

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):

Grupo de prácticas y profesor de prácticas:

Fecha de entrega:

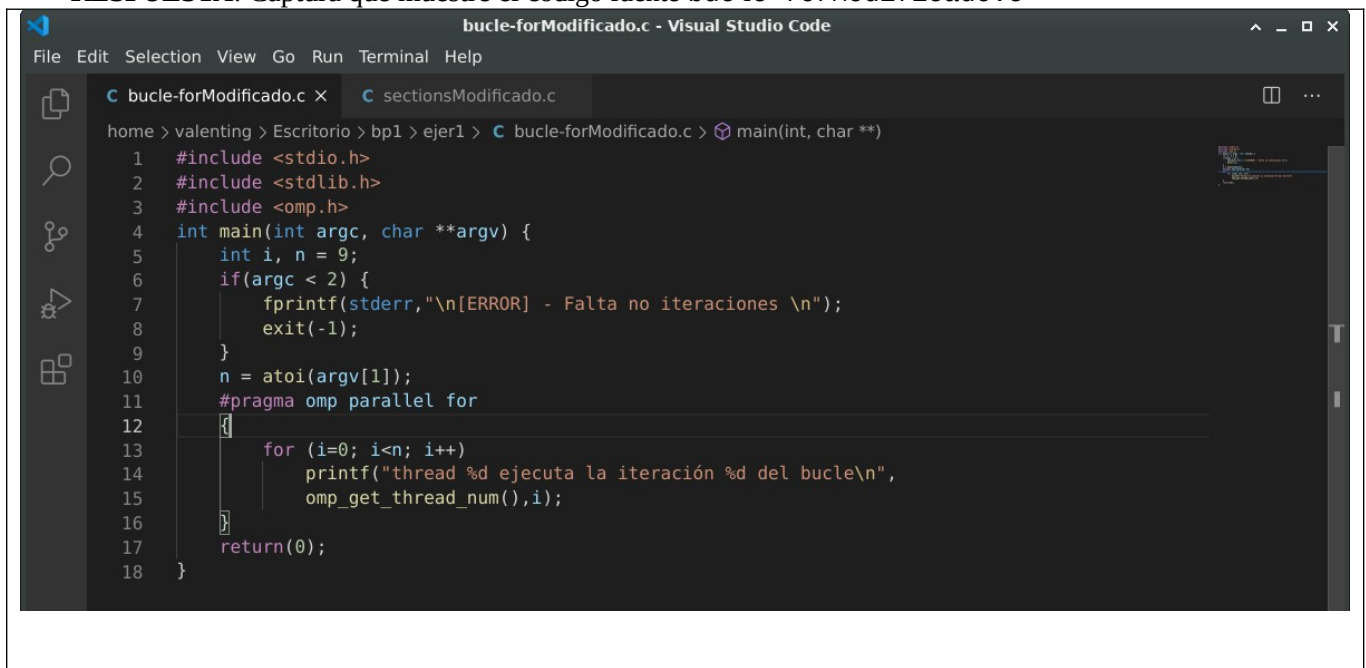
Fecha evaluación en clase:

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

Ejercicios basados en los ejemplos del seminario práctico

1. Usar la 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`



```
bucle-forModificado.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help

C bucle-forModificado.c X C sectionsModificado.c
home > valenting > Escritorio > bp1 > ejer1 > C bucle-forModificado.c > main(int, char **)
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <omp.h>
4  int main(int argc, char **argv) {
5      int i, n = 9;
6      if(argc < 2) {
7          fprintf(stderr, "\n[ERROR] - Falta no iteraciones \n");
8          exit(-1);
9      }
10     n = atoi(argv[1]);
11     #pragma omp parallel for
12     for (i=0; i<n; i++)
13         printf("thread %d ejecuta la iteración %d del bucle\n",
14             omp_get_thread_num(), i);
15     return(0);
16 }
```

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

```

sectionsModificado.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help

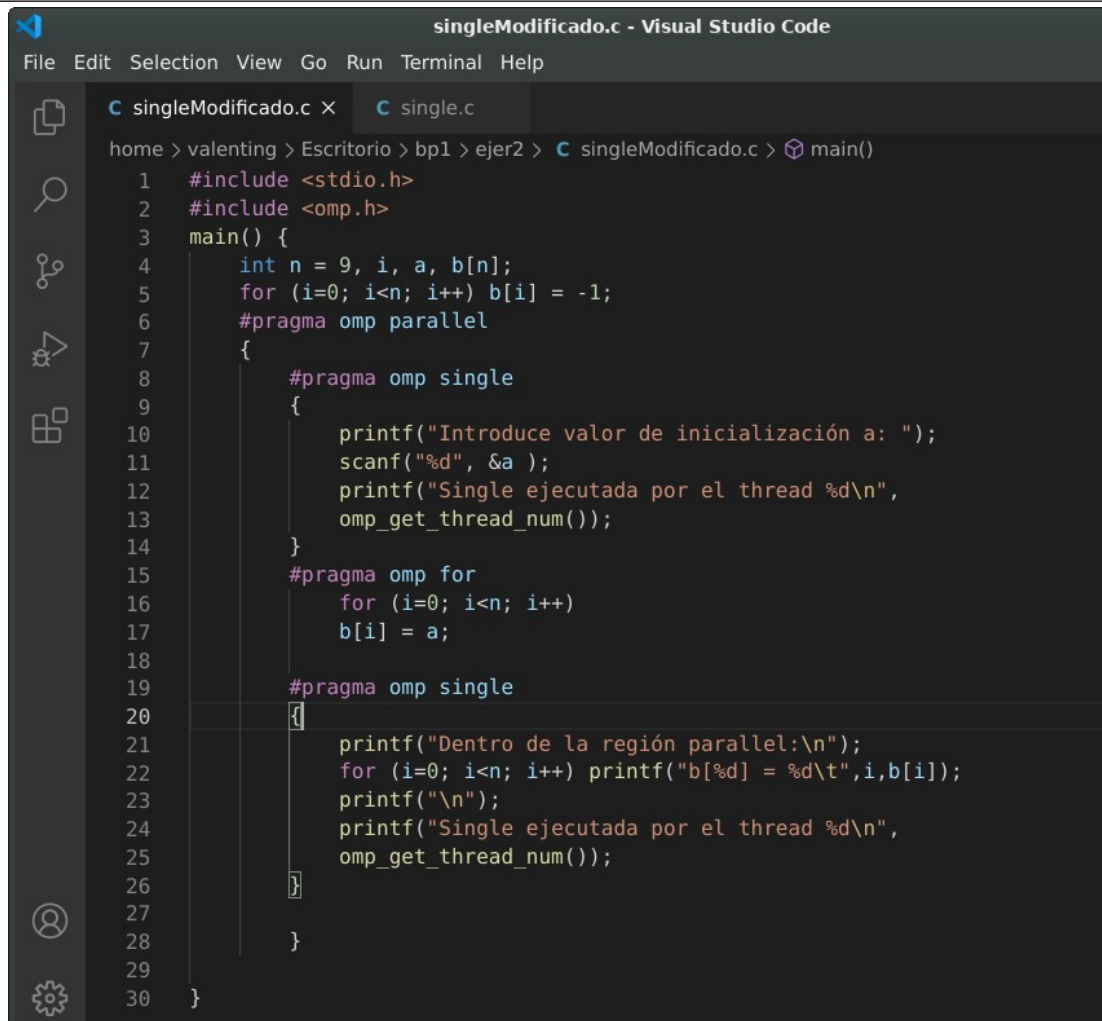
C bucle-forModificado.c  C sectionsModificado.c X

home > valenting > Escritorio > bp1 > ejer1 > C sectionsModificado.c > funcB()
1  #include <stdio.h>
2  #include <omp.h>
3  void funcA() {
4      printf("En funcA: esta sección la ejecuta el thread
5          %d\n",
6          omp_get_thread_num());
7  }
8  void funcB() {
9      printf("En funcB: esta sección la ejecuta el thread
10         %d\n",
11         omp_get_thread_num());
12 }
13 main(){
14     #pragma omp parallel sections
15     {
16         #pragma omp section
17         (void) funcA();
18         #pragma omp section
19         (void) funcB();
20     }
21 }

