Universidade de Aveiro
Departamento de Física

Perguntas Modelo para Exame Teórico

Computação Paralela — Módulo MPI
2019/2020

1.

a) No bloco de código seguinte, que faz parte de um hipotético programa paralelizado com MPI, `count` é um inteiro positivo conhecido por todos os processos.

```
if (myid == 0) {
  MPI_Recv(a, count, MPI_REAL, 1, 0, comm, MPI_STATUS_IGNORE);
  MPI_Send(b, count, MPI_REAL, 1, 0, comm);
}
else if (myid == 1) {
  MPI_Recv(c, count, MPI_REAL, 0, 0, comm, MPI_STATUS_IGNORE);
  MPI_Send(d, count, MPI_REAL, 0, 0, comm);
}
```

Diga, justificando, se o programa vai ficar sempre bloqueado, se vai sempre completar as duas comunicações e prosseguir, ou se vai prosseguir condicionalmente. Se a resposta foi que vai prosseguir condicionalmente, explique em que condições é que as comunicações vão ter sucesso.

b) No bloco de código seguinte, que faz parte de um hipotético programa paralelizado com MPI, `count` é um inteiro positivo conhecido por todos os processos.

```
if (myid == 0) {
  MPI_Send(b, count, MPI_REAL, 1, 0, comm);
  MPI_Recv(a, count, MPI_REAL, 1, 0, comm, MPI_STATUS_IGNORE);
}
else if (myid == 1) {
  MPI_Send(d, count, MPI_REAL, 0, 0, comm);
  MPI_Recv(c, count, MPI_REAL, 0, 0, comm, MPI_STATUS_IGNORE);
}
```

Diga, justificando, se o programa vai ficar sempre bloqueado, se vai sempre completar as duas comunicações e prosseguir, ou se vai prosseguir condicionalmente. Se a resposta foi que vai prosseguir condicionalmente, explique em que condições é que as comunicações vão ter sucesso

c) Em pelo menos uma das alíneas anteriores, a resposta certa é vai que vai prosseguir condicionalmente. Reescreva esse(s) bloco(s) de código sem alterar a sequência dos processos de *send* e *receive*, apenas alterando o modo de comunicação, de maneira a que o processo avance incondicionalmente. Atenção: adicione todas as novas linhas de código necessárias.

2. No bloco de código seguinte, que faz parte de um hipotético programa paralelizado com MPI, `count` é um inteiro positivo conhecido por todos os processos e `source` e `target` são *arrays* com `count` números reais de precisão dupla.

```
if (myid == 0) {
  MPI_Type_contiguous(count, MPI_DOUBLE, &newtype);
  MPI_Type_commit(&newtype);
  MPI_Send(source, 1, newtype, 1, 0, comm);
  MPI_Type_free(&newtype);
}
else if (myid == 1) {
  MPI_Recv(target, count, MPI_DOUBLE, 0, 0, comm, MPI_STATUS_IGNORE);
}
```

a) Explique como é que a comunicação funciona apesar de serem usados *datatypes* diferentes.

b) Imagine agora que a *array* `source` tem `3 count` elementos e que devem ser enviados para o *processo* 1 apenas os elementos $3, 6, 9 \ldots$. Reescreva o código executado pelo *processo* 0 usando um novo *datatype*.

3. Um dado programa MPI corre sempre num número de processos múltiplo de 4. Pretende-se dividir (*split*) os processos por 4 comunicadores de igual tamanho. A variável `subcomm`, do tipo `MPI_Comm`, vai ter o mesmo valor para todos os processos que ficaram em cada um dos novos 4 comunicadores. Os inteiros `subid` e `subsize` identificam, respetivamente, o *rank* de cada processo no seu novo comunicador e o tamanho desse novo comunicador.

a) Escreva o bloco de código que cria os novos comunicadores e que permite a cada processo identificar os seus valores de `subid` e `subsize`.

b) Escreva um exemplo de uma chamada de uma comunicação coletiva qualquer que inclua

   i) todos os processos;

   ii) apenas os processos que ficaram no mesmo comunicador `subcomm`.

   Basta escrever uma linha.

c) A função `double f2(double x, double y)` deve ser executada apenas pelos processos que ficaram no terceiro dos 4 novos comunicadores. Escreva o código que faz com que isto aconteça.

