

Bacharelado em Ciência da Computação - DCC/IM-UFRJ
Programação Paralela e Distribuída
Prof. Gabriel P. Silva
1º Trabalho – 21/01/2020

- 1) Neste trabalho você deve paralelizar o programa **calpi**, que calcula o valor de π (3.14159265358979323846264 ...) usando uma aproximação do método de integral com o método do trapézio ($f(x)=4/(1+x^2)$) e com o método de Monte Carlo. É uma boa aplicação para se paralelizar porque a divisão do trabalho entre as tarefas é muito simples, e nenhum padrão complicado de comunicação é usado. Ao paralelizar o código do **calpi**, você aumentará sua compreensão dos modelos de programação, ganhará experiência no desenvolvimento de uma aplicação MPI e vai fazer o uso de muitas chamadas básicas do MPI. Para isso observe os seguintes passos:
 - Você necessitará de um arquivo começar o trabalho: as versões sequenciais do calpi, que podem ser obtidas na página da disciplina no AVA;
 - Compile, verifique o seu funcionamento e familiarize-se com o modo com que o programa sequencial funciona;
 - Certifique-se que a divisão de tarefas entre os processos seja a mais justa possível, independente de número de processos.
 - Utilize a rotina **MPI_Wtime** para verificar o tempo de execução do programa.
 - Avalie o desempenho das duas versões paralela, com valores de N para 10^9 , 10^{10} e 10^{11} . Utilize 1, 2, 4, 8, 12 e 16 processos.
 - Trace gráficos com o tempo total de execução, speedup e eficiência.
 - Apresente um relatório com código fonte, resultados e comentários sobre todo esse processo.

- 2) A segunda tarefa consiste em elaborar um programa de ordenação em paralelo de um vetor. Considere para isso um vetor com N números aleatórios gerados previamente. A seguir este vetor é distribuído pelo processo raiz para os demais processos e a ordenação parcial é realizada por cada um dos processos. O processo raiz deve ter o resultado final do arquivo. Para realizar esse programa considere:
 - Utilizar rotinas de **comunicação coletiva** para a distribuição e coleta dos vetores.
 - Manter o maior número possível de processos ocupados no processo de ordenação.
 - Escolher um algoritmo local sequencial de ordenação de melhor desempenho. Justifique a sua escolha.
 - Utilize a rotina **MPI_Wtime** para verificar o tempo de execução do programa.
 - Executar o programa com 1, 2, 4, 8, 12 e 16 processos e ordenando 10, 20 e 40 milhões de elementos.
 - Trace gráficos com o tempo total de execução, speedup e eficiência.
 - Apresente um relatório com código fonte, resultados e comentários sobre todo esse processo.

- 3) Neste trabalho você deve modificar um programa que calcula a equação de uma reta utilizando o método de mínimos quadrados. Antes disso, contudo, você deve escrever um pequeno programa que gere um arquivo cujo primeiro dado é um valor **N**, seguido de **N** pares de coordenadas **X** e **Y** que vão servir como entrada de dados. Você pode usar uma equação conhecida e atribuir um pequeno erro aleatoriamente para cada y gerado. Para modificar o programa observe as seguintes instruções:
 - Uma versão MPI do programa do método dos mínimos quadrados pode ser obtida no AVA.
 - Envie o valor de **n** e de todos os valores lidos do arquivo para todos os processos utilizando uma rotina de comunicação coletiva;

- Escreva uma versão com uso da rotina **MPI_Bsend** para o envio dos valores de x e y para cada um dos processos;
- Utilize rotinas de **comunicação coletiva** (redução) para a recepção das somas parciais;
- Avalie o desempenho da versão paralela do programa com valores de N para 10^6 , 10^7 e 10^8 . Utilize 1, 2, 4, 8, 12 e 16 processos.
- Trace gráficos com o tempo total de execução, speedup e eficiência.
- Apresente um relatório com código fonte, resultados e comentários sobre todo esse processo.

4) Neste trabalho você deve paralelizar o programa **mandelbrot**, que calcula um fractal de mandelbrot, utilizando rotinas do MPI. Ao paralelizar o código do **mandelbrot**, você aumentará sua compreensão dos modelos de programação, ganhará experiência no desenvolvimento de uma aplicação MPI. Para isso observe os seguintes passos:

- Você necessitará de um arquivo começar o trabalho: a versão sequencial do mandelbrot, que pode ser obtida no AVA.
- Compile, verifique o seu funcionamento e familiarize-se com o modo com que o programa sequencial funciona.
- Experimente modificar o código para imprimir o **mandelbrot** colorido, aumente o tamanho da imagem para obter um tempo significativo de computação.
- Paralelize o programa. Para conseguir isto, imagine que cada processo irá calcular faixas (horizontais ou verticais) da imagem. Paralelize o envio dos resultados já calculados com o cálculo de novas faixas com uso da rotina **MPI_Isend**.
- Note que apenas uma tarefa deverá escrever o resultado final da imagem para exibição posterior para o usuário.
- Compare o desempenho rodando em 1, 2, 4, 8, 12, e 16 processos.
- Apresente um relatório com código fonte, resultados e comentários sobre todo esse processo.

5) Execute todas as versões do programa de cálculo de números primos do livro. Compare as duas versões paralelas com 1, 2, 4, 8, 12 e 16 processos. Utilize valores de N igual a 10^9 , 10^{10} e 10^{11} . Verifique a escalabilidade com base na Lei de Gustafson, ou seja, aumentar a carga de trabalho proporcionalmente ao aumento número de processadores. Use o N que considerar adequado neste caso.

Em todos os trabalhos os valores de N podem ser ajustados caso resultem em tempos de execução muito pequenos (< 1s) ou muito grandes (> 1h) . Todos os itens tem o mesmo valor. Boa sorte!