

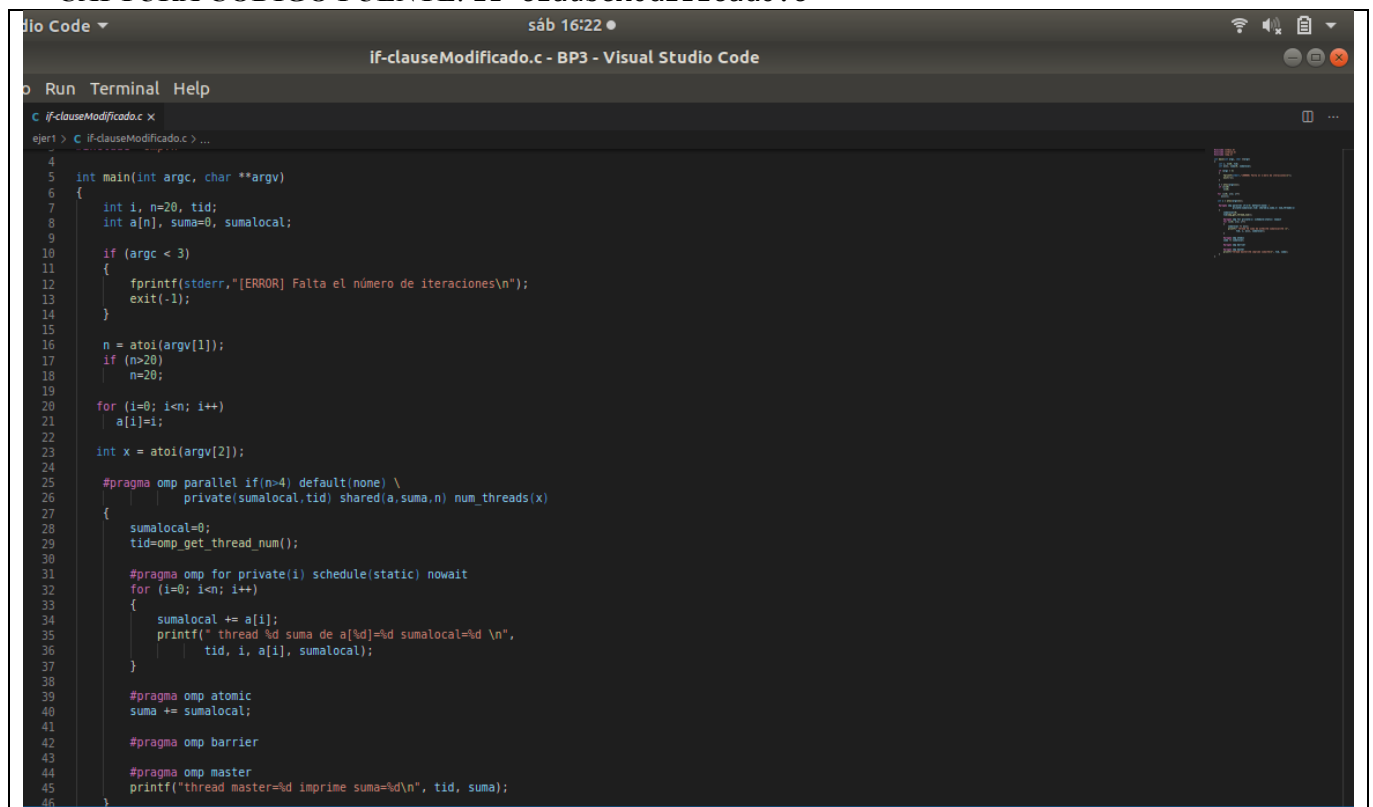
<p>2º curso / 2º cuatr. Grado Ing. Inform.</p>	<h1>Arquitectura de Computadores (AC)</h1> <h2>Cuaderno de prácticas.</h2> <h3>Bloque Práctico 3. Programación paralela III: Interacción con el entorno en OpenMP</h3> <p>Estudiante (nombre y apellidos): Alberto Llamas González Grupo de prácticas: D3 Fecha de entrega: Fecha evaluación en clase:</p>
--	--

