

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): Juan Valentín Guerrero Cano

Grupo de prácticas: 3

Fecha de entrega: 18/05/21

Fecha evaluación en clase: 19/05/21

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

CAPTURAS DE PANTALLA:

```

e3estudiante11@atcgrid:~/bp3/ejer1
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer1] 2021-05-16 domingo
$srunc -pac ./if-clauseModificado 10 5
thread 1 suma de a[2]=2 sumalocal=2
thread 1 suma de a[3]=3 sumalocal=5
thread 2 suma de a[4]=4 sumalocal=4
thread 0 suma de a[0]=0 sumalocal=0
thread 2 suma de a[5]=5 sumalocal=9
thread 0 suma de a[1]=1 sumalocal=1
thread 3 suma de a[6]=6 sumalocal=6
thread 4 suma de a[8]=8 sumalocal=8
thread 3 suma de a[7]=7 sumalocal=13
thread 4 suma de a[9]=9 sumalocal=17
thread master=0 imprime suma=45
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer1] 2021-05-16 domingo
$srunc -pac ./if-clauseModificado 10 2
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 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 master=0 imprime suma=45
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer1] 2021-05-16 domingo
$

```

RESPUESTA:

En la captura podemos observar como en dos ejecuciones a las que le hemos pasado un segundo parámetro distinto (a una 5 y a otra 2) el número de hebras que ejecutan el bucle es 5 y 2 respectivamente, dando la suma (variable global) el mismo valor independientemente de el número de threads fijadas. El resultado de suma se mantiene igual en ambas ejecuciones porque el primer parámetro introducido es mayor que 4 ($n > 4$).

2. Rellenar la Tabla 1 (se debe poner en la tabla el id del *thread* que ejecuta cada iteración) usando `schedulerc -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=1	x=2	x=1	x=2	x=1	x=2
0	0	0	0	0	0	2	1	0
1	1	0	1	0	2	2	1	0
2	2	1	2	1	1	0	1	0
3	0	1	0	1	2	0	1	0
4	1	2	1	2	2	0	1	0
5	2	2	2	2	2	0	1	0
6	0	0	0	0	0	0	2	2
7	1	0	1	0	2	0	2	2
8	2	1	2	1	0	2	2	2
9	0	1	0	1	0	2	2	2
10	1	2	1	2	0	0	0	0
11	2	2	2	2	0	0	0	0
12	0	0	0	0	0	2	1	2
13	1	0	1	0	0	2	1	2
14	2	1	2	1	0	0	2	0
15	0	1	0	1	0	0	2	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:

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 (en la segunda columna de la tabla la primer hebra realiza las primeras "chunks" iteraciones, en ese caso 2, mientras que la siguiente hebra en orden lógico creciente realiza siguientes dos iteraciones). 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`

```

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; int chunk_size;
14     if(argc < 3)
15     {
16         fprintf(stderr, "\nFalta iteraciones o chunk\n");
17         exit(-1);
18     }
19
20     n = atoi(argv[1]);
21     if (n>200)
22         n=200;
23
24     chunk = atoi(argv[2]);
25
26     for (i=0; i<n; i++)
27         a[i] = i;
28
29     #pragma omp parallel
30     {
31         #pragma omp for firstprivate(suma) \
32             lastprivate(suma) schedule(dynamic, chunk)
33         for (i=0; i<n; i++)
34         {
35             suma = suma + a[i];
36             printf(" thread %d suma a[%d]=%d suma=%d \n",
37                 omp_get_thread_num(), i, a[i], suma);
38         }
39
40         #pragma omp single
41         {
42             omp_get_schedule(&kind, &chunk_size);
43             printf("Valor de las variables de control dyn-var: %d, nthreads-var: %d, thread-limit-var: %d y run-sched-var: (%d, %d) dentro de parallel:\n",
44                 omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), kind, chunk_size);
45         }
46     }
47
48     printf("Fuera de 'parallel for' suma=%d\n", suma);
49
50     omp_get_schedule(&kind, &chunk_size);
51     printf("Valor de las variables de control dyn-var: %d, nthreads-var: %d, thread-limit-var: %d y run-sched-var: (%d, %d) fuera de parallel:\n",
52         omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), kind, chunk_size);
53
54 }
55

