

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): Miguel Ángel Fernández Gutiérrez

Grupo de prácticas: GIM2, Francisco Barranco

Fecha de entrega: 15 de mayo, 2019

Fecha evaluación en clase: 16 de mayo, 2019

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`

```
GNU nano 2.9.3 if-clauseModificado.c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main(int argc, char **argv)
{
    int i, n=20, tid;
    int a[n], suma=0, sumalocal;
    if(argc < 3) {
        fprintf(stderr, "[ERROR]-Falta iteraciones y número de threads\n");
        exit(-1);
    }

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

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

    int x = atoi(argv[2]);

    #pragma omp parallel if(n>4) default(none) \
    private(sumalocal,tid) shared(a,suma,n) num_threads(x)
    {
        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);
    }
}
```

CAPTURAS DE PANTALLA:

```

ejer1 : bash — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer1] 2019-05-13 lunes
$ ./if-clauseModificado 20 20
thread 0 suma de a[0]=0 sumalocal=0
thread 7 suma de a[7]=7 sumalocal=7
thread 6 suma de a[6]=6 sumalocal=6
thread 10 suma de a[10]=10 sumalocal=10
thread 1 suma de a[1]=1 sumalocal=1
thread 3 suma de a[3]=3 sumalocal=3
thread 5 suma de a[5]=5 sumalocal=5
thread 13 suma de a[13]=13 sumalocal=13
thread 14 suma de a[14]=14 sumalocal=14
thread 15 suma de a[15]=15 sumalocal=15
thread 16 suma de a[16]=16 sumalocal=16
thread 17 suma de a[17]=17 sumalocal=17
thread 9 suma de a[9]=9 sumalocal=9
thread 8 suma de a[8]=8 sumalocal=8
thread 11 suma de a[11]=11 sumalocal=11
thread 12 suma de a[12]=12 sumalocal=12
thread 18 suma de a[18]=18 sumalocal=18
thread 19 suma de a[19]=19 sumalocal=19
thread 4 suma de a[4]=4 sumalocal=4
thread 2 suma de a[2]=2 sumalocal=2
thread master=0 imprime suma=190
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer1] 2019-05-13 lunes
$ ./if-clauseModificado 20 2
thread 0 suma de a[0]=0 sumalocal=0
thread 0 suma de a[1]=1 sumalocal=1
thread 0 suma de a[2]=2 sumalocal=3
thread 0 suma de a[3]=3 sumalocal=6
thread 0 suma de a[4]=4 sumalocal=10
thread 0 suma de a[5]=5 sumalocal=15
thread 0 suma de a[6]=6 sumalocal=21
thread 0 suma de a[7]=7 sumalocal=28
thread 0 suma de a[8]=8 sumalocal=36
thread 0 suma de a[9]=9 sumalocal=45
thread 1 suma de a[10]=10 sumalocal=10
thread 1 suma de a[11]=11 sumalocal=21
thread 1 suma de a[12]=12 sumalocal=33
thread 1 suma de a[13]=13 sumalocal=46
thread 1 suma de a[14]=14 sumalocal=60
thread 1 suma de a[15]=15 sumalocal=75
thread 1 suma de a[16]=16 sumalocal=91
thread 1 suma de a[17]=17 sumalocal=108
thread 1 suma de a[18]=18 sumalocal=126
thread 1 suma de a[19]=19 sumalocal=145
thread master=0 imprime suma=190
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer1] 2019-05-13 lunes

```

RESPUESTA:

Claramente, en las capturas de pantalla vemos cómo la variación del número de threads en entrada modifica el número de threads que ejecutan las instrucciones del for paralelizado.