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`



```

1  #include <stdio.h>
2  #include <omp.h>
3
4  int main(int argc, char **argv)
5  {
6      int i, n=20, tid;
7      int a[n], suma=0, sumalocal;
8
9      if (argc < 3)
10     {
11         fprintf(stderr, "[ERROR] Falta el número de iteraciones\n");
12         exit(-1);
13     }
14
15     n = atoi(argv[1]);
16     if (n>20)
17         n=20;
18
19     for (i=0; i<n; i++)
20         a[i]=i;
21
22     int x = atoi(argv[2]);
23
24     #pragma omp parallel if(n>4) default(none) \
25         private(sumalocal,tid) shared(a,suma,n) num_threads(x)
26     {
27         sumalocal=0;
28         tid=omp_get_thread_num();
29
30         #pragma omp for private(i) schedule(static) nowait
31         for (i=0; i<n; i++)
32         {
33             sumalocal += a[i];
34             printf(" thread %d suma de a[%d]=%d sumalocal=%d \n",
35                 tid, i, a[i], sumalocal);
36         }
37
38         #pragma omp atomic
39         suma += sumalocal;
40
41         #pragma omp barrier
42
43         #pragma omp master
44         printf("thread master=%d imprime suma=%d\n", tid, suma);
45     }
46 }

```

#### CAPTURAS DE PANTALLA:

```

albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/BP3
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertollamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3] 2021-05-22 sábado
$cd ejer1
[AlbertollamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer1] 2021-05-22 sábado
$gcc -O2 -fopenmp -o if-clauseModificado if-clauseModificado.c
[AlbertollamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer1] 2021-05-22 sábado
$./if-clauseModificado 10 5
thread 1 suma de a[2]=2 sumalocal=2
thread 1 suma de a[3]=3 sumalocal=5
thread 4 suma de a[8]=8 sumalocal=8
thread 4 suma de a[9]=9 sumalocal=17
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 3 suma de a[6]=6 sumalocal=6
thread 3 suma de a[7]=7 sumalocal=13
thread 2 suma de a[5]=5 sumalocal=9
thread master=0 imprime suma=45
[AlbertollamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer1] 2021-05-22 sábado
$./if-clauseModificado 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
[AlbertollamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer1] 2021-05-22 sábado
$

```

**RESPUESTA:** Se han realizado dos ejecuciones del programa `if-clause.c` con un segundo parámetro distinto correspondiente al número de hebras que ejecutan el bucle. Como vemos el resultado de la suma se mantiene igual para ambas ejecuciones debido a que el primer parámetro es  $> 4$ .

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

Iteración	"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	1	0	0	1	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	0
2	1	0	0	1	0	0	0	0	1	0	0	0
3	1	0	0	1	0	0	0	0	1	0	0	0
4	1	0	0	1	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	0	0	1	0	1	1
6	0	2	0	0	2	0	0	1	0	0	1	1
7	0	2	0	0	2	0	0	1	0	0	0	1
8	0	2	1	0	2	1	0	1	2	0	0	1
9	0	2	1	0	2	1	0	1	2	0	0	1

10	0	1	1	0	1	1	0	1	2	1	1	1
11	2	1	1	2	1	1	0	1	2	1	1	0
12	2	1	2	2	1	2	0	2	1	1	2	2
13	2	1	2	2	1	2	0	2	1	1	2	2
14	2	1	2	2	1	2	1	0	1	2	2	2
15	2	1	2	2	1	2	2	0	1	2	2	2

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:

*Static* divide las iteraciones en un número igual al chunk pasado como parámetro y se van asignando en round robin. *Dynamic* divide las iteraciones de la misma forma, pero la asignación se hace en tiempo de ejecución es decir, se el chunk correspondiente de iteraciones a la hebra que termine antes. *Guided* obliga a que el tamaño mínimo del bloque de iteraciones que realiza una misma hebra sea igual a los chunks pasados como parámetros.

La diferencia entre *monotonic* y *nonmonotonic* es que la primera obliga a que las iteraciones del bucle las realicen la correspondiente hebra en orden lógico creciente. *Nonmonotonic* simplemente no mantiene este orden lógico creciente.

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.

Para `OMP_SCHEDULE = "static"`, los valores son 1 para *modifier* y 0 para *chunk*.

Para `OMP_SCHEDULE = "dynamic"`, los valores son 2 para *modifier* y 1 para *chunk*.

Para `OMP_SCHEDULE = "guided"`, los valores son 3 para *modifier* y 1 para *chunk*.

En el programa anterior he utilizado la función `omp_get_schedule` para obtener dichos valores. Además los únicos argumentos que podemos pasar a la función `omp_set_schedule` para obtener *schedule* como *static*, *dynamic* o *guided*, son 1, 2 ó 3 respectivamente.

