

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): Jose Luis Martínez Ortiz

Grupo de prácticas: C3

Fecha de entrega: 17 - 05 - 2016

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.

CÓDIGO FUENTE: `if-clauseModificado.c`

```
/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 o 9 */
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ */
/* INTERLINEADO SENCILLO */

#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 < 3) {
        fprintf(stderr, "[ERROR]-Falta iteraciones y/o num_threads\n");
        exit(-1);
    }

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

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

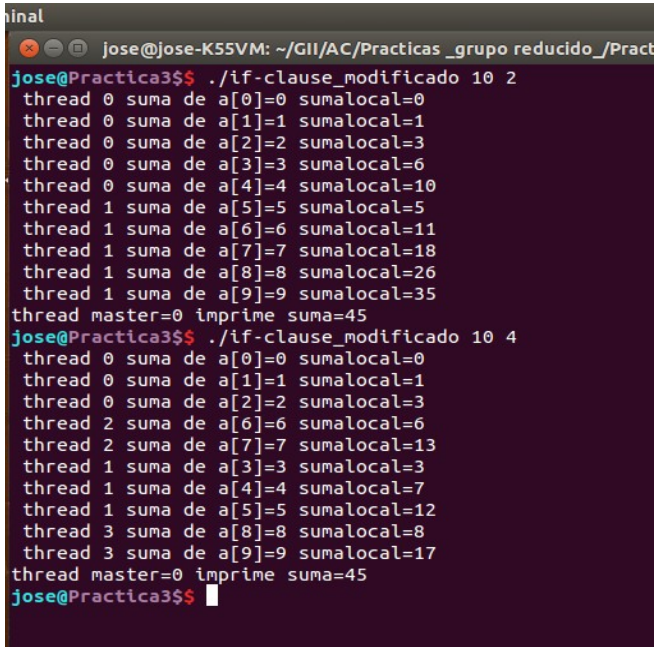
    #pragma omp parallel num_threads(x) if(n>4) default(none) \
        private(sumalocal,tid) shared(a,suma,n)
    {
        sumalocal=0;
        tid=omp_get_thread_num();

        #pragma omp for private(i) schedule(static) nowait
        for (i=0; i<n; i++)
        {
            sumalocal += a[i];
            printf(" thread %d suma de a[%d]=%d\n",
                tid,i,a[i],sumalocal);
        }
    }

    suma+=sumalocal;
    printf(" suma total = %d\n", suma);
}
```

```
        #pragma omp atomic
        suma += sumalocal;
    #pragma omp barrier
    #pragma omp master
        printf("thread master=%d imprime suma=
%d\n",tid,suma);
    }
}
```

CAPTURAS DE PANTALLA:



RESPUESTA: En la captura se puede observar que dependiendo del 2º parametro, en primer lugar con 10 iteraciones y con 2 num_threads observamos que solo esta el thread 0 y 1. En la segunda ejecución con 10 iteraciones y con 4 num_threads vemos que realiza el mismo cálculo pero lo reparte entre 4 threads.

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 *scheduleleg-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			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	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	0	0	0
15	1	1	1	0	0	0	0	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			schedule- clausd.c			schedule- clauseg.c		
	1	2	4	1	2	4	1	2	4
0	0	0	0	3	2	2	1	1	3
1	1	0	0	1	2	2	1	1	3
2	2	1	0	2	0	2	1	1	3
3	3	1	0	0	0	2	1	1	3
4	0	2	1	0	1	1	2	2	1
5	1	2	1	0	1	1	2	2	1
6	2	3	1	0	3	1	2	2	1
7	3	3	1	0	3	1	0	3	1
8	0	0	2	0	0	3	0	3	0
9	1	0	2	0	0	3	0	3	0
10	2	1	2	0	0	3	3	0	0

11	3	1	2	0	0	3	3	0	0
12	0	2	3	0	0	0	1	0	2
13	1	2	3	0	0	0	1	0	2
14	2	3	3	0	0	0	1	0	2
15	3	3	3	0	0	0	1	0	2

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

RESPUESTA: Con el `Schedule static` se reparte las iteraciones entre los threads que están fijados, asignándoles a cada uno el chunk que se le indica. Con el `Schedule dynamic` vemos que prácticamente todo lo ejecuta el thread 0, con este esquema se hace el reparto en tiempo de ejecución y el thread más rápido ejecuta más iteraciones, por ello al ser una operación sencilla es el thread 0 quien lo hace prácticamente todo. Con el `Schedule guided` verificamos que divide el nº de iteraciones entre el nº de threads y repite con el resto.

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?

CÓDIGO FUENTE: `scheduled-clauseModificado.c`

```
/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 o 9 .*/
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/
/* INTERLINEADO SENCILLO */

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

int main(int argc, char ** argv)
{
    int i, n=16, 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]);
```

```

omp_sched_t kind;
int modifier;
omp_get_schedule(&kind, & modifier);

printf("Fuera \ndyn-var=%d  nthreads_var=%d  thread-limit-var=%d
run-sched-var=(%d,%d)\n",

omp_get_dynamic(),omp_get_max_threads(),omp_get_thread_limit(),kind,modifier);

for (i=0; i<n; i++) a[i] = i;
#pragma omp parallel num_threads(4)
{
    #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("Dentro\n");
    omp_get_schedule(&kind,
    & modifier);

    printf("dyn-var=%d
nthreads_var=%d  thread-limit-var=%d  run-sched-var=(%d,%d)\n",

omp_get_dynamic(),omp_get_max_threads(),omp_get_thread_limit(),kind,modifier);
}

    }

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

```

CAPTURAS DE PANTALLA:

```

thread 3 suma a[14]=14 suma=39
thread 3 suma a[15]=15 suma=54
Dentro
dyn-var=0  nthreads_var=8  thread-limit-var=2147483647  run-sched-var=(2,1)
Fuera de 'parallel for' suma=54
jose@Practica3$ export OMP_DYNAMIC=TRUE
jose@Practica3$ ./scheduled-clause_modificado 16 4
Fuera
dyn-var=1  nthreads_var=8  thread-limit-var=2147483647  run-sched-var=(2,1)
thread 0 suma a[12]=12 suma=12
thread 0 suma a[13]=13 suma=25
thread 0 suma a[14]=14 suma=39
thread 0 suma a[15]=15 suma=54
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 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[8]=8 suma=8
thread 3 suma a[9]=9 suma=17
thread 3 suma a[10]=10 suma=27
thread 3 suma a[11]=11 suma=38
Dentro
dyn-var=1  nthreads_var=8  thread-limit-var=2147483647  run-sched-var=(2,1)
Fuera de 'parallel for' suma=54
jose@Practica3$

```

```

jose@Jose-K55VM: ~/GII/AC/Practicas_grupo reducido_/Practica3
jose@Practica3$ ./scheduled-clause_modificado 16 4
Fuera
dyn-var=0  nthreads_var=8  thread-limit-var=2147483647  run-sched-var=(2,1)
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
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 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
Dentro
dyn-var=0  nthreads_var=8  thread-limit-var=2147483647  run-sched-var=(2,1)
Fuera de 'parallel for' suma=54
jose@Practica3$

```

```

final
jose@jose-K55VM: ~/Gil/AC/Practicas_grupo reducido_/Practica3
jose@Practica3$ export OMP_THREAD_LIMIT=8
jose@Practica3$ ./scheduled-clause_modificado 16 4
Fuera
dyn-var=1 nthreads_var=8 thread-limit-var=8 run-sched-var=(2,1)
thread 2 suma a[8]=8 suma=8
thread 2 suma a[9]=9 suma=17
thread 2 suma a[10]=10 suma=27
thread 2 suma a[11]=11 suma=38
thread 0 suma a[12]=12 suma=12
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 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
thread 0 suma a[13]=13 suma=25
thread 0 suma a[14]=14 suma=39
thread 0 suma a[15]=15 suma=54
Dentro
dyn-var=1 nthreads_var=8 thread-limit-var=8 run-sched-var=(2,1)
Fuera de 'parallel for' suma=54
jose@Practica3$

```

RESPUESTA: No, ya que las variables de entorno son fijas y no se alteran en la ejecución del programa.

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.

