Relatório

Aluno: Vinícius Tadeu Soares Silva - UC22102142

Introdução

Nos sistemas computacionais modernos, a eficiência e o desempenho são frequentemente melhorados através da utilização de threads. Threads são unidades básicas de processamento que permitem que um programa execute múltiplas tarefas de forma concorrente. Este texto explora o conceito de threads, seu funcionamento computacional, impacto no tempo de execução de algoritmos e a relação entre modelos de computação concorrente e paralela na performance de algoritmos.

O que são threads?

Threads são sequências independentes de execução que compartilham o mesmo espaço de endereçamento e recursos de um processo maior. Em um sistema operacional, um processo pode ter uma ou múltiplas threads, cada uma representando uma linha de execução dentro do contexto do processo. Diferente dos processos, threads compartilham memória e recursos diretamente, o que as torna mais leves e rápidas de criar e de comutar entre si.

Como as threads funcionam computacionalmente?

Cada thread possui seu próprio contador de programa, conjunto de registradores e pilha de execução, mas compartilha memória e recursos com outros threads do mesmo processo. Isso permite que threads cooperem e coordenem suas atividades de forma eficiente, ideal para operações que podem ser subdivididas em tarefas menores que podem ser executadas simultaneamente ou de forma concorrente.

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 processamento. Algoritmos que possuem partes independentes ou que podem ser paralelizadas podem ser divididos em threads que executam simultaneamente, acelerando assim a execução. No entanto, é importante gerenciar corretamente a concorrência entre threads para evitar condições de corrida e garantir consistência nos dados compartilhados.

Segundo Silberschatz, Galvin e Gagne (2018), "threads são populares em sistemas operacionais modernos porque podem melhorar a performance através da exploração de múltiplos núcleos de CPU e podem melhorar a responsividade das aplicações que interagem com dispositivos de I/O" (p. 197).

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

Os modelos de computação concorrente e paralela desempenham papéis essenciais na performance dos algoritmos. Na computação concorrente, múltiplas threads cooperam para resolver problemas simultaneamente, dividindo a carga de trabalho entre si e potencialmente reduzindo o tempo total de execução. Por outro lado, na computação paralela, o objetivo é executar múltiplas tarefas ao mesmo tempo, aproveitando eficientemente os recursos disponíveis.

A escolha entre modelos depende das características do problema e das capacidades do hardware. Algoritmos que podem ser decompostos em tarefas independentes são candidatos ideais para a computação paralela, enquanto problemas que envolvem interação entre tarefas ou dependências de dados são mais adequados para a computação concorrente.

Conclusão

Em resumo, threads são unidades básicas de execução que permitem a implementação de computação concorrente eficiente. A utilização de threads pode reduzir o tempo de execução de algoritmos ao explorar a paralelização de tarefas. A escolha entre modelos de computação concorrente e paralela é crucial para otimizar a performance de algoritmos, garantindo que o software aproveite ao máximo o potencial do hardware disponível.

Referência bibliográfica:

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

https://www.devmedia.com.br/programacao-com-threads/6152

https://www.techtudo.com.br/noticias/2019/01/o-que-sao-threads-e-para-que-servem-em-um-processador.ghtml