```

CAPTURAS DE PANTALLA:

```

e3estudiante11@atcgrid:~/bp3/ejer4
Archivo Editar Ver Buscar Terminal Ayuda
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$run -pac ./scheduled-clauseModificado 10 2
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 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 1 suma a[6]=6 suma=11
thread 1 suma a[7]=7 suma=18
thread 0 suma a[5]=5 suma=10
thread 1 suma a[8]=8 suma=26
thread 1 suma a[9]=9 suma=35
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) dentro de parallel:
Fuera de 'parallel for' suma=35
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) fuera de parallel:
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$run -pac ./scheduled-clauseModificado 10 3
thread 1 suma a[3]=3 suma=3
thread 1 suma a[4]=4 suma=7
thread 1 suma a[5]=5 suma=12
thread 0 suma a[0]=0 suma=0
thread 0 suma a[1]=1 suma=1
thread 0 suma a[2]=2 suma=3
thread 0 suma a[9]=9 suma=12
thread 1 suma a[6]=6 suma=18
thread 1 suma a[7]=7 suma=25
thread 1 suma a[8]=8 suma=33
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) dentro de parallel:
Fuera de 'parallel for' suma=12
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) fuera de parallel:
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$run -pac ./scheduled-clauseModificado 4 2
thread 1 suma a[0]=0 suma=0
thread 1 suma a[1]=1 suma=1
thread 0 suma a[2]=2 suma=2
thread 0 suma a[3]=3 suma=5
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) dentro de parallel:
Fuera de 'parallel for' suma=5
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) fuera de parallel:
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$

```

```

e3estudiante11@atcgrid:~/bp3/ejer4
Archivo Editar Ver Buscar Terminal Ayuda
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$export OMP_NUM_THREADS=4
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$run -pac ./scheduled-clauseModificado 4 2
thread 1 suma a[0]=0 suma=0
thread 1 suma a[1]=1 suma=1
thread 0 suma a[2]=2 suma=2
thread 0 suma a[3]=3 suma=5
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) dentro de parallel:
Fuera de 'parallel for' suma=5
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) fuera de parallel:
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$export OMP_SCHEDULE="dynamic"
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$run -pac ./scheduled-clauseModificado 4 2
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 0 suma a[0]=0 suma=0
thread 0 suma a[1]=1 suma=1
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) dentro de parallel:
Fuera de 'parallel for' suma=5
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) fuera de parallel:
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$export OMP_SCHEDULE="guided"
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$run -pac ./scheduled-clauseModificado 4 2
thread 1 suma a[0]=0 suma=0
thread 1 suma a[1]=1 suma=1
thread 0 suma a[2]=2 suma=2
thread 0 suma a[3]=3 suma=5
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (3, 1) dentro de parallel:
Fuera de 'parallel for' suma=5
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (3, 1) fuera de parallel:
[juanvalentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer4] 2021-05-17 lunes
$

```

RESPUESTA:

Sí, se imprimen los mismos valores debido a que una vez los valores se asignaron en la región paralela, al imprimirlos en la región no paralela estos valores se mantienen y no cambian.

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

```

13  omp_sched_t kind; int chunk_size;
14  if(argc < 3)
15  {
16      fprintf(stderr, "\nFalta iteraciones o chunk\n");
17      exit(-1);
18  }
19
20  n = atoi(argv[1]);
21  if (n>200)
22      n=200;
23
24  chunk = atoi(argv[2]);
25
26  for (i=0; i<n; i++)
27      a[i] = i;
28
29  #pragma omp parallel
30  {
31      #pragma omp for firstprivate(suma) \
32          lastprivate(suma) schedule(dynamic, chunk)
33      for (i=0; i<n; i++)
34      {
35          suma = suma + a[i];
36          printf(" thread %d suma a[%d]=%d suma=%d \n",
37              omp_get_thread_num(), i, a[i], suma);
38      }
39      #pragma omp single
40      {
41          printf("Dentro de region paralela:")
42          omp_get_schedule(&kind, &chunk_size);
43          printf("Valor de las variables de control dyn-var: %d, nthreads-var: %d, thread-limit-var: %d y run-sched-var: (%d, %d)\n",
44              omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), kind, chunk_size);
45
46          printf("Valor de num-threads: %d, de num-procs: %d, de in_parallel: %d\n"
47              omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
48      }
49  }
50
51
52  printf("Fuera de 'parallel for' suma=%d\n", suma);
53  omp_get_schedule(&kind, &chunk_size);
54  printf("Valor de las variables de control dyn-var: %d, nthreads-var: %d, thread-limit-var: %d y run-sched-var: (%d, %d) fuera de parallel:\n",
55      omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), kind, chunk_size);
56
57  printf("Valor de num-threads: %d, de num-procs: %d, de in_parallel: %d\n"
58      omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
59
60  }
61

