2º curso / 2º cuatr.

Grado Ing. Inform.

Doble Grado Ing. Inform. y Mat.

Arquitectura de Computadores (AC)

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

Estudiante (nombre y apellidos): SERGIO AGUILERA RAMIREZ
Grupo de prácticas: A2
Fecha de entrega:12-05-18
Fecha evaluación en clase: 15-05-18

Ejercicios basados en los ejemplos del seminario práctico

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.

CAPTURA CÓDIGO FUENTE: `if-clauseModificado.c`

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


            int main(int argc, char **argv)
            {
                        int i, n=20, tid;
                        int a[n],suma=0,sumalocal;
                        int x;

                        if(argc < 3){
                                    fprintf(stderr,"[ERROR]-Falta
iteraciones\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 num_threads(x) if(n>4)
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);
                        }
                }
```

CAPTURAS DE PANTALLA:

```
[Sergio Aguilera Ramirez usuario@cvihs220185:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-04-24 martes
$gcc-7 -O2 -fopenmp if-clauseModificado.c -o if-clauseModificado
[Sergio Aguilera Ramirez usuario@cvihs220185:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-04-24 martes
$./if-clauseModificado 4 4
 thread 0 suma de a[0]=0 sumalocal=0
 thread 0 suma de a[1]=1 sumalocal=1
 thread 0 suma de a[2]=2 sumalocal=3
 thread 0 suma de a[3]=3 sumalocal=6
thread master=0 imprime suma=6
[Sergio Aguilera Ramirez usuario@cvihs220185:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-04-24 martes
$./if-clauseModificado 5 4
 thread 0 suma de a[0]=0 sumalocal=0
 thread 1 suma de a[2]=2 sumalocal=2
 thread 3 suma de a[4]=4 sumalocal=4
 thread 2 suma de a[3]=3 sumalocal=3
 thread 0 suma de a[1]=1 sumalocal=1
thread master=0 imprime suma=10
```

RESPUESTA: al incorporar el nuevo argumento nos permite cambiar el numero de hebras, al declarar la if (n>4) el número mínimo de hebras para entrar en la región paralela es 5. Como podemos observar si n <= 4 el código solo lo ejecuta la hebra máster.

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	1	1	0	0	1	0
1	1	0	0	0	1	0	0	1	0

2	0	1	0	0	0	0	0	1	0
3	1	1	0	1	0	0	0	1	0
4	0	0	1	0	1	1	0	1	0
5	1	0	1	1	1	1	0	1	0
6	0	1	1	0	0	1	0	1	0
7	1	1	1	1	0	1	0	1	0
8	0	0	0	0	1	0	1	0	1
9	1	0	0	1	1	0	1	0	1
10	0	1	0	0	0	0	1	0	1
11	1	1	0	1	0	0	1	0	1
12	0	0	1	0	1	1	1	0	1
13	1	0	1	1	1	1	1	0	1
14	0	1	1	0	0	1	1	0	1
15	1	1	1	1	0	1	1	0	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	0	0	0	0	0	2	1	2	1
1	1	0	0	3	0	2	1	2	1
2	2	1	0	2	3	2	1	2	1
3	3	1	0	3	3	2	1	2	1
4	0	2	1	3	2	0	2	2	0
5	1	2	1	0	2	0	2	2	0
6	2	3	1	2	1	0	2	2	0
7	3	3	1	1	1	0	0	2	0
8	0	0	2	3	3	1	0	2	2
9	1	0	2	0	3	1	0	2	2
10	2	1	2	2	0	1	3	2	2
11	3	1	2	1	0	1	3	2	2
12	0	2	3	3	2	3	3	2	3
13	1	2	3	0	2	3	3	2	3
14	2	3	3	2	1	3	0	0	3
15	3	3	3	1	1	3	2	0	3

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

RESPUESTA: con static las tareas se reparten igualmente entre los threads , y con dynamic y guided no se sabe como se repartirá la tarea entre los threads, pero si se sabe que

el tamaño de la tarea vendrá definido por chunk (cada hebra realizará chunk iteraciones)

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?

CAPTURA CÓDIGO FUENTE: `scheduled-clauseModificado.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, valor_chunk;
        omp_sched_t schedule_type;

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

        n = atoi(argv[1]);

        if (n>200){
            n=200;
        }

        chunk = atoi(argv[2]);

        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");
                printf("  static = 1, dynamic = 2, guided = 3, auto = 4\n");
                omp_get_schedule(&schedule_type, & valor_chunk);
                printf(" dyn-var: %d, num_threads-var: %d, limit_threads-var:
%d, run_sched-var: %d, chunk: %d\n", omp_get_dynamic(), omp_get_max_threads(),
omp_get_thread_limit(), schedule_type, valor_chunk);
            }
        }

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

```
    printf("  static = 1, dynamic = 2, guided = 3, auto = 4\n");
    omp_get_schedule(&schedule_type, &valor_chunk);
    printf("  dyn-var: %d, num_threads-var: %d, limit_threads-var: %d,
run_sched-var: %d, chunk: %d\n", omp_get_dynamic(), omp_get_max_threads(),
omp_get_thread_limit(), schedule_type, valor_chunk);

    }
```

CAPTURAS DE PANTALLA:

```
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$gcc-7 -O2 -fopenmp scheduled-clauseModificado.c -o scheduled-clauseModificado
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$./scheduled-clauseModificado 5 2
 thread 0 suma a[0]=0 suma=0
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
 thread 0 suma a[1]=1 suma=1
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
 thread 0 suma a[2]=2 suma=3
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
 thread 0 suma a[3]=3 suma=6
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
 thread 0 suma a[4]=4 suma=10
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
Fuera de 'parallel for' suma=10
  static = 1, dynamic = 2, guided = 3, auto = 4
  dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
```

OMP_SCHEDULE=”static,2”

