Universidade Católica de Brasília

Pró-Reitoria Acadêmica

Curso de Ciência da Computação

RELATÓRIO – AVALIAÇÃO SUBSTITUTIVA

Aluno(a):Mariana de Sousa Guarino

Professor: João Robson Santos Martins

Brasília – DF

2024

Sumário

O QUE SÃO THREADS

Os threads fazem parte da programação paralela e distribuída, a possibilidade de criar um programa e que este tenha a capacidade de executar múltiplas tarefas ou processos, é o que se espera do papel das threads, threads (fio) traduzido diretamente do inglês, para as linguagens de programações significa a capacidade que um software opere em atividades subdivididas processos ou tarefas de forma simultânea ou coordenadas, portanto podemos entender que "Uma thread (também denominada contexto de execução ou processo peso-leve) é um fluxo sequencial único de execução dentro de um programa, que permite executar tarefas isoladamente."(FERRÃO e FORTES,2005). O conceito de thread apresentado, exemplifica o que são e qual o sentido de sua utilização.

COMO THREADS FUNCIONAM COMPUTACIONALMENTE

Os threads são uma vertente da programação concorrente que visa permitir a paralelização de uma ou mais procedimentos sendo executados, a funcionalidade dos threads é possibilitar o atual funcionamento dos dispositivos eletrônicos, como celulares, computadores, visto que cada vez mais se exigem uma capacidade multifuncional desses aparelhos.

Ao ter um programa executado, tem-se início ao processo que por vezes pode ter uma ou mais threads sendo inicializadas, a forma que cada thread lida com uma parte específica do código possibilita ao programa realizar múltiplas tarefas ao mesmo tempo.

Computacionalmente falando "Esse recurso é chamado multithreading, ou multiescalonamento e possibilita a realização de múltiplas atividades em paralelo, proporcionando melhoras evidentes no desempenho, principalmente de tarefas mais complexas." (FERRÃO e FORTES,2005). Ao realizarem tais procedimentos, utilizar threads permite ganhos de eficiência e desempenho, como já apresentado paralelismo, ou seja, execução simultânea de várias partes de um mesmo programa, comunicação rápida, uma vez que partilham os mesmos espaços de memória, uso mais eficiente dos recursos disponíveis.

COMO O USO DE THREADS PODE AFETAR O TEMPO DE EXECUÇÃO DE UM ALGORITMO

O uso de threads pode ter um grande impacto no tempo de execução de algoritmos. Quando aplicado corretamente, o uso de threads pode levar a ganhos significativos de desempenho, assim “Os resultados mostram que a paralelização demonstra ser mais eficiente do que o processamento puramente sequencial.” (GARCIA, 2013) conforme exposto.

Ocorre que os threads permitiram que várias requisições fossem processadas de forma concorrente, reduzindo o tempo de resposta. Além disso, a criação e destruição de threads são mais rápidas do que as de processos, trazendo uma abordagem mais eficiente em termos de recursos, conforme já mencionado.

QUAL A RELAÇÃO ENTRE OS MODELOS DE COMPUTAÇÃO CONCORRENTE E PARALELO E A PERFORMANCE DOS ALGORITMOS.

Os modelos de computação concorrente e paralela são essenciais para melhorar a performance dos algoritmos. Na computação concorrente, múltiplas tarefas são executadas de forma intercalada, o que é útil em ambientes com alta latência de E/S, permitindo o uso eficiente do tempo ocioso “Assim, ao descrever a concorrência de uma aplicação, o programador atenta em explicitar quais atividades podem ser executadas de forma simultânea e como devem estas reagir a mudanças de estado no programa” (MORAIS, TORCHELSEN e CAVALHEIRO, 2022).

Ao detalhar cuidadosamente a concorrência em uma aplicação durante o desenvolvimento. O programador precisa especificar quais atividades podem ser executadas simultaneamente e definir claramente como essas atividades devem reagir às mudanças de estado do programa. Não obstante, na programação paralela “A programação paralela, por sua vez, envolve a seleção de recursos de programação visando a melhor execução, em termos de obtenção de desempenho, em uma máquina específica” (MORAIS, TORCHELSEN e CAVALHEIRO, 2022). Esse cuidado é essencial para detalhar a execução simultânea de atividades e sua reação a mudanças de estado.

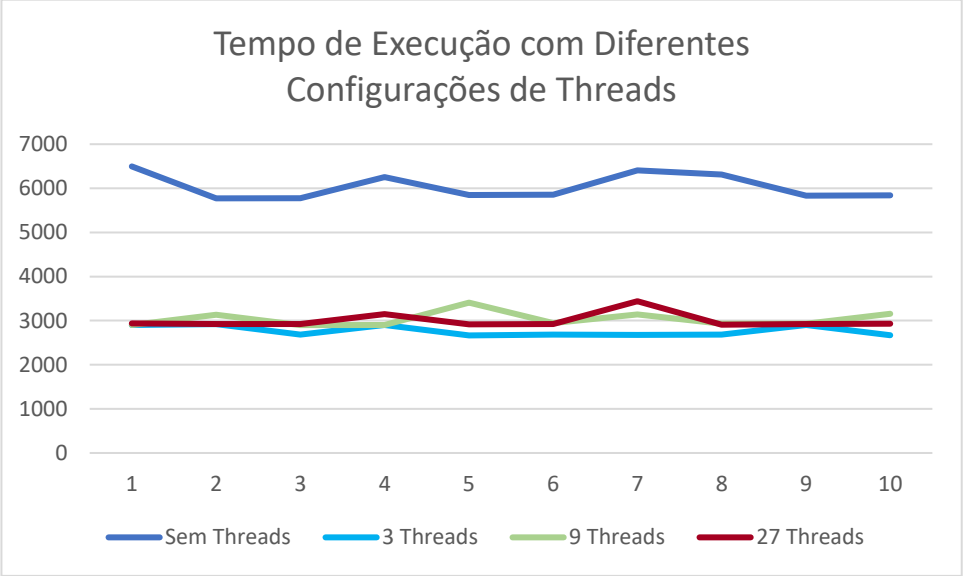
Além disso, enfatizam a programação paralela como um processo de selecionar recursos de programação para otimizar o desempenho em uma máquina específica, visando alcançar maior eficiência.

