2º curso / 2º cuatr.

Grado Ing. Inform.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 3. Programación paralela III: Interacción con el entorno en OpenMP

Félix Ramírez García
Grupo A3 con profesor Christian Morillas
Fecha de entrega: 19-05-2019
Fecha evaluación en clase: 20-05-2019

---

1. Usar la cláusula `num_threads(x)` en el ejemplo del seminario `if_clause.c`, y añadir un parámetro de entrada al programa que fije el valor `x` que se va a usar en la cláusula. Incorporar en el cuaderno de trabajo de esta práctica volcados de pantalla con ejemplos de ejecución que ilustren la funcionalidad de esta cláusula y explicar por qué lo ilustran.

La siguiente imagen muestra el código de if-clauseModificado.c

if-clauseModificado.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(int argc, char **argv){
    int i, n=20, tid, x;
    int a[n], suma=0, sumalocal;
    if(argc<2){
        fprintf(stderr, "[ERROR]-Faltan iteraciones\n");
        exit(-1);
    }

    if(argc<3){
        fprintf(stderr, "[ERROR]-Falta numero de threads\n");
        exit(-1);
    }

    n=atoi(argv[1]);
    x=atoi(argv[2]);

    if(n>20) n=20;
    for(i=0;i<n;i++){
        a[i]=i;
    }

    #pragma omp parallel if(n>4) num_threads(x) default(none) private(sumalocal,tid) shared(a, suma, n)
    {
        sumalocal=0;
        tid=omp_get_thread_num();
        #pragma omp for private(i) schedule(static) nowait
        for(i=0;i<n;i++){
            sumalocal += a[i];
            printf("thread %d suma de a[%d]=%d sumalocal=%d\n", tid, i, a[i], sumalocal);
        }
        #pragma omp atomic
            suma+=sumalocal;
        #pragma omp barrier
        #pragma omp master
            printf("thread master=%d imprime suma=%d\n", tid, suma);
    }
}

```

Y la siguiente imagen muestra la compilación y las ejecuciones de if-clause y de if-clauseModificado:

```
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer1] 2019-05-18 Saturday
$gcc -O2 -fopenmp -o if-clause if-clause.c
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer1] 2019-05-18 Saturday
$gcc -O2 -fopenmp -o if-clauseModificado if-clauseModificado.c
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer1] 2019-05-18 Saturday
$./if-clause 5
thread 0 suma de a[0]=0 sumalocal=0
thread 0 suma de a[1]=1 sumalocal=1
thread 3 suma de a[4]=4 sumalocal=4
thread 1 suma de a[2]=2 sumalocal=2
thread 2 suma de a[3]=3 sumalocal=3
thread master0 imprime suma=10
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer1] 2019-05-18 Saturday
$./if-clauseModificado 5 4
thread 0 suma de a[0]=0 sumalocal=0
thread 0 suma de a[1]=1 sumalocal=1
thread 3 suma de a[4]=4 sumalocal=4
thread 1 suma de a[2]=2 sumalocal=2
thread 2 suma de a[3]=3 sumalocal=3
thread master=0 imprime suma=10
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer1] 2019-05-18 Saturday
$./if-clause 6
thread 0 suma de a[0]=0 sumalocal=0
thread 0 suma de a[1]=1 sumalocal=1
thread 3 suma de a[5]=5 sumalocal=5
thread 2 suma de a[4]=4 sumalocal=4
thread 1 suma de a[2]=2 sumalocal=2
thread 1 suma de a[3]=3 sumalocal=5
thread master0 imprime suma=15
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer1] 2019-05-18 Saturday
$./if-clauseModificado 6 6
thread 5 suma de a[5]=5 sumalocal=5
thread 0 suma de a[0]=0 sumalocal=0
thread 3 suma de a[3]=3 sumalocal=3
thread 2 suma de a[2]=2 sumalocal=2
thread 4 suma de a[4]=4 sumalocal=4
thread 1 suma de a[1]=1 sumalocal=1
thread master=0 imprime suma=15
```

El código solo se puede paralelizar si el numero de iteraciones de superior a 4. Al poner num_threads() le podemos fijar el número de hebras sin tener que recompilar. Aunque solo se fijara si el numero de iteraciones es mayor que 4.

2. (a) Rellenar la Tabla 1 (se debe poner en la tabla el id del *thread* que ejecuta cada iteración) ejecutando los ejemplos del seminario `schedule-clause.c`, `scheduled-clause.c` y `scheduleg-clause.c` con dos *threads* (0,1) y unas entradas de:

- iteraciones: 16 (0,…15)
- chunck= 1, 2 y 4

Tabla 1 . Tabla schedule. En la segunda fila, 1, 2 4 representan el tamaño del chunk (consulte seminario)

Iteración	schedule-clause.c			schedule-claused.c			schedule-clauseg.c		
	1	2	4	1	2	4	1	2	4
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0	0
6	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	0
8	1	1	1	0	1	0	0	0	0
9	1	1	1	0	1	0	0	0	0
10	1	1	1	0	1	0	0	0	0
11	1	1	1	0	1	0	0	0	0
12	1	1	1	0	0	1	1	1	1
13	1	1	1	0	0	1	1	1	1
14	1	1	1	0	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1

(b) Rellenar otra tabla como la de la figura pero esta vez usando cuatro *threads* (0,1,2,3).

Tabla 2 . Tabla schedule. En la segunda fila, 1, 2 4 representan el tamaño del chunk (consulte seminario)

Iteración	schedule-clause.c			schedule-claused.c			schedule-clauseg.c		
	1	2	4	1	2	4	1	2	4
0	3	2	0	1	0	0	0	0	0
1	3	2	0	1	0	0	0	0	0
2	3	2	0	1	0	0	0	0	0
3	3	1	0	1	0	0	0	0	2
4	1	0	3	1	0	3	0	0	0
5	1	0	3	3	0	3	0	0	2
6	1	0	3	3	0	3	0	0	2
7	1	0	3	3	0	3	0	0	2
8	2	3	1	3	0	1	0	3	1
9	2	3	1	3	0	1	3	3	1
10	2	3	1	3	1	1	2	3	3
11	2	3	1	3	1	1	2	1	3
12	0	1	2	3	3	2	2	1	3
13	0	1	2	2	3	2	1	2	3
14	0	1	2	1	2	2	1	2	1
15	0	1	2	0	2	2	1	2	1

Escriba en el cuaderno de prácticas las diferencias en el comportamiento de `schedule()` con `static`, `dynamic` y `guided`.

Con static las tareas se reparten equitativamente usando round-robin, con Dynamic y Guided se repartirán las tareas aleatoriamente (pero su tamaño vendrá definido por el chunk).

3. Añadir al programa `scheduled-clause.c` lo necesario para que imprima el valor de las variables de control `dyn-var`, `nthreads-var`, `thread-limit-var` y `run-sched-var` dentro (debe imprimir sólo un thread) y fuera de la región paralela. Realizar varias ejecuciones usando variables de entorno para modificar estas variables de control antes de la ejecución. Incorporar en su cuaderno de prácticas volcados de pantalla de estas ejecuciones. ¿Se imprimen valores distintos dentro y fuera de la región paralela?

El código fuente de `scheduled-clauseModificado.c` se muestra en la siguiente imagen:

scheduled-clauseModificado.c