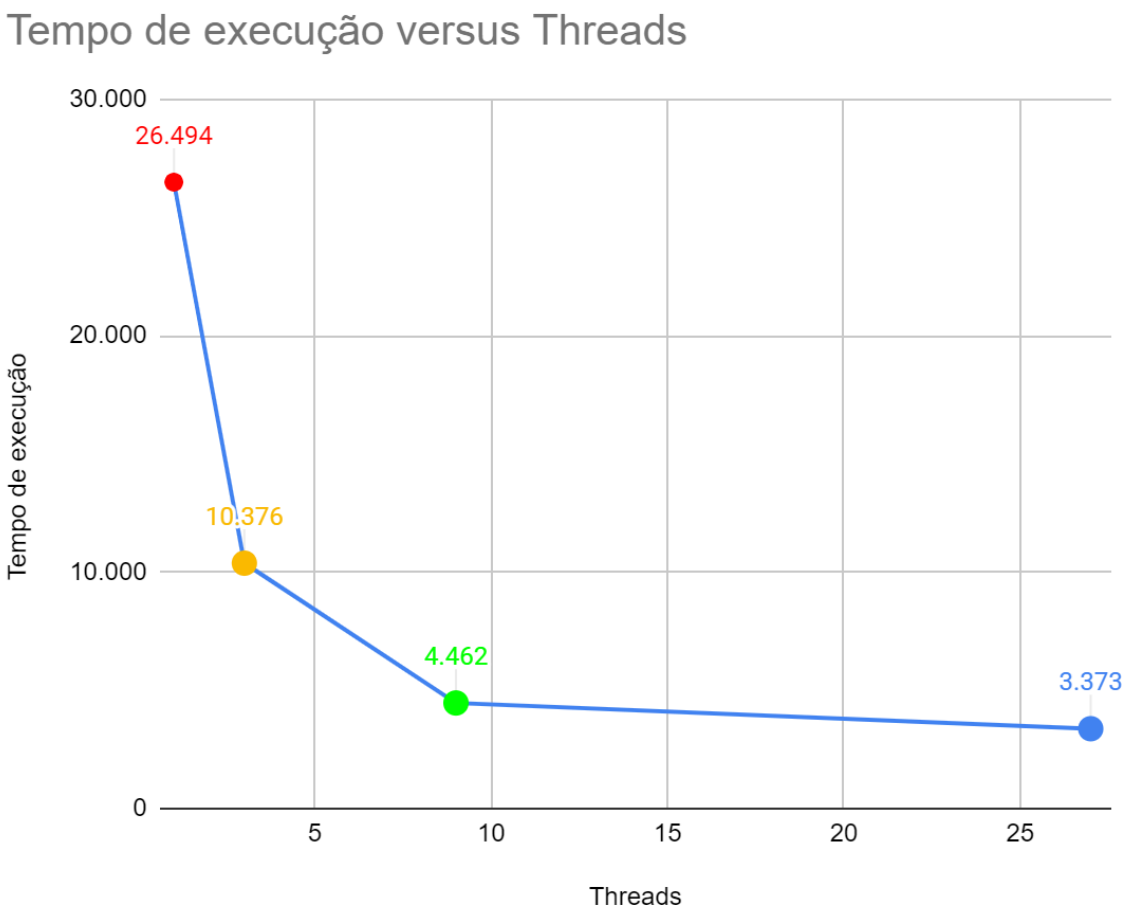
Experimento

Após o teste com 1 Thread, o tempo de execução foi de **26.494** milissegundos.

Após o teste com 3 Threads, o tempo de execução foi de **10.376** milissegundos.

Após o teste com 9 Threads, o tempo de execução foi de **4.462** milissegundos.

Após o teste com 27 Threads, o tempo de execução foi de **3.373** milissegundos.



Portanto, pode-se concluir que quanto mais threads utilizadas, menor o tempo de execução do programa.

Threads	Execução	Tempo de execução
1	1	26.494
	2	26.492
	3	26.471
	4	26.546
	5	26.514
	6	26.410
	7	27.218
	8	26.313
	9	26.766
	10	26.485
3	1	10.376
	2	9.845
	3	9.951
	4	9.951
	5	9.839
	6	9.821
	7	9.871
	8	9.821
	9	10.060
	10	9.862
9	1	4.462
	2	4.251
	3	4.253
	4	4.227
	5	4.173
	6	4.517
	7	4.355
	8	4.269
	9	4.332
	10	4.163
27	1	3.373

	2	2.607
	3	2.602
	4	2.784
	5	2.619
	6	2.672
	7	3.279
	8	2.708
	9	3.335
	10	2.884