CÓDIGO FUENTE: scheduled-clauseModificado4.c

```

/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 o 9 */
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ */
/* INTERLINEADO SENCILLO */

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

int main(int argc, char ** argv)
{
    int i, n=16, 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]);

    omp_sched_t kind;
    int modifier;
    omp_get_schedule(&kind, & modifier);

    printf("Fuera.\n");
    printf("omp_get_num_threads()=%d  omp_get_num_procs()=%d\n",
    omp_in_parallel()=%d \n",

    omp_get_num_threads(), omp_get_num_procs(), omp_in_parallel());

```

```

        for (i=0; i<n; i++) a[i] = i;
        #pragma omp parallel num_threads(4)
        {
            #pragma omp for firstprivate(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("Dentro\n");

                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);
        }
}

```

CAPTURAS DE PANTALLA:

The left screenshot shows the terminal output of the first run of the program. It displays the output of the first thread (thread 0) and the output of the first thread (thread 0) inside the parallel region. The output shows the thread number, the index i, the value of a[i], and the current value of suma. The output is as follows:

```

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[4]=4 suma=10
thread 0 suma a[5]=5 suma=15
thread 0 suma a[6]=6 suma=21
thread 0 suma a[7]=7 suma=28
thread 0 suma a[8]=8 suma=36
thread 0 suma a[9]=9 suma=45
thread 0 suma a[10]=10 suma=55
thread 0 suma a[11]=11 suma=66
thread 0 suma a[12]=12 suma=78
thread 0 suma a[13]=13 suma=91
thread 0 suma a[14]=14 suma=105
thread 0 suma a[15]=15 suma=120
thread 0 suma a[16]=16 suma=136
thread 0 suma a[17]=17 suma=153
thread 0 suma a[18]=18 suma=171
thread 0 suma a[19]=19 suma=190
thread 0 suma a[20]=20 suma=210
thread 0 suma a[21]=21 suma=231
thread 0 suma a[22]=22 suma=253
thread 0 suma a[23]=23 suma=276
thread 0 suma a[24]=24 suma=300
thread 0 suma a[25]=25 suma=325
thread 0 suma a[26]=26 suma=351
thread 0 suma a[27]=27 suma=378
thread 0 suma a[28]=28 suma=406
thread 0 suma a[29]=29 suma=435
thread 0 suma a[30]=30 suma=465
thread 0 suma a[31]=31 suma=496
thread 0 suma a[32]=32 suma=528
thread 0 suma a[33]=33 suma=561
thread 0 suma a[34]=34 suma=595
thread 0 suma a[35]=35 suma=630
thread 0 suma a[36]=36 suma=666
thread 0 suma a[37]=37 suma=703
thread 0 suma a[38]=38 suma=741
thread 0 suma a[39]=39 suma=780
thread 0 suma a[40]=40 suma=820
thread 0 suma a[41]=41 suma=861
thread 0 suma a[42]=42 suma=903
thread 0 suma a[43]=43 suma=946
thread 0 suma a[44]=44 suma=990
thread 0 suma a[45]=45 suma=1035
thread 0 suma a[46]=46 suma=1081
thread 0 suma a[47]=47 suma=1128
thread 0 suma a[48]=48 suma=1176
thread 0 suma a[49]=49 suma=1225
thread 0 suma a[50]=50 suma=1275
thread 0 suma a[51]=51 suma=1326
thread 0 suma a[52]=52 suma=1378
thread 0 suma a[53]=53 suma=1431
thread 0 suma a[54]=54 suma=1485
thread 0 suma a[55]=55 suma=1540
thread 0 suma a[56]=56 suma=1596
thread 0 suma a[57]=57 suma=1653
thread 0 suma a[58]=58 suma=1711
thread 0 suma a[59]=59 suma=1770
thread 0 suma a[60]=60 suma=1830
thread 0 suma a[61]=61 suma=1891
thread 0 suma a[62]=62 suma=1953
thread 0 suma a[63]=63 suma=2016
thread 0 suma a[64]=64 suma=2080
thread 0 suma a[65]=65 suma=2145
thread 0 suma a[66]=66 suma=2211
thread 0 suma a[67]=67 suma=2278
thread 0 suma a[68]=68 suma=2346
thread 0 suma a[69]=69 suma=2415
thread 0 suma a[70]=70 suma=2485
thread 0 suma a[71]=71 suma=2556
thread 0 suma a[72]=72 suma=2628
thread 0 suma a[73]=73 suma=2701
thread 0 suma a[74]=74 suma=2775
thread 0 suma a[75]=75 suma=2850
thread 0 suma a[76]=76 suma=2926
thread 0 suma a[77]=77 suma=3003
thread 0 suma a[78]=78 suma=3081
thread 0 suma a[79]=79 suma=3160
thread 0 suma a[80]=80 suma=3240
thread 0 suma a[81]=81 suma=3321
thread 0 suma a[82]=82 suma=3403
thread 0 suma a[83]=83 suma=3486
thread 0 suma a[84]=84 suma=3570
thread 0 suma a[85]=85 suma=3655
thread 0 suma a[86]=86 suma=3741
thread 0 suma a[87]=87 suma=3828
thread 0 suma a[88]=88 suma=3916
thread 0 suma a[89]=89 suma=4005
thread 0 suma a[90]=90 suma=4095
thread 0 suma a[91]=91 suma=4186
thread 0 suma a[92]=92 suma=4278
thread 0 suma a[93]=93 suma=4371
thread 0 suma a[94]=94 suma=4465
thread 0 suma a[95]=95 suma=4560
thread 0 suma a[96]=96 suma=4656
thread 0 suma a[97]=97 suma=4753
thread 0 suma a[98]=98 suma=4851
thread 0 suma a[99]=99 suma=4950
thread 0 suma a[100]=100 suma=5050
thread 0 suma a[101]=101 suma=5151
thread 0 suma a[102]=102 suma=5253
thread 0 suma a[103]=103 suma=5356
thread 0 suma a[104]=104 suma=5460
thread 0 suma a[105]=105 suma=5565
thread 0 suma a[106]=106 suma=5671
thread 0 suma a[107]=107 suma=5778
thread 0 suma a[108]=108 suma=5886
thread 0 suma a[109]=109 suma=5995
thread 0 suma a[110]=110 suma=6105
thread 0 suma a[111]=111 suma=6216
thread 0 suma a[112]=112 suma=6328
thread 0 suma a[113]=113 suma=6441
thread 0 suma a[114]=114 suma=6555
thread 0 suma a[115]=115 suma=6670
thread 0 suma a[116]=116 suma=6786
thread 0 suma a[117]=117 suma=6903
thread 0 suma a[118]=118 suma=7021
thread 0 suma a[119]=119 suma=7140
thread 0 suma a[120]=120 suma=7260
thread 0 suma a[121]=121 suma=7381
thread 0 suma a[122]=122 suma=7503
thread 0 suma a[123]=123 suma=7626
thread 0 suma a[124]=124 suma=7750
thread 0 suma a[125]=125 suma=7875
thread 0 suma a[126]=126 suma=8001
thread 0 suma a[127]=127 suma=8128
thread 0 suma a[128]=128 suma=8256
thread 0 suma a[129]=129 suma=8385
thread 0 suma a[130]=130 suma=8515
thread 0 suma a[131]=131 suma=8646
thread 0 suma a[132]=132 suma=8778
thread 0 suma a[133]=133 suma=8911
thread 0 suma a[134]=134 suma=9045
thread 0 suma a[135]=135 suma=9180
thread 0 suma a[136]=136 suma=9316
thread 0 suma a[137]=137 suma=9453
thread 0 suma a[138]=138 suma=9591
thread 0 suma a[139]=139 suma=9730
thread 0 suma a[140]=140 suma=9870
thread 0 suma a[141]=141 suma=10011
thread 0 suma a[142]=142 suma=10153
thread 0 suma a[143]=143 suma=10296
thread 0 suma a[144]=144 suma=10440
thread 0 suma a[145]=145 suma=10585
thread 0 suma a[146]=146 suma=10731
thread 0 suma a[147]=147 suma=10878
thread 0 suma a[148]=148 suma=11026
thread 0 suma a[149]=149 suma=11175
thread 0 suma a[150]=150 suma=11325
thread 0 suma a[151]=151 suma=11476
thread 0 suma a[152]=152 suma=11628
thread 0 suma a[153]=153 suma=11781
thread 0 suma a[154]=154 suma=11935
thread 0 suma a[155]=155 suma=12090
thread 0 suma a[156]=156 suma=12246
thread 0 suma a[157]=157 suma=12403
thread 0 suma a[158]=158 suma=12561
thread 0 suma a[159]=159 suma=12720
thread 0 suma a[160]=160 suma=12880
thread 0 suma a[161]=161 suma=13041
thread 0 suma a[162]=162 suma=13203
thread 0 suma a[163]=163 suma=13366
thread 0 suma a[164]=164 suma=13530
thread 0 suma a[165]=165 suma=13695
thread 0 suma a[166]=166 suma=13861
thread 0 suma a[167]=167 suma=14028
thread 0 suma a[168]=168 suma=14196
thread 0 suma a[169]=169 suma=14365
thread 0 suma a[170]=170 suma=14535
thread 0 suma a[171]=171 suma=14706
thread 0 suma a[172]=172 suma=14878
thread 0 suma a[173]=173 suma=15051
thread 0 suma a[174]=174 suma=15225
thread 0 suma a[175]=175 suma=15400
thread 0 suma a[176]=176 suma=15576
thread 0 suma a[177]=177 suma=15753
thread 0 suma a[178]=178 suma=15931
thread 0 suma a[179]=179 suma=16110
thread 0 suma a[180]=180 suma=16290
thread 0 suma a[181]=181 suma=16471
thread 0 suma a[182]=182 suma=16653
thread 0 suma a[183]=183 suma=16836
thread 0 suma a[184]=184 suma=17020
thread 0 suma a[185]=185 suma=17205
thread 0 suma a[186]=186 suma=17391
thread 0 suma a[187]=187 suma=17578
thread 0 suma a[188]=188 suma=17766
thread 0 suma a[189]=189 suma=17955
thread 0 suma a[190]=190 suma=18145
thread 0 suma a[191]=191 suma=18336
thread 0 suma a[192]=192 suma=18528
thread 0 suma a[193]=193 suma=18721
thread 0 suma a[194]=194 suma=18915
thread 0 suma a[195]=195 suma=19110
thread 0 suma a[196]=196 suma=19306
thread 0 suma a[197]=197 suma=19503
thread 0 suma a[198]=198 suma=19701
thread 0 suma a[199]=199 suma=19900
thread 0 suma a[200]=200 suma=20100
thread 0 suma a[201]=201 suma=20301
thread 0 suma a[202]=202 suma=20503
thread 0 suma a[203]=203 suma=20706
thread 0 suma a[204]=204 suma=20910
thread 0 suma a[205]=205 suma=21115
thread 0 suma a[206]=206 suma=21321
thread 0 suma a[207]=207 suma=21528
thread 0 suma a[208]=208 suma=21735
thread 0 suma a[209]=209 suma=21943
thread 0 suma a[210]=210 suma=22152
thread 0 suma a[211]=211 suma=22362
thread 0 suma a[212]=212 suma=22573
thread 0 suma a[213]=213 suma=22785
thread 0 suma a[214]=214 suma=22997
thread 0 suma a[215]=215 suma=23210
thread 0 suma a[216]=216 suma=23424
thread 0 suma a[217]=217 suma=23639
thread 0 suma a[218]=218 suma=23855
thread 0 suma a[219]=219 suma=24071
thread 0 suma a[220]=220 suma=24288
thread 0 suma a[221]=221 suma=24505
thread 0 suma a[222]=222 suma=24723
thread 0 suma a[223]=223 suma=24942
thread 0 suma a[224]=224 suma=25161
thread 0 suma a[225]=225 suma=25381
thread 0 suma a[226]=226 suma=25602
thread 0 suma a[227]=227 suma=25823
thread 0 suma a[228]=228 suma=26045
thread 0 suma a[229]=229 suma=26267
thread 0 suma a[230]=230 suma=26490
thread 0 suma a[231]=231 suma=26713
thread 0 suma a[232]=232 suma=26937
thread 0 suma a[233]=233 suma=27161
thread 0 suma a[234]=234 suma=27386
thread 0 suma a[235]=235 suma=27611
thread 0 suma a[236]=236 suma=27837
thread 0 suma a[237]=237 suma=28063
thread 0 suma a[238]=238 suma=28290
thread 0 suma a[239]=239 suma=28517
thread 0 suma a[240]=240 suma=28745
thread 0 suma a[241]=241 suma=28973
thread 0 suma a[242]=242 suma=29202
thread 0 suma a[243]=243 suma=29431
thread 0 suma a[244]=244 suma=29661
thread 0 suma a[245]=245 suma=29891
thread 0 suma a[246]=246 suma=30122
thread 0 suma a[247]=247 suma=30353
thread 0 suma a[248]=248 suma=30584
thread 0 suma a[249]=249 suma=30815
thread 0 suma a[250]=250 suma=31047
thread 0 suma a[251]=251 suma=31279
thread 0 suma a[252]=252 suma=31511
thread 0 suma a[253]=253 suma=31744
thread 0 suma a[254]=254 suma=31977
thread 0 suma a[255]=255 suma=32211
thread 0 suma a[256]=256 suma=32445
thread 0 suma a[257]=257 suma=32680
thread 0 suma a[258]=258 suma=32915
thread 0 suma a[259]=259 suma=33150
thread 0 suma a[260]=260 suma=33385
thread 0 suma a[261]=261 suma=33621
thread 0 suma a[262]=262 suma=33857
thread 0 suma a[263]=263 suma=34093
thread 0 suma a[264]=264 suma=34330
thread 0 suma a[265]=265 suma=34567
thread 0 suma a[266]=266 suma=34804
thread 0 suma a[267]=267 suma=35041
thread 0 suma a[268]=268 suma=35279
thread 0 suma a[269]=269 suma=35517
thread 0 suma a[270]=270 suma=35755
thread 0 suma a[271]=271 suma=35993
thread 0 suma a[272]=272 suma=36231
thread 0 suma a[273]=273 suma=36470
thread 0 suma a[274]=274 suma=36709
thread 0 suma a[275]=275 suma=36948
thread 0 suma a[276]=276 suma=37187
thread 0 suma a[277]=277 suma=37426
thread 0 suma a[278]=278 suma=37665
thread 0 suma a[279]=279 suma=37905
thread 0 suma a[280]=280 suma=38145
thread 0 suma a[281]=281 suma=38385