```c
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
   #include <omp.h>
#else
   #define omp_get_thread_num() 0
   #define omp_get_num_threads() 1
   #define omp_set_num_threads(int)
#endif

int main(int argc, char **argv){
   int i, n=200, chunk, a[n], suma=0;
   omp_sched_t scheduleType;
   int chunkValue;

   if(argc<3){
      fprintf(stderr, "\nFalta iteraciones o chunk\n");
      exit(-1);
   }

   n=atoi(argv[1]);
   chunk=atoi(argv[2]);

   if(n>200) n=200;

   for(i=0;i<n;i++) a[i]=i;

   #pragma omp parallel for firstprivate(suma) lastprivate(suma) schedule(dynamic, chunk)
   for(i=0;i<n;i++){
      suma=suma+a[i];
      printf("thread %d suma a[%d]=%d suma=%d\n", omp_get_thread_num(), i, a[i], suma);

      if(omp_get_thread_num()==0){
         printf("Dentro de parallel for: \n");

         omp_get_schedule(&scheduleType, &chunkValue);

         printf("dyn-var: %d, nthreads-var:%d, thread-limit-var:%d,run-sched-var: %d, chunk: %d\n", omp_get_dynamic(),omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);
      }
   }

   printf("Fuera de parallel for suma=%d\n", suma);

   omp_get_schedule(&scheduleType, &chunkValue);

   printf("dyn-var: %d, nthreads-var:%d, thread-limit-var:%d,run-sched-var: %d, chunk: %d\n", omp_get_dynamic(),omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);

   return 0;
}
```

La siguiente imagen muestra la compilacion y ejecución de scheduled-clauseModificado cambiando OMP_DYNAMIC :

```
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarcia
Felix/ejer3] 2019-05-18 Saturday
$gcc -O2 -fopenmp -o scheduled-clauseModificado scheduled-clauseModificado.c
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarcia
Felix/ejer3] 2019-05-18 Saturday
$export OMP_DYNAMIC=false && ./scheduled-clauseModificado 5 2
thread 1 suma a[4]=4 suma=4
thread 3 suma a[0]=0 suma=0
thread 3 suma a[1]=1 suma=1
thread 0 suma a[2]=2 suma=2
Dentro de parallel for:
dyn-var: 0, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1thread 0 suma a[3]=3 suma=5
Dentro de parallel for:
dyn-var: 0, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1Fuera de parallel for suma=4
dyn-var: 0, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1[FelixRamirezGarcia felix@DESKTOP-8PO8
DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer3] 2019-05-18 Saturday
$export OMP_DYNAMIC=true && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
thread 3 suma a[2]=2 suma=2
thread 3 suma a[3]=3 suma=5
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1thread 1 suma a[4]=4 suma=4
Fuera de parallel for suma=4
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1[FelixRamirezGarcia felix@DESKTOP-8PO8
```

La siguiente imagen muestra la compilación y ejecución de scheduled-clauseModificado cambiando OMP_NUM_THREADS :

```
DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer3] 2019-05-18 Saturday
$export OMP_NUM_THREADS=4 && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
thread 3 suma a[2]=2 suma=2
thread 3 suma a[3]=3 suma=5
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1thread 2 suma a[4]=4 suma=4
Fuera de parallel for suma=4
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1[FelixRamirezGarcia felix@DESKTOP-8PO8
DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer3] 2019-05-18 Saturday
$export OMP_NUM_THREADS=2 && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 1 suma a[4]=4 suma=9
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1Fuera de parallel for suma=9
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1[FelixRamirezGarcia felix@DESKTOP-8PO8
DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer3] 2019-05-18 Saturday
```

La siguiente imagen muestra la compilación y ejecución de scheduled-clauseModificado cambiando OMP_SCHEDULE :

```
DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer3] 2019-05-18 Saturday
$export OMP_SCHEDULE="static" && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 1 suma a[4]=4 suma=9
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 1, chunk: 0thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 1, chunk: 0Fuera de parallel for suma=9
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 1, chunk: 0[FelixRamirezGarcia felix@DESKTOP-8PO8
DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer3] 2019-05-18 Saturday
$export OMP_SCHEDULE="dynamic" && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1thread 0 suma a[4]=4 suma=5
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
Fuera de parallel for suma=5
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1[FelixRamirezGarcia felix@DESKTOP-8PO8
DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer3] 2019-05-18 Saturday
```

La siguiente imagen muestra la compilación y ejecución de scheduled-clauseModificado cambiando OMP_THREAD_LIMIT :

