

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): Javier Victoria Mohamed

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`

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

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

    if(argc < 3) {
        fprintf(stderr, "[ERROR]-Falta iteraciones\n y/o Num_Threads");
        exit(-1);
    }

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

    if (n>20) n=20;

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

    #pragma omp parallel if(n>4) default(none) num_threads(x) \
        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 \nprine suma=%d\n",tid,suma);
    }
}
```

#### CAPTURAS DE PANTALLA:

```

Actividades Terminal lun 16:08
javizyv@javizyv-VirtualBox: ~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3

Archivo Editar Ver Buscar Terminal Ayuda
javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3$ PS1="[JavlerVlctoriaMohamed \u@h:\w] \D{%F %A}\n$"
[JavlerVlctoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3] 2019-05-06 Lunes
$cd ejer1
[JavlerVlctoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer1] 2019-05-06 Lunes
$ls
tf-clauseModificado tf-clauseModificado.c
[JavlerVlctoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer1] 2019-05-06 Lunes
$g++ -o tf-clauseModificado tf-clauseModificado.c -fopenmp
[JavlerVlctoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer1] 2019-05-06 Lunes
$./tf-clauseModificado 3 3
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 master=0 imprime suma=3
[JavlerVlctoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer1] 2019-05-06 Lunes
$./tf-clauseModificado 6 5
thread 0 suma de a[0]=0 sumalocal=0
thread 0 suma de a[1]=1 sumalocal=1
thread 3 suma de a[4]=4 sumalocal=4
thread 2 suma de a[3]=3 sumalocal=3
thread 4 suma de a[5]=5 sumalocal=5
thread 1 suma de a[2]=2 sumalocal=2
thread master=0 imprime suma=15
[JavlerVlctoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer1] 2019-05-06 Lunes
$]

```

## RESPUESTA:

Esto nos permite cambiar el número de threads que ejecutan la sección, sin embargo esto solo ocurrirá como podemos ver cuando ponemos 4 o más iteraciones del bucle debido al if que hay en el parallel. En otro caso la hebra master hará todas las iteraciones.

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

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

#### RESPUESTA:

Static:

Observamos como las tareas se distribuyen de forma equitativa en paquete de chunk en chunk.

Dynamic:

La distribución es un poco arbitraria aunque todas siguen un patrón, para `chunk = 1` las hebras harán chunk iteraciones a partir de la iteración = chunk, el resto las hará la hebra 0.

Guided:

Se distribuyen siempre de una forma similar, cada hebra hace mínimo el número de chunk iteraciones y el resto las hará la hebra 0.

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`

```

Actividades Editor de textos mié 10:43
scheduled-clauseModificado.c
~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3/ejer3 Guardar

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

main(int argc, char **argv) {
    int i, n=200, chunk, a[n], suma=0;
    omp_sched_t sched_value;
    //int x=4;

    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;

    //num_threads(x)

    #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) {
                omp_get_schedule(&sched_value, &chunk);
                printf("Dentro de 'parallel for': \n dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, \run-sched-var: %d \n", omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), sched_value);
            }
        }

    printf("Fuera de 'parallel for' suma=%d\n", suma);
    omp_get_schedule(&sched_value, &chunk);
    printf("dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d \n", omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), sched_value);
}

Cargando archivo ~/home/javizzzy/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3/ejer3/scheduled-clauseModificado.c... C Anchura del tabulador: 8 Ln 1, Col 1 INS

```

### CAPTURAS DE PANTALLA:

#### Ejecución sin cambiar nada:

```

Actividades Terminal mié 10:49
javizzzy@javizzzy-VirtualBox: ~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3
Archivo Editar Ver Buscar Terminal Ayuda

[javterVictoriaMohamed javizzzy@javizzzy-VirtualBox:~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
$ ./scheduled-clauseModificado 16 2
thread 0 suma a[0]=0 suma=0
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
thread 0 suma a[1]=1 suma=1
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
thread 0 suma a[12]=12 suma=13
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
thread 0 suma a[13]=13 suma=26
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
thread 0 suma a[14]=14 suma=40
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
thread 0 suma a[15]=15 suma=55
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
thread 3 suma a[4]=4 suma=4
thread 3 suma a[5]=5 suma=9
thread 4 suma a[6]=6 suma=6
thread 4 suma a[7]=7 suma=13
thread 5 suma a[2]=2 suma=2
thread 5 suma a[3]=3 suma=5
thread 1 suma a[8]=8 suma=8
thread 1 suma a[9]=9 suma=17
thread 2 suma a[10]=10 suma=10
thread 2 suma a[11]=11 suma=21
Fuera de 'parallel for' suma=55
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
[javterVictoriaMohamed javizzzy@javizzzy-VirtualBox:~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
$

```

## Ejecución cambiando el num\_threads a 4:

```

Actividades Terminal mié 10:48
javizyv@javizyv-VirtualBox: ~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3

Archivo Editar Ver Buscar Terminal Ayuda
[javivictoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
$.scheduled-clauseModificado 16 2
thread 0 suma a[0]=0 suma=0
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 4, thread-limit-var:2147483647, run-sched-var: 2
thread 0 suma a[1]=1 suma=1
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 4, thread-limit-var:2147483647, run-sched-var: 2
thread 0 suma a[8]=8 suma=9
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 4, thread-limit-var:2147483647, run-sched-var: 2
thread 0 suma a[9]=9 suma=18
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 1 suma a[10]=10 suma=15
thread 1 suma a[11]=11 suma=26
thread 1 suma a[12]=12 suma=38
thread 1 suma a[13]=13 suma=51
thread 1 suma a[14]=14 suma=65
thread 1 suma a[15]=15 suma=80
thread 2 suma a[6]=6 suma=6
thread 2 suma a[7]=7 suma=13
thread 3 suma a[4]=4 suma=4
thread 3 suma a[5]=5 suma=9
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 4, thread-limit-var:2147483647, run-sched-var: 2
Fuera de 'parallel for' suma=80
dyn-var: 0, nthreads-var: 4, thread-limit-var: 2147483647, run-sched-var: 2
[javivictoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
8]

```

## Ejecución cambiando omp\_schedule a static:

```

Actividades Terminal mié 10:53
javizyv@javizyv-VirtualBox: ~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3

Archivo Editar Ver Buscar Terminal Ayuda
[javivictoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
$export OMP_SCHEDULE="static"
[javivictoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
$.scheduled-clauseModificado 16 2
thread 0 suma a[2]=2 suma=2
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:2147483647, run-sched-var: 1
thread 0 suma a[3]=3 suma=5
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:2147483647, run-sched-var: 1
thread 0 suma a[12]=12 suma=17
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:2147483647, run-sched-var: 1
thread 0 suma a[13]=13 suma=30
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:2147483647, run-sched-var: 1
thread 0 suma a[14]=14 suma=44
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:2147483647, run-sched-var: 1
thread 0 suma a[15]=15 suma=59
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:2147483647, run-sched-var: 1
thread 2 suma a[6]=6 suma=6
thread 2 suma a[7]=7 suma=13
thread 4 suma a[4]=4 suma=4
thread 4 suma a[5]=5 suma=9
thread 1 suma a[8]=8 suma=8
thread 1 suma a[9]=9 suma=17
thread 3 suma a[10]=10 suma=10
thread 3 suma a[11]=11 suma=21
thread 5 suma a[0]=0 suma=0
thread 5 suma a[1]=1 suma=1
Fuera de 'parallel for' suma=59
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 1
[javivictoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/Zdo/Zdo cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
8]

```

## Ejecución cambiando omp\_thread\_limit a 10:

```

Actividades Terminal mié 10:55
javizzyv@javizzyv-VirtualBox: ~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3

Archivo Editar Ver Buscar Terminal Ayuda
[javierVictoriaMohamed javizzyv@javizzyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
$export OMP_THREAD_LIMIT=10
[javierVictoriaMohamed javizzyv@javizzyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
$./scheduled-clauseModificado4 10 2
thread 0 suma a[4]=4 suma=4
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:10, run-sched-var: 2
thread 0 suma a[5]=5 suma=9
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:10, run-sched-var: 2
thread 0 suma a[12]=12 suma=21
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:10, run-sched-var: 2
thread 0 suma a[13]=13 suma=34
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:10, run-sched-var: 2
thread 0 suma a[14]=14 suma=48
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:10, run-sched-var: 2
thread 0 suma a[15]=15 suma=63
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var:10, run-sched-var: 2
thread 4 suma a[8]=8 suma=8
thread 4 suma a[9]=9 suma=17
thread 5 suma a[10]=10 suma=27
thread 5 suma a[11]=11 suma=38
thread 3 suma a[6]=6 suma=6
thread 3 suma a[7]=7 suma=13
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 2 suma a[0]=0 suma=0
thread 2 suma a[1]=1 suma=1
Fuera de 'parallel for' suma=63
dyn-var: 0, nthreads-var: 6, thread-limit-var: 10, run-sched-var: 2
[javierVictoriaMohamed javizzyv@javizzyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer3] 2019-05-15 miércoles
$]

```

**RESPUESTA:**

No, devuelve los mismo valores tanto fuera como dentro 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`

```

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

main(int argc, char **argv) {
    int t, n=200, chunk, a[n], suma=0;
    omp_sched_t sched_value;
    //int x=4;

    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;

    //num_threads(x)

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

            if(omp_get_thread_num() == 0) {
                omp_get_schedule(&sched_value, &chunk);
                printf("Dentro de 'parallel for': \n dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d \n", omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), sched_value);
                printf(" get_num_threads: %d, get_num_procs: %d, in_parallel():%d\n", omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
            }
        }

    printf("Fuera de 'parallel for' suma=%d\n", suma);
    omp_get_schedule(&sched_value, &chunk);
    printf("dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d \n", omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), sched_value);
    printf(" get_num_threads: %d, get_num_procs: %d, in_parallel():%d\n", omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
}

```

## CAPTURAS DE PANTALLA:

```

mié 11:03
javizyv@javizyv-VirtualBox: ~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer4
[ javierVictorIaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer4 ] 2019-05-15 miércoles
$ ./scheduled-clauseModificado4 16 2
thread 0 suma a[0]=0 suma=0
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
thread 4 suma a[4]=4 suma=4
thread 4 suma a[5]=5 suma=9
thread 4 suma a[12]=12 suma=21
thread 4 suma a[13]=13 suma=34
thread 4 suma a[14]=14 suma=48
thread 4 suma a[15]=15 suma=63
thread 3 suma a[6]=6 suma=6
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 5 suma a[8]=8 suma=8
thread 2 suma a[10]=10 suma=10
thread 5 suma a[9]=9 suma=17
thread 2 suma a[11]=11 suma=21
get_num_threads: 6, get_num_procs: 6, in_parallel():1
thread 0 suma a[1]=1 suma=1
Dentro de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
get_num_threads: 6, get_num_procs: 6, in_parallel():1
thread 3 suma a[7]=7 suma=13
Fuera de 'parallel for' suma=63
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
get_num_threads: 1, get_num_procs: 6, in_parallel():0
[ javierVictorIaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer4 ] 2019-05-15 miércoles
$

```

## RESPUESTA:

Cambian las variables `get_num_threads` e `in_parallel`. Dentro del `parallel` `get_num_threads` da 6 porque hay 6 hebras ejecutando y fuera 1 porque no es una región paralela. En cuanto `in_parallel` dentro da 1 porque es una región paralela y fuera da 0 porque no lo es.



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

main(int argc, char **argv) {
    int i, n=200, chunk, a[n], suma=0;
    omp_sched_t sched_value;
    //int x=4;

    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;

    //num_threads(x)

    omp_get_schedule(&sched_value, &chunk);

    printf("\nAntes de 'parallel for': \n dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d \n", omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), sched_value);
    printf(" get_num_threads: %d, get_num_procs: %d, in_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(omp_sched_static, 4);

    #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) {
                omp_get_schedule(&sched_value, &chunk);
                printf("Dentro de 'parallel for': \n dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d \n", omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), sched_value);
                printf(" get_num_threads: %d, get_num_procs: %d, in_parallel(): %d\n", omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
            }
        }

    printf("Fuera de 'parallel for' suma=%d\n", suma);
    omp_get_schedule(&sched_value, &chunk);
    printf("dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d \n", omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), sched_value);
    printf(" get_num_threads: %d, get_num_procs: %d, in_parallel(): %d\n", omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
}
```

## CAPTURAS DE PANTALLA:

```
mié 11:27
javizyv@javizyv-VirtualBox: ~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer5

Antes de 'parallel for':
dyn-var: 0, nthreads-var: 6, thread-limit-var: 2147483647, run-sched-var: 2
get_num_threads: 1, get_num_procs: 6, in_parallel(): 0
thread 0 suma a[0]=0 suma=0
Dentro de 'parallel for':
dyn-var: 1, nthreads-var: 2, thread-limit-var: 2147483647, run-sched-var: 1
thread 1 suma a[1]=1 suma=1
thread 1 suma a[2]=2 suma=3
thread 1 suma a[3]=3 suma=6
thread 1 suma a[4]=4 suma=10
thread 1 suma a[5]=5 suma=15
thread 1 suma a[6]=6 suma=21
thread 1 suma a[7]=7 suma=28
thread 1 suma a[8]=8 suma=36
get_num_threads: 2, get_num_procs: 6, in_parallel(): 1
thread 0 suma a[10]=10 suma=10
Dentro de 'parallel for':
dyn-var: 1, nthreads-var: 2, thread-limit-var: 2147483647, run-sched-var: 1
get_num_threads: 2, get_num_procs: 6, in_parallel(): 1
thread 0 suma a[11]=11 suma=21
Dentro de 'parallel for':
dyn-var: 1, nthreads-var: 2, thread-limit-var: 2147483647, run-sched-var: 1
get_num_threads: 2, get_num_procs: 6, in_parallel(): 1
thread 0 suma a[12]=12 suma=33
Dentro de 'parallel for':
dyn-var: 1, nthreads-var: 2, thread-limit-var: 2147483647, run-sched-var: 1
get_num_threads: 2, get_num_procs: 6, in_parallel(): 1
thread 0 suma a[13]=13 suma=46
Dentro de 'parallel for':
dyn-var: 1, nthreads-var: 2, thread-limit-var: 2147483647, run-sched-var: 1
get_num_threads: 2, get_num_procs: 6, in_parallel(): 1
thread 0 suma a[14]=14 suma=60
Dentro de 'parallel for':
dyn-var: 1, nthreads-var: 2, thread-limit-var: 2147483647, run-sched-var: 1
thread 1 suma a[9]=9 suma=45
thread 1 suma a[15]=15 suma=60
get_num_threads: 2, get_num_procs: 6, in_parallel(): 1
Fuera de 'parallel for' suma=60
dyn-var: 1, nthreads-var: 2, thread-limit-var: 2147483647, run-sched-var: 1
get_num_threads: 1, get_num_procs: 6, in_parallel(): 0
[javievictorl@Mohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer5] 2019-05-15 miércoles
8]
```

