UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
CAMPUS NATAL CENTRAL
CURSO DE BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

ANÁLISE EMPÍRICA DOS ALGORITMOS DE ORDENAÇÃO

Esther Maria Da Silveira Wanderley
Thuanny Carvalho Rolim de Albuquerque

Natal-RN
Outubro, 2022

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
CAMPUS NATAL CENTRAL
CURSO DE BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

ANÁLISE EMPÍRICA DOS ALGORITMOS DE ORDENAÇÃO

Esther Maria Da Silveira Wanderley
Thuanny Carvalho Rolim de Albuquerque

Relatório técnico do desempenho do algoritmo sequencial, de processos e threads durante a multiplicação de matrizes.

Natal-RN
Outubro, 2022

Sumário

1 Introdução

O relatório propõe a análise de 3 algoritmos distintos na realização do produto de matrizes. Um dos algoritmos foi feito de maneira sequencial, onde é realizada a multiplicação de toda a matriz resultante sequencialmente, ou seja, uma de cada vez. O segundo algoritmo se utiliza de processos, onde cada vez que se calcula P elementos da matriz resultante é criado um processo, o que permite com que seja feita paralelamente o cálculo das outras linhas a partir da criação de outros processos. Este valor, P, é recebido pela linha de comando. O terceiro e último algoritmo deriva da criação de threads a cada P elementos a serem calculados, o que permite, também, a realização do cálculo paralelamente ao cálculo das próximas linhas.

A partir da observação dos gráficos e resultados coletados durante a execução dos códigos, será possível analisar e concluir qual dos três algoritmos apresentou um melhor desempenho para o cálculo de multiplicação de matrizes.

A matriz de maior tamanho foi decidida a partir do tempo de execução do algoritmo sequencial, que no caso foi de mais de 2 minutos na matriz $3200X3200$.

Após a obtenção dos resultados serão realizados os experimentos $1)$ e 2, que consiste em:
$1)$ analisar o comportamento de todos os três algoritmos dado um P na matriz $3200X3200$
$2)$ analisar o comportamento dos algoritmos paralelos a partir de um incremento/decremento de P.

Este trabalho está dividido em uma seção para cada algoritmo que explicará os resultados observados, contando com o algoritmo em pseudo-código, o gráfico e os experimentos 1 e 2.

2 Metodologia

2.1 Materiais utilizados

2.1.1 Computador

Computador	Especificações
Samsung 550XCJ/ 550XCR	Processador Intel (R) Core (TM) i3-10110U 2.59 GHz 2 x 4 GB DDR4 1333 MHz SSD 256 GB Intel (R) UHD Graphics

Para a implementação dos algoritmos foi usado o sistema operacional Linux na versão Ubuntu.

2.1.2 Ferramentas de programação

Os três algoritmos escolhidos foram implementados na linguagem C++, padrão ISO/IEC 14882:2017, ou simplesmente C++17.

Eles foram compilados utilizando o g++ pelo terminal do Linux com o comando:

g++ -Wall -std=c++17 nome_do_arquivo.cpp -o exec

A biblioteca *chrono* foi responsável pelas medições de tempo.

2.1.3 Algoritmos

2.1.3.1 Sequencial

A solução utilizada consiste na multiplicação de cada linha da matriz 1 por cada coluna da matriz 2. Cada multiplicação é feita sequencialmente, linha por linha

Algorithm 1: Sequencial

Input: Dois vetores de vetores de inteiros representando as matrizes M1 e M2 e três inteiros: um representando o número de linhas de M1 (m1_linhas); outro, o número de colunas de M1 e linhas de M2 (m1_col_m2_lin); e o terceiro, o número de colunas de M2 (m2_colunas).

Result: Um vetor de vetor de inteiros com o resultado da multiplicação

$resultado[m1\_linhas][m2\_colunas]$;
$linha[m2\_colunas]$;
for $i \leftarrow 0$ ***to*** $m1\_linhas - 1$ **do**
    $linha.clear()$;
    for $j \leftarrow 0$ ***to*** $m2\_colunas - 1$ **do**
        $soma \leftarrow 0$;
        for $k \leftarrow 0$ ***to*** $m1\_col\_m2\_lin - 1$ **do**
            $soma \leftarrow soma + (M1[i][k] * M2[k][j])$;
        end
        $linha[j] \leftarrow soma$;
    end
    $resultado[i] \leftarrow linha$;
end
return $result$;

2.1.3.2 Processos & Threads

A solução utilizada consiste na multiplicação de cada P elementos da matriz resultado. Cada multiplicação é feita paralelamente, seja com uso de processos ou de threads.

Algorithm 2: Processos & Threads

Input: Dois vetores de vetores de inteiros representando M1 e M2, um inteiro que representa o índice (idx) e uma string com o nome do arquivo em que iremos escrever o resultado (file_name).

