

Sistemas Operacionais I

Programando com Múltiplas Threads

Francisco José da Silva e Silva
Departamento de Informática, UFMA

1º Semestre de 2020

1 Objetivo, Datas e Observações

- Objetivo: Desenvolver a habilidade de programação com múltiplas threads.
- **Data limite para a entrega: 26 de novembro.**
- Dúvidas podem e devem ser tiradas com o assistente de ensino: Ivan Rodrigues / ivan.rodrigues@lsdi.ufma.br

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 e contabilizar o tempo necessário para realizar esta computação. Isso deve ser feito de duas formas:

1. De modo sequencial, ou seja, a contagem dos primos deve ser feita um número após o outro. O tempo para realizar essa computação será considerado seu tempo de referência.
2. De modo paralelo. Para tanto, o programa deve subdividir a matriz em "macroblocos" (submatrizes) sem efetuar cópias da mesma, baseando-se apenas em índices. A verificação dos números primos de cada submatriz 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 macro-bloco é composto por 9 elementos. O macro-bloco 1 vai da coluna 0 a 2 e da linha 0 a 2, e assim sucessivamente. Os macro-blocos serão as unidades de trabalho de cada thread. Atenção: nem a matriz nem os macro-blocos deverão ser obrigatoriamente quadradas. A única exigência é que todos os macro-blocos

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

Figura 1: Exemplo de Matriz

tenham o mesmo tamanho. Além disso, você deve encontrar alguma forma de PARAMETRIZAR essa divisão (usando a diretiva `#define`, por exemplo) a fim de poder efetuar os testes para diferentes tamanhos de macro-blocos. Os macro-blocos terão tamanhos que podem variar desde um único elemento até a matriz toda (equivalente ao caso sequencial).

A matriz e a variável que contabiliza o número de números primos encontrados deverão ser globais (logo, compartilhadas entre as threads) e únicas.

Diretrizes básicas a serem seguidas:

- Geração de uma matriz de números naturais aleatórios (intervalo 0 a 29999) usando 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.
- 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)
- **Cálculo sequencial** da quantidade de números primos da matriz. Exiba a quantidade de números primos encontrados e o tempo decorrido nessa busca.

- **Cálculo paralelo** da quantidade de números primos na mesma matriz usada na busca sequencial. Cabe aqui, algumas dicas:

A escolha do número 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 $N_c = 2$ núcleos reais, crie N threads.

A atribuição de qual macrobloco será processado em cada momento deverá ser feita da seguinte forma: suponha que foram criadas 4 threads. A thread 1 deverá começar a busca no macrobloco 1, a thread 2 no macrobloco 2, a thread 3 no macrobloco 3 e a thread 4 no macrobloco 4. Devido a natureza aleatória dos números (e consequentemente do tempo de verificação se o número é primo ou não depender da magnitude do número) e do escalonador do sistema operacional, nada se pode afirmar sobre que thread terminará sua computação primeiro. No entanto, digamos que a thread 2 termine sua computação primeiro. Ela deverá: somar o número de primos encontrado à variável que esteja contabilizando o número total de primos da matriz (ou seja, a thread usará uma variável temporária para contar o número de primos e, após terminada a busca no macrobloco, somará esse valor à variável global) e reiniciar a busca no próximo macrobloco que ainda não foi atribuído a nenhuma thread. A variável que controla quais macroblocos estão livres/alocados deverá ser global (compartilhada). Exemplo: Thread 2 termina. Logo, ela verificará se o macrobloco 5 já foi atribuído a alguma thread. Se não, ela o marca como já atribuído e reinicia seus trabalhos nele. Caso contrário, busca pelo próximo macrobloco livre, até que por fim se esgote os macroblocos a serem buscados. Neste ponto, a thread termina e a thread principal fica esperando que as demais threads terminem.

Atenção: É proibido criar um vetor para armazenar a quantidade de números primos encontrados em cada macrobloco e depois somá-los. Como já mencionado, as variáveis que armazenam a quantidade total de número de primos, a que controla a alocação dos macroblocos e a matriz principal deverão ser globais e o acesso compartilhado deverá ser controlado.

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
- Um relatório no formato PDF, que deve conter:

Como compilar e executar seu código;
Bugs encontrados e não tratados;
Dificuldades enfrentadas;
Avaliação do seu Trabalho: Seja Criativo :)

Avalie/critique os resultados dos testes, faça análises relativas ao tempo de processamento considerando as diferentes quantidades de threads utilizadas e tamanhos de macro-blocos. Você encontrará resultados bem interessantes quando o tamanho dos macroblocos for muito grande ou muito pequenos! Em resumo, brigue com seu trabalho, e analise: o que você pôde aprender com esse trabalho?

Boa Sorte :)