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): Paula Ruiz García
Grupo de prácticas: D1
Fecha de entrega: 01/04/2018
Fecha evaluación en clase: 09/04/2018

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

```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

```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();
        }

   return 0;
}
```

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 `single`Modificado.c

```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 ("Resultados de la hebra %d\n", omp_get_thread_num());
      for (i=0; i<n; i++)   printf("b[%d] = %d\t",i,b[i]);
      printf("\n");
    }
  }
return 0;
}
```

CAPTURAS DE PANTALLA:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$gcc -O2 -fopenmp single.c -o single
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$export OMP_DYNAMIC=FALSE
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$export OMP_NUM_THREADS=8
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$./single
Introduce valor de inicialización a: 23
Single ejecutada por el thread 7
Resultados de la hebra 1
b[0] = 23       b[1] = 23       b[2] = 23       b[3] = 23       b[4] = 23       b[5] = 23       b[6] = 23       b[7]
= 23    b[8] = 23
```

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 `single`Modificado2.c

```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 ("Resultados de la hebra: %d\n", omp_get_thread_num());
      for (i=0; i<n; i++)   printf("b[%d] = %d\t",i,b[i]);
      printf("\n");
    }
  }
return 0;
}
```

CAPTURAS DE PANTALLA:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$gcc -O2 -fopenmp singleModificado2.c -o master
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$export OMP_NUM_THREADS=3
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$./master
Introduce valor de inicialización a: 6
Single ejecutada por el thread 0
Resultados de la hebra master 0
b[0] = 6        b[1] = 6        b[2] = 6        b[3] = 6        b[4] = 6        b[5] = 6        b[6] = 6        b[7]
= 6     b[8] = 6
```

RESPUESTA A LA PREGUNTA:

Al mostrar la respuesta con la directiva master los resultados siempre los pintara la hebra 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:

Porque la directiva `barrier` hace que las hebras se esperen entre si, entonces si eliminamos esta directiva podria ocurrir que la hebra 0 se ejecutara mas rapido, lo que provocaria que imprimiera los resultados sin haber esperado que terminasen los otros hilos.

Resto de ejercicios

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

   RESPUESTA:

   La suma de los tiempos de usuario y sistema es menor que el tiempo real.

   CAPTURAS DE PANTALLA:

   PC:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$gcc -O2 SumaVectores.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$cat STDIN.o69523
Tiempo(seg.):0.060303090         / Tamaño Vectores:10000000     / V1[0]+V2[0]=V3[0](1000000.000000+1000000.000000=200
0000.000000) V1[9999999]+V2[9999999]=V3[9999999](1999999.900000+0.100000=2000000.000000) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$cat STDIN.e69523

real    0m0.173s
user    0m0.049s
sys     0m0.120s
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-29 Thursday
$
```

   sftp:

```
sftp> put SumaVectores
Uploading SumaVectores to /home/D1estudiante18/SumaVectores
SumaVectores                                                  100% 8968     8.8KB/s   00:00
sftp> get STDIN.*
Fetching /home/D1estudiante18/STDIN.e69523 to STDIN.e69523
/home/D1estudiante18/STDIN.e69523                             100%   42     0.0KB/s   00:00
Fetching /home/D1estudiante18/STDIN.o69523 to STDIN.o69523
/home/D1estudiante18/STDIN.o69523                             100%  198     0.2KB/s   00:00
sftp> ls
STDIN.e69523  STDIN.o69523  SumaVectores
sftp>
```

   ssh:

```
[D1estudiante18@atcgrid ~]$ echo 'time ./SumaVectores 10000000' | qsub -q ac
69523.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
69523.atcgrid              STDIN            D1estudiante18  00:00:00 C ac
[D1estudiante18@atcgrid ~]$ ls
STDIN.e69523  STDIN.o69523  SumaVectores
[D1estudiante18@atcgrid ~]$
```

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). Incorpore **el código ensamblador de la parte de la suma de vectores** en el cuaderno.

   CAPTURAS DE PANTALLA:

   TAM = 10

   cat

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$cat STDIN.o70059
Tiempo(seg.):0.000002813         / Tamaño Vectores:10   / V1[0]+V2[0]=V3[0](1.000000+1.000000=2.000000) V1[9]+V2[9]=V
3[9](1.900000+0.100000=2.000000) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$cat STDIN.e70059

real    0m0.002s
user    0m0.000s
sys     0m0.001s
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$
```

sftp

```
sftp> get STDIN.*
Fetching /home/D1estudiante18/STDIN.e70059 to STDIN.e70059
/home/D1estudiante18/STDIN.e70059                                    100%   42     0.0KB/s   00:00
Fetching /home/D1estudiante18/STDIN.o70059 to STDIN.o70059
/home/D1estudiante18/STDIN.o70059                                    100%  144     0.1KB/s   00:01
sftp>
```

ssh