```
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$export OMP_SCHEDULE="static,2"
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$./scheduled-clauseModificado 5 2
 thread 0 suma a[0]=0 suma=0
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 1, chunk: 2
 thread 0 suma a[1]=1 suma=1
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 1, chunk: 2
 thread 0 suma a[2]=2 suma=3
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 1, chunk: 2
 thread 0 suma a[3]=3 suma=6
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 1, chunk: 2
 thread 0 suma a[4]=4 suma=10
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 1, chunk: 2
Fuera de 'parallel for' suma=10
  static = 1, dynamic = 2, guided = 3, auto = 4
  dyn-var: 0, num_threads-var: 1, limit_threads-var: 2147483647, run_sched-var: 1, chunk: 2
```

OMP_THREAD_LIMIT = 4

```
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$export OMP_THREAD_LIMIT=4
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$./scheduled-clauseModificado 5 2
 thread 0 suma a[0]=0 suma=0
 thread 1 suma a[2]=2 suma=2
 thread 2 suma a[4]=4 suma=4
Dentro de 'parallel for':
 thread 1 suma a[3]=3 suma=5
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 9, limit_threads-var: 4, run_sched-var: 1, chunk: 2
 thread 0 suma a[1]=1 suma=1
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 0, num_threads-var: 9, limit_threads-var: 4, run_sched-var: 1, chunk: 2
Fuera de 'parallel for' suma=4
  static = 1, dynamic = 2, guided = 3, auto = 4
  dyn-var: 0, num_threads-var: 9, limit_threads-var: 4, run_sched-var: 1, chunk: 2
```

OMP_NUM_THREADS=9

```
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$export OMP_NUM_THREADS=9
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$./scheduled-clauseModificado 5 2
 thread 3 suma a[2]=2 suma=2
 thread 4 suma a[0]=0 suma=0
 thread 5 suma a[4]=4 suma=4
 thread 3 suma a[3]=3 suma=5
 thread 4 suma a[1]=1 suma=1
Fuera de 'parallel for' suma=4
  static = 1, dynamic = 2, guided = 3, auto = 4
  dyn-var: 0, num_threads-var: 9, limit_threads-var: 2147483647, run_sched-var: 1, chunk: 2
```

OMP_DYNAMIC=TRUE

```
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$export OMP_DYNAMIC=TRUE
[Sergio Aguilera Ramirez usuario@cvihs218060:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-04 viernes
$./scheduled-clauseModificado 5 2
 thread 0 suma a[0]=0 suma=0
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 1, num_threads-var: 9, limit_threads-var: 4, run_sched-var: 1, chunk: 2
 thread 0 suma a[1]=1 suma=1
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 1, num_threads-var: 9, limit_threads-var: 4, run_sched-var: 1, chunk: 2
 thread 0 suma a[2]=2 suma=3
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 1, num_threads-var: 9, limit_threads-var: 4, run_sched-var: 1, chunk: 2
 thread 0 suma a[3]=3 suma=6
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 1, num_threads-var: 9, limit_threads-var: 4, run_sched-var: 1, chunk: 2
 thread 0 suma a[4]=4 suma=10
Dentro de 'parallel for':
  static = 1, dynamic = 2, guided = 3, auto = 4
 dyn-var: 1, num_threads-var: 9, limit_threads-var: 4, run_sched-var: 1, chunk: 2
Fuera de 'parallel for' suma=10
  static = 1, dynamic = 2, guided = 3, auto = 4
  dyn-var: 1, num_threads-var: 9, limit_threads-var: 4, run_sched-var: 1, chunk: 2
```

RESPUESTA: El resultado es el mismo tanto fuera de la región parallel como dentro de ella.

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.

CAPTURA CÓDIGO FUENTE: `scheduled-clauseModificado4.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, valor_chunk;
        omp_sched_t schedule_type;

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

        n = atoi(argv[1]);

        if (n>200){
            n=200;
        }

        chunk = atoi(argv[2]);

        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("\nDentro de 'parallel for':\n");
                printf("static = 1, dynamic = 2, guided = 3, auto = 4\n");
                omp_get_schedule(&schedule_type, & valor_chunk);
                printf("dyn-var: %d, num_threads-var: %d, limit_threads-var:
%d, run_sched-var: %d, chunk: %d\n", omp_get_dynamic(),
```

```
omp_get_max_threads(), omp_get_thread_limit(), schedule_type, valor_chunk);
                printf("get_num_htreads: %d \nget_num_procs: %d \
nin_parallel(): %d \n", omp_get_num_threads(), omp_get_num_procs(),
omp_in_parallel());
            }
        }

    printf("\nFuera de 'parallel for' suma=%d\n",suma);
    printf("static = 1, dynamic = 2, guided = 3, auto = 4\n");
    omp_get_schedule(&schedule_type, &valor_chunk);
    printf("dyn-var: %d, num_threads-var: %d, limit_threads-var: %d,
run_sched-var: %d, chunk: %d\n", omp_get_dynamic(), omp_get_max_threads(),
omp_get_thread_limit(), schedule_type, valor_chunk);
    printf("get_num_htreads: %d \nget_num_procs: %d \nin_parallel(): %d \n",
omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());

    }
```

CAPTURAS DE PANTALLA:

```
$gcc-7 -O2 -fopenmp scheduled-clauseModificado4.c -o scheduled-clauseModificado4
[Sergio Aguilera Ramirez usuario@cvi096052:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-07 lunes
$./scheduled-clauseModificado4 5 2
 thread 0 suma a[0]=0 suma=0
 thread 1 suma a[2]=2 suma=2
 thread 3 suma a[4]=4 suma=4

Dentro de 'parallel for':
 thread 1 suma a[3]=3 suma=5
static = 1, dynamic = 2, guided = 3, auto = 4
dyn-var: 0, num_threads-var: 4, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
get_num_htreads: 4
get_num_procs: 4
in_parallel(): 1
 thread 0 suma a[1]=1 suma=1

Dentro de 'parallel for':
static = 1, dynamic = 2, guided = 3, auto = 4
dyn-var: 0, num_threads-var: 4, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
get_num_htreads: 4
get_num_procs: 4
in_parallel(): 1

Fuera de 'parallel for' suma=4
static = 1, dynamic = 2, guided = 3, auto = 4
dyn-var: 0, num_threads-var: 4, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
get_num_htreads: 1
get_num_procs: 4
in_parallel(): 0
```

```
[Sergio Aguilera Ramirez usuario@cvi096052:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-07 lunes
$./scheduled-clauseModificado4 5 1
 thread 2 suma a[1]=1 suma=1
 thread 1 suma a[0]=0 suma=0
 thread 0 suma a[2]=2 suma=2
 thread 3 suma a[3]=3 suma=3
 thread 2 suma a[4]=4 suma=5

Dentro de 'parallel for':
static = 1, dynamic = 2, guided = 3, auto = 4
dyn-var: 0, num_threads-var: 4, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
get_num_htreads: 4
get_num_procs: 4
in_parallel(): 1

Fuera de 'parallel for' suma=5
static = 1, dynamic = 2, guided = 3, auto = 4
dyn-var: 0, num_threads-var: 4, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
get_num_htreads: 1
get_num_procs: 4
in_parallel(): 0
```