**RESPUESTA:**

Podemos apreciar que el set en el propio código tiene mucha más preferencia que los export o que los argumentos.

**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**

```

Actividades Editor de textos mié 12:11
pmtv-secuencial.c
~/Escritorio/Universidad/2do/2do cuatrimestre/JAC/Practicas/P3/ejer6 Guardar 96%

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

int main(int argc, char ** argv) {
    int i, j;

    if(argc < 2) {
        fprintf(stderr, "Falta filas/columnas\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 (i=0; i<N; i++){
        matriz[i] = (int*) malloc(N*sizeof(int));
    }

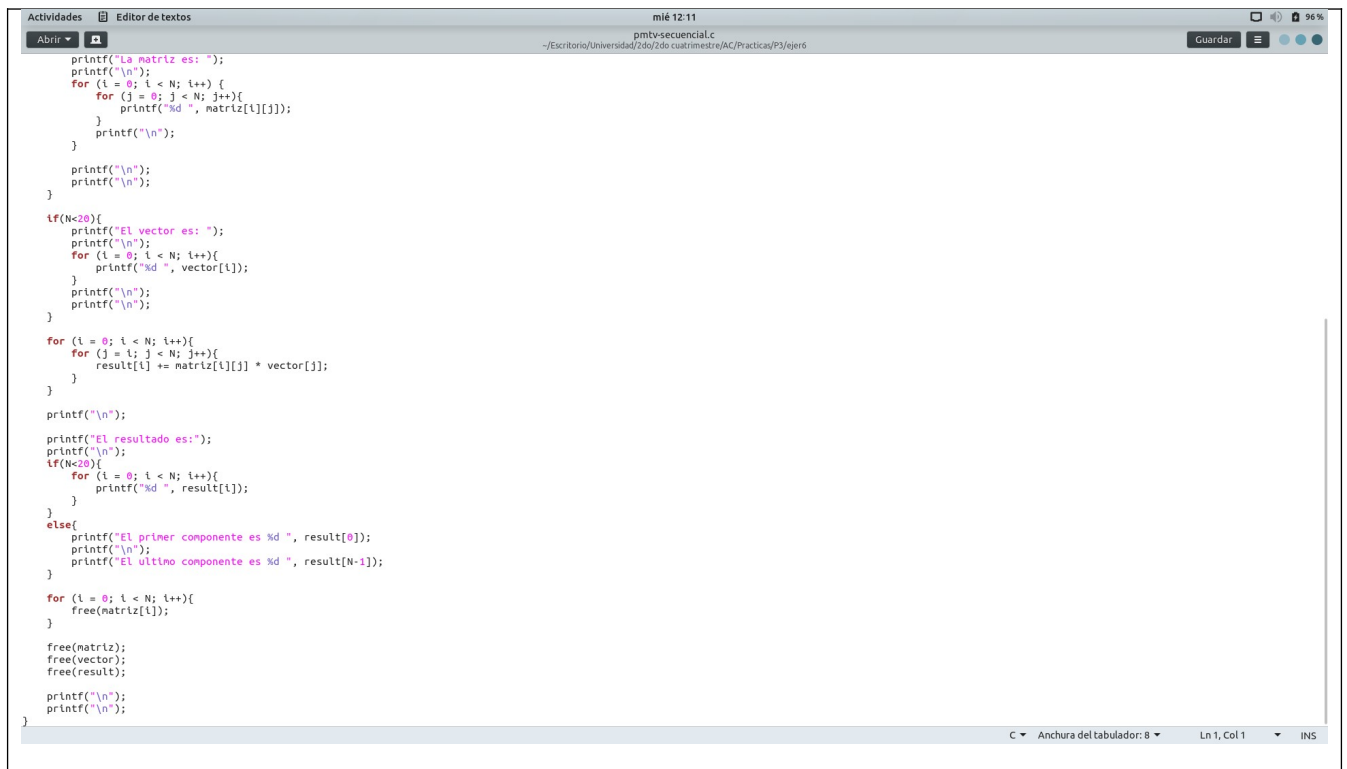
    for (i = 0; i < N; i++) {
        vector[i] = 2;
        result[i] = 0;
        for (j = 0; j < N; j++){
            if(j >= i)
                matriz[i][j] = 3;
            else
                matriz[i][j] = 0;
        }
    }

    if(N<20){
        printf("La matriz es: ");
        printf("\n");
        for (i = 0; i < N; i++) {
            for (j = 0; j < N; j++){
                printf("%d ", matriz[i][j]);
            }
            printf("\n");
        }

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

    if(N<30){
        printf("El vector es: ");
        printf("\n");
        for (i = 0; i < N; i++){
            printf("%d ", vector[i]);
        }
        printf("\n");
    }
}
C Anchura del tabulador: 8 Ln 1, Col 1 INS

