Assignment 3
Distributed Data Management

Team NoCommentCode

1 Basic Architecture

We use a pull-based propagation to distribute work. When a worker is idle, it sends a `WorkRequest` to the master. The master will then send a work item back. If the master has no work to be done (i. e., there are no more persons left from a batch with unknown password), it puts the worker into a queue. Once a new batch message arrives, every worker in that queue will get some work again.

2 Solving the Hints

We generate all subsets of the *password characters* with one char missing. One after another, these subsets are passed to the workers. We try to avoid iterating over all possible permutations multiple times. Therefore, we also send all the hashes to a worker together with the subset of characters. While producing the permutations, the worker hashes the current permutation and compares it with the hashes. If it found a match, a message to the master is sent and the hash is removed from further comparison both for the worker and the master.

When the worker generated all the permutations of one character subset and did not find any match for one specific person, the worker informs the master that the char that was not part of this subset is definitely part of the person's password.

3 Password Cracking

Once we know either all the characters that are definitely part of a person's password or have excluded all characters except of four, we mark that password as ready for cracking.

We ensure that the password cracking has a higher priority than the hint solving. Therefore we return a `CrackingRequest` when asked and any password is ready for cracking. Once a password is cracked, we remove all of the person's information.

4 Worker Scaling

When our system retrieves a `WorkRequest` and no password is able to be cracked, we send the next subset of characters to solve the hints. If we sent all of those subsets, we go back in the sending order and resend a subset. Since every subset is shuffled at the beginning of the permutation generation, we hope that this leads to finding all results faster. If we got as many different hash matches for a subset as we have persons or we once ran through all of its permutations, we will skip this subset for further permutation generation.