2. (a) Rellenar la Tabla 1 (se debe poner en la tabla el id del *thread* que ejecuta cada iteración) ejecutando los ejemplos del seminario `schedule-clause.c`, `scheduled-clause.c` y `scheduleg-clause.c` con dos *threads* (0,1) y unas entradas de:

- iteraciones: 16 (0,...15)
- chunk= 1, 2 y 4

Tabla 1 . Tabla schedule. En la segunda fila, 1, 2 4 representan el tamaño del chunk (consulte seminario)

Iteración	schedule-clause.c			scheduled-clause.c			scheduleg-clause.c		
	1	2	4	1	2	4	1	2	4
0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0
2	0	1	0	0	1	0	0	0	0
3	1	1	0	0	1	0	0	0	0
4	0	0	1	0	0	1	0	0	0
5	1	0	1	0	0	1	0	0	0
6	0	1	1	0	0	1	0	0	0
7	1	1	1	0	0	1	0	0	0
8	0	0	0	0	0	0	1	1	1
9	1	0	0	0	0	0	1	1	1
10	0	1	0	0	0	0	1	1	1
11	1	1	0	0	0	0	1	1	1
12	0	0	1	0	0	0	0	0	0
13	1	0	1	0	0	0	0	0	0
14	0	1	1	0	0	0	1	0	0
15	1	1	1	0	0	0	1	0	0

(b) Rellenar otra tabla como la de la figura pero esta vez usando cuatro *threads* (0,1,2,3).

Tabla 2 . Tabla schedule. En la segunda fila, 1, 2 4 representan el tamaño del chunk (consulte seminario)

Iteración	schedule-clause.c			scheduled-clause.c			scheduleg-clause.c		
	1	2	4	1	2	4	1	2	4
0	0	0	0	2	2	1	0	1	1
1	1	0	0	0	2	1	0	1	1
2	2	1	0	1	0	1	0	1	1
3	3	1	0	3	0	1	0	1	1
4	0	2	1	2	3	0	1	0	0
5	1	2	1	2	3	0	1	0	0
6	2	3	1	2	1	0	1	0	0
7	3	3	1	2	1	0	3	3	0
8	0	0	2	2	0	3	3	3	3
9	1	0	2	2	0	3	3	3	3
10	2	1	2	2	0	3	2	2	3
11	3	1	2	2	0	3	2	2	3
12	0	2	3	2	0	2	2	0	2
13	1	2	3	2	0	2	2	0	2
14	2	3	3	2	1	2	2	0	2
15	3	3	3	2	1	2	2	0	2

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

RESPUESTA:

Con `static`, las iteraciones se dividen en “paquetes” de un número de iteraciones igual al `chunk`, y se van asignando en round robin. Con `dynamic`, aunque el paquete tenga `chunk` iteraciones, la asignación se hace en tiempo de ejecución, es decir, se asigna dicho `chunk` a la hebra que termine antes. En el caso de `guided`, el `chunk` indica el tamaño mínimo de bloque.

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

CAPTURA CÓDIGO FUENTE: `scheduled-clauseModificado.c`

```
GNU nano 2.9.3 scheduled-clauseModificado.c

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

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

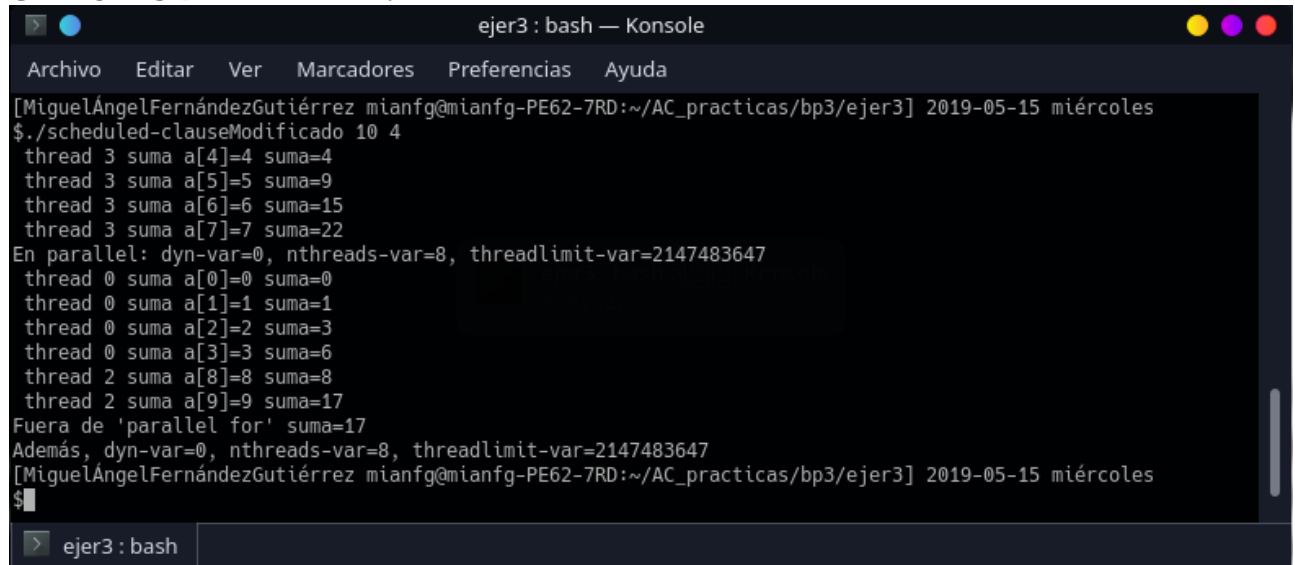
int main(int argc, char **argv) {
    int i, n=200, chunk, a[n], suma=0;
    if(argc < 3) {
        fprintf(stderr, "\nFalta iteraciones o chunk \n");
        exit(-1);
    }
    n = atoi(argv[1]); if (n>200) n=200; chunk = atoi(argv[2]);

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

    #pragma omp parallel for firstprivate(suma) \
    lastprivate(suma) schedule(dynamic,chunk)
    for (i=0; i<n; i++) {
        if ( i == 0 )
            printf("En parallel: dyn-var=%d, nthreads-var=%d, threadlimit-var=%d\n",
                omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit());
        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);
    printf("Además, dyn-var=%d, nthreads-var=%d, threadlimit-var=%d\n",
        omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit());
}
```

