

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): Alberto Llamas González

Grupo de prácticas y profesor de prácticas: D3, Juan Carlos Gómez López

Fecha de entrega: 18 abril 2021

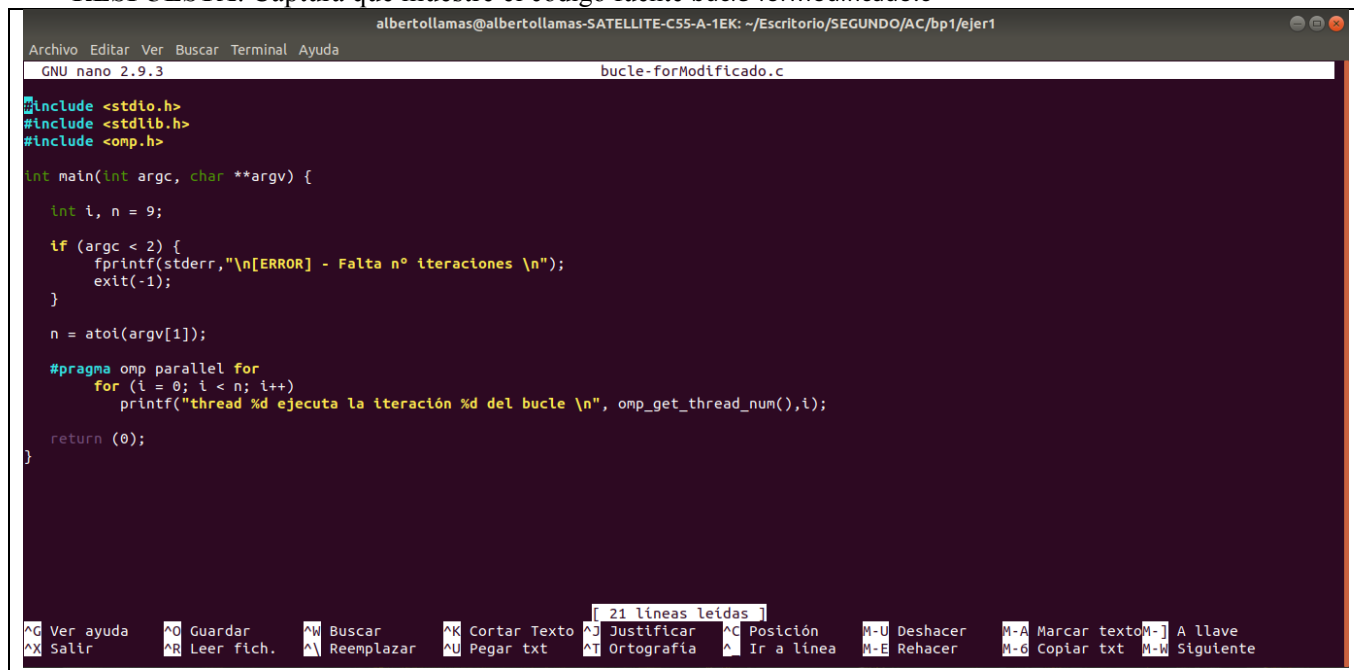
Fecha evaluación en clase: 19 abril 2021

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 bucle-forModificado.c



```
albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/bp1/ejer1
GNU nano 2.9.3 bucle-forModificado.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 \n");
        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

```

albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/bp1
GNU nano 2.9.3 sectionsModificado.c

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

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

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

int main(){
    #pragma omp parallel sections
    {
        #pragma omp section
        (void) funcA();
        #pragma omp section
        (void) funcB();
    }
}

```

- 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`

```

albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/bp1
GNU nano 2.9.3 singleModificado.c

#include <stdio.h>
#include <omp.h>
int main(){
    int n = 9, i, a, b[n];
    for (i = 0; i < n; i++) b[i] = -1;

    #pragma omp parallel
    {
        #pragma omp single
        {
            printf("Introduce valor de inicialización a: ");
            scanf("%d", &a);
            printf("Single ejecutada por el thread %d\n", omp_get_thread_num());
        }

        #pragma omp for
        for (i = 0; i < n; i++) b[i] = a;

        #pragma omp single
        {
            printf("Segundo single dentro de la región parallel:\n");
            for (i = 0; i < n; i++)
                printf("b[%d] = %d\t", i, b[i]);
            printf("\n");
            printf("Single ejecutada por el thread %d\n", omp_get_thread_num());
        }
    }
}

```

CAPTURAS DE PANTALLA:

```

d3estudiante26@atcgrid:~/bp1/ejer2
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer2] 2021-04-12 lunes
$./singleModificado
Introduce valor de inicialización a: 11
Single ejecutada por el thread 0
Segundo single dentro de la región parallel:
b[0] = 11      b[1] = 11      b[2] = 11      b[3] = 11      b[4] = 11      b[5] = 11      b[6] = 11      b[7] = 11      b[8] = 11
Single ejecutada por el thread 4
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer2] 2021-04-12 lunes
$./singleModificado
Introduce valor de inicialización a: 19
Single ejecutada por el thread 0
Segundo single dentro de la región parallel:
b[0] = 19      b[1] = 19      b[2] = 19      b[3] = 19      b[4] = 19      b[5] = 19      b[6] = 19      b[7] = 19      b[8] = 19
Single ejecutada por el thread 6
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer2] 2021-04-12 lunes
$./singleModificado
Introduce valor de inicialización a: 100
Single ejecutada por el thread 7
Segundo single dentro de la región parallel:
b[0] = 100     b[1] = 100     b[2] = 100     b[3] = 100     b[4] = 100     b[5] = 100     b[6] = 100     b[7] = 100     b[8] = 100
Single ejecutada por el thread 3
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer2] 2021-04-12 lunes
$

```

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

```

albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/bp1
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.9.3 singleModificado2.c
#include <stdio.h>
#include <omp.h>
int main()
{
    int n = 9, i, a, b[n];
    for (i = 0; i < n; i++) b[i] = -1;

    #pragma omp parallel
    {
        #pragma omp single
        {
            printf("Introduce valor de inicialización a: ");
            scanf("%d", &a);
            printf("Single ejecutada por el thread %d\n", omp_get_thread_num());
        }

        #pragma omp for
        for (i = 0; i < n; i++) b[i] = a;

        #pragma omp master
        {
            printf("Master dentro de la región parallel:\n");
            for (i = 0; i < n; i++)
                printf("b[%d] = %d\t", i, b[i]);
            printf("\n");
            printf("Master ejecutada por el thread %d\n", omp_get_thread_num());
        }
    }
}
[ 28 líneas leídas ]
^G Ver ayuda  ^O Guardar  ^W Buscar  ^K Cortar Texto  ^J Justificar  ^C Posición  M-U Deshacer  M-A Marcar texto  M-] A llave
^X Salir      ^R Leer fich.  ^E Reemplazar  ^U Pegar txt  ^I Ortografía  ^_ Ir a línea  M-E Rehacer  M-G Copiar txt  M-W Siguiente

```

CAPTURAS DE PANTALLA:

```

d3estudiante26@atcgrid:~/bp1/ejer3
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp1/ejer3] 2021-04-12 lunes
$ ./singleModificado2
Introduce valor de inicialización a: 13
Single ejecutada por el thread 5
Master dentro de la región parallel:
b[0] = 13    b[1] = 13    b[2] = 13    b[3] = 13    b[4] = 13    b[5] = 13    b[6] = 13    b[7] = 13    b[8] = 13
Master ejecutada por el thread 0
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp1/ejer3] 2021-04-12 lunes
$ ./singleModificado2
Introduce valor de inicialización a: 15
Single ejecutada por el thread 6
Master dentro de la región parallel:
b[0] = 15    b[1] = 15    b[2] = 15    b[3] = 15    b[4] = 15    b[5] = 15    b[6] = 15    b[7] = 15    b[8] = 15
Master ejecutada por el thread 0
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp1/ejer3] 2021-04-12 lunes
$ ./singleModificado2
Introduce valor de inicialización a: 100
Single ejecutada por el thread 7
Master dentro de la región parallel:
b[0] = 100   b[1] = 100   b[2] = 100   b[3] = 100   b[4] = 100   b[5] = 100   b[6] = 100   b[7] = 100   b[8] = 100
Master ejecutada por el thread 0
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp1/ejer3] 2021-04-12 lunes
$

```

RESPUESTA A LA PREGUNTA: La principal diferencia es que todas las instrucciones contenidas en la directiva "master" se ejecutan siempre por el thread maestro, es decir, el thread que corresponde al identificador 0. Para comprobar esto, podemos ver que tras varias ejecuciones el identificador que se imprime tras el resultado es siempre 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: Esto ocurre porque tras "atomic" no hay barrera implícita. Esto, en consecuencia, implica que es posible ejecutar el "printf" antes de que se hayan acumulado todas las sumas, por lo que el resultado sería incorrecto (uno de los hilos, tan pronto como no esté ocupado, ejecutará dicho "printf"). La barrera impide esto: espera a que todos los hilos hayan realizado la instrucción de adición acumulativa sobre "suma", y a continuación se ejecuta el "printf", teniendo en consecuencia un resultado correcto.

1.1.1

Resto de ejercicios (usar en atcgrid la cola ac a no ser que se tenga que usar atcgrid4)

5. El programa secuencial C del Listado 1 calcula la suma de dos vectores ($v3 = v1 + v2$; $v3(i) = v1(i) + v2(i)$, $i=0, \dots, N-1$). 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:

```

albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/bp1/ejer5
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.9.3 Listado1.c

double v1[N], v2[N], v3[N]; // Tamaño variable local en tiempo de ejecución ...
                             // disponible en C a partir de C99
#ifdef VECTOR_GLOBAL
if (N>MAX) N=MAX;
#endif
#ifdef VECTOR_DYNAMIC
double *v1, *v2, *v3;
v1 = (double*) malloc(N*sizeof(double)); // malloc necesita el tamaño en bytes
v2 = (double*) malloc(N*sizeof(double));
v3 = (double*) malloc(N*sizeof(double));
if ((v1 == NULL) || (v2 == NULL) || (v3 == NULL)) {
    printf("No hay suficiente espacio para los vectores \n");
    exit(-2);
}
#endif

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

double inicio = omp_get_wtime();

#pragma omp parallel for
//Calcular suma de vectores
for(i=0; i<N; i++)
    v3[i] = v1[i] + v2[i];

ncgt= omp_get_wtime() - inicio;

//Imprimir resultado de la suma y el tiempo de ejecución
if (N<10) {
    printf("Tiempo(seg.):%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,v1[i],v2[i],v3[i]);
}
else
    printf("Tiempo(seg.):%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]);

```

```

d3estudiante26@atcgrid:~
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlomasGonzalez d3estudiante26@atcgrid:~] 2021-04-07 miércoles
$time bp1/ejer5/Listado1 10000000
Tiempo:0.031498677      / Tamaño Vectores:10000000      / V1[0]+V2[0]=V3[0](0.60
7726+1.922949=2.530675) / / V1[9999999]+V2[9999999]=V3[9999999](546.735109+0.920
654=547.655763) /

real    0m0.304s
user    0m0.240s
sys      0m0.063s
[AlbertoLlomasGonzalez d3estudiante26@atcgrid:~] 2021-04-07 miércoles
$

```

RESPUESTA: La suma del tiempo de CPU de usuario y del tiempo del sistema va a ser menor o igual al tiempo real. En mi caso, $0,063 + 0,240 = 0,303 < 0,304$. Esto se debe a que el tiempo real es una aproximación del tiempo que puede tardar el programa en ejecutarse. El tiempo de usuario es el tiempo que tarda el usuario en recuperar el control de la terminal tras ejecutar el programa y el tiempo del sistema es el tiempo de espera debido a operaciones de E/S, por ejemplo, es decir, operaciones que utilizan recursos del sistema.

- 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. Incorporar **el código ensamblador de la parte de la suma de vectores** (no de todo el programa) en el cuaderno.

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

```
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer6] 2021-04-15 jueves
$gcc -S -O2 Listado1.c -o Listado1.s -lrt
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer6] 2021-04-15 jueves
$ls
Listado1.c Listado1.s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer6] 2021-04-15 jueves
$gcc -O2 Listado1.c -o Listado1 -lrt
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer6] 2021-04-15 jueves
$ls
Listado1 Listado1.c Listado1.s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer6] 2021-04-15 jueves
$
```

Cálculo para 10 y 10000000 componentes

