Ministério da Educação
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
UNED Nova Friburgo
Bacharelado em Sistemas de Informação
Disciplina de Programação Paralela e Concorrente
Professor Bruno Policarpo Toledo Freitas

Trabalho: Paralelização de algoritmos

1. **Objetivo**
   Aplicar os conceitos de paralelização com OpenMP ou MPI e análise de desempenho de programas paralelos vistas em aula

2. **Especificação**
   O trabalho consistirá em paralelizar e apresentar uma análise dos seguintes quatro algoritmos utilizando a biblioteca OpenMP em C ou C++:

- Multiplicação Matricial
- Algoritmos de ordenação:
  - Bucket sort
  - Shell sort
- Eliminação Gaussiana (ou Triangulação de matrizes)
  - https://en.wikipedia.org/wiki/Gaussian_elimination

Para cada um dos algoritmos, criar uma seção do algoritmo, criando uma subseção para realizar os seguintes passos, em ordem :

2.1. Descrever o algoritmo (preferencialmente com exemplos). Não é para explicar o código! É o algoritmo! São 2 coisas diferentes!

2.2. Implementar os algoritmos seriais

Não precisa criar uma subseção para este passo, irei ler o código

2.3. A partir da implementação do algoritmo do item anterior, indicar no código:
   a) Regiões Críticas
   b) Dependências de dados

2.4. Implementar o algoritmo em paralelo, baseado no algoritmo serial.

2.5. Uma comparação de desempenho entre os algoritmos serial e sua implementação paralela, sob os critérios de *Speedup* e *Eficiência*.

Para os critérios de Speedup e Eficiência, crie uma tabela com os valores obtidos sob diferentes tamanhos de entradas e quantidade de threads, conforme o exemplo abaixo.

SIZE_1, SIZE_2, e SIZE_3 devem ser três valores escolhidos para realizar as análises.

Procure escolher valores que cresçam de duas em duas vezes. Por exemplo, se o problema trabalhar com vetores unidimensionais: 100, 200, 400.

Critério (Tempo de execução, Speedup ou Eficiência)			
	Threads		
Tamanho das entradas (SIZE)	1 (serial)	2	4
SIZE_1			
SIZE_2			
SIZE_3			

2.6. Crie os seguintes gráficos de **colunas**, baseados nos resultados do item 2.5:

a) *Speedup* x tamanho das entradas, para 2 e 4 threads, e variando o tamanho das entradas em SIZE_1, SIZE_2, e SIZE_3, no mesmo gráfico.

b) Eficiência x tamanho das entradas, para 2 e 4 threads, e variando o tamanho das entradas em SIZE_1, SIZE_2, e SIZE_3, no mesmo gráfico

2.7. Faça uma análise dos resultados obtidos nos itens 2.5 e 2.6. Explique as diferenças observadas para o Speedup e Eficiência.

2.8. Faça uma análise da Escalabilidade do algoritmo. O seu algoritmo é forte ou fracamente escalável? Justifique.

2.9. **(Somente para os algoritmos de ordenação)** Faça um comparativo da utilização de memória cache e desempenho entre os algoritmos de ordenação na implementação paralela.

Qual deles ficou mais rápido? Qual deles gerou mais cache-misses? Explique.

3. **Avaliação**

   O aluno deverá entregar um relatório em formato de artigo resumido, descrevendo a estratégia de paralelização e os resultados obtidos

   O relatório deverá ser escrito no formato de artigo resumido da SBC, disponível aqui.

   O aluno deverá entregar um PDF do artigo e um arquivo compactado contendo os código-fonte desenvolvidos.

   No último dia letivo, será realizada uma competição de alto desempenho. Os critérios serão apresentados no dia. Os grupos com os algoritmos mais rápidos (e corretos, obviamente) irão receber:

   1. Primeiro lugar: 1 ponto extra na média da P2
   2. Segundo lugar: 0,5 pontos extras na média da P2
   3. Terceiro lugar: 0,2 pontos extras na média da P2

4. **Dicas**

   - Caso os algoritmos sejam executados “rápido demais”, repitam a execução do algoritmo diversas vezes e considerem que o tempo de execução é a soma de todas as execuções;

   - Caso seja necessário repetir a execução do algoritmo, lembre-se de usar os mesmos dados de entrada em todas repetições;

   - Cuidado com os *timings*! Ou seja, meça o tempo de execução apenas durante a execução do algoritmo em si!

     Por exemplo: não meça o tempo em que os dados de entrada do algoritmo são definidos!

   - Lembre-se, também, e verificar se seu programa paralelo está funcionando corretamente – ou seja, no mínimo compare a saída do programa serial com o paralelo, no mínimo!

     Uma maneira rápida de fazer essa comparação é salvar os resultados dos programas serial e paralelo em um arquivo e comparar o hashing obtido entre eles.