UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE TECNOLOGIA

Sistemas Operacionais

Projeto “Localiza na Matriz”

Grupo : CoffeeLovers

Nomes:

João Victor Mathias Netto 199831
Keyth Kazumi Kanashiro 200669
Samuel Lahr Azorli 205698

Limeira

2018

Sumário

❏ Descrição do problema e Objetivo

Considerando uma matriz M x N (M linhas, N colunas) formada por valores em ponto flutuante, positivos ou negativos de números aleatórios, não ordenados e talvez repetidos, o programa desenvolvido ira que utilizar múltiplas threads para procurar um determinado valor na matriz e a partir dos resultados realizar uma análise do desempenho do programa com 2, 4, 8, e 16 threads a partir de gráficos .

❏ Descrição da solução do problema:

Primeiramente o programa irá lê o arquivo e armazenará a matrix na memória usando uma estrutura de índice e uma outra com a linha, então o número de threads informadas serão criadas com a instrução de busca(o algoritmo que crias as threads são um loop que roda ‘n’ vezes, sendo ‘n’ o número de threads desejadas).

A função “buscar” faz com que cada thread inicie em uma determinada linha(por exemplo a thread 2 inicia na linha 2) procure o valor e então pule a quantidade de threads criadas, assim prevenindo que uma linha seja lida por mais de uma thread.

Após as threads terminarem a busca elas se juntam e o resultado é mostrado, sendo uma mensagem dizendo que encontrou e informando a posição, uma que não existe na matriz e uma default de tratamento.

❏ Instrução para a compilação

As instruções para a compilação do programa no sistema operacional Linux seguem as seguintes etapas:

1. Primeiramente para compilar o programa é necessário a instalação da biblioteca POSIX threads, o que pode ser realizado em sistemas ubuntu com o seguinte comando:

   sudo apt-get install libpthread-stubs0-dev

2. Após a instalação da biblioteca, vamos compilar o programa com o seguinte comando:

   gcc -pthread <nomedoprograma.c> -o nome

   Ex: gcc -pthread SOprojeto.c -o so

3. Pegar o arquivo que contém a matriz NxM, que será gerada aleatoriamente a partir da execução de um programa que o professor disponibilizou no diretório no github da disciplina e copiar um número qualquer para a realização do teste do programa.

   Ex: 2574284576,739285

4. Volte para o terminal e insira os seguintes comandos e tecle enter:

   ./nome qtd_linhas_do_arquivo qtd_colunas_do_arquivo num_threads valor_buscado nome_arquivo

   Ex: ./so 1000 1000 2 2574284576,739285 matrix.txt

5. A saída do programa será o seguinte:

   Criando a thread numero: 1….
   Criando a thread numero: n

   Obs: a saída dessa parte se altera de acordo com o num_thread escolhido

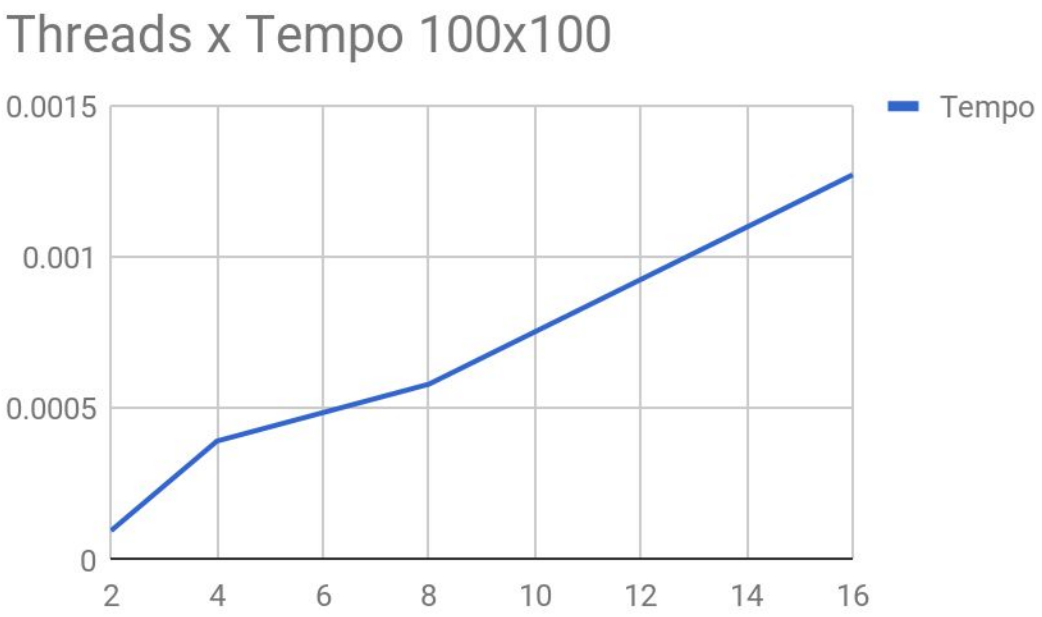
   Busca terminada, tempo gasto : xxx.xx s
   Linha: x, Coluna: y

❏ Gráficos da análises:

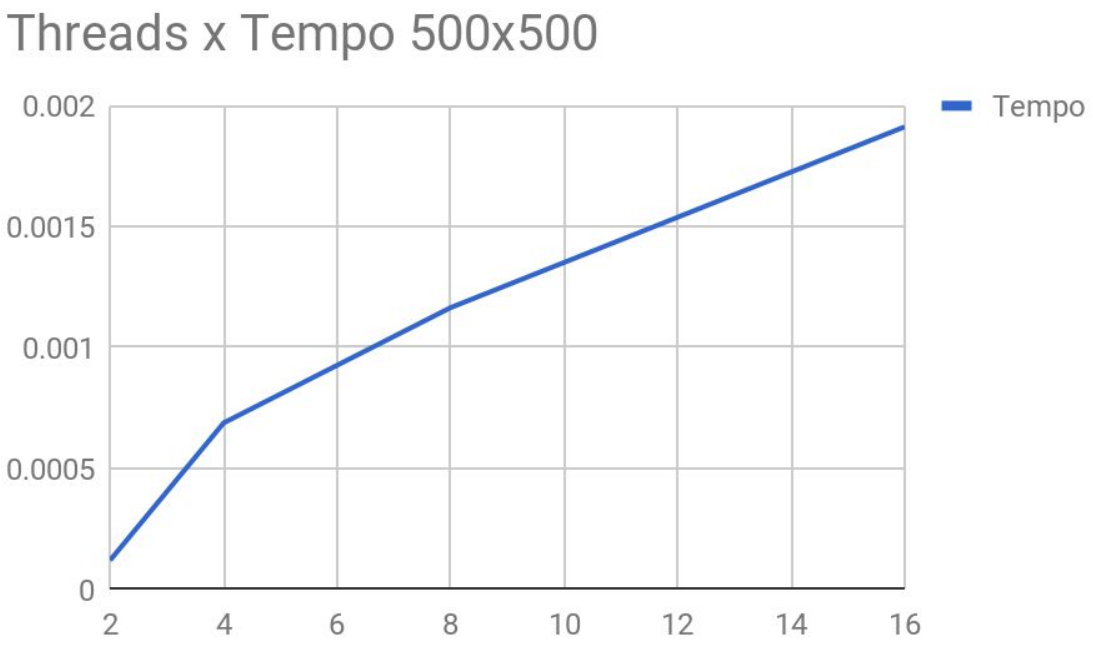
As representação do desempenho das threads de 2, 4, 8 e 16, são médias de tempo de cinco execuções do programa para cada matriz.

Informações sobre o PC usado para testar: samsung expert x30, com i5 5200u e 8GB de ram 1600mhz ddr3.

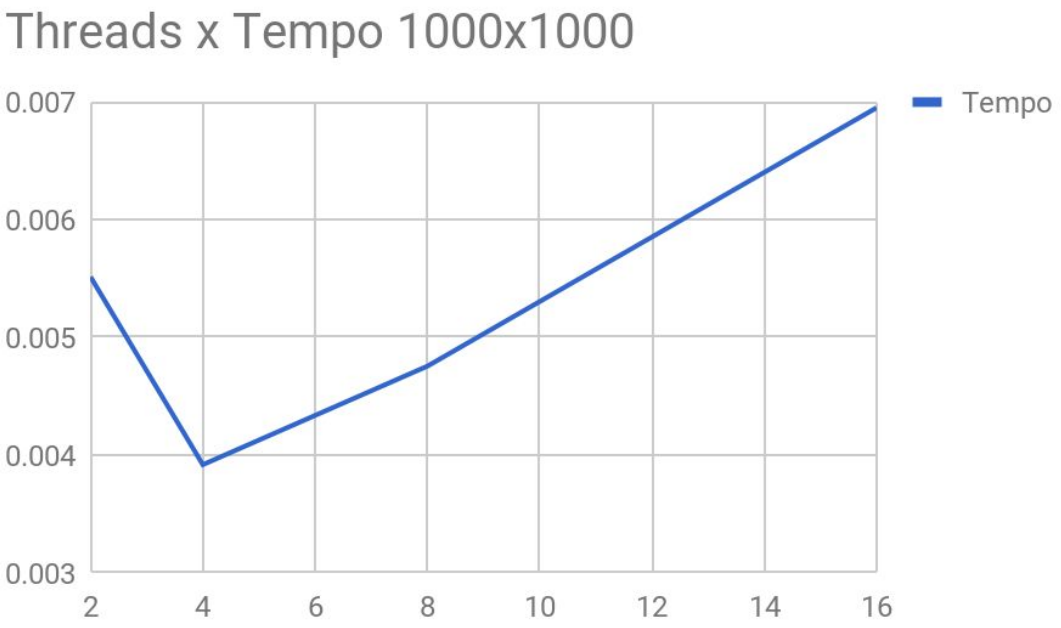
Para uma Matriz 100x100:



Para uma matriz 500x500:



Para uma matriz 1000x1000:



❏ Conclusão a respeito dos resultados obtidos

A conclusão do trabalho é o problema da multiprogramação, para atividades simples, como buscar o valor em uma matriz 100x100 normalmente será mais rápido executá-la de modo monoprogramação do que dividir a tarefa em várias threads, pois ao fazer isso, a própria divisão pode levar mais tempo que a execução da atividade, a divisão compensa apenas em atividades mais complexas, no caso do nosso programa, em matrizes 1000x1000 ou em atividades que são necessariamente paralelas.

Outro ponto em que as threads podem trazer perda de desempenho é o próprio controle delas, pois maior a quantidade de threads mais concorrência é gerada, no caso do nosso programa evitamos o uso de semáforos por este motivo, e mesmo em atividades que se beneficiam da utilização da multiprogramação, haverá um limite de recursos no sistema o que traz uma diminuição no desempenho caso a aplicação utilize mais threads do que o necessário.

Link para o repositório GitHub:

https://github.com/SamuelAzorli/SO

Link do Drive para acesso do video:

https://drive.google.com/drive/folders/1qtDmUa4bM_Hy6knWEzoUzym7UtJvoFt0?usp=sharing