Result: Um arquivo com parte do resultado da multiplicação

openWrite($file\_name$);

$j \leftarrow 0$;

while $j < (M1.size() * M2[0].size())$ ***and*** $idx < (M1.size() * M2[0].size())$ **do**

  $sum \leftarrow 0$;

  for $k \leftarrow 0$ ***to*** $M1[0].size()$ **do**

    $sum \leftarrow sum + (M1[idx/M1[0].size()][k] * M2[k][idx\%M2[0].size()])$;

  end

  $file\_name.Write(sum)$;

  $idx \leftarrow idx + 1$;

  $j \leftarrow j + 1$;

end

3 Desenvolvimento e resultados

A partir de agora, serão mostrados os resultados da análise dos dados obtidos em cada Experimento.

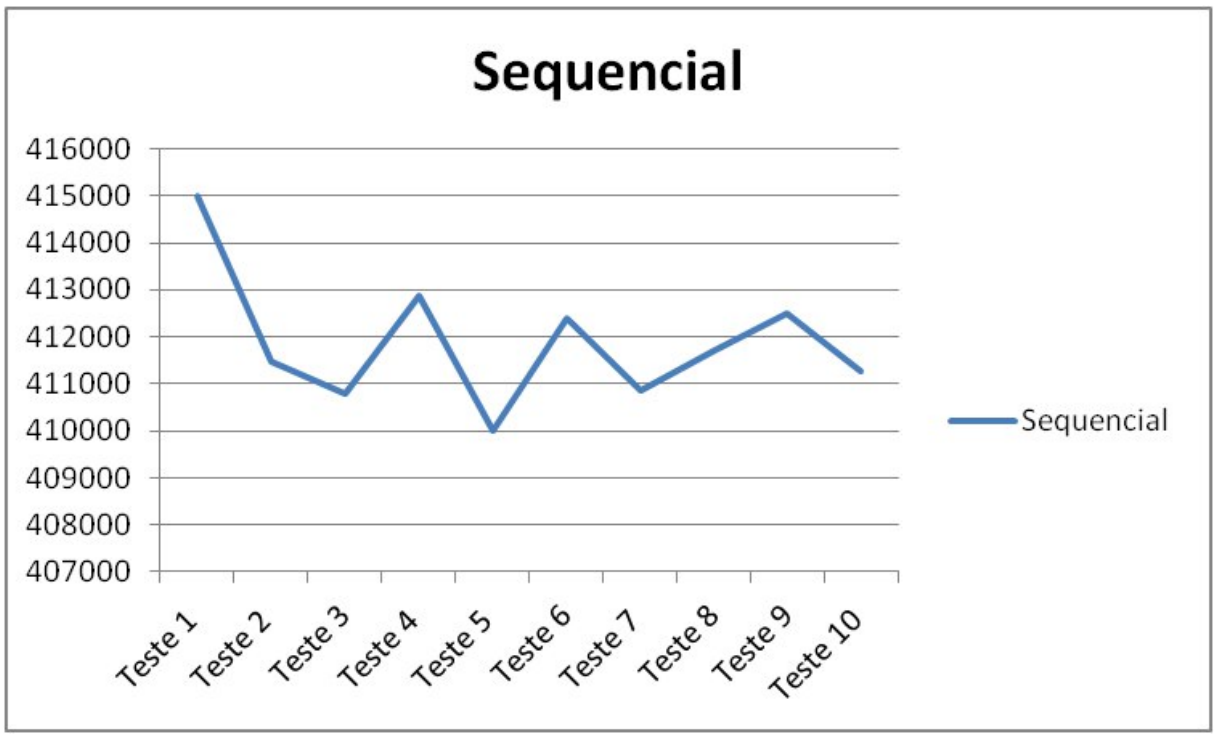
Cada algoritmo foi colocado em um mesmo cenário: multiplicação de duas matrizes com valores aleatórios de 1 a 100 e executado 10 vezes, a partir disso foi obtido o tempo de cada execução e o gráfico de seus comportamentos na matriz de 3.200 x 3.200.

Faremos a análise desses dados a partir dos gráficos que foram obtidos, utilizando-se o número de testes e o tempo de execução (que no caso das threads e dos processos é feito a partir do maior tempo de uma thread ou processo) como parâmetros para os gráficos.

Utilizaremos os dados e gráficos apresentados para cada experimento neste capítulo para a discussão no capítulo posterior.

