

2º curso / 2º cuatr.
Grado Ing. Inform.
Doble Grado Ing.
Inform. y Mat.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 3. Programación paralela III: Interacción con el entorno en OpenMP

Estudiante (nombre y apellidos): Javier Gómez Luzón

Grupo de prácticas: C1

Fecha de entrega:

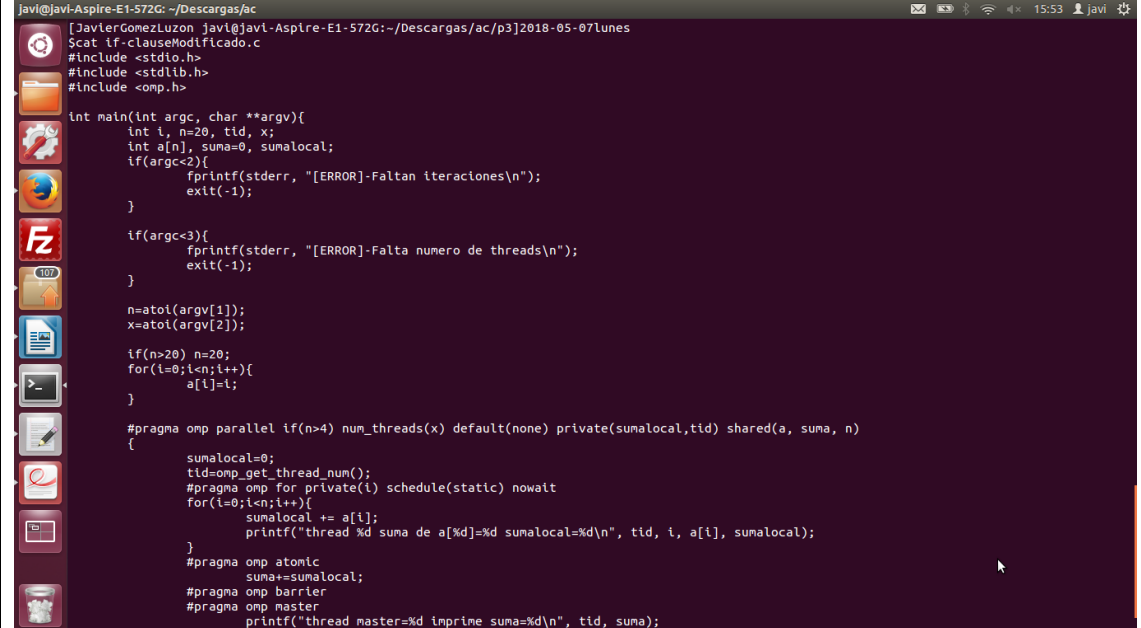
Fecha evaluación en clase:

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`

Imagen 1: Cat `if-clauseModificado.c`.



```
javi@javi-Aspire-E1-572G: ~/Descargas/ac
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07Lunes
$cat if-clauseModificado.c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

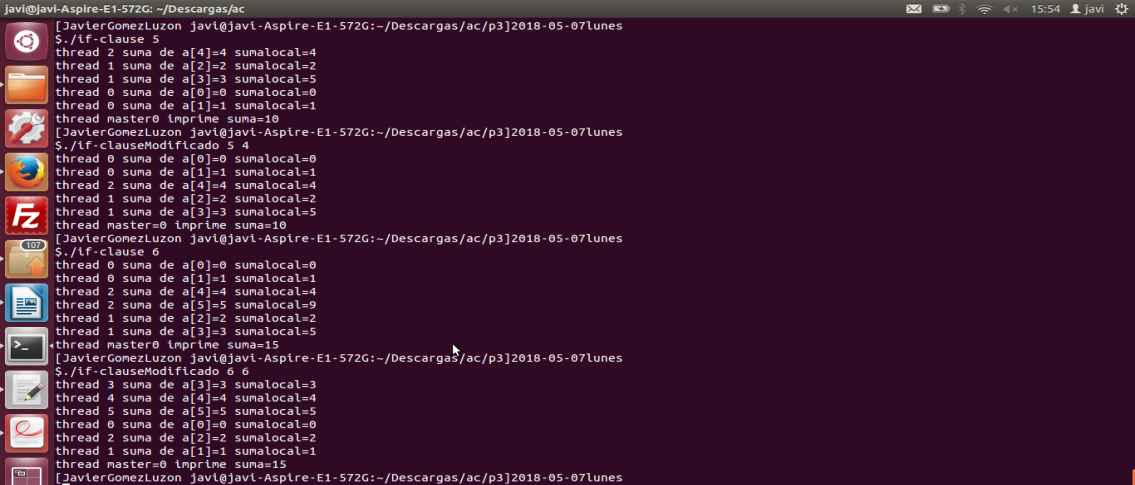
int main(int argc, char **argv){
    int i, n=20, tid, x;
    int a[n], suma=0, sumalocal;
    if(argc<2){
        fprintf(stderr, "[ERROR]-Faltan iteraciones\n");
        exit(-1);
    }
    if(argc<3){
        fprintf(stderr, "[ERROR]-Falta numero de threads\n");
        exit(-1);
    }
    n=atoi(argv[1]);
    x=atoi(argv[2]);

    if(n>20) n=20;
    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);
    }
}
```

CAPTURAS DE PANTALLA:

Imagen 2: Ejecución if-clause.c y if-clauseModificado.c.



RESPUESTA:

El código solo se puede paralelizar si el numero de iteraciones de superior a 4. Al poner num_threads() le podemos fijar el número de hebras sin tener que recompilar. Aunque solo se fijara si el numero de iteraciones es mayor que 4.