```
[D1estudiante18@atcgrid ~]$ echo 'time ./SumaVectores 10' | qsub -q ac
70059.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
70059.atcgrid              STDIN            D1estudiante18  00:00:00 C ac
[D1estudiante18@atcgrid ~]$ ls
STDIN.e70059  STDIN.o70059  SumaVectores
[D1estudiante18@atcgrid ~]$
```

TAM = 10000000

cat

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$cat STDIN.o70060
Tiempo(seg.):0.068611082         / Tamaño Vectores:10000000     / V1[0]+V2[0]=V3[0](1000000.000000+1000000.000000=200
0000.000000) V1[9999999]+V2[9999999]=V3[9999999](1999999.900000+0.100000=2000000.000000) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$cat STDIN.e70060

real    0m0.189s
user    0m0.079s
sys     0m0.107s
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$
```

sftp

```
sftp> get STDIN.o70060
Fetching /home/D1estudiante18/STDIN.o70060 to STDIN.o70060
/home/D1estudiante18/STDIN.o70060                                    100%  198     0.2KB/s   00:00
sftp> get STDIN.e70060
Fetching /home/D1estudiante18/STDIN.e70060 to STDIN.e70060
/home/D1estudiante18/STDIN.e70060                                    100%   42     0.0KB/s   00:00
```

ssh

```
[D1estudiante18@atcgrid ~]$ echo 'time ./SumaVectores 10000000' | qsub -q ac
70060.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
70060.atcgrid              STDIN            D1estudiante18  00:00:00 C ac
[D1estudiante18@atcgrid ~]$ ls
STDIN.e70059  STDIN.e70060  STDIN.o70059  STDIN.o70060  SumaVectores
```

RESPUESTA: cálculo de los MIPS y los MFLOPS

Tamaño del vector	10	10000000
Nº Instrucciones	65 = 5 + (6*10)	60000005 = 5 + (6*10000000)
Instrucciones en Coma Flotante	10 = 1*10	10000000 = 1*10000000

Tiempo	0.002s	0.189s
MIPS	65/(0.002*10^6) = 0.0325	60000005/(0.189*10^6) = 317.4603439
MFLOPS	10/(0.002*10^6) = 5*10^(-3)	10000000/(0.189*10^6) = 52.91005291

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

```
70    call  clock_gettime
71    xorl  %eax, %eax
72    .p2align 4,,10
73    .p2align 3
74  .L5:
75    movsd v1(%rax), %xmm0
76    addq  $8, %rax
77    addsd v2-8(%rax), %xmm0
78    movsd %xmm0, v3-8(%rax)
79    cmpq  %rax, %rbx
80    jne .L5
81  .L6:
82    leaq  16(%rsp), %rsi
83    xorl  %edi, %edi
84    call  clock_gettime
```

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

RESPUESTA: Captura que muestre el código fuente implementado

```c
#include <stdlib.h> // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h> // biblioteca donde se encuentra la función printf()
#include <time.h>   // biblioteca donde se encuentra la función clock_gettime()

#ifdef _OPENMP
  #include <omp.h>
#else
  #define omp_get_thread_num() 0
#endif
//#define PRINTF_ALL  // comentar para quitar el printf que imprime todos los componentes

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

  int i;
  double cgt1,cgt2;
  double ncgt; //para tiempo de ejecución

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

  unsigned int N = atoi(argv[1]); // Máximo N =2^32-1=4294967295 (sizeof(unsigned int) = 4 B)
                                  // disponible en C a partir de actualización C99
  double *v1, *v2, *v3;
  v1 = (double*) malloc(N*sizeof(double));// malloc necesita el tamaño en bytes
  v2 = (double*) malloc(N*sizeof(double)); //si no hay espacio suficiente malloc devuelve NULL
  v3 = (double*) malloc(N*sizeof(double));
    if ( (v1==NULL) || (v2==NULL) || (v3==NULL) ){
    printf("Error en la reserva de espacio para los vectores\n");
    exit(-2);
  }
  #pragma omp parallel for
    for(i=0; i<N; i++){
        v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //Los valores dependen de N
      }
    cgt1 = omp_get_wtime();
  #pragma omp parallel for
    for(i=0; i<N; i++){
      v3[i] = v1[i] + v2[i];
    }
    cgt2 = omp_get_wtime();
  ncgt= cgt2 - cgt1;

  //Imprimir resultado de la suma y el tiempo de ejecución
  #ifdef PRINTF_ALL
  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,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]);
  #endif

  free(v1); // libera el espacio reservado para v1
  free(v2); // libera el espacio reservado para v2
  free(v3); // libera el espacio reservado para v3
  return 0;
}
```

