Comp 480/580: Assignment #2

Rice University — Due Date: Tuesday, 2/19/2019

1 Consistent Hashing with Variable Server Capacities: A comparison of Three Algorithms.

Task: Assign load to servers, using a hashing scheme, such that the load is maximally balanced and the assignment latency (or cost of the evaluation is small). Servers can get added or deleted (due to crash), but the hash function should be able to handle the load balancing gracefully. **Initial Server:** Start with $n = 5$ servers with each having integer capacity between $1 - 3$. For every server randomly generate a number between 1 and 3 and the generated number will be its assigned capacity. We will keep track of the list of jobs assigned to each server in a simple dictionary list. If a server is down, we will also have another bit array A, with A[server id] showing its status, whether it is down or up. Note, this list will grow as the number of servers are added.

Iteration: Let P_f be the probability of a server failure and P_A be the probability of server addition. Now go in a for loop over variable t (time) from 0 to 1000. In each, iteration, a job with id t is created and assigned based on one of the following algorithms. Besides, with probability, P_f a randomly chosen server fails. Also, with probability P_A a new server is added. The id of the new server is equal to the total number of servers created so far + 1. For simplicity and to avoid dealing with old servers coming back, lets follow a simple rule: if a server fails ever we ignore it, or assume is added later as a new server. This way, we don't have to keep track of which server came back.

Deletion and Addition: Whenever a server is deleted and added, there is additional handling to transfer data from servers which we will ignore for this assignment.
Implement the following three Algorithms for assigning load to server:

1. **SPOCA Algorithm taught in class:**
    - Calculate the expected total capacity of servers you will create in terms of $n,\ P_f,\ P_A$ and total iteration t, given the process described above. Reserve space 10x more than the expected load.
2. **Consistent Hashing with copies of server:**
    - Note that the number of copies should be equal to the capacity of the server
    - Also, reserve the hash space to be around 10x more than the expected load.
3. **Consistent Hashing with Bounded Loads:**
    - Use the usual consistent hashing, (just one copy of server), but if the load of the server is exhausted then go over to the next one in the same direction.

In all the above algorithm, use any reasonable way to resolve low probability collisions of servers to the same address, etc. The main function will have a signature int assign(int jobid), which will take an input as jobid and outputs the assigned server. Check that for the same job id and no change in the status of servers, the assignment is same for different function calls (this is called consistency)
Things to keep track Of: Start with $n = 2$ severs. Plot the following with t:

1. The mean number of (only active) servers running at f times their capacity over time where the ranges are $f = \{0 - 0.3, 0.3 - 0.7, 0.7 - 1, 1 - 1.3, 1.3+\}$. One plot per range. Each plot will have all three methods (three lines).

2. The total overhead of addition. This is the total time spend in the assignment function call.
3. Do the above four times with (All four combinations below.) $P_f = 0.1, 0.05$ $P_A = 0.01, 0.005$.
4. Compare contrast different methods and summarize your findings in half a page.