REQUISITOS

Dependências:

Python 3: O código foi escrito em Python e precisa do interpretador Python 3.

mpi4py: Biblioteca Python para interface com o MPI. Para instalá-la, execute:

```
pip install mpi4py
```

OpenMPI: Implementação do MPI que será usada para comunicação entre os processos distribuídos. Foi utilizado Microsoft MPI v10.1.3: https://www.microsoft.com/en-us/download/details.aspx?id=105289

Execução do Programa:

- Executar o código com o comando mpiexec (ou mpirun) para iniciar múltiplos processos MPI. A quantidade de processos é otimizada para um quadrado perfeito, uma vez que as matrizes são divididas em blocos, mas o código adequa-se para fora dessa regra.

Exemplo de Comando de Execução:

Para executar o código com 4 processos (por exemplo):

```
mpiexec -n 4 python MPI_entrega.py
```

- mpiexec -n 4 especifica que 4 processos MPI serão criados.
- python MPI_entrega.py é o nome do arquivo Python contendo o código.

Observações:

O número de processos deve ser o quadrado perfeito de um número (1, 4, 9, 16, etc.) para garantir que as matrizes sejam divididas corretamente.

Se você estiver trabalhando em um ambiente com múltiplos nós, pode precisar configurar o mpiexec para especificar os nós nos quais os processos serão executados.

Verificando a Execução:

Ao rodar o código, o processo de rank 0 (root) imprimirá as matrizes A, B e o resultado da multiplicação de matrizes A x B = C no terminal.

Notas Finais:

Para ajuste da dimensão de matrizes é necessário ajustar os valores das variáveis **m, n, p** na sessão *##definir as dimensões das matrizes*.

Para grandes matrizes e muitos processos, é recomendado executar o programa em um cluster de alto desempenho para melhor escalabilidade e uso eficiente dos recursos.