2º curso / 2º cuatr.

Grado Ing. Inform.

Doble Grado Ing. Inform. y Mat.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 1. Programación paralela I: Directivas OpenMP

Estudiante (nombre y apellidos): Alejandro Molina Criado
Grupo de prácticas y profesor de prácticas: A2
(Christian Agustín Morillas Gutierrez)
Fecha de entrega: 25/03/2020
Fecha evaluación en clase: 25/03/2020

Antes de comenzar a realizar el trabajo de este cuaderno consultar el fichero con los normas de prácticas que se encuentra en SWAD

Ejercicios basados en los ejemplos del seminario práctico

1. Usar la directiva `parallel` combinada con directivas de trabajo compartido en los ejemplos `bucle-for.c` y `sections.c` del seminario. Incorporar el código fuente resultante al cuaderno de prácticas.

RESPUESTA: Captura que muestre el código fuente

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

int main(int argc , char **argv){

    int i,n=9;

    if (argc < 2){
        fprintf(stderr,"\n[ERROR] - FALTA Nº ITERACIONES");
        exit(-1);
    }

    n = atoi(argv[1]);

    #pragma omp parallel for

        for(i=0;i<n;i++){
            printf("thread %d ejecuta la iteración %d del bucle\n",omp_get_thread_num(),i);
        }


    return 0;

}
```

RESPUESTA: Captura que muestre el código fuente `sectionsModificado.c`

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


void funcionA(){
    printf("En funcA: esta seccion ejecuta el thread %d\n",omp_get_thread_num());
}

void funcionB(){
    printf("En funcB: esta seccion ejecuta el thread %d\n",omp_get_thread_num());
}


int main(int argc , char **argv){

    #pragma omp parallel sections
    {
            #pragma omp section
                (void) funcionA();
            #pragma omp section
                (void) funcionB();

    }

}
```

2. Imprimir los resultados del programa `single.c` usando una directiva `single` dentro de la construcción `parallel` en lugar de imprimirlos fuera de la región `parallel`. Añadir lo necesario, dentro de la nueva directiva `single` incorporada, para que se imprima el identificador del thread que ejecuta el bloque estructurado de la directiva `single`. Incorpore en su cuaderno de trabajo el código fuente y volcados de pantalla con los resultados de ejecución obtenidos.

RESPUESTA: Captura que muestre el código fuente `singleModificado.c`

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

int main(){

    int n = 9 ;
    int i , a , b[n];

    for(i=0 ; i<n ; i++) b[i]=-1;

    #pragma omp parallel
    {

        #pragma omp single
        {
            printf("Escribe valor de inicialización a:");
            scanf("%d",&a);
            printf("Single ejecutada por el hilo %d\n",omp_get_thread_num());
        } // <-- Barrera
        #pragma omp for
        for(i=0;i<n;i++){
            b[i]=a;
        }


        #pragma omp single
        {
            for(i=0;i<n;i++) printf("b[%d]=%d\t",i,b[i]);
            printf("\n");
        }
    }
}
```

CAPTURAS DE PANTALLA:

Ejecución en PC LOCAL:

```
alex@alex-CX62-6QD:~/bp1/ejercicio2$ gcc -O2 -fopenmp -o singleModificado singleModificado.c
alex@alex-CX62-6QD:~/bp1/ejercicio2$ ./singleModificado
Escribe valor de inicialización a:8
Single ejecutada por el hilo 2
b[0]=8  b[1]=8  b[2]=8  b[3]=8  b[4]=8  b[5]=8  b[6]=8  b[7]=8  b[8]=8
alex@alex-CX62-6QD:~/bp1/ejercicio2$ ./singleModificado
Escribe valor de inicialización a:10
Single ejecutada por el hilo 1
b[0]=10 b[1]=10 b[2]=10 b[3]=10 b[4]=10 b[5]=10 b[6]=10 b[7]=10 b[8]=10
```

3. Imprimir los resultados del programa `single.c` usando una directiva `master` dentro de la construcción `parallel` en lugar de imprimirlos fuera de la región `parallel`. Añadir lo necesario, dentro de la nueva directiva `master` incorporada, para que se imprima el identificador del thread que ejecuta el bloque estructurado de la directiva `master`. Incorpore en su cuaderno el código fuente y volcados de pantalla con los resultados de ejecución obtenidos. ¿Qué diferencia observa con respecto a los resultados de ejecución del ejercicio anterior?

RESPUESTA: Captura que muestre el código fuente `singleModificado2.c`

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

int main(){

    int n = 9 ;
    int i , a , b[n];

    for(i=0 ; i<n ; i++) b[i]=-1;

    #pragma omp parallel
    {

        #pragma omp single
        {
            printf("Escribe valor de inicialización a:");
            scanf("%d",&a);
            printf("Single ejecutada por el hilo %d\n",omp_get_thread_num());
        } // <-- Barrera
        #pragma omp for
        for(i=0;i<n;i++){
            b[i]=a;
        }


        #pragma omp master
        {
            printf("Dentro de la región parallel:\n");
            for(i=0;i<n;i++) printf("b[%d]=%d\t",i,b[i]);
            printf("Directiva master ejecutada por el hilo %d\n",omp_get_thread_num());
        }
    }
}
```

