TADS

NOITE

DS142 – LINGUAGEM DE PROGRAMAÇÃO ORIENTADA A OBJETOS II

ATIVIDADE – THREADS

ALUNO: CASSIANO KRUCHELSKI VIDAL
GRR20184619

DESCRIÇÃO DA ATIVIDADE:

O presente exercício tem como objetivo realizar a análise de desempenho de um computador na execução de um programa de multiplicação de matrizes quadradas utilizando single thread e multi threads.

O programa em questão faz o seguinte: utiliza X threads para multiplicar duas matrizes, sendo que cada thread se responsabiliza por um conjunto de linhas da matriz que pode ser desde 1 linha (caso com maior número de threads) ou todas as linhas (quando o programa é executado com thread única).

O programa recebe dois argumentos na sua execução que se referem ao tamanho da matriz e ao número de threads. Por exemplo:

$ java -jar desafioMatrizNoPrint.jar 100 10

No exemplo, seriam construídas matrizes de tamanho 100 x 100 e o programa seria executado com 10 threads.

SOBRE O ALGORITMO

Inicialmente o programa converte os argumentos passados na hora da execução do programa para números inteiros e faz as devidas atribuições. Na sequência são criadas as duas matrizes com números aleatórios e é declarada a matriz de resultado com números 0.

Após são instanciados objetos da classe MultMatriz (que estende a classe Thread) que possui o método para multiplicação utilizando threads. Detalhe que a quantidade de objetos instanciados está ligada à quantidade de threads passadas como argumento ao programa.

Na sequência é iniciada a multiplicação (com o método start()), sendo que cada thread assume um número de linhas calculadas da seguinte forma:

Tamanho da matriz / Número de threads

Em caso de haver um resto para essa conta, este é somado ao número de linhas que a última thread irá calcular. Por fim, cada objeto da classe MultMatriz executa o método join() para que seja feito o sincronismo dos cálculos e a matriz resultante esteja correta.

CONSIDERAÇÕES SOBRE O ALGORITMO

Existem duas versões do algoritmo:

- desafioMatriz
- desafioMatrizNoPrint

Ambas as versões realizam a multiplicação através do mesmo algoritmo, a diferença é que o primeiro faz a impressão das matrizes e é destinado para conferência do resultado. Já o segundo é utilizado para benchmark, sem impressões e, portanto, sem chamadas de sistema visando acesso no Output padrão.

RESULTADOS E MENSURAÇÕES

Para a realização das diversas execuções do programa, juntamente com as respectivas medições, foi utilizado o programa hyperfine. Esta aplicação realiza execuções automatizadas, mediante parâmetros, com finalidade de realizar benchmarks. A execução ocorreu na seguinte máquina:

Apple MacBook Air (13”, 2017)

- Processador: Intel Core i5 dual core de 1,8 GHz (Turbo Boost de até 2,9 GHz) e 3 MB de cache L3 compartilhado
- Memória: Memória integrada LPDDR3 de 8 GB com 1600 MHz
- Armazenamento: SSD PCIe de 128 GB
- Vídeo: Intel HD Graphics 6000
- Sistema operacional macOS Catalina 10.15.1

E realização dos testes ocorreu logo após ligar o computador, com o carregador conectado e cerca de 50% da carga de bateria. A execução do programa foi feita na aplicação *Terminal* (simulador de terminal nativo do Sistema Operacional macOS), variando o número de threads de 1 a 10 e executando 10 vezes para *warmup* (visando reservar espaços de memória e criar tabelas de paginação) e então mais 10 vezes para cada quantidade de thread. Por fim, através do hyperfine, foi calculada a média entre as 10 execuções e o desvio padrão. Um exemplo de comando utilizado no terminal para a execução, conforme descrita, foi:

$ hyperfine -r 10 -w 10 --parameter-scan x 1 10 ‘java -jar desafioMatrizNoPrint.jar 100 {x}’ --export-csv results100.csv

Sendo que:

- -r 10 é o argumento que indica para que sejam executadas 10 vezes
- -w 10 é o argumento que indica para ser executadas 10 vezes para *warmup*

- --parameter-scan x 1 10 é o argumento que define x como sendo a variável que será utilizada como argumento do programa correspondente ao número de threads e os valores (1 10) indicando que a variação será de 1 até 20 com incremento de 1
- ‘java -jar desafioMatrizNoPrint.jar 100 {x}’ é a chamada do programa sendo que x é a variável que será automaticamente incrementada e que representa as threads e o 100 é o argumento para a quantidade de linhas e colunas (este valor foi alterado ao longo das execuções).
- --export-csv results1000.csv é o nome do arquivo exportado com os resultados.