```



```
Actividades Editor de textos mié 12:11
pmtv-secuencial.c
~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer6 Guardar

printf("La matriz es: ");
printf("\n");
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++){
        printf("%d ", matriz[i][j]);
    }
    printf("\n");
}

printf("\n");
printf("\n");

if(N<20){
    printf("El vector es: ");
    printf("\n");
    for (i = 0; i < N; i++){
        printf("%d ", vector[i]);
    }
    printf("\n");
    printf("\n");
}

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

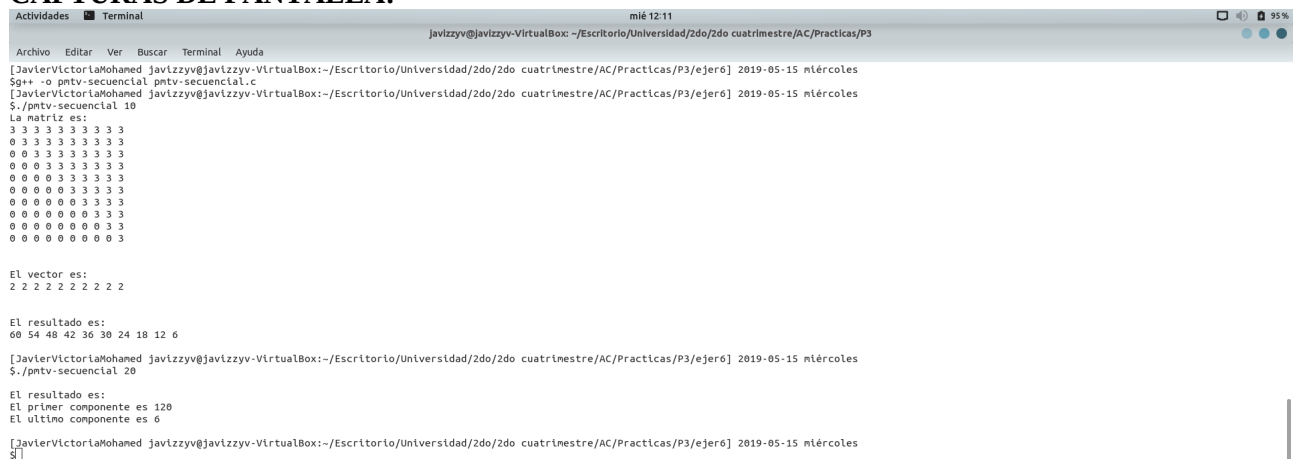
printf("\n");
printf("El resultado es:");
printf("\n");
if(N<20){
    for (i = 0; i < N; i++){
        printf("%d ", result[i]);
    }
}
else{
    printf("El primer componente es %d ", result[0]);
    printf("\n");
    printf("El ultimo componente es %d ", result[N-1]);
}

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

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

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

## CAPTURAS DE PANTALLA:



```
Actividades Terminal mié 12:11
javizyv@javizyv-VirtualBox: ~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3
Archivo Editar Ver Buscar Terminal Ayuda

[javierVictoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer6] 2019-05-15 miércoles
$g++ -o pmtv-secuencial pmtv-secuencial.c
[javierVictoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer6] 2019-05-15 miércoles
$./pmtv-secuencial 10
La matriz es:
3 3 3 3 3 3 3 3
0 3 3 3 3 3 3 3
0 0 3 3 3 3 3 3
0 0 0 3 3 3 3 3
0 0 0 0 3 3 3 3
0 0 0 0 0 3 3 3
0 0 0 0 0 0 3 3
0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 3

El vector es:
2 2 2 2 2 2 2 2

El resultado es:
60 54 48 42 36 30 24 18 12 6

[javierVictoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer6] 2019-05-15 miércoles
$./pmtv-secuencial 20
El resultado es:
El primer componente es 120
El ultimo componente es 6

[javierVictoriaMohamed javizyv@javizyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer6] 2019-05-15 miércoles
$]
```

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:**

#### CAPTURA CÓDIGO FUENTE: pmtv-OpenMP.c

```

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

int main(int argc, char ** argv) {
    int i, j;
    double int, fin;

    if(argc < 2) {
        fprintf(stderr, "Falta filas/columnas\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 (i=0; i<N; i++){
        matriz[i] = (int*) malloc(N*sizeof(int));
    }

    for (i = 0; i < N; i++) {
        vector[i] = 2;
        result[i] = 0;
        for (j = 0; j<N; j++){
            if(j >= i)
                matriz[i][j] = 3;
            else
                matriz[i][j] = 0;
        }
    }

    if(N<20){
        printf("La matriz es: ");
        printf("\n");
        for (i = 0; i < N; i++) {
            for (j = 0; j < N; j++){
                printf("%d ", matriz[i][j]);
            }
            printf("\n");
        }
        printf("\n");
        printf("\n");
    }

    if(N<20){
        printf("El vector es: ");
        printf("\n");
        for (i = 0; i < N; i++){
            printf("%d ", vector[i]);
        }
    }
}

```

```

Actividades Editor de textos mié 12:43
Abrir Guardar 100%
C:\estudiante19 en atcgrindugr.es/home/C3estudiante19/bp3/ejer7

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

if(N<20){
printf("El vector es: ");
printf("\n");
for (i = 0; i < N; i++){
printf("%d ", vector[i]);
}
printf("\n");
printf("\n");
}

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

printf("El resultado es:");
printf("\n");
if(N<20){
for (i = 0; i < N; i++){
printf("%d ", result[i]);
}
}
else{
printf("El primer componente es %d ", result[0]);
printf("\n");
printf("El ultimo componente es %d ", result[N-1]);
}

printf("\n");
printf("Tiempo: %11.9f\t", fin-int);

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

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

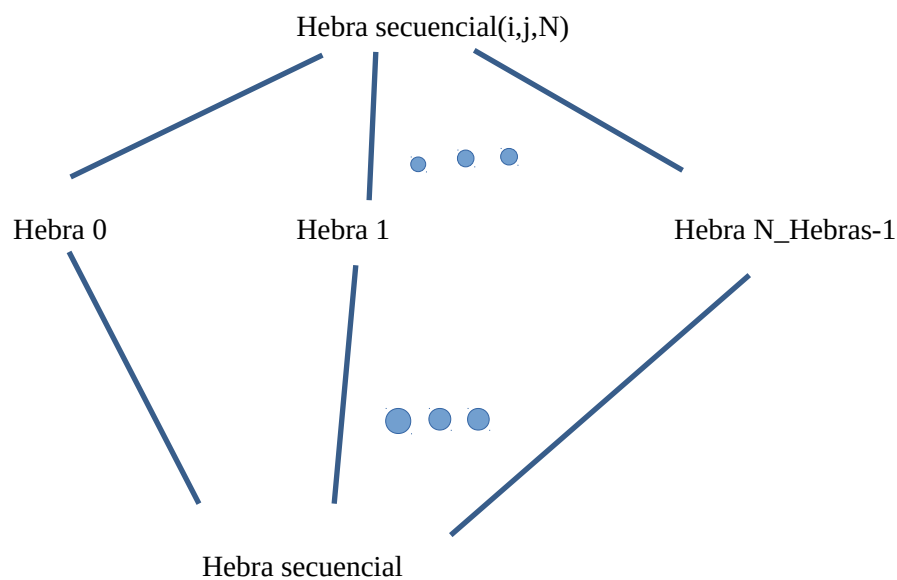
printf("\n");
printf("\n");
}

```

C Anchura del tabulador: 8 Ln 1, Col 1 INS

## DESCOMPOSICIÓN DE DOMINIO:

La hebra secuencial difundiría tanto las componentes  $i, j$  como el tamaño del problema  $N$ . Después cuando todas terminasen seguiría la hebra secuencial.



**CAPTURAS DE PANTALLA:**