CAPTURAS DE PANTALLA:

```
[AlejandroMolinaCriado a2estudiante13@atcgrid:~/bp1/ejercicio2] 2020-03-25 miércoles
$gcc -O2 -fopenmp -o singleModificado2 singleModificado2.c
[AlejandroMolinaCriado a2estudiante13@atcgrid:~/bp1/ejercicio2] 2020-03-25 miércoles
$./singleModificado2
Escribe valor de inicialización a:8
Single ejecutada por el hilo 4
Dentro de la región parallel:
b[0]=8  b[1]=8  b[2]=8  b[3]=8  b[4]=8  b[5]=8  b[6]=8  b[7]=8  b[8]=8  Directiva master ejecutada por el hilo 0
[AlejandroMolinaCriado a2estudiante13@atcgrid:~/bp1/ejercicio2] 2020-03-25 miércoles
$./singleModificado2
Escribe valor de inicialización a:9
Single ejecutada por el hilo 1
Dentro de la región parallel:
b[0]=9  b[1]=9  b[2]=9  b[3]=9  b[4]=9  b[5]=9  b[6]=9  b[7]=9  b[8]=9  Directiva master ejecutada por el hilo 0
[AlejandroMolinaCriado a2estudiante13@atcgrid:~/bp1/ejercicio2] 2020-03-25 miércoles
$./singleModificado2
Escribe valor de inicialización a:10
Single ejecutada por el hilo 0
Dentro de la región parallel:
b[0]=10 b[1]=10 b[2]=10 b[3]=10 b[4]=10 b[5]=10 b[6]=10 b[7]=10 b[8]=10 Directiva master ejecutada por el hilo 0
```

RESPUESTA A LA PREGUNTA:

La diferencia entre la directiva single y master **está en la hebra que ejecuta la sección** que en el caso de master siempre es la hebra número 0.

4. ¿Por qué si se elimina directiva `barrier` en el ejemplo `master.c` la suma que se calcula e imprime no siempre es correcta? Responda razonadamente.

RESPUESTA:

A diferencia de la directiva single , la directiva master no tiene barrera impícita , es decir , las hebras no esperarán a que las otras terminen su ejecución en un punto determinado .

Si no la pusieramos , la hebra master no esperaría a que las demás terminasen y el resultado de la suma sería erróneo e indeterminado.

Resto de ejercicios

5. El programa secuencial C del Listado 1 calcula la suma de dos vectores (v3 = v1 + v2; v3(i) = v1(i) + v2(i), i=0,…N-1). Generar el ejecutable del programa del Listado 1 para **vectores globales**. Usar `time` (Lección 3/ Tema 1) en la línea de comandos para obtener, en atcgrid, el tiempo de ejecución (*elapsed time*) y el tiempo de CPU del usuario y del sistema generado. Obtenga los tiempos para vectores con 10000000 componentes. ¿La suma de los tiempos de CPU del usuario y del sistema es menor, mayor o igual que el tiempo real (*elapsed*)? Justifique la respuesta.

