15-440: Distributed Systems
Problem Solving Assignment 5

School of Computer Science
Carnegie Mellon University, Qatar
Fall 2014

Assigned date: November 19, 2014

Due date: December 4, 2014

I) Fault-Tolerance & Recovery (35 Points):

1. For each of the following applications, discuss whether the *at-least-once* or the *at-most-once* semantic is more suitable.

   a) Reading and writing files from a file server.
   b) Compiling a program.
   c) Remote banking.

2. The Two-Phase Commit (2PC) discussed in class is a solution to the general *distributed commit problem* which states that an operation performed by a group of processes is either performed by all group members, or none at all.

   Consider a distributed transaction involving the participation of two processes, A and B. Each process runs on a different machine installed with a database. The transaction is split amongst A and B, and cannot be committed until both A and B have committed their parts of the transaction.

   Now, A and B successfully execute their parts and send the coordinator a VOTE_COMMIT message implying that each is ready to locally commit its part. In return, the coordinator initiates a GLOBAL_COMMIT message affirming to proceed with the commit. While A receives the GLOBAL_COMMIT message and performs the commit, just before the GLOBAL_COMMIT message arrives at B, it crashes. Answer the following questions precisely and concisely:

   a) When B revives, which stage in the 2PC protocol does B resume from? Explain.

   b) Once B has resumed its current stage, does it perform the usual sequence of operations in that stage? Explain.

   c) Does B ever learn about the GLOBAL_COMMIT message sent by the coordinator? Explain.

II) Distributed File Systems (25 Points):

1. Suppose we have a file server that connects to a storage server via a storage area network (SAN). The storage server provides a collection of disk blocks for use by the file server. The file server uses these disk blocks to store files, and provides these files to its clients. Since its clients are fully capable of implementing a file system from the disk blocks provided by the file server, is the file server really needed? Can't the clients deal directly with the storage server? Explain.
2. An approach for implementing file locking in distributed file systems (DFSs) is to maintain the locks on the server in volatile storage (i.e., storage whose contents disappear in the event of a crash). Thus, after a server crashes, the server must recover its prior state by obtaining information from its clients about who had which files locked. If servers maintained locks in non-volatile storage, then, it would seem, state recovery after a crash would not be necessary. Explain why a DFS like NFS (for instance) does not maintain such lock information in non-volatile storage.

III) Writing Critiques (40 points):

1. Read the paper titled "***The Google File System***" by Sanjay Ghemawat *et. al.* and answer the following questions:
    a) Summarize the paper in 2 paragraphs.
    b) Despite the fact that the Google File System scales well, it could be argued that the master is a potential bottleneck. What would be a reasonable alternative to replace it?
    c) State 3 strengths and 3 weaknesses in the paper.
2. Read the paper titled "***MC2: Map Concurrency Characterization for MapReduce on the Cloud***" by Mohammad Hammoud and Majd Sakr, and write a critique about it.