UNIVERSIDADE FEDERAL DE PELOTAS

REDES

CIÊNCIA DA COMPUTAÇÃO

TRABALHO PRÁTICO 1

Problema das N Rainhas utilizando Openmp em Multicomputadores

Heliks Mersenburg

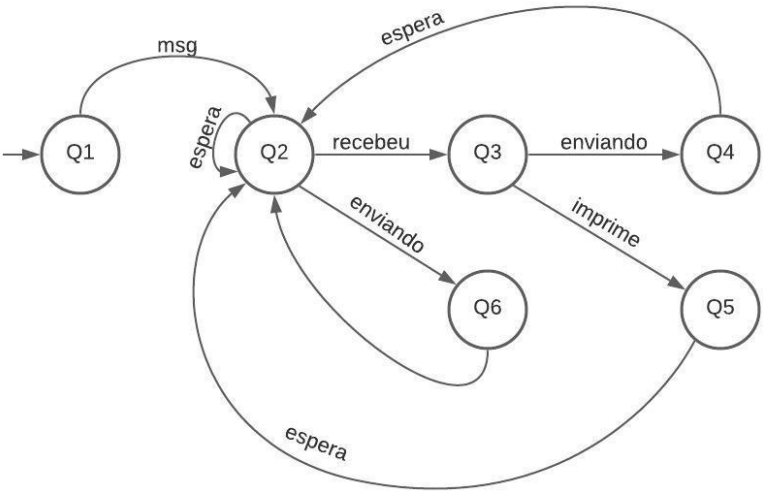
Pelotas, 24 de maio de 2021.

1. **Objetivos / Informações relevantes:** Meu objetivo é simular paralelismo em rede utilizando o algoritmo N-Queen e protocolo TCP.
    - ***a.*** *Problema N-Queen*: É o problema de colocar N rainhas de xadrez em um tabuleiro N × N de modo que duas rainhas não se ameacem; assim, uma solução requer que duas rainhas não compartilhem a mesma linha, coluna ou diagonal.
    - ***b.*** *Algoritmo N-Queen implementado*: Para implementar o algoritmo N-Queen, utilizei a linguagem C juntamente com OpenMP para explorar o paralelismo nas máquinas.
    - ***c.*** *Observações sob o algoritmo N-Queen*: Para executar o algoritmo é necessário passar como argumento o valor de N e o intervalo que será computado. O retorno dessa função é o número de soluções encontradas para aquele intervalo. O algoritmo é dividido em 4 intervalos, onde cada máquina tem o dever de executar seu intervalo proposto.
    - ***d.*** *Observações sob o algoritmo TCP*: As máquinas não possuem objetivo de compartilhar memória, cada máquina deve computar o intervalo que foi recebido e entregar ao servidor sua ordem. Seguindo um padrão de mestre/escravo.

2. **Características do protocolo:**

    O protocolo possui uma abordagem ***cliente-servidor*** onde o primeiro cliente é definido como Mestre e ele e somente ele pode fazer pedidos ao servidor para executar o algoritmo. Ao receber a mensagem informando que deve ser computado um valor de N, o servidor ordena aos escravos e ao mestre que comecem a executar o algoritmo (Diversos computadores podem estar conectados como cliente, mas apenas 4 executam o algoritmo N-queen). A conexão é ***persistente***, pois a cada pedido o usuário pode fazer uma nova requisição e os computadores ficam sempre conectados aguardando novas tarefas. O protocolo possui ***estado***, é armazenado uma informação logo que é iniciado informando quem é o cliente mestre. De modo geral, o usuário mestre faz um pedido ao servidor, o servidor envia as ordens aos clientes (incluindo ao mestre) e aguarda receber as respostas dos clientes para enviar o somatório total de soluções ao cliente mestre que havia lhe solicitado o pedido. Ao fechar o servidor, todos clientes são fechados.

3. **Máquina de estados Cliente**
    - ***a. Cliente - Mestre*:**



Estados:

Q1: O cliente é iniciado

Q1 -> Q2: Ao ser iniciado o cliente espera uma mensagem do servidor definindo quem é o mestre;

Q2: Estado de espera até o cliente ou o servidor tomar uma ação;

Q2 -> Q3: O cliente recebeu mensagem do servidor;

Q3 -> Q4: O cliente envia a resposta ao servidor;

Q3 -> Q5: O cliente imprime o resultado;

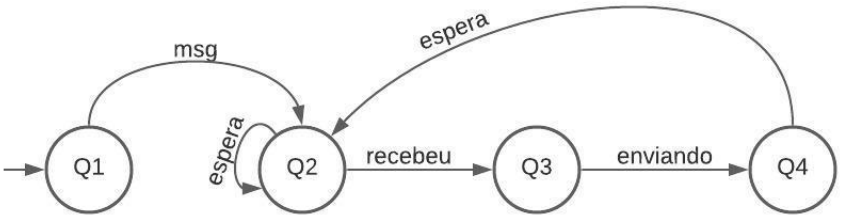
Q4 -> Q2: Volta para o estado de espera;

Q5 -> Q2: Volta ao estado de espera;

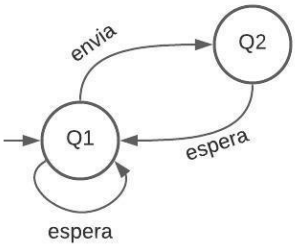
Q2 -> Q6: O cliente envia uma mensagem ao servidor requisitando cálculo;