```

CAPTURAS DE PANTALLA:

```

e3estudiante11@atcgrid:~/bp3/ejer5
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer5] 2021-05-17 lunes
$srunc -pac ./scheduled-clauseModificado4 10 3
thread 1 suma a[0]=0 suma=0
thread 1 suma a[1]=1 suma=1
thread 1 suma a[2]=2 suma=3
thread 1 suma a[6]=6 suma=9
thread 1 suma a[7]=7 suma=16
thread 1 suma a[8]=8 suma=24
thread 1 suma a[9]=9 suma=33
thread 0 suma a[3]=3 suma=3
thread 0 suma a[4]=4 suma=7
thread 0 suma a[5]=5 suma=12
Dentro de region paralela:Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1)
Valor de num-threads: 2, de num-procs: 2, de in_parallel: 1
Fuera de 'parallel for' suma=33
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (2, 1) fuera de parallel:
Valor de num-threads: 1, de num-procs: 2, de in_parallel: 0
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer5] 2021-05-17 lunes
$export OMP_SCHEDULE="guided"
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer5] 2021-05-17 lunes
$srunc -pac ./scheduled-clauseModificado4 10 3
thread 1 suma a[3]=3 suma=3
thread 1 suma a[4]=4 suma=7
thread 1 suma a[5]=5 suma=12
thread 0 suma a[0]=0 suma=0
thread 0 suma a[1]=1 suma=1
thread 0 suma a[2]=2 suma=3
thread 1 suma a[6]=6 suma=18
thread 1 suma a[7]=7 suma=25
thread 1 suma a[8]=8 suma=33
thread 0 suma a[9]=9 suma=12
Dentro de region paralela:Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (3, 1)
Valor de num-threads: 2, de num-procs: 2, de in_parallel: 1
Fuera de 'parallel for' suma=12
Valor de las variables de control dyn-var: 0, nthreads-var: 2, thread-limit-var: 2147483647 y run-sched-var: (3, 1) fuera de parallel:
Valor de num-threads: 1, de num-procs: 2, de in_parallel: 0
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer5] 2021-05-17 lunes
$

```

```

e3estudiante11@atcgrid:~/bp3/ejer5
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer5] 2021-05-17 lunes
$export OMP_NUM_THREADS=8
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer5] 2021-05-17 lunes
$srunc -pac4 ./scheduled-clauseModificado4 10 3
thread 4 suma a[0]=0 suma=0
thread 4 suma a[1]=1 suma=1
thread 4 suma a[2]=2 suma=3
thread 1 suma a[3]=3 suma=3
thread 1 suma a[4]=4 suma=7
thread 1 suma a[5]=5 suma=12
thread 5 suma a[9]=9 suma=9
thread 3 suma a[6]=6 suma=6
thread 3 suma a[7]=7 suma=13
thread 3 suma a[8]=8 suma=21
Dentro de region paralela:Valor de las variables de control dyn-var: 0, nthreads-var: 8, thread-limit-var: 2147483647 y run-sched-var: (3, 1)
Valor de num-threads: 8, de num-procs: 2, de in_parallel: 1
Fuera de 'parallel for' suma=9
Valor de las variables de control dyn-var: 0, nthreads-var: 8, thread-limit-var: 2147483647 y run-sched-var: (3, 1) fuera de parallel:
Valor de num-threads: 1, de num-procs: 2, de in_parallel: 0
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer5] 2021-05-17 lunes
$

```

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 cuál, 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`

```

15  {
16      fprintf(stderr, "\nFalta iteraciones o chunk\n");
17      exit(-1);
18  }
19
20  n = atoi(argv[1]);
21  if (n>200)
22      n=200;
23
24  chunk = atoi(argv[2]);
25
26  for (i=0; i<n; i++)
27      a[i] = i;
28
29  #pragma omp parallel
30  {
31      #pragma omp for firstprivate(suma) \
32          lastprivate(suma) schedule(dynamic, chunk)
33      for (i=0; i<n; i++)
34      {
35          suma = suma + a[i];
36          printf(" thread %d suma a[%d]=%d suma=%d \n",
37              omp_get_thread_num(), i, a[i], suma);
38      }
39
40      #pragma omp single
41      {
42          omp_set_dynamic(1);
43          omp_set_num_threads(4);
44          omp_set_schedule(3, chunk);
45
46          printf("Dentro de region paralela:");
47          omp_get_schedule(&kind, &chunk);
48          printf("Valor de las variables de control dyn-var: %d, nthreads-var: %d, thread-limit-var: %d y run-sched-var: (%d, %d)\n",
49              omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), kind, chunk);
50      }
51  }
52
53  }
54
55  omp_set_dynamic(1);
56  omp_set_num_threads(4);
57  omp_set_schedule(3, chunk);
58
59  printf("Fuera de 'parallel for' suma=%d\n", suma);
60  omp_get_schedule(&kind, &chunk);
61  printf("Valor de las variables de control dyn-var: %d, nthreads-var: %d, thread-limit-var: %d y run-sched-var: (%d, %d) fuera de parallel:\n",
62      omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit(), kind, chunk);
63
64  }

