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): Valentino Lugli
Grupo de prácticas: C1
Fecha de entrega: 17/05/21
Fecha evaluación en clase: 18/05/21

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`

```
#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;

    if(argc < 3)
    {
        fprintf(stderr,"[ERROR]-Faltan componentes\n");
        exit(-1);
    }

    n = atoi(argv[1]); if (n>20) n=20;

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

    // Definiendo número de hilos
    int nThread = atoi(argv[2]); if (nThread < 1) nThread = 1;

    #pragma omp parallel if(n>4) default(none) \
    private(sumalocal,tid) shared(a,suma,n) num_threads(nThread)
    {
        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:

```
File  Edit  View  Bookmarks  Settings  Help
>>make
gcc -O2 -fopenmp -o if-clauseModificado_EXE if-clauseModificado.c
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer1] 2021-04-27 martes
>>./if-clauseModificado_EXE 10 2
 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 0 suma de a[4]=4 sumalocal=10
 thread 1 suma de a[5]=5 sumalocal=5
 thread 1 suma de a[6]=6 sumalocal=11
 thread 1 suma de a[7]=7 sumalocal=18
 thread 1 suma de a[8]=8 sumalocal=26
 thread 1 suma de a[9]=9 sumalocal=35
thread master=0 imprime suma=45
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer1] 2021-04-27 martes
>>./if-clauseModificado_EXE 6 3
 thread 1 suma de a[2]=2 sumalocal=2
 thread 1 suma de a[3]=3 sumalocal=5
 thread 0 suma de a[0]=0 sumalocal=0
 thread 0 suma de a[1]=1 sumalocal=1
 thread 2 suma de a[4]=4 sumalocal=4
 thread 2 suma de a[5]=5 sumalocal=9
thread master=0 imprime suma=15
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer1] 2021-04-27 martes
```

```
File  Edit  View  Bookmarks  Settings  Help
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer1] 2021-04-27 martes
>>./if-clauseModificado_EXE 6 3
 thread 1 suma de a[2]=2 sumalocal=2
 thread 1 suma de a[3]=3 sumalocal=5
 thread 0 suma de a[0]=0 sumalocal=0
 thread 0 suma de a[1]=1 sumalocal=1
 thread 2 suma de a[4]=4 sumalocal=4
 thread 2 suma de a[5]=5 sumalocal=9
thread master=0 imprime suma=15
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer1] 2021-04-27 martes
>>./if-clauseModificado_EXE 2 3
 thread 0 suma de a[0]=0 sumalocal=0
 thread 0 suma de a[1]=1 sumalocal=1
thread master=0 imprime suma=1
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer1] 2021-04-27 martes
>>./if-clauseModificado_EXE 6 10
 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 4 suma de a[4]=4 sumalocal=4
 thread 0 suma de a[0]=0 sumalocal=0
 thread 5 suma de a[5]=5 sumalocal=5
thread master=0 imprime suma=15
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer1] 2021-04-27 martes
>>
```

RESPUESTA:
Esta cláusula permite indicar el número de hilos a utilizarse para la sección paralela en la que se incluye, se puede ver esto en las capturas de que, variando el segundo parámetro de entrada que es el valor que se le asigna, aparecen la cantidad de hebras especificadas si las iteraciones la permiten.

2. Rellenar la Tabla 1 (se debe poner en la tabla el id del *thread* que ejecuta cada iteración) usando `scheduler-clause.c` con tres *threads* (0,1,2) y un número de iteraciones de 16 (0 a 15 en la tabla). Con este ejercicio se pretende comparar distintas alternativas de planificación de bucles. Se van a usar distintos tipos (`static`, `dynamic`, `guided`), modificadores (`monotonic` y `nonmonotonic`) y tamaños de chunk (x = 1, 2 y 4).

Tabla 1 . Tabla schedule. Rellenar esta tabla ejecutando `scheduler-clause.c` asignando previamente a la variable de entorno `OMP_SCHEDULE` los valores que se indican en la tabla (por ej.: `export OMP_SCHEDULE="nonmonotonic:static,2)`. En la segunda fila, 1, 2 4 representan el tamaño del chunk

Iter	"monotonic:static,x"			"nonmonotonic:static,x"			"monotonic:dynamic,x"			"monotonic:guided,x"		
	x=1	x=2	x=4	x=1	x=2	x=4	x=1	x=2	x=4	x=1	x=2	x=4
0	0	0	0	0	0	0	0	0	0	1	0	1
1	1	0	0	1	0	0	1	0	0	1	0	1
2	2	1	0	2	1	0	2	2	0	1	0	1
3	0	1	0	0	1	0	0	1	0	1	0	1
4	1	2	1	1	2	1	0	1	1	1	0	1
5	2	2	1	2	2	1	0	2	1	1	0	1
6	0	0	1	0	0	1	0	0	1	0	2	0
7	1	0	1	1	0	1	0	0	1	0	2	0
8	2	1	2	2	1	2	0	1	2	0	2	0
9	0	1	2	0	1	2	0	1	2	0	2	0
10	1	2	2	1	2	2	0	1	2	2	1	2
11	2	2	2	2	2	2	0	1	2	2	1	2
12	0	0	0	0	0	0	0	1	0	2	1	2
13	1	0	0	1	0	0	0	1	0	2	1	2
14	2	1	0	2	1	0	0	1	0	2	1	0
15	0	1	0	0	1	0	0	1	0	2	1	0

Destacar las diferencias entre las 4 alternativas de planificación de la tabla, en particular, las que hay entre `static`, `dynamic` y `guided` y las diferencias entre usar `monotonic` y `nonmonotonic`.

RESPUESTA:
La planificación `static` realiza la asignación de manera round-robin: es decir, una hebra realiza una iteración y luego la siguiente hebra a esa hasta que se llega a la última hebra y comienza de nuevo la primera, independientemente de que sea monotónica o no, y la manera en que los asigna depende del número del chunk, cada hebra realizará tantas iteraciones seguidas dependiendo del tamaño del chunk.

La planificación `dynamic` realiza la asignación de manera, como bien indica, dinámica: cada hebra realiza el número de chunks indicado, al finalizar las ejecuciones pide más iteraciones de tamaño chunk hasta que no hay más iteraciones, los hilos más rápidos realizan más unidades.

La planificación `guided` realiza la asignación de manera decreciente, comienza con un tamaño de bloque

máximo y este se va reduciendo. El tamaño de cada bloque en cada iteración es el número de iteraciones que restan entre el número de hilos; se reduce hasta el tamaño que se ha especificado en el chunk.

El utilizar `monotonic` indica que el hilo ejecuta las iteraciones que tiene asignada de manera lógica creciente, mientras que utilizar `nonmonotonic` hace que el hilo ejecute las iteraciones asignadas en un orden no especificado.

3. ¿Qué valor por defecto usa OpenMP para `chunk` y `modifier` con `static`, `dynamic` y `guided`? Explicar qué ha hecho para contestar a esta pregunta.

- Con `static`, el valor por defecto de `chunk` es aquel valor que divida equitativamente las iteraciones a realizar. El `modifier` que utiliza por defecto es `monotonic`.
- Con `dynamic` y `guided`, el valor por defecto del `chunk` es 1. El `modifier` por defecto que utiliza es `nonmonotonic`.

Se ha ido al `manual de OpenMP 5.1` para encontrar estas interrogantes.

4. 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`

