LAB 4

Branch and bound with OpenMP: N-queens puzzle

2018-2019 Q1

Par2013

Daniel Palomo Cabrera i David Soldevila Puigbi

Introduction

First of all, we are going to understand the nqueens algorithm. It consist of a recursive algorithm that tries to put N chess queens into a N x N chess table in order that every queen doesn’t attack any other queen (no column, row or diagonal is shared).

Understanding the potential parallelism in Nqueens

Executing the sequential version of the program with 12 queens in a table of 12 x 12 the program gives us the following output:

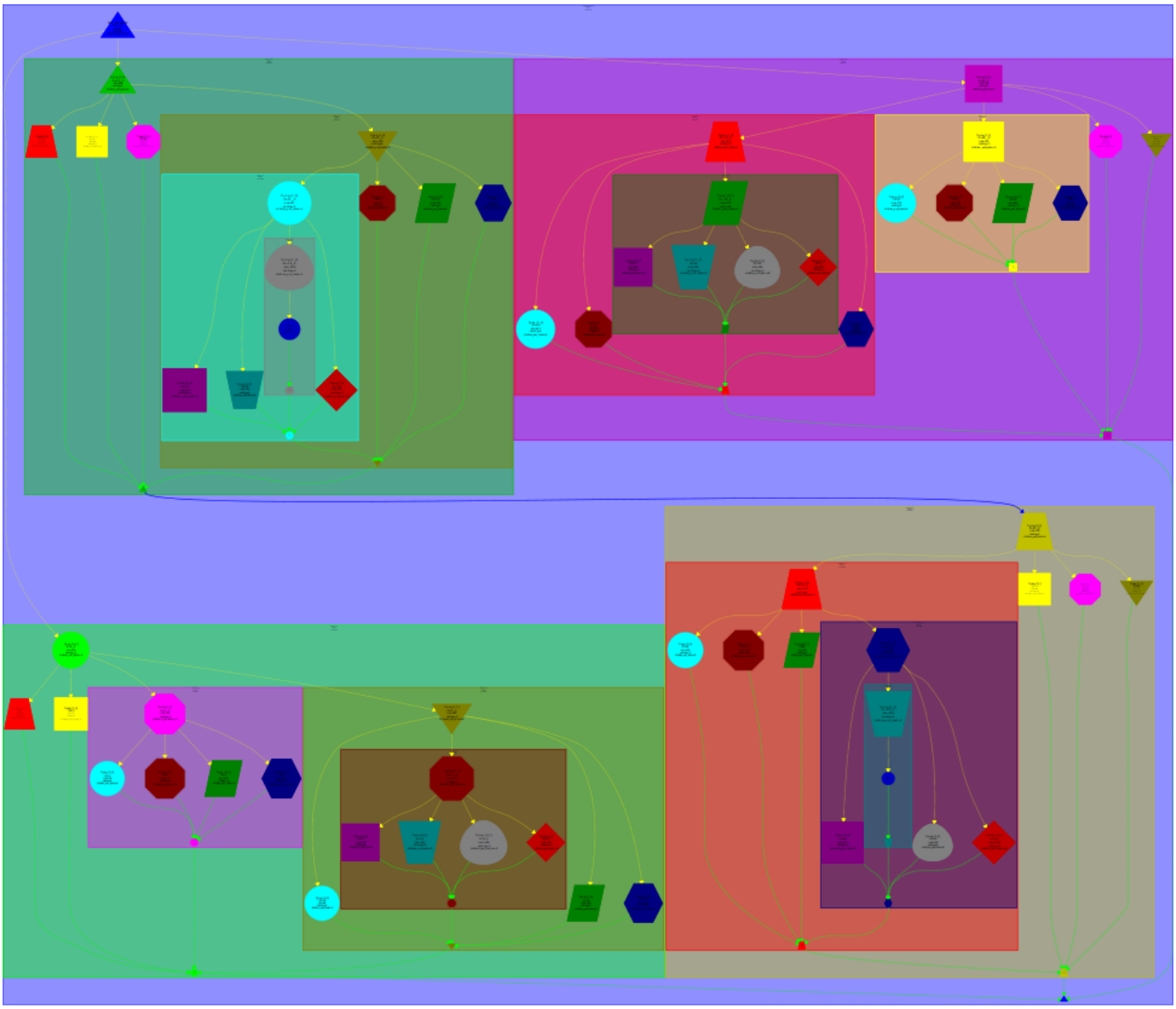
```
  board size:              12
  recursion cutoff level:  8

one solution:  0  2  4  7  9 11  5 10  1  6  8  3

number of solutions: 14200
```