```

CAPTURAS DE PANTALLA:


```

e3estudiante11@atcgrid:~/bp3/ejer6
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer6] 2021-05-17 lunes
$srunc -pac ./scheduled-clauseModificado6 10 2
thread 1 suma a[0]=0 suma=0
thread 1 suma a[1]=1 suma=1
thread 2 suma a[2]=2 suma=2
thread 2 suma a[3]=3 suma=5
thread 3 suma a[4]=4 suma=4
thread 3 suma a[5]=5 suma=9
thread 4 suma a[6]=6 suma=6
thread 4 suma a[7]=7 suma=13
thread 5 suma a[8]=8 suma=8
thread 5 suma a[9]=9 suma=17
Dentro de region paralela:Valor de las variables de control dyn-var: 1, nthreads-var: 4, thread-limit-var: 2147483647 y run-sched-var: (3, 2)
Fuera de 'parallel for' suma=17
Valor de las variables de control dyn-var: 1, nthreads-var: 4, thread-limit-var: 2147483647 y run-sched-var: (3, 2) fuera de parallel:
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer6] 2021-05-17 lunes
$export OMP_NUM_THREADS=8
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer6] 2021-05-17 lunes
$srunc -pac ./scheduled-clauseModificado6 20 2
thread 2 suma a[2]=2 suma=2
thread 2 suma a[3]=3 suma=5
thread 1 suma a[0]=0 suma=0
thread 5 suma a[8]=8 suma=8
thread 5 suma a[9]=9 suma=17
thread 1 suma a[1]=1 suma=1
thread 1 suma a[10]=10 suma=11
thread 5 suma a[12]=12 suma=29
thread 1 suma a[11]=11 suma=22
thread 5 suma a[13]=13 suma=42
thread 1 suma a[14]=14 suma=36
thread 5 suma a[16]=16 suma=58
thread 1 suma a[15]=15 suma=51
thread 5 suma a[17]=17 suma=75
thread 1 suma a[18]=18 suma=69
thread 1 suma a[19]=19 suma=88
thread 0 suma a[4]=4 suma=4
thread 0 suma a[5]=5 suma=9
thread 2 suma a[6]=6 suma=11
thread 2 suma a[7]=7 suma=18
Dentro de region paralela:Valor de las variables de control dyn-var: 1, nthreads-var: 4, thread-limit-var: 2147483647 y run-sched-var: (3, 2)
Fuera de 'parallel for' suma=88
Valor de las variables de control dyn-var: 1, nthreads-var: 4, thread-limit-var: 2147483647 y run-sched-var: (3, 2) fuera de parallel:
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer6] 2021-05-17 lunes
$

e3estudiante11@atcgrid:~/bp3/ejer6
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer6] 2021-05-17 lunes
$export OMP_SCHEDULE="dynamic"
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer6] 2021-05-17 lunes
$srunc -pac ./scheduled-clauseModificado6 20 2
thread 2 suma a[2]=2 suma=2
thread 2 suma a[3]=3 suma=5
thread 1 suma a[0]=0 suma=0
thread 3 suma a[8]=8 suma=8
thread 3 suma a[9]=9 suma=17
thread 1 suma a[1]=1 suma=1
thread 1 suma a[12]=12 suma=13
thread 4 suma a[14]=14 suma=14
thread 4 suma a[15]=15 suma=29
thread 4 suma a[16]=16 suma=45
thread 1 suma a[13]=13 suma=26
thread 4 suma a[17]=17 suma=62
thread 1 suma a[18]=18 suma=44
thread 1 suma a[19]=19 suma=63
thread 3 suma a[10]=10 suma=27
thread 2 suma a[6]=6 suma=11
thread 0 suma a[4]=4 suma=4
thread 2 suma a[7]=7 suma=18
thread 3 suma a[11]=11 suma=38
thread 0 suma a[5]=5 suma=9
Dentro de region paralela:Valor de las variables de control dyn-var: 1, nthreads-var: 4, thread-limit-var: 2147483647 y run-sched-var: (3, 2)
Fuera de 'parallel for' suma=63
Valor de las variables de control dyn-var: 1, nthreads-var: 4, thread-limit-var: 2147483647 y run-sched-var: (3, 2) fuera de parallel:
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer6] 2021-05-17 lunes
$

```

RESPUESTA:

Como podemos observar pese a que cambiemos las variables de entorno, al utilizar las funciones modificadoras se mantiene la modificación imprimiéndose los valores asignados en el código. He adjuntado varias ejecuciones modificando distintas variables del entorno. 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

```

File Edit Selection View Go Run Terminal Help

C pmtv-secuencial.c X
home > valenting > Escritorio > BP3 > ejer7 > C pmtv-secuencial.c > main(int, char **)
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <omp.h>
4
5  int main(int argc, char ** argv){
6      if (argc<2) {
7          printf("[ERROR]-Debe insertar tamaño de matriz y vector\n");
8          exit(-1);
9      }
10
11     unsigned int tam = atoi(argv[1]);
12     double **mat, *v, *res;
13
14     // Reserva de espacio
15
16     v = (double*) malloc(tam*sizeof(double));
17     res = (double*) malloc(tam*sizeof(double));
18     mat = (double**) malloc(tam*sizeof(double*));
19
20     if ( ( v == NULL ) || ( res == NULL ) || ( mat == NULL ) ){
21         printf("[ERROR]-----Reserva para vectores\n");
22         exit(-2);
23     }
24
25     int i, j;
26     double suma, time;
27
28     for ( i = 0; i < tam; i++){
29         mat[i] = (double*) malloc(tam*sizeof(double));
30         if ( mat[i] == NULL ){
31             printf("[ERROR]-----Reserva para matriz\n");
32             exit(-2);
33         }
34     }
35
36     for ( i = 0; i < tam; i++){
37         for( j = 0; j < tam; j++){
38             mat[i][j] = (i+1)*(j+1);
39         }
40     }
41
42     for ( i = 0; i < tam; i++ )
43         v[i] = i;
44     time = omp_get_wtime();
45
46     for ( i = 0; i < tam; i++ ) {

