Universidade Federal do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES

Processamento Paralelo

Prof. Oberlan Romão

Exercício Programa 2

Password Cracking[1]

Atenção: Esse EP pode ser feito em dupla. Nesse caso, apenas um dos integrantes deve enviar o EP pelo AVA (identifique a dupla no relatório).

Em criptografia, um ataque de força bruta, ou busca exaustiva, é um ataque criptoanalítico que pode, em teoria, ser usado contra quaisquer dados criptografados. Tal tipo de ataque pode ser usado quando não é possível tomar vantagem de outras fraquezas em um sistema de criptografia (se existir) que tornariam a tarefa mais fácil. Ele consiste de verificação sistemática de todas as possíveis senhas até que a correta seja encontrada.

Esse é o objetivo desse EP: testar todas as possibilidades de senhas para tentar descriptografar um determinada senha (criptografada usado SHA1). Ao fim, seu programa deve, basicamente, imprimir a senha descriptografada (caso obtenha sucesso). Claro que isso só é possível tendo algumas informações sobre as senhas, por exemplo, o tamanho máximo de caracteres, se é composta apenas de letras (minusculas e maiúsculas), se possui números, acentos, símbolos, etc. Como todos já sabem, quando maior for a senha e/ou quando mais tipos de caracteres for usados mais difícil é de se quebrar uma senha usando força-bruta, já que esse algoritmo é exponencial.

Por exemplo, suponha que a senha que se deseja descriptografar seja: `902ba3cda1883801594b6e1b452790cc53948fda`. Sabendo que senha descriptografada possui apenas um caractere e esse é composto por um dos caracteres $\{a, b, 0, 1, ..., 9\}$, geramos todas as possibilidades (12), criptografamos cada uma usando um algoritmo SHA1 e comparamos o valor obtido com a senha criptografa.

Senha	SHA1
a	86f7e437faa5a7fce15d1ddcb9eaeaea377667b8
b	e9d71f5ee7c92d6dc9e92ffdad17b8bd49418f98
0	b6589fc6ab0dc82cf12099d1c2d40ab994e8410c
1	356a192b7913b04c54574d18c28d46e6395428ab
2	da4b9237bacccdf19c0760cab7aec4a8359010b0
3	77de68daecd823babbb58edb1c8e14d7106e83bb
4	1b6453892473a467d07372d45eb05abc2031647a
5	ac3478d69a3c81fa62e60f5c3696165a4e5e6ac4
6	c1dfd96eea8cc2b62785275bca38ac261256e278
7	902ba3cda1883801594b6e1b452790cc53948fda
8	fe5dbbcea5ce7e2988b8c69bcfdfde8904aabc1f
9	0ade7c2cf97f75d009975f4d720d1fa6c19f4897

Nesse caso, o programa deve retornar, como resposta, 7 (indicando que a senha descriptografada é 7).

[1] Descrição baseada em: https://en.wikipedia.org/wiki/Brute-force_attack

Como você já deve saber a forma mais simples de se evitar que alguém consiga quebrar uma senha (usando força-bruta) é aumentar a "força" da senha, ou seja, ter uma senha com muitos dígitos e combinar letras (minúsculas e maiúsculas) com números e símbolos. Por isso, usar uma senha que várias outras pessoas utilizam[2] (`12345`, `pass`, `ninja`, `jesus`, `money`, `54321`, `abc12`, `AbCdE`, `Red`, `love`, `senha`, `globo`, `amor`, `012ab`, `A1B2`, `A2019`, `sexy`, `admin`, etc) é uma péssima ideia.

Tarefa

Você deve implementar um programa que faça uma busca exaustiva paralela, **usando MPI**, para tentar descriptografar uma senha (criptografada usando SHA1). Seu programa deve receber como argumento, a senha a ser descriptografada. Por exemplo, supondo que seu programa se chame EP2.exe, a seguinte execução:

```
./EP2.exe 902ba3cda1883801594b6e1b452790cc53948fda
```

irá tentar descriptografar `902ba3cda1883801594b6e1b452790cc53948fda`. Como resposta, seu programa deve imprimir, caso consiga, a senha descriptografada.