thread 0 suma a[282]=282 suma=38625
thread 0 suma a[283]=283 suma=38865
thread 0 suma a[284]=284 suma=39105
thread 0 suma a[285]=285 suma=39345
thread 0 suma a[286]=286 suma=39585
thread 0 suma a[287]=287 suma=39825
thread 0 suma a[288]=288 suma=40065
thread 0 suma a[289]=289 suma=40305
thread 0 suma a[290]=290 suma=40545
thread 0 suma a[291]=291 suma=40785
thread 0 suma a[292]=292 suma=41025
thread 0 suma a[293]=293 suma=41265
thread 0 suma a[294]=294 suma=41505
thread 0 suma a[295]=295 suma=41745
thread 0 suma a[296]=296 suma=41985
thread 0 suma a[297]=297 suma=42225
thread 0 suma a[298]=298 suma=42465
thread 0 suma a[299]=299 suma=42705
thread 0 suma a[300]=300 suma=42945
thread 0 suma a[301]=301 suma=43185
thread 0 suma a[302]=302 suma=43425
thread 0 suma a[303]=303 suma=43665
thread 0 suma a[304]=304 suma=43905
thread 0 suma a[305]=305 suma=44145
thread 0 suma a[306]=306 suma=44385
thread 0 suma a[307]=307 suma=44625
thread 0 suma a[308]=308 suma=44865
thread 0 suma a[309]=309 suma=45105
thread 0 suma a[310]=310 suma=45345
thread 0 suma a[311]=311 suma=45585
thread 0 suma a[312]=312 suma=45825
thread 0 suma a[313]=313 suma=46065
thread 0 suma a[314]=314 suma=46305
thread 0 suma a[315]=315 suma=46545
thread 0 suma a[316]=316 suma=46785
thread 0 suma a[317]=317 suma=47025
thread 0 suma a[318]=318 suma=47265
thread 0 suma a[319]=319 suma=47505
thread 0 suma a[320]=320 suma=47745
thread 0 suma a[321]=321 suma=47985
thread 0 suma a[322]=322 suma=48225
thread 0 suma a[323]=323 suma=48465
thread 0 suma a[324]=324 suma=48705
thread 0 suma a[325]=325 suma=48945
thread 0 suma a[326]=326 suma=49185
thread 0 suma a[327]=327 suma=49425
thread 0 suma a[328]=328 suma=49665
thread 0 suma a[329]=329 suma=49905
thread 0 suma a[330]=330 suma=50145
thread 0 suma a[331]=331 suma=50385
thread 0 suma a[332]=332 suma=50625
thread 0 suma a[333]=333 suma=50865
thread 0 suma a[334]=334 suma=51105
thread 0 suma a[335]=335 suma=51345
thread 0 suma a[336]=336 suma=51585
thread 0 suma a[337]=337 suma=51825
thread 0 suma a[338]=338 suma=52065
thread 0 suma a[339]=339 suma=52305
thread 0 suma a[340]=340 suma=52545
thread 0 suma a[341]=341 suma=52785
thread 0 suma a[342]=342 suma=53025
thread 0 suma a[343]=343 suma=53265
thread 0 suma a[344]=344 suma=53505
thread 0 suma a[345]=345 suma=53745
thread 0 suma a[346]=346 suma=53985
thread 0 suma a[347]=347 suma=54225
thread 0 suma a[348]=348 suma=54465
thread 0 suma a[349]=349 suma=54705
thread 0 suma a[350]=350 suma=54945
thread 0 suma a[351]=351 suma=55185
thread 0 suma a[352]=352 suma=55425
thread 0 suma a[353]=353 suma=55665
thread 0 suma a[354]=354 suma=55905
thread 0 suma a[355]=355 suma=56145
thread 0 suma a[356]=356 suma=56385
thread 0 suma a[357]=357 suma=56625
thread 0 suma a[358]=358 suma=56865
thread 0 suma a[359]=359 suma=57105
thread 0 suma a[360]=360 suma=57345
thread 0 suma a[361]=361 suma=57585
thread 0 suma a[362]=362 suma=57825
thread 0 suma a[363]=363 suma=58065
thread 0 suma a[364]=364 suma=58305
thread 0 suma a[365]=365 suma=58545
thread 0 suma a[366]=366 suma=58785
thread 0 suma a[367]=367 suma=59025
thread 0 suma a[368]=368 suma=59265
thread 0 suma a[369]=369 suma=59505
thread 0 suma a[370]=370 suma=59745
thread 0 suma a[371]=371 suma=59985
thread 0 suma a[372]=372 suma=60225
thread 0 suma a[373]=373 suma=60465
thread 0 suma a[374]=374 suma=60705
thread 0 suma a[375]=375 suma=60945
thread 0 suma a[376]=376 suma=61185
thread 0 suma a[377]=377 suma=61425
thread 0 suma a[378]=378 suma=61665
thread 0 suma a[379]=379 suma=61905
thread 0 suma a[380]=380 suma=62145
thread 0 suma a[381]=381 suma=62385
thread 0 suma a[382]=382 suma=62625
thread 0 suma a[383]=383 suma=62865
thread 0 suma a[384]=384 suma=63105
thread 0 suma a[385]=385 suma=63345
thread 0 suma a[386]=386 suma=63585
thread 0 suma a[387]=387 suma=63825
thread 0 suma a[388]=388 suma=64065
thread 0 suma a[389]=389 suma=64305
thread 0 suma a[390]=390 suma=64545
thread 0 suma a[391]=391 suma=64785
thread 0 suma a[392]=392 suma=65025
thread 0 suma a[393]=393 suma=65265
thread 0 suma a[394]=394 suma=65505
thread 0 suma a[395]=395 suma=65745
thread 0 suma a[396]=396 suma=65985
thread 0 suma a[397]=397 suma=66225
thread 0 suma a[398]=398 suma=66465
thread 0 suma a[399]=399 suma=66705
thread 0 suma a[400]=400 suma=66945
thread 0 suma a[401]=401 suma=67185
thread 0 suma a[402]=402 suma=67425
thread 0 suma a[403]=403 suma=67665
thread 0 suma a[404]=404 suma=67905
thread 0 suma a[405]=405 suma=68145
thread 0 suma a[406]=406 suma=68385
thread 0 suma a[407]=407 suma=68625
thread 0 suma a[408]=408 suma=68865
thread 0 suma a[409]=409 suma=69105
thread 0 suma a[410]=410 suma=69345
thread 0 suma a[411]=411 suma=69585
thread 0 suma a[412]=412 suma=69825
thread 0 suma a[413]=413 suma=70065
thread 0 suma a[414]=414 suma=70305
thread 0 suma a[415]=415 suma=70545
thread 0 suma a[416]=416 suma=70785
thread 0 suma a[417]=417 suma=71025
thread 0 suma a[418]=418 suma=71265
thread 0 suma a[419]=419 suma=71505
thread 0 suma a[420]=420 suma=71745
thread 0 suma a[421]=421 suma=71985
thread 0 suma a[422]=422 suma=72225
thread 0 suma a[423]=423 suma=72465
thread 0 suma a[424]=424 suma=72705
thread 0 suma a[425]=425 suma=72945
thread 0 suma a[426]=426 suma=73185
thread 0 suma a[427]=427 suma=73425
thread 0 suma a[428]=428 suma=73665
thread 0 suma a[429]=429 suma=73905
thread 0 suma a[430]=430 suma=74145
thread 0 suma a[431]=431 suma=74385
thread 0 suma a[432]=432 suma=74625
thread 0 suma a[433]=433 suma=74865
thread 0 suma a[434]=434 suma=75105
thread 0 suma a[435]=435 suma=75345
thread 0 suma a[436]=436 suma=75585
thread 0 suma a[437]=437 suma=75825
thread 0 suma a[438]=438 suma=76065
thread 0 suma a[439]=439 suma=76305
thread 0 suma a[440]=440 suma=76545
thread 0 suma a[441]=441 suma=76785
thread 0 suma a[442]=442 suma=77025
thread 0 suma a[443]=443 suma=77265
thread 0 suma a[444]=444 suma=77505
thread 0 suma a[445]=445 suma=77745
thread 0 suma a[446]=446 suma=77985
thread 0 suma a[447]=447 suma=78225
thread 0 suma a[448]=448 suma=78465
thread 0 suma a[449]=449 suma=78705
thread 0 suma a[450]=450 suma=78945
thread 0 suma a[451]=451 suma=79185
thread 0 suma a[452]=452 suma=79425
thread 0 suma a[453]=453 suma=79665
thread 0 suma a[454]=454 suma=79905
thread 0 suma a[455]=455 suma=80145
thread 0 suma a[456]=456 suma=80385
thread 0 suma a[457]=457 suma=80625
thread 0 suma a[458]=458 suma=80865
thread 0 suma a[459]=459 suma=81105
thread 0 suma a[460]=460 suma=81345
thread 0 suma a[461]=461 suma=81585
thread 0 suma a[462]=462 suma=81825
thread 0 suma a[463]=463 suma=82065
thread 0 suma a[464]=464 suma=82305
thread 0 suma a[465]=465 suma=82545
thread 0 suma a[466]=466 suma=82785
thread 0 suma a[467]=467 suma=83025
thread 0 suma a[468]=468 suma=83265
thread 0 suma a[469]=469 suma=83505
thread 0 suma a[470]=470 suma=83745
thread 0 suma a[471]=471 suma=83985
thread 0 suma a[472]=472 suma=84225
thread 0 suma a[473]=473 suma=84465
thread 0 suma a[474]=474 suma=84705
thread 0 suma a[475]=475 suma=84945
thread 0 suma a[476]=476 suma=85185
thread 0 suma a[477]=477 suma=85425
thread 0 suma a[478]=478 suma=85665
thread 0 suma a[479]=479 suma=85905
thread 0 suma a[480]=480 suma=86145
thread 0 suma a[481]=481 suma=86385
thread 0 suma a[482]=482 suma=86625
thread 0 suma a[483]=483 suma=86865
thread 0 suma a[484]=484 suma=87105
thread 0 suma a[485]=485 suma=87345
thread 0 suma a[486]=486 suma=87585
thread 0 suma a[487]=487 suma=87825
thread 0 suma a[488]=488 suma=88065
thread 0 suma a[489]=489 suma=88305
thread 0 suma a[490]=490 suma=88545
thread 0 suma a[491]=491 suma=88785
thread 0 suma a[492]=492 suma=89025
thread 0 suma a[493]=493 suma=89265
thread 0 suma a[494]=494 suma=89505
thread 0 suma a[495]=495 suma=89745
thread 0 suma a[496]=496 suma=89985
thread 0 suma a[497]=497 suma=90225
thread 0 suma a[498]=498 suma=90465
thread 0 suma a[499]=499 suma=90705
thread 0 suma a[500]=500 suma=90945
thread 0 suma a[501]=501 suma=91185
thread 0 suma a[502]=502 suma=91425
thread 0 suma a[503]=503 suma=91665
thread 0 suma a[504]=504 suma=91905
thread 0 suma a[505]=505 suma=92145
thread 0 suma a[506]=506 suma=92385
thread 0 suma a[507]=507 suma=92625
thread 0 suma a[508]=508 suma=92865
thread 0 suma a[509]=509 suma=93105
thread 0 suma a[510]=510 suma=93345
thread 0 suma a[511]=511 suma=93585
thread 0 suma a[512]=512 suma=93825
thread 0 suma a[513]=513 suma=94065
thread 0 suma a[514]=514 suma=94305
thread 0 suma a[515]=515 suma=94545
thread 0 suma a[516]=516 suma=94785
thread 0 suma a[517]=517 suma=95025
thread 0 suma a[518]=518 suma=95265
thread 0 suma a[519]=519 suma=95505
thread 0 suma a[520]=520 suma=95745
thread 0 suma a[521]=521 suma=95985
thread 0 suma a[522]=522 suma=96225
thread 0 suma a[523]=523 suma=96465
thread 0 suma a[524]=524 suma=96705
thread 0 suma a[525]=525 suma=96945
thread 0 suma a[526]=526 suma=97185
thread 0 suma a[527]=527 suma=97425
thread 0 suma a[528]=528 suma=97665
thread 0 suma a[529]=529 suma=97905
thread 0 suma a[530]=530 suma=98145
thread 0 suma a[531]=531 suma=98385
thread 0 suma a[532]=532 suma=98625
thread 0 suma a[533]=533 suma=98865
thread 0 suma a[534]=534 suma=99105
thread 0 suma a[535]=535 suma=99345
thread 0 suma a[536]=536 suma=99585
thread 0 suma a[537]=537 suma=99825
thread 0 suma a[538]=538 suma=100065
thread 0 suma a[539]=539 suma=100305
thread 0 suma a[540]=540 suma=100545
thread 0 suma a[541]=541 suma=100785
thread 0 suma a[542]=542 suma=101025
thread 0 suma a[543]=543 suma=101265
thread 0 suma a[544]=544 suma=101505
thread 0 suma a[545]=545 suma=101745
thread 0 suma a[546]=546 suma=101985
thread 0 suma a[547]=547 suma=102225
thread 0 suma a[548]=548 suma=102465
thread 0 suma a[549]=549 suma=102705
thread 0 suma a[550]=550 suma=102945
thread 0 suma a[551]=551 suma=103185
thread 0 suma a[552]=552 suma=103425
thread 0 suma a[553]=553 suma=103665
thread 0 suma a[554]=554 suma=103905
thread 0 suma a[555]=555 suma=104145
thread 0 suma a[556]=556 suma=104385
thread 0 suma a[557]=557 suma=104625
thread 0 suma a[558]=558 suma=104865
thread 0 suma a[559]=559 suma=105105
thread 0 suma a[560]=560 suma=105345
thread 0 suma a[561]=561 suma=105585
thread 0 suma a[562]=562 suma=105825
thread 0 suma a[563]=563 suma=106065
thread 0 suma a[564]=564 suma=106305
thread 0 suma a[565]=565 suma=106545
thread 0 suma a[566]=566 suma=106785
thread 0 suma a[567]=567 suma=107025
thread 0 suma a[568]=568 suma=107265
thread 0 suma a[569]=569 suma=107505
thread 0 suma a[570]=570 suma=107745
thread 0 suma a[571]=571 suma=107985
thread 0 suma a[572]=572 suma=108225
thread 0 suma a[573]=573 suma=108465
thread 0 suma a[574]=574 suma=108705
thread 0 suma a[575]=575 suma=108945
thread 0 suma a[576]=576 suma=109185
thread 0 suma a[577]=577 suma=109425
thread 0 suma a[578]=578 suma=109665
thread 0 suma a[579]=579 suma=109905
thread 0 suma a[580]=580 suma=110145
thread 0 suma a[581]=581 suma=110385
thread 0 suma a[582]=582 suma=110625
thread 0 suma a[583]=583 suma=110865
thread 0 suma a[584]=584 suma=111
```


