Projeto 2 -- "Build matrix" - Grupo Cronus

Membros:

- Henrique de Freitas Andreoli – 091415
- Vinicius Soares Brito da Silva – 245447
- Tiago Oliveira Mendes – 244849

Repositório Git: https://github.com/hfandreoli/ProjetoSO

Descrição da Solução

Para realizar a geração da matriz de forma ordenada e dividida entre múltiplas threads o programa monta um vetor com os números inteiros lidos dos arquivos de entradas e divide em partes iguais para serem processadas pelas threads. As threads fazem a ordenação da parte que lhes compete utilizando a técnica *insertion sort,* quando todas as threads finalizam a ordenação parcial, a thread do número 0 faz a ordenação final do vetor utilizando a técnica de *merge sort* quanto esta finaliza a ordenação final, o vetor novamente é divido entre as threads para a montagem da matriz que será gravada do arquivo de saída. O tempo de execução começa a ser contado antes da criação das threads e a contagem termina após o fim de execução das threads.

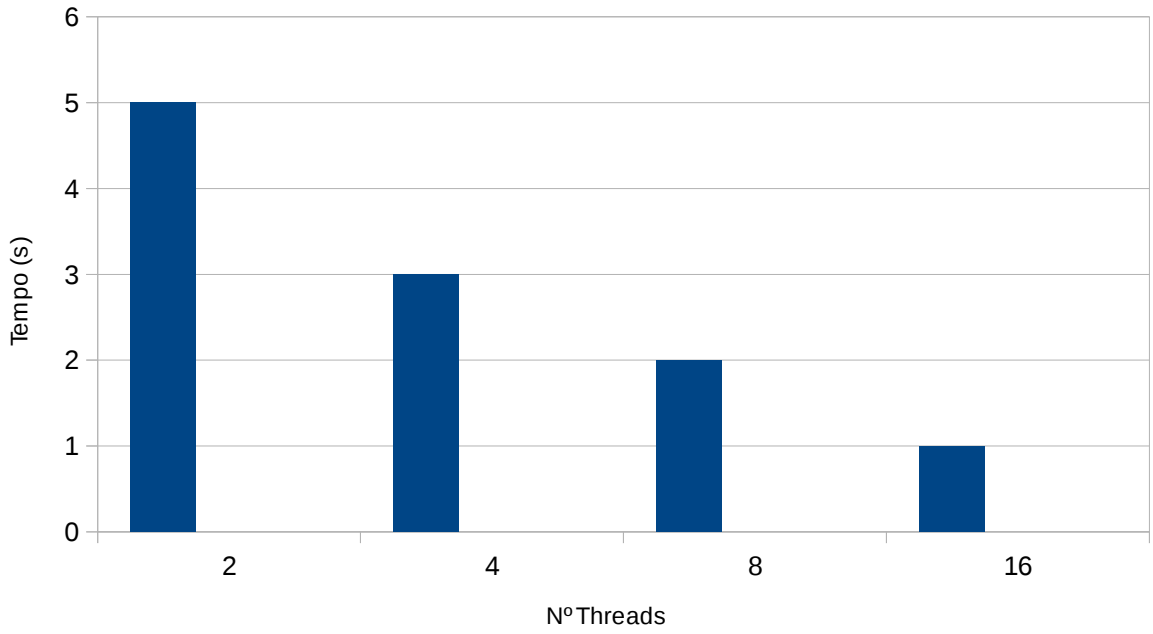
Inscrições para Compilar

```
gcc -pthreads trabalho.c -o trabalho
```

