|  |  |  |
| --- | --- | --- |
| **2º curso / 2º cuatr.**  **Grado Ing. Inform.** |  | **Arquitectura de Computadores (AC)**  **Cuaderno de prácticas.**  **Bloque Práctico 3.** **Programación paralela III: Interacción con el entorno en OpenMP**  Estudiante (nombre y apellidos): Alberto Llamas González  Grupo de prácticas: D3  Fecha de entrega:  Fecha evaluación en clase: |

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

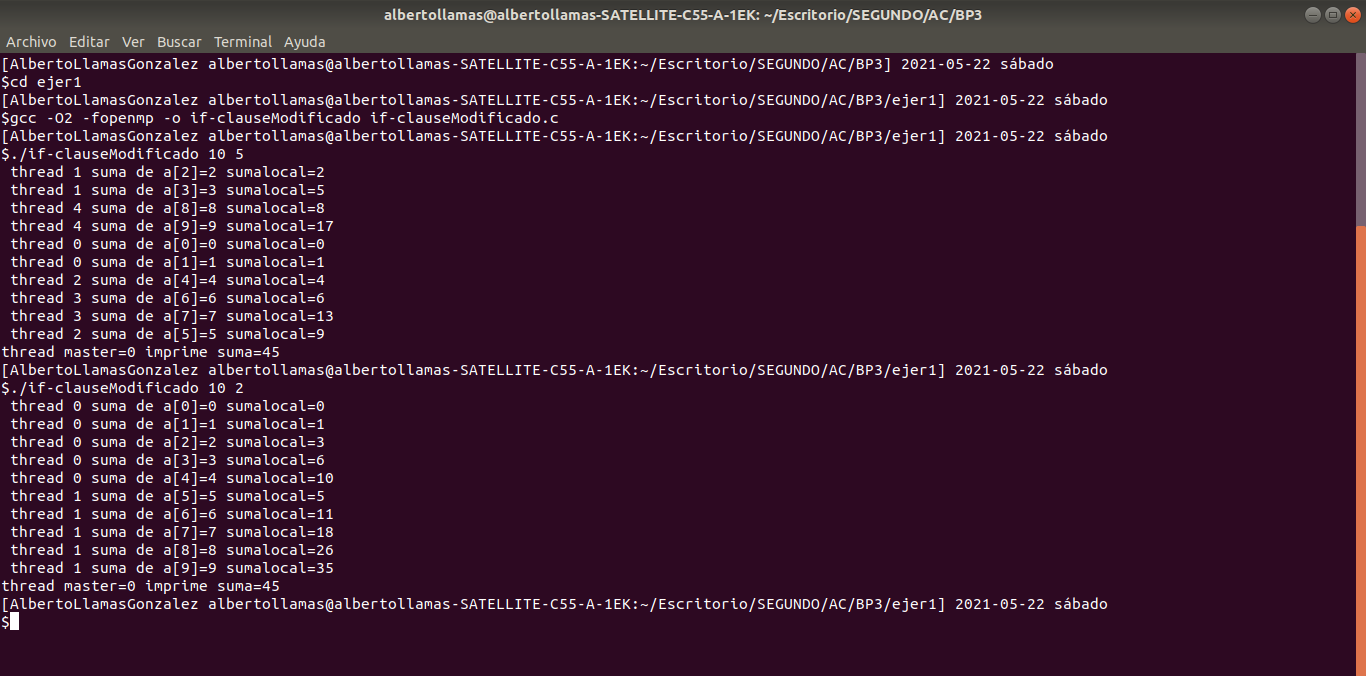
Ejercicios basados en los ejemplos del seminario práctico

1. Usar la cláusula num\_threads(x) en el ejemplo del seminario if\_clause.c, y añadir un parámetro de entrada al programa que fije el valor x que se va a usar en la cláusula. Incorporar en el cuaderno de trabajo de esta práctica volcados de pantalla con ejemplos de ejecución que ilustren la funcionalidad de esta cláusula y explicar por qué lo ilustran.

**CAPTURA CÓDIGO FUENTE**: if-clauseModificado.c

|  |
| --- |
| ../../../../../../Downloads/ejer/ejer1/Captura%20de%20pantalla%20de%202021-05-22%2016-21-53.png |

**CAPTURAS DE PANTALLA:**

****

**RESPUESTA**: Se han realizado dos ejecuciones del programa if-clause.c con un segundo parámetro distinto correspondiente al número de hebras que ejecutan el bucle. Como vemos el resultado de la suma se mantiene igual para ambas ejecuciones debido a que el primer parámetro es > 4.

