TP3 HPC

Villalba Guillermo David , Martin Gustavo Nicolas
DNI: 32944121 , DNI:39375821
Lunes, Grupo 6

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina

1 Desarrollo

Ejercicio 1 - Hola Mundo estilo SOA

1.a) Identifique los 3 componentes de openMP en el ejercicio 1.

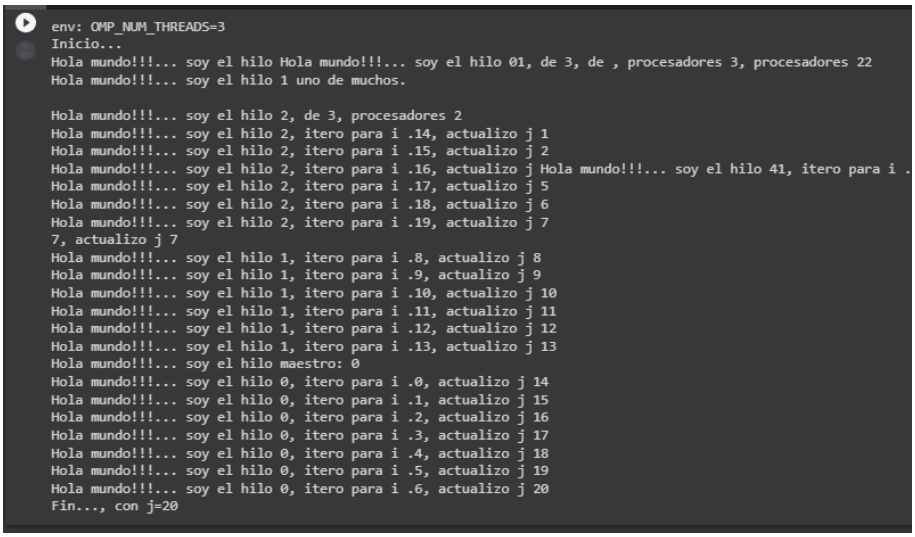
- Directivas del compilador. Por ejemplo: #pragma omp parallel
- Funciones de la librería de openMP: omp_get_num_threads()
- Variables de ambiente que puede utilizarse en la solución: %env OMP_NUM_THREADS= XXXXX

1.b) Realice pruebas modificando la cantidad de hilos (1, 3, 7).

RESULTADOS
THREADS: 1

```
env: OMP_NUM_THREADS=1
Inicio...
Hola mundo!!!... soy el hilo 0, de 1, procesadores 2
Hola mundo!!!... soy el hilo 0 uno de muchos.
Hola mundo!!!... soy el hilo maestro: 0
Hola mundo!!!... soy el hilo 0, itero para i .0, actualizo j 1
Hola mundo!!!... soy el hilo 0, itero para i .1, actualizo j 2
Hola mundo!!!... soy el hilo 0, itero para i .2, actualizo j 3
Hola mundo!!!... soy el hilo 0, itero para i .3, actualizo j 4
Hola mundo!!!... soy el hilo 0, itero para i .4, actualizo j 5
Hola mundo!!!... soy el hilo 0, itero para i .5, actualizo j 6
Hola mundo!!!... soy el hilo 0, itero para i .6, actualizo j 7
Hola mundo!!!... soy el hilo 0, itero para i .7, actualizo j 8
Hola mundo!!!... soy el hilo 0, itero para i .8, actualizo j 9
Hola mundo!!!... soy el hilo 0, itero para i .9, actualizo j 10
Hola mundo!!!... soy el hilo 0, itero para i .10, actualizo j 11
Hola mundo!!!... soy el hilo 0, itero para i .11, actualizo j 12
Hola mundo!!!... soy el hilo 0, itero para i .12, actualizo j 13
Hola mundo!!!... soy el hilo 0, itero para i .13, actualizo j 14
Hola mundo!!!... soy el hilo 0, itero para i .14, actualizo j 15
Hola mundo!!!... soy el hilo 0, itero para i .15, actualizo j 16
Hola mundo!!!... soy el hilo 0, itero para i .16, actualizo j 17
Hola mundo!!!... soy el hilo 0, itero para i .17, actualizo j 18
Hola mundo!!!... soy el hilo 0, itero para i .18, actualizo j 19
Hola mundo!!!... soy el hilo 0, itero para i .19, actualizo j 20
Fin..., con j=20
```

