# Programação Paralela e Distribuída

**Programação Paralela com MPI – Modelo Pipeline**

**Exercício 2 – *Insertion Sort* paralelo**

O objetivo do trabalho é implementar, usando a biblioteca MPI, uma versão paralela usando o modelo *pipeline* do algoritmo de ordenação *Insertion Sort*. Após implementado, o programa deve ser executado no *cluster* para realização das medições de desempenho e, por fim, deve-se gerar o gráfico de *speed up*.

O algoritmo ordenação *Insertion* Sort funciona da seguinte forma: "Cada número é inserido de forma ordenada em um vetor. O valor é comparado com cada posição do vetor até encontrar a posição em que deve ser 'encaixado'. Após a inserção do valor, todos os outros valores conseguintes são recolocados uma posição adiante."

A versão sequencial do algoritmo está apresentada abaixo:

```
for(i=0; i<n; i++) {
   val = A[i];
   for(j=0; j<i; j++) {
      if(val < B[j]) {
         aux = B[j];
         B[j] = val;
         val = aux;
      }
   }
   B[i] = val;
}
```

A versão paralela usando *pipeline* consiste em dividir o vetor resultado de tamanho n em P pedaços, sendo P o número de processos utilizados para executar o programa. Cada processo aloca um espaço para armazenar o seu pedaço do vetor resultado (n/P). O processo 0 contém o vetor de entrada e inicia a inserção ordenada no seu vetor resultado. Quando o vetor resultado local estiver cheio, a inserção de um valor resultará na saída de outro. Este valor que é "expelido" do vetor resultado local é enviado para o próximo processo que o utiliza para realizar o mesmo processo no seu vetor resultado. Ao fim da inserção de todos os valores, todos os processos enviam os seus vetores resultado para o processo 0 que os armazena corretamente.

O programa deverá receber como parâmetros de entrada um número N, representando o número de valores a ser testado e o nome de um arquivo, que contém a lista de valores inteiros a serem ordenados. Utilizar 3 casos de teste para realização das medições no cluster. A saída que deve ser gerada é a lista ordenada (crescente) dos valores de entrada (1 valor por linha) e também o tempo de execução da aplicação.

Os itens para avaliação são:

- implementação da versão sequencial do algoritmo;
- implementação da versão paralela do algoritmo
- medição dos tempos de execução para as versões sequencial e paralelas (usando 2, 4 e 8 processadores) para os 3 casos de teste (arquivos de entrada);
- cálculo do *SpeedUp* e eficiência de cada versão paralela para cada caso de teste;
- geração do gráfico de *SpeedUp* e eficiência para cada versão paralela com os 3 casos de teste;
- otimização do código;
- clareza do código (utilização de comentários e nomes de variáveis adequadas).

PS: A nota do Trabalho 2 será uma composição das notas dos exercícios que serão realizados em aula. **O exercício pode ser feito em duplas.**