Lab 3: Embarrassingly parallelism with OpenMP: Mandelbrot set

par4111

Adrià Cabeza, Xavier Lacasa
Departament d' Arquitectura de Computadors

April 9, 2019
2018 - 19 SPRING

Contents

1 Introduction

This second laboratory assigment consists on a series of tests and modifications on a program called Mandelbrot. The program computes a particular set of points belongig to the complex domain,whose boundary generates a distinctive and recognisable two-dimensional fractal shape.

2 Task decomposition and granularity analysis

In this section we will explain the different task decomposition strategies and granularities explored using Tareador and the Mandolbrot program.

Which are the two most important common characteristics of the task graphs generated for the two task granularities (Row and Point) for the non-graphical version of mandel-tareador? Obtain the task graphs that are generated in both cases for -w 8.

Firstly, we will do a task at each single point (row,col) level. Please look at the Figure **??** to see the dependency graph obtained at that level of granularity.

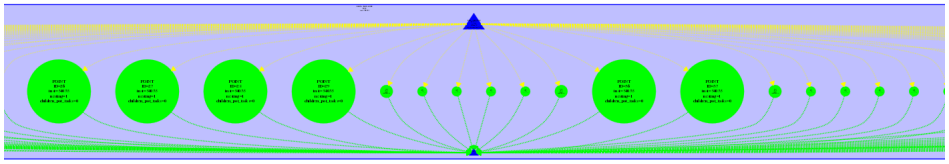

Figure 1: Part of the dependency graph for point decomposition

The second one consists on doing the tasks at the row level. Please look at the Figure **??** to see the dependency graph obtained at that level of granularity.

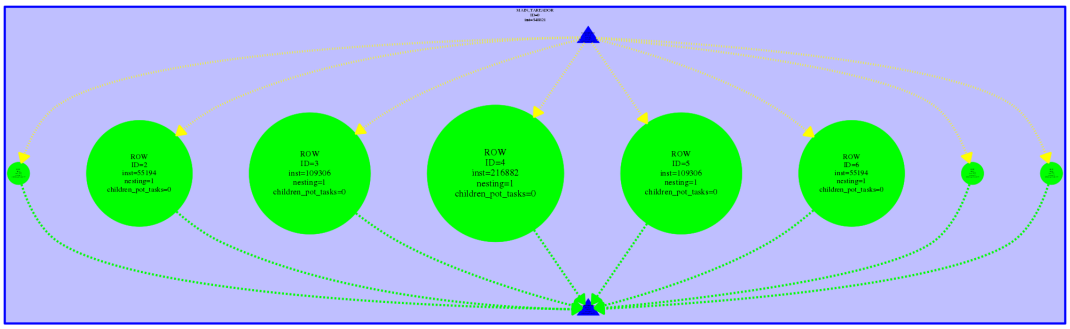

Figure 2: Dependency graph for point decomposition

If we compare each dependency graph we can observe that the load between different tasks is not equally distributed: the sizes of the tasks have very different

values. For this reason we decided that to create tasks in the *static mode* is not the best option. We should use *dynamic mode* instead. Also we can observe that there is no shared data between tasks so there are no dependencies.

Which section of the code is causing the serialization of all tasks in mandeld-tareador? How do you plan to protect this section of code in the parallel OpenMP code?

The previous analysis was made using the *mandel-tar* program which does not show the fractal on the screen. If we analyse the program that shows the image (*mandeld-tar*, we get the following dependency graph (Figure **??**).

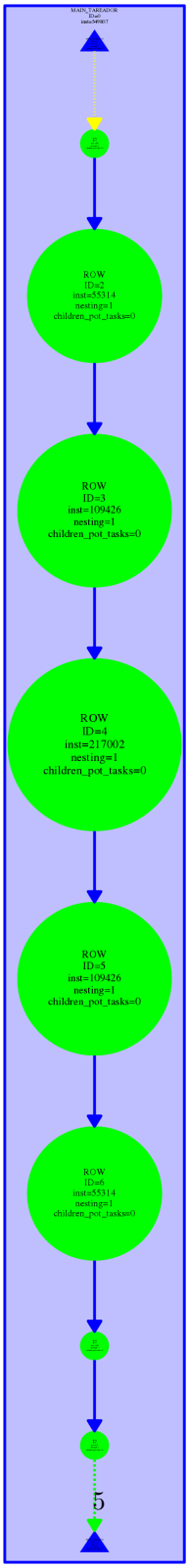

Figure 3: Dependency graph for *mandeld* program

It is easy to conclude that we have obtained a sequential version of the program with a lot of dependencies that do not allow to parallelize. The reason for this behaviour is the way the results are printed in the screen (we need to print the dots line by line so parallelism is useless here). In order to parallelize the code we should modify it as follows:

```
#pragma omp critical
{
   XSetForeground(display,gc,color);
   XDrawPoint(display,win,gc,col,row);
}
```