```

Actividades Terminal mié 12:43
C3estudiante19@atcgriid:~/bp3/ejer7

Archivo Editar Ver Buscar Terminal Ayuda

pmtv-OpenMP pmtv-OpenMP.c pmtv-OpenMP.e21437 pmtv-OpenMP.o21437 script.sh
[JavierVictoriaMohamed C3estudiante19@atcgriid:~/bp3/ejer7] 2019-05-15 miércoles
$cat pmtv-OpenMP.o21437
Id. usuario del trabajo: C3estudiante19
Id. del trabajo: 21437.atcgriid
Nombre del trabajo especificado por usuario: pmtv-OpenMP
Directorio en el que se ha ejecutado qsub: /home/C3estudiante19/bp3/ejer7
Directorio de trabajo: /home/C3estudiante19
Cola: ac
Modo que ejecuta qsub: atcgriid
Nodos asignados al trabajo:
atcgriid2
Nº de threads inicial: 12
estatico con chunk por defecto

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.035314373

estatico con chunk de 1

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.036739649

estatico con chunk de 64

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.030951973

dinamico con chunk por defecto

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.032565041

dinamico con chunk de 1

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.035944406

dinamico con chunk de 64

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.028559766

```

```

Actividades Terminal mié 12:43
C3estudiante19@atcgriid:~/bp3/ejer7

Archivo Editar Ver Buscar Terminal Ayuda

Tiempo: 0.036739649

estatico con chunk de 64

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.030951973

dinamico con chunk por defecto

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.032565041

dinamico con chunk de 1

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.035944406

dinamico con chunk de 64

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.028559766

guiado con chunk por defecto

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.036856757

guiado con chunk de 1

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.050059728

guiado con chunk de 64

El resultado es:
El primer componente es 92160
El ultimo componente es 6
Tiempo: 0.035769613

[JavierVictoriaMohamed C3estudiante19@atcgriid:~/bp3/ejer7] 2019-05-15 miércoles
$cat pmtv-OpenMP.o21437 > resultado.txt
[JavierVictoriaMohamed C3estudiante19@atcgriid:~/bp3/ejer7] 2019-05-15 miércoles
$]

```

**TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid****SCRIPT: pmvt-OpenMP\_atcgrid.sh**

Script:

```

mié 12:44
script.sh
C:\estudiante19 en atcgrid.ugr.es/home/C\estudiante19\bp3\ejer7

Abrir
Guardar

echo "Id. del trabajo: $PBS_JOBID"
echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
echo "Directorio de trabajo: $PBS_JOBDIR"
echo "Cola: $PBS_QUEUE"
echo "Nodo que ejecuta qsub: $PBS_O_HOST"
echo "Nodos asignados al trabajo:"
cat $PBS_NODEFILE
# FIN del trozo que deben incluir todos los scripts

#Se fija a 12 el nº de threads máximo (tantas como cores en un nodo)
export OMP_THREAD_LIMIT=12
export OMP_NUM_THREADS=12
echo "Nº de threads inicial: $OMP_THREAD_LIMIT"

export OMP_SCHEDULE="static"
echo "estatico con chunk por defecto"
$PBS_O_WORKDIR/pmtv-OpenMP 15360

export OMP_SCHEDULE="static,1"
echo "estatico con chunk de 1"
$PBS_O_WORKDIR/pmtv-OpenMP 15360

export OMP_SCHEDULE="static,64"
echo "estatico con chunk de 64"
$PBS_O_WORKDIR/pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic"
echo "dinamico con chunk por defecto"
$PBS_O_WORKDIR/pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic,1"
echo "dinamico con chunk de 1"
$PBS_O_WORKDIR/pmtv-OpenMP 15360

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

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

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

export OMP_SCHEDULE="guided,64"
echo "guiado con chunk de 64"
$PBS_O_WORKDIR/pmtv-OpenMP 15360

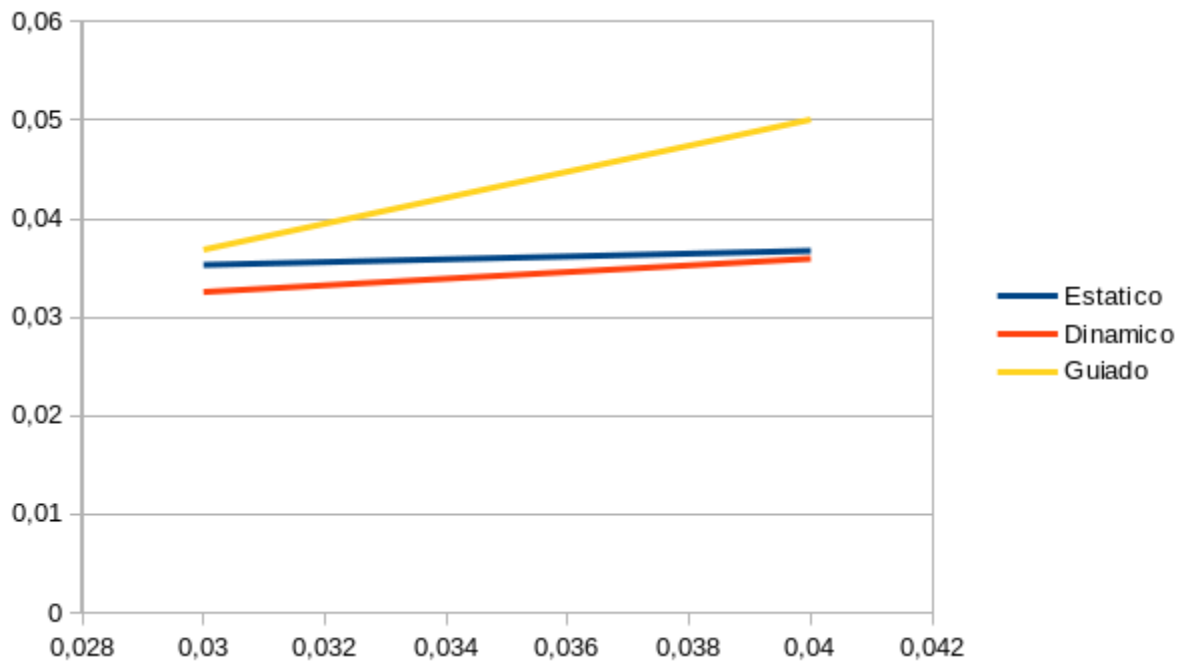
unset OMP_SCHEDULE
unset OMP_THREAD_LIMIT
unset OMP_NUM_THREADS
sh Anchura del tabulador: 8 Ln 1, Col 1 INS

```

**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	0,035314373	0,032565041	0,036856757
1	0,036739649	0,035944406	0,050059728
64	0,030951973	0,028559766	0,035769613
Chunk	Static	Dynamic	Guided
por defecto	0.040378354	0.033395811	0.033954578
1	0.038311711	0.034458528	0.037165847
64	0.030965774	0.028647606	0.049648141

Grafica:



-¿Qué alternativa ofrece mejores prestaciones?

Observamos en la gráfica como es mejor la planificación dinámica en cualquier caso, al menos en la traza realizada. Esto es debido a que la planificación dinámica optimiza la asignación de hebras.

-a)

Estatica: 1, porque los tiempos en chunk 1 y por defecto son similares.

Dinámica: 1, porque los tiempos en chunk 1 y por defecto son similares.

Guiada: En este caso parece que el chunk depende de la carga y se establece según esto. Más o menos sería el número de iteraciones/número de hebras.

-b) Sería el número del chunk \* el número de fila.

-c) Se repartirá según la carga de cada iteración, siendo la hebra que hace una iteración con más carga hace menos iteraciones y viceversa.



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

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

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