RESPUESTA: La única función que devuelve el mismo valor fuera y dentro de la región parallel es get_num_procs().

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.

CAPTURA CÓDIGO FUENTE: `scheduled-clauseModificado5.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, valor_chunk;
        omp_sched_t schedule_type;

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

        n = atoi(argv[1]);

        if (n>200){
            n=200;
        }
```

```
        chunk = atoi(argv[2]);

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


        printf("Antes de cambiar las variables\n");
        printf("static = 1, dynamic = 2, guided = 3, auto = 4\n");
        omp_get_schedule(&schedule_type, & valor_chunk);
        printf("dyn-var: %d, num_threads-var: %d, limit_threads-var: %d,
run_sched-var: %d, chunk: %d\n", omp_get_dynamic(), omp_get_max_threads(),
omp_get_thread_limit(), schedule_type, valor_chunk);
        printf("get_num_htreads: %d \nget_num_procs: %d \nin_parallel(): %d \
n", omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());

        omp_set_dynamic(2);
        omp_set_num_threads(2);
        omp_set_schedule(2,1);

    #pragma omp parallel for firstprivate(suma) \
        lastprivate(suma) schedule(dynamic,chunk)
        for (i=0; i<n; i++){
            suma = suma + a[i];
            printf("\nthread %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");
                //printf("static = 1, dynamic = 2, guided = 3, auto = 4\n");
                //omp_get_schedule(&schedule_type, & valor_chunk);
                //printf("dyn-var: %d, num_threads-var: %d, limit_threads-var:
%d, run_sched-var: %d, chunk: %d\n", omp_get_dynamic(), omp_get_max_threads(),
omp_get_thread_limit(), schedule_type, valor_chunk);
                //printf("get_num_htreads: %d \nget_num_procs: %d \
nin_parallel(): %d \n", omp_get_num_threads(), omp_get_num_procs(),
omp_in_parallel());
            }
        }

    printf("\nFuera de 'parallel for' suma=%d\n",suma);
    printf("static = 1, dynamic = 2, guided = 3, auto = 4\n");
    omp_get_schedule(&schedule_type, &valor_chunk);
    printf("dyn-var: %d, num_threads-var: %d, limit_threads-var: %d,
run_sched-var: %d, chunk: %d\n", omp_get_dynamic(), omp_get_max_threads(),
omp_get_thread_limit(), schedule_type, valor_chunk);
    printf("get_num_htreads: %d \nget_num_procs: %d \nin_parallel(): %d \n",
omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());

    }
```

CAPTURAS DE PANTALLA:

```
[Sergio Aguilera Ramirez usuario@cvi096052:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-07 lunes
$gcc-7 -O2 -fopenmp scheduled-clauseModificado5.c -o scheduled-clauseModificado5
[Sergio Aguilera Ramirez usuario@cvi096052:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-07 lunes
$./scheduled-clauseModificado5 5 1
Antes de cambiar las variables
static = 1, dynamic = 2, guided = 3, auto = 4
dyn-var: 0, num_threads-var: 4, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
get_num_htreads: 1
get_num_procs: 4
in_parallel(): 0

thread 0 suma a[0]=0 suma=0
Dentro de 'parallel for':

thread 0 suma a[1]=1 suma=1
Dentro de 'parallel for':

thread 0 suma a[2]=2 suma=3
Dentro de 'parallel for':

thread 0 suma a[3]=3 suma=6
Dentro de 'parallel for':

thread 0 suma a[4]=4 suma=10
Dentro de 'parallel for':

Fuera de 'parallel for' suma=10
static = 1, dynamic = 2, guided = 3, auto = 4
dyn-var: 1, num_threads-var: 2, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
get_num_htreads: 1
get_num_procs: 4
in_parallel(): 0
```

```
[Sergio Aguilera Ramirez usuario@cvi096052:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-07 lunes
$./scheduled-clauseModificado5 5 2
Antes de cambiar las variables
static = 1, dynamic = 2, guided = 3, auto = 4
dyn-var: 0, num_threads-var: 4, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
get_num_htreads: 1
get_num_procs: 4
in_parallel(): 0

thread 0 suma a[0]=0 suma=0
Dentro de 'parallel for':

thread 0 suma a[1]=1 suma=1
Dentro de 'parallel for':

thread 0 suma a[2]=2 suma=3
Dentro de 'parallel for':

thread 0 suma a[3]=3 suma=6
Dentro de 'parallel for':

thread 0 suma a[4]=4 suma=10
Dentro de 'parallel for':

