1. Introdução Teórica

O que são threads

Threads são a menor unidade de processamento que pode ser gerenciada de maneira independente pelo sistema operacional. Elas são partes de um processo maior, permitindo que diversas operações sejam realizadas simultaneamente dentro de um mesmo programa. Threads são úteis para dividir tarefas complexas em partes menores que podem ser executadas em paralelo, aumentando a eficiência e a performance do software.

Como threads funcionam computacionalmente

Computacionalmente, threads compartilham o mesmo espaço de memória dentro de um processo, mas operam de maneira independente. Cada thread tem seu próprio conjunto de registradores, contador de programa e pilha. O sistema operacional gerencia a execução das threads utilizando um escalonador, que determina qual thread deve ser executada em um dado momento, permitindo a execução concorrente de múltiplas threads em sistemas com múltiplos núcleos de processamento.

Como o uso de threads pode afetar o tempo de execução de um algoritmo

O uso de threads pode reduzir significativamente o tempo de execução de um algoritmo ao permitir que várias operações sejam realizadas simultaneamente. Quando um algoritmo é dividido em várias threads, cada thread pode ser executada em paralelo em diferentes núcleos de um processador multi-core. Isso pode levar a uma redução drástica no tempo total de execução, especialmente para tarefas que são computacionalmente intensivas ou que envolvem operações de I/O.

Relação entre os modelos de computação concorrente e paralelo e a performance dos algoritmos

A computação concorrente e a computação paralela são modelos que permitem a execução simultânea de múltiplas operações. Na computação concorrente, múltiplas threads são gerenciadas pelo escalonador do sistema operacional e podem não ser executadas ao mesmo tempo, mas sim de maneira intercalada. Já na computação paralela, múltiplas threads ou processos são executados simultaneamente em diferentes núcleos ou máquinas. Ambos os modelos podem melhorar a performance dos algoritmos, mas a computação paralela tende a ser mais eficiente em termos de redução do tempo de execução em sistemas com múltiplos núcleos de processamento.

Citação de referência bibliográfica: "Operating System Concepts" by Abraham Silberschatz, Greg Gagne, and Peter B. Galvin.

2. Exibição e Explicação dos Resultados Obtidos

Neste projeto, foi utilizada a API da Open-Meteo para recuperar as temperaturas a cada 2 horas, durante o período de 1 de janeiro de 2024 a 31 de janeiro de 2024, de todas as capitais do Brasil. O objetivo principal foi avaliar o impacto do uso de threads na performance do algoritmo responsável por essas requisições. Para isso, um experimento foi conduzido no qual um conjunto de operações foi executado em quatro versões distintas:

1. Sem o uso de threads (apenas a thread principal);
2. Utilizando 3 threads;
3. Utilizando 9 threads;
4. Utilizando 27 threads.

A escolha de diferentes quantidades de threads visa compreender como a paralelização das requisições pode influenciar o desempenho e a eficiência do processo. O uso de múltiplas threads é uma prática comum em computação para otimizar tarefas que envolvem I/O intensivo, como no caso das requisições de dados via API. No entanto, a implementação de multi-threading deve ser cuidadosamente manejada para evitar problemas como concorrência de threads e sobrecarga do sistema.

