Assignment 3 – password cracking

Getting Started

The corresponding .jar file for this project is located in the assignment3 folder. A master system can be created from there using the following command:

```
java -jar ddm-pc-1.0.jar master -dp ../ddm-pc/data/
```

And additional slave systems on the same machine using this expression:

```
java -jar ddm-pc-1.0.jar slave -mh 127.0.0.1
```

Program Flow

1. The Input file is read by the Reader and transferred to the Master in chunks.
2. The Master waits for all the chunks and combines them in *handle(BatchMessage)*.
3. When all data is received, the master creates a dictionary, mapping from hint hashes to all the indices of inputs which contain this hint. This prevents duplicate calculation of duplicate hint hashes.
4. The master also analyses the hint character universe (= pw char universe) and calculates all possible prefixes (draw x without replacement) of hints of length x. (in our case x=2). These prefixes are later used to assign specific ranges of possible hints to the workers.
5. The master proceeds by informing all available slaves about all hashes that need to be found using a broadcasted *hashesOfInterestMessage*. (from here on, he will also send this message to new slaves)
6. After receiving this message, the slaves know that it is time to work and they will send an *idleMessage* to the master, to indicate that they can be assigned to new work.
7. While not all hints are solved (= not all prefixes distributed), the master responds with a *hashRangeMessage* and keeps track of which range each slave is currently working on, in case any of those terminate/loose connection. (Note: At this point the last character of each originally calculated prefix is interpreted as the character to ignore in this cracking task).
8. The worker brute forces all combinations using the provided prefix and excluded character and compares his results to the *hashesOfInterest*. If he finds a hash, he sends the master a *foundHashMessage*. And once he is finished, he concludes his current task with another *idleMessage*.
9. Upon receiving a *foundHashMessage*, the master updates his lists of solved hints for all input lines that contained the corresponding hint hash.
10. When he finds out that all hints for one input have been cracked, he adds this input to a queue of elements that are ready for password cracking.
11. The above continues until all prefixes are distributed.
12. When this is the case, idle workers will be collected in a waiting queue and will be sent *crackPasswordMessages* immediately when new passwords are ready to be tackled (all hints solved).
13. Those messages include all possible characters that each password can still contain, the hash and the length of the password.
14. After the correct password is found by a slave it informs the master using a *foundPasswordMessage*.
15. The master happily receives all the final messages and forwards them to the collector, who then outputs the results.