**2.**  Rellenar la Tabla 1 (se debe poner en la tabla el id del *thread* que ejecuta cada iteración) usando scheduler-clause.c con tres *threads* (0,1,2) y un número de iteraciones de 16 (0 a 15 en la tabla). Con este ejercicio se pretende comparar distintas alternativas de planificación de bucles. Se van a usar distintos tipos (static, dynamic, guided), modificadores (monotonic y nonmonotonic) y tamaños de chunk (x = 1, 2 y 4).

|  |
| --- |
| 1. Tabla schedule. Rellenar esta tabla ejecutando scheduler-clause.c asignando previamente a la variable de entorno OMP\_SCHEDULE los valores que se indican en la tabla (por ej.: export OMP\_SCHEDULE=“nonmonotonic:static,2). En la segunda fila, 1, 2 4 representan el tamaño del chunk |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteración** | “monotonic:static,x“ | | | “nonmonotonic:static,x“ | | | “monotonic:dynamic,x“ | | | “monotonic:guided,x“ | | |
| x=1 | x=2 | x=4 | x=1 | x=2 | x=4 | x=1 | x=2 | x=4 | x=1 | x=2 | x=4 |
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 1 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| **2** | 2 | 1 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| **3** | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 0 | 0 |
| **4** | 1 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 0 | 0 | 0 |
| **5** | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 0 | 1 | 0 | 1 | 1 |
| **6** | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| **7** | 1 | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 0 | 0 | 0 | 1 |
| **8** | 2 | 1 | 2 | 2 | 1 | 2 | 0 | 1 | 2 | 0 | 0 | 1 |
| **9** | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 0 | 1 |
| **10** | 1 | 2 | 2 | 1 | 2 | 2 | 0 | 1 | 2 | 1 | 1 | 1 |
| **11** | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 1 | 2 | 1 | 1 | 0 |
| **12** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 1 | 2 | 2 |
| **13** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 1 | 1 | 2 | 2 |
| **14** | 2 | 1 | 0 | 2 | 1 | 0 | 0 | 0 | 1 | 2 | 2 | 2 |
| **15** | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 | 2 | 2 |

Destacar las diferencias entre las 4 alternativas de planificación de la tabla, en particular, las que hay entre static, dynamic y guided y las diferencias entre usar monotonic y nonmonotonic.

**RESPUESTA**:

*Static* divide las iteraciones en un número igual al chunk pasado como parámetro y se van asignando en round robin. *Dynamic* divide las iteraciones de la misma forma, pero la asignación se hace en tiempo de ejecución es decir, se el chunk correspondiente de iteraciones a la hebra que termine antes. *Guided* obliga a que el tamaño mínimo del bloque de iteraciones que realiza una misma hebra sea igual a los chunks pasados como parámetros.

La diferencia entre *monotonic* y *nonmonotonic* es que la primera obliga a que las iteraciones del bucle las realicen la correspondiente hebra en orden lógico creciente. *Nonmonotonic* simplemente no mantiene este orden lógico creciente.

**3.** ¿Qué valor por defecto usa OpenMP para chunk y modifier con static, dynamic y guided? Explicar qué ha hecho para contestar a esta pregunta.

Para OMP\_SCHEDULE = “*static*”, los valores son 1 para modifier y 0 para chunk.

Para OMP\_SCHEDULE = “*dynamic*”, los valores son 2 para modifier y 1 para chunk.

Para OMP\_SCHEDULE = “*guided*”, los valores son 3 para modifier y 1 para chunk.

En el programa anterior he utilizado la función *omp\_get\_schedule* para obtener dichos valores. Además los únicos argumentos que podemos pasar a la función *omp\_set\_schedule* para obtener *schedule* como *static*, *dynamic* o *guided*, son 1, 2 ó 3 respectivamente.

**4.** Añadir al programa scheduled-clause.c lo necesario para que imprima el valor de las variables de control dyn-var, nthreads-var, thread-limit-var y run-sched-var dentro (debe imprimir sólo un thread) y fuera de la región paralela. Realizar varias ejecuciones usando variables de entorno para modificar estas variables de control antes de la ejecución. Incorporar en su cuaderno de prácticas volcados de pantalla de estas ejecuciones. ¿Se imprimen valores distintos dentro y fuera de la región paralela?