```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;

    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;

    // Variables de estado
    int modifier;
    omp_sched_t kind;
    omp_get_schedule(&kind, &modifier);

    printf("\nVariables fuera de región paralela:\n");
    printf("(Ajuste dinámico de hilos) dyn-var = %d\n", omp_get_dynamic());
    printf("(Hilos en ejecución paralela) nthreads-var = %d\n",
```

```
omp_get_max_threads());
    printf("(Máximo de hilos) thread-limit-var = %d\n", omp_get_thread_limit());
    printf("(Planificación) run-sched-var = %X %d\n\n", 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("\nVariables dentro de región paralela (ejecutado por hilo
%d):\n", omp_get_thread_num());
            printf("(Ajuste dinámico de hilos) dyn-var = %d\n",
omp_get_dynamic());
            printf("(Hilos en ejecución paralela) nthreads-var = %d\n",
omp_get_max_threads());
            printf("(Máximo de hilos) thread-limit-var = %d\n",
omp_get_thread_limit());
            printf("(Planificación) run-sched-var = %X %d\n\n", kind, modifier);
        }
    }

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

CAPTURAS DE PANTALLA:

```
File  Edit  View  Bookmarks  Settings  Help
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>./scheduled-clauseModificado_EXE 10 1
Variables fuera de región paralela:
(Ajuste dinámico de hilos) dyn-var = 1
(Hilos en ejecución paralela) nthreads-var = 8
(Máximo de hilos) thread-limit-var = 2147483647
(Planificación) run-sched-var = 2 1

 thread 0 suma a[2]=2 suma=2
 thread 0 suma a[7]=7 suma=9
 thread 0 suma a[8]=8 suma=17
 thread 0 suma a[9]=9 suma=26
 thread 3 suma a[6]=6 suma=6
 thread 4 suma a[0]=0 suma=0
 thread 5 suma a[3]=3 suma=3
 thread 2 suma a[4]=4 suma=4
 thread 1 suma a[1]=1 suma=1
 thread 6 suma a[5]=5 suma=5
Variables dentro de región paralela (ejecutado por hilo 4):
(Ajuste dinámico de hilos) dyn-var = 1
(Hilos en ejecución paralela) nthreads-var = 8
(Máximo de hilos) thread-limit-var = 2147483647
(Planificación) run-sched-var = 2 1

Fuera de 'parallel for' suma=26
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>
```

```
Fuera de 'parallel for' suma=17
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>export OMP_NUM_THREADS=2
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>./scheduled-clauseModificado_EXE 10 1
Variables fuera de región paralela:
(Ajuste dinámico de hilos) dyn-var = 0
(Hilos en ejecución paralela) nthreads-var = 2
(Máximo de hilos) thread-limit-var = 2147483647
(Planificación) run-sched-var = 2 1

 thread 0 suma a[0]=0 suma=0
 thread 0 suma a[2]=2 suma=2
 thread 0 suma a[3]=3 suma=5
 thread 0 suma a[4]=4 suma=9
 thread 0 suma a[5]=5 suma=14
 thread 0 suma a[6]=6 suma=20
 thread 0 suma a[7]=7 suma=27
 thread 0 suma a[8]=8 suma=35
 thread 0 suma a[9]=9 suma=44
 thread 1 suma a[1]=1 suma=1
Variables dentro de región paralela (ejecutado por hilo 0):
(Ajuste dinámico de hilos) dyn-var = 0
(Hilos en ejecución paralela) nthreads-var = 2
(Máximo de hilos) thread-limit-var = 2147483647
(Planificación) run-sched-var = 2 1

Fuera de 'parallel for' suma=44
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>
```

```
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>export OMP_SCHEDULE="auto,3"
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>./scheduled-clauseModificado_EXE 10 1

Variables fuera de región paralela:
(Ajuste dinámico de hilos) dyn-var = 0
(Hilos en ejecución paralela) nthreads-var = 2
(Máximo de hilos) thread-limit-var = 2147483647
(Planificación) run-sched-var = 4 3

 thread 0 suma a[0]=0 suma=0
 thread 0 suma a[2]=2 suma=2
 thread 0 suma a[3]=3 suma=5
 thread 0 suma a[4]=4 suma=9
 thread 0 suma a[5]=5 suma=14
 thread 0 suma a[6]=6 suma=20
 thread 0 suma a[7]=7 suma=27
 thread 1 suma a[1]=1 suma=1
 thread 1 suma a[9]=9 suma=10
 thread 0 suma a[8]=8 suma=35

Variables dentro de región paralela (ejecutado por hilo 1):
(Ajuste dinámico de hilos) dyn-var = 0
(Hilos en ejecución paralela) nthreads-var = 2
(Máximo de hilos) thread-limit-var = 2147483647
(Planificación) run-sched-var = 4 3

Fuera de 'parallel for' suma=10
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>
```

```
File  Edit  View  Bookmarks  Settings  Help
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>export OMP_DYNAMIC=false
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>./scheduled-clauseModificado_EXE 10 1
Variables fuera de región paralela:
(Ajuste dinámico de hilos) dyn-var = 0
(Hilos en ejecución paralela) nthreads-var = 8
(Máximo de hilos) thread-limit-var = 2147483647
(Planificación) run-sched-var = 2 1

 thread 0 suma a[6]=6 suma=6
 thread 0 suma a[8]=8 suma=14
 thread 0 suma a[9]=9 suma=23
 thread 1 suma a[2]=2 suma=2
 thread 3 suma a[7]=7 suma=7
 thread 4 suma a[3]=3 suma=3
 thread 6 suma a[4]=4 suma=4
 thread 2 suma a[5]=5 suma=5
 thread 5 suma a[0]=0 suma=0
 thread 7 suma a[1]=1 suma=1
Variables dentro de región paralela (ejecutado por hilo 5):
(Ajuste dinámico de hilos) dyn-var = 0
(Hilos en ejecución paralela) nthreads-var = 8
(Máximo de hilos) thread-limit-var = 2147483647
(Planificación) run-sched-var = 2 1

Fuera de 'parallel for' suma=23
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer4] 2021-05-04 martes
>>
```

