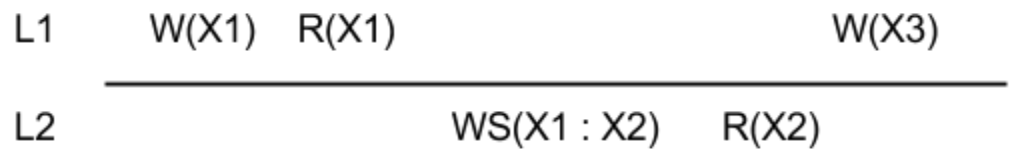
Assignment 6 - ECE 454

Piyush Gadigone, 20366910
Yash Malik, 20379044

1.

No, if a data store provides “Read your Writes” property, we cannot infer that it provides the “Writes follow Read” property. Consider the following scenario:

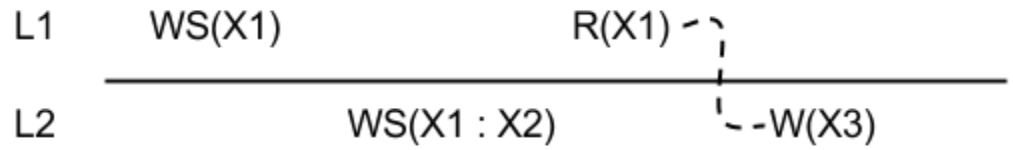


This meets the “Read your Writes” property as the Read always gets the value from the latest Write. It however does not meet the “Writes follow Read” property as the W(X3) write does not see the X2 change, which was the latest Read.

2.

No, if a data store provides the “Writes follow read” property, it is not guaranteed to provide the “Read your writes” property.

Consider the following scenario:



Here, the “Write follows reads” property is followed. The “Write follows reads” property states that a write operation by a process on a data item x following a previous read operation on x by the same process, is guaranteed to take place on the same or a more recent value of x that was read. L1 reads X1 and L2 writes X3, which is on the most recent value of X. But, the “Read your writes” property is violated when X1 is read at L1, as L2 writes X2, and L1 should have read X2.

3

a)

Lets say the total number of servers is *i* and the number of clients is *n*.

- Using brute force, the first step is to figure out all possible combinations of servers. This would have a run time of $O(2^i)$. Also, Each server has a variant cost f where the number of bits allocated for the cost is $\log_2(f)$.

- Now consider a single combination, S_j. For each client, we have to loop through all the servers in S_j and select the server which gives the minimum cost. This would have a run time of O(i*n) in the worst case.
- Finally, we have to loop through all the combinations of servers and find the combination that gives the minimum cost.

Hence the run time for this entire algorithm would equal $O(2^i \log_2(f)) * O(i*n) = O(2^i \log_2(f)*i*n)$.

b)
No, the above algorithm is not efficient if the cost is constant. If the cost (f) is constant, $\log_2(f)$ is constant and hence $O(2^i \log_2(f)*i*n)$ simplifies to $O(2^i*i*n)$. This is still exponential to the number of servers.

4)
Yes, there is an efficient algorithm for the original k-median problem if we have an oracle that outputs the minimum cost given an input of *i* servers, *n* clients, and *k*.

- The first step of the algorithm is to figure out the set of servers that yield the minimum cost. This can be done in O(i) as follows:
  Pass in the input of *i* servers, *n* clients and *k* to the oracle to get the minimum cost. Now for each server S_j, remove it from the set of servers and see the cost output by the oracle. If this cost is higher than the minimum cost, you add S_j back into the set of servers. Else you discard S_j from the set of servers. After the loop is completed, you will have a set of servers that must be used to attain the minimum cost.
- The next step is to find a mapping between the selected servers and *n* clients. As described in question 3, this can be done by having a nested for loop that runs in the worst case time complexity of O(*i***n*).

Combining these two steps, the time complexity of the algorithm is O(i)+O(i*n) = O(i*n). This is an efficient polynomial time algorithm.

5) Assume all $n_1 + n_2$ accesses go to P.

a)
P would drop a file if the storage cost of F exceeds the cost of doing all $n_1 + n_2$ accesses from Q via P. In the question, no storage cost is given. Hence we can assume that the storage cost is 0, and hence P would never drop F. Hence the drop-threshold is 0.

b)
P would replicate a file if the cost of accessing F from Q via P exceeds the cost of replicating F at P plus the cost of accessing F directly from P.

Cost of $n_1 + n_2$ accesses of F from Q via P = $(p + q)(n_1 + n_2)$
Cost of $n_1 + n_2$ accesses of F directly from P = $p(n_1 + n_2)$
Cost of replicating F at P from Q = e

If $(p + q)(n_1 + n_2) > p(n_1 + n_2) + e$, then P replicates file F.

$(p + q)(n_1 + n_2) - p(n_1 + n_2) > e$
$q(n_1 + n_2) > e$
$(n_1 + n_2) > e/q$

The replicate threshold R is defined as the number of requests before F is replicated at P. Hence, R is e/q.

6)
a)
Proving that a client always reads the latest version of x can be done by contradiction. Lets assume that a client does not read the latest version of a file F from N_r servers. This means that the latest write on servers N_w does not include any of the servers in N_r. This means that $N_r + N_w <= N$. This contradicts what is given in the problem, thus a client always reads the latest version of x.

b) Yes, we have causal consistency across clients. Causal consistency states that Writes that are potentially causally related must be seen by all processes in the same order. In this case since there is no propagation delay between servers and since updates are immediate, a Read operation by any client that follows a Write operation will always yield the latest result.