# **Báo cáo bài tập: Giải mã MD5**

# I. Giới thiệu

Chương trình giải mã md5 sử dụng thư viện openssl và mpi để tính toán song song tăng hiệu năng của chương trình.

Trong bài tập này em chỉ giới hạn các ký tự thường và biết trước số lượng ký tự trong mật khẩu.

# II. Phương pháp

## Luồng thực hiện chính

Đầu tiên thực hiện mã hóa mật khẩu mình nhập vào thành MD5.

Truyền mật khẩu mã hóa và số lượng ký tự của mật khẩu, chương trình sẽ sinh số mật khẩu có thể có theo đúng số ký tự và ghi vào file, số lượng mật khẩu cho mỗi node tính toán = tất cả mật khẩu / số node tính toán.

Sau đó chia file ở trên để lấy mật khẩu đầu và cuối gửi cho mỗi tiến trình. Ví dụ có 2 tiến trình, số lượng mật khẩu giả sử có aaa, aab, aac, aad aae, aaf thì sẽ chia làm 2 và gửi cho node tính toán aaa, aac; sau đó các node sẽ tự tính toán tạo mật khẩu trong khoảng mật khẩu mà node nhận được.

Cuối cùng là tìm mật khẩu và gửi về cho node FrontEnd.

## Mã hóa mật khẩu thành MD5

Void encodePasswordMD5(const char \*string, char \*mdString), string chứa chuỗi mật khẩu truyền vào, mdString là mã MD5 được lưu vào sau khi kết thúc chương trình.

## Giải mã MD5

Các hàm trong chương trình:

* Void genPassword(char tempPassword[], int \*isPause, int point, int i) : Hàm tạo mật khẩu tiếp theo. Ví dụ aaa → aab
* Void splitData(list<string> \*passwordBound, int numberProcess, int lenPassword): chia đều mật khẩu cho các các node.

Ví dụ mật khẩu gồm có 2 ký tự và có 2 node tính toán, sẽ có tất cả (262 – 1) mật khẩu. Lấy 262 / 2 = 338, chuyển 338 về ký tự bằng cách 338 / 26 = 13 dư 0 . Vậy ký đầu tiên là ‘a’; tiếp tục lấy 13 chia 26 = 0 dư 13, ký tự tiếp theo là ‘n’ và ta tạo ra các cận ‘aa’, ‘an’, ‘zz’. FrontEnd gửi ‘aa’ và ‘an’ cho node 1, gửi ‘an’ và ‘zz’ cho node 2 sau đó chúng sẽ gọi hàm genPassword() và sinh mật khẩu.

Char \*findPassword(char passwordBegin[], char passwordEnd[], char \*mdString, char \*tempPasswordFound, int lenPassword): hàm tìm mật khẩu đúng.

# III. Cài đặt

## Cài đặt rock cluster

* Cài đặt frontend:
* B1:Cấu hình virtuabox: Chọn 2 mạng là NAT để kết nối với internet và Internal Network để các node kết nối với nhau.
* B2: Start virtualbox để cài đặt. Sau khi vào giao diện thì gõ “build”
* B3: Chọn roll bao gồm base, os, kernel, ganglia, hpc, webserver.
* B4: Cấu hình eth0:

IP: 35.0.0.1

Netmask: 255.255.0.0

* B5: Cấu hình eth1.

IP: 10.0.3.1

Netmask: 255.255.0.0

* B6: Cấu hình gateway:

10.0.3.2

DNS: 192.168.2.1

* B7: Chọn Install để cài đặt

Sau khi cài xong FrontEnd thì mở terminal rồi gõ lệnh "insert-ethers" để insert node.

* Cài đặt compute node
* B1: Cấu hình virtuabox: Để mạng Internal Network giống với Internal Network của FrontEnd. Cho thứ tự boot bằng network lên đầu.
* B2: Start virtuabox để cài đặt. Quá trình cài đặt sẽ diễn ra tự động.

## Cài đặt chạy chương trình

Ta phải chạy chương trình bằng user thường. Đăng nhập FrontEnd bằng tài khoản root và gõ các lệnh:

* useradd 'username' (thêm user mới)
* passwd 'username' (tạo password cho user vừa tạo)
* rocks sync users (đồng bộ user vừa tạo vào rockscluster)

Sau đó chuyển qua user thường để chạy chương trình MPI. Bật terminal và gõ các lệnh:

* ssh-agent $SHELL
* ssh-add

Tạo hostfile chứa tên các node compute để thực hiện tính toán.

Chạy chương trình: mpic++ mpi\_md5.cpp –o mpi\_md5 –lcrypto

Chạy mpirun –hostfile [hostfile] –np [number\_process] mpi\_md5 [mã MD5] [number\_password]

# IV. Thực nghiệm

Với 4 ký tự:

* 1 node tính toán : 0.58(s)
* 2 node tính toán : 0.37(s)
* 3 node tính toán : 0.27(s)

Với 5 ký tự:

* 1 node tính toán : 13.57(s)
* 2 node tính toán : 9.56(s)
* 3 node tính toán : 6.17(s)

# V. Kết Luận

Với số lượng node tính toán tăng lên thời gian có giảm nhưng không tuyến tính. Tăng 1 lên thành 2 node tính toán thời gian giảm 0.2(s) đối với 4 ký tự và 4(s) đối với 5 ký tự. Nhưng khi thêm node thì nó không giảm như vậy nữa mà chỉ giảm 0.1(s) đối với 4 ký tự và 3(s) đối với 5 ký tự. Điều này cho thấy không phải cứ tăng node là giảm thời gian tính toán.