

**BÁO CÁO MÔN HỌC**

**TÍNH TOÁN PHÂN TÁN**

**ĐỀ TÀI: Lập trình MPI, thực hành giải mã MD5**

**SINH VIÊN THỰC HIỆN:**

|  |  |
| --- | --- |
| **Họ tên** | **MSSV** |
| *Bùi Văn Trung* | *20144686* |

**GIẢNG VIÊN HƯỚNG DẪN:**

**TS. Nguyễn Hữu Đức**

Mục lục

[I, GIỚI THIỆU BÀI TOÁN: 5](#_Toc514344989)

[1, Giải thuật mã hóa MD5: 5](#_Toc514344990)

[2, Giải mã MD5: 5](#_Toc514344991)

[II, PHƯƠNG PHÁP: 5](#_Toc514344992)

[1, Vét cạn: 5](#_Toc514344993)

[2, Sử dụng MPI trong Rocks Cluster: 6](#_Toc514344994)

[3, Thực hiện vét cạn trên Rocks Cluster sử dụng MPI: 6](#_Toc514344995)

[III, CÀI ĐẶT: 6](#_Toc514344996)

[IV, KẾT QUẢ THỬ NGHIỆM: 7](#_Toc514344997)

[V, TỔNG KẾT : 8](#_Toc514344998)

[VI, Tài liệu tham khảo 9](#_Toc514344999)

# **I, GIỚI THIỆU BÀI TOÁN:**

## **1, Giải thuật mã hóa MD5:**

* MD5 (Message-Digest algorithm 5) là một thuật toán mã hoá, theo chuẩn RFC 1321 (http://tools.ietf.org/html/rfc1321). Các chương trình mã hoá (tính) MD5 thường được gọi là MD5 CheckSum.
* Cũng như các hàm băm khác như MD4 hay SHS (Secure Hash Standard), MD5 là phương pháp có ưu điểm là tốc độ xử lý nhanh, thích hợp với các thông điệp dài. Nó lấy đầu vào là các khối 512 bit và sinh ra các giá trị băm 128 bit. Đây là phương pháp được tin là phi đụng độ và one-way, do đó nó thường được dùng để kiểm tra tính toàn vẹn dữ liệu của một tập tin.
* Thuật toán MD5 được thiết kế cho phép chạy tốt nhất trên các máy tính 32 bit. Nó sử dụng các phép toán đơn giản như phép cộng module 32, do đó thích hợp với công việc mã hóa cho các bộ xử lý 32 bit.
* Một bảng băm MD5 thường được diễn tả bằng một số trong hệ thập lục phân với 32 ký tự.

**2, Giải mã MD5:**

* Ngày nay, khi năng lực tính toán của các thiết bị ngày càng cao, kích thước 128 bit của mã MD5 là đủ nhỏ để có thể tiến hành giải mã theo phương pháp vét cạn.
* Giới giạn độ dài của mật khẩu <= 6 ký tự với không gian khóa từ a 🡪 z, A 🡪 Z,

0🡪9

* Thực hiện giải mã trên môi trường máy ảo (Rocks Cluster).

# **II, PHƯƠNG PHÁP:**

## **1, Vét cạn:**

* Tạo các chuỗi password ngẫu nhiên để tiến hành mã hóa MD5.
* Cho chạy qua bộ mã hóa MD5 thu được các Hash tương ứng.
* So sánh các chuỗi Hash vừa thu được với chuỗi Hash của mã cần giải mã:

(+) Nếu trùng thì dừng -> Tìm được mật mã.

(+) Nếu không có kết quả trùng thì lặp lại tất cả các bước phía trên.

* Để nâng cao hiệu năng của chương trình, ta sẽ sử dụng chuẩn MPI cho phép chạy song song trên nhiều máy.

## **2, Sử dụng MPI trong Rocks Cluster:**

* MPI (Message Passing Interface) là một chuẩn cho lập trình truyền thông điệp; đồng thời là một thư viện các hàm có thể chèn thêm vào mã nguồn để truyền dữ liệu giữa các tiến trình.
* Rock Clusters là một bản phân phối Linux cho phép xây dựng các cụm máy tính hiệu năng cao. Rocks ban đầu được dựa trên phân phối Red Hat Linux, tuy nhiên các phiên bản hiện đại của Rocks được dựa trên CentOS, với một trình cài đặt Anaconda sửa đổi nhằm đơn giản hóa quá trình cài đặt trên nhiều máy tính.

## **3, Thực hiện vét cạn trên Rocks Cluster sử dụng MPI:**

* Cài đặt một cụm máy tính (bao gồm 1 front-end và một số compute-node) trên môi trường máy ảo VirtualBox sử dụng Rocks Cluster 6.2.
* Sử dụng MPI giải mã MD5:

(+) Đầu vào là một mã Hash của password cần tìm.