(RECUERDE ADJUNTAR CÓDIGO FUENTE AL .ZIP)

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

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$gcc -O2 -fopenmp SumaVectoresEj7.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$./SumaVectores 8
Tiempo(seg.):0.000004000          / Tamaño Vectores:8    / V1[0]+V2[0]=V3[0](0.800000+0.800000=1.600000) V1[7]+V2[7]=V
3[7](1.500000+0.100000=1.600000) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$./SumaVectores 11
Tiempo(seg.):0.000005000          / 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) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$
```

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 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

```c
#include <stdlib.h> // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h> // biblioteca donde se encuentra la función printf()
#include <time.h>   // biblioteca donde se encuentra la función clock_gettime()

#ifdef _OPENMP
  #include <omp.h>
#else
  #define omp_get_thread_num() 0
#endif
//#define PRINTF_ALL  // comentar para quitar el printf que imprime todos los componentes

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

  int i;
  double cgt1,cgt2;
  double ncgt; //para tiempo de ejecución

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

  unsigned int N = atoi(argv[1]); // Máximo N =2^32-1=4294967295 (sizeof(unsigned int) = 4 B)
                                  // disponible en C a partir de actualización C99
  double *v1, *v2, *v3;
  v1 = (double*) malloc(N*sizeof(double));// malloc necesita el tamaño en bytes
  v2 = (double*) malloc(N*sizeof(double)); //si no hay espacio suficiente malloc devuelve NULL
  v3 = (double*) malloc(N*sizeof(double));
    if ( (v1==NULL) || (v2==NULL) || (v3==NULL) ){
    printf("Error en la reserva de espacio para los vectores\n");
    exit(-2);
  }
  #pragma omp parallel private(i)
  {
    #pragma omp sections
      {
       #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;
       }
      }
    #pragma omp single
    {
      cgt1 = omp_get_wtime();
    }
    #pragma omp 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];

      }
    #pragma omp single
    {
      cgt2 = omp_get_wtime();
    }
  }
  ncgt= cgt2 - cgt1;

  //Imprimir resultado de la suma y el tiempo de ejecución
  #ifdef PRINTF_ALL
  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,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]);
  #endif

  free(v1); // libera el espacio reservado para v1
  free(v2); // libera el espacio reservado para v2
  free(v3); // libera el espacio reservado para v3
  return 0;
}
```

(RECUERDE ADJUNTAR CÓDIGO FUENTE AL .ZIP)

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

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$gcc -O2 -fopenmp SumaVectoresEj8.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$./SumaVectores 8
Tiempo(seg.):0.000003000        / Tamaño Vectores:8    / V1[0]+V2[0]=V3[0](0.800000+0.800000=1.600000) V1[7]+V2[7]=V
3[7](1.500000+0.100000=1.600000) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$./SumaVectores 11
Tiempo(seg.):0.000003000        / 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) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-03-31 Saturday
$
```

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

RESPUESTA:

Ejercicio 7:

Para este ejercicio como no hemos definido la variable de entorno `OMP_NUM_THREADS` se utilizarán todos los `cores/threads` que tenga disponibles la máquina que estemos utilizando.

Ejercicio 8:

En este ejercicio ocurre lo mismo que en el ejercicio 7, la única diferencia es que, en este caso, al dividirlo en secciones fijas, habrá hebras que no realicen ningún trabajo, aunque como máximo el número de `cores/threads` será las que tenga disponible la maquina utilizada.

10. Rellenar una tabla como la Tabla 210 para atcgrid y otra para su PC con los tiempos de ejecución de los programas paralelos implementados en los ejercicios 7 y 8 y el programa secuencial del Listado 1. Generar los ejecutables usando -O2. En la tabla debe aparecer el tiempo de ejecución del trozo de código que realiza la suma en paralelo (este es el tiempo que deben imprimir los programas). Ponga en la tabla el número de threads/cores que usan los códigos. Represente en una gráfica los tres tiempos. NOTA: Nunca ejecute código que imprima todos los componentes del resultado cuando este número sea elevado.

