cache-storage), or you could use your own in-memory data-structure in the Filesystem. Caching is a limited storage resource which should be used with care- you can only put a limited number of objects in the cache (say you can only store contents for 5 files), which implies that you are required to write policies to manage the cache. For simplicity in this assignment we assume that the granularity is at the level of a file, i.e. you can only store a limited number of files in the cache.

In addition, the cache has to be synchronized with the main storage to ensure that it does not contains stale content. In essence, you are to implement a caching layer in your file-system complete with the

management policies- for instance, LRU- and justify it by demonstrating the performance improvement by having a cache by comparing it with the implementation having only persistent storage.

The assignment is partly open-ended to allow you to explore the design of both the implementation and the experiment that you would use to demonstrate the effectiveness of your cache. It will also expose you to two open-source object storage/caching services that are widely used in practice (MongoDB and memcached). You are responsible for going through the documentation of both these software packages and to configure your own virtual machine environment with them. While most of the caching complexity is already handled by memcached, textbook's section 6.2 can be used as a reference for a discussion on caching and replacement policies. The assignment will be graded based on the correctness of FS operations, design of the implementation, and the performance improvements. Partial credit will be given if you only complete one of the two aspects of the assignment (persistent storage and/or caching).

Submission

The submission must be a zip file containing the below--

1. submit a pdf file named homework3part2.pdf detailing the design of your implementation and the experiment along with results.
2. Also submit all the source code files, both for the implementations and the automated test scripts.
3. Only one member of the team is required to submit the solution.