Considere que as senhas que serão descriptografadas (tabela abaixo), terão as seguintes características:

- **Tamanho**: até 5 caracteres;
- **Caracteres**: letras minúsculas e maiúsculas e números (`a-zA-Z0-9`);

Supondo que o tamanho da senha seja 2, para gerar todas as possibilidades, você pode usar a seguinte estratégia (fique livre para usar outra estratégia):

```
char caracteres[]="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

for(int i = 0; i < strlen(caracteres); i++)
    for(int j = 0; j < strlen(caracteres); j++)
        cout << caracteres[i] << caracteres[j] << endl;
```

Se o tamanho da senha for 3, basta adicionar mais um for alinhado:

```
for(int i = 0; i < strlen(caracteres); i++)
   for(int j = 0; j < strlen(caracteres); j++)
      for(int k = 0; k < strlen(caracteres); k++)
         cout << caracteres[i] << caracteres[j] << caracteres[k] << endl;
```

Dessa forma, você pode testar primeiro senhas de tamanho 1, depois de tamanho 2, e assim por diante. O grande desafio é dividir e balancear a geração das senhas entre os processos.

[2] https://www.tiforense.com.br/as-100-senhas-mais-utilizadas-de-2018/

Experimentos

Todos os experimentos devem ser feitos no Cluster do Ceunes (use sempre o `qsub` para submeter os jobs) considerando o número de processadores (`-np`) = $\{1, 8, 16, 32\}$. Para o relatório, você deve apresentar a seguinte tabela (preenchida) para os diferentes quantidade de processadores:

#	SHA1	Senha	Tempo (seg) 1 8 16 32
1	8cb2237d0679ca88db6464eac60da96345513964		
2	7b151de317f2547df39e1a1ff2850a6abfa6128f		
3	230cdfcdb6f2aa33b7acf19edaae5a216a14155f		
4	b2cdbf0601d8ae90d3cda1c978566ace86c4eac0		
5	a045b7efa463c6ed195c644163f4168952fbd34a		

Apresente também o tempo médio gasto para descriptografar as 5 senhas para cada valor de `np`. Sinta-se livre para quebrar outras senhas.

Desafio: quebrar a senha `d27e086d60a993d203717509c46a6752daccc967`

O que entregar

- Implementação (.c ou .cpp) usando MPI do EP;
- Um relatório (.pdf) reportando a estratégia usada para paralelizar os códigos, os experimentos feitos e os gráficos obtidos. Considere o tempo gasto para comparar as diferentes versões. Seu relatório deve conter, no mínimo, uma discussão (gráficos podem e vão ajudar) a respeito da estratégia usada para distribuir a geração de senhas entre os diferentes processadores e o tempo gasto para quebrar cada senha variando a quantidade de processadores utilizados.

Data de entrega: até às 6h do dia 06/07/2019

Observações:

1. Você deve implementar o código usando a linguagem de programação C/C++ e MPI;
2. Para gerar os gráficos e relatório, execute seu programa, no mínimo, 3 vezes;
3. Não envie nada mais que os códigos-fontes e o arquivo do relatório;
4. Todos os experimentos devem ser feitos do cluster, usando o comando `qsub`;
5. Para medir o tempo, use, preferencialmente, a função `MyClock()` ou o comando `time` do Linux;
6. Contabilize o tempo total do seu programa, algo assim (em C++ e usando a biblioteca chrono):

```
int main(){
    auto start_time = timeNow();
    ...
    auto end_time = timeNow();
    cout << duration(end_time - start_time) / 1000.f << " ms\n" << flush;
    return 0;
}
```

7. Em caso de plágio, será atribuído 0 a todos os envolvidos.

Critérios de avaliação:

A nota geral do EP será da seguinte forma:

- Implementação do algoritmo: 5.0 pontos;
- Relatório técnico: 5.0 pontos (caprichem). O relatório deve conter, pelo menos, os seguintes pontos:
  - Análise e discussão da(s) estratégia(s) usada(s) para tentar quebrar a senha;
  - A eficiência do seu programa: número de senhas testadas por segundo;
  - Gráficos (preferencialmente, feitos com o `Matplotlib`) comparando os resultados. Sempre que possível, exiba a média e o desvio padrão.

Bom trabalho!!!