RESPUESTA:

Se imprimen los mismos resultados dentro y fuera de la región paralela.

5. 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`

```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;

    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
    {
        #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("\nVariables dentro de región paralela (ejecutado por hilo
%d):\n", omp_get_thread_num());
            printf("omp_get_num_threads = %d\n", omp_get_num_threads());
            printf("omp_get_num_procs = %d\n", omp_get_num_procs());
            printf("omp_in_parallel = %d\n", omp_in_parallel());
        }
    }

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

    printf("\nVariables fuera de región paralela:\n");
    printf("omp_get_num_threads = %d\n", omp_get_num_threads());
    printf("omp_get_num_procs = %d\n", omp_get_num_procs());
    printf("omp_in_parallel = %d\n", omp_in_parallel());

}
```

CAPTURAS DE PANTALLA:

```
File  Edit  View  Bookmarks  Settings  Help
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer5] 2021-05-04 martes
>>make
gcc -O2 -fopenmp -o scheduled-clauseModificado4_EXE scheduled-clauseModificado4.c
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer5] 2021-05-04 martes
>>./scheduled-clauseModificado4_EXE 10 1
 thread 5 suma a[0]=0 suma=0
 thread 5 suma a[8]=8 suma=8
 thread 5 suma a[9]=9 suma=17
 thread 3 suma a[2]=2 suma=2
 thread 4 suma a[7]=7 suma=7
 thread 7 suma a[3]=3 suma=3
 thread 2 suma a[4]=4 suma=4
 thread 0 suma a[5]=5 suma=5
 thread 1 suma a[6]=6 suma=6
 thread 6 suma a[1]=1 suma=1

Variables dentro de región paralela (ejecutado por hilo 1):
omp_get_num_threads = 8
omp_get_num_procs = 8
omp_in_parallel = 1
Fuera de 'parallel for' suma=17

Variables fuera de región paralela:
omp_get_num_threads = 1
omp_get_num_procs = 8
omp_in_parallel = 0
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer5] 2021-05-04 martes
>>
```

RESPUESTA:
El número de hilos y si se está en una región paralela o no varían acordemente como es de esperarse, `get_num_procs()` se mantiene estático ya que esa función devuelve el número de procesadores disponibles en el dispositivo.

6. Añadir al programa `scheduled-clause.c` lo necesario para, usando funciones, modificar las variables de control `dyn-var`, `nthreads-var` y `run-sched-var` dentro de la región paralela y fuera de la región paralela. En la modificación de `run-sched-var` se debe usar un valor de `kind` distinto al utilizado en la cláusula `schedule()`. Añadir lo necesario para imprimir el contenido de estas variables antes y después de cada una de las dos modificaciones. Comentar los resultados.

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;

    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;


    // Variables de estado
    int modifier;
    omp_sched_t kind;
    omp_get_schedule(&kind, &modifier);

    // Cambio de variables antes de región paralela
    printf("Fuera de región paralela:\n\tAntes de modificación: dyn-var=%d,
nthreads-var=%d, run-sched-var=%X %d\n",  omp_get_dynamic(),
omp_get_max_threads(), kind, modifier);

    omp_set_dynamic(1);
    omp_set_num_threads(5);
    omp_set_schedule(omp_sched_auto, chunk);

    omp_get_schedule(&kind, &modifier);
    printf("\tDespués de modificación: dyn-var=%d, nthreads-var=%d, run-sched-
var=%X %d\n",  omp_get_dynamic(), omp_get_max_threads(), 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("Dentro de región paralela:\n\tAntes de modificación: dyn-var=
%d, nthreads-var=%d, run-sched-var=%X %d\n",  omp_get_dynamic(),
omp_get_max_threads(), kind, modifier);

            omp_set_dynamic(0);
            omp_set_num_threads(3);
            omp_set_schedule(omp_sched_guided, chunk+2);

            omp_get_schedule(&kind, &modifier);

            printf("\tDespués de modificación: dyn-var=%d, nthreads-var=%d, run-
sched-var=%X %d\n",  omp_get_dynamic(), omp_get_max_threads(), kind, modifier);

