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

Estudiante (nombre y apellidos):
Grupo de prácticas:
Fecha de entrega:
Fecha evaluación en clase:

Antes de comenzar a realizar el trabajo de este cuaderno consultar el fichero con los normas de prácticas que se encuentra en SWAD

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`

```
int i, n=20,x=4, tid;
int a[n], suma=0, sumalocal;

if (argc < 3)
{
    fprintf(stderr,"[ERROR] Falta el número de iteraciones\n");
    exit(-1);
}
x = atoi(argv[2]);
n = atoi(argv[1]);
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)
```

RESPUESTA:

Como veremos en la captura de la ejecución, observamos que en if-clause solo realiza la ejecución de forma paralela si cumple la condición de if. En if-clauseModificado, sucede lo mismo, pero lo que conseguimos en caso de cumplir la condición del if, es modificar el número de hebras que van a realizar la ejecución paralela. En caso contrario, en los 2 programas, ejecuta todo ese código la thread master (la 0) .

CAPTURAS DE PANTALLA:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer1] 2020-04-27 lunes
$gcc -O2 -fopenmp -o if-clause if-clause.c
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer1] 2020-04-27 lunes
$gcc -O2 -fopenmp -o if-clauseModificado if-clauseModificado.c
```

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer1] 2020-04-27 lunes
$./if-clause 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
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer1] 2020-04-27 lunes
$./if-clause 5
 thread 4 suma de a[4]=4 sumalocal=4
 thread 0 suma de a[0]=0 sumalocal=0
 thread 1 suma de a[1]=1 sumalocal=1
 thread 3 suma de a[3]=3 sumalocal=3
 thread 2 suma de a[2]=2 sumalocal=2
thread master=0 imprime suma=10
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer1] 2020-04-27 lunes
$./if-clauseModificado 5 5
 thread 0 suma de a[0]=0 sumalocal=0
 thread 2 suma de a[2]=2 sumalocal=2
 thread 3 suma de a[3]=3 sumalocal=3
 thread 1 suma de a[1]=1 sumalocal=1
 thread 4 suma de a[4]=4 sumalocal=4
thread master=0 imprime suma=10
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer1] 2020-04-27 lunes
$./if-clauseModificado 4 5
 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
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer1] 2020-04-27 lunes
$./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 2 suma de a[3]=3 sumalocal=3
 thread 1 suma de a[2]=2 sumalocal=2
thread master=0 imprime suma=10
```

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

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

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

RESPUESTA:

Tenemos diferencias, entre static y dynamic con guided, ya que las dos primeras, el valor de chunk establece el tamaño de los bloques que asignan, mientras que en guided, el valor de chunk establece el mínimo tamaño del bloque, por los que si es 2 el valor, guided podría asignar bloques de tamaño >= 2 .

Otra diferencia la tenemos entre dynamic y guided con static, esto es porque las dos primeras asignan los bloques en tiempo de ejecución mientras que static no, por lo que los identificadores de las hebras van en orden en static(0,1,2,3,..) mientras que en dynamic y guided cambia dependiendo de que hebra esté disponible antes.

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`

```
int dyn_var = omp_get_dynamic();
int nthreads_var = omp_get_max_threads();
int thread_limit_var = omp_get_thread_limit();
int modifier;
omp_sched_t kind;
omp_get_schedule(&kind,&modifier);
    printf("kind:%d\n",kind);

#pragma omp parallel
{
#pragma omp 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);
}
    #pragma omp single
    {
    printf("\nDentro del parallel\nthread %d imprime dyn-var %d\n",omp_get_thread_num(),dyn_var);
    printf("thread %d imprime nthreads_var %d\n",omp_get_thread_num(),nthreads_var);
    printf("thread %d imprime thread-limit-var %d\n",omp_get_thread_num(),thread_limit_var);
    printf("thread %d imprime run-sched-var:\n",omp_get_thread_num());
    if(omp_sched_dynamic == kind) printf("Tipo dynamic, modificador %d\n\n",modifier);
    else if(omp_sched_guided == kind) printf("Tipo guided, modificador %d\n\n",modifier);
    else printf("Tipo static, modificador %d\n\n",modifier);
    }
}

printf("Fuera de 'parallel for' suma=%d\n",suma);
printf("\ndyn-var %d\n",dyn_var);
printf("nthreads_var %d\n",nthreads_var);
printf("thread-limit-var %d\n",thread_limit_var);
printf("run-sched-var:%d\n",kind);
    if(omp_sched_dynamic == kind) printf("Tipo dynamic, modificador %d\n\n",modifier);
    else if(omp_sched_guided == kind) printf("Tipo guided, modificador %d\n\n",modifier);
    else printf("Tipo static, modificador %d\n\n",modifier);
```

CAPTURAS DE PANTALLA:

EJECUCIÓN NORMAL:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$gcc -O2 -fopenmp -o scheduled-clauseModificado scheduled-clauseModificado.c
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$./scheduled-clauseModificado 7 2
 thread 0 suma a[2]=2 suma=2
 thread 0 suma a[3]=3 suma=5
 thread 1 suma a[0]=0 suma=0
 thread 1 suma a[1]=1 suma=1
 thread 2 suma a[4]=4 suma=4
 thread 2 suma a[5]=5 suma=9
 thread 3 suma a[6]=6 suma=6

Dentro del parallel
thread 0 imprime dyn-var 0
thread 0 imprime nthreads_var 4
thread 0 imprime thread-limit-var 2147483647
thread 0 imprime run-sched-var:
Tipo dynamic, modificador 1

