HEITOR MAURO CHAVEZ HUARACHI

RELATÓRIO DE SISTEMAS OPERACIONAIS

BAGÉ-RIO GRANDE DO SUL

2023

https://github.com/elcabriton/SO-PI

1. INTRODUÇÃO

O trabalho tem como objetivo apresentar alguns conceitos de threads e algumas informações da máquina operada pelo software.

Todo o código desenvolvido ao decorrer deste trabalho pode ser conferido no github.

2. HARDWARE EXECUTADO

O trabalho foi executado em um desktop de mesa, onde suas especificações estão listadas abaixo:

- Processador ryzen 5 3400g (4cores e 8 threads)
- Placa Mãe B450
- Memoria ram 32gb 3200mhz ddr4
- SSD 500gb

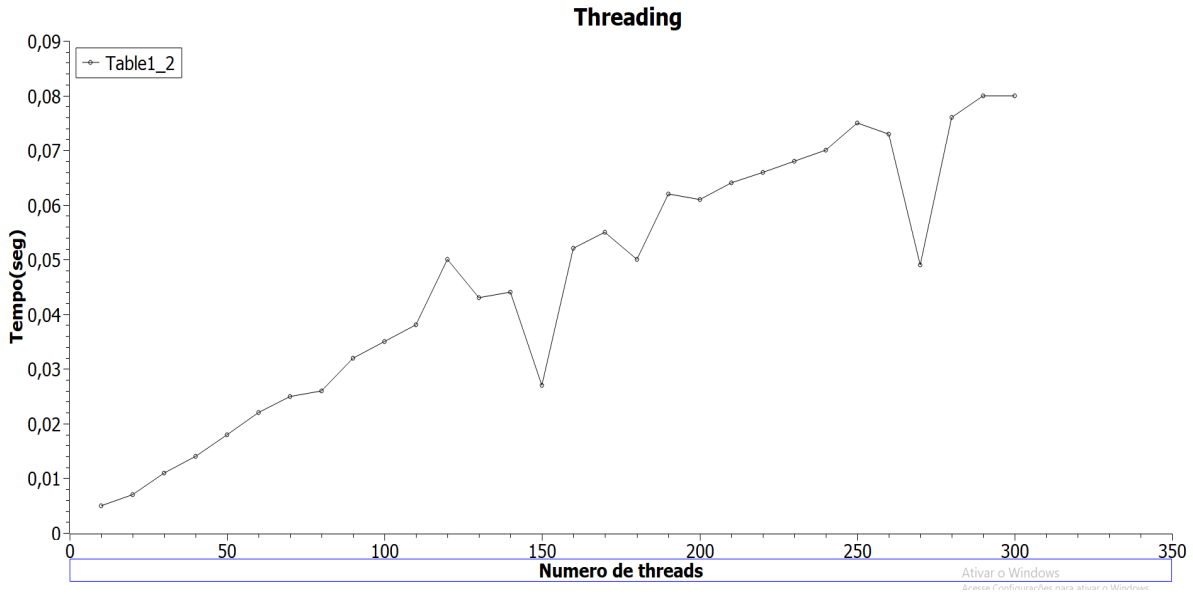
Com isso demos progresso ao código e testando a capacidade e tempo em que executa o programa criado com os desejos do professor.

3. Gráfico de desempenho

Para fazer o desempenho do código foi utilizado duas ferramentas para fazer esse processo com conhecimentos de laboratório de física 1, utilizamos um software chamado “Scidavis” para criação do gráfico, junto com o comando “htop” de linux, onde é mostrado todas as informações necessárias para definir o trabalho com as threads criadas, a cache e a quantidade de memória RAM utilizada. Foram desenvolvidas duas versões de código: a primeira utilizando a biblioteca “multiprocessing”, e a outra utilizando “threading” para ser feita uma comparação entre as duas.

3.1. THREADING

Essa biblioteca de Python ela executa como se fosse uma thread mesmo, pegando uma parte da memória, e se dividindo para concluir o que lhe foi salientado. Dito isto podemos ver como ficou o trabalho no gráfico abaixo.



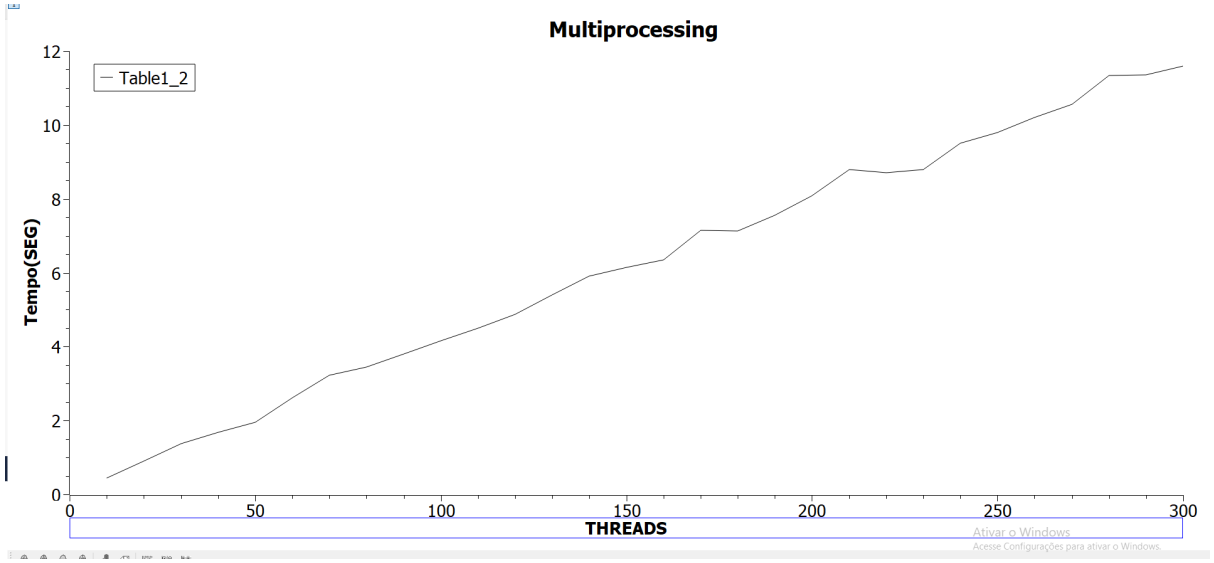
1. Gráfico Threading

Como podemos ver analisando o gráfico acima, vemos muita mudança repentina no tempo, mesmo aumentando o número de threads e a quantidade a mais de interações para chegar no PI, aumentando e diminuindo, mesmo com esse número de threads crescendo para ajudar a finalizar a contagem.

Com isso podemos dizer que essa “biblioteca threading” somente simula uma thread não ajudando no tempo de execução do arquivo em quase nada, apenas de uso acadêmico servindo.

3.2. MULTIPROCESSING

O multiprocessing ele faz criar vários processos, para executar um devido arquivo, sendo possível gerenciar o número de processos que irá ser criado, isso simula muito bem como funciona threads na teoria, com isso damos inicio ao nosso gráfico. Lembrando que, por conta de abrir muitos processos, acaba abrindo muitas GUI(interface grafico), e para o tempo ser exato e necessário fechar todas as GUI aberta para que me de os resultados, sendo assim utilizamos o arquivo pelo terminal mesmo, mas o trabalho final será entregue com a GUI, como apresentado ao professor.



2. Gráfico Multiprocessing

Como se pode ver no gráfico quanto mais threads mais desempenho ganha o código, fazendo aumentar extremamente pouco o tempo, conforme as threads aumentam, isso para um código grande faz extrema diferença, com isso ele acaba ganhando muito desempenho. Além disso, se acredita que quanto mais threads forem colocados além do máximo do gráfico, vai haver um pouco de desempenho.

4. CONCLUSÃO

Podemos concluir com este trabalho na primeira abordagem que na “figura 2”, houve maior desempenho conforme mais tempo vai aumentando, pois o tempo sobe extremamente pouco, além de não haver divergência de tempo como na figura 1, onde aquela forma de uso, não acaba sendo na eficaz pois, não te dá resultados sólidos para um trabalho ou projeto.

Uma dificuldade que foi encontrada ao adaptar o código para receber a GUI, foi que conforme o programa fosse criando novas threads, novas interfaces gráficas eram abertas, matando o desempenho do processador e forçando a reinicialização do computador. Isso foi driblado ao adotar a saída pelo terminal, pois assim não há abertura de várias GUI, mas no trabalho estará as duas formas de código, lembrando que para utilizar a “multiprocessing” deve se fechar todas as GUIS para que dê o resultado final, porém isso implica no tempo,pois enquanto essas abas estiverem abertas, mais tempo vai ser perdido e isso contará na contagem do programa. Além disso alguns momentos no windows por abrir muitos arquivos

acaba dando erro, isso foi driblado no linux utilizando o comando “unlimit -n 100000” onde me deixa abrir esse número exorbitante de arquivos por vez, permutando o programa rodar.