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

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

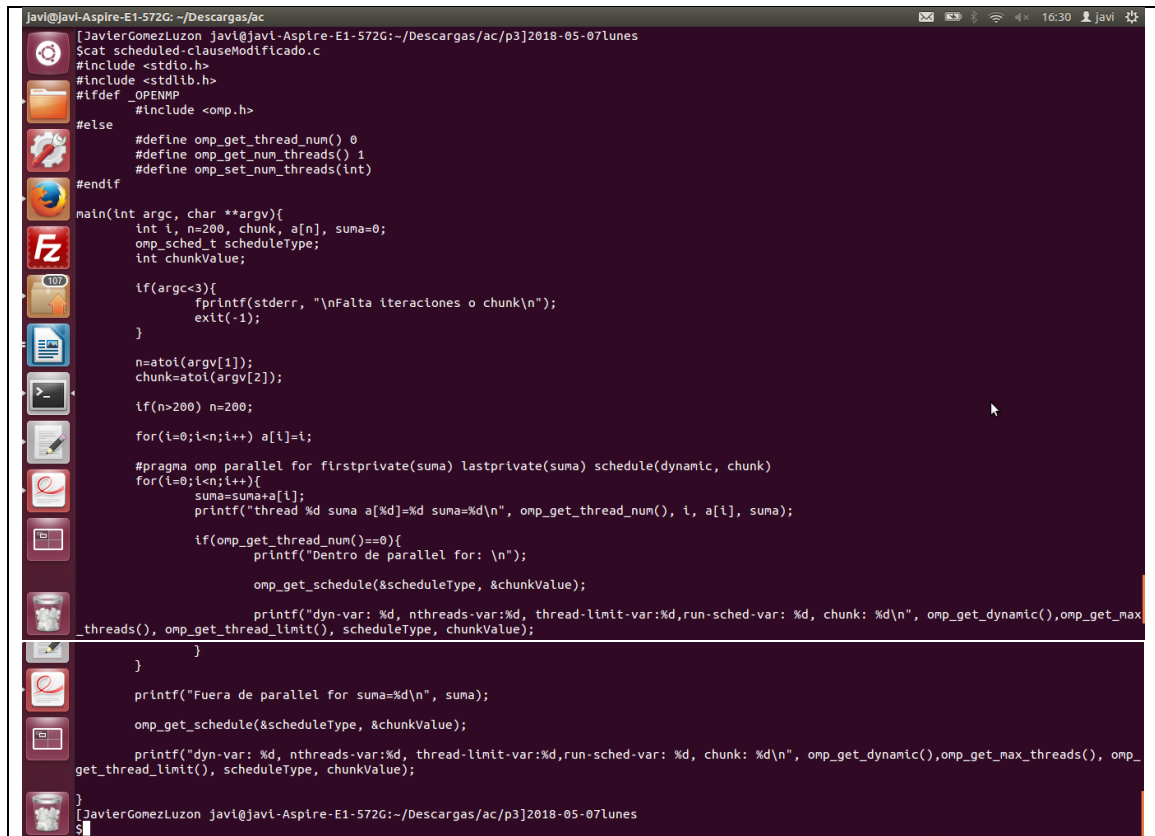
RESPUESTA:

Con `static` las tareas se reparten equitativamente usando `round-robin`, con `Dynamic` y `Guided` se repartirán las tareas aleatoriamente (pero su tamaño vendrá definido por el `chunk`).

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`

Imagen 3 y 4: Cat `scheduled-clauseModificado.c`.



```

javi@javi-Aspire-E1-572G: ~/Descargas/ac
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
Scat scheduled-clauseModificado.c
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#define omp_get_num_threads() 1
#define omp_set_num_threads(int)
#endif

main(int argc, char **argv){
    int i, n=200, chunk, a[n], suma=0;
    omp_sched_t scheduleType;
    int chunkValue;

    if(argc<3){
        fprintf(stderr, "\nFalta iteraciones o chunk\n");
        exit(-1);
    }

    n=atoi(argv[1]);
    chunk=atoi(argv[2]);

    if(n>200) n=200;

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

        if(omp_get_thread_num()==0){
            printf("Dentro de parallel for: \n");

            omp_get_schedule(&scheduleType, &chunkValue);

            printf("dyn-var: %d, nthreads-var:%d, thread-limit-var:%d,run-sched-var: %d, chunk: %d\n", omp_get_dynamic(),omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);

        }

        printf("Fuera de parallel for suma=%d\n", suma);

        omp_get_schedule(&scheduleType, &chunkValue);

        printf("dyn-var: %d, nthreads-var:%d, thread-limit-var:%d,run-sched-var: %d, chunk: %d\n", omp_get_dynamic(),omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);

    }

    printf("Fuera de parallel for suma=%d\n", suma);

    omp_get_schedule(&scheduleType, &chunkValue);

    printf("dyn-var: %d, nthreads-var:%d, thread-limit-var:%d,run-sched-var: %d, chunk: %d\n", omp_get_dynamic(),omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);

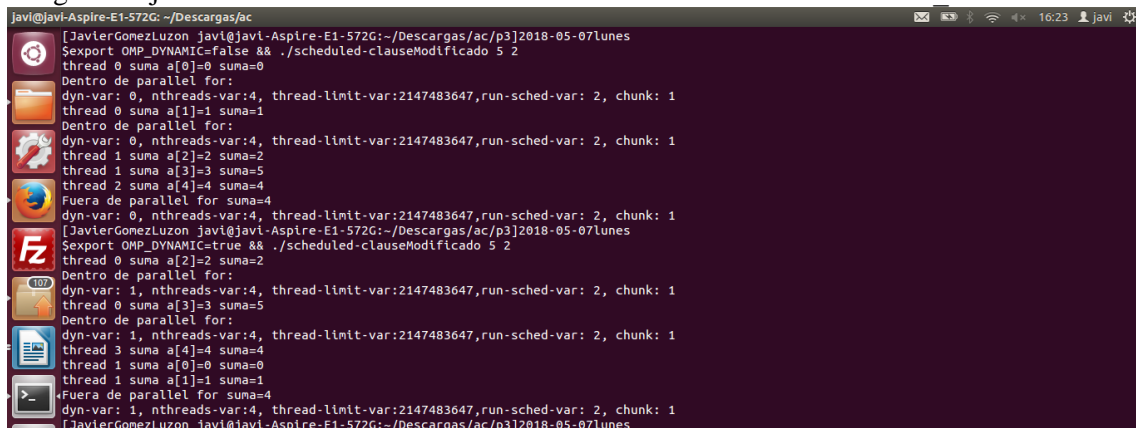
}

[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
$

```

CAPTURAS DE PANTALLA:

Imagen 5: Ejecucion de scheduled-clauseModificado.c cambiando OMP_DYNAMIC.