**CAPTURA CÓDIGO FUENTE**: scheduled-clauseModificado.c

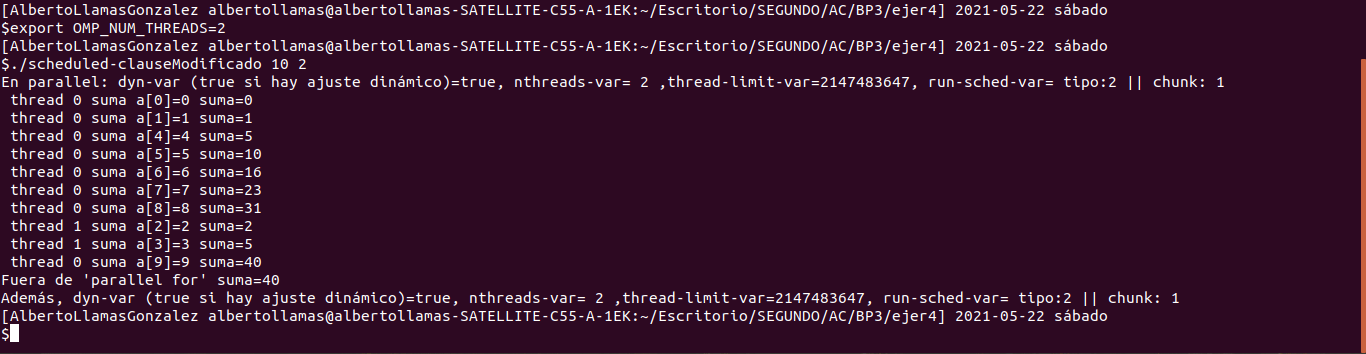
|  |
| --- |
| ../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017-../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017- |

**CAPTURAS DE PANTALLA:**

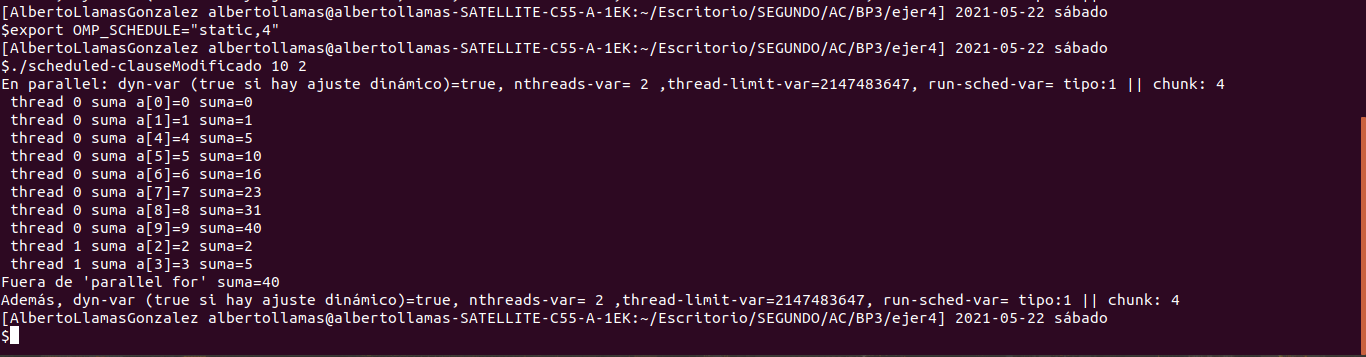
**En la primera ejecución cambio la asignación dinámica, activándola: export OMP\_DYNAMIC=TRUE**

