UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

RELATÓRIO DA Nº 01 EXPERIÊNCIA
TRABALHO PRÁTICO DA UNIDADE 1– PROCESSOS E THREADS

TURMA: T03

Leonardo David Galvão de Melo: 20210047151

Lucas Vinicius Gois Nogueira: 20210076805

Natal-RN
2022

Leonardo David Galvão de Melo: 20210047151

Lucas Vinicius Gois Nogueira: 20210076805

Threads vs. Processos

Primeiro relatório apresentado à disciplina de Sistemas Operacionais, correspondente ao trabalho da 1º unidade do semestre 2022.2 do 4º período do curso de Tecnologia da Informação da Universidade Federal do Rio Grande do Norte, sob orientação do **Prof. Gustavo Girao Barreto Da Silva.**

Professor: Gustavo Girao Barreto Da Silva.

Natal-RN

2022

RESUMO

Este relatório apresenta o processo e resultados de uma análise empírica da eficiência de programas com Threads, Processos e Sequenciais quando comparados um ao outro. Para tal foram implementados programas para realizar multiplicação de matrizes utilizando as três técnicas. Com isso, realizou-se a medição e plotagem de seus tempos de execução para matrizes com tamanhos variados. Os resultados e certos aspectos de interesse são discutidos finalmente.

Palavras-chave: threads; processos; sistemas; teste empírico

Sumário

1 INTRODUÇÃO

Até meados do século 20 os computadores só podiam executar uma tarefa por vez, chamados computadores monotarefa. Tais dispositivos limitam o usuário e as possibilidades para os desenvolvedores. Pois, por exemplo, quando um programa necessita esperar um dado, como algo que o usuário deve digitar, o processador permanece inativo até que essa atividade se encerre, sem que o usuário possa usufruir de outros programas ou da capacidade máxima de seu processador.

A solução para esse problema são os dispositivos multitarefa e o multiprocessamento. Ambos envolvem a máquina ter programas solicitando recursos do processador concorrentemente. Em outras palavras: ter mais de um programa executando por vez.

Há duas formas principais para o programador criar esse paralelismo em seu programa: as threads e os processos. A criação de novos processos envolve a duplicação do programa em seu estado atual, de onde os processos decidem o que farão ser feito individualmente. Já as threads, não duplicam o programa. Cada thread tem o mesmo contexto de software e compartilha o mesmo espaço de memória, com stacks individuais. Em C, a thread recebe o endereço de uma função em sua criação e encerra quando retorna (ou prematuramente).

Nesse trabalho será analisada empiricamente a performance dessas duas abordagens e da programação sequencial. Para tanto será implementado um programa para calcular a multiplicação de duas matrizes utilizando cada um desses métodos. O tempo de cada um será medido para diferentes tamanhos de entrada e posteriormente disposto e analisado.

2 METODOLOGIA

Para averiguar o comportamento das diferentes formas de processamento três programas foram elaborados para realizar a multiplicação de matrizes (Sequencial, Paralelo, Threads). Primeiramente diversos tamanhos de matrizes foram passados. Começando com 100x100 e multiplicando os tamanhos por dois até o tempo de cálculo do programa Sequencial passar de 2 minutos. Os programas paralelos foram executados com 8 instâncias de seus respectivos paralelismos. Para cada tamanho de entrada o programa foi executado 10 vezes para retirar a média.

Em um segundo momento a quantidade de paralelismos ideal foi buscada. Utilizando as matrizes máximas do teste anterior a quantidade de elementos calculados por instância foi reduzido 10 vezes até chegar em um quarto do valor inicial. Isso têm efeito direto na quantidade de paralelismo que será realizado, quanto menor o número de elementos calculados por instância, maior é o número de instâncias que são criadas. Depois o número de elementos foi aumentado até 4 vezes o valor original.

A tabela abaixo apresenta as principais especificações do computador utilizado nos testes.

Tabela 1: Especificações da máquina utilizada.