```
Archivo Editar Ver Buscar Terminal Ayuda
d3estudiante26@atcgrid:~/bp1/ejer6
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer6] 2021-04-15 jueves
$time srtn -pac ./Listado1 10000000
Tiempo:0.039321731 / Tamaño Vectores:10000000 / V1[0]+V2[0]=V3[0](1.249169+0.075493=1.324662) / / V1[9999999]+V2[9999999]=V3[9999999](1.071070+1.278516=2.349585) /
real 0m0.688s
user 0m0.006s
sys 0m0.009s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer6] 2021-04-15 jueves
$time srtn -pac ./Listado1 10
Tiempo:0.000391119 / Tamaño Vectores:10 / V1[0]+V2[0]=V3[0](0.270327+1.406922=1.677249) / / V1[9]+V2[9]=V3[9](3.241782+0.039467=3.281249) /
real 0m0.105s
user 0m0.005s
sys 0m0.010s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer6] 2021-04-15 jueves
$
```

RESPUESTA: cálculo de los MIPS y los MFLOPS

Para 10 componentes:

$$\text{MIPS} = \text{NI} / (\text{T}_{\text{cpu}} * 10^6) = (6 * 10) / (0,000391119 * 10^6) = 0,1534059966 \text{ MIPS}$$

$$\text{MFLOPS} = \text{NI}_{\text{float}} / (\text{T}_{\text{cpu}} * 10^6) = (1 * 10) / (0,000391119 * 10^6) = 0,02556766611 \text{ MFLOPS}$$

Para 10000000 componentes:

$$\text{MIPS} = \text{NI} / (\text{T}_{\text{cpu}} * 10^6) = (6 * 10000000) / (0,039321731 * 10^6) = 1525,8738227979 \text{ MIPS}$$

$$\text{MFLOPS} = \text{NI}_{\text{float}} / (\text{T}_{\text{cpu}} * 10^6) = (1 * 10000000) / (0,039321731 * 10^6) = 254,3123037996 \text{ MFLOPS}$$

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

```

Archivo Editar Ver Buscar Terminal Ayuda
d3estudiante26@atcgird:~/bp1/ejer6

.L6:
    ja     .L5
    movq   %rsp, %rsi
    xorl   %edi, %edi
    call   clock_gettime
    xorl   %eax, %eax
    .p2align 4,,10
    .p2align 3

.L8:
    movsd  v1(,%rax,8), %xmm0
    addsd  v2(,%rax,8), %xmm0
    movsd  %xmm0, v3(,%rax,8)
    addq   $1, %rax
    cmpl   %eax, %ebp
    ja     .L8
    leaq   16(%rsp), %rsi
    xorl   %edi, %edi
    call   clock_gettime
    movq   24(%rsp), %rax
    pxor   %xmm0, %xmm0
    subq   8(%rsp), %rax
    cvtsi2sdq %rax, %xmm0
    pxor   %xmm1, %xmm1
    movq   16(%rsp), %rax
    subq   (%rsp), %rax
    cvtsi2sdq %rax, %xmm1
    divsd  .LC5(%rip), %xmm0
    addsd  %xmm1, %xmm0
    cmpl   $9, %ebp
    jbe    .L25
    leal   -1(%rbp), %eax
    movl   %ebp, %esi
    movl   $.LC3, %edi
    movsd  v3(%rip), %xmm3
    movsd  v2(%rip), %xmm2

```

7. Implementar un programa en C con OpenMP, a partir del código del Listado 1, que calcule en paralelo la suma de dos vectores ($v3 = v1 + v2$; $v3(i) = v1(i) + v2(i)$, $i = 0, \dots, N-1$) 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 `sp-OpenMP-for.c`

```

C Listado1Ejer7.c X C Listado1Ejer8.c
ejer7 > C Listado1Ejer7.c > main(int, char **)
65 |     exit(-2);
66 | }
67 | #endif
68 |
69 | #pragma omp parallel for
70 | //Inicializar vectores
71 | for (i = 0; i < N; i++)
72 | {
73 |     v1[i] = N * 0.1 + i * 0.1;
74 |     v2[i] = N * 0.1 - i * 0.1;
75 | }
76 |
77 | double inicio = omp_get_wtime();
78 |
79 | #pragma omp parallel for
80 | //Calcular suma de vectores
81 | for (i = 0; i < N; i++)
82 |     v3[i] = v1[i] + v2[i];
83 |
84 | ncgt = omp_get_wtime() - inicio;
85 |
86 | //Imprimir resultado de la suma y el tiempo de ejecución
87 | if (N < 10)
88 | {
89 |     printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\n", ncgt, N);
90 |     for (i = 0; i < N; i++)
91 |         printf("/ V1[%d]+V2[%d]=V3[%d] (%8.6f+%8.6f=%8.6f) /\n",
92 |             i, i, i, v1[i], v2[i], v3[i]);
93 | }

```


(RECUERDE ADJUNTAR CÓDIGO FUENTE AL .ZIP)

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

```

albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/bp1/ejer7
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/bp1/ejer7] 2021-04-12 lunes
$gcc -fopenmp -O2 -o Listado1Ejer7 Listado1Ejer7.c
Listado1Ejer7.c: In function 'main':
Listado1Ejer7.c:168:18: warning: implicit declaration of function 'omp_get_wtime'; did you mean 'timer_gettime'? [-Wimplicit-function-declaration]
    double start = omp_get_wtime();
                   ^~~~~~
                   timer_gettime
[AlbertoLlamasGonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/bp1/ejer7] 2021-04-12 lunes
$clear

```

```

d3estudiante26@atcgrid:~/bp1/ejer7
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer7] 2021-04-12 lunes
$time srun -pac -Aac ./Listado1Ejer7 8
Tiempo(seg.):0.000000000 / 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) /

real    0m0.097s
user    0m0.009s
sys      0m0.006s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer7] 2021-04-12 lunes
$time srun -pac -Aac ./Listado1Ejer7 11
Tiempo(seg.):0.000000000 / Tamaño Vectores:11 / V1[0]+V2[0]=V3[0](1.100000+1.100000=2.200000) / / V1[10]+V2[10]=V3[10](2.100000+0.100000=2.200000) /

real    0m0.091s
user    0m0.005s
sys      0m0.010s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer7] 2021-04-12 lunes
$

```

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 `sp-OpenMP-sections.c`

The image shows two screenshots of a C code editor, likely Visual Studio Code, with a dark theme. The editor has two tabs open: 'Listado1Ejer7.c' and 'Listado1Ejer8.c'. The active tab is 'Listado1Ejer8.c', which contains C code using OpenMP for parallelizing vector operations. The code is written in a syntax-highlighted style with green and blue comments. The first screenshot shows lines 67 to 95, defining four parallel sections for calculating vector elements v1[i] and v2[i]. The second screenshot shows lines 96 to 124, which calculate the sum of vectors v1 and v2 into v3, and then print the execution time and the first few elements of the vectors. The status bar at the bottom of both screenshots indicates 'Ln 97, Col 18', 'Tab Size: 4', 'UTF-8', 'LF', 'C', 'Linux', and icons for search, run, and other editor functions.