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`

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #ifdef _OPENMP
5  #include <omp.h>
6  #else
7  #define omp_get_thread_num() 0
8  #endif
9
10 int main(int argc, char **argv)
11 {
12     int i, n=200, chunk, a[n], suma=0;
13     omp_sched_t kind;
14     int modif;
15     if(argc < 3)
16     {
17         fprintf(stderr, "\nFalta iteraciones o chunk\n");
18         exit(-1);
19     }
20
21     n = atoi(argv[1]);
22     if (n>200)
23         n=200;
24
25     chunk = atoi(argv[2]);
26
27     for (i=0; i<n; i++)
28         a[i] = i;
29
30     #pragma omp parallel for firstprivate(suma) \
31         lastprivate(suma) schedule(dynamic, chunk)
32     for (i=0; i<n; i++)
33     {
34
35         if (i == 0){
36             omp_get_schedule(&kind, &modif);
37             printf("En parallel: dyn-var (true si hay ajuste dinámico)=%s, nthreads-var= %d ,thread-limit-var=%d, run-sched-var= tipo:%d || chunk: %d\n",
38                 omp_get_dynamic()? "true": "false", omp_get_max_threads(), omp_get_thread_limit(), kind, modif);
39
40             }suma = suma + a[i];
41             printf(" thread %d suma a[%d]=%d suma=%d \n",
42                 omp_get_thread_num(), i, a[i], suma);
43         }
44
45     #pragma omp parallel for firstprivate(suma) \
46         lastprivate(suma) schedule(dynamic, chunk)
47     for (i=0; i<n; i++)
48     {
49
50         if (i == 0){
51             omp_get_schedule(&kind, &modif);
52             printf("En parallel: dyn-var (true si hay ajuste dinámico)=%s, nthreads-var= %d ,thread-limit-var=%d, run-sched-var= tipo:%d || chunk: %d\n",
53                 omp_get_dynamic()? "true": "false", omp_get_max_threads(), omp_get_thread_limit(), kind, modif);
54
55             }suma = suma + a[i];
56             printf(" thread %d suma a[%d]=%d suma=%d \n",
57                 omp_get_thread_num(), i, a[i], suma);
58         }
59
60     printf("Fuera de 'parallel for' suma=%d\n", suma);
61     printf("Además, dyn-var (true si hay ajuste dinámico)=%s, nthreads-var= %d ,thread-limit-var=%d, run-sched-var= tipo:%d || chunk: %d\n",
62         omp_get_dynamic()? "true": "false", omp_get_max_threads(), omp_get_thread_limit(), kind, modif);
63 }

```

## CAPTURAS DE PANTALLA:

## En la primera ejecución cambio la asignación dinámica, activándola: `export OMP_DYNAMIC=TRUE`

```
albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/BP3
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$gcc -O2 -fopenmp -o scheduled-clauseModificado scheduled-clauseModificado.c
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$export OMP_DYNAMIC=TRUE
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$./scheduled-clauseModificado 10 2
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
En parallel: dyn-var (true si hay ajuste dinámico)=true, nthreads-var= 4 ,thread-limit-var=2147483647, run-sched-var= tipo:2 || chunk: 1
thread 3 suma a[6]=6 suma=6
thread 3 suma a[7]=7 suma=13
thread 2 suma a[0]=0 suma=0
thread 2 suma a[1]=1 suma=1
thread 1 suma a[8]=8 suma=13
thread 1 suma a[9]=9 suma=22
thread 0 suma a[4]=4 suma=4
thread 0 suma a[5]=5 suma=9
Fuera de 'parallel for' suma=22
Además, dyn-var (true si hay ajuste dinámico)=true, nthreads-var= 4 ,thread-limit-var=2147483647, run-sched-var= tipo:2 || chunk: 1
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$
```

## En la segunda ejecución cambio el número de threads usados, con `export OMP_NUM_THREADS`

```
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$export OMP_NUM_THREADS=2
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$./scheduled-clauseModificado 10 2
En parallel: dyn-var (true si hay ajuste dinámico)=true, nthreads-var= 2 ,thread-limit-var=2147483647, run-sched-var= tipo:2 || chunk: 1
thread 0 suma a[0]=0 suma=0
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 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 0 suma a[9]=9 suma=40
Fuera de 'parallel for' suma=40
Además, dyn-var (true si hay ajuste dinámico)=true, nthreads-var= 2 ,thread-limit-var=2147483647, run-sched-var= tipo:2 || chunk: 1
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$
```

## En la tercera ejecución cambio la planificación, `export OMP_SCHEDULE="static, 4"`

