Projeto 2 - Sistemas Distribuídos

Prof. Dr. Aleardo Manacero Junior

UNESP - IBILCE

27/07/2020

Giovana Giardini Borges, Rafael Fernandez Campos, Vinícius Martins Pinto da Silva

Objetivo

Este trabalho tem como objetivo calcular a integral da equação (1) de forma distribuída, utilizando *Message Passing Interface* (MPI) em uma arquitetura mestre-escravo. A integral deve ser calculada pelo método do trapézio.

$$\int_0^{100} f(x)dx = \int_0^{100} \sqrt{100^2 + x^2} \qquad (1)$$

Metodologia

O problema foi modelado com a arquitetura mestre-escravo. Assim, existe um mestre que requisita a resolução da integral em *n* subintervalos para *n* escravos. Dessa forma, cada escravo realiza o cálculo da integral descrita na equação (1) em um subintervalo e retorna o valor parcial.

O cálculo, por sua vez, é realizado utilizando o método do trapézio. Este consiste em dividir o intervalo de integração [a, b] em *n* subintervalos discretos de tamanho Δx. Dessa forma, tem-se os limites da integral definidos na Equação (2):

$$\begin{aligned} x_0 &= a \\ x_{i+1} &= x_i + \Delta x \qquad (2) \\ x_n &= b \end{aligned}$$

Assim, a integral pode ser aproximada para o somatório das áreas do trapézio formados pelos subintervalos, conforme mostra a equação (3):

$$\int_0^{100} f(x)dx \simeq \sum_{i=0}^{n-1} \frac{1}{2} (f(x_i) + f(x_{i+1})) * \Delta x \qquad (3)$$

Por fim, esses resultados parciais são enviados por cada escravo para o mestre, o qual realiza a soma dos resultados parciais, exibindo o resultado final do cálculo da integral. Para implementar o somatório dos resultados parciais, utilizamos a função MPI_REDUCE disponível na biblioteca *mpi.h*.

Resultados e discussão

O comportamento do programa foi analisado de acordo com o tempo de execução em relação aos diferentes dados de entrada, os quais variam o número de escravos entre 1, 2, 4 e 10, com intervalos de discretização(k) iguais a 0.0001, 0.00001 e 0.000001.

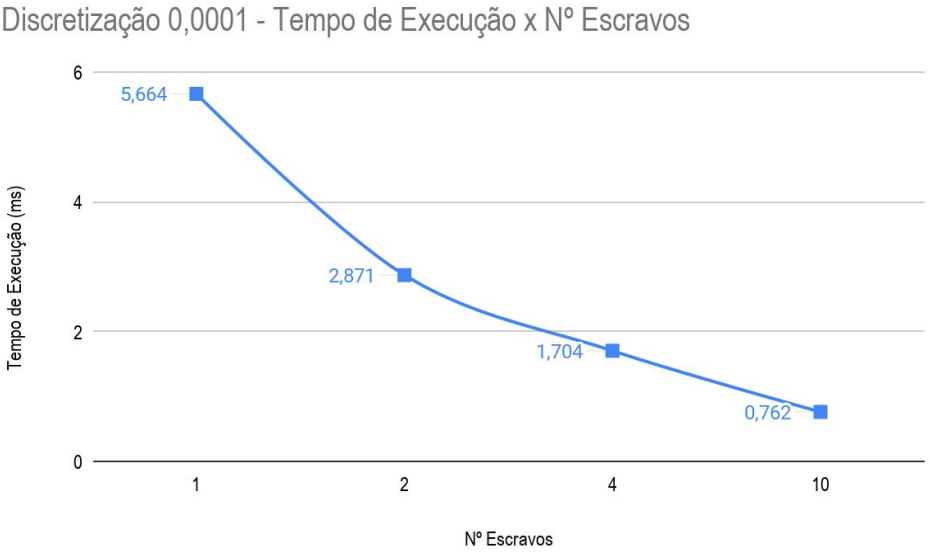

Figura 1: Gráfico TxE com discretização 0,0001

Por meio da *Figura 1,* percebe-se que, ao ter um maior número de escravos, tem-se uma diminuição do tempo de execução. Esse comportamento deve-se à divisão de tarefas entre os escravos, o que permite que os cálculos dos intervalos sejam realizados de forma simultânea, diminuindo o tempo e aumentando a eficiência do programa.