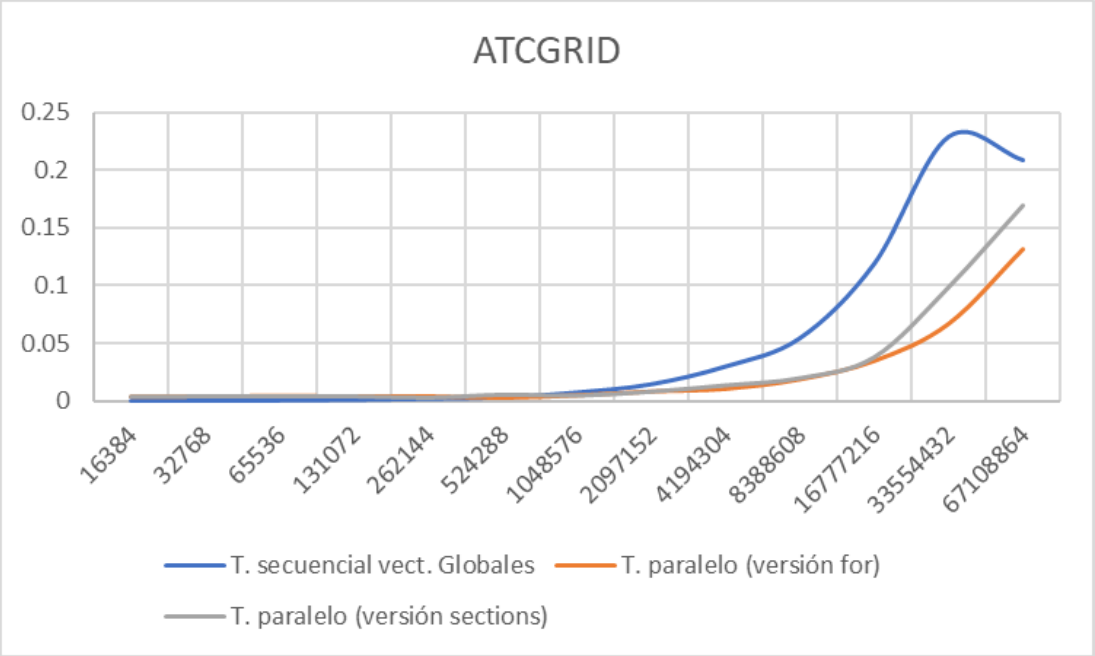
RESPUESTA:

Tabla 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 utilizados.

ATCGRID

Nº de Componentes	T. secuencial vect. Globales 1 thread/core	T. paralelo (versión for) 24 threads/ 12 cores	T. paralelo (versión sections) 24 threads/ 12 cores
16384	0.000109295	0.003628514	0.003831479
32768	0.000132056	0.003991864	0.004169485
65536	0.000432580	0.004351563	0.004232457
131072	0.000916586	0.004064233	0.004153133
262144	0.001859043	0.004015537	0.003017462
524288	0.003090178	0.002723464	0.005554617
1048576	0.007355660	0.004844075	0.004742870
2097152	0.014539068	0.008204784	0.008273282

4194304	0.029920417	0.010436159	0.013658007
8388608	0.054520638	0.018717208	0.019876556
16777216	0.119134173	0.034661092	0.037964507
33554432	0.229320361	0.066802821	0.098484611
67108864	0.209037665	0.131385369	0.169065687



PANTALLAZOS:

Globales:

Cat

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$gcc -O2 SumaVectores.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.o70409
Tiempo(seg.):0.000109295         / 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.):0.000132056         / 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.):0.000432580         / 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.):0.000916586         / 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.):0.001859043         / 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.):0.003090178         / 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.):0.007355660         / 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.):0.014539068         / 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.):0.029920417         / 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.):0.054520638         / 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.):0.119134173         / 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.):0.229320361         / 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.):0.209037665         / 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) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.e70409
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$
```

Ssh

```
[D1estudiante18@atcgrid ~]$ qsub SumadorVectores.sh -q ac
70409.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
70409.atcgrid              ...resC_vlocales D1estudiante18         0 Q ac
[D1estudiante18@atcgrid ~]$ ls
SumadorVectores.sh  SumaVectores  SumaVectoresC_vlocales.e70409  SumaVectoresC_vlocales.o70409
[D1estudiante18@atcgrid ~]$
```

sftp

```
sftp> put SumadorVectores.sh
Uploading SumadorVectores.sh to /home/D1estudiante18/SumadorVectores.sh
SumadorVectores.sh                                            100%  739     0.0KB/s   00:26
sftp> put SumaVectores
Uploading SumaVectores to /home/D1estudiante18/SumaVectores
SumaVectores                                                  100% 8968     8.8KB/s   00:00
sftp> ls
SumaVectores           SumadorVectores.sh
sftp> get SumaVectoresC*
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.e70409 to SumaVectoresC_vlocales.e70409
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.o70409 to SumaVectoresC_vlocales.o70409
/home/D1estudiante18/SumaVectoresC_vlocales.o70409             100% 2451     2.4KB/s   00:00
sftp>
```

Parallel-for:

Cat

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$gcc -O2 -fopenmp SumaVectoresEj7.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.o70411
Tiempo(seg.):0.003628514         / 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.):0.003991864         / 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.):0.004351563         / 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.):0.004064233         / 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.):0.004015537         / 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.):0.002723464         / 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.):0.004844075         / 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.):0.008204784         / 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.):0.010436159         / 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.):0.018717208         / 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.):0.034661092         / 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.):0.066802821         / 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.):0.131385369         / 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) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.e70411
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$
```