****

**En la segunda ejecución cambio el número de threads usados, con export OMP\_NUM\_THREADS**

****

**En la tercera ejecución cambio la planificación, export OMP\_SCHEDULE="static, 4"**

****

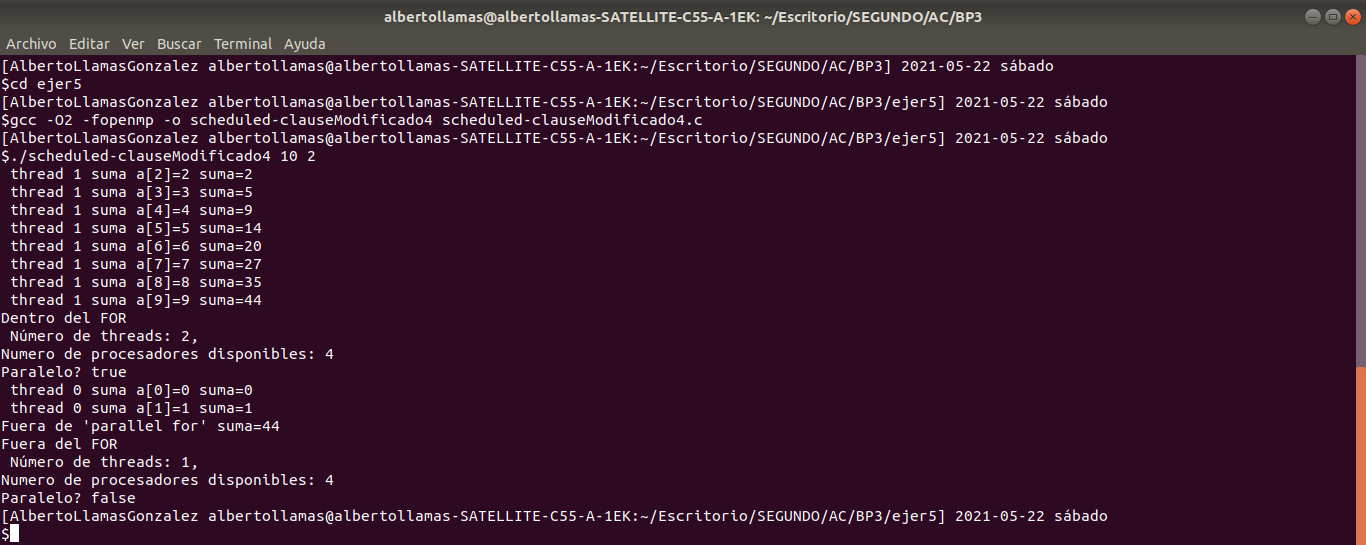
**RESPUESTA**: Los valores no cambian entre la zona paralela y la zona no paralela, y esto es porque cada variable de control es general al sistema donde se está ejecutando la tarea. Cuando modificamos una variable de entorno, modificamos la variable de control que está asignada a dicha de entorno.

**5.** Usar en el ejemplo anterior las funciones omp\_get\_num\_threads(), omp\_get\_num\_procs() y omp\_in\_parallel() dentro y fuera de la región paralela. Imprimir los valores que obtienen estas funciones dentro (lo debe imprimir sólo uno de los threads) y fuera de la región paralela. Incorporar en su cuaderno de prácticas volcados de pantalla con los resultados de ejecución obtenidos. Indicar en qué funciones se obtienen valores distintos dentro y fuera de la región paralela.

**CAPTURA CÓDIGO FUENTE**: scheduled-clauseModificado4.c

|  |
| --- |
| ../../../../../../Downloads/ejer/ejer5/Captura%20de%20pantalla%20de%202021-05-22%2016-33-13.png |

**CAPTURAS DE PANTALLA:**

****

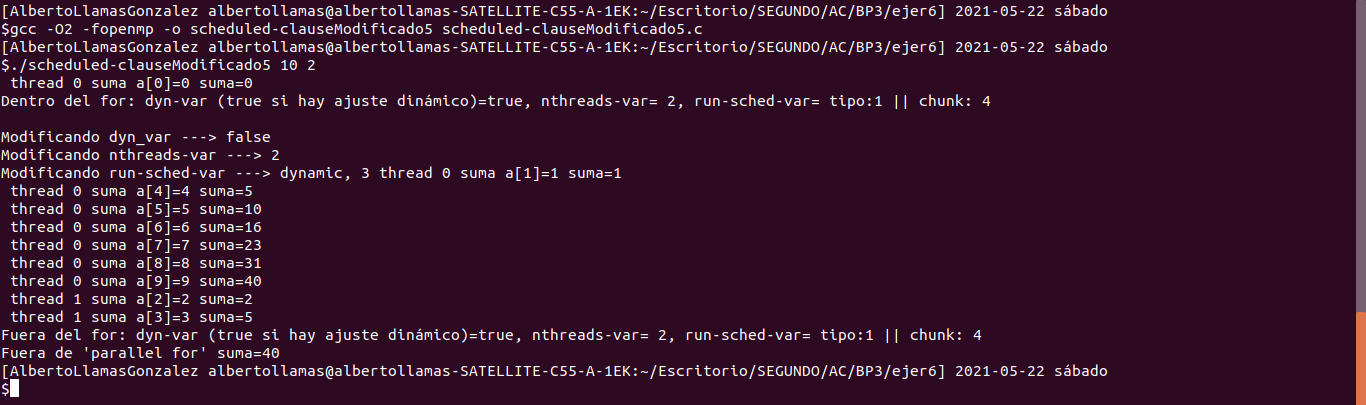
**RESPUESTA**: Se obtienen valores distintos en el número de threads que intervienen, ya que en una parte obtiene el valor de los threads creados en la paralelización, y fuera del parallel solo obtiene el número 1 (única hebra que ejecuta dicha parte del código, la hebra master). Num-procs es idéntico en ambas partes ya que en eso no influye la paralelización, y la variable in-parallel nos informa si estamos en región paralela, por lo cual, dentro devuelve un 1 y fuera un 0.