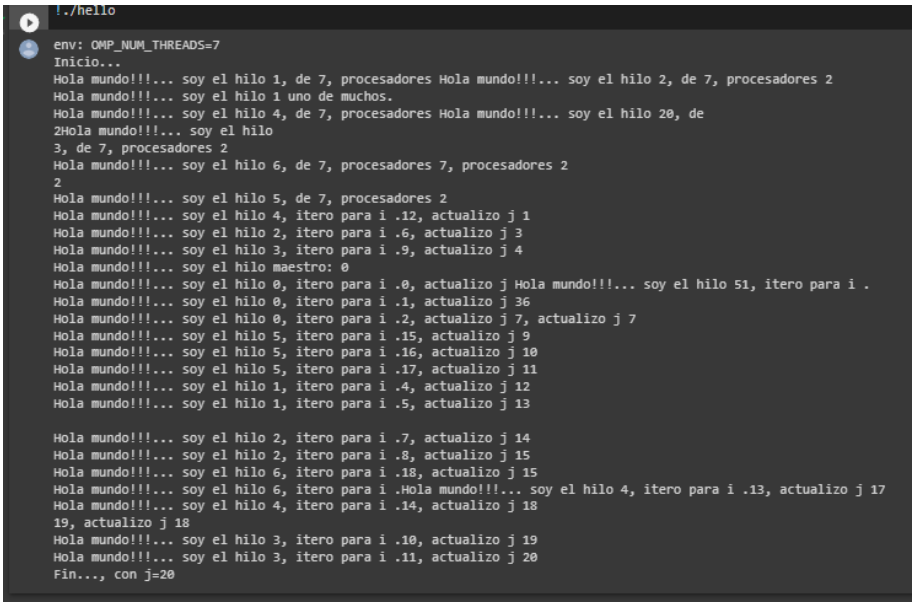
THREADS: 3



```
env: OMP_NUM_THREADS=3
Inicio...
Hola mundo!!!... soy el hilo Hola mundo!!!... soy el hilo 01, de 3, de , procesadores 3, procesadores 22
Hola mundo!!!... soy el hilo 1 uno de muchos.

Hola mundo!!!... soy el hilo 2, de 3, procesadores 2
Hola mundo!!!... soy el hilo 2, itero para i .14, actualizo j 1
Hola mundo!!!... soy el hilo 2, itero para i .15, actualizo j 2
Hola mundo!!!... soy el hilo 2, itero para i .16, actualizo j Hola mundo!!!... soy el hilo 41, itero para i .
Hola mundo!!!... soy el hilo 2, itero para i .17, actualizo j 5
Hola mundo!!!... soy el hilo 2, itero para i .18, actualizo j 6
Hola mundo!!!... soy el hilo 2, itero para i .19, actualizo j 7
7, actualizo j 7
Hola mundo!!!... soy el hilo 1, itero para i .8, actualizo j 8
Hola mundo!!!... soy el hilo 1, itero para i .9, actualizo j 9
Hola mundo!!!... soy el hilo 1, itero para i .10, actualizo j 10
Hola mundo!!!... soy el hilo 1, itero para i .11, actualizo j 11
Hola mundo!!!... soy el hilo 1, itero para i .12, actualizo j 12
Hola mundo!!!... soy el hilo 1, itero para i .13, actualizo j 13
Hola mundo!!!... soy el hilo maestro: 0
Hola mundo!!!... soy el hilo 0, itero para i .0, actualizo j 14
Hola mundo!!!... soy el hilo 0, itero para i .1, actualizo j 15
Hola mundo!!!... soy el hilo 0, itero para i .2, actualizo j 16
Hola mundo!!!... soy el hilo 0, itero para i .3, actualizo j 17
Hola mundo!!!... soy el hilo 0, itero para i .4, actualizo j 18
Hola mundo!!!... soy el hilo 0, itero para i .5, actualizo j 19
Hola mundo!!!... soy el hilo 0, itero para i .6, actualizo j 20
Fin..., con j=20
```