CÓDIGO FUENTE: scheduled-clauseModificado5.c

```

/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 o 9 .*/
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/
/* INTERLINEADO SENCILLO */

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

int main(int argc, char ** argv)
{
    int i, n=16, 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]);

    omp_sched_t kind;
    int modifier;

    omp_get_schedule(&kind, & modifier);
    printf("Fuera sin modificar\n");
    printf("dyn-var=%d  nthreads_var=%d  run-sched-var=(%d,%d)\n",

    omp_get_dynamic(), omp_get_max_threads(), kind, modifier);
    // Las modifico
    /*Un valor que indica si el número de subprocesos disponibles en
la
de
número
número de subprocesos
*/
    omp_set_dynamic(4);
    omp_set_num_threads(3);
    kind = 1;
    modifier = 2;
    omp_set_schedule(kind, modifier);
    printf("Fuera Modificadas\n");
    printf("dyn-var=%d  nthreads_var=%d  run-sched-var=(%d,%d)\n",

    omp_get_dynamic(), omp_get_max_threads(), kind, modifier);

    for (i=0; i<n; i++) a[i] = i;
        #pragma omp parallel num_threads(4)
        {
            #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
                                {
                                omp_get_schedule(&kind,
& modifier);
                                printf("Fuera sin
                                modificar\n");
                                printf("dyn-var=%d
                                nthreads_var=%d run-sched-var=(%d,%d)\n",
                                omp_get_dynamic(),omp_get_max_threads(),kind,modifier);
                                // Las modifico
                                omp_set_dynamic(0);
                                omp_set_num_threads(7);
                                kind=3;
                                modifier=2;
                                omp_set_schedule(kind,
                                modifier);
                                printf("Fuera
                                Modificadas\n");
                                printf("dyn-var=%d
                                nthreads_var=%d run-sched-var=(%d,%d)\n",
                                omp_get_dynamic(),omp_get_max_threads(),kind,modifier);
                                }
                                }

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