```

```

36
37
38     for ( i = 0; i < tam; i++ ){
39         for( j = 0; j < tam; j++ ){
40             mat[i][j] = (i+1)*(j+1);
41         }
42     }
43
44
45
46     for ( i = 0; i < tam; i++ )
47     {
48         v[i] = i;
49         time = omp_get_wtime();
50
51         for ( i = 0; i < tam; i++ ) {
52             suma = 0;
53
54             for ( j = i; j < tam; j++ ){
55                 suma += mat[i][j]*v[j];
56             }
57             res[i] = suma;
58         }
59
60         time = omp_get_wtime() - time; //Cálculo del tiempo en que se realiza el producto
61
62         printf("Tiempo: %f s\n", time);
63
64
65         printf("\nResultado:\n");
66         if ( tam < 20 )
67             for ( i = 0; i < tam; i++ )
68                 printf("res[%d]=%f ", i, res[i]);
69         else
70             printf("res[0]=%f res[%d]=%f", res[0], tam-1, res[tam-1]);
71         printf("\n");
72
73         free(v);
74         free(res);
75
76
77         for ( i=0; i<tam; i++){
78             free(mat[i]);
79         }
80         free(mat);
81
82         return 0;
83     }

```

CAPTURAS DE PANTALLA:

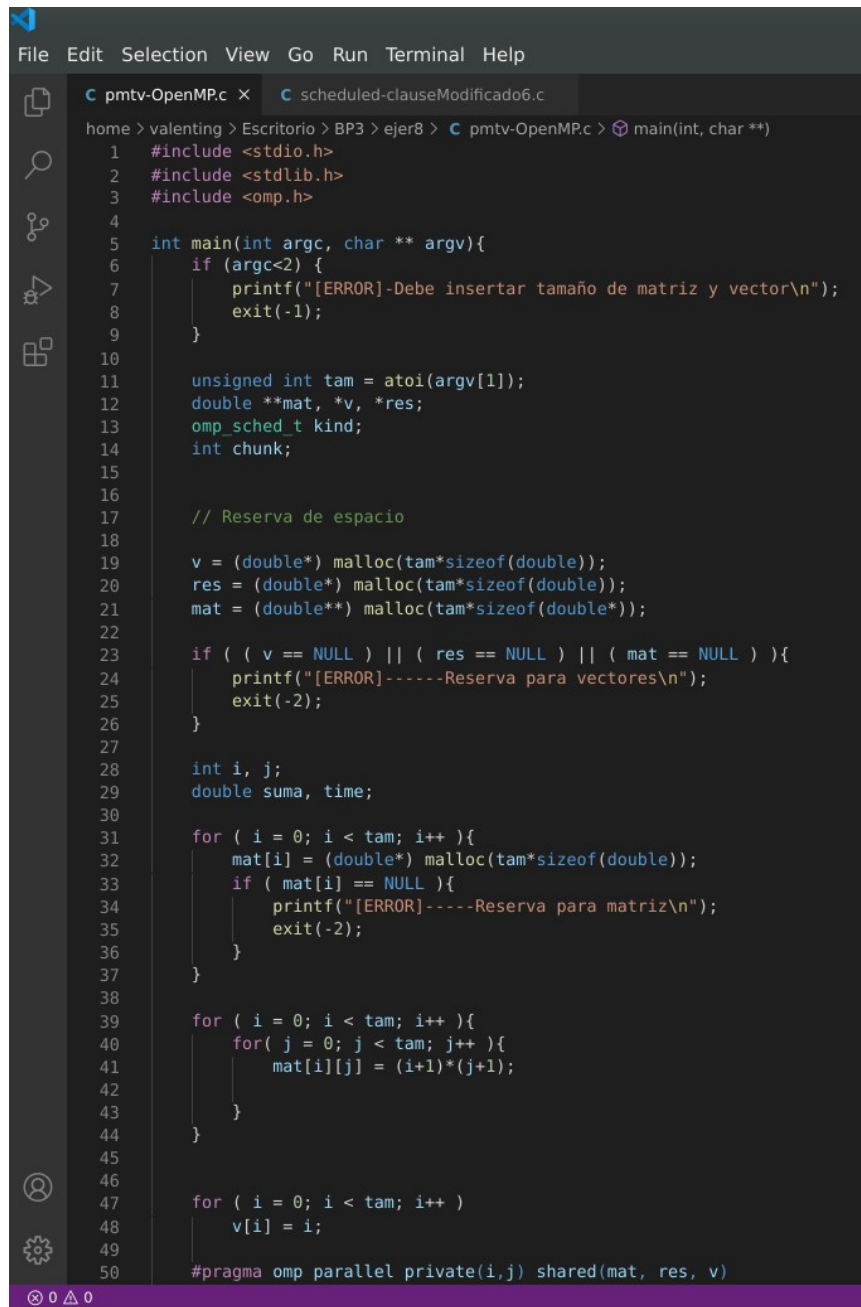
```

e3estudiante11
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer7] 2021-05-17 lunes
$gcc -O2 -fopenmp pmtv-secuencial.c -o pmtv-secuencial -lrt
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer7] 2021-05-17 lunes
$srn -pac ./pmtv-secuencial 10000
Tiempo: 0.068933 s

Resultado:
res[0]=49995000.000000 res[9999]=9999.000000
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer7] 2021-05-17 lunes
$