The code of nqueens fuctions with tareadors calls:

```
void nqueens(int n, int j, char *a, int d) {
     int i;

     if (n == j) {
#ifdef _TAREADOR_
          tareador_start_task("Solution");
#endif
          /* put good solution in heap. */
        if( sol == NULL ) {
                        sol = malloc(n * sizeof(char));
                        memcpy(sol, a, n * sizeof(char));
        }
        sol_count += 1;

#ifdef _TAREADOR_
          tareador_end_task("Solution");
#endif
     } else {
          /* try each possible position for queen <j> */
          for ( i=0 ; i < n ; i++ ) {

#ifdef _TAREADOR_
                  sprintf(stringMessage,"Trying [%d,%d]",j,i);
                  tareador_start_task(stringMessage);
#endif
                        b[j] = (char) i;
                        if (ok(j + 1, b)) {
                                nqueens(n, j + 1, b,d+1);
                        }
#ifdef _TAREADOR_
                        tareador_end_task(stringMessage);
#endif
          }
     }
}
```

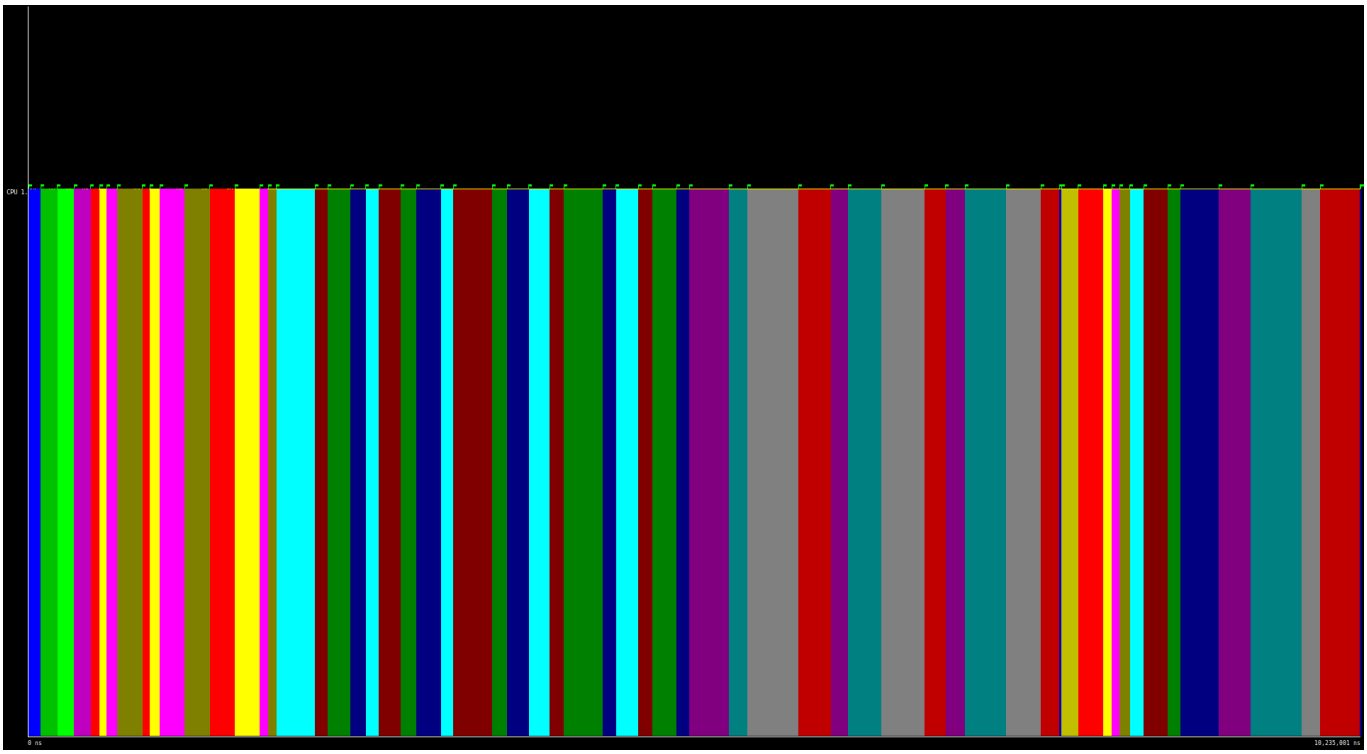
nqueens-tar.c

Lets discuss about the parallelization of the algorithm using Tareador. First take a look at the dependence graph for the execution of the algorithm with n equals 4.