```

CAPTURAS DE PANTALLA:

```

terminal
jose@jose-K55VM: ~/GII/AC/Practicas_grupo reducido_/Practica3
jose@Practica3$ gcc -O2 scheduled-clause_modificado5.c -o scheduled-clause_modificado5 -lrt -fopenmp
jose@Practica3$ clear
jose@Practica3$ ./scheduled-clause_modificado5 20 4
Fuera sin modificar
dyn-var=0 nthreads_var=8 run-sched-var=(2,1)
Fuera Modificadas
dyn-var=1 nthreads_var=3 run-sched-var=(1,2)
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[12]=12 suma=18
thread 0 suma a[13]=13 suma=31
thread 0 suma a[14]=14 suma=45
thread 0 suma a[15]=15 suma=60
thread 0 suma a[16]=16 suma=76
thread 0 suma a[17]=17 suma=93
thread 0 suma a[18]=18 suma=111
thread 0 suma a[19]=19 suma=130
thread 2 suma a[8]=8 suma=8
thread 2 suma a[9]=9 suma=17
thread 2 suma a[10]=10 suma=27
thread 2 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
Fuera sin modificar
dyn-var=1 nthreads_var=3 run-sched-var=(1,2)
Fuera Modificadas
dyn-var=0 nthreads_var=7 run-sched-var=(3,2)
Fuera de 'parallel for' suma=130
jose@Practica3$
jose@Practica3$
jose@Practica3$
jose@Practica3$

```

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.

