

Grai2º curso / 2º
cuatr.

Grado Ing. Inform.

Doble Grado Ing.
Inform. y Mat.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 2. Programación paralela II: Cláusulas OpenMP

Estudiante (nombre y apellidos):

Grupo de prácticas:

Fecha de entrega:

Fecha evaluación en clase:

Ejercicios basados en los ejemplos del seminario práctico

1. ¿Qué ocurre si en el ejemplo del seminario `shared-clause.c` se añade a la directiva `parallel` la cláusula `default(none)`? (añada una captura de pantalla que muestre lo que ocurre) **(b)** Resuelva el problema generado sin eliminar `default(none)`. Añada el código con la modificación al cuaderno de prácticas.

RESPUESTA:

Lo que ocurre al añadir `default(none)` es que el compilador te pide que especifiques el ámbito de las variables `n` y `a`.

CAPTURA CÓDIGO FUENTE: `shared-clauseModificado.c`

```
#include <stdio.h>
#ifdef _OPENMP
#include <omp.h>
#endif

int main(){
    int i, n = 7;
    int a[n];
    for (i=0; i<n; i++)
        a[i] = i+1;

    #pragma omp parallel for shared(a, n) default(none)
    for (i=0; i<n; i++){
        a[i] += i;
    }
    printf("Después de parallel for:\n");
    for (i=0; i<n; i++)
        printf("a[%d] = %d\n", i, a[i]);
}
```

CAPTURAS DE PANTALLA:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp shared-clauseModificado.c -o shared-clauseM
shared-clauseModificado.c: In function 'main':
shared-clauseModificado.c:13:10: error: 'n' not specified in enclosing parallel
    #pragma omp parallel for default(none)
    ^
shared-clauseModificado.c:13:10: error: enclosing parallel
shared-clauseModificado.c:14:30: error: 'a' not specified in enclosing parallel
    for (i=0; i<n; i++)    a[i] += i;
                        ^
shared-clauseModificado.c:13:10: error: enclosing parallel
    #pragma omp parallel for default(none)
    ^
```

2. ¿Qué ocurre si en `private-clause.c` se inicializa la variable `suma` fuera de la construcción `parallel` en lugar de dentro? (inicialice `suma` a un valor distinto de 0 dentro y fuera de `parallel`) Razone su respuesta. Añada el código con la modificación al cuaderno de prácticas.

RESPUESTA:

Si inicializamos la variable suma fuera de la construcción del `parallel` esta contendrá basura y no realizará bien la suma.

CAPTURA CÓDIGO FUENTE: `private-clauseModificado.c`

```
#include <stdio.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#endif

int main()
{
    int i, n = 7;
    int a[n], suma;

    for (i=0; i<n; i++)
        a[i] = i;
    //suma = 1; Mal
    #pragma omp parallel private(suma)
    {
        suma = 1; //Bien
        #pragma omp for
        for (i=0; i<n; i++)
        {
            suma = suma + a[i];
            printf(
                "thread %d suma a[%d] / ", omp_get_thread_num(), i);
        }
        printf(
            "\n* thread %d suma= %d", omp_get_thread_num(), suma);
    }

    printf("\n");

    return 0;
}
```

CAPTURAS DE PANTALLA:

Cuando lo inicializamos fuera de la construcción del `parallel`:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp private-clauseModificado.c -o private-clauseM
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./private-clauseM
thread 2 suma a[4] / thread 2 suma a[5] / thread 1 suma a[2] / thread 1 suma a[3] / thread 0 suma a[0] / thread 0 suma
a a[1] / thread 3 suma a[6] /
* thread 2 suma= 4196569
* thread 1 suma= 4196565
* thread 3 suma= 4196566
* thread 0 suma= 5
```

Cuando lo inicializamos dentro:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp private-clauseModificado.c -o private-clauseM
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./private-clauseM
thread 0 suma a[0] / thread 0 suma a[1] / thread 2 suma a[4] / thread 2 suma a[5] / thread 3 suma a[6] / thread 1 suma
a a[2] / thread 1 suma a[3] /
* thread 0 suma= 2
* thread 3 suma= 7
* thread 1 suma= 6
* thread 2 suma= 10
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$
```

3. ¿Qué ocurre si en `private-clause.c` se elimina la cláusula `private(suma)`? ¿A qué cree que es debido?

RESPUESTA:

Es debido a que, al eliminar el `private (suma)`, la variable `suma` pasara a ser compartida con lo que los distintos hilos podrán modificarla a la vez y machacar unos valores con otros.

CAPTURA CÓDIGO FUENTE: `private-clauseModificado3.c`

```
#include <stdio.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#endif

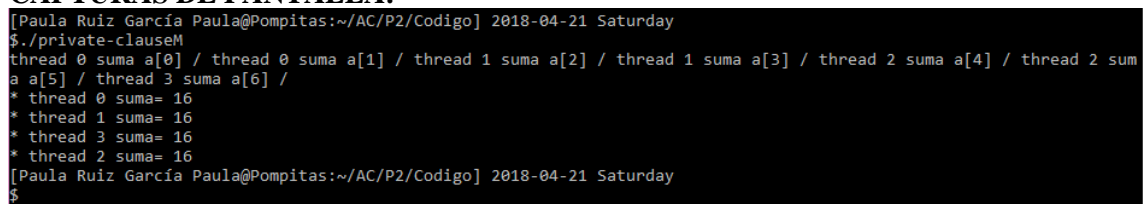
int main()
{
    int i, n = 7;
    int a[n], suma;

    for (i=0; i<n; i++)
        a[i] = i;
    //suma = 1; Mal
    #pragma omp parallel
    {
        suma = 1; //Bien
        #pragma omp for
        for (i=0; i<n; i++)
        {
            suma = suma + a[i];
            printf(
                "thread %d suma a[%d] / ", omp_get_thread_num(), i);
        }
        printf(
            "\n* thread %d suma= %d", omp_get_thread_num(), suma);
    }

    printf("\n");

    return 0;
}
```

CAPTURAS DE PANTALLA:



```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$ ./private-clauseM
thread 0 suma a[0] / thread 0 suma a[1] / thread 1 suma a[2] / thread 1 suma a[3] / thread 2 suma a[4] / thread 2 sum
a a[5] / thread 3 suma a[6] /
* thread 0 suma= 16
* thread 1 suma= 16
* thread 3 suma= 16
* thread 2 suma= 16
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$
```

- En la ejecución de `firstlastprivate.c` de la pag. 21 del seminario se imprime un 6 fuera de la región `parallel`. ¿El código imprime siempre 6 fuera de la región `parallel`? Razone su respuesta.

RESPUESTA:

Si imprime siempre 6 ya que `last/firstprivate` asigna siempre a la variable el ultimo valor que se le asignaría en una ejecución secuencial.

CAPTURAS DE PANTALLA:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp firstlastprivate-clause.c -o firstlastprivate-clause
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./firstlastprivate-clause
thread 2 suma a[4] suma=4
thread 2 suma a[5] suma=9
thread 3 suma a[6] suma=6
thread 1 suma a[2] suma=2
thread 1 suma a[3] suma=5
thread 0 suma a[0] suma=0
thread 0 suma a[1] suma=1

Fuera de la construcción parallel suma=6
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./firstlastprivate-clause
thread 2 suma a[4] suma=4
thread 2 suma a[5] suma=9
thread 3 suma a[6] suma=6
thread 0 suma a[0] suma=0
thread 0 suma a[1] suma=1
thread 1 suma a[2] suma=2
thread 1 suma a[3] suma=5

Fuera de la construcción parallel suma=6
```

5. ¿Qué se observa en los resultados de ejecución de `copyprivate-clause.c` cuando se elimina la cláusula `copyprivate(a)` en la directiva `single`? ¿A qué cree que es debido?

RESPUESTA:

Se observa un mal funcionamiento, lo que produce un resultado incorrecto, y esto es debido a que sin `copyprivate`, cuando termina `single`, no se copia la variable a los demás hilos, por lo que seguiremos usando la variable `a` (en este caso) pero contendrá basura.

CAPTURA CÓDIGO FUENTE: `copyprivate-clauseModificado.c`

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

int main() {
    int n = 9, i, b[n];

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

    #pragma omp parallel
    { int a;
      #pragma omp single
      {
          printf("\nIntroduce valor de inicialización a: ");
          scanf("%d", &a);
          printf("\nSingle ejecutada por el thread %d\n",
                omp_get_thread_num());
      }
      #pragma omp for
      for (i=0; i<n; i++) b[i] = a;
    }

    printf("Después de la región parallel:\n");
    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/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp copyprivate-clauseModificado.c -o copyprivate-clauseM
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./copyprivate-clauseM

Introduce valor de inicialización a: 8

Single ejecutada por el thread 3
Depués de la región parallel:
b[0] = 0      b[1] = 0      b[2] = 0      b[3] = 0      b[4] = 0      b[5] = 0      b[6] = 0      b[7]
= 8      b[8] = 8
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./copyprivate-clauseM

Introduce valor de inicialización a:
11

Single ejecutada por el thread 0
Depués de la región parallel:
b[0] = 11     b[1] = 11     b[2] = 11     b[3] = 0      b[4] = 0      b[5] = 0      b[6] = 0      b[7]
= 0      b[8] = 0
```

6. En el ejemplo `reduction-clause.c` sustituya `suma=0` por `suma=10`. ¿Qué resultado se imprime ahora? Justifique el resultado

RESPUESTA:

Suma = 0 equivale al valor inicial de la variable donde se almacenen los resultados de la suma con la cláusula `reduction`, por lo que si lo cambiamos a Suma = 10, el resultado final será el mismo más 10.

CAPTURA CÓDIGO FUENTE: `reduction-clauseModificado.c`

```
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#endif

int main(int argc, char **argv) {
    int i, n=20, a[n], suma=10;

    if(argc < 2) {
        fprintf(stderr, "Falta iteraciones\n");
        exit(-1);
    }
    n = atoi(argv[1]); if (n>20) {n=20; printf("n=%d", n);}

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

    #pragma omp parallel for reduction(+:suma)
    for (i=0; i<n; i++) suma += a[i];

    printf("Tras 'parallel' suma=%d\n", suma);
}
```

CAPTURAS DE PANTALLA:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp reduction-clauseModificado.c -o reduction-clauseM
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./reduction-clauseM
Falta iteraciones
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./reduction-clauseM 8
Tras 'parallel' suma=38
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./reduction-clauseM 11
Tras 'parallel' suma=65
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$
```

7. En el ejemplo `reduction-clause.c`, elimine `reduction()` de `#pragma omp parallel for reduction(+:suma)` y haga las modificaciones necesarias para que se siga realizando la suma de los componentes del vector `a` en paralelo sin usar directivas de trabajo compartido

RESPUESTA:

Si eliminamos la `clausula reduction()` podemos ver que aunque se realice la suma no nos devuelve un resultado correcto, sino que cada vez nos dará un resultado distinto.

Para solucionarlo he añadido una variable privada auxiliar que nos proporciona los resultados obtenidos del `for`, y a se acumulan en la variable `suma`, utilizando una `directiva atomic` para poder evitar las condiciones de carrera.

CAPTURA CÓDIGO FUENTE: `reduction-clauseModificado7.c`

```
#include <stdio.h>
#include <stdlib.h>
#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#endif

int main(int argc, char **argv) {
    int i, n=20, a[n], suma=0, aux;

    if(argc < 2) {
        fprintf(stderr, "Falta iteraciones\n");
        exit(-1);
    }
    n = atoi(argv[1]); if (n>20) {n=20; printf("n=%d", n);}

    for (i=0; i<n; i++) a[i] = i;
    #pragma omp parallel firstprivate(aux) shared(suma)
    {
        #pragma omp for
        for (i=0; i<n; i++) aux+= a[i];
        #pragma omp atomic
        suma += aux;
    }
    printf("Tras 'parallel' suma=%d\n", suma);
}
```

CAPTURAS DE PANTALLA:

La suma bien realizada:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp reduction-clauseModificado7.c -o reduction-clauseM
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./reduction-clauseM 11
Tras 'parallel' suma=55
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./reduction-clauseM 11
Tras 'parallel' suma=55
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./reduction-clauseM 11
Tras 'parallel' suma=55
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$
```

La suma sin `reduction`:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp reduction-clauseModificado.c -o reduction-clauseM
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./reduction-clauseM 11
Tras 'parallel' suma=34
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./reduction-clauseM 11
Tras 'parallel' suma=31
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./reduction-clauseM 11
Tras 'parallel' suma=52
```

Resto de ejercicios

8. Implementar un programa secuencial en C que calcule el producto de una matriz cuadrada, M, por un vector, v1 (implemente una versión para variables globales y otra para variables dinámicas, use una de estas versiones en los siguientes ejercicios):

$$v2 = M \bullet v1; \quad v2(i) = \sum_{k=0}^{N-1} M(i, k) \bullet v(k), \quad i = 0, \dots, N-1$$

NOTAS: (1) el número de filas /columnas N de la matriz deben ser argumentos de entrada al programa; (2) se debe inicializar la matriz y el vector 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 calcula el producto matriz vector y, al menos, el primer y último componente del resultado (esto último evita que las optimizaciones del compilador eliminen el código de la suma).

