Exame de Programação Paralela e Distribuída

Departamento de Ciência de Computadores
Faculdade de Ciências – Universidade do Porto
27 de Janeiro de 2009
Duração: 2 horas e 30 minutos

1. Um dos métodos mais conhecidos para desenhar algoritmos paralelos é a metodologia de programação de Ian Foster. Diga em que consistem as etapas de *Decomposição* e de *Comunicação* da metodologia de Ian Foster e descreva de forma sucinta as principais características e conceitos relacionados com cada uma dessas etapas.

2. Os *semáforos* e as *variáveis de condição* são dois dos mecanismos mais conhecidos para sincronização no acesso a zonas críticas sem envolver espera activa. Explique em que consiste cada um dos mecanismos e indique as suas principais diferenças.

3. Considere o algoritmo de Fox tal como utilizado no trabalho prático I. Numa das etapas desse algoritmo todos os processos devem enviar a sua submatriz coluna para o processo acima e receber do processo abaixo a submatriz coluna deste. Utilizando o modelo de programação MPI, esta etapa poderia ser implementada utilizando as funções `MPI_Send`/`MPI_Recv`, as funções `MPI_Isend`/`MPI_Irecv`/`MPI_Wait` ou a função `MPI_Sendrecv_replace`, tal como a seguir se apresenta:

   (a)
   ```
   MPI_Send(col_matrix,size,MPI_INT,up,tag,col_comm);
   MPI_Recv(col_matrix,size,MPI_INT,down,tag,col_comm,&status);
   ```

   (b)
   ```
   MPI_Isend(col_matrix,size,MPI_INT,up,tag,col_comm,&req_send);
   MPI_Irecv(col_matrix,size,MPI_INT,down,tag,col_comm,&req_recv);
   MPI_Wait(&req_send,&status);
   MPI_Wait(&req_recv,&status);
   ```

   (c)
   ```
   MPI_Sendrecv_replace(col_matrix,size,MPI_INT,up,tag,down,tag,
   col_comm,&status);
   ```

   Tendo por base as 3 sequências de comunicação acima apresentadas, explique os conceitos e as diferenças entre elas e indique os eventuais problemas (se algum) que podem surgir com cada uma delas.

4. Dada uma matriz $M_{L\times C}$ de valores inteiros, considere o problema de calcular num vector V os maiores elementos de cada linha da matriz M e na variável max o maior elemento de toda a matriz. Escreva um programa que tire partido do modelo de programação MPI para implementar uma solução utilizando P processadores. Assuma que P divide L e que o processo 0 é o responsável por carregar os valores de L, C e da matriz M através da chamada à função `load_values()` e de apresentar os valores calculados em V e max através da chamada à função `show_results()`.

5. A directiva `#pragma omp critical` do OpenMP permite definir zonas críticas de código que devem ser executadas de forma atómica. As sequências de código que se seguem usam incorrectamente esta directiva para resolver o problema de calcular em paralelo o valor máximo de um determinado vector `a[]`. Identifique qual é o problema em cada uma delas e, mantendo inalterável a declaração `#pragma omp parallel for`, apresente uma solução que utilize igualmente a directiva `#pragma omp critical`.

   (a)
   ```
   max = a[0];
   #pragma omp parallel for private(i) shared(a,max)
   for (i = 1; i < N; i++)
     if (a[i] > max)
       #pragma omp critical
       max = a[i];
   ```

   (b)
   ```
   max = a[0];
   #pragma omp parallel for private(i) shared(a,max)
   for (i = 1; i < N; i++)
     #pragma omp critical
     if (a[i] > max)
       max = a[i];
   ```

6. Identifique as dependências *loop-carried* existentes no bloco de código que se segue, classifique-as quanto ao tipo de fluxo de dados (*flow dependence/anti dependence/output dependence*) e escreva uma versão paralela do código em OpenMP com as dependências removidas (assuma que o valor da variável `i` não é necessário na continuação da computação).

   ```
   for (i = 1; i < N; i++) {
     x = x + a[i];
     y = a[i] + i;
     b[i] = b[i] + c[i - 1];
     c[i] = c[i] + y;
   }
   ```

7. Considere uma aplicação paralela cujo tempo de computação divide-se em três partes:

   A - computação sequencial utilizando um único processador

   B - computação paralela utilizando metade dos processadores disponíveis

   C - computação paralela utilizando todos os processadores disponíveis

   Por experimentação, num multiprocessador com 100 processadores, verificou-se que 5% do tempo de computação é passado na parte A. Calcule o tempo máximo da computação que pode ser passado na parte B para que o *speedup* da aplicação nesse multiprocessador seja 75. Segundo a lei de Gustafson-Barsis, qual é o speedup máximo da aplicação nesse multiprocessador.