```
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$export OMP_SCHEDULE="static,4"
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$./scheduled-clauseModificado 10 2
En parallel: dyn-var (true si hay ajuste dinámico)=true, nthreads-var= 2 ,thread-limit-var=2147483647, run-sched-var= tipo:1 || chunk: 4
thread 0 suma a[0]=0 suma=0
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[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
Fuera de 'parallel for' suma=40
Además, dyn-var (true si hay ajuste dinámico)=true, nthreads-var= 2 ,thread-limit-var=2147483647, run-sched-var= tipo:1 || chunk: 4
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer4] 2021-05-22 sábado
$
```

**RESPUESTA:** Los valores no cambian entre la zona paralela y la zona no paralela, y esto es porque cada variable de control es general al sistema donde se está ejecutando la tarea. Cuando modificamos una variable de entorno, modificamos la variable de control que está asignada a dicha de entorno.

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

```

14 int main()
15 {
16     if (argc < 3)
17     {
18         fprintf(stderr, "\nFalta iteraciones o chunk\n");
19         exit(-1);
20     }
21     n = atoi(argv[1]);
22     if (n > 200)
23         n = 200;
24     chunk = atoi(argv[2]);
25     for (i = 0; i < n; i++)
26         a[i] = i;
27
28     #pragma omp parallel for firstprivate(suma) \
29     lastprivate(suma) schedule(dynamic, chunk)
30     for (i = 0; i < n; i++)
31     {
32         if (i == 0)
33         {
34             omp_get_schedule(&kind, &modif);
35             printf("Dentro del FOR \n Numero de threads: %d, \nNumero de procesadores disponibles: %d\n",
36                   "Paralelo? %s\n", omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel() ? "true" : "false");
37         }
38         suma = suma + a[i];
39         printf(" thread %d suma a[%d]=%d suma=%d \n",
40               omp_get_thread_num(), i, a[i], suma);
41     }
42     printf("Fuera de 'parallel for' suma=%d\n", suma);
43     printf("Fuera del FOR \n Numero de threads: %d, \nNumero de procesadores disponibles: %d\n",
44           "Paralelo? %s\n", omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel() ? "true" : "false");
45 }

```

**CAPTURAS DE PANTALLA:**

```

[AlbetoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3] 2021-05-22 sábado
$ cd ejer5
[AlbetoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer5] 2021-05-22 sábado
$ gcc -O2 -fopenmp -o scheduled-clauseModificado4 scheduled-clauseModificado4.c
[AlbetoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer5] 2021-05-22 sábado
$ ./scheduled-clauseModificado4 10 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 1 suma a[7]=7 suma=27
thread 1 suma a[8]=8 suma=35
thread 1 suma a[9]=9 suma=44
Dentro del FOR
Numero de threads: 2,
Numero de procesadores disponibles: 4
Paralelo? true
thread 0 suma a[0]=0 suma=0
thread 0 suma a[1]=1 suma=1
Fuera de 'parallel for' suma=44
Fuera del FOR
Numero de threads: 1,
Numero de procesadores disponibles: 4
Paralelo? false
[AlbetoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer5] 2021-05-22 sábado

```

**RESPUESTA:** Se obtienen valores distintos en el número de threads que intervienen, ya que en una parte obtiene el valor de los threads creados en la paralelización, y fuera del parallel solo obtiene el número 1 (única hebra que ejecuta dicha parte del código, la hebra master). Num-procs es idéntico en ambas partes ya que en eso no influye la paralelización, y la variable in-parallel nos informa si estamos en región paralela, por lo cual, dentro devuelve un 1 y fuera un 0.

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