Fuera de 'parallel for' suma=6

dyn-var 0
nthreads_var 4
thread-limit-var 2147483647
run-sched-var:
Tipo dynamic, modificador1
```

MODIFICANDO DYN-VAR:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$export OMP_DYNAMIC=FALSE
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$./scheduled-clauseModificado 7 2
 thread 0 suma a[0]=0 suma=0
 thread 0 suma a[1]=1 suma=1
 thread 2 suma a[2]=2 suma=2
 thread 2 suma a[3]=3 suma=5
 thread 3 suma a[4]=4 suma=4
 thread 3 suma a[5]=5 suma=9
 thread 1 suma a[6]=6 suma=6

Dentro del parallel
thread 3 imprime dyn-var 0
thread 3 imprime nthreads_var 4
thread 3 imprime thread-limit-var 2147483647
thread 3 imprime run-sched-var:
Tipo dynamic, modificador 1

Fuera de 'parallel for' suma=6

dyn-var 0
nthreads_var 4
thread-limit-var 2147483647
run-sched-var:
Tipo dynamic, modificador 1

[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$export OMP_DYNAMIC=TRUE
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$./scheduled-clauseModificado 7 2
 thread 1 suma a[2]=2 suma=2
 thread 1 suma a[3]=3 suma=5
 thread 2 suma a[4]=4 suma=4
 thread 2 suma a[5]=5 suma=9
 thread 1 suma a[6]=6 suma=11
 thread 0 suma a[0]=0 suma=0
 thread 0 suma a[1]=1 suma=1

Dentro del parallel
thread 2 imprime dyn-var 1
thread 2 imprime nthreads_var 4
thread 2 imprime thread-limit-var 2147483647
thread 2 imprime run-sched-var:
Tipo dynamic, modificador 1

Fuera de 'parallel for' suma=11

dyn-var 1
nthreads_var 4
thread-limit-var 2147483647
run-sched-var:
Tipo dynamic, modificador 1
```

MODIFICANDO NTHREADS-VAR:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$export OMP_NUM_THREADS=4
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$./scheduled-clauseModificado 7 2
 thread 0 suma a[2]=2 suma=2
 thread 0 suma a[3]=3 suma=5
 thread 1 suma a[6]=6 suma=6
 thread 2 suma a[0]=0 suma=0
 thread 2 suma a[1]=1 suma=1
 thread 3 suma a[4]=4 suma=4
 thread 3 suma a[5]=5 suma=9

Dentro del parallel
thread 0 imprime dyn-var 0
thread 0 imprime nthreads_var 4
thread 0 imprime thread-limit-var 2147483647
thread 0 imprime run-sched-var:
Tipo dynamic, modificador 1

Fuera de 'parallel for' suma=6

dyn-var 0
nthreads_var 4
thread-limit-var 2147483647
run-sched-var:
Tipo dynamic, modificador 1

[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$export OMP_NUM_THREADS=8
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$./scheduled-clauseModificado 7 2
 thread 0 suma a[6]=6 suma=6
 thread 2 suma a[4]=4 suma=4
 thread 2 suma a[5]=5 suma=9
 thread 4 suma a[0]=0 suma=0
 thread 4 suma a[1]=1 suma=1
 thread 6 suma a[2]=2 suma=2
 thread 6 suma a[3]=3 suma=5

Dentro del parallel
thread 0 imprime dyn-var 0
thread 0 imprime nthreads_var 8
thread 0 imprime thread-limit-var 2147483647
thread 0 imprime run-sched-var:
Tipo dynamic, modificador 1

Fuera de 'parallel for' suma=6

dyn-var 0
nthreads_var 8
thread-limit-var 2147483647
run-sched-var:
Tipo dynamic, modificador 1
```

MODIFICANDO THREAD-LIMIT-VAR:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$export OMP_NUM_THREADS=4
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$export OMP_THREAD_LIMIT=2
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$./scheduled-clauseModificado 7 2
 thread 1 suma a[2]=2 suma=2
 thread 1 suma a[3]=3 suma=5
 thread 1 suma a[4]=4 suma=9
 thread 1 suma a[5]=5 suma=14
 thread 1 suma a[6]=6 suma=20
 thread 0 suma a[0]=0 suma=0
 thread 0 suma a[1]=1 suma=1

Dentro del parallel
thread 1 imprime dyn-var 0
thread 1 imprime nthreads_var 4
thread 1 imprime thread-limit-var 2
thread 1 imprime run-sched-var:
Tipo dynamic, modificador 1

Fuera de 'parallel for' suma=20

dyn-var 0
nthreads_var 4
thread-limit-var 2
run-sched-var:
Tipo dynamic, modificador 1
```

MODIFICANDO RUN-SCHED-VAR:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$export OMP_SCHEDULE="dynamic,4"
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$./scheduled-clauseModificado 7 2
 thread 2 suma a[0]=0 suma=0
 thread 2 suma a[1]=1 suma=1
 thread 0 suma a[2]=2 suma=2
 thread 0 suma a[3]=3 suma=5
 thread 3 suma a[6]=6 suma=6
 thread 1 suma a[4]=4 suma=4
 thread 1 suma a[5]=5 suma=9

Dentro del parallel
thread 0 imprime dyn-var 0
thread 0 imprime nthreads_var 4
thread 0 imprime thread-limit-var 2147483647
thread 0 imprime run-sched-var:
Tipo dynamic, modificador 4

Fuera de 'parallel for' suma=6

dyn-var 0
nthreads_var 4
thread-limit-var 2147483647
run-sched-var:
Tipo dynamic, modificador 4

[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$export OMP_SCHEDULE="static,3"
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer3] 2020-04-27 lunes
$./scheduled-clauseModificado 7 2
 thread 0 suma a[6]=6 suma=6
 thread 1 suma a[0]=0 suma=0
 thread 1 suma a[1]=1 suma=1
 thread 3 suma a[4]=4 suma=4
 thread 3 suma a[5]=5 suma=9
 thread 2 suma a[2]=2 suma=2
 thread 2 suma a[3]=3 suma=5

Dentro del parallel
thread 3 imprime dyn-var 0
thread 3 imprime nthreads_var 4
thread 3 imprime thread-limit-var 2147483647
thread 3 imprime run-sched-var:
Tipo static, modificador 3

Fuera de 'parallel for' suma=6

dyn-var 0
nthreads_var 4
thread-limit-var 2147483647
run-sched-var:
Tipo static, modificador 3
```

Sobre run-sched-var hay que comentar que esta hecha para usar schedule(runtime), por lo que como se ve en las ejecuciones, aunque modifiquemos OMP_SCHEDULE cambiando el modifier, no modifica el chunk que usa la región paralela, ya que esa esta introducida por argumento, aun así si modifica el valor dentro de run-sched-var como se puede comprobar en las imágenes

RESPUESTA:

Se puede observar claramente que no influye para nada que la impresión de las variables sea dentro o fuera de la región del parallel.

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`