```

67 #endif
68 #pragma omp parallel sections
69 {
70     //Inicializar vectores
71     #pragma omp section
72     for (i = 0; i < N / 4; i++)
73     {
74         v1[i] = N * 0.1 + i * 0.1;
75         v2[i] = N * 0.1 - i * 0.1; //los valores dependen de N
76     }
77     #pragma omp section
78     for (i = N / 4; i < N / 2; i++)
79     {
80         v1[i] = N * 0.1 + i * 0.1;
81         v2[i] = N * 0.1 - i * 0.1; //los valores dependen de N
82     }
83     #pragma omp section
84     for (i = N / 2; i < 3 * N / 4; i++)
85     {
86         v1[i] = N * 0.1 + i * 0.1;
87         v2[i] = N * 0.1 - i * 0.1; //los valores dependen de N
88     }
89     #pragma omp section
90     for (i = 3 * N / 4; i < N; i++)
91     {
92         v1[i] = N * 0.1 + i * 0.1;
93         v2[i] = N * 0.1 - i * 0.1; //los valores dependen de N
94     }
95 }

96
97 double inicio = omp_get_wtime();
98
99 #pragma omp parallel sections
100 {
101     //Calcular suma de vectores
102     #pragma omp section
103     for (i = 0; i < N / 4; i++)
104         v3[i] = v1[i] + v2[i];
105     #pragma omp section
106     for (i = N / 4; i < N / 2; i++)
107         v3[i] = v1[i] + v2[i];
108     #pragma omp section
109     for (i = N / 2; i < 3 * N / 2; i++)
110         v3[i] = v1[i] + v2[i];
111     #pragma omp section
112     for (i = 3 * N / 2; i < N; i++)
113         v3[i] = v1[i] + v2[i];
114 }
115
116 ncgt = omp_get_wtime() - inicio;
117 //Imprimir resultado de la suma y el tiempo de ejecución
118 if (N < 10)
119 {
120     printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\n", ncgt, N);
121     for (i = 0; i < N; i++)
122         printf(" / V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
123             i, i, i, v1[i], v2[i], v3[i]);
124 }

```

(RECUERDE ADJUNTAR CÓDIGO FUENTE AL .ZIP)

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

```

d3estudiante26@atcgrid:~/bp1/ejer8
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer8] 2021-04-12 lunes
$time srun -pac -Aac ./Listado1Ejer8 8
Tiempo(seg.):0.000674214 / 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) /

real 0m0.106s
user 0m0.009s
sys 0m0.005s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer8] 2021-04-12 lunes
$time srun -pac -Aac ./Listado1Ejer8 11
Tiempo(seg.):0.000576563 / Tamaño Vectores:11 / V1[0]+V2[0]=V3[0](1.100000+1.100000=2.200000) / / V1[10]+V2[10]=V3[10](2.100000+0.100000=2.200000) /

real 0m0.094s
user 0m0.008s
sys 0m0.006s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer8] 2021-04-12 lunes
$

```

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 cuántos cores como máximo podría utilizar la versión que ha implementado en el ejercicio 8? Razone su respuesta. NOTA: Al contestar piense sólo en el código, no piense en el computador en el que lo va a ejecutar.

RESPUESTA: En el ejercicio 7, podríamos utilizar un máximo de N threads, sin embargo, este número se limita por el número de cores lógicos del ordenador en el que estemos trabajando, o de la constante definida por OpenMP. Por otra parte, el ejercicio 8 siempre usará un máximo de 4 threads, porque hemos dividido el “for” en cuatro secciones paralelas.

10. Rellenar una tabla como la Tabla 2; **Error! Marcador no definido.** 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. **Escribir un script para realizar las ejecuciones necesarias utilizando como base el script del seminario de BP0 (se deben imprimir en el script al menos las variables de entorno que ya se imprimen en el script de BP0).** 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. **Observar que el número de componentes en la tabla llega hasta 67108864.**

RESPUESTA: Captura del script implementado sp-OpenMP-script10.sh

```

albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/bp1/ejer11
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.9.3 sp-OpenMP-script10.sh

#!/bin/bash
#Autor: Alberto Llamas González
#Órdenes para el sistema de colas:
#1. Asigna al trabajo un nombre
#SBATCH --job-name=sp-OpenMP-script10
#2. Asignar el trabajo a una cola (partición)
#SBATCH --partition=ac
#2. Asignar el trabajo a un account
#SBATCH --account=ac
#Obtener información de las variables del entorno del sistema de colas:

echo "Id. usuario del trabajo: $SLURM_JOB_USER"
echo "Id. del trabajo: $SLURM_JOBID"
echo "Nombre del trabajo especificado por usuario: $SLURM_JOB_NAME"
echo "Directorio de trabajo (en el que se ejecuta el script): $SLURM_SUBMIT_DIR"
echo "Cola: $SLURM_JOB_PARTITION"
echo "Nodo que ejecuta este trabajo:$SLURM_SUBMIT_HOST"
echo "Nº de nodos asignados al trabajo: $SLURM_JOB_NUM_NODES"
echo "Nodos asignados al trabajo: $SLURM_JOB_NODELIST"
echo "CPUs por nodo: $SLURM_JOB_CPUS_PER_NODE"

for (( N= 16384 ; N <= 67108864 ; N=N*2 ))
do
    ./S1 $N
done

```

(RECUERDE ADJUNTAR LOS CÓDIGOS AL .ZIP)

CAPTURAS DE PANTALLA (mostrar la ejecución en atcgrid – envío(s) a la cola):

Compilación en atcgrid:

```

d3estudiante26@atcgrid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertollamasGonzalez d3estudiante26@atcgrid:~] 2021-04-16 viernes
$cd bp1/ejer10
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$ls
Listado1.c Listado1Ejer7.c Listado1Ejer8.c scrip-ejer10.sh
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$gcc -O2 -fopenmp Listado1.c -o Listado1 -lrt
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$gcc -O2 -fopenmp Listado1Ejer7.c -o Listado1Ejer7 -lrt
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$gcc -O2 -fopenmp Listado1Ejer8.c -o Listado1Ejer8 -lrt
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$ls -l
total 60
-rwxrwxr-x 1 d3estudiante26 d3estudiante26 8520 abr 16 00:11 Listado1
-rw-rw-r-- 1 d3estudiante26 d3estudiante26 3526 abr 16 00:10 Listado1.c
-rwxrwxr-x 1 d3estudiante26 d3estudiante26 8656 abr 16 00:12 Listado1Ejer7
-rw-rw-r-- 1 d3estudiante26 d3estudiante26 3409 abr 16 00:10 Listado1Ejer7.c
-rwxrwxr-x 1 d3estudiante26 d3estudiante26 12856 abr 16 00:12 Listado1Ejer8
-rwxr-xr-x 1 d3estudiante26 d3estudiante26 4165 abr 16 00:10 Listado1Ejer8.c
-rwxrwxr-x 1 d3estudiante26 d3estudiante26 895 abr 16 00:10 scrip-ejer10.sh
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$

```

Envío a la cola de ejecución:

```

d3estudiante26@atcgrid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$sbatch -n1 -pac -Aac -c12 --hint=nomultithread ./scrip-ejer10.sh Listado1
Submitted batch job 91836
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$sbatch -n1 -pac -Aac -c12 --hint=nomultithread ./scrip-ejer10.sh Listado1Ejer7
Submitted batch job 91839
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$sbatch -n1 -pac -Aac -c12 --hint=nomultithread ./scrip-ejer10.sh Listado1Ejer8
Submitted batch job 91840
[AlbertollamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$

```

Ejecución en atcgrid:

Listado1 (Secuencial):

```

d3estudiante26@atcgriid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
Directorio de trabajo (en el que se ejecuta el script): /home/d3estudiante26/bp1/ejer10
Cola: ac
Nodo que ejecuta este trabajo:atcgriid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgriid1
CPUs por nodo: 24
Tiempo:0.000434682 / Tamaño Vectores:16384 / V1[0]+V2[0]=V3[0](0.643976+0.963920=1.607896) / / V1[16383]+V2[16383]=V3[16383](0.295304+0.9
89207=1.284511) /
Tiempo:0.000467208 / Tamaño Vectores:32768 / V1[0]+V2[0]=V3[0](0.643976+0.963920=1.607896) / / V1[32767]+V2[32767]=V3[32767](0.088541+0.8
60534=0.949076) /
Tiempo:0.000356291 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.643976+0.963920=1.607896) / / V1[65535]+V2[65535]=V3[65535](3.726647+1.7
97219=5.523866) /
Tiempo:0.000499056 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.643976+0.963920=1.607896) / / V1[131071]+V2[131071]=V3[131071](0.139147+
0.420969=0.560116) /
Tiempo:0.001367306 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.643976+0.963920=1.607896) / / V1[262143]+V2[262143]=V3[262143](0.626483+
0.308373=0.934856) /
Tiempo:0.002516822 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.643976+0.963920=1.607896) / / V1[524287]+V2[524287]=V3[524287](0.808991+
34.729316=35.538307) /
Tiempo:0.004973934 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](3.512490+0.912919=4.425409) / / V1[1048575]+V2[1048575]=V3[1048575](0.1137
62+1.897123=2.010885) /
Tiempo:0.008847148 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](3.512490+0.912919=4.425409) / / V1[2097151]+V2[2097151]=V3[2097151](50.660
640+1.273937=51.934577) /
Tiempo:0.017095919 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](3.512490+0.912919=4.425409) / / V1[4194303]+V2[4194303]=V3[4194303](0.3545
09+0.726450=1.080959) /
Tiempo:0.032331859 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](3.512490+0.912919=4.425409) / / V1[8388607]+V2[8388607]=V3[8388607](0.9023
68+0.988769=1.891137) /
Tiempo:0.063942061 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](3.512490+0.912919=4.425409) / / V1[16777215]+V2[16777215]=V3[16777215](0.5
63470+0.727299=1.290769) /
Tiempo:0.126561664 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](1.553202+0.289424=1.842626) / / V1[33554431]+V2[33554431]=V3[33554431](0.8
74565+0.277005=1.151570) /
Tiempo:0.126731809 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.478010+0.900141=1.378151) / / V1[33554431]+V2[33554431]=V3[33554431](1.0
35951+1.928379=2.964330) /
[AlbertollamasGonzalez d3estudiante26@atcgriid:~/bp1/ejer10] 2021-04-16 viernes
$

```

Listado1Ejer7 (Parallel-for):

```

d3estudiante26@atcgriid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
Directorio de trabajo (en el que se ejecuta el script): /home/d3estudiante26/bp1/ejer10
Cola: ac
Nodo que ejecuta este trabajo:atcgriid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgriid1
CPUs por nodo: 24
Tiempo(seg.):0.005891461 / Tamaño Vectores:16384 / V1[0]+V2[0]=V3[0](1638.400000+1638.400000=3276.800000) / / V1[16383]+V2[16383]=V3[16
383](3276.700000+0.100000=3276.800000) /
Tiempo(seg.):0.006270841 / Tamaño Vectores:32768 / V1[0]+V2[0]=V3[0](3276.800000+3276.800000=6553.600000) / / V1[32767]+V2[32767]=V3[32
767](6553.500000+0.100000=6553.600000) /
Tiempo(seg.):0.004906844 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / / V1[65535]+V2[65535]=V3[6
5535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.006947633 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) / / V1[131071]+V2[131071]=
V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.006786712 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=
V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.007437937 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / / V1[524287]+V2[524287]=
V3[524287](104857.500000+0.100000=104857.600000) /
Tiempo(seg.):0.008533470 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) / / V1[1048575]+V2[1048
575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.010104209 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) / / V1[2097151]+V2[2097
151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.012020752 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) / / V1[4194303]+V2[4194
303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.018717587 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) / / V1[8388607]+V2[838
8607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo(seg.):0.032735396 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / / V1[16777215]+V2[
16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
Tiempo(seg.):0.059547167 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[
33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
Tiempo(seg.):0.060793776 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[
33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
[AlbertollamasGonzalez d3estudiante26@atcgriid:~/bp1/ejer10] 2021-04-16 viernes
$

```

Listado1Ejer8 (Parallel-sections):

```

d3estudiante26@atcgrid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
Id. del trabajo: 91840
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/d3estudiante26/bp1/ejer10
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24
Tiempo(seg.):0.004376493 / Tamaño Vectores:16384 / V1[0]+V2[0]=V3[0](1638.400000+1638.400000=3276.800000) / V1[16383]+V2[16383]=V3[16383](3276.700000+0.100000=3276.800000) /
Tiempo(seg.):0.004195131 / Tamaño Vectores:32768 / V1[0]+V2[0]=V3[0](3276.800000+3276.800000=6553.600000) / V1[32767]+V2[32767]=V3[32767](6553.500000+0.100000=6553.600000) /
Tiempo(seg.):0.004140746 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.004204031 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.002125777 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.005466901 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tiempo(seg.):0.008355312 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.011571925 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.025399286 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.035702657 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo(seg.):0.084965989 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
/var/spool/slurmd/job91840/slurm_script: línea 25: 12854 Violación de segmento ('core' generado) ./ $1 $N
/var/spool/slurmd/job91840/slurm_script: línea 25: 12888 Violación de segmento ('core' generado) ./ $1 $N
[AlbertoLlanasGonzalez d3estudiante26@atcgrid:~/bp1/ejer10] 2021-04-16 viernes
$

```

Ejecución en PC:

Listado1 (secuencial):