        }
    }

}
```

CAPTURAS DE PANTALLA:

```
File  Edit  View  Bookmarks  Settings  Help
       Antes de modificación: dyn-var=1, nthreads-var=5, run-sched-var=4 1
       Después de modificación: dyn-var=0, nthreads-var=3, run-sched-var=3 3
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer6] 2021-05-04 martes
>>make
gcc -O2 -fopenmp -o scheduled-clauseModificado5_EXE scheduled-clauseModificado5.c
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer6] 2021-05-04 martes
>>./scheduled-clauseModificado5_EXE 10 1
Fuera de región paralela:
       Antes de modificación: dyn-var=0, nthreads-var=8, run-sched-var=2 1
       Después de modificación: dyn-var=1, nthreads-var=5, run-sched-var=4 1
 thread 0 suma a[1]=1 suma=1
 thread 0 suma a[4]=4 suma=5
 thread 0 suma a[5]=5 suma=10
 thread 0 suma a[6]=6 suma=16
 thread 0 suma a[7]=7 suma=23
 thread 0 suma a[8]=8 suma=31
 thread 0 suma a[9]=9 suma=40
 thread 1 suma a[0]=0 suma=0
 thread 2 suma a[2]=2 suma=2
 thread 3 suma a[3]=3 suma=3
Dentro de región paralela:
       Antes de modificación: dyn-var=1, nthreads-var=5, run-sched-var=4 1
       Después de modificación: dyn-var=0, nthreads-var=3, run-sched-var=3 3
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer6] 2021-05-04 martes
>>
```

RESPUESTA:

Se puede comprobar que es posible de modificar los valores de estas variables de estado tanto dentro como fuera de la región paralela, y que, además estas modificaciones tienen más prioridad que las definidas en la cláusula del `for` por ejemplo en respecto al tipo de `schedule`.

Resto de ejercicios (usar en atcgrid la cola ac a no ser que se tenga que usar atcgrid4)

7. Implementar un programa secuencial en C que multiplique una matriz triangular inferior por un vector (use variables dinámicas y tipo de datos double). Comparar el orden de complejidad y el número total de operaciones (sumas y productos) de este código respecto al 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>

int main(int argc, char **argv)
{
    if(argc < 2)
    {
        printf("Uso: ./pmtv-sencuencial_EXE n\nn    Dimensión del vector y de la
matriz.\n");
        exit(-1);
    }


    // Declaración de datos.
    struct timespec cgt1,cgt2; double ncgt;

    int N = atoi(argv[1]);

    double *v1, *v3;
        v1 = (double*) malloc(N*sizeof(double));
        v3 = (double*) malloc(N*sizeof(double));

    double **matrix;
        matrix = (double**) malloc(N*sizeof(double*));

    if(v1 == NULL || matrix == NULL || v3 == NULL)
    {
        printf("Error en la reserva de espacio\n");
        exit(-2);
    }

    for(int i=0; i<N; i++)
    {
        matrix[i] = (double*) malloc(N*sizeof(double));
        if(matrix[i] == NULL)
        {
            printf("Error en la reserva de espacio\n");
            exit(-2);
        }
    }

    // Inicialización de datos.
```

```
    for (int i = 0; i < N; i++)
    {
        v1[i] = 1;
        for (int j = 0; j < N; j++)
        {
            if (j <= i)
            {
                matrix[i][j] = 1;
            }
            else
            {
                matrix[i][j] = 0;
            }

        }

    }

    // Cálculo de la multiplicación de v1 * matrix = v3
    int aux;

    clock_gettime(CLOCK_REALTIME,&cgt1);

    for (int i = 0; i < N; i++)
    {
        aux = 0;
        for (int j = i; j < N; j++)
        {
            aux += matrix[j][i]*v1[j];
        }
        v3[i] = aux;
    }

    clock_gettime(CLOCK_REALTIME,&cgt2);

    ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
         (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

    // Imprimir los datos
    printf("Dimension: %d Tiempo: %11.9f \n", N, ncgt);

    printf("Resultado: ");
    if(N<12)
    {
        for (int i = 0; i < N; i++)
        {
            printf("[%d] = %0.2f ", i, v3[i]);
        }

    }
    else
    {
        printf("[0] = %2.2f, ..., [%d] = %2.2f", v3[0], N-1, v3[N-1]);
    }
    printf("\n");


    // Liberar memoria
    free(v1);
    free(v3);
```

```
    for (int i = 0; i < N; i++) free(matrix[i]);
    free(matrix);

    return 0;
}
```

CAPTURAS DE PANTALLA:

8. Implementar en paralelo la multiplicación de una matriz triangular inferior 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. La inicialización de los datos la debe hacer el thread 0. 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`. Mostrar en una captura de pantalla que el código resultante funciona correctamente. NOTA: usar para generar los valores aleatorios, por ejemplo, `drand48_r()`.

