**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

****

**BÁO CÁO TỔNG KẾT**

**Bài Tập Lớn Môn Học : Tính Toán Phân Tán**

**ĐỀ TÀI: Lập Trình MPI ,Thực Hành Giải Mã MD5 Sử Dụng Phương Pháp Vét Cạn**

**SINH VIÊN : Nguyễn Huy Phát**

**MSSV : 20143397**

**GIẢNG VIÊN: TS.Nguyễn Hữu Đức**

*Hà nội ngày 01 tháng 04 năm 2018*

**Mục lục**

[**I)** **Giới Thiệu Bài Toán :** 3](#_Toc510679479)

[1. Mã hóa MD5: 3](#_Toc510679480)

[2. Giải Mã MD5: 3](#_Toc510679481)

[3. Giải mã MD5 khi biết được dãy Hash của mật khẩu: 3](#_Toc510679482)

[**II)** **Phương Pháp :** 3](#_Toc510679483)

[1. Bruteforce (vét cạn): 3](#_Toc510679484)

[2. Sử dựng MPI trong Rockscluster : 4](#_Toc510679485)

[3. Thực hiện Bruteforce trên Rockscluster sử dụng MPI: 4](#_Toc510679486)

[**III)** **Cài Đặt :** 4](#_Toc510679487)

[**IV)** **Thử Nghiệm :** 5](#_Toc510679488)

[1. Tốc độ tính toán khi tăng số node: 5](#_Toc510679489)

[2. Thời gian tính toán khi cố định số node và số tiến trình: 5](#_Toc510679490)

[**V)** **Tổng Kết :** 5](#_Toc510679491)

[**VI)** **Tài Liệu Tham Khảo:** 5](#_Toc510679492)

1. **Giới Thiệu Bài Toán :**
2. Mã hóa MD5:

Trong mật mã học,MD5 (Message-Digest algorithm 5) là một hàm băm mật mã học được sử dụng phổ biến với giá trị Hash dài 128 bit.Là một chuẩn của Internet,MD5 được sử dụng nhiều trong các ứng dụng bảo mật,đồng thời cũng được dùng để xác định tính toàn vẹn của tập tin.Một bản băm MD5 được biểu diễn bằng một dãy 32 ký tự trong hệ thập lục phân(Hexa)

1. Giải Mã MD5:

* Tồn tại một dạng “sung đột ảo” của hàm nén MD5
* Kích thước 128 bit của mã MD5 là đủ nhỏ để có thể tấn công: Bruteforce (vét cạn)
* Càng ngày năng lực tính toán của các thiết bị càng cao,các hệ thống tính toán phân tán cực mạnh,dẫn đến việc mã MD5 không còn an toàn nữa.

1. Giải mã MD5 khi biết được dãy Hash của mật khẩu:

Đầu vào:

* Biết được dãy Hash của mật khẩu
* Khuôn khổ đề tài giới hạn không gian khóa : a 🡺 z
* Độ dài mật khẩu thực tế < 8 ký tự
* Thực hiện trên môi trường máy ảo

Đầu ra: mật khẩu của chuỗi hash đầu vào

1. **Phương Pháp :**
2. Bruteforce (vét cạn):
3. Sinh ra các chuỗi test (có thể là ngẫu nhiên hoặc theo một quy luật nào đó)
4. Cho chạy qua bộ mã hóa MD5,thu được các Hash tương ứng
5. So sánh kết quả thu được với chuỗi Hash của mã cần giải mã

+ Nếu giống nhau thì dừng

+ Nếu không giống,tiếp tục lặm lại các bước trên

1. Sử dựng MPI trong Rockscluster :

* MPI là một chuẩn cho lập trình truyền thông điệp
* MPI là một thư viện có thể chèn vào mã nguồn để truyền dữ liệu dữa các tiến trình
* Rockscluster là một phân phối opensource cho phép xây dựng các cụm máy tính

1. Thực hiện Bruteforce trên Rockscluster sử dụng MPI:

* Không gian mật khẩu cần vét cạn ( giới hạn mậ khẩu : a🡺z)
* 1 ký tự: a🡺z (26)
* 2 ký tự: aa 🡺 zz ( 26\*26)
* 3 ký tự: aaa 🡺 zzz (26\*26\*26)
* 4 ký tự: aaaa 🡺 zzzz (26\*26\*26\*26\*26)
* 5 ký tự: aaaaa 🡺 zzzzz (26\*26\*26\*26\*26\*26)
* 6 ký tự: aaaaaa 🡺 zzzzzz (26\*26\*26\*26\*26\*26\*26)
* 7 ký tự: aaaaaaa 🡺 zzzzzzz (26\*26\*26\*26\*26\*26\*26\*26)
* Cài đặt một cụm máy tính sử dụng rockscluster 6.2 bao gồm 1 frontend và 5 compute-node
* Phân chia mật khẩu trên các tiến trình:

+ Ta chia không gian mật khẩu đều ra cho tất cả tiến trình:

Ở đây do số node tính toán triển khai chỉ gồm có 6 node tính toán nên ta chỉ giới hạn số tiến trình tối đa là 26 tiến trình để thuận tiện cho việc phân chia mật khẩu:

Với N tiến trình ( N< 26) thì ta thực hiện chia 26 cho N, mỗi tiến trình sẽ tính trên khoản không gian N/26:

Ví dụ : Ta có 1 mảng gồm 26 phần tử là 26 phần tử trong bản chữ cái, ta có 5 tiến trình, ta chia 26 cho 5 được 5 dư 1. Ta chia mảng 26 phần tử cho 5 tiến trình:

* Tiến trình 0 sẽ nhận đầu vào là mảng gồm 5 phần tử :{a,b,c,d,e}.
* Tiến trình 1 sẽ nhận các phần tử đầu vào gồm 5 phần tử:{f,g,h,i,j}.
* Tiến trình 2 sẽ nhận các phần tử đầu vào gồm 5 phần Tử :{k,l,m,n,o},
* Tiến trình 3 sẽ nhận các phần tử đầu vào gồm 5 phần tử :{p,q,r,s,t},
* Tiến trình 4 sẽ nhận các phần tử đầu vào gồm 6 phần tử: {u,v,w,x,y,z}.

Từ mảng đầu vào đó các tiến trình sẽ tính toán trên không gian của mình:

Mỗi tiến trình sẽ thực hiện tính toán trên không gian tất cả các mật khẩu 1 ký tự, 2 ký tự, 3 ký tự, 4 ký tự, 5 ký tự, 6 ký tự, 7 ký tự mà ký tự đầu tiên là các ký tự trong mảng đầu vào:

* Tính toán trên mỗi tiến trình:

+ Mỗi tiến trình sẽ dựa vào rank của mình để xác định không gian mật khẩu cần tính toán như đã trình bày ở bước trên.

+ Mỗi tiến trình nhận đầu vào là đoạn mã MD5. Mỗi tiến trình thực hiện hash từng mật khẩu của không gian mà tiến trình đó chịu trách nhiệm tính toán, rồi so sánh với mã hash MD5 đầu vào, nếu khớp thì thực hiện ghi nhận vào 1 mảng. Khi kết thúc quá trình tính toán thì tiến trình đó sẽ gửi tất cả các mật khẩu mà có mã hash MD5 trùng với mã MD5 đầu vào cho tiến trình gốc tổng hợp sử dụng MPI\_Send.

+ Tiến trình gốc ngoài nhiệm vụ tính toán giống như các tiến trình khác còn làm thêm việc tổng hợp các mật khẩu mà các tiến trình khác gửi về, sử dụng MPI\_Recv.

* Lập trình MPI thực hiện giải mã MD5:

1. Nhận đầu vào là một mã Hash 32 bit
2. Tạo một mảng chứa 26 ký tự : a🡺z
3. Sinh ra các xâu có độ dài từ : 1🡺25
4. Phân phối các chuỗi cho các tiến trình
5. Đưa xâu qua bộ mã hóa MD5 để thu được dãy Hash của nó
6. Xo sánh với dãy Hash đầu vào tại mỗi tiến trình
7. Nếu node nào tìm thấy kế quả ,xẽ gửi kết quả về cho tiến trình gốc để tổng hợp.
8. Kết thúc trương trình,hiển thi mật khẩu,kết quả giải mã MD5 của chuỗi Hash đầu vào.
9. **Cài Đặt :**
10. Cài đặt rockscluster 6.2 trên môi trường máy ảo Vitrulbox,1 core và 1GB ram
11. Cài đặt 5 compute-node ,1 core và 1GB ram
12. Tạo chương trình : md5\_decode\_mpi.cpp tại frontend
13. Ngôn ngữ : C++
14. Thư viện : OpenMPI
15. Luồng thực hiện:
16. Tạo mảng chứ 26 ký tự : **khong\_gian\_khoa[26] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z}**
17. Xây dựng hàm mã hóa MD5: **void ma\_hoa\_MD5 (const char \*string, char \*mdString)**
18. Xây dựng hàm so sánh hai Hash: **bool so\_sanh\_hash (char \*md1, char \*md2)**
19. Xây dựng hàm chia dữ liệu cho các tiến trình: **int chia\_du\_lieu(int so\_tien\_trinh,int rank,char \*mang)**
20. Xây dựng hàm vét cạn tại các node:
21. **void rank0(int so\_tien\_trinh,char \*hash)**
22. **void ranki(int so\_tien\_trinh,int rank,char \*hash)**
23. Hàm : **int main(int argc, char \*\* argv)**, nơi thực thi chương trình