Os valores de colunas/linhas utilizados foram os seguintes: 10, 50, 100, 250, 500, 750, 1000.

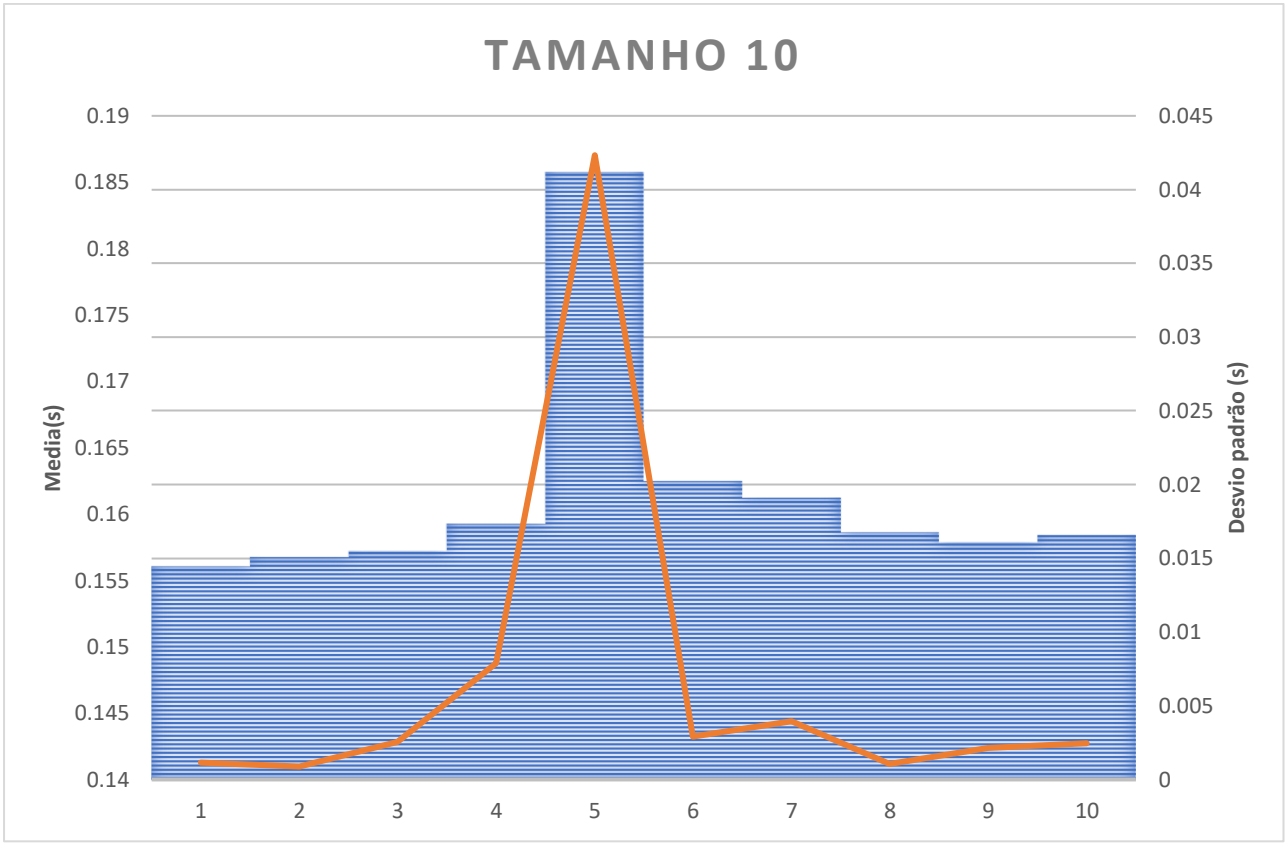
A seguir, seguem as tabelas e gráficos obtidos pelo hyperfine e processados no Microsoft Excel.