Componente	Especificações
Máquina	Tipo: Notebook Marca: Acer Produto: Aspire 3 Modelo: A315-23-R5DQ UEFI: Acer
Sistema	Kernel: 5.4.0-125-generic x86_64 Bits: 64 Desktop: Cinnamon 5.2.7 Distro: Mint 20.3 Cinnamon
CPU	Topologia: Dual Core Modelo: Ryzen 3 3250U Bits: 64
Memória:	5.8 GiB

3 RESULTADOS

3.1 Experimento 1

Como requisitado no Experimento 1, foi realizada a execução dos programas **Sequencial**, **Paralelo Threads** e **Paralelo Processos** com matrizes de um intervalo de $100x100$ até $3200x3200$. Para os programas paralelos, o cálculo de P foi obtido pela fórmula $\lceil \frac{n1*m2}{8} \rceil$. O gráfico obtido é exibido a seguir na Figura 1.

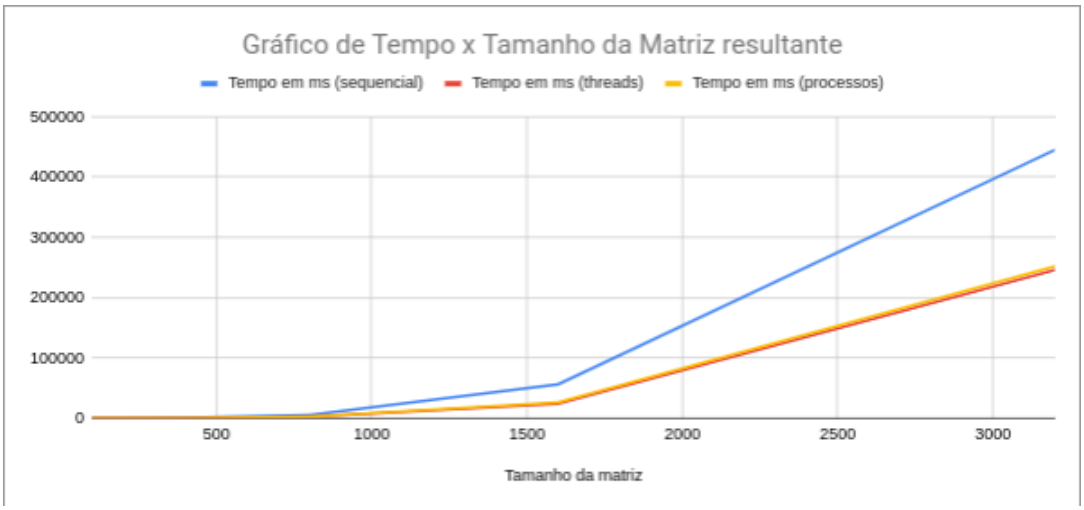

Figura 1: Gráfico de tempo em milissegundos em relação ao tamanho da matriz final.

3.2 Experimento 2

Para o Experimento 2, foram feitas duas análises, uma partindo do número de elementos como sendo $\lceil \frac{n1*m2}{8} \rceil$ e diminuindo-o até $\lceil \frac{n1*m2}{32} \rceil$, ou seja, uma sequência de decrementos, e a outra, uma sucessiva sequência de incrementos até $\lceil \frac{n1*m2}{2} \rceil$, isto é, quatro vezes o tamanho anterior.

