Politechnika Poznańska
Systemy równoległe i rozproszone

Krzysztof Bojakowski

Rozproszony system do łamania pliku z hasłami

Rozproszony system do łamania pliku z hasłami 1

Wprowadzenie 1

Konfiguracja klastra 1

Zasada działania 3

Funkcje i metody 4

1. Skrypt python 4
2. Program C 4

Zrzuty ekranu 5

Źródła 7

1. Wprowadzenie

Celem projektu było utworzenie rozproszonego systemu do łamania pliku z hasłami. Program został zrealizowany w języku C, wykorzystując bibliotekę MPICH2 na system Linux - Ubuntu. W pliku zapisane są hasła zahaszowane algorytmem MD5, które zostały wygenerowane za pomocą skryptu w języku python 2.7.

2. Konfiguracja klastra

Konfigurację należy zacząć od utworzenia minimum dwóch maszyn wirtualnych, jeden master oraz jeden lub więcej slave'ów. Należy dodać maszyny do tej samej sieci np. 192.168.0.1/24, 192.168.0.2/24, należy ustawić jedną z kart sieciowych każdej maszyny wirtualnej w tryb "Internal Network".

Zaczynamy od zmiany hostname, przykładowy schemat to maszyna "Master", kolejne "SlaveX", gdzie X to liczba.

Wykonujemy polecenie:

hostname nazwa

Dla pewności w pliku */etc/hostname*, również wprowadzamy podaną wyżej nazwę. Na niektórych dystrybucjach konieczne będzie wymuszenie zmian przez komendę:

/etc/init.d/hostname.sh start

Ważne, aby katalog /home/user/.ssh posiadał odpowiednie uprawnienia, w tym celu wpisujemy:

chmod 700 /home/user/.ssh

Pobieramy MPICH2 (http://www.mpich.org/), następnie wypakowujemy archiwum i przemieszczamy się do katalogu MPICH2. Rozpoczynamy instalację wpisując:

./configure --disable-fortran

Następnie, po wstępnej konfiguracji wpisujemy:

make; sudo make install

Sprawdzamy czy wszystko zostało zainstalowane poprawnie wywołując poniższy kod, jeżeli instalacja się udała, powinna zostać wyświetlona wersja MPI.

***mpiexec --version*.**

Wypełniamy plik */etc/hosts*, który mapuje nazwy hostów do adresów IP. Przykładowa zawartość przedstawiona niżej, każda z konfigurowanych maszyn musi posiadać takie wpisy:

127.0.0.1 ***localhost***
192.168.0.1 master
192.168.0.2 slave1
192.168.0.2 slave2

Kolejnym krokiem jest konfiguracja SSH. W tym celu wywołujemy:

sudo apt-get install openssh-server

W tym momencie możemy wygenerować klucze SSH na maszynach slave, wpisujemy:

ssh-keygen -t rsa

Wygenerowany klucz publiczny dla danej maszyny przekazujemy zewnętrznemu hostowi - w tym wypadku maszynie o nazwie master:

ssh-copy-id master

Alternatywnie możemy posłużyć się adresem IP.

ssh-copy-id 192.168.0.1

Dla MPICH2 istotne jest by móc logować się za pomocą SSH bez hasła, w tym celu wykonujemy:

eval `ssh-agent`
ssh-add ~/.ssh/id_dsa

Następnie testujemy połączenie SSH, powinno nastąpić logowanie bez podawania hasła z maszyny master do maszyn slave. W przeciwnym wypadku możliwe, że będzie trzeba dokonać odpowiednich konfiguracji w pliku */etc/ssh/sshd_config*.

ssh slave1
ssh slave2
...