Fuera de 'parallel for' suma=10
static = 1, dynamic = 2, guided = 3, auto = 4
dyn-var: 1, num_threads-var: 2, limit_threads-var: 2147483647, run_sched-var: 2, chunk: 1
get_num_htreads: 1
get_num_procs: 4
in_parallel(): 0
```

RESPUESTA: Los valores devueltos por las funciones antes y después de cambiar los valores son iguales.

Resto de ejercicios

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.

CAPTURA CÓDIGO FUENTE: `pmtv-secuencial.c`

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

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

                        unsigned int N = atoi(argv[1]);

                        int *vector, *result, **matriz;
                        vector = (int *) malloc(N*sizeof(int));
                        result = (int *) malloc(N*sizeof(int));
                        matriz = (int **) malloc(N*sizeof(int*));

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

                        for( int i=0; i<N; i++){
                                    for( int j=i; j<N; j++){
                                                matriz[i][j] = 4;
                                                vector[i] = 2;
                                                result[i] = 0;
                                    }
                        }

                        printf("\nMATRIZ\n");
                        for( int i=0; i<N; i++ ){
                                    for( int j=0; j<N; j++ ){
                                                if (j>= i){
                                                            printf("%d ",
matriz[i][j]);
                                                }
                                                else{
                                                            printf("0 ");
                                                }
                                    }
                                    printf("\n");
                        }


                        printf("\nVECTOR\n");
                        for ( int i=0; i<N; i++ ){
                                    printf("%d ", vector[i] );
                        }
                        printf("\n");
```

```
                        for (int i = 0; i < N; i++){
                                for ( int j=i; j<N; j++ ){
                                        result[i] += matriz[i][j]
* vector[j];
                                }

                        }

                        printf("\nRESULTADOS\n");
                        for ( int i=0; i<N; i++ ){
                                printf("%d ", result[i]);
                        }
                                printf("\n\n");

                        for (int i = 0; i < N; ++i){
                                free(matriz[i]);
                        }

                        free(matriz);
                        free(vector);
                        free(result);

                        return 0;

                }
```

CAPTURAS DE PANTALLA:

```
[Sergio Aguilera Ramirez usuario@cvihs223089:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$gcc-7 -O2 -fopenmp pmtv-secuencial.c -o pmtv-secuencial
[Sergio Aguilera Ramirez usuario@cvihs223089:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmtv-secuencial 5

MATRIZ
4 4 4 4 4
0 4 4 4 4
0 0 4 4 4
0 0 0 4 4
0 0 0 0 4

VECTOR
2 2 2 2 2

RESULTADOS
40 32 24 16 8

Primera: 40      Ultima: 8
```

```
[Sergio Aguilera Ramirez usuario@cvihs223089:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmtv-secuencial 10

MATRIZ
4 4 4 4 4 4 4 4 4 4
0 4 4 4 4 4 4 4 4 4
0 0 4 4 4 4 4 4 4 4
0 0 0 4 4 4 4 4 4 4
0 0 0 0 4 4 4 4 4 4
0 0 0 0 0 4 4 4 4 4
0 0 0 0 0 0 4 4 4 4
0 0 0 0 0 0 0 4 4 4
0 0 0 0 0 0 0 0 4 4
0 0 0 0 0 0 0 0 0 4

VECTOR
2 2 2 2 2 2 2 2 2 2

RESULTADOS
80 72 64 56 48 40 32 24 16 8

Primera: 80      Ultima: 8
```

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?

RESPUESTA:

(a) Los valores por defecto de chunk son: static no tiene y para dynamic y guided el valor de chunk es 1.

(b) Este hará N operaciones, siendo N = chunk * número de fila

(c) En la planificación guided el último valor calculado no será igual, el resto puede que se ejecuten 64 operaciones al mismo tiempo o 1 sola operación.

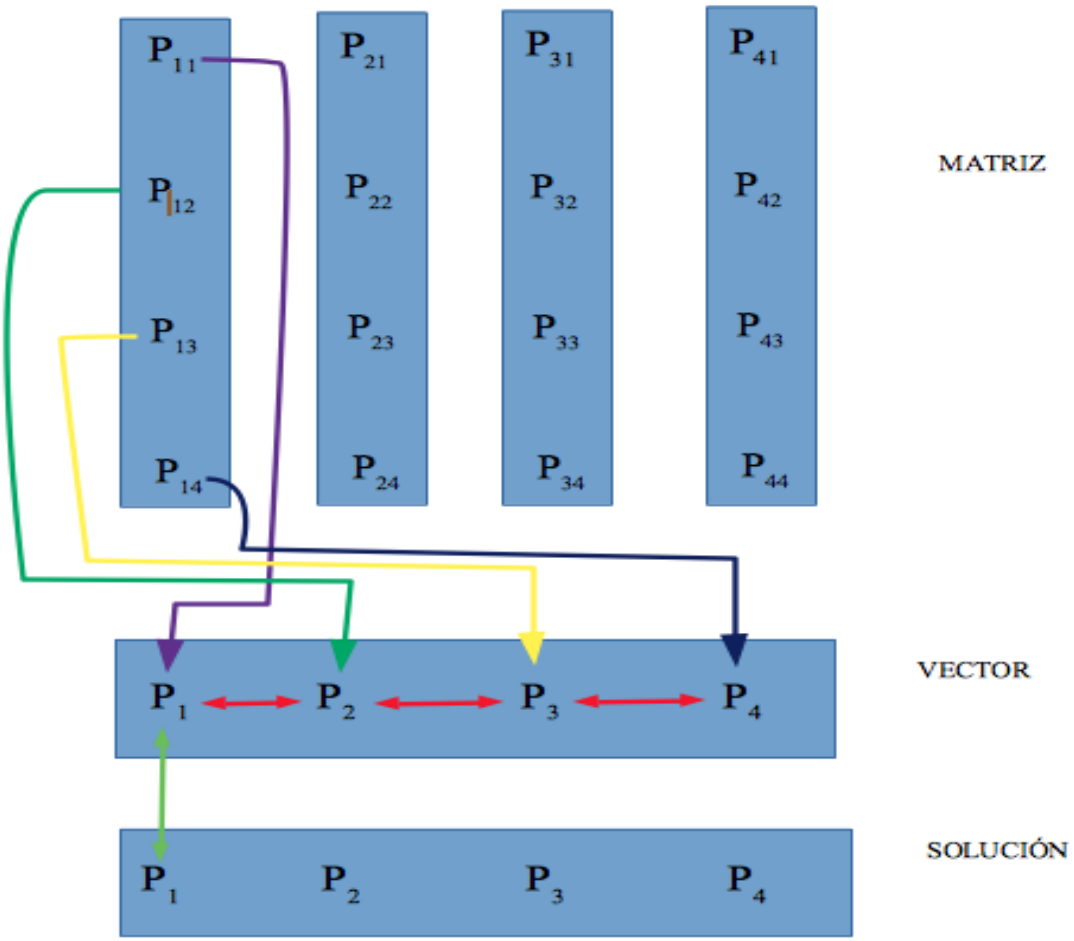
CAPTURA CÓDIGO FUENTE: `pmtv-OpenMP.c`

```
#include <stdlib.h>
#include <stdio.h>
#include <time.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
            #define omp_set_dynamic(int)
#endif

            int main ( int argc, char **argv ){

                        int j,debug = 0;

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

                        if ( argc == 3 ){
                                    debug = atoi(argv[2]);
                        }

                        unsigned int N = atoi(argv[1]);

                        int *vector, *result, **matriz;
                        vector = (int *) malloc(N*sizeof(int));
                        result = (int *) malloc(N*sizeof(int));
                        matriz = (int **) malloc(N*sizeof(int*));

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

                        for( int i=0; i<N; i++){
                                    for(  j=i; j<N; j++){
                                                matriz[i][j] = 2;
                                                vector[i] = 4;
                                                result[i] = 0;
                                    }
                        }

                        if ( debug == 1 ){

                                    printf("\nMATRIZ\n");
                                    for( int i=0; i<N; i++ ){
                                                for(  j=0; j<N; j++ ){
                                                            if (j>= i){

            printf("%d ", matriz[i][j]);
                                                            }
                                                            else{

            printf("0 ");
                                                            }
                                                }
                                                printf("\n");
                                    }


                                    printf("\nVECTOR\n");
                                    for ( int i=0; i<N; i++ ){
```

```
                                                printf("%d ",
vector[i] );
                                        }
                                        printf("\n");

                        }

                        double comienzo, fin, total;
                        comienzo = omp_get_wtime();


                        #pragma omp parallel for private(j)
schedule(runtime)
                        for (int i = 0; i < N; i++){
                                for ( int j=i; j<N; j++ ){
                                        result[i] += matriz[i]
[j] * vector[j];
                                }
                        }

                        fin = omp_get_wtime();
                        total = fin - comienzo;

                        if ( debug == 1 ){
                                printf("\nRESULTADOS\n");
                                for ( int i=0; i<N; i++ ){
                                        printf("%d ",
result[i]);
                                }
                                        printf("\n\n");
                        }


                        printf("Tiempo: %11.9f\t Primera: %d\t Ultima: %d\
n", total, result[0], result[N-1]);


                        for (int i = 0; i < N; ++i){
                                free(matriz[i]);
                        }

                        free(matriz);
                        free(vector);
                        free(result);

                        return 0;

                }
```