CAPTURAS DE PANTALLA:

```
[AlejandroMolinaCriado a2estudiante13@alexmolinaatcgrid:cx626QD~/bp1/ejercicio5] 2020-03-25 miércoles
$gcc  -O2  sumavectores.c -o SumaVectores -lrt
[AlejandroMolinaCriado a2estudiante13@alexmolinaatcgrid:cx626QD~/bp1/ejercicio5] 2020-03-25 miércoles
$srun -p ac time ./SumaVectores 10000000
TamaÃ±o Vectores:10000000 (4 B)
Tiempo:0.040156621       / TamaÃ±o Vectores:10000000  / V1[0]+V2[0]=V3[0](1000000.000000+1000000.000000=2000000.000000) / / V1[9999999
]+V2[9999999]=V3[9999999](1999999.900000+0.100000=2000000.000000) /
0.05user 0.04system 0:00.09elapsed 96%CPU (0avgtext+0avgdata 240252maxresident)k
24inputs+0outputs (0major+835minor)pagefaults 0swaps
```

Respuesta a la pregunta : El tiempo real es igual que el tiempo de CPU .

Si el programa tuviese alguna E/S , el tiempo real sería mayor al tiempo de CPU.

Si el programa tuviese distintos hilos , el tiempo de CPU podría ser mayor al tiempo real (podría ejecutar a la vez distintos flujos de instrucciones)

6. Generar el código ensamblador a partir del programa secuencial C del Listado 1 para **vectores globales** (para generar el código ensamblador tiene que compilar usando `-S` en lugar de `-o`). Utilice el fichero con el código fuente ensamblador generado y el fichero ejecutable generado en el ejercicio 5 para obtener para atcgrid los MIPS (*Millions of Instructions Per Second*) y los MFLOPS (*Millions of FLOating-point Per Second*) del código que obtiene la suma de vectores (código entre las funciones `clock_gettime()`); el cálculo se debe hacer para 10 y 10000000 componentes en los vectores (consulte la Lección 3/Tema1 AC). Razonar cómo se han obtenido los valores que se necesitan para calcular los MIPS y MFLOPS. Incorpore **el código ensamblador de la parte de la suma de vectores** en el cuaderno.

(nota mia) : El codigo que ejecuta la suma está entre dos call _clockgettime

(para MFLOPS tengo que contar aquellas instrucciones que operan en coma flotante (son aquella que llevan como operando %xmm_)

CAPTURAS DE PANTALLA (que muestren la generación del código ensamblador y del código ejecutable, y la obtención de los tiempos de ejecución):

```
[AlejandroMolinaCriado a2estudiante13@alexmolinaatcgrid:cx626QD~/bp1/ejercicio6] 2020-03-25 miércoles
$gcc -O2 -S sumavectores.c
[AlejandroMolinaCriado a2estudiante13@alexmolinaatcgrid:cx626QD~/bp1/ejercicio6] 2020-03-25 miércoles
$gcc  -O2  sumavectores.c -o SumaVectores -lrt
[AlejandroMolinaCriado a2estudiante13@alexmolinaatcgrid:cx626QD~/bp1/ejercicio6] 2020-03-25 miércoles
$
```

RESPUESTA: cálculo de los MIPS y los MFLOPS (de forma empírica)

T = 10	T = 10000000
MIPS = NI / Tcpu*10^6 = (6*10 + 3) / 0.000376837 * 10^6 MFLOPS = NopComaFlotante / Tcpu*10^6 = 30 / 0.000376837 * 10^6	MIPS = (6*10000000+3)/(0.041640940*10^6) MFLOPS = 30000000 / (0.041640940 * 10^6)

RESPUESTA: Captura que muesre el código ensamblador generado de la parte de la suma de vectores

```
        call    clock_gettime
        xorl    %eax, %eax
        .p2align 4,,10
        .p2align 3
.L6:
        movsd   v1(,%rax,8), %xmm0
        addsd   v2(,%rax,8), %xmm0
        movsd   %xmm0, v3(,%rax,8)
        addq    $1, %rax
        cmpl    %eax, %ebp
        ja      .L6
```

7. Implementar un programa en C con OpenMP, a partir del código del Listado 1, que calcule en paralelo la suma de dos vectores (v3 = v1 + v2; v3(i)=v1(i)+v2(i), i=0,…N-1) usando las directivas `parallel` y `for`. Se debe paralelizar también las tareas asociadas a la inicialización de los vectores. Como en el código del Listado 1 se debe obtener el tiempo (*elapsed time*) que supone el cálculo de la suma. Para obtener este tiempo usar la función `omp_get_wtime()`, que proporciona el estándar OpenMP, en lugar de `clock_gettime()`. NOTAS: (1) el número de componentes N de los vectores debe ser un argumento de entrada al programa; (2) se deben inicializar los vectores antes del cálculo; (3) se debe asegurar que el programa calcula la suma correctamente imprimiendo todos los componentes del vector resultante, v3, para varios tamaños pequeños de los vectores (por ejemplo, N = 8 y N=11); (5) se debe imprimir sea cual sea el tamaño de los vectores el tiempo de ejecución del código paralelo que suma los vectores y, al menos, el primer y último componente de v1, v2 y v3 (esto último evita que las optimizaciones del compilador eliminen el código de la suma).