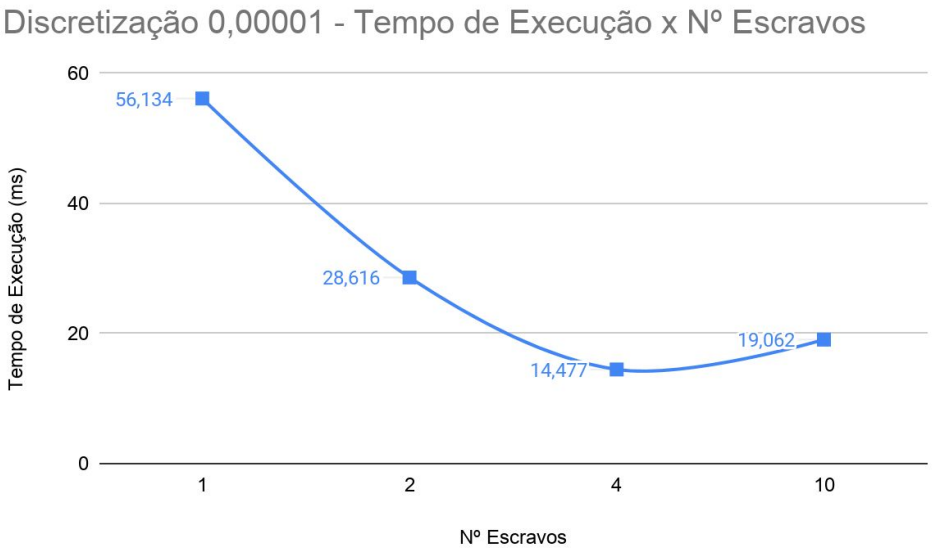

Figura 2: Gráfico TxE com discretização 0,00001

Já no gráfico da *Figura 2,* observa-se que, para o intervalo de discretização 0,00001, o comportamento do tempo de execução em relação ao número de escravos sofre algumas mudanças em relação ao intervalo anterior. Inicialmente, para 1 escravo, observa-se um tempo de execução igual a 56,134 ms que decai para 28,616 ms e, na sequência, foi obtido o melhor valor para discretização 0,00001, 14,477 ms, utilizando 4 servidores escravos. Após isso, na utilização de 10 escravos, percebemos um certo aumento no tempo de execução.

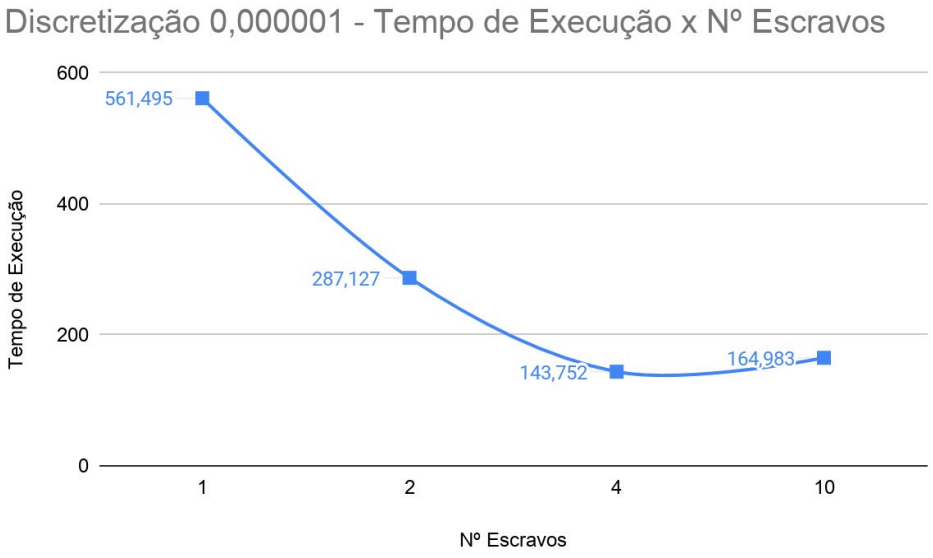

Figura 3: Gráfico TxE com discretização 0,000001

Por fim, na *Figura 3,* a qual apresenta os resultados do menor intervalo de discretização testado, 0.000001, tem-se uma redução no tempo de execução ao utilizar entre 1 e 4 escravos. Após isso, isto é, após executar com seu valor ótimo, sofre um aumento no tempo ao utilizar 10 escravos. Dessa forma, apresenta comportamento semelhante ao intervalo 0.00001, apresentado na *Figura 2.*

Conclusão

Diante dos testes apresentados, pôde-se concluir que o tempo de execução diminui com a paralelização da tarefa utilizando múltiplos processos escravos até que o programa atinja o menor tempo de execução, apresentando, após isso, uma queda de desempenho. Porém, é possível perceber, claramente, que a abordagem do sistema distribuído possibilita uma execução em tempos muito menores do que aqueles obtidos ao utilizar apenas um processo escravo para o cálculo, isto é, ao realizar os cálculos de maneira sequencial.

Uma possível explicação para o aumento de tempo de execução ao utilizar 10 processos escravos seria as limitações de hardware, visto que o processador no qual foi executado o programa possui 6 núcleos e 6 threads. Tal configuração possibilita o processamento simultâneo de no máximo 5 escravos, sendo 1 núcleo utilizado pelo mestre.

Execução do programa

Para compilar o projeto é necessário, em um terminal linux, utilize o comando **mpicc -o main main.c -lm.** Após compilado, para execução, deve-se utilizar o comando **mpirun -np *n* --hostfile hostfile main *k***, sendo *n* o número de processos que o MPI criará para a execução e *k* o intervalo de discretização, que de acordo com a especificação do projeto varia entre 0.0001, 0.00001 e 0.000001. Dessa forma, um exemplo utilizando 4 escravos e intervalo de discretização de 0.0001, possui comando igual a **mpirun -np 5 --hostfile hostfile main 0.0001**, criando um processo mestre e 4 escravos.