

Lab3 SO

The top screenshot shows the Visual Studio Code interface with the file `SudokuValidator.c` open. The terminal output displays a 9x9 grid of numbers and logs for multiple threads checking rows and columns. The bottom screenshot shows the same interface with a modified `SudokuValidator.c` file. The terminal output shows a different 9x9 grid and logs for threads checking rows and columns. The status bar at the bottom of the second screenshot indicates the file is `Lab4_2WithoutMutex.cpp`.

1. ¿Qué es una race condition y porqué hay que evitarlas? Una race condition es una condición donde dos o más threads pueden acceder a cierta memoria compartida generando que se modifique dicha memoria de manera que no sea lo esperable.
2. ¿Cuál es la relación, en Linux, entre pthreads y clone()? ¿Hay diferencia al crear threads con uno o con otro? ¿Qué es más recomendable? Un pthread o posix thread es un lightweight process o un proceso de bajo consumo, así mismo, estos son creados a partir de funciones de pthreads, sin embargo, en el background realmente se ejecuta una llamada a sistema de clone que sirve para crear/clonar procesos y poder indicarle a un proceso que recursos compartirá, así como de datos y demás. Realmente no existe una diferencia al crear uno u otro.

3. ¿Dónde, en su programa, hay paralelización de tareas, y dónde de datos? En el caso de mi programa la paralelización de tareas ocurre en:

```

21  pid_t thread_id = syscall(SYS_gettid);
22  printf("The thread that makes the checking rows is: %d\n", thread_id);
23  #pragma omp parallel for
24  for (int j = 0; j < 9; j++)
25  {
26      #pragma omp parallel for
27      for (int k = 0; k < 9; k++){
28          // printf("val1: %d val2: %d\n", matrix[j][i], check_array[k]);
29          if (matrix[j][i] != check_array[k])
30          {
31              check_array[k] = matrix[j][i];
32          }
33          else{
34              return (void *) false;
35          }
36      }
37  }
38  // #pragma omp parallel for
39

```

PROBLEMS (12) OUTPUT TERMINAL DEBUG CONSOLE

```

SudokuValidator.c: In function 'checkColumns':
SudokuValidator.c:18: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
18 | #pragma omp parallel for
   |
SudokuValidator.c:23: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
23 | #pragma omp parallel for
   |
SudokuValidator.c:26: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
26 | #pragma omp parallel for
   |

```

Dado que se asigna la tarea de verificar que estén todos los números del uno al nueve en la columna.

Y se realiza la paralelización de datos al momento de realizar el fork debido a que se divide y se ejecutan diferentes tareas para datos distintos.

```

163  char pid_txt[4];
164  if (fork() == 0)
165  {
166      sprintf(pid_txt, "%d", pid);
167      execvp("ps", "ps", "-p", pid_txt, "-lfl", NULL);
168  }
169  else{
170      pthread_create(&columns_checkT, NULL, &checkColumns, NULL);
171      pthread_join(columns_checkT, (void **)&pointer);
172      pid_t thread_id = syscall(SYS_gettid);
173      printf("In the thread No. %d is checking the columns \n", thread_id);
174      wait(NULL);
175      pthread_create(&rows_checkT, NULL, &checkRows, NULL);
176      pthread_join(rows_checkT, (void **)&pointer2);
177      printf("Unique columns?: %d\n", Unique_rows?: %d\n", pointer, pointer2);
178      if (subset == pointer2 && subset == pointer)
179      {
180          printf("The solution to the sudoku is valid\n");
181      }

```

PROBLEMS (12) OUTPUT TERMINAL DEBUG CONSOLE

```

SudokuValidator.c: In function 'checkColumns':
SudokuValidator.c:18: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
18 | #pragma omp parallel for
   |
SudokuValidator.c:23: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
23 | #pragma omp parallel for
   |
SudokuValidator.c:26: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
26 | #pragma omp parallel for
   |

