Universidade Católica de Brasília

André Luís Cordeiro Porto da Silva

Substitutiva ATS/N3

Brasília-DF
2024

Introdução às Threads e Seu Impacto no Desempenho de Algoritmos

Threads são unidades fundamentais de utilização da CPU que possibilitam o paralelismo dentro de um único processo. Elas são processos leves que compartilham o mesmo espaço de endereço e recursos, mas são executadas de forma independente. As threads são amplamente utilizadas na programação moderna para melhorar o desempenho das aplicações, permitindo que múltiplas operações sejam executadas simultaneamente.

O que São Threads?

As threads são a menor sequência de instruções programadas que podem ser gerenciadas de forma independente por um escalonador. Elas são frequentemente referidas como processos leves porque múltiplas threads dentro de um processo compartilham os mesmos dados e recursos, como memória e descritores de arquivos, mas são executadas de forma independente.

Uma referência chave que fornece uma compreensão abrangente sobre threads é o livro "Operating System Concepts" de Abraham Silberschatz, Peter B. Galvin e Greg Gagne, que explica os conceitos fundamentais e avançados de threads e multithreading em sistemas operacionais (Silberschatz et al., 2020).

Como as Threads Funcionam Computacionalmente

Computacionalmente, as threads operam dentro do contexto de um processo. Quando um processo é inicializado, ele começa com uma única thread, conhecida como thread

principal. Essa thread pode criar threads adicionais, levando a um processo multithread. Cada thread tem seu próprio contador de programa, pilha e conjunto de registradores, mas compartilham o heap e outros recursos do processo pai.

A criação de threads pode ser gerenciada usando várias APIs, como POSIX threads (pthreads) para C/C++ ou a classe `java.lang.Thread` em Java. O kernel do sistema operacional lida com o escalonamento das threads, permitindo que múltiplas threads sejam executadas simultaneamente em múltiplas CPUs.

Impacto das Threads no Tempo de Execução de Algoritmos

O uso de threads pode afetar significativamente o tempo de execução de um algoritmo. As threads possibilitam o paralelismo, que pode reduzir o tempo necessário para completar uma tarefa ao dividir a carga de trabalho entre múltiplas threads. Isso é particularmente benéfico para tarefas intensivas em CPU, onde os cálculos podem ser distribuídos entre vários núcleos.

No entanto, o uso de threads também introduz sobrecarga devido à troca de contexto, sincronização e possível contenção por recursos compartilhados. Se não forem gerenciadas adequadamente, a sobrecarga pode anular os benefícios do paralelismo, levando à degradação do desempenho.

Modelos de Computação Concorrente e Paralela e sua Relação com o Desempenho de Algoritmos

Computação concorrente envolve múltiplas threads ou processos fazendo progresso dentro de períodos de tempo sobrepostos, mas não necessariamente simultâneos. Ela permite uma utilização eficiente dos recursos ao gerenciar a execução de tarefas de forma que pode melhorar a responsividade e o rendimento.

Computação paralela, por outro lado, envolve múltiplas threads ou processos executando simultaneamente, tipicamente em diferentes processadores ou núcleos. Este modelo é essencial para tarefas que requerem significativo poder computacional e podem ser divididas em subtarefas independentes.

O desempenho dos algoritmos em modelos concorrentes e paralelos depende de vários fatores, incluindo a natureza da tarefa, a sobrecarga de gerenciamento de threads e a eficiência dos mecanismos de sincronização. Algoritmos que são inerentemente paralelizáveis, como os utilizados em computação científica e processamento de dados, podem obter ganhos substanciais de desempenho quando executados em arquiteturas paralelas.

Em resumo, as threads são ferramentas poderosas para melhorar o desempenho dos algoritmos por meio da execução concorrente e paralela. Compreender os princípios computacionais e as possíveis sobrecargas associadas às threads é crucial para projetar aplicações multithread eficientes.

