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): Brian Sena Simons
Grupo de prácticas: 2ºB – B2
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`

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

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

  if(argc<3){
    fprintf(stderr,"[ERROR] Exec: ./{this} {nº iter} {nº threads}");
    exit(-1);
  }

  n=atoi(argv[1]); if(n>20) n=20;
  x=atoi(argv[2]); if(x>32) x=32;

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

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

  return 0;

}
```

CAPTURAS DE PANTALLA:

```
2021-04-27-09:18 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer1$ gcc -O2 -fopenmp -o if-clauseModificado if-clauseModi
ficado.c
2021-04-27-09:18 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer1$ ./if-clauseModificado 5 3
Thread: 2 suma de a[4] = 4, sumalocal= 4
Thread: 0 suma de a[0] = 0, sumalocal= 0
Thread: 0 suma de a[1] = 1, sumalocal= 1
Thread: 1 suma de a[2] = 2, sumalocal= 2
Thread: 1 suma de a[3] = 3, sumalocal= 5
Thread master = 0 imprime suma = 10
2021-04-27-09:18 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer1$ ./if-clauseModificado 3 5
Thread: 0 suma de a[0] = 0, sumalocal= 0
Thread: 0 suma de a[1] = 1, sumalocal= 1
Thread: 0 suma de a[2] = 2, sumalocal= 3
Thread master = 0 imprime suma = 3
2021-04-27-09:18 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer1$ ./if-clauseModificado 10 10
Thread: 3 suma de a[3] = 3, sumalocal= 3
Thread: 2 suma de a[2] = 2, sumalocal= 2
Thread: 0 suma de a[0] = 0, sumalocal= 0
Thread: 9 suma de a[9] = 9, sumalocal= 9
Thread: 5 suma de a[5] = 5, sumalocal= 5
Thread: 8 suma de a[8] = 8, sumalocal= 8
Thread: 6 suma de a[6] = 6, sumalocal= 6
Thread: 1 suma de a[1] = 1, sumalocal= 1
Thread: 7 suma de a[7] = 7, sumalocal= 7
Thread: 4 suma de a[4] = 4, sumalocal= 4
Thread master = 0 imprime suma = 45
2021-04-27-09:18 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer1$
```

RESPUESTA:

Básicamente con la cláusula "if" puedo establecer un umbral para el cual no se ejecutará el código en paralelo si no que más bien utilizará apenas una hebra, es decir, de forma secuencial. Sin embargo, si n>4 entonces podré establecer el número de hebras que quiero utilizar para repartir el trabajo. Dando así mayor flexibilidad a los posibles problemas a resolver.

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=1	x=2	x=1	x=2	x=1	x=2
0	0	0	0	0	1	1	1	1
1	1	0	1	0	0	1	1	1
2	2	1	2	1	2	2	1	1
3	0	1	0	1	0	2	1	1
4	1	2	1	2	2	0	1	1
5	2	2	2	2	2	0	1	1
6	0	0	0	0	2	2	2	2
7	1	0	1	0	0	2	2	2
8	2	1	2	1	2	2	2	2
9	0	1	0	1	0	2	2	2
10	1	2	1	2	0	2	0	0
11	2	2	2	2	2	2	0	0

12	0	0	0	0	2	2	2	0
13	1	0	1	0	2	2	2	0
14	2	1	2	1	0	0	2	0
15	0	1	0	1	2	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:

La cláusula “static” divide uniformemente los “chunks”, normalmente de tamaño 1, a todas las hebras. Mientas que la cláusula “Dynamic”, que normalmente es para cuando se desconoce el tiempo de ejecución de las iteraciones, permite que las hebras más rápidas ejecuten más iteraciones del bucle. Esta última es igual que “Guided” solo que ahora el tamaño del bloque empieza siendo largo y luego va disminuyendo de tamaño, no más pequeño que chunk.

Según: **“Advanced OpenMP**”: La diferencia entre “monotoic” y “nonmonotonic” es que “nonmonotonic” le da más flexibilidad a la distribución de las iteraciones. Permitiendo que una hebra, por ejemplo, no siempre tenga que ejecutar secuencialmente “1,3,6” sino que pueda ir hacia atrás “3,6,1”;

• **simd**: the chunk_size must be a multiple of the simd width. 1 • **monotonic**: If a thread executed iteration i, then the thread must execute iterations larger than i subsequently. 1 • **non-monotonic**: Execution order not subject to the monotonic restriction

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.

Tras realizar un par de ejecuciones en mi terminal he determinado que el chunk por defecto de las reglas chunk/modifier es de un tamaño adaptativo al número de hebras y tamaño del bucle (trabajo/nºhebra), y solo cambiaría las hebras que participan y/o trabajan debido al modifier (diferencias entre Static, Dynamic and Guided).

4. Añadir al programa `scheduled-clause.c` lo necesario para que imprima el valor de las variables de control `dyn-var`, `nthreads-var`, `thread-limit-var` y `run-sched-var` dentro (debe imprimir sólo un thread) y fuera de la región paralela. Realizar varias ejecuciones usando variables de entorno para modificar estas variables de control antes de la ejecución. Incorporar en su cuaderno de prácticas volcados de pantalla de estas ejecuciones. ¿Se imprimen valores distintos dentro y fuera de la región paralela?

CAPTURA CÓDIGO FUENTE: `scheduled-clauseModificado.c`

```c
void print_stuff(){
  omp_sched_t tipo;
  int modif;
  char type[20];
  omp_get_schedule(&tipo,&modif);
  printf("dyn-var:%d\nnthreads-var:%d\nthread-limit-var:%d\nrun-schedvar:%d\n",
        omp_get_dynamic(),omp_get_max_threads(),omp_get_thread_limit(),modif);
 switch(tipo){
         case omp_sched_static:
           strcpy(type,"STATIC");
           break;
         case omp_sched_dynamic:
           strcpy(type,"DYNAMIC");
           break;
         case omp_sched_auto:
           strcpy(type,"AUTO");
           break;
         case omp_sched_guided:
           strcpy(type,"GUIDED");
           break;
         default:
           strcpy(type,"[ERROR SWITCH]");
           break;

       }
        printf("tipo:%s\n", type);
}
```

```c

#pragma omp parallel for firstprivate(suma) lastprivate(suma) schedule(dynamic, chunk)
  for(i=0;i<n;i++){
    if(first){
        first = false;
        #pragma single
        printf("\n\t############################## \n ");
        printf("\t Dentro del Bucle-for-paralelo \n");
                        print_stuff();
        printf("\t############################## \n\n");
    }
    suma+=a[i];
    //printf("thread:%d suma de a[%d], suma=%d\n",omp_get_thread_num(),i,suma);
  }

  printf("\n\t############################## \n");
  printf("\t Fuera del Bucle-for-paralelo \n");
                    print_stuff();
  printf("\t ############################## \n\n");
  return 0;
}

77
NORMAL  schedule-clauseModificado.c                    unix | utf-8 | c  100%  77:1
:HardTimeOff
```

CAPTURAS DE PANTALLA:

```
2021-04-30-01:08 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$ export OMP_DYNAMIC=TRUE
2021-04-30-01:09 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$ ./schedule-clauseModificado 5 2

        ##############################
         Dentro del Bucle-for-paralelo
dyn-var:1
nthreads-var:8
thread-limit-var:2147483647
run-schedvar:1
tipo:DYNAMIC
        ##############################


        ##############################
         Fuera del Bucle-for-paralelo
dyn-var:1
nthreads-var:8
thread-limit-var:2147483647
run-schedvar:1
tipo:DYNAMIC
         ##############################

2021-04-30-01:09 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$ export OMP_THREAD_LIMIT=2
2021-04-30-01:09 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$ ./schedule-clauseModificado 5 2

        ##############################
         Dentro del Bucle-for-paralelo
dyn-var:1
nthreads-var:8
thread-limit-var:2
run-schedvar:1
tipo:DYNAMIC
        ##############################


        ##############################
         Fuera del Bucle-for-paralelo
dyn-var:1
nthreads-var:8
thread-limit-var:2
run-schedvar:1
tipo:DYNAMIC
         ##############################

2021-04-30-01:09 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$
```

```
2021-04-30-01:15 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$ export OMP_SCHEDULE="auto"
2021-04-30-01:15 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$ ./schedule-clauseModificado 5 2

        ##############################
         Dentro del Bucle-for-paralelo
dyn-var:1
nthreads-var:8
thread-limit-var:2
run-schedvar:1
tipo:AUTO
        ##################################


        ##############################
         Fuera del Bucle-for-paralelo
dyn-var:1
nthreads-var:8
thread-limit-var:2
run-schedvar:1
tipo:AUTO
         ##############################

2021-04-30-01:15 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$ export OMP_SCHEDULE="guided"
2021-04-30-01:16 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$ export OMP_SCHEDULE="guided,4"
2021-04-30-01:16 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$ ./schedule-clauseModificado 5 2

        ##############################
         Dentro del Bucle-for-paralelo
dyn-var:1
nthreads-var:8
thread-limit-var:2
run-schedvar:4
tipo:GUIDED
        ##################################


        ##############################
         Fuera del Bucle-for-paralelo
dyn-var:1
nthreads-var:8
thread-limit-var:2
run-schedvar:4
tipo:GUIDED
         ##############################

2021-04-30-01:16 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer4$
```

RESPUESTA:
Siempre se imprime los mismos valores, ya sea dentro o fuera del bucle-for-paralelo.

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`

```
10
11 void print_stuff(){
12   omp_sched_t tipo;
13   int modif;
14   char type[20];
15   omp_get_schedule(&tipo,&modif);
16   printf("num_threads:%d\nnum_procs:%d\nin_parallel:%d\n",
17         omp_get_num_threads(),omp_get_num_procs(),omp_in_parallel());
18   /*printf("dyn-var:%d\nnthreads-var:%d\nthread-limit-var:%d\nrun-schedvar:%d\n",
19         omp_get_dynamic(),omp_get_max_threads(),omp_get_thread_limit(),modif);
20   */
21  switch(tipo){
22          case omp_sched_static:
23            strcpy(type,"STATIC");
```

CAPTURAS DE PANTALLA:

```
2021-04-30-01:26 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer5$ export OMP_DYNAMIC=TRUE
2021-04-30-01:26 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer5$ export OMP_SCHEDULE=dynamic
2021-04-30-01:26 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer5$ export OMP_THREAD_LIMIT=1000
2021-04-30-01:27 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer5$ ./schedule-clauseModificado 20 4

        ################################
         Dentro del Bucle-for-paralelo
num_threads:8
num_procs:8
in_parallel:1
tipo:DYNAMIC
        ################################


        ################################
         Fuera del Bucle-for-paralelo
num_threads:1
num_procs:8
in_parallel:0
tipo:DYNAMIC
         ################################

2021-04-30-01:27 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer5$
```

RESPUESTA:

Como podemos ver con "**in_parallel"** podemos verificar si estamos o no dentro de una zona de trabajo compartido. (1->Sí, 0->No). Y también otra diferencia es el número de "**threads**" en ejecución, como es de esperar, en la zona compartida contra la zona secuencial, 8 vs. 1 "**num_threads**". Sin embargo, sigue siendo el mismo **"num_procs".**

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`

```
    #define omp_get_thread_num() 0
#endif

void modify_stuff(int a, int b, int c){
  omp_set_dynamic(a);
  omp_set_num_threads(b);
  omp_set_schedule(omp_sched_static,c);
}

void print_stuff(){
  omp_sched_t tipo;
```

```
#pragma single
    {
    if(first){
      first = false;
      printf("\n\t############################### \n ");
      printf("\t Dentro del Bucle-for-paralelo \n");
      print_stuff();
      printf("\t############################### \n\n");
      }
    if(i==2){
      modify_stuff(3,2,2); //dynamic_value, num_threads, size_chunk
      printf("\n\t############################### \n ");
      printf("\tTras Modificar Dentro del Bucle-for-paralelo \n");
      print_stuff();
      printf("\t############################### \n\n");
      }
    }
    suma+=a[i];
    //printf("thread:%d suma de a[%d], suma=%d\n",omp_get_thread_num(),i,suma);
  }

  printf("\n\t############################## \n");
  printf("\t Fuera del Bucle-for-paralelo \n");
  print_stuff();
```

```
NORMAL  schedule-clauseModificado.c                    unix | utf-8 | c   50%   49:1
```

CAPTURAS DE PANTALLA:

```
2021-04-30-01:45 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer6$ gcc -O2 -fopenmp -o schedule-clauseM
dificado schedule-clauseModificado.c
2021-04-30-01:45 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer6$ ./schedule-clauseModificado 10 4

        ##############################
         Dentro del Bucle-for-paralelo
num_threads:8
num_procs:8
in_parallel:1
dyn-var:1
nthreads-var:8
thread-limit-var:1000
run-schedvar:1
tipo:DYNAMIC
        ##############################


        ##############################
        Tras Modificar Dentro del Bucle-for-paralelo
num_threads:8
num_procs:8
in_parallel:1
dyn-var:1
nthreads-var:2
thread-limit-var:1000
run-schedvar:2
tipo:STATIC
        ##############################


        ##############################
         Fuera del Bucle-for-paralelo
num_threads:1
num_procs:8
in_parallel:0
dyn-var:1
nthreads-var:8
thread-limit-var:1000
run-schedvar:1
tipo:DYNAMIC
         ##############################

2021-04-30-01:45 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer6$
```

RESPUESTA:

Como podemos observar si se han modificado correctamente los valores dentro de la ejecución de la zona de trabajo compartido, y al salir, los valores volvieron a ser los mismos de antes. **(mirar nthreads-var, run-schedvar y tipo para percibir las diferencias antes/después de modificar).**

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

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

NOTAS: (1) el número de filas/columnas debe ser un argumento de entrada; (2) se debe inicializar las matrices antes del cálculo; (3) se debe imprimir siempre la primera y última componente del resultado antes de que termine el programa.

CAPTURA CÓDIGO FUENTE: `pmtv-secuencial.c`

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

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

int main(int argc, char const* argv[])
{
  if(argc<2){
    fprintf(stderr,"[ERROR] Exec: ./{this} {size}\n");
    exit(-1);
  }

  omp_sched_t tipo;
  int modif;

  int i,n,j,suma;
  int **m, *v, *v_result;
  double t_ini, t_fin, t_elap;

  n=atoi(argv[1]);

  v = (int*)malloc(n*sizeof(int));
  v_result = (int*)malloc(n*sizeof(int));
  m = (int**)malloc(n*sizeof(int*));

  if((v==NULL)||(v_result==NULL)||(m==NULL)){
    fprintf(stderr,"[ERROR] Couldn't reserve memory\n");
    exit(-2);
  }

  for(int i=0;i<n;++i){
    m[i]=(int*)malloc(n*sizeof(int));
    if(m[i]==NULL){
      fprintf(stderr,"[ERROR] Not enough memory for column: %d\n",i);
      exit(-2);
    }
  }
```

```
32
31  //inicialización
30  for(i=0;i<n;++i){
29    for(j=0;j<n;++j){
28      if(i<=j)
27        m[i][j]=2;
26      else
25        m[i][j]=0;
24    }
23    v[i]=3;
22    v_result[i]=0;
21  }
20
19
18  //Cálculo
17  t_ini = omp_get_wtime();
16
15    for(i=0;i<n;++i){
14      suma=0;
13      for(j=0;j<n;++j){
12        suma+=m[i][j]*v[j];
11      }
10      v_result[i]=suma;
 9    }
 8
 7  t_fin = omp_get_wtime();
 6
 5  t_elap = t_fin - t_ini;
 4  printf("Tiempo (seg.):%11.9f\t tamaño: %d\n", t_elap,n);
 3  printf("v_result[%d]=%d\n",0,v_result[0]);
 2  printf("v_result[%d]=%d\n",n-1,v_result[n-1]);
 1
74  omp_get_schedule(&tipo,&modif);
 1  if(tipo==omp_sched_static)
 2    printf("[STATIC]");
 3  else if(tipo==omp_sched_dynamic)
 4    printf("[DYNAMIC]");
 5  else if(tipo==omp_sched_guided)
 6    printf("[GUIDED]");
 7  else
 8    printf("[AUTO]");
 9
10  printf("chunk:%d\n",modif);
11  free(v);
12  free(v_result);
```

```
 5  for(int i=0;i<n;++i)
 4    free(m[i]);
 3
 2  free(m);
 1  return 0;
93  }
NORMAL  pmtv-secuencial.c                    unix | utf-8 | c  100%  93:1
:HardTimeOff
```

CAPTURAS DE PANTALLA:

```
2021-05-02-11:57 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer7$ ls
pmtv-secuencial  pmtv-secuencial.c  slurm-102200.out  task.sh
2021-05-02-11:59 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer7$ cat slurm-102200.out
Tiempo (seg.):0.000000302        tamaño: 10
v_result[0]=60
v_result[9]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000000548        tamaño: 20
v_result[0]=120
v_result[19]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000001088        tamaño: 30
v_result[0]=180
v_result[29]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000001840        tamaño: 40
v_result[0]=240
v_result[39]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000002798        tamaño: 50
v_result[0]=300
v_result[49]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000003979        tamaño: 60
v_result[0]=360
v_result[59]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000005856        tamaño: 70
v_result[0]=420
v_result[69]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000007551        tamaño: 80
v_result[0]=480
v_result[79]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000009466        tamaño: 90
v_result[0]=540
v_result[89]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000011526        tamaño: 100
v_result[0]=600
v_result[99]=6
[DYNAMIC]chunk:1
```

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`

```
  //inicialización
  for(i=0;i<n;++i){
    for(j=0;j<n;++j){
      if(i<=j)
        m[i][j]=2;
      else
        m[i][j]=0;
    }
    v[i]=3;
    v_result[i]=0;
  }

55
  //Cálculo
  #pragma omp parallel private(j)
  {
    #pragma single
    t_ini = omp_get_wtime();

    #pragma omp for firstprivate(suma) schedule(runtime)
    for(i=0;i<n;++i){
      suma=0;
      for(j=0;j<n;++j){
        suma+=m[i][j]*v[j];
      }
    #pragma single
      v_result[i]=suma;
    }

    #pragma single
    t_fin = omp_get_wtime();

  }

  t_elap = t_fin - t_ini;
  printf("Tiempo (seg.):%11.9f\t tamaño: %d\n", t_elap,n);
  printf("v_result[%d]=%d\n",0,v_result[0]);
  printf("v_result[%d]=%d\n",n-1,v_result[n-1]);

  omp_get_schedule(&tipo,&modif);
  if(tipo==omp_sched_static)
    printf("[STATIC]");
  else if(tipo==omp_sched_dynamic)
    printf("[DYNAMIC]");
  else if(tipo==omp_sched_guided)
    printf("[GUIDED]");
NORMAL  pmtv-secuencial.c                    unix | utf-8 | c  54%  55:0
```

DESCOMPOSICIÓN DE DOMINIO:

$$c = A \bullet b; \quad c_i = \sum_{k=0}^{M-1} a_{ik} \bullet b_k = a_i^T \bullet b, \quad c(i) = \sum_{k=0}^{M-1} A(i,k) \bullet b(k), \; i = 0,\ldots N-1$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} \end{pmatrix}_{8x6} \bullet \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{pmatrix} \quad N=8 \; M=6 \quad = \quad \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{pmatrix} \begin{matrix} a_{11}b_1 + a_{12}b_2 + a_{13}b_3 + a_{14}b_4 + a_{15}b_5 + a_{16}b_6 \\ a_{21}b_1 + a_{22}b_2 + a_{23}b_3 + a_{24}b_4 + a_{25}b_5 + a_{26}b_6 \\ a_{31}b_1 + a_{32}b_2 + a_{33}b_3 + a_{34}b_4 + a_{35}b_5 + a_{36}b_6 \\ a_{41}b_1 + a_{42}b_2 + a_{43}b_3 + a_{44}b_4 + a_{45}b_5 + a_{46}b_6 \\ a_{51}b_1 + a_{52}b_2 + a_{53}b_3 + a_{54}b_4 + a_{55}b_5 + a_{56}b_6 \\ a_{61}b_1 + a_{62}b_2 + a_{63}b_3 + a_{64}b_4 + a_{65}b_5 + a_{66}b_6 \\ a_{71}b_1 + a_{72}b_2 + a_{73}b_3 + a_{74}b_4 + a_{75}b_5 + a_{76}b_6 \\ a_{81}b_1 + a_{82}b_2 + a_{83}b_3 + a_{84}b_4 + a_{85}b_5 + a_{86}b_6 \end{matrix}$$

CAPTURAS DE PANTALLA:

```
2021-05-02-12:00 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3$ cd ejer8
2021-05-02-12:00 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer8$ cat slurm-102201.out
Tiempo (seg.):0.000006050        tamaño: 10
v_result[0]=60
v_result[9]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000004835        tamaño: 20
v_result[0]=120
v_result[19]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000005875        tamaño: 30
v_result[0]=180
v_result[29]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000006717        tamaño: 40
v_result[0]=240
v_result[39]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000007108        tamaño: 50
v_result[0]=300
v_result[49]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000008415        tamaño: 60
v_result[0]=360
v_result[59]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000015009        tamaño: 70
v_result[0]=420
v_result[69]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000011764        tamaño: 80
v_result[0]=480
v_result[79]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000014566        tamaño: 90
v_result[0]=540
v_result[89]=6
[DYNAMIC]chunk:1
Tiempo (seg.):0.000016287        tamaño: 100
v_result[0]=600
v_result[99]=6
[DYNAMIC]chunk:1
2021-05-02-12:00 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer8$
```

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:

Observando la estructura del cálculo, vemos que en cada iteración se realiza una cantidad total de N multiplicaciones y N sumas (Inner-loop); Por lo que podemos suponer que con una versión static monotonic, cada hebra realizará un total de N/nºhebras ejecuciones del loop-exterior.

Dando un total estimado de (N/nºhebras)*(2N) operaciones de suma y multiplicaciones.

(Sin contar aquella hebra que llegue a ejecutar la última suma en el #pragma omp single).

(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:

No podemos suponer la carga laboral para cada hebra ya que depende de la rapidez de ejecución de cada una de ellas.

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

RESPUESTA:

He hecho varios tests (incluyo alguno a continuación) y al parecer la versión dinámica le gana a la versión estática por algunos segundos (cada vez más notable para un n < no). En mi opinión debe de ser porque la versión dinámica permite obviar aquellas hebras que estén tardando en finalizar su tarea por cualquier razón y así favorecer el rendimiento total de la ejecución. (Repartición no equitativa => sobrecarga en algunas hebras es mayor => pero tiempo de ejecución es menor).

```
[AUTO]chunk:1
2021-05-07-12:50 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer8$ OMP_SCHEDULE="monotonic:static,1" ./pmtv-OpenMP 10000
 10
Tiempo (seg.):0.031985000        tamaño: 10000
v_result[0]=60000
v_result[9999]=6
[AUTO]chunk:1
2021-05-07-12:50 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer8$ OMP_SCHEDULE="dynamic,1" ./pmtv-OpenMP 10000 10
Tiempo (seg.):0.029778300        tamaño: 10000
v_result[0]=60000
v_result[9999]=6
[DYNAMIC]chunk:1
2021-05-07-12:50 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer8$ OMP_SCHEDULE="static,1" ./pmtv-OpenMP 50000 10
Tiempo (seg.):11.927013600       tamaño: 50000
v_result[0]=300000
v_result[49999]=6
[AUTO]chunk:1
2021-05-07-12:51 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer8$ OMP_SCHEDULE="dynamic,1" ./pmtv-OpenMP 50000 10
Tiempo (seg.):8.114220300        tamaño: 50000
v_result[0]=300000
v_result[49999]=6
[DYNAMIC]chunk:1
2021-05-07-12:52 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer8$
```

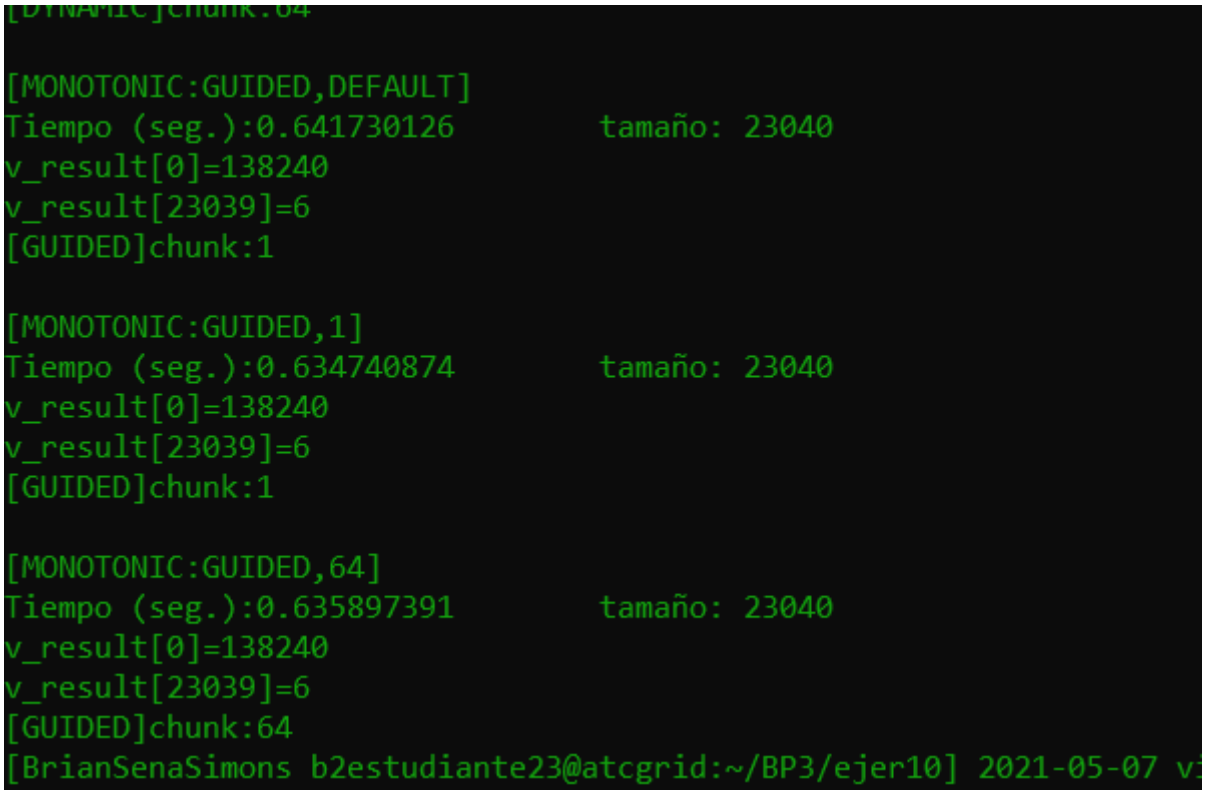
10. Obtener en atcgrid los tiempos de ejecución del código paralelo (usando, como siempre, -O2 al compilar) que multiplica una matriz triangular por un vector con las alternativas de planificación `static`, `dynamic` y `guided` para `chunk` de 1, 64 y el `chunk` por defecto para la alternativa (con `monotonic` 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`

He Utilizado el mismo código de paralelización por filas que en el **ejericicio8**

CAPTURAS DE PANTALLA:

```
[BrianSenaSimons b2estudiante23@atcgrid:~/BP3/ejer10] 2021-05-07 viern
$cat slurm-103602.out

[MONOTONIC:STATIC,DEFAULT]
Tiempo (seg.):0.650871214        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[AUTO]chunk:0

[MONOTONIC:STATIC,1]
Tiempo (seg.):0.634872735        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[AUTO]chunk:1

[MONOTONIC:STATIC,64]
Tiempo (seg.):0.640068561        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[AUTO]chunk:64

[MONOTONIC:DYNAMIC,DEFAULT]
Tiempo (seg.):0.639345739        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[DYNAMIC]chunk:1

[MONOTONIC:DYNAMIC,1]
Tiempo (seg.):0.635532711        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[DYNAMIC]chunk:1

[MONOTONIC:DYNAMIC,64]
Tiempo (seg.):0.636108361        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[DYNAMIC]chunk:64
```