RESPUESTA: Captura que muestre el código fuente implementado

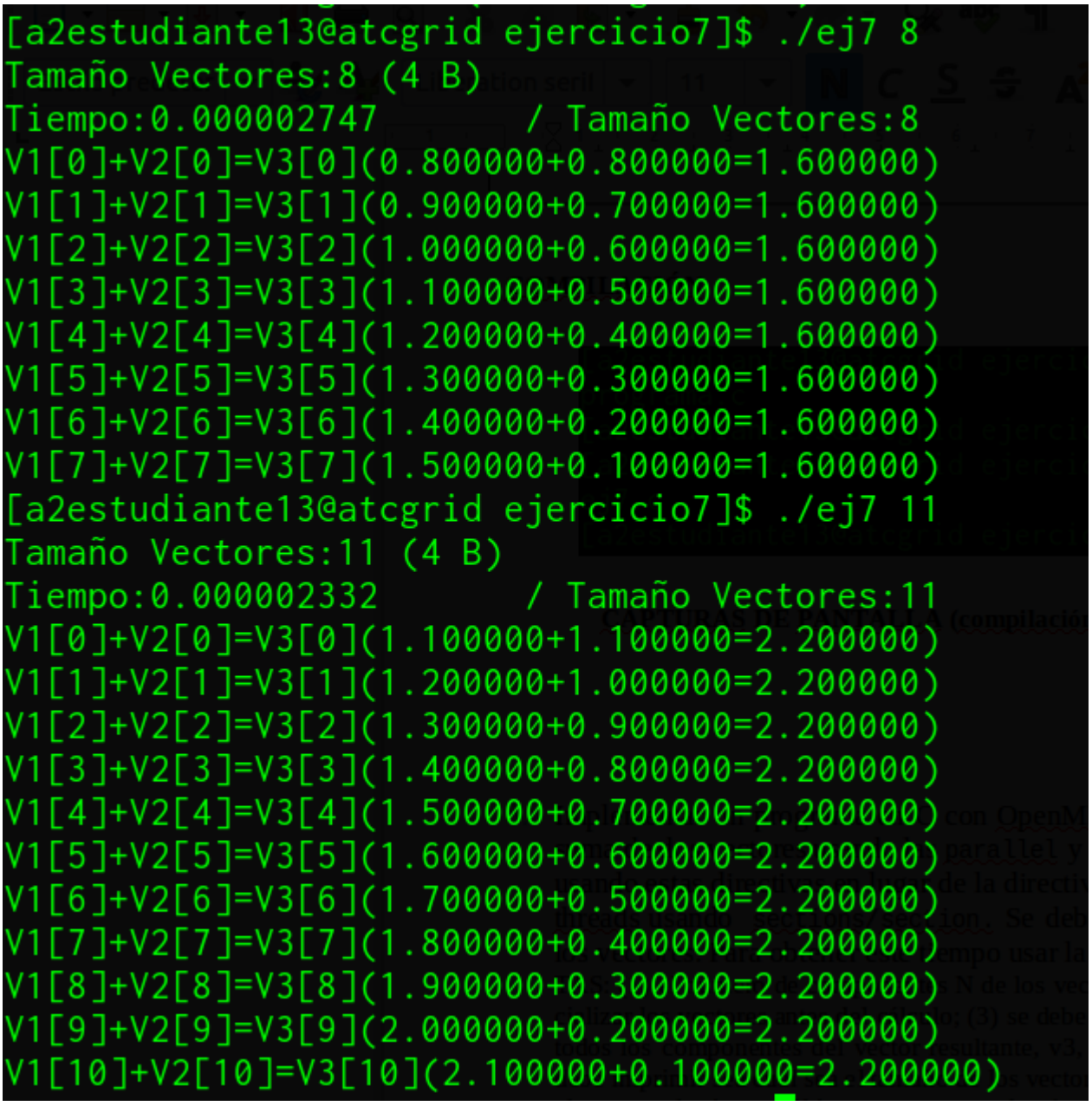
ejercicio7 > C programa.c > main(int, char **)