Na maszynie master instalujemy serwer NFS:

NFS server *sudo apt-get install nfs-kernel-server*

Następnie tworzymy katalog, w którym będzie program, współdzielone pliki - jest to miejsce na dysku, do którego będą miały dostęp wszystkie maszyny w klastrze.

mkdir cloud

W pliku */etc/exports* dopisz utworzony katalog, wpsiując:

/home/master/cloud *(rw,sync,no_root_squash,no_subtree_check)

Następnie wymuszamy zmiany i restartujemy NFS server za pomocą komendy:

exportfs -a; sudo service nfs-kernel-server restart

Na maszynach slave instalujemy NFS klienta:

sudo apt-get install nfs-common

Następnie tworzymy katalog *cloud tak jak wyżej*, montujemy go wywołując:

sudo mount -t nfs master:/home/master/cloud ~/cloud

Sprawdzamy zamontowany katalog za pomocą poniższej komendy, powinna wyświetlić się wzmianka o lokalizacji master:/home/master/cloud.

df -h

Skompilowany program przenosimy do folderu cloud, lub tam go kompilujemy, dokonujemy tego przez polecenie:

mpicc -o test test.c

Aby wykonać program w rozproszonym środowisku wykonujemy polecenie mpirun, gdzie np to liczba hostów, zaś hosts to konkretne aliasy dla hostów oddzielone przecinkiem, ./test to nazwa pliku do wykonania.

mpirun -np 5 -hosts slave1,slave2,localhost ./test

Na niektórych dystrybucjach odnotowano problem kiedy nazwa domyślna użytkownika ssh, jest różna od podstawowego konta w systemie linux.

Również konieczne jest zainstalowanie zewnętrznej biblioteki OpenSSL, możemy tego dokonać wpisując:

apt-get install libssl-dev

Można też pobrać package ze strony producenta i wpisać:

cd openssl-1.1.0c
./config shared --prefix=/usr/local/ssl --openssldir=/usr/local/ssl
make
make install
cd /usr/local/ssl/bin
./openssl version

Jeżeli po wykonaniu ostatniej komendy nie wyświetli się wersja SSL, możemy spróbować dodać SSL do PATH.

export PATH="/usr/local/ssl/bin:$PATH”

3. Zasada działania

Właściwy kod programu kompilujemy za pomocą narzędzia mpicc z parametrami poniżej, są to dołączone biblioteki do haszowania MD5 z biblioteki OpenSSL.

-L/usr/lib -lssl -lcrypto

mpicc -o mpi_hello_world mpi_hello_world.c -L/usr/lib -lssl -lcrypto

Program uruchamiamy podając liczbę hostów biorących udział w wykonywaniu programu oraz ich nazwy. Hosty muszą być zdefiniowane w pliku /etc/hosts.

mpirun -np 5 -hosts prime,slave1 ./mpi_hello_world

W pliku ‘passwords’ przechowywane są hasła w postaci hasza MD5. Hasła mogą składać się z dowolnych liter ASCII, o dowolnej długości. Celem programu jest znalezienie wartości odpowiadającej wartości odczytanej z pliku przez wywołanie funkcji haszującej na permutacji ciągu znaków. W tym celu zaimplementowana została prosta metoda oparta o metodę brute force. Ciągi znaków są generowane poziomami (pierw permutacje ciągów 1-znakowych, później 2, 3-znakowych, etc.). Podział zadań na procesy polega na podziale zbioru ASCII dla pierwszej litery przez procesy. Przykład dla alfabetu łacińskiego, gdzie mamy 26 znaków, przy podziale na 26 procesy.

1. proces - aa, ab, ac, …, az
2. proces - ba, bb, bc, ..., bz
...
25. proces - ya, yb, yc, …, yz
26. proces - za, zb, zc, …, zz

Jeżeli podzielilibyśmy powyższy zbiór na 13 procesów, wtedy pierwszy proces będzie miał litery a, b, drugi c oraz d, itd. Dzięki temu każdy z procesów wykonuje taką samą pracę. Warto wspomnieć, że procesy pracują nad tym samym plikiem, który jest wczytywany tylko przez proces o numerze 0, który następnie przesyła dane do reszty procesów za pomocą MPI bdcast, która jednocześnie jest barierą - zsynchronizuje procesy. W momencie, gdy znajdowany jest wynik dla danego hasza - wtedy proces, który go znalazł rozsyła informację o tym. Inne procesy po otrzymaniu wiadomości natychmiast zaprzestają pracę. System rozproszony składa się z maszyny głównej oraz maszyn podlegających. Maszyna główna również bierze udział w wykonywaniu obliczeń. Program permutuje ciągi znaków aż do momentu kiedy liczba znaków permutowanych jest mniejsza niż równa polu “BRUTEFORCE_DEPTH”. Zmieniając parametry CHAR_OFFSET oraz CHARS możemy zarządzać tym jakie znaki ASCII mają być brane pod uwagę. CHAR_OFFSET to liczba dziesiętna pierwszego znaku, który ma być brany pod uwagę w analizie, zaś CHARS to liczba kolejnych znaków.

