UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GABRIEL CARVALHO DO ESPÍRITO SANTO

**PROBLEMA DAS N-CIFRAS: SIMULAÇÃO DE QUEBRA DE SENHAS COM O PROCESSO SEQUECIAL, PARALELO E DISTRIBUÍDO**

CORNÉLIO PROCÓPIO

2025

**GABRIEL CARVALHO DO ESPIRITO SANTO**

1. Introdução

O problema da N-Cifras, é um problema clássico na área da computação, que envolve os conceitos de criptografia e força bruta. O objetivo desse problema é encontrar a senha testando todas as possíveis combinações.

Muito utilizado nos estudos de algoritmos paralelos e distribuídos, o desafio contribui ao comparar o tempo de resolução das simulações.

Este trabalho tem como objetivo implementar e comparar o resultado de três abordagens para resolver o problema das N-Cifras:

* **Sequencial:** solução que executa todo o processo em um único fluxo de execução;
* **Paralela**: utiliza múltiplas *threads* para dividir o trabalho, explorando o poder de processamento de múltiplos núcleos;
* **Distribuída:** reparte o processamento entre diferentes máquinas ou processos, utilizando comunicação via *Sockets* ou RMI.

Assim, ao fim da implementação, será possível analisar os resultados e compreender as características das versões construídas

1. Desenvolvimento

Parte principal do trabalho, que contém a exposição ordenada e pormenorizada do assunto. É composta de revisão de literatura, dividida em seções e subseções, material e métodos e/ou metodologia e resultados, agora descritos detalhadamente. Cada seção ou subseção deverá ter um título apropriado ao conteúdo.

Deve-se utilizar sempre a terceira pessoa do singular na elaboração do texto, mantendo-se a forma impessoal.

## Solução Paralela

Durante a execução do algoritmo de força bruta para resolução do problema da N-Cifras, inicialmente é informado o número de dígitos que compõem a senha alvo. Com base nessa informação, calcula-se o total de combinações possíveis, considerando que cada dígito pode assumir valores de 0 a 9. Esse total é obtido por meio da exponenciação da base 10 pelo número de dígitos da senha, ou seja, 10^n, onde n representa a quantidade de posições.

A partir desse valor, o algoritmo inicia um processo iterativo que percorre, incrementalmente, todas as combinações possíveis de senhas, iniciando do valor 0 até o limite superior estabelecido. Em cada iteração, o número corrente é formatado como uma *string* com preenchimento de zeros à esquerda, de modo a manter o comprimento compatível com o da senha alvo. A cada nova tentativa, a *string* gerada é comparada com a senha correta, encerrando-se o processo assim que uma correspondência for encontrada.

Assim, já possível presumir que quanto mais próxima de zero, mais rápida a solução será encontrada. O que mostra um considerável desafio quando iniciar senhas com quantidade de caracteres maiores e com início no número 9.

## Solução Sequencial

Para a resolução do problema por meio de força bruta, foi implementada uma abordagem utilizando o módulo *threading* da linguagem Python. Nessa estratégia, o espaço total de combinações possíveis 10^N, sendo N o número de dígitos da senha) foi dividido igualmente entre a quantidade de threads definidas, de forma análoga à solução paralela convencional. Neste trabalho, optou-se pela utilização de quatro*threads*, resultando em uma divisão de 10^N/4 combinações por thread.

Durante a execução, cada thread opera de forma independente, percorrendo seu intervalo de busca sem a necessidade de sincronização contínua com as demais. Essa independência reduz significativamente o overhead associado a bloqueios e comunicação entre threads. O algoritmo é encerrado assim que um dos threads encontra a senha correta, momento em que o sistema finaliza a execução e exibe a senha descoberta juntamente com o tempo total de processamento.

O método demonstrou resultados mais caros em senhas menores quando comparado a solução paralela, mas rapidamente superou em velocidade o seu antecessor, quando simuladas senhas maiores.

## Solução Distribuída

Nesta solução, a parte lógica é semelhante à solução sequencial, mas com o diferencial de que a carga de trabalho é dividida entre múltiplas máquinas (ou nós), em vez de apenas múltiplas threads em uma única máquina.

Na abordagem distribuída, o espaço total de busca 10^N, onde N representa a quantidade de dígitos da senha, é dividido entre diversos nós trabalhadores (*workers*) que operam em máquinas distintas ou em processos isolados. Um nó coordenador (mestre) é responsável por orquestrar a execução da tarefa, atribuindo a cada *worker* um intervalo específico de combinações a ser verificado.

Cada *worker* realiza sua busca local de forma autônoma, utilizando um loop de força bruta dentro de seu intervalo designado. Essa descentralização permite que a carga de trabalho seja distribuída horizontalmente, aumentando a escalabilidade da solução e reduzindo o tempo total de execução, especialmente em ambientes com múltiplos recursos computacionais disponíveis.