```

4. Al agregar los #pragmas a los ciclos for, ¿cuántos LWP's hay abiertos antes de terminar el main() y cuántos durante la revisión de columnas? ¿Cuántos user threads deben haber abierto en cada caso, entonces? Hint: recuerde el modelo de multithreading que usan Linux y Windows.
Tal como se observa la cantidad de lightweight process abiertos antes de terminar el

main es de 4 y de revisión de las columnas también es de 4. Así mismo, los user threads que debieron abrir en cada caso debió de ser de 4 dado que el modelo que implementa el modelo de Linux y Windows es de 1 a 1.

```

6 2 4 5 3 9 1 8 7
5 1 9 7 2 8 6 3 4
8 3 7 6 1 4 2 9 5
1 4 3 8 6 5 7 2 9
9 5 8 2 4 7 3 6 1
7 6 2 3 9 1 4 5 8
3 7 1 9 5 6 8 4 2
4 9 6 1 8 2 5 7 3
2 8 5 4 7 3 9 1 6
The thread that makes the checking rows is: 28031
The thread that makes the checking rows is: 28032
The thread that makes the checking rows is: 28032
The thread that makes the checking rows is: 28031
The thread that makes the checking rows is: 28030
The thread that makes the checking rows is: 28030
The thread that makes the checking rows is: 28030
The thread that makes the checking rows is: 28033
The thread that makes the checking rows is: 28033
In the thread No. 28025 is checking the columns
F S UID      PID PPID  LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S dieggs+ 28025 27070 28025 99   4 80  0 - 47831 do_wai 20:24 pts/5    00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28026  0   4 80  0 - 47831 futex_ 20:24 pts/5    00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28027  0   4 80  0 - 47831 futex_ 20:24 pts/5    00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28028  0   4 80  0 - 47831 futex_ 20:24 pts/5    00:00:00 ./SudokuValidator
Unique columns? 1
Unique rows? 1
The solution to the sudoku is valid
The status of the process 28025 is:
F S UID      PID PPID  LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S dieggs+ 28025 27070 28025 25   4 80  0 - 47831 do_wai 20:24 pts/5    00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28026  0   4 80  0 - 47831 futex_ 20:24 pts/5    00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28027  0   4 80  0 - 47831 futex_ 20:24 pts/5    00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28028  0   4 80  0 - 47831 futex_ 20:24 pts/5    00:00:00 ./SudokuValidator
  
```

Se abrieron 4 threads de LWP

Solo quería destacar que estos errores me aparecían.

```

135     }
136     printf("\n");
137 }
138 }
139
140 void convert_matrix99(char* array){
141     for (int i = 0; i < 81; i++)
142     {
143         matrix[((int)i/9)][((int)i%9)] = (int)array[i]-48;
144         // printf("i:%d j:%d result:%c \n", ((int)i/9), ((int)i%9), array[i] );
145     }
146 }
147
148 int main(){
149     omp_set_num_threads(1);
150     // This pointer is to receive the returns from the kernel threads.
151     void *pointer;
152     void *pointer2;
153     pthread_t columns_checkT;
154     pthread_t rows_checkT;
155     int op = open("./sudoku", O_RDONLY );
156     struct stat sb;
  
```

y en el caso de la paralelización por for's no servía por estos warnings donde en sí el

compilador no reconocía el pragma

```

SudokuValidator.c: In function 'checkColumns':
SudokuValidator.c:18: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
18 | #pragma omp parallel for
   | ~~~~~^
SudokuValidator.c:23: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
23 | #pragma omp parallel for
   | ~~~~~^
SudokuValidator.c:26: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
26 | #pragma omp parallel for
   | ~~~~~^
SudokuValidator.c:16:26: warning: unused parameter 'arg' [-Wunused-parameter]
16 | void* checkColumns(void* arg){
   |                      ~~~~~^
SudokuValidator.c: In function 'checkRows':
SudokuValidator.c:49: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
49 | #pragma omp parallel for
   | ~~~~~^
SudokuValidator.c:55: warning: ignoring '#pragma omp parallel' [-Wunknown-pragmas]
55 | #pragma omp parallel for
   | ~~~~~^
SudokuValidator.c:47:23: warning: unused parameter 'arg' [-Wunused-parameter]
47 | void* checkRows(void* arg){
   |                  ~~~~~^
SudokuValidator.c: In function 'checkSubset':

