Computação de Alto Desempenho 2014/15

2nd Test – 5/6/2015

Duration: 2h00

Closed book test; possible doubts about the contents should be solved by the student; please include in your answers the assumptions you made.

1. Assume an application taking too long to execute that is to be executed in a GPU platform. Write, and justify, which are the performance optimisation strategies to be considered to adapt that application in order to take the best advantage of a GPU execution.

2. The "*bi-segmentation*" algorithm is extensively used in image processing, e.g. it is one of the steps of object identification in Materials Science's images. For instance, starting from a 2D tomographic image of a composite material with two materials A and B that have to be differentiated, the algorithm uses two values between 0 and 255 (0- black colour; 255- white colour), *L1* and *L2*, which represent the colour of those two materials. Based on the two values, the algorithm applies the following operation to all the pixels in the 2D image:

```
if ( pixel_color < L1 )
   pixel_color = 0;   //pixel gets the black color (material A)
else if (pixel_color > L2 ) {
   pixel_color = 255; //the pixel gets the white color (material B)
else
   pixel_color = 128; //the pixel gets the grey color (indetermined)
```

Implement this algorithm using ***Cuda C*** assuming that the 2D image occupies N*N bytes and each byte contains the pixel's colour at that position. Thus fill in the following C code with your code in ***Cuda C*** assuming there is enough space in the *device* for at least two images:

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

#define N ...
unsigned char material[N][N]; // material's initial image
unsigned char *pt_img;        // pointer to the processed image

// your kernel function

int main (int argc, char *argv[])
{
   FILE *f;
   // Image's input
   f = fopen("input.bin", "r");
   fread(material, sizeof(unsigned char), N*N, f);
   fclose(f);

   // your CUDA C code

   // Writing the processed image
```

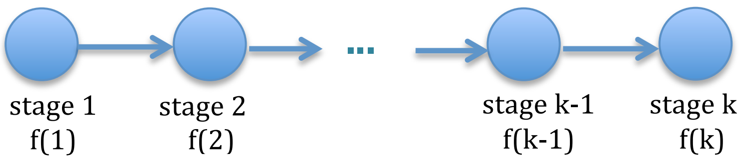
```
   f= fopen("simulation.bin", "w");
   fwrite(pt_img, sizeof(float), N*N, f);

   return 0;
}
```

3. Consider a computational platform composed of ***N*** personal computers connected by an 1 Gbps local network where the *MPI* is installed. Explain how the following functionalities may be implemented in this setting, specifically which *MPI collective communication functions* can be used in each case:

a) Processing a vector ***V*** of integer numbers with dimension ***d*** *(**d** is much bigger than **N**)* to count how many elements in the vector have the same value as ***val***; the result must be available at all nodes (and not only at the node with *rank 0*).

b) Implementing a *pipeline with **k** stages (**k** < **N**)* to process a *stream* of data values; at each *pipeline's stage i*, the function *f(int stage)* is applied with *argument i* to transform the data, as shown in the following figure:



Assume that the *rank 0 node* is continuously receiving the data from a channel *f_in* and the data processed by the *pipeline* is written to the channel *f_out*. The command line of the program might be:

```
mpiexec -np k_stage –hostfilename stream_processing f_in f_out
```

4. The problem of calculating the *Mandelbrot's figure* use the following function as basis:

```
#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;
}
```

This function identifies if a point belongs or not to the Mandelbrot's figure and it may be applied in parallel to all points in an image; the returned value may be subsequently used to identify the pixel's colour.
The goal of the program in this question is to use the above function to calculate the *Mandelbrot's histogram*, i.e. to calculate for how many points the returned number of iterations is equal to 1, for how many points is returned the value of 2 iterations, and so on, until the returned value of 255. The calculation of the histogram is to be parallelised using the functions of the *MPI* assuming a cluster of 16 personal computers connected by a *1Gbps* network where the MPI was installed.

Assume that the command line could be:

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

where

- *x1*, *y1* are the coordinates of the left corner on top of the rectangular area defining the image to be built;
- *x2*, *y2* are the coordinates of the right corner on bottom of the rectangular area defining the image to be built;
- L is the number of pixels in the horizontal dimension (width);
- H is the number of pixels in the vertical dimension (height).

Describe how to apply the Foster's methodology to develop a parallel solution in the described conditions. Justify, for each phase of this methodology, the options you make.

5. Identify in which situations a *hybrid implementation* with *OpenMP+MPI* to parallelise a problem is better than its implementation in CUDA. You may consider as example the problem of applying a function to all elements in a matrix of bytes like the example in *question 2* above.