Ssh

```
[D1estudiante18@atcgrid ~]$ qsub SumadorVectores.sh -q ac
70411.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
70411.atcgrid              ...resC_vlocales D1estudiante18         0 R ac
[D1estudiante18@atcgrid ~]$ ls
SumadorVectores.sh  SumaVectores  SumaVectoresC_vlocales.e70411  SumaVectoresC_vlocales.o70411
[D1estudiante18@atcgrid ~]$
```

Sftp

```
sftp> put SumaVectores
Uploading SumaVectores to /home/D1estudiante18/SumaVectores
SumaVectores                                                  100%   13KB  13.0KB/s   00:01
sftp> get SumaVectoresC*
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.e70411 to SumaVectoresC_vlocales.e70411
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.o70411 to SumaVectoresC_vlocales.o70411
/home/D1estudiante18/SumaVectoresC_vlocales.o70411             100% 2454     2.4KB/s   00:01
sftp>
```

Parallel-Sections:

Cat

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$gcc -O2 -fopenmp SumaVectoresEj8.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.o70412
Tiempo(seg.):0.003831479         / 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.):0.004169485         / 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.):0.004232457         / 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.):0.004153133         / 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.):0.003017462         / 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.):0.005554617         / 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.):0.004742870         / 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.):0.008273282         / 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.):0.013658007         / 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.):0.019876556         / 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.):0.037964507         / 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.):0.098484611         / 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.):0.169065687         / 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) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.e70412
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$
```

Ssh

```
[D1estudiante18@atcgrid ~]$ qsub SumadorVectores.sh -q ac
70412.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
70412.atcgrid              ...resC_vlocales D1estudiante18         0 Q ac
[D1estudiante18@atcgrid ~]$ ls
SumadorVectores.sh  SumaVectores
[D1estudiante18@atcgrid ~]$ ls
SumadorVectores.sh  SumaVectores  SumaVectoresC_vlocales.e70412  SumaVectoresC_vlocales.o70412
```

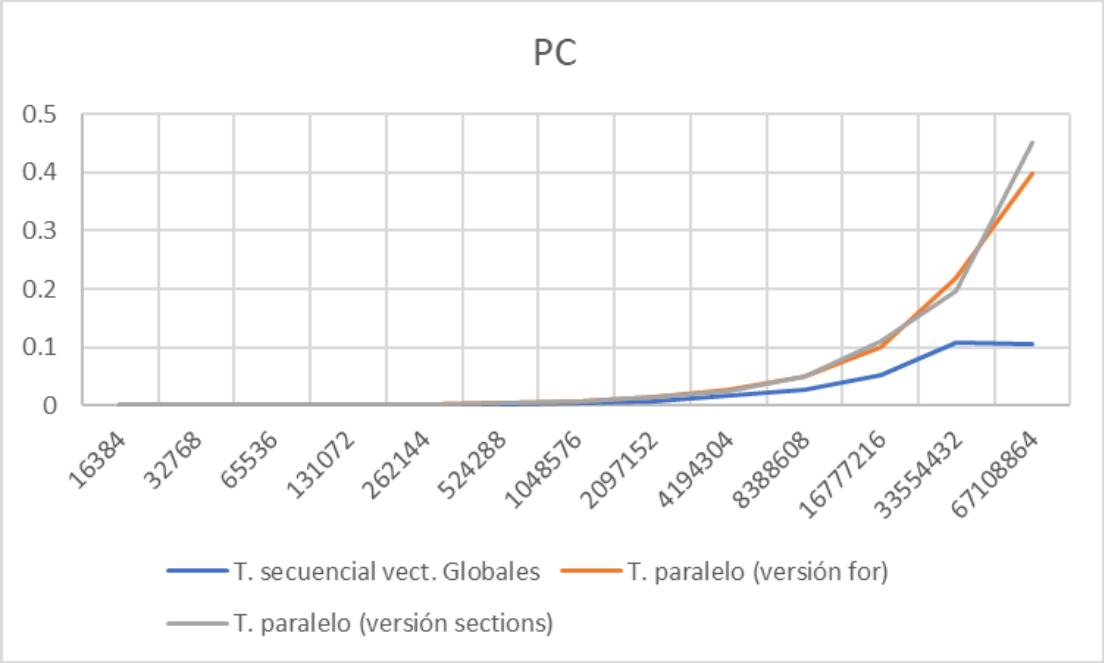
Sftp