```
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarcia
Felix/ejer3] 2019-05-18 Saturday
$export OMP_THREAD_LIMIT=3 && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:3,run-sched-var: 2, chunk: 1
thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:3,run-sched-var: 2, chunk: 1
thread 0 suma a[4]=4 suma=5
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:3,run-sched-var: 2, chunk: 1
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
Fuera de parallel for suma=5
dyn-var: 1, nthreads-var:2, thread-limit-var:3,run-sched-var: 2, chunk: 1
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarcia
Felix/ejer3] 2019-05-18 Saturday
$export OMP_THREAD_LIMIT=4 && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:4,run-sched-var: 2, chunk: 1
thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:4,run-sched-var: 2, chunk: 1
thread 0 suma a[4]=4 suma=5
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:4,run-sched-var: 2, chunk: 1
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
Fuera de parallel for suma=5
dyn-var: 1, nthreads-var:2, thread-limit-var:4,run-sched-var: 2, chunk: 1
```

Como podemos ver, dentro de la región parallel y fuera obtenemos los mismos resultados.

4. Usar en el ejemplo anterior las funciones `omp_get_num_threads()`, `omp_get_num_procs()` y `omp_in_parallel()` dentro y fuera de la región paralela. Imprimir los valores que obtienen estas funciones dentro (lo debe imprimir sólo uno de los threads) y fuera de la región paralela. Incorporar en su cuaderno de prácticas volcados de pantalla con los resultados de ejecución obtenidos. Indicar en qué funciones se obtienen valores distintos dentro y fuera de la región paralela.

El código de `scheduled-clauseModificado4.c` se muestra en la siguiente imagen:

scheduled-clauseModificado4.c

```c
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
    #include <omp.h>
#else
    #define omp_get_thread_num() 0
    #define omp_get_num_threads() 1
    #define omp_set_num_threads(int)
    #define omp_in_parallel() 0
#endif

int main(int argc, char **argv){
    int i, n=200, chunk, a[n], suma=0;
    omp_sched_t scheduleType;
    int chunkValue;

    if(argc<3){
        fprintf(stderr, "\nFalta iteraciones o chunk\n");
        exit(-1);
    }

    n=atoi(argv[1]);
    chunk=atoi(argv[2]);

    if(n>200) n=200;

    for(i=0;i<n;i++) a[i]=i;

    #pragma omp parallel for firstprivate(suma) lastprivate(suma) schedule(dynamic, chunk)
    for(i=0;i<n;i++){
        suma=suma+a[i];
        printf("thread %d suma a[%d]=%d suma=%d\n", omp_get_thread_num(), i, a[i], suma);

        if(omp_get_thread_num()==0){
            printf("Dentro de parallel for: \n");

            omp_get_schedule(&scheduleType, &chunkValue);

            printf("dyn-var: %d, nthreads-var:%d, thread-limit-var:%d,run-sched-var: %d, chunk: %d\n", omp_get_dynamic(),omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);

            printf("omp_get_num_threads: %d, omp_get_num_procs: %d, omp_in_parallel(): %d\n",omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
        }
    }

    printf("Fuera de parallel for suma=%d\n", suma);

    omp_get_schedule(&scheduleType, &chunkValue);

    printf("dyn-var: %d, nthreads-var:%d, thread-limit-var:%d,run-sched-var: %d, chunk: %d\n", omp_get_dynamic(),omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);

    printf("omp_get_num_threads: %d, omp_get_num_procs: %d, omp_in_parallel(): %d\n",omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());

    return 0;
}
```

La siguiente imagen muestra la compilación y ejecución:

```
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarcia
Felix/ejer4] 2019-05-18 Saturday
$gcc -O2 -fopenmp -o scheduled-clauseModificado4 scheduled-clauseModificado4.c
ç[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarci
aFelix/ejer4] 2019-05-18 Saturday
$./scheduled-clauseModificado4 5 2
thread 0 suma a[0]=0 suma=0
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 1 suma a[4]=4 suma=9
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:4,run-sched-var: 2, chunk: 1
omp_get_num_threads: 2, omp_get_num_procs: 4, omp_in_parallel(): 1
thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:4,run-sched-var: 2, chunk: 1
omp_get_num_threads: 2, omp_get_num_procs: 4, omp_in_parallel(): 1
Fuera de parallel for suma=9
dyn-var: 1, nthreads-var:2, thread-limit-var:4,run-sched-var: 2, chunk: 1
omp_get_num_threads: 1, omp_get_num_procs: 4, omp_in_parallel(): 0
```

Dentro y fuera cambian los valores de omp_in_parallel() y omp_get_num_threads().

5. Añadir al programa `scheduled-clause.c` lo necesario para modificar las variables de control `dyn-var`, `nthreads-var` y `run-sched-var` y para poder imprimir el valor de estas variables antes y después de dicha modificación. Incorporar en su cuaderno de prácticas volcados de pantalla con los resultados de ejecución obtenidos.

La siguiente imagen muestra el código de `scheduled-clauseModificado5.c`

```c
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
    #include <omp.h>
#else
    #define omp_get_thread_num() 0
#endif

int main(int argc, char **argv){
    int i, n=200, chunk, a[n], suma=0, chunkValue;
    omp_sched_t scheduleType;

    if(argc<3){
        fprintf(stderr, "\nFalta iteraciones o chunk\n");
        exit(-1);
    }

    n=atoi(argv[1]);
    chunk=atoi(argv[2]);

    if(n>200) n=200;


    for(i=0;i<n;i++) a[i]=i;

    printf("Antes de hacer el cambio\n");
    omp_get_schedule(&scheduleType, &chunkValue);
    printf("dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d, chunk: %d\n ", omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);


    printf("Cambiando valores a:\n\tdyn-var: 1\n\tnthreads-var: 3\n\trun-sched-var: 2\n\tchunk: 2\n");
    omp_set_dynamic(1);
    omp_set_num_threads(3);
    omp_set_schedule(2, 2);