```
#pragma omp parallel
{
#pragma omp 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);
}
    #pragma omp single
    {
    printf("\nDentro del parallel\n");
    printf("\nomp_get_num_threads = %d\n", omp_get_num_threads());
    printf("\nomp_get_num_procs = %d\n", omp_get_num_procs());
    printf("\nomp_in_parallel = %d\n\n", omp_in_parallel());
    }
}

printf("Fuera de 'parallel for' suma=%d\n",suma);
printf("\nomp_get_num_threads = %d\n", omp_get_num_threads());
printf("\nomp_get_num_procs = %d\n", omp_get_num_procs());
printf("\nomp_in_parallel = %d\n\n", omp_in_parallel());
```

CAPTURAS DE PANTALLA:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer4] 2020-04-27 lunes
$gcc -O2 -fopenmp -o scheduled-clauseModificado4 scheduled-clauseModificado4.c
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer4] 2020-04-27 lunes
$./scheduled-clauseModificado4 7 2
 thread 0 suma a[2]=2 suma=2
 thread 0 suma a[3]=3 suma=5
 thread 3 suma a[6]=6 suma=6
 thread 2 suma a[0]=0 suma=0
 thread 2 suma a[1]=1 suma=1
 thread 1 suma a[4]=4 suma=4
 thread 1 suma a[5]=5 suma=9

Dentro del parallel

omp_get_num_threads = 4

omp_get_num_procs = 8

omp_in_parallel = 1

Fuera de 'parallel for' suma=6

omp_get_num_threads = 1

omp_get_num_procs = 8

omp_in_parallel = 0
```

RESPUESTA:

En omp_get_num_threads y omp_in_parallel se obtienen diferentes valores si se está dentro o no del parallel.

Cuando están dentro de la región, omp_get_num_threads muestra el valor con el número de threads que están ejecutando esa región y omp_in_parallel muestra 1, indicandonos que eso es una región parallel.

Cuando están fuera, la primera muestra 1, indicando que 1 hebra sola está ejecutando (hay 1 porque es secuencial y es la master) y omp_in_parallel nos muestra 0, diciendonos que no está en una región parallel.

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`

```
int modifier;
omp_sched_t kind;
omp_get_schedule(&kind,&modifier);


#pragma omp parallel
{
#pragma omp 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);
}
    #pragma omp single
    {
    printf("\nValores sin modificar:\ndyn-var %d\nnthreads_var %d\nrun-sched-var:\n",omp_get_dynamic(),omp_get_max_threads());
    if(omp_sched_dynamic == kind) printf("Tipo dynamic, modificador %d\n\n",modifier);
    else if(omp_sched_guided == kind) printf("Tipo guided, modificador %d\n\n",modifier);
    else printf("Tipo static, modificador %d\n\n",modifier);

    omp_set_dynamic(1);
    omp_set_num_threads(8);
    omp_set_schedule(omp_sched_guided,5);
    omp_get_schedule(&kind,&modifier);

    printf("\nValores modificados:\ndyn-var %d\nnthreads_var %d\nrun-sched-var:\n",omp_get_dynamic(),omp_get_max_threads());
    if(omp_sched_dynamic == kind) printf("Tipo dynamic, modificador %d\n\n",modifier);
    else if(omp_sched_guided == kind) printf("Tipo guided, modificador %d\n\n",modifier);
    else printf("Tipo static, modificador %d\n\n",modifier);

    }
}

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

CAPTURAS DE PANTALLA:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer5] 2020-04-27 lunes
$gcc -O2 -fopenmp -o scheduled-clauseModificado5 scheduled-clauseModificado5.c
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer5] 2020-04-27 lunes
$./scheduled-clauseModificado5 7 2
 thread 3 suma a[4]=4 suma=4
 thread 3 suma a[5]=5 suma=9
 thread 2 suma a[2]=2 suma=2
 thread 2 suma a[3]=3 suma=5
 thread 0 suma a[6]=6 suma=6
 thread 1 suma a[0]=0 suma=0
 thread 1 suma a[1]=1 suma=1

Valores sin modificar:
dyn-var 0
nthreads_var 4
run-sched-var:
Tipo dynamic, modificador 1


Valores modificados:
dyn-var 1
nthreads_var 8
run-sched-var:
Tipo guided, modificador 5

Fuera de 'parallel for' suma=6
```

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`

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



int main(int argc, char const *argv[]){
        if(argc != 2){                                  //Esto es para aceptar el argumento N(numero de filas/columnas)
        printf("Error de argumentos %s", argv[0]);
        return(EXIT_FAILURE);
    }

    struct timespec cgt1, cgt2;                 //Definimos estructuras para medir el tiempo
    double ncgt;

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


        double **matriz, *vector, *vr;                  //Inicializacion dinámica de la matriz y los vectores
        matriz = (double**) malloc(N * sizeof(double*));
        for(int i = 0; i < N; ++i)
            matriz[i] = (double*) malloc(N * sizeof(double));

        vector = (double*) malloc(N * sizeof(double));
        vr = (double*) malloc(N * sizeof(double));



    for(int i = 0; i < N; ++i)
    {                                   //Se rellenan el vector y la matriz con valores
        for(int j = 0; j < N; ++j)
        {
                if(j >= i)
                        matriz[i][j] = 6.12345;
                else
                        matriz[i][j] = 0;
        }
        vector[i] = 2.321;
        vr[i] = 0;
    }

        clock_gettime(CLOCK_REALTIME, &cgt1);           //Cojemos este instante de tiempo

        for(int i = 0; i < N; i++)
        {                       //Calculamos el vector resultante
                for(int j = i; j < N; j++)
                        vr[i] += matriz[i][j] * vector[i];
        }

        clock_gettime(CLOCK_REALTIME, &cgt2);           //Cojemos el instante de tiempo de justo cuando termina el algoritmo
        ncgt = (double) (cgt2.tv_sec - cgt1.tv_sec) + (double) (cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9); //Obtenemos el tiempo de ejecucion
```