```
sftp> put SumaVectores
Uploading SumaVectores to /home/D1estudiante18/SumaVectores
SumaVectores                                                  100%   13KB  13.2KB/s   00:01
sftp> ls
SumaVectores                        SumaVectoresC_vlocales.e70412         SumaVectoresC_vlocales.o70412
SumadorVectores.sh
sftp> get SumaVectoresC*
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.e70412 to SumaVectoresC_vlocales.e70412
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.o70412 to SumaVectoresC_vlocales.o70412
/home/D1estudiante18/SumaVectoresC_vlocales.o70412                 100% 2454     2.4KB/s   00:00
sftp>
```

PC

Nº de Componentes	T. secuencial vect. Globales 1 thread/core	T. paralelo (versión for) ¿?threads/cores	T. paralelo (versión sections) ¿?threads/cores
16384	0.000080100	0.000186000	0.000151000
32768	0.000166100	0.000215000	0.000239000
65536	0.000299800	0.000474000	0.000401000
131072	0.000497300	0.000989000	0.001006000
262144	0.000883900	0.001649000	0.001683000
524288	0.001750100	0.003946000	0.003832000
1048576	0.003425400	0.006499000	0.006948000
2097152	0.006919900	0.012941000	0.013288000

4194304	0.016005300	0.025729000	0.025105000
8388608	0.026763300	0.049268000	0.048833000
16777216	0.053169800	0.099658000	0.109647000
33554432	0.106675400	0.218542000	0.195592000
67108864	0.105540700	0.398799000	0.451608000



PANTALLAZOS:

Globales

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$gcc -O2 SumaVectores.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$bash SumadorVectores.sh
Tiempo(seg.):0.000080100         / 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.):0.000166100         / 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.):0.000299800         / 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.):0.000497300         / 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.):0.000883900         / 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.):0.001750100         / 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.):0.003425400         / 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.):0.006919900         / 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.):0.016005300         / 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.):0.026763300         / 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.):0.053169800         / 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.):0.106675400         / 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.):0.105540700         / 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) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$
```

Parallel-for

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$gcc -O2 -fopenmp SumaVectoresEj7.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$bash SumadorVectores.sh
Tiempo(seg.):0.000186000         / Tamaño Vectores:16384        / V1[0]+V2[0]=V3[0](1638.400000+1638.400000=3276.8000
00) V1[16383]+V2[16383]=V3[16383](3276.700000+0.100000=3276.800000) /
Tiempo(seg.):0.000215000         / Tamaño Vectores:32768        / V1[0]+V2[0]=V3[0](3276.800000+3276.800000=6553.6000
00) V1[32767]+V2[32767]=V3[32767](6553.500000+0.100000=6553.600000) /
Tiempo(seg.):0.000474000         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200
000) V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.000989000         / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.4
00000) V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.001649000         / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.8
00000) V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.003946000         / 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.):0.006499000         / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=20971
5.200000) V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.012941000         / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=41943
0.400000) V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.025729000         / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=83886
0.800000) V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.049268000         / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=16777
21.600000) V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo(seg.):0.099658000         / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=335
5443.200000) V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
Tiempo(seg.):0.218542000         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=671
0886.400000) V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
Tiempo(seg.):0.398799000         / Tamaño Vectores:67108864     / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=134
21772.800000) V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$
```

Parallel-Sections

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$gcc -O2 -fopenmp SumaVectoresEj8.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$bash SumadorVectores.sh
Tiempo(seg.):0.000151000         / Tamaño Vectores:16384        / V1[0]+V2[0]=V3[0](1638.400000+1638.400000=3276.8000
00) V1[16383]+V2[16383]=V3[16383](3276.700000+0.100000=3276.800000) /
Tiempo(seg.):0.000239000         / Tamaño Vectores:32768        / V1[0]+V2[0]=V3[0](3276.800000+3276.800000=6553.6000
00) V1[32767]+V2[32767]=V3[32767](6553.500000+0.100000=6553.600000) /
Tiempo(seg.):0.000401000         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200
000) V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.001006000         / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.4
00000) V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.001683000         / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.8
00000) V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.003832000         / 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.):0.006948000         / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=20971
5.200000) V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.013288000         / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=41943
0.400000) V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.025105000         / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=83886
0.800000) V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.048833000         / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=16777
21.600000) V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo(seg.):0.109647000         / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=335
5443.200000) V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
Tiempo(seg.):0.195592000         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=671
0886.400000) V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
Tiempo(seg.):0.451608000         / Tamaño Vectores:67108864     / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=134
21772.800000) V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$
```

11. Rellenar una tabla como la 16Tabla 3 para atcgrid con el tiempo de ejecución, tiempo de CPU del usuario y tiempo CPU del sistema obtenidos con time para el ejecutable del ejercicio 7 y para el programa secuencial del Listado 1. Ponga en la tabla el número de threads/cores que usan los códigos. ¿El tiempo de CPU que se obtiene es mayor o igual que el tiempo real (*elapsed*)? Justifique la respuesta.