```

Actividades Editor de textos mié 13:38
pmm-secuencial.c
~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer8 Guardar

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

int main(int argc, char ** argv) {
    int i, j, k;

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

    unsigned int N = atoi(argv[1]);
    int **m1, **m2, **m3;
    m1 = (int **) malloc(N*sizeof(int*));
    m2 = (int **) malloc(N*sizeof(int*));
    m3 = (int **) malloc(N*sizeof(int*));

    for (i = 0; i < N; ++i) {
        m1[i] = (int*) malloc(N*sizeof(int));
        m2[i] = (int*) malloc(N*sizeof(int));
        m3[i] = (int*) malloc(N*sizeof(int));
    }

    for (i = 0; i < N; ++i) {
        for (j = 0; j < N; ++j) {
            m1[i][j] = 2;
            m2[i][j] = 3;
            m3[i][j] = 0;
        }
    }

    for (i = 0; i < N; ++i)
        for (j = 0; j < N; ++j)
            for (k = 0; k < N; ++k)
                m3[i][j] = m1[i][k] * m2[k][j];

    printf("\n");

    if(N<20){
        for (i = 0; i < N; ++i) {
            for (j = 0; j < N; ++j)
                printf("%d ", m3[i][j]);
            printf("\n");
        }
    }
    else{
        printf("Primer componente: %d ", m3[0][0]);
        printf("\n");
        printf("Segundo componente: %d ", m3[N-1][N-1]);
        printf("\n");
    }

    for (i = 0; i < N; ++i) {
        free(m1[i]);
    }
}
C Anchura del tabulador: 8 Ln 1, Col 1 INS

```

```
mié 13:38
pmm-secuencialC
~/Escritorio/Universidad/zdo/zdo cuatrimestre/AC/Practicas/P3/ejer8

Actividades Editor de textos
Abrir Guardar

printf(stderr, "Falta el tamaño del vector/matriz\n");
exit(-1);
}

unsigned int N = atoi(argv[1]);
int **m1, **m2, **m3;
m1 = (int **) malloc(N*sizeof(int*));
m2 = (int **) malloc(N*sizeof(int*));
m3 = (int **) malloc(N*sizeof(int*));

for (i = 0; i < N; ++i) {
    m1[i] = (int*) malloc(N*sizeof(int));
    m2[i] = (int*) malloc(N*sizeof(int));
    m3[i] = (int*) malloc(N*sizeof(int));
}

for (i = 0; i < N; ++i) {
    for (j = 0; j < N; ++j) {
        m1[i][j] = 2;
        m2[i][j] = 3;
        m3[i][j] = 0;
    }
}

for (i = 0; i < N; ++i)
    for (j = 0; j < N; ++j)
        for (k = 0; k < N; ++k)
            m3[i][j] = m1[i][k] * m2[k][j];

printf("\n");

if (N <= 20) {
    for (i = 0; i < N; ++i) {
        for (j = 0; j < N; ++j)
            printf("%d ", m3[i][j]);
        printf("\n");
    }
} else {
    printf("Primer componente: %d ", m3[0][0]);
    printf("\n");
    printf("Segundo componente: %d ", m3[N-1][N-1]);
    printf("\n");
}

for (i = 0; i < N; ++i) {
    free(m1[i]);
    free(m2[i]);
    free(m3[i]);
}

free(m1);
free(m2);
free(m3);
}
```

C Anchura del tabulador: 8 Ln 1, Col 1 INS

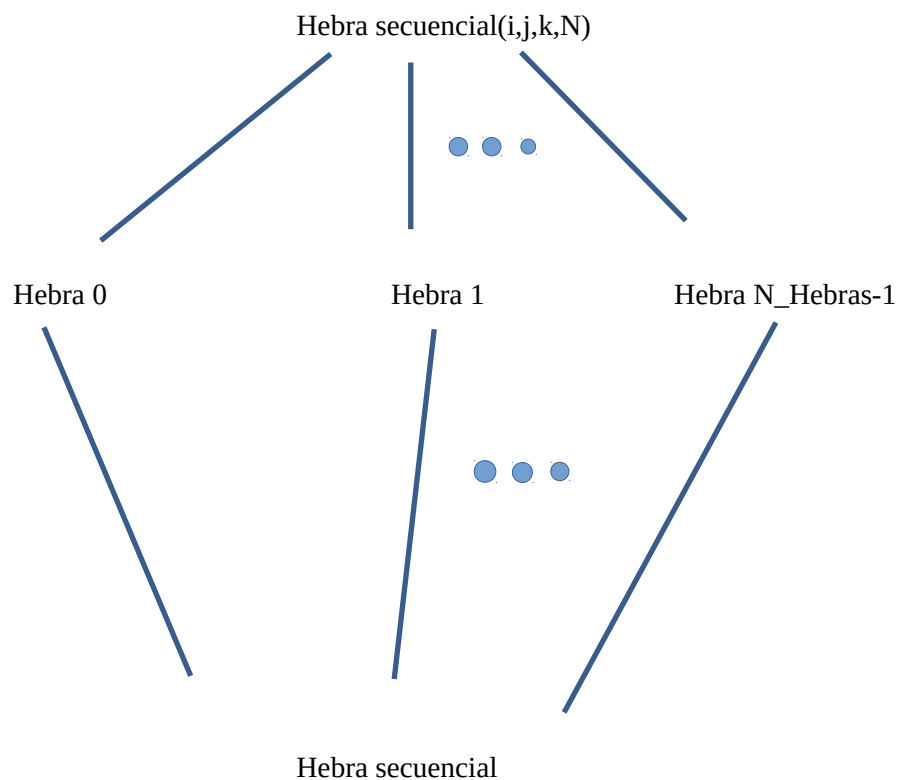
## CAPTURAS DE PANTALLA:

[illegible]

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:

La hebra secuencial difundiría tanto las componentes  $i, j, k$  como el tamaño del problema  $N$ . Después cuando todas terminasen seguiría la hebra secuencial.



## CAPTURA CÓDIGO FUENTE: pmm-OpenMP.c

```

Actividades Editor de textos mié 14:05
pmm-OpenMP.c
~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer9
Guardar

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

int main(int argc, char ** argv) {
    int l, j, k;
    double ini, fin;