```
	printf("Matriz trinagular superior: \n");
	for(int i = 0; i < N; ++i){
		for(int j = 0; j < N; ++j)
			printf("%f ",matriz[i][j]);
		printf("\n");
	}

	printf("Vector que va a multiplicar a la matriz: \n");
	for(int i = 0; i < N; ++i)
		printf("%f ",vector[i]);
	printf("\n");


	printf("Vector resultante: \n");
	for(int i = 0; i < N; ++i)
		printf("%f ",vr[i]);
	printf("\n");



	printf("Tiempo(seg.): %11.9f\t / Tamaño vectores y matriz: %u\n", ncgt, N);
	if(N < 15)                                                    //Con este if/else mostramos el vector con valores pequeños
		for(int i = 0; i < N; i++){
			printf("Vector Resultante[%d] = %f  ", i, vr[i]);
			printf("\n");
		}
	else{
		printf("Vector Resultante[0] = %f  ", vr[0]);
		printf("Vector Resultante[%d] = %f  ", N - 1, vr[N - 1]);
		printf("\n");
	}


	for(int i = 0; i < N; i++)                    //Liberación del espacio cuando usamos matriz y vectores dinamicos
		free(matriz[i]);

	free(matriz); free(vector); free(vr);


return 0;
```

CAPTURAS DE PANTALLA:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer6] 2020-04-30 jueves
$gcc -O2 -fopenmp -o pmtv-secuencial pmtv-secuencial.c
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer6] 2020-04-30 jueves
$./pmtv-secuencial 5
Matriz trinagular superior:
6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 6.123450
Vector que va a multiplicar a la matriz:
2.321000 2.321000 2.321000 2.321000 2.321000
Vector resultante:
71.062637 56.850110 42.637582 28.425055 14.212527
Tiempo(seg.): 0.000000129        / Tamaño vectores y matriz: 5
Vector Resultante[0] = 71.062637
Vector Resultante[1] = 56.850110
Vector Resultante[2] = 42.637582
Vector Resultante[3] = 28.425055
Vector Resultante[4] = 14.212527
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer6] 2020-04-30 jueves
$./pmtv-secuencial 16
Matriz trinagular superior:
6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450 6.123450
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 6.123450
Vector que va a multiplicar a la matriz:
2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000 2.321000
Vector resultante:
227.400439 213.187912 198.975384 184.762857 170.550329 156.337802 142.125274 127.912747 113.700220 99.487692 85.275165 71.062637 56.850110 42.637582 28.425055 14.212527
Tiempo(seg.): 0.000000256        / Tamaño vectores y matriz: 16
Vector Resultante[0] = 227.400439  Vector Resultante[15] = 14.212527
```

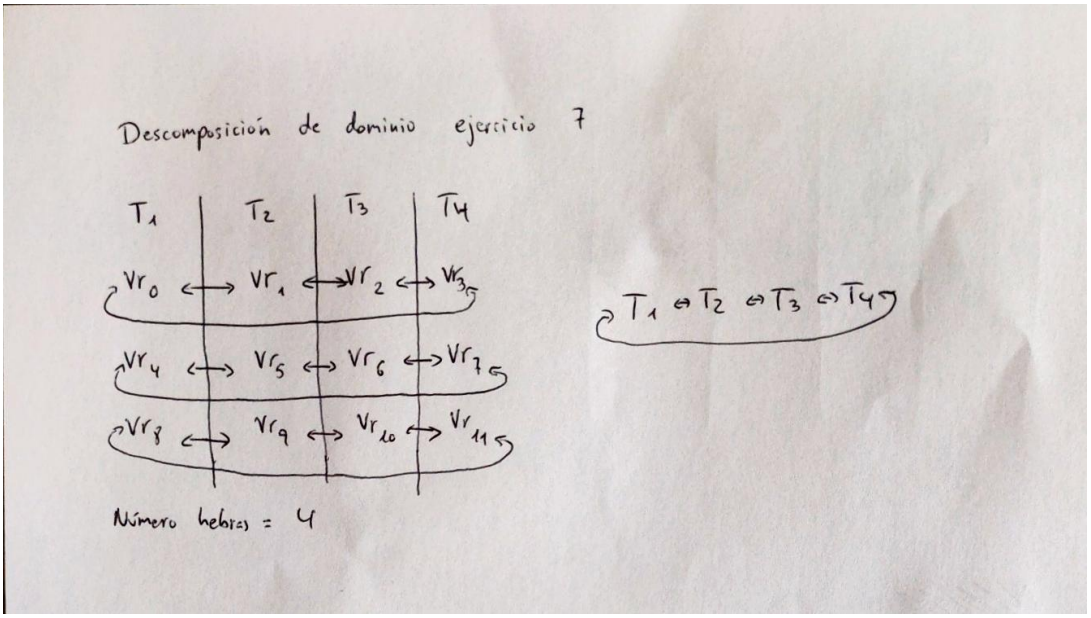
7. Implementar en paralelo la multiplicación de una matriz triangular por un vector a partir del código secuencial realizado para el ejercicio anterior utilizando la directiva `for` de OpenMP. El código debe repartir entre los threads las iteraciones del bucle que recorre las filas. Dibujar en el cuaderno 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?

CAPTURA CÓDIGO FUENTE: `pmtv-OpenMP.c`

```
ti = omp_get_wtime();           //Cojemos este instante de tiempo

#pragma omp parallel for schedule(runtime)
for(int i = 0; i < N; i++)
{                       //Calculamos el vector resultante
        for(int j = i; j < N; j++)
                vr[i] += matriz[i][j] * vector[i];
}

tf = omp_get_wtime();           //Cojemos el instante de tiempo de justo cuando termina el algoritmo
tt = tf - ti; //Obtenemos el tiempo de ejecucion
```

DESCOMPOSICIÓN DE DOMINIO: Las thread calcula cada una ,un valor del vector resultante, recorriendo filas diferentes de la matriz