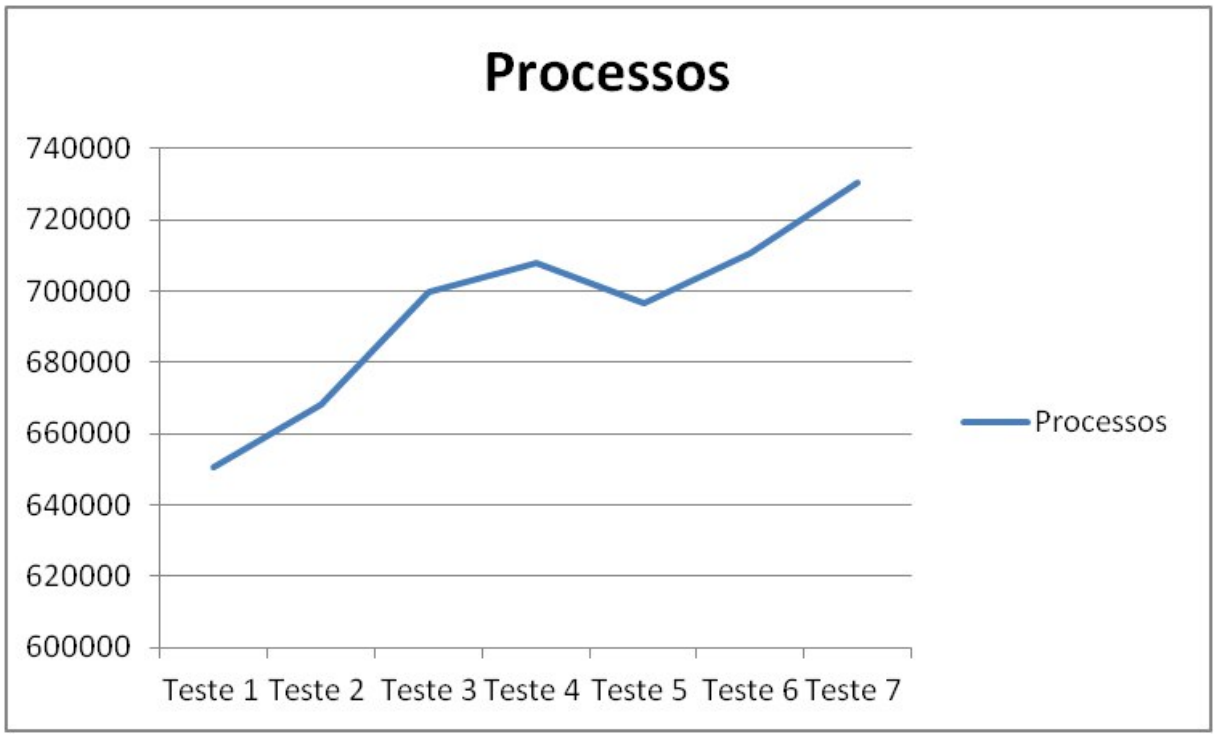
3.1 Experimento 1

3.1.1 Sequencial



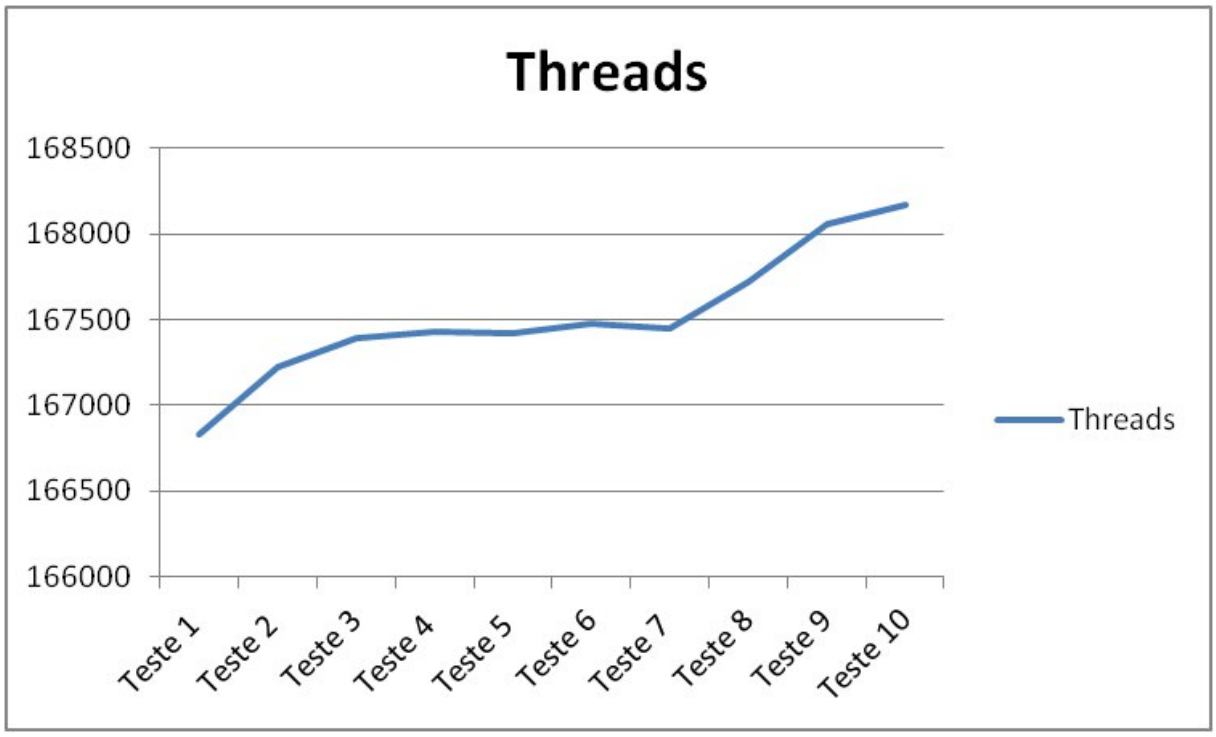
Comportamento do algoritmo sequencial na matriz 3200 x 3200

