UNIVERSITY OF GRONINGEN

PARALLEL COMPUTING

Homework 6: Parallelization of brute forcing of passwords.

Author:
Konstantinos MAZGALTZIDIS
(s5100453)

June 17, 2022

1 Introduction to Open MPI

Open MPI works a little different compared to, POSIX Threads. In POSIX Threads one must spawn, manage, terminate each thread.

In the other hand Open MPI does something different, in essence all processing nodes execute the same program, it up to the programmer to place checks to do different tasks depending on the process ID.

It is like running the same program but with different command line arguments. In Open MPI there is no shared memory and every process has it own private memory space, communication is done thought the exchange of messages.

2 Introduction to the problem

Let's define the problem in the input/output model for better understanding.

- **Input :**
  1. A list of usernames and hashed passwords, also the hashing algorithm is specified.
  2. A dictionary containing commonly used passwords (*stored in plaintext*).
- **Output :** The passwords that were matched/-found in the list of hashed password, along with the username.

3 Preprocessing of dictionaries

As we have freedom to do any prepossessing we like, I chose to do the following:

3.1 E-books

The provided book was written in a format that the program couldn't read easily, so I did some modifications to the text files. I wrote a small Python script to get all the words of the e-books, remove duplicate words, lowercase them and put them into a file called `combided.txt`.

3.2 Plain passwords

In the provided files, I noticed some passwords stored in plain text, it will be a shame to not add those passwords in dictionary list.

With some `awk` magic `awk -F : 'print $2' sample1-plain.txt` I got the second collumn of the plain text files and I added them to the dictionary `combided.txt`.

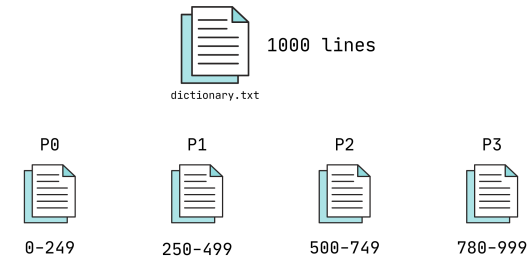
4 Algorithm parallelization

Inspecting the provided code, we observe the following pipeline:

- First, the files that have the usernames and hashed passwords are being read and loaded to the programs memory.
- Secondly, `tryPasswords` function takes three arguments the first one being the dictionary, the second being the passwords of the users and lastly the `user.hashSetting` that contains information about the users.
- Free memory.

Because the dictionaries contain a lot of lines of text, a simple idea to try to parallelize the work is to split the hashing of the dictionaries into multiple process and give at each process a different section of the dictionary to work on.

For example, if we have a 1000 lines dictionary and 4 processes, each process will take a 250 lines segment to process:



With this implementation, each process doesn't need to communicate with any other process, remember message passing is expensive!

One big disadvantage, this splitting method, has is in permutations, there will not be as good compared doing with the whole file because every process will work with only a part of the array.

5 Permutetunions

The brute force algorithm, it first tries all the combinations from the dictionary after that, it tries to uppercase everything, after that it tries all combinations between the lowercase and the uppercase words.