```

d3estudiante26@atcgrid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlanasGonzalez albertollanas@albertollanas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/bp1/ejer10] 2021-04-16 viernes
$./scrip-ejer10.sh Listado1
Id. usuario del trabajo:
Id. del trabajo:
Nombre del trabajo especificado por usuario:
Directorio de trabajo (en el que se ejecuta el script):
Cola:
Nodo que ejecuta este trabajo:
Nº de nodos asignados al trabajo:
Nodos asignados al trabajo:
CPUs por nodo:
Tiempo:0.000178382 / Tamaño Vectores:16384 / V1[0]+V2[0]=V3[0](0.064837+1.912136=1.976973) / V1[16383]+V2[16383]=V3[16383](0.927578+4.051384=4.978962) /
Tiempo:0.000215182 / Tamaño Vectores:32768 / V1[0]+V2[0]=V3[0](0.064837+1.912136=1.976973) / V1[32767]+V2[32767]=V3[32767](0.128127+0.032795=0.160922) /
Tiempo:0.000353213 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.064837+1.912136=1.976973) / V1[65535]+V2[65535]=V3[65535](2.043505+1.567675=3.611180) /
Tiempo:0.000650210 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.064837+1.912136=1.976973) / V1[131071]+V2[131071]=V3[131071](1.594049+3.467294=5.061343) /
Tiempo:0.001315974 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.064837+1.912136=1.976973) / V1[262143]+V2[262143]=V3[262143](2.278838+16.763015=19.041853) /
Tiempo:0.002718411 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.064837+1.912136=1.976973) / V1[524287]+V2[524287]=V3[524287](1.691176+1.100579=2.791755) /
Tiempo:0.005278967 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](0.064837+1.912136=1.976973) / V1[1048575]+V2[1048575]=V3[1048575](0.061201+2.488302=2.549503) /
Tiempo:0.010407676 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](0.064837+1.912136=1.976973) / V1[2097151]+V2[2097151]=V3[2097151](1.356908+0.421655=1.778564) /
Tiempo:0.021181350 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](0.064837+1.912136=1.976973) / V1[4194303]+V2[4194303]=V3[4194303](1.463780+0.364969=1.828749) /
Tiempo:0.043924646 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](2.037993+5.560920=7.598912) / V1[8388607]+V2[8388607]=V3[8388607](0.696489+0.311659=1.008148) /
Tiempo:0.087851159 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](2.037993+5.560920=7.598912) / V1[16777215]+V2[16777215]=V3[16777215](1.209909+1.322245=2.532153) /
Tiempo:0.173085675 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.175016+1.069770=2.044786) / V1[33554431]+V2[33554431]=V3[33554431](0.844915+1.223897=2.068812) /
Tiempo:0.170341434 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.657452+1.813088=2.470461) / V1[33554431]+V2[33554431]=V3[33554431](0.192610+0.590430=0.783039) /
[AlbertoLlanasGonzalez albertollanas@albertollanas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/bp1/ejer10] 2021-04-16 viernes
$

```

Listado1Ejer7 (Parallel-for):


```

d3estudiante26@atcgriid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
[albertollamasgonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/bp1/ejer10] 2021-04-16 viernes
S./scrip-ejer10.sh Listado1Ejer7
Id. usuario del trabajo:
Id. del trabajo:
Nombre del trabajo especificado por usuario:
Directorio de trabajo (en el que se ejecuta el script):
Cola:
Nodo que ejecuta este trabajo:
Nº de nodos asignados al trabajo:
Nodos asignados al trabajo:
CPUS por nodo:
Tiempo:0.000104315 / Tamaño Vectores:16384 / V1[0]+V2[0]=V3[0](0.092031+0.933669=1.025700) / / V1[16383]+V2[16383]=V3[16383](0.801926+0.945059=1.746985) /
Tiempo:0.000283921 / Tamaño Vectores:32768 / V1[0]+V2[0]=V3[0](0.092031+0.933669=1.025700) / / V1[32767]+V2[32767]=V3[32767](0.699208+0.578389=1.277596) /
Tiempo:0.000306561 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.092031+0.933669=1.025700) / / V1[65535]+V2[65535]=V3[65535](0.192448+0.540322=0.732762) /
Tiempo:0.000655123 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.092031+0.933669=1.025700) / / V1[131071]+V2[131071]=V3[131071](1.188549+0.566722=1.755271) /
Tiempo:0.001386243 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.092031+0.933669=1.025700) / / V1[262143]+V2[262143]=V3[262143](0.027290+0.897453=0.924743) /
Tiempo:0.002659000 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.092031+0.933669=1.025700) / / V1[524287]+V2[524287]=V3[524287](0.755703+1.048162=1.803865) /
Tiempo:0.005449120 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](0.092031+0.933669=1.025700) / / V1[1048575]+V2[1048575]=V3[1048575](0.415040+1.538204=1.953244) /
Tiempo:0.010705989 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](0.092031+0.933669=1.025700) / / V1[2097151]+V2[2097151]=V3[2097151](0.383693+0.881414=1.265107) /
Tiempo:0.021579533 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](0.465586+0.385001=0.850586) / / V1[4194303]+V2[4194303]=V3[4194303](0.237091+0.084965=0.322056) /
Tiempo:0.042370652 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](0.465586+0.385001=0.850586) / / V1[8388607]+V2[8388607]=V3[8388607](0.808536+0.136449=0.944985) /
Tiempo:0.085049523 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](0.980903+0.203076=1.183979) / / V1[16777215]+V2[16777215]=V3[16777215](0.719776+1.580552=2.300327) /
Tiempo:0.169654567 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.168510+1.758537=1.927047) / / V1[33554431]+V2[33554431]=V3[33554431](0.353369+0.697629=1.050999) /
Tiempo:0.170111649 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.748291+0.67854=679.426144) / / V1[33554431]+V2[33554431]=V3[33554431](1.457187+18.877970=20.335157) /
[albertollamasgonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/bp1/ejer10] 2021-04-16 viernes
$

```

Listado1Ejer8 (Parallel-sections):