CAPTURAS DE PANTALLA:

```
[b2estudiante4@atcgrid ejer7]$ sbatch -p ac -n1 --cpus-per-task=12 --hint=nomultithread --exclusive script-pmtv.sh
Submitted batch job 46318
[b2estudiante4@atcgrid ejer7]$ cat slurm-46318.out
static defecto
Tiempo(seg.): 0.058980908        / Tamaño vectores y matriz: 15360
Vector Resultante[0] = 218304.421632  Vector Resultante[15359] = 14.212527
static 1
Tiempo(seg.): 0.064978741        / Tamaño vectores y matriz: 15360
Vector Resultante[0] = 218304.421632  Vector Resultante[15359] = 14.212527
static 64
Tiempo(seg.): 0.059894707        / Tamaño vectores y matriz: 15360
Vector Resultante[0] = 218304.421632  Vector Resultante[15359] = 14.212527
dynamic defecto
Tiempo(seg.): 0.064229503        / Tamaño vectores y matriz: 15360
Vector Resultante[0] = 218304.421632  Vector Resultante[15359] = 14.212527
dynamic 1
Tiempo(seg.): 0.060847849        / Tamaño vectores y matriz: 15360
Vector Resultante[0] = 218304.421632  Vector Resultante[15359] = 14.212527
dynamic 64
Tiempo(seg.): 0.056507260        / Tamaño vectores y matriz: 15360
Vector Resultante[0] = 218304.421632  Vector Resultante[15359] = 14.212527
guided defecto
Tiempo(seg.): 0.058330674        / Tamaño vectores y matriz: 15360
Vector Resultante[0] = 218304.421632  Vector Resultante[15359] = 14.212527
guided 1
Tiempo(seg.): 0.063545700        / Tamaño vectores y matriz: 15360
Vector Resultante[0] = 218304.421632  Vector Resultante[15359] = 14.212527
guided 64
Tiempo(seg.): 0.063062880        / Tamaño vectores y matriz: 15360
Vector Resultante[0] = 218304.421632  Vector Resultante[15359] = 14.212527
```

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid

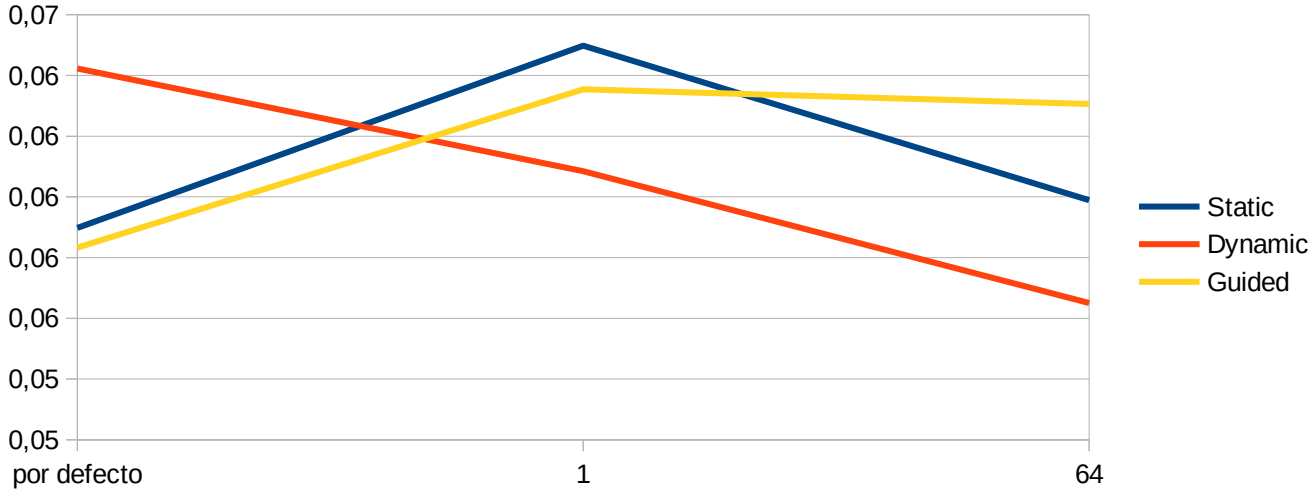
SCRIPT: pmvt-OpenMP_atcgrid.sh

```
#!/bin/bash

export OMP_DYNAMIC=FALSE
export OMP_NUM_THREADS=12

export OMP_SCHEDULE="static"
echo "static defecto"
srun ./pmtv-OpenMP 15360

export OMP_SCHEDULE="static,1"
echo "static 1"
srun ./pmtv-OpenMP 15360

export OMP_SCHEDULE="static,64"
echo "static 64"
srun ./pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic"
echo "dynamic defecto"
srun ./pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic,1"
echo "dynamic 1"
srun ./pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic,64"
echo "dynamic 64"
srun ./pmtv-OpenMP 15360

export OMP_SCHEDULE="guided"
echo "guided defecto"
srun ./pmtv-OpenMP 15360

export OMP_SCHEDULE="guided,1"
echo "guided 1"
srun ./pmtv-OpenMP 15360

export OMP_SCHEDULE="guided,64"
echo "guided 64"
srun ./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	0,058980908	0,064229503	0,058330674
1	0,064978741	0,060847849	0,063545700
64	0,059894707	0,056507260	0,063062880

Chunk	Static	Dynamic	Guided
por defecto	0,074869808	0,061633594	0,061438240
1	0,057358220	0,062831197	0,075869579
64	0,059947450	0,056492981	0,059907883



RESPUESTA:

A) Static tiene por defecto 0 y dynamic y guided tienen por defecto 1. He usado la función omp_get_schedule(&kind,&mod) donde guarda en mod el tamaño de chunk que usa.

B) EL numero de chunk que establezacamos * el numero de filas/columnas

C) Static y dynamic realizaran 64 o 1 operaciones a la vez. En guided dependiendo de como asigne los chunks, no sera igual, ya que el tamaño de chunk es el minimo, luego puede realizar 3 operaciones a la vez o 65.

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`

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



int main(int argc, char const *argv[]){
        if(argc != 2){                                  //Esto es para aceptar el argumento N(numero de filas/columnas)
        printf("Error de argumentos %s", argv[0]);
        return(EXIT_FAILURE);
    }

    struct timespec cgt1, cgt2;                 //Definimos estructuras para medir el tiempo
    double ncgt;

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


        double **A,**B,**C;                     //Inicializacion dinámica de las matrices
        A = (double**) malloc(N * sizeof(double*));
        B = (double**) malloc(N * sizeof(double*));
        C = (double**) malloc(N * sizeof(double*));
        for(int i = 0; i < N; ++i){
            A[i] = (double*) malloc(N * sizeof(double));
            B[i] = (double*) malloc(N * sizeof(double));
            C[i] = (double*) malloc(N * sizeof(double));
        }

    for(int i = 0; i < N; ++i)
    {
                                    //Se rellenan el vector y la matriz con valores
        for(int j = 0; j < N; ++j)
        {
                        B[i][j] = 3.12345 + i;
                        C[i][j] = 2.98765 + j;
                        A[i][j] = 0;
        }
    }

        clock_gettime(CLOCK_REALTIME, &cgt1);           //Cojemos este instante de tiempo

        for(int i = 0; i < N; i++){                                             //Calculamos la matriz resultante
                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);           //Cojemos el instante de tiempo de justo cuando termina el algoritmo
        ncgt = (double) (cgt2.tv_sec - cgt1.tv_sec) + (double) (cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9); //Obtenemos el tiempo de ejecucion
```

```
	printf("Matriz B: \n");
	for(int i = 0; i < N; ++i){
		for(int j = 0; j < N; ++j)
			printf("%f ",B[i][j]);
		printf("\n");
	}

	printf("Matriz C: \n");
	for(int i = 0; i < N; ++i){
		for(int j = 0; j < N; ++j)
			printf("%f ",C[i][j]);
		printf("\n");
	}


	printf("Matriz Resultante: \n");
	for(int i = 0; i < N; ++i){
		for(int j = 0; j < N; ++j)
			printf("%f ",A[i][j]);
		printf("\n");
	}



	printf("Tiempo(seg.): %11.9f\t / Tamaño matrices: %u\n", ncgt, N);



	printf("Matriz Resultante[0] = %f  ", A[0][0]);
	printf("Matriz Resultante[%d] = %f  ", N - 1, A[N - 1][N - 1]);
	printf("\n");



	for(int i = 0; i < N; i++){                    //Liberación del espacio cuando usamos matrices dinamicas
		free(A[i]);
		free(B[i]);
		free(C[i]);
	}

	free(A); free(B); free(C);


return 0;
```

