UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE TECNOLOGIA

RELATÓRIO

Filipe Bernardo 197054
Lucas Watanuki 202143
Rodrigo Zanco 205541

Limeira
2018

Projeto “Localiza na Matriz”

Grupo Miguezeiros:
Filipe S. Bernardo - 197054
Lucas Watanuki - 202143
Rodrigo M. Zanco - 205541

➔ **Repositório GitHub: https://github.com/lucaswatanuki/Miguezeiros-Projeto1**
➔ **Vídeo no Google Drive: https://drive.google.com/file/d/1_fLo_8Drn_qODf1cXg6CptJ18HAhifly/view?usp=sharing**

1. Descrição da Solução do Problema (Algoritmo em alto nível)

Algoritmo

1. Incluir bibliotecas necessárias;
2. Declarar variáveis;
3. Ler arquivo contendo dados;
4. Ler parâmetros matriz;
5. Ler numero de threads a serem utilizadas;
6. Ler elemento a ser encontrado na matriz;
7. Verifica se arquivo é válido/existe, se não, volta ao passo 3;
8. Cria matriz com base nos parâmetros do passo 4;
9. Leitura dos elementos que estão no arquivo e transposição para a matriz criada no passo 8;
10. Disparo das Threads e procura elemento entre as linhas;
11. Printa as posições do elemento procurado dentro da matriz, se existir, ou retorna erro se não existir.

Fim Algoritmo

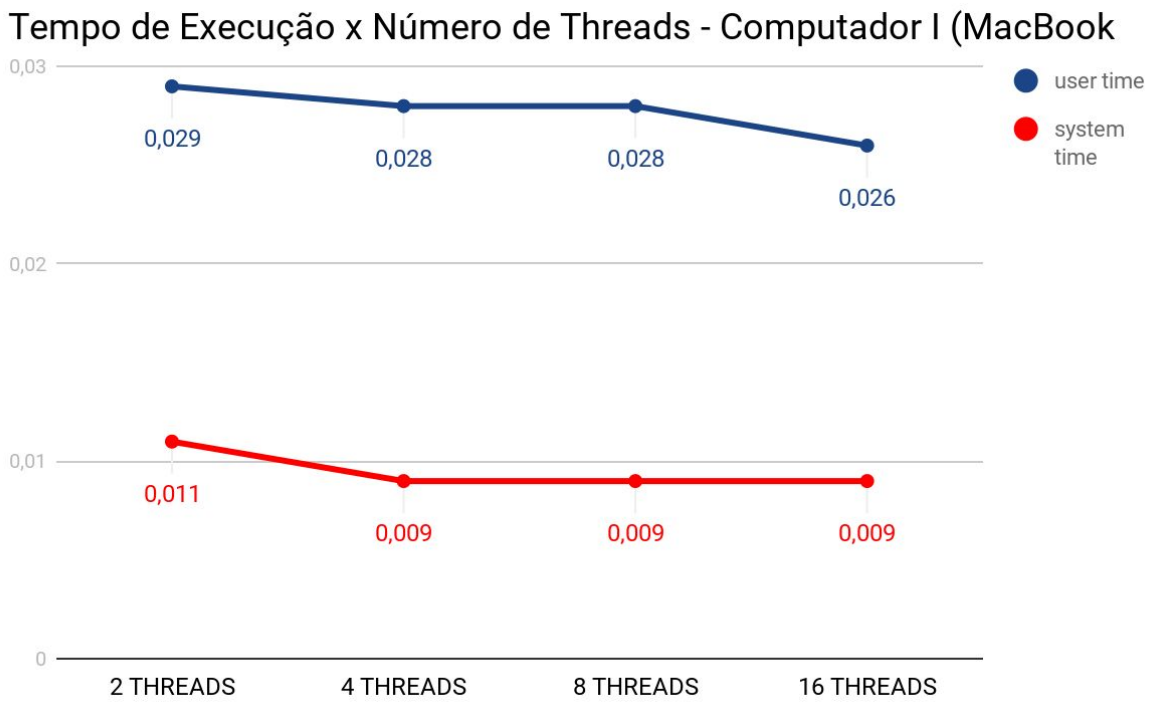
2. Instruções para Compilação e Execução (Linux)