4. Funkcje i metody

1. Skrypt python

Skrypt opiera się o bibliotekę md5.

h(f, str, last=False)

Metoda h służy do zapisywania zahaszowanych ciągów znaków do pliku. F to uchwyt do pliku, str to ciąg znaków do zahaszowania (dowolne znaki ASCII), parametr last mówi, czy jest to ostatni wiersz w pliku (jest to istotne do parsowania).

2. Program C

Program napisany w C opiera się o OpenSSL. Zostały zadeklarowane podstawowe metody:

bool bruteforceLevel(char *source, char* str, int index, int maxLength, int world_size, int world_rank)
Funkcja rekurencyjna, odpowiadająca za generowanie permutacji ciągu znaków, tutaj porównywane są hasze, odbywa komunikacja z innymi procesami. Parametry oznaczają kolejno: source - zahaszowany tekst nieznany, str - ciąg znaków do zahaszowania, tablica char, index - wskaźnik na indeks tablicy char (str), maxLength - głębokość do jakiej szukany jest pasujący hasz, int world_size - liczba procesorów, int world_rank - numer procesora.

void bruteforce(char *source, int maxLen, int world_size, int world_rank)
Z tej metody wywoływana jest metoda bruteforceLevel, parametry oznaczają dokładnie to samo. Inicjalizuje atak brute force na hasła. Odpowiada za wywoływanie haszowania na kolejnych poziomach permutacji (1 znak, 2 znaki, etc.)

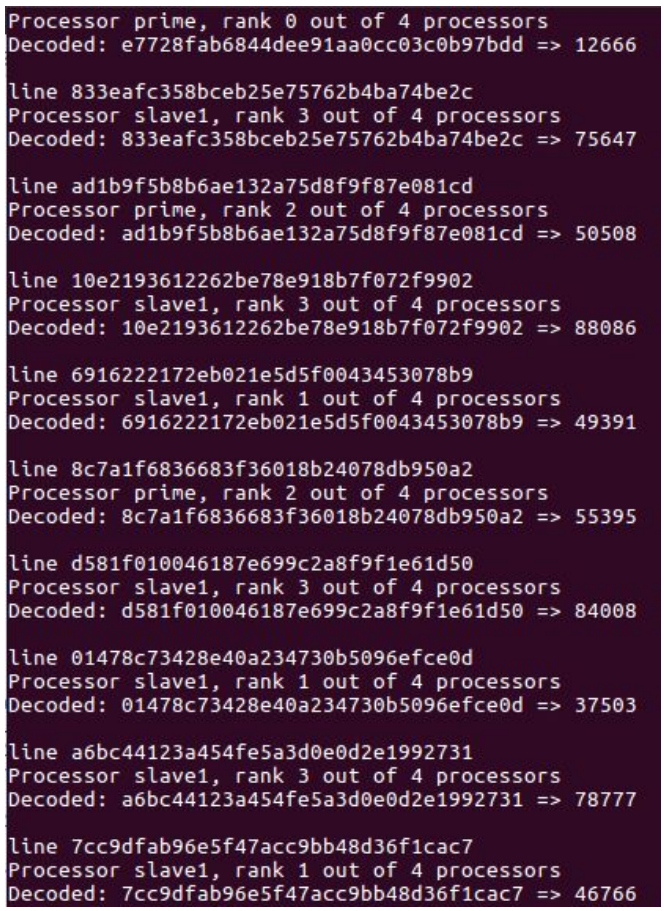
bool compareMD5(char *str1, char *str2) -
Funkcja porównuje czy wartości haszów MD5 są sobie równe.

void getMD5(char *md5string, char *string) -
Pobierany jest hasz z ciągu string i zapisywany do md5string.