```

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`



```

singleModificado.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help

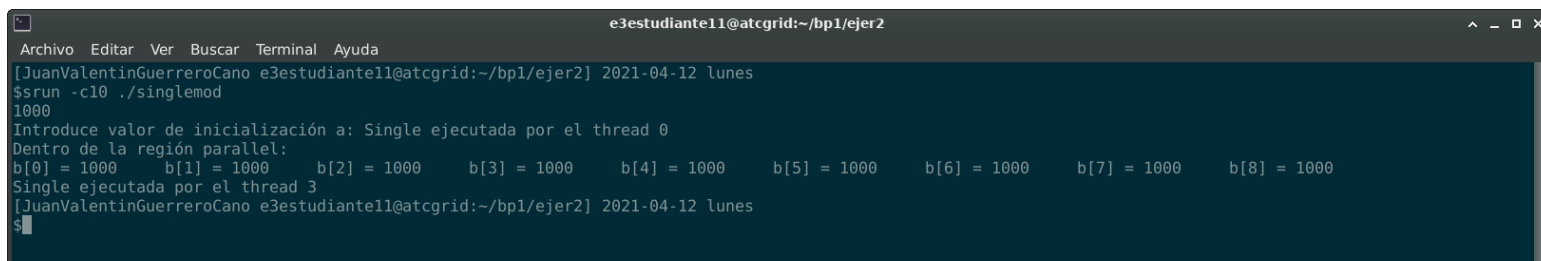
C singleModificado.c x C single.c

home > valenting > Escritorio > bp1 > ejer2 > C singleModificado.c > main()

1  #include <stdio.h>
2  #include <omp.h>
3  main() {
4      int n = 9, i, a, b[n];
5      for (i=0; i<n; i++) b[i] = -1;
6      #pragma omp parallel
7      {
8          #pragma omp single
9          {
10             printf("Introduce valor de inicialización a: ");
11             scanf("%d", &a );
12             printf("Single ejecutada por el thread %d\n",
13                 omp_get_thread_num());
14         }
15         #pragma omp for
16         for (i=0; i<n; i++)
17             b[i] = a;
18
19         #pragma omp single
20         {
21             printf("Dentro de la región parallel:\n");
22             for (i=0; i<n; i++) printf("b[%d] = %d\t",i,b[i]);
23             printf("\n");
24             printf("Single ejecutada por el thread %d\n",
25                 omp_get_thread_num());
26         }
27     }
28 }
29
30

```

CAPTURAS DE PANTALLA:



```

e3estudiante11@atcgrid:~/bp1/ejer2
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiantell@atcgrid:~/bp1/ejer2] 2021-04-12 lunes
$srn -c10 ./singlemod
1000
Introduce valor de inicialización a: Single ejecutada por el thread 0
Dentro de la región parallel:
b[0] = 1000    b[1] = 1000    b[2] = 1000    b[3] = 1000    b[4] = 1000    b[5] = 1000    b[6] = 1000    b[7] = 1000    b[8] = 1000
Single ejecutada por el thread 3
[JuanValentinGuerreroCano e3estudiantell@atcgrid:~/bp1/ejer2] 2021-04-12 lunes
$

```

- 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 singleModificado.c x
home > valenting > Escritorio > bp1 > ejer3 > C singleModificado.c > main()
1  #include <stdio.h>
2  #include <omp.h>
3  main() {
4      int n = 9, i, a, b[n];
5      for (i=0; i<n; i++) b[i] = -1;
6      #pragma omp parallel
7      {
8          #pragma omp master
9          {
10             printf("Introduce valor de inicialización a: ");
11             scanf("%d", &a);
12             printf("Single ejecutada por el thread %d\n",
13                 omp_get_thread_num());
14         }
15         #pragma omp barrier
16
17         #pragma omp for
18         for (i=0; i<n; i++)
19             b[i] = a;
20
21         #pragma omp master
22         {
23             printf("Dentro de la región parallel:\n");
24             for (i=0; i<n; i++) printf("b[%d] = %d\t",i,b[i]);
25             printf("\n");
26             printf("Single ejecutada por el thread %d\n",
27                 omp_get_thread_num());
28         }
29         #pragma omp barrier
30
31         printf("Después de la región parallel:\n");
32         for (i=0; i<n; i++) printf("b[%d] = %d\t",i,b[i]);
33         printf("\n");
34     }
35 }

```

CAPTURAS DE PANTALLA:

```

[JuanValentinGuerreroCano e3estudiantell@atcgriid:~/bp1/ejer3] 2021-04-12 lunes
$srn -pac -Aac ./singlemod
10000
Introduce valor de inicialización a:
Single ejecutada por el thread 0
Dentro de la región parallel:
b[0] = 10000  b[1] = 10000  b[2] = 10000  b[3] = 10000  b[4] = 10000  b[5] = 10000  b[6] = 10000  b[7] = 10000  b[8] = 10000
Single ejecutada por el thread 0
Después de la región parallel:
b[0] = 10000  b[1] = 10000  b[2] = 10000  b[3] = 10000  b[4] = 10000  b[5] = 10000  b[6] = 10000  b[7] = 10000  b[8] = 10000
[JuanValentinGuerreroCano e3estudiantell@atcgriid:~/bp1/ejer3] 2021-04-12 lunes
$

```

RESPUESTA A LA PREGUNTA:

Los resultados obtenidos son los mismos a excepción de que la hebra que ejecuta el trozo de código que muestra por pantalla el vector es la hebra 0 (master), en lugar de cualquiera de las otras hebras en el caso del ejercicio anterior.

- ¿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:

Porque en dicho caso las hebras continúan la ejecución sin esperar al resto de hebras, y la hebra master llegará a imprimir el resultado antes de que el resto de hebras lo hayan calculado correctamente. En caso de que la directiva `barrier` se hubiese mantenido, todas las hebras hubiesen parado en el momento de leer esa directiva, esperando al resto de hebras. De esta forma, al esperar la hebra 0 a las demás, imprime el 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:

```
e3estudiante11@atcgrid:~/bp1/ejer5
Archivo Editar Ver Buscar Terminal Ayuda
[juanvalentinguerreroCano e3estudiantell@atcgrid:~/bp1/ejer5] 2021-04-03 sábado
$ time srun -p ac ./sumavectores 10000000
Tiempo(seg.):0.040905265 / Tamaño Vectores:10000000 / V1[0]+V2[0]=V3[0](0.455492+0.130770=0.586261) / V1[9999999]+V2[9999999]=V3[9999999](1.255737+0.945709=2.201446)/
real    0m0.698s
user    0m0.007s
sys      0m0.008s
[juanvalentinguerreroCano e3estudiantell@atcgrid:~/bp1/ejer5] 2021-04-03 sábado
$
```

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[juanvalentinguerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer5] 2021-04-03 sábado
$ gcc -O2 -o sumavectores sumavectores.c
sumavectores.c: In function 'main':
sumavectores.c:88:53: warning: format '%lu' expects argument of type 'long unsigned int', but argument 3 has type 'unsigned int' [-Wformat=]
   88 | printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\n",ncgt,N);
      |                                     ~~~~^
      |                                     |
      |                                     long unsigned int
      |                                     unsigned int
      |                                     %u
[juanvalentinguerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer5] 2021-04-03 sábado
$
```

RESPUESTA: La suma de dichos tiempos es menor que el tiempo real. La diferencia de tiempo entre dicha suma y el tiempo real es el asociado a las esperas por I/O o asociado a la ejecución de otros programas.

