2º curso / 2º cuatr. Grados Ingeniería Informática	**Arquitectura de Computadores (AC)** **Cuaderno de prácticas.** **Bloque Práctico 3. Programación paralela III: Interacción con el entorno en OpenMP** Estudiante (nombre y apellidos): Grupo de prácticas y profesor de prácticas:

Parte I. 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
int main(int argc, char **argv)
{
    int i, n=20, tid, x = 4;
    int a[n], suma=0, sumalocal;

    if (argc < 2)
    {
        fprintf(stderr,"[ERROR] Falta el nÃºmero de iteraciones\n");
        exit(-1);
    }

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

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

    #pragma omp parallel num_threads(x) if(n>4) 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
```

CAPTURAS DE PANTALLA:

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ gcc if-clauseModificado.c -o if-clauseModificado -fopenmp -O2
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./if-clauseModificado
[ERROR] Falta el nÃºmero de iteraciones
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./if-clauseModificado 5
 thread 0 suma de a[0]=0 sumalocal=0
 thread 0 suma de a[1]=1 sumalocal=1
 thread 3 suma de a[4]=4 sumalocal=4
 thread 2 suma de a[3]=3 sumalocal=3
 thread 1 suma de a[2]=2 sumalocal=2
thread master=0 imprime suma=10
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./if-clauseModificado 4
 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 master=0 imprime suma=6
```

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ gcc if-clauseModificado.c -o if-clauseModificado -fopenmp -O2
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./if-clauseModificado 40 5
thread 0 suma de a[0]=0 sumalocal=0
thread 7 suma de a[18]=18 sumalocal=18
thread 5 suma de a[14]=14 sumalocal=14
thread 4 suma de a[12]=12 sumalocal=12
thread 4 suma de a[13]=13 sumalocal=25
thread 0 suma de a[1]=1 sumalocal=1
thread 1 suma de a[3]=3 sumalocal=3
thread 1 suma de a[4]=4 sumalocal=7
thread 2 suma de a[6]=6 sumalocal=6
thread 2 suma de a[7]=7 sumalocal=13
thread 2 suma de a[8]=8 sumalocal=21
thread 0 suma de a[2]=2 sumalocal=3
thread 6 suma de a[16]=16 sumalocal=16
thread 6 suma de a[17]=17 sumalocal=33
thread 7 suma de a[19]=19 sumalocal=37
thread 1 suma de a[5]=5 sumalocal=12
thread 5 suma de a[15]=15 sumalocal=29
thread 3 suma de a[9]=9 sumalocal=9
thread 3 suma de a[10]=10 sumalocal=19
thread 3 suma de a[11]=11 sumalocal=30
thread master=0 imprime suma=190
```

RESPUESTA:
El programa reparte las iteraciones del bucle para que se vayan sumando con esta cláusula lo que hacemos es fijar el número de hebras que se van a usar

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 y 2).

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 y 2 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	0	0	0
1	1	0	0	1	1	0	1	0
2	2	1	0	2	2	1	2	1
3	3	1	0	3	3	1	3	1
4	4	2	0	4	4	2	4	2
5	5	2	0	5	5	2	5	2
6	6	3	0	6	6	3	6	3
7	7	3	0	7	7	3	7	3
8	0	4	0	0	0	4	0	4
9	1	4	0	1	1	4	1	4
10	2	5	0	2	2	5	2	5
11	3	5	0	3	3	5	3	5
12	4	6	0	4	4	6	4	6
13	5	6	0	5	5	6	5	6
14	6	7	0	6	6	7	6	7
15	7	7	0	7	7	7	7	7

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:
En static se distribuyen las hebras en tiempo de compilación y se dividen en chunks, todas ejecutan las mismas iteraciones que el resto en orden.
En dynamic se distribuyen en tiempo de ejecución y si ejecutas varias veces se ejecutan en un orden distinto cada vez.
En guided se distribuyen en tiempo de ejecución pero la primera hebra ejecuta nIte/nhebras y la siguiente hace lo mismo con el número de iteraciones restantes.
En monotonic si una hebra ejecuta la iteración I ya no puede ejecutar una iteración de un número menor.
En nonmonotonic se anula el efecto de monotonic.

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.

Static usa un valor por defecto de 1 mientras que dynamic y guided depende de la carga de trabajo que se tenga.

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