[DYNAMIC]chunk:64
[MONOTONIC:GUIDED,DEFAULT]
Tiempo (seg.):0.641730126 tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[GUIDED]chunk:1
[MONOTONIC:GUIDED,1]
Tiempo (seg.):0.634740874 tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[GUIDED]chunk:1
[MONOTONIC:GUIDED,64]
Tiempo (seg.):0.635897391 tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[GUIDED]chunk:64
[BrianSenaSimons b2estudiante23@atcgrid:~/BP3/ejer10] 2021-05-07 v

```
Submitted batch job 103603
[BrianSenaSimons b2estudiante23@atcgrid:~/BP3/ejer10] 2021-05-07 viernes
$ls
pmtv-OpenMP  pmtv-OpenMP.c  slurm-103602.out  slurm-103603.out  task.sh
[BrianSenaSimons b2estudiante23@atcgrid:~/BP3/ejer10] 2021-05-07 viernes
$cat slurm-103603.out

[MONOTONIC:STATIC,DEFAULT]
Tiempo (seg.):0.651854269        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[AUTO]chunk:0

[MONOTONIC:STATIC,1]
Tiempo (seg.):0.642978173        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[AUTO]chunk:1

[MONOTONIC:STATIC,64]
Tiempo (seg.):0.635197040        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[AUTO]chunk:64

[MONOTONIC:DYNAMIC,DEFAULT]
Tiempo (seg.):0.642870516        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[DYNAMIC]chunk:1

[MONOTONIC:DYNAMIC,1]
Tiempo (seg.):0.633240532        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[DYNAMIC]chunk:1

[MONOTONIC:DYNAMIC,64]
Tiempo (seg.):0.640496396        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[DYNAMIC]chunk:64
```

```
[MONOTONIC:GUIDED,DEFAULT]
Tiempo (seg.):0.637744170        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[GUIDED]chunk:1

[MONOTONIC:GUIDED,1]
Tiempo (seg.):0.640282393        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[GUIDED]chunk:1

[MONOTONIC:GUIDED,64]
Tiempo (seg.):0.639107812        tamaño: 23040
v_result[0]=138240
v_result[23039]=6
[GUIDED]chunk:64
[BrianSenaSimons b2estudiante23@atcgrid:~/BP3/ejer10] 2021-05-0
```

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid

SCRIPT: pmvt-OpenMP_atcgrid.sh

```
echo -e "\n[MONOTONIC:STATIC,DEFAULT]"
  export OMP_SCHEDULE="monotonic:static"
  ./pmtv-OpenMP 17280

echo -e "\n[MONOTONIC:STATIC,1]"
  export OMP_SCHEDULE="monotonic:static,1"
  ./pmtv-OpenMP 17280

echo -e "\n[MONOTONIC:STATIC,1]"
  export OMP_SCHEDULE="monotonic:static,64"
  ./pmtv-OpenMP 17280

echo -e "\n[MONOTONIC:DYNAMIC,DEFAULT]"
  export OMP_SCHEDULE="monotonic:dynamic"
  ./pmtv-OpenMP 17280

echo -e "\n[MONOTONIC:DYNAMIC,1]"
  export OMP_SCHEDULE="monotonic:dynamic,1"
  ./pmtv-OpenMP 17280

echo -e "\n[MONOTONIC:DYNAMIC,64]"
  export OMP_SCHEDULE="monotonic:dynamic,64"
  ./pmtv-OpenMP 17280

echo -e "\n[MONOTONIC:GUIDED,DEFAULT]"
  export OMP_SCHEDULE="monotonic:guided"
  ./pmtv-OpenMP 17280

echo -e "\n[MONOTONIC:GUIDED,1]"
  export OMP_SCHEDULE="monotonic:guided,1"
  ./pmtv-OpenMP 17280

echo -e "\n[MONOTONIC:GUIDED,64]"
  export OMP_SCHEDULE="monotonic:guided,64"
  ./pmtv-OpenMP 17280
2021-05-07-01:21 BrianSenaSimons@Ubuntu-20.04:~/Uni2/AC/BP3/ejer10$
```

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	0.650871214	0.639345739	0.641730126
1	0.634872735	0.635532711	0.634740874

64	0.640068561	0.636108361	0.635897391

Chunk	**Static**	**Dynamic**	**Guided**
por defecto	0.651854269	0.642870516	0.637744170
1	0.642978173	0.633240532	0.640282393
64	0.635197040	0.640496396	0.639107812