```

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


```

File Edit Selection View Go Run Terminal Help

C pmtv-OpenMP.c x C scheduled-clauseModificado6.c

home > valenting > Escritorio > BP3 > ejer8 > C pmtv-OpenMP.c > main(int, char **)
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <omp.h>
4
5  int main(int argc, char ** argv){
6      if (argc<2) {
7          printf("[ERROR]-Debe insertar tamaño de matriz y vector\n");
8          exit(-1);
9      }
10
11     unsigned int tam = atoi(argv[1]);
12     double **mat, *v, *res;
13     omp_sched_t kind;
14     int chunk;
15
16     // Reserva de espacio
17
18     v = (double*) malloc(tam*sizeof(double));
19     res = (double*) malloc(tam*sizeof(double));
20     mat = (double**) malloc(tam*sizeof(double*));
21
22     if ( ( v == NULL ) || ( res == NULL ) || ( mat == NULL ) ){
23         printf("[ERROR]-----Reserva para vectores\n");
24         exit(-2);
25     }
26
27     int i, j;
28     double suma, time;
29
30     for ( i = 0; i < tam; i++){
31         mat[i] = (double*) malloc(tam*sizeof(double));
32         if ( mat[i] == NULL ){
33             printf("[ERROR]-----Reserva para matriz\n");
34             exit(-2);
35         }
36     }
37
38     for ( i = 0; i < tam; i++){
39         for( j = 0; j < tam; j++){
40             mat[i][j] = (i+1)*(j+1);
41         }
42     }
43
44     for ( i = 0; i < tam; i++)
45         v[i] = i;
46
47     #pragma omp parallel private(i,j) shared(mat, res, v)

```

```

47     for ( i = 0; i < tam; i++ )
48         v[i] = i;
49
50     #pragma omp parallel private(i,j) shared(mat, res, v)
51     {
52
53         #pragma omp single
54         {
55             time = omp_get_wtime();
56         }
57
58         #pragma omp for schedule(runtime)
59         for ( i = 0; i < tam; i++ ) {
60             suma = 0;
61
62             for ( j = i; j < tam; j++ ){
63
64                 suma += mat[i][j]*v[j];
65             }
66             res[i] = suma;
67         }
68
69         #pragma omp single
70         {
71             time = omp_get_wtime() - time; //Cálculo del tiempo en que se realiza el producto
72             printf("Dentro de region paralela:");
73             omp_get_schedule(&kind,&chunk);
74             printf("Valor de run-sched-var: (%d, %d)\n", kind, chunk),
75         }
76
77     }
78
79     printf("Tiempo: %f s\n", time);
80
81
82     printf("\nResultado:\n");
83     if ( tam < 20 )
84         for ( i = 0; i < tam; i++)
85             printf("res[%d]=%f ", i, res[i]);
86     else
87         printf("res[0]=%f res[%d]=%f", res[0], tam-1, res[tam-1]);
88     printf("\n");
89
90     free(v);
91     free(res);
92
93     for ( i=0; i<tam; i++){
94         free(mat[i]);
95     }

```

DESCOMPOSICIÓN DE DOMINIO:**CAPTURAS DE PANTALLA:**

```

e3estudiante11@atcgrid:~/bp3/ejer8
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer8] 2021-05-17 lunes
$gcc -O2 -fopenmp pmtv-OpenMP.c -o pmtv-OpenMP -lrt
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer8] 2021-05-17 lunes
$export OMP_SCHEDULE="guided"
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer8] 2021-05-17 lunes
$run -pac ./pmtv-OpenMP 100
Dentro de region paralela:Valor de run-sched-var: (3, 1)
Tiempo: 0.000019 s

Resultado:
res[0]=333300.000000 res[99]=990000.000000
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer8] 2021-05-17 lunes
$export OMP_SCHEDULE="dynamic"
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer8] 2021-05-17 lunes
$run -pac ./pmtv-OpenMP 100
Dentro de region paralela:Valor de run-sched-var: (2, 1)
Tiempo: 0.000023 s

Resultado:
res[0]=333300.000000 res[99]=990000.000000
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer8] 2021-05-17 lunes
$export OMP_SCHEDULE="static"
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer8] 2021-05-17 lunes
$run -pac ./pmtv-OpenMP 100
Dentro de region paralela:Valor de run-sched-var: (-2147483647, 0)
Tiempo: 0.000018 s

Resultado:
res[0]=333300.000000 res[99]=990000.000000
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer8] 2021-05-17 lunes
$

