3/11/2015

PAC 2

SISTEMES DISTRIBUITS AULA 2

MANUEL MARTÍNEZ DÍAZ

QUESTIONS

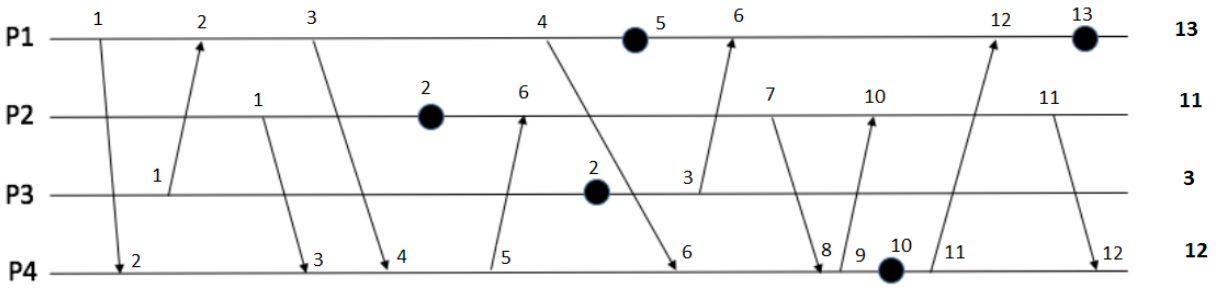
1. In the next scenarios, answer these questions:

a. With Lamport clocks. All clocks start at zero. Dots indicate progress of the Lamport clock in each process. Label the diagram with the clock values. Show the final state of the clocks.

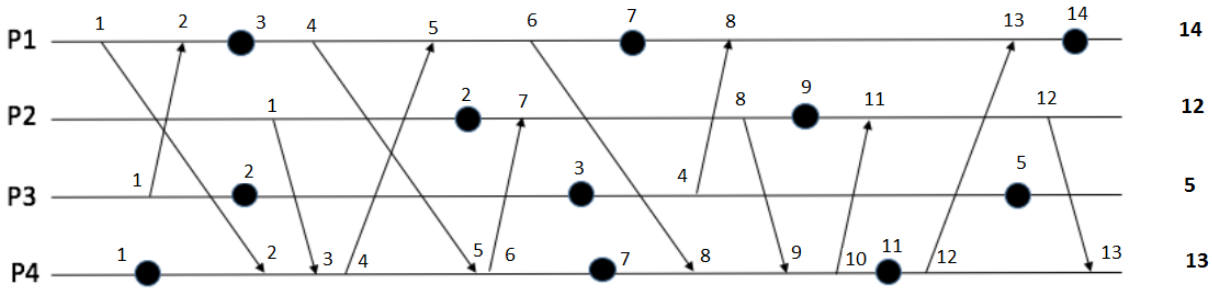
Each event sums one in each process. When one process send a message it send its value. The new value at the target process is the result from max(Li,t) plus one.

The final state of the clocks is show in the last column.

Case 1



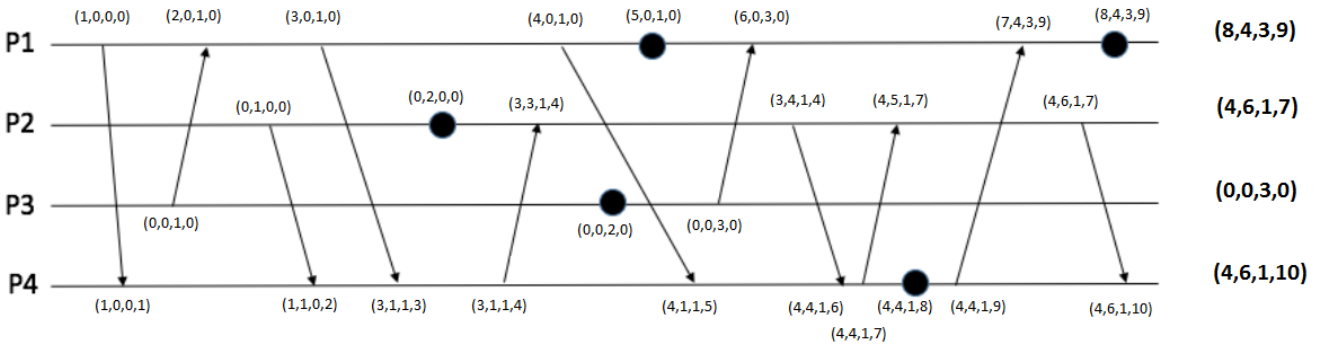
Case 2



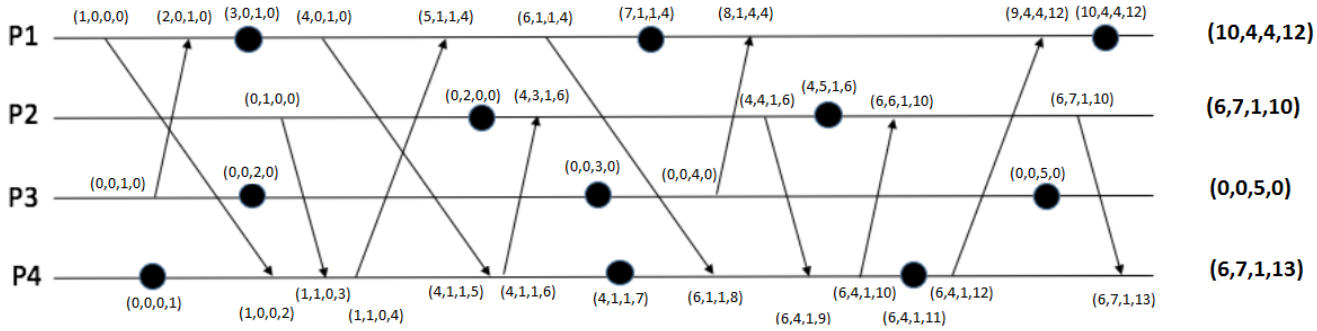
b. With vector clocks. All clocks start at (0,0,0,0). Label the diagram with the vector clock values. Show the final state of the clocks.