CAPTURA CÓDIGO FUENTE: pmv-secuencial.c

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

// #define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
// globales (su longitud no estará limitada
// por el ...
// tamaño de la pila del programa)
#define VECTOR_DYNAMIC // descomentar para que los vectores sean variables ...
// dinámicas (memoria reutilizable durante la ejecución)

#ifndef VECTOR_GLOBAL
#define MAX 8192 // = 2^13
double V1[MAX], V2[MAX], M[MAX][MAX];
#endif

int main(int argc, char** argv){
    int i, k;
    struct timespec cgt1, cgt2;
    double ncgt;

    // 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)
    #ifdef VECTOR_GLOBAL
    if (N > MAX) N = MAX;
    #endif
    #ifdef VECTOR_DYNAMIC
    double *V1, *V2, **M;
    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
    M = (double**) malloc(N*sizeof(double));
    if ( (V1==NULL) || (V2==NULL) || (M==NULL) ){
        printf("Error en la reserva de espacio para los vectores\n");
        exit(-2);
    }
    }
```

```

for(i=0; i<N; i++){
    M[i]=(double*) malloc(N*sizeof(double));
    if ( M[i]==NULL ){
        printf("Error en la reserva de espacio para los vectores\n");
        exit(-2);
    }
}
#endif

//Inicializar matriz y vectores
for(i=0; i<N; i++){
    V1[i] = i;
    V2[i] = 0;
    for(k=0; k<N; k++){
        M[i][k] = i + k;
    }
}
//Calculo del tiempo
clock_gettime(CLOCK_REALTIME,&cgt1);

//Calcular el producto de la matriz
for(i=0; i<N; i++){
    for (k=0; k<N; k++){
        V2[i] += M[i][k] * V1[k];
    }
}

//Calculo del tiempo
clock_gettime(CLOCK_REALTIME,&cgt2);
ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
    (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

//Imprimir resultado de la suma y el tiempo de ejecución
printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t / V2[0]=%8.6f V2[%d]=%8.6f\n",ncgt, N, V2[0], N-1, V2[N-1]);
if (N<20){
    for(i=0; i<N; i++){
        printf("/ V2[%d]=%5.2f\n", i, V2[i]);
    }
}

#ifdef VECTOR_DYNAMIC
free(V1); // libera el espacio reservado para v1
free(V2); // libera el espacio reservado para v2
for (i=0; i<N; i++){
    free(M[i]);
}
free(M);
#endif
return 0;
}

```

CAPTURAS DE PANTALLA:

Dinámicos:


```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 pmv-secuencial.c -o pmv-secuencial
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-secuencial 20
Tiempo(seg.):0.000001600      / Tamaño Vectores:20      / V2[0]=2470.000000 V2[19]=6080.000000
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-secuencial 11
Tiempo(seg.):0.000001200      / Tamaño Vectores:11      / V2[0]=385.000000 V2[10]=935.000000
/ V2[0]=385.00
/ V2[1]=440.00
/ V2[2]=495.00
/ V2[3]=550.00
/ V2[4]=605.00
/ V2[5]=660.00
/ V2[6]=715.00
/ V2[7]=770.00
/ V2[8]=825.00
/ V2[9]=880.00
/ V2[10]=935.00
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$
```

Globales:

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 pmv-secuencial.c -o pmv-secuencial
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-secuencial 20
Tiempo(seg.):0.000001500      / Tamaño Vectores:20      / V2[0]=2470.000000 V2[19]=6080.000000
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-secuencial 11
Tiempo(seg.):0.000001100      / Tamaño Vectores:11      / V2[0]=385.000000 V2[10]=935.000000
/ V2[0]=385.00
/ V2[1]=440.00
/ V2[2]=495.00
/ V2[3]=550.00
/ V2[4]=605.00
/ V2[5]=660.00
/ V2[6]=715.00
/ V2[7]=770.00
/ V2[8]=825.00
/ V2[9]=880.00
/ V2[10]=935.00
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$
```

/*A partir de aquí solo utilizo la parte de vectores dinámicos para las practicas*/

9. Implementar en paralelo el producto matriz por vector con OpenMP a partir del código escrito en el ejercicio anterior usando la directiva for. Debe implementar dos versiones del código (consulte la lección 5/Tema 2):

- una primera que paralelice el bucle que recorre las filas de la matriz y
- una segunda que paralelice el bucle que recorre las columnas.

Use las directivas que estime oportunas y las cláusulas que sean necesarias **excepto la cláusula reduction**. Se debe paralelizar también la inicialización de las matrices. Respecto a este ejercicio:

- Anote en su cuaderno de prácticas todos los errores de compilación que se han generado durante la realización del ejercicio y explique cómo los ha resuelto (especifique qué ayudas externas ha usado o recibido).
- Anote todos los errores en tiempo de ejecución que se han generado durante la realización del ejercicio y explique cómo los ha resuelto (especifique qué ayudas externas ha usado o recibido).

NOTAS: (1) el número de filas /columnas N de la matriz deben ser argumentos de entrada; (2) se debe inicializar la matriz y el vector 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 que calcula el producto matriz vector y, al menos, el primer y último componente del resultado (esto último evita que las optimizaciones del compilador eliminen el código de la suma).

CAPTURA CÓDIGO FUENTE : pmv-OpenMP-a.c

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

#ifdef _OPENMP
```

```

#include <omp.h>
#else
#define omp_get_thread_num() 0
#define omp_get_num_threads() 1
#endif

// #define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
// globales (su longitud no estará limitada
por el ...
// tamaño de la pila del programa)
#define VECTOR_DYNAMIC // descomentar para que los vectores sean variables ...