Sobre os gráficos:

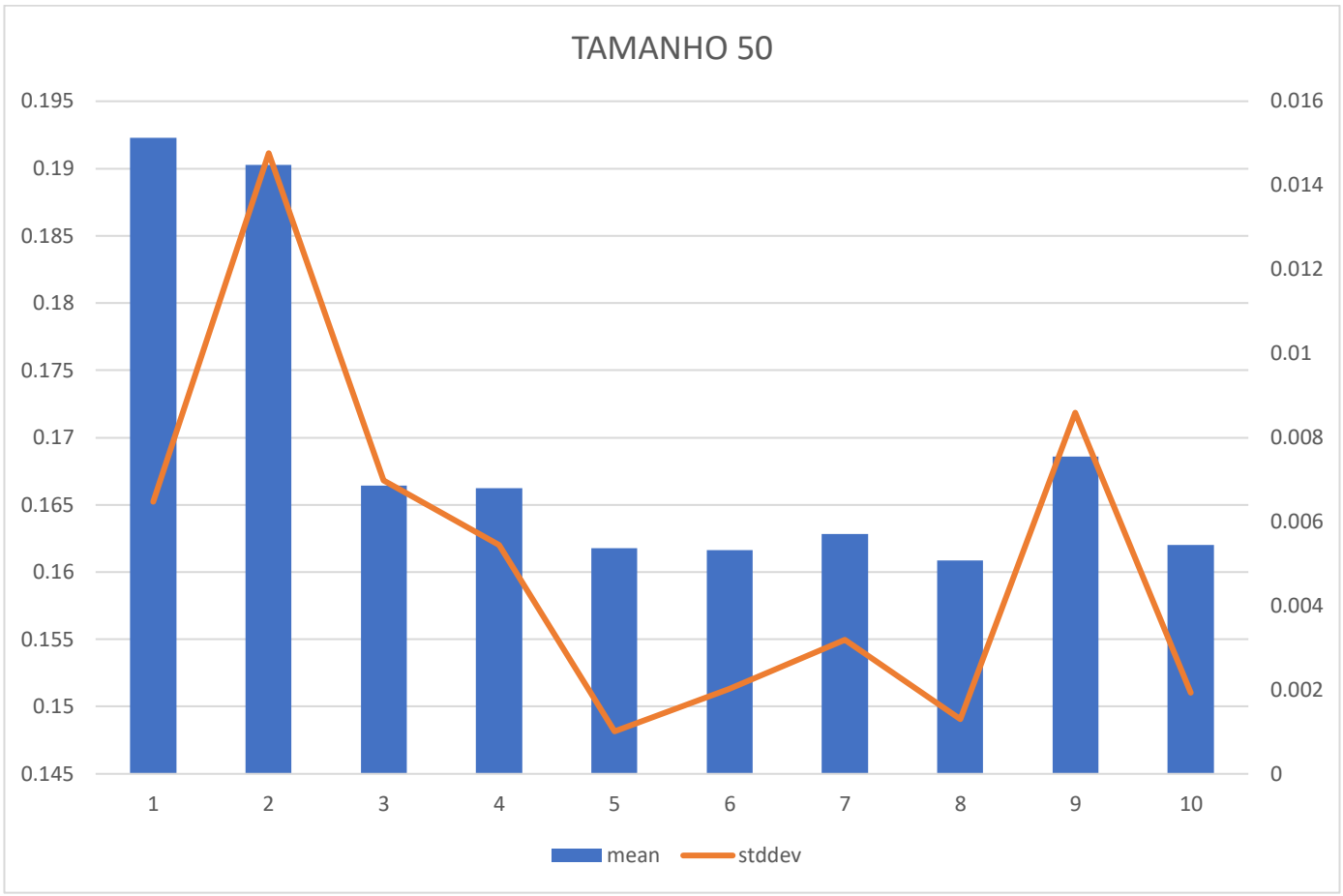
- mean, cujos valores estão no eixo Y esquerdo, representa o tempo médio (em ms)
- stddev, cujos valores estão no eixo Y direito, representa o desvio padrão (em ms)
- threads, cujos valores estão no eixo X, indicam a quantidade de threads na execução