Durante a execução, os *workers* não necessitam de comunicação constante entre si, o que minimiza o overhead de rede. O nó mestre pode periodicamente monitorar o progresso ou simplesmente aguardar que um dos *workers* retorne com a combinação correta. Assim que a senha é encontrada por qualquer *worker*, este envia a resposta ao coordenador, que então encerra o processamento nos demais nós e registra o tempo total de execução.

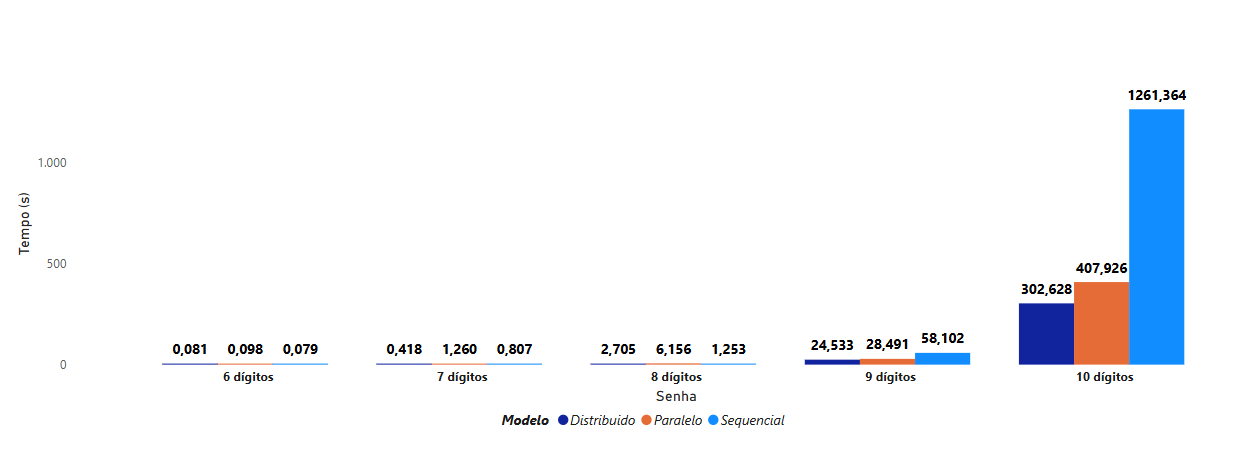
De forma geral, essa solução demonstrou equilíbrio em seus testes primários, mas evolui se distanciando positivamente cada vez mais rápido das soluções anteriores. Um dos complicadores foi a adição de parâmetros para retornar resposta durante a execução, visto o custo temporal em senhas maiores.

1. RESULTADOS

Com o intuito de executar as soluções de forma diversa, foram geradas 5 senhas numéricas aleatórias de tamanhos diferentes, iniciando com o tamanho de número 6. Cada senha foi inserida em cada solução e registrado o tempo para que fosse encontrada. A tabela abaixo mostra as senhas e os tempos (em segundos) de cada solução:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Senha** | **Dígitos** | **Sequencial** | **Paralela** | **Distribuída** |
| 928471 | *6 Dígitos* | 0.079004 | 0,097676 | 0.080595 |
| 7391284 | *7 Dígitos* | 0.807013 | 1,260402 | 0.418480 |
| 10394827 | *8 Dígitos* | 1,253184 | 6,156008 | 2,705106 |
| 547293106 | *9 Dígitos* | 58,101577 | 28,491373 | 24,533192 |
| 8192047356 | *10 Dígitos* | 1.261,363949 | 407,926086 | 302,627896 |

Para uma melhor visualização na comparação dos resultados obtidos, a figura abaixo ilustra com um gráfico de colunas as três soluções utilizadas:



1. CONCLUSÃO

Com os experimentos realizados neste trabalho, foi possível observar que a solução sequencial apresentou melhor desempenho em termos de tempo de execução para senhas com um número reduzido de dígitos. Isso se deve à sua simplicidade e à ausência de sobrecarga com criação de threads ou comunicação entre processos.

Por outro lado, a solução distribuída, embora apresentasse um desempenho competitivo já nos testes iniciais, manteve-se mais estável e eficiente à medida que o número de dígitos da senha aumentava, demonstrando maior escalabilidade. Essa característica a torna particularmente vantajosa em cenários com grande espaço de busca, onde a distribuição de carga entre diferentes nós compensa o custo de comunicação.

A solução paralela, por sua vez, apresentou resultados intermediários entre as abordagens sequencial e distribuída. Embora seja mais eficiente que a sequencial em entradas maiores, ainda está limitada ao ambiente de execução de uma única máquina, não alcançando o mesmo nível de desempenho da solução distribuída em casos de maior complexidade.

1. REFERÊNCIAS

**ATAQUE DE FORÇA BRUTA**. Em: *Wikipedia, a enciclopédia livre*. Disponível em: [https://pt.wikipedia.org/wiki/Ataque\_de\_força\_bruta](https://pt.wikipedia.org/wiki/Ataque_de_for%C3%A7a_bruta). Acesso em: 03 jul. 2025.

**OPENAI**. *ChatGPT (versão GPT-4o)*. Disponível em: <https://chat.openai.com>. Acesso em: 03 jul. 2025.