    #pragma omp parallel for firstprivate(suma) lastprivate(suma) schedule(dynamic, chunk)
    for(i=0;i<n;i++){
        suma=suma+a[i];
        printf("thread %d suma a[%d]=%d suma=%d\n", omp_get_thread_num(), i, a[i], suma);
    }

    printf("Fuera de parallel for suma=%d\n", suma);

    printf("Despues de hacer el cambio\n");
    omp_get_schedule(&scheduleType, &chunkValue);
    printf("dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d, chunk: %d\n ", omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);

    return 0;
}
```

La siguiente imagen muestra la compilación y ejecución :

```
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarcia
Felix/ejer5] 2019-05-18 Saturday
$gcc -O2 -fopenmp -o scheduled-clauseModificado5 scheduled-clauseModificado5.c
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/AC/Practicas/Entrega/bp3_RamirezGarcia
Felix/ejer5] 2019-05-18 Saturday
$./scheduled-clauseModificado5 2 3
Antes de hacer el cambio
dyn-var: 1, nthreads-var: 2, thread-limit-var: 4, run-sched-var: 2, chunk: 1
 Cambiando valores a:
        dyn-var: 1
        nthreads-var: 3
        run-sched-var: 2
        chunk: 2
thread 2 suma a[0]=0 suma=0
thread 2 suma a[1]=1 suma=1
Fuera de parallel for suma=1
Despues de hacer el cambio
dyn-var: 1, nthreads-var: 3, thread-limit-var: 4, run-sched-var: 2, chunk: 2
```

Como se puede apreciar los cambios se realizan correctamente.

6. Implementar un programa secuencial en C que multiplique una matriz triangular por un vector (use variables dinámicas). Compare el orden de complejidad del código que ha implementado con el código que implementó para el producto matriz por vector.

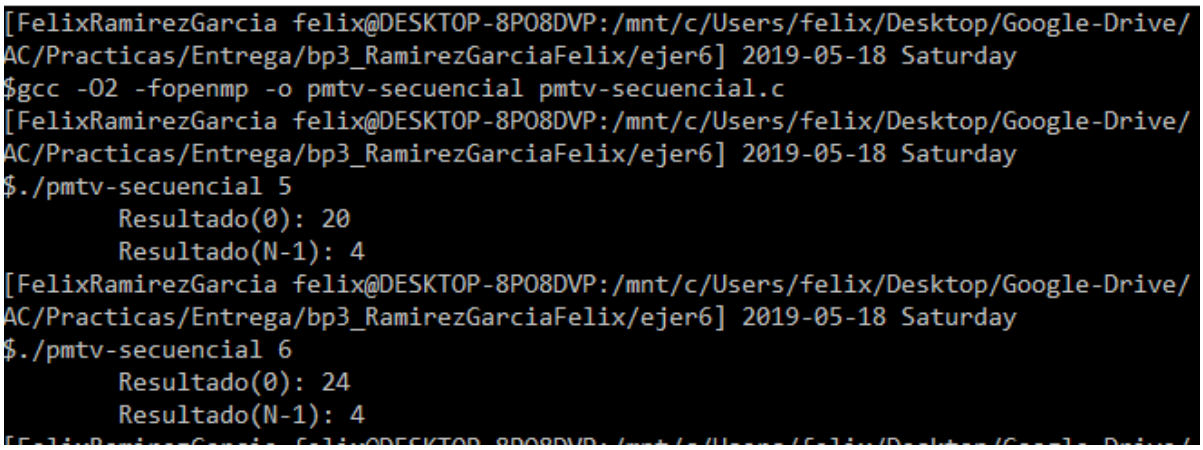
NOTAS: (1) el número de filas/columnas debe ser un argumento de entrada; (2) se debe inicializar las matrices antes del cálculo; (3) se debe imprimir siempre la primera y última componente del resultado antes de que termine el programa.

El código de `pmtv-secuencial.c` es el siguiente:

pmtv-secuencial.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv){
    int i, j, N;
    int *v, *resultado;
    int **m;

    if(argc<2){
        fprintf(stderr, "Falta el tamaño\n");
        exit(-1);
    }

    N=atoi(argv[1]);

    v=(int*)malloc(N*sizeof(int));
    resultado=(int*)malloc(N*sizeof(int));
    m=(int**)malloc(N*sizeof(int*));

    for(i=0;i<N;i++) m[i]=(int*)malloc(N*sizeof(int));

    for(i=0;i<N;i++){
        for(j=i;j<N;j++) m[i][j]=2;
        v[i]=2;
        resultado[i]=0;
    }

    for(i=0;i<N;i++){
        for(j=i;j<N;j++) resultado[i]+=m[i][j]*v[j];
    }

  printf("\tResultado(0): %d\n\tResultado(N-1): %d\n", resultado[0], resultado[N-1]);

  for (i=0; i<N; i++) free(m[i]);
  free(m);
  free(v);
  free(resultado);
```

La siguiente imagen muestra la compilación y ejecución :



7. Implementar en paralelo la multiplicación de una matriz triangular por un vector a partir del código secuencial realizado para el ejercicio anterior utilizando la directiva `for` de OpenMP. El código debe repartir entre los threads las iteraciones del bucle que recorre las filas. Dibujar en el cuaderno de prácticas la descomposición de dominio utilizada (Lección 4/Tema 2) en el código paralelo implementado para asignar tareas a los threads (Lección 5/Tema 2). Añadir lo necesario para que el usuario pueda fijar la planificación de tareas usando la variable de entorno `OMP_SCHEDULE`. Obtener en atcgrid los tiempos de ejecución del código paralelo (usando, como siempre, -O2 al compilar) que multiplica una matriz triangular por un vector con las alternativas de planificación `static`, `dynamic` y `guided` para `chunk` de 1, 64 y el `chunk` por defecto para la alternativa. Use un tamaño de vector N múltiplo del número de cores y de 64 que no sea inferior a 15360. El número de threads en las ejecuciones debe coincidir con el número de cores. Rellenar la Tabla 3 dos veces con los tiempos obtenidos. Representar el tiempo para `static`, `dynamic` y `guided` en función del tamaño del `chunk` en una gráfica. ¿Qué alternativa ofrece mejores prestaciones? Razone por qué. Incluya los scripts utilizado en el cuaderno de prácticas. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

Conteste a las siguientes preguntas: (a) ¿Qué valor por defecto usa OpenMP para `chunk` con `static`, `dynamic` y `guided`? Indique qué ha hecho para obtener este valor por defecto para cada alternativa. (b) ¿Qué número de operaciones de multiplicación y suma realizan cada uno de los threads en la asignación `static` para cada uno de los `chunks`? (c) Con la asignación `dynamic` y `guided`, ¿qué cree que debe ocurrir con el número de operaciones de multiplicación y suma que realizan cada uno de los threads?