EXIBIÇÃO E EXPLICAÇÃO DOS RESULTADOS OBTIDOS

Os dados fornecidos pelo programa, possibilitaram a construção de gráficos de linhas e gráficos de barras para a aferição dos resultados obtidos.

O gráfico de linhas terá o papel de demonstrar o tempo de execução para cada configuração de threads ao longo das 10 execuções, permitindo através desse tipo gráfico visualizar as tendências e variações no desempenho de cada configuração de threads.

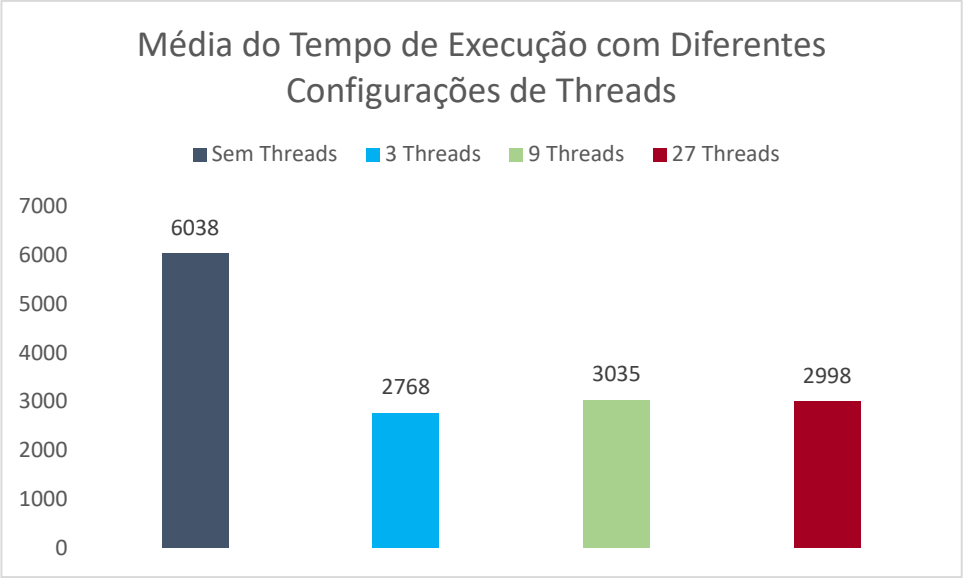
Em seguida, será apresentado um gráfico de barras para comparar as médias de desempenho das diferentes configurações de threads. Este tipo de gráfico será útil para visualizar claramente a comparação direta das médias de desempenho.



Este é o gráfico de linhas que mostra o tempo de execução para cada configuração de threads ao longo das 10 execuções. Assim pode-se observar que:

- O tempo de execução sem threads é significativamente maior em comparação com qualquer configuração de threads.
- As configurações com threads 3, 9 e 27 apresentam tempos de execução bem mais baixos e consistentes, com pequenas variações entre elas.
- A configuração sem threads mostra uma variação maior entre as execuções, enquanto as configurações com threads são mais estáveis.

Agora, para o gráfico de tempo médio de execução, será utilizado um gráfico de barras para comparar as médias de desempenho das diferentes configurações de threads



O gráfico de barras acima compara a média do tempo de execução para cada configuração de threads. Assim observar-se que:

- A média do tempo de execução sem threads é significativamente maior, cerca de duas vezes mais alta que a das configurações com threads.
- As configurações com 3, 9 e 27 threads têm tempos de execução médios bastante próximos, indicando que o aumento do número de threads além de 3 não proporciona uma melhoria significativa no desempenho.
- A configuração com 9 threads tem um leve aumento no tempo médio de execução em comparação com 3 e 27 threads, sugerindo que pode haver uma sobrecarga na criação e gerenciamento de um número maior de threads.

Análise:

- Usar threads reduz em muito o tempo de execução em comparação com a execução sem threads.
- A utilização de 3 threads parece ser a configuração mais eficiente, já que os tempos médios adicionais de threads não resultam em ganhos substanciais de desempenho.
- Além de 3 threads, a variação no desempenho é mínima, indicando que o a utilização mais otimizada para esse sistema é usar um número moderado de threads.

BIBLIOGRAFIA

FERRÃO, Lucilene Baêta; FORTES, Reinaldo Silva. Programação Paralela e Distribuída em Java.

GARCIA, Adriano Marques et al. Analisando Duas Técnicas de Paralelização em um Algoritmo Mergesort in-place. In: **Workshop de Iniciação Científica (WSCAD-WIC 2013) evento integrante do XIV Simpósio em Sistemas Computacionais (WSCAD-SSC 2013)**. 2013. p. 258-261.

MORAIS, Lucas; TORCHELSEN, Rafael; CAVALHEIRO, Gerson Geraldo H. Incentivando a adoção de modelos de programação concorrente/paralela a partir da visualização de ganhos na geração de fractais em tempo-real. In: **Anais do XXXIII Simpósio Brasileiro de Informática na Educação**. SBC, 2022. p. 311-322.