(+) Tạo một mảng alphabet chứa các ký tự: a -> z, A -> Z, 0 -> 9.

(+) Sinh các mã hash gồm 32 kí tự.

(+) Phân phối các chuỗi cho các node.

(+) Mã hóa MD5 các xâu.

(+) So sánh các Hash thu được với Hash đầu vào tại từng node.

(+) Thông báo kết quả sau khi tấn công vét cạn.

# **III, CÀI ĐẶT:**

1. Ngôn ngữ sử dụng: C
2. Cài đặt rockscluster 6.2 trên môi trường máy ảo Vitrulbox,1 core và 1GB ram
3. Cài đặt 4 compute-node ,1 core và 1GB ram
4. Tạo chương trình : mpi-md5.c tại frontend
5. Thư viện sử dụng : <openssl/md5.h> để tạo mã băm , thư viện <mpi.h> để tính toán phân tán
6. Các hàm sử dụng:

* str2digest\_md5(const char \*str, unsigned char \*digest) : chuyển từ chuỗi thành 1 mã băm digest.
* hex2dig(const char \*str, unsigned char \*digest): chuyển từ mã hexa đầu vào gồm 32 kí tự thành 1 mã digest.
* matches(const unsigned char \*s1, const unsigned char \*s2) : so sánh chuỗi đã mã hóa với chuỗi đích
* bruteForce(int process\_size, int len, int rank) : tấn công vét cạn với đầu vào là số tiến trình, độ dài tối đa của chuỗi đầu vào, rank của tiến trình.

1. Cài đặt : mpicc –Wall –o mpi-md5 mpi-md5.c –lcrypto –lssl
2. Để chạy chương trình: mpirun –np <số tiến trình> -host [tên host,] ./ mpi-md5 (đường dẫn tới file vừa make) <mã md5 32 kí tự> <số kí tự tối đa của mật khẩu>.
3. Ví dụ: mpirun –np 4 –host hust.edu.vn,compute-0-1,compute-0-3,compute-0-4 ./mpi-md5 d3eb9a9233e52948740d7eb8c3062d14 5

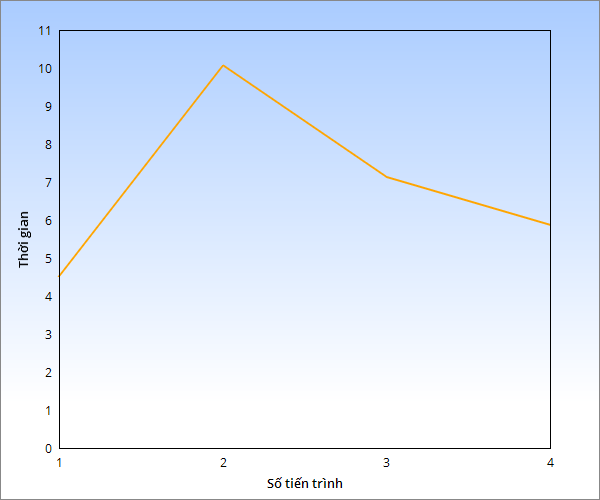
(\*\*\*)Source code cài đặt :

# **IV, KẾT QUẢ THỬ NGHIỆM:**

* Chương trình chạy với số tiến trình và độ dài khóa thay đổi:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Số tiến trình  Độ dài mk | **1** | **2** | **3** | **4** |
| **3** | 0.07s | 0.162s | 0.115s | 0.09s |
| **4** | 4.53s | 10.07s | 7.13s | 5.87s |
| **5** | 268s | 618s | 423s | 342s |

* Đồ thị theo số tiến trình với độ dài mật khẩu là 4:



* Nhận xét kết quả :
* Ta thấy thời gian thực hiện với số tiến trình là 1 ngắn nhất
* Thời gian thực hiện giảm dần tương ứng với số tiến trình tăng từ 2 🡪 4
* Giải thích kết quả:
* Do cài đặt chương trình trên máy ảo với tài nguyên là số core hạn chế nên thời gian thực hiện với số tiến trình là 1 ngắn nhất.
* 1 lý do nữa là thời gian truyền thông giữa các tiến trình ở các node phân tán.
* Thời gian thực hiện giảm dần với số tiến trình từ 2 trở lên, chứng tỏ sự hiệu quả của việc phân chia công việc và thực hiện song song các tiến trình trên nhiều node.

# **V, 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 nhược điểm của phương pháp vét cạn đối với quá trình thám mã( rất tốn tài nguyên hệ thống,tốn thời gian)

# **VI, Tài liệu tham khảo**

1. <https://vi.wikipedia.org/wiki/MD5>
2. <http://www.rocksclusters.org/docs/guides-6-2.html>
3. <https://www.open-mpi.org/>
4. <https://github.com/waplet/unix_hw3>