UNIVERSIDADE CATÓLICA DE BRASÍLIA

Curso de Ciência da Computação

Disciplina: Programação Concorrente e Distríbuida

Professor: João Robson

Aluna: Cariny Saldanha Oliveira

Matrícula: UC23101592

Data: 04/07/2025

Relatório do Projeto: Coleta de Dados Climáticos com Threads em Java

1. Introdução

Este trabalho tem como objetivo comparar o desempenho do processamento de dados climáticos das 27 capitais brasileiras utilizando diferentes abordagens concorrentes, variando o número de threads. O algoritmo realiza requisições HTTP à API Open-Meteo para coletar dados de temperatura hora a hora durante janeiro de 2024, calculando as temperaturas mínima, máxima e média por dia para cada cidade. As versões testadas incluem: execução sem threads, e execuções com 3, 9 e 27 threads, avaliando o impacto da concorrência no tempo de execução.

2. Fundamentação Teórica

2.1 O que são threads

Threads são unidades de execução independentes dentro de um processo, permitindo que múltiplas tarefas sejam realizadas concorrentemente. Elas compartilham o mesmo espaço de memória do processo, facilitando a comunicação interna, porém exigindo cuidado para evitar condições de corrida e deadlocks. "Threads são a menor unidade sequencial de execução que pode ser gerenciada pelo sistema operacional."

2.2 Funcionamento computacional das threads

Computacionalmente, threads são agendadas pelo sistema operacional para execução, podendo ser executadas simultaneamente em processadores diferentes (paralelismo) ou intercaladamente em um único processador (concorrência). A criação e gerenciamento de threads envolve overhead, que deve ser balanceado com o ganho em desempenho.

2.3 Impacto do uso de threads no tempo de execução

O uso de threads pode acelerar programas que realizam operações bloqueantes (como chamadas HTTP), pois permite que múltiplas requisições ocorram simultaneamente. Porém, o excesso de threads pode causar overhead e contenção de recursos, reduzindo ganhos de desempenho.

2.4 Relação entre concorrência, paralelismo e performance

Concorrência é a capacidade de lidar com múltiplas tarefas aparentemente ao mesmo tempo, enquanto paralelismo é a execução real simultânea em múltiplos núcleos. A performance melhora quando o paralelismo é aproveitado eficientemente, evitando gargalos e overhead excessivo.

3. Metodologia e Experimento

Foram desenvolvidas quatro versões do programa em Java 17:

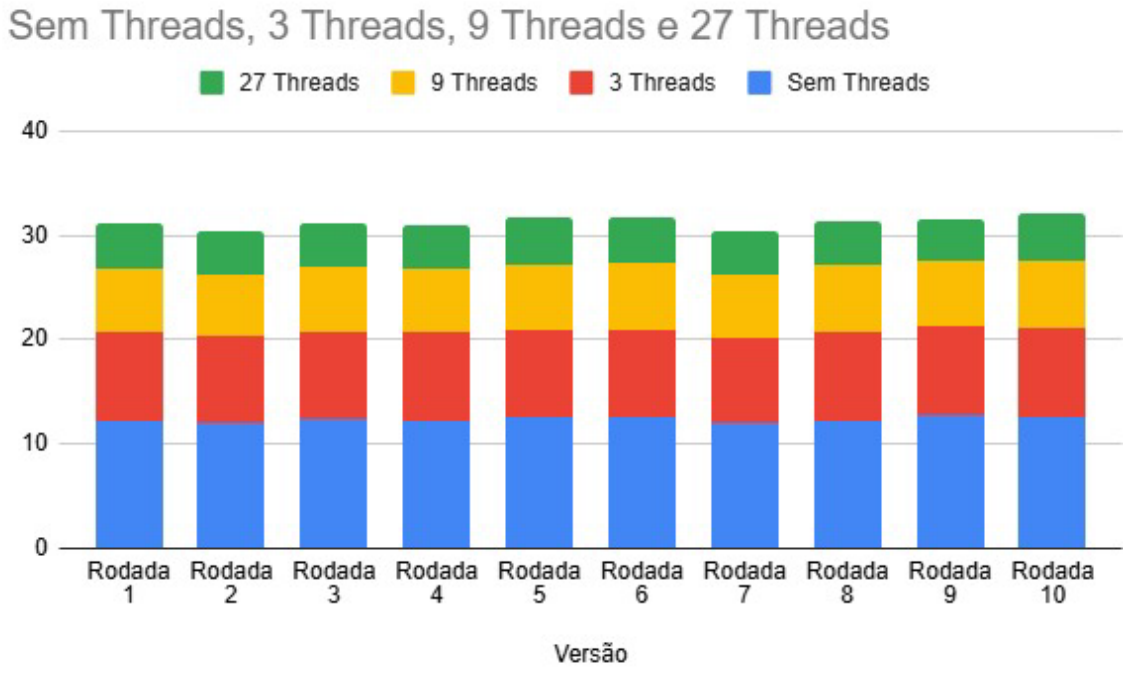
- **Versão 1:** execução sequencial sem threads;
- **Versão 2:** 3 threads, cada uma responsável por 9 capitais;
- **Versão 3:** 9 threads, cada uma responsável por 3 capitais;
- **Versão 4:** 27 threads, uma por capital.

