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):
Grupo de prácticas:
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, n=20, tid;
    int a[n], suma=0, sumalocal;
    if(argc<3){
        fprintf(stderr,"Uso: ./%s <iteraciones> <hebras>",argv[0]);
        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 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
        #pragma omp master
            printf("thread master=%d imprime suma=%d\n",tid,suma);
    }
}
```

CAPTURAS DE PANTALLA:

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer1] 2019-05-08 miércoles
$gcc -O2 -fopenmp if-clauseModificado.c -o if-clauseModificado
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer1] 2019-05-08 miércoles
$./if-clauseModificado 3 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 master=0 imprime suma=3
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer1] 2019-05-08 miércoles
$./if-clauseModificado 5 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 2 suma de a[3]=3 sumalocal=3
 thread 3 suma de a[4]=4 sumalocal=4
thread master=0 imprime suma=10
```

RESPUESTA:

Si uso un número de iteraciones menor que el que me indica el if, usará una sola hebra ya que es lo que viene especificado, sino usará el número de hebras que le especifiquemos en el segundo argumento.

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)
- chunck= 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			schedule-claused.c			schedule-clauseg.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	1	0	1	1	1
9	1	0	0	0	1	0	1	1	1
10	0	1	0	0	1	0	1	1	1
11	1	1	0	0	1	0	1	1	1
12	0	0	1	1	1	1	0	0	1
13	1	0	1	1	1	1	0	0	1
14	0	1	1	1	1	1	0	0	1
15	1	1	1	0	1	1	0	0	1

(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			schedule-claused.c			schedule-clauseg.c		
	1	2	4	1	2	4	1	2	4
0	0	0	0	2	0	2	2	1	0
1	1	0	0	0	0	2	2	1	0
2	2	1	0	1	1	2	2	1	0
3	3	1	0	3	1	2	2	1	0
4	0	2	1	1	2	1	0	2	1
5	1	2	1	1	2	1	0	2	1
6	2	3	1	1	3	1	0	2	1
7	3	3	1	1	3	1	3	3	1
8	0	0	2	3	0	3	3	3	2
9	1	0	2	2	0	3	3	3	2
10	2	1	2	2	0	3	1	0	2
11	3	1	2	2	0	3	1	0	2
12	0	2	3	2	0	0	2	1	3
13	1	2	3	2	0	0	2	1	3
14	2	3	3	2	0	0	2	1	3
15	3	3	3	2	0	0	2	1	3

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

RESPUESTA:

Con static se asigna un único chunk a cada thread, las iteraciones se dividen en unidaes de chunk iteraciones y las unidades se asignan en round-robin. con dynamic las iteraciones se dividen en unidades de chunk iteraciones y las unidades se asignan en tiempo de ejecución por lo que los threads más rápidos ejecutan más unidades. Con guided la distribución se hace en tiempo de ejecución, comienza con un bloque grande y el tamaño del bloque va menguando en cada iteración pero nunca es más pequeño que chunk (excepto la última):

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`

```
#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, modifier;
    omp_sched_t kind;
    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
    {
        #pragma omp for firstprivate(suma) lastprivate(suma) schedule(dynamic,chunk)
        for(i=0;i<n;i++){
            suma =suma +a[i];
            printf(" thread %d suma a[%d]=%d suma=%d \n",
                omp_get_thread_num(),i,a[i],suma);
        }
        #pragma omp single
        {
            printf("Variables de control dentro de parallel:\n");
            printf("dyn-var=%d\n",omp_get_dynamic());
            printf("nthreads-var=%d\n",omp_get_max_threads());
            printf("thread-limit-var=%d\n",omp_get_thread_limit());
            omp_get_schedule(&kind,&modifier);
            printf("run-sched-var: kind=%d, modifier=%d\n",kind, modifier);
        }
    }
```

```
    printf("Variables de control fuera de parallel:\n");
            printf("dyn-var=%d\n",omp_get_dynamic());
            printf("nthreads-var=%d\n",omp_get_max_threads());
            printf("thread-limit-var=%d\n",omp_get_thread_limit());
            omp_get_schedule(&kind,&modifier);
            printf("run-sched-var: kind=%d, modifier=%d\n",kind, modifier);
    printf("Fuera de 'parallelfor' suma=%d\n",suma);

}
```

CAPTURAS DE PANTALLA:

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer3] 2019-05-14 martes
$export OMP_NUM_THREADS=4
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer3] 2019-05-14 martes
$export OMP_DYNAMIC=FALSE
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer3] 2019-05-14 martes
$gcc -O2 -fopenmp scheduled-clause.c -o scheduled-clause
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer3] 2019-05-14 martes
$./scheduled-clause 16 4
 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[3]=3 suma=6
 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 3 suma a[12]=12 suma=12
 thread 3 suma a[13]=13 suma=25
 thread 3 suma a[14]=14 suma=39
 thread 3 suma a[15]=15 suma=54
 thread 0 suma a[8]=8 suma=8
 thread 0 suma a[9]=9 suma=17
 thread 0 suma a[10]=10 suma=27
 thread 0 suma a[11]=11 suma=38
Variables de control dentro de parallel:
dyn-var=0
nthreads-var=4
thread-limit-var=2147483647
run-sched-var: kind=2, modifier=1
Variables de control fuera de parallel:
dyn-var=0
nthreads-var=4
thread-limit-var=2147483647
run-sched-var: kind=2, modifier=1
Fuera de 'parallelfor' suma=54
```

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer3] 2019-05-14 martes
$export OMP_NUM_THREADS=2
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer3] 2019-05-14 martes
$export OMP_DYNAMIC=TRUE
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer3] 2019-05-14 martes
$export OMP_SCHEDULE="static,4"
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer3] 2019-05-14 martes
$export OMP_THREAD_LIMIT=4
```