The final state of the clocks is show in the last column.

Case 1



Case 2



c. Using the happened before relationship, explain what can be achieved with vector clocks that is impossible with Lamport clocks.

Lamport clocks say, if events $a \rightarrow b$ it's true that event logical clock $C(a) < C(b)$, where $\rightarrow$ means happened-before, however we don`t know for sure if the event logical clock is $C(a) < C(b)$ the events are $a \rightarrow b$.

With vector clocks we have the same property that Lamport clocks, if events $a \rightarrow b$ it's true that event logical clock $C(a) < C(b)$, but int vector clocks is also true that if event logical clock $VC(a) < VC(b)$, then $a \rightarrow b$.

2. Compare the following multicast ordering: FIFO, causal and total. Present an example of each of them.

FIFO ordering assures that messages by the same sender are delivered in the order that they were sent.

An application for FIFO ordering is a video and software Distribution replicated in different servers. When the primary copy is updated, the owner sends a message to the other processes that have copies and that copies are updated in the same order.

Causal ordering, a process P0 send a message to process P1 and P2, P1 receive the message and to reply send a message to P0 and P2. The messages from P0 and P1 arrive at P2, but if a delay occurs on message from P0, P2 could receive first from P2. Both of the messages are related, and causal ordering ensures to ordered first message P0 and second the reply message P1.

One use of causal ordering is in a bulletin board, where we can also use total ordering. In a bulletin boards is critical that messages related that users post on it are displayed in the right ordered and assure that there is no answer displayed before the previous one to maintain a timeline.

Total ordering, if a process P0 sends messages x, y to processes P1, P2, then all the processes P1, P2 receive the messages in the same order that have been send. Ensures that all processes receive all messages from other processes in the same order.

An application of total ordering is applied on a faulty tolerant and distributed servers, were there are a group of identical servers that must be in the same state, receiving the same client messages in the same order. This is used in case of a server crashes there are other servers to manage the client messages.

3. Compare Two-Phase Commit (2PC) with Three-Phase Commit (3PC) taking into account functionality and performance of these protocols.

2PC vs 3PC

Comparing their functionality

2PC is a blocking Two Phase Commit protocol, offer validity of the data but there is a problem when the coordinator and participants crashes or fail. If the coordinator fails permanently, some participants will never resolve their transactions. After a participant has sent a doCommit message to the coordinator, it will block until a commit or rollback is received.

3PC is a non blocking Three Phase Commit protocol, offers validity of the data and more reliability when crashes or fails occurs, except with network partitions or asynchronous communication. There is a PreCommit message after the canCommit message and before the doCommit. This new message is used to wait until all participants sends a Yes and then start the doCommit and change the participant state to commit the transaction.

Comparing their performance

The 2PC costs 2 round trip times, one for each phase and in messages 4N where N is the number of participants.

The 3PC costs more messages than a Two Phase. The extra message adds more messages, a minimum of 3 round trip times. This is potentially a long latency to complete each transaction.

4. Compare BASE and ACID databases. Describe two examples of real applications that are based on BASE/ACID databases.

Where ACID is pessimistic and forces consistency at the end of every operation, BASE is optimistic and accepts that the database consistency will be in a state of flux.

When an ACID operation does not complete in all databases/server, the modifications will be undone in the rest of databases.

An ACID database is the perfect fit for use cases where data reliability and consistency are essential and BASE is more suitable when the priority is high availability and scalability.

BASE reduces consistency and isolation to gain availability and performance.

The availability on BASE is higher than ACID, for example in ACID the total availability is the sum of the availability of each server divided by the number of databases, for 2 with an availability of 99,9%, the total will be an 99,8% availability. In BASE will be 99,9% because support partial failures.

Banks use ACID databases because despite the scalability is needed to be sure when a transaction is to be done that there is enough money to do it. Another example is ticketmaster that in 2008 migrate to MySQL using replication servers.

Amazon use BASE on his web shopping service to improve the scalability, is named Dynamo.

5. Describe and compare passive replication with active replication. Explain how does the gossip architecture apply replication, how does it detect and solve conflicts, and what can be done to provide scalability?

The passive replication is composed by a single primary replica manager and one or more secondary replica managers, but in active replication all of servers have equivalent role.

In passive the front ends communicate only with the primary replica manager but in active replication, the front ends multicast their request to all servers.

The operations in passive are executed in the primary and sends copies of the updated data to the backups, but in active all servers execute and reply the operations.

The gossip architecture is composed by different replica managers that receive query (read) and update (write) messages from the front end. The front ends received the client messages and forward them to a replica manager.

All replicas managers will receive all updates with ordering with a weak consistency. The consistency is achieved using casual ordering to update the data, but it can also support total ordering and immediate ordering.

The failure detection is achieved by combining the reachability data from a lot of different nodes you can quickly determine when a node is down.

Conflict are detected the timestamp vector and the log provide sufficient information about the update history of each file replica to detect conflicts.

The scalability is a problem and one solution is to make most of the replicas read-only, which are updated by gossip messages but do not receive update from the front ends. This reduces the traffic but extend the delay to update de data in the replica managers. It is only applicable if the ratio update/query is small.