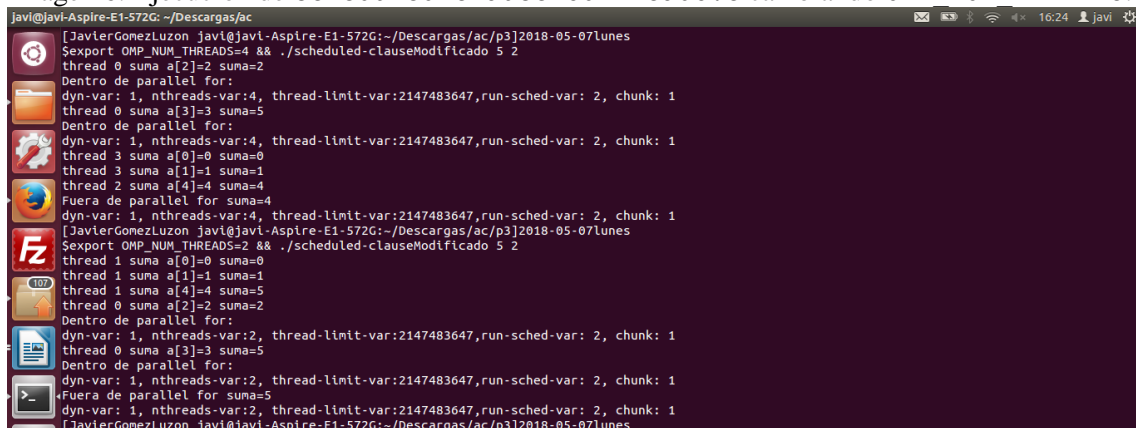


```

javi@javi-Aspire-E1-572G: ~/Descargas/ac
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
$export OMP_DYNAMIC=true && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
Dentro de parallel for:
dyn-var: 0, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 0, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
thread 2 suma a[4]=4 suma=4
Fuera de parallel for suma=4
dyn-var: 0, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
$export OMP_DYNAMIC=true && ./scheduled-clauseModificado 5 2
thread 0 suma a[2]=2 suma=2
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
thread 0 suma a[3]=3 suma=5
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
thread 3 suma a[4]=4 suma=4
thread 1 suma a[0]=0 suma=0
thread 1 suma a[1]=1 suma=1
Fuera de parallel for suma=4
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes

```

Imagen 6: Ejecucion de scheduled-clauseModificado.c cambiando OMP_NUM_THREADS.



```

javi@javi-Aspire-E1-572G: ~/Descargas/ac
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
$export OMP_NUM_THREADS=2 && ./scheduled-clauseModificado 5 2
thread 0 suma a[2]=2 suma=2
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
thread 0 suma a[3]=3 suma=5
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
thread 3 suma a[0]=0 suma=0
thread 3 suma a[1]=1 suma=1
thread 2 suma a[4]=4 suma=4
Fuera de parallel for suma=4
dyn-var: 1, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
$export OMP_NUM_THREADS=2 && ./scheduled-clauseModificado 5 2
thread 1 suma a[0]=0 suma=0
thread 1 suma a[1]=1 suma=1
thread 1 suma a[4]=4 suma=5
thread 0 suma a[2]=2 suma=2
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
thread 0 suma a[3]=3 suma=5
Dentro de parallel for:
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
Fuera de parallel for suma=5
dyn-var: 1, nthreads-var:2, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes

```

Imagen 7: Ejecucion de scheduled-clauseModificado.c cambiando OMP_SCHEDULE.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
Sexport OMP_SCHEDULE="static" && ./scheduled-clauseModificado 5 2
thread 0 suma a[2]=2 suma=2
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:4,run-sched-var: 1, chunk: 1
thread 0 suma a[3]=3 suma=5
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:4,run-sched-var: 1, chunk: 1
thread 1 suma a[0]=0 suma=0
thread 1 suma a[1]=1 suma=1
thread 2 suma a[4]=4 suma=4
Fuera de parallel for suma=4
dyn-var: 1, nthreads-var:4, thread-limit-var:4,run-sched-var: 1, chunk: 1
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
Sexport OMP_SCHEDULE="dynamic" && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
Dentro de parallel for:
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
dyn-var: 1, nthreads-var:4, thread-limit-var:4,run-sched-var: 2, chunk: 1
thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:4,run-sched-var: 2, chunk: 1
thread 2 suma a[4]=4 suma=4
Fuera de parallel for suma=4
dyn-var: 1, nthreads-var:4, thread-limit-var:4,run-sched-var: 2, chunk: 1
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes

```

Imagen 8: Ejecucion de `scheduled-clauseModificado.c` cambiando OMP_THREAD_LIMIT.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
Sexport OMP_THREAD_LIMIT=3 && ./scheduled-clauseModificado 5 2
thread 0 suma a[0]=0 suma=0
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:3,run-sched-var: 2, chunk: 1
thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:3,run-sched-var: 2, chunk: 1
thread 2 suma a[4]=4 suma=4
thread 1 suma a[2]=2 suma=2
thread 1 suma a[3]=3 suma=5
Fuera de parallel for suma=4
dyn-var: 1, nthreads-var:4, thread-limit-var:3,run-sched-var: 2, chunk: 1
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes
Sexport OMP_THREAD_LIMIT=4 && ./scheduled-clauseModificado 5 2
thread 1 suma a[0]=0 suma=0
thread 1 suma a[1]=1 suma=1
thread 3 suma a[2]=2 suma=2
thread 3 suma a[3]=3 suma=5
thread 0 suma a[4]=4 suma=4
Dentro de parallel for:
dyn-var: 1, nthreads-var:4, thread-limit-var:4,run-sched-var: 2, chunk: 1
Fuera de parallel for suma=4
dyn-var: 1, nthreads-var:4, thread-limit-var:4,run-sched-var: 2, chunk: 1
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-07lunes

```

RESPUESTA:

Como podemos ver, dentro de la región parallel y fuera obtenemos los mismos resultados.