```

C pmtv-secuencial x
ejer7 > C pmtv-secuencial > ...
43
44 // Inicialización de matriz, mat
45 for (i = 0; i < tam; i++)
46     for (j = 0; j < tam; j++)
47         if (j < i)
48             mat[i][j] = 0;
49         else
50             mat[i][j] = 1;
51
52 // Inicialización de vector, v
53 for (i = 0; i < tam; i++)
54     v[i] = i;
55
56 // Medición de tiempos
57 t = omp_get_wtime();
58
59 // Cálculo de mat*v, res
60 for (i = 0; i < tam; i++)
61 {
62     suma = 0;
63
64     for (j = i; j < tam; j++)
65         suma += mat[i][j] * v[j];
66
67     res[i] = suma;
68 }
69
70 // Medición de tiempos
71 t = omp_get_wtime() - t;
72
73 // Impresión de tiempo de ejecución
74 printf("Tiempo (seg): %f\n", t);
75
76 // Impresión de resultado
77 printf("\n\n\t\t\t\t\tResultado:\n");
78 if (tam < 20)
79     for (i = 0; i < tam; i++)
80         printf("res[%d]=%f ", i, res[i]);
81 else
82     printf("res[0]=%f res[%d]=%f", res[0], tam - 1, res[tam - 1]);
83 printf("\n");
84
85 // Liberación de memoria

```

```

scheduled-clauseModificado5.c - BP3 - Visual Studio Code
C scheduled-clauseModificado5.c 2 X
ejers > C scheduled-clauseModificado5.c > @ main(int, char **)
26
27 for (i = 0; i < n; i++)
28     a[i] = i;
29
30 #pragma omp parallel for firstprivate(suma) \
31     lastprivate(suma) schedule(dynamic, chunk)
32 for (i = 0; i < n; i++)
33 {
34     suma = suma + a[i];
35     printf(" thread %d suma a[%d]=%d suma=%d \n",
36           omp_get_thread_num(), i, a[i], suma);
37
38     if (i == 0)
39     {
40         omp_get_schedule(&kind, &modif);
41         printf("Dentro del for: dyn-var (true si hay ajuste dinámico)=%s, nthreads-var= %d, run-sched-var= tipo:%d || chunk: %d\n",
42               omp_get_dynamic() ? "true" : "false", omp_get_max_threads(), kind, modif);
43
44         printf("\nModificando dyn_var ---> false");
45         omp_set_dynamic(0);
46
47         printf("\nModificando nthreads-var ---> 2");
48         omp_set_num_threads(7);
49
50         printf("\nModificando run-sched-var ---> dynamic, 3");
51         omp_set_schedule(omp_sched_dynamic, 3);
52     }
53
54     omp_get_schedule(&kind, &modif);
55     printf("Fuera del for: dyn-var (true si hay ajuste dinámico)=%s, nthreads-var= %d, run-sched-var= tipo:%d || chunk: %d\n",
56           omp_get_dynamic() ? "true" : "false", omp_get_max_threads(), kind, modif);
57     printf("Fuera de 'parallel for' suma=%d\n", suma);
58 }
59
60

```

### CAPTURAS DE PANTALLA:

```

[AlbertoLlomasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer6] 2021-05-22 sábado
$gcc -O2 -fopenmp -o scheduled-clauseModificado5 scheduled-clauseModificado5.c
[AlbertoLlomasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer6] 2021-05-22 sábado
$./scheduled-clauseModificado5 10 2
 thread 0 suma a[0]=0 suma=0
Dentro del for: dyn-var (true si hay ajuste dinámico)=true, nthreads-var= 2, run-sched-var= tipo:1 || chunk: 4
Modificando dyn_var ---> false
Modificando nthreads-var ---> 2
Modificando run-sched-var ---> dynamic, 3 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[2]=2 suma=2
 thread 1 suma a[3]=3 suma=5
Fuera del for: dyn-var (true si hay ajuste dinámico)=true, nthreads-var= 2, run-sched-var= tipo:1 || chunk: 4
Fuera de 'parallel for' suma=40
[AlbertoLlomasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/BP3/ejer6] 2021-05-22 sábado
$

```

**RESPUESTA:** El resultado en la región paralela y fuera de ella es el mismo ya que hemos modificado los valores fuera y dentro.

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 pmtv-secuencial x
ejer7 > C pmtv-secuencial > ...
43
44 // Inicialización de matriz, mat
45 for (i = 0; i < tam; i++)
46     for (j = 0; j < tam; j++)
47         if (j < i)
48             mat[i][j] = 0;
49         else
50             mat[i][j] = 1;
51
52 // Inicialización de vector, v
53 for (i = 0; i < tam; i++)
54     v[i] = i;
55
56 // Medición de tiempos
57 t = omp_get_wtime();
58
59 // Cálculo de mat*v, res
60 for (i = 0; i < tam; i++)
61 {
62     suma = 0;
63
64     for (j = i; j < tam; j++)
65         suma += mat[i][j] * v[j];
66
67     res[i] = suma;
68 }
69
70 // Medición de tiempos
71 t = omp_get_wtime() - t;
72
73 // Impresión de tiempo de ejecución
74 printf("Tiempo (seg): %f\n", t);
75
76 // Impresión de resultado
77 printf("\n\n\t\t\t\t\tResultado:\n");
78 if (tam < 20)
79     for (i = 0; i < tam; i++)
80         printf("res[%d]=%f ", i, res[i]);
81 else
82     printf("res[0]=%f res[%d]=%f", res[0], tam - 1, res[tam - 1]);
83 printf("\n");
84
85 // Liberación de memoria