```

5. Al limitar el número de threads en main() a uno, ¿cuántos LWP's hay abiertos durante la revisión de columnas? Compare esto con el número de LWP's abiertos antes de limitar el número de threads en main(). ¿Cuántos threads (en general) crea OpenMP por defecto?

```

3 7 1 9 5 6 8 4 2
4 9 6 1 8 2 5 7 3
2 8 5 4 7 3 9 1 6
The thread that makes the checking rows is: 29923
The thread that makes the checking rows is: 29923
The thread that makes the checking rows is: 29923
The thread that makes the checking rows is: 29923
F S UID PID PPID LWP C NLWP PRI NI ADDR SZ WCHAN STIME TTY TIME CMD
0 S diegosp 29921 27870 29921 99 5 80 0 - 25268 futex 22:43 pts/5 00:00:00 ./SudokuValidator
1 R diegosp 29921 27870 29923 99 5 80 0 - 25268 - 22:43 pts/5 00:00:00 ./SudokuValidator
1 R diegosp 29921 27870 29924 99 5 80 0 - 25268 - 22:43 pts/5 00:00:00 ./SudokuValidator
The thread that makes the checking rows is: 29923
1 R diegosp 29921 27870 29925 99 5 80 0 - 25268 - 22:43 pts/5 00:00:00 ./SudokuValidator
1 R diegosp 29921 27870 29926 0 5 80 0 - 25268 - 22:43 pts/5 00:00:00 ./SudokuValidator
The thread that makes the checking rows is: 29923
The thread that makes the checking rows is: 29923
The thread that makes the checking rows is: 29923
The thread that makes the checking rows is: 29923
In the thread No. 29921 is checking the columns
Unique columns:1
Unique rows:0
The solution to the sudoku is valid
The status of the process 29921 is:
F S UID PID PPID LWP C NLWP PRI NI ADDR SZ WCHAN STIME TTY TIME CMD
0 S diegosp 29921 27870 29921 99 1 80 0 - 41684 do_wai 22:43 pts/5 00:00:00 ./SudokuValidator

```

La cantidad de lightweights process que crea es de 5 por durante la revisión de las columnas, mientras que en el caso del número de threads al final es de solamente uno. De manera que contrasta ya que al momento de limitar la cantidad de threads se genera un modelo de multitasking similar al modelo de uno ya que crea 5 threads pero luego al pasar a la ejecución del kernel se asigna a solo uno, generando 1 único thread. Así mismo, al momento de crear los hilos crea 4 por defecto.

6. Observe cuáles LWP's están abiertos durante la revisión de columnas según ps. ¿Qué significa la primera columna de resultados de este comando? ¿Cuál es el LWP que está inactivo y por qué está inactivo? Hint: consulte las páginas del manual sobre ps.

La primera columna significa la flag del thread, en el caso de la flag que indica 1 significa que se realizó un fork pero no se realizó un exec. Así mismo, el LWP que está inactivo es aquel que dice FUTEX debajo de wchan, este indica que está bloqueado por una llamada a futex en el kernel, así mismo, se encuentra inactivo o en sleep porque es el parent id esperando a que termine de ejecutarse el execp por parte del fork.

```

7 6 2 3 9 1 4 5 8
3 7 1 9 5 6 8 4 2
4 9 6 1 8 2 5 7 3
2 8 5 4 7 3 9 1 6

The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
F S UID      PID PPID  LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY      TIME CMD
0 S dieggsp+ 30392 27070 30392 99  5 80  0 - 25268 futex_ 22:45 pts/5  00:00:00 ./SudokuValidator
1 R dieggsp+ 30392 27070 30394 99  5 80  0 - 25268 -    22:45 pts/5  00:00:00 ./SudokuValidator
1 R dieggsp+ 30392 27070 30395 99  5 80  0 - 25268 -    22:45 pts/5  00:00:00 ./SudokuValidator
1 R dieggsp+ 30392 27070 30396 99  5 80  0 - 25268 -    22:45 pts/5  00:00:00 ./SudokuValidator
1 R dieggsp+ 30392 27070 30397 99  5 80  0 - 25268 -    22:45 pts/5  00:00:00 ./SudokuValidator
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
In the thread No. 30392 is checking the columns
Unique columns?:1
Unique rows?:1
The solution to the sudoku is valid
The status of the process 30392 is:
F S UID      PID PPID  LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY      TIME CMD
0 S dieggsp+ 30392 27070 30392 99  1 80  0 - 41684 do_wai 22:45 pts/5  00:00:00 ./SudokuValidator
  