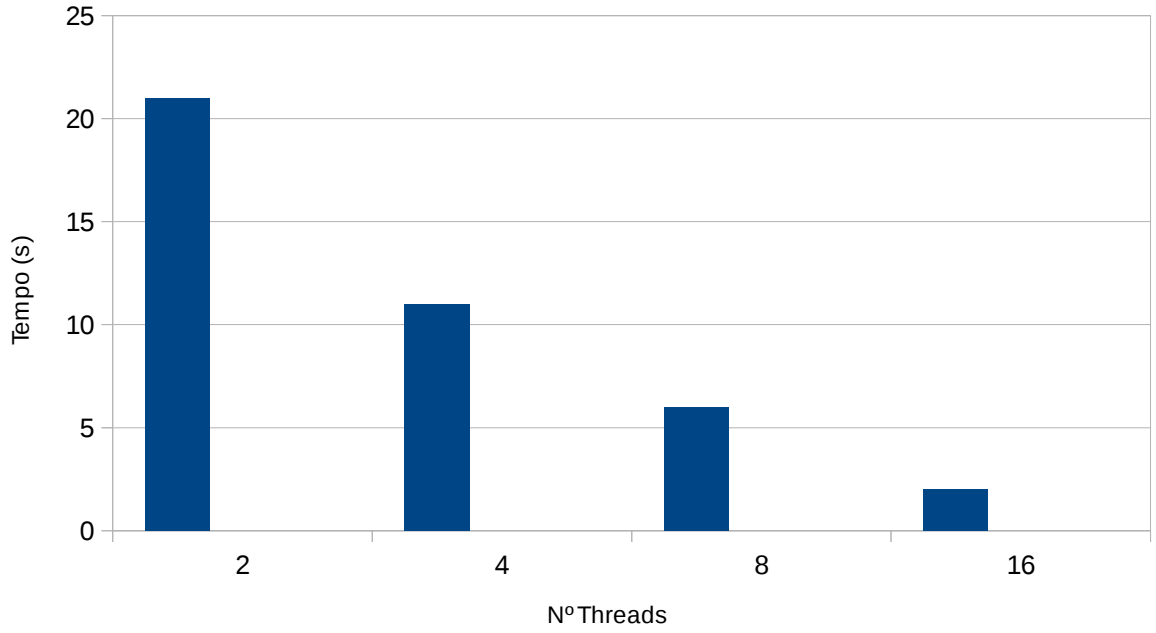
Execução

A seguir alguns resultados obtidos da execução com diferentes parâmetros

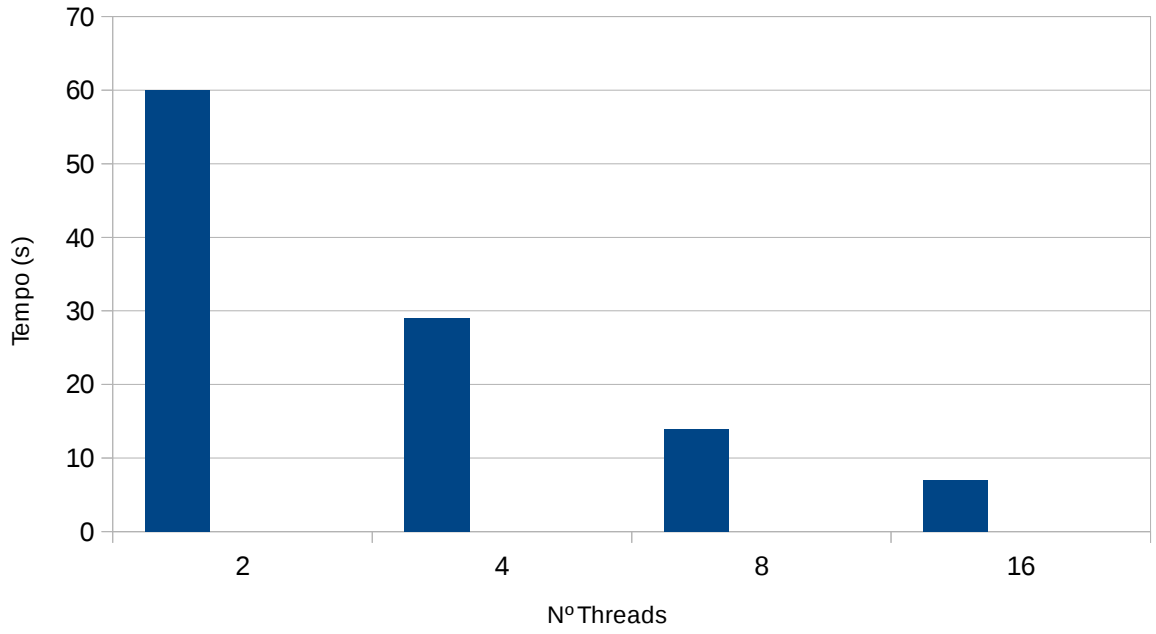
12 arquivos com 10000 números cada



16 arquivos com 15000 números cada



20 arquivos com 20000 números cada



Conclusões

Em relação aos resultados obtidos, é possível perceber que ao se executar diversos arquivos que contém vários números inteiros, com várias quantidades de Threads, sendo 2, 4, 8 ou 16 Threads, os valores que dizem respeito ao tempo de execução variam, ou seja, quanto mais threads são utilizadas, mais rápido o programa é executado, ou seja, leva menos tempo para que todos os valores dos arquivos gerados, sejam executados, ordenados e alocados no diretório.

Em alguns casos, dependendo do número total de valores utilizados nos testes, o tempo tende a reduzir sempre pela metade, até que chegue a 0, o que nem sempre acontece pois, por mais que tenha 16 threads em funcionamento, ainda não é totalmente suficiente a ponto de executar tudo em 0 segundos. Quanto maior a quantidade de valores a serem executados, mais tempo leva para que a execução seja realizada, e quanto mais threads, menor será o tempo em questão, pois o processamento estaria sendo divido em várias partes teoricamente iguais, cada uma para um núcleo, aumentando o desempenho e a eficiência da leitura dos dados.

Houve casos também, que por mais que 16 threads sejam bastante a ponto de agilizar o processo, dependendo da quantidade de números a serem analisados e processados, o sistema sofre uma falha, mais precisamente no terminal, ao tentar executar um valor alto, com 2, 4, 8 ou 16 Threads, o sistema simplesmente não roda, e apresenta uma falha de segmentação.