CAPTURAS DE PANTALLA:



```
ejer3 : bash — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer3] 2019-05-15 miércoles
$ ./scheduled-clauseModificado 10 4
thread 3 suma a[4]=4 suma=4
thread 3 suma a[5]=5 suma=9
thread 3 suma a[6]=6 suma=15
thread 3 suma a[7]=7 suma=22
En parallel: dyn-var=0, nthreads-var=8, threadlimit-var=2147483647
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[3]=3 suma=6
thread 2 suma a[8]=8 suma=8
thread 2 suma a[9]=9 suma=17
Fuera de 'parallel for' suma=17
Además, dyn-var=0, nthreads-var=8, threadlimit-var=2147483647
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer3] 2019-05-15 miércoles
$
```

RESPUESTA:

Vemos que el valor es el mismo independientemente de que se encuentre dentro de la región paralelizada o no.

4. Usar en el ejemplo anterior las funciones `omp_get_num_threads()`, `omp_get_num_procs()` y `omp_in_parallel()` dentro y fuera de la región paralela. Imprimir los valores que obtienen estas funciones dentro (lo debe imprimir sólo uno de los threads) y fuera de la región paralela. Incorporar en su cuaderno de prácticas volcados de pantalla con los resultados de ejecución obtenidos. Indicar en qué funciones se obtienen valores distintos dentro y fuera de la región paralela.

CAPTURA CÓDIGO FUENTE: `scheduled-clauseModificado4.c`

```
GNU nano 2.9.3 scheduled-clauseModificado4.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 **argv) {
    int i, n=200, chunk, a[n], suma=0;
    if(argc < 3) {
        fprintf(stderr, "\nFalta iteraciones o chunk \n");
        exit(-1);
    }
    n = atoi(argv[1]); if (n>200) n=200; chunk = atoi(argv[2]);

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

    #pragma omp parallel for firstprivate(suma) \
        lastprivate(suma) schedule(dynamic, chunk)
    for (i=0; i<n; i++) {
        if ( i == 0 )
            printf("En parallel: num-threads=%d, num-procs=%d, in-parallel=%d\n",
                omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
        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);
    printf("Además, num-threads=%d, num-procs=%d, in-parallel=%d\n",
        omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
}
```

