**Universidade de Aveiro**
**Departamento de Física**

# Exame Teórico

## Computação Paralela — Módulo MPI
## 2019/2020

30 de junho de 2020

Duração: 1h15 (Teste com consulta)

**1.** No bloco de código seguinte, que faz parte de um hipotético programa paralelizado com MPI, `b` é um *array* de 2 000 000 números reais.

```
if (myid == 0) {
  for (int i = 0; i < n; ++i) {
    MPI_Ssend(b, 2000000, MPI_REAL, 3, i,MPI_COMM_WORLD);
     /* Seguem-se cálculos extensos que não alteram os valores de b */
      (...);
      /* Os cálculos extensos terminaram */
      for (int j=0; j < 2000000; ++j) b[j] = ...;
  }
}
```

a) Descreva o processo de comunicação.

b) Escreva uma linha de código para o processo 3 que garanta que ele recebe corretamente as mensagens. Assuma que o processo 3 não conhece previamente o *rank* do processo remetente nem a etiqueta da mensagem. É claro que a linha deverá estar dentro de um ciclo `for` repetido $n$ vezes, mas não se preocupe com isso.

c) Suponha que se quer substituir a função `MPI_Ssend` pela função `MPI_Issend`. Escreva a nova linha. Qual é a função do parâmetro extra?

d) Altere o bloco de código do enunciado fazendo a alteração da alínea anterior e acrescentando o que for necessário para garantir que não são enviados dados errados. Explique.

e) Em termos de performance, quais são as vantagens de usar o `MPI_Issend` em vez do `MPI_Ssend`?

**2.** Num programa MPI, o processo de *rank* 1 faz operações sobre `gr`, um *array* quadrado de dimensões $m \times m$, onde $m$ é um inteiro par. Num dado instante do programa, pretende-se que o processo 1 envie para o processo 0 a quarta parte inferior direita do *array* `gr`, de dimensões $(m/2) \times (m/2)$. Escreva as linhas de código que preparam e executam o envio, fazendo uso de

a) `MPI_Type_vector`;

b) `MPI_Type_create_subarray`.

Não tem que escrever a parte do código respeitante à receção pelo processo 0.

**3.** Na unidade curricular de Física Computacional estudou numericamente a condução de calor numa barra fina unidimensional. No programa série que desenvolveu, viu que o objetivo do trabalho era escrever uma matriz de $N_x$ linhas e $N_t$ colunas. A primeira linha contém os valores iniciais da temperatura ao longo da barra. A primeira coluna contém os valores da temperatura numa das extremidades da barra e a última coluna contém os valores da temperatura na outra extremidade. Assuma que estas condições fronteira não variam com o tempo: todos os valores da primeira coluna são iguais, todos os valores da última coluna são iguais. Cada linha representa o perfil de temperaturas da barra num dado instante. Cada coluna representa a evolução ao longo do tempo da temperatura de um ponto da barra. Usando um simples algoritmo progressivo, a temperatura do ponto $j$ no instante $i+1$ é calculada explicitamente a partir dos valores da temperatura desse ponto e dos seus vizinhos próximos no instante anterior $i$.

---

(...)
$N_x = 40\,000$
$N_t = 100\,000$
(...)
**for** $i = 0$ **to** $N_t - 2$ **do**
  **for** $j = 0$ **to** $N_x - 1$ **do**
    $T(i+1, j) = T(i, j) + \frac{C\Delta t}{(\Delta x)^2} \left[T(i, j-1) - 2T(i, j) + T(i, j+1)\right]$

---

O ciclo interior é paralelizável (cada processo pode calcular os novos valores da temperatura apenas para um subconjunto de pontos), o exterior não: antes de avançar para o instante $i+1$ têm que ser conhecidos os valores da temperatura no instante $i$ em todos os pontos $j$.

O programa foi paralelizado através das bibliotecas MPI. Foi criado um novo comunicador cartesiano `comm1` com 4 processos.

a) Escreva as linhas de código que permitem a cada processo identificar, no novo comunicador, o seu *rank* `newid` e os dos seus vizinhos próximos, `left` e `right`.

b) No programa paralelizado, cada processo trabalha com um *array* `Tlocal` com 100 000 linhas. Os *arrays* `Tlocal` dos processos 0 e 3 têm 10 001 colunas e os dos processos

1 e 2 têm 10 002 colunas. Explique porquê. Cada um dos processos fica responsável pela determinação da temperatura de quantos pontos $x$?

c) Assuma que em cada processo o inteiro **cols** contém o valor do número de colunas de **Tlocal**. O processo 0 lê inicialmente um *array* **perfilinicial** com os $N_x$ valores da temperatura no instante inicial $i = 0$. A seguir, todos os processos executam o seguinte bloco de código:

```
int mystart = 1;
if (newid == 0) mystart = 0;
MPI_Scatter(&linhainicial, Nx/4, MPI_DOUBLE, &Tlocal[0][mystart],
                ↪ Nx/4, MPI_DOUBLE, 0, comm1);
MPI_Sendrecv(&Tlocal[0][cols-2], 1, MPI_DOUBLE, right, 0,
                ↪ &Tlocal[0][0], 1, MPI_DOUBLE, left, 0, comm1,
                ↪ MPI_STATUS_IGNORE);
MPI_Sendrecv(&Tlocal[0][1], 1, MPI_DOUBLE, left, 1,
                ↪ &Tlocal[0][cols-1], 1, MPI_DOUBLE, right, 1,
                ↪ comm1, MPI_STATUS_IGNORE);
```

Explique cuidadosamente os objetivos e o funcionamento deste bloco de código.

d) Para cada valor de $i$, após ter terminado o ciclo em $j$, tem que haver comunicação entre os processos. Escreva as linhas de código necessárias.

e) No fim do ciclo em $i$, pretende-se que o processo 0 (e só o processo 0) recolha os valores da temperatura final em todos os pontos e os escreva num *array* **perfilfinal**. Escreva a linha de código com a função de comunicação coletiva a usar e as linhas de código auxiliares, quando necessárias. Se quiséssemos que todos os processos recebessem o *array* **perfilfinal**, o que tinha que ser alterado?

**4.** No fim da execução paralela de um programa MPI em 3 processos, todos eles têm na sua memória um *array* unidimensional **a** com 100 000 000 números reais de precisão dupla e um *array* unidimensional **b** com 100 000 000 números inteiros. Os valores estão sincronizados entre os processos.

Pretende-se escrever um ficheiro binário com os 100 000 000 números reais no início, seguidos dos 100 000 000 inteiros, usando as funções de I/O paralelo do MPI, distribuindo as tarefas de escrita de forma otimizada. Assumindo que um número real de precisão dupla ocupa 8 bytes de memória e um inteiro 4 bytes, escreva as linhas de código que devem ser acrescentadas ao programa para o fazer. Note que não tem que criar *subarrays* para selecionar valores de **a** ou **b** para escrever no ficheiro: como eles estão contíguos na memória, basta apontar para o endereço do primeiro.