1. Kleppmann Chap 5

a) When should you use multi-leader replication, and why should you use it in these cases?

Ans: The answer would be No, in most cases. As, multi leader replication is hard to maintain and there are/could be a lot of update loss. But there are some cases like when you have multiple data center and acting like GEO data center to serve different target group, then its fine. Also, some f the benefits it provides like faster write and better fault tolerance.

When is leader-based replication better to use?

b) Why should you use log shipping as a replication means instead of replicating the SQL statements?

Ans: Yes, we should, as if we use SQL statement, for replication, we have to face a lot of problems. For example: nondeterministic functions, like Rand (), Now () or concurrent updates or autoincrement ids or trigger would cause a lot of problems. As these things are not constant and mostly depends on local machine.

2. Kleppmann Chap 6

a) What is the best way of supporting re-partitioning? And why is this the best way? (According to Kleppmann).

Ans: According to Kleppmann, the best way to do re-partitioning is to use range partitioning as in hash-based partitioning, it's a lot of data moving and can affect the performance during concurrent queries. On the other hand, If a new node added to the system/configuration you probably just split your data and move them to the new node, though for doing this – the nodes need to know their data range.

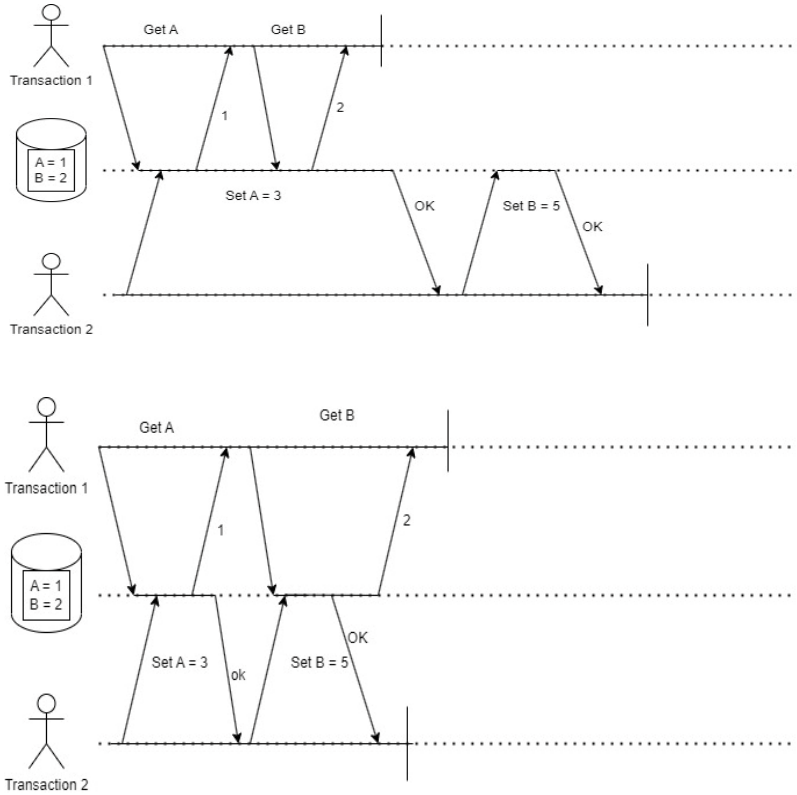
b) Explain when you should use local indexing, and when you should use global indexing?

Ans: For me, so far what I have understand is that, if you need a faster read then its better to use the global indexing and opposite case on should use local indexing. As, we know in local indexing every node just does their own stuff or do their own indexing, because of that for reading a request has to go to each and every node to look for a specific data until it found that and for writing one can just write to any nod and index that node, simple. And for global indexing, after performing the writing, you have to fix the indexing also, multiple writes happen hence the process could be slow. As people say, its better to use local indexing as its faster.