**6.** Añadir al programa scheduled-clause.c lo necesario para, usando funciones, modificar las variables de control dyn-var, nthreads-var y run-sched-var dentro de la región paralela y fuera de la región paralela. En la modificación de run-sched-var se debe usar un valor de kind distinto al utilizado en la cláusula schedule(). Añadir lo necesario para imprimir el contenido de estas variables antes y después de cada una de las dos modificaciones. Comentar los resultados.

**CAPTURA CÓDIGO FUENTE**: scheduled-clauseModificado5.c

|  |
| --- |
| ../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017-../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017-../../../../../../Downloads/ejer/ejer6/Captura%20de%20pantalla%20de%202021-05-22%2016-32-57.png |

**CAPTURAS DE PANTALLA:**

****

**RESPUESTA**: El resultado en la región paralela y fuera de ella es el mismo ya que hemos modificado los valores fuera y dentro.

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

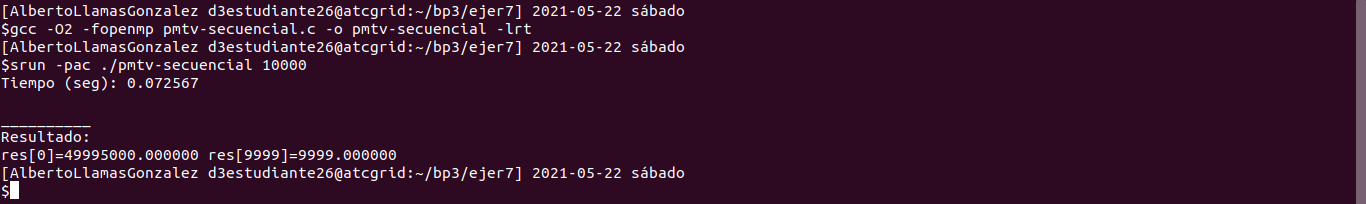
**7.** Implementar un programa secuencial en C que multiplique una matriz triangular inferior por un vector (use variables dinámicas y tipo de datos double). Comparar el orden de complejidad y el número total de operaciones (sumas y productos) de este código respecto al que implementó para el producto matriz por vector.

NOTAS: (1) el número de filas/columnas debe ser un argumento de entrada; (2) se debe inicializar las matrices antes del cálculo; (3) se debe imprimir siempre la primera y última componente del resultado antes de que termine el programa.

**CAPTURA CÓDIGO FUENTE**: pmtv-secuencial.c

|  |
| --- |
| ../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017-../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017-../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017- |