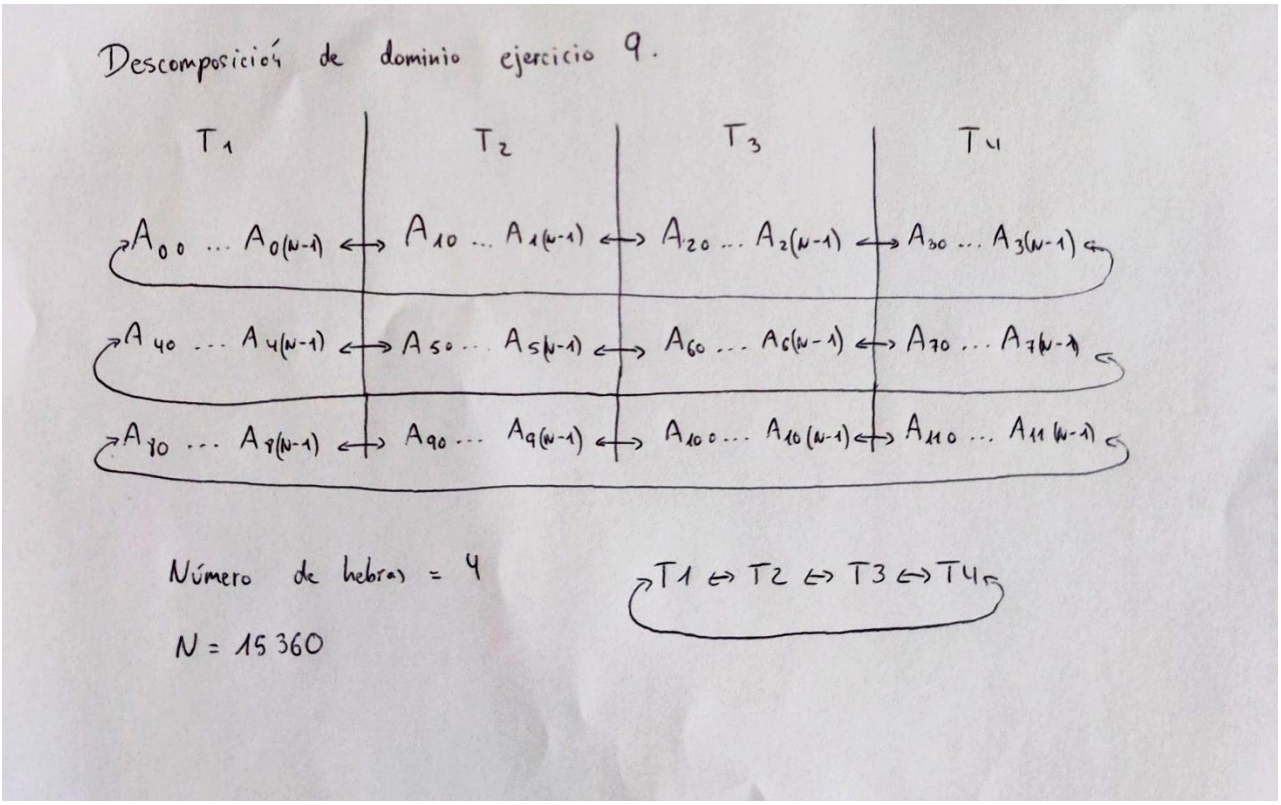
CAPTURAS DE PANTALLA:

```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer8] 2020-04-30 jueves
$gcc -o pmm-secuencial pmm-secuencial.c
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer8] 2020-04-30 jueves
$./pmm-secuencial 5
Matriz B:
3.123450 3.123450 3.123450 3.123450 3.123450
4.123450 4.123450 4.123450 4.123450 4.123450
5.123450 5.123450 5.123450 5.123450 5.123450
6.123450 6.123450 6.123450 6.123450 6.123450
7.123450 7.123450 7.123450 7.123450 7.123450
Matriz C:
2.987650 3.987650 4.987650 5.987650 6.987650
2.987650 3.987650 4.987650 5.987650 6.987650
2.987650 3.987650 4.987650 5.987650 6.987650
2.987650 3.987650 4.987650 5.987650 6.987650
2.987650 3.987650 4.987650 5.987650 6.987650
Matriz Resultante:
46.658877 62.276127 77.893377 93.510627 109.127877
61.597127 82.214377 102.831627 123.448877 144.066127
76.535377 102.152627 127.769877 153.387127 179.004377
91.473627 122.090877 152.708127 183.325377 213.942627
106.411877 142.029127 177.646377 213.263627 248.880877
Tiempo(seg.): 0.000000710         / Tamaño matrices: 5
Matriz Resultante[0] = 46.658877  Matriz Resultante[4] = 248.880877
```

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:

Cada hebra calcula una fila de la matriz resultante recorriendo las filas de una de las matrices, y las columnas de la otra.



