Sam Pastoriza

Chris Lambert

Joel Shapiro

Scalability using Replication

**Question:**

**For instance, your database could bring about improved scalability by replication and or Sharding. Which option are you using? Why are you choosing this option?  How does the option affect consistency? Why is this consistency the correct option for you?**

To account for scaling, Riak uses both replication and partitions. Replication is the act of duplicating data across multiple servers, which Riak does by default. Since Riak automatically distributes data across multiple servers, it achieves a near-linear performance increase as capacity rises. Riak will automatically redistribute data as nodes are added or removed to balance data across nodes. Riak eliminates manual sharding and eliminates hot spots through replication of data. Partitioning is how we decide to divide a set of keys among different servers. We can choose a specific server to have a set range of keys, and then have the rest of the remaining servers host the rest of the different non-overlapping keys. The total capacity can increase as the number of partitions increase since there are more nodes on more servers, but each server has to do less work. However, if a node goes down, the partition goes down with it. This is why Riak uses the combination of replication and partitioning for scaling and availability.

We currently only storing usernames and hashed passwords in Riak. We will divide the usernames, since they are unique, into two partitions on two different nodes, and then replicated those two nodes, so there will be a total of 4 nodes. This means there are two pairs of nodes that have two sets of the same data. Riak uses a variable called n\_val to describe the number of duplicated nodes to create. The default is three, but we will use n\_val = 2, since this will divide the number of usernames equally.

From version 2.0, Riak supports strong consistency and high availability. This should break the CAP theorem, but Riak only sets a bucket type property as strongly consistent. This means that either a request is successfully replicated to a majority of partitions or it will fail. However, if too many nodes go down, the write will fail and you will need to try again after repairing the node. This means you have lost high availability. Riak defaults to high availability, but since we are dealing with usernames and passwords, it is critical that we have strong consistency.