// dinámicas (memoria reutilizable durante la ejecución)
#ifdef VECTOR_GLOBAL
#define MAX 8192 // = 2^13
double V1[MAX], V2[MAX], M[MAX][MAX];
#endif

int main(int argc, char** argv){
    int i, k;
    double t1, t2, t_total;

    // 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)
    #ifdef VECTOR_GLOBAL
    if (N > MAX) N = MAX;
    #endif
    #ifdef VECTOR_DYNAMIC
    double *V1, *V2, **M;
    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
    M = (double**) malloc(N*sizeof(double));
    if ( (V1==NULL) || (V2==NULL) || (M==NULL) ){
        printf("Error en la reserva de espacio para los vectores\n");
        exit(-2);
    }

    for(i=0; i<N; i++){
        M[i] = (double*) malloc(N*sizeof(double));
        if ( M[i]==NULL ){
            printf("Error en la reserva de espacio para los vectores\n");
            exit(-2);
        }
    }
    #endif

    // Inicializar matriz y vectores
    #pragma omp parallel
    {
        #pragma omp for private(k)
        for(i=0; i<N; i++){
            V1[i] = i;

```

```

V2[i] = 0;
for(k=0; k<N; k++){
    M[i][k] = i + k;
}
}

//Medida de tiempo
#pragma omp single
t1 = omp_get_wtime();

//Calcular el producto de la matriz
#pragma omp for private(k)
for(i=0; i<N; i++){
    for (k=0; k<N; k++){
        V2[i] += M[i][k] * V1[k];
    }
}

//Medida de tiempo
#pragma omp single
t2 = omp_get_wtime();
}
t_total = t2 - t1;

//Imprimir resultado de la suma y el tiempo de ejecución
printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t / V2[0]=%8.6f V2[%d]=%8.6f\n", t_total, N, V2[0], N-1,
V2[N-1]);

if (N<20){
    for(i=0; i<N; i++){
        printf("/ V2[%d]=%5.2f\n", i, V2[i]);
    }
}

#ifdef VECTOR_DYNAMIC
free(V1); // libera el espacio reservado para v1
free(V2); // libera el espacio reservado para v2
for (i=0; i<N; i++){
    free(M[i]);
}
free(M); //libera el espacio reservado para M
#endif
return 0;
}

```

CAPTURA CÓDIGO FUENTE: pmv-OpenMP-b.c

```

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

#ifdef _OPENMP
    #include <omp.h>
#else
    #define omp_get_thread_num() 0
    #define omp_get_num_threads() 1
#endif

// #define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
// globales (su longitud no estará limitada

```

```

por el ...

// tamaño de la pila del programa)
#define VECTOR_DYNAMIC // descomentar para que los vectores sean variables ...

// dinámicas (memoria reutilizable durante la ejecución)
#ifdef VECTOR_GLOBAL
#define MAX 8192 // = 2^13
double V1[MAX], V2[MAX], M[MAX][MAX];
#endif

int main(int argc, char** argv){
    int i, k;
    double t1, t2, t_total;

    //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)
    #ifdef VECTOR_GLOBAL
    if (N>MAX) N=MAX;
    #endif
    #ifdef VECTOR_DYNAMIC
    double *V1, *V2, **M;
    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
    M = (double**) malloc(N*sizeof(double));
    if ( (V1==NULL) || (V2==NULL) || (M==NULL) ){
        printf("Error en la reserva de espacio para los vectores\n");
        exit(-2);
    }

    for(i=0; i<N; i++){
        M[i]=(double*) malloc(N*sizeof(double));
        if ( M[i]==NULL ){
            printf("Error en la reserva de espacio para los vectores\n");
            exit(-2);
        }
    }
    #endif

    //Inicializar matriz y vectores
    for(i=0; i<N; i++){
        V1[i] = i;
        V2[i] = 0;
        #pragma omp parallel for
        for(k=0; k<N; k++){
            M[i][k] = i + k;
        }
    }

    //Medida de tiempo
    t1 = omp_get_wtime();

    //Calcular el producto de la matriz

```

```

for(i=0; i<N; i++){
    #pragma omp parallel
    {
        double aux = 0;
        #pragma omp for
        for (k=0; k<N; k++){
            aux += M[i][k] * V1[k];
        }
        #pragma omp critical
        V2[i] += aux;
    }
}
//Medida de tiempo
t2 = omp_get_wtime();
t_total = t2 - t1;

//Imprimir resultado de la suma y el tiempo de ejecución
printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t / V2[0]=%8.6f V2[%d]=%8.6f\n", t_total, N, V2[0], N-1,
V2[N-1]);

if (N<20){
    for(i=0; i<N; i++){
        printf("/ V2[%d]=%5.2f\n", i, V2[i]);
    }
}

#ifdef VECTOR_DYNAMIC
free(V1); // libera el espacio reservado para v1
free(V2); // libera el espacio reservado para v2
for (i=0; i<N; i++){
    free(M[i]);
}
free(M); //libera el espacio reservado para M
#endif
return 0;
}

```

RESPUESTA:

El ejercicio A me ha compilado y ejecutado sin ningún tipo de error, mientras que el ejercicio B me ha devuelto valores erróneos, por lo que he preferido crear una variable global `double auxiliar` la cual igualo a 0 y a continuación creo una sección crítica para que no lie los valores si no que cada uno se suma cuando se debe sumar.

Para la resolución he obtenido ayudas de los seminarios 1 y 2 de prácticas.

CAPTURAS DE PANTALLA:

pmv-OpenMP-a.c

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp pmv-OpenMP-a.c -o pmv-a
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-a 20
Tiempo(seg.):0.000003000 / Tamaño Vectores:20 / V2[0]=2470.000000 V2[19]=6080.000000
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-a 11
Tiempo(seg.):0.000002000 / Tamaño Vectores:11 / V2[0]=385.000000 V2[10]=935.000000
/ V2[0]=385.00
/ V2[1]=440.00
/ V2[2]=495.00
/ V2[3]=550.00
/ V2[4]=605.00
/ V2[5]=660.00
/ V2[6]=715.00
/ V2[7]=770.00
/ V2[8]=825.00
/ V2[9]=880.00
/ V2[10]=935.00
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$
```

pmv-OpenMP-b.c

```
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp pmv-OpenMP-b.c -o pmv-b
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-a 20
Tiempo(seg.):0.000003000 / Tamaño Vectores:20 / V2[0]=2470.000000 V2[19]=6080.000000
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-a 11
Tiempo(seg.):0.000003000 / Tamaño Vectores:11 / V2[0]=385.000000 V2[10]=935.000000
/ V2[0]=385.00
/ V2[1]=440.00
/ V2[2]=495.00
/ V2[3]=550.00
/ V2[4]=605.00
/ V2[5]=660.00
/ V2[6]=715.00
/ V2[7]=770.00
/ V2[8]=825.00
/ V2[9]=880.00
/ V2[10]=935.00
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$
```

10. A partir de la segunda versión de código paralelo desarrollado en el ejercicio anterior, implementar una versión paralela del producto matriz por vector con OpenMP que use para comunicación/sincronización la cláusula `reduction`. Respecto a este ejercicio:

- Anote en su cuaderno de prácticas todos los errores de compilación que se han generado durante la realización del ejercicio y explique cómo los ha resuelto (especifique qué ayudas externas ha usado o recibido).
- Anote todos los errores en tiempo de ejecución que se han generado durante la realización del ejercicio y explique cómo los ha resuelto (especifique qué ayudas externas ha usado o recibido).

CAPTURA CÓDIGO FUENTE: `pmv-OpenmMP-reduction.c`

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

#ifdef _OPENMP
#include <omp.h>
#else
#define omp_get_thread_num() 0
#define omp_get_num_threads() 1
#endif

// #define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
// globales (su longitud no estará limitada
// por el ...
// tamaño de la pila del programa)
#define VECTOR_DYNAMIC // descomentar para que los vectores sean variables ...
// dinámicas (memoria reutilizable durante la ejecución)
```

```

#ifndef VECTOR_GLOBAL
#define MAX 8192  //2^13
double V1[MAX], V2[MAX], M[MAX][MAX];
#endif

int main(int argc, char** argv){
    int i, k;
    double t1, t2, t_total;

    //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)
    #ifndef VECTOR_GLOBAL
    if (N>MAX) N=MAX;
    #endif
    #ifndef VECTOR_DYNAMIC
    double *V1, *V2, **M;
    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
    M = (double**) malloc(N*sizeof(double));
    if ( (V1==NULL) || (V2==NULL) || (M==NULL)){
        printf("Error en la reserva de espacio para los vectores\n");
        exit(-2);
    }

    for(i=0; i<N; i++){
        M[i]=(double*) malloc(N*sizeof(double));
        if ( M[i]==NULL ){
            printf("Error en la reserva de espacio para los vectores\n");
            exit(-2);
        }
    }
    #endif

    //Inicializar matriz y vectores
    for(i=0; i<N; i++){
        V1[i] = i;
        V2[i] = 0;
        #pragma omp parallel for
        for(k=0; k<N; k++){
            M[i][k] = i + k;
        }
    }

    //Medida de tiempo
    t1 = omp_get_wtime();

    //Calcular el producto de la matriz

    for(i=0; i<N; i++){
        double aux = 0;
        #pragma omp parallel for reduction(+:aux)
        for (k=0; k<N; k++){
            aux += M[i][k] * V1[k];

```

```

    }
    V2[i] += aux;
}
//Medida de tiempo
t2 = omp_get_wtime();
t_total = t2 - t1;

//Imprimir resultado de la suma y el tiempo de ejecución
printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t / V2[0]=%8.6f V2[%d]=%8.6f\n", t_total, N, V2[0], N-1,
V2[N-1]);

if (N<20){
    for(i=0; i<N; i++){
        printf("/ V2[%d]=%5.2f\n", i, V2[i]);
    }
}

#ifdef VECTOR_DYNAMIC
free(V1); // libera el espacio reservado para v1
free(V2); // libera el espacio reservado para v2
for (i=0; i<N; i++){
    free(M[i]);
}
free(M); //libera el espacio reservado para M
#endif
return 0;
}

```

RESPUESTA:

No he obtenido fallos a la hora de compilarlo y ejecutarlo.

CAPTURAS DE PANTALLA:

```

[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$gcc -O2 -fopenmp pmv-OpenMP-reduction.c -o pmv-reduction
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-reduction 20
Tiempo(seg.):0.000035000      / Tamaño Vectores:20      / V2[0]=2470.000000 V2[19]=6080.000000
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$./pmv-reduction 11
Tiempo(seg.):0.000019000      / Tamaño Vectores:11      / V2[0]=385.000000 V2[10]=935.000000
/ V2[0]=385.00
/ V2[1]=440.00
/ V2[2]=495.00
/ V2[3]=550.00
/ V2[4]=605.00
/ V2[5]=660.00
/ V2[6]=715.00
/ V2[7]=770.00
/ V2[8]=825.00
/ V2[9]=880.00
/ V2[10]=935.00
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-21 Saturday
$

```

11. Ayudándose de una hoja de cálculo (recuerde que en las aulas está instalado OpenOffice) realice una tabla y una gráfica que permitan comparar la escalabilidad (ganancia en velocidad en función del número de cores) en atcgrid y en su PC del mejor código paralelo de los tres implementados en los ejercicios anteriores para dos tamaños (N) distintos (consulte la Lección 6/Tema 2). Usar -O2 al compilar. Justificar por qué el código escogido es el mejor. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

CAPTURAS DE PANTALLA (que justifique el código elegido):**Solución ATCGRID:**


```
sftp> get pmv-OpenMPa.*
Fetching /home/D1estudiante18/pmv-OpenMPa.e75677 to pmv-OpenMPa.e75677
Fetching /home/D1estudiante18/pmv-OpenMPa.o75677 to pmv-OpenMPa.o75677
/home/D1estudiante18/pmv-OpenMPa.o75677 100% 1023 1.0KB/s 00:00
sftp> get pmv-OpenMPb.*
Fetching /home/D1estudiante18/pmv-OpenMPb.e75678 to pmv-OpenMPb.e75678
/home/D1estudiante18/pmv-OpenMPb.e75678 100% 51 0.1KB/s 00:00
Fetching /home/D1estudiante18/pmv-OpenMPb.o75678 to pmv-OpenMPb.o75678
/home/D1estudiante18/pmv-OpenMPb.o75678 100% 121 0.1KB/s 00:00
sftp> get pmv-OpenMP-reduction.*
Fetching /home/D1estudiante18/pmv-OpenMP-reduction.e75679 to pmv-OpenMP-reduction.e75679
Fetching /home/D1estudiante18/pmv-OpenMP-reduction.o75679 to pmv-OpenMP-reduction.o75679
/home/D1estudiante18/pmv-OpenMP-reduction.o75679 100% 1031 1.0KB/s 00:00
sftp>
```

```
[D1estudiante18@atcgrid ~]$ qsub script-a.sh -q ac
75677.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID Name User Time Use S Queue
-----
75677.atcgrid pmv-OpenMPa D1estudiante18 0 R ac
[D1estudiante18@atcgrid ~]$ qsub script-b.sh -q ac
75678.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID Name User Time Use S Queue
-----
75677.atcgrid pmv-OpenMPa D1estudiante18 00:00:00 C ac
75678.atcgrid pmv-OpenMPb D1estudiante18 0 R ac
[D1estudiante18@atcgrid ~]$ qsub script-reduction.sh -q ac
75679.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID Name User Time Use S Queue
-----
75678.atcgrid pmv-OpenMPb D1estudiante18 0 R ac
75679.atcgrid ...nMP-reduction D1estudiante18 0 R ac
[D1estudiante18@atcgrid ~]$
```

(*Aunque el OpenMP-b no vale esta solución*)

```
sftp> get STDIN*
Fetching /home/D1estudiante18/STDIN.e75746 to STDIN.e75746
Fetching /home/D1estudiante18/STDIN.e75747 to STDIN.e75747
Fetching /home/D1estudiante18/STDIN.e75748 to STDIN.e75748
Fetching /home/D1estudiante18/STDIN.e75749 to STDIN.e75749
Fetching /home/D1estudiante18/STDIN.e75750 to STDIN.e75750
Fetching /home/D1estudiante18/STDIN.e75751 to STDIN.e75751
Fetching /home/D1estudiante18/STDIN.e75752 to STDIN.e75752
Fetching /home/D1estudiante18/STDIN.e75753 to STDIN.e75753
Fetching /home/D1estudiante18/STDIN.e75754 to STDIN.e75754
Fetching /home/D1estudiante18/STDIN.o75746 to STDIN.o75746
/home/D1estudiante18/STDIN.o75746 100% 108 0.1KB/s 00:00
Fetching /home/D1estudiante18/STDIN.o75747 to STDIN.o75747
/home/D1estudiante18/STDIN.o75747 100% 109 0.1KB/s 00:00
Fetching /home/D1estudiante18/STDIN.o75748 to STDIN.o75748
/home/D1estudiante18/STDIN.o75748 100% 112 0.1KB/s 00:01
Fetching /home/D1estudiante18/STDIN.o75749 to STDIN.o75749
/home/D1estudiante18/STDIN.o75749 100% 112 0.1KB/s 00:00
Fetching /home/D1estudiante18/STDIN.o75750 to STDIN.o75750
/home/D1estudiante18/STDIN.o75750 100% 113 0.1KB/s 00:00
Fetching /home/D1estudiante18/STDIN.o75751 to STDIN.o75751
/home/D1estudiante18/STDIN.o75751 100% 113 0.1KB/s 00:00
Fetching /home/D1estudiante18/STDIN.o75752 to STDIN.o75752
/home/D1estudiante18/STDIN.o75752 100% 114 0.1KB/s 00:00
Fetching /home/D1estudiante18/STDIN.o75753 to STDIN.o75753
/home/D1estudiante18/STDIN.o75753 100% 115 0.1KB/s 00:00
Fetching /home/D1estudiante18/STDIN.o75754 to STDIN.o75754
/home/D1estudiante18/STDIN.o75754 100% 115 0.1KB/s 00:00
sftp>
```

```
[D1estudiante18@atcgrid ~]$ echo './pmv-OpenMP-b 5000' | qsub -q ac
75746.atcgrid
[D1estudiante18@atcgrid ~]$ echo './pmv-OpenMP-b 10000' | qsub -q ac
75747.atcgrid
[D1estudiante18@atcgrid ~]$ echo './pmv-OpenMP-b 15000' | qsub -q ac
75748.atcgrid
[D1estudiante18@atcgrid ~]$ echo './pmv-OpenMP-b 20000' | qsub -q ac
75749.atcgrid
[D1estudiante18@atcgrid ~]$ echo './pmv-OpenMP-b 25000' | qsub -q ac
75750.atcgrid
[D1estudiante18@atcgrid ~]$ echo './pmv-OpenMP-b 30000' | qsub -q ac
75751.atcgrid
[D1estudiante18@atcgrid ~]$ echo './pmv-OpenMP-b 40000' | qsub -q ac
75752.atcgrid
[D1estudiante18@atcgrid ~]$ echo './pmv-OpenMP-b 50000' | qsub -q ac
75753.atcgrid
[D1estudiante18@atcgrid ~]$ echo './pmv-OpenMP-b 60000' | qsub -q ac
75754.atcgrid
[D1estudiante18@atcgrid ~]$ qstat
Job ID Name User Time Use S Queue
-----
75746.atcgrid STDIN D1estudiante18 0 R ac
75751.atcgrid STDIN D1estudiante18 00:00:00 C ac
75752.atcgrid STDIN D1estudiante18 00:00:00 C ac
75753.atcgrid STDIN D1estudiante18 00:00:00 E ac
75754.atcgrid STDIN D1estudiante18 0 R ac
```

Solución PC LOCAL:

```

[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-22 Sunday
$gcc -O2 -fopenmp pmv-OpenMP-a.c -o pmv-OpenMP-a
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-22 Sunday
$bash script-a.sh
Id. usuario del trabajo:
Id. del trabajo:
Nombre del trabajo especificado por usuario:
Nodo que ejecuta qsub:
Directorio en el que se ha ejecutado qsub:
Cola:
script-a.sh: line 13: '$\r': command not found
pmv-OpenMP-a
Tiempo(seg.):0.023037000 / Tamaño Vectores:5000 / V2[0]=41654167500.000000 V2[4999]=104129170000.000000
Tiempo(seg.):0.090988000 / Tamaño Vectores:10000 / V2[0]=333283335000.000000 V2[9999]=833183340000.00
0000
Tiempo(seg.):0.258760000 / Tamaño Vectores:15000 / V2[0]=1124887502500.000000 V2[14999]=2812162510000
.000000
Tiempo(seg.):0.389647000 / Tamaño Vectores:20000 / V2[0]=2666466670000.000000 V2[19999]=6666066680000
.000000
Tiempo(seg.):2.239161000 / Tamaño Vectores:25000 / V2[0]=5208020837500.000000 V2[24999]=1301989585000
0.000000
Tiempo(seg.):16.365290000 / Tamaño Vectores:30000 / V2[0]=8999550005000.000000 V2[29999]=2249865002000
0.000000
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-22 Sunday
$