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer3] 2019-05-14 martes
$./scheduled-clause 16 4
 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 0 suma a[8]=8 suma=14
 thread 0 suma a[9]=9 suma=23
 thread 0 suma a[10]=10 suma=33
 thread 0 suma a[11]=11 suma=44
 thread 0 suma a[12]=12 suma=56
 thread 0 suma a[13]=13 suma=69
 thread 0 suma a[14]=14 suma=83
 thread 0 suma a[15]=15 suma=98
 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
Variables de control dentro de parallel:
dyn-var=1
nthreads-var=2
thread-limit-var=4
run-sched-var: kind=1, modifier=4
Variables de control fuera de parallel:
dyn-var=1
nthreads-var=2
thread-limit-var=4
run-sched-var: kind=1, modifier=4
Fuera de 'parallelfor' suma=98
```

RESPUESTA:

Sí se imprimen los mismos valores dentro y fuera de parallel.

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`

```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, modifier;
    omp_sched_t kind;
    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
    {
        #pragma omp for firstprivate(suma) lastprivate(suma) schedule(dynamic,chunk)
        for(i=0;i<n;i++){
            suma =suma +a[i];
            printf(" thread %d suma a[%d]=%d suma=%d \n",
                omp_get_thread_num(),i,a[i],suma);
        }
        #pragma omp single
        {
            printf("Variables de control dentro de parallel:\n");
            printf("num_threads=%d\n",omp_get_num_threads());
            printf("num_procs=%d\n",omp_get_num_procs());
            printf("omp_in_parallel=%d\n",omp_in_parallel());
        }
    }

    printf("Variables de control fuera de parallel:\n");
            printf("num_threads=%d\n",omp_get_num_threads());
            printf("num_procs=%d\n",omp_get_num_procs());
            printf("omp_in_parallel=%d\n",omp_in_parallel());
    printf("Fuera de 'parallelfor' suma=%d\n",suma);

}
```

CAPTURAS DE PANTALLA:

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer4] 2019-05-14 martes
$gcc -O2 -fopenmp scheduled-clauseModificado4.c -o scheduled-clauseModificado4
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer4] 2019-05-14 martes
$./scheduled-clauseModificado4 16 4
 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[3]=3 suma=6
 thread 3 suma a[12]=12 suma=12
 thread 3 suma a[13]=13 suma=25
 thread 3 suma a[14]=14 suma=39
 thread 3 suma a[15]=15 suma=54
 thread 0 suma a[8]=8 suma=8
 thread 0 suma a[9]=9 suma=17
 thread 0 suma a[10]=10 suma=27
 thread 0 suma a[11]=11 suma=38
 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
Variables de control dentro de parallel:
num_threads=4
num_procs=8
omp_in_parallel=1
Variables de control fuera de parallel:
num_threads=1
num_procs=8
omp_in_parallel=0
Fuera de 'parallelfor' suma=54
```

RESPUESTA:

Cambian las variables num_threads y 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`

```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, modifier;
    omp_sched_t kind;
    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
    {
        #pragma omp for firstprivate(suma) lastprivate(suma) schedule(dynamic,chunk)
        for(i=0;i<n;i++){
            suma =suma +a[i];
            printf(" thread %d suma a[%d]=%d suma=%d \n",
                omp_get_thread_num(),i,a[i],suma);
        }
    }

    printf("Antes de la modificación:\n");
    printf("dyn-var=%d\n",omp_get_dynamic());
    printf("nthreads-var=%d\n",omp_get_max_threads());
    omp_get_schedule(&kind,&modifier);
    printf("run-sched-var: kind=%d, modifier=%d\n",kind, modifier);

    omp_set_dynamic(1);
    omp_set_num_threads(2);
    omp_set_schedule(4, 2);

    printf("Después de la modificación:\n");
    printf("dyn-var=%d\n",omp_get_dynamic());
    printf("nthreads-var=%d\n",omp_get_max_threads());
    omp_get_schedule(&kind,&modifier);
    printf("run-sched-var: kind=%d, modifier=%d\n",kind, modifier);
    printf("Fuera de 'parallelfor' suma=%d\n",suma);

}
```

CAPTURAS DE PANTALLA:

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer5] 2019-05-14 martes
$gcc -O2 -fopenmp scheduled-clauseModificado5.c -o scheduled-clauseModificado5
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer5] 2019-05-14 martes
$./scheduled-clauseModificado5 16 4
 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[3]=3 suma=6
 thread 3 suma a[12]=12 suma=12
 thread 3 suma a[13]=13 suma=25
 thread 0 suma a[8]=8 suma=8
 thread 0 suma a[9]=9 suma=17
 thread 0 suma a[10]=10 suma=27
 thread 0 suma a[11]=11 suma=38
 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 3 suma a[14]=14 suma=39
 thread 3 suma a[15]=15 suma=54
Antes de la modificación:
dyn-var=0
nthreads-var=4
run-sched-var: kind=1, modifier=4
Después de la modificación:
dyn-var=1
nthreads-var=2
run-sched-var: kind=4, modifier=4
Fuera de 'parallelfor' suma=54
```

RESPUESTA:

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`

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

int main(int argc, char ** argv){
    if (argc != 2){
        printf("Faltan las dimensiones de la matriz y vector\n");
        exit(1);
    }

    int i, j, contador, N = atoi(argv[1]);
    double **m, *v, *resultado, t_inicio, t_fin;

    m = (double **)malloc(N*sizeof(double *));

    for(i=0; i<N; i++){
        m[i] = (double*)malloc(N*sizeof(double));
    }

    resultado = (double *)malloc(N*sizeof(double));

    v = (double*)malloc(N*sizeof(double));

    contador = 1;

    // Matriz triangular inferior

    for(i=0; i<N; i++){
        for(j=0; j<N; j++){
            m[i][j] = contador;
            contador++;
        }
    }

    for(i=0; i<N; i++){
        for(j=i+1; j<N; j++){
            m[i][j] = 0;
        }
    }

    contador = 1;

    for(i=0; i<N; i++){
        v[i] = contador;
        contador++;
    }

    for(i=0; i<N; i++){
        resultado[i] = 0;
    }

    t_inicio = omp_get_wtime();

    for(i=0; i<N; i++){
        for(j=0; j<i+1; j++){
            resultado[i] += m[i][j]*v[j];
        }
    }

    t_fin = omp_get_wtime();

    double tiempo = t_fin - t_inicio;

    printf("resultado[0]=%f",resultado[0]);
    printf("\nresultado[N-1]=%f\n",resultado[N-1]);

    printf("Tiempo: %f\n", tiempo);

    free(v);

    for(i=0; i<N; i++){
        free(m[i]);
    }

    free(m);

    free(resultado);


    return (0);
}
```

CAPTURAS DE PANTALLA:

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer6] 2019-05-13 lunes
$gcc -O2 -fopenmp pmtv-secuencial.c -o pmtv-secuencial
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer6] 2019-05-13 lunes
$./pmtv-secuencial 4
resultado[0]=1.000000
resultado[N-1]=150.000000
Tiempo: 0.000001
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer6] 2019-05-13 lunes
$./pmtv-secuencial 15
resultado[0]=1.000000
resultado[N-1]=26440.000000
Tiempo: 0.000001
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer6] 2019-05-13 lunes
$./pmtv-secuencial 20
resultado[0]=1.000000
resultado[N-1]=82670.000000
Tiempo: 0.000002
```

El código de la práctica anterior era de n^2 y el de esta práctica es de $(n^2-c)/2$.

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:

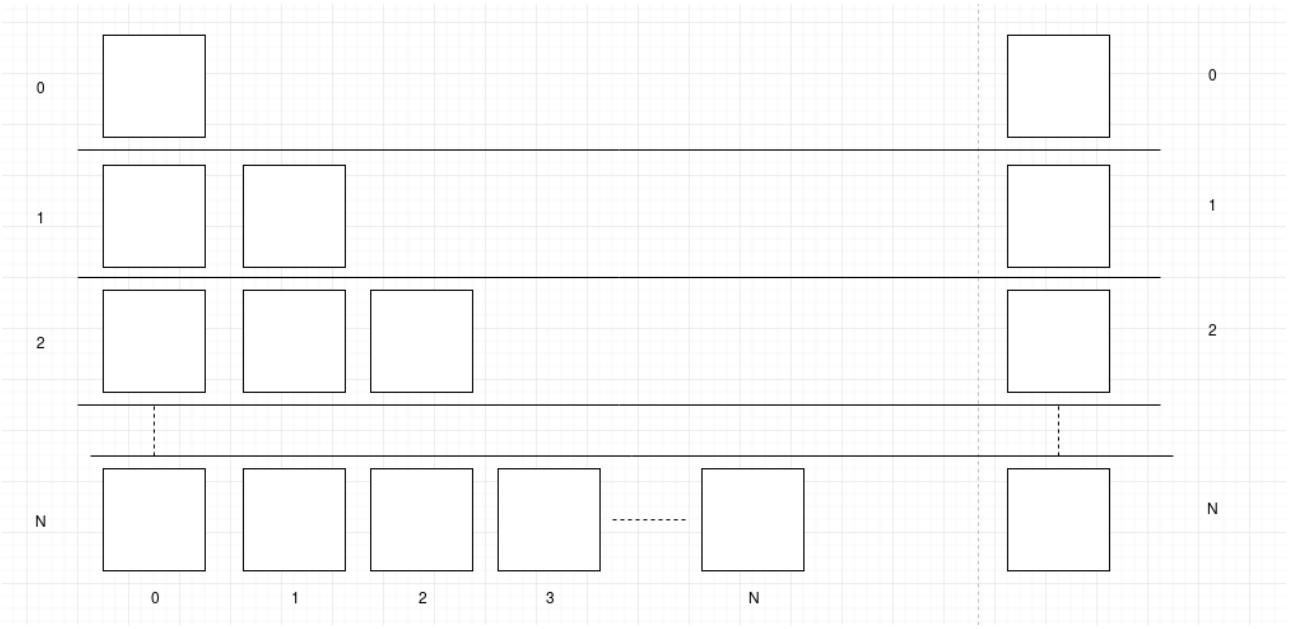
a) El valor por defecto del chunk en static es 0, para dynamic es 1 y para guided es 1 también. Para obtener el valor por defecto he cambiado la variable de entorno OMP_SCHEDULE a “static”, “dynamic” y “guided”, luego lo he comprobado con omp_get_schedule(&kind,&chunk) en el programa.

c) No se sabe exactamente ya que la asignación se realiza en tiempo de ejecución.

CAPTURA CÓDIGO FUENTE: `pmtv-OpenMP.c`

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

int main(int argc, char ** argv){
    if (argc != 2){
        printf("Faltan las dimensiones de la matriz y vector\n");
        exit(1);
    }

    int i, j, contador, N = atoi(argv[1]);
    double **m, *v, *resultado, t_inicio, t_fin;

    m = (double **)malloc(N*sizeof(double *));

    for(i=0; i<N; i++){
        m[i] = (double*)malloc(N*sizeof(double));
    }

    resultado = (double *)malloc(N*sizeof(double));

    v = (double*)malloc(N*sizeof(double));

    contador = 1;

    // Matriz triangular inferior

    for(i=0; i<N; i++){
        for(j=0; j<N; j++){
            m[i][j] = contador;
            contador++;
        }
    }

    for(i=0; i<N; i++){
        for(j=i+1; j<N; j++){
            m[i][j] = 0;
        }
    }

    contador = 1;

    for(i=0; i<N; i++){
        v[i] = contador;
        contador++;
    }

    for(i=0; i<N; i++){
        resultado[i] = 0;
    }

    t_inicio = omp_get_wtime();

    #pragma omp parallel private(j)
    {
        #pragma omp for schedule(runtime)
        for(i=0; i<N; i++){
            for(j=0; j<i+1; j++){
                resultado[i] += m[i][j]*v[j];
            }
        }

    }

    t_fin = omp_get_wtime();

    double tiempo = t_fin - t_inicio;

    printf("resultado[0]=%f",resultado[0]);
    printf("\nresultado[N-1]=%f\n",resultado[N-1]);

    printf("Tiempo: %f\n", tiempo);

    free(v);

    for(i=0; i<N; i++){
        free(m[i]);
    }

    free(m);

    free(resultado);


    return (0);
}
```