Todas tienen unos tiempos de respuesta parecidos. Aunque el que es ligeramente mejor es static, ya que no tiene que hacer un reparto de tareas tan elaborado.

a) Static no tiene y para dynamic y guided su valor es 0.
b) Realizarán chunk * numero_de_fila

El codigo fuente de pmtv-OpenMP.c es el que se muestra a continuacion:

pmtv-OpenMP.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(int argc, char **argv){
    int i, j, N;
    int *v, *r;
    int **m;
    double inicio, fin;

    if(argc<2){
        fprintf(stderr, "Falta tamaño\n");
        exit(-1);
    }

    N=atoi(argv[1]);

    r=(int*)malloc(N*sizeof(int));
    m=(int**)malloc(N*sizeof(int*));
    v=(int*)malloc(N*sizeof(int));

    for(i=0;i<N;i++) m[i]=(int*)malloc(N*sizeof(int));

    for(i=0;i<N;i++){
        for(j=i;j<N;j++) m[i][j]=2;
        v[i]=2;
        r[i]=0;
    }

    inicio=omp_get_wtime();
    #pragma omp parallel for private(j) schedule(runtime)
    for(i=0;i<N;i++){
        for(j=i;j<N;j++) r[i]+=m[i][j]*v[j];
    }
    fin=omp_get_wtime();

    printf("Tiempo= %11.9f\tPrimera=%d\tUltima=%d\n", fin-inicio, r[0], r[N-1]);

    for(i=0;i<N;i++) free(m[i]);
    free(m);
    free(v);
    free(r);
}
```

DESCOMPOSICIÓN DE DOMINIO:

Cada hebra trabaja con una parte de los datos asignados (las filas de la matriz), y cada hebra conoce la fila con la que trabaja.

La siguiente imagen muestra la compilación y ejecución en local:

```
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/
AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer7] 2019-05-18 Saturday
$gcc -O2 -fopenmp -o pmtv-OpenMP pmtv-OpenMP.c
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/
AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer7] 2019-05-18 Saturday
$./pmtv-OpenMP 100
Tiempo= 0.000276000     Primera=400     Ultima=4
```

La siguiente imagen muestra la ejecución del script en local:

```
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Use
Felix/ejer7] 2019-05-18 Saturday
$./pmtv-OpenMP_PCaula
schedule(static) chunk(defecto)
Tiempo= 0.094122000     Primera=61440   Ultima=4
schedule(static) chunk(1)
Tiempo= 0.066223000     Primera=61440   Ultima=4
schedule(static) chunk(64)
Tiempo= 0.066384000     Primera=61440   Ultima=4
schedule(dynamic) chunk(defecto)
Tiempo= 0.093506000     Primera=61440   Ultima=4
schedule(dynamic) chunk(1)
Tiempo= 0.068243000     Primera=61440   Ultima=4
schedule(dynamic) chunk(64)
Tiempo= 0.068390000     Primera=61440   Ultima=4
schedule(guided) chunk(defecto)
Tiempo= 0.122639000     Primera=61440   Ultima=4
schedule(guided) chunk(1)
Tiempo= 0.152661000     Primera=61440   Ultima=4
schedule(guided) chunk(64)
Tiempo= 0.091013000     Primera=61440   Ultima=4
```

La siguiente imagen muestra el código del script pmvt-OpenMP_atcgrid.sh :

```
#!/bin/bash

export OMP_SCHEDULE="static"
echo "schedule(static) chunk(defecto)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="static, 1"
echo "schedule(static) chunk(1)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="static, 64"
echo "schedule(static) chunk(64)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic"
echo "schedule(dynamic) chunk(defecto)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic, 1"
echo "schedule(dynamic) chunk(1)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic, 64"
echo "schedule(dynamic) chunk(64)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="guided"
echo "schedule(guided) chunk(defecto)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="guided, 1"
echo "schedule(guided) chunk(1)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="guided, 64"
echo "schedule(guided) chunk(64)"
./pmtv-OpenMP 15360
```

La siguiente imagen muestra la ejecución del script pmvt-OpenMP_atcgrid.sh en atcgrid:

```
[FelixRamirezGarcia A3estudiante20@atcgrid:~] 2019-05-18 Saturday
$echo './pmtv-OpenMP_atcgrid' | qsub -q ac
21866.atcgrid
```

La siguiente imagen muestra la salida del script pmvt-OpenMP_atcgrid.sh en atcgrid:

