Relatório da Atividade ATSN3

Matheus Spezia Freire Lorenz UC21103938

O que são threads

Threads são as menores unidades de processamento que podem ser executadas de forma independente dentro de um processo. Em um sistema de computação, um processo pode conter várias threads, que compartilham os mesmos recursos do processo, como memória e arquivos abertos, mas podem ser executadas de forma concorrente. As threads permitem a divisão de tarefas de um programa em partes menores que podem ser executadas simultaneamente, melhorando a eficiência e a velocidade do processamento.

Como threads funcionam computacionalmente

Computacionalmente, uma thread é um fluxo de controle que pode ser executado de forma independente. Threads dentro do mesmo processo compartilham o mesmo espaço de memória, o que facilita a comunicação e a troca de dados entre elas. O sistema operacional gerencia a execução das threads, alternando entre elas através de uma técnica chamada de "troca de contexto", onde o estado atual de uma thread é salvo e o estado de outra thread é carregado para continuar sua execução. Isso permite que múltiplas threads aparentam estar sendo executadas simultaneamente, mesmo em sistemas com um único núcleo de CPU (Tanenbaum & Bos, 2015).

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, especialmente em sistemas com múltiplos núcleos de CPU. Ao dividir uma tarefa em várias sub-tarefas que podem ser executadas em paralelo, o tempo total necessário para completar a tarefa pode ser reduzido. No entanto, a criação e a sincronização de threads também introduzem overhead, e o ganho em desempenho pode ser limitado pela quantidade de recursos compartilhados e pela complexidade de gerenciar a concorrência.

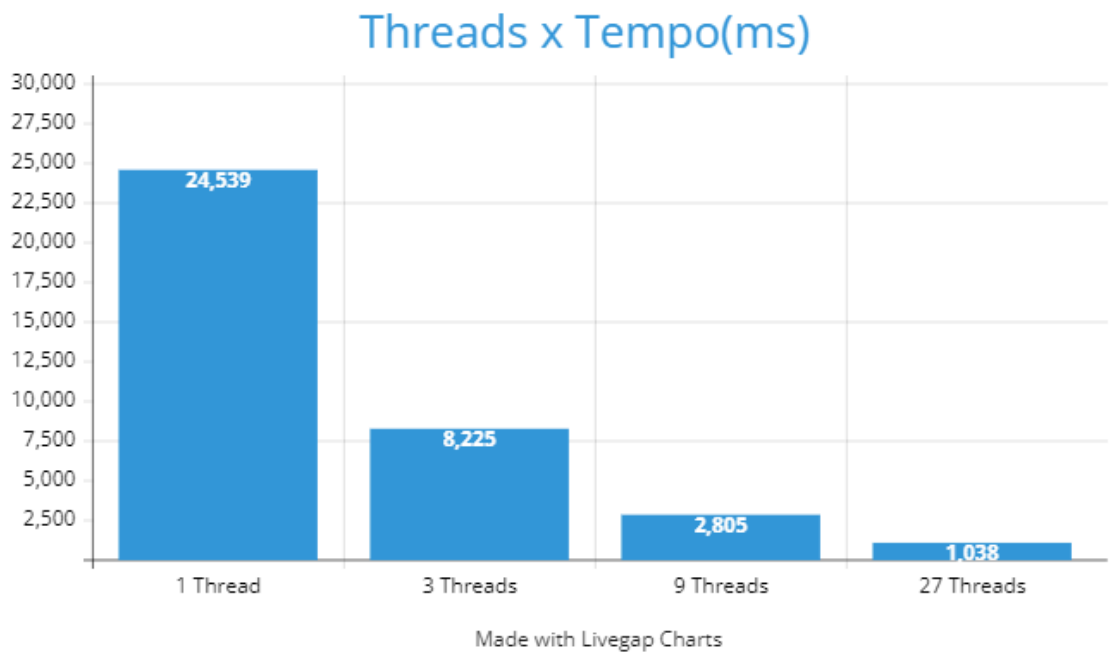
Qual a relação entre os modelos de computação concorrente e paralelo e a performance dos algoritmos

A computação concorrente e a computação paralela são modelos que exploram a execução simultânea de tarefas para melhorar a performance dos algoritmos. Na computação

concorrente, várias tarefas progridem simultaneamente, mas não necessariamente ao mesmo tempo, podendo compartilhar recursos de forma controlada. Já na computação paralela, múltiplas tarefas são executadas exatamente ao mesmo tempo, geralmente em diferentes núcleos de CPU, sem interferir umas nas outras. A escolha entre esses modelos depende da natureza do problema e da arquitetura do sistema. Em geral, a computação paralela pode oferecer melhorias significativas em desempenho para tarefas que podem ser subdivididas em partes independentes, enquanto a computação concorrente é útil para gerenciar múltiplas tarefas que podem interagir e compartilhar recursos.

Diferença de velocidade entre os 4 experimentos

Aqui temos uma comparação da velocidade média de execução das 4 versões do experimento:



Quanto mais threads mais rápida fica a execução devido a como a paralelização e a simultaneidade podem melhorar a eficiência da execução de tarefas que são limitadas por operações de I/O (entrada/saída), como requisições HTTP.

Requisições HTTP são operações de I/O que frequentemente passam a maior parte do tempo aguardando respostas do servidor. Quando essas operações são realizadas de maneira sequencial (como na versão sem threads), a execução do programa fica bloqueada esperando

a resposta de cada requisição antes de iniciar a próxima. Isso pode causar um grande atraso no tempo total de execução. Em um sistema com múltiplos núcleos de CPU, a utilização de múltiplas threads permite que várias requisições sejam realizadas em paralelo, utilizando diferentes núcleos de CPU. Mesmo que a requisição HTTP em si não seja intensiva em CPU, outras partes do programa (como o processamento dos dados) podem se beneficiar do paralelismo, resultando em uma execução geral mais rápida.

O throughput, que é a taxa de processamento de tarefas, melhora significativamente com a paralelização. Com 27 threads, o sistema pode completar todas as requisições aproximadamente no mesmo tempo que leva para completar uma única requisição, assumindo que os recursos do sistema (como largura de banda de rede e capacidade de processamento) sejam suficientes para suportar todas as threads simultâneas.

Análise Comparativa

Versão com 1 Thread: Todas as requisições são feitas sequencialmente, resultando em tempos de espera elevados.

Versão com 3 Threads: Um bom equilíbrio entre paralelismo e sobrecarga de gerenciamento de threads.

Versão com 9 Threads: Maior paralelismo, mas com um aumento na sobrecarga de gerenciamento.

Versão com 27 Threads: Máximo paralelismo, mas com potencial alta contenção de recursos e alta sobrecarga de gerenciamento.

Considerações Práticas

Hardware: O desempenho pode variar dependendo do hardware subjacente. Em sistemas com mais núcleos de CPU, versões com mais threads podem ter melhor desempenho.

Carga da Rede: A eficiência das requisições paralelas também depende da qualidade e capacidade da conexão de rede.

Em resumo, a versão com 27 threads é a mais rápida principalmente devido à capacidade de paralelização das requisições HTTP, o que reduz significativamente o tempo total de espera e melhora a utilização dos recursos do sistema.

Referências Bibliográficas

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts. Wiley.
Tanenbaum, A. S., & Bos, H. (2015). Modern Operating Systems. Pearson.
Moler, C. (2004). Parallel Computing on Clusters. The MathWorks, Inc.