DESCOMPOSICIÓN DE DOMINIO:



CAPTURAS DE PANTALLA:

```
[NoeliaEscaleraMejias A2estudiante5@atcgrid:~/bp3/ejer7] 2019-05-13 lunes
$gcc -O2 -fopenmp pmtv-OpenMP.c -o pmtv-OpenMP
[NoeliaEscaleraMejias A2estudiante5@atcgrid:~/bp3/ejer7] 2019-05-13 lunes
$echo '/home/A2estudiante5/bp3/ejer7/pmtv-OpenMP_atcgrid.sh' | qsub -q ac -N 'pmtv'
21058.atcgrid
[NoeliaEscaleraMejias A2estudiante5@atcgrid:~/bp3/ejer7] 2019-05-13 lunes
$ls
pmtv-OpenMP  pmtv-OpenMP_atcgrid.sh  pmtv-OpenMP.c
[NoeliaEscaleraMejias A2estudiante5@atcgrid:~/bp3/ejer7] 2019-05-13 lunes
$ls
pmtv.e21058  pmtv.o21058  pmtv-OpenMP  pmtv-OpenMP_atcgrid.sh  pmtv-OpenMP.c
```

```
[NoeliaEscaleraMejias A2estudiante5@atcgrid:~/bp3/ejer7] 2019-05-13 lunes
$echo '/home/A2estudiante5/bp3/ejer7/pmtv-OpenMP_atcgrid.sh' | qsub -q ac -N 'pmtv2'
21060.atcgrid
[NoeliaEscaleraMejias A2estudiante5@atcgrid:~/bp3/ejer7] 2019-05-13 lunes
$ls
pmtv2.e21060  pmtv2.o21060  pmtv.e21058  pmtv.o21058  pmtv-OpenMP  pmtv-OpenMP_atcgrid.sh  pmtv-OpenMP.c
```

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid

SCRIPT: pmvt-OpenMP_atcgrid.sh

```
#!/bin/bash
echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
echo "Id. del trabajo: $PBS_JOBID"
echo "Nombre  del trabajo especificado por usuario: $PBS_JOBNAME"
echo "Nodo que ejecuta qsub: $PBS_O_HOST"
echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
echo "Cola: $PBS_QUEUE"
echo "Nodos asignados al trabajo:"
cat $PBS_NODEFILE
export OMP_NUM_THREADS=12;

export OMP_SCHEDULE="static";
~/bp3/ejer7/pmtv-OpenMP 15360

export OMP_SCHEDULE="static,1";
~/bp3/ejer7/pmtv-OpenMP 15360

export OMP_SCHEDULE="static,64";
~/bp3/ejer7/pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic";
~/bp3/ejer7/pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic,1";
~/bp3/ejer7/pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic,64";
~/bp3/ejer7/pmtv-OpenMP 15360

export OMP_SCHEDULE="guided";
~/bp3/ejer7/pmtv-OpenMP 15360

export OMP_SCHEDULE="guided,1";
~/bp3/ejer7/pmtv-OpenMP 15360

export OMP_SCHEDULE="guided,64";
~/bp3/ejer7/pmtv-OpenMP 15360
```