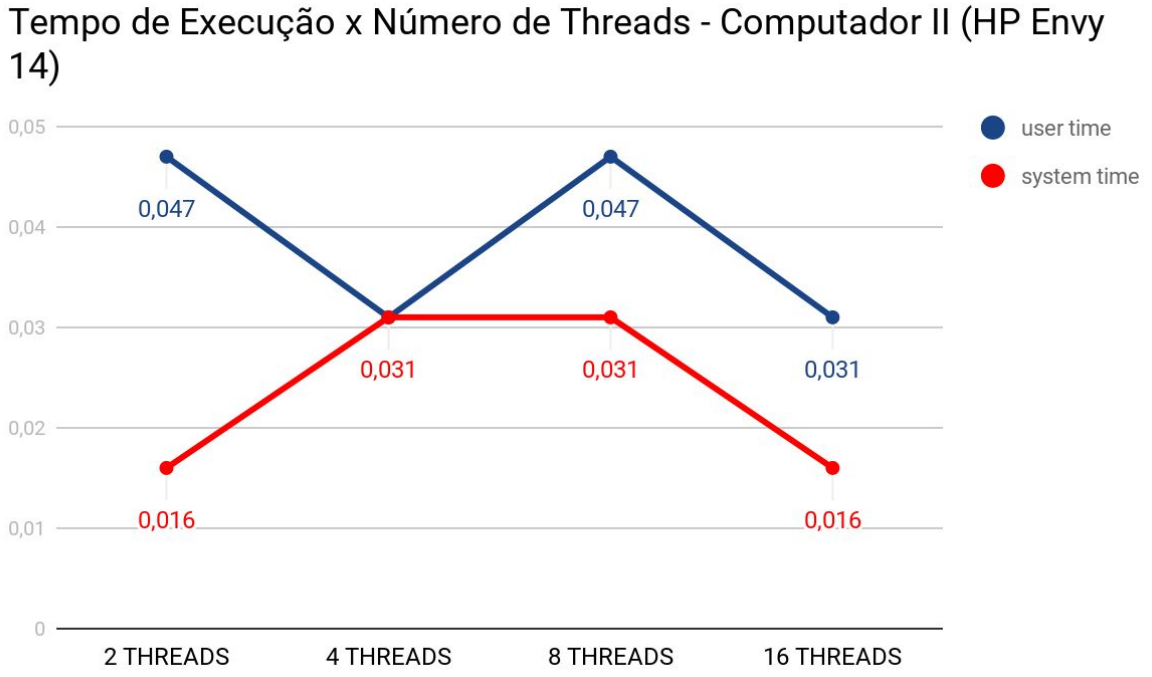
Utilizando o sistema operacional Linux, distribuição Ubuntu 18.04, utilizamos o terminal para a compilação e execução do projeto desenvolvido. Para tal, foi necessário utilizar as seguintes instruções:

i) Compilação: gcc projeto.c -o projeto -lpthread
*Observe que na instrução acima foi necessário o uso do argumento “-lpthread”, já que estamos trabalhando com a biblioteca POSIX.

ii) Execução: time ./projeto
*No comando acima, utilizamos o argumento “time” para obter os tempos de execução do programa.

3. Gráficos de tempo de execução por número de Threads e resultados obtidos





Computador I - MacBook Pro - Intel Core i5 (7th Generation Dual Core
Computador II - HP Envy 14 - Intel Core i5 (Dual Core) 1.7Ghz

Para montagem dos gráficos, executamos o programa junto ao comando “time”, e procuramos o valor “9” em uma matriz 300 x 300, utilizando 2, 4, 8 e 16 threads; os testes foram realizados em 2 computadores de configurações bem diferentes, porém com sistemas operacionais baseados no UNIX, sendo MacOS X e Linux Ubuntu 18.04 LTS, para efeito de comparação.

Como resultado do Computador I obtivemos um decréscimo de tempo enquanto o número de threads aumentava, entretanto esse decréscimo é muito baixo e o consideramos praticamente nulo. Uma vez que realizando os testes diversas vezes, não pudemos identificar um padrão de redução do tempo significativa relacionado ao aumento do número de Threads.

No Computador II pudemos observar que o tempo de execução para o usuário é maior que o Computador I, atribuímos este fato principalmente ao processador e diferença de "idade" das duas máquinas, sendo a primeira, mais nova. De mesma forma observamos uma pequena mudança no tempo, mas novamente sem padrão.

4. Conclusão

Após realizar todos os testes e obter os resultados comentados anteriormente, podemos concluir que ao executar esse "pequeno" código, é perceptível uma pequena influência do número de Threads em relação ao tempo. Essa pequena influência mostra-se melhor no exemplo do Computador I, no qual aumentando o número de Threads, o tempo tende a diminuir. Pensamos que se o programa fosse mais complexo e mais demorado, como um software de computador bem mais desenvolvido, as threads participariam bem mais na redução do tempo. Entretanto, após estudar mais o assunto, também precisamos ressaltar a importância do processador relacionado ao desempenho das Threads. Se o processador for de apenas um núcleo, talvez utilizar diversas threads seja mais custoso em termos de tempo do que utilizar apenas uma thread.