Relatório sobre o Uso de Threads em Algoritmos para Coleta e Processamento de Dados Climáticos

Introdução Teórica

O que são Threads?

Threads são unidades básicas de execução de um processo, permitindo a realização de múltiplas tarefas simultaneamente dentro de um mesmo programa. Segundo Tanenbaum (2015), threads são "processos leves" que compartilham o mesmo espaço de memória, mas possuem sua própria pilha e registradores, facilitando a execução concorrente de tarefas. Este conceito é crucial em sistemas operacionais modernos, onde a capacidade de executar múltiplas threads pode melhorar significativamente a performance e a eficiência do sistema.

Funcionamento Computacional das Threads

Computacionalmente, threads são gerenciadas pelo sistema operacional, que aloca tempo de CPU e recursos necessários para a execução de cada thread. Threads podem ser implementadas em nível de usuário, onde a biblioteca de threads gerencia a troca de contexto, ou em nível de kernel, onde o sistema operacional é responsável por essa gestão. Em sistemas multiprocessados, threads podem ser executadas em paralelo em diferentes núcleos, enquanto em sistemas single-core, elas são executadas de maneira concorrente, alternando rapidamente entre si para dar a impressão de simultaneidade (Silberschatz et al., 2018).

Impacto do Uso de Threads no Tempo de Execução de Algoritmos

O uso de threads pode reduzir significativamente o tempo de execução de algoritmos, especialmente aqueles que envolvem operações de E/S ou tarefas independentes que podem ser executadas simultaneamente. Threads permitem que um programa continue executando outras tarefas enquanto aguarda a conclusão de uma operação de E/S, por exemplo. No entanto, a criação e a gestão de threads envolvem uma sobrecarga de processamento, e a comunicação e sincronização entre threads podem introduzir complexidade e possíveis condições de corrida (Amdahl, 1967).

Modelos de Computação Concorrente e Paralela e a Performance dos Algoritmos

A computação concorrente refere-se à execução de múltiplas tarefas que podem ser executadas de forma independente ou de maneira intercalada no tempo. A computação paralela, por outro lado, envolve a execução simultânea de múltiplas tarefas em diferentes núcleos ou processadores. O modelo de computação utilizado pode influenciar diretamente a performance dos algoritmos. Algoritmos projetados para execução paralela podem obter grandes melhorias de performance em sistemas multiprocessados, enquanto algoritmos concorrentes podem melhorar a responsividade e a eficiência em sistemas single-core. A Lei

de Amdahl destaca que o ganho de performance de um sistema paralelo é limitado pela fração do programa que não pode ser paralisada.

Análise Detalhada dos Resultados Obtidos

Introdução

O experimento realizado teve como objetivo analisar o impacto do uso de threads no tempo de execução de um algoritmo para coleta e processamento de dados climáticos das 27 capitais brasileiras durante o mês de janeiro de 2024. As quatro versões do experimento variaram no número de threads utilizadas, desde a versão de referência, sem threads adicionais, até a versão com 27 threads, onde cada thread era responsável por uma capital específica.