```
[FelixRamirezGarcia A3estudiante20@atcgrid:~] 2019-05-18 Saturday
$cat STDIN.e21866
[FelixRamirezGarcia A3estudiante20@atcgrid:~] 2019-05-18 Saturday
$cat STDIN.o21866
schedule(static) chunk(defecto)
Tiempo= 0.039028727     Primera=61440   Ultima=4
schedule(static) chunk(1)
Tiempo= 0.038059705     Primera=61440   Ultima=4
schedule(static) chunk(64)
Tiempo= 0.035381175     Primera=61440   Ultima=4
schedule(dynamic) chunk(defecto)
Tiempo= 0.038480219     Primera=61440   Ultima=4
schedule(dynamic) chunk(1)
Tiempo= 0.037924551     Primera=61440   Ultima=4
schedule(dynamic) chunk(64)
Tiempo= 0.034598569     Primera=61440   Ultima=4
schedule(guided) chunk(defecto)
Tiempo= 0.036386421     Primera=61440   Ultima=4
schedule(guided) chunk(1)
Tiempo= 0.041688609     Primera=61440   Ultima=4
schedule(guided) chunk(64)
Tiempo= 0.037753926     Primera=61440   Ultima=4
```

Tabla 3 . Tiempos de ejecución de la versión paralela del producto de una matriz triangular por un vector r para vectores de tamaño N= 15360 , 12 threads

Tabla para la ejecución en local:

Chunk	Static	Dynamic	Guided
por defecto	0.094122000	0.093506000	0.122639000
1	0.066223000	0.068243000	0.152661000
64	0,066384000	0.068390000	0.091913000

A continuación se muestra el grafico de la ejecución del script en local:

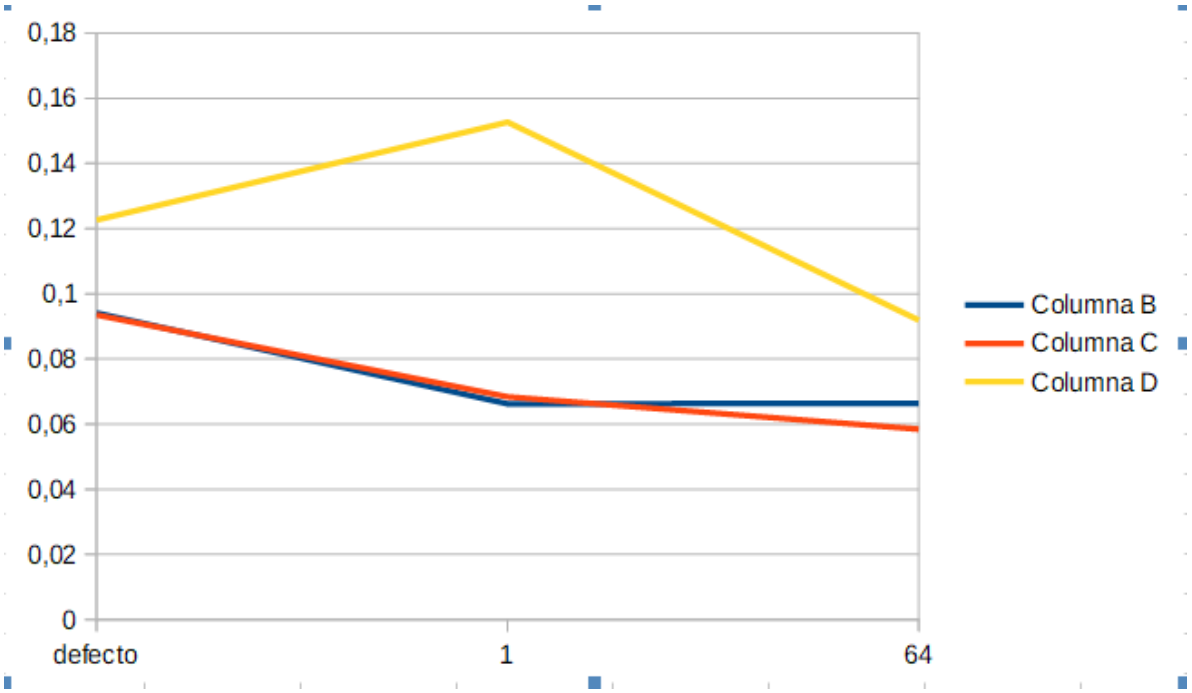
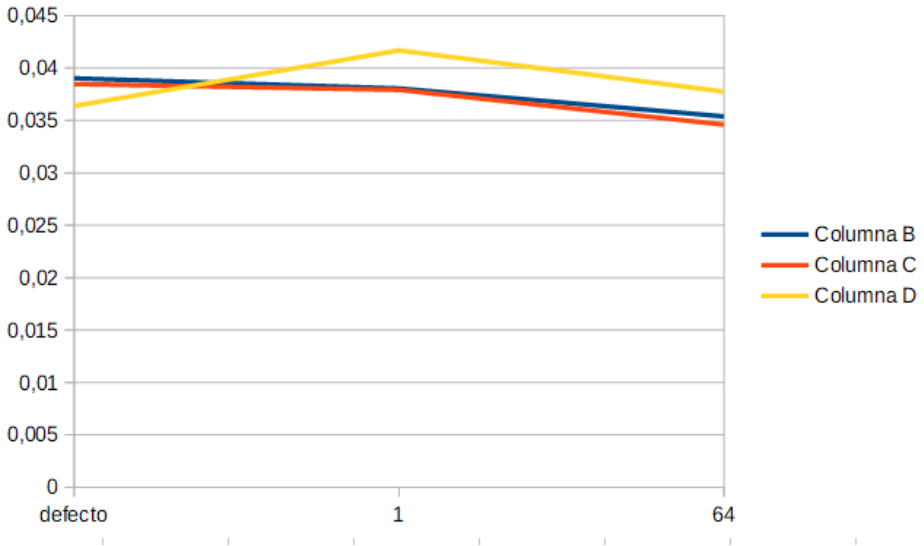


Tabla para la ejecución en atcgrid:

Chunk	Static	Dynamic	Guided
por defecto	0.039028727	0.038480219	0.036386421
1	0.038059705	0.037924551	0.041688609
64	0.035381175	0.034598569	0.037753926

A continuación se muestra el grafico de la ejecución del script en atcgrid



8. Implementar un programa secuencial en C que calcule la multiplicación de matrices cuadradas, B y C:

$$A = B \bullet C;\ \ A(i,j) = \sum_{k=0}^{N-1} B(i,k) \bullet C(k,j),\ \ i,j = 0,\ldots N-1$$

NOTAS: (1) el número de filas/columnas debe ser un argumento de entrada; (2) se deben inicializar las matrices antes del cálculo; (3) se debe imprimir siempre las componentes (0,0) y (N-1, N-1) del resultado antes de que termine el programa.

El código fuente de pmm-secuencial.c es el que se muestra a continuación:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, char **argv){
    int i, j, k, N;
    int **m1, **m2, **mResultado;
    struct timespec inicio, fin;
    double time;

    if(argc<2){
        fprintf(stderr, "falta el tamaño\n");
        exit(-1);
    }

    N=atoi(argv[1]);

    m1=(int**)malloc(N*sizeof(int*));
    m2=(int**)malloc(N*sizeof(int*));
    mResultado=(int**)malloc(N*sizeof(int*));

    for (i=0;i<N;i++){
        m1[i]=(int*)malloc(N*sizeof(int));
        m2[i]=(int*)malloc(N*sizeof(int));
        mResultado[i]=(int*)malloc(N*sizeof(int));
    }

    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            m1[i][j]=2;
            m2[i][j]=2;
            mResultado[i][j]=0;
        }
    }

    clock_gettime(CLOCK_REALTIME,&inicio);
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            for(k=0;k<N;k++) mResultado[i][j]+=m1[i][k]*m2[k][j];
        }
    }
    clock_gettime(CLOCK_REALTIME,&fin);

    time=(double)(fin.tv_sec-inicio.tv_sec)+(double)((fin.tv_nsec-inicio.tv_nsec)/(1.e+9));

    printf("Tiempo=%11.9f\n\t(0,0)=%d\n\t(N-1, N-1)=%d\n", time,mResultado[0][0], mResultado[N-1][N-1]);

    for (i=0; i<N; i++){
        free(m1[i]);
        free(m2[i]);
        free(mResultado[i]);
    }
    free(m1);
    free(m2);
    free(mResultado);
}
```