3. Kleppmann Chap 7

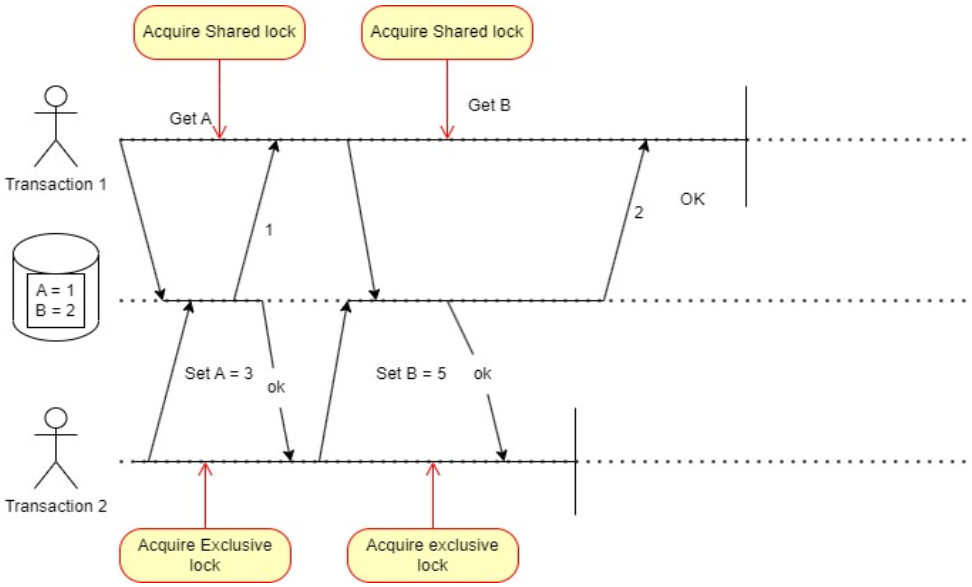
a) Read committed vs snapshot isolation. We want to compare read committed with snapshot isolation. We assume the traditional way of implementing read committed, where write locks are held to the end of the transaction, while read locks are set and released when doing the read itself. Show how the following schedule is executed using these two approaches: r1(A); w2(A); w2(B); r1(B); c1; c2.

Ans:



b) Also show how this is executed using serializable with 2PL (two-phase locking).

Ans:



4. Kleppmann Chap 8

a) If you send a message in a network and you do not get a reply, what could have happened? List some alternatives.

Ans: A lot of things could have happened. The request could be lost or waiting in the queue and will be delivered later or the remote node is stopped/lost, or the request is processed, and

response is lost, or request is processed but response is delayed. Or is could be network issue or network is very slow and it could be a server/node configuration error.

b) Explain why and how using clocks for last write wins could be dangerous.

Ans: If you consider two message and trying to figure out which one comes last based on the clock then its would be very tough decision and probably not accurate. It's obvious that, the receiving time will always be later than the sending time. But how much, its going to be based on the network. So, its very hard to determine this, as network is unreliable and not consistent.

5. Kleppmann Chap 9

a) Explain the connection between ordering, linearizability and consensus.

Ans: Ordering, linearizability and consensus are highly correlated. Ordering helps preserve causality. E.g.: A row must be created before its updated. If a system obeys the ordering imposes by causality, we say its causality consistent. And linearizability implies causality. If a system is linearizable then its ordered.

b) Are there any distributed data systems which are usable even if they are not linearizable? Explain your answer.

Ans: Yes, they are. Linearizability is a string consistency model in distributed system. Which comes with cost. Depending on the system it might require additional communication with nodes to preserve the property. Which in terms will produce late response. So, for the system like Facebook or Twitter where Linearizability is not mandatory then its better to leave it.

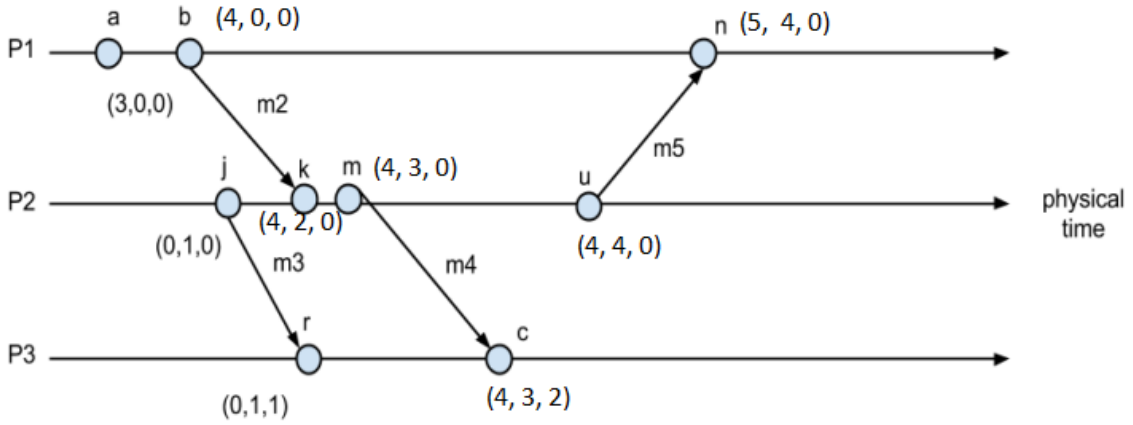
6. Coulouris Chap 14

a) Given two events e and f. Assume that the logical (Lamport) clock values L is such that L (e) < L (f). Can we then deduce that e "happened before" f? Why? What happens if one uses vector clocks instead? Explain.