3.1.2 Processos



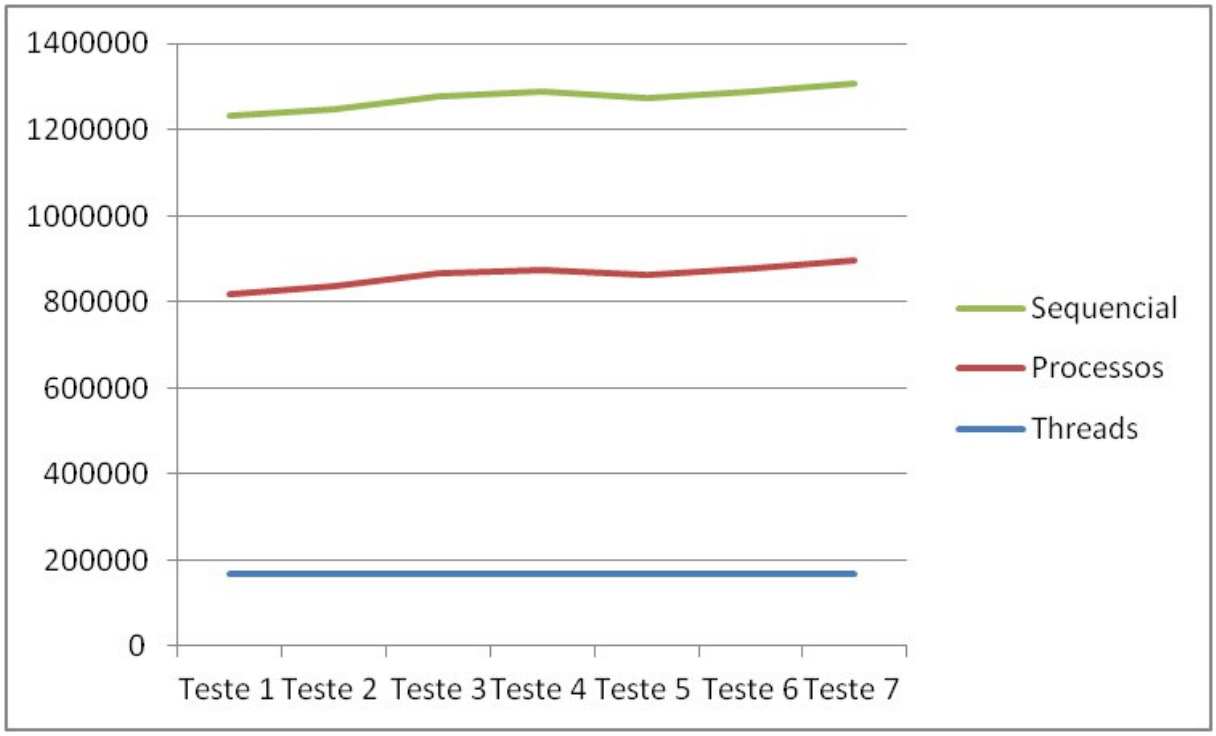
Comportamento do algoritmo de processos na matriz 3200 x 3200

