UNIVERSIDADE FEDERAL DO CARIRI
CENTRO DE CIÊNCIAS E TECNOLOGIAS
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ALAN GABRIEL SILVA OLIVEIRA
PEDRO DA SILVA VIANA

PROJETO FINAL
PROGRAMAÇÃO CONCORRENTE

SUMÁRIO

1. Introdução

O objetivo do projeto é implementar o Crivo de Eratóstenes utilizando programação paralela e, em seguida, comparar seus resultados com a execução sequencial do algoritmo. O Crivo de Eratóstenes é reconhecido como um método eficaz e antigo para identificar todos os números primos dentro de um limite especificado. Ao adotar uma abordagem paralela para este algoritmo, buscamos explorar o poder de processamento distribuído dos sistemas computacionais modernos, com o intuito de aprimorar a eficiência e o desempenho na identificação de números primos. A análise comparativa entre a execução paralela e sequencial nos permitirá avaliar o impacto da paralelização no tempo de execução e na utilização dos recursos computacionais, oferecendo valiosas perspectivas sobre a escalabilidade e eficiência do algoritmo em diversos contextos de aplicação.

Para a implementação paralela com memória compartilhada, foi utilizada a biblioteca OpenMP. Infelizmente, devido a limitações técnicas, não foi viável realizar a implementação com memória compartilhada usando a MPI.

2. Recursos computacionais utilizados

Os algoritmos foram executados em um notebook Lenovo IdeaPad 3, com processador AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz e 8Gb de Ram.

3. Implementação

3.1. Visão geral do algoritmo

O funcionamento do algoritmo, tanto na versão sequencial quanto na paralela, é substancialmente semelhante. Partindo de um número "n", inicializamos um vetor com "n - 1" posições, representando os números de 2 até "n", e preenchemos todo o vetor com 1 's, indicando inicialmente que todos são números primos. Em seguida, iteramos sobre o vetor novamente. Ao

encontrar um número marcado como primo, exploramos todos os seus múltiplos até "n", marcando-os com 0 para indicar que não são primos. Esse procedimento exclui os números compostos, deixando apenas os números primos no vetor ao final da execução. Após a conclusão do algoritmo, percorremos o vetor e imprimimos as posições que permanecem marcadas como 1, identificando-os como números primos.

3.2. Implementação paralelizada

Para aproveitar ao máximo o paralelismo, distribuímos tanto a iteração que verifica os múltiplos de um número primo entre vários processadores distintos quanto a iteração do laço responsável pelo preenchimento do vetor com 1 's. Isso contribui significativamente para a eficiência do algoritmo. É crucial observar que o laço que percorre o vetor para verificar se um número é primo não pode ser paralelizado, pois dependemos de iterações anteriores para essa verificação.

4. Resultados

O Crivo de Eratóstenes demonstra uma complexidade assintótica de O(nlog(log n)). Em vista disso, é notável que para entradas de tamanho reduzido, a discrepância temporal entre a execução paralela e sequencial é mínima, mas essa disparidade aumenta progressivamente conforme o tamanho da entrada cresce. Por exemplo, ao considerar n = 1000000000 (1 bilhão), constatamos que o tempo de execução paralela é de aproximadamente 10,148188 segundos* utilizando 3 threads, enquanto na execução sequencial o tempo é de 17,928973 segundos*. Isso resulta em um Speedup de 1,76671667 e uma eficiência de 0,58890555. Antecipa-se que tanto o Speedup quanto a eficiência tendem a melhorar conforme o tamanho da entrada aumenta. No entanto, uma limitação significativa se faz presente em relação ao tamanho da entrada, visto que a máquina não suporta um vetor de inteiros de 32 bits com tamanho 1 bilhão, o que nos obrigou a utilizar o tipo int8_t da biblioteca <inttypes.h> para lidar com essa magnitude de dados.

* Desconsiderando o tempo gasto para imprimir os resultados na tela.