DESCOMPOSICIÓN DE DOMINIO:



CAPTURAS DE PANTALLA:

```
[Sergio Aguilera Ramirez usuario@cvihs223089:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$gcc-7 -O2 -fopenmp pmtv-OpenMP.c -o pmtv-OpenMP
[Sergio Aguilera Ramirez usuario@cvihs223089:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmtv-OpenMP 10 1

MATRIZ
2 2 2 2 2 2 2 2 2 2
0 2 2 2 2 2 2 2 2 2
0 0 2 2 2 2 2 2 2 2
0 0 0 2 2 2 2 2 2 2
0 0 0 0 2 2 2 2 2 2
0 0 0 0 0 2 2 2 2 2
0 0 0 0 0 0 2 2 2 2
0 0 0 0 0 0 0 2 2 2
0 0 0 0 0 0 0 0 2 2
0 0 0 0 0 0 0 0 0 2

VECTOR
4 4 4 4 4 4 4 4 4 4

RESULTADOS
80 72 64 56 48 40 32 24 16 8

Tiempo: 0.000411000      Primera: 80     Ultima: 8
```

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid

SCRIPT: pmvt-OpenMP_PCaula.sh

```
    #!/bin/bash
10. #Se asigna al trabajo el nombre pmtv-OpemMp_atcgrid
11. #PBS -N pmtv-OpemMp_atcgrid
12. #Se asigna al trabajo la cola ac
13. #PBS -q ac
14. #Se imprime información del trabajo usando variables de entorno de PBS
15. echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
16. echo "Id. del trabajo: $PBS_JOBID"
17. echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
18. echo "Nodo que ejecuta qsub: $PBS_O_HOST"
19. echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
20. echo "Cola: $PBS_QUEUE"
21. echo "Nodos asignados al trabajo:"
22. #cat $PBS_NODEFILE
23.
24.
25. export OMP_SCHEDULE="static"
26. echo "static y chunk por defecto"
27. $PBS_O_WORKDIR/pmtv-OpenMP 15360
28.
29. export OMP_SCHEDULE="static,1"
30. echo "static y chunk 1"
31. $PBS_O_WORKDIR/pmtv-OpenMP 15360
32.
33. export OMP_SCHEDULE="static,64"
34. echo "static y chunk 64"
35. $PBS_O_WORKDIR/pmtv-OpenMP 15360
36.
37. export OMP_SCHEDULE="dynamic"
38. echo "dynamic y chunk por defecto"
39. $PBS_O_WORKDIR/pmtv-OpenMP 15360
40.
41. export OMP_SCHEDULE="dynamic,1"
42. echo "dynamic y chunk 1"
43. $PBS_O_WORKDIR/pmtv-OpenMP 15360
44.
45. export OMP_SCHEDULE="dynamic,64"
46. echo "dynamic y chunk 64"
```

```
47. $PBS_O_WORKDIR/pmtv-OpenMP 15360
48.
49. export OMP_SCHEDULE="guided"
50. echo "guided y chunk por defecto"
51. $PBS_O_WORKDIR/pmtv-OpenMP 15360
52.
53. export OMP_SCHEDULE="guided,1"
54. echo "guided y chunk 1"
55. $PBS_O_WORKDIR/pmtv-OpenMP 15360
56.
57. export OMP_SCHEDULE="guided,64"
58. echo "guided y chunk 64"
59. $PBS_O_WORKDIR/pmtv-OpenMP 15360
```