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

int main(int nargs, char** argv){

int i;
double ncgt,cgt1;
unsigned int N = atoi(argv[1]);
double v1[N], v2[N], v3[N];
printf("Tamaño Vectores:%u (%lu B)\n",N, sizeof(unsigned int));

#pragma omp parallel for
for(int i = 0; i<N; i++){
    v1[i] = N*0.1+i*0.1;
    v2[i] = N*0.1-i*0.1;
}

cgt1 = omp_get_wtime();

#pragma omp parallel for
for(i=0; i<N; i++)
    v3[i] = v1[i] + v2[i];

ncgt = omp_get_wtime() - cgt1;

 if (N<12) {
 printf("Tiempo:%11.9f\t / Tamaño Vectores:%u\n",ncgt,N);
 for(i=0; i<N; i++)
   printf("V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f)\n", i,i,i,v1[i],v2[i],v3[i]);
 }
 else
   printf("Tiempo:%11.9f\t / Tamaño Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f=%8.6f) / / V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
   ncgt,N,v1[0],v2[0],v3[0],N-1,N-1,N-1,v1[N-1],v2[N-1],v3[N-1]);
}
```

COMPILACIÓN

```
[a2estudiante13@atcgrid ejercicio7]$ ls
programa.c
[a2estudiante13@atcgrid ejercicio7]$ mv programa.c ej7.c
[a2estudiante13@atcgrid ejercicio7]$ ls
ej7.c
[a2estudiante13@atcgrid ejercicio7]$ gcc -O2 -fopenmp -o ej7 ej7.c
```

CAPTURAS DE PANTALLA (compilación y ejecución para N=8 y N=11):



8. Implementar un programa en C con OpenMP, a partir del código del Listado 1, que calcule en paralelo la suma de dos vectores usando las `parallel` y `sections/section` (se debe aprovechar el paralelismo de datos usando estas directivas en lugar de la directiva `for`); es decir, hay que repartir el trabajo (tareas) entre varios threads usando `sections/section`. Se debe paralelizar también las tareas asociadas a la inicialización de los vectores. Para obtener este tiempo usar la función `omp_get_wtime()` en lugar de `clock_gettime()`. NOTAS: (1) el número de componentes N de los vectores debe ser un argumento de entrada al programa; (2) se deben inicializar los vectores antes del cálculo; (3) se debe asegurar que el programa calcula la suma correctamente imprimiendo todos los componentes del vector resultante, v3, para tamaños pequeños de los vectores (por ejemplo, N = 8); (5) se debe imprimir sea cual sea el tamaño de los vectores el tiempo de ejecución del código paralelo que suma los vectores y, al menos, el primer y último componente de v1, v2 y v3 (esto último evita que las optimizaciones del compilador eliminen el código de la suma).

RESPUESTA: Captura que muestre el código fuente implementado

ejercicio8 > C ej8.c > main(int, char **)

```c
int main(int nargs, char** argv)
{
int i;
double ncgt,cgt1;
unsigned int N = atoi(argv[1]);
double v1[N], v2[N], v3[N];
printf("Tamaño Vectores:%u (%lu B)\n",N, sizeof(unsigned int));

#pragma omp parallel sections private(i)
{
    #pragma omp section
    for (i=0 ; i<1/4*N; i++){
        v1[i] = N*0.1+i*0.1;
        v2[i] = N*0.1-i*0.1;
    }
    #pragma omp section
    for (i=1/4*N ; i<2/4*N; i++){
        v1[i] = N*0.1+i*0.1;
        v2[i] = N*0.1-i*0.1;
    }
    #pragma omp section
    for (i=2/4*N ; i<3/4*N; i++){
        v1[i] = N*0.1+i*0.1;
        v2[i] = N*0.1-i*0.1;
    }
    #pragma omp section
    for (i=3/4*N ; i<N; i++){
        v1[i] = N*0.1+i*0.1;
        v2[i] = N*0.1-i*0.1;
    }
}

cgt1=omp_get_wtime();
#pragma omp parallel sections private(i)
{
    #pragma omp section
    for (i=0 ; i<1/4*N; i++)
        v3[i] = v1[i] + v2[i];

    #pragma omp section
    for (i=1/4*N ; i<2/4*N; i++)
        v3[i] = v1[i] + v2[i];