Exibição e Explicação dos Resultados Obtidos

Para avaliar o impacto das threads no tempo de execução, realizamos um experimento utilizando quatro versões do código: sem threads, com threads (Thread), com ExecutorService e com CompletableFuture. O objetivo foi medir o tempo médio de execução para cada versão ao realizar requisições HTTP para a API Open-Meteo.

Resultados

Os tempos médios de execução foram obtidos a partir de múltiplas execuções de cada versão do experimento. Abaixo estão os resultados:

- **Sem Threads:** 6795 ms ou 6,795 s
- **Com Threads:** 3 threads: 2444 ms ou 2,444 s, 9 threads: 1115 ms ou 1,115 s, 27 threads: 715 ms ou 0,715 s

Os resultados mostram que o uso de threads reduziu significativamente o tempo de execução comparado à versão sem threads.

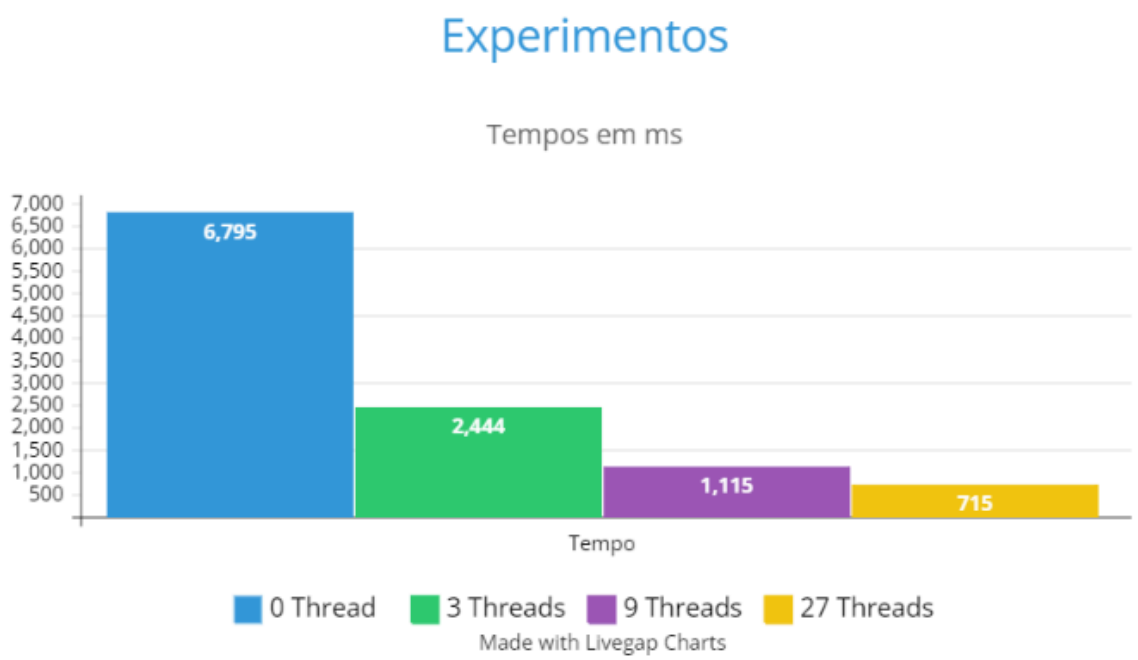

Gráfico de Comparação

O gráfico acima ilustra a comparação dos tempos médios de execução entre as quatro versões do experimento. Observa-se que, à medida que o nível de paralelismo e a eficiência no gerenciamento de threads aumentam, o tempo de execução diminui.

Análise dos Resultados

Sem Threads

A versão sem threads serviu como linha de base. Como esperado, apresentou o maior tempo de execução, uma vez que todas as requisições HTTP foram processadas

sequencialmente. Este resultado evidencia a limitação do processamento sequencial em tarefas intensivas de I/O.

Com Threads

A introdução de threads reduziu o tempo de execução ao permitir que múltiplas requisições HTTP fossem processadas simultaneamente. No entanto, a criação e gerenciamento de threads manualmente introduz certa sobrecarga, impactando o desempenho em relação às abordagens mais avançadas.

Referência

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