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

Chunk	Static	Dynamic	Guided
por defecto	0,038131004	0,039453949	0,038453000
1	0,035756012	0,035645208	0,041927476
64	0,039494647	0,033914737	0,035406357
Chunk	**Static**	**Dynamic**	**Guided**
por defecto	0,039636854	0,037144384	0,035714374
1	0,038236322	0,037052903	0,037632747
64	0,036148358	0,032630344	0,038906933

TABLA 1:

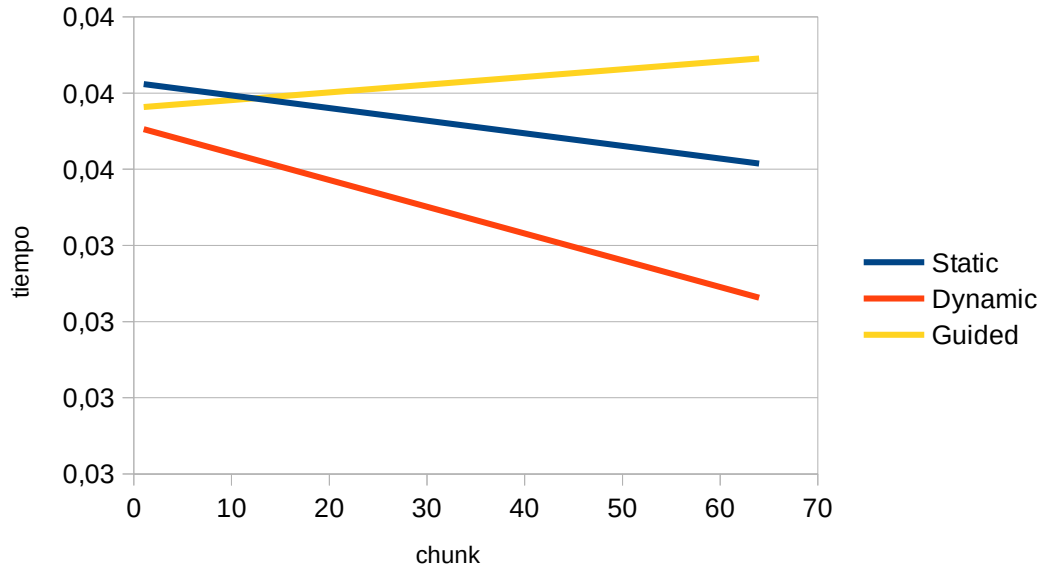
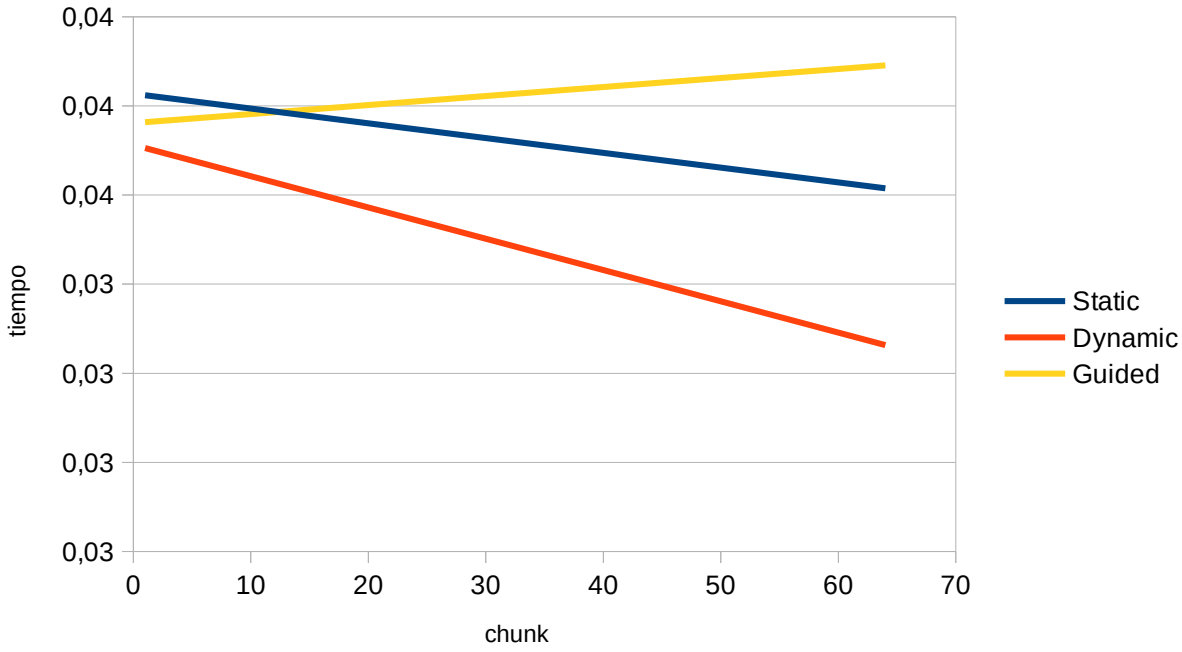


TABLA 2:



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.

CAPTURA CÓDIGO FUENTE: `pmm-secuencial.c`

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

              int main (int argc, char **argv){

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


                            unsigned int N = atoi(argv[1]);

                            int **a, **b, **c;

                            a = (int **) malloc(N*sizeof(int*));
                            b = (int **) malloc(N*sizeof(int*));
                            c = (int **) malloc(N*sizeof(int*));

                            for (int i=0; i<N; i++){
                                          a[i] = (int *) malloc(N*sizeof(int));
                                          b[i] = (int *) malloc(N*sizeof(int));
                                          c[i] = (int *) malloc(N*sizeof(int));
                            }

                            for(int i=0; i<N; i++){
                                          for( int j=0; j<N; j++){
                                                        a[i][j] = 0;
                                                        b[i][j] = 2;
                                                        c[i][j] = 2;
                                          }
                            }

                            struct timespec cgt1,cgt2; double ncgt;

                            clock_gettime(CLOCK_REALTIME, &cgt1);

                            for( int i=0; i<N; i++){
                                          for( int j=0; j<N; j++){
                                                        for ( int k=0; k<N; k++)
{
                                                                      a[i][j] +=
b[i][k] * c[k][j];
                                                        }
                                          }
                            }

