Assignment 3 – password cracking in akka

Salih Mezraoui and Tobias Niedling
Group Herbstlaub

Solution

1) For each char in the charset generate a hint cracking task (one character is missing in the hin)

   Example: Charset: [‘A’,’B’,’C’,’D’]
   Tasks: [‘A’,’B’,’C’], [‘B’,’C’,‘D’], [‘A’,’C’,’D’], [‘A’,’B’,’D’]

2) Devide the task for hint cracking in smaller subtasks (each worker gets a fixed start to reduce the number of permutations)

   Example: Task 1: [‘A’,‘B’,’C’]
   Tasks: {[‘A’,’B’], start: ‘C’}, {[‘A’,’C’], start: ‘B’}, {[‘B’,’C’], start: ‘A’}

   The subdivision is repeated until a threshold, which is either a too small remaining charset or maximum length of the fixed start

Solution

3) Put all tasks in a task queue and shuffle it. Shuffling leads to a higher performance, because similar tasks are not executed in parallel

   Example: From 2) → only one task will crack a hint, if the first one succeeds, the other two don’t need to be executed. If the queue is sequentially those tasks will usually be assigned in short intervals one after another. If the queue is shuffled the unnecessary tasks have a higher chance of not already being assigned to a worker and they can be removed form the queue

4) Schedule tasks to the workers. To react on later joining workers and because of point 3), a maximum of 3 tasks is assigned to a worker.

   All scheduled tasks are again appended to the back of the queue, if a worker fails (silently).

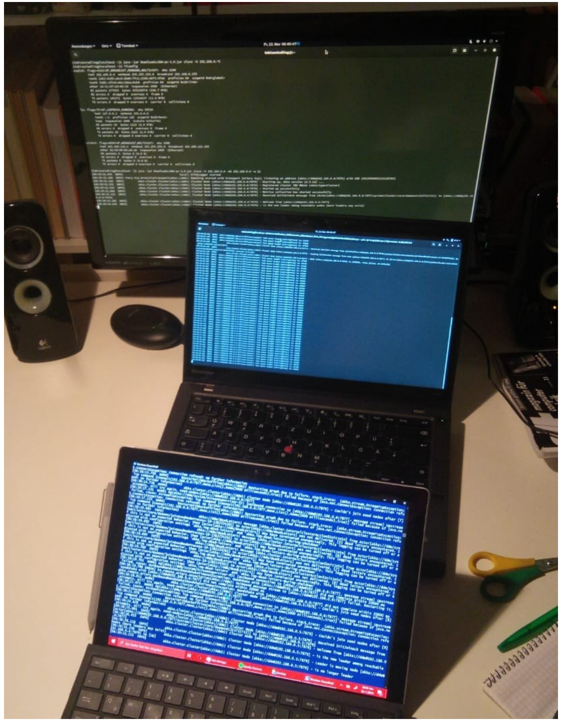
Solution

5) If all hints are cracked (queue length 0), start cracking passwords

   Therefore, again, a list of tasks is generated (one for each password). This list is scheduled as mentioned before

6) If the queue is empty again, the next batch is pulled or the program finishes

Testing



- 3 PC’s in local WiFi
- Intel Core m3-6Y30 (2*0,9GHz)
- Intel Core i7-5600U (4*2,6GHz)
- AMD Ryzen 5 2600 (6*3,4GHz)
- Runtime ~16 min

Possible weaknesses

- The subdivision of tasks happens in the master, for big character sets or longer passwords the master might be stuck for several seconds, so outsourcing this task to another worker could have been useful

- The task queue needs to be in main memory for fast processing, so for larger batches, longer passwords, bigger charsets, … performance issues might occur (or even memory errors). The solution here is to reduce the batch size via the command line.

- The scheduling is a compromise of keeping the workers busy and not assigning to many tasks to them at once in case more workers join. This is implemented by counting the send tasks and received answers, so that their difference gets no higher than 3. If a message is lost, the effective limit is reduced by one. So if this happens 3 times, no more tasks will be assigned to the worker, although it might still be available. (We didn’t saw any message loss in our testing scenario, so we thought that this risk is really low and loosing one worker should not be that serious in a (big) cluster of at least 20 workers for this task)