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. 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):

```
e3estudiante11@atcgrid:~/bp1/ejer6
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer6] 2021-04-13 martes
$gcc -S -O2 sumavectores.c -o sumavectores.s -lrt
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer6] 2021-04-13 martes
$gcc -O2 sumavectores.c -o sumavectores -lrt
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer6] 2021-04-13 martes
$ls
sumavectores sumavectores.c sumavectores.s
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer6] 2021-04-13 martes
$
```

Para 10 000 0000 componentes:

```
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer6] 2021-04-12 lunes
$time srun -pac ./sumavectores 10000000
Tiempo(seg.):0.059893436 / Tamaño Vectores:10000000 / V1[0]+V2[0]=V3[0](0.067952+0.895260=0.963212) / /V1[9999999]+V2[9999999]=V3[9999999](0.794059+0.797098=1.591157)/
real 0m0.701s
user 0m0.005s
sys 0m0.010s
```

Para 10 componentes:

```
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer6] 2021-04-12 lunes
$time srun -pac ./sumavectores 10
Tiempo(seg.):0.000390649 / Tamaño Vectores:10 / V1[0]+V2[0]=V3[0](2.894951+1.064315=3.959267) / /V1[9]+V2[9]=V3[9](0.267163+0.685184=0.952347)/
real 0m0.124s
user 0m0.007s
sys 0m0.008s
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer6] 2021-04-12 lunes
$
```

RESPUESTA: cálculo de los MIPS y los MFLOPS

Para 10000000 componentes

$$\text{MIPS} = (6 * 10000000) / (0,059893436 * 10^6) = 1001.779 \text{ MIPS}$$

$$\text{MFLOPS} = (1 * 10000000) / (0,059893436 * 10^6) = 166.96 \text{ MFLOPS}$$

Para 10 componentes

$$\text{MIPS} = (6 * 10) / (0,000390649 * 10^6) = 0,15357 \text{ MIPS}$$

$$\text{MFLOPS} = (1 * 10) / (0,000390649 * 10^6) = 0,02519 \text{ MFLOPS}$$

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

```

e3estudiante11@atcgriid:~/bp1/ejer6
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
.L6:
    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

```

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`


```

#pragma omp parallel
{
    //Paralelización
    //Inicializar vectores
    if(N<9)
    {
        #pragma omp for //Bucle for paralelizado para inicializar los vectores
        for(i=0;i<N;i++)
        {
            v1[i]=N*0.1+i*0.1;
            v2[i]=N*0.1-i*0.1;
        }
    }
    else
    {
        srand(time(0));
        #pragma omp for //Bucle for paralelizado para inicializar los vectores
        for(i=0;i<N;i++)
        {
            v1[i]=rand()/((double)rand());
            v2[i]=rand()/((double)rand());//printf("%d:%f,%f/",i,v1[i],v2[i]);
        }
    }

    //clock_gettime(CLOCK_REALTIME,&cgt1);
    #pragma omp single
    {
        //Nos interesa que el tiempo únicamente nos lo tome una hebra
        start = omp_get_wtime(); // omp_get_wtime() nos devuelve un double, de ahí las nuevas variables double start y end.
    }

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

    //clock_gettime(CLOCK_REALTIME,&cgt2);
    #pragma omp single
    {
        //Nos interesa que el tiempo únicamente nos lo tome una hebra
        end = omp_get_wtime();
    }

    ncgt=(end - start); //Calculo del tiempo mediante la resta del tiempo tomado antes de la suma y despues de la suma
}

```

(RECUERDE ADJUNTAR CÓDIGO FUENTE AL .ZIP)

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

```

e3estudiante11@atcgrid:~/bp1/ejer7
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer7] 2021-04-13 martes
$gcc -O2 -fopenmp sumavectores.c -o sumavectores
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer7] 2021-04-13 martes
$export OMP_NUM_THREADS=8
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer7] 2021-04-13 martes
$srn -pac ./sumavectores 8
Tiempo(seg.):0.000535995 / 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) /
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer7] 2021-04-13 martes
$srn -pac ./sumavectores 11
Tiempo(seg.):0.000743493 / Tamaño Vectores:11
/ V1[0]+V2[0]=V3[0](1.277671+0.884196=2.161867) / /V1[10]+V2[10]=V3[10](1.001188+1.188898=2.190086)/
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer7] 2021-04-13 martes
$

```


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`

```
#pragma omp parallel sections
{
    //No nos importa que el numero de componentes de los vectores no sea divisible
    //por 4 ya que la división se queda con la parte entera
    #pragma omp section
    for(i=0;i<N/4;i++){
        v1[i]=N*0.1+i*0.1;
        v2[i]=N*0.1-i*0.1;
    }

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

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

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

//clock_gettime(CLOCK_REALTIME,&cgt1);
start = omp_get_wtime(); // omp_get_wtime() nos devuelve un double, de ahí las nuevas variables double start y end.

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

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

```

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

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

//clock_gettime(CLOCK_REALTIME,&cgt2);
end = omp_get_wtime();

ncgt=(end - start);          //Calculo del tiempo mediante la resta del tiempo tomado antes de la suma y despues de la suma

//Imprimir resultado de la suma y el tiempo de ejecución
if (N<10) {
    printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\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) /",
        ncgt,N,v1[0],v2[0],v3[0],v1[N/4],v2[N/4],v3[N/4],v1[N/2],v2[N/2],v3[N/2],v1[3*N/4],v2[3*N/4],v3[3*N/4],v1[N-1],v2[N-1],v3[N-1]);
}

```

(RECUERDE ADJUNTAR CÓDIGO FUENTE AL .ZIP)

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

```

e3estudiante11@atcgrid:~/bp1/ejer8
Archivo Editar Ver Buscar Terminal Ayuda
$gcc -O2 -fopenmp sumavectores.c -o sumavectores
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer8] 2021-04-13 martes
$export OMP_NUM_THREADS=8
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer8] 2021-04-13 martes
$srn -pac ./sumavectores 8
Tiempo(seg.):0.000510372          / 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) /
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer8] 2021-04-13 martes
$srn -pac ./sumavectores 11
Tiempo(seg.):0.000523783          / 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)/
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer8] 2021-04-13 martes
$

```

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. 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 se utilizarán como máximo el número de hebras restringido al ejecutar “export OMP_NUM_THREADS = 8”. Podrá tener como máximo el número de cores disponibles en el momento de ejecución pues no se ha impuesto ningún tipo de restricción o límite en el número de cores.

En el ejercicio 8 se utilizarán como máximo tantas hebras como sections tengamos, en nuestro caso, 4 ya que hemos dividido el vector en 4 partes, cada una de las cuales será inicializada y sumada por una hebra distinta. El número de cores que podrá emplear como máximo es el número de hebras que podamos utilizar, en este caso 4, suponiendo que cada core emplease una única hebra.

10. Rellenar una tabla como la Tabla 215 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