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`

Imagen 9 y 10: Cat de scheduled-clauseModificado4.c.

```

javi@javi-Aspire-E1-572G: ~
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-13domingo
$ cat scheduled-clauseModificado4.c
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#define omp_get_num_threads() 1
#define omp_set_num_threads(int)
#define omp_in_parallel() 0
#endif

main(int argc, char **argv){
    int i, n=200, chunk, a[n], suma=0;
    omp_sched_t scheduleType;
    int chunkValue;

    if(argc<3){
        fprintf(stderr, "\nFalta iteraciones o chunk\n");
        exit(-1);
    }

    n=atoi(argv[1]);
    chunk=atoi(argv[2]);
    if(n>200) n=200;

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

        if(omp_get_thread_num()==0){
            printf("Dentro de parallel for: \n");

            omp_get_schedule(&scheduleType, &chunkValue);

            printf("dyn-var: %d, nthreads-var:%d, thread-limit-var:%d,run-sched-var: %d, chunk: %d\n", omp_get_dynamic(),omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);

            printf("omp_get_num_threads: %d, omp_get_num_procs: %d, omp_in_parallel(): %d\n",omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
        }

        printf("Fuera de parallel for suma=%d\n", suma);

        omp_get_schedule(&scheduleType, &chunkValue);

        printf("dyn-var: %d, nthreads-var:%d, thread-limit-var:%d,run-sched-var: %d, chunk: %d\n", omp_get_dynamic(),omp_get_max_threads(), omp_get_thread_limit(), scheduleType, chunkValue);

        printf("omp_get_num_threads: %d, omp_get_num_procs: %d, omp_in_parallel(): %d\n",omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());
    }
}

```

CAPTURAS DE PANTALLA:

Imagen 11: Ejecución scheduelld-clauseModificado4.c.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac/p3
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-12sábado
$ ./scheduled-clauseModificado4 5 2
thread 2 suma a[2]=2 suma=2
thread 2 suma a[3]=3 suma=5
thread 3 suma a[4]=4 suma=4
thread 0 suma a[0]=0 suma=0
Dentro de parallel for:
dyn-var: 0, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
omp_get_num_threads: 4, omp_get_num_procs: 4, omp_in_parallel(): 1
thread 0 suma a[1]=1 suma=1
Dentro de parallel for:
dyn-var: 0, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
omp_get_num_threads: 4, omp_get_num_procs: 4, omp_in_parallel(): 1
Fuera de parallel for suma=4
dyn-var: 0, nthreads-var:4, thread-limit-var:2147483647,run-sched-var: 2, chunk: 1
omp_get_num_threads: 1, omp_get_num_procs: 4, omp_in_parallel(): 0
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-12sábado

```

RESPUESTA:

Dentro y fuera cambian los valores de `omp_in_parallel()` y `omp_get_num_threads()`.

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`

Imagen 12 y 13: Cat scheduled-clauseModificado5.c.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac/p3
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-12sábado
$ cat scheduled-clauseModificado5.c
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#endif

main(int argc, char **argv){
    int i, n=200, chunk, a[n], suma=0, chunkValue;
    omp_sched_t scheduleType;

    if(argc<3){
        fprintf(stderr, "\nFalta iteraciones o chunk\n");
        exit(-1);
    }

    n=atoi(argv[1]);
    chunk=atoi(argv[2]);

    if(n>200) n=200;

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

    printf("Antes de hacer el cambio\n");
    omp_get_schedule(&scheduleType, &chunkValue);
    printf("dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d, chunk: %d\n ", omp_get_dynamic(), omp_get_max_threads(),
    omp_get_thread_limit(), scheduleType, chunkValue);

    printf("Cambiando valores a:\n\tdyn-var: 1\n\tnthreads-var: 3\n\trun-sched-var: 2\n\tchunk: 2\n");
    omp_set_dynamic(1);
    omp_set_num_threads(3);
    omp_set_schedule(2, 2);

    #pragma omp parallel 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("Fuera de parallel for suma=%d\n", suma);

    printf("Despues de hacer el cambio\n");
    omp_get_schedule(&scheduleType, &chunkValue);
    printf("dyn-var: %d, nthreads-var: %d, thread-limit-var: %d, run-sched-var: %d, chunk: %d\n ", omp_get_dynamic(), omp_get_max_threads(),
    omp_get_thread_limit(), scheduleType, chunkValue);
}
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-12sábado
$

```

CAPTURAS DE PANTALLA:

Imagen 14: Ejecución scheduled-clauseModificado5.c.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac/p3
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-12sábado
$ ./scheduled-clauseModificado5 2 3
Antes de hacer el cambio
dyn-var: 0, nthreads-var: 4, thread-limit-var: 2147483647, run-sched-var: 2, chunk: 1
Cambiando valores a:
dyn-var: 1
nthreads-var: 3
run-sched-var: 2
chunk: 2
thread 0 suma a[0]=0 suma=0
thread 0 suma a[1]=1 suma=1
Fuera de parallel for suma=1
Despues de hacer el cambio
dyn-var: 1, nthreads-var: 3, thread-limit-var: 2147483647, run-sched-var: 2, chunk: 2
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-12sábado
$

```

RESPUESTA:

Los cambios se realizan correctamente.

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

Imagen 15: Cat pmtv-secuencial.c.

```

javi@javi-Aspire-E1-572G: ~/Descargas/ac/p3
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-12sábado
Scat pmtv-secuencial.c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv){
    int i, j, N;
    int *v, *resultado;
    int **m;

    if(argc<2){
        fprintf(stderr, "Falta el tamaño\n");
        exit(-1);
    }

    N=atoi(argv[1]);

    v=(int*)malloc(N*sizeof(int));
    resultado=(int*)malloc(N*sizeof(int));
    m=(int**)malloc(N*sizeof(int*));

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

    for(i=0;i<N;i++){
        for(j=i;j<N;j++){ m[i][j]=2;
            v[i]=2;
            resultado[i]=0;
        }

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

    printf("\tResultado(0): %d\n\tResultado(N-1): %d\n", resultado[0], resultado[N-1]);

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

```

CAPTURAS DE PANTALLA:

Imagen 16: Ejecucion pmtv-secuencial.c.