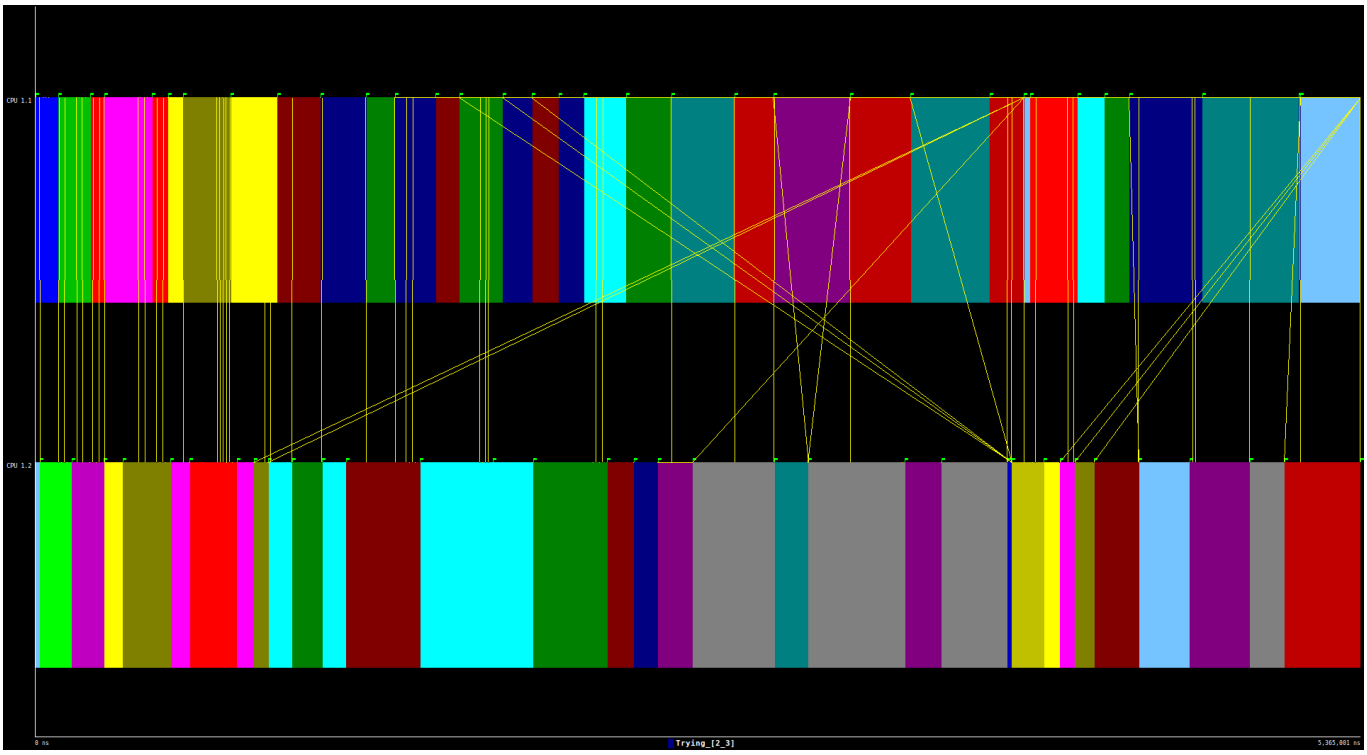


Dependence grpah generated by Tareador

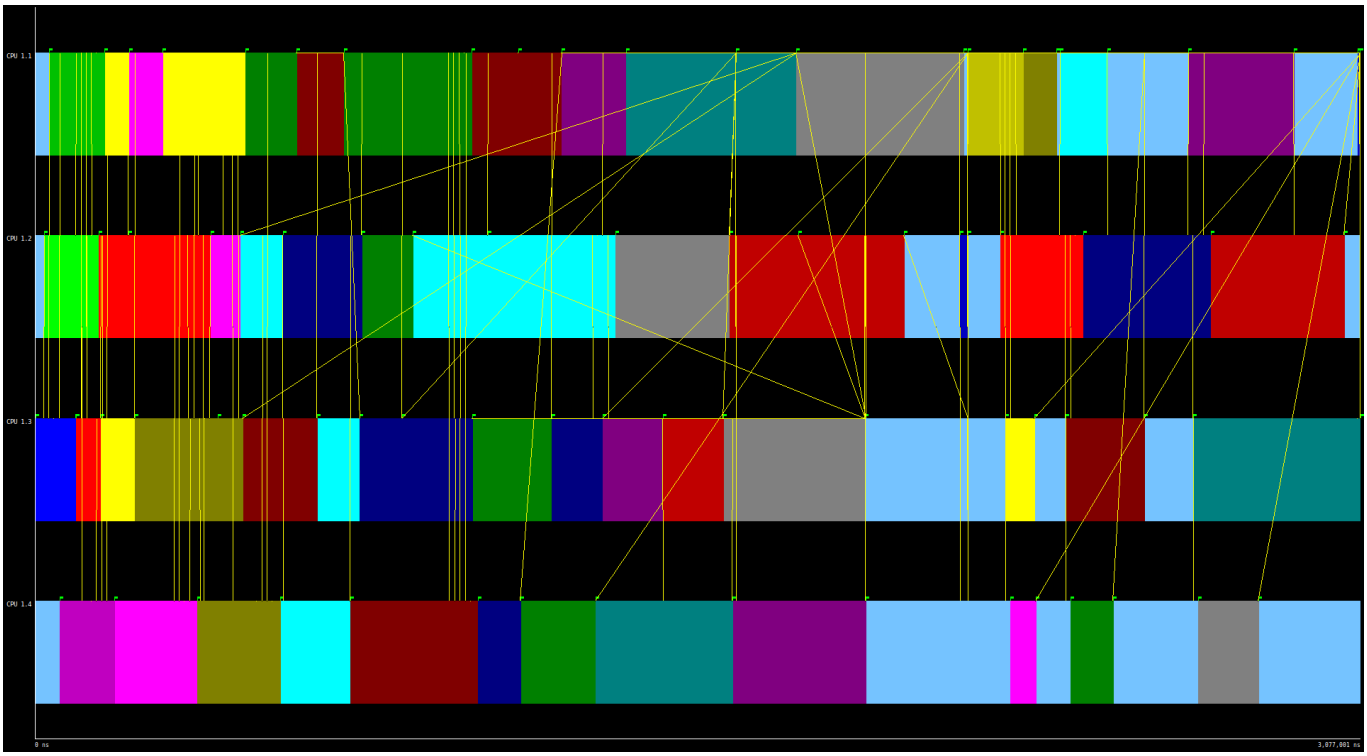
Even thought Tareador shows that block with id 1 (bottom left) at the second row it can be executed at the same time of the blocks in the first row. Note this at the simulation with different number of cores.



