Distributed System To crack 5-char Password

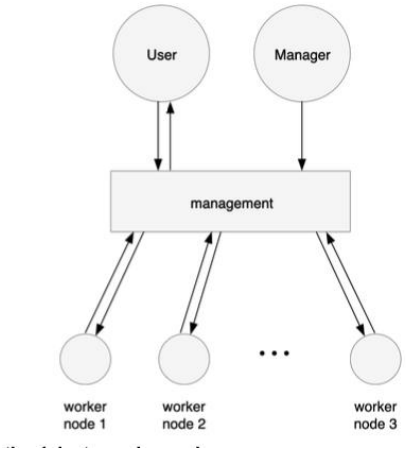
Minghui Yang, Yinan An, Jingyi Huang, Yuhe Peng
Github (management node): https://github.com/minghuiyang1998/PAFinal_MNode
Github (worker node): https://github.com/JessiePen/PAFinal_WNode

1. Problem Statement

Definition
This project is to design a Hadoop like distributed system to crack a 5- character (a-z, A-Z) password hashed by md5. The crack process is done through a brute force approach with the help of multiple worker nodes. The whole system is parallel and scalable. The manager node dedicates an equal amount of work units to free workers. A worker works on the units assigned and will request more from the manager if all units have been calculated. During the process, a user should be able to add/remove workers on the fly.
The difficulty of the project is to divide calculations to equal partition and assign them to every worker. Furthermore, what actions should be taken if one of the workers timeouts.
Learning Outcomes:
Through the project, we could know more about socket mechanisms, front-backend programming, and distributed systems.

2. Design

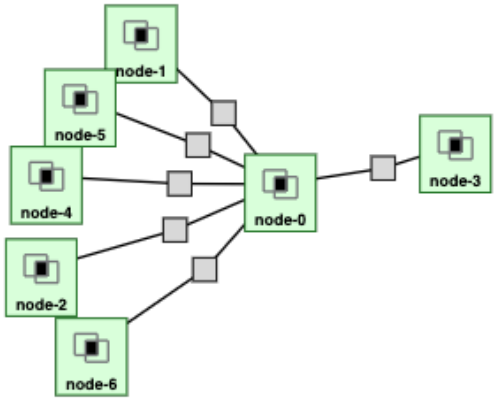
1. Setup diagram



2. Environment/resources
Deploy:
URL:
User: http://pcvm5-1.instageni.idre.ucla.edu:8000/
Admin: http://pcvm5-1.instageni.idre.ucla.edu:8000/admin

Available Worker Node:

Node-1: 10.10.1.2 port: 58001
Node-2: 10.10.2.2 port: 58001
Node-3: 10.10.3.2 port: 58001
Node-4: 10.10.4.2 port: 58001
Node-5: 10.10.5.2 port: 58001
Node-6: 10.10.6.2 port: 58001



3. Execution/results

1. Configuration and usage

Manager Node:

wget https://raw.githubusercontent.com/minghuiyang1998/PAFinal_MNode/install/ManagerNode.sh
sudo bash ManagerNode.sh

Worker Node:

wget https://raw.githubusercontent.com/JessiePen/PAFinal_WNode/yuhe/WorkerNode.sh
sudo bash WorkerNode.sh

Run locally

1. Use IDEA to start management program locally (current port is 8000)
2. Use IDEA to start worker program locally (current port is 58001)
3. Add localhost:58001 in localhost:8000/admin
4. Use localhost:8000/ to submit password.

Run in Internet

(Please use chrome, didn't consider website compatibility)

Website For Submitting password:

Website For managing working nodes:

1. Add worker node
2. Submit password cracking request

2. Framework
   - Front End:
     1. JQuery: deal with md5 and ajax
     2. Vue: deal with interactions
   - Back End (management)
     1. Spring Boot
     2. ThymeLeaf: template engine

3.Exception Handling

When a worker node crashes (manager node could not establish socket or could not get response), the manager node will reassign the former worker node jobs in thread pool to the new available nodes. And also set the crashed worker node as unavailable so that the following jobs will not be assigned to that node.

4. metrics, graphs, analysis

The table for confidence Intervals of total delay for cracking a md5 password are as follows

(sec)	n=3	n=5	n=7	n=10
$l = 26^3$	512.9649~717.9485	282.9833~404.2974	212.5909~286.4135	130.7369~180.5002
$l = 26^4$	544.4878~708.2046	336.8132~455.8628	225.5338~291.5073	149.0008~211.9388

Table 1: Confidence intervals of total time for cracking

(ms)	n=3	n=5	n=7	n=10
$l = 26^3$	0.406	0.547	0.703	0.62
$l = 26^4$	0.503	0.647	0.734	0.627

Table 2: Average RTT of each node in experiment

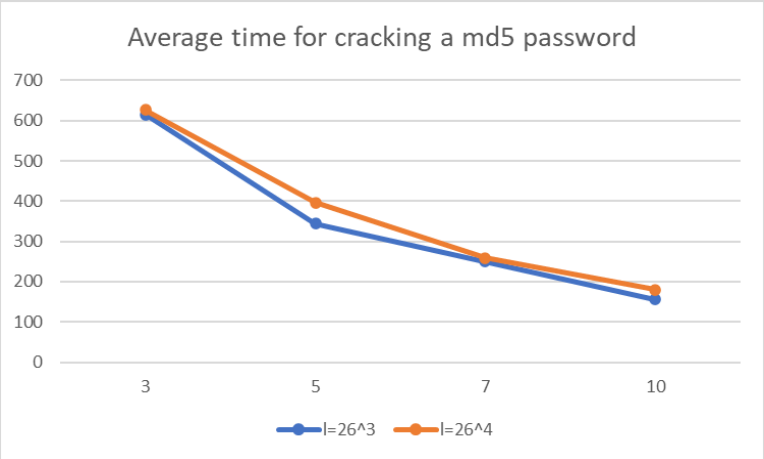


Figure 1: Average time for cracking a md5 password

	n=3	n=5	n=7	n=10
$l = 26^3$	1426	1271	1381	1255
$l = 26^4$	28	28	27	24

Table 3: Average times of distributing tasks

2.3 Analysis

- By increasing the number of worker nodes, the total time reduces since every worker node is cracking the md5 password simultaneously.
- Compared with the delay of calculation, the communication RTT is very small and can be neglected in our case. However, for tasks whose sizes are small enough (i.e. the time for calculation is close to RTT), we should take RTT into consideration.
- While a larger size of task can reduce the time for communication between the management node and some worker node, chances of timeout for redistribution increase (which is also shown in the Table 3). This explains why the average time for cracking when l=26^3 is greater than that when l=26^4.
- The number of requests can affect the actual time of cracking because the management node creates a thread for each request and the required time increases because of thread switching and thread sleep.

4. Conclusion

- With more resources (i.e. the worker nodes), the distributed system works more efficiently. However, it will also be more unstable so we need to deal with some errors of the subnodes to make sure other works.
- Use cache to store the calculated passwords, and thus save time for calculation.
- Instead of setting the same size of each task, consider an adaptive size when distributing tasks. For example, distribute a task of more passwords to the worker node which responds in a shorter time.

5. Labor

MInghui Yang	Design project structure, implement UI and front-end functionality.
Yinan An	Implement the manager node. Design and implement a distribution algorithm.
Yuhe Peng	Implement the worker node. Project deployment.
Jingyi Huang	Design experiments. Implement the operations on adding and deleting worker nodes.