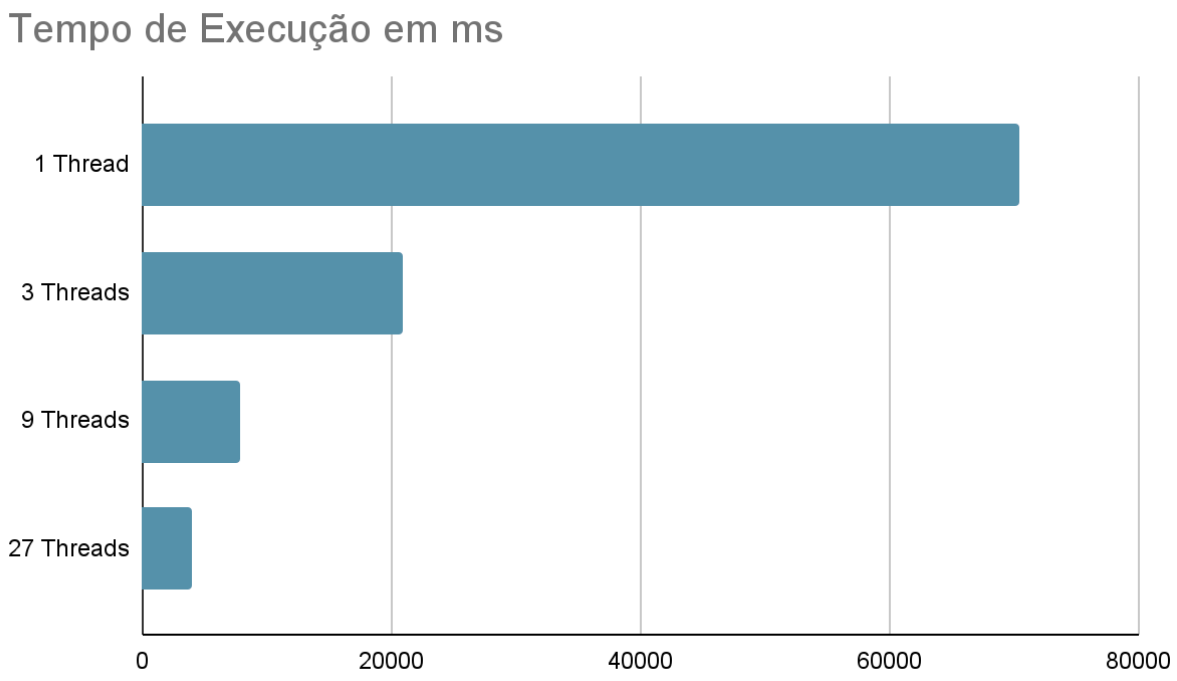
Devido à grande quantidade de requisições efetuadas, foi necessário espaçar a execução das diferentes versões para evitar o retorno do código de erro 429 (Too Many Requests), que ocorre quando um limite de requisições à API é excedido em um curto período de tempo. Este espaçamento entre as execuções garantiu que cada versão pudesse ser testada de forma justa, sem interferências externas causadas por limitações da API.

Após a execução das quatro versões mencionadas, os resultados obtidos foram os seguintes:

Resultados

Versão do experimento	Tempo médio de execução (ms)
Uma Thread	70499
Três Threads	20894
Nove Threads	7957
Vinte sete Threads	4051

Gráfico de Comparação



Esses resultados fornecem uma análise comparativa entre as diferentes abordagens de multi-threading e a execução sequencial. A partir dessa análise, pode-se observar as vantagens e desvantagens de cada método em termos de tempo de execução, uso de recursos do sistema e capacidade de lidar com grandes volumes de dados de maneira eficiente.

Este estudo é relevante para entender as implicações práticas do uso de threads em aplicações reais que dependem de múltiplas requisições simultâneas a serviços externos. Além disso, oferece insights valiosos para a otimização de algoritmos em contextos onde o desempenho é crítico.

Explicação dos Resultados

Observamos que o uso de threads teve um impacto significativo na redução do tempo de execução. A versão sem threads teve o maior tempo médio de execução, enquanto as versões com threads apresentaram uma redução progressiva no tempo de execução conforme aumentamos o número de threads. Isso demonstra a eficiência do uso de threads para paralelizar tarefas e reduzir o tempo de processamento em algoritmos intensivos. No entanto, também é importante notar que o aumento no número de threads não resulta em uma redução linear do tempo de execução, devido a fatores como sobrecarga de gerenciamento de threads e limitações no número de núcleos de processamento disponíveis.

---

3. Referências Bibliográficas

- Silberschatz, A., Gagne, G., & Galvin, P. B. (2020). *Operating System Concepts* (10th ed.). Wiley.