Computação de Alto Desempenho 2013/14
Exame – Parte B – 23/6/2014
Duração: 1h30

O exame é sem consulta e em caso de dúvida na interpretação do enunciado, deve explicitar todos os pressupostos na elaboração das suas respostas.

1. Considere o problema de calcular quantos elementos comuns existem entre duas matrizes de inteiros de igual dimensão (i.e. quantos elementos iguais existem na mesma posição nas duas matrizes), e em que posição se encontram. Pretende-se assim construir uma matriz binária (com zeros e uns) onde são registadas as posições dos elementos comuns, bem como escrever quantos elementos comuns foram encontrados.
Considere que os valores de entrada são colocado em duas matrizes designadas por *matrix_in1* e *matrix_in2*, ambas com dimensão 1000 por 1000 bytes, e o resultado é guardado numa matriz denominada *matrix_out*, também de 1000 por 1000 bytes. Já o número de ocorrências comuns é afixado no final do programa.
Implemente esta problema num programa em ***Cuda C***, preenchendo o seguinte esqueleto de código em C, com o seu código necessário em ***Cuda C***.

```
#include <stdio.h>
#include <stdlib.h>

#define N  1000
int matrix_in1[N][N], matrix_in2[N][N];
int matrix_out[N][N];

// kernel function

int main (int argc, char *argv[])
{
   FILE *f; int elementos_iguais = 0;
   // Leitura das matrizes
   f = fopen("input.bin", "r");
   fread(matrix_in1, sizeof(int), N*N, f);
   fread(matrix_in2, sizeof(int), N*N, f);
   fclose(f);

   // o seu código CUDA C

    // Escrita dos resultados
   f= fopen("result.bin", "w");
   fwrite(matrix_out, sizeof(int), N*N, f);
   fclose(f);
   printf("Número de elementos iguais: %d\n", elementos_iguais);
   return 0;
 }
```

2. Considere um programa massivamente paralelo escrito em MPI que processa um vector de inteiros ***V*** de dimensão ***n***, usando para o efeito um conjunto de nós ligados por uma rede de comunicação. Explique como poderia usar as

funções de comunicação colectiva, de modo a implementar as seguintes operações:

a) Inicialmente apenas o nó de rank 0 contém os valores do vector V, e pretende-se calcular de forma distribuída qual o maior elemento do vector. Deve também ser o nó de rank 0 a afixar o resultado.

b) o vector V inicialmente não tem valores válidos, mas no final deve ficar com as potências de dois correspondente a cada posição. Ou seja, V[0] == 1, V[1] == 2, V[2] == 4, V[3] == 8, ... V[n] == 2^n. Para mais, o vector V alterado deve ficar disponível em todos os nós.

3. Considere o problema do cálculo da figura de Mandelbrot. Suponha disponível a seguinte função:

```
#define max_iterations 255
int compute_point(double ci, double cr) {
   int iterations = 0;
   double zi = 0;
   double zr = 0;
   while ((zr*zr + zi*zi < 4) && (iterations < max_iterations))
   {
        double nr, ni;
        nr = zr*zr - zi*zi + cr; ni = 2*zr*zi + ci;
        zi = ni; zr = nr;
        iterations ++;
   }
   return iterations;
 }
```

Pretende-se construir o chamado histograma de Mandelbrot, i.e. saber para quantos pontos é o número de iterações igual a 1, par quantos é 2, etc, até para quantos é 255.
Usando as primitivas do MPI, programe uma versão paralela deste algoritmo para um conjunto de 16 computadores pessoais interligados por uma rede local a 1 Gbps, no qual está instalado o MPI. Suponha a seguinte linha de comando:

```
mpiexec -np 16 –hostfilename  mandelbrot_histogram x1 y1 x2 y2 L H
```

Em que

- *x1*, *y1* são as coordenadas do canto superior esquerdo da área rectangular considerada.
- *x2*, *y2* são as coordenadas do canto inferior direito da área rectangular considerada.
- L é o número de pixels na horizontal.
- H é o número de pixels na vertical.

4. Considere o problema de calcular quantos elementos comuns existem entre duas matrizes de inteiros de igual dimensão e em que posições ocorrem, tal como descrito na pergunta 1. Discuta em que situações haveria vantagem em ter uma implementação híbrida (OpenMP+MPI) do problema, em vez de uma implementação em CUDA.