CAPTURA CÓDIGO FUENTE: `pmm-OpenMP.c`

```
    int i,j,k;
    double ti,tf,tt;

#pragma omp parallel for schedule(runtime) private(j)
for(int i = 0; i < N; ++i)
{                                   //Se rellenan el vector y la matriz con valores
    for(int j = 0; j < N; ++j)
    {
                    B[i][j] = 3.12345 + i;
                    C[i][j] = 2.98765 + j;
                    A[i][j] = 0;
    }
}

    ti = omp_get_wtime();           //Cojemos este instante de tiempo

    #pragma omp parallel for schedule(runtime) private(j,k)
    for(int i = 0; i < N; i++){                                     //Calculamos la matriz resultante
            for(int j = 0; j < N; j++){
                    for(int k=0; k < N ; k++)
                            A[i][j] += B[i][k] * C[k][j];
            }
    }

    tf = omp_get_wtime();           //Cojemos el instante de tiempo de justo cuando termina el algoritmo
    tt = tf - ti;//Obtenemos el tiempo de ejecucion
```

CAPTURAS DE PANTALLA:

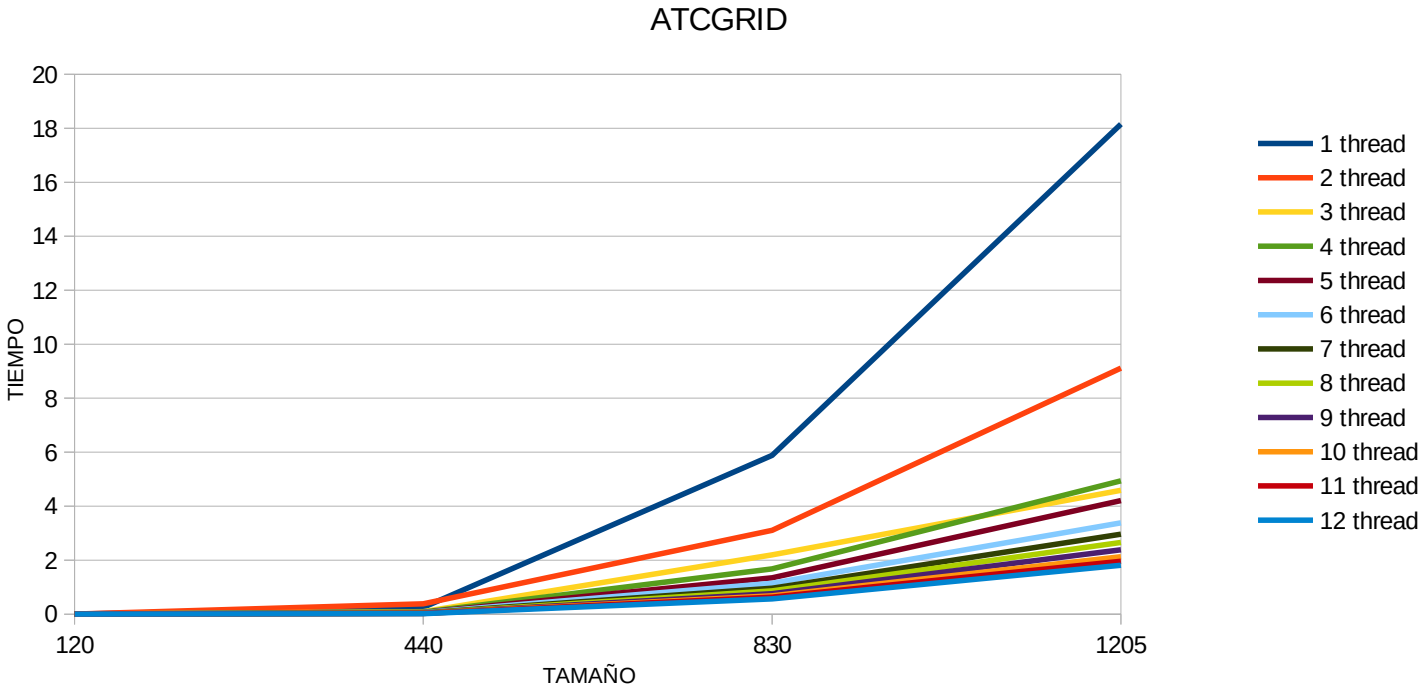
```
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer9] 2020-05-01 viernes
$gcc -O2 -fopenmp -o pmm-OpenMP pmm-OpenMP.c
[RaulCastroMoreno raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/AC/bp3/PC/ejer9] 2020-05-01 viernes
$./pmm-OpenMP 5
Matriz B:
3.123450 3.123450 3.123450 3.123450 3.123450
4.123450 4.123450 4.123450 4.123450 4.123450
5.123450 5.123450 5.123450 5.123450 5.123450
6.123450 6.123450 6.123450 6.123450 6.123450
7.123450 7.123450 7.123450 7.123450 7.123450
Matriz C:
2.987650 3.987650 4.987650 5.987650 6.987650
2.987650 3.987650 4.987650 5.987650 6.987650
2.987650 3.987650 4.987650 5.987650 6.987650
2.987650 3.987650 4.987650 5.987650 6.987650
2.987650 3.987650 4.987650 5.987650 6.987650
Matriz Resultante:
46.658877 62.276127 77.893377 93.510627 109.127877
61.597127 82.214377 102.831627 123.448877 144.066127
76.535377 102.152627 127.769877 153.387127 179.004377
91.473627 122.090877 152.708127 183.325377 213.942627
106.411877 142.029127 177.646377 213.263627 248.880877
Tiempo(seg.): 0.006591572        / Tamaño matrices: 5
Matriz Resultante[0] = 46.658877  Matriz Resultante[4] = 248.880877
```

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:

TAMAÑO	1 thread	2 thread	3 thread	4 thread	5 thread	6 thread	7 thread	8 thread	9 thread	10 thread	11 thread	12 thread
120	0,004736759	0,00248934	0,001311503	0,000942707	0,000854582	0,000649583	0,000814199	0,000524033	0,000645265	0,000437111	0,000398118	0,000368923
440	0,247123986	0,38296504	0,09152396	0,074499626	0,060606144	0,053907599	0,046736334	0,040320266	0,0333222	0,027028061	0,02605373	0,019439008
830	5,880426969	3,107342768	2,195836637	1,677204285	1,345133252	1,144857887	0,961022329	0,863937989	0,761235166	0,679989737	0,626850542	0,567520343
1205	18,154316157	9,11389586	4,588118177	4,940340113	4,207592349	3,385066852	2,963227935	2,65921472	2,384052098	2,124642592	1,960177928	1,80886168

Se observa perfectamente que a cuantas más cores/hebras usemos, mucho más rápido se hará la ejecución.