**CAPTURAS DE PANTALLA:**

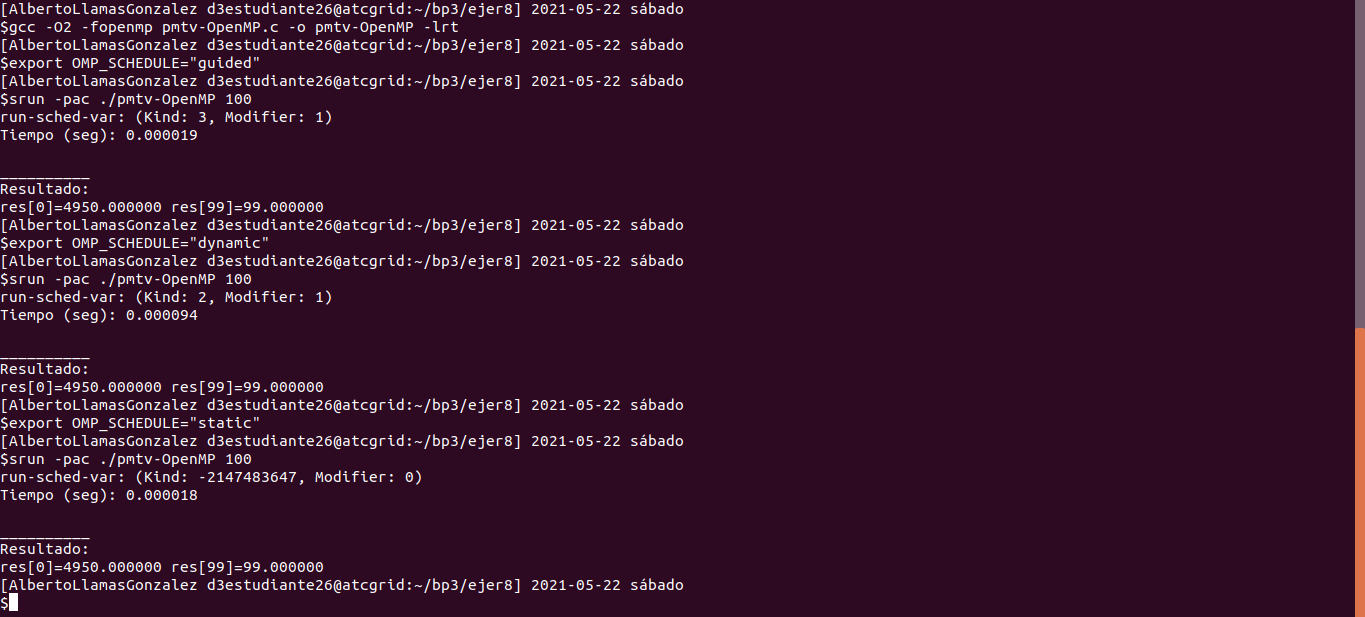
****

**8.** Implementar en paralelo la multiplicación de una matriz triangular inferior por un vector a partir del código secuencial realizado para el ejercicio anterior utilizando la directiva for de OpenMP. El código debe repartir entre los threads las iteraciones del bucle que recorre las filas. La inicialización de los datos la debe hacer el thread 0. Dibujar en el cuaderno de prácticas la descomposición de dominio utilizada (Lección 4/Tema 2) en el código paralelo implementado para asignar tareas a los threads (Lección 5/Tema 2). Añadir lo necesario para que el usuario pueda fijar la planificación de tareas usando la variable de entorno OMP\_SCHEDULE. Mostrar en una captura de pantalla que el código resultante funciona correctamente. NOTA: usar para generar los valores aleatorios, por ejemplo, drand48\_r().

**CAPTURA CÓDIGO FUENTE**: pmtv-OpenMP.c

|  |
| --- |
| ../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017-../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017-../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-22%2017-5/Captura%20de%20pantalla%20de%202021-05-22%2017- |

**CAPTURAS DE PANTALLA:**



**9.** Contestar a las siguientes preguntas sobre el código del ejercicio anterior:

**(a)** ¿Qué número de operaciones de multiplicación y qué número de operaciones de suma realizan cada uno de los threads en la asignación static con monotonic y un chunk de 1?

**RESPUESTA**: El comportamiento de static ha sido definido anteriormente, el número de instrucciones se distribuye en round-robin a cada thread, en bloques de tamaño del chunk.

**(b)** Con la asignación dynamic y guided, ¿qué cree que debe ocurrir con el número de operaciones de multiplicación y suma que realizan cada uno de los threads?

**RESPUESTA**: En el caso de dynamic y guided, esto varía. Es muy probable que un thread acumule varias iteraciones por terminar el primero.

