Trabalho 3: Programação Paralela para

Multiprocessadores com Memoria Distribuída Usando MPI

- **Remoção dos 0s de um vetor**:
  - **Entrada**:
    - Vetor vIn: n elementos inteiros
  - **Saída**:
    - Vetor $vOut$: m elementos inteiros, $m \leq n$
    - $vOut$: cópia de vIn com elementos em 0 removidos
    - Elementos restantes **contíguos** em $vOut$
- **Exemplo**:
  - **Entrada**:
    - $n = 10$
    - $vIn =$

10	77	0	24	0	0	31	58	9	2
0	1	2	3	4	5	6	7	8	9

  - **Saída**:
    - $m = 7$
    - $vOut =$

10	77	24	31	58	9	2
0	1	2	3	4	5	6

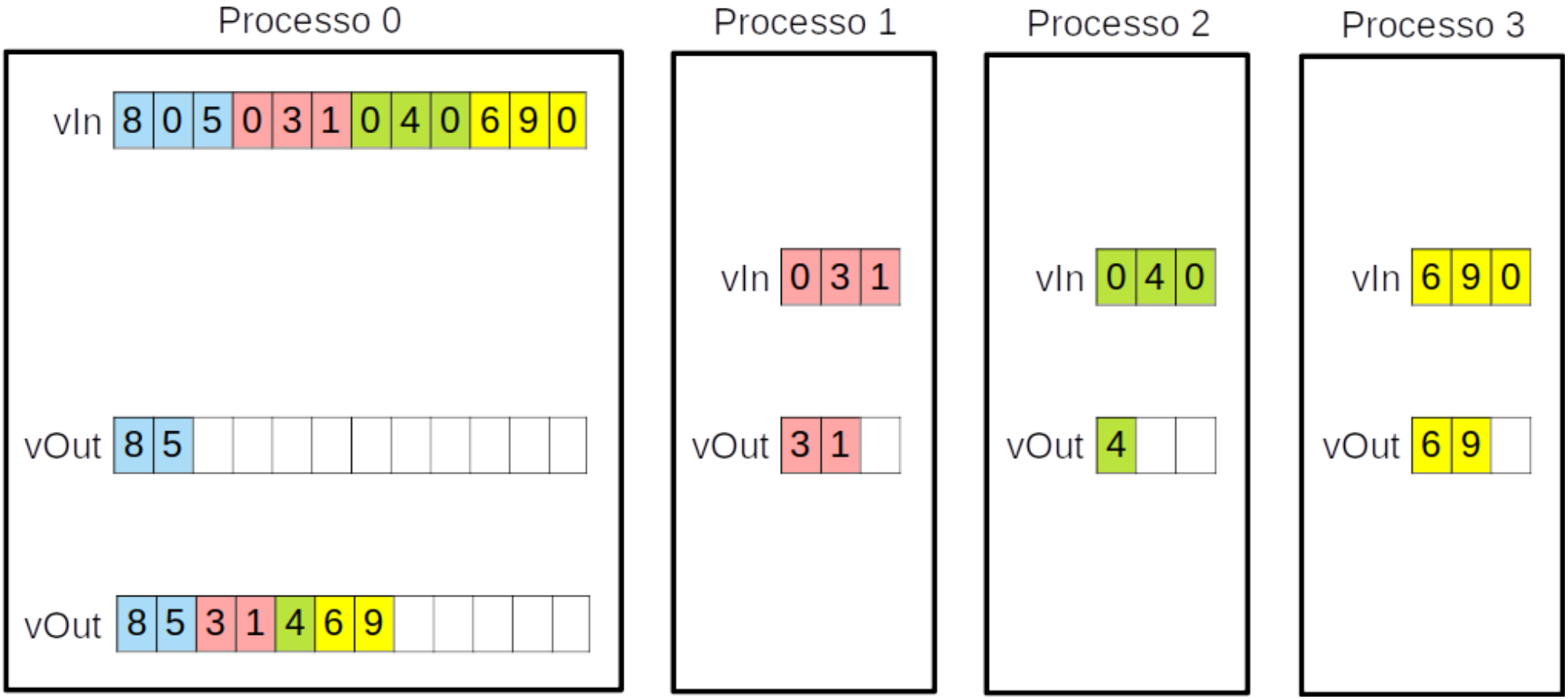
Algoritmo Sequencial para Remoção dos 0s de um Vetor

```
void remove0(int n, int *vIn, int *m, int *vOut)
{
    int i,
        c = 0;

    for (i = 0; i < n; i++)
    {
       if (vIn[i] != 0)
       {
          vOut[c] = vIn[i];
          c++;
       }
    }
    *m = c;
}
```