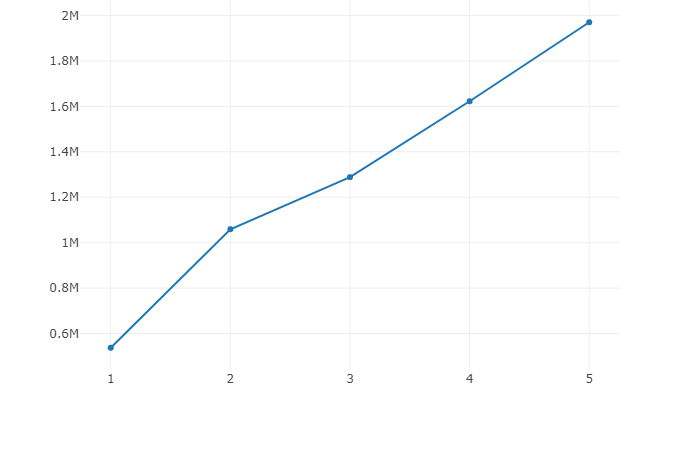
\*) Source code cài đặt :

<https://github.com/PhatBK/distributed_computing/blob/master/mpi_md5_decode.cpp>

1. **Thử Nghiệm :**
2. Tốc độ tính toán khi tăng số node:

|  |  |
| --- | --- |
| Số node | Số phép tính/s |
| 1 | 537692 |
| 2 | 1059307 |
| 3 | 1288655 |
| 4 | 1622437 |
| 5 | 1970144 |

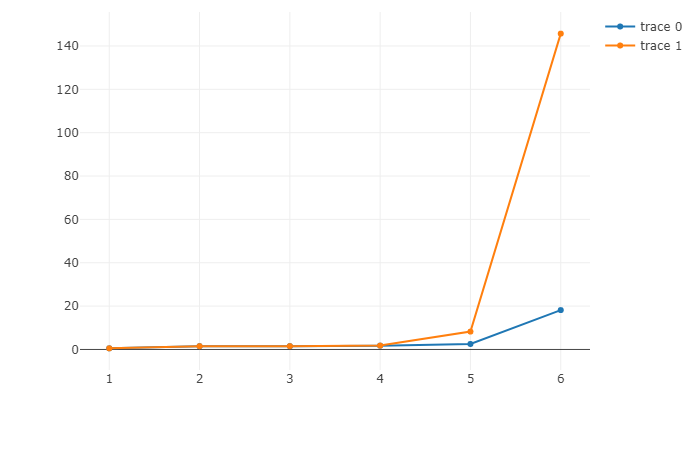
Biểu đồ:



1. Thời gian tính toán khi cố định số node và số tiến trình:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Độ dài | Mật khẩu | Thời gian dải mã | Mật khẩu | Thời gian giải mã |
| 1 | a | 0.512 | z | 0.5121 |
| 2 | ac | 1.505 | xz | 1.5085 |
| 3 | abd | 1.508 | zzz | 1.5199 |
| 4 | acdf | 1.7662 | zxwv | 1.7848 |
| 5 | dcfge | 2.5267 | zzxxw | 8.235 |
| 6 | bcfgek | 18.1586 | wzwzxz | 145.6799 |

Biểu đồ:



1. **Tổng Kết :**

* Cài đặt thử nghiệm được một hệ thống cụm máy tính sử dụng rockscluster 6.2
* Cài đặt thành công chương trình tính toán phân tán sử dụng MPI để truyền thông giữa các tiến trình
* Từ việc xây dựng chương trình giải mã MD5 bằng phương pháp vét cạn,ta thấy được tốc độ giải mã tăng đáng kể khi triển khai trên các node tính toán phân tán
* Thấy được điểm yếu của mã hóa MD5 trước các hệ thống tính toán phân tán
* Thấy được khi đặt mật khẩu theo bảng chữ cái nếu ta đặt theo các chữ cái ở đầu bảng chữ cái (thứ tự : a🡺z) thì dễ bị tấn công vét cạn hơn là đặt mật khẩu theo các ký tự cuối bẳng chữ cái
* Thấy được nhược điểm của phương pháp vét cạn đối với quá trình thám mã( rất tốn tài nguyên hệ thống,tốn thời gian)

1. **Tài Liệu Tham Khảo:**
2. <https://vi.wikipedia.org/wiki/MD5>
3. <http://www.rocksclusters.org/docs/guides-6-2.html>
4. <http://www.rocksclusters.org/assets/tutorials//tutorial-1.pdf>
5. <https://www.open-mpi.org/>
6. <https://www.open-mpi.org/doc/current/>
7. <https://github.com/waplet/unix_hw3>
8. <https://www.quora.com/What-is-the-best-way-to-create-an-MD5-hash-in-C>++
9. <https://stackoverflow.com/questions/31166313/calculate-md5-of-a-string-in-c>