CAPTURA CÓDIGO FUENTE: `pmtv-OpenMP.c`

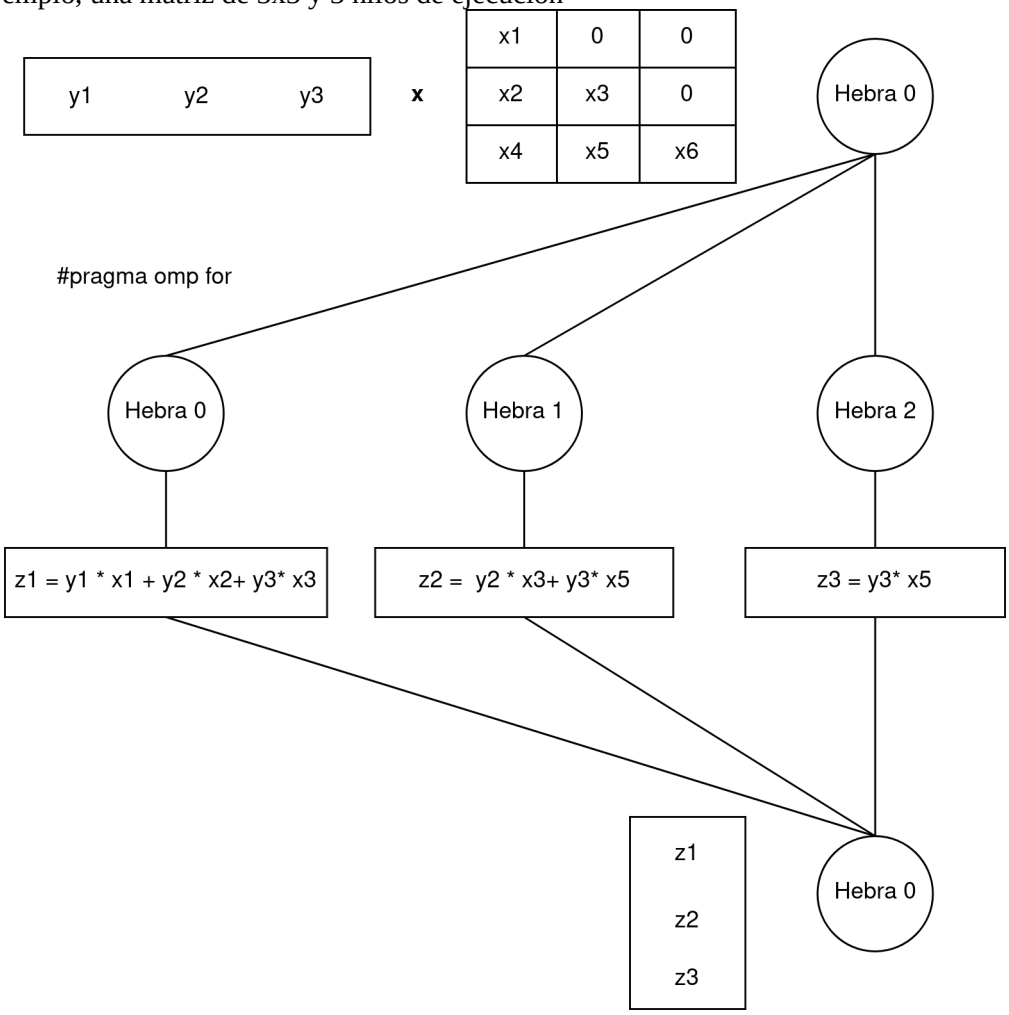
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

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

int main(int argc, char **argv)
{
    if(argc < 2)
    {
        printf("Uso: ./pmtv-OpenMP_EXE n\nn    Dimensión del vector y de la
matriz.\n");
        exit(-1);
    }


    // Declaración de datos.
    double ncgt, cgt1, cgt2;
```

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

    double *v1, *v3;
        v1 = (double*) malloc(N*sizeof(double));
        v3 = (double*) malloc(N*sizeof(double));

    double **matrix;
        matrix = (double**) malloc(N*sizeof(double*));

    if(v1 == NULL || matrix == NULL || v3 == NULL)
    {
        printf("Error en la reserva de espacio\n");
        exit(-2);
    }

    for(int i=0; i<N; i++)
    {
        matrix[i] = (double*) malloc(N*sizeof(double));
        if(matrix[i] == NULL)
        {
            printf("Error en la reserva de espacio\n");
            exit(-2);
        }
    }

    // Inicialización de datos.
    int aux, j;


    #pragma omp parallel private(aux, j) shared(v3)
    {
        #pragma omp master
        {
            struct drand48_data randBuffer;
            double randomNumber;
            srand48_r(time(NULL), &randBuffer);

            for (int i = 0; i < N; i++)
            {
                drand48_r(&randBuffer, &randomNumber);
                v1[i] = randomNumber * 10;
                for (int j = 0; j < N; j++)
                {
                    if (j <= i)
                    {
                        drand48_r(&randBuffer, &randomNumber);
                        matrix[i][j] = randomNumber * 10;
                    }
                    else
                    {
                        matrix[i][j] = 0;
                    }

                }

            }
        }

        #pragma omp barrier

        // Cálculo de la multiplicación de v1 * matrix = v3
```

```
        cgt1 = omp_get_wtime();

        // Filas
        #pragma omp for
        for (int i = 0; i < N; i++)
        {
            aux = 0;
            // Columnas
            for (j = i; j < N; j++)
            {
                aux += matrix[j][i]*v1[j];
            }
            v3[i] = aux;
        }

        cgt2 = omp_get_wtime();
    }
        ncgt = cgt2 - cgt1;

        // Imprimir los datos
        printf("Dimension: %d Tiempo: %11.9f \n", N, ncgt);

        printf("Resultado: ");
        if(N<12)
        {
            for (int i = 0; i < N; i++)
            {
                printf("[%d] = %0.2f ", i, v3[i]);
            }

        }
        else
        {
            printf("[0] = %2.2f, ..., [%d] = %2.2f", v3[0], N-1, v3[N-1]);
        }
        printf("\n");



    // Liberar memoria
    free(v1);
    free(v3);

    for (int i = 0; i < N; i++) free(matrix[i]);
    free(matrix);

    return 0;
}
```