3.1.3 Threads



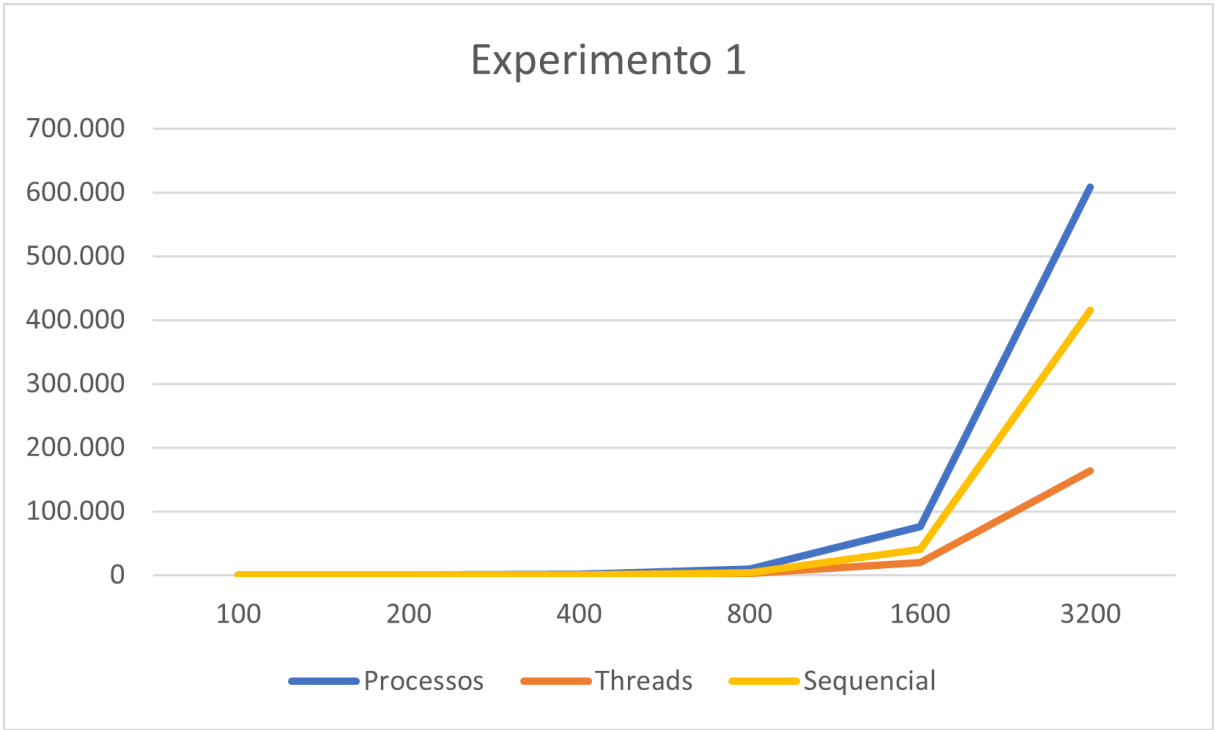
Comportamento do algoritmo de threads na matriz 3200 x 3200