```

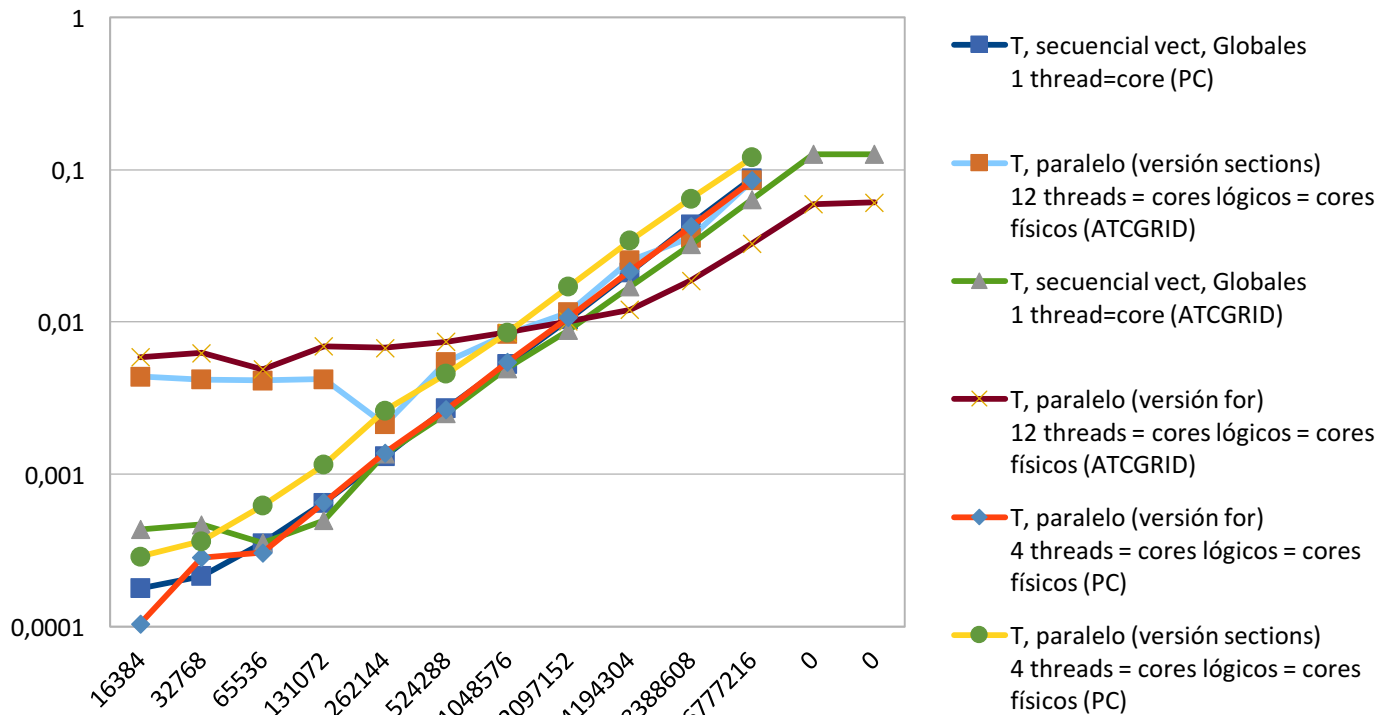
d3estudiante26@atcgriid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
[albertollamasgonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/bp1/ejer10] 2021-04-16 viernes
Sexport OMP_NUM_THREADS=4
[albertollamasgonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/bp1/ejer10] 2021-04-16 viernes
S./scrip-ejer10.sh Listado1Ejer8
Id. usuario del trabajo:
Id. del trabajo:
Nombre del trabajo especificado por usuario:
Directorio de trabajo (en el que se ejecuta el script):
Cola:
Nodo que ejecuta este trabajo:
Nº de nodos asignados al trabajo:
Nodos asignados al trabajo:
CPUS por nodo:
Tiempo(seg.):0.000280413 / Tamaño Vectores:16384 / V1[0]+V2[0]=V3[0](1638.400000+1638.400000=3276.800000) / / V1[16383]+V2[16383]=V3[16383](3276.700000+0.100000=3276.800000) /
Tiempo(seg.):0.000364683 / Tamaño Vectores:32768 / V1[0]+V2[0]=V3[0](3276.800000+3276.800000=6553.600000) / / V1[32767]+V2[32767]=V3[32767](6553.500000+0.100000=6553.600000) /
Tiempo(seg.):0.000621874 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200000) / / V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.001155135 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.400000) / / V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.002616417 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.800000) / / V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.004577958 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / / V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tiempo(seg.):0.008508678 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=209715.200000) / / V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.017055570 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=419430.400000) / / V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.034186747 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=838860.800000) / / V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.064451489 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo(seg.):0.120477282 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / / V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
./scrip-ejer10.sh: línea 25: 3670 Violación de segmento ('core' generado) ./S1 $N
./scrip-ejer10.sh: línea 25: 3675 Violación de segmento ('core' generado) ./S1 $N
[albertollamasgonzalez albertollamas@albertollamas-SATELLITE-C55-A-1EK:~/Escritorio/SEGUNDO/AC/bp1/ejer10] 2021-04-16 viernes
$

```

Tabla 2. Tiempos de ejecución de la versión secuencial de la suma de vectores y de las dos versiones paralelas. Sustituir en el encabezado de la tabla “?” por el número de threads utilizados, que debe coincidir con el número de cores físicos y cores lógicos utilizados.

Nº de Componentes	T. secuencial vect. Globales 1 thread=core	PARA ATCGRID:	
		T. paralelo (versión for) 12 threads = cores lógicos = cores físicos	T. paralelo (versión sections) 12 threads = cores lógicos = cores físicos
16384	0.000434682	0.005891461	0.004376493
32768	0.000467208	0.006270841	0.004195131
65536	0.000356291	0.004906844	0.004140746
131072	0.000499056	0.006947633	0.004204031
262144	0.001367306	0.006786712	0.002125777
524288	0.002516822	0.007437937	0.005466901
1048576	0.004973934	0.008533470	0.008355312
2097152	0.008847148	0.010104209	0.011571925
4194304	0.017095919	0.012020752	0.025399286
8388608	0.032331859	0.018717587	0.035702657
16777216	0.063942061	0.032735396	0.084965989
33554432	0.126561664	0.059547167	-
67108864	0.126731809	0.060793776	-

Nº de Componentes	T. secuencial vect. Globales 1 thread=core	PARA PC:	
		T. paralelo (versión for) 4 threads = cores lógicos = cores físicos	T. paralelo (versión sections) 4 threads = cores lógicos = cores físicos
16384	0.000178382	0.000104315	0.000288413
32768	0.000215182	0.000283921	0.000364683
65536	0.000353213	0.000306561	0.000621874
131072	0.000650210	0.000655123	0.001155135
262144	0.001315974	0.001386243	0.002616417
524288	0.002718411	0.002659000	0.004577958
1048576	0.005278967	0.005449120	0.008508678
2097152	0.010407676	0.010705089	0.017055570
4194304	0.021181350	0.021579533	0.034186747
8388608	0.043924646	0.042370052	0.064451489
16777216	0.087851159	0.085049523	0.120477282
33554432	0.173085675	0.169654567	-
67108864	0.170341434	0.170111649	-



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 (que debe coincidir con el número cores físicos y lógicos) que usan los códigos. Escribir un script para realizar las ejecuciones necesarias utilizando como base el script del seminario de BP0 (se deben imprimir en el script al menos las variables de entorno que ya se imprimen en el script de BP0) ¿El tiempo de CPU que se obtiene es mayor o igual que el tiempo real (*elapsed*)? Justifique la respuesta.

RESPUESTA: Captura del script implementado sp-OpenMP-script11.sh