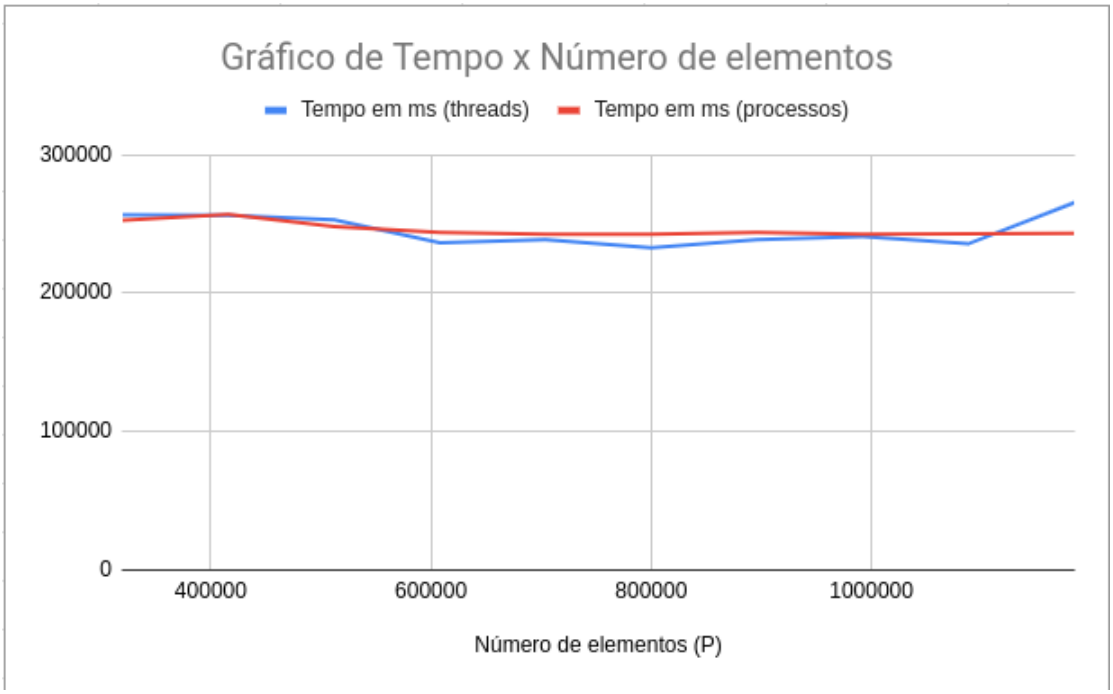

Figura 2: Gráfico de tempo em milissegundos em relação ao número de elementos (abordagem decremental).

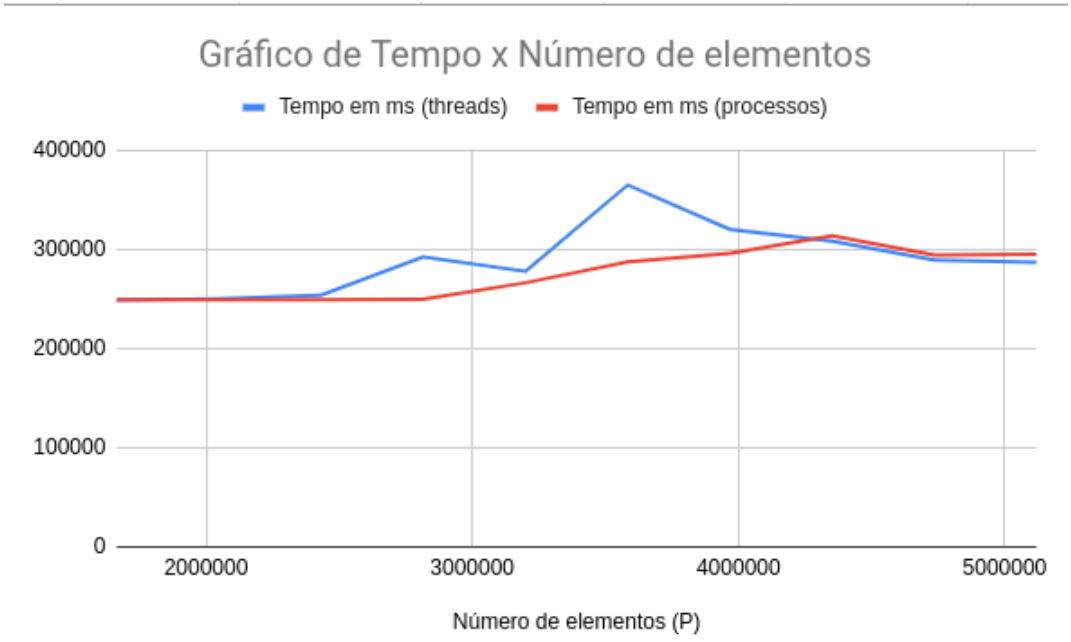

Figura 3: Gráfico de tempo em milissegundos em relação ao número de elementos (abordagem incremental).

