Project – 2
(Akanksha Bansal, Prateek Agarwal)
Slip days used: 4
Total used: 4

The read only command can be implemented in the following ways according to the Paper Section 4.4:
The client sends a read only command to just one of the replicas. This replica then sends over this read-only command to the Leader to ask when the given command can be executed because the replica might not be aware of all the decisions that have been made by the system. Thus the replica needs to know the latest state so that it doesn't return stale data.
The reply that the replica is looking for is ideally a slot number based on which it comes to know the number of decisions that have been made by Paxos. So that it can execute the read only command after it has been able to process the given number of decisions.
Now when the active leader receives this read-only command, it cannot just reply to the request of the replica with the maximum of his proposal set, because the acceptors might have adopted another leader. In order to come out of this situation, the leader sends a scout to the acceptors to make sure that he is the latest acceptor.
Once the leader gets adopted by majority of acceptors the leader then sends a message to the replicas informing them that they can execute the read only command after the slot number mentioned in the message from the leader.

Flaw in this approach:
Because the messages are being sent asynchronously there is always the possibility that the message being sent by the leader to the replica, informing about the slot no. at which the read-only command needs to be executed, gets delayed by an amount which results in stale data being returned by the replica. The staleness can arise if Paxos was able to make a decision on the slot number mentioned by the leader to the replica before the replica could send out the reply to the client's read-only request.
In other words, the leader cannot be sure that even after it has been able to get itself adopted, its decision might not be the latest decision by the time the message reaches the replica.

An example of this implementation:
Environment consists of: 2 Replicas, 2 Leaders and 3 Acceptors.
Replica 1: Sends a request to all the leaders with the read only request while it waiting for a decision on slot_no =5.
Leader 1(Active): Gets the read only message from Replica 1. Gets a scout adopted by majority of acceptors. Sends a message to Replica 1 informing it to run the read only command before executing slot no: 6 (because that is the number of decisions that have been made uptill now).
Leader 2: The failure detector of Leader 2 informs the current leader has died. Gets adopted by majority of acceptors.
Replica 2: Sends a proposal message for slot_no = 7.
Leader 2: Gets the slot_no 7 decided by majority of acceptors.
Replica 2: Executes the command at slot_no=5.
Replica 1: Gets the read only decision from the leader 1. But now the data that is held by the Replica 1 is stale.

The paper mentions lease as a solution to this problem. Where the active leader is ensured that for a given time frame the acceptors will not adopt any new leader. This assumes that the messages being sent by the leader to replicas will get delivered by the time the lease of the leader expires. But in an asynchronous system we can never have this guarantee.
The paper mentions another approach where the leader waits for all the pending operations to get completed, but again the leader can never guarantee if no new updates to the system will be made by the time replica executes the given read only command.

By increasing the delay time between the read-only decision message between leader and replica we can always find an example where the above two solutions will fail.

Our Approach:

If we are able to ensure that a leader before informing the replica about the decisions (lets say at slot number *S* to be the point at which the read only command can be executed), it is also able to inform the rest of the leaders that a read only command needs to be executed before slot number *S+1*, then we can side step the problem at hand.

Acceptors are the shared memory of the leaders. Also every leader before becoming active will be communicating to the acceptors so as to be aware of all the decisions that have been made by any of the leader.

Now as per the paper even an active leader when gets a read only message needs to send a scout to the acceptors to check is they have not adopted some other leader in the meantime. Now if this message in enhanced by leader to inform the acceptors about the read only commands that need to be executed after a slot number *S*, then we can be sure that the next leader will also be able to come to know of this decision. (Because the majority of acceptors will have added in their PValues this new command which needs to be executed after the slot number *S,* and the next leader overrides its proposal with Pvalues).

If the scout gets pre-empted by some acceptor, in that case the leader will recognize that it is no longer the active leader and sets his active flag as false.

In case the leader is not pre-empted, it can send the decision message to all the replicas by a new command informing them about the slot number *S+1* at which they can execute the read-only command. Any new proposal that comes for *S+1* will get added into this read-only command. So that the decision for *S+1* tells the replicas about the read-only commands that need to be executed by them before executing the update command.

Implementation details

The decision message is enhanced to inform the replicas about the read-only operations to be addressed before execution of the said update operation. Thus command will also contain a *Set of read only commands.* Command now tells the system that this set of read-only operations need to be performed before the execution of the update operation.

Replica:

- Creates a new type of Paxos message for Read only requests.
- Executes the read-only commands before the update command whenever a decision message comes form leader. Slot number is not increased in case the message doesn't have any update command.

Leader:

- When it sees a read only request, it creates a scout and tells the scout to also inform the acceptors about the slot no *S+1* at which the leader plans to ask the replicas to execute the read-only command. The scout in this case doesn't increase its ballot number.
- The slot number *S+1* at which the read only command needs to be executed is decided by the leader's proposal set. The first non-null command in the proposal set is chosen to be location at which the leader plans to execute the read-only message.
- Leader adds an empty update command into its proposal set that whenever a proposal is received by the leader for this slot number *S+1*, the leader's commander will be able to get complete proposal accepted by the leaders. (Helpful in the scenario where the read-only scout gets delayed in getting acceptance by the acceptors but the commander was quick in getting the proposal accepted).
- The leader sends out the message to all the replicas with a decision message without any update command as well as soon as the read-only scout informs the leader that the leader is still an active leader.

Acceptors:

- Accept read-only commands that are being provided by the scout.
- Merge the requests that have been received from the same ballot no. and for same slot. That is combine the accepted value of the commands where one is only a read only message set and another is an update command for the same slot for same ballot number.