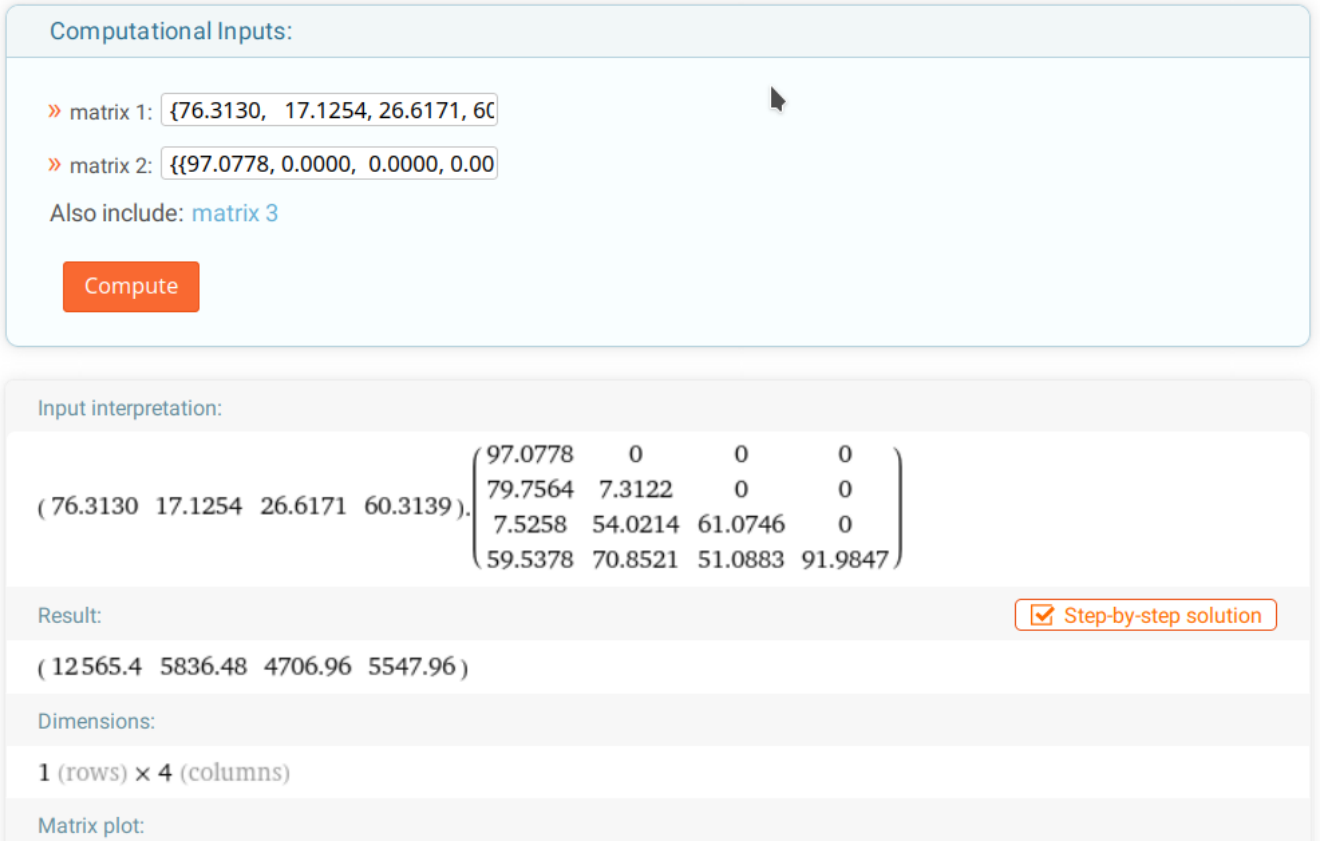
DESCOMPOSICIÓN DE DOMINIO:

En este ejemplo, una matriz de 3x3 y 3 hilos de ejecución



CAPTURAS DE PANTALLA:

```
File  Edit  View  Bookmarks  Settings  Help
99.0760 0.0000  0.0000  0.0000
40.0795 79.3782 0.0000  0.0000
85.6141 61.8478 13.8570 0.0000
38.0423 31.1890 10.8773 87.9234
Dimension: 4 Tiempo: 0.000000901
Resultado: [0] = 10380.0000 [1] = 3787.0000 [2] = 900.0000 [3] = 6253.0000
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer8] 2021-05-11 martes
>>make
gcc -O2 -fopenmp -o pmtv-OpenMP_EXE pmtv-OpenMP.c
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer8] 2021-05-11 martes
>>./pmtv-OpenMP_EXE 4
v1 = [76.3130   17.1254 26.6171 60.3139 ]
M =
97.0778 0.0000  0.0000  0.0000
79.7564 7.3122  0.0000  0.0000
7.5258  54.0214 61.0746 0.0000
59.5378 70.8521 51.0883 91.9847
Dimension: 4 Tiempo: 0.000829136
Resultado: [0] = 12565.4324 [1] = 5836.4847 [2] = 4706.9663 [3] = 5547.9526
[ValentinoLugli valentino@FX505DT:~/Desktop/Git/ac2021/bp3/ejer8] 2021-05-11 martes
```



9. Contestar a las siguientes preguntas sobre el código del ejercicio anterior:

(a) ¿Qué número de operaciones de multiplicación y qué número de operaciones de suma realizan cada uno de los threads en la asignación `static` con `monotonic` y un `chunk` de 1?

RESPUESTA:

Con un número de hilos igual a la dimensión de la matriz, cada hilo realizará una cantidad de multiplicaciones equivalentes a el identificador del hilo – la dimensión de la matriz, y realizará un número de sumas igual al de multiplicaciones – 1.

Con un número de dimensiones mayor al número de hilos, se mantiene que mientras más bajo sea el identificador del hilo, mayores multiplicaciones y sumas realizará, mientras que el hilo que posea el identificador de mayor valor puede que solamente realice una multiplicación sin sumar.

