Password Cracker

Jiahua Zhang

GitHub link: https://github.com/jnzbfgjd/zjhpassword
Name of GENI slice: PwcrackZjh
Public link: http://204.102.244.52:3890/

1.Problem Statement

1.1 Definition

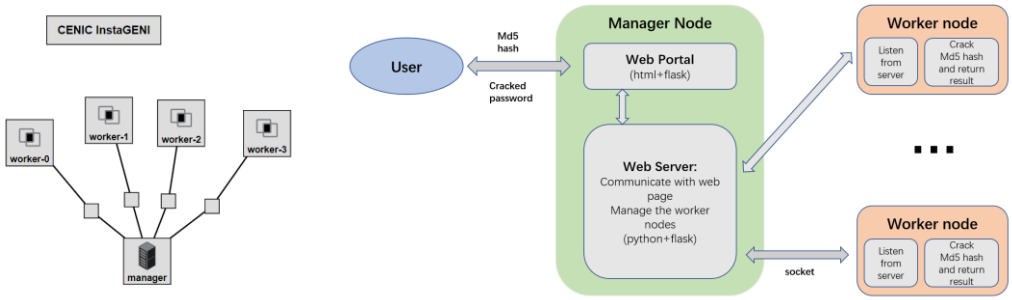
In this project our aim is to design a distributed system where a user submits the md5 hash of a 5-character password (either lowercase or uppercase) to the system using a web interface. The web interface, with the help of worker nodes cracks the password by a brute force approach. Which means that the worker nodes will check through every md5 hash of all possibilities and find the one that match the input hash.

1.2 Learning Outcomes

I.How to implement simple html webpage and how to realize the data interaction between the front end and the back end.
II.How to design a distributed system and how to manage the asynchronous communication between each node via sockets programming and multi-threads programming.
III.How to use brute force method to crack a md5 hash.

2.Experimental Methodology

The following figure shows the architecture of the distributed system, which is consist of 4 worker nodes and a manager node.



2.1 Manager node

There are two main components in manager node.
The first is the web portal which is responsible for displaying a web page for user to input md5 hash and the number of worker nodes to use for the cracking, and after the hash has been cracked display the result and the time used in total. I use flask to realize the data interaction between the front end and the back end.
The second is the web server which is responsible for receiving the hash as while as communicate with worker nodes. After getting the number of workers and md5 hash, the server will create corresponding number of processes using multiprocessing to

communicate with and monitor the status of the worker node. For each process, it will connect to the worker node with socket and send the number of workers, the id of the corresponding worker, the hash to crack to the worker node. When the worker node returns the result, the process will return the status to the main program and terminate all the other processes that are still running, then the main program will send a terminate message to all the worker node to stop the process on the worker node that is still running. After that the result will be pass to the web portal and then display to user.

2.2 Worker node

The worker node will listen and receive message from server.

I use an integer-to-str function that converts any integer from range 0 to 380204031 (52^5-1) into a 5-letter string, then after the number of workers and id is passed, I iterate from the umber of id to 380204031 with steps of number of workers to make sure all possibilities have been covered when all worker nodes working together. After the result has been found, the worker node will send a message back to the server. And if the worker node receives a terminate message from the server when the matching process is still working, it will terminate the process anyway.

2.3 Assumptions

A. The input md5 hash should come from a 5-letter string only using capital and lowercase letters. The number of the worker node should be no less than 1 and no more than 4 and must be an integer.

B. The connection between the manager node and the worker nodes should be flawless, user cannot resend or remove requests after the data is submitted.

C. Only one user at one time when using the interface.

3. Results

3.1 Usage instructions

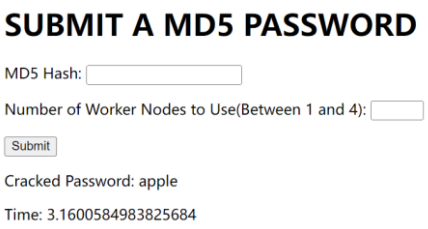
I.Setup the GENI resources with PwcrackZjh_request_rspec.xml.

II. SSH into all of the nodes and clone the repository via the command git clone https://github.com/jnzbfgjd/zjhpassword.git.

III. For the manager node, enter the directory by cd zjhpassword/server, modify the managerhost.txt and workerhost.txt with your own IP address, then run the server setup script via sh serversetup.sh.

IV. For the worker node, enter the directory by cd zjhpassword/client, run the worker setup script via sh workersetup.sh ip-address, while the ip-address is the address of the corresponding worker node.

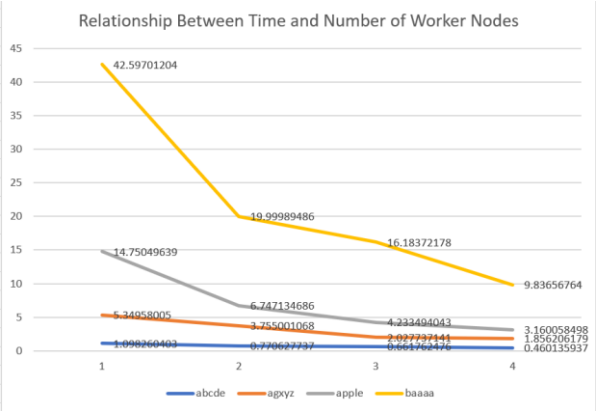
V. Enter the web address of http://manager-ip-address:3890/ ,the manager-ip-address is the address of the manager node, then input an md5 hash and the number of worker nodes before click submit.



VI. Wait until the web page shows the cracked password. The web page should be look like the above figure.

3.2 Analysis

After the distributed system being implemented, I tried to study the relationship between the time to crack the md5 hash and the number of worker nodes. My hypothesis was that time will decrease with the increase of the number of worker nodes, and the product of these two values should be very close. For here I choose four md5 hash and run multi times to get the result and time on different conditions, then take the average time as the final value. The final result is shown in the following figure.



From the result we can know that with the increase of the serial number of the password, the time also increase, and it's almost proportional to the serial number. And when the number of worker nodes increases, the efficiency in cracking the md5 hash also increase. And when the serial number is large, the relationship between them is more likely to be linear. When the serial number is quite small, the time to create process and communicate between the manager and worker nodes takes a major proportion, so the relationship is not that close to linear.

4.Conclusion

4.1 Summary

In this project I implemented a simply distributed system to crack md5 hash. I learned how to realize the data interaction between the front end and the back end. I also learned how to use multi-process and socket to communicate between server and client. Meanwhile I learned how to use brute force method to crack a md5 hash. Finally, I performed an experiment to find the relationship between time and the number of worker nodes.

4.2 Extensions

There are few possible extensions for the project. The first is to add more worker nodes, which could be realize by add the address to the txt file and then download the code from github and run the shell script on the corresponding worker node. The second is to support several users to use the interface at the same time, these can be realized by using multi-process pool. The third is to improve the algorithm on the worker node, which could support more types of hash and increase the efficiency.

5.Divsion of Labor

The project is finished on my own.