Sistemas Operacionais I
Programando com Múltiplas Threads

Francisco José da Silva e Silva
Departamento de Informática, UFMA

1º Semestre de 2022

1 Objetivo, Datas e Observações

- Objetivo: Desenvolver a habilidade de programação com múltiplas threads.
- **Data limite para a entrega: 21 de junho de 2022**.

2 Implementação

O objetivo do programa a ser desenvolvido é: dada uma matriz de números naturais aleatórios (dentro do intervalo 0 a 29999) verificar quantos números primos existem, indicar suas posições na matriz e contabilizar o tempo necessário para realizar esta computação. Isso deve feito de duas formas:

1. De modo sequencial, ou seja, a contagem dos primos e inclusão em uma estrutura de dados das posições dos números primos encontrados na matriz de primos deve ser feita um número após o outro. O tempo para realizar essa computação será considerado seu tempo de referência.
2. De modo concorrente. Para tanto, o programa deve subdividir a matriz em "blocos" (submatrizes) sem efetuar cópias da mesma, baseando-se apenas em índices. A verificação dos números primos de cada submatriz bem como o registro de suas posições deve ser realizada por uma thread independente. Um exemplo é ilustrado na Figura 2.

A matriz ilustrada na figura é de tamanho 9 x 9 e cada bloco é composto por 9 elementos. O bloco 1 vai da coluna 0 a 2 e da linha 0 a 2, e assim sucessivamente. Os blocos serão as unidades de trabalho de cada thread. Pode-se arbitrar que a matriz é quadrada e todos os blocos devem ter o mesmo tamanho. Através de diretivas `#define`, o código deve permitir a parametrização tanto do tamanho da matriz, quanto dos blocos.

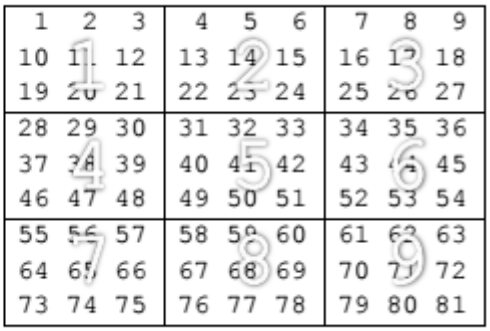

Figura 1: Exemplo de Matriz e Submatrizes

A matriz de números primos, a variável que contabiliza a quantidade de números primos e a estrutura de dados a ser utilizada para guardar as posições nas quais os números primos foram encontrados deverão ser globais (logo, compartilhadas entre as threads) e únicas.

Diretrizes básicas a serem seguidas:

- Para a geração da matriz de números naturais aleatórios (intervalo 0 a 29999) utilize uma semente pré-definida no código, a fim de sempre ter a mesma "matriz aleatória" para todos os testes. A geração de números aleatórios em C pode ser realizada com o uso das funções `srand()` e `rand()`.

- O tamanho da matriz deverá ser consideravelmente grande a fim de que possam ser efetuadas medidas de desempenho consistentes. Para efeito de comparação, em um desktop com 8 GB de RAM e CPU core i5 com 4

  núcleos reais, a verificação serial de uma matriz de 20000 x 20000 levou aproximadamente 42 segundos.

- Você deve tomar cuidado na escolha do tamanho da matriz. Em testes efetuados, o processo que utilizou a matriz 20000 x 20000 ocupou aproximadamente 1,5 GB de RAM. No entanto, visto ter o computador uma quantidade de RAM expressivamente maior que isso, pode-se concluir que a memória secundaria (HD ou SSD, que é muito mais lento que a RAM, obviamente) não fora usada. A escolha das dimensões da matriz deverá levar em conta a configuração do computador onde os testes serão executados a fim de evitar medidas incorretas devido ao uso da memória virtual. Faça testes de olho no consumo de memória ao executar o programa, usando o Gerenciador de Tarefas, se estiver

no Windows, ou os comandos top, htop, vmstat no Linux, a fim de definir um tamanho razoável para a matriz.

- Para contabilizar o tempo de processador gasto utilize uma ferramenta de *profile*, como o GProf (`https://www.ibm.com/developerworks/br/local/linux/gprof_introduction/index.html`)
- A escolha da quantidade de threads para o teste principal deverá levar em conta o número de núcleos reais da CPU que equipa o computador. Se a CPU tiver $N \geq 2$ núcleos reais, crie N threads.
- O programa deve executar e imprimir o resultado da quantidade de números primos encontrados e suas posições com a abordagem sequencial e, após isso, executar e imprimir o mesmo resultado com a abordagem concorrente.

Realize vários testes considerando uma matriz de bom tamanho e alterando a quantidade de blocos. Escreva um relatório que analise os resultados dos testes. Faça análises relativas ao tempo de processamento considerando as diferentes quantidades de threads utilizadas. Lembre de considerar tamanhos de blocos muito grandes e pequenos. O relatório deve conter uma conclusão: o que você pôde aprender com esse trabalho?

O relatório deve conter: uma descrição da máquina na qual os testes foram realizados (qual a CPU, quantidade de memória e sistema operacional); os resultados dos experimentos realizados variando-se a quantidade de threads e, eventualmente, o tamanho da matriz; e as conclusões que você tirou da execução dos experimentos.

3 O que enviar ?

A entrega do trabalho deve ser realizada exclusivamente através da plataforma Google Classroom. Deve-se enviar um arquivo compactado (tar.gz ou zip) contendo:

- Código fonte do programa implementado
- O relatório no formato PDF
- Um arquivo texto contendo:

  Como compilar e executar seu código;

  Bugs encontrados e não tratados;

  Dificuldades enfrentadas.