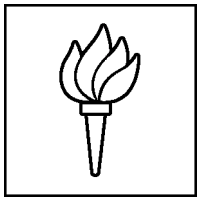

Computer Science Department

New York University

G22.3033-001 Distributed Systems: Fall 2009

Quiz II

All problems are open-ended questions. In order to receive credit you must answer the question *as precisely as possible*. You have 100 minutes to answer this quiz.

Some questions may be much harder than others. Read them all through first and attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit!

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.

I (xx/20)	II (xx/30)	III (xx/10)	IV (xx/20)	VI (xx/10)	Total (xx/90)

Name:

I MapReduce

Ben Bitdiddle gains access to a cluster of 100 nodes. Ben aims to run the wordcount MapReduce computation on the cluster with 200GB of input data. Ben configures the MapReduce job to divide the input into 1000 map tasks and to use 100 reduce tasks (i.e. reduce partitions).

1. [10 points]: Ben notices that the reduce phase of his MapReduce computation takes a long time to finish. In addition, if a reducer happens to fail, the overall reduce phase would take even longer. Alyssa P. Hacker advises Ben to increase his reduce tasks (partitions) from 100 to 200. Explain why Alyssa's suggestion might speed up the reduce phase both in normal circumstances and in scenarios when some reducers fail.

Ben takes Alyssa's advice to the extreme and increases the number of reduce tasks to 4000. The number of map tasks remains the same (1000) as before. Ben was expecting better performance and is disappointed to find out that the overall MapReduce computation actually takes much longer than before. Ben checks the usage statistics of the cluster and finds that the network and CPUs are not fully utilized but the disk activities on all machines are very high. Ben asks Alyssa to help find the likely cause for this slowdown.

2. [5 points]: What type of intermediate data does MapReduce store? How and where does MapReduce store them?

3. [5 points]: Alyssa tells Ben that his new configuration makes MapReduce unable to write and read intermediate data efficiently, resulting in slowdown. Is Alyssa correct? Please explain using *concrete numbers* based on Ben's MapReduce configuration and typical hardware performance numbers.

II Paxos

4. [10 points]: Ben Bitdiddle decides to implement a different join protocol than that in Lab 7. Recall in Lab 7's join protocol, a joining node X asks the primary (in the latest view known to X) to initiate Paxos that will include X in the new view. In Ben's protocol, a joining node X directly initiates Paxos by becoming a proposer itself and seeks a majority of "accepts" messages from all nodes in the current view *plus itself*. For example, if the current view is {A,B} and node X joins, the Paxos protocol inititated by X will succeed in deciding on the new view when any majority of nodes in {A,B,X} accepts the new view.

Is Ben's new join protocol correct? If yes, give a proof sketch. If no, give a concrete counter-example where his protocol gives incorrect result.

5. [10 points]: Ben decides to simplify the Paxos protocol and speed up the decision process. He proposes to change Paxos by having a proposing node use its node identifier as the proposal number. As a result, the node with the biggest node identifier will always be able to win over other potential proposers. In contrast, in the original Paxos design, a proposer generates a proposal number by concatenating an (increasing) integer together its node identifier instead of just using the node identifier alone. Therefore, unlike Ben's design, the original Paxos does not ensure that any one node will always win over other potential proposers.

Is Ben's change desirable? If yes, explain its advantages over the original Paxos. If no, give a concrete example to explain why the original Paxos is more preferrable.

6. [10 points]:

Ben and Alyssa P. Hacker are discussing about the need for every Paxos node to log the highest proposal number that it has accepted (n_a) and the corresponding accepted value v_a on disk. Alyssa agrees with Ben that logging is essential for the correctness of the original Paxos design. However, she also thinks that it is *not* necessary to log n_a and v_a if a recovered node (after a previous crash) always joins the system with a new unique node identifier.

Is Alyssa correct? If yes, give a proof sketch and discuss *the pros and cons* of Alyssa's new design with the original Paxos that requires logging of n_a and v_a. If no, give a counter-example.

III Replicated State Machine

In Lab 8's RSM, the primary processes a client's request by first assigning it a sequence number and sends the request to all other replicas. The primary waits for the acknowledgements of all non-primary replicas before executing the request locally and returns the reply to the client. Ben proposes to reduce the latency of RSM by having the primary wait for the acknowledgements of a majority of non-primary replicas (instead of all non-primary replicas). In Ben's new design, the primary assigns a client request a sequence number and sends the request to all non-primary replicas. However, as soon as the primary receives acknowledgements from a majority of non-primary replicas, it executes the request and returns the reply to the client.

7. [10 points]: Is Ben's improvement correct? If yes, give a proof sketch. If no, give a counter-example that shows an incorrect outcome. (Hint: Recall that Lab 8 copes with node failures as follows: first, upon detection of a node failure, existing nodes use Paxos to decide on the next view (and the corresponding new primary). Second, non-primary nodes in the new view synchronize their state with the new primary before start processing requests. The new primary waits for all non-primary nodes to finish synchronizing state before start processing client requests itself.)

IV Consistency in Dynamo/BigTable

8. [10 points]: Dynamo is an eventually-consistent key-value storage system. Using the shopping cart as an example application, describe a scenario where the weak consistency model causes the application to produce an ”abnormal” or ”confusing” result to the end-users.

9. [5 points]: One important design decision of Dynamo is that nodes do not necessarily have the same view of the current set of available (or live) nodes. Explain why this design decision causes the overall Dynamo storage system to lose any guarantee for strong (sequential) consistency.

10. [5 points]: Does BigTable guarantee that all nodes have the same view of the current set of available tablet servers? If yes, how does BigTable achieve such guarantee? If no, describe a concrete scenario where two nodes have different views.

V G22.3033-001

11. [5 points]: Describe the most memorable error you have made so far in one of the labs. (Provide enough details so that we can understand your answer.)

We would like to hear your opinions about the class so far, so please answer the following two questions.

12. [3 points]: What is the best aspect of this class?

13. [2 points]: What is the worst aspect of this class?

End of Quiz I. Happy Holidays!