CÓDIGO FUENTE: pmtv-secuencial.c

```
/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 o 9 .*/
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/
/* INTERLINEADO SENCILLO */

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

int main(int argc, char ** argv)
{
    // Variables
    int i,j;

    //Leer argumento de entrada (no de componentes de la matriz)
    if (argc<2){
        printf("Faltan no componentes del vector\n");
        exit(-1);
    }

    unsigned int N = atoi(argv[1]);

    double *M, *v1, *v2;
    M = (double*) malloc(N*N*sizeof(double)); // malloc necesita el
tamaño en bytes
    v1 = (double*) malloc(N*sizeof(double)); //si no hay espacio
suficiente malloc devuelve NULL
    v2 = (double*) malloc(N*sizeof(double));
    if ( (M==NULL) || (v1==NULL) || (v2==NULL) ){
        printf("Error en la reserva de espacio para los
vectores\n");
        exit(-2);
    }

    // Inicialización de la matriz y vector;
    for(i=0;i<N ;i++){
        for(j=0;j<N;j++){
            if(j<=i){
                M[i*N+j] = i+j+1;
            }else{
                M[i*N+j] = 0;
            }
            v1[i] = i+1;
        }
    }

    // Calculos
    for(i=0; i<N; i++){
        for(j=0; j<=i; j++)
```

```

        v2[i]+=M[i*N+j]*v1[j];
    }

    // Resultados
    printf("Primer componente= %2.3f \n",v2[0] );
    printf("Ultimo componente= %2.3f \n",v2[N-1] );

    // Visualiza las matrices si no son muy grandes
    // Se recomienda redirigir la salida a un fichero.
    if (N < 20) {
        printf("\n\t M \n");
        for(i=0; i<N; i++){
            printf("| ");
            for(j=0; j<N; j++)
                printf(" %2.2f ",
M[i*N+j]);

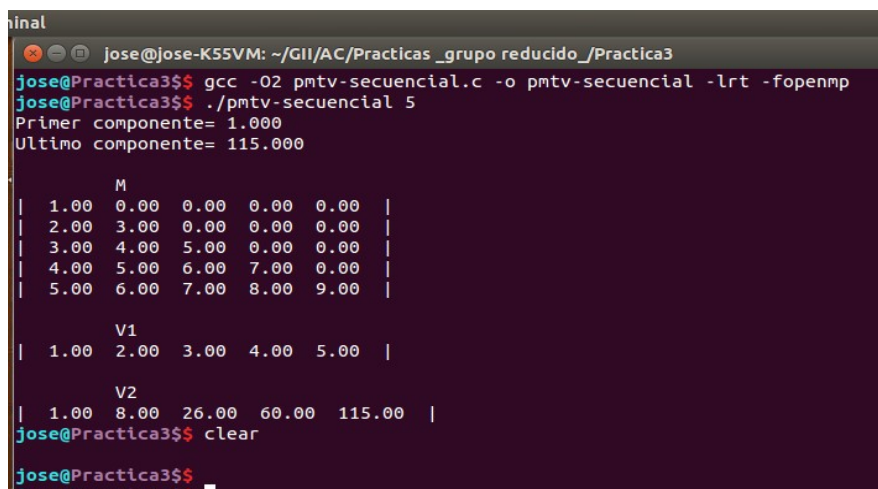
            printf(" |\n");
        }
        printf("\n\t V1 \n| ");
        for(i=0; i<N; i++){
            printf(" %2.2f ", v1[i]);
        }
        printf(" |\n");

        printf("\n\t V2 \n| ");
        for(i=0; i<N; i++){
            printf(" %2.2f ", v2[i]);
        }
        printf(" |\n");
    }

    free(M); // libera el espacio reservado para v1
    free(v1); // libera el espacio reservado para v2
    free(v2); // libera el espacio reservado para v3
}

```

CAPTURAS DE PANTALLA: (ADJUNTAR CÓDIGO FUENTE AL .ZIP)



```

jose@jose-K55VM: ~/Gil/AC/Practicas_grupo reducido_/Practica3
jose@Practica3$ gcc -O2 pmtv-secuencial.c -o pmtv-secuencial -lrt -fopenmp
jose@Practica3$ ./pmtv-secuencial 5
Primer componente= 1.000
Ultimo componente= 115.000

      M
| 1.00 0.00 0.00 0.00 0.00 |
| 2.00 3.00 0.00 0.00 0.00 |
| 3.00 4.00 5.00 0.00 0.00 |
| 4.00 5.00 6.00 7.00 0.00 |
| 5.00 6.00 7.00 8.00 9.00 |

      V1
| 1.00 2.00 3.00 4.00 5.00 |

      V2
| 1.00 8.00 26.00 60.00 115.00 |
jose@Practica3$ clear
jose@Practica3$

```

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:

CÓDIGO FUENTE: `pmtv-OpenMP.c`

```
/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 o 9 */
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ */
/* INTERLINEADO SENCILLO */

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

int main(int argc, char ** argv)
{
    // Variables
    int i,j;
    struct timespec cgt1,cgt2; double ncgt; //para tiempo de
ejecución

    omp_sched_t kind;
    int modifier;

    //Leer argumento de entrada (no de componentes de la matriz)
    if (argc<2){
        printf("Error: ./pmtv-OpenMP N kind chunk\n");
        exit(-1);
    }

    unsigned int N = atoi(argv[1]);
    // kind = atoi(argv[2]);
    // modifier = atoi(argv[3]);

    double *M, *v1, *v2;
    M = (double*) malloc(N*N*sizeof(double)); // malloc necesita el
tamaño en bytes
    v1 = (double*) malloc(N*sizeof(double)); //si no hay espacio
suficiente malloc devuelve NULL
    v2 = (double*) malloc(N*sizeof(double));
    if ( (M==NULL) || (v1==NULL) || (v2==NULL) ){
        printf("Error al reservar de espacio para los
vectores\n");
```

```

        exit(-2);
    }

    // Pido los parametros de OMP_SCHEDULE
    // indicar que pido por la funcion por que tiene mas sentido
//
    #ifdef _OPENMP
        //omp_set_schedule(kind);
        omp_get_schedule(&kind, & modifier);
        printf("Asignado run-sched-var=(%d,
%d)\n", kind, modifier);
//s
    #endif

    // Inicialización de la matriz y vector;
    for(i=0; i<N ;i++){
        for(j=0; j<N; j++){
            if(j<=i){
                M[i*N+j] = i+j+1;
            }else{
                M[i*N+j] = 0;
            }
        }
        v1[i] = i+1;
    }

    // Calculos
    clock_gettime(CLOCK_REALTIME, &cgt1);
    #pragma omp parallel for private(j)
    for(i=0; i<N; i++){
        for(j=0; j<=i; j++){
            v2[i] += M[i*N+j] * v1[j];
        }
    }
    clock_gettime(CLOCK_REALTIME, &cgt2);

    ncgt = (double) (cgt2.tv_sec - cgt1.tv_sec) + (double)
((cgt2.tv_nsec - cgt1.tv_nsec) / (1.e+9));

    // Resultados
//
    printf("Tiempos de schedule= %d \t chunk= %d
-----\n", kind, modifier );
    printf("Tiempo de ejecución= %2.11f\n", ncgt);
    printf("Primer componente= %2.3f \n", v2[0] );
    printf("Ultimo componente= %2.3f \n", v2[N-1] );
//

printf("-----\n");

    // Visualiza las matrices si no son muy grandes
    // Se recomienda redirigir la salida a un fichero.
    if (N < 20) {
        printf("\n\t M \n");
        for(i=0; i<N; i++){
            printf("| ");
            for(j=0; j<N; j++){
                printf(" %2.2f ",
M[i*N+j]);

                printf(" |\n");
            }
            printf("\n\t v1 \n| ");
            for(i=0; i<N; i++){
                printf(" %2.2f ", v1[i]);
            }
        }
    }

