SF2568: Homework 1

Michael Hanke

November 18, 2015

1. Describe the difference between a *process* and a *processor*! (1)

2. A multiprocessor consists of 100 processors, each capable of a peak execution rate of 2 Gflops (i.e. 2×10^9 floating operations per second). What is the performance of the system as measured in Gflops when 10% of the code is sequential and 90% is parallelizable? (1)

3. Is it possible for a system to have a system efficiency (η_P) of greater than 100%? Discuss. (1)

4. In the Parallel Rank Sort method presented in the lecture, the number of processors must be the same as the number of elements in the list. Assume that the number of elements in the list is actually ten times larger than the number of processors in the computer. Describe a modification to handle this situation. (1)

5. You are given some time on a new parallel computer. You run a program which parallelizes perfectly during certain phases, but which must run serially during others.

   (a) If the best serial algorithm runs in 64s on one processor, and your parallel algorithm runs in 22s on 8 processors, what fraction of the serial running time is spent in the serial section? (1)

   (b) Determine the parallel speedup. (1)

   (c) You have another program that has three distinct phases. The first phase runs optimally on 5 processors; this is, performance remains constant on 6 or more processors. The second phase runs optimally on 10 processors, and the third on 15. The phases consume 20%, 20%, and 60% of the serial running time, respectively. What is the speedup on 15 processors? (2)

6. For the following problem, you need to have access to MPI on one of the platforms (preferably tegner).

   Implement the Mandelbrot algorithm in a parallel environment!

   There are some issues which need to be considered:

- When using MPI you usually do not have access to a graphics system. Therefore, the display of the final image cannot be handled by your parallel program. I recommend that you write the final result to hard disk. Then use matlab or another program of your choice for a graphical representation.
- The file can be saved either in binary or in ASCII (human readable) format to hard disk. Please note that different machines have usually different binary formats! On the other hand, an ASCII file needs much more disk space. With the size of the image given below, the ASCII file may occupy up to 30MB on hard disk.
- The recommended data type for `color` is `unsigned char` (in C) or `INTEGER*2` (in Fortran). The following code snippets indicate how an ASCII file could be generated:
  - in C:

```
#include <stdio.h>
#define M 2048
#define N 2048
unsigned char color[M*N];
FILE *fp;
/* your computations */
fp = fopen(‘‘color.txt’’,’’w’’);
for (j = 0; j < N; j++) {
   for (i = 0; i < M; i++)
      fprintf(fp, ‘‘%hhu ‘‘, color[i+j*M]);
   fprintf(fp, ‘‘\n’’);
}
fclose(fp);
```

  - in Fortran:

```
      PARAMETER (m = 2048, n = 2048)
      INTEGER color(m,n)
C  your computations
      OPEN(1, FILE='color.txt', ACTION='WRITE')
      DO j = 1, n
         WRITE (1,*) (color(i,j), i = 1,m)
      ENDDO
      CLOSE(1)
```

  - In matlab, both files can simply be read using the command

```
load -ascii color.txt
```

  - Note that for this to work, the array `color` must be gathered at the master process.

- Compare the description of Lab 1 in order to understand how to run programs on tegner.

(a) Implement the Mandelbrot program using MPI. You can assume that the number of processors divides the number of columns evenly. (Test this in your program!!) (4)

(b) Reproduce the figure from the lecture notes. (You may play around with different palettes in matlab.) (1)

(c) Magnify some interesting parts of the figure. Do this by computing only parts of the complete picture using higher resolutions. (2)

You should hand in: the source code of your program, a printout of your figures (if possible in color).