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

    unsigned int N = atoi(argv[1]);
    int **m1, **m2, **m3;
    m1 = (int **) malloc(N*sizeof(int*));
    m2 = (int **) malloc(N*sizeof(int*));
    m3 = (int **) malloc(N*sizeof(int*));

    for (l = 0; l < N; ++l) {
        m1[l] = (int*) malloc(N*sizeof(int));
        m2[l] = (int*) malloc(N*sizeof(int));
        m3[l] = (int*) malloc(N*sizeof(int));
    }

    #pragma omp parallel for private(j,k)
    for (l = 0; l < N; ++l) {
        for (j = 0; j < N; ++j) {
            m1[l][j] = 2;
            m2[l][j] = 3;
            m3[l][j] = 0;
        }
    }

    ini = omp_get_wtime();
    #pragma omp parallel for private(j,k)
    for (l = 0; l < N; ++l){
        for (j = 0; j < N; ++j)
            for (k = 0; k < N; ++k)
                m3[l][j] = m1[l][k] * m2[k][j];
    }
    fin = omp_get_wtime();

    printf("\n");

    if(N<20){
        printf("El resultado sería:");
        printf("\n");
        for (l = 0; l < N; ++l) {
            for (j = 0; j < N; ++j)
                printf("%d ", m3[l][j]);
            printf("\n");
        }
    }
    else{

Actividades Editor de textos mié 14:05
pmm-OpenMP.c
~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer9
Guardar

        m2[l] = (int*) malloc(N*sizeof(int));
        m3[l] = (int*) malloc(N*sizeof(int));
    }

    #pragma omp parallel for private(j,k)
    for (l = 0; l < N; ++l) {
        for (j = 0; j < N; ++j) {
            m1[l][j] = 2;
            m2[l][j] = 3;
            m3[l][j] = 0;
        }
    }

    ini = omp_get_wtime();
    #pragma omp parallel for private(j,k)
    for (l = 0; l < N; ++l){
        for (j = 0; j < N; ++j)
            for (k = 0; k < N; ++k)
                m3[l][j] = m1[l][k] * m2[k][j];
    }
    fin = omp_get_wtime();

    printf("\n");

    if(N<20){
        printf("El resultado sería:");
        printf("\n");
        for (l = 0; l < N; ++l) {
            for (j = 0; j < N; ++j)
                printf("%d ", m3[l][j]);
            printf("\n");
        }
    }
    else{
        printf("Primer componente: %d ", m3[0][0]);
        printf("\n");
        printf("Segundo componente: %d ", m3[N-1][N-1]);
        printf("\n");
    }

    printf("\n");
    printf("Tiempo: %11.9f\t", fin-ini);
    printf("\n");
    printf("\n");

    for (l = 0; l < N; ++l) {
        free(m1[l]);
        free(m2[l]);
        free(m3[l]);
    }

    free(m1);
    free(m2);
    free(m3);
}

```

### CAPTURAS DE PANTALLA:

```
Actividades Terminal mié 14:04 javizzyv@javizzyv-VirtualBox: ~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3
```

---

```
Archivo Editar Ver Buscar Terminal Ayuda
```

```
[JavierVictoriaMohamed javizzyv@javizzyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer9] 2019-05-15 miércoles  
$g++ -O2 -o pmm-OpenMP pmm-OpenMP.c -fopenmp  
[JavierVictoriaMohamed javizzyv@javizzyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer9] 2019-05-15 miércoles  
$. /pmm-OpenMP 10
```

```
El resultado seria:  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
Tiempo: 0.000013922
```

```
[JavierVictoriaMohamed javizzyv@javizzyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer9] 2019-05-15 miércoles  
$. /pmm-OpenMP 20
```

```
Primer componente: 6  
Segundo componente: 6
```

```
Tiempo: 0.000021720
```

```
[JavierVictoriaMohamed javizzyv@javizzyv-VirtualBox:~/Escritorio/Universidad/2do/2do cuatrimestre/AC/Practicas/P3/ejer9] 2019-05-15 miércoles  
$
```

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 utilizados en el cuaderno de prácticas. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

### ESTUDIO DE ESCALABILIDAD EN atcgrid:

Threads = 12

N = 1500

Chunk	Static	Dynamic	Guided
<b>por defecto</b>	3,301642434	3,293433338	3,555089287
<b>1</b>	3,292049173	4,430911989	3,288901772
<b>64</b>	3,323104274	3,287058310	3,377660259

### SCRIPT: pmm-OpenMP\_atcgrid.sh

```

#!/bin/bash
#Todos los scripts que se hagan para atcgrid deben incluir lo siguiente:
#1. Se asigna al trabajo el nombre helloomp
#PBS -N pmm-OpenMP
#2. Se asigna el trabajo a la cola ac
#PBS -q ac
#3. Se imprime información del trabajo usando variables de entorno de PBS
echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
echo "Id. del trabajo: $PBS_JOBID"
echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
echo "Directorio de trabajo: $PBS_JOBDIR"
echo "Cola: $PBS_QUEUE"
echo "Nodo que ejecuta qsub: $PBS_O_HOST"
echo "Nodos asignados al trabajo:"
cat $PBS_NODEFILE
# FIN del trozo que deben incluir todos los scripts

#Se fija a 12 el nº de threads máximo (tantas como cores en un nodo)
export OMP_THREAD_LIMIT=12
export OMP_NUM_THREADS=12
echo "Nº de threads inicial: $OMP_THREAD_LIMIT"

export OMP_SCHEDULE="static"
echo "estático con chunk por defecto"
$PBS_O_WORKDIR/pmm-OpenMP 1500

export OMP_SCHEDULE="static,1"
echo "estático con chunk de 1"
$PBS_O_WORKDIR/pmm-OpenMP 1500

export OMP_SCHEDULE="static,64"
echo "estático con chunk de 64"
$PBS_O_WORKDIR/pmm-OpenMP 1500

export OMP_SCHEDULE="dynamic"
echo "dinámico con chunk por defecto"
$PBS_O_WORKDIR/pmm-OpenMP 1500