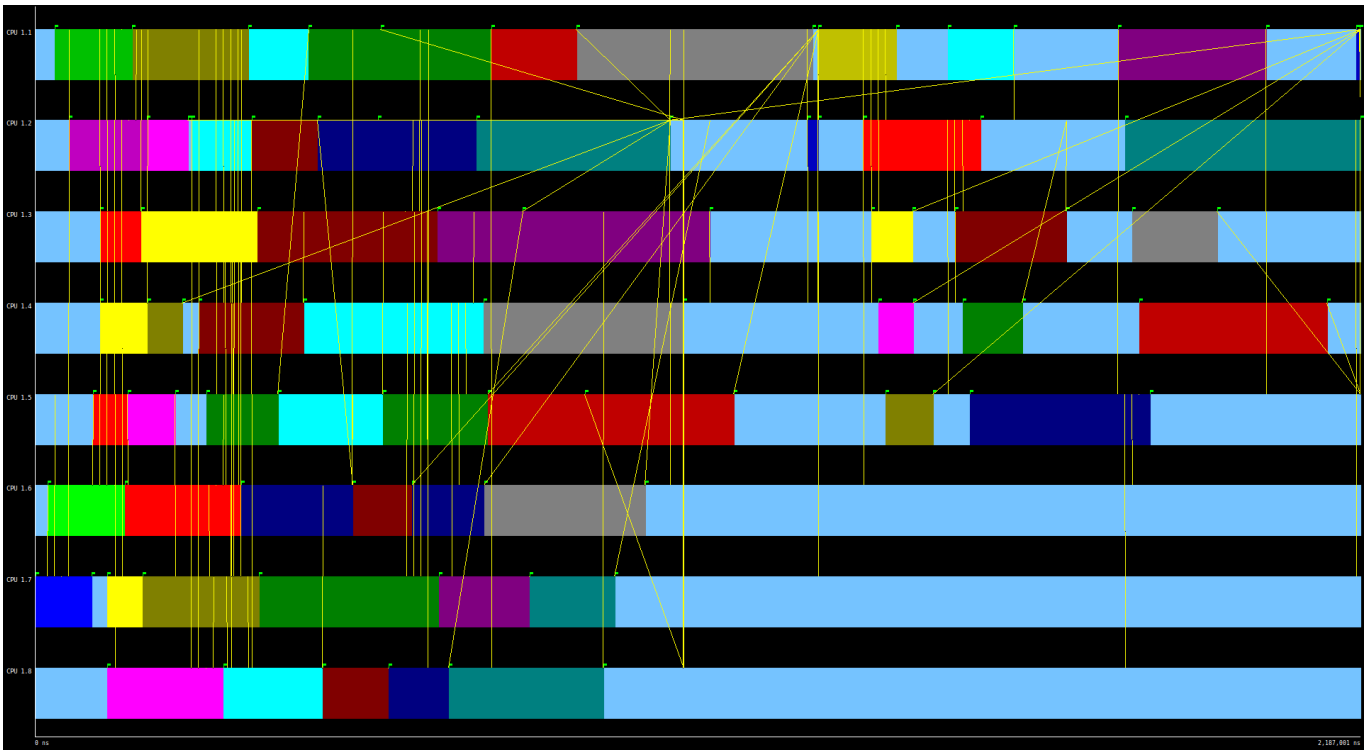
Simulation of execution of nqeens using 1 core