3.1.4 Comparação entre os três



Comparação do comportamento dos três algoritmos na matriz 3200 x 3200

3.1.5 Comparação entre os três variando na dimensão

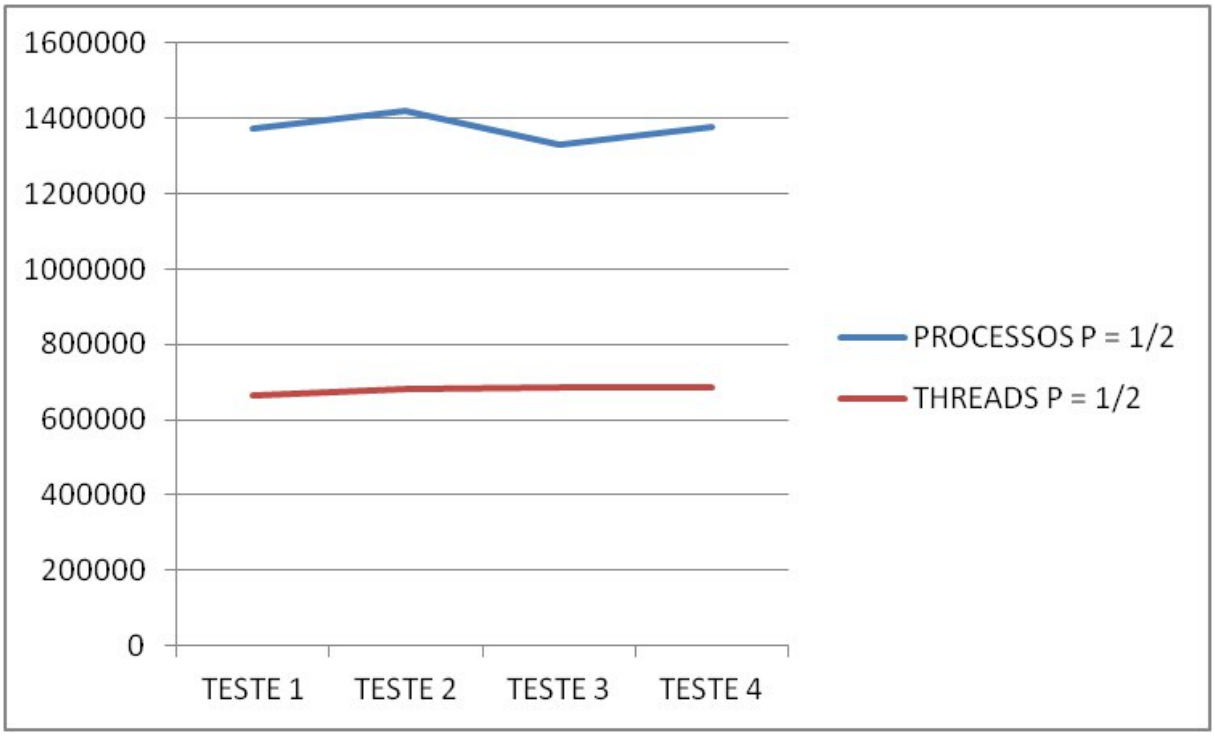


Comparação do comportamento dos três algoritmos em cada matriz

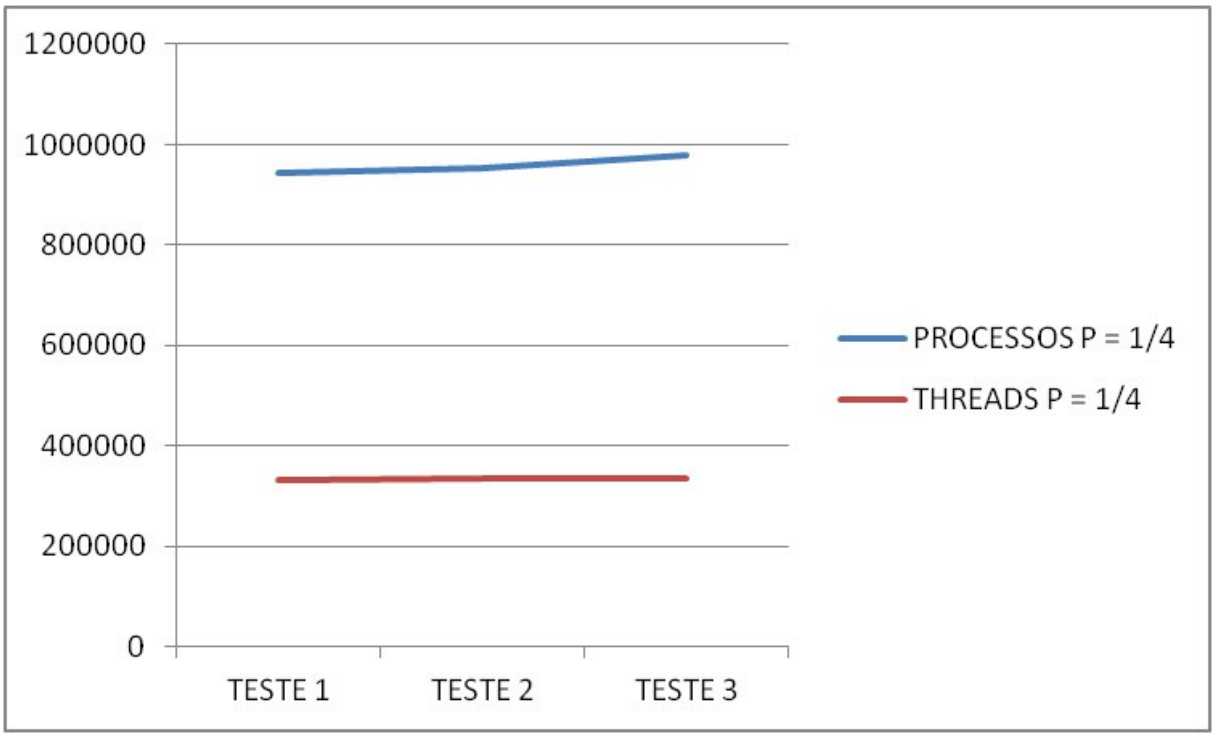
3.2 Experimento 2

Análise do Comportamento do algoritmo de Processos e Threads de acordo com o decremento de P