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

Chunk	Static	Dynamic	Guided
por defecto	0.064298	0.074974	0.055008
1	0.062365	0.061492	0.054753
64	0.058868	0.055325	0.055255

Chunk	Static	Dynamic	Guided
por defecto	0.098162	0.063314	0.054138
1	0.060467	0.064071	0.054804
64	0.059724	0.055424	0.054269

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

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

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`

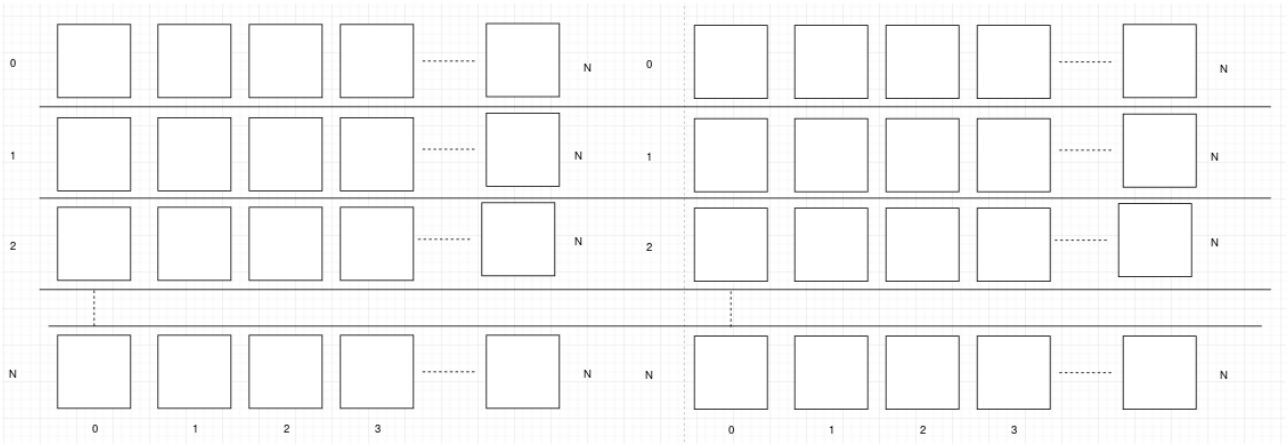
```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>

int main(int argc, char ** argv){
    if (argc != 2){
        printf("Faltan las dimensiones de la matriz y vector\n");
        exit(1);
    }

    int i, j, k, contador, N = atoi(argv[1]);
    double **m1, **m2, **resultado, t_inicio, t_fin;

    m1 = (double **)malloc(N*sizeof(double *));

    for(i=0; i<N; i++){
        m1[i] = (double*)malloc(N*sizeof(double));
    }

    resultado = (double **)malloc(N*sizeof(double));

    for(i=0; i<N; i++){
        resultado[i] = (double*)malloc(N*sizeof(double));
    }

    m2 = (double **)malloc(N*sizeof(double *));

    for(i=0; i<N; i++){
        m2[i] = (double*)malloc(N*sizeof(double));
    }

    contador = 1;


    for(i=0; i<N; i++){
        for(j=0; j<N; j++){
            m1[i][j] = contador;
            contador++;
        }
    }

    contador = 1;
    for(i=0; i<N; i++){
        for(j=0; j<N; j++){
            m2[i][j] = contador;
            contador++;
        }
    }

    for(i=0; i<N; i++){
        for(j=0; j<N; j++){
            resultado[i][j] = 0;
        }
    }

    t_inicio = omp_get_wtime();

     for (i=0;i<N;i++){
        for (j=0;j<N;j++){
            for (k=0;k<N;k++){
                resultado[i][j]=resultado[i][j]+m1[i][k]*m2[k][j];
            }
        }
    }
    t_fin = omp_get_wtime();

    double tiempo = t_fin - t_inicio;

    printf("resultado[0][0]=%f",resultado[0][0]);
    printf("\nresultado[N-1][N-1]=%f\n",resultado[N-1][N-1]);

    printf("Tiempo: %f\n", tiempo);

    for(i=0; i<N; i++){
        free(m1[i]);
    }

    free(m1);

    for(i=0; i<N; i++){
        free(m2[i]);
    }

    free(m2);

    for(i=0; i<N; i++){
        free(resultado[i]);
    }

    free(resultado);


    return (0);
}
```