CAPTURAS DE PANTALLA:

```
ejer4: bash — Konsole

Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda

[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer4] 2019-05-15 miércoles
$ ./scheduled-clauseModificado4 10 4
thread 1 suma a[4]=4 suma=4
thread 1 suma a[5]=5 suma=9
thread 1 suma a[6]=6 suma=15
thread 1 suma a[7]=7 suma=22
thread 2 suma a[8]=8 suma=8
thread 2 suma a[9]=9 suma=17
En parallel: num-threads=8, num-procs=8, in-parallel=1
thread 3 suma a[0]=0 suma=0
thread 3 suma a[1]=1 suma=1
thread 3 suma a[2]=2 suma=3
thread 3 suma a[3]=3 suma=6
Fuera de 'parallel for' suma=17
Además, num-threads=1, num-procs=8, in-parallel=0
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer4] 2019-05-15 miércoles
$
```

RESPUESTA:

Obtenemos valores diferentes en `omp_num_threads` y en `omp_in_parallel`.

5. Añadir al programa `scheduled-clause.c` lo necesario para modificar las variables de control `dyn-var`, `nthreads-var` y `run-sched-var` y para poder imprimir el valor de estas variables antes y después de dicha modificación. Incorporar en su cuaderno de prácticas volcados de pantalla con los resultados de ejecución obtenidos.

CAPTURA CÓDIGO FUENTE: `scheduled-clauseModificado5.c`

```
GNU nano 2.9.3 scheduled-clauseModificado5.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 **argv) {
    int i, n=200, chunk, a[n], suma=0;
    if(argc < 3) {
        fprintf(stderr, "\nFalta iteraciones o chunk \n");
        exit(-1);
    }
    n = atoi(argv[1]); if (n>200) n=200; chunk = atoi(argv[2]);

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

    int modifier; omp_sched_t kind;

    #pragma omp parallel for firstprivate(suma) \
    lastprivate(suma) schedule(dynamic,chunk)
    for (i=0; i<n; i++) {
        if ( i == 0 )
        {
            printf("En parallel: dyn-var=%d, nthreads-var=%d, threadlimit-var=%d\n",
                omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit());
            printf("dyn-var: %d \n", omp_get_dynamic());
            omp_set_dynamic(3);
            printf("Modificamos dyn-var con omp_set_dynamic(1) y el resultado es: %d\n", omp_get_dynamic());
            printf("nthreads-var: %d \n", omp_get_max_threads());
            omp_set_num_threads(8);
            printf("Modificamos nthreads-var con omp_set_num_threads(8); y el resultado es: %d\n", omp_get_max_threads());

            omp_get_schedule(&kind, &modifier);
            printf("run-sched-var: (Kind: %d, Modifier: %d)\n", kind, modifier);
            omp_set_schedule(2, 2);
            omp_get_schedule(&kind, &modifier);
            printf("Modificamos run-sched-var con omp_set_schedule(2,2) y el resultado de Kind es %d y el de Modifier es: %d \n", kind, 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);
    printf("Además, dyn-var=%d, nthreads-var=%d, threadlimit-var=%d ",
        omp_get_dynamic(), omp_get_max_threads(), omp_get_thread_limit());
    omp_get_schedule(&kind, &modifier);
    printf("run-sched-var: (Kind: %d, Modifier: %d)\n", kind, modifier);
}
```

CAPTURAS DE PANTALLA:

```

ej5: bash — Konsole
Archivo Editar Ver Marcadores Preferencias Ayuda
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer5] 2019-05-15 miércoles
$ ./scheduled-clauseModificado5 10 4
thread 2 suma a[4]=4 suma=4
thread 2 suma a[5]=5 suma=9
thread 2 suma a[6]=6 suma=15
thread 2 suma a[7]=7 suma=22
thread 6 suma a[8]=8 suma=8
thread 6 suma a[9]=9 suma=17
En parallel: dyn-var=0, nthreads-var=8, threadlimit-var=2147483647
dyn-var: 0
Modificamos dyn-var con omp_set_dynamic(1) y el resultado es: 1
nthreads-var: 8
Modificamos nthreads-var con omp_set_num_threads(8); y el resultado es: 8
run-sched-var: (Kind: 2, Modifier: 1)
Modificamos run-sched-var con omp_set_schedule(2,2) y el resultado de Kind es 2 y el de Modifier es: 2
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 5 suma a[3]=3 suma=6
Fuera de 'parallel for' suma=17
Además, dyn-var=0, nthreads-var=8, threadlimit-var=2147483647 run-sched-var: (Kind: 2, Modifier: 1)
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer5] 2019-05-15 miércoles
$

```

Resto de ejercicios

6. Implementar un programa secuencial en C que multiplique una matriz triangular por un vector (use variables dinámicas). Compare el orden de complejidad del código que ha implementado con el código que implementó para el producto matriz por vector.

