**MIGUEL SOUZA DA SILVA**

Avaliação substitutiva - Programação Concorrente e Distribuída

**Brasília**

**202****4**

**O que são threads**

Threads são unidades de execução dentro de um processo, permitindo que múltiplas tarefas sejam executadas simultaneamente. Um processo pode conter um ou mais threads e todos os threads dentro de um processo compartilham o mesmo espaço de memória e recursos do sistema. Threading é um método de multitarefa no qual várias tarefas podem ser executadas simultaneamente dentro do mesmo programa.

Threads são frequentemente usados ​​para melhorar o desempenho do programa, especialmente em sistemas com muitos núcleos de CPU. Em um sistema multithread, diferentes threads podem executar diferentes tarefas em paralelo, utilizando os recursos do sistema de forma mais eficaz e aumentando o desempenho.

**Como threads funcionam computacionalmente**

Teoricamente, os threads são gerenciados pelo sistema operacional, que é responsável por organizar e gerenciar a execução dos threads. Cada thread possui seu próprio contador de programa, conjunto de registros e pilha, mas compartilha espaço de memória e recursos com outros threads no mesmo processo.

Quando um programa é iniciado, o sistema operacional cria um processo e um thread principal. Novos threads podem ser criados a partir do thread principal ou de outros threads existentes. Os threads são então ordenados pelo sistema operacional, que decide qual thread será executado a cada vez, com base na prioridade e em outros fatores.

Threads podem ser usados ​​no nível do usuário ou no nível do kernel:

Threads de nível de usuário: eles são gerenciados pela biblioteca, não pelo sistema operacional. Eles são rápidos de criar e gerenciar, mas podem não funcionar bem em sistemas com muitos núcleos de CPU porque o sistema operacional não reconhece os threads e não pode agendá-los em núcleos diferentes.

Threads de nível de kernel: Eles são gerenciados diretamente pelo sistema operacional. Eles podem ser mais lentos para criar e gerenciar, mas permitem melhor uso de vários núcleos de CPU porque o sistema operacional pode agendar threads em núcleos diferentes.

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

O uso de threads pode afetar significativamente o tempo de execução do algoritmo, tanto positiva quanto negativamente:

Tempo de execução reduzido: Em sistemas com múltiplos núcleos de CPU, o uso de threading pode reduzir o tempo de execução de um algoritmo ao permitir a execução simultânea de diversas partes do algoritmo. Isto é especialmente útil para algoritmos que podem ser divididos em partes independentes que podem ser executadas em paralelo.

Sobrecarga de criação e gerenciamento de threads: A criação e o gerenciamento de threads introduzem sobrecarga, o que pode aumentar o tempo de execução se o número de threads for muito grande ou se o número de operações executadas por cada thread for muito pequeno.

Conflitos de recursos e condições de corrida: Threads que compartilham recursos podem entrar em conflito, levando a condições de corrida e outros problemas de sincronização que podem aumentar o tempo de execução e causar erros difíceis de corrigir.

Eficiência de agendamento de threads: A eficiência com a qual o sistema operacional agenda threads pode afetar o tempo de execução. Se o agendador não equilibrar adequadamente a carga entre os núcleos da CPU, alguns threads poderão ficar ociosos enquanto outros threads estarão muito carregados.

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

Computação paralela e computação paralela são dois modelos de programação que lidam com múltiplas tarefas ao mesmo tempo, mas de maneiras diferentes:

Computação paralela: Refere-se à execução de múltiplas tarefas que podem ser divididas no tempo, mas podem ser executadas ao mesmo tempo. No mesmo sistema, vários trabalhos são executados um após o outro, utilizando os mesmos recursos. A simultaneidade tem mais a ver com a estrutura do programa do que com a simultaneidade real.

Computação paralela: Refere-se à execução simultânea de múltiplas tarefas em múltiplos núcleos de CPU. Num sistema paralelo, múltiplas tarefas são executadas simultaneamente, cada uma em um contexto separado.

O desempenho dos algoritmos no mesmo ambiente ou em ambiente semelhante depende de vários fatores:

Grau de paralelismo: Algoritmos que podem ser divididos em partes independentes que podem ser executadas em paralelo tendem a ter melhor desempenho no mesmo ambiente.

Alta comunicação e sincronização: A comunicação e a sincronização entre diferentes threads criam sobrecarga que pode afetar o desempenho. Em casos semelhantes, a sobrecarga de comunicação entre diferentes núcleos pode ser muito grande.

Balanceamento de carga: o desempenho for maximizado quando a carga de trabalho é balanceada uniformemente entre todos os threads. O desequilíbrio de carga pode levar à subutilização de recursos e ao aumento do tempo de execução.

Escalabilidade: A capacidade dos algoritmos de escalar bem com o aumento da contagem de núcleos é importante para a execução de ambientes paralelos.

**resultados obtidos**

O experimento foi realizado para comparar o desempenho de quatro versões diferentes de um algoritmo que faz solicitações HTTP para coletar e processar dados meteorológicos de 27 capitais brasileiras durante o mês de janeiro de 2024. As versões são:

Versão de referência (threads únicos)

Versão de 3 threads

Versão de 9 threads

Versão de 27 threads

Realizar 27 requisições HTTP (uma para cada capital). Processar os dados recebidos, calculando as temperaturas média, mínima e máxima por capital por dia. Armazenar os dados gerados em memória. Exibir os dados gerados no console.

**Resultados**

Os tempos de execução das 10 repetições para cada versão foram registrados e a média desses tempos foi calculada

|  |  |
| --- | --- |
| **Versão** | **Tempo Médio (ms)** |
| Single-threads | 2424ms |
| 3 Threads | 900ms |
| 9 Threads | 918ms |
| 27 Threads | 903ms |

**Analisar resultados**

Single-threads. Na versão de referência, todas as solicitações e processamento de dados são feitos um único thread. Este método é fácil de usar, mas também lento porque as solicitações HTTP são feitas sequencialmente. Portanto, o tempo total de execução é o maior entre todas as versões testadas.

3 Threads. Na versão 3 threads, cada thread é responsável por 9 solicitações. O tempo de execução é muito menor que a versão single-threads porque as solicitações podem ser executadas em paralelo, aproveitando melhor os recursos do sistema. No entanto, o custo de processamento de vários threads torna-se um fator a ser considerado.

9 threads. Com 9 threads, cada thread é responsável por 3 solicitações. Esta configuração proporciona mais melhorias no tempo de execução em comparação com a versão de 3 threads, pois as solicitações são mais paralelas. No entanto, a redução no tempo de execução não é tão pronunciada como na mudança de um para três threads, sugerindo um ponto de retornos decrescentes.

27 threads. A versão de 27 threads, onde cada thread é responsável por uma solicitação, possui tempo médio de execução menor. Isso porque todas as solicitações podem ser feitas simultaneamente, reduzindo o tempo total de espera. No entanto, essa abordagem pode levar ao alto uso de recursos do sistema e causar problemas de escalabilidade e gerenciamento de threads.

# REFERÊNCIAS

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.

Bovet, D. P., & Cesati, M. (2005). Understanding the Linux Kernel (3rd ed.). O'Reilly Media.

Andrews, G. R. (2000). Foundations of Multithreaded, Parallel, and Distributed Programming. Addison-Wesley.