Trabalho de Programação Paralela.
Paralelizando o algoritmo de busca do problema das n-rainhas utilizando o MPI.

Alunos: Lucas Nardon, Willian Albeche.

1 - Introdução

O problema das N-rainhas é um problema clássico de busca onde o objetivo é posicionar N rainhas em um tabuleiro de xadrez de MxM de forma que nenhuma rainha possa atacar outra, ou seja, nenhuma rainha pode estar na mesma linha, coluna ou diagonal de outra rainha.

2 - Objetivo

O objetivo deste código é resolver o problema das N-Rainhas usando o algoritmo de força bruta distribuída em um cluster MPI.

O programa é projetado para ser executado em um cluster de computadores, onde cada computador (ou processo) desempenha um papel na busca por soluções válidas. O programa usa a biblioteca MPI para permitir a comunicação e coordenação entre os processos.

3 - Implementação

A ideia é dividir o espaço de busca em intervalos menores e atribuir a cada processo uma parte desses intervalos para realizar a busca. Cada processo executa a função brute_force, que itera por todas as permutações possíveis de rainhas em seu intervalo e verifica se cada permutação é uma solução válida usando a função is_solution. O contador solutions_count é atualizado a cada vez que uma solução válida é encontrada.

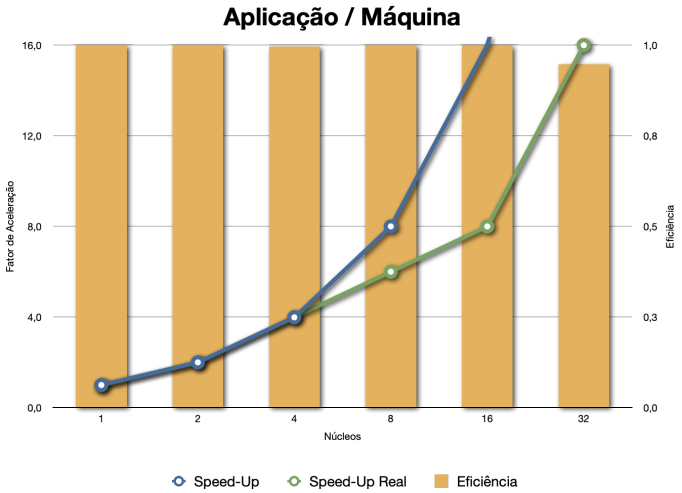
A implementação aproveita o ambiente MPI para coordenação e comunicação entre os processos a partir de uma divisão de trabalho mestre-escravo entre os processos, onde o mestre apenas fica responsável por coordenar e agregar as soluções encontradas pelos processos escravos que realizam a parte de busca de soluções, possibilitando uma execução mais rápida ao utilizar o poder de processamento paralelo e distribuído do cluster.

No final, o programa exibe o número total de soluções encontradas e o tempo decorrido para executar a tarefa. O código desenvolvido encontra-se no repositório do Github, acessado através do seguinte link: https://github.com/lnardon/ParallelComputing

4 - Resultados

Com base nos resultados obtidos, é possível notar que o tempo de execução diminui à medida que o número de threads aumenta. Isso ocorre porque, ao utilizar mais threads, o trabalho é dividido entre elas, permitindo a execução simultânea de diferentes partes do espaço de busca. Esse paralelismo resulta em um tempo total menor necessário para encontrar todas as soluções.

Vale ressaltar que no gráfico foram extrapolados os valores para 1 ou 2 processos, pois o tempo estimado para a execução do algoritmo com 4 processos já alcançava a casa dos 40 minutos, tornando inviável a execução com menos processos.



Ao distribuir o trabalho entre os processos em um cluster, o MPI permitiu explorar o poder de processamento paralelo, resultando em tempos de execução menores à medida que o número de nodos ou threads aumentou. Além disso, o MPI facilitou a comunicação entre os processos, permitindo a transmissão de dados e a combinação dos resultados de forma coordenada.

Contabilizando um ganho entre 4 a 32 processos, de 8x, mostrando como o algoritmo se beneficiou com o aumento do paralelismo.