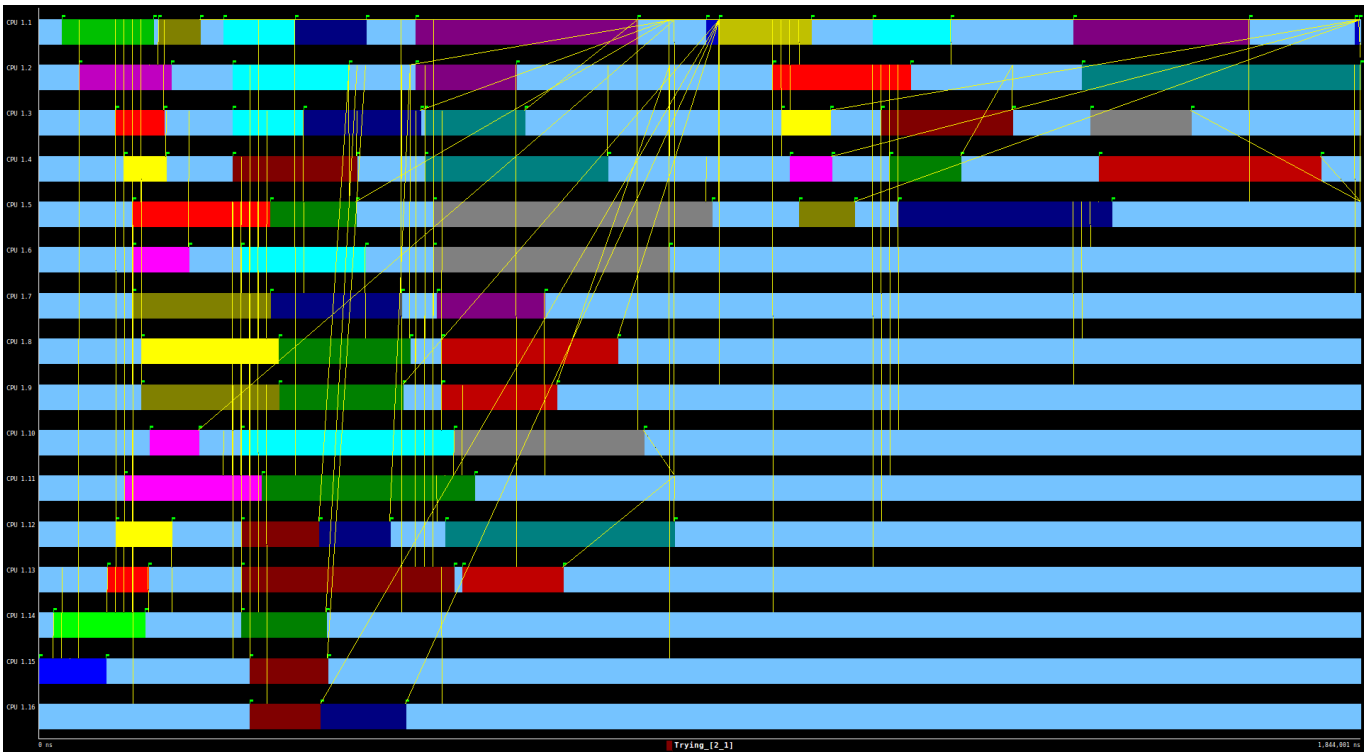
Simulation of execution of nqeens using 2 core



Simulation of execution of nqeens using 4 core



Simulation of execution of nqeens using 8 core



Simulation of execution of nqeens using 16 core

With 32 cores the time is exactly the same as the 16 cores simulation.

Shared-memory parallelization

At last we will parallelice the nqueen algorithm using OpenMP.

In order to reach a good level of parallelization for each task the program creates a copy of the chess board. That strategy eliminates the dependence of reading and writing the chess board.