                            clock_gettime(CLOCK_REALTIME, &cgt2);

                            ncgt = (double) (cgt2.tv_sec - cgt1.tv_sec)+
(double) ((cgt2.tv_nsec - cgt1.tv_nsec))/(1.e+9);
                            printf("Tiempo: %11.9F\t Primera: %d\t Ultima: %d\
n",ncgt,a[0][0], a[N-1][N-1]);
```

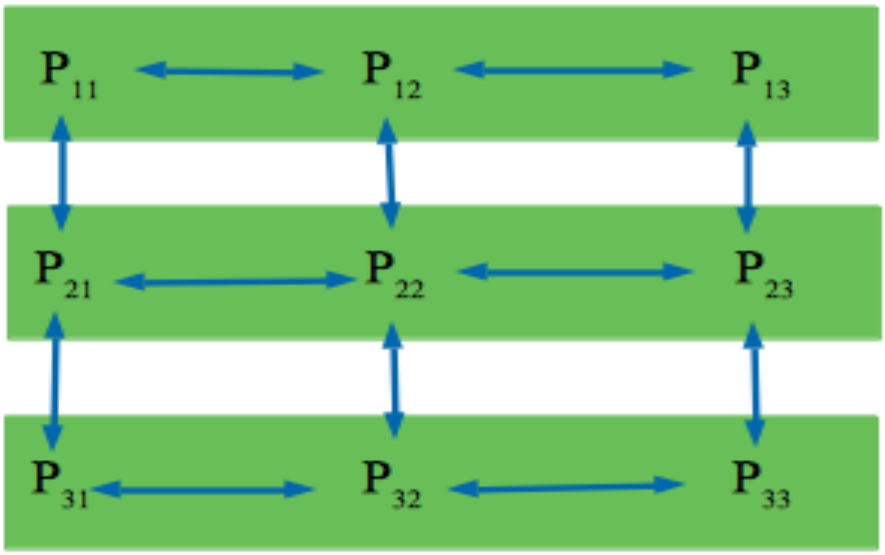
```
                        for(int i=0; i<N; i++){
                                    free(a[i]);
                                    free(b[i]);
                                    free(c[i]);
                        }

                        free(a);
                        free(b);
                        free(c);

                        return 0;
            }
```

CAPTURAS DE PANTALLA:

```
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$gcc-7 -O2 -fopenmp pmm-secuencial.c -o pmm-secuencial
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-secuencial 10
Tiempo: 0.000003000      Primera: 40     Ultima: 40
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-secuencial 20
Tiempo: 0.000013000      Primera: 80     Ultima: 80
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-secuencial 30
Tiempo: 0.000028000      Primera: 120    Ultima: 120
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-secuencial 100
Tiempo: 0.000746000      Primera: 400    Ultima: 400
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-secuencial 1000
Tiempo: 9.645668000      Primera: 4000   Ultima: 4000
```

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).

DESCOMPOSICIÓN DE DOMINIO:



CAPTURA CÓDIGO FUENTE: `pmm-OpenMP.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <time.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
              #define omp_set_dynamic(int)
#endif

              int main (int argc, char **argv){

                            int i, j, k;

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


                            unsigned int N = atoi(argv[1]);

                            int **a, **b, **c;

                            a = (int **) malloc(N*sizeof(int*));
                            b = (int **) malloc(N*sizeof(int*));
                            c = (int **) malloc(N*sizeof(int*));

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

```
                                        b[i] = (int *) malloc(N*sizeof(int));
                                        c[i] = (int *) malloc(N*sizeof(int));
                        }

                        for( i=0; i<N; i++){
                                        for( int j=0; j<N; j++){
                                                        a[i][j] = 0;
                                                        b[i][j] = 2;
                                                        c[i][j] = 2;
                                        }
                        }

                        double comienzo, fin, total;
                        comienzo = omp_get_wtime();



                        #pragma omp parallel for private(k,j)
                        for(  i=0; i<N; i++){
                                        for(  j=0; j<N; j++){
                                                        for (  k=0; k<N; k++){
                                                                        a[i][j] +=
b[i][k] * c[k][j];
                                                        }
                                        }
                        }

                        fin = omp_get_wtime();

                        total = fin - comienzo;



                        printf("Tiempo: %11.9F\t Primera: %d\t Ultima: %d\
n",total,a[0][0], a[N-1][N-1]);

                        for(int i=0; i<N; i++){
                                        free(a[i]);
                                        free(b[i]);
                                        free(c[i]);
                        }

                        free(a);
                        free(b);
                        free(c);

                        return 0;
                }
```

CAPTURAS DE PANTALLA:

```
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$gcc-7 -O2 -fopenmp pmm-OpenMP.c -o pmm-OpenMP
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-OpenMP 10
Tiempo: 0.000518000      Primera: 40     Ultima: 40
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-OpenMP 20
Tiempo: 0.000297000      Primera: 80     Ultima: 80
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-OpenMP 30
Tiempo: 0.000461000      Primera: 120    Ultima: 120
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-OpenMP 100
Tiempo: 0.000844000      Primera: 400    Ultima: 400
[Sergio Aguilera Ramirez usuario@MacBook-Pro-de-Sergio-Aguilera:~/Desktop/BP4_AguileraRamirezSergio/codigos] 2018-05-08 martes
$./pmm-OpenMP 1000
Tiempo: 5.180231000      Primera: 4000   Ultima: 4000
```

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.

ESTUDIO DE ESCALABILIDAD EN atcgrid:

SCRIPT: pmm-OpenMP_atcgrid.sh