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

CAPTURA CÓDIGO FUENTE: pmtv-secuencial.c

```

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

```


CAPTURAS DE PANTALLA:

```

ejer6 : bash — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer6] 2019-05-15 miércoles
$ ./pmtv-secuencial 1500
Tiempo (seg): 0.001574

Resultado:
res[0]=1124250.000000 res[1499]=1499.000000
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer6] 2019-05-15 miércoles
$

```

7. Implementar en paralelo la multiplicación de una matriz triangular por un vector a partir del código secuencial realizado para el ejercicio anterior utilizando la directiva `for` de OpenMP. El código debe repartir entre los threads las iteraciones del bucle que recorre las filas. Dibujar en el cuaderno de prácticas la descomposición de dominio utilizada (Lección 4/Tema 2) en el código paralelo implementado para asignar tareas a los threads (Lección 5/Tema 2). Añadir lo necesario para que el usuario pueda fijar la planificación de tareas usando la variable de entorno `OMP_SCHEDULE`. Obtener en atcgrid los tiempos de ejecución del código paralelo (usando, como siempre, `-O2` al compilar) que multiplica una matriz triangular por un vector con las alternativas de planificación `static`, `dynamic` y `guided` para chunk de 1, 64 y el chunk por defecto para la alternativa. Use un tamaño de vector N múltiplo del número de cores y de 64 que no sea inferior a 15360. El número de threads en las ejecuciones debe coincidir con el número de cores. Rellenar la Tabla 3 dos veces con los tiempos obtenidos. Representar el tiempo para `static`, `dynamic` y `guided` en función del tamaño del chunk en una gráfica. ¿Qué alternativa ofrece mejores prestaciones? Razone por qué. Incluya los scripts utilizado en el cuaderno de prácticas. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

Conteste a las siguientes preguntas: (a) ¿Qué valor por defecto usa OpenMP para chunk con `static`, `dynamic` y `guided`? Indique qué ha hecho para obtener este valor por defecto para cada alternativa. (b) ¿Qué número de operaciones de multiplicación y suma realizan cada uno de los threads en la asignación `static` para cada uno de los chunks? (c) Con la asignación `dynamic` y `guided`, ¿qué cree que debe ocurrir con el número de operaciones de multiplicación y suma que realizan cada uno de los threads?

RESPUESTA:

a) Vemos el valor por defecto del chunk para cada planificación a continuación:

```

ejer7 : bash — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda

[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$export OMP_SCHEDULE="static"
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$./pmtv-OpenMP 1500
run-sched-var: (Kind: 1, Modifier: 0)
Tiempo (seg): 0.001271

Resultado:
res[0]=1124250.000000 res[1499]=1499.000000
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$export OMP_SCHEDULE="dynamic"
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$./pmtv-OpenMP 1500
run-sched-var: (Kind: 2, Modifier: 1)
Tiempo (seg): 0.000948

Resultado:
res[0]=1124250.000000 res[1499]=1499.000000
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$export OMP_SCHEDULE="guided"
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$./pmtv-OpenMP 1500
run-sched-var: (Kind: 3, Modifier: 1)
Tiempo (seg): 0.000840

Resultado:
res[0]=1124250.000000 res[1499]=1499.000000
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$

```

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

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

CAPTURA CÓDIGO FUENTE: pmtv-OpenMP.c

```

51 // cálculo de mat*v, res
52 #pragma omp parallel shared(mat,v,res) private(i,j)
53 {
54     #pragma omp single
55     {
56         omp_get_schedule(skind,&chunk_size);
57         printf("run-sched-var: (Kind: %d, Modifier: %d) \n",kind,chunk_size);
58     }
59     //Medida de tiempo
60     t = omp_get_wtime();
61 }
62
63 #pragma omp for schedule(runtime)
64 for ( i = 0; i < tam; i++ ) {
65     suma = 0;
66
67     for ( j = i; j < tam; j++ )
68         suma += mat[i][j]*v[j];
69
70     res[i] = suma;
71 }
72
73 // Medida de tiempo
74 #pragma omp single
75 {
76     t = omp_get_wtime() - t;
77 }
78
79 // Impresión de tiempo de ejecución
80