    #pragma omp section
    for (i=2/4*N ; i<3/4*N; i++)
        v3[i] = v1[i] + v2[i];

    #pragma omp section
    for (i=3/4*N ; i<N; i++)
        v3[i] = v1[i] + v2[i];
}

ncgt=omp_get_wtime() - cgt1;

//Imprimir resultados
  if (N<12) {
  printf("Tiempo:%11.9f\t / Tamaño Vectores:%u\n",ncgt,N);
  for(i=0; i<N; i++)
    printf("/ V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
           i,i,i,v1[i],v2[i],v3[i]);
  }
  else
    printf("Tiempo:%11.9f\t / Tamaño Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f=%8.6f) / / V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n"
    ,ncgt,N,v1[0],v2[0],v3[0],N-1,N-1,N-1,v1[N-1],v2[N-1],v3[N-1]);


return 0;
}
```

(RECUERDE ADJUNTAR CÓDIGO FUENTE AL .ZIP)

CAPTURAS DE PANTALLA (compilación y ejecución para N=8 y N=11):

```
[a2estudiante13@atcgrid ejercicio8]$ gcc -O2 -fopenmp -o ej8 ej8.c
[a2estudiante13@atcgrid ejercicio8]$ ./ej8 8
Tamaño Vectores:8 (4 B)
Tiempo:0.000002639       / Tamaño Vectores:8
/ V1[0]+V2[0]=V3[0](0.800000+0.800000=1.600000) /
/ V1[1]+V2[1]=V3[1](0.900000+0.700000=1.600000) /
/ V1[2]+V2[2]=V3[2](1.000000+0.600000=1.600000) /
/ V1[3]+V2[3]=V3[3](1.100000+0.500000=1.600000) /
/ V1[4]+V2[4]=V3[4](1.200000+0.400000=1.600000) /
/ V1[5]+V2[5]=V3[5](1.300000+0.300000=1.600000) /
/ V1[6]+V2[6]=V3[6](1.400000+0.200000=1.600000) /
/ V1[7]+V2[7]=V3[7](1.500000+0.100000=1.600000) /
[a2estudiante13@atcgrid ejercicio8]$ ./ej8 11
Tamaño Vectores:11 (4 B)
Tiempo:0.000002572       / Tamaño Vectores:11
/ V1[0]+V2[0]=V3[0](1.100000+1.100000=2.200000) /
/ V1[1]+V2[1]=V3[1](1.200000+1.000000=2.200000) /
/ V1[2]+V2[2]=V3[2](1.300000+0.900000=2.200000) /
/ V1[3]+V2[3]=V3[3](1.400000+0.800000=2.200000) /
/ V1[4]+V2[4]=V3[4](1.500000+0.700000=2.200000) /
/ V1[5]+V2[5]=V3[5](1.600000+0.600000=2.200000) /
/ V1[6]+V2[6]=V3[6](1.700000+0.500000=2.200000) /
/ V1[7]+V2[7]=V3[7](1.800000+0.400000=2.200000) /
/ V1[8]+V2[8]=V3[8](1.900000+0.300000=2.200000) /
/ V1[9]+V2[9]=V3[9](2.000000+0.200000=2.200000) /
/ V1[10]+V2[10]=V3[10](2.100000+0.100000=2.200000) /
```

