Name: Raman Kumar Sonkhla
Person Name: RSONKHLA
UB #: 50026724
Group Name: Buffalo wings

Simple Dynamo

1. Components

a. Front End

As the name suggests, this component is used by the front end user to interact with simple dynamo. It takes the command from the user and acts as a bridge between user and simple dynamo. In our case, front end acts as the interface having 3 put buttons, 1 get button and 1 dump button. When any of the put buttons is pressed a thread is spawned which produces a key-value every 3 sec and gives to the content provider. The content provider then decides whom it should forward the key value pair to store in simple dynamo. The emulator to which it forwards is called coordinator. If the coordinator does not respond within timeout time then it sends to its successor.

b. Coordinator

The role of this component is to receive key-value pair from the content provider insert in its own database and send to two of its successor for replica. After sending it waits for reply from its successor and in total if it is unable to get 3 replicas including itself then it sends an error message to the content provider.

Also when a query button is pressed from the front end the front end decides who will be the coordinator. The coordinator then retrieves the value from its own database and waits for its successor to send him the value. Once it reaches a quorum of 2 or 3 it replies back to the Front End.

c. Replicas

The role of these replicas is to simply receive the message from its coordinator, put it in its own database. After inserting to its own database it sends a reply back to its coordinator intimating successful insertion.

When it receives a query message it retrieves the value corresponding to the given key and sends the reply to its coordinator.

2. Consistency model

The consistency model achieved by our application depends on following two factors:-

1) Method of replicating data

2) Method of recovering data for failed-recovered node.

Since data is being replicated in two replicas in same order as in the data arriving at coordinator, under no failure condition the application will achieve sequential consistency. But since replication of data is more or less same as mentioned in specification, it's our method of recovery for failed node that governs the consistency achieved.

Failed handling of nodes:-

1) Power up the node.
2) Ask first successor node to provide relevant data from its database. This node will provide failed node's data as well as the node previous (i.e. keys for which failed node was first replica).
3) If first successor is not alive, ask next successor to provide relevant data from its database. This node will provide failed node's data as well as the node previous.
4) Ask predecessor node to provide data relevant data from its database. This node will provide both its data and the node previous, i.e. keys for which the failed node was a first or second replica.
5) If predecessor is not alive, ask next predecessor. This will only provide data for which it is coordinate i.e. for which it was second replica.
6) After failed node has recovered completely, start server thread so that this node can now respond to incoming insert or query request.

Based on these steps, we can establish that any node that fails first makes itself consistent with existing data and then only responds to any more requests. However following cases must be considered:-

1) Node A, B, C and D are part of dynamo in order.

   B dies. Insert request comes to A for which coordinator was B.

   Case 1: Request comes before A recovers: A directs request to C, which replicates at D and completes successfully. A then recovers and gets latest values from C. Hence overall consistency is maintained.

   Case 2: Request comes after A recovers: In this case request will go to A and A will properly replicate, maintaining overall consistency.

2) Node A, B, C and D are part of dynamo in order.
   Node B dies, meanwhile insert request comes for which coordinator was A, then A will skip B and replicate in C. Also since A's server thread is busy doing this, it won't respond to B's update request message until current operation is done. Hence B will get only updated values, thus maintaining overall consistency.

Based on above we can infer that our application maintains sequential consistency.