5. Problemy

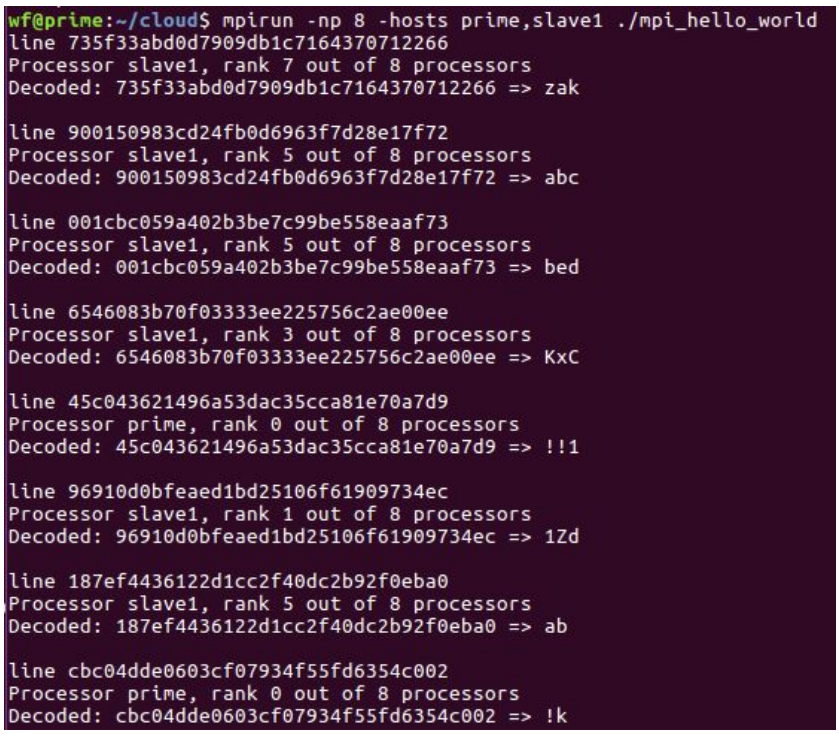
Ze względu na pewien narzut, związany wykorzystywaniem rekurencji, program im dłużej działa - tym wolniej wykonuje obliczenia. Dzieje się tak, gdyż ramka stosu rośnie wraz z wywoływaniem tysięcy wywołań rekurencyjnych. Również posiada ograniczenie co do liczby jednocześnie pracujących procesów - od 0 do 255, a praktycznie od X do Y, gdzie X to początek rozważanego zbioru ASCII, a Y to liczba kolejnych znaków następujących po X. W celu rozwiązania tych problemów należałoby zmienić funkcję w iteracyjną - jednakże nie ma gwarancji co do znacznej poprawy wydajności. W celu umożliwienia większej ilości procesów, można by by dzielić również podzbiory w celu ich permutowania.

6. Zrzuty ekranu



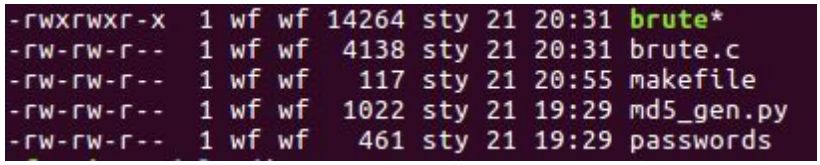
Rys. 1.: Wykonywanie programu, w którym zostały zahaszowane liczby.

Na rys. 1., line jest to aktualnie wczytany hasz z pliku, informacja o procesorze jest wyświetlana kiedy, któremuś z procesów uda się złamać hasz. Następnie wypisywana jest wartość odszyfrowana.



Rys. 2.: Wykonywanie programu, w którym zostały zahaszowane ciągi znaków.

Na rys. 2. możemy zaobserwować, że zostały odszyfrowane również znaki. Oznacza to, że wykorzystany został szerszy zbiór znaków ASCII.



Rys. 3.: Wylistowanie zawartości katalogu wspólnego cloud.

Na rys 3., możemy zobaczyć pliki, które znajdują się w katalogu, z czego maszyny slave dostępu potrzebują do pliku wykonywalnego brute oraz pliku z hasłami passwords.

7. Źródła

Instalacja MPICH2 na systemie linux, konfiguracja klastrów, obsługa systemu linux
http://mpitutorial.com/tutorials/running-an-mpi-cluster-within-a-lan/
http://www.ducea.com/2006/08/07/how-to-change-the-hostname-of-a-linux-system/

Sprawdzanie poprawności zahaszowanych haseł
http://md5.gromweb.com/?string=