```

home > valenting > Escritorio > bp1 > ejer10 > sp-OpenMP-script10.sh
1  #!/bin/bash
2  #Autor: Juan Valentín Guerrero Cano
3  #Órdenes para el Gestor de carga de trabajo:
4  #1. Asigna al trabajo un nombre
5  #SBATCH --job-name=helloOMP2
6  #2. Asignar el trabajo a una partición (cola)
7  #SBATCH --partition=ac
8  #2. Asignar el trabajo a un account
9  #SBATCH --account=ac
10
11 #Obtener información de las variables del entorno del Gestor de carga de trabajo:
12 echo "Id. usuario del trabajo: $SLURM_JOB_USER"
13 echo "Id. del trabajo: $SLURM_JOBID"
14 echo "Nombre del trabajo especificado por usuario: $SLURM_JOB_NAME"
15 echo "Directorio de trabajo (en el que se ejecuta el script): $SLURM_SUBMIT_DIR"
16 echo "Cola: $SLURM_JOB_PARTITION"
17 echo "Nodo que ejecuta este trabajo: $SLURM_SUBMIT_HOST"
18 echo "No de nodos asignados al trabajo: $SLURM_JOB_NUM_NODES"
19 echo "Nodos asignados al trabajo: $SLURM_JOB_NODELIST"
20 echo "CPUs por nodo: $SLURM_JOB_CPUS_PER_NODE"
21
22 echo -e "\nVECTORES GLOBALES\n\n"
23
24 for (( N = 16384 ; N<= 67108864 ; N=N*2))
25 do
26     ./$1 $N
27 done
28

```

(RECUERDE ADJUNTAR LOS CÓDIGOS AL .ZIP)

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

Compilación en atcgrid:

```
e3estudiante11@atcgrid:~/bp1/e
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$gcc -O2 -fopenmp sumavectores1.c -o sumavectores1 -lrt
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$gcc -O2 -fopenmp sumavectores7.c -o sumavectores7 -lrt
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$gcc -O2 -fopenmp sumavectores8.c -o sumavectores8 -lrt
```

Ejecución en atcgrid y envío a la cola:

```
e3estudiante11@atcgrid:~/bp1/
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$sbatch -n1 -pac -Aac -c12 --hint=nomultithread ./sp-OpenMP-script10.sh sumavectores1
Submitted batch job 89659
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$sbatch -n1 -pac -Aac -c12 --hint=nomultithread ./sp-OpenMP-script10.sh sumavectores7
Submitted batch job 89660
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$sbatch -n1 -pac -Aac -c12 --hint=nomultithread ./sp-OpenMP-script10.sh sumavectores8
Submitted batch job 89662
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$
```

-Secuencial

```
e3estudiante11@atcgrid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
Scat slurm-89659.out
Id. usuario del trabajo: e3estudiante11
Id. del trabajo: 89659
Nombre del trabajo especificado por usuario: ejer10
Directorio de trabajo (en el que se ejecuta el script): /home/e3estudiante11/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

VECTORES GLOBALES

Tiempo(seg.):0.000437267 / Tamaño Vectores:16384 / V1[0]+V2[0]=V3[0](0.837885+1.282785=2.120670) / /V1[16383]+V2[16383]=V3[16383](1.129823+1.747311=2.877134)/
Tiempo(seg.):0.000468730 / Tamaño Vectores:32768 / V1[0]+V2[0]=V3[0](0.837885+1.282785=2.120670) / /V1[32767]+V2[32767]=V3[32767](0.017050+1.865032=1.882082)/
Tiempo(seg.):0.000388674 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.837885+1.282785=2.120670) / /V1[65535]+V2[65535]=V3[65535](1.205301+0.254837=1.460138)/
Tiempo(seg.):0.000543215 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.837885+1.282785=2.120670) / /V1[131071]+V2[131071]=V3[131071](0.147019+5.957723=6.104742)/
Tiempo(seg.):0.001400244 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.554391+8.781118=9.335510) / /V1[262143]+V2[262143]=V3[262143](1.739970+0.982386=2.722356)/
Tiempo(seg.):0.002561063 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.554391+8.781118=9.335510) / /V1[524287]+V2[524287]=V3[524287](0.073336+0.669686=0.743022)/
Tiempo(seg.):0.004719868 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](0.554391+8.781118=9.335510) / /V1[1048575]+V2[1048575]=V3[1048575](0.765201+1.009600=1.774801)/
Tiempo(seg.):0.008690762 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](0.554391+8.781118=9.335510) / /V1[2097151]+V2[2097151]=V3[2097151](2.125773+0.376863=2.502635)/
Tiempo(seg.):0.017157876 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](0.554391+8.781118=9.335510) / /V1[4194303]+V2[4194303]=V3[4194303](2.429029+0.755111=3.184139)/
Tiempo(seg.):0.033072116 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](3.097501+0.106665=3.204166) / /V1[8388607]+V2[8388607]=V3[8388607](0.950088+0.912508=1.862595)/
Tiempo(seg.):0.065456833 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](3.097501+0.106665=3.204166) / /V1[16777215]+V2[16777215]=V3[16777215](0.760988+1.320521=2.081509)/
Tiempo(seg.):0.129765270 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](11.531533+3.832866=15.364399) / /V1[33554431]+V2[33554431]=V3[33554431](0.907342+0.159234=1.066576)/
Tiempo(seg.):0.248747145 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](0.174990+1.353232=1.528222) / /V1[67108863]+V2[67108863]=V3[67108863](5.428365+0.506681=5.935046)/
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$
```


- Parallel-for

```

e3estudiante11@atcgrid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
Scat slurm-89660.out
Id. usuario del trabajo: e3estudiante11
Id. del trabajo: 89660
Nombre del trabajo especificado por usuario: ejer10
Directorio de trabajo (en el que se ejecuta el script): /home/e3estudiante11/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

VECTORES GLOBALES

Tiempo(seg.):0.008041292      / Tamaño Vectores:16384      / V1[0]+V2[0]=V3[0](0.532436+1.009967=1.542403) / /V1[16383]+V2[16383]=V3[16383](0.241872+0.636028=0.877900)/
Tiempo(seg.):0.007218782      / Tamaño Vectores:32768      / V1[0]+V2[0]=V3[0](0.895508+0.532436=1.427945) / /V1[32767]+V2[32767]=V3[32767](1.589006+0.858408=2.447415)/
Tiempo(seg.):0.006528758      / Tamaño Vectores:65536      / V1[0]+V2[0]=V3[0](0.674034+3.621718=4.295752) / /V1[65535]+V2[65535]=V3[65535](5.711178+26.625648=32.336826)
Tiempo(seg.):0.003131967      / Tamaño Vectores:131072     / V1[0]+V2[0]=V3[0](1.029299+5.056526=6.085826) / /V1[131071]+V2[131071]=V3[131071](6.957120+0.093197=7.050317)
Tiempo(seg.):0.006734565      / Tamaño Vectores:262144     / V1[0]+V2[0]=V3[0](1.998247+5.056526=7.054774) / /V1[262143]+V2[262143]=V3[262143](1.240387+0.915156=2.155543)
Tiempo(seg.):0.006833177      / Tamaño Vectores:524288     / V1[0]+V2[0]=V3[0](1.910976+1.029299=2.940275) / /V1[524287]+V2[524287]=V3[524287](1.880871+0.752738=2.633609)
Tiempo(seg.):0.006305639      / Tamaño Vectores:1048576    / V1[0]+V2[0]=V3[0](0.277859+1.014563=1.292422) / /V1[1048575]+V2[1048575]=V3[1048575](0.807089+0.412172=1.219261)
Tiempo(seg.):0.009416416      / Tamaño Vectores:2097152    / V1[0]+V2[0]=V3[0](0.374020+0.135222=0.509242) / /V1[2097151]+V2[2097151]=V3[2097151](0.643096+0.927629=1.570726)
Tiempo(seg.):0.012201764      / Tamaño Vectores:4194304    / V1[0]+V2[0]=V3[0](0.349934+1.252873=1.602807) / /V1[4194303]+V2[4194303]=V3[4194303](9.882284+1.742008=11.624292)
Tiempo(seg.):0.020933412      / Tamaño Vectores:8388608    / V1[0]+V2[0]=V3[0](12.696043+1.970145=14.666187) / /V1[8388607]+V2[8388607]=V3[8388607](5.660479+0.721352=6.381831)
Tiempo(seg.):0.033943541      / Tamaño Vectores:16777216   / V1[0]+V2[0]=V3[0](0.363701+0.628851=0.992552) / /V1[16777215]+V2[16777215]=V3[16777215](2.051018+5.481743=7.532762)
Tiempo(seg.):0.063719124      / Tamaño Vectores:33554432   / V1[0]+V2[0]=V3[0](1.701221+1.139344=2.840565) / /V1[33554431]+V2[33554431]=V3[33554431](0.677395+0.952111=1.629506)
slurmstepd: error: *** JOB 89660 ON atcgrid1 CANCELLED AT 2021-04-13T20:07:15 DUE TO TIME LIMIT ***
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$

```

- Parallel-sections