4. O algoritmo da parte essencial de um programa de dinâmica molecular que estuda a evolução de N partículas iguais, num domínio cúbico, é

---

begin
- declaram-se as variáveis, escreve-se o valor de N, etc.
- inicializa-se um array de N linhas e 3 colunas com as coordenadas iniciais das partículas
- inicializa-se um array de N linhas e 3 colunas com as componentes cartesianas das velocidades iniciais das partículas
- **for** *número de passos* **do**
  - **for** *partícula* **do**
    - calcula-se a força total que atua sobre a partícula a partir da sua posição e das posições das outras partículas
    - atualiza-se a posição da partícula
    - atualiza-se a velocidade da partícula
- escreve-se os resultados

Assuma que N é múltiplo de 4. Pretende-se paralelizar o programa de tal maneira que cada um de 4 processos fica responsável por atualizar as posições e velocidades de um quarto das partículas. Cada processo é sempre responsável pelas mesmas partículas, independentemente das suas posições. Todos os processos sabem o valor de N e da massa. Só o processo de *rank* 0 tem inicialmente acesso às posições iniciais `posglobal` e velocidades iniciais `velglobal` das partículas. Durante o programa, cada processo trabalha com a *array* `posglobal` que contém as posições de todas as partículas e com as *arrays* `poslocal` e `vellocal` que contêm as posições e velocidades instantâneas das partículas que é da sua responsabilidade atualizar. O novo algoritmo é

---

begin
- declaram-se as variáveis, escreve-se o valor de N, etc.
- **if** $rank = 0$ **then**
  - inicializa-se *posglobal* e *velglobal*
- cada processo recebe *posglobal* do processo de rank 0
- cada processo recebe *poslocal* e *vellocal* do processo de rank 0
- **for** *número de passos* **do**
  - **for** *partícula local* **do**
    - calcula-se a força total que atua sobre a partícula a partir da sua posição e das posições das outras partículas
    - atualiza-se a posição da partícula em *poslocal*
    - atualiza-se a velocidade da partícula em *vellocal*
  - todos os processos comunicam entre si para atualizar *posglobal*
- escreve-se os resultados

---

a) Escreva as linhas de código de todas as comunicações coletivas mencionadas no algoritmo. Não se preocupe com as declarações de variáveis, etc.

b) Suponha agora que o número de partículas não é múltiplo de N. Qual é a nova função de comunicação que tem que ser usada no processo atualização de `posglobal` no fim de cada passo? Tem que dizer qual é e explicar porquê, mas não tem que escrever o novo código.

5. Um programa de dinâmica molecular num domínio cúbico é paralelizado de tal maneira que as $N/2$ partículas com menores valores de x são atualizadas pelo processo de *rank* 0 ($x0max$ é o valor máximo de x deste conjunto de partículas) e as outras pelo processo de *rank* 1 ($x1min$ é o valor mínimo de x deste segundo conjunto de partículas).

Em cada instante, o processo 0 trabalha com um *array* de números reais de precisão dupla, com $N0$ linhas e 7 colunas. Cada linha corresponde a uma partícula. As primeiras $N/2$ linhas contêm informação sobre as $N/2$ partículas com menores valores de x e as seguintes $N0 - N/2$ linhas contêm informação sobre as partículas com valores de x entre $x0max$ e $x0max + xcut$. A justificação para isto é que das partículas com maiores x só estas $N0 - N/2$ partículas podem estar suficientemente perto das primeiras $N/2$ partículas para que a interação entre elas não possa ser desprezada.

Por seu lado, o processo 0 trabalha com um *array* de números reais de precisão dupla, com $N1$ linhas e 7 colunas. Cada linha corresponde a uma partícula. As primeiras $N1 - N/2$ linhas contêm informação sobre as partículas com valores de x entre $x1min - xcut$ e $x1min$ e as seguintes $N/2$ linhas contêm informação sobre as $N/2$ partículas com maiores valores de x. Note que em geral $N0 \neq N1$.

Em ambos os casos, para cada linha, o primeiro valor é a massa da partícula, os 3 seguintes são as coordenadas x, y e z, e os 3 últimos são as componentes cartesianas da velocidade.

No fim do programa, pretende-se escrever um ficheiro binário com as coordenadas de todas as partículas, usando as funções de I/O paralelo do MPI. Escreva as linhas de código que devem ser acrescentadas ao programa para o fazer. Assuma que para ambos os processos a *array* se chama `parts`.