```

albertollamas@albertollamas-SATELLITE-C55-A-1EK: ~/Escritorio/SEGUNDO/AC/bp1/ejer11
GNU nano 2.9.3 sp-OpenMP-script11.sh

#!/bin/bash
#Autor: Alberto Llamas González
#Órdenes para el sistema de colas:
#1. Asigna al trabajo un nombre
#SBATCH --job-name=sp-OpenMP-script11
#2. Asignar el trabajo a una cola (partición)
#SBATCH --partition=ac
#2. Asignar el trabajo a un account
#SBATCH --account=ac
#Obtener información de las variables del entorno del sistema de colas:
echo "Id. usuario del trabajo: $SLURM_JOB_USER"
echo "Id. del trabajo: $SLURM_JOBID"
echo "Nombre del trabajo especificado por usuario: $SLURM_JOB_NAME"
echo "Directorio de trabajo (en el que se ejecuta el script): $SLURM_SUBMIT_DIR"
echo "Cola: $SLURM_JOB_PARTITION"
echo "Nodo que ejecuta este trabajo:$SLURM_SUBMIT_HOST"
echo "Nº de nodos asignados al trabajo: $SLURM_JOB_NUM_NODES"
echo "Nodos asignados al trabajo: $SLURM_JOB_NODELIST"
echo "CPUs por nodo: $SLURM_JOB_CPUS_PER_NODE"

for (( N= 8388608 ; N <= 67108864 ; N=N*2 ))
do
    time ./s1 $N
done
    
```

(RECUERDE ADJUNTAR LOS CÓDIGOS AL .ZIP)**CAPTURAS DE PANTALLA (ejecución en atcgrid):**

Envío a la cola de ejecución atcgrid:

```

d3estudiante26@atcgrid:~/bp1/ejer11
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer11] 2021-04-16 viernes
$ sbatch -n1 -pac -Ac -c12 --hint=nomultithread ./sp-OpenMP-script11.sh Listado1
Submitted batch job 91859
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer11] 2021-04-16 viernes
$ sbatch -n1 -pac -Ac -c12 --hint=nomultithread ./sp-OpenMP-script11.sh Listado1Ejer7
Submitted batch job 91860
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer11] 2021-04-16 viernes
$

```

Listado1 (secuencial):

```

d3estudiante26@atcgrid:~/bp1/ejer11
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer11] 2021-04-16 viernes
$ cat slurm-91859.out
Id. usuario del trabajo: d3estudiante26
Id. del trabajo: 91859
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/d3estudiante26/bp1/ejer11
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24
Tiempo: 0.034536878 / Tamaño Vectores: 8388608 / V1[0]+V2[0]=V3[0](0.227033+0.093071=0.320103) / / V1[8388607]+V2[8388607]=V3[8388607](20.637907+1.134527=21.772434) /
real 0m0.536s
user 0m0.447s
sys 0m0.039s
Tiempo: 0.066343436 / Tamaño Vectores: 16777216 / V1[0]+V2[0]=V3[0](2.552964+0.900482=3.453447) / / V1[16777215]+V2[16777215]=V3[16777215](3.153828+1.778736=4.932564) /
real 0m1.027s
user 0m0.914s
sys 0m0.081s
Tiempo: 0.128762606 / Tamaño Vectores: 33554432 / V1[0]+V2[0]=V3[0](2.836406+0.219313=3.055719) / / V1[33554431]+V2[33554431]=V3[33554431](1.607273+0.194025=1.801298) /
real 0m1.862s
user 0m1.662s
sys 0m0.151s
Tiempo: 0.129602213 / Tamaño Vectores: 33554432 / V1[0]+V2[0]=V3[0](0.879384+0.840590=1.719974) / / V1[33554431]+V2[33554431]=V3[33554431](0.394469+0.941979=1.336448) /
real 0m1.874s
user 0m1.673s
sys 0m0.143s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer11] 2021-04-16 viernes
$

```

Listado1Ejer7 (Parallel-for):

```

d3estudiante26@atcgrid:~/bp1/ejer11
Archivo Editar Ver Buscar Terminal Ayuda
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer11] 2021-04-16 viernes
$ cat slurm-91860.out
Id. usuario del trabajo: d3estudiante26
Id. del trabajo: 91860
Nombre del trabajo especificado por usuario: helloOMP
Directorio de trabajo (en el que se ejecuta el script): /home/d3estudiante26/bp1/ejer11
Cola: ac
Nodo que ejecuta este trabajo: atcgrid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24
Tiempo(seg.): 0.018119700 / Tamaño Vectores: 8388608 / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=1677721.600000) / / V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1.600000) /
real 0m0.103s
user 0m0.569s
sys 0m0.221s
Tiempo(seg.): 0.035296280 / Tamaño Vectores: 16777216 / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=3355443.200000) / / V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
real 0m0.101s
user 0m1.047s
sys 0m0.407s
Tiempo(seg.): 0.064416870 / Tamaño Vectores: 33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
real 0m0.210s
user 0m1.539s
sys 0m0.744s
Tiempo(seg.): 0.063335910 / Tamaño Vectores: 33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=6710886.400000) / / V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
real 0m0.199s
user 0m1.984s
sys 0m0.833s
[AlbertoLlamasGonzalez d3estudiante26@atcgrid:~/bp1/ejer11] 2021-04-16 viernes
$

```

Tabla 3. Tiempos de ejecución de la versión secuencial de la suma de vectores y de las dos versiones paralelas. Sustituir en el encabezado de la tabla “i?” por el número de threads utilizados.

Nº de Componentes	Tiempo secuencial vect. Globales 1 thread = 1 core lógico = 1 core físico			Tiempo paralelo/versión for 12 Threads = cores lógicos=cores físicos		
	<i>Elapsed</i>	<i>CPU-user</i>	<i>CPU- sys</i>	<i>Elapsed</i>	<i>CPU-user</i>	<i>CPU- sys</i>
8388608	0.536s	0.447s	0.039s	0.103s	0.569s	0.221s
16777216	1.027s	0.914s	0.081s	0.101s	1.047s	0.407s
33554432	1.862s	1.662s	0.151s	0.210s	1.839s	0.744s
67108864	1.874s	1.673s	0.143s	0.199s	1.984s	0.833s

En la secuencial la suma del tiempo cpu-user y del cpu-sys es igual al elapsed time, sin embargo, en la que utiliza paralelización la suma de ambos es mayor que el elapsed time y esto se debe a que se cuenta el tiempo que consume cada core físico y al sumarlo no suma el tiempo transcurrido realmente ya que son simultáneos y no secuenciales.