```

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:

ACLARACIÓN: Para la realización de este ejercicio he modificado el código del ejercicio anterior, permitiéndole pasar como parámetro el número de chunks. (simplemente `int chunk = atoi(argv[2]);`)

El número de multiplicaciones y sumas será el mismo, ya que por cada multiplicación en el segundo bucle anidado se realiza una suma. Sin embargo no podemos saber con exactitud dicha cantidad porque dependerá del número de hebras que tengamos en la región paralela y del número de componentes que tenga nuestro vector (equivalentemente el número de columnas y filas que tendrá nuestra matriz).

Primero, cada thread realizará un número de iteraciones del bucle concretas :

$$x = \text{número de iteraciones (tam)} / \text{número de threads.}$$

Suponiendo que x es un valor entero; el número de multiplicaciones y sumas como hemos dicho será idéntico, pero sin embargo cada hebra no realiza la misma cantidad ya que la forma en la que está declarada el índice j en el segundo bucle anidado permite obviar las multiplicaciones en las que el factor de la matriz es 0. Esto ahorra tiempo, sin embargo la primera hebra realizará un total de “tam” multiplicaciones en la primera iteración, y la segunda hebra realizará un total de “tam-1”. Es por ello que no podemos saber con exactitud el número de multiplicaciones y sumas de cada hebra.

Suponiendo que x no sea un valor entero; esto implicaría que algunas hebras realizarían más iteraciones que otras (una más). Es por ello que tampoco podríamos saber en este caso la cantidad de operaciones de multiplicación y de suma que realiza cada hebra.

Sí es seguro que la hebra 0 sería la que más operaciones realizaría y sucesivamente en orden creciente.

(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` ocurriría lo mismo, sin embargo las hebras que más rápido acabaran su correspondiente iteración realizarían más iteraciones que en el caso “`schedule(static)`” (distribución en tiempo de ejecución).

En el caso de `guided`, la distribución de las iteraciones entre las hebras sería más organizada y por tanto cada hebra realizaría el mismo número de operaciones de multiplicación y de suma.

(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 (n° iteraciones que restan dividido por n° 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`



```
File Edit Selection View Go Run Terminal Help

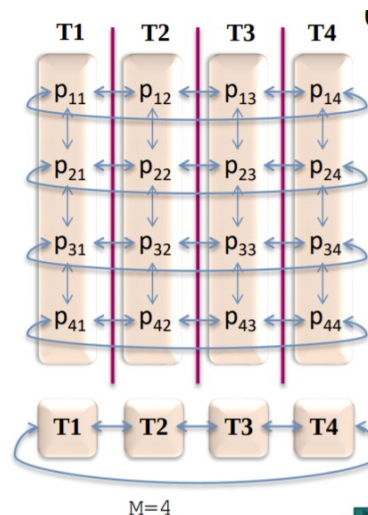
C pmtv-OpenMP.c X
home > valenting > Escritorio > BP3 > ejer10 > C pmtv-OpenMP.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 int main(int argc, char ** argv){
6     if (argc>3) {
7         printf("[ERROR]\n");
8         exit(-1);
9     }
10
11     unsigned int tam = atoi(argv[1]);
12     double **mat, *v, *res;
13
14
15     omp_sched_t kind;
16     int chunk;
17     //Permite introducir los chunks como parámetro al ejecutar el script
18     if(argc>2){
19         chunk = atoi(argv[2]);
20     }
21
22     // Reserva de espacio
23
24     v = (double*) malloc(tam*sizeof(double));
25     res = (double*) malloc(tam*sizeof(double));
26     mat = (double**) malloc(tam*sizeof(double*));
27
28     if ( ( v == NULL ) || ( res == NULL ) || ( mat == NULL ) ){
29         printf("[ERROR]-----Reserva para vectores\n");
30         exit(-2);
31     }
32
33     int i, j;
34     double suma, time;
35
36     for ( i = 0; i < tam; i++){
37         mat[i] = (double*) malloc(tam*sizeof(double));
38         if ( mat[i] == NULL ){
39             printf("[ERROR]-----Reserva para matriz\n");
40             exit(-2);
41         }
42     }
43
44     for ( i = 0; i < tam; i++){
45         for( j = 0; j < tam; j++){
46             mat[i][j] = (i+1)*(j+1);
47         }
48     }
49
50 }
```

```

50 }
51
52
53 for ( i = 0; i < tam; i++ )
54     v[i] = i;
55
56 #pragma omp parallel private(i,j) shared(mat, res, v)
57 {
58     #pragma omp single
59     {
60         printf("Dentro de region paralela:");
61         omp_get_schedule(&kind,&chunk);
62         printf("Valor de run-sched-var: (%d, %d)\n", kind, chunk);
63         time = omp_get_wtime();
64     }
65
66     #pragma omp for schedule(runtime)
67     for ( i = 0; i < tam; i++ ) {
68         suma = 0;
69
70         for ( j = i; j < tam; j++ ){
71             suma += mat[i][j]*v[j];
72         }
73         res[i] = suma;
74     }
75
76     #pragma omp single
77     {
78         time = omp_get_wtime() - time; //Cálculo del tiempo en que se realiza el producto
79     }
80
81 }
82
83 printf("Tiempo: %f s\n", time);
84
85 printf("\nResultado:\n");
86 if ( tam < 20 )
87     for ( i = 0; i < tam; i++)
88         printf("res[%d]=%f ", i, res[i]);
89 else
90     printf("res[0]=%f res[%d]=%f", res[0], tam-1, res[tam-1]);
91 printf("\n");
92
93 free(v);
94 free(res);
95
96
97
98