```

e3estudiante11@atcgrid:~/bp1/ejer10
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
Scat slurm-89662.out
Id. usuario del trabajo: e3estudiante11
Id. del trabajo: 89662
Nombre del trabajo especificado por usuario: ejer10
Directorio de trabajo (en el que se ejecuta el script): /home/e3estudiante11/bp1/ejer10
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
No de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid3
CPUs por nodo: 24

VECTORES GLOBALES

Tiempo(seg.):0.004564829      / 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.):0.000633370      / 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.):0.000846434      / 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.):0.000814691      / 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.):0.001363307      / 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.):0.002205960      / 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.):0.004434153      / 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.):0.005259294      / 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.):0.015688993      / 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.):0.027820121      / 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.):0.029576942      / 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.):0.094521433      / 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.):0.153732225      / Tamaño Vectores:67108864   / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) / /V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000)/
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer10] 2021-04-13 martes
$

```

Ejecución en PC:

-Secuencial:

```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
[JuanValentinGuerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer10] 2021-04-13 martes
Sexport OMP_NUM_THREADS=4
[JuanValentinGuerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer10] 2021-04-13 martes
S./sp-OpenMP-script10.sh sumavectores1
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:

VECTORES GLOBALES

Tiempo(seg.):0.000103411 / Tamaño Vectores:16384 / V1[0]+V2[0]=V3[0](0.775270+3.155773=3.931043) / /V1[16383]+V2[16383]=V3[16383](2.479621+1.251876=3.731497)/
Tiempo(seg.):0.000192207 / Tamaño Vectores:32768 / V1[0]+V2[0]=V3[0](0.775270+3.155773=3.931043) / /V1[32767]+V2[32767]=V3[32767](5.014032+1.160957=6.174989)/
Tiempo(seg.):0.000170040 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](0.612336+1.020754=1.633090) / /V1[65535]+V2[65535]=V3[65535](14.873708+1.472968=16.346676)
Tiempo(seg.):0.000343345 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.612336+1.020754=1.633090) / /V1[131071]+V2[131071]=V3[131071](0.806089+0.799611=1.60570)
Tiempo(seg.):0.001098176 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](0.612336+1.020754=1.633090) / /V1[262143]+V2[262143]=V3[262143](0.144173+2.128963=2.27313)
Tiempo(seg.):0.001755363 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.612336+1.020754=1.633090) / /V1[524287]+V2[524287]=V3[524287](1.271203+6.201766=7.47296)
Tiempo(seg.):0.003688164 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](0.612336+1.020754=1.633090) / /V1[1048575]+V2[1048575]=V3[1048575](0.830661+0.554318=1.384979)
Tiempo(seg.):0.006049583 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](0.612336+1.020754=1.633090) / /V1[2097151]+V2[2097151]=V3[2097151](15.585644+0.038037=15.623681)
Tiempo(seg.):0.010985464 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](0.612336+1.020754=1.633090) / /V1[4194303]+V2[4194303]=V3[4194303](0.799028+2.459200=3.258228)
Tiempo(seg.):0.021325183 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](0.612336+1.020754=1.633090) / /V1[8388607]+V2[8388607]=V3[8388607](1.529332+2.133493=3.662825)
Tiempo(seg.):0.042316946 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](0.612336+1.020754=1.633090) / /V1[16777215]+V2[16777215]=V3[16777215](0.899287+2.534576=3.433863)
Tiempo(seg.):0.083761573 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.367369+0.994666=1.362035) / /V1[33554431]+V2[33554431]=V3[33554431](1.077836+0.140258=1.218094)
Tiempo(seg.):0.184244157 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](0.367369+0.994666=1.362035) / /V1[67108863]+V2[67108863]=V3[67108863](3.024513+1.902243=4.926756)
[JuanValentinGuerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer10] 2021-04-13 martes

```

-Parallel-for

```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
S./sp-OpenMP-script10.sh sumavectores7
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:

VECTORES GLOBALES

Tiempo(seg.):0.000020691 / Tamaño Vectores:16384 / V1[0]+V2[0]=V3[0](0.528622+1.471808=2.000429) / /V1[16383]+V2[16383]=V3[16383](0.480444+1.228768=1.709212)/
Tiempo(seg.):0.000221795 / Tamaño Vectores:32768 / V1[0]+V2[0]=V3[0](1.865455+11.732170=13.597626) / /V1[32767]+V2[32767]=V3[32767](0.997715+2.704823=3.702537)
Tiempo(seg.):0.000355809 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](1.471808+1.847276=3.319084) / /V1[65535]+V2[65535]=V3[65535](0.811450+0.720150=1.531600)/
Tiempo(seg.):0.000177999 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](0.528622+1.471808=2.000429) / /V1[131071]+V2[131071]=V3[131071](1.971265+3.802034=5.77329)
Tiempo(seg.):0.000348668 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](1.471808+1.847276=3.319084) / /V1[262143]+V2[262143]=V3[262143](0.611214+1.200206=1.81142)
Tiempo(seg.):0.001321537 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](0.528622+1.471808=2.000429) / /V1[524287]+V2[524287]=V3[524287](0.382224+1.260955=1.64317)
Tiempo(seg.):0.002601611 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](2.357225+0.541744=2.898968) / /V1[1048575]+V2[1048575]=V3[1048575](1.047829+0.431562=1.479391)
Tiempo(seg.):0.004367006 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](2.357225+0.541744=2.898968) / /V1[2097151]+V2[2097151]=V3[2097151](1.064184+2.042061=3.106245)
Tiempo(seg.):0.008026686 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](0.782154+0.710049=1.492202) / /V1[4194303]+V2[4194303]=V3[4194303](0.228949+5.536156=5.765105)
Tiempo(seg.):0.014602151 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](0.410190+1.419671=1.829861) / /V1[8388607]+V2[8388607]=V3[8388607](2.452862+1.566369=4.019232)
Tiempo(seg.):0.029365428 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1.472813+0.646533=2.119346) / /V1[16777215]+V2[16777215]=V3[16777215](1.668853+1.791663=3.460516)
Tiempo(seg.):0.054184099 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](0.705179+5.467215=6.172394) / /V1[33554431]+V2[33554431]=V3[33554431](0.854682+0.296487=1.151169)
Tiempo(seg.):0.106660684 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](1.600927+0.747786=2.348713) / /V1[67108863]+V2[67108863]=V3[67108863](0.487534+1.418990=1.906524)
[JuanValentinGuerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer10] 2021-04-13 martes
S
[JuanValentinGuerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer10] 2021-04-13 martes
S

```

- Parallel-sections

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[JuanValentinGuerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer10] 2021-04-13 martes
Sexport OMP_NUM_THREADS=4
[JuanValentinGuerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer10] 2021-04-13 martes
$ ./sp-OpenMP-script10.sh sumavectores8
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:

VECTORES GLOBALES

Tiempo(seg.):0.000063377 / 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)/
0=3276.800000)/
Tiempo(seg.):0.000113854 / 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)/
0=6553.600000)/
Tiempo(seg.):0.000241421 / 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)/
000=13107.200000)/
Tiempo(seg.):0.000342300 / 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)/
0.100000=26214.400000)/
Tiempo(seg.):0.000691996 / 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)/
0.100000=52428.800000)/
Tiempo(seg.):0.001502023 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](52428.800000+52428.800000=104857.600000) / /V1[524287]+V2[524287]=V3[524287](104857.500000+0+0.100000=104857.600000)/
0+0.100000=104857.600000)/
Tiempo(seg.):0.002683686 / 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)/
100000=209715.200000)/
Tiempo(seg.):0.003615653 / 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)/
300000=419430.400000)/
Tiempo(seg.):0.011222828 / 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)/
700000=838860.800000)/
Tiempo(seg.):0.015618511 / 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)/
1.500000=1677721.600000)/
Tiempo(seg.):0.027662726 / 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)/
355443.200000=3355443.200000)/
Tiempo(seg.):0.053513053 / 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)/
710886.300000=6710886.400000)/
Tiempo(seg.):0.105896613 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=13421772.800000) / /V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000)/
13421772.700000=13421772.800000)/
[JuanValentinGuerreroCano valenting@valenting-Aspire-A515-54:~/Escritorio/bp1/ejer10] 2021-04-13 martes