```

javi@javi-Aspire-E1-572G: ~
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-13domingo
$ ./pmtv-secuencial 5
Resultado(0): 20
Resultado(N-1): 4
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-13domingo
$ ./pmtv-secuencial 6
Resultado(0): 24
Resultado(N-1): 4
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3]2018-05-13domingo

```

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:

Todas tienen unos tiempo de respuesta parecidos. Aunque el que es ligeramente mejor es

static ya que no tiene que hacer un reparto de tareas mas elaborado.

- Static no tiene y para dynamic y guided su valor es 0.
- Realizarán chunk * numero_de_filas

CAPTURA CÓDIGO FUENTE: pmtv-OpenMP.c

Imagen 17 y 18: Cat pmtv-OpenMP.c.

```
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
$ cat pmtv-OpenMP.c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(int argc, char **argv){
    int i, j, N;
    int *v, *r;
    int **m;
    double inicio, fin;

    if(argc<2){
        fprintf(stderr, "Falta tamaño\n");
        exit(-1);
    }

    N=atoi(argv[1]);

    r=(int*)malloc(N*sizeof(int));
    m=(int**)malloc(N*sizeof(int*));
    v=(int*)malloc(N*sizeof(int));

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

    for(i=0;i<N;i++){
        for(j=i;j<N;j++){
            m[i][j]=2;
            v[i]=2;
            r[i]=0;
        }
    }

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

    printf("Tiempo= %11.9f\tPrimera=%d\tUltima=%d\n", fin-inicio, r[0], r[N-1]);

    for(i=0;i<N;i++) free(m[i]);
    free(m);
    free(v);
    free(r);
}
```

DESCOMPOSICIÓN DE DOMINIO:

Cada hebra trabaja con una parte de los datos asignados (las filas de la matriz), y cada hebra conoce la fila con la que trabaja.

CAPTURAS DE PANTALLA:

Imagen 19: Ejecución pmtv-OpenMP.c.

```
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
$ ./pmtv-OpenMP 100
Tiempo= 0.007994653 Primera=400 Ultima=4
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
```

Imagen 20: Ejecución del script pmtv-OpenMP_PCaula en local.

```
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
$ ./pmtv-OpenMP_PCaula
schedule(static) chunk(defecto)
Tiempo= 0.059206379 Primera=61440 Ultima=4
schedule(static) chunk(1)
Tiempo= 0.060054683 Primera=61440 Ultima=4
schedule(static) chunk(64)
Tiempo= 0.059672128 Primera=61440 Ultima=4
schedule(dynamic) chunk(defecto)
Tiempo= 0.063617422 Primera=61440 Ultima=4
schedule(dynamic) chunk(1)
Tiempo= 0.063112092 Primera=61440 Ultima=4
schedule(dynamic) chunk(64)
Tiempo= 0.058461828 Primera=61440 Ultima=4
schedule(guided) chunk(defecto)
Tiempo= 0.080606061 Primera=61440 Ultima=4
schedule(guided) chunk(1)
Tiempo= 0.066946543 Primera=61440 Ultima=4
schedule(guided) chunk(64)
Tiempo= 0.068406362 Primera=61440 Ultima=4
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
```

Imagen 21: Ejecución del script pmtv-OpenMP_PCaula en atcgrid.

```
C1estudiante9@atcgrid:~
[JavierGomezLuzon C1estudiante9@atcgrid:~]2018-05-13domingo
Secho './pmtv-OpenMP_PCaula' | qsub -q ac
78503.atcgrid
[JavierGomezLuzon C1estudiante9@atcgrid:~]2018-05-13domingo
$ls
pmtv-OpenMP pmtv-OpenMP_PCaula STDIN.e78503 STDIN.o78503
[JavierGomezLuzon C1estudiante9@atcgrid:~]2018-05-13domingo
```

Imagen 22: Muestra de la salida y del error de la ejecución anterior.

```
javi@javi-Aspire-E1-572G:~
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
$cat STDIN.e*
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
$cat STDIN.o*
schedule(static) chunk(defecto)
Tiempo= 0.035475573 Primera=61440 Ultima=4
schedule(static) chunk(1)
Tiempo= 0.035963950 Primera=61440 Ultima=4
schedule(static) chunk(64)
Tiempo= 0.03523402 Primera=61440 Ultima=4
schedule(dynamic) chunk(defecto)
Tiempo= 0.038429376 Primera=61440 Ultima=4
schedule(dynamic) chunk(1)
Tiempo= 0.038120823 Primera=61440 Ultima=4
schedule(dynamic) chunk(64)
Tiempo= 0.03925342 Primera=61440 Ultima=4
schedule(guided) chunk(defecto)
Tiempo= 0.037341984 Primera=61440 Ultima=4
schedule(guided) chunk(1)
Tiempo= 0.041073302 Primera=61440 Ultima=4
schedule(guided) chunk(64)
Tiempo= 0.040120856 Primera=61440 Ultima=4
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
```

TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid

SCRIPT: pmtv-OpenMP_PCaula.sh

Imagen 23: Cat del script pmtv-OpenMP_PCaula.

```
javi@javi-Aspire-E1-572G:~
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
$cat pmtv-OpenMP_PCaula
#!/bin/bash

export OMP_SCHEDULE="static"
echo "schedule(static) chunk(defecto)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="static, 1"
echo "schedule(static) chunk(1)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="static, 64"
echo "schedule(static) chunk(64)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic"
echo "schedule(dynamic) chunk(defecto)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic, 1"
echo "schedule(dynamic) chunk(1)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="dynamic, 64"
echo "schedule(dynamic) chunk(64)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="guided"
echo "schedule(guided) chunk(defecto)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="guided, 1"
echo "schedule(guided) chunk(1)"
./pmtv-OpenMP 15360

