Trabalho Final de Sistemas Distribuídos

Fernanda Takao

2025-1

Introdução

A crescente demanda por aplicações mais rápidas e eficientes impulsionou o avanço da programação paralela e distribuída, cujo objetivo é dividir o processamento de tarefas complexas entre múltiplos núcleos, threads ou até mesmo máquinas, de forma a reduzir o tempo de execução e aumentar o desempenho. Em um cenário onde o volume de dados e a complexidade dos algarismos são cada vez maiores, essa estrutura se faz necessária e proveitosa.

Na programação paralela, uma série de instruções são executadas simultaneamente em diferentes núcleos de um mesmo processador; dessa forma, partes independentes de um algoritmo são processadas em paralelo, aumentando a eficiência. Na programação distribuída, por sua vez, o processamento é repartido entre diferentes computadores conectados em rede, ampliando o poder computacional, embora exija mecanismos de comunicação e sincronização entre os nós.

Ambas as técnicas marcam presença expressiva em áreas como inteligência artificial, simulações científicas, jogos em tempo real e sistemas embarcados. Neste trabalho, utilizou-se esses dois paradigmas — paralelismo via threads e distribuição via sockets — para resolver um problema clássico da ciência da computação: o problema do Caixeiro Viajante (TSP), cuja complexidade é exponencial e se beneficia fortemente dessas abordagens.

Apresentação do Problema

O problema do Caixeiro Viajante (Traveling Salesman Problem – TSP) consiste em encontrar o caminho de menor custo que percorra todas as cidades de um mapa, partindo de uma cidade inicial, visitando todas as demais uma única vez e retornando ao ponto de origem. Trata-se de um problema de otimização que pertence à classe NP-difícil, o que significa que sua resolução por métodos exatos torna-se inviável para instâncias maiores.

Sua importância é notável em diversas áreas como logística, roteamento de veículos, organização de circuitos eletrônicos e planejamento de rotas. Apesar de ser simples de descrever, a quantidade de caminhos possíveis se multiplica com o número de cidades, criando assim um cenário favorável para testar algoritmos de paralelismo e distribuição.

Considerando um grafo com oito cidades conectadas por arestas com pesos que representam as distâncias, cada solução possível seria um caminho fechado que passa uma única vez por todas as cidades. A resolução desse problema com base nesse exemplo já envolveria 5040 permutações (7!); com 10 cidades, ultrapassaria 360 mil caminhos possíveis.

Proposta de Resolução

Neste trabalho, desenvolveu-se três versões distintas para resolver o TSP:

- **Versão Sequencial**: percorre todas as permutações possíveis de caminhos de forma linear. Utiliza força bruta para testar cada permutação e calcular seu custo.

- **Versão Paralela**: divide a lista de permutações em dois subconjuntos e os processa simultaneamente com oito threads. Ao final, compara os resultados parciais para determinar o melhor caminho.

- **Versão Distribuída**: executa a lógica em um servidor que divide a lista de permutações em dois grupos e os envia para dois clientes via sockets. Cada cliente calcula o menor caminho local e retorna a resposta ao servidor.

A classe *TSPLogica.java* é utilizada pelas três versões e contém duas funções principais: *gerarPermutacoes*, que gera todas as rotas possíveis, e *calcularDistancia*, que retorna o custo total de um caminho. O código completo pode ser consultado no seguinte repositório:

Resultados

Os testes foram realizados em uma máquina com processador Intel Core i5, 8 GB de RAM, com sistema Windows 10. Abaixo, uma tabela com os tempos de execução médios para o problema com 8 cidades:

Versão	Tempo médio (ms)	Observações
Sequencial	7	Custo computacional elevado
Paralela	3	Uso de 2 threads
Distribuída	20000	Comunicação via sockets com 2 clientes

O desempenho da versão paralela foi superior ao da sequencial, reduzindo o tempo em até 57,14%. A versão distribuída, por ter sido simulada em uma única máquina, apresentou um tempo real muito superior às outras devido à sobrecarga de serialização de objetos grandes, estabelecimento de conexões TCP e tempo de espera para a resposta de cada cliente. No entanto, em condições ideais — execução em máquinas distintas com comunicação otimizada —, é de se esperar que esse tempo seja

significativamente menor, superando a versão sequencial e aproximando-se da eficiência da versão paralela.

Conclusão

O trabalho demonstrou a efetividade da programação paralela na resolução de problemas de alta complexidade computacional, tais qual o TSP. A versão sequencial, embora simples, mostrou-se ineficiente em termos de tempo, ao passo que a paralela foi significativamente mais rápida e fácil de implementar. Já a versão distribuída, mais complexa, permitiu a execução do cálculo em múltiplos processos; seria mais escalável para um contexto onde houvesse mais de uma máquina trabalhando ao mesmo tempo.

Referências

1 c0rdurb0y, 2024
https://www.reddit.com/r/algorithms/comments/1bckkm9/discussion_of_traveling_salesman_problem/

2 Pozza, Rogério, 2025. Moodle da UTFPR de Cornélio Procópio.

3 Takao, Fernanda. Github.