```
                lastprivate(suma) schedule(dynamic,chunk)
    for (i=0; i<n; i++)
    {
        if (i == 0){
                int nthreads_var = omp_get_max_threads();
                int thread_limit_var = omp_get_thread_limit();
                int dyn_var = omp_get_dynamic();
                int modifier;
                omp_sched_t kind;
                omp_get_schedule(&kind, &modifier);

                printf("Dinámica (T 0, F 1): %d\n",dyn_var);
                printf("N threads: %d\n",nthreads_var);
                printf("Límite de threads: %d\n", thread_limit_var);
                printf("kind (static 1, dynamic 2, guided 3): %d\n",kind);
                printf("Chunk: %d\n",modifier);

        }
        suma = suma + a[i];
        printf(" thread %d suma a[%d]=%d suma=%d \n",
               omp_get_thread_num(),i,a[i],suma);
    }

    printf("Fuera de 'parallel for' suma=%d\n",suma);
    int nthreads_var = omp_get_max_threads();
                int thread_limit_var = omp_get_thread_limit();
                int dyn_var = omp_get_dynamic();
                int modifier;
                omp_sched_t kind;
                omp_get_schedule(&kind, &modifier);

                printf("Dinámica (T 0, F 1): %d\n",dyn_var);
                printf("N threads: %d\n",nthreads_var);
                printf("Límite de threads: %d\n", thread_limit_var);
                printf("kind (static 1, dynamic 2, guided 3): %d\n",kind);
                printf("Chunk: %d\n",modifier);
}
```

CAPTURAS DE PANTALLA:

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ export OMP_SCHEDULE=nonmonotonic:static,1
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./shedeuled 12 3
 thread 0 suma a[9]=9 suma=9
 thread 0 suma a[10]=10 suma=19
 thread 0 suma a[11]=11 suma=30
Dinámica (T 0, F 1): 0
N threads: 8
Límite de threads: 2147483647
kind (static 1, dynamic 2, guided 3): 1
Chunk: 1
 thread 5 suma a[0]=0 suma=0
 thread 5 suma a[1]=1 suma=1
 thread 5 suma a[2]=2 suma=3
 thread 7 suma a[3]=3 suma=3
 thread 7 suma a[4]=4 suma=7
 thread 7 suma a[5]=5 suma=12
 thread 3 suma a[6]=6 suma=6
 thread 3 suma a[7]=7 suma=13
 thread 3 suma a[8]=8 suma=21
Fuera de 'parallel for' suma=30
Dinámica (T 0, F 1): 0
N threads: 8
Límite de threads: 2147483647
kind (static 1, dynamic 2, guided 3): 1
Chunk: 1
```

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ export OMP_SCHEDULE=nonmonotonic:dynamic,1
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./shedeuled 12 3
 thread 3 suma a[3]=3 suma=3
 thread 3 suma a[4]=4 suma=7
 thread 3 suma a[5]=5 suma=12
Dinámica (T 0, F 1): 0
N threads: 8
Límite de threads: 2147483647
kind (static 1, dynamic 2, guided 3): 2
Chunk: 1
 thread 2 suma a[0]=0 suma=0
 thread 2 suma a[1]=1 suma=1
 thread 2 suma a[2]=2 suma=3
 thread 1 suma a[6]=6 suma=6
 thread 1 suma a[7]=7 suma=13
 thread 1 suma a[8]=8 suma=21
 thread 6 suma a[9]=9 suma=9
 thread 6 suma a[10]=10 suma=19
 thread 6 suma a[11]=11 suma=30
Fuera de 'parallel for' suma=30
Dinámica (T 0, F 1): 0
N threads: 8
Límite de threads: 2147483647
kind (static 1, dynamic 2, guided 3): 2
Chunk: 1
```

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ export OMP_SCHEDULE=nonmonotonic:guided,1
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./shedeuled 12 3
Dinámica (T 0, F 1): 0
N threads: 8
Límite de threads: 2147483647
kind (static 1, dynamic 2, guided 3): 3
Chunk: 1
 thread 5 suma a[0]=0 suma=0
 thread 5 suma a[1]=1 suma=1
 thread 5 suma a[2]=2 suma=3
 thread 6 suma a[9]=9 suma=9
 thread 6 suma a[10]=10 suma=19
 thread 6 suma a[11]=11 suma=30
 thread 1 suma a[6]=6 suma=6
 thread 1 suma a[7]=7 suma=13
 thread 1 suma a[8]=8 suma=21
 thread 3 suma a[3]=3 suma=3
 thread 3 suma a[4]=4 suma=7
 thread 3 suma a[5]=5 suma=12
Fuera de 'parallel for' suma=30
Dinámica (T 0, F 1): 0
N threads: 8
Límite de threads: 2147483647
kind (static 1, dynamic 2, guided 3): 3
Chunk: 1
```

