UNIVERSIDADE CATÓLICA DE BRASÍLIA

Gabriel Eduardo Barbosa da Silva

Victor Inácio Ferreira Soares

AT2/N1 - RELATÓRIO ATIVIDADE PRÁTICA COLETIVA

BRASÍLIA-DF

2024

O QUE SÃO THREADS?

Threads são unidades básicas de execução dentro de um processo. Elas são frequentemente chamadas de “subprocessos” porque operam dentro do contexto de um processo maior. Cada thread em um processo compartilha o mesmo espaço de memória e recursos, mas pode executar tarefas de forma independente.

COMO THREADS FUNCIONAM COMPUTACIONALMENTE?

Threads são criadas por um processo principal. Cada uma possui seu próprio contexto de execução, que inclui um conjunto de registros, uma pilha para armazenar variáveis locais e o contador de programa que aponta para a próxima instrução a ser executada. O sistema operacional possui um escalonador que decide qual thread deve ser executada em um dado momento. Quando o escalonador decide mudar a execução de uma thread para outra, ele realiza uma troca de contexto, salvando o estado da thread atual e carregando o estado da próxima thread a ser executada.

COMO O USO DE THREADS PODE AFETAR O TEMPO DE EXECUÇÃO DE UM ALGORITMO?

Threads permitem que diferentes partes de um algoritmo sejam executadas simultaneamente em múltiplos núcleos de CPU. Isso pode reduzir drasticamente o tempo de execução, especialmente para tarefas que podem ser divididas em sub-tarefas independentes.

A RELAÇÃO ENTRE OS MODELOS DE COMPUTAÇÃO CONCORRENTE E PARALELO E A PERFORMANCE DOS ALGORITMOS

A computação concorrente envolve a execução de múltiplas tarefas que podem ser intercaladas no tempo, mas não necessariamente ao mesmo tempo. A concorrência melhora a responsividade e a eficiência dos sistemas, especialmente em ambientes onde múltiplas tarefas precisam ser gerenciadas simultaneamente. No entanto, a sobrecarga de gerenciamento de tarefas concorrentes pode introduzir latência e complexidade.

A computação paralela envolve a execução simultânea de múltiplas tarefas em diferentes processadores ou núcleos. É focada em dividir uma tarefa grande em partes menores que podem ser executadas ao mesmo tempo. O paralelismo pode levar a ganhos significativos de performance, especialmente em algoritmos que podem ser facilmente divididos em sub-tarefas independentes. No entanto, a eficiência do paralelismo depende da capacidade de dividir a tarefa e da comunicação entre as sub-tarefas.

EXPERIMENTO

O experimento consiste em um projeto em Java que lê 320 arquivos 320 contendo temperaturas diárias de 320 cidades de diversos países. Após a leitura de cada arquivo, o programa deve calcular as temperaturas média, máxima e mínima por mês, para cada cidade. Cada versão do experimento consiste na execução de 10 repetições/rodadas de um algoritmo composto por:

1. Leitura dos dados de cada cidade;
2. Processamento do dado recebido, cálculo das temperaturas média, mínima e máxima por cidade por mês;
3. Armazenamento dos dados gerado na memória;
4. Exibição dos dados gerados no console.

O objetivo do experimento é computar o tempo de execução de cada rodada e, ao fim das 10 rodadas, calcular o tempo médio para execução(soma dos tempos de cada rodada dividido por 10). No gráfico abaixo estão os resultados do experimento:

O gráfico demonstra que o uso de threads melhora a eficiência do algoritmo. Porém, o uso excessivo de threads causa uma sobrecarga no gerenciamento dessas tarefas que resulta em

uma lentidão e aumento do tempo necessário para a sua execução. Ou seja, usar muitas threads para tarefas simples não tem um bom custo-benefício, já que o sistema passa mais tempo gerenciando as threads do que executando a tarefa em si. Esse gerenciamento inclui alocar recursos para cada thread, sincronizar as threads para evitar que elas tentem acessar o mesmo recurso simultaneamente, e a troca de contexto entre threads, que envolve interromper uma thread para permitir que outra tenha acesso à CPU.

REFERÊNCIAS BIBLIOGRÁFICAS

DevMedia. “Programação com Threads”. Disponível em: https://www.devmedia.com.br/programacao-com-threads/6152

Tecnoblog. "O que são threads do processador e quais os benefícios do multithreading?". Disponível em: https://tecnoblog.net/responde/o-que-e-thread-processador/

GeeksforGeeks. “Thread in Operating System”. Disponível em: https://www.geeksforgeeks.org/thread-in-operating-system/

Wikipedia. “Thread (computing)”. Disponível em: https://en.wikipedia.org/wiki/Thread_(computing)

Stack Overflow. “What is a thread, really?”. Disponível em: https://stackoverflow.com/questions/5201852/what-is-a-thread-really

Dev.to. “Processos, Paralelismo e Concorrência”. Disponível em: https://dev.to/starch1/processos-paralelismo-e-concorrencia-iom

Dev.to. “O que é concorrência em um sistema operacional?”. Disponível em: https://dev.to/nfo94/o-que-e-concorrencia-em-um-sistema-operacional-1pj1

Wikipedia. “Programação concorrente”. Disponível em: https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_concorrente