```

7. Compare los resultados de ps en la pregunta anterior con los que son desplegados por la función de revisión de columnas per se. ¿Qué es un thread team en OpenMP y cuál es el master thread en este caso? ¿Por qué parece haber un thread “corriendo”, pero que no está haciendo nada? ¿Qué significa el término busy-wait? ¿Cómo maneja OpenMP su thread pool? Resultados dados por el ps usando el `omp_set_num_threads(1)`:


```

File Edit Selection View Go Run Terminal Help
SudokuValidator.c - s_o [WSL: kali-linux] - Visual Studio Code

tasks.json C SudokuValidator.c x
labs > lab3 > C SudokuValidator.c > main()

222 print_matrix(matrix);
223 bool subset = checkSubset(matrix);
224 pid_t pid = getpid();
225 char pid_txt[4];
226 if (fork()==0)
227 {
228     sprintf(pid_txt, "%d", pid);

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
7 6 2 3 9 1 4 5 8
3 7 1 9 5 6 8 4 2
4 9 6 1 8 2 5 7 3
2 8 5 4 7 3 9 1 6
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
F S UID PID PPID LWP C NLWP PRI NI ADDR SZ WCHAN STIME TTY TIME CMD
0 S dieggs+ 30392 27070 30392 99 5 80 0 - 25268 futex_ 22:45 pts/5 00:00:00 ./SudokuValidator
1 R dieggs+ 30392 27070 30394 99 5 80 0 - 25268 - 22:45 pts/5 00:00:00 ./SudokuValidator
1 R dieggs+ 30392 27070 30395 99 5 80 0 - 25268 - 22:45 pts/5 00:00:00 ./SudokuValidator
1 R dieggs+ 30392 27070 30396 99 5 80 0 - 25268 - 22:45 pts/5 00:00:00 ./SudokuValidator
1 R dieggs+ 30392 27070 30397 99 5 80 0 - 25268 - 22:45 pts/5 00:00:00 ./SudokuValidator
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
The thread that makes the checking rows is: 30394
In the thread No. 30392 is checking the columns
Unique columns?:1
Unique rows?:1
The solution to the sudoku is valid
The status of the process 30392 is:
F S UID PID PPID LWP C NLWP PRI NI ADDR SZ WCHAN STIME TTY TIME CMD
0 S dieggs+ 30392 27070 30392 99 1 80 0 - 41684 do_wai 22:45 pts/5 00:00:00 ./SudokuValidator

WSL: kali-linux 0 9 Select folder. Compile & Run Compile Debug Ln 210, Col 26 Spaces: 4 UTF-8 LF C Linux 11:06 PM 3/13/2023

```

Resultados dados con la paralelización:

```

File Edit Selection View Go Run Terminal Help
SudokuValidator.c - s_o [WSL: kali-linux] - Visual Studio Code

tasks.json C SudokuValidator.c x
labs > lab3 > C SudokuValidator.c > main()

222 print_matrix(matrix);
223 bool subset = checkSubset(matrix);
224 pid_t pid = getpid();
225 char pid_txt[4];
226 if (fork()==0)
227 {
228     sprintf(pid_txt, "%d", pid);

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
6 2 4 5 3 9 1 8 7
5 1 9 7 2 8 6 3 4
0 3 7 6 1 4 2 9 5
1 4 3 8 6 5 7 2 9
9 5 8 2 4 7 3 6 1
7 6 2 3 9 1 4 5 8
3 7 1 9 5 6 8 4 2
4 9 6 1 8 2 5 7 3
2 8 5 4 7 3 9 1 6
The thread that makes the checking rows is: 28031
The thread that makes the checking rows is: 28032
The thread that makes the checking rows is: 28032
The thread that makes the checking rows is: 28031
The thread that makes the checking rows is: 28030
The thread that makes the checking rows is: 28030
The thread that makes the checking rows is: 28033
The thread that makes the checking rows is: 28033
In the thread No. 28025 is checking the columns
F S UID PID PPID LWP C NLWP PRI NI ADDR SZ WCHAN STIME TTY TIME CMD
0 S dieggs+ 28025 27070 28025 99 4 80 0 - 47831 do_wai 20:24 pts/5 00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28026 0 4 80 0 - 47831 futex_ 20:24 pts/5 00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28027 0 4 80 0 - 47831 futex_ 20:24 pts/5 00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28028 0 4 80 0 - 47831 futex_ 20:24 pts/5 00:00:00 ./SudokuValidator
Unique columns?:1
Unique rows?:1
The solution to the sudoku is valid
The status of the process 28025 is:
F S UID PID PPID LWP C NLWP PRI NI ADDR SZ WCHAN STIME TTY TIME CMD
0 S dieggs+ 28025 27070 28025 25 4 80 0 - 47831 do_wai 20:24 pts/5 00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28026 0 4 80 0 - 47831 futex_ 20:24 pts/5 00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28027 0 4 80 0 - 47831 futex_ 20:24 pts/5 00:00:00 ./SudokuValidator
1 S dieggs+ 28025 27070 28028 0 4 80 0 - 47831 futex_ 20:24 pts/5 00:00:00 ./SudokuValidator

WSL: kali-linux 0 9 Select folder. Compile & Run Compile Debug Ln 207, Col 28 Spaces: 4 UTF-8 LF C Linux 8:30 PM 3/13/2023

