Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Exercício de Programação 1 (EP1)

1. Identificação da Disciplina: Programação Paralela

2. Introdução

Dado o conjunto dos números naturais $\mathbb{N}$, um número p, onde $p \neq 0$ e $p \neq 1$, é considerado primo se ele possui exatamente dois divisores, um e ele mesmo.

O crivo de Eratóstenes é um método para encontrar todos os números primos até um número limite n. O método pode ser descrito pelos seguintes passos:

- Crie uma lista de inteiros consecutivos de 2 até n: $(2, 3, 4 \ldots, n)$.
- Seja p=2, o primeiro número primo.
- Enumere todos os múltiplos de p até n e marque-os como não primos na lista criada no primeiro passo. Esses números serão da forma $2p, 3p, 4p \ldots$. Note que p deve continuar na lista (pois é primo).
- Encontre o primeiro número na lista que seja maior do que p e que não esteja marcado como não primo. Caso não exista tal número, pare. Caso contrário, seja p igual a este número (que é o próximo primo já que caso não fosse primo ele teria sido marcado no passo 3), repita a partir do passo 3.
- Quando o algoritmo terminar, os números não marcados presentes na lista serão todos os números primos entre 1 e n.

3. Enunciado

Deve-se implementar um algoritmo usando C/C++ e paralelizar o mesmo utilizando os pragmas de OpenMP. O algoritmo precisa gerar uma lista ordenada de números primos de 2 até um inteiro n passado na entrada, onde $1 < n \leq 10^9$.

3.1. Entrada do Programa

O programa deverá receber três parâmetros como entrada: um inteiro n que representa o limite superior $[2, n]$ da lista de primos e uma string s que representará os tipos de saídas: "$time$" para imprimir somente o tempo, "$list$" para imprimir somente a lista de números e "all" para imprimir a lista de números na primeira linha e o tempo na segunda linha.

3.2. Saída do programa

O programa deverá imprimir a lista de primos com um espaço após cada número e o tempo de execução em segundos com precisão de seis casas decimais. O resultado deverá ser impresso na saída padrão (stdout) no seguinte o formato, de acordo com a string s de entrada:

- "$time$": Uma linha contendo o tempo de execução do programa.
- "$list$": Uma linha contendo a lista de números primos.
- "all": A primeira linha contendo a lista de números primos e a segunda linha o tempo de execução do programa.

Após a exibição, acrescente uma quebra de linha - `printf("\n")`- na saída. O parâmetro p refere-se ao número de threads utilizadas. Abaixo apresentamos alguns exemplos de execuções:

Nr	Exemplo de entrada	Exemplo de Saída
01	10 a p	2 3 5 7 0.000007
02	42 l p	2 3 5 7 11 13 17 19 23 29 31 37 41
03	100000 t p	0.000596

4. Entregas e avaliações

O código deverá ser entrege no moodle da disciplina juntamente com o relatório.

4.1. Código

- O código deve conter todo o fonte do seu programa (C ou C++) juntamente com instruções claras sobre sua compilação e execução (Makefile). Você deverá entregar o código sequencial e o paralelo.
- O ambiente de testes será Linux com OpenMPI e GCC.
- Programas com erros de compilação receberão nota 0.
- Programas que não cumprirem os requisitos (resultados incorretos, erros durante a execução...) descritos na seção anterior receberão nota 0.
- Programas sem boa documentação no código terão a sua nota reduzida.
- Programas desorganizados (nomes de variáveis/funções ruins, falta de identação,...) terão a sua nota reduzida.

4.2. Relatório

Juntamente com o seu código você deverá entregar um relatório (máximo de 5 páginas) que contenha:

- Uma explicação detalhada de como o particionamento das tarefas foi feito entre os processos na sua implementação.
- Você deverá avaliar o impacto de tipo de escalonador e *chunkSize* no desempenho.
- Para o melhor escalonador e chunkSize, Tabelas e gráficos do speedup, eficiência e tempo variando os valores de N e P (não esqueça de incluir a especificação da máquina utilizada para a execução - recomendo o uso da máquina n1-standard-8 no google cloud). Você deverá utilizar p até o valor de 8.
- Interpretação dos valores mostrados na tabela anterior. Descreva e interprete a variação do speedup e da eficiência em relação à N e P. Explique os resultados obtidos levando em consideração as características da máquina utilizada.
- Análise da escalabilidade da sua implementação. Ela é escalável? Apresenta escalabilidade forte ou fraca?
- Análise do balanceamento de carga. Todos os processos recebem a mesma quantidade de trabalho? Todos acabam a execução aproximadamente ao mesmo tempo?

Nesses experimentos sugerimos utilizar a execução com o parâmetro t, para evitar impacto do I/O no tempo.

4.3. Desempenho

- É esperado que você consiga speedup > 1 na versão paralela.
- Os desempenhos tanto da versão sequencial (otimize seu código) quanto da versão paralela do seu algoritmo serão avaliados.

A descrição desse trabalho utilizou materiais dos Profs. Emilio Francesquini e Prof. Fikret Ercal.