```

DESCOMPOSICIÓN DE DOMINIO:



Captura extraída de las transparencias correspondientes al Tema 2 de la asignatura Arquitectura de Computadores.

CAPTURAS DE PANTALLA:

```

e3estudiante11@atcgrid:~/bp3/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer10] 2021-05-18 martes
$ sbatch -pac *.sh
Submitted batch job 106438
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp3/ejer10] 2021-05-18 martes
$ cat *.out
Dentro de region paralela: Valor de run-sched-var: (-2147483647, 0)
Tiempo: 0.193449 s

Resultado:
res[0]=1365333328000.000000 res[15999]=4095744000000.000000
Dentro de region paralela: Valor de run-sched-var: (-2147483647, 1)
Tiempo: 0.173698 s

Resultado:
res[0]=1365333328000.000000 res[15999]=4095744000000.000000
Dentro de region paralela: Valor de run-sched-var: (-2147483647, 64)
Tiempo: 0.167811 s

Resultado:
res[0]=1365333328000.000000 res[15999]=4095744000000.000000
Dentro de region paralela: Valor de run-sched-var: (-2147483646, 1)
Tiempo: 0.171387 s

Resultado:
res[0]=1365333328000.000000 res[15999]=4095744000000.000000
Dentro de region paralela: Valor de run-sched-var: (-2147483646, 1)
Tiempo: 0.170991 s

Resultado:
res[0]=1365333328000.000000 res[15999]=4095744000000.000000
Dentro de region paralela: Valor de run-sched-var: (-2147483646, 64)
Tiempo: 0.170557 s

Resultado:
res[0]=1365333328000.000000 res[15999]=4095744000000.000000
Dentro de region paralela: Valor de run-sched-var: (-2147483645, 1)
Tiempo: 0.177397 s

Resultado:
res[0]=1365333328000.000000 res[15999]=4095744000000.000000
Dentro de region paralela: Valor de run-sched-var: (-2147483645, 1)
Tiempo: 0.180403 s

Resultado:
res[0]=1365333328000.000000 res[15999]=4095744000000.000000

```

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid**SCRIPT:** pmvt-OpenMP_atcgrid.sh

```

File Edit Selection View Go Run Terminal Help
pmvt-OpenMP_atcgrid.sh x
home > valenting > Escritorio > BP3 > ejer10 > pmvt-OpenMP_
1  #!/bin/bash
2
3  export OMP_SCHEDULE="monotonic:static"
4  srun -pac ./pmtv-OpenMP 16000
5  export OMP_SCHEDULE="monotonic:static,1"
6  srun -pac ./pmtv-OpenMP 16000
7  export OMP_SCHEDULE="monotonic:static,64"
8  srun -pac ./pmtv-OpenMP 16000
9
10
11 export OMP_SCHEDULE="monotonic:dynamic"
12 srun -pac ./pmtv-OpenMP 16000
13 export OMP_SCHEDULE="monotonic:dynamic,1"
14 srun -pac ./pmtv-OpenMP 16000
15 export OMP_SCHEDULE="monotonic:dynamic,64"
16 srun -pac ./pmtv-OpenMP 16000
17
18
19 export OMP_SCHEDULE="monotonic:guided"
20 srun -pac ./pmtv-OpenMP 16000
21 export OMP_SCHEDULE="monotonic:guided,1"
22 srun -pac ./pmtv-OpenMP 16000
23 export OMP_SCHEDULE="monotonic:guided,64"
24 srun -pac ./pmtv-OpenMP 16000

```

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	0.181098	0.171525	0.178428
1	0.171858	0.171502	0.178541
64	0.171142	0.177021	0.178433
Chunk	Static	Dynamic	Guided
por defecto	0.178490	0.171489	0.17839
1	0.171758	0.17180	0.178470
64	0.171024	0.171047	0.178714