    RESPUESTA: El tiempo de CPU es parecido o igual al tiempo real ya que solo se esta usando un procesador.

    En la versión paralela el tiempo de CPU es mayor que el real, esto se debe a que el tiempo de CPU es la suma de los tiempos de cada núcleo, mientras que el tiempo real es el tiempo que ha tardado, realmente, en ejecutarse el programa.

Nº de Componentes	Tiempo secuencial vect. Globales 1 thread/core			Tiempo paralelo/versión for 2 Threads/ 2 cores		
	Elapsed	*CPU-user*	*CPU- sys*	*Elapsed*	*CPU-user*	*CPU- sys*
65536	0m0.003s	0m0.001s	0m0.001s	0m0.017s	0m0.213s	0m0.002s
131072	0m0.004s	0m0.001s	0m0.003s	0m0.011s	0m0.168s	0m0.023s
262144	0m0.005s	0m0.001s	0m0.004s	0m0.010s	0m0.184s	0m0.018s
524288	0m0.013s	0m0.004s	0m0.009s	0m0.013s	0m0.218s	0m0.006s
1048576	0m0.017s	0m0.007s	0m0.010s	0m0.016s	0m0.232s	0m0.015s
2097152	0m0.043s	0m0.013s	0m0.029s	0m0.019s	0m0.270s	0m0.076s
4194304	0m0.084s	0m0.032s	0m0.051s	0m0.033s	0m0.336s	0m0.146s
8388608	0m0.157s	0m0.059s	0m0.097s	0m0.048s	0m0.433s	0m0.344s
16777216	0m0.311s	0m0.113s	0m0.194s	0m0.093s	0m0.706s	0m0.616s
33554432	0m0.606s	0m0.218s	0m0.382s	0m0.172s	0m1.215s	0m1.326s
67108864	0m0.603s	0m0.206s	0m0.392s	0m0.341s	0m2.226s	0m2.412s

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.

PANTALLAZOS:

Globales:

Cat

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$gcc -O2 SumaVectores.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.o70414
Tiempo(seg.):0.000469236         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200
000) V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.000801845         / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.4
00000) V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.001141799         / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.8
00000) V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.003819740         / 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.):0.005775767         / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=20971
5.200000) V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.014586706         / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=41943
0.400000) V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.030013567         / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=83886
0.800000) V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.056359758         / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=16777
21.600000) V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo(seg.):0.115928053         / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=335
5443.200000) V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
Tiempo(seg.):0.221317293         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=671
0886.400000) V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
Tiempo(seg.):0.217963527         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=671
0886.400000) V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
```

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.e70414

real    0m0.003s
user    0m0.001s
sys     0m0.001s

real    0m0.004s
user    0m0.001s
sys     0m0.003s

real    0m0.005s
user    0m0.001s
sys     0m0.004s

real    0m0.013s
user    0m0.004s
sys     0m0.009s

real    0m0.017s
user    0m0.007s
sys     0m0.010s

real    0m0.043s
user    0m0.013s
sys     0m0.029s

real    0m0.084s
user    0m0.032s
sys     0m0.051s

real    0m0.157s
user    0m0.059s
sys     0m0.097s

real    0m0.311s
user    0m0.113s
sys     0m0.194s

real    0m0.606s
user    0m0.218s
sys     0m0.382s

real    0m0.603s
user    0m0.206s
sys     0m0.392s
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$
```

Ssh

```
[D1estudiante18@atcgrid ~]$ qsub CalculadorTiempos.sh -q ac
70414.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
70414.atcgrid              ...resC_vlocales D1estudiante18         0 R ac
[D1estudiante18@atcgrid ~]$ ls
CalculadorTiempos.sh  SumaVectores  SumaVectoresC_vlocales.e70414  SumaVectoresC_vlocales.o70414
[D1estudiante18@atcgrid ~]$
```