RESPUESTA:

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-clauseModificado2.c

```
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 for firstprivate(suma) \
30                  lastprivate(suma) schedule(dynamic,chunk)
31      for (i=0; i<n; i++)
32      {
33          if (i == 0){
34
35                  printf("N procesadores disponibles: %d\n",omp_get_num_procs());
36                  printf("N threads: %d\n",omp_get_num_threads());
37                  printf("En paralelo(T 1, F 0): %d\n", omp_in_parallel());
38
39
40          }
41          suma = suma + a[i];
42          printf(" thread %d suma a[%d]=%d suma=%d \n",
43                 omp_get_thread_num(),i,a[i],suma);
44      }
45
46      printf("Fuera de 'parallel for' suma=%d\n",suma);
47
48          printf("N procesadores disponibles: %d\n",omp_get_num_procs());
49          printf("N threads: %d\n",omp_get_num_threads());
50          printf("En paralelo(T 1, F 0): %d\n", omp_in_parallel());
51
52  }
53
```

CAPTURAS DE PANTALLA:

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ gcc scheduled-clauseModificado2.c -o shedeuled2 -fopenmp -O2
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ export OMP_NUM_THREADS=3
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./shedeuled2 12

Falta iteraciones o chunk
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./shedeuled2 12 3
 thread 1 suma a[6]=6 suma=6
 thread 1 suma a[7]=7 suma=13
 thread 1 suma a[8]=8 suma=21
 thread 1 suma a[9]=9 suma=30
 thread 1 suma a[10]=10 suma=40
N procesadores disponibles: 8
N threads: 3
En paralelo(T 1, F 0): 1
 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[11]=11 suma=51
 thread 2 suma a[3]=3 suma=3
 thread 2 suma a[4]=4 suma=7
 thread 2 suma a[5]=5 suma=12
Fuera de 'parallel for' suma=51
N procesadores disponibles: 8
N threads: 1
En paralelo(T 1, F 0): 0
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$
```

RESPUESTA:

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-clauseModificado3.c

```
34                int modifier;
35                omp_sched_t kind;
36                omp_get_schedule(&kind,&modifier);
37
38                printf("REGION PARALELA ANTES DE MODIF\n");
39                printf("Dinamica (true 0, false 1): %d\n",omp_get_dynamic());
40                printf("N threads: %d\n",omp_get_max_threads());
41                printf("Kind (static 1, dynamic 2, guided 3): %d\n",kind);
42                printf("Chunk: %d\n\n",modifier);
43
44                omp_set_dynamic(5);
45                omp_set_num_threads(6);
46                omp_set_schedule(1,2);
47                omp_get_schedule(&kind,&modifier);
48
49                printf("REGION PARALELA DESPUES DE MODIF\n");
50                printf("Dinamica (true 0, false 1): %d\n",omp_get_dynamic());
51                printf("N threads: %d\n",omp_get_max_threads());
52                printf("Kind (static 1, dynamic 2, guided 3): %d\n",kind);
53                printf("Chunk: %d\n\n",modifier);
54
55
56
57        }
58        suma = suma + a[i];
59        printf(" thread %d suma a[%d]=%d suma=%d \n",
60                omp_get_thread_num(),i,a[i],suma);
61    }
62
63    printf("Fuera de 'parallel for' suma=%d\n",suma);
64
65    int modifier;
66    omp_sched_t kind;
67    omp_get_schedule(&kind,&modifier);
68
69    printf("REGION SECUENCIAL ANTES DE MODIF\n");
70        printf("Dinamica (true 0, false 1): %d\n",omp_get_dynamic());
71        printf("N threads: %d\n",omp_get_max_threads());
72        printf("Kind (static 1, dynamic 2, guided 3): %d\n",kind);
73        printf("Chunk: %d\n\n",modifier);
74
75        omp_set_dynamic(4);
76        omp_set_num_threads(4);
77        omp_set_schedule(2,1);
78        omp_get_schedule(&kind,&modifier);
79
80        printf("REGION SECUENCIAL DESPUES DE MODIF\n");
81        printf("Dinamica (true 0, false 1): %d\n",omp_get_dynamic());
82        printf("N threads: %d\n",omp_get_max_threads());
83        printf("Kind (static 1, dynamic 2, guided 3): %d\n",kind);
84        printf("Chunk: %d\n\n",modifier);
```

CAPTURAS DE PANTALLA:

```
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ gcc scheduled-clauseModificado3.c -o shedeuled3 -fopenmp -O2
jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$ ./shedeuled3 12 3
REGION PARALELA ANTES DE MODIF
Dinamica (true 0, false 1): 0
N threads: 3
Kind (static 1, dynamic 2, guided 3): 3
Chunk: 1

