Computação Distribuída

Projeto Final
Distributed Password Cracker

Fábio Martins NMEC -98119

Luca Pereira NMEC -97689

Arquitetura

A arquitetura desenvolvida foi uma que fizesse o mínimo uso de mensagens entre os *slaves* possível , estas apenas foram usadas no inicio, de maneira a coordenar os esforços dos mesmos na geração de passwords , e no fim para quando a password for descoberta notificar os restantes que o objetivo foi alcançado. Desta maneira após a sincronização dos slaves ao inicio, estes podem usar todos os seus recursos para realizar tentativas ao servidor.

Para tal definiu-se um algoritmo em que todos os caracteres possíveis (letras maiúsculas, minúsculas e números *ascii*) , seriam divididos em grupos de tamanhos iguais por todos os *slaves*, sendo cada grupo distinto entre si. Após cada slave ter o seu grupo, este irá começar a gerar passwords iniciadas por cada um das letras do seu grupo e com um tamanho crescente, sem repetição através do comando *iteratools.product*. Este gera um producto cartesiano entre os caracteres possíveis mencionados anteriormente, por exemplo (a,b,c,d,…,aa,ab,ac,ad…). Com este algoritmo fica assim garantido que não só os slaves não reptem as suas próprias passwords como não reptem as dos outros slaves tornando o processo de geração de passwords eficiente.

Arquitetura Cont.

Um pormenor a notar, a geração das passwords é determinística, ou seja dado que as passwords são geradas do resultado de produtos cartesianos, a ordem da geração das passwords é a mesma para cada número de slaves. Isto não causa ganho nem perda na eficiência visto que a password alvo é aleatória, podendo ser tanto muito ou pouco "favorável". Passwords que têm as suas letras da esquerda iguais aos índices iniciais de cada grupo serão extremamente rápidas de decifrar e vice-versa, A média deverá tender para os tempos dos índices do meio.

Obviamente que esta arquitetura tem também desvantagens. Por exemplo, a comunicação entre slaves é feita apenas no início de forma a fazer a distribuição dos grupos. O sistema fica assim extremamente dependente das condições iniciais, ou seja, se, por exemplo, estiverem 3 slaves a trabalhar simultaneamente e um deles falhar, o sistema fica comprometido pois deixa de ser capaz de reproduzir 33% das passwords possíveis. Num caso desses seria necessário um mecanismo de deteção de falhas e um algoritmo para fazer uma redistribuição dos grupos pelos dois slaves restantes. Neste projeto não foram implementados tais sistemas pelo que este sistema em concreto não é tolerante a falhas.

Protocolo

- Neste trabalho, como já referido foram apenas usados 2 tipos de mensagens:

  - JOIN: A mensagem JOIN é enviada por todos os slaves entre si (por multicast), onde nela segue os ID de cada um deles. Com estes dados cada slave consegue calcular quantos slaves existem. Através da comparação de ID e do número de slaves cada um deles consegue descobrir o seu grupo de caracteres.
  - DONE: A mensagem DONE é enviada pelo slave que descobrir a password em primeiro lugar , e apenas indica a todos os outros que a password foi descoberta e que podem parar as suas execuções

Resultados

- No final do desenvolvimento deste projeto conseguimos ter um grupo de 3 slaves que comunicam entre si nas condições referidas anteriormente.
- Em termos de tempo, como já havia sido referido, a sua execução irá depender dos caracteres da password, sendo que o algoritmo pode acertar na password no seu “início” ou no seu “final”
- Com N palavras passes diferentes, o tempo médio de decifração da palavra-passe será de 2 minutos para 2 caracteres.

Notas Finais

- No desenvolver deste projeto, temos em conta que não iremos saber o tamanho verdadeiro da password, portanto geramos todas as combinações possíveis até ao tamanho máximo das palavras-passes.