```

```

    }
    printf(" |\n");

    printf("\n\t V2 |\n");
    for(i=0; i<N; i++){
        printf(" %2.2f ", v2[i]);
    }
    printf(" |\n");
}

free(M); // libera el espacio reservado para v1
free(v1); // libera el espacio reservado para v2
free(v2); // libera el espacio reservado para v3
}

```

DESCOMPOSICIÓN DE DOMINIO:**CAPTURAS DE PANTALLA:
(ADJUNTAR CÓDIGO FUENTE AL .ZIP)****TABLA RESULTADOS, SCRIPT Y GRÁFICA ATCGRID****SCRIPT:** pmtv-OpenMP_atcgrid.sh

```

#!/bin/bash
10. #Se asigna al trabajo el nombre pmtv-OpenMP
11. #PBS -N pmtv-OpenMP
12. #Se asigna al trabajo la cola ac
13. #PBS -q ac
14.
15. #Se ejecuta pmtv-OpenMP, que está en el directorio en el que se ha
    ejecutado qsub,
16. #          N SC CH
17. export OMP_SCHEDULE="static"
18. echo "static - default"
19. ./pmtv-OpenMP 576
20. export OMP_SCHEDULE="static,1"
21. echo "static - 1"
22. ./pmtv-OpenMP 576
23. export OMP_SCHEDULE="static,64"
24. echo "static - 64"
25. ./pmtv-OpenMP 576
26. echo "-----"
27. export OMP_SCHEDULE="dynamic"
28. echo "dynamic - default"
29. ./pmtv-OpenMP 576
30. export OMP_SCHEDULE="dynamic,1"
31. echo "dynamic - 1"
32. ./pmtv-OpenMP 576
33. export OMP_SCHEDULE="dynamic,64"
34. echo "dynamic - 64"
35. ./pmtv-OpenMP 576
36. echo "-----"
37. export OMP_SCHEDULE="guided"
38. echo "guided - default"
39. ./pmtv-OpenMP 576
40. export OMP_SCHEDULE="guided,1"
41. echo "guided - 1"
42. ./pmtv-OpenMP 576
43. export OMP_SCHEDULE="guided,64"

```

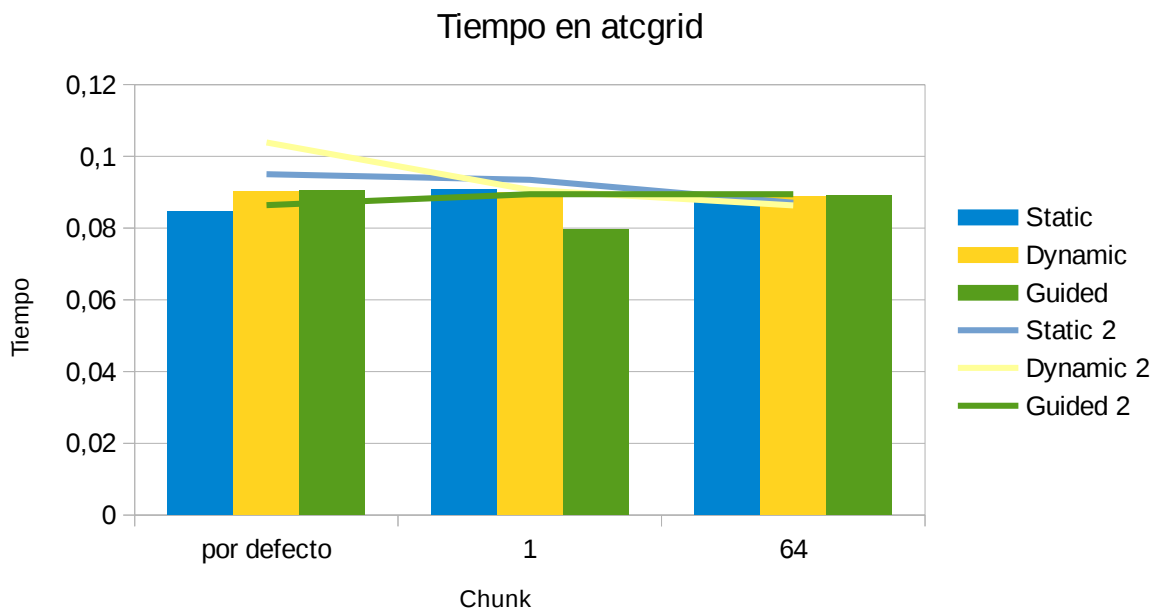
```

44. echo "guided - 64"
45. ./pmtv-OpenMP 576
46. echo "fin correctamente"

```

Tabla 3. Tiempos de ejecución de la versión paralela del producto de una matriz triangular por un vector r para vectores de tamaño $N=17280$, 12 threads

Chunk	Static	Dynamic	Guided
por defecto	0.08452796500	0.09025187500	0.09054618800
1	0.09086678700	0.08993644200	0.07955836900
64	0.08825790800	0.08876919800	0.08903223400
Chunk	Static	Dynamic	Guided
por defecto	0.09503212700	0.10390033000	0.08642198600
1	0.09344813500	0.09065560900	0.08944094100
64	0.08704507500	0.08632124100	0.08942839800



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

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

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.

CÓDIGO FUENTE:

```

/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 o 9 .*/
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ*/
/* INTERLINEADO SENCILLO */

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

int main(int argc, char ** argv)
{
    // Variables
    int i,j,k;
    struct timespec cgt1,cgt2; double ncgt; //para tiempo de
ejecución
    omp_sched_t kind;
    int modifier;

    //Leer argumento de entrada (no de componentes de la matriz)
    if (argc<2){
        printf("Faltan no componentes del vector\n");
        exit(-1);
    }

    unsigned int N = atoi(argv[1]);

    double *M1, *M2, *M3;
    M1 = (double*) malloc(N*N*sizeof(double)); // malloc necesita el
tamaño en bytes
    M2 = (double*) malloc(N*N*sizeof(double)); //si no hay espacio
suficiente malloc devuelve NULL
    M3 = (double*) malloc(N*N*sizeof(double));
    if ( (M1==NULL) || (M1==NULL) || (M2==NULL) ){
        printf("Error en la reserva de espacio para los
vectores\n");
        exit(-2);
    }

    // Inicialización de la matriz y vector;
    for(i=0;i<N ;i++){
        for(j=0;j<N;j++){
            M1[i*N+j] = i+1;
            M2[i*N+j] = j+1;
            M3[i*N+j] = 0;
        }
    }

    // Calculos
    clock_gettime(CLOCK_REALTIME,&cgt1);
    //#pragma omp parallel for private(j)
    for(i=0; i<N; i++){
        for(j=0; j<N; j++)
            for(k=0; k <N; k ++){
                M3[i*N+j]
+=M1[i*N+k]*M2[k*N+j];
            }
        clock_gettime(CLOCK_REALTIME,&cgt2);

        ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+ (double)

```

```

((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

// Resultados
printf("Tiempo de ejecución= %2.11f\n",ncgt);
printf("Primer componente= %2.3f \n",M3[0] );
printf("Ultimo componente= %2.3f \n",M3[N*N-1] );

// Visualiza las matrices si no son muy grandes
// Se recomienda redirigir la salida a un fichero.
if (N < 20) {
    printf("\n\t M \n");
    for(i=0; i<N; i++){
        printf("| ");
        for(j=0; j<N; j++)
            printf(" %2.2f ",
M1[i*N+j]);

        printf(" |\n");
    }
    printf("\n\t V1 \n| ");
    for(i=0; i<N; i++){
        for(j=0; j<N;j++)
            printf(" %2.2f ",
M2[i*N+j]);

        printf(" |\n");
    }
    printf(" |\n");

    printf("\n\t V2 \n| ");
    for(i=0; i<N; i++){
        for(j=0; j<N;j++)
            printf(" %2.2f ",
M3[i*N+j]);

        printf(" |\n");
    }
    printf(" |\n");
}

free(M1); // libera el espacio reservado para v1
free(M2); // libera el espacio reservado para v2
free(M3); // libera el espacio reservado para v3
}

```

CAPTURAS DE PANTALLA:
(ADJUNTAR CÓDIGO FUENTE AL .ZIP)

```
@jose-K55VM: ~/GII/AC/Practicas_grupo reducido_/Practica3
|
jose@Practica3$ gcc -O2 pmm-secuencial.c -o pmm-secuencial -lrt -fopenmp
jose@Practica3$ ./pmm-secuencial 5 1
Tiempo de ejecución= 0.00000083000
Primer componente= 5.000
Ultimo componente= 125.000

      M
| 1.00 1.00 1.00 1.00 1.00 |
| 2.00 2.00 2.00 2.00 2.00 |
| 3.00 3.00 3.00 3.00 3.00 |
| 4.00 4.00 4.00 4.00 4.00 |
| 5.00 5.00 5.00 5.00 5.00 |

      V1
| 1.00 2.00 3.00 4.00 5.00 |
1.00 2.00 3.00 4.00 5.00 |
1.00 2.00 3.00 4.00 5.00 |
1.00 2.00 3.00 4.00 5.00 |
1.00 2.00 3.00 4.00 5.00 |

      V2
| 5.00 10.00 15.00 20.00 25.00 |
10.00 20.00 30.00 40.00 50.00 |
15.00 30.00 45.00 60.00 75.00 |
20.00 40.00 60.00 80.00 100.00 |
25.00 50.00 75.00 100.00 125.00 |
```

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).