REGION PARALELA DESPUES DE MODIF
Dinamica (true 0, false 1): 1
N threads: 6
Kind (static 1, dynamic 2, guided 3): 1
Chunk: 2

 thread 2 suma a[0]=0 suma=0
 thread 2 suma a[1]=1 suma=1
 thread 2 suma a[2]=2 suma=3
 thread 2 suma a[9]=9 suma=12
 thread 2 suma a[10]=10 suma=22
 thread 2 suma a[11]=11 suma=33
 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[6]=6 suma=6
 thread 0 suma a[7]=7 suma=13
 thread 0 suma a[8]=8 suma=21
Fuera de 'parallel for' suma=33
REGION SECUENCIAL ANTES DE MODIF
Dinamica (true 0, false 1): 0
N threads: 3
Kind (static 1, dynamic 2, guided 3): 3
Chunk: 1

REGION SECUENCIAL DESPUES DE MODIF
Dinamica (true 0, false 1): 1
N threads: 4
Kind (static 1, dynamic 2, guided 3): 2
Chunk: 1

jesus@jesus-HP-EliteBook-830-G7-Notebook-PC:~/AC/Practicas/Practica3$
```

RESPUESTA: Todo se actualiza bien

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

7. 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. 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
            printf("No hay suficiente espacio para m \n");
            exit(EXIT_FAILURE);
        }
    }
    #endif

    // Inicializar vector y matriz
    for (i = 0; i < N; i++){
        v1[i] = 0.1*i;
        v2[i] = 0;
        for (j = 0; j < i; j++)
            m[i][j] = 0;
        for (j = i; j < N; j++)
            m[i][j] = i*N+j;
    }

    // Calcular v2 = m * v1
    clock_gettime(CLOCK_REALTIME,&cgt1);

    #pragma omp parallel for schedule(runtime)
    for(i = 0; i < N; i++){
        for (j = i; j < N; j++)
            v2[i] += m[i][j] * v1[j];
    }
    clock_gettime(CLOCK_REALTIME,&cgt2);
    ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
        (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

    // Imprimir resultados
    if (N < 10){
        printf("Tiempo: %11.9f\t Tamaño: %u\n", ncgt, N);
        printf("Matriz:\n\t");
        for (i = 0; i < N; i++){
            for (j = 0; j < N; j++)
                printf("%8.6f \t ", m[i][j]);
            printf("\n\t");
```

DESCOMPOSICIÓN DE DOMINIO:

CAPTURAS DE PANTALLA:

```
ac125@atcgrid:~/Practica3
Try "srun --help" for more information
[ac125@atcgrid Practica3]$ gcc -O2 -openmp -o XD pmtv-OpenMP.c
[ac125@atcgrid Practica3]$ srun -pac4 -Aac -O2 -openmp XD
srun: invalid option -- '2'
srun: invalid option -- '2'
Try "srun --help" for more information
[ac125@atcgrid Practica3]$ srun -pac4 -Aac XD
Uso: XD tamaño
srun: error: atcgrid4: task 0: Exited with exit code 1
[ac125@atcgrid Practica3]$ srun -pac4 -Aac XD 5
Tiempo: 0.000000192      Tamaño: 5
Matriz:
        0.000000        1.000000        2.000000        3.000000        4.000000
        0.000000        6.000000        7.000000        8.000000        9.000000
        0.000000        0.000000        12.000000       13.000000       14.000000
        0.000000        0.000000        0.000000        18.000000       19.000000
        0.000000        0.000000        0.000000        0.000000        24.000000

Vector:
        0.000000 0.100000 0.200000 0.300000 0.400000

Vector resultado:
        3.000000 8.000000 11.900000 13.000000 9.600000
```

8. 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:

Pues en la fila i habrá i multiplicaciones y i-1 sumas.

(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:

Con dynamic se le van a ir asignando nuevas filas según vaya acabando y con guided se reparte dependiendo del número de iteraciones y el número de hebras así que puede haber hebras con más iteraciones que otras.

(c) ¿Qué alternativa cree que debería ofrecer mejores prestaciones? Razonar la respuesta.

RESPUESTA:

Dynamic porque reparte mejor las hebras además de que en el momento de que una este ociosa le dará más trabajo.

9. 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

DESCOMPOSICIÓN DE DOMINIO:

CAPTURAS DE PANTALLA:

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid

SCRIPT: pmvt-OpenMP_atcgrid.sh

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			