OBS: Neste arquivo serão apresentados apenas os gráficos. As tabelas com os dados obtidos serão enviados em arquivos separados

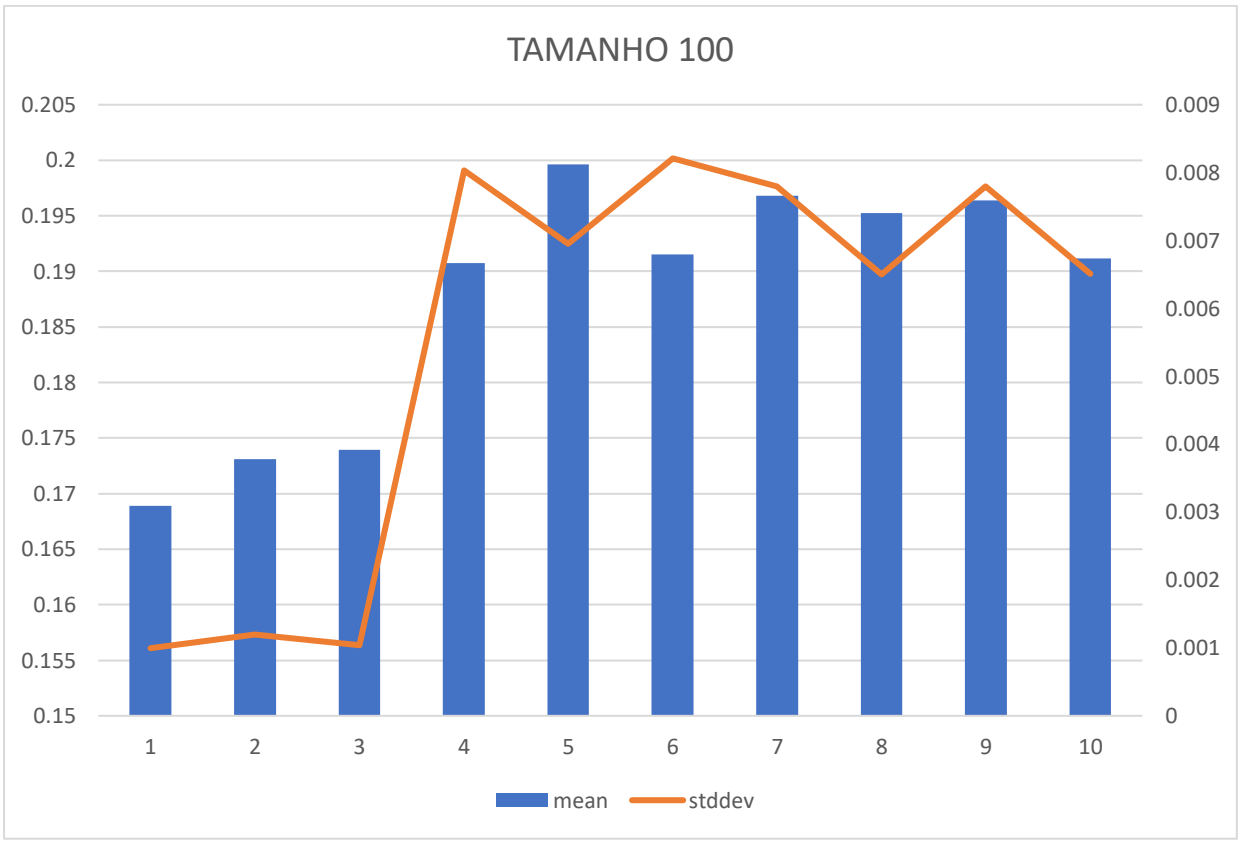
10 linhas x 10 colunas



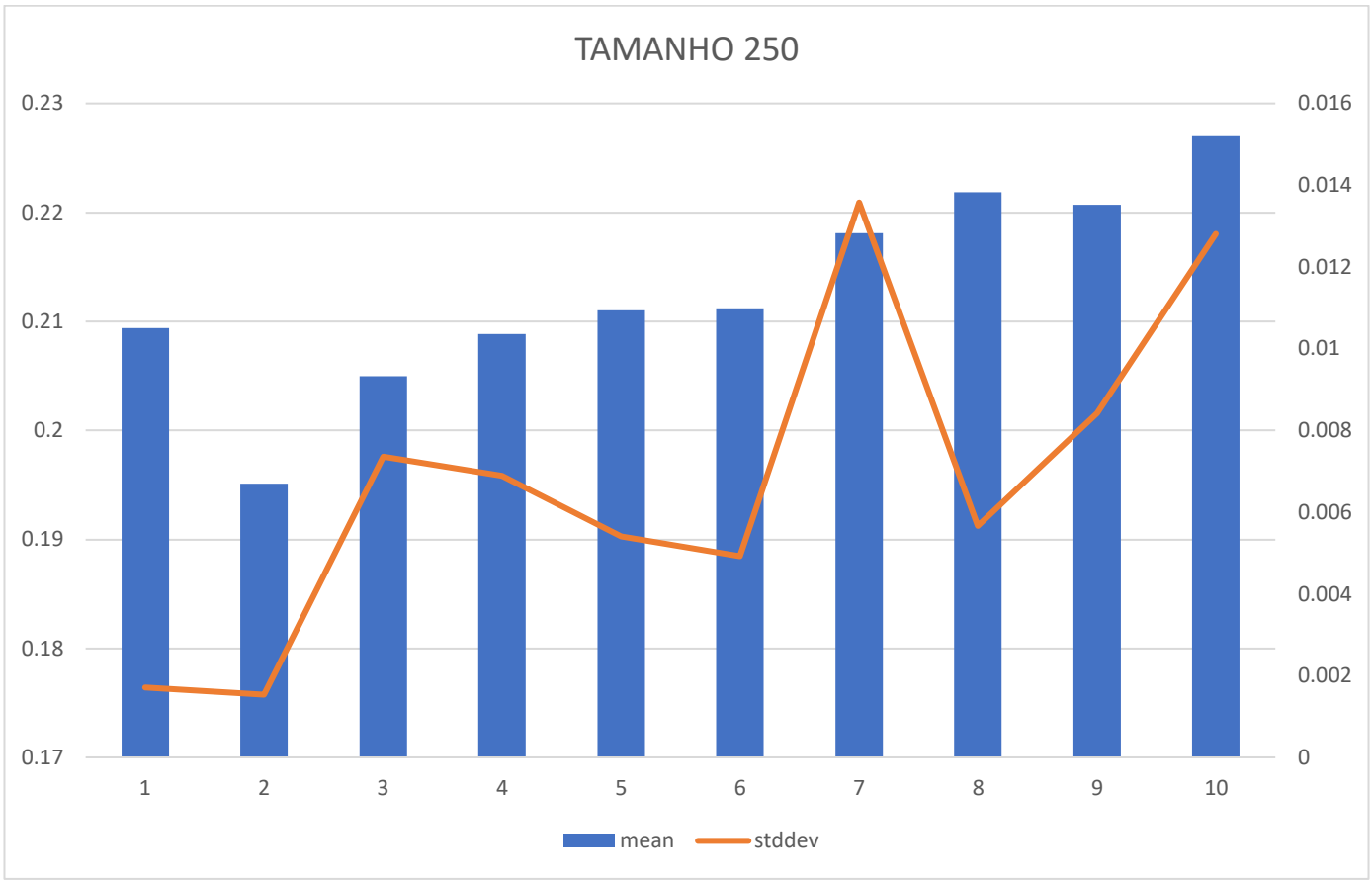