Sftp

```
sftp> put CalculadorTiempos.sh
Uploading CalculadorTiempos.sh to /home/D1estudiante18/CalculadorTiempos.sh
CalculadorTiempos.sh                                          100%  744     0.7KB/s   00:00
sftp> put SumaVectores
Uploading SumaVectores to /home/D1estudiante18/SumaVectores
SumaVectores                                                  100% 8968     8.8KB/s   00:00
sftp> get SumaVectoresC*
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.e70414 to SumaVectoresC_vlocales.e70414
/home/D1estudiante18/SumaVectoresC_vlocales.e70414             100%  462     0.5KB/s   00:00
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.o70414 to SumaVectoresC_vlocales.o70414
/home/D1estudiante18/SumaVectoresC_vlocales.o70414             100% 2103     2.1KB/s   00:00
sftp>
```

Parallel-for:

Cat

```
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$gcc -O2 -fopenmp SumaVectoresEj7.c -o SumaVectores -lrt
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.o70416
Tiempo(seg.):0.005202846         / Tamaño Vectores:65536        / V1[0]+V2[0]=V3[0](6553.600000+6553.600000=13107.200
000) V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.004407540         / Tamaño Vectores:131072       / V1[0]+V2[0]=V3[0](13107.200000+13107.200000=26214.4
00000) V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.004036348         / Tamaño Vectores:262144       / V1[0]+V2[0]=V3[0](26214.400000+26214.400000=52428.8
00000) V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.005006007         / 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.):0.005740574         / Tamaño Vectores:1048576      / V1[0]+V2[0]=V3[0](104857.600000+104857.600000=20971
5.200000) V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.007027409         / Tamaño Vectores:2097152      / V1[0]+V2[0]=V3[0](209715.200000+209715.200000=41943
0.400000) V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.011992455         / Tamaño Vectores:4194304      / V1[0]+V2[0]=V3[0](419430.400000+419430.400000=83886
0.800000) V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.018780702         / Tamaño Vectores:8388608      / V1[0]+V2[0]=V3[0](838860.800000+838860.800000=16777
21.600000) V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000) /
Tiempo(seg.):0.036430617         / Tamaño Vectores:16777216     / V1[0]+V2[0]=V3[0](1677721.600000+1677721.600000=335
5443.200000) V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.200000) /
Tiempo(seg.):0.067156632         / Tamaño Vectores:33554432     / V1[0]+V2[0]=V3[0](3355443.200000+3355443.200000=671
0886.400000) V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.400000) /
Tiempo(seg.):0.134204283         / Tamaño Vectores:67108864     / V1[0]+V2[0]=V3[0](6710886.400000+6710886.400000=134
21772.800000) V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772.800000) /
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$cat SumaVectoresC_vlocales.e70416

real    0m0.017s
user    0m0.213s
sys     0m0.002s

real    0m0.011s
user    0m0.168s
sys     0m0.023s

real    0m0.010s
user    0m0.184s
sys     0m0.018s

real    0m0.013s
user    0m0.218s
sys     0m0.006s

real    0m0.016s
user    0m0.232s
sys     0m0.015s

real    0m0.019s
user    0m0.270s
sys     0m0.076s

real    0m0.033s
user    0m0.336s
sys     0m0.146s

real    0m0.048s
user    0m0.433s
sys     0m0.344s

real    0m0.093s
user    0m0.706s
sys     0m0.616s

real    0m0.172s
user    0m1.215s
sys     0m1.326s

real    0m0.341s
user    0m2.266s
sys     0m2.412s
[Paula Ruiz García Paula@Pompitas:~/AC/P1/Codigos] 2018-04-01 Sunday
$
```

Ssh

```
[D1estudiante18@atcgrid ~]$ qsub CalculadorTiempos.sh -q ac
70416.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
70416.atcgrid              ...resC_vlocales D1estudiante18  00:00:00 C ac
[D1estudiante18@atcgrid ~]$ ls
CalculadorTiempos.sh  SumaVectores  SumaVectoresC_vlocales.e70416  SumaVectoresC_vlocales.o70416
[D1estudiante18@atcgrid ~]$
```

Sftp

```
sftp> put SumaVectores
Uploading SumaVectores to /home/D1estudiante18/SumaVectores
SumaVectores                                                  100%   13KB  13.0KB/s   00:00
sftp> get SumaVectoresC*
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.e70416 to SumaVectoresC_vlocales.e70416
/home/D1estudiante18/SumaVectoresC_vlocales.e70416             100%  462     0.5KB/s   00:00
Fetching /home/D1estudiante18/SumaVectoresC_vlocales.o70416 to SumaVectoresC_vlocales.o70416
/home/D1estudiante18/SumaVectoresC_vlocales.o70416             100% 2106     2.1KB/s   00:00
sftp>
```