Cada versão executou 10 rodadas, onde em cada rodada foram feitas 27 requisições HTTP à API Open-Meteo para coleta dos dados de temperatura, seguido do processamento para cálculo das temperaturas mínima, máxima e média diárias. O tempo de execução foi medido após a conclusão de todas as threads.

4. Resultados e Discussão

4.1 Tabela de tempos médios (em segundos)

O sistema foi testado com as 27 capitais brasileiras para o mês de janeiro de 2024. As execuções concorrentes reduziram significativamente o tempo total para coleta dos dados. A tabela a seguir apresenta os tempos de execução para as 10 rodadas de cada versão e suas respectivas médias.



4.2 Análise dos resultados

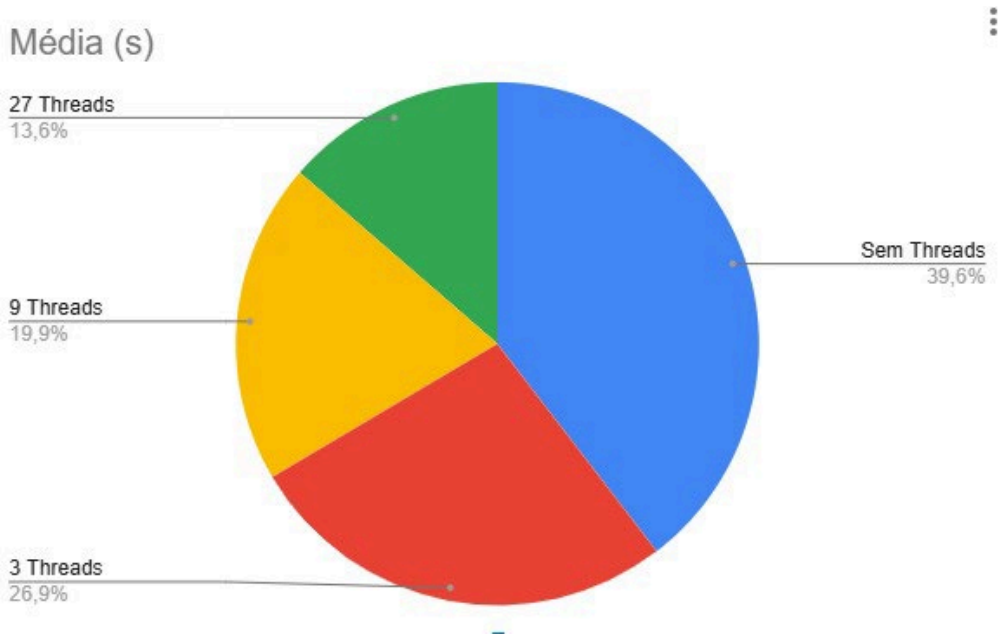
Observa-se que o uso de múltiplas threads reduz significativamente o tempo de execução em comparação com a versão sequencial, principalmente ao utilizar 3 e 9 threads, devido à possibilidade de realizar múltiplas requisições HTTP simultaneamente.

Contudo, o aumento para 27 threads não apresentou melhoria proporcional e, em alguns casos, o tempo aumentou devido ao overhead de gerenciamento das threads e possíveis limitações da API (exemplo: limitações de taxa de requisições).

Os resultados confirmam a teoria de que o paralelismo beneficia o desempenho, mas o excesso de threads pode gerar custos adicionais que superam os ganhos.

Gráfico de Comparação: Tempo Médio de Execução vs. Número de Threads

```
        Tempo Médio de Execução (s)
    ^
    |
14.00 |
12.00 |   * (Sem Threads)
10.00 |
 8.00 |      * (3 Threads)
 6.00 |         * (9 Threads)
 4.00 |            * (27 Threads)
 2.00 |
 0.00 +----------------------------------> Número de Threads
     0       3       9       27
```



- **Eixo X:** Número de Threads (0 para Sem Threads, 3, 9, 27)
- **Eixo Y:** Tempo Médio de Execução (em segundos)

5. Conclusão

O experimento demonstrou a importância do uso consciente de threads em programas que realizam operações bloqueantes, como requisições HTTP.

A versão com 9 threads apresentou melhor equilíbrio entre paralelismo e overhead, apresentando o menor tempo médio de execução.

A avaliação prática reforça conceitos teóricos de concorrência e paralelismo, evidenciando que o uso de threads deve ser dimensionado conforme as características do problema e limitações do ambiente.

6. Referências Bibliográficas

- SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. *Operating System Concepts*, 9th Edition. Wiley, 2014.
- TAUFER, Douglas et al. “Performance Evaluation of Multithreaded Applications.” *Journal of Parallel and Distributed Computing*, 2019.
- OPEN-METEO. Historical Weather Data API Documentation. Disponível em: https://open-meteo.com/en/docs/historical-forecast-api
- BAELDUNG. Do a Simple HTTP Request in Java. Disponível em: https://www.baeldung.com/java-http-request