10.

DESCOMPOSICIÓN DE DOMINIO:

CÓDIGO FUENTE: pmm-OpenMP.c

```
/* Tipo de letra Courier new o Liberation Mono. Tamaño 8 o 9 */
/* COPIAR Y PEGAR CÓDIGO FUENTE AQUÍ */
/* INTERLINEADO SENCILLO */

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

int main(int argc, char ** argv)
{
    // Variables
    int i,j,k;
    struct timespec cgt1,cgt2; double ncgt; //para tiempo de
ejecución

    //Leer argumento de entrada (no de componentes de la matriz)
    if (argc<2){
        printf("Faltan no componentes del vector\n");
        exit(-1);
    }

    unsigned int N = atoi(argv[1]);

    double *M1, *M2, *M3;
    M1 = (double*) malloc(N*N*sizeof(double)); // malloc necesita el
tamaño en bytes
    M2 = (double*) malloc(N*N*sizeof(double)); //si no hay espacio
suficiente malloc devuelve NULL
    M3 = (double*) malloc(N*N*sizeof(double));
    if ( (M1==NULL) || (M1==NULL) || (M2==NULL) ){
```

```

        printf("Error en la reserva de espacio para los
vectores\n");
        exit(-2);
    }

    // Inicialización de la matriz y vector;
    #pragma omp parallel for private(j)
    for(i=0; i<N; i++){
        for(j=0; j<N; j++){
            M1[i*N+j] = i+1;
            M2[i*N+j] = j+1;
            M3[i*N+j] = 0;
        }
    }

    // Calculos
    clock_gettime(CLOCK_REALTIME,&cgt1);
    #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 ++){
                M3[i*N+j]
+=M1[i*N+k]*M2[k*N+j];
            }
        }
        clock_gettime(CLOCK_REALTIME,&cgt2);

        ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+ (double)
((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

    // Resultados
    printf("Tiempo de ejecución= %2.11f\n",ncgt);
    printf("Primer componente= %2.3f \n",M3[0] );
    printf("Ultimo componente= %2.3f \n",M3[N*N-1] );

    // Visualiza las matrices si no son muy grandes
    // Se recomienda redirigir la salida a un fichero.
    if (N < 20) {
        printf("\n\t M \n");
        for(i=0; i<N; i++){
            printf("| ");
            for(j=0; j<N; j++){
                printf(" %2.2f ",
M1[i*N+j]);

                printf(" |\n");
            }
            printf("\n\t V1 \n| ");
            for(i=0; i<N; i++){
                for(j=0; j<N; j++){
                    printf(" %2.2f ",
M2[i*N+j]);

                    printf(" |\n");
                }
                printf(" |\n");

                printf("\n\t V2 \n| ");
                for(i=0; i<N; i++){
                    for(j=0; j<N; j++){
                        printf(" %2.2f ",
M3[i*N+j]);

                        printf(" |\n");
                    }
                }
            }
        }
    }
}

```

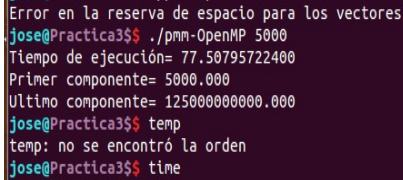
```

        printf(" |\n");
    }

    free(M1); // libera el espacio reservado para v1
    free(M2); // libera el espacio reservado para v2
    free(M3); // libera el espacio reservado para v3
}

```

CAPTURAS DE PANTALLA: (ADJUNTAR CÓDIGO FUENTE AL .ZIP)



```

Error en la reserva de espacio para los vectores
jose@Practica3$ ./pmm-OpenMP 5000
Tiempo de ejecución= 77.50795722400
Primer componente= 5000.000
Ultimo componente= 125000000000.000
jose@Practica3$ temp
temp: no se encontró la orden
jose@Practica3$ time

```

47. Hacer un estudio de escalabilidad (ganancia en velocidad en función del número de cores) en atcgrid y en el PC local del código paralelo implementado para dos tamaños de las matrices. Debe recordar usar `-O2` al compilar. 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.

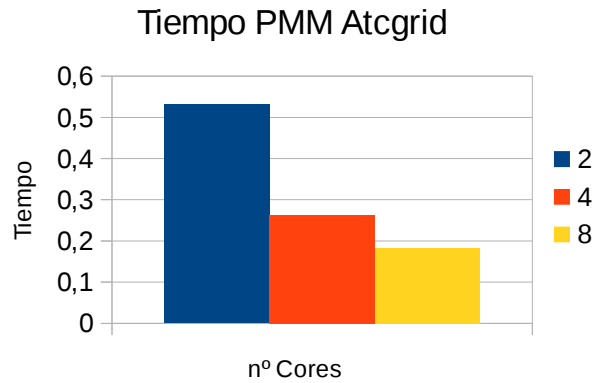
ESTUDIO DE ESCALABILIDAD EN ATCGRID:

SCRIPT: pmm-OpenMP_atcgrid.sh

```

#!/bin/bash
48. #Se asigna al trabajo el nombre pmm-OpenMP
49. #PBS -N pmm-OpenMP
50. #Se asigna al trabajo la cola ac
51. #PBS -q ac
52.
53. #Se ejecuta pmm-OpenMP, que está en el directorio en el que se ha
    ejecutado qsub,
54. #
55. export OMP_NUM_THREADS=2
56. echo "threads = 2 - 1"
57. ./pmm-OpenMP 576
58. echo "threads = 2 - 2"
59. ./pmm-OpenMP 576
60. export OMP_NUM_THREADS=4
61. echo "threads = 4 - 1"
62. ./pmm-OpenMP 576
63. echo "threads = 4 - 2"
64. ./pmm-OpenMP 576
65. export OMP_NUM_THREADS=8
66. echo "threads = 8 - 1"
67. ./pmm-OpenMP 576
68. echo "threads = 8 - 2"
69. ./pmm-OpenMP 576

```



ESTUDIO DE ESCALABILIDAD EN PCLOCAL:

SCRIPT: pmm-OpenMP_pclocal.sh

```
#!/bin/bash
70. #Se asigna al trabajo el nombre pmtv-OpenMP
71. #PBS -N pmtv-OpenMP
72. #Se asigna al trabajo la cola ac
73. #PBS -q ac
74.
75. #Se ejecuta pmtv-OpenMP, que está en el directorio en el que se ha
    ejecutado qsub,
76. #          N SC CH
77. export OMP_NUM_THREADS=2
78. echo "threads = 2 - 1"
79. ./pmm-OpenMP 576
80. echo "threads = 2 - 2"
81. ./pmm-OpenMP 576
82. export OMP_NUM_THREADS=4
83. echo "threads = 4 - 1"
84. ./pmm-OpenMP 576
85. echo "threads = 4 - 2"
86. ./pmm-OpenMP 576
87. export OMP_NUM_THREADS=8
88. echo "threads = 8 - 1"
89. ./pmm-OpenMP 576
90. echo "threads = 8 - 2"
91. ./pmm-OpenMP 576
```