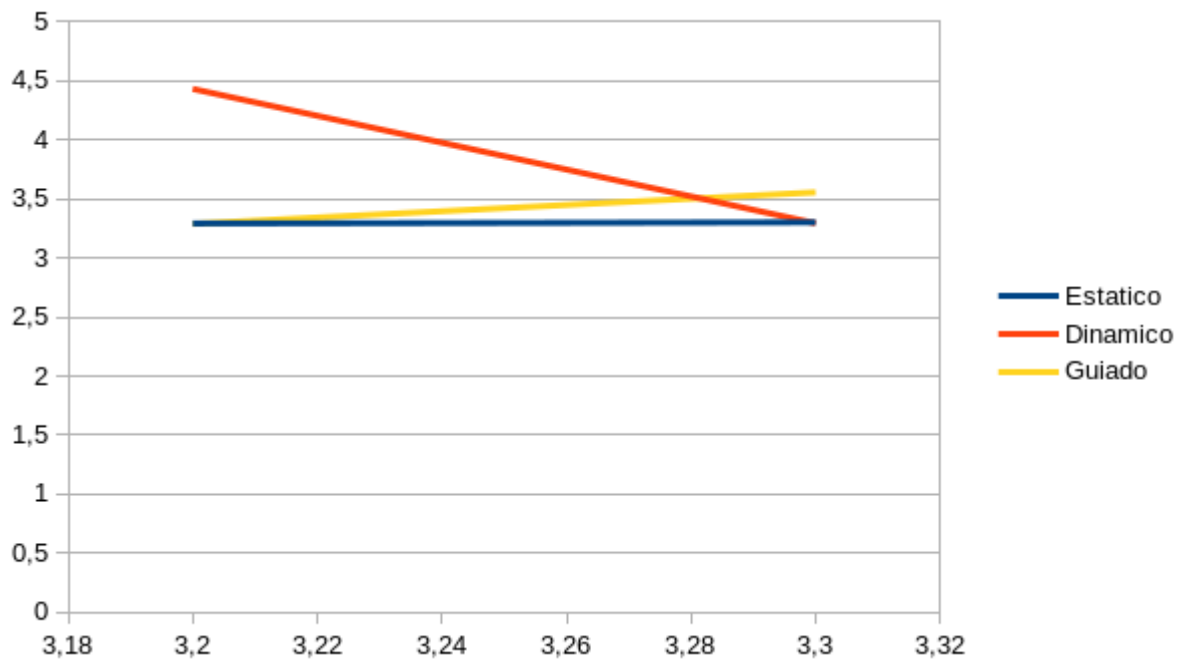
export OMP_SCHEDULE="dynamic,1"
echo "dinámico con chunk de 1"
$PBS_O_WORKDIR/pmm-OpenMP 1500

export OMP_SCHEDULE="dynamic,64"
echo "dinámico con chunk de 64"
$PBS_O_WORKDIR/pmm-OpenMP 1500

export OMP_SCHEDULE="guided"
echo "guiado con chunk por defecto"
$PBS_O_WORKDIR/pmm-OpenMP 1500

export OMP_SCHEDULE="guided,1"
echo "guiado con chunk de 1"
$PBS_O_WORKDIR/pmm-OpenMP 1500
  
```

Gráfica:



Aquí observamos algo ligeramente diferente, vemos como es más optimo planificar estáticamente en ATCgrid y que para valores bajos de chunk la planificación dinámica se comporta bastante peor. Sin embargo podemos apreciar un aumento significativo en las prestaciones con la planificación dinámica para valores de chunk altos por lo que seguramente que se comporte mejor que el estático para valores mayores.

## ESTUDIO DE ESCALABILIDAD EN PCLOCAL:

Threads = 6

N = 1000

Chunk	Static	Dynamic	Guided
<b>por defecto</b>	0,205095852	0,166112108	0,176633466
<b>1</b>	0,173139166	0,165984120	0,162882676
<b>64</b>	0,175329787	0,164251844	0,170652225

### SCRIPT: pmm-OpenMP\_pcllocal.sh

```

#i/bt/bash
#Se fija a 12 el n° de threads máximo (tantas como cores en un nodo)
export OMP_THREAD_LIMIT=6
export OMP_NUM_THREADS=6
echo "Nº de threads inicial: $OMP_THREAD_LIMIT"

export OMP_SCHEDULE="static"
echo "estatico con chunk por defecto"
./pmm-OpenMP 1000

export OMP_SCHEDULE="static,1"
echo "estatico con chunk de 1"
./pmm-OpenMP 1000

export OMP_SCHEDULE="static,64"
echo "estatico con chunk de 64"
./pmm-OpenMP 1000

export OMP_SCHEDULE="dynamic"
echo "dinamico con chunk por defecto"
./pmm-OpenMP 1000

export OMP_SCHEDULE="dynamic,1"
echo "dinamico con chunk de 1"
./pmm-OpenMP 1000

export OMP_SCHEDULE="dynamic,64"
echo "dinamico con chunk de 64"
./pmm-OpenMP 1000

export OMP_SCHEDULE="guided"
echo "guiado con chunk por defecto"
./pmm-OpenMP 1000

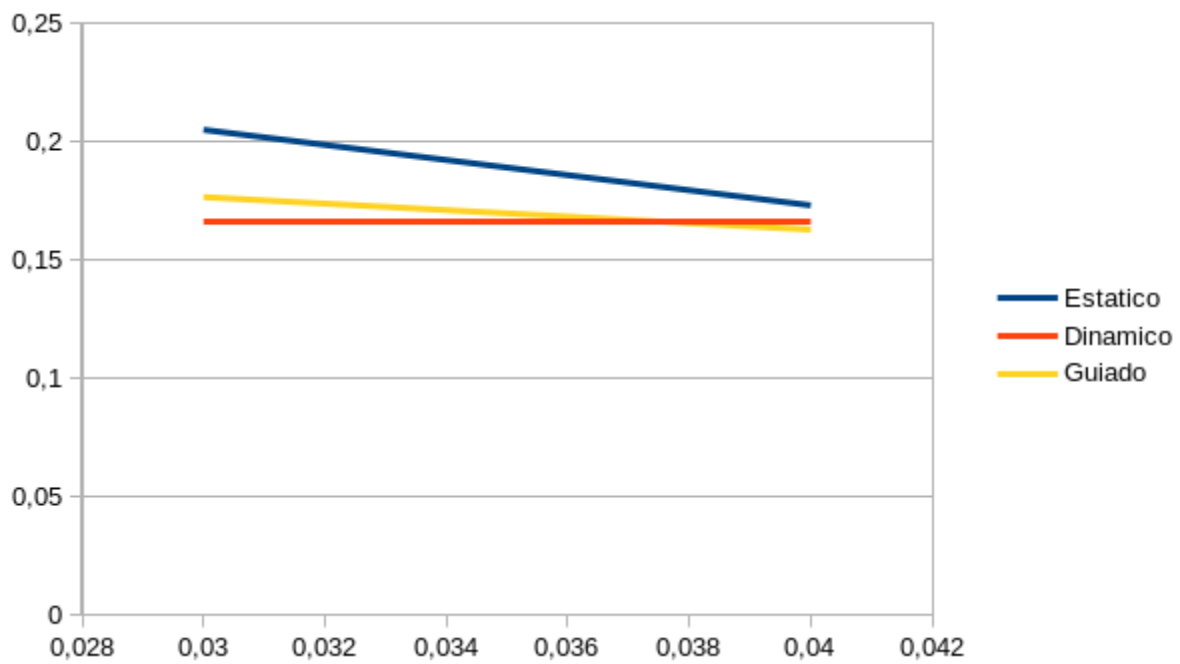
export OMP_SCHEDULE="guided,1"
echo "guiado con chunk de 1"
./pmm-OpenMP 1000

export OMP_SCHEDULE="guided,64"
echo "guiado con chunk de 64"
./pmm-OpenMP 1000

unset OMP_SCHEDULE
unset OMP_THREAD_LIMIT
unset OMP_NUM_THREADS
  
```



Gráfica:



Podemos ver como en mi PC la planificación dinámica sigue siendo mejor que el resto extecto para valores de chunk más altos.