[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-22 Sunday
$gcc -O2 -fopenmp pmv-OpenMP-b.c -o pmv-OpenMP-b
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-22 Sunday
$bash script-b.sh
Id. usuario del trabajo:
Id. del trabajo:
Nombre del trabajo especificado por usuario:
Nodo que ejecuta qsub:
Directorio en el que se ha ejecutado qsub:
Cola:
script-b.sh: line 13: '$\r': command not found
pmv-OpenMP-b
Tiempo(seg.):0.045818000 / Tamaño Vectores:5000 / V2[0]=41654167500.000000 V2[4999]=104129170000.000000
Tiempo(seg.):0.134104000 / Tamaño Vectores:10000 / V2[0]=333283335000.000000 V2[9999]=833183340000.00
0000
Tiempo(seg.):0.342490000 / Tamaño Vectores:15000 / V2[0]=1124887502500.000000 V2[14999]=2812162510000
.000000
Tiempo(seg.):0.860512000 / Tamaño Vectores:20000 / V2[0]=2666466670000.000000 V2[19999]=6666066680000
.000000
Tiempo(seg.):1.021111000 / Tamaño Vectores:25000 / V2[0]=5208020837500.000000 V2[24999]=1301989585000
0.000000
Tiempo(seg.):28.698976000 / Tamaño Vectores:30000 / V2[0]=8999550005000.000000 V2[29999]=2249865002000
0.000000
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-22 Sunday
$^C

[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-22 Sunday
$gcc -O2 -fopenmp pmv-OpenMP-reduction.c -o pmv-OpenMP-reduction
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-22 Sunday
$bash script-reduction.sh
Id. usuario del trabajo:
Id. del trabajo:
Nombre del trabajo especificado por usuario:
Nodo que ejecuta qsub:
Directorio en el que se ha ejecutado qsub:
Cola:
script-reduction.sh: line 13: '$\r': command not found
pmv-OpenMP-reduction
Tiempo(seg.):0.039720000 / Tamaño Vectores:5000 / V2[0]=41654167500.000000 V2[4999]=104129170000.000000
Tiempo(seg.):0.148902000 / Tamaño Vectores:10000 / V2[0]=333283335000.000000 V2[9999]=833183340000.00
0000
Tiempo(seg.):0.286898000 / Tamaño Vectores:15000 / V2[0]=1124887502500.000000 V2[14999]=2812162510000
.000000
Tiempo(seg.):0.759272000 / Tamaño Vectores:20000 / V2[0]=2666466670000.000000 V2[19999]=6666066680000
.000000
Tiempo(seg.):1.372572000 / Tamaño Vectores:25000 / V2[0]=5208020837500.000000 V2[24999]=1301989585000
0.000000
Tiempo(seg.):22.152438000 / Tamaño Vectores:30000 / V2[0]=8999550005000.000000 V2[29999]=2249865002000
0.000000
[Paula Ruiz García Paula@Pompitas:~/AC/P2/Codigo] 2018-04-22 Sunday
$

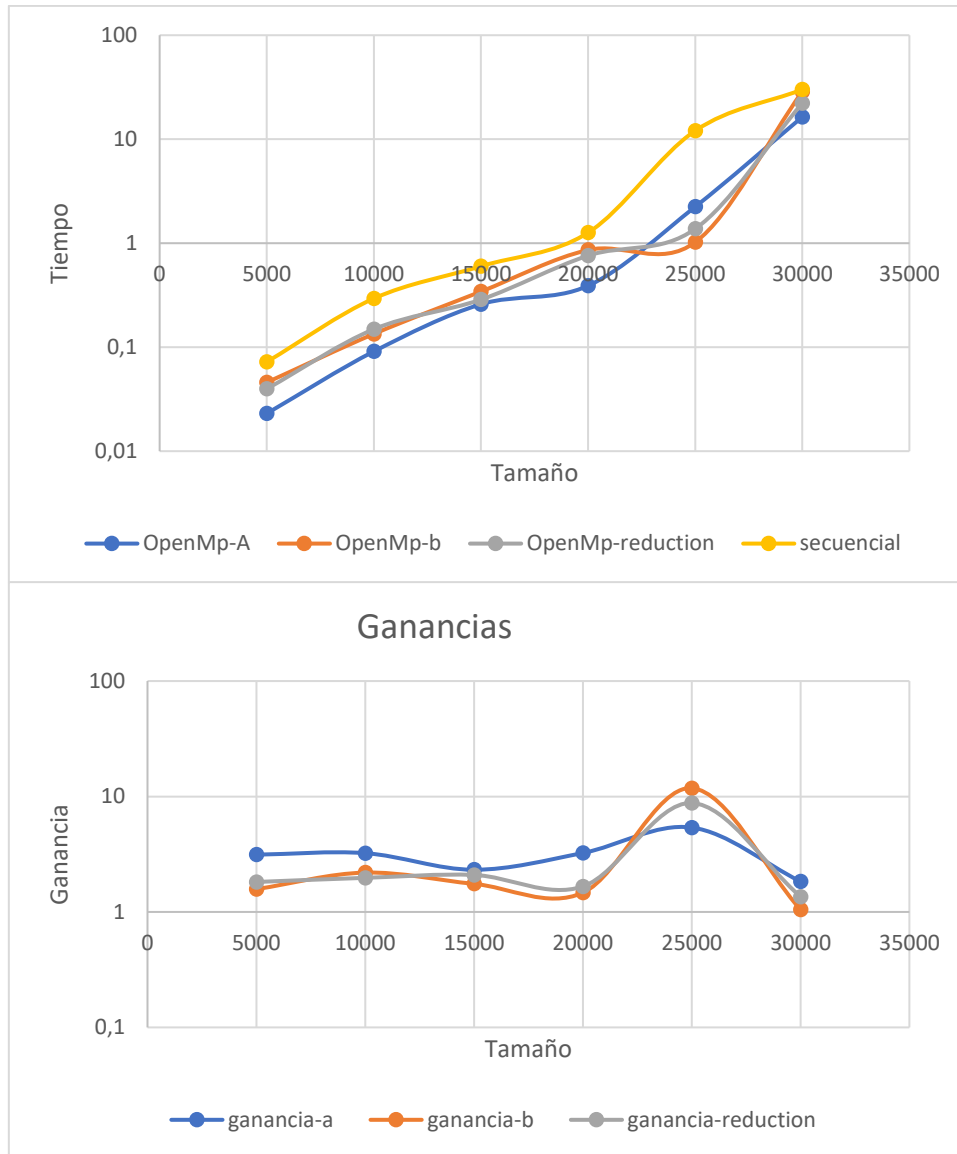
```

TABLA Y GRÁFICA (por *ejemplo* para 1-4 threads PC local, y para 1-12 threads en atcgrid, tamaños-N-: un N entre 30000 y 100000, y otro entre 5000 y 30000):

PC Local:

Tamaño	Tiempo paralelo			Tiempo Secuencial	Ganancia		
	OpenMP-A	OpenMP-B	OpenMP-Reduction		OpenMP-A	OpenMP-B	OpenMP-Reduction
5000	0.023037	0.045818	0.03972	0.0721976	3.13398446	1.57574752	1.81766365
10000	0.090988	0.134104	0.148902	0.2939232	3.23035126	2.19175565	1.97393722

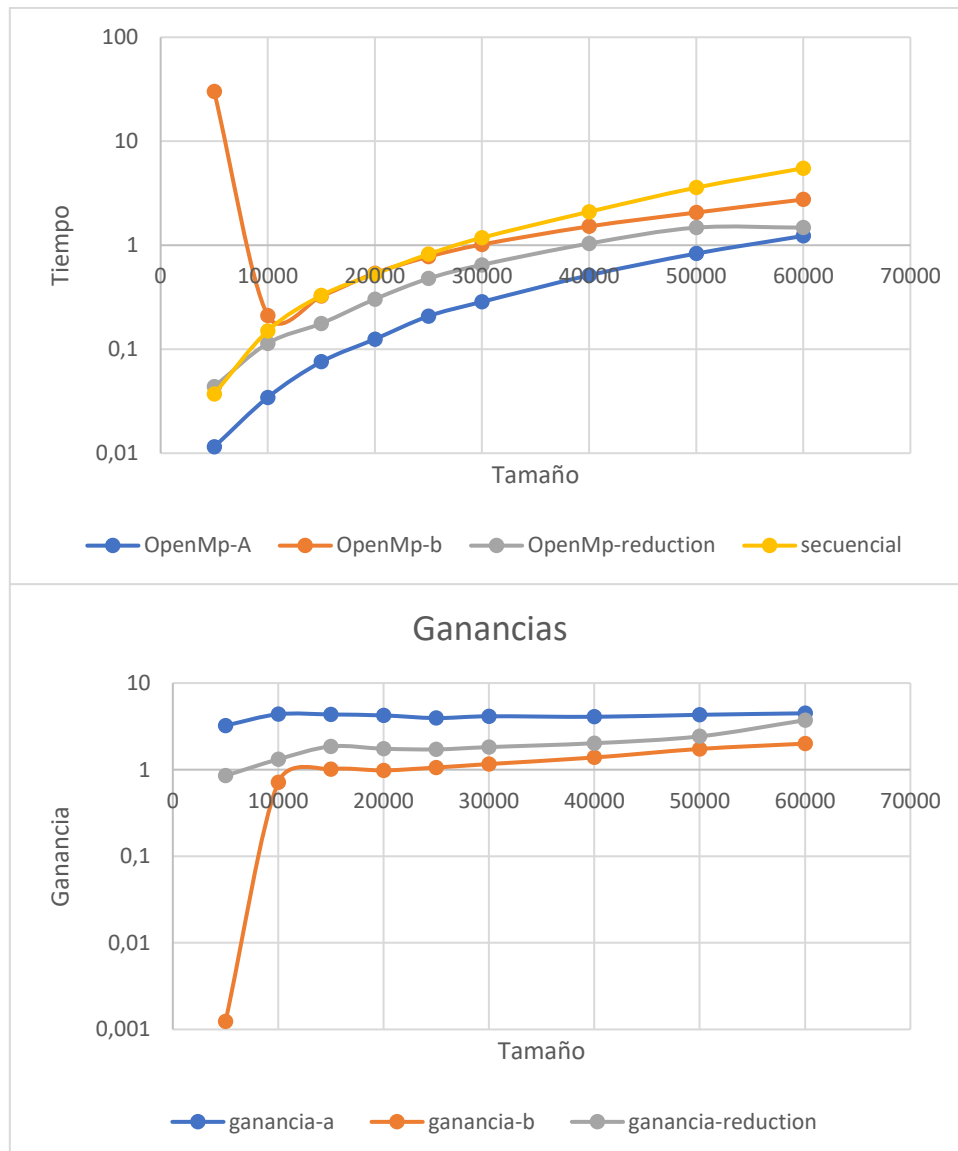
15000	0.25876	0.34249	0.286898	0.6001198	2.31921394	1.75222576	2.09175317
20000	0.389647	0.860512	0.759272	1.2619807	3.23877946	1.46654631	1.66209303
25000	2.239161	1.021111	1.372572	12.0547957	5.38362168	11.8055683	8.78263268
30000	16.36529	28.698976	22.152438	30.029992	1.83498074	1.04637852	1.35560664



Atcgrid:

Tamaño	Tiempo paralelo			Tiempo Secuencial	Ganancia		
	OpenMP-A	OpenMP-B	OpenMP-Reduction		OpenMP-A	OpenMP-B	OpenMP-Reduction
5000	0.01148346	30.074293	0.043601656	0.0371385	3.23408514	0.00123489	0.85176804
10000	0.03438983	0.2110546	0.114000345	0.15010782	4.36489052	0.71122739	1.31673129
15000	0.07564722	0.3241489	0.176970354	0.32841723	4.34143181	1.01316772	1.85577543
20000	0.1249276	0.5385644	0.302927153	0.5285558	4.230897	0.98141607	1.74482805
25000	0.20769645	0.7742249	0.477766434	0.8193631	3.94500297	1.05830112	1.71498674
30000	0.2854622	1.0161545	0.647523444	1.18059126	4.13571831	1.16182261	1.82324096
40000	0.51409905	1.5172791	1.038386867	2.09702611	4.07903125	1.38209651	2.01950369

50000	0.83164705	2.0629978	1.47598814	3.57938448	4.30397066	1.73504039	2.42507672
60000	1.22864198	2.7518292	1.47598814	5.5017781	4.47793433	1.99931673	3.72752189



COMENTARIOS SOBRE LOS RESULTADOS:

A la hora de obtener los resultados para atcgrid he tenido que coger menos valores de los iniciales porque el calculo se sobrepasaba del tiempo de espera, y además, para OpenMP-b he tenido que realizarlos manualmente y varias veces ya que siempre sobrepasaba el primero el tiempo limite de espera y además los segundos también durante la primeras ejecuciones, he intentado cambiar el código y nada de nada hasta que he conseguido que casi todos me de valores mas o menos regulares.

También me he dado cuenta que tanto en mi pc local como en atcgrid el más eficiente es el OpenMp-a ya que es el que realiza la multiplicación más rápidamente y es el único que no me ha dado problemas.