```

NOTA: se muestra la parte del código que ha sido modificada

DESCOMPOSICIÓN DE DOMINIO:

CAPTURAS DE PANTALLA:

```

ejer7 : bash — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda

[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$export OMP_SCHEDULE="static"
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$./pmtv-OpenMP 1500
run-sched-var: (Kind: 1, Modifier: 0)
Tiempo (seg): 0.001271

Resultado:
res[0]=1124250.000000 res[1499]=1499.000000
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$export OMP_SCHEDULE="dynamic"
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$./pmtv-OpenMP 1500
run-sched-var: (Kind: 2, Modifier: 1)
Tiempo (seg): 0.000948

Resultado:
res[0]=1124250.000000 res[1499]=1499.000000
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$export OMP_SCHEDULE="guided"
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$./pmtv-OpenMP 1500
run-sched-var: (Kind: 3, Modifier: 1)
Tiempo (seg): 0.000840

Resultado:
res[0]=1124250.000000 res[1499]=1499.000000
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer7] 2019-05-15 miércoles
$

```

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

```

1  #!/bin/bash
2
3  #PBS -N pmtv-OpenMP
4  #PBS -q ac
5
6  export OMP_NUM_THREADS=12
7  export OMP_SCHEDULE="static"
8  $PBS_O_WORKDIR/pmtv-OpenMP 24832
9  export OMP_SCHEDULE="static, 1"
10 $PBS_O_WORKDIR/pmtv-OpenMP 24832
11 export OMP_SCHEDULE="static, 64"
12 $PBS_O_WORKDIR/pmtv-OpenMP 24832
13
14 export OMP_NUM_THREADS=12
15 export OMP_SCHEDULE="dynamic"
16 $PBS_O_WORKDIR/pmtv-OpenMP 24832
17 export OMP_SCHEDULE="dynamic, 1"
18 $PBS_O_WORKDIR/pmtv-OpenMP 24832
19 export OMP_SCHEDULE="dynamic, 64"
20 $PBS_O_WORKDIR/pmtv-OpenMP 24832
21
22 export OMP_NUM_THREADS=12
23 export OMP_SCHEDULE="guided"
24 $PBS_O_WORKDIR/pmtv-OpenMP 24832
25 export OMP_SCHEDULE="guided, 1"
26 $PBS_O_WORKDIR/pmtv-OpenMP 24832
27 export OMP_SCHEDULE="guided, 64"
28 $PBS_O_WORKDIR/pmtv-OpenMP 24832
29

```

Tabla 3. Tiempos de ejecución de la versión paralela del producto de una matriz triangular por un vector r **para vectores de tamaño N= 24832 , 12 threads**

Chunk	Static	Dynamic	Guided
por defecto	0.205197	0.169254	0.167756
1	0.228637	0.176050	0.184019
64	0.170376	0.168399	0.164816

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

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

NOTAS: (1) el número de filas/columnas debe ser un argumento de entrada; (2) se deben inicializar las matrices antes del cálculo; (3) se debe imprimir siempre las componentes (0,0) y (N-1, N-1) del resultado antes de que termine el programa.