4 DISCUSSÕES

4.1 a) Qual o motivo dos resultados obtidos no experimento E1? O que pode ter causado o comportamento observado?

A diferença entre os programas com paralelismo e o sequencial é clara. Enquanto os com paralelismo se aproveitam de mais de um núcleo de processamento da CPU, o sequencial só é capaz de utilizar um. Foi observado que o computador onde foram realizados os testes tem dois cores e o tempo foi reduzido aproximadamente pela metade nos testes que se utilizam de paralelismo.

Por outro lado não se observou grande diferença de tempo de execução entre as Threads e Processos. Os processos tendem a ser mais lentos pois ocorre a duplicação do programa, mas neste caso estamos lidando com um programa extremamente simples. Acarretando assim em impactos insignificativos nos tempos de execução.

A diferença de prioridade dada aos processos e às threads pode ser um fator relevante também. Pois certas variáveis como o processador, sistema operacional e a implementação do paralelismo na linguagem podem afetar o tempo de execução no processador.

4.2 b) Qual o motivo dos resultados obtidos no experimento E2? O que pode ter causado o comportamento observado?

Os gráficos do experimento 2 mostram uma leve vantagem para as threads em termos de tempo de execução quando se tratou da variação do número de elementos calculados, mas com a mesmas dimensões na matriz. Já os processos obtiveram uma ligeira vantagem nos outros gráficos. Uma das principais razões para isso é o fato de que a criação de threads se dá de forma menos custosa do que a criação de processos. Inclusive, embora os valores de tempo estejam melhores para threads, a diferença de valores não é grande.

4.3 c) Qual é o valor de P ideal para a multiplicação das matrizes M1 e M2? Justifique sua resposta através dos experimentos realizados.

De acordo com os experimentos o melhor tamanho de P foi 800000 gerando 13 threads/processos e resultou na execução mais rápida das threads e segundo melhor com os processos.

Inicialmente foi pensado que o número ideal de instâncias seria próximo do número de cores da máquina utilizada. Porém, como são realizadas muitas operações de I/O, que não envolvem o processador mas são demoradas, é possível que outras instâncias aproveitem esse tempo.

Outro possível fator relevante para esse valor ideal é encontrar o equilíbrio entre quantidade de elementos a ser calculado por thread/processo e o número de threads/processos. Valores de P muito altos geram tendências de comportamento do programa **Sequencial**. Já a criação de muitas threads e/ou processos também é algo custoso para o Sistema Operacional. Este valor de P foi capaz de entregar o menor tempo de execução para o **Paralelo Threads** e um dos menores para o **Paralelo Processos**, pois provavelmente possui um equilíbrio entre os pontos supracitados.

5 CONCLUSÃO

É possível observar, com base nos experimentos E1 e E2, que o programa **Sequencial** é muito limitado em termos de escala e gerenciamento de recursos do computador. Faz uso de apenas um núcleo do processador e, mesmo que seu programa possua algoritmos altamente eficientes, pode causar longas esperas por ter escassez de recurso ao seu dispor, diferentemente dos programas **Paralelo Processos** e **Paralelo Threads**.

A programação executada de forma paralela visa uma melhor distribuição de recursos para o seu programa, de modo que utilize de mais recursos do seu computador, garantindo maior eficiência em termos de tempo de execução.

Os tempos de execução para programação paralela foram bem próximos, apesar da ciência de que a criação de processos é mais cara do que a criação de threads. Devido à simplicidade do problema, é plausível o que pode-se observar nos resultados.