Ideia da Paralelização da Remoção dos 0s de um Vetor

- **Supor**: n múltiplo de p
  - $n =$ número de elementos de vIn
  - $p =$ número de processos
- **Decomposição**:
  - Cada processo remove 0s de um pedaço de vIn com n/p elementos
- **Exemplo**: $n = 12, \quad p = 4, \quad n/p = 3$



Ideia da Paralelização da Remoção dos 0s de um Vetor

- **Processo 0**:
  - **Entrada**:
    (A) Lê vIn do arquivo de entrada
  - **Corpo principal do processo**:
    (B) Passos de comunicação e computação (se necessário)
    (C) Remove 0s de um pedaço de vIn com n/p elementos
    (D) Passos de comunicação e computação (se necessário)
  - **Saída**:
    (E) Escreve $vOut$ no arquivo de saída

- **Demais processos**:
  - **Corpo principal do processo**:
    (B) Passos de comunicação e computação (se necessário)
    (C) Remove 0s de um pedaço de vIn com n/p elementos
    (D) Passos de comunicação e computação (se necessário)

ATENÇÃO: Paralelização da Remoção dos 0s de um Vetor

- **Medição de tempo**:
  - Deve englobar todo o corpo principal do processo
  - Não deve incluir entrada e saída
- **Decomposição do trabalho e paralelização**:
  - **Estruturas de dados**:
    - Processo 0: único que possui vetores vIn e $vOut$ de tamanho n
    - Demais processos: possuem vetores de tamanho n/p
  - **Processamento em cada processo**:
    - Deve ser no máximo $O(\frac{n}{p} + p)$

Entrada e Saída do Programa

- **Entrada**:
  - **Em um arquivo texto**:
    - n
    - Vetor vIn
- **Saída**:
  - **Na tela**:
    - Tempo de execução (em segundos, medido com `MPI_Wtime`)
  - **Em um arquivo texto**:
    - m
    - Vetor $vOut$
- **ATENÇÃO**:
  - **Não deve haver nenhuma outra impressão na tela ou no arquivo de saída**
  - **Arquivo de saída produzido será comparado com gabarito, usando `diff`**

Entrada e Saída do Programa

- **Arquivos de entrada e saída fornecidos**:

Arquivo	n	Arquivo	m
entrada0.txt	10	saida0.txt	7
entrada1.txt	100	saida1.txt	72
entrada2.txt	1000	saida2.txt	745
entrada3.txt	10000	saida3.txt	7574
entrada4.txt	100000	saida4.txt	76138
entrada5.txt	1000000	saida5.txt	760366
entrada6.txt	10000000	saida6.txt	7595231

- **Exemplo**: Arquivo entrada0.txt

```
10
10  77  0  24  0  0  31  58  9  2
```

n

vIn

- **Exemplo**: Arquivo saida0.txt

```
7
10  77  24  31  58  9  2
```

m

$vOut$

Programa a ser Desenvolvido

- **Programa deve ser em C ou C++**
- **Programa sequencial fornecido**
- **Desenvolver programa paralelo `remove0_par`, usando MPI**
- **Programa será compilado com**:

```
mpicc  remove0_par.c   -o remove0_par -Wall          ou
mpic++ remove0_par.cpp -o remove0_par -Wall
```

- **Programa será executado com argumentos por linha de comando**:

```
mpirun -oversubscribe -np 10 remove0_par entrada.txt saida.txt
```

  Obs.: Também serão usados outros valores para `-np`

- **Submissão no AVA**: um único arquivo .zip com programa fonte paralelo
  - **Programa deve ter no cabeçalho**:
    - Nome dos alunos do grupo: **máximo 3 alunos**
    - Outras informações (se necessário):
      - Comando diferente de compilação, ...
    - **Não** submeter programa sequencial, executável, arquivos de entrada e saída
  - **Um único aluno do grupo deve submeter**