export OMP_SCHEDULE="guided, 64"
echo "schedule(guided) chunk(64)"
./pmtv-OpenMP 15360
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
$
```

Tabla 3 . Tiempos de ejecución de la versión paralela del producto de una matriz triangular por un vector r **para vectores de tamaño N= 15360, 12 threads**

Tabla para la ejecución en local.

| Chunk | Static | Dynamic | Guided |
|-------------|-------------|-------------|-------------|
| por defecto | 0,059206379 | 0,063617422 | 0,080606061 |
| 1 | 0,060054683 | 0,063112092 | 0,066946543 |
| 64 | 0,059672128 | 0,058461828 | 0,068406362 |

Imagen 24: Gráfico de la ejecución del script en local.

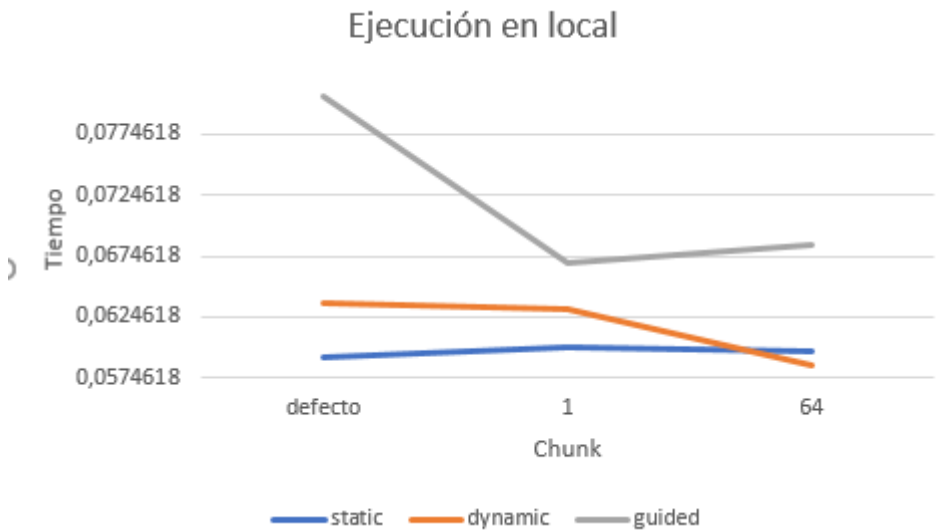
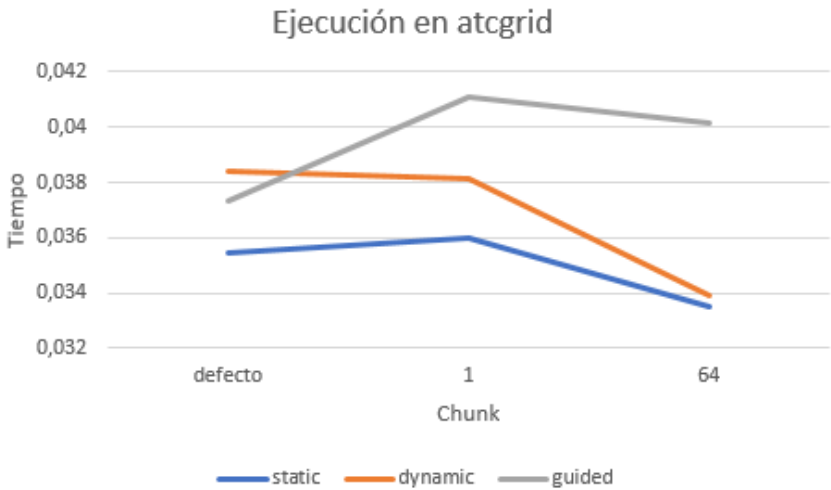


Tabla para la ejecución en atcgrid.

| Chunk | Static | Dynamic | Guided |
|-------------|-------------|-------------|-------------|
| por defecto | 0,035475573 | 0,038429376 | 0,037341984 |
| 1 | 0,035963950 | 0,038126823 | 0,041073302 |
| 64 | 0,033523402 | 0,033925342 | 0,040120856 |

Imagen 25: Gráfico de la ejecución del script en atcgrid.



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

Imagen 26 y 27: Cat pmm-secuencial.c.

```
[JavierGomezLuzon javi@javi-Aspire-E1-572G: ~/Descargas/ac/p3]2018-05-12sábado
$ cat pmm-secuencial.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, char **argv){
    int i, j, k, N;
    int **m1, **m2, **mResultado;
    struct timespec inicio, fin;
    double time;

    if(argc<2){
        fprintf(stderr, "falta el tamaño\n");
        exit(-1);
    }

    N=atoi(argv[1]);

    m1=(int**)malloc(N*sizeof(int*));
    m2=(int**)malloc(N*sizeof(int*));
    mResultado=(int**)malloc(N*sizeof(int*));

    for (i=0;i<N;i++){
        m1[i]=(int*)malloc(N*sizeof(int));
        m2[i]=(int*)malloc(N*sizeof(int));
        mResultado[i]=(int*)malloc(N*sizeof(int));
    }

    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            m1[i][j]=2;
            m2[i][j]=2;
            mResultado[i][j]=0;
        }
    }

    clock_gettime(CLOCK_REALTIME,&inicio);
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            for(k=0;k<N;k++){ mResultado[i][j]+=m1[i][k]*m2[k][j];
            }
        }
    }
    clock_gettime(CLOCK_REALTIME,&fin);

    time=(double)(fin.tv_sec-inicio.tv_sec)+(double)((fin.tv_nsec-inicio.tv_nsec)/(1.e+9));

    printf("Tiempo=%11.9f\n\t(0,0)=%d\n\t(N-1, N-1)=%d\n", time,mResultado[0][0], mResultado[N-1][N-1]);

    for (i=0; i<N; i++){
        free(m1[i]);
        free(m2[i]);
        free(mResultado[i]);
    }
    free(m1);
    free(m2);
    free(mResultado);
}
```

CAPTURAS DE PANTALLA:

Imagen 28: Ejecucion pmm-secuencial.c.

```
[JavierGomezLuzon javi@javi-Aspire-E1-572G: ~/Descargas/ac/p3]2018-05-12sábado
$ ./pmm-secuencial 5
Tiempo=0.000001043
(0,0)=20
(N-1, N-1)=20
[JavierGomezLuzon javi@javi-Aspire-E1-572G: ~/Descargas/ac/p3]2018-05-12sábado
$ ./pmm-secuencial 6
Tiempo=0.000002175
(0,0)=24
(N-1, N-1)=24
[JavierGomezLuzon javi@javi-Aspire-E1-572G: ~/Descargas/ac/p3]2018-05-12sábado
```

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).

CAPTURA CÓDIGO FUENTE: pmm-OpenMP.c

Imagen 29 y 30: Cat de pmm-OpenMP . c .