THREAD: 7



```
!./hello

env: OMP_NUM_THREADS=7
Inicio...
Hola mundo!!!... soy el hilo 1, de 7, procesadores Hola mundo!!!... soy el hilo 2, de 7, procesadores 2
Hola mundo!!!... soy el hilo 1 uno de muchos.
Hola mundo!!!... soy el hilo 4, de 7, procesadores Hola mundo!!!... soy el hilo 20, de
2Hola mundo!!!... soy el hilo
3, de 7, procesadores 2
Hola mundo!!!... soy el hilo 6, de 7, procesadores 7, procesadores 2
2
Hola mundo!!!... soy el hilo 5, de 7, procesadores 2
Hola mundo!!!... soy el hilo 4, itero para i .12, actualizo j 1
Hola mundo!!!... soy el hilo 2, itero para i .6, actualizo j 3
Hola mundo!!!... soy el hilo 3, itero para i .9, actualizo j 4
Hola mundo!!!... soy el hilo maestro: 0
Hola mundo!!!... soy el hilo 0, itero para i .0, actualizo j Hola mundo!!!... soy el hilo 51, itero para i .
Hola mundo!!!... soy el hilo 0, itero para i .1, actualizo j 36
Hola mundo!!!... soy el hilo 0, itero para i .2, actualizo j 7, actualizo j 7
Hola mundo!!!... soy el hilo 5, itero para i .15, actualizo j 9
Hola mundo!!!... soy el hilo 5, itero para i .16, actualizo j 10
Hola mundo!!!... soy el hilo 5, itero para i .17, actualizo j 11
Hola mundo!!!... soy el hilo 1, itero para i .4, actualizo j 12
Hola mundo!!!... soy el hilo 1, itero para i .5, actualizo j 13

Hola mundo!!!... soy el hilo 2, itero para i .7, actualizo j 14
Hola mundo!!!... soy el hilo 2, itero para i .8, actualizo j 15
Hola mundo!!!... soy el hilo 6, itero para i .18, actualizo j 15
Hola mundo!!!... soy el hilo 6, itero para i .Hola mundo!!!... soy el hilo 4, itero para i .13, actualizo j 17
Hola mundo!!!... soy el hilo 4, itero para i .14, actualizo j 18
19, actualizo j 18
Hola mundo!!!... soy el hilo 3, itero para i .10, actualizo j 19
Hola mundo!!!... soy el hilo 3, itero para i .11, actualizo j 20
Fin..., con j=20
```

1.c) ¿Cuál es la diferencia las directivas master y single?

La diferencia es que la directiva Master hace que el código siguiente reaccione al momento que se ejecuta en el hilo principal del programa y la directiva single, hace que se pueda ejecutar el código en el hilo secundario, no necesariamente puede ser el hilo principal que se encuentre corriendo.

1.d) En el for: ¿Que sucede con el valor de j, sí se parametriza como lastprivate?

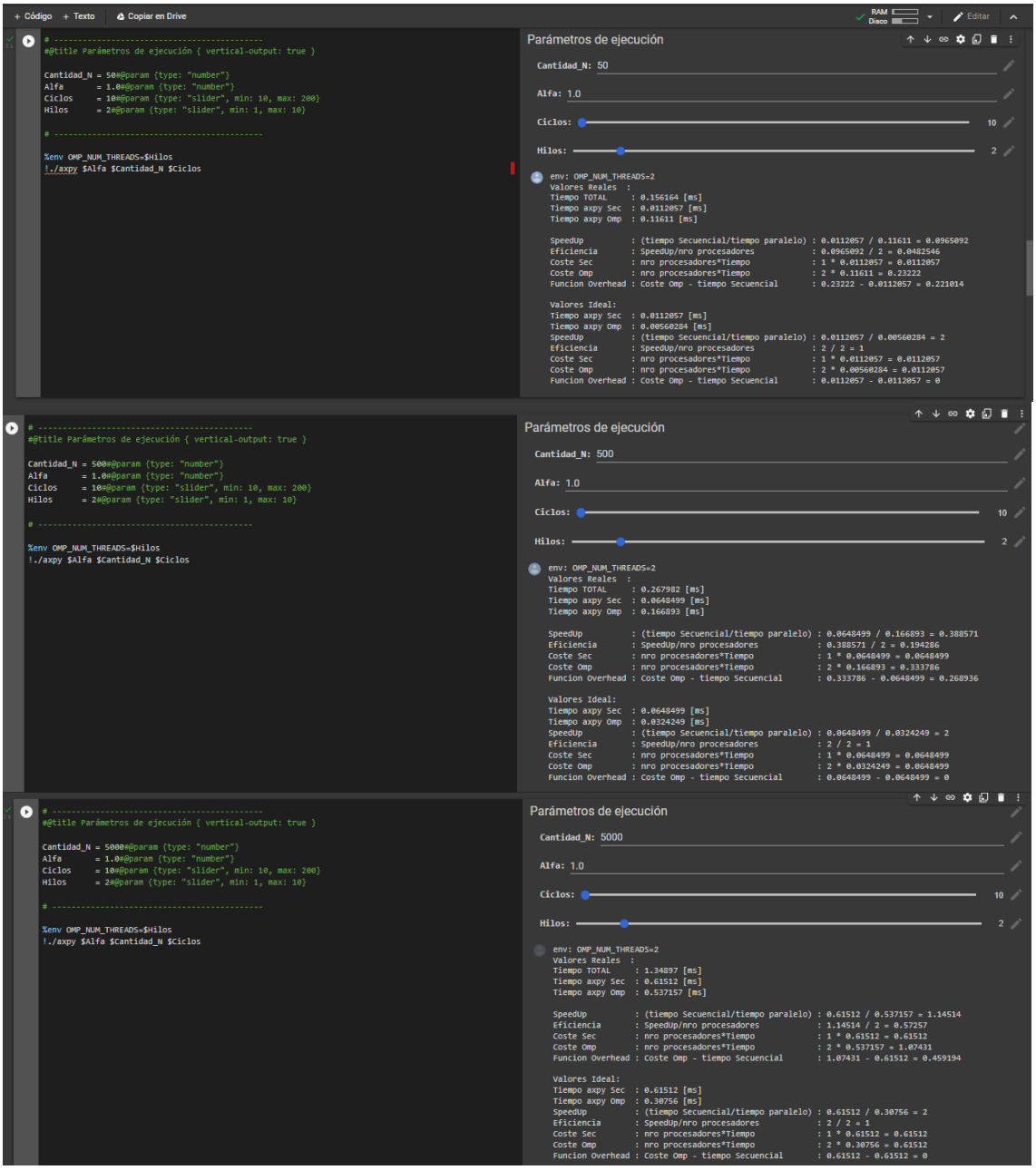
Si j se parametriza como lastprivate, lo que sucederá es que en la última ejecución del for, se copiará el valor resultante de j para el context principal (main)

