PAR – Final Exam Laboratory – Course 2024/2025-Q1

Student name: .... Subgroup: ....

January 16th, 2025
Grade Publication January 22^{nd}, 2025 - Exam Review 23^{rd} - 16:00h–17:00h - C6E106

Disclamer: Only few solutions are given since all the answers are part of the practices done during the lab sessions.

Problem 1: Lab 3 (3.0 points) Given the following excerpt of the instrumentation with `Tareador` of the iterative `mandelbrot` code:

```
for (int py = y; py < y + TILE; py++) {
   tareador_start_task("mandelbrot_py_fill");
   for (int px = x; px < x + TILE; px++) {
      M[py][px] = M[y][x];
      if (output2display) {
         if (setup_return == EXIT_SUCCESS) {
             XSetForeground(display, gc, color);
             XDrawPoint(display, win, gc, px, py);
   } } }
   tareador_end_task("mandelbrot_py_fill");
}
```

We ask you to: Draw the Task Dependency Graph (TDG) corresponding to this part of the code, assuming $TILE = 4$ and the mandelbrot program is executed with `Tareador` and: case 1) no arguments and case 2) `-d` argument. For each TDG, and in case you find dependencies, indicate if they are due to data sharing and how you would deactivate them using `Tareador` API. Finally, how will you protect or avoid them, if necessary, in your OpenMP implementation?.

Solution:

No comments are given because it is part of the lab sessions.

Problem 2: Lab 4 (4.0 points) After doing the analysis with `Modelfactors` and `Paraver`, the speedup and efficiency (speedup, efficiency in %) of three of the four `Mandebrot` iterative and recursive parallel implementations in lab4, when running with 16 threads, are: 1) (1.1x,7%), 2) (4.4x,27%) and 3) (13.0x,80%) approximately. As a reminder, the names of the four implementations are: Iterative `Tile`, Iterative `Fine Grain`, Recursive `Leaf` and Recursive `Tree`. **We ask you to answer the following questions:**

1. (1.0 points) Which implementation is executed to obtain each result? Briefly reason your answer.

   Solution:

   No comments are given because it is part of the lab sessions. 1) Leaf Recursive, 2) Tile (original), 3) Fine grain Iterative.

2. (1.5 points) For the fine grain task decomposition, the recursive tree task decomposition and the recursive leaf task decompositon, assume you visualize a `Paraver` execution trace for each of them using `OpenMP tasking/explicit tasks function created and executed` hint. How many threads do you expect to find creating tasks for each implementation? Briefly reason your answer.

   Solution:

   No comments are given because it is part of the lab sessions.

3. (1.5 points) For the tile (original) task decomposition, the recursive tree task decomposition and the recursive leaf task decompositon, draft a picture of the expected Instantaneous Parallelism view using 16 threads. Briefly explain your pictures.

   Solution:

   No comments are given because it is part of the lab sessions.

Problem 3: Lab 5 (3.0 points) Assuming the implementation of `Mandelbrot` using a 1D cyclic data decomposition by rows, **we ask you to answer the following questions:**

1. (0.25 points) What is the value of *phi* you expect to obtain with `Modelfactors`?

   Solution:

   No comments are given because it is part of the lab sessions.

2. (0.5 points) Assume the loop body of the inner-most loop of the mandel function to be: `M[py][px]=...` (you do not have to write the code to display the image and to update histogram, just write "..."). Write the excerpt of the parallel code of the 1D cyclic data decomposition by rows of `mandel` function.

   Solution:

   No comments are given because it is part of the lab sessions.

3. (0.75 points) What is the approximate expected order of magnitude in the number of *l*2 misses of your code using 20 threads? Do you think a 1D cyclic data decomposition **by columns** will be more efficient thant yours in number of *l*2 misses? Briefly reason your answer.

   Solution:

   No comments are given because it is part of the lab sessions. However, it cannot be better than the version by columns because the version by columns will have much more number of l2 misses; it will not exploit the spatial locality at all.

4. (0.25 points) What is the approximate expected % load balancing (`Modelfactors` metric) of your code using 20 threads? Briefly reason your answer.

   Solution:

   No comments are given because it is part of the lab sessions.

5. (0.5 points) What is the approximate expected speedup with 20 threads? And, if we decide to use 21 threads, instead of the 20 threads, which is the expected speedup increment? Briefly reason your answer.

   Solution:

   No comments are given because it is part of the lab sessions. Remember lab1: with 21 threads we will have to share one physical core and we saw that this is not good for performance.

6. (0.75 points) Are 1D block-cyclic data decomposition by columns or 1D block data decomposition by columns strategies more globally efficient (`Modelfactors` metric) than 1D cyclic data decomposition by rows? Why? Briefly reason your answer.

   Solution:

   No comments are given because it is part of the lab sessions.