```

Resultados dados per se sin la paralelización:

```

135     }
136     printf("\n");
137 }
138 }
139
140 void convert_matrix99(char* array){
141     for (int i = 0; i < 81; i++)

```

```

7 6 2 3 9 1 4 5 8
3 7 1 9 5 6 8 4 2
4 9 6 1 8 2 5 7 3
2 8 5 4 7 3 9 1 6
The thread that makes the checking rows is: 7172
The thread that makes the checking rows is: 7172
The thread that makes the checking rows is: 7172
The thread that makes the checking rows is: 7172
The thread that makes the checking rows is: 7172
The thread that makes the checking rows is: 7172
The thread that makes the checking rows is: 7172
The thread that makes the checking rows is: 7172
In the thread No. 7155 is checking the columns
F S UID      PID PPID  LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY      TIME CMD
0 S dieggspp 7155 7145 7155 0 1 80  0 - 2665 do_wai 23:58 ?      00:00:00 /home/dieggsppu/s_o/labs/lab3/output/S
Unique columns?:1
Unique rows?:1
The solution to the sudoku is valid
The status of the process 7155 is:
F S UID      PID PPID  LWP  C NLWP PRI  NI ADDR SZ WCHAN  STIME TTY      TIME CMD
0 S dieggspp 7155 7145 7155 0 1 80  0 - 2665 do_wai 23:58 ?      00:00:00 /home/dieggsppu/s_o/labs/lab3/output/S
[1] + Done          "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0x"/tmp/Microsoft-MIEngine-In-ygeh5v5e.0nu" 1>
"/tmp/Microsoft-MIEngine-Out-z33t5sdj.kmx"

```

Se debe de notar que al momento de utilizar el per se se realiza un wait para esperar una cola, así mismo, en el caso de utilizar el paralelizado hay un thread que realiza la espera (que sería el parent) y los demás realizan una llamada a futex, y en el caso del thread único solamente 1 tiene un futex. Un thread team es cuando un thread entra a una región paralelizable de manera que el thread que entra se forkea y se genera un thread team, es similar a un fork-join de threading implícito. En el caso del thread que está corriendo, sin embargo, no está haciendo nada es debido a que es el main thread que está esperando a los otros. Busy wait es es una técnica de sincronización de procesos en la que un proceso espera y constantemente chequea por una condición para proseguir con su ejecución. Open MP maneja su thread pool creando un thread por cada core que está en la computadora, de esta forma mantiene el modelo uno a uno de multithreading.