```

javi@javi-Aspire-E1-572G:~$ cat pmm-OpenMP.c
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
$cat pmm-OpenMP.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, char **argv){
    int i, j, k, N;
    int **m1, **m2, **mResultado;
    struct timespec inicio, fin;
    double time;

    if(argc<2){
        fprintf(stderr, "falta el tamaño\n");
        exit(-1);
    }

    N=atoi(argv[1]);

    m1=(int**)malloc(N*sizeof(int*));
    m2=(int**)malloc(N*sizeof(int*));
    mResultado=(int**)malloc(N*sizeof(int*));

    for (i=0;i<N;i++){
        m1[i]=(int*)malloc(N*sizeof(int));
        m2[i]=(int*)malloc(N*sizeof(int));
        mResultado[i]=(int*)malloc(N*sizeof(int));
    }

    #pragma omp parallel for private(j)
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            m1[i][j]=2;
            m2[i][j]=2;
            mResultado[i][j]=0;
        }
    }

    clock_gettime(CLOCK_REALTIME,&inicio);
    #pragma omp parallel for private(j, k)
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            for(k=0;k<N;k++){ mResultado[i][j]=m1[i][k]*m2[k][j];
            }
        }
    }
    clock_gettime(CLOCK_REALTIME,&fin);
    time=(double)(fin.tv_sec-inicio.tv_sec)+(double)((fin.tv_nsec-inicio.tv_nsec)/(1.e+9));
    printf("Tiempo=%11.9f\n\t(0,0)=%d\n\t(N-1, N-1)=%d\n", time,mResultado[0][0], mResultado[N-1][N-1]);

    for (i=0; i<N; i++){
        free(m1[i]);
        free(m2[i]);
        free(mResultado[i]);
    }
    free(m1);
    free(m2);
    free(mResultado);
}

```

CAPTURAS DE PANTALLA:

Imagen 31: Ejecución de pmm-OpenMP . c .