CAPTURA CÓDIGO FUENTE: pmm-secuencial.c

```

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 matriz\n");
8          exit(-1);
9      }
10
11     unsigned int N = atoi(argv[1]);
12     double **A, **B,**C;
13
14     // Reserva de espacio
15
16     A = (double**) malloc(N*sizeof(double *));
17     B = (double**) malloc(N*sizeof(double *));
18     C = (double**) malloc(N*sizeof(double *));
19
20     if ( (A==NULL) || (B==NULL) || (C==NULL) ){
21         printf("Error en la reserva de espacio para las matrices\n");
22         exit(-2);
23     }
24
25     int i, j, k;
26     double suma;
27
28     for ( i = 0; i < N; i++ ) {
29         A[i] = (double*) malloc(N*sizeof(double));
30         B[i] = (double*) malloc(N*sizeof(double));
31         C[i] = (double*) malloc(N*sizeof(double));
32
33         if ( A[i] == NULL || B[i] == NULL || C[i] == NULL ) {
34             printf("Error en la reserva de espacio para los vectores\n");
35             exit(-2);
36         }
37     }
38 }
39

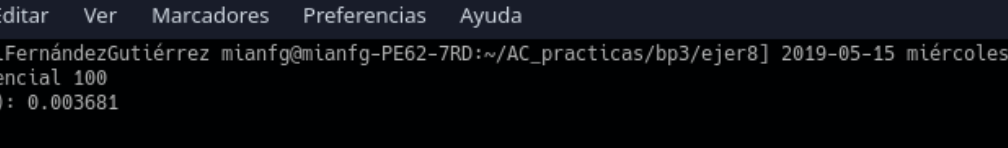
```

```

41 // Inicialización matrices B, C
42 for ( i = 0; i < N; i++ )
43     for ( j = 0; j < N; j++ ) {
44         B[i][j]=j+1;
45         C[i][j]=j+1;
46     }
47
48 // Tiempo
49 double t = omp_get_wtime();
50
51 // Cálculo de A=B*C
52 for ( i = 0; i < N; i++ )
53     for ( j = 0; j < N; j++ ) {
54         A[i][j] = 0;
55         for ( k = 0; k < N; k++ )
56             A[i][j] = A[i][j]+B[i][k]*C[k][j];
57     }
58
59 // Tiempo
60 t = omp_get_wtime() - t;
61
62 // Impresión de tiempo de ejecución
63 printf("Tiempo (seg): %f\n", t);
64
65 // Impresión de resultados
66 printf("\n_____ \nResultados:\n");
67
68 printf("\nMatriz B: \n");
69 if ( N < 10 )
70     for ( i = 0; i < N; i++ ) {
71         for ( j = 0; j < N; j++ )
72             printf("%f ", B[i][j]);
73         printf("\n");
74     }
75 else
76     printf("B[0][0]=%f B[%d][%d]=%f\n", B[0][0], N-1, N-1, B[N-1][N-1]);
77
78 printf("\nMatriz C: \n");
79 if ( N < 10 )
80     for ( i = 0; i < N; i++ ) {
81         for ( j = 0; j < N; j++ )
82             printf("%f ", C[i][j]);
83         printf("\n");
84     }
85 else
86     printf("C[0][0]=%f C[%d][%d]=%f\n", C[0][0], N-1, N-1, C[N-1][N-1]);
87
88 printf("\nMatriz A=B*C: \n");
89 if ( N < 10 )
90     for ( i = 0; i < N; i++ ) {
91         for ( j = 0; j < N; j++ )
92             printf("%f ", A[i][j]);
93         printf("\n");
94     }
95 else
96     printf("A[0][0]=%f A[%d][%d]=%f\n", A[0][0], N-1, N-1, A[N-1][N-1]);
97
98 printf("\n");
99
100 // Liberar
101 for (i=0; i<N; i++)
102     free(A[i]); free(B[i]); free(C[i]);
103
104 free(A); free(B); free(C);
105
106 return 0;
107 }
108

```

CAPTURAS DE PANTALLA:



```
ejer8 : bash — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer8] 2019-05-15 miércoles
$./pmm-secuencial 100
Tiempo (seg): 0.003681

Resultados:

Matriz B:
B[0][0]=1.000000 B[99][99]=100.000000

Matriz C:
C[0][0]=1.000000 C[99][99]=100.000000

Matriz A=B*C:
A[0][0]=5050.000000 A[99][99]=505000.000000

[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer8] 2019-05-15 miércoles
$
```

9. Implementar en paralelo la multiplicación de matrices cuadradas con OpenMP a partir del código escrito en el ejercicio anterior. Use las directivas, las cláusulas y las funciones de entorno que considere oportunas. Se debe paralelizar también la inicialización de las matrices. Dibuje en su cuaderno de prácticas la descomposición de dominio que ha utilizado en el código paralelo implementado para asignar tareas a los threads (Lección 4/Tema 2, Lección 5/Tema 2).