SCRIPT: pmm-OpenMP_atcgrid.sh

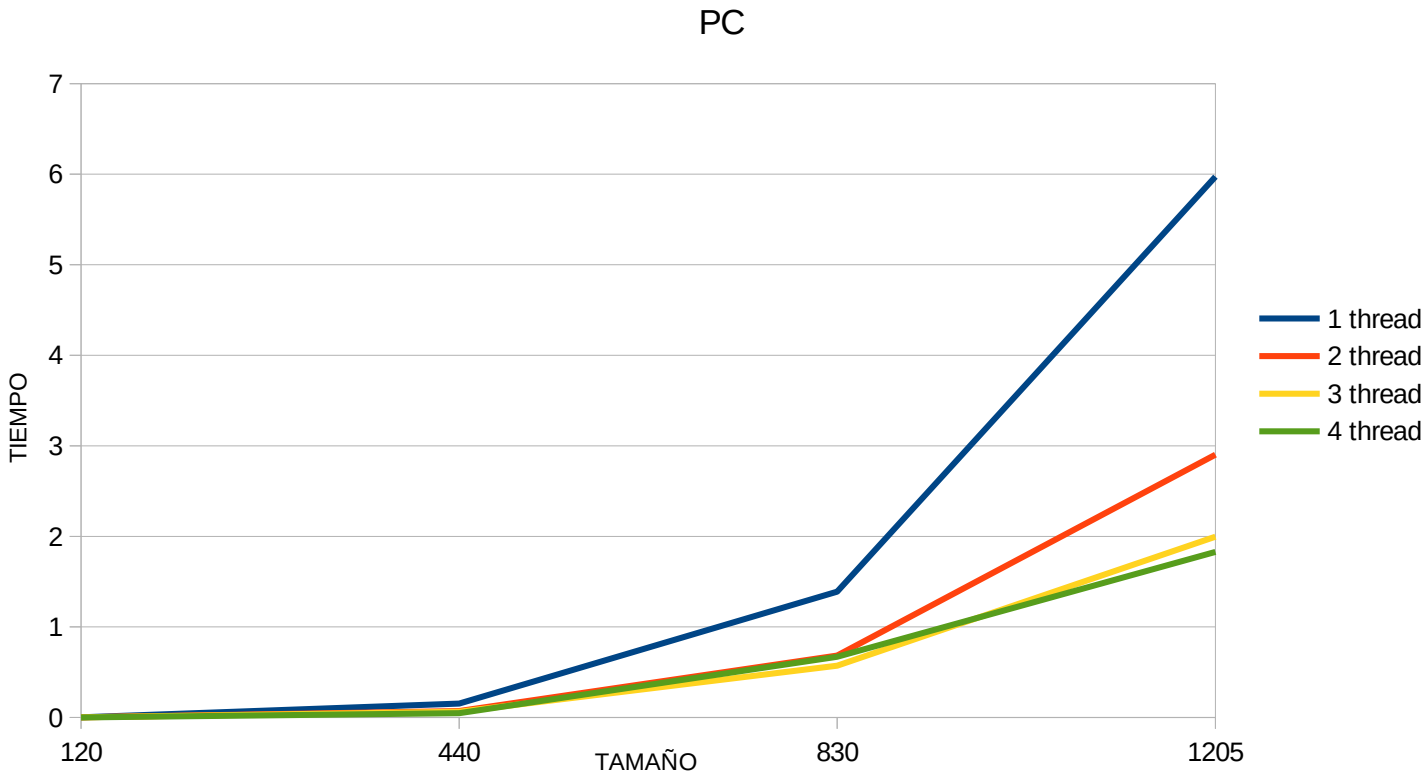
```
#!/bin/bash

export OMP_DYNAMIC=FALSE
export OMP_SCHEDULE="static"


export OMP_NUM_THREADS=1
echo "1 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=2
echo "2 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=3
echo "3 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=4
echo "4 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=5
echo "5 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=6
echo "6 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205
```

```
#!/bin/bash

export OMP_DYNAMIC=FALSE
export OMP_SCHEDULE="static"


export OMP_NUM_THREADS=7
echo "7 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=8
echo "8 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=9
echo "9 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=10
echo "10 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=11
echo "11 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205

export OMP_NUM_THREADS=12
echo "12 thread"
srun ./pmm-OpenMP 120
srun ./pmm-OpenMP 440
srun ./pmm-OpenMP 830
srun ./pmm-OpenMP 1205
```

He tenido que crear 2 scripts para atcgrid porque no me dejaba realizar más de 6 , por estar demasiado tiempo ejecutando.

ESTUDIO DE ESCALABILIDAD EN PCLOCAL:

TAMAÑO	1 thread	2 thread	3 thread	4 thread
120	0,002185287	0,001005411	0,00120743	0,000714785
440	0,153015694	0,070207024	0,063555089	0,047212427
830	1,388343581	0,683116753	0,571721595	0,670158669
1205	5,971616782	2,901850236	1,996355292	1,828835837



SCRIPT: pmm-OpenMP_pclocal.sh

```
#!/bin/bash

export OMP_DYNAMIC=FALSE
export OMP_SCHEDULE="static"


export OMP_NUM_THREADS=1
echo "1 thread"
./pmm-OpenMP 120
./pmm-OpenMP 440
./pmm-OpenMP 830
./pmm-OpenMP 1205

export OMP_NUM_THREADS=2
echo "2 thread"
./pmm-OpenMP 120
./pmm-OpenMP 440
./pmm-OpenMP 830
./pmm-OpenMP 1205

export OMP_NUM_THREADS=3
echo "3 thread"
./pmm-OpenMP 120
./pmm-OpenMP 440
./pmm-OpenMP 830
./pmm-OpenMP 1205

export OMP_NUM_THREADS=4
echo "4 thread"
./pmm-OpenMP 120
./pmm-OpenMP 440
./pmm-OpenMP 830
./pmm-OpenMP 1205
```