50 linhas x 50 colunas



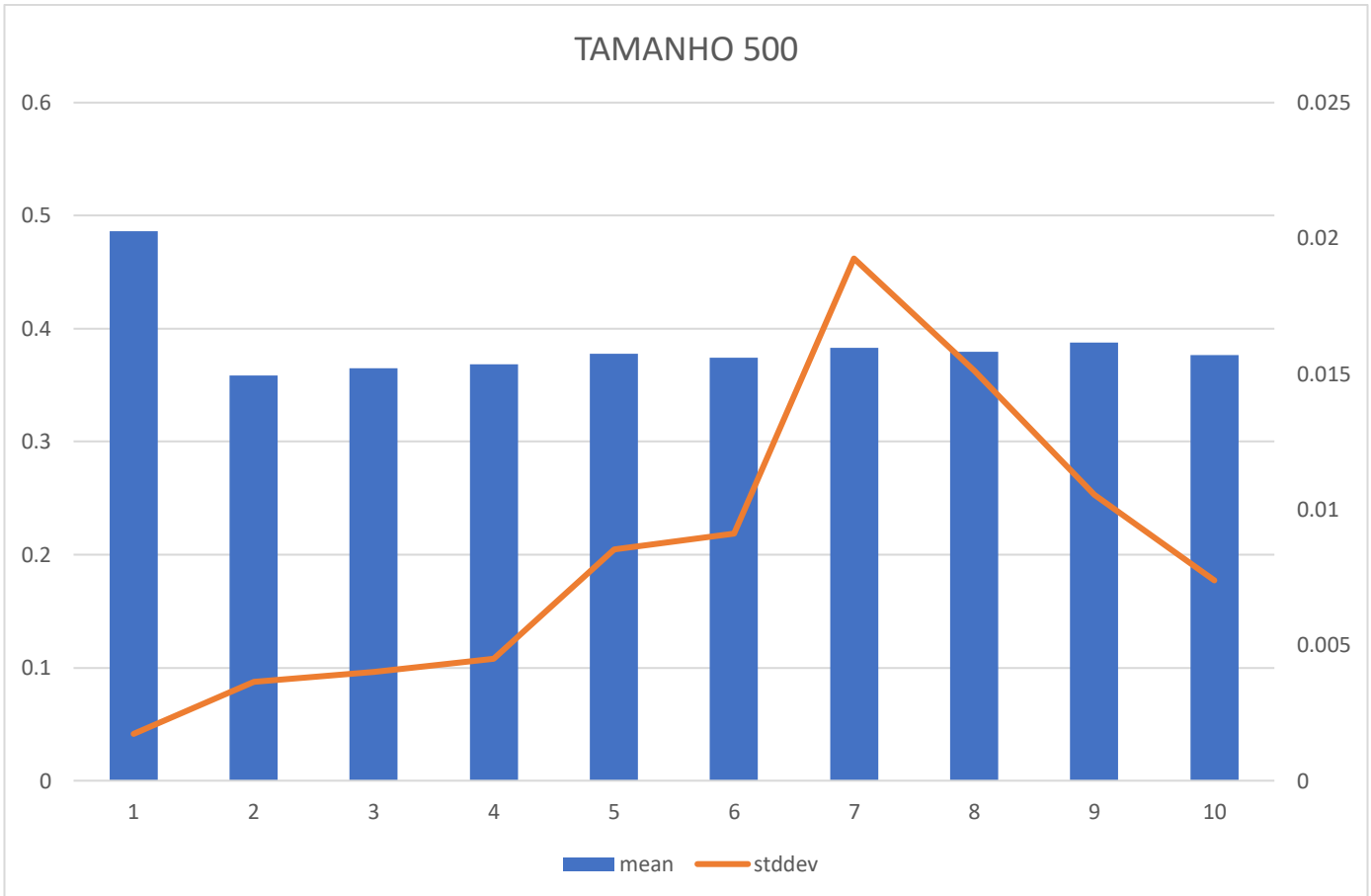
100 linhas x 100 colunas



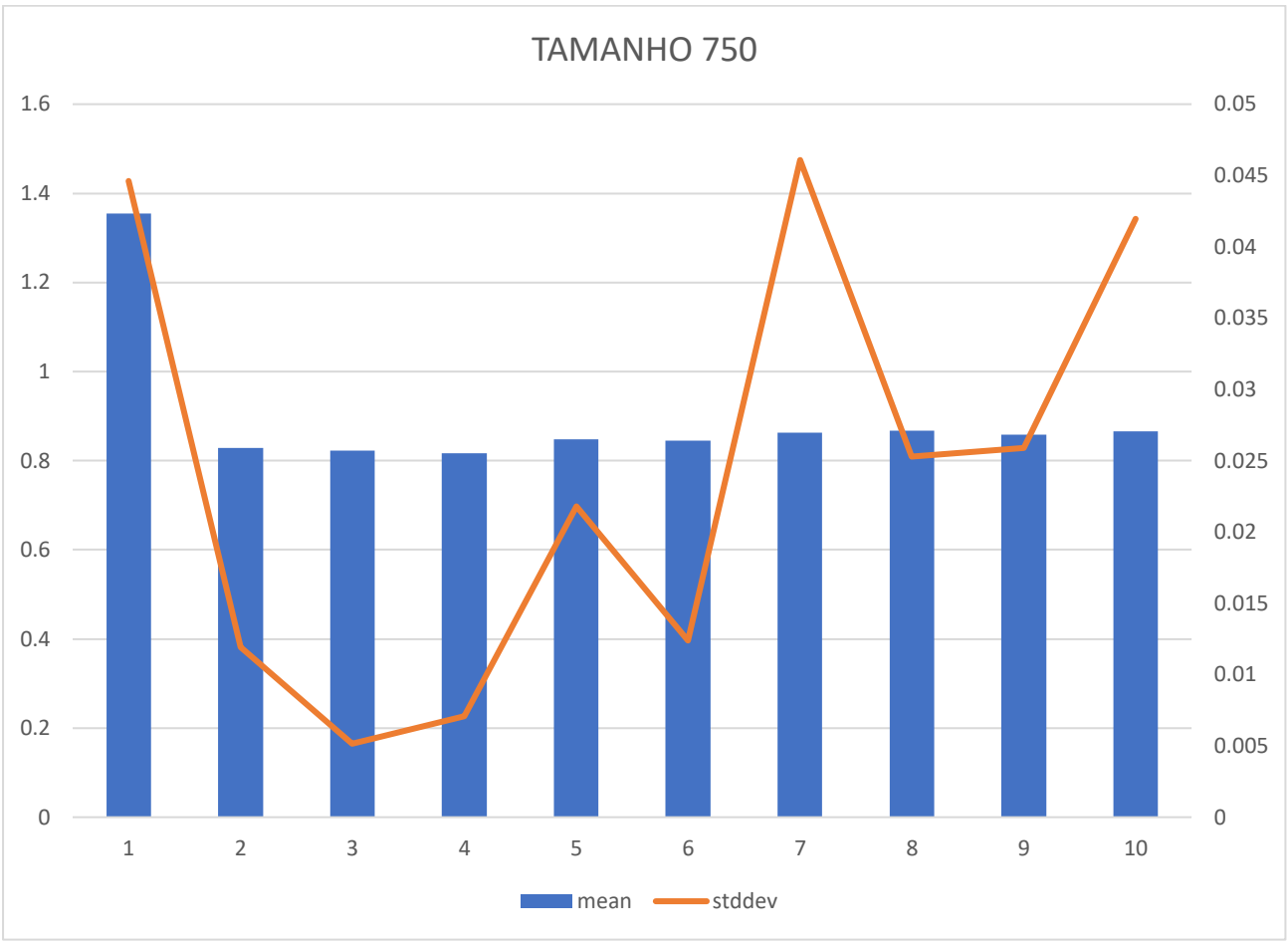
250 linhas x 250 colunas



500 linhas x 500 colunas



750 linhas x 750 colunas



1000 linhas x 1000 colunas