Ans: No, we can't. As, Lamport clock property says that, if e-> f then L(e) < L(f). Not the other way around. So, its not possible to say e -> f given L(e) < L(f). Yes, if we use vector clock instead, then its possible to say that e->f given V(e) < V(f). As vector clock use local participants number in the vector along with index. So, if any of the vector value is less than one is less than another. You just do pair wise comparison to find out if one is smaller than others.

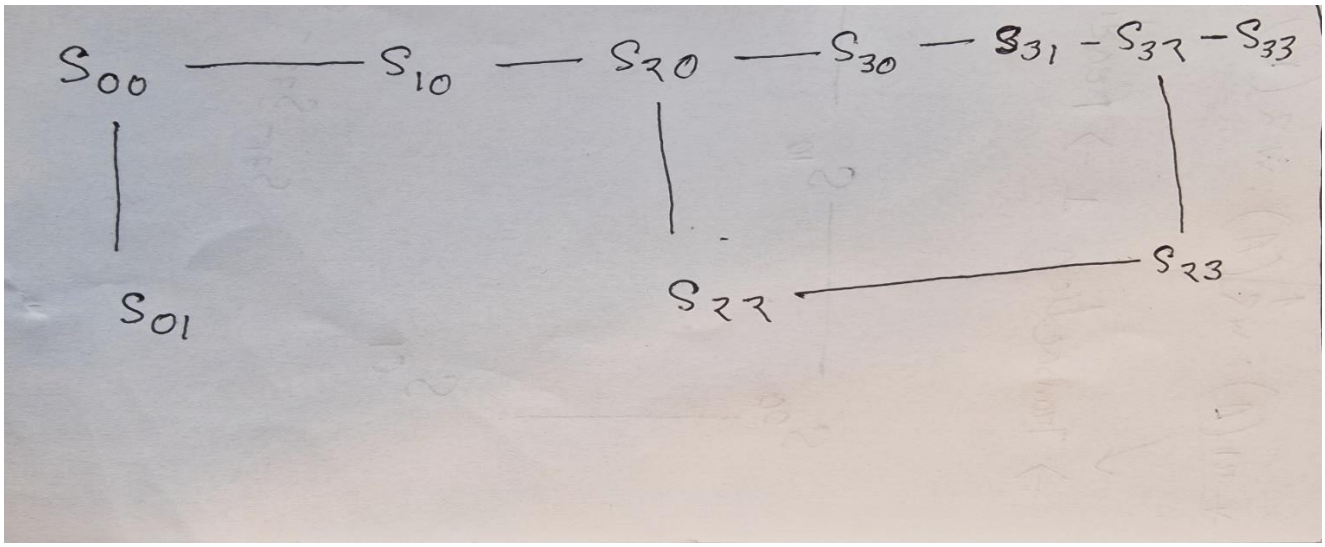
b) The figure below shows three processes and several events. Vector clock values are given for some of the events. Give the vector clock values for the remaining events.

Ans:



c) The figure below shows the events that occur at two processes P1 and P2. The arrows mean sending of messages. Show the alternative consistent states the system can have had. Start from state S00. (S_{xy} where x is p1's state and y is p2's state)

Ans:



7. RAFT

a) RAFT has a concept where the log is replicated to all participants. How does RAFT ensure that the log is equal on all nodes in case of a crash and a new leader?

Ans: In case of a crash, I new leader is selected. The process of new leader selection is also connected with log consistency. The new leader will have the most complete log, based on – term and index. Now the question about – how RUFT ensure – The leader will try to finish committing entries from its earlier term and make followers log consistent with its own. By delete extraneous entries and fill the missing entries. Leader sends a AppendEntries request to its followers initializing to (1+ leaders last index). When the checks fail, the leader decrement the index by 1 and try the process until all the logs get consistent with the leader.