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer8] 2019-05-13 lu
nes
$gcc -O2 -fopenmp pmm-secuencial.c -o pmm-secuencial
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer8] 2019-05-13 lu
nes
$./pmm-secuencial 8
resultado[0][0]=1380.000000
resultado[N-1][N-1]=17760.000000
Tiempo: 0.000001
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer8] 2019-05-13 lu
nes
$./pmm-secuencial 20
resultado[0][0]=53410.000000
resultado[N-1][N-1]=1653400.000000
Tiempo: 0.000020
```

CAPTURAS DE PANTALLA:

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`

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

int main(int argc, char ** argv){
    if (argc != 2){
        printf("Faltan las dimensiones de la matriz y vector\n");
        exit(1);
    }

    int i, j, k, contador, N = atoi(argv[1]);
    double **m1, **m2, **resultado, t_inicio, t_fin;

    m1 = (double **)malloc(N*sizeof(double *));

    for(i=0; i<N; i++){
        m1[i] = (double*)malloc(N*sizeof(double));
    }

    resultado = (double **)malloc(N*sizeof(double));

    for(i=0; i<N; i++){
        resultado[i] = (double*)malloc(N*sizeof(double));
    }

    m2 = (double **)malloc(N*sizeof(double *));

    for(i=0; i<N; i++){
        m2[i] = (double*)malloc(N*sizeof(double));
    }

    contador = 1;
    int contador_2 = 1;

    #pragma omp parallel
    {
        #pragma omp sections
        {
            #pragma omp section
            for(i=0; i<N; i++){
                for(j=0; j<N; j++){
                    m1[i][j] = contador;
                    contador++;
                }
            }

            #pragma omp section
            for(i=0; i<N; i++){
                for(j=0; j<N; j++){
                    m2[i][j] = contador_2;
                    contador_2++;
                }
            }

            #pragma omp section
            for(i=0; i<N; i++){
                for(j=0; j<N; j++){
                    resultado[i][j] = 0;
                }
            }
        }
    }

    t_inicio = omp_get_wtime();

    #pragma omp parallel
    {
        #pragma omp for
        for (i=0;i<N;i++){
            for (j=0;j<N;j++){
                for (k=0;k<N;k++){
                    resultado[i][j]=resultado[i][j]+m1[i][k]*m2[k][j];
                }
            }
        }
    }

    t_fin = omp_get_wtime();

    double tiempo = t_fin - t_inicio;

    printf("resultado[0][0]=%f",resultado[0][0]);
    printf("\nresultado[N-1][N-1]=%f\n",resultado[N-1][N-1]);

    printf("Tiempo: %f\n", tiempo);

    for(i=0; i<N; i++){
        free(m1[i]);
    }

    free(m1);

    for(i=0; i<N; i++){
        free(m2[i]);
    }

    free(m2);

    for(i=0; i<N; i++){
        free(resultado[i]);
    }

    free(resultado);


    return (0);
}
```

CAPTURAS DE PANTALLA:

```
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer9] 2019-05-14 martes
$gcc -O2 -fopenmp pmm-OpenMP.c -o pmm-OpenMP
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer9] 2019-05-14 martes
$./pmm-OpenMP 8
resultado[0][0]=1380.000000
resultado[N-1][N-1]=17760.000000
Tiempo: 0.000008
[NoeliaEscaleraMejias noelia@noelia-HP-ENVY-17-Notebook-PC:~/Escritorio/Universidad/Segundo/AC/Prácticas/bp3/ejer9] 2019-05-14 martes
$./pmm-OpenMP 20
resultado[0][0]=53410.000000
resultado[N-1][N-1]=1653400.000000
Tiempo: 0.000050
```

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 utilizado en el cuaderno de prácticas. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

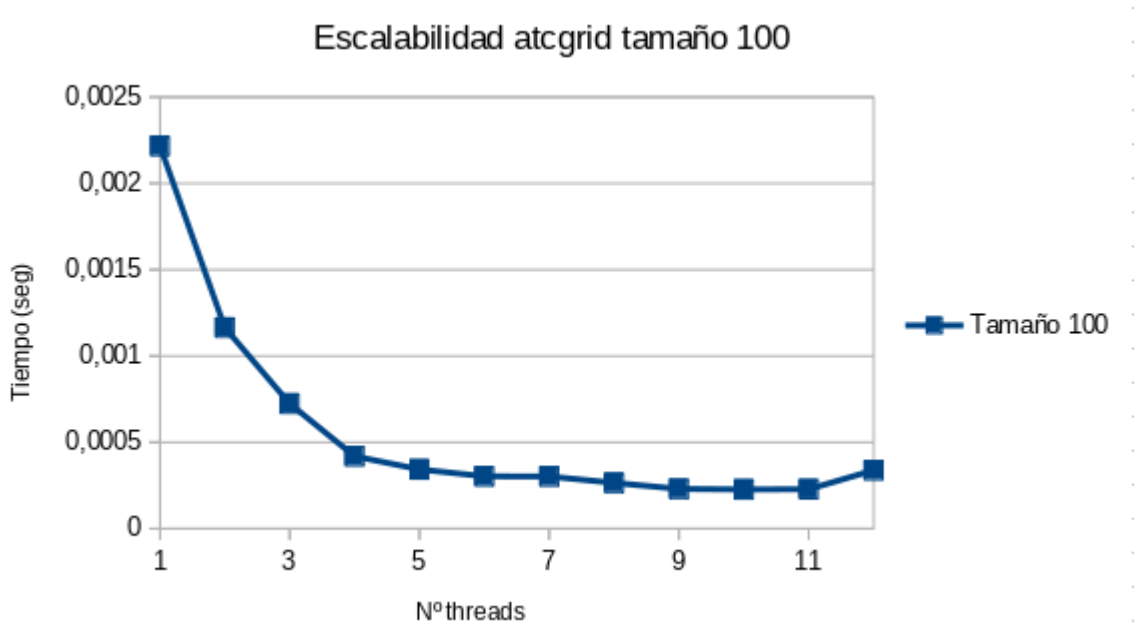
ESTUDIO DE ESCALABILIDAD EN atcgrid:

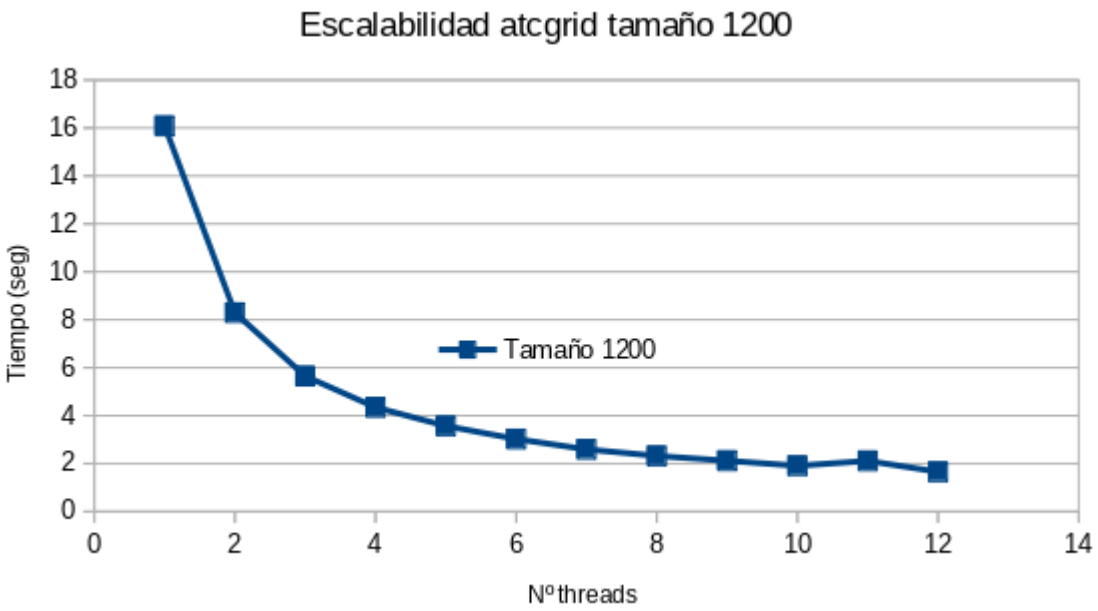
SCRIPT: pmm-OpenMP_atcgrid.sh

```
#!/bin/bash
echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
echo "Id. del trabajo: $PBS_JOBID"
echo "Nombre  del trabajo especificado por usuario: $PBS_JOBNAME"
echo "Nodo que ejecuta qsub: $PBS_O_HOST"
echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
echo "Cola: $PBS_QUEUE"
echo "Nodos asignados al trabajo:"
cat $PBS_NODEFILE
export OMP_DYNAMIC=FALSE;

for ((i=1;i<=12;i=i+1))
do
    echo "tiempos $i hebras"
    export OMP_NUM_THREADS=$i;

    echo "Tamaño 100"

    ~/bp3/ejer10/pmm-OpenMP 100

    echo "Tamaño 1200"

    ~/bp3/ejer10/pmm-OpenMP 1200

done
```

```
[NoeliaEscaleraMejias A2estudiante5@atcgrid:~/bp3/ejer10] 2019-05-14 martes
$gcc -O2 -fopenmp pmm-OpenMP.c -o pmm-OpenMP
[NoeliaEscaleraMejias A2estudiante5@atcgrid:~/bp3/ejer10] 2019-05-14 martes
$echo '/home/A2estudiante5/bp3/ejer10/pmm-OpenMP_atcgrid.sh' | qsub -q ac -N 'pmm-OpenMP'
21220.atcgrid
[NoeliaEscaleraMejias A2estudiante5@atcgrid:~/bp3/ejer10] 2019-05-14 martes
$ls
pmm-OpenMP  pmm-OpenMP_atcgrid.sh  pmm-OpenMP.c  pmm-OpenMP.e21220  pmm-OpenMP.o21220
```

N.º threads	Tamaño 100	Tamaño 1200
1	0,002216	16,083168
2	0,001164	8,292648
3	0,000724	5,648455
4	0,000419	4,345912
5	0,000343	3,567937
6	0,000303	3,025236
7	0,000302	2,594714
8	0,000265	2,319185
9	0,00023	2,117861
10	0,000227	1,908393
11	0,000228	2,110994
12	0,000338	1,656917