```

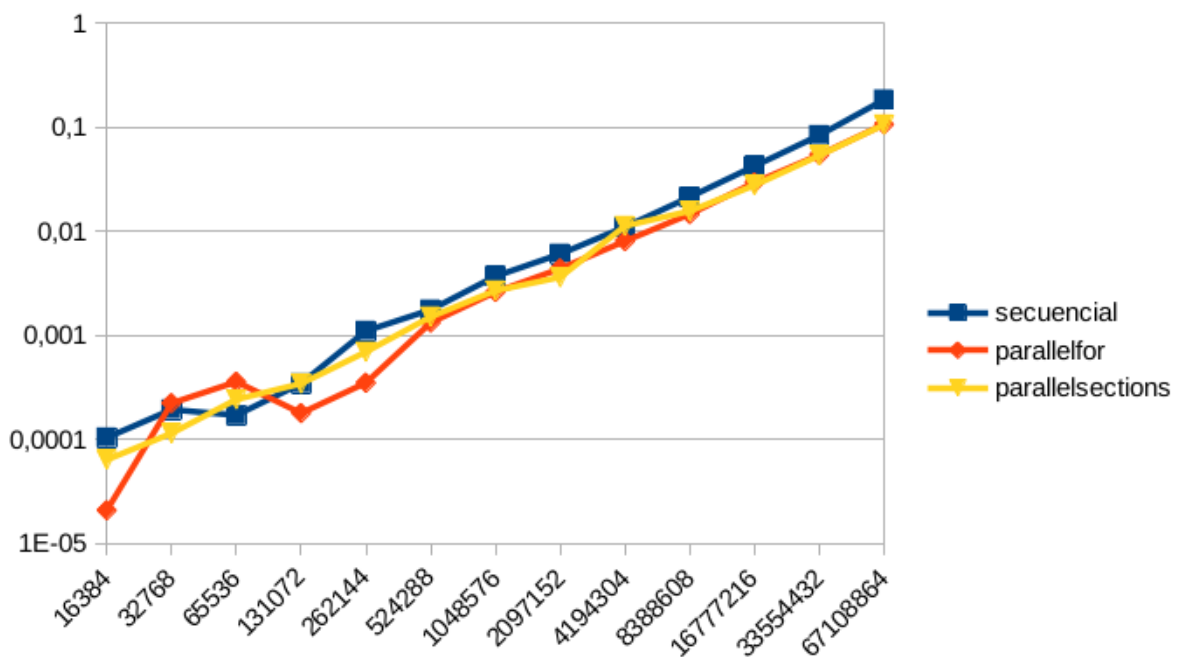
Tabla 2. Tiempos de ejecución de la versión secuencial de la suma de vectores y de las dos versiones paralelas. Sustituir en el encabezado de la tabla “¿?” por el número de threads utilizados, que debe coincidir con el número de cores físicos y cores lógicos utilizados.

Nº de Componentes	T. secuencial vect. Globales 1 thread=core	PARA PC:	
		T. paralelo (versión for) 4 threads = cores lógicos = cores físicos	T. paralelo (versión sections) 4 threads = cores lógicos = cores físicos
16384	0.000103411	0.000020691	0.000063377
32768	0.000192207	0.000221795	0.000113854
65536	0.000170040	0.000355809	0.000241421
131072	0.000343345	0.000177999	0.000342300
262144	0.001098176	0.000348668	0.000691996
524288	0.001755363	0.001321537	0.001502023
1048576	0.003688164	0.002601611	0.002683686
2097152	0.006049583	0.004367006	0.003615653
4194304	0.010985464	0.008026686	0.011222828
8388608	0.021325183	0.014602151	0.015618511
16777216	0.042316946	0.029365428	0.027662726
33554432	0.083761573	0.054184099	0.053513053
67108864	0.184244157	0.106660684	0.105896613

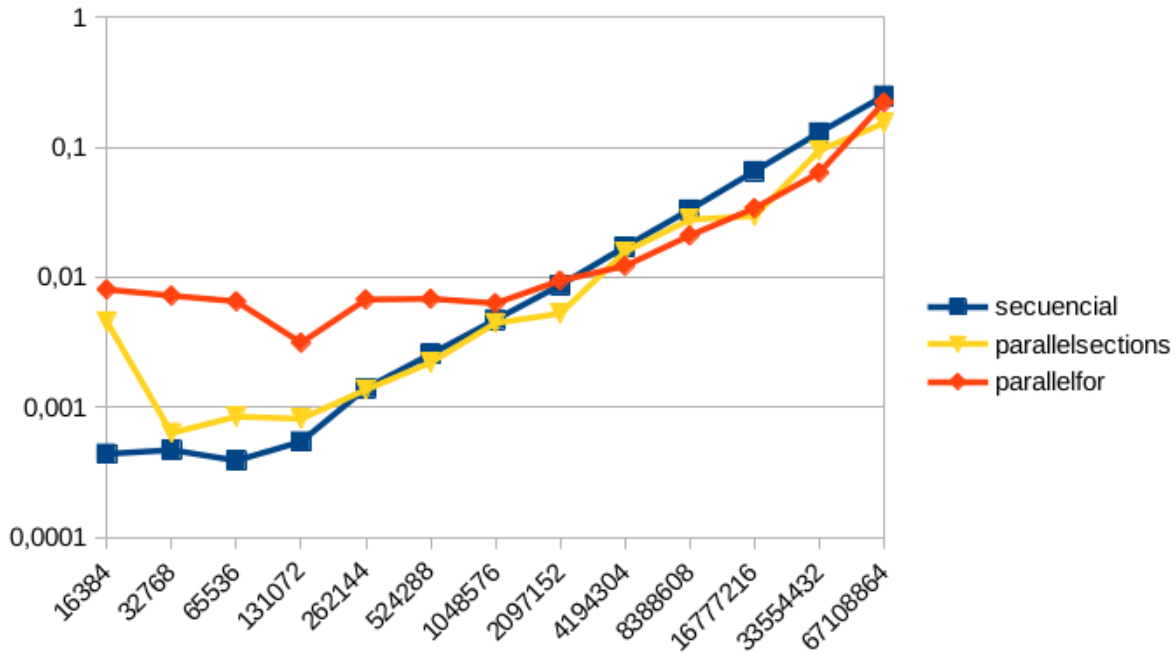
PARA ATCGRID:

Nº de Componentes	T. secuencial vect. Globales 1 thread=core	T. paralelo (versión for) 12 threads = cores lógicos = cores físicos	T. paralelo (versión sections) 12 threads = cores lógicos = cores físicos
16384	0.000437267	0.008041292	0.004564829
32768	0.000468730	0.007218782	0.000633370
65536	0.000388674	0.006528758	0.000846434
131072	0.000543215	0.003131967	0.000814691
262144	0.001400244	0.006734565	0.001363307
524288	0.002561063	0.006833177	0.002205960
1048576	0.004719868	0.006305639	0.004434153
2097152	0.008690762	0.009416416	0.005259294
4194304	0.017157876	0.012201764	0.015688993
8388608	0.033072116	0.020933412	0.027820121
16777216	0.065456833	0.033943541	0.029576942
33554432	0.129765270	0.063719124	0.094521433
67108864	0.248747145	0.221397992	0.153732225

GRÁFICA PARA PC:

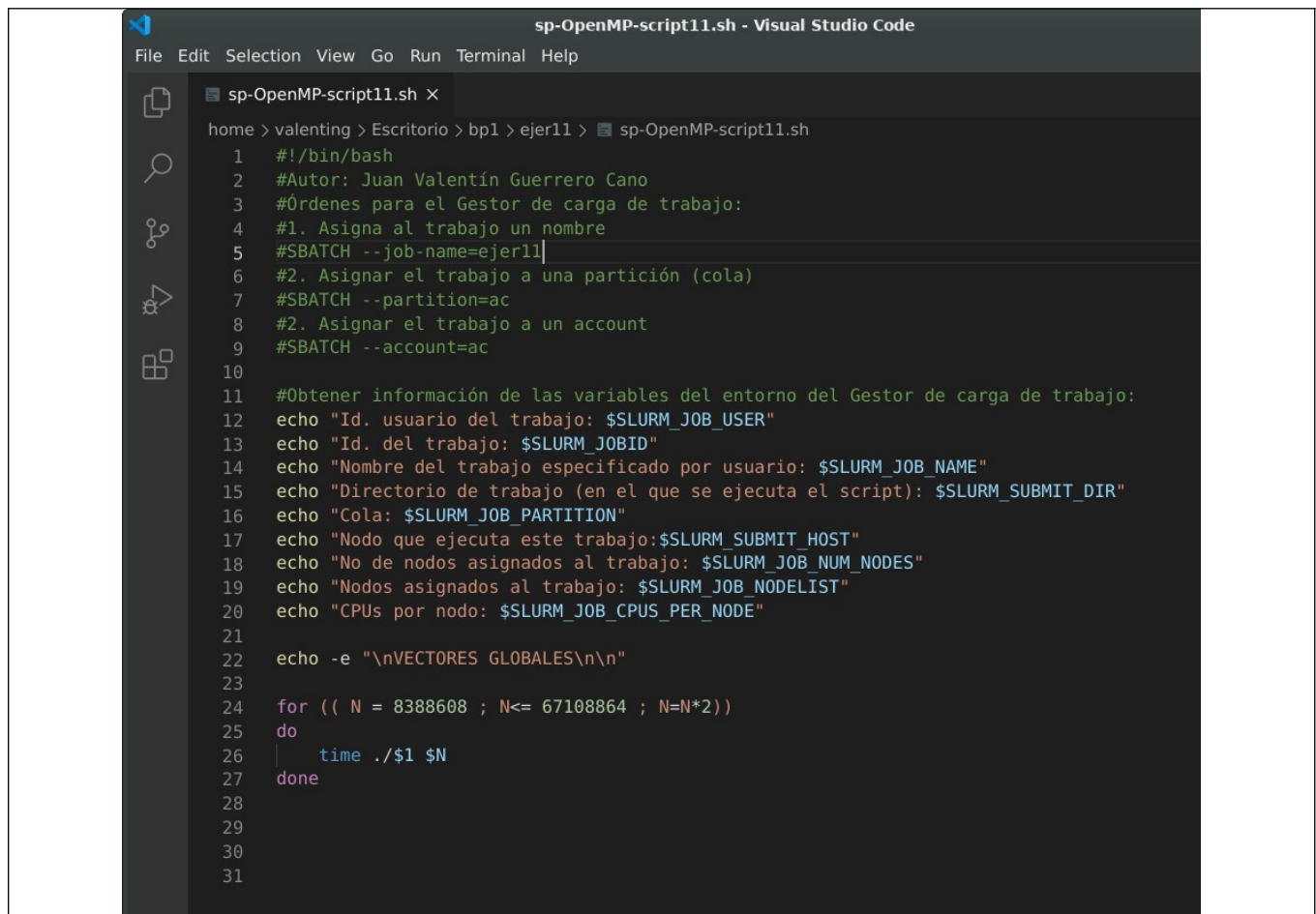


GRÁFICA PARA ATCGRID:



11. Rellenar una tabla como la 21Tabla 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`



```

sp-OpenMP-script11.sh - Visual Studio Code
File Edit Selection View Go Run Terminal Help

home > valenting > Escritorio > bp1 > ejer11 > sp-OpenMP-script11.sh
1  #!/bin/bash
2  #Autor: Juan Valentín Guerrero Cano
3  #Ordenes para el Gestor de carga de trabajo:
4  #1. Asigna al trabajo un nombre
5  #SBATCH --job-name=ejer11
6  #2. Asignar el trabajo a una partición (cola)
7  #SBATCH --partition=ac
8  #2. Asignar el trabajo a un account
9  #SBATCH --account=ac
10
11 #Obtener información de las variables del entorno del Gestor de carga de trabajo:
12 echo "Id. usuario del trabajo: $SLURM_JOB_USER"
13 echo "Id. del trabajo: $SLURM_JOBID"
14 echo "Nombre del trabajo especificado por usuario: $SLURM_JOB_NAME"
15 echo "Directorio de trabajo (en el que se ejecuta el script): $SLURM_SUBMIT_DIR"
16 echo "Cola: $SLURM_JOB_PARTITION"
17 echo "Nodo que ejecuta este trabajo:$SLURM_SUBMIT_HOST"
18 echo "No de nodos asignados al trabajo: $SLURM_JOB_NUM_NODES"
19 echo "Nodos asignados al trabajo: $SLURM_JOB_NODELIST"
20 echo "CPUs por nodo: $SLURM_JOB_CPUS_PER_NODE"
21
22 echo -e "\nVECTORES GLOBALES\n\n"
23
24 for (( N = 8388608 ; N<= 67108864 ; N=N*2))
25 do
26     time ./$1 $N
27 done
28
29
30
31

```

(RECUERDE ADJUNTAR LOS CÓDIGOS AL .ZIP)

CAPTURAS DE PANTALLA (ejecución en atcgrid):