(b) 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:

La asignación `dynamic` hace que los hilos tomen en principio una cantidad de iteraciones igual al tamaño del chunk, una vez que las terminan, piden más hasta que no quedan más iteraciones, mientras que en guided, el tamaño de los chunks comienza grande y se va disminuyendo hasta llegar al valor del chunk especificado, es muy similar a `dynamic` salvo ese detalle, en este caso, no es posible saber con exactitud cuantas multiplicaciones y sumas realizarán cada hebras ya que esta asignación no es predecible, aún así, por lo general deberían distribuirse de una manera más uniforme debido a que las iteraciones son otorgadas de manera que la hebra más rápida pueda realizar más operaciones, y al mismo tiempo, la asignación guided está diseñada para intentar balancear cargas

(c) ¿Qué alternativa ofrece mejores prestaciones? Razonar la respuesta.

RESPUESTA:

Tomando en cuenta lo que se sabe sobre `static`, `dynamic` y `guided`, se puede inferir que la puede que la mejor asignación sea `dynamic` o `guided`, en teoría `guided` debería ser mejor ya que se supone intenta equilibrar la carga que realizan las hebras, pero también es la que más overhead posee de las tres, por lo

tanto en la práctica la mejor asignación será `dynamic`; de igual manera se debe de experimentar para saber que planificación da mejores prestaciones dependiendo de los datos y de la máquina que lo ejecuta.

10. 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 (con `monotonic` en todos los casos). Usar un tamaño de vector N múltiplo del número de cores y de 64 que esté entre 11520 y 23040. El número de threads en las ejecuciones debe coincidir con el número de núcleos del computador. 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 (representar los valores de las dos tablas). Incluir los scripts utilizado en el cuaderno de prácticas.
NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

CAPTURAS DE PANTALLA:

```
File  Edit  View  Bookmarks  Settings  Help
>FIN
[ValentinoLugli c1estudiante16@atcgrid:~/bp3/ejer10] 2021-05-17 lunes
>>exit
logout
Connection to atcgrid.ugr.es closed.
[ValentinoLugli valentino@FX505DT:~/Documents/Git/ac2021/bp3/ejer10] 2021-05-17 lunes
>>ssh -X c1estudiante16@atcgrid.ugr.es
c1estudiante16@atcgrid.ugr.es's password:
Last login: Mon May 17 12:04:36 2021 from vpn-s245187.ugr.es
[ValentinoLugli c1estudiante16@atcgrid:~] 2021-05-17 lunes
>>cd bp3/ejer10/
[ValentinoLugli c1estudiante16@atcgrid:~/bp3/ejer10] 2021-05-17 lunes
>>sbatch -pac -Aac -n1 -c12 --hint=nomultithread  --exclusive pmvt-OpenMP_atcgrid.sh
Submitted batch job 106091
[ValentinoLugli c1estudiante16@atcgrid:~/bp3/ejer10] 2021-05-17 lunes
>>la
bash: la: no se encontró la orden...
[ValentinoLugli c1estudiante16@atcgrid:~/bp3/ejer10] 2021-05-17 lunes
>>ls
pmtv-OpenMP_EXE  pmvt-OpenMP_atcgrid.sh  slurm-106090.out  slurm-106091.out
[ValentinoLugli c1estudiante16@atcgrid:~/bp3/ejer10] 2021-05-17 lunes
>>cat slurm-106090.out
>INICIO SCRIPT\n>>COLA: ac
>>>EJECUTANDO static,...
ejer10 : bash  (c1estudiante16) atcgrid.ugr.es  ejer10 : sftp
```

```
File  Edit  View  Bookmarks  Settings  Help
Dimension: 15360 Tiempo: 0.842538986
Resultado: [0] = 382537.6154, ..., [15359] = 10.1104
Finalizado
>>>EJECUTANDO static,64...
Dimension: 15360 Tiempo: 0.427413050
Resultado: [0] = 385637.6985, ..., [15359] = 93.7518
Finalizado
>>>EJECUTANDO dynamic,64...
Dimension: 15360 Tiempo: 0.418437518
Resultado: [0] = 385750.3127, ..., [15359] = 22.0689
Finalizado
>>>EJECUTANDO guided,64...
Dimension: 15360 Tiempo: 0.824048381
Resultado: [0] = 384028.2101, ..., [15359] = 40.8453
Finalizado
>FIN
[ValentinoLugli c1estudiante16@atcgrid:~/bp3/ejer10] 2021-05-17 lunes
>>sbatch -pac -Aac -n1 -c12 --hint=nomultithread  --exclusive pmvt-OpenMP_atcgrid.sh
Submitted batch job 106092
[ValentinoLugli c1estudiante16@atcgrid:~/bp3/ejer10] 2021-05-17 lunes
>>ls
pmtv-OpenMP_EXE  pmvt-OpenMP_atcgrid.sh  slurm-106091.out  slurm-106092.out
[ValentinoLugli c1estudiante16@atcgrid:~/bp3/ejer10] 2021-05-17 lunes
>>
ejer10 : bash  (c1estudiante16) atcgrid.ugr.es  ejer10 : sftp
```