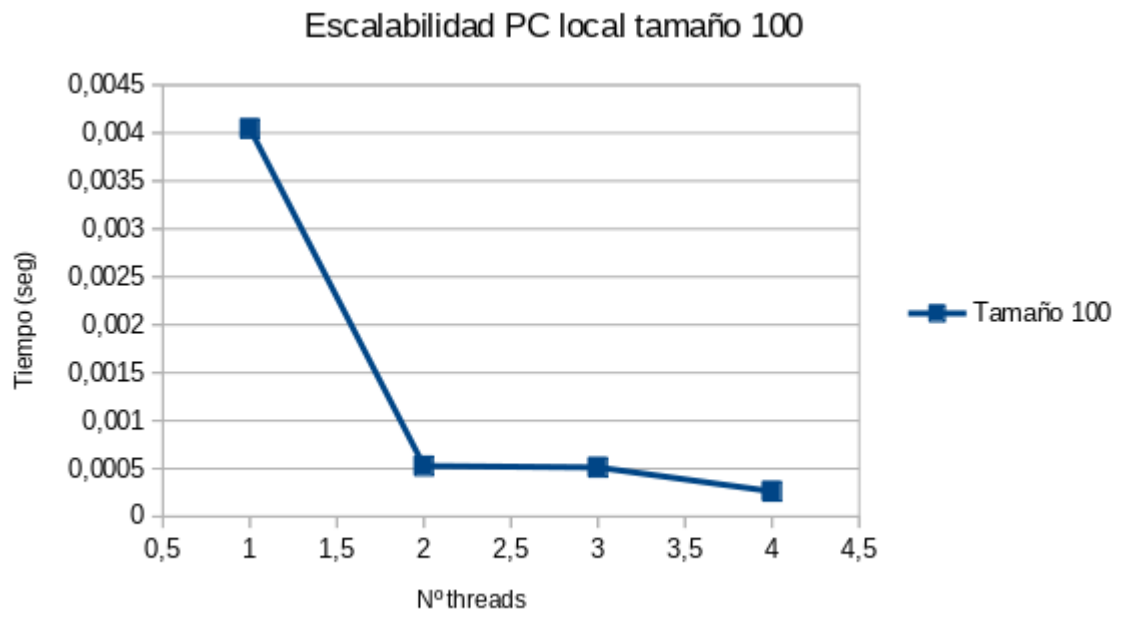


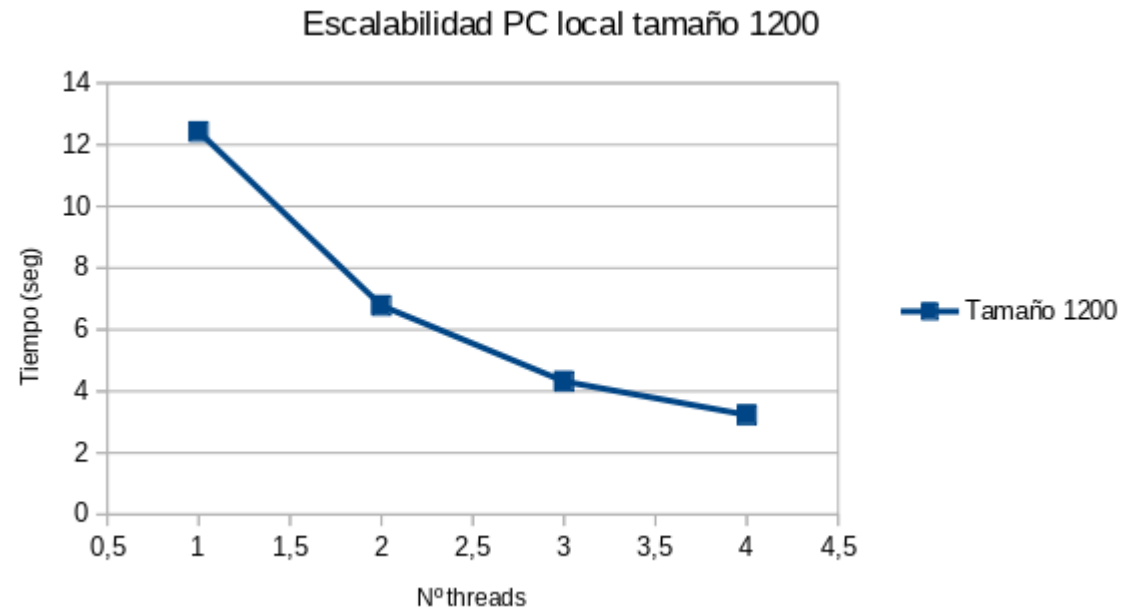
ESTUDIO DE ESCALABILIDAD EN PCLOCAL:

SCRIPT: pmm-OpenMP_pclocal.sh

```
#!/bin/bash
for ((i=1;i<=4;i=i+1))
do
    echo "tiempos $i hebras"
    export OMP_NUM_THREADS=$i;

    echo "Tamaño 100"

    ./pmm-OpenMP 100

    echo "Tamaño 1200"

    ./pmm-OpenMP 1200

done
```

N.º threads	Tamaño 100	Tamaño 1200
1	0,004046	12,42833
2	0,000528	6,781029
3	0,000513	4,319853
4	0,000263	3,23333


Escalabilidad PC local tamaño 100
Tiempo (seg)
0,0045
0,004
0,0035
0,003
0,0025
0,002
0,0015
0,001
0,0005
0
0,5
1
1,5
2
2,5
3
3,5
4
4,5
Nº threads
Tamaño 100


Escalabilidad PC local tamaño 1200
Tiempo (seg)
14
12
10
8
6
4
2
0
0,5
1
1,5
2
2,5
3
3,5
4
4,5
Nº threads
Tamaño 1200