La siguiente imagen muestra la ejecución:

```
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/
AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer8] 2019-05-18 Saturday
$./pmm-secuencial 5
Tiempo=0.000002500
        (0,0)=20
        (N-1, N-1)=20
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/
AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer8] 2019-05-18 Saturday
$./pmm-secuencial 6
Tiempo=0.000001700
        (0,0)=24
        (N-1, N-1)=24
```

9. Implementar en paralelo la multiplicación de matrices cuadradas con OpenMP a partir del código escrito en el ejercicio anterior. Use las directivas, las cláusulas y las funciones de entorno que considere oportunas. Se debe paralelizar también la inicialización de las matrices. Dibuje en su cuaderno de prácticas la descomposición de dominio que ha utilizado en el código paralelo implementado para asignar tareas a los threads (Lección 4/Tema 2,Lección 5/Tema 2).

El código fuente de pmm-OpenMP.c es el que se muestra a continuación:

```c
int main(int argc, char **argv){
    int i, j, k, N;
    int **m1, **m2, **mResultado;
    struct timespec inicio, fin;
    double time;

    if(argc<2){
        fprintf(stderr, "falta el tamaño\n");
        exit(-1);
    }

    N=atoi(argv[1]);

    m1=(int**)malloc(N*sizeof(int*));
    m2=(int**)malloc(N*sizeof(int*));
    mResultado=(int**)malloc(N*sizeof(int*));

    for (i=0;i<N;i++){
        m1[i]=(int*)malloc(N*sizeof(int));
        m2[i]=(int*)malloc(N*sizeof(int));
        mResultado[i]=(int*)malloc(N*sizeof(int));
    }

    #pragma omp parallel for private(j)
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            m1[i][j]=2;
            m2[i][j]=2;
            mResultado[i][j]=0;
        }
    }

    clock_gettime(CLOCK_REALTIME,&inicio);
    #pragma omp parallel for private(j, k)
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            for(k=0;k<N;k++) mResultado[i][j]+=m1[i][k]*m2[k][j];
        }
    }
    clock_gettime(CLOCK_REALTIME,&fin);

    time=(double)(fin.tv_sec-inicio.tv_sec)+(double)((fin.tv_nsec-inicio.tv_nsec)/(1.e+9));

    printf("Tiempo=%11.9f\n\t(0,0)=%d\n\t(N-1, N-1)=%d\n", time,mResultado[0][0], mResultado[N-1][N-1]);

    for (i=0; i<N; i++){
        free(m1[i]);
        free(m2[i]);
        free(mResultado[i]);
    }
    free(m1);
    free(m2);
    free(mResultado);
}
```

La siguiente imagen muestra la ejecución:

```
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/
AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer9] 2019-05-18 Saturday
$gcc -O2 -fopenmp -o pmm-OpenMP pmm-OpenMP.c
[FelixRamirezGarcia felix@DESKTOP-8PO8DVP:/mnt/c/Users/felix/Desktop/Google-Drive/
AC/Practicas/Entrega/bp3_RamirezGarciaFelix/ejer9] 2019-05-18 Saturday
$./pmm-OpenMP 100
Tiempo=0.000603700
       (0,0)=400
       (N-1, N-1)=400
```

10. Hacer un estudio de escalabilidad (ganancia en velocidad en función del número de cores) en atcgrid y en su PC del código paralelo implementado para dos tamaños de las matrices. Debe recordar usar `-O2` al compilar. El número de núcleos máximo en este estudio debe ser el igual al de núcleos físicos del computador. Presente los resultados del estudio en tablas de valores y en gráficas. Escoger los tamaños de manera que se observe diferentes curvas de escalabilidad en las gráficas que entregue en su cuaderno de prácticas (pruebe con valores de N entre 100 y 1500). Consulte la Lección 6/Tema 2. Incluya los scripts utilizado en el cuaderno de prácticas. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

Antes de comenzar con el estudio de escalabilidad vamos a mostrar el código del script pmm-OpenMP_atcgrid.sh :