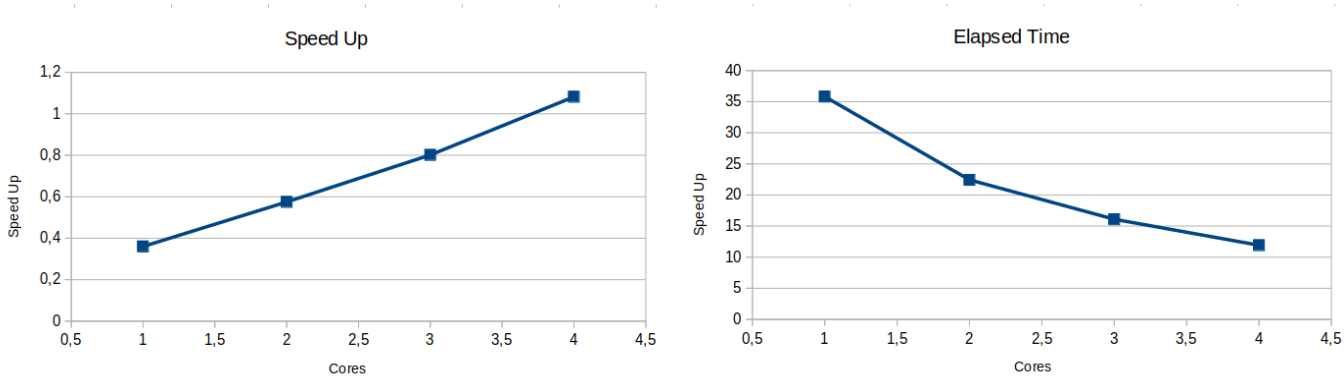
Last, we introduced a critical zone at the basic case, only one thread should be the one who edits the number of solutions. And the resulting code is:

```
void nqueens(int n, int j, char *a, int d) {
        int i;
        if (n == j) {
        #ifdef _TAREADOR_
                tareador_start_task("Solution");
        #endif
                /* put good solution in heap. */
        #pragma omp critical
        {
                if( sol == NULL ) {
                        sol = malloc(n * sizeof(char));
                        memcpy(sol, a, n * sizeof(char));
                }
                sol_count += 1;
        }
        #ifdef _TAREADOR_
        tareador_end_task("Solution");
        #endif
        } else {
                /* try each possible position for queen <j> */
                for ( i=0 ; i < n ; i++ ) {
                        #ifdef _TAREADOR_
                                sprintf(stringMessage,"Trying
[%d,%d]",j,i);
                                tareador_start_task(stringMessage);
                        #endif
                        char* b = alloca(sizeof(a));
                        memcpy(b, a, sizeof(a));
                        #pragma omp task final (d<CUTOFF) mergeable
                        {
                                b[j] = (char) i;
                                if (ok(j + 1, b)) {
                                        nqueens(n, j + 1, b,d+1);
                                }
                        }
                        #ifdef _TAREADOR_
                                tareador_end_task(stringMessage);
                        #endif
                }
                #pragma omp taskwait
```

```
            }
    }
```

nqueens-omp.c

With that changes we got the following elapsed time and speed up polts.



Speed Up and elapsed time plots for n=13

Note that scalability is not as perfect as the previous sessions. Now we are going to discuss the performance with paraver captures (if it works).

```
david@xps-13-9360:~$ ssh -X par2103@boada.ac.upc.edu
par2103@boada.ac.upc.edu's password:
Warning: No xauth data; using fake authentication data for X11 forwarding.
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-39-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

*** System restart required ***
Last login: Tue Dec  4 22:32:39 2018 from 196.red-83-39-13.dynamicip.rima-tde.ne
t
Warning: No xauth data; using fake authentication data for X11 forwarding.
Last login: Tue Dec  4 22:32:39 2018 from boada-server-int.ac.upc.es
Intel(R) Parallel Studio XE 2018 Update 2 for Linux*
Copyright (C) 2009-2018 Intel Corporation. All rights reserved.
```

However, we can't execute paraver, tareador or any other program with GUI, in addition, we cannot create any paraver trace because theres no available disk space. But we only have the code of the previous laboratory and the necessary for this one.

Therefore we can't fully complete this deliverable as asked