**(c)** ¿Qué alternativa ofrece mejores prestaciones? Razonar la respuesta.

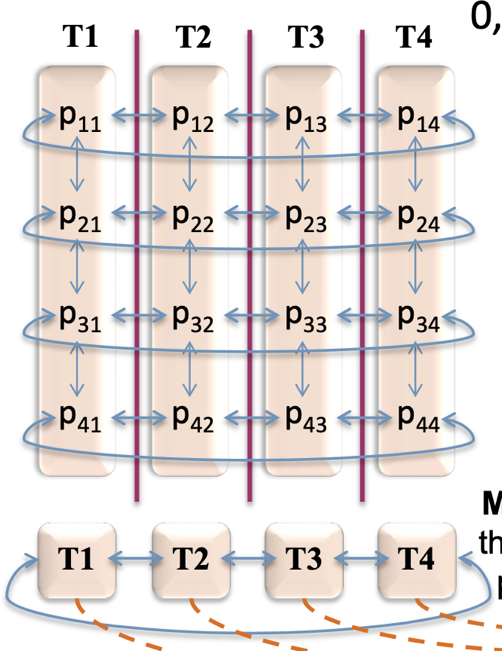
**RESPUESTA**: La que mejores prestaciones ofrece es la asignación *guided* ya que el tamaño de bloque (no iteraciones que restan dividido por no threads) va menguando lo cual lleva a que haya una mayor y mejor distribución del trabajo entre las hebras, trabajando todas por igual lo que evita que hebras se sobrecarguen.

**10.** Obtener en atcgrid los tiempos de ejecución del código paralelo (usando, como siempre, -O2 al compilar) que multiplica una matriz triangular por un vector con las alternativas de planificación static, dynamic y guided para chunk de 1, 64 y el chunk por defecto para la alternativa (con monotonic en todos los casos). Usar un tamaño de vector N múltiplo del número de cores y de 64 que esté entre 11520 y 23040. El número de threads en las ejecuciones debe coincidir con el número de núcleos del computador. Rellenar la Tabla 3 dos veces con los tiempos obtenidos. Representar el tiempo para static, dynamic y guided en función del tamaño del chunk en una gráfica (representar los valores de las dos tablas). Incluir los scripts utilizado en el cuaderno de prácticas. NOTA: Nunca ejecute en atcgrid código que imprima todos los componentes del resultado.

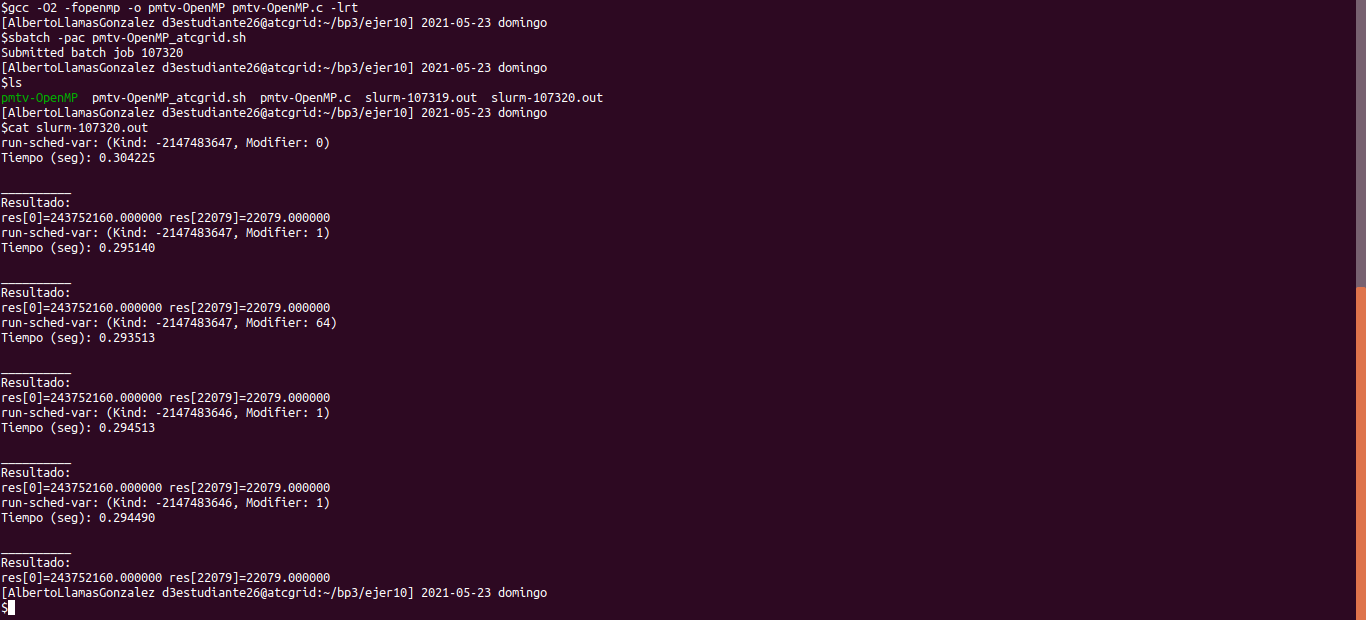
**CAPTURA CÓDIGO FUENTE**: pmtv-OpenMP.c