8. Luego de agregar por primera vez la cláusula schedule(dynamic) y ejecutar su programa repetidas veces, ¿cuál es el máximo número de threads trabajando según la función de revisión de columnas? Al comparar este número con la cantidad de LWP's que se creaban antes de agregar schedule(), ¿qué deduce sobre la distribución de trabajo que OpenMP hace por defecto?

```

7 6 2 3 9 1 4 5 8
3 7 1 9 5 6 8 4 2
4 9 6 1 8 2 5 7 3
2 8 5 4 7 3 9 1 6
5 1 9 7 2 8 6 3 4
3 7 6 1 4 2 9 5
The thread that makes the checking rows is: 2332
The thread that makes the checking rows is: 2332
The thread that makes the checking rows is: 2332
The thread that makes the checking rows is: 2332
The thread that makes the checking rows is: 2332
The thread that makes the checking rows is: 2332
The thread that makes the checking rows is: 2332
The thread that makes the checking rows is: 2332
In the thread No. 2321 is checking the columns
F S UID      PID PPID  LWP  C NLMP PRI NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S dieggsp+ 2321 27870 2321 99  8  80  0 - 47799 futex_ 23:28 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 2321 27870 2322  0  8  80  0 - 47799 futex_ 23:28 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 2321 27870 2323  0  8  80  0 - 47799 futex_ 23:28 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 2321 27870 2324  0  8  80  0 - 47799 futex_ 23:28 pts/5    00:00:00 ./SudokuValidator
1 R dieggsp+ 2321 27870 2332  0  8  80  0 - 47799 futex_ 23:28 pts/5    00:00:00 ./SudokuValidator
1 R dieggsp+ 2321 27870 2333  0  8  80  0 - 47799 -      23:28 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 2321 27870 2334  0  7  80  0 - 47838 futex_ 23:28 pts/5    00:00:00 ./SudokuValidator
1 R dieggsp+ 2321 27870 2335  0  5  80  0 - 47831 ?        23:28 pts/5    00:00:00 ./SudokuValidator
Unique columns?:1
Unique rows?:1
The solution to the sudoku is valid
The status of the process 2321 is:
F S UID      PID PPID  LWP  C NLMP PRI NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S dieggsp+ 2321 27870 2321 50  4  80  0 - 47831 do_wai 23:28 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 2321 27870 2322  0  4  80  0 - 47831 futex_ 23:28 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 2321 27870 2323  0  4  80  0 - 47831 futex_ 23:28 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 2321 27870 2324  0  4  80  0 - 47831 futex_ 23:28 pts/5    00:00:00 ./SudokuValidator

```

El número máximo de threads según la revisión de columnas es 8, así mismo, al comparar este número con la cantidad de antes de que se agregara el schedule puedo suponer que se optimizan los recursos de manera que puede optimizarse de mejor manera ya que cuando se acabe de utilizar el thread para una iteración en el for, el mismo thread se puede asignar a la siguiente iteración realizando 2 veces su trabajo.

9. Luego de agregar las llamadas `omp_set_num_threads()` a cada función donde se usa OpenMP y probar su programa, antes de agregar `omp_set_nested(true)`, ¿hay más o menos concurrencia en su programa? ¿Es esto sinónimo de un mejor desempeño? Explique

```

17 bool columns = true;
18 bool rows = true;
19 void* checkColumns1(void* arg){
20     int check_array[9] = {};
21     omp_set_num_threads(9);
22     #pragma omp parallel for
23     // #pragma omp parallel for schedule(dynamic)
The thread that makes the checking rows is: 4645
The thread that makes the checking rows is: 4638
In the thread No. 4628 is checking the columns
F S UID      PID PPID  LWP  C NLMP PRI NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S dieggsp+ 4628 27870 4628 99  9  80  0 - 58076 do_wai 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4629  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4630  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4631  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4632  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4633  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4634  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4635  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4636  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
Unique columns?:1
Unique rows?:1
The solution to the sudoku is valid
The status of the process 4628 is:
F S UID      PID PPID  LWP  C NLMP PRI NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S dieggsp+ 4628 27870 4628 25  9  80  0 - 58076 do_wai 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4629  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4630  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4631  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4632  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4633  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4634  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator
1 S dieggsp+ 4628 27870 4635  0  9  80  0 - 58076 futex_ 23:44 pts/5    00:00:00 ./SudokuValidator

```

Hay más concurrencia debido a que lo que hace la función es setear la cantidad de threads que se crearán, de manera que en este caso desde un principio se crea la

cantidad de threads que se utilizarán para cada paralelización, es por ello que en este caso son 9. Sin embargo, esto no es sinónimo de un mejor desempeño ya que cómo se puede observar en la columna "C" la utilización del cpu es de 0 exceptuando en 1 sola columna. Por ende es bastante deficiente ya que puede ser que algunos threads utilicen más o menos recursos que otros y entonces muchos threads no se les asigne toda su utilización.

10. ¿Cuál es el efecto de agregar `omp_set_nested(true)`? Explique
Lo que permite que se pueda realizar la paralelización anidada, es decir cuando un thread entra en una región paralela y esta región paralela pueda crear otras regiones paralelas, es paralelizar lo paralelizado.