2.a) Ejecute el ejercicio 1.2.2:

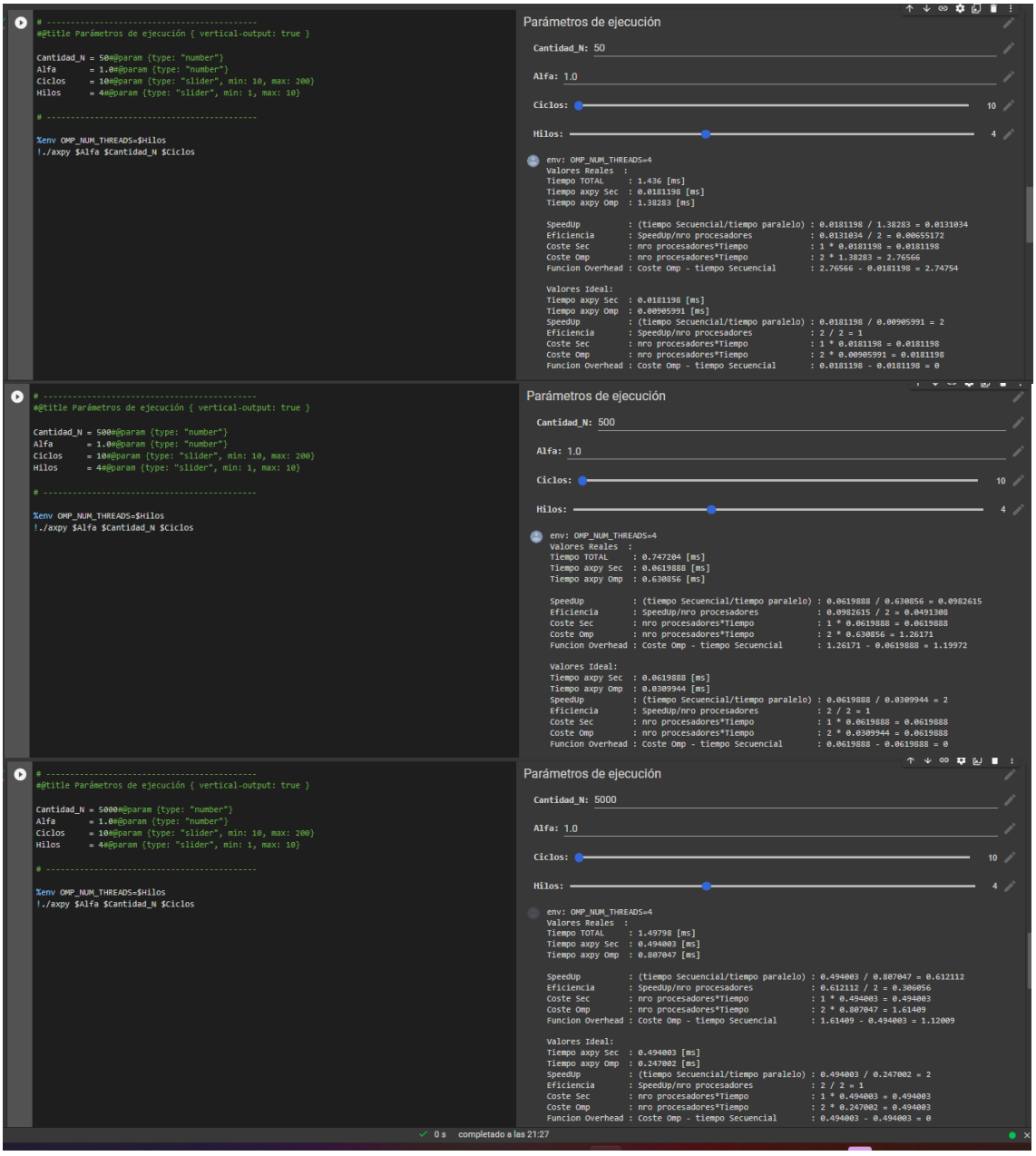
- Con la variable cantidad_N para: 50, 500, 5.000.

- Con los valores hilos_ 2, 4, 6.

Dos hilos:



Cuatro hilos:



```
# ----------------------------------------------
#@title Parámetros de ejecución { vertical-output: true }

Cantidad_N = 50#@param {type: "number"}
Alfa       = 1.0#@param {type: "number"}
Ciclos     = 10#@param {type: "slider", min: 10, max: 200}
Hilos      = 4#@param {type: "slider", min: 1, max: 10}

# ----------------------------------------------

%env OMP_NUM_THREADS=$Hilos
!./axpy $Alfa $Cantidad_N $Ciclos
```

Parámetros de ejecución

Cantidad_N: 50

Alfa: 1.0

Ciclos: 10

Hilos: 4

```
env: OMP_NUM_THREADS=4
Valores Reales  :
Tiempo TOTAL     : 1.436 [ms]
Tiempo axpy Sec  : 0.0181198 [ms]
Tiempo axpy Omp  : 1.38283 [ms]

SpeedUp          : (tiempo Secuencial/tiempo paralelo) : 0.0181198 / 1.38283 = 0.0131034
Eficiencia       : SpeedUp/nro procesadores            : 0.0131034 / 2 = 0.00655172
Coste Sec        : nro procesadores*Tiempo             : 1 * 0.0181198 = 0.0181198
Coste Omp        : nro procesadores*Tiempo             : 2 * 1.38283 = 2.76566
Funcion Overhead : Coste Omp - tiempo Secuencial       : 2.76566 - 0.0181198 = 2.74754

Valores Ideal:
Tiempo axpy Sec  : 0.0181198 [ms]
Tiempo axpy Omp  : 0.00905991 [ms]
SpeedUp          : (tiempo Secuencial/tiempo paralelo) : 0.0181198 / 0.00905991 = 2
Eficiencia       : SpeedUp/nro procesadores            : 2 / 2 = 1
Coste Sec        : nro procesadores*Tiempo             : 1 * 0.0181198 = 0.0181198
Coste Omp        : nro procesadores*Tiempo             : 2 * 0.00905991 = 0.0181198
Funcion Overhead : Coste Omp - tiempo Secuencial       : 0.0181198 - 0.0181198 = 0
```

```
# ----------------------------------------------
#@title Parámetros de ejecución { vertical-output: true }

Cantidad_N = 500#@param {type: "number"}
Alfa       = 1.0#@param {type: "number"}
Ciclos     = 10#@param {type: "slider", min: 10, max: 200}
Hilos      = 4#@param {type: "slider", min: 1, max: 10}

# ----------------------------------------------

%env OMP_NUM_THREADS=$Hilos
!./axpy $Alfa $Cantidad_N $Ciclos
```

Parámetros de ejecución

Cantidad_N: 500

Alfa: 1.0

Ciclos: 10

Hilos: 4

```
env: OMP_NUM_THREADS=4
Valores Reales  :
Tiempo TOTAL     : 0.747204 [ms]
Tiempo axpy Sec  : 0.0619888 [ms]
Tiempo axpy Omp  : 0.630856 [ms]

SpeedUp          : (tiempo Secuencial/tiempo paralelo) : 0.0619888 / 0.630856 = 0.0982615
Eficiencia       : SpeedUp/nro procesadores            : 0.0982615 / 2 = 0.0491308
Coste Sec        : nro procesadores*Tiempo             : 1 * 0.0619888 = 0.0619888
Coste Omp        : nro procesadores*Tiempo             : 2 * 0.630856 = 1.26171
Funcion Overhead : Coste Omp - tiempo Secuencial       : 1.26171 - 0.0619888 = 1.19972

Valores Ideal:
Tiempo axpy Sec  : 0.0619888 [ms]
Tiempo axpy Omp  : 0.0309944 [ms]
SpeedUp          : (tiempo Secuencial/tiempo paralelo) : 0.0619888 / 0.0309944 = 2
Eficiencia       : SpeedUp/nro procesadores            : 2 / 2 = 1
Coste Sec        : nro procesadores*Tiempo             : 1 * 0.0619888 = 0.0619888
Coste Omp        : nro procesadores*Tiempo             : 2 * 0.0309944 = 0.0619888
Funcion Overhead : Coste Omp - tiempo Secuencial       : 0.0619888 - 0.0619888 = 0
```

```
# ----------------------------------------------
#@title Parámetros de ejecución { vertical-output: true }

Cantidad_N = 5000#@param {type: "number"}
Alfa       = 1.0#@param {type: "number"}
Ciclos     = 10#@param {type: "slider", min: 10, max: 200}
Hilos      = 4#@param {type: "slider", min: 1, max: 10}

# ----------------------------------------------

%env OMP_NUM_THREADS=$Hilos
!./axpy $Alfa $Cantidad_N $Ciclos
```

Parámetros de ejecución

Cantidad_N: 5000

Alfa: 1.0

Ciclos: 10

Hilos: 4

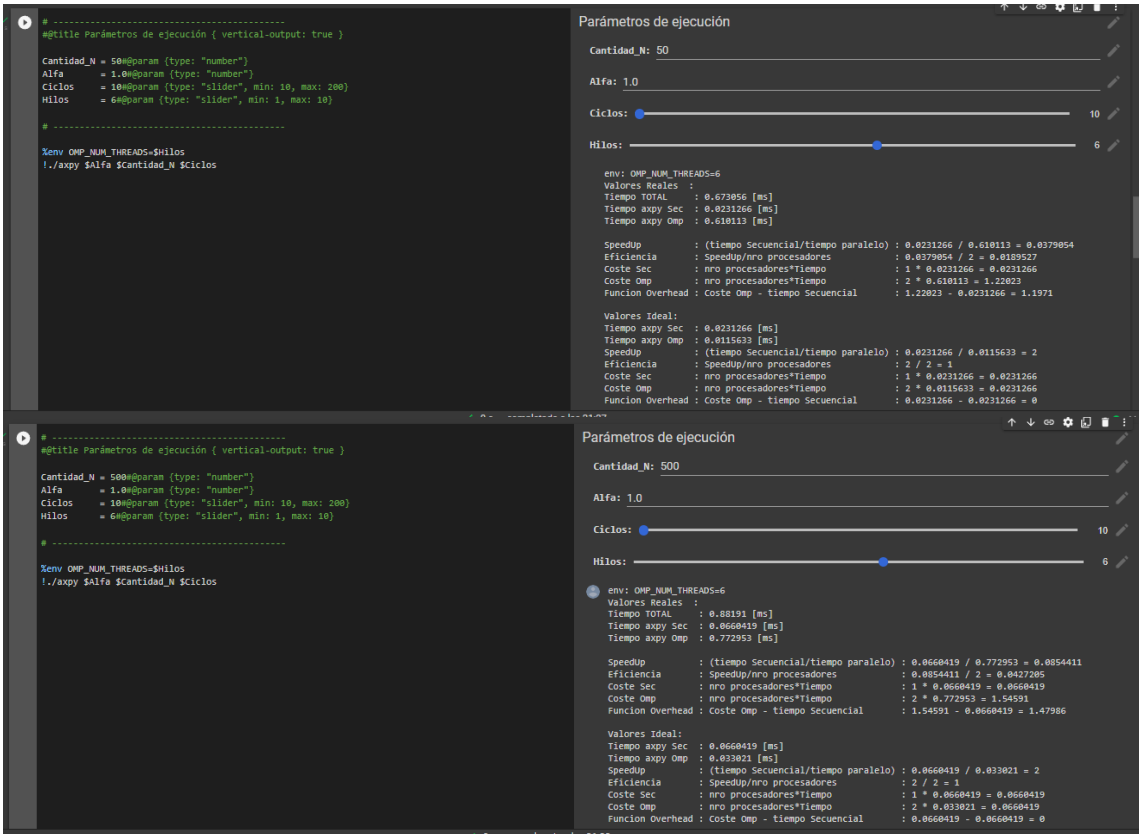
```
env: OMP_NUM_THREADS=4
Valores Reales  :
Tiempo TOTAL     : 1.49798 [ms]
Tiempo axpy Sec  : 0.494003 [ms]
Tiempo axpy Omp  : 0.807047 [ms]

SpeedUp          : (tiempo Secuencial/tiempo paralelo) : 0.494003 / 0.807047 = 0.612112
Eficiencia       : SpeedUp/nro procesadores            : 0.612112 / 2 = 0.306056
Coste Sec        : nro procesadores*Tiempo             : 1 * 0.494003 = 0.494003
Coste Omp        : nro procesadores*Tiempo             : 2 * 0.807047 = 1.61409
Funcion Overhead : Coste Omp - tiempo Secuencial       : 1.61409 - 0.494003 = 1.12009

Valores Ideal:
Tiempo axpy Sec  : 0.494003 [ms]
Tiempo axpy Omp  : 0.247002 [ms]
SpeedUp          : (tiempo Secuencial/tiempo paralelo) : 0.494003 / 0.247002 = 2
Eficiencia       : SpeedUp/nro procesadores            : 2 / 2 = 1
Coste Sec        : nro procesadores*Tiempo             : 1 * 0.494003 = 0.494003
Coste Omp        : nro procesadores*Tiempo             : 2 * 0.247002 = 0.494003
Funcion Overhead : Coste Omp - tiempo Secuencial       : 0.494003 - 0.494003 = 0
```

0 s completado a las 21:27

Seis hilos:



2.b) Repita la prueba (a), con ciclos 1, 10, 30. ¿Cuál fue la diferencia?

Al ejecutar con aumentos de ciclos, va disminuyendo el rendimiento del programa.

2.c) ¿Por qué el SpeedUp no mejora a medida que aumentan la cantidad de hilos?

no mejora, ya que el hardware no tiene la capacidad de correr en paralelo la cantidad de hilos que uno quiera. A medida que se excede la cantidad de hilos respecto la cantidad de cores del procesador, el rendimiento irá disminuyendo por temas de overhead y concurrencia.

En el caso ejecutado, solo existen dos cores físicos, por lo que cantidad de hilos > 2 implica una disminución de rendimiento

2.d) ¿Qué sucede con la eficiencia a medida que aumentan la cantidad_N?¿Porque no llega al valor ideal?

No llega al valor ideal por temas de hardware del Sistema en donde corre el programa

Ejercicio 2 - Hola Mundo con MPI

a) ¿Qué función realiza la instancia maestra? ¿Qué función realizan las instancias esclavas?

La instancia maestra ejecuta la función init() crea las tareas y las reparte entre las instancias esclavas. A medida que las instancias esclavas le responden la instancia maestra les vuelve a asignar una tarea hasta que no haya mas tareas para asignar, entonces se queda esperando a que las instancias esclavas le respondan con su resultado.

Las instancias esclavas ejecutan la tarea que se les asigno y le responden a la instancia maestra dentro de un bucle infinito esperando una nueva tarea.

b) ¿Cuántas de esas instancias ejecuta la función main(), initWork() y doWork()?

Todas ejecutan la función main(), solo el maestro ejecuta el initWork(), y solo las instancias esclavas ejecutan dowork().

c) ¿Cómo se diferencian los mensajes de trabajo o de finalización?

Se diferencian con los tags WORK_FLAG = 1 y END_WORK_FLAG = 2, que la instancia maestra envía a las esclavas.

d) ¿Cómo implementaría la función BLAS axpy() con este patrón?¿Sería eficiente? Tips: Pide solo el planteo, no la implementación.

No sería posible agregar axpy ya que MPI implementa procesos pesados sin memoria compartida y axpy se implementa con memoria compartida. No tendría sentido desarrollar una lógica asociada a este concepto en este caso.

e) ¿Qué sucede cuando solo ejecuta con una sola instancia?

El programa ejecuta, pero no se realiza ninguna tarea ya que la única instancia es la maestra y no genera ninguna tarea.

f) Punto opcional: El código que ejecutan las instancias esclavas, tienen un error en su lógica. ¿Cómo se podría solucionar?

Ejercicio 3

a) El programa del punto 3.1.2 se encuentra dividido en bloques, que se identifican con nombres de colores. El problema consiste en ordenar los bloques, ya que se encuentran desordenados. Para ello, haga foco en cada

celda y utilice las flechas y , para reorganizar los bloques según el orden correcto.

Nombre del bloque	Orden correcto
Verde	Naranja
Rosa	Rosa
Blanco	Amarillo
Amarillo	Violeta
Gris	Gris
Naranja	Azul
Violeta	Blanco
Negro	Rojo
Rojo	Verde
Azul	Negro

Tips: Para ejecutar todas las celdas, ocúltelas con la flecha que está al lado del título "3.1.2 Ejecución del programa".

b) Punto opcional: Para verificar que el cálculo en GPU (junto con sus etapas) es correcto. Desarrolle la función axpy en forma secuencial (en lenguaje python) y compare los vectores resultados.

c) Realice pruebas para diferentes valores de cantidad_N (500, 5.000, 50.000, 500.000 y 5.000.000). Compare los resultados con los obtenidos en el ejercicio 1.2.1 (OpenMP y secuencial).

Tamaño	Tiempo secuencial	Tiempo openmp	Tiempo GPU
500	0,0648499	0,166893	0.356
5.000	0,61512	0,537157	0.459
50.000	9,89318	4,51803	0.275
500.000	57.8742	57.4911	0.387
5.000.000	544.325	448.715	4.411

Se puede observar que para N pequeños el tiempo secuencial es mejor, pero a medida que N crece el tiempo de GPU se incrementa mucho mas lento, por lo que para N grandes es mucho mejor.

d) A la comparación anterior, incluya como impactan el SpeedUP y Eficiencia al usar GPU.

e) A la comparación anterior, analice como fueron aumentando las dimensiones. ¿Se desperdiciaron muchos hilos? ¿Cómo se podría instanciar una cantidad de hilos exactamente igual al tamaño del vector?

f) ¿Cómo hace el programa, para almacenar el resultado en el vector r_cpu, si este vector nunca se parametriza en el kernel?

Copia a nivel posicion de memoria con la funcion “cuda.memcpy_dtoh(r_cpu, y_gpu)”

g) Compare la función del kernel con la función axpy() del ejercicio 1.2.2 (OpenMP). ¿Qué cambios nota en el algoritmo? ¿El índice funciona igual? ¿Qué paso con el ciclo for en la función del kernel?

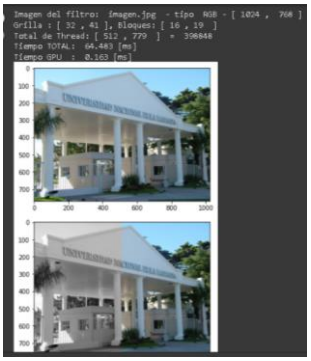
Ejercicio 4

a) Comparé el tiempo de respuesta con el ejercicio secuencial URL-wiki. ¿Cuál es la complejidad computacional del filtro secuencial?

Lo que pasa es que en el ejercico de prueba es que la complejidad computacional es O^2 lo tanto va a tardar mas que en el ejercico del tp ya que la complejidad computacional en este es constante (O1).

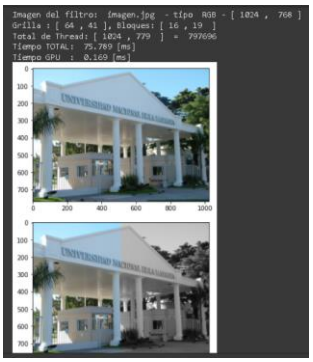
b) ¿Qué cambios realizaría para que se planifiquen la mitad de los hilos-gpu (sobre el total)? ¿Qué sucede con la imagen resultante? ¿Qué sucede con la velocidad de respuesta?

Dividi la cantidad de bloques x a la mitad.
En la imagen resultante solo se procesaron la mitad de los pixeles. No se observaron cambios grandes en la velocidad.



c) ¿Qué cambios realizaría para que solo se procese la parte derecha de la imagen? ¿Qué sucede con la parte izquierda en la imagen resultante? ¿Qué sucede con la velocidad de respuesta?

Para que solo se procese la parte derecha de la imagen agregue la condicion en el if del kernel_img si idx es mayor que ancho dividido 2. La imagen resultante tiene la mitad izquierda igual a la original y la derecha en escala de grises. La velocidad de respuesta se incremento levemente



d) ¿Qué sucede con el tiempo de respuesta si se utilizan bloques del tamaño máximo soportado por el GPU? Tips: ver ejemplo *"Prueba 0 - Hola Mundo GPU"*.

e) ¿Cúantos hilos se planifican de más? Tips: Los que no tengan condición verdadera en la condición dentro del kernel.

Hilos necesitados : 1024 * 768 = 786432
Hilos creados : 1024 * 779 = 797696
Hilos de mas: 797696 – 786432 = 11264

f) ¿Qué cambios hay que realizar, para que solo se convierta a la mitad de la imagen? NOTA: La otra mitad de la imagen, debe tener la imagen original.

En el if de la funcion kernel_img hay que preguntar si idx es menor a ancho dividido dos hacer la conversion, y agregar un else para que a los demas pixeles los copie tal cual.