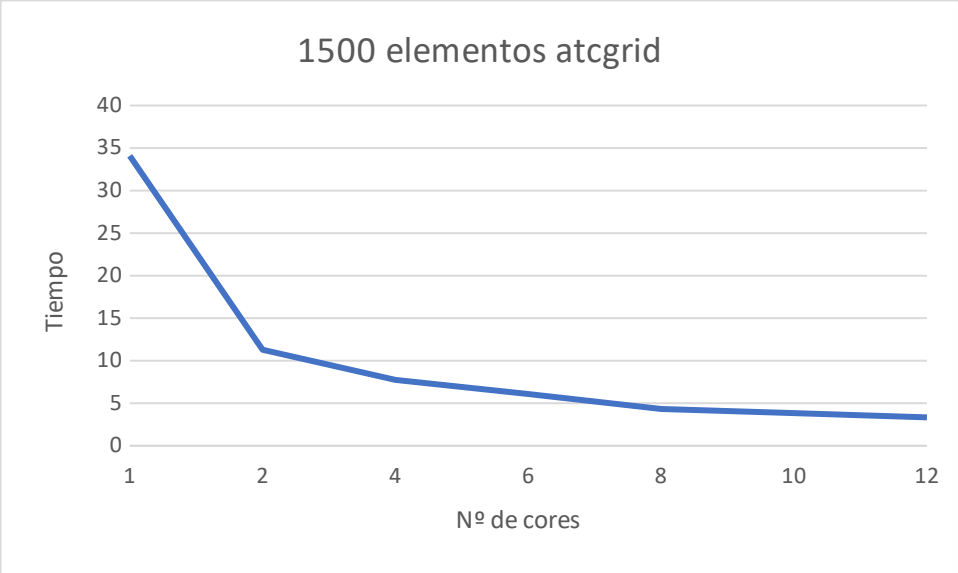
```
#!/bin/bash

export OMP_SCHEDULE="static"
echo "secuencial"
echo "100 elementos"
./pmm-secuencial 100
echo "1500 elementos"
./pmm-secuencial 1500

export OMP_NUM_THREADS=2
echo "2 cores"
echo "100 elementos"
./pmm-OpenMP 100
echo "1500 elementos"
./pmm-OpenMP 1500

export OMP_NUM_THREADS=4
echo "4 cores"
echo "100 elementos"
./pmm-OpenMP 100
echo "1500 elementos"
./pmm-OpenMP 1500

export OMP_NUM_THREADS=6
echo "6 cores"
echo "100 elementos"
./pmm-OpenMP 100
echo "1500 elementos"
./pmm-OpenMP 1500

export OMP_NUM_THREADS=8
echo "8 cores"
echo "100 elementos"
./pmm-OpenMP 100
echo "1500 elementos"
./pmm-OpenMP 1500

export OMP_NUM_THREADS=10
echo "10 cores"
echo "100 elementos"
./pmm-OpenMP 100
echo "1500 elementos"
./pmm-OpenMP 1500

export OMP_NUM_THREADS=12
echo "12 cores"
echo "100 elementos"
./pmm-OpenMP 100
echo "1500 elementos"
./pmm-OpenMP 1500
```

ESTUDIO DE ESCALABILIDAD EN atcgrid:

Cores	100 elementos	1500 elementos
1	0,002035586	34,084211285
2	0,001124368	11,293998479
4	0,000413716	7,744093488
6	0,000285647	6,089357064
8	0,000227823	4,336290691
10	0,000193739	3,844367677
12	0,000172742	3,341043423

A continuación se muestra la grafica de la ejecución del script en atcgrid para 100 elementos:

A continuación se muestra la grafica de la ejecución del script en atcgrid para 1500 elementos:



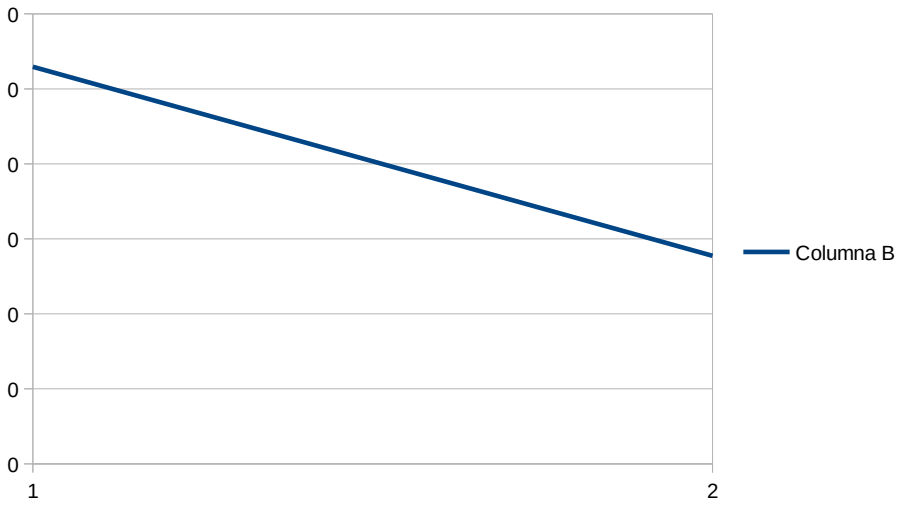
Antes de comenzar con el estudio de escalabilidad vamos a mostrar el código del script pmm-OpenMP_pclocal.sh:

```
#!/bin/bash

export OMP_SCHEDULE="static"
echo "secuencial"
echo "100 elementos"
./pmm-secuencial 100
echo "1500 elementos"
./pmm-secuencial 1500

export OMP_NUM_THREADS=2
echo "2 cores"
echo "100 elementos"
./pmm-OpenMP 100
echo "1500 elementos"
./pmm-OpenMP 1500
```

ESTUDIO DE ESCALABILIDAD EN PCLOCAL:

Cores	100 elementos	1500 elementos
1	0,001058900	13,837992500
2	0,000555100	9,872938600

A continuación se muestra la grafica de la ejecución del script en local para 100 elementos:



A continuación se muestra la grafica de la ejecución del script en local para 1500 elementos: