UNIVERSIDADE FEDERAL DO MARANHÃO
ALUNO: THIAGO CHAVES MENEZES
MATRÍCULA:
PROFESSOR: FRANCISCO JOSE DA SILVA E SILVA
COCOM: COORDENAÇÃO DE COMPUTAÇÃO.
SISTEMAS OPERACIONAIS

São Luís – MA

RELATÓRIO:

INTRODUÇÃO

Em primeiro contato com a implementação do código do trabalho apresentado na disciplina de Sistemas Operacionais, tive que me ater primeiramente com a geração das função de criação das matrizes randomizadas e as listas que guarda as coordenadas dos números primos. Sem encontrar muitas dificuldades nessas implementações, iniciei a criação da função responsável por retornar se o número encontrado é primo ou não.

No início, a função de achar o número primo usava um custo computacional muito grande pelo fato de percorrer dentro de for, onde era testado se o resto da divisão pelo contador seria igual a zero até chegar no valor testado menos um. Seu custo computacional era alto e fazia com que a execução do programa com matriz de ordem 10000(dez mil) demorasse mais do que o aceitável.

A implementação da threads foi o maior desafio encontrado até então pela falta de compatibilidade com Sistema Operacional Windows e sua biblioteca "pthread" contém uma quantidade relevante de funções com parâmetros confusos. A pesquisa feita para encontrar a funções necessárias para o projeto levaram tempo e consulta com colegas de sala que fizeram e conseguiram encontrar as funções úteis.

Para resolução da implementação da “pthread” no projeto, tive que fazer um dual boot para utilizar S.O. baseado em linux com kde da GNU ubuntu 20.04. No ubuntu, a instalação do gcc é feita apenas com uma linha de comando no terminal que ajuda bastante os programadores. Em seguida reutilizei o código já implementado e utilizei algumas funções necessárias para o projeto.

Uma outra facilidade muito útil encontrada foi o comando para criar o executável junto com o Makefile que permite criar o executável apenas com uma linha de comando.

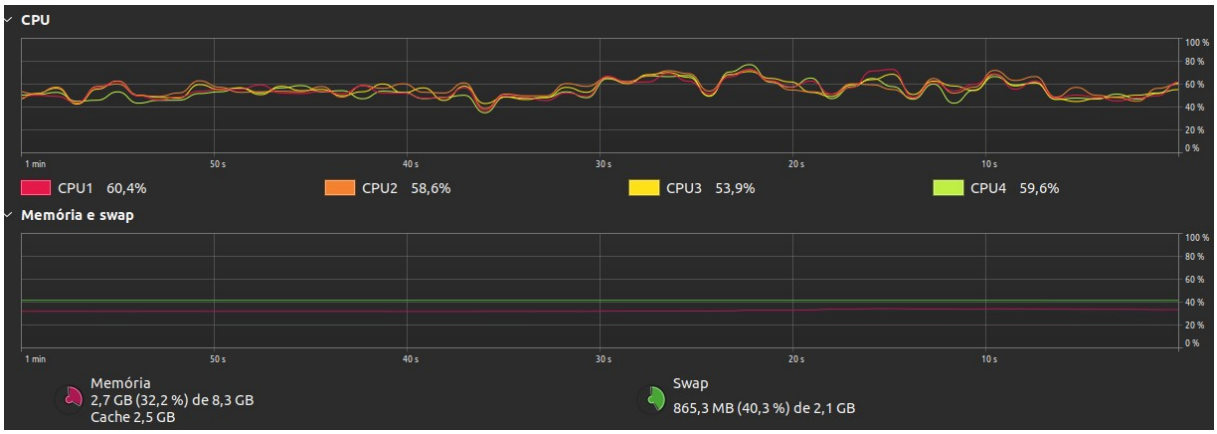
Iniciando fase de teste:

Para começar farei testes com single-thread utilizando matrizes quadradas de ordem de 1000(mil), 10000(dez mil), 20000(vinte mil), 25000(vinte e cinco mil). Em seguida utilizarei testes com multi-threads utilizando matrizes de mesma ordem.

ESPECIFICAÇÕES:

CPU: intel i5 7500
Memória RAM: 8GB
HD: 1TB
SSD: 240GB
SO dualboot Windows 10/Ubuntu 20.04

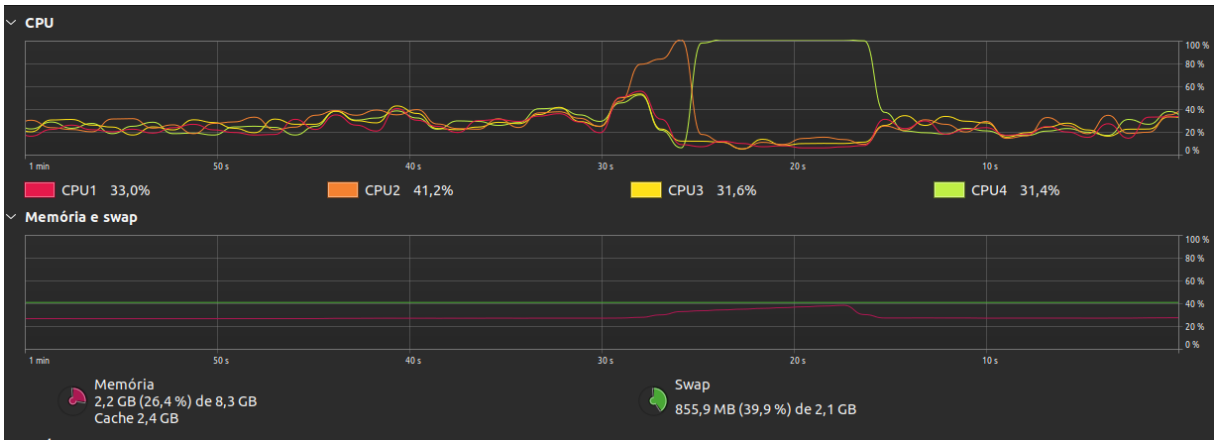
Single-thread matriz de ordem 1000:

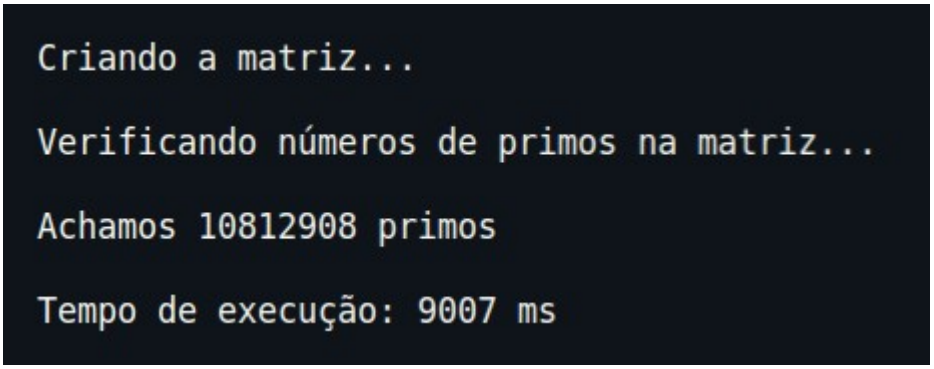



```
Criando a matriz...

Verificando números de primos na matriz...

Achamos 107620 primos

Tempo de execução: 101 ms
```

- O uso da thread e o uso de memória para um matriz de ordem de 1000 linhas e colunas não deixa perceptível qualquer alteração em ambas. O gráfico da CPU possui somente alterações casuais de usos de programas externos. O tempo de execução é quase que imperceptível.

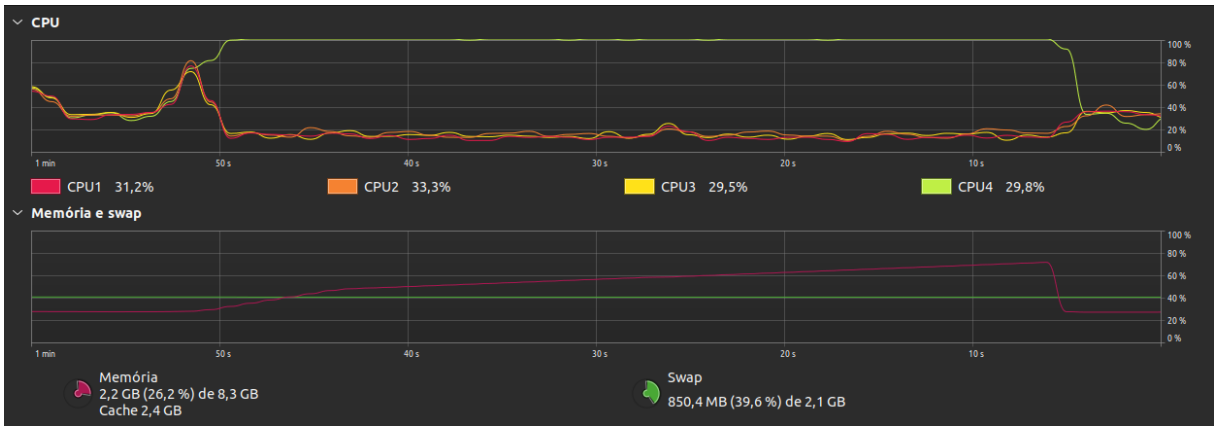
Single-thread matriz de ordem 10000:

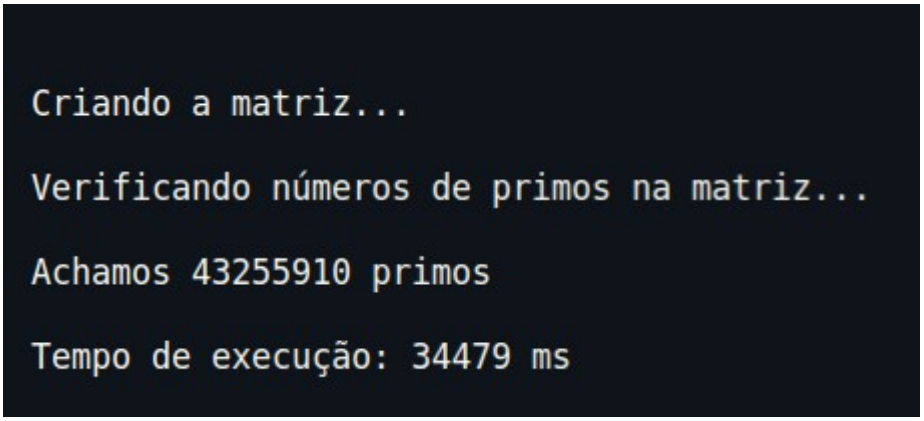




- A parti de uma matriz de 10x mais linhas e colunas já começamos a notar uma pequena alteração no uso da memória. E podemos ver também o uso do núcleo 2 e 4 passam por um tempo maior de uso. O núcleo 2 possivelmente foi responsável pela criação das matrizes e o núcleo 4 fez a separação dos números primos e das suas coordenadas na lista. Tempo de execução é quase 90 vezes maior que no primeiro caso.

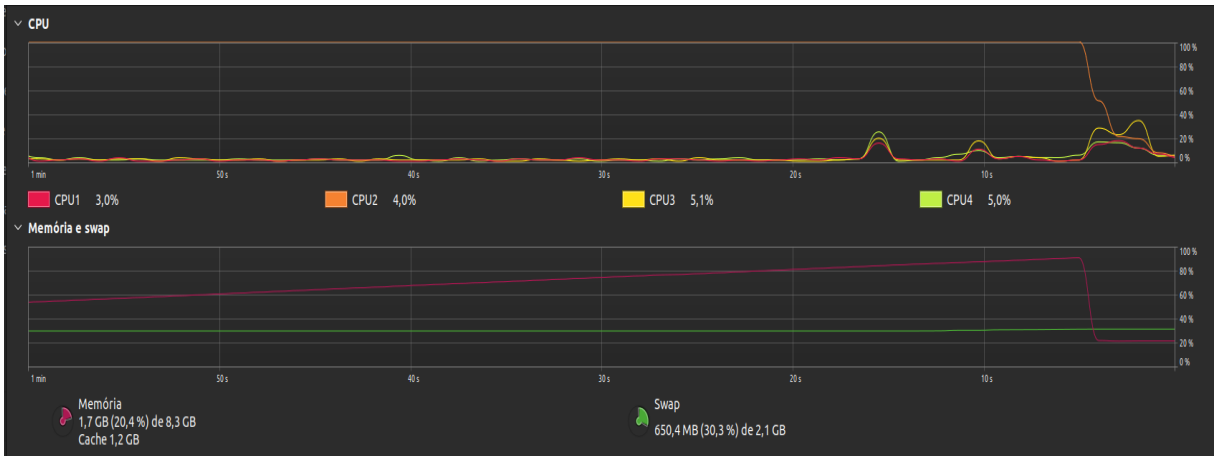
Single-thread matriz de ordem 20000:

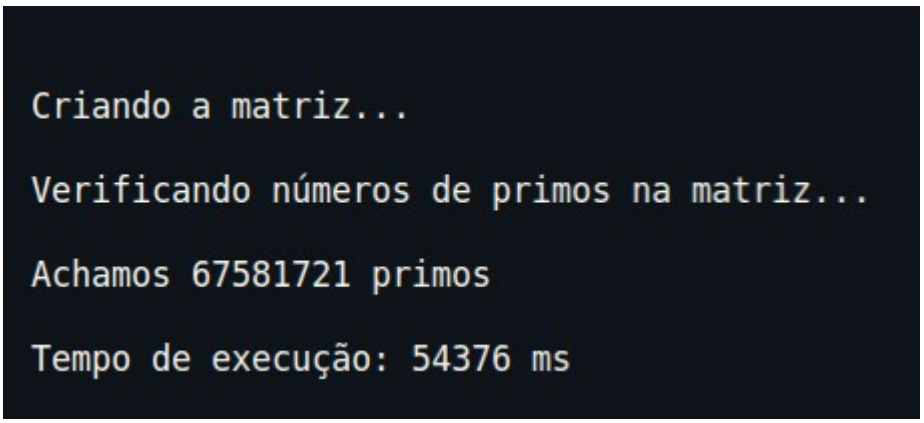




- Com a matriz de ordem 20000, a memória chega em 70% de uso, mostrando que o sistema começa a perceber uma dificuldade em armazenar a quantidade de número primos e suas respectivas coordenadas. O tempo de execução fica 4 vezes maior à matriz anterior e estende bastante o uso do núcleo 4 da CPU.

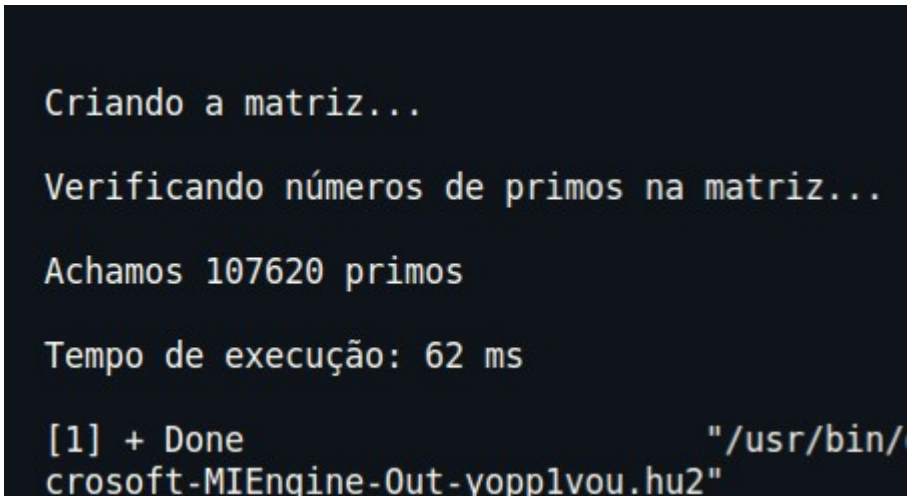
Single-thread matriz de ordem 25000:





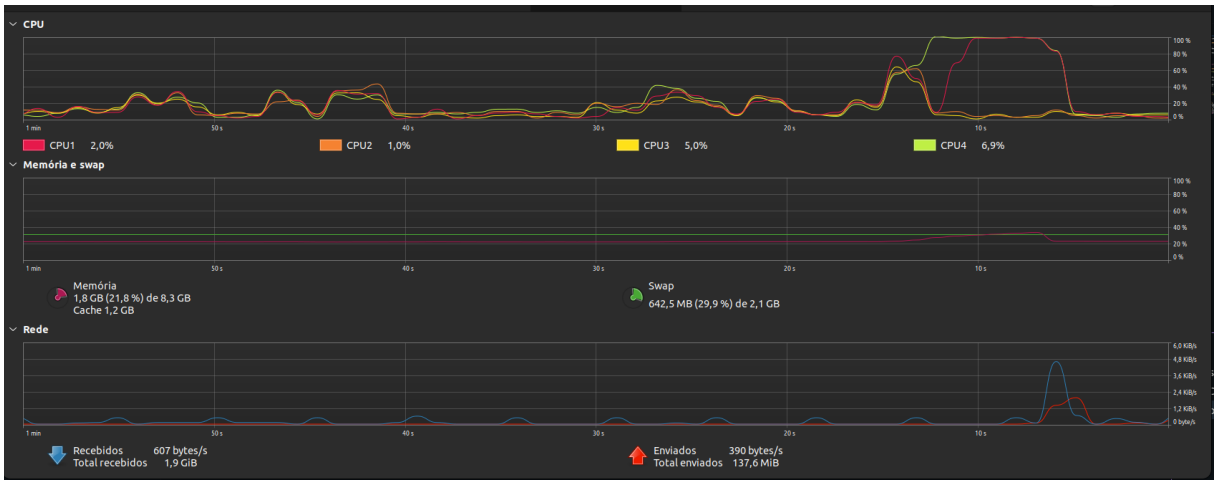
-Nesse momento podemos perceber que a matriz de ordem 25000 fez quase o uso máximo de memória da máquina em questão, mostrando que é o tamanho limite para a execução com uma memória de RAM de 8GB. O uso excessivo do núcleo 2 da CPU e da memória RAM fez com os outros núcleos hibernassem. Caso utilizasse outra operações paralelas na máquina enquanto o programa estivesse em execução, toda a operação seria comprometida fazendo com que a CPU “matasse” o programa.

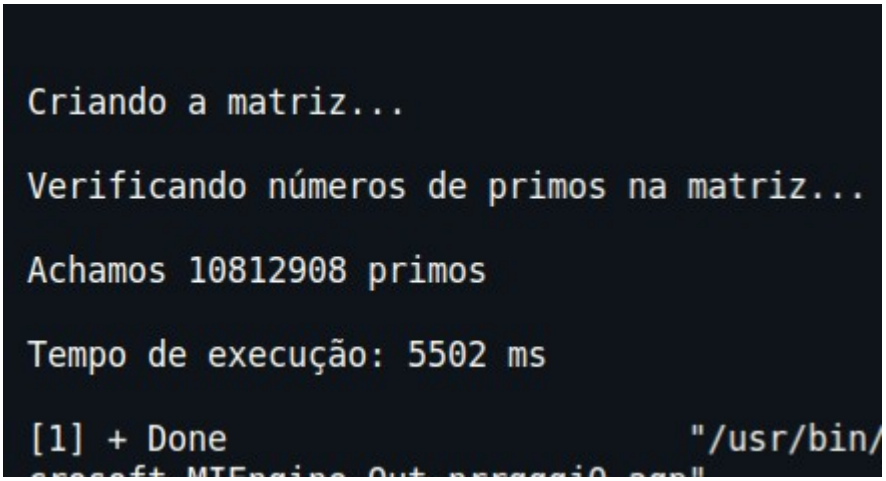
Multi-threads matriz de ordem 1000:



- A matriz de ordem 1000 continua não demonstrando nenhuma alteração significativa na memória. Porém nesse momento demonstra um pico de alteração no uso da CPU no momento de execução, mas não demonstra que que algo devidamente relacionado ao programa executado. O tempo em comparação a single-threads é relativamente bom mesmo com apenas 29ms de diferença.

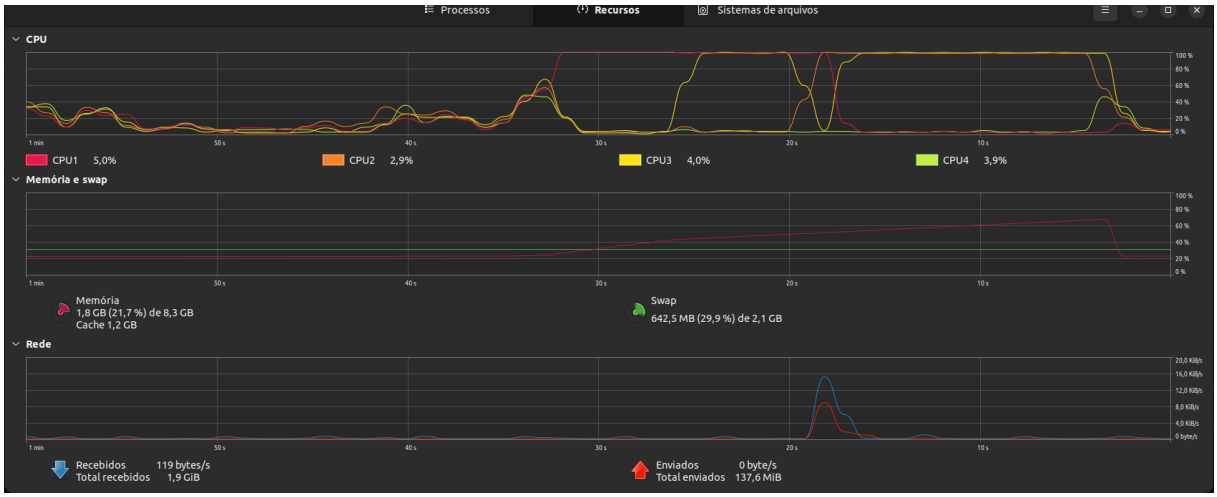
Multi-threads matriz de ordem 10000:

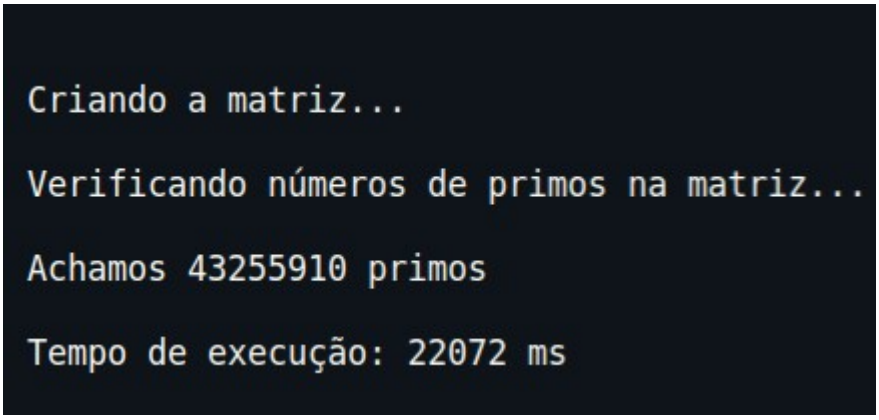




- Podemos perceber uma alteração efêmera no comportamento da memória e da CPU, demonstrando dessa vez o uso da duas threads utilizadas agora no programa. O tempo de execução em comparação ao single threads é notável com ganho de 4 segundos, sendo assim, perceptível a eficácia do uso de multi-threads no programa.

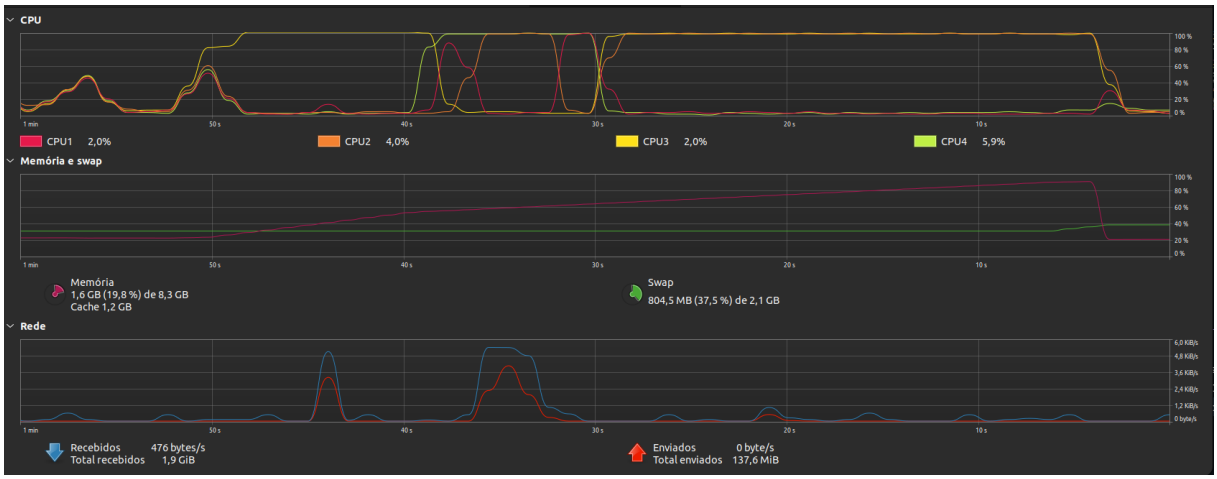
Multi-threads matriz de ordem 20000:

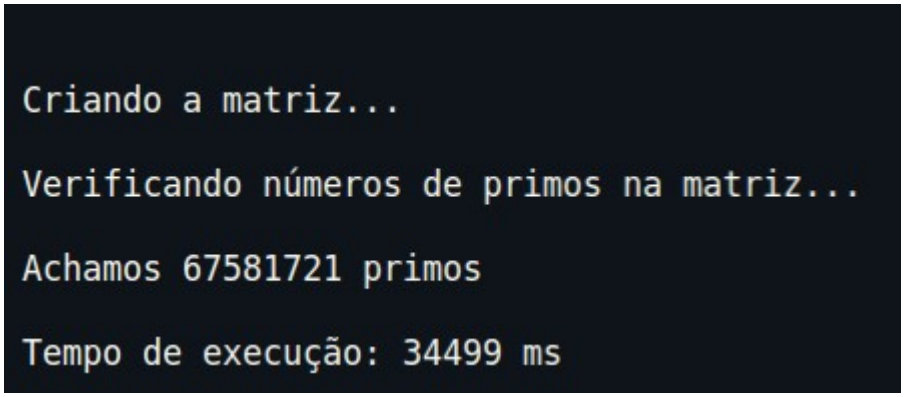




- Na matriz de ordem 20000 utilizando multi - threads podemos perceber novamente o uso de 70% da memória, porém temos tempo de execução de 12 seg a menos que o uso de single – threads, mostrando que o uso da memória sempre será o mesmo e utilizar multi – threads apenas melhorá o tempo de execução. E notável também a alternação no uso dos núcleos da GPU.

Multi-threads matriz de ordem 25000:





- Notamos novamente um ganho significativo no tempo de execução, porém o uso da memória continua no limite mostrando que a matriz ainda continua sendo o limite aceitável da memória.

DIFICULDADE ENCONTRADAS:

Em primeiro momento na implementação do trabalho não tive muitas dificuldades em criar as funções responsáveis pela alocação de lista e das matrizes. No atrito inicial foi o custo computacional utilizado pela função de encontrar um número primo. Então foi necessário fazer uma pesquisa para diminuir o número de iterações necessárias para a execução da função. A segunda observação importante a declarar é que o Sistema Operacional Windows não possui suporte nem acessibilidade para execução de programas em C, nem compatibilidade com a biblioteca “pthread” dificultando o interesse de programação nesse sistema. O uso da biblioteca “pthread” foi um dos maiores atritos pois possui uma quantidade de funções extensiva e os parâmetros utilizados nessa projeto foi de relativa complexidade.

O QUE APRENDI COM ESSE TRABALHO:

Com o trabalho ao todo, fazendo uma observação entre single – thread e multi – threads eu aprendi que o uso das threads faz uma diferença significativa no desempenho do programa. O programa ganha velocidade em execução, mas nos põem a observar que o uso da memória continua sendo a mesma no single quanto no multi. Assim, mesmo utilizando uma matriz de ordem de 30000 e fazendo uso das 4 threads disponíveis a CPU iria “matar” o processo por exceder a quantidade de memória utilizada para armazenar os número primos e suas coordenadas.