9. ¿Cuántos threads y cuántos cores como máximo podría utilizar la versión que ha implementado en el ejercicio 7? Razone su respuesta. ¿Cuántos threads y cuantos cores como máximo podría utilizar la versión que ha implementado en el ejercicio 8? Razone su respuesta.

   RESPUESTA:

   Podemos tener mas hebras que iteraciones en el ejercicio 7 , pero eso significaría un desperdicio de tiempo en crear y destruirlas lo lógico es tener tantas hebras como iteraciones

   Igual que en el ejercicio 7 , en el 8 lo mejor es tener tantas hebras como tareas-

10. Rellenar una tabla como la Tabla 2Error: no se encontró el origen de la referencia para atcgrid y otra para su PC con los tiempos de ejecución de los programas paralelos implementados en los ejercicios 7 y 8 y el programa secuencial del Listado 1. Generar los ejecutables usando -O2. En la tabla debe aparecer el tiempo de ejecución del trozo de código que realiza la suma en paralelo (este es el tiempo que deben imprimir los programas). Ponga en la tabla el número de threads/cores que usan los códigos (use el máximo número de cores físicos del computador que como máximo puede aprovechar el código, no use un número de threads superior al número de cores físicos). Represente en una gráfica los tres tiempos. NOTA: Nunca ejecute código que imprima todos los componentes del resultado cuando este número sea elevado.

RESPUESTA:

TABLA 1

Nº de Componentes	**T. secuencial vect. Globales 1 thread/core**	**T. paralelo (versión** for**)** ¿?threads/cores	**T. paralelo (versión** sections**)** ¿?threads/cores
16384	0,000137205	0.005286152	0,000829035
32768	0,000259559	0,006310421	0,004891549
65536	0,000925751	0,000141042	0,002892876
131072	0,00112932	0,002332412	0,00053659
262144	0,003282788	0,006571844	0,000852717
524288	0,005828421	0,007645742	0,005234347
1048576	0,01273564	0,017386637	0,011130889
2097152	0,014186846	0,01683644	0,013378558
4194304	0,029876169	0,021915858	0,017278152
8388608	0,06150479	0,033372433	0,025644984
16777216	0,127521518	0,074292028	0,060120946
33554432	0,250292431	0,137889103	0,11049867
67108864	0,62745358	0,330815533	0,453886542