```

```

c pmtv-secuencial.c x
ejer7 > c pmtv-secuencial.c ...
64     for (j = 1; j < tam; j++)
65         suma += mat[i][j] * v[j];
66
67     res[i] = suma;
68 }
69
70 // Medición de tiempos
71 t = omp_get_wtime() - t;
72
73 // Impresión de tiempo de ejecución
74 printf("Tiempo (seg): %f\n", t);
75
76 // Impresión de resultado
77 printf("\n          \nResultado:\n");
78 if (tam < 20)
79     for (i = 0; i < tam; i++)
80         printf("res[%d]=%f ", i, res[i]);
81 else
82     printf("res[0]=%f res[%d]=%f", res[0], tam - 1, res[tam - 1]);
83 printf("\n");
84
85 // Liberación de memoria
86 free(v);
87 free(res);
88 for (i = 0; i < tam; i++)
89     free(mat[i]);
90 free(mat);
91
92 return 0;
93 }
94

```

## CAPTURAS DE PANTALLA:

```
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer7] 2021-05-22 sábado
$gcc -O2 -fopenmp pmtv-secuencial.c -o pmtv-secuencial -lrt
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer7] 2021-05-22 sábado
$sruncpac ./pmtv-secuencial 10000
Tiempo (seg): 0.072567

Resultado:
res[0]=49995000.000000 res[9999]=9999.000000
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer7] 2021-05-22 sábado
$
```

**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`

```

C: pmtv-OpenMPC x
ejers > C: pmtv-OpenMPC > ...
6 #include <omp.h>
7
8 int main(int argc, char **argv)
9 {
10     if (argc < 2)
11     {
12         printf("[ERROR]-Debe insertar tamaño de matriz y vector\n");
13         exit(-1);
14     }
15
16     unsigned int tam = atoi(argv[1]);
17     double **mat, *v, *res;
18     omp_sched_t kind;
19     int chunk_size;
20
21     // Reserva de espacio
22
23     v = (double *)malloc(tam * sizeof(double));
24     res = (double *)malloc(tam * sizeof(double));
25     mat = (double **)malloc(tam * sizeof(double *));
26
27     if ((v == NULL) || (res == NULL) || (mat == NULL))
28     {
29         printf("[ERROR]-Reserva para vectores\n");
30         exit(-2);
31     }
32
33     int i, j;
34     double suma, t;
35
36     for (i = 0; i < tam; i++)
37     {
38         mat[i] = (double *)malloc(tam * sizeof(double));
39         if (mat[i] == NULL)
40         {
41             printf("[ERROR]-Reserva para matriz\n");
42             exit(-2);
43         }
44     }
45
46     // Inicialización de matriz, mat
47     for (i = 0; i < tam; i++)
48         for (j = 0; j < tam; j++)
49             mat[i][j] = 0;
50
51     // Inicialización de vector, v
52     for (i = 0; i < tam; i++)
53         v[i] = i;
54
55     // Cálculo de mat*v, res
56     #pragma omp parallel shared(mat, v, res) private(i, j)
57     {
58         #pragma omp single
59         {
60             omp_get_schedule(&kind, &chunk_size);
61             printf("run-sched-var: (Kind: %d, Modifier: %d) \n", kind, chunk_size);
62
63             //Medida de tiempo
64             t = omp_get_wtime();
65
66             #pragma omp for schedule(runtime)
67             for (i = 0; i < tam; i++)
68             {
69                 suma = 0;
70
71                 for (j = 1; j < tam; j++)
72                     suma += mat[i][j] * v[j];
73
74                 res[i] = suma;
75             }
76
77             // Medida de tiempo
78             #pragma omp single
79             {
80                 t = omp_get_wtime() - t;
81             }
82
83             // Impresión de tiempo de ejecución
84         }
85     }
86
87     // Impresión de tiempo de ejecución
88 }