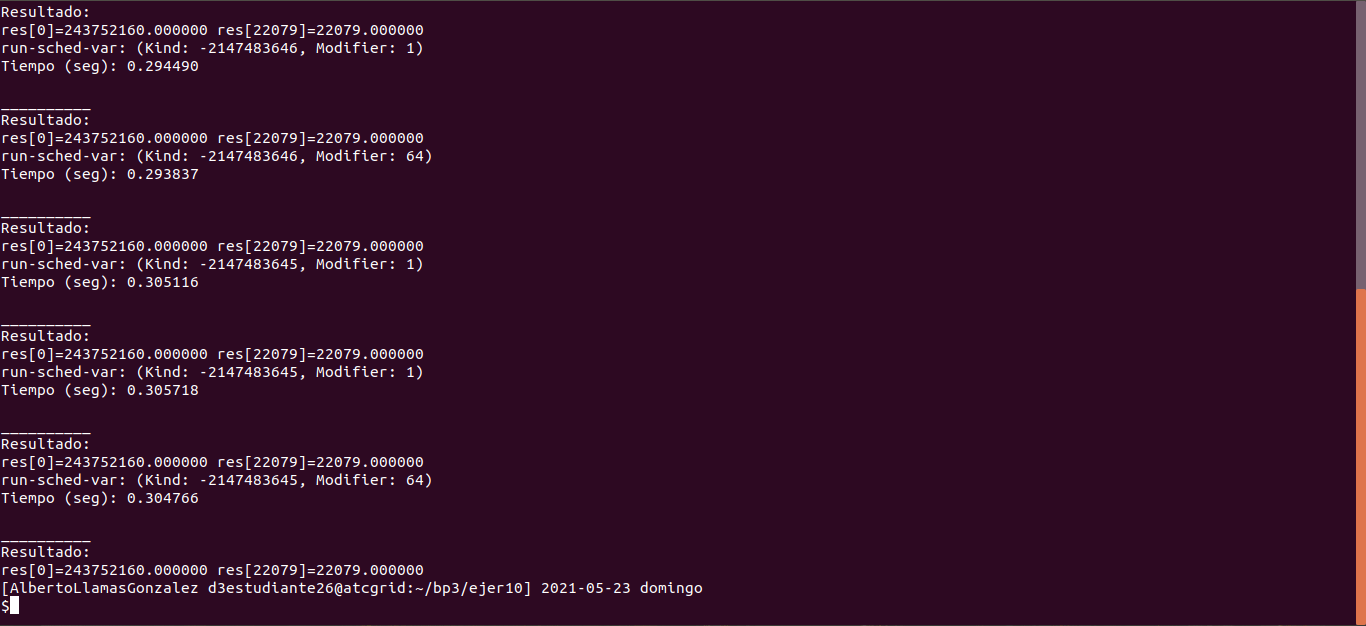
|  |
| --- |
| ../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-23%2000-/Captura%20de%20pantalla%20de%202021-05-23%2000- |

**DESCOMPOSICIÓN DE DOMINIO:**

****

**CAPTURAS DE PANTALLA:**

****

****

**TABLA RESULTADOS, SCRIPT Y GRÁFICA atcgrid**

**SCRIPT:** pmvt-OpenMP\_atcgrid.sh

|  |
| --- |
| ../../../../../../Downloads/Captura%20de%20pantalla%20de%202021-05-23%2000-/Captura%20de%20pantalla%20de%202021-05-23%2000- |

|  |
| --- |
| 1. Tiempos de ejecución de la versión paralela del producto de una matriz triangular por un vector para vectores de tamaño N=  (solo se ha paralelizado el producto, no la inicialización de los datos). |
| |  |  |  |  | | --- | --- | --- | --- | | **Chunk** | **Static** | **Dynamic** | **Guided** | | **por defecto** |  |  |  | | **1** |  |  |  | | **64** |  |  |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Chunk** | **Static** | **Dynamic** | **Guided** | | **por defecto** |  |  |  | | **1** |  |  |  | | **64** |  |  |  | |