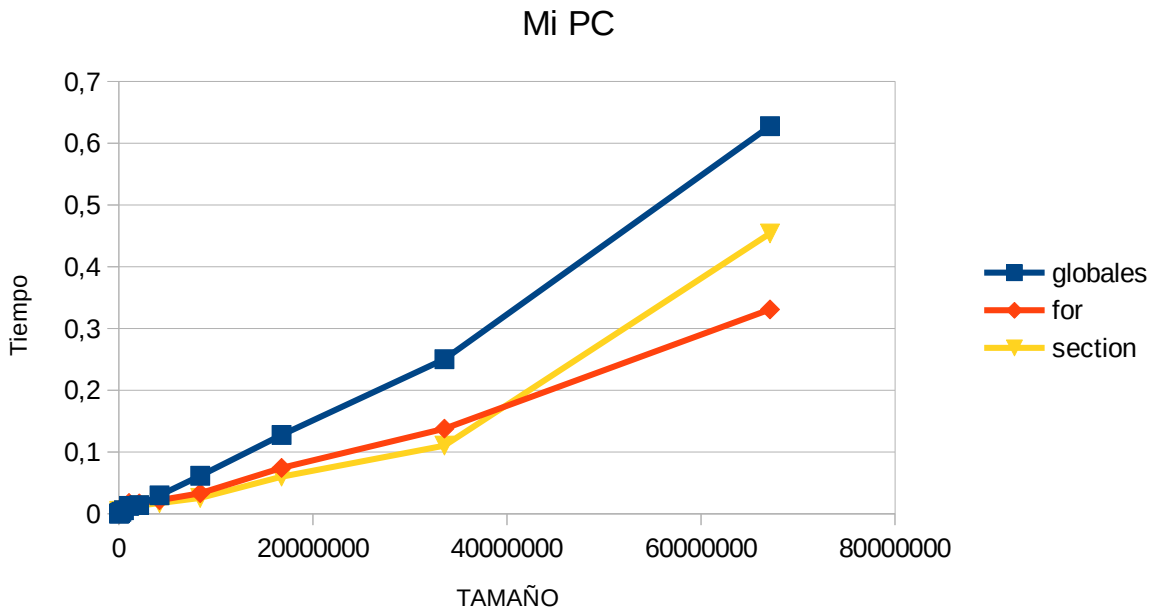
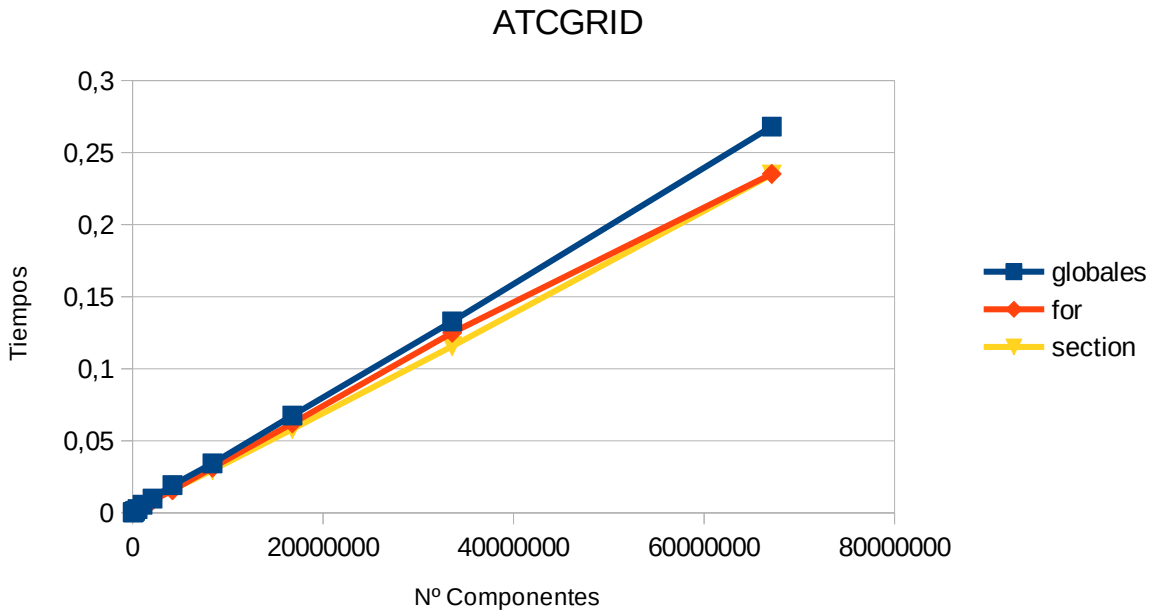


TABLA 2

Nº de Componentes	T. secuencial vect. Globales 1 thread/core	T. paralelo (versión for) ¿?threads/cores	T. paralelo (versión sections) ¿?threads/cores
16384	0,000442134	0.000669451	6,5755E-05
32768	0,000287011	0,000535183	0,000125743
65536	0,000355886	0,000422593	0,000255639
131072	0,000510913	0,00074994	0,000580113
262144	0,001350195	0,001002993	0,001066139
524288	0,002637612	0,002109515	0,001802385
1048576	0,005649643	0,004318478	0,004331317
2097152	0,009913711	0,008470235	0,008763473
4194304	0,019272389	0,0155924	0,016032891
8388608	0,03434113	0,031358868	0,029711843
16777216	0,067535624	0,062078897	0,058121467
33554432	0,132854233	0,124944916	0,115629446
67108864	0,268012789	0,235166177	0,23471123



11. Rellenar una tabla como la tabla 3 para atcgrid con el tiempo de ejecución, tiempo de CPU del usuario y tiempo CPU del sistema obtenidos con `time` para el ejecutable del ejercicio 7 y para el programa secuencial del Listado 1. Ponga en la tabla el número de threads/cores que usan los códigos. ¿El tiempo de CPU que se obtiene es mayor o igual que el tiempo real (*elapsed*)? Justifique la respuesta.

RESPUESTA:

Nº de Componentes	Tiempo secuencial vect. Globales 1 thread/core			Tiempo paralelo/versión for ¿? Threads/cores		
	Elapsed	CPU-user	CPU- sys	Elapsed	CPU-user	CPU- sys
65536	0m0.004s	0m0.000s	0m0.003s	0m0.004s	0m0.000s	0m0.005s
131072	0m0.004s	0m0.000s	0m0.004s	0m0.005s	0m0.005s	0m0.003s
262144	0m0.005s	0m0.002s	0m0.003s	0m0.008s	0m0.007s	0m0.006s
524288	0m0.010s	0m0.004s	0m0.005s	0m0.012s	0m0.011s	0m0.012s
1048576	0m0.014s	0m0.007s	0m0.007s	0m0.021s	0m0.021s	0m0.019s
2097152	0m0.028s	0m0.014s	0m0.014s	0m0.036s	0m0.040s	0m0.031s
4194304	0m0.044s	0m0.023s	0m0.021s	0m0.069s	0m0.094s	0m0.042s
8388608	0m0.090s	0m0.050s	0m0.040s	0m0.070s	0m0.089s	0m0.047s
16777216	0m0.158s	0m0.089s	0m0.068s	0m0.134s	0m0.166s	0m0.099s
33554432	0m0.313s	0m0.159s	0m0.154s	0m0.266s	0m0.341s	0m0.186s
67108864	0m0.619s	0m0.311s	0m0.307s	0m0.528s	0m0.684s	0m0.364s