```

```
C pmtv-OpenMP.c x
ejers > C pmtv-OpenMP.c > ...

80
81 // Medida de tiempo
82 #pragma omp single
83 {
84     t = omp_get_wtime() - t;
85 }
86
87 // Impresión de tiempo de ejecución
88 printf("Tiempo (seg): %f\n", t);
89
90 // Impresión de resultado
91 printf("\n\nResultado:\n");
92 if (tam < 20)
93     for (i = 0; i < tam; i++)
94         printf("res[%d]=%f ", i, res[i]);
95 else
96     printf("res[0]=%f res[%d]=%f", res[0], tam - 1, res[tam - 1]);
97     printf("\n");
98
99 // Liberación de memoria
100 free(v);
101 free(res);
102 for (i = 0; i < tam; i++)
103     free(mat[i]);
104 free(mat);
105
106 return 0;
107 }
108
109
```

### CAPTURAS DE PANTALLA:

```

C# FUENTES DE PARTIDA:
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer8] 2021-05-22 sábado
$gcc -O2 -fopenmp pmtv-OpenMP.c -o pmtv-OpenMP -lrt
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer8] 2021-05-22 sábado
$export OMP_SCHEDULE="guided"
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer8] 2021-05-22 sábado
$srunc -pac ./pmtv-OpenMP 100
run-sched-var: (Kind: 3, Modifier: 1)
Tiempo (seg): 0.000019

Resultado:
res[0]=4950.000000 res[99]=99.000000
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer8] 2021-05-22 sábado
$export OMP_SCHEDULE="dynamic"
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer8] 2021-05-22 sábado
$srunc -pac ./pmtv-OpenMP 100
run-sched-var: (Kind: 2, Modifier: 1)
Tiempo (seg): 0.000094

Resultado:
res[0]=4950.000000 res[99]=99.000000
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer8] 2021-05-22 sábado
$export OMP_SCHEDULE="static"
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer8] 2021-05-22 sábado
$srunc -pac ./pmtv-OpenMP 100
run-sched-var: (Kind: -2147483647, Modifier: 0)
Tiempo (seg): 0.000018

Resultado:
res[0]=4950.000000 res[99]=99.000000
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp3/ejer8] 2021-05-22 sábado
$

```

**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:** El comportamiento de static ha sido definido anteriormente, el número de instrucciones se distribuye en round-robin a cada thread, en bloques de tamaño del chunk.

**(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:** En el caso de dynamic y guided, esto varía. Es muy probable que un thread acumule varias iteraciones por terminar el primero.

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

**RESPUESTA:** La que mejores prestaciones ofrece es la asignación *guided* ya que el tamaño de bloque (no iteraciones que restan dividido por no threads) va menguando lo cual lleva a que haya una mayor y mejor distribución del trabajo entre las hebras, trabajando todas por igual lo que evita que hebras se sobrecarguen.

**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.

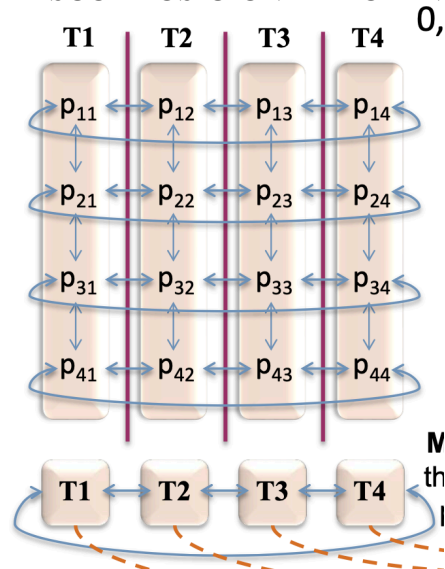
### CAPTURA CÓDIGO FUENTE: pmtv-OpenMP.c

```

C pmtv-OpenMP.c ejer10 x pmtv-OpenMP.c ejer10 x pmtv-OpenMP_atcgrid.sh
ejer10 > C pmtv-OpenMP.c > ...
43 // Inicialización de matriz, mat
44 for (i = 0; i < tam; i++)
45     for (j = 0; j < tam; j++)
46         if (j < i)
47             mat[i][j] = 0;
48         else
49             mat[i][j] = 1;
50
51 // Inicialización de vector, v
52 for (i = 0; i < tam; i++)
53     v[i] = i;
54
55 // Cálculo de mat*v, res
56 #pragma omp parallel shared(mat, v, res) private(i, j)
57 {
58     #pragma omp single
59     {
60         omp_get_schedule(&kind, &chunk_size);
61         printf("run-sched-var: (Kind: %d, Modifier: %d) \n", kind, chunk_size);
62
63         //Medida de tiempo
64         t = omp_get_wtime();
65     }
66
67     #pragma omp for schedule(runtime)
68     for (i = 0; i < tam; i++)
69     {
70         suma = 0;
71
72         for (j = i; j < tam; j++)
73             suma += mat[i][j] * v[j];
74
75         res[i] = suma;
76     }
77
78 // Medida de tiempo
79 #pragma omp single
80 {
81     t = omp_get_wtime() - t;
82 }
83
84 // Impresión de tiempo de ejecución
85