Secuencial:

```

e3estudiante11@atcgrid:~/bp1/ejer11
Archivo Editar Ver Buscar Terminal Ayuda

[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer11] 2021-04-13 martes
Ssrun -pac -Aac --hint=nomultithread -c12 ./*.sh sumavectores1
Id. usuario del trabajo: e3estudiante11
Id. del trabajo: 90133
Id. usuario del trabajo: e3estudiante11
Id. del trabajo: 90133
Nombre del trabajo especificado por usuario: sp-OpenMP-script11.sh
Nombre del trabajo especificado por usuario: sp-OpenMP-script11.sh
Directorio de trabajo (en el que se ejecuta el script): /home/e3estudiante11/bp1/ejer11
Directorio de trabajo (en el que se ejecuta el script): /home/e3estudiante11/bp1/ejer11
Cola: ac
Cola: ac
Nodo que ejecuta este trabajo:atcgrid.ugr.es
No de nodos asignados al trabajo: 1
Nodo que ejecuta este trabajo:atcgrid.ugr.es
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24
No de nodos asignados al trabajo: 1

VECTORES GLOBALES

Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

VECTORES GLOBALES

Tiempo(seg.):0.046471811 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](1.523594+2.696130=4.219725) / /V1[8388607]+V2[8388607]=V3[8388607](2.221635+1.038037=3.259672)/
real 0m0.510s
user 0m0.465s
sys 0m0.042s
Tiempo(seg.):0.047327877 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](1.523594+2.696130=4.219725) / /V1[8388607]+V2[8388607]=V3[8388607](2.221635+1.038037=3.259672)/
real 0m0.515s
user 0m0.461s
sys 0m0.051s
Tiempo(seg.):0.070055475 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1.523594+2.696130=4.219725) / /V1[16777215]+V2[16777215]=V3[16777215](10.026885+0.939934=10.966819)/
real 0m0.935s

```

```

e3estudiante11@atcgrid:~/bp1/ejer11
Archivo Editar Ver Buscar Terminal Ayuda

Tiempo(seg.):0.047327877 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](1.523594+2.696130=4.219725) / /V1[8388607]+V2[8388607]=V3[8388607](2.221635+1.038037=3.259672)/
real 0m0.515s
user 0m0.461s
sys 0m0.051s
Tiempo(seg.):0.070055475 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1.523594+2.696130=4.219725) / /V1[16777215]+V2[16777215]=V3[16777215](10.026885+0.939934=10.966819)/
real 0m0.935s
user 0m0.867s
sys 0m0.067s
Tiempo(seg.):0.070345264 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1.523594+2.696130=4.219725) / /V1[16777215]+V2[16777215]=V3[16777215](10.026885+0.939934=10.966819)/
real 0m1.017s
user 0m0.941s
sys 0m0.075s
Tiempo(seg.):0.176214673 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](7.036775+0.031309=7.068083) / /V1[33554431]+V2[33554431]=V3[33554431](9.016123+1.373561=10.389684)/
real 0m1.838s
user 0m1.679s
sys 0m0.158s
Tiempo(seg.):0.170779641 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](7.036775+0.031309=7.068083) / /V1[33554431]+V2[33554431]=V3[33554431](9.016123+1.373561=10.389684)/
real 0m1.816s
user 0m1.660s
sys 0m0.155s
Tiempo(seg.):0.313310886 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](0.369091+0.532357=0.901448) / /V1[67108863]+V2[67108863]=V3[67108863](0.756153+0.777922=1.534076)/
real 0m3.566s
user 0m3.246s
sys 0m0.318s
Tiempo(seg.):0.308180911 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](0.369091+0.532357=0.901448) / /V1[67108863]+V2[67108863]=V3[67108863](0.756153+0.777922=1.534076)/
real 0m3.564s
user 0m3.269s
sys 0m0.293s
[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer11] 2021-04-13 martes
$

```

Parallel for:

```

e3estudiante11@atcgrid:~/bp1/ejer11
Archivo Editar Ver Buscar Terminal Ayuda

[JuanValentinGuerreroCano e3estudiante11@atcgrid:~/bp1/ejer11] 2021-04-13 martes
$ srun -pac -Aac --hint=nomultithread -c12 ./*.sh sumavectores7
Id. usuario del trabajo: e3estudiante11
Id. del trabajo: 90484
Nombre del trabajo especificado por usuario: sp-OpenMP-script11.sh
Directorio de trabajo (en el que se ejecuta el script): /home/e3estudiante11/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

VECTORES GLOBALES

Id. usuario del trabajo: e3estudiante11
Id. del trabajo: 90484
Nombre del trabajo especificado por usuario: sp-OpenMP-script11.sh
Directorio de trabajo (en el que se ejecuta el script): /home/e3estudiante11/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

VECTORES GLOBALES

Tiempo(seg.):0.029765580      / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](2.451264+0.640924=3.092188) / /V1[8388607]+V2[8388607]=V3[8388607](0.648796+2.123828=2.77
2624)/

real      0m4.555s
user      0m6.384s
sys       0m16.920s
Tiempo(seg.):0.030869324      / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](2.451264+107.896226=110.347490) / /V1[8388607]+V2[8388607]=V3[8388607](3.296098+0.563849=
3.859947)/

real      0m4.761s
user      0m6.814s
sys       0m21.490s
Tiempo(seg.):0.059600111      / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](0.022620+1.477983=1.500603) / /V1[16777215]+V2[16777215]=V3[16777215](39.358333+1.212901=
40.571233)/

real      0m8.830s

```

```

e3estudiante11@atcgrid:~/bp1/ejer11
Archivo Editar Ver Buscar Terminal Ayuda

VECTORES GLOBALES

Tiempo(seg.):0.029765580      / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](2.451264+0.640924=3.092188) / /V1[8388607]+V2[8388607]=V3[8388607](0.648796+2.123828=2.77
2624)/

real      0m4.555s
user      0m6.384s
sys       0m16.920s
Tiempo(seg.):0.030869324      / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](2.451264+107.896226=110.347490) / /V1[8388607]+V2[8388607]=V3[8388607](3.296098+0.563849=
3.859947)/

real      0m4.761s
user      0m6.814s
sys       0m21.490s
Tiempo(seg.):0.059600111      / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](0.022620+1.477983=1.500603) / /V1[16777215]+V2[16777215]=V3[16777215](39.358333+1.212901=
40.571233)/

real      0m8.830s
user      0m12.166s
sys       0m32.252s
Tiempo(seg.):0.054848794      / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](1.520806+10.826125=12.346931) / /V1[16777215]+V2[16777215]=V3[16777215](2.193322+2.325115
=4.518437)/

real      0m9.520s
user      0m13.936s
sys       0m42.144s
Tiempo(seg.):0.115806889      / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](2.331417+0.092494=2.423911) / /V1[33554431]+V2[33554431]=V3[33554431](0.611858+1.522503=2
.134361)/

real      0m17.950s
user      0m24.901s
sys       1m6.500s
Tiempo(seg.):0.115659546      / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](0.108165+2.208581=2.316746) / /V1[33554431]+V2[33554431]=V3[33554431](0.222300+3.178585=3
.400884)/

real      0m19.152s
user      0m27.493s
sys       1m27.019s
Tiempo(seg.):0.226885553      / Tamaño Vectores:67108864     / V1[0]+V2[0]=V3[0](0.850146+0.171944=1.022090) / /V1[67108863]+V2[67108863]=V3[67108863](3.289885+0.717957=4
.007842)/

real      0m35.722s
user      0m49.195s
sys       2m12.761s

```

No entiendo muy bien por qué se me ejecutan dos veces cada una de las iteraciones del vector, aún así escogeremos para realizar la tabla la primera iteración para cada una de las componentes.

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 “¿?” 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	0.510s	0.465s	0.042s	4.555s	6.384s	16.920s
16777216	0.935s	0.867s	0.067s	8.830s	12.166s	32.252s
33554432	1.838s	1.679s	0.158s	17.950s	24.901s	1m6.500s
67108864	3.566s	3.246s	0.318s	35.722s	49.195s	2m12.761s

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 sumar lo no suma el tiempo transcurrido realmente ya que son simultáneos y no secuenciales.