UMEÅ UNIVERSITY, DEPARTMENT OF COMPUTING SCIENCE

Written Assignment 2 (30 points)

Distributed Systems - 5DV147

You need to complete and submit this assignment individually. Collaboration is permitted only as described in the Guidelines for compulsory assignments and the Honor code. You need to submit this assignment by October 3 at 14:00 (hard deadline). You can either email your solution to us (5dv147-staff(at)cs.umu.se) or drop off your solution in the box outside the department.

1 WEAK CONSISTENCY (4 POINTS)

- Explain in your own words what is the reason for considering weak consistency models. (**2 points**)
- It is often argued that weak consistency models impose an extra burden for programmers. To what extent is this statement actually true? (**2 points**)

2 CAUSAL CONSISTENCY (10 POINTS)

During class we studied *Causal consistency*. In this model all writes that are causally related must be seen by every process in the same order, reads must be consistent with this order, and writes that are not causally related to one another can be seen in any order.

Propose an approach for implementing a causally consistent data store. As part of your answer you should identify the key difficulty(s) that you need to consider and explain how to overcome those difficulties.

3 Quorums (9 Points)

A quorum-based replica management system consisting of **N** servers engages only a designated fraction of replicas for every read and write operation. These are known as read quorum **(R)** and write quorum **(W)**.

Remember that in sequential consistency all writes must be seen in the same order by all processes. Assuming that **N** is an even number,

- Will sequential consistency be satisfied if **W = N/2, R = N/2**? (Justify your answer) (**3 Point)**
- Will sequential consistency be satisfied if **W = (N/2) + 1, R = (N/2) + 1**? (Justify your answer) (**3 Point)**
- If **N = 10, W = 7**, then what should be the minimum value of **R** so that sequential consistency is satisfied? (**3 Point)**

4 System Performance (7 points)

As we saw in class, high level requests such as loading a web page, fetching a data file from the network, or querying a database, depend on a number of low level operations. Each low level operation adds a little bit of latency to the overall high level request latency.

When you program web applications, it is very important that requests are served with very short latencies . If you were to design an on-line store, you would lose many customers if it takes a minute to check whether certain product is available. Thus, it is very important for you to understand where the time is spent so that you can optimize at the right places.

Taking into account the latencies presented in class[1] and reproduced above[2], propose 3 strategies that you could use to reduce latency in a web application that you (theoretically) have developed.

[1] From "Latency numbers every programmer should know", Jeff Dean

[2] You can also take a look at `http://www.eecs.berkeley.edu/~rcs/research/interactive_latency.html` if you want to know how those numbers have varied in the past (and into the future).

Operation	Latency (ns)
L1 cache reference	0.5
Branch mis-predict	5
L2 cache reference	7
Mutex lock/unlock	25
Main memory reference	100
Compress 1K bytes with Zippy	3,000
Send 2K bytes over 1 Gbps network	20,000
Read 1 MB sequentially from memory	250,000
Round trip within same datacenter	500,000
Disk seek	10,000,000
Read 1 MB sequentially from disk	20,000,000
Send packet CA->Netherlands->CA	150,000,000

Table 4.1: Latency numbers every programmer should know.