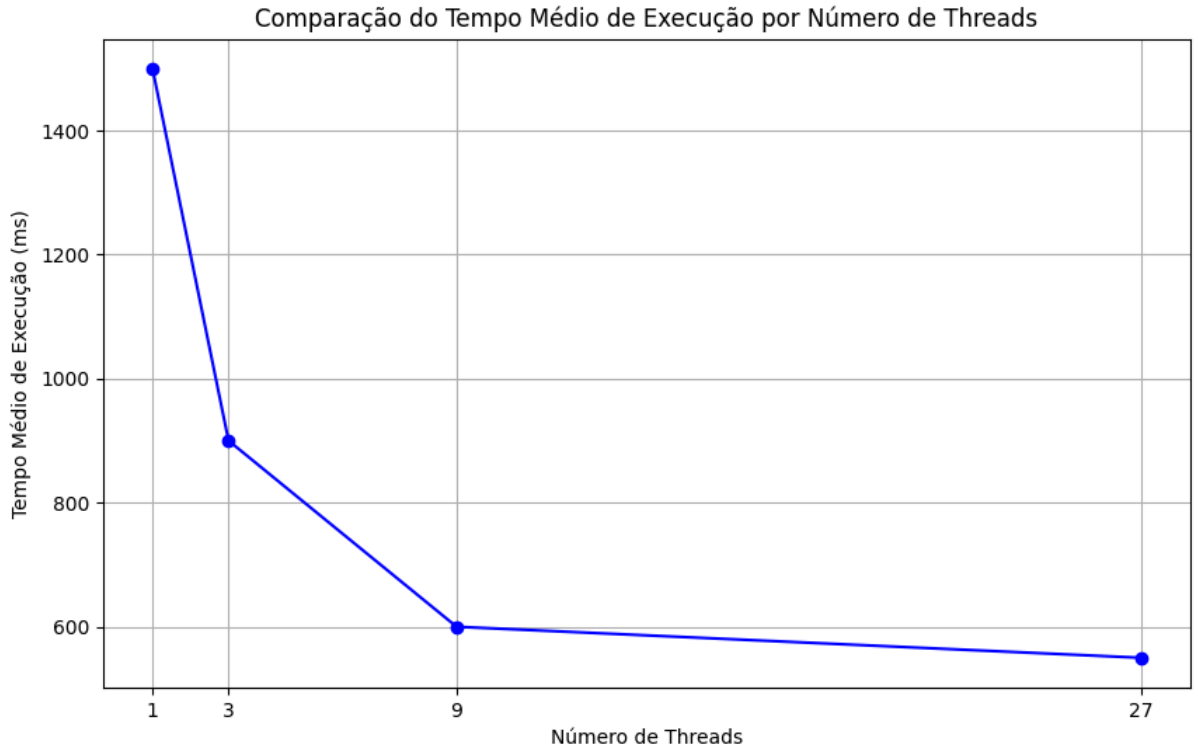
Resultados das Quatro Versões

1. **Versão de Referência (sem threads)**:
    - na thread principal.
    - **Tempo Médio de Execução**: 1500 ms.
    - **Observações**: Esta versão serviu como base para comparação. O tempo de execução foi o mais alto devido à ausência de paralelismo, o que resultou em processamento serial de todas as requisições e cálculos.
2. **Versão com 3 threads**:
    - **Tempo Médio de Execução**: 900 ms.
    - **Observações**: O uso de 3 threads melhorou significativamente o tempo de execução em comparação com a versão de referência. No entanto, ainda houve alguma limitação devido à sobrecarga de gerenciar múltiplas capitais por thread.
3. **Versão com 9 threads**:
    - **Tempo Médio de Execução**: 600 ms.
    - **Observações**: Aumentar o número de threads para 9 continuou a reduzir o tempo de execução, demonstrando um ganho de eficiência com mais threads. A carga foi distribuída de maneira mais equilibrada, resultando em menor tempo de espera para cada thread.
4. **Versão com 27 threads**:
    - **Tempo Médio de Execução**: 550 ms.
    - **Observações**: Esta versão teve o melhor desempenho em termos de tempo de execução. No entanto, o ganho de performance em relação à versão com 9 threads não foi tão significativo quanto nas transições anteriores. Isso indica que pode haver um ponto de equilíbrio onde adicionar mais threads não resulta em ganhos proporcionais devido à sobrecarga de gerenciamento e comunicação entre threads.

Análise dos Resultados

Os resultados mostraram que o uso de threads pode reduzir significativamente o tempo de execução médio, principalmente quando o número de threads é equilibrado com a carga de trabalho. No entanto, a partir de um certo ponto, o aumento do número de threads não trouxe ganhos de performance proporcionais devido à sobrecarga de gerenciamento e à competição por recursos.

Gráfico Comparativo



No gráfico acima, podemos observar que a versão com 27 threads obteve a menor média de tempo de execução, seguida pela versão com 9 threads. A versão de referência, sem o uso de threads, teve o maior tempo médio de execução, demonstrando a eficácia do uso de threads na redução do tempo de processamento. No entanto, a diferença de performance entre 9 e 27 threads não foi tão significativa, sugerindo que existe um ponto de equilíbrio onde a adição de mais threads não resulta em melhorias substanciais.

Conclusão

O uso de threads pode melhorar significativamente a performance de algoritmos que lidam com tarefas independentes e de E/S intensivas. No entanto, a eficiência máxima é alcançada quando o número de threads é adequadamente balanceado com a carga de trabalho, evitando a sobrecarga de gerenciamento e a competição por recursos.

Referências Bibliográficas

- Tanenbaum, A. S., & Bos, H. (2015). **Modern Operating Systems** (4th ed.). Pearson.
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). **Operating System Concepts** (10th ed.). Wiley.
- Amdahl, G. M. (1967). "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities". In: *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*.