Universidade Federal do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES

Processamento Paralelo

Prof. Oberlan Romão

Números primos da forma $2n^2 - 1$

Como sabemos, um número é classificado como primo se ele é maior do que um e é divisível apenas por um e por ele mesmo. Apenas números naturais são classificados como primos. O dez primeiros números primos são: 2, 3, 5, 7, 11, 13, 17, 19, 23 e 29.

Existem várias estratégias/algoritmos para verificar se um número é primo ou não. A mais simples e trivial é verificar se um dado número n possui dois divisores (veja o código abaixo). Claro que essa estratégia não é a única e nem a mais eficiente.

```
bool ehPrimo(int n) {
    int numDiv = 0;
    for (int d = 1; d <= n; d++) {
        if (n % d == 0)
            numDiv++;
    }
    return numDiv == 2;
}
```

Considere os números da forma $t(n) = 2n^2 - 1$ com $n > 1$. Os primeiros números dessa sequencia são 7, 17, 31, 49, 71, 97, 127 e 161 (veja a tabela abaixo). Desses, apenas $49 = 7 \times 7$ e $161 = 7 \times 23$ não são primos. Para $n \leq 10.000$, existem 2202 números da sequencia $t(n)$ que são primos.

n	2	3	4	5	6	7	8	9
$t(n)$	7	17	31	49	71	97	127	161
Primo?	Sim	Sim	Sim	Não	Sim	Sim	Sim	Não

Tarefa

Você deve implementar um código paralelo, usando Pthread ou std::thread, que encontre quantos número de $t(n)$ são primos para $n \leq N$. Você pode usar qualquer estratégia para verificar se um número é primo ou não, mas você deve explicar o que fez (use comentários para isso). Caso parte do código seja baseado de algum artigo/livro/site, deixe isso claro. Seu código deve imprimir **apenas** a quantidade de números primos de $t(n)$, para $n \leq N$, e o tempo gasto em segundos.

Após a implementação (tenha certeza que seu código paralelo está gerando a resposta correta), você deve fazer alguns experimentos do seu programa no DevCloud da Intel. Nos experimentos, você deve contabilizar o tempo total (em segundos) do programa considerando os valores de N e número de *threads*

mostrados na tabela abaixo. Nesse miniEP, você pode executar apenas uma vez o seu código. **Essa tabela deve ser entregue preenchida.**
Obs.: Caso o seu programa demore mais que 6h (tempo limite padrão de execução de um *job* no DevCloud) para algum valor N e número de *threads*, o mesmo será encerrado e você não saberá a resposta. Nesse caso, coloque na célula da tabela o valor `>6h`.

Tempo gasto (em segundos)

N	**Número de Threads**				
	1	**6**	**12**	**18**	**24**
10.000					
100.000					
1.000.000					
5.000.000					
10.000.000					

Dica 1: Faça com que N e o número de *threads* sejam argumentos do seu programa (veja o código da Aula Prática 5).
Dica 2: Crie um arquivo `job.sh` para cada valor de N. Dentro do arquivo, faça um `for` para o número de *thread* (veja o arquivo `job.sh` da Aula Prática 5). Após isso é só submeter os 5 *jobs* e rezar :)

O que entregar?

Nessa tarefa você deve enviar o arquivo com o código do seu programa (.c ou .cpp) paralelo (usando std::thread ou Pthread) e um arquivo (.pdf) com a tabela preenchida. A explicação da estratégia usada pode ser descrita como comentário no código ou estar no arquivo pdf.

Data de entrega: até às 6h do dia 03/11/2020.

Observações:

1. Você deve implementar o código usando a linguagem de programação C/C++ (minha sugestão é usar C++ e std::thread);
2. Envie apenas o código e o arquivo relatório (não compacte os arquivos);
3. Contabilize o tempo total do seu programa, algo assim:

```
int main() {
    double inicio = MyClock();
    //Implementação
    printf("%.5lf\n", (MyClock()-inicio)/CLOCKS_PER_SEC);
    return 0;
}
```

4. Entregas com atraso, sem justificativa, serão desconsideradas;
5. Em caso de plágio, será atribuído 0 a todos os envolvidos.