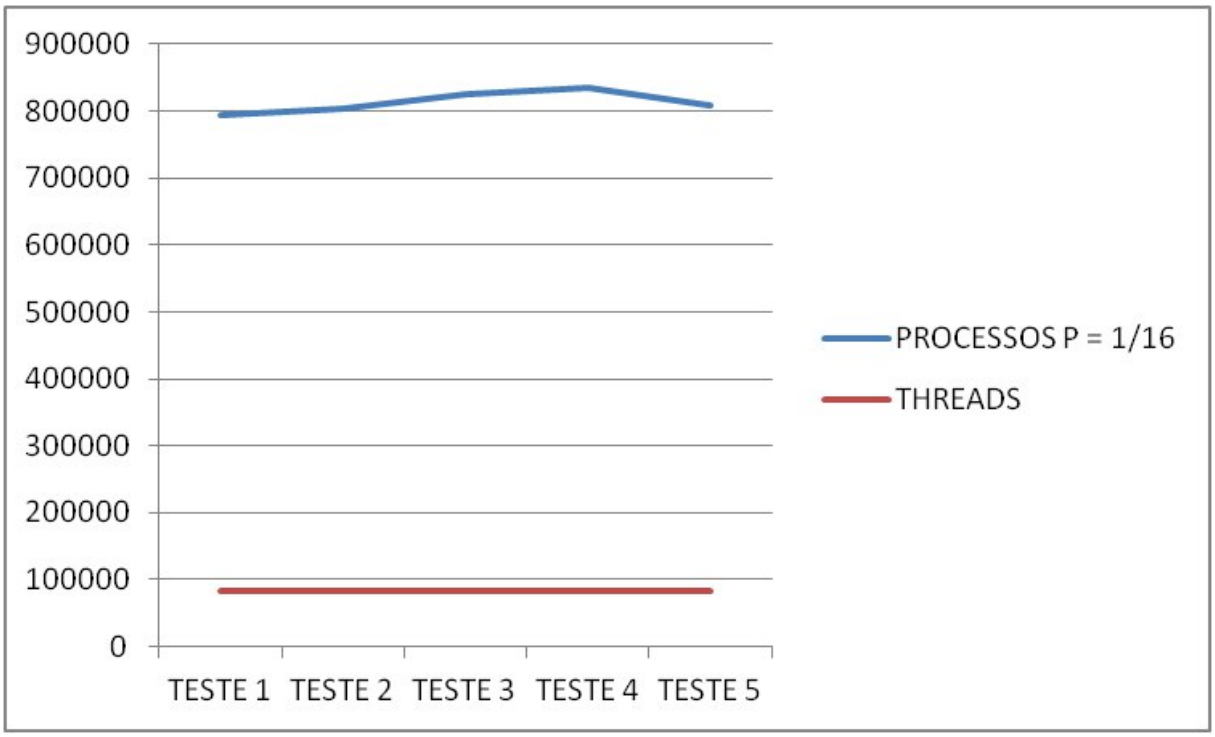
3.2.1 incremento = x4



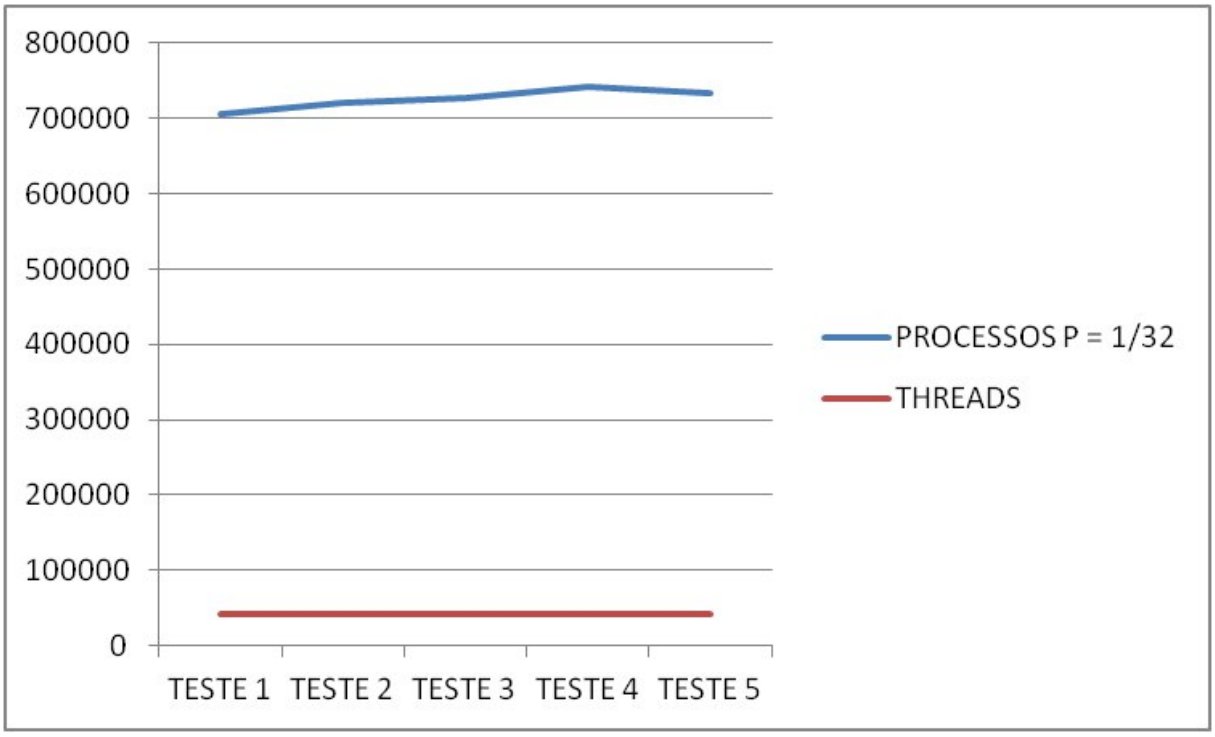
3.2.2 incremento = x2



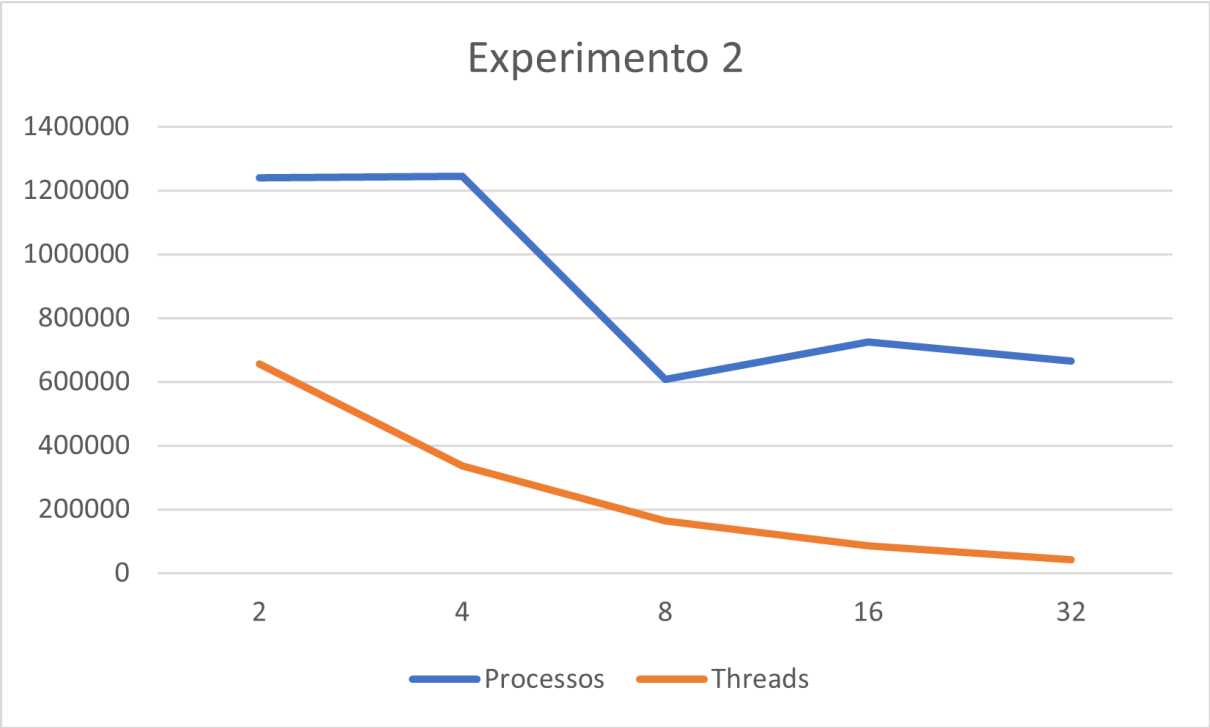
3.2.3 decremento = x1/2



3.2.4 decremento = x1/4



3.2.5 Variação de acordo com o número de threads/processos criados



4 Discussão

4.0.1 Conclusões dos dados obtidos

Para a discussão, são trazidos também alguns questionamentos. Apresentaremos, a seguir, algumas respostas:

a) Qual o motivo dos resultados obtidos no experimento E1? O que pode ter causado o comportamento observado?
RESPOSTA: Com a análise dos gráficos apresentados, é possível perceber que o sequencial obteve um desempenho inferior ao de processos e ao de threads, pois ele não é executado paralelamente; pelo contrário, é necessário esperar até o fim de um cálculo para começar o cálculo do próximo elemento.
Em contrapartida, observamos que o algoritmo executado com threads obteve um desempenho superior ao do algoritmo executado com processos. Isso se dá pelo fato de que a criação de uma thread é muito mais rápida do que a criação de um processo. A partir disso, observamos que a execução em paralelo é muito mais eficiente do que a execução em sequência, pois não há a necessidade de esperar pelo término de uma ação para assim começar a outra, elas são executadas ao mesmo tempo.

b) Qual o motivo dos resultados obtidos no experimento E2? O que pode ter causado o comportamento observado?
RESPOSTA:
Observa-se que o algoritmo de threads obteve um desempenho superior ao de Processos. Isso se dá pelo fato da criação de threads ser mais rápida que a criação de processos.

c) Qual é o valor de P ideal para a multiplicação das matrizes M1 e M2? Justifique sua resposta através dos experimentos realizados.
O valor de P ideal para a de processos seria $\frac{n_1 \cdot m_2}{8}$, onde n_1 é o número de linhas da Matriz 1 e m_2 é o número de colunas da Matriz 2, pois existe um equilíbrio entre o tempo de criação de cada processo e de cada processo e o tempo do cálculo de cada elemento que cada um iria fazer; no entanto, para as threads, há um custo tão baixo na criação da thread que é mais vantajoso P $= \frac{n_1 \cdot m_2}{32}$.