Q6 -> Q2: Volta ao estado de espera.

b. Cliente - Escravo: *Consiste nos estados básicos do cliente-mestre, mas sem o estado de enviar pedido ao servidor, nem de imprimir o resultado.*



4. **Máquina de estados Servidor:** As máquinas de estados foram divididas em 2 figuras, a primeira se refere ao processo servidor-pai, onde é iniciado novos clientes e armazenado em uma lista de clientes e a outra se refere a thread do processo servidor-cliente-mestre.
    a. **Processo-Pai**:

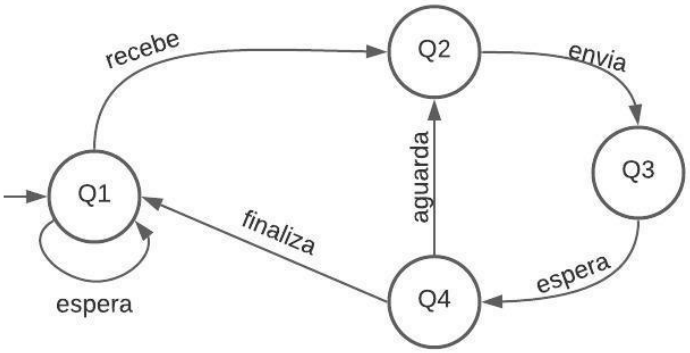


Q1: Estado de espera até que um novo cliente seja iniciado.

Q1 -> Q2 : O servidor envia ao cliente a informação de quem é o mestre (Refere-se ao estado Q1->Q2 da máquina cliente);

Q2 -> Q1 : O servidor volta ao estado 1.

b. Thread Servidor-Mestre:



Q1: O servidor aguarda uma mensagem;

Q1 -> Q2: O servidor recebe uma mensagem;

Q2 -> Q3: O servidor envia uma mensagem para cada cliente ativo;

Q3 -> Q4: O servidor espera a resposta dos clientes;

Q4 -> Q2: Caso não hajam 4 computadores conectados, volta ao estado 2 para executar as ações restantes até que as 4 soluções sejam encontradas;

Q4 -> Q1: Envia o resultado ao Cliente-Mestre e volta ao estado 1.

Obs.: *Existem estados que não foram acrescentados: O servidor verifica se os clientes estão conectados, caso eles não estejam, remove eles da lista de threads existentes.*

5. **Listagem de mensagens cliente - servidor:**

1. O cliente envia uma mensagem contendo o valor de N para o servidor, esse valor deve ser maior que cinco e é um número inteiro.

2. O cliente envia uma mensagem após o cálculo N-Queen contendo a quantidade de soluções naquele intervalo. Este valor é um número inteiro.

6. **listagem de mensagens servidor - cliente:**

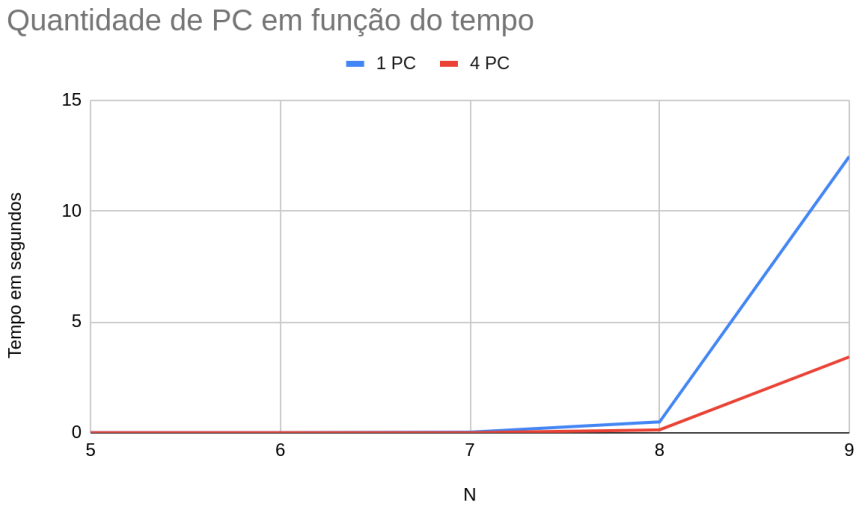
1. O servidor envia uma mensagem contendo o numero do computador (A informação serve para identificar quem é mestre e quem é servo)

2. O servidor envia uma mensagem para cada cliente contendo duas informações, a primeira é o intervalo que deve ser calculado e a segunda é o valor N.

3. O servidor envia para o cliente mestre o resultado encontrado. Esta mensagem contém 3 informações: o último intervalo calculado, o valor de N e a solução total.

7. **Considerações finais:**

O gráfico a seguir compara o desempenho do algoritmo N-Queen executando em um computador sem protocolo e em 4 computadores com o protocolo implementado.



Ao executar a proposta percebi que minha abordagem deveria ter sido peer-to-peer e sem a utilização de um cliente-mestre, certamente haveria um ganho de desempenho se o cliente não precisasse se comunicar com um servidor e suas ordens passassem direto para os clientes conectados a rede.

Utilizei a linguagem Python para implementação do cliente-servidor e C junto com a API Openmp para implementar o N-Queen.

8. Instruções de uso:

Execute o servidor digitando no terminal:
python serverTCP.py

Execute o cliente digitando no terminal:
python clientTCP.py

O **primeiro** cliente executado é o **mestre**, insira um valor para N (valores acima de 10 podem levar horas para serem executados)