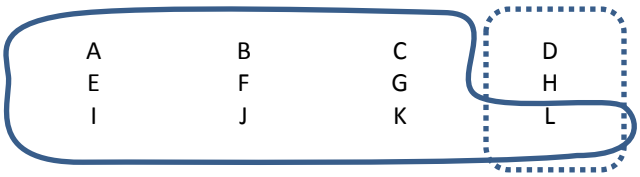
Assignment 3 CS454/654
Due: 2:30 PM March 29, 2012

1. Recall the Quorum-based replication example we discussed in class where the write quorum size N_w is 10, read quorum size N_r is 3, and the number of nodes N is 12:



A pragmatic problem with Quorum-based replication is that the data must be replicated to N_w number of nodes, where N_w must be bigger than N / 2. For our example, we would need to create replicas at 10 nodes, which is generally more than one would normally replicate for fault tolerance.

   a. Assume that you only want 2 replicas for each data item. Describe how you can partition the data-space to create a collection of quorums with each having a N_w of 2 if the total number of nodes is 12.

   b. What problem would a partitioning approach have if the data popularity is highly skewed? Consider the case where the most popular data item is an order of magnitude more popular than the second most popular item, and the second most popular item is an order of magnitude more popular than the third most popular item, etc. You can assume that this is a read mostly workload (99.9% reads, 0.1% writes).

2. An alternative approach to reducing replication overhead in quorum systems is to use erasure coding. Erasure coding is a family of techniques that divides each data item into *m* fragments, generates a recoded representation that consists of *n* fragments where $n > m$, and enables the reconstruction of the original data item by combining any *m* out of *n* fragments from the recoded data representation. For example, a 1000 byte data item can be divided into 10 fragments of 100 bytes each and recoded into a new representation with 20 fragments of 100 bytes each. In this example, any 10 fragments from the recoded representation can be used to recreate the original data item.

   a. Using erasure coding, propose a quorum replication scheme with a storage requirement that is equivalent to 2 replicas (i.e. if a data item is 1000 bytes, your quorum replication scheme should only store 2000 bytes of data for this data item) for N = 12 and $N_w = 10$.

   b. How many nodes can fail before your replication scheme loses data? Explain why.

   c. Does an erasure coding approach perform better than a data partitioning approach when the data popularity is highly skewed? Assume the distribution is the same as the distribution in question 1b. You can again assume that this is a read mostly workload (99.9% reads, 0.1% writes). Explain your reasoning.

3. Sketch out an efficient replication approach that provides:
   a. Monotonic reads
   b. Monotonic writes
   c. Read your writes
   d. Write follows read

   Explain why your approach provides the desired client-centric consistency. Try to avoid providing more consistency than necessary if it reduces performance. For example, primary-backup or chain replication would provide all of these properties, but they also restrict which nodes can handle reads and writes.

4. Given the following real-time execution ordering:

   $$W_1(x, 3), R_2(x), R_3(x), W_2(x, 2), R_1(y), W_2(y, 1), R_3(y), R_2(z), R_3(z)$$

   where $W_2(x, 3)$ means that transaction 2 sets the value of x to 3. The original values of x, y, z are 4, 5, 6 respectively and the transaction number serves as the transaction's timestamp. Show how the following concurrency control schemes affect the ordering of the reads and writes in the execution history, and the value returned from each read operation.

   a. Strict 2PL locking: Assume a transaction commits immediately after its last operation.
   b. Timestamp ordering
   c. Multiversion timestamp ordering

   Assume that if a transaction aborts, it will be restarted as a new transaction with its reads and writes added to the end of the execution history. The new transaction will be assigned the next available transaction ID.

5. The main capability missing in your DHT from assignment 2 is replication. Describe what modifications you would need to make to your DHT to incorporate the following replication schemes:

   a. Primary Copy Remote-Write: Assume that you want 5 replicas. The replicas should be stored at the closest counter clockwise node (CCCN) to the key, and at the nodes in the CCCN's neighbor set.
   b. Quorum-based replication: Assume that $N_w = 3$, $N_r = 3$, and only the CCCN to the key and the CCCN's neighbor set can be part of the Quorum.

   Consider how your design would handle nodes joining and leaving the DHT. How does it handle a failure of the primary replica? Also consider how and when it would delete a replica when a node is no longer part of the CCCN's neighbor set.