```

### DESCOMPOSICIÓN DE DOMINIO:



**CAPTURAS DE PANTALLA:**

```

$gcc -O2 -fopenmp -o pmtv-OpenMP pmtv-OpenMP.c -lrt
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp3/ejer10] 2021-05-23 domingo
$sbatch -pac pmtv-OpenMP_atcgrid.sh
Submitted batch job 107320
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp3/ejer10] 2021-05-23 domingo
$ls
pmtv-OpenMP pmtv-OpenMP_atcgrid.sh pmtv-OpenMP.c slurm-107319.out slurm-107320.out
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp3/ejer10] 2021-05-23 domingo
$cat slurm-107320.out
run-sched-var: (Kind: -2147483647, Modifier: 0)
Tiempo (seg): 0.304225

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
run-sched-var: (Kind: -2147483647, Modifier: 1)
Tiempo (seg): 0.295140

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
run-sched-var: (Kind: -2147483647, Modifier: 64)
Tiempo (seg): 0.293513

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
run-sched-var: (Kind: -2147483646, Modifier: 1)
Tiempo (seg): 0.294513

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
run-sched-var: (Kind: -2147483646, Modifier: 1)
Tiempo (seg): 0.294490

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp3/ejer10] 2021-05-23 domingo
$

```

```

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
run-sched-var: (Kind: -2147483646, Modifier: 1)
Tiempo (seg): 0.294490

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
run-sched-var: (Kind: -2147483646, Modifier: 64)
Tiempo (seg): 0.293837

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
run-sched-var: (Kind: -2147483645, Modifier: 1)
Tiempo (seg): 0.305116

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
run-sched-var: (Kind: -2147483645, Modifier: 1)
Tiempo (seg): 0.305718

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
run-sched-var: (Kind: -2147483645, Modifier: 64)
Tiempo (seg): 0.304766

Resultado:
res[0]=243752160.000000 res[22079]=22079.000000
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp3/ejer10] 2021-05-23 domingo
$

```

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

```

C pmtv-OpenMP.c ejer8  C pmtv-OpenMP.c ejer10  pmtv-OpenMP_atcgrid.sh X
ejer10 > pmtv-OpenMP_atcgrid.sh
1  #!/bin/bash
2
3  export OMP_SCHEDULE="monotonic:static"
4  srun -pac ./pmtv-OpenMP 22080
5  export OMP_SCHEDULE="monotonic:static, 1"
6  srun -pac ./pmtv-OpenMP 22080
7  export OMP_SCHEDULE="monotonic:static, 64"
8  srun -pac ./pmtv-OpenMP 22080
9
10
11 export OMP_SCHEDULE="monotonic:dynamic"
12 srun -pac ./pmtv-OpenMP 22080
13 export OMP_SCHEDULE="monotonic:dynamic, 1"
14 srun -pac ./pmtv-OpenMP 22080
15 export OMP_SCHEDULE="monotonic:dynamic, 64"
16 srun -pac ./pmtv-OpenMP 22080
17
18
19 export OMP_SCHEDULE="monotonic:guided"
20 srun -pac ./pmtv-OpenMP 22080
21 export OMP_SCHEDULE="monotonic:guided, 1"
22 srun -pac ./pmtv-OpenMP 22080
23 export OMP_SCHEDULE="monotonic:guided, 64"
24 srun -pac ./pmtv-OpenMP 22080
25

```

**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			