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid

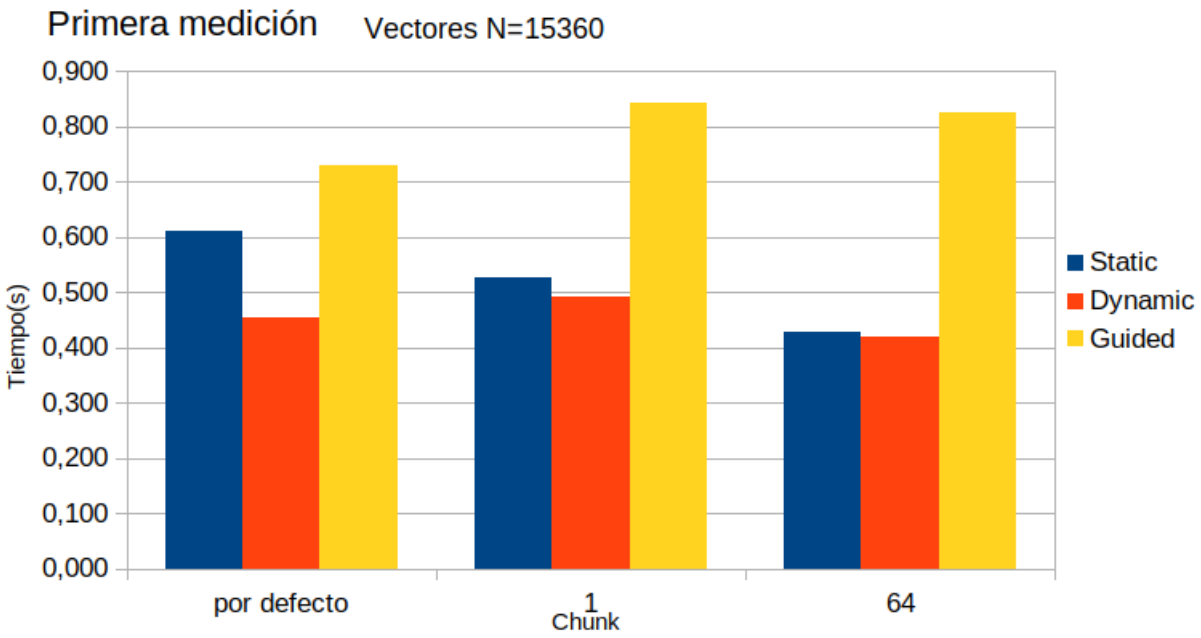
SCRIPT: pmvt-OpenMP_atcgrid.sh

```
#!/bin/bash
#Ordenes para el Gestor de carga de trabajo:
#1. Asigna al trabajo un nombre
#SBATCH --job-name=bp3_10

echo ">INICIO SCRIPT"
echo ">>COLA: $SLURM_JOB_PARTITION"

schedule=("static" "dynamic" "guided" "static,1" "dynamic,1" "guided,1"
"static,64" "dynamic,64" "guided,64")


for (( i=0; i<9; i+=1 )); do

    echo -e ">>>EJECUTANDO ${schedule[$i]}..."

    export OMP_SCHEDULE="monotonic:${schedule[$i]}"

    srun ./pmtv-OpenMP_EXE 15360

    echo "Finalizado"
done

echo ">FIN"
```

Para vectores de N=15360

Chunk	Static	Dynamic	Guided
por defecto	0,61092861	0,454954367	0,728995036
1	0,527765665	0,492020015	0,842538986
64	0,42741305	0,418437518	0,824048381

Chunk	Static	Dynamic	Guided
por defecto	0,602022868	0,462909214	0,805224452
1	0,557505533	0,491554864	0,817123774
64	0,428355489	0,422202423	0,788883634



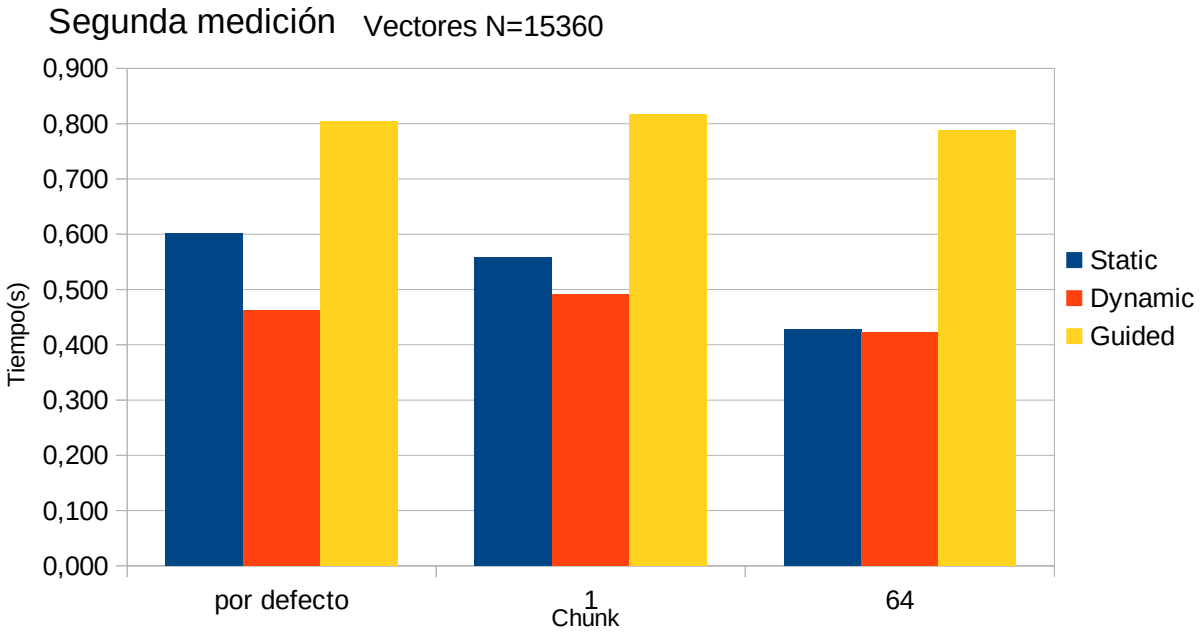


Tabla 2 . Tiempos de ejecución de la versión paralela del producto de una matriz triangular por un vector para vectores de tamaño N= (solo se ha paralelizado el producto, no la inicialización de los datos).

Chunk	Static	Dynamic	Guided
por defecto			
1			
64			
Chunk	Static	Dynamic	Guided

por defecto
1
64