```

javi@javi-Aspire-E1-572G:~$ ./pmm-OpenMP 100
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
$./pmm-OpenMP 100
Tiempo=0.000775215
(0,0)=400
(N-1, N-1)=400
[javierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo

```

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:

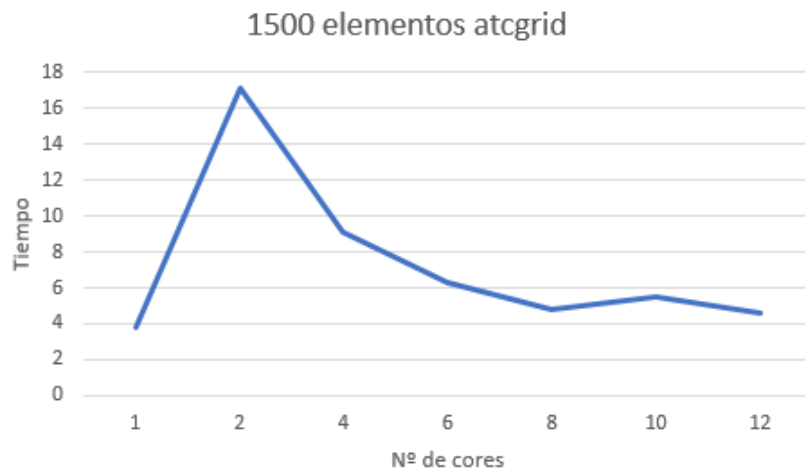
Tabla de valores para la ejecución en atcgrid.

| Cores | 100 elementos | 1500 elementos |
|-------|---------------|----------------|
| 1 | 0.0046046 | 3.790517125 |
| 2 | 0.001306141 | 17.10128142 |
| 4 | 0.00742971 | 9.113018113 |
| 6 | 0.000356639 | 6.301190018 |
| 8 | 0.000354023 | 4.835182903 |
| 10 | 0.000251833 | 5.52525334 |
| 12 | 0.000380848 | 4.574818653 |

Imagen 32: Gráfico de ejecución del script pmm-OpenMP_atcgrid en atcgrid para 100 elementos.

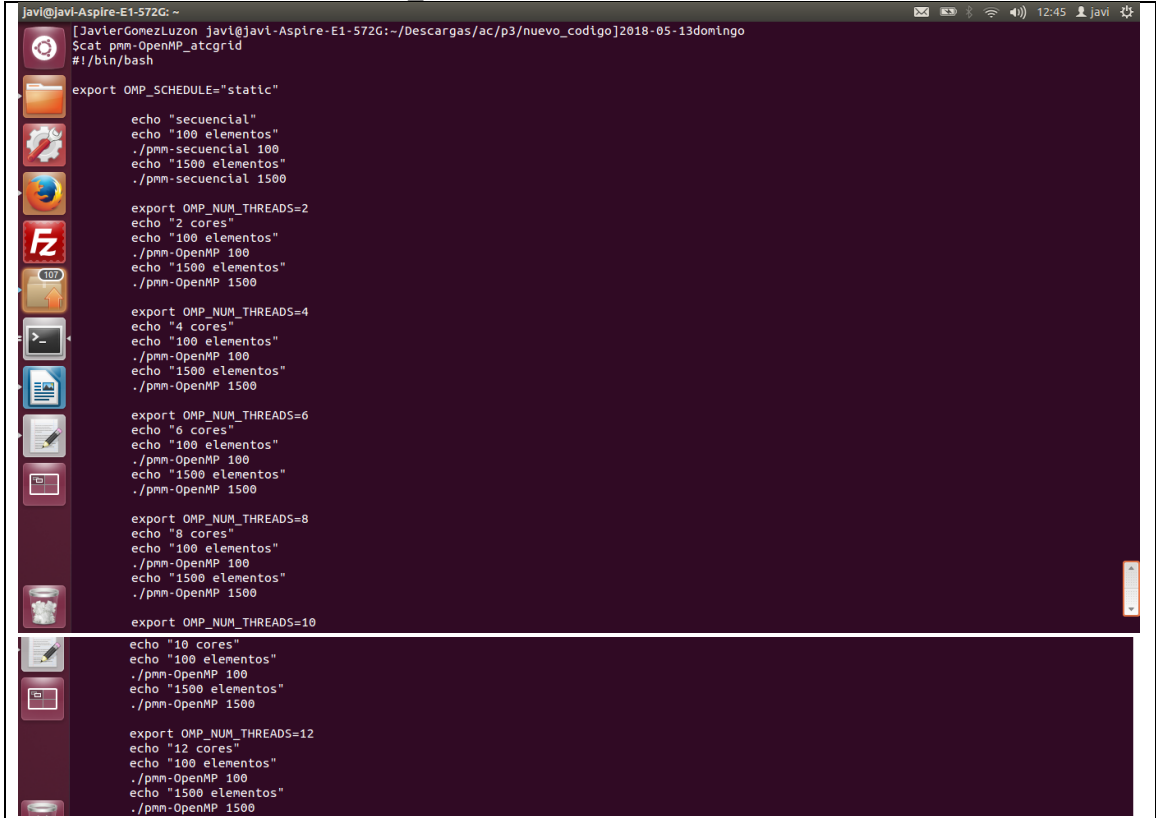


Imagen 33: Gráfico de ejecución del script pmm-OpenMP_atcgrid en atcgrid para 1500 elementos.



SCRIPT: pmm-OpenMP_atcgrid.sh

Imagen 34 y 35: Cat pmm-OpenMP_atcgrid.



ESTUDIO DE ESCALABILIDAD EN PCLOCAL:

Tabla de valores para la ejecución en local.

| Cores | 100 elementos | 1500 elementos |
|-------|---------------|----------------|
| 1 | 0.00078313 | 32.93705332 |
| 2 | 0.000696979 | 17.00792599 |

Imagen 36: Gráfico de ejecución del script pmm-OpenMP_pclocal en local para 100 elementos.

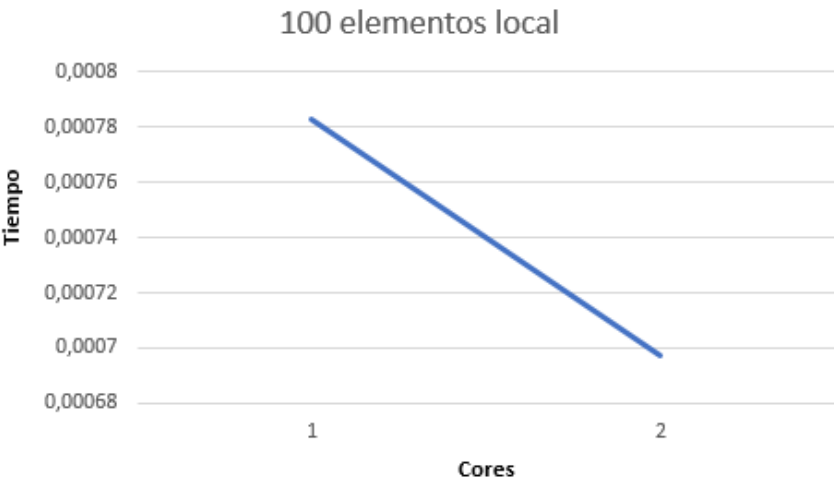
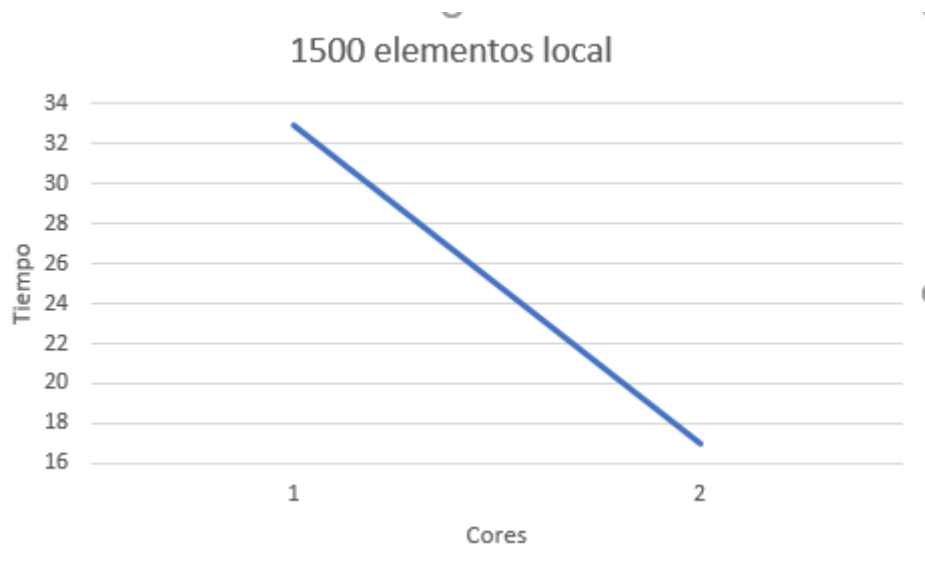


Imagen 37: Gráfico de ejecución del script `pmm-OpenMP_pcllocal` en local para 1500 elementos.



SCRIPT: `pmm-OpenMP_pcllocal.sh`

Imagen 38: Cat `pmm-OpenMP_pcllocal`.

```
javi@javi-Aspire-E1-572G: ~  
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo  
$cat pmm-OpenMP_pcllocal  
#!/bin/bash  
  
export OMP_SCHEDULE="static"  
  
echo "secuencial"  
echo "100 elementos"  
./pmm-secuencial 100  
echo "1500 elementos"  
./pmm-secuencial 1500  
  
export OMP_NUM_THREADS=2  
echo "2 cores"  
echo "100 elementos"  
./pmm-OpenMP 100  
echo "1500 elementos"  
./pmm-OpenMP 1500  
[JavierGomezLuzon javi@javi-Aspire-E1-572G:~/Descargas/ac/p3/nuevo_codigo]2018-05-13domingo
```