DESCOMPOSICIÓN DE DOMINIO:

CAPTURA CÓDIGO FUENTE: pmm-OpenMP.c

```

41 // Inicialización matrices B, C
42 #pragma omp parallel shared(B,C) private(i,j)
43 {
44     #pragma omp for schedule(runtime)
45     for ( i = 0; i < N; i++ )
46         for ( j = 0; j < N; j++ )
47             A[i][j]=0;
48
49     #pragma omp for schedule(runtime)
50     for ( i = 0; i < N; i++ )
51         for ( j = 0; j < N; j++ )
52             B[i][j]=j+1;
53
54     #pragma omp for schedule(runtime)
55     for ( i = 0; i < N; i++ )
56         for ( j = 0; j < N; j++ )
57             C[i][j]=j+1;
58 }
59
60 // Cálculo de A=B*C
61 #pragma omp parallel shared(A,B,C) private(i,j,k)
62 {
63     // Tiempo
64     #pragma omp single
65     {
66         t = omp_get_wtime();
67     }
68
69     #pragma omp for schedule(runtime)
70     for ( i = 0; i < N; i++ )
71         for ( j = 0; j < N; j++ ) {
72             A[i][j] = 0;
73             for ( k = 0; k < N; k++ )
74                 A[i][j] = A[i][j]+B[i][k]*C[k][j];
75         }
76
77     #pragma omp single
78     {
79         t = omp_get_wtime() - t;
80     }
81 }
82
83 // Impresión de tiempo de ejecución
84 printf("Tiempo (seg): %f\n", t);
85
86 // Impresión de resultados
87

```

Nota: aparecen únicamente las modificaciones al código anterior

CAPTURAS DE PANTALLA:

```

ejer9 : bash — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer9] 2019-05-15 miércoles
$ ./pmm-OpenMP 100
Tiempo (seg): 0.005113

Resultados:

Matriz B:
B[0][0]=1.000000 B[99][99]=100.000000

Matriz C:
C[0][0]=1.000000 C[99][99]=100.000000

Matriz A=B*C:
A[0][0]=5050.000000 A[99][99]=505000.000000

[MiguelÁngelFernándezGutiérrez mianfg@mianfg-PE62-7RD:~/AC_practicas/bp3/ejer9] 2019-05-15 miércoles
$

```

10. Hacer un estudio de escalabilidad (ganancia en velocidad en función del número de cores) en atcgrid y en su PC del código paralelo implementado para dos tamaños de las matrices. Debe recordar usar `-O2` al compilar. El número de núcleos máximo en este estudio debe ser el igual al de núcleos físicos del computador. Presente los resultados del estudio en tablas de valores y en gráficas. Escoger los tamaños de manera que se observe diferentes curvas de escalabilidad en las gráficas que entregue en su cuaderno de prácticas (pruebe con valores de N entre 100 y 1500). Consulte la Lección 6/Tema 2. Incluya los scripts utilizados en el cuaderno de prácticas. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

nmtbdpdpb

ESTUDIO DE ESCALABILIDAD EN atcgrid:

SCRIPT: pmm-OpenMP_atcgrid.sh

```
#!/bin/bash

#PBS -N pmtv-OpenMP
#PBS -q ac

echo "N=200"
for i in `seq 1 8`
do
    export OMP_NUM_THREADS=$i
    $PBS_O_WORKDIR/pmm-OpenMP 200
done

echo "N=1000"
for i in `seq 1 8`
do
    export OMP_NUM_THREADS=$i
    $PBS_O_WORKDIR/pmm-OpenMP 1000
done
```

ESTUDIO DE ESCALABILIDAD EN PCLOCAL:

SCRIPT: pmm-OpenMP_pclocal.sh

```
#!/bin/bash

echo "N=200"
for i in `seq 1 8`
do
    export OMP_NUM_THREADS=$i
    ./pmm-OpenMP 200
done

echo "N=1000"
for i in `seq 1 8`
do
    export OMP_NUM_THREADS=$i
    ./pmm-OpenMP 1000
done
```