```
    #!/bin/bash
60. #Se asigna al trabajo el nombre pmtv-OpemMp_atcgrid
61. #PBS -N pmtv-OpemMp_atcgrid
62. #Se asigna al trabajo la cola ac
63. #PBS -q ac
64. #Se imprime información del trabajo usando variables de entorno de PBS
65. echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
66. echo "Id. del trabajo: $PBS_JOBID"
67. echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
68. echo "Nodo que ejecuta qsub: $PBS_O_HOST"
69. echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
70. echo "Cola: $PBS_QUEUE"
71. echo "Nodos asignados al trabajo:"
72. #cat $PBS_NODEFILE
73.
74.
75. export OMP_SCHEDULE="static"
76. echo "static y chunk por defecto"
77. $PBS_O_WORKDIR/pmtv-OpenMP 15360
78.
79. export OMP_SCHEDULE="static,1"
80. echo "static y chunk 1"
81. $PBS_O_WORKDIR/pmtv-OpenMP 15360
82.
83. export OMP_SCHEDULE="static,64"
84. echo "static y chunk 64"
85. $PBS_O_WORKDIR/pmtv-OpenMP 15360
86.
87. export OMP_SCHEDULE="dynamic"
88. echo "dynamic y chunk por defecto"
89. $PBS_O_WORKDIR/pmtv-OpenMP 15360
90.
```

```
export OMP_SCHEDULE="dynamic,1"
echo "dynamic y chunk 1"
$PBS_O_WORKDIR/pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic,64"
echo "dynamic y chunk 64"
$PBS_O_WORKDIR/pmtv-OpenMP 15360

export OMP_SCHEDULE="guided"
      echo "guided y chunk por defecto"
      $PBS_O_WORKDIR/pmtv-OpenMP 15360

      export OMP_SCHEDULE="guided,1"
      echo "guided y chunk 1"
      $PBS_O_WORKDIR/pmtv-OpenMP 15360

      export OMP_SCHEDULE="guided,64"
      echo "guided y chunk 64"
      $PBS_O_WORKDIR/pmtv-OpenMP 15360
```

ESTUDIO DE ESCALABILIDAD EN PCLOCAL:

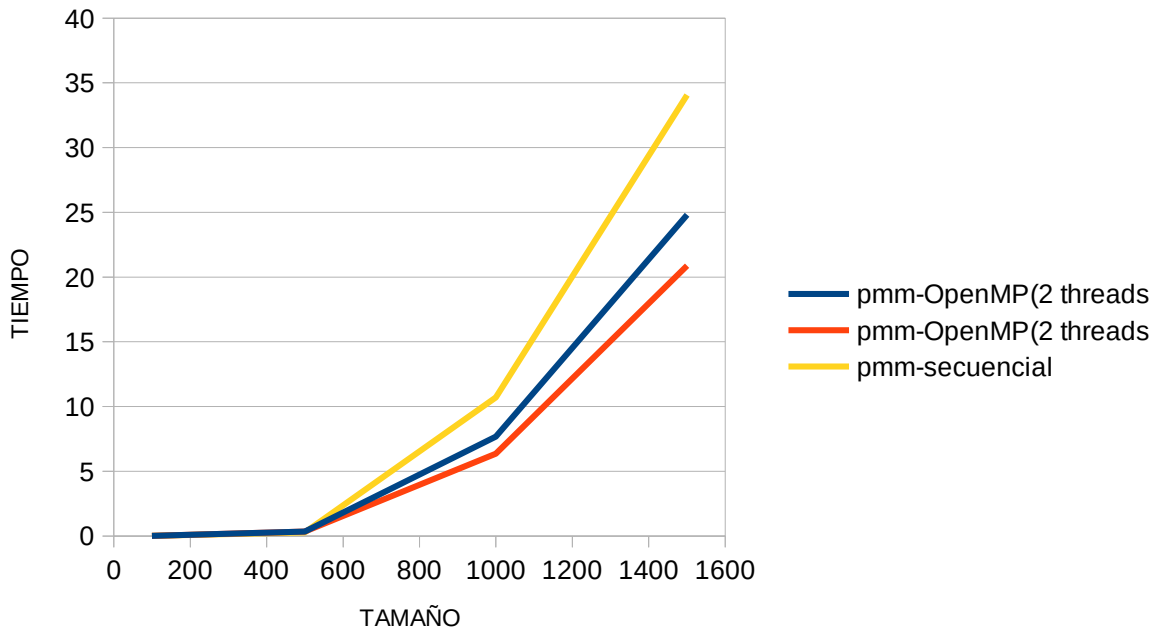
SCRIPT: pmm-OpenMP_pclocal.sh

```
   #!/bin/bash

      export OMP_DYNAMIC=FALSE

      echo "\nTiempo secuencial con 100"
      ./pmm-secuencial 100

      echo "Tiempo secuencial con 500"
      ./pmm-secuencial 500

      echo "\nTiempo secuencial con 1000"
      ./pmm-secuencial 1000

      echo "Tiempo secuencial con 1500"
      ./pmm-secuencial 1500









      export OMP_NUM_THREADS=2
      echo "\nTiempo con dos threads con 100"
      ./pmm-OpenMP 100


      echo "Tiempo con dos threads con 500"
      ./pmm-OpenMP 500


      echo "Tiempo con dos threads con 1000"
      ./pmm-OpenMP 1000


      echo "Tiempo con dos threads con 1500"
      ./pmm-OpenMP 1500
```

```
147.
148.      export OMP_NUM_THREADS=4
149.      echo "\nTiempo con cuatro threads con 100"
150.      ./pmm-OpenMP 100
151.
152.
153.      echo "Tiempo con cuatro threads con 500"
154.      ./pmm-OpenMP 500
155.
156.      echo "Tiempo con cuatro threads con 1000"
157.      ./pmm-OpenMP 1000
158.
159.
160.      echo "Tiempo con cuatro threads con 1500"
161.      ./pmm-OpenMP 1500
162.
```

PCLOCAL:

TAMAÑO	pmm-OpenMP(2 threads	pmm-OpenMP(2 threads	pmm-secuencial
100	0,001299000	0,001035000	0,000745000
500	0,333274000	0,349641000	0,276665000
1000	7,680000000	6,360792000	10,692037000
1500	24,810609000	20,865036000	34,054651000



ATCGRID:

TAMAÑO	pmm-OpenMP(2 threads	pmm-OpenMP(2 threads	pmm-secuencial
100	0,001280194	0,000784686	0,002081807
500	0,181897490	0,093967012	0,349973386
1000	4,925504415	2,623921874	9,722031564
1500	17,140168763	8,803202207	34,885217722