OpenMP coprocesadores

1. Consideraciones previas

- Se usará el compilador nvc de Nvidia, en particular, se utilizará la versión 21.2 que está instalado en el nodo atcgrid4 (se debe tener en cuenta que distintas versiones de nvc podrían generar distinto código ejecutable).
- El objetivo de estos ejercicios es habituarse a la organización de la GPU y al compilador, y entender la sobrecarga que introduce el uso del coprocesador (GPU, en este caso).
- El compilador nvc espera que el código termine con un salto de línea
- Entregar este fichero en pdf (AC_OpenMPCoprocesadores_ApellidosNombre.pdf) en un zip junto con los códigos implementados (AC_OpenMPCoprocesadores_ApellidosNombre.zip).
- Los códigos debe mantenerlos en su directorio de atcgrid hasta final del curso.
- El tamaño de la letra en las capturas debe ser similar al tamaño de la letra en este documento.
- En las capturas de pantalla se debe ver el nombre del usuario, fecha y hora.
  - Debe modificar el prompt en los computadores que utilice para que aparezca su nombre y apellidos, su usuario (`\u`), el computador (`\h`), el directorio de trabajo del bloque práctico (`\w`), la fecha (`\D`) completa (`%F`) y el día (`%A`). Para modificar el prompt utilice lo siguiente (si es necesario, use `export` delante):

    ```
    PS1="[NombreApellidos \u@\h:\w] \D{%F %A}\n$"
    ```
    (.bash_profile)

    donde NombreApellidos es su nombre seguido de sus apellidos, por ejemplo: JuanOrtuñoVilariño

2. Ejercicios basados en los ejemplos del seminario

1. **(a)** Compilar el ejemplo `omp_offload.c` del seminario en el nodo atcgrid4::

```
srun -pac4 -Aac nvc -O2 -openmp -mp=gpu omp_offload.c -o omp_offload_GPU
```

(`-openmp` para que tenga en cuenta las directivas OpenMP y `-mp=gpu` para que el código delimitado con `target` se genere para un dispositivo gpu)

Ejecutar `omp_offload_GPU` usando:

```
srun -pac4 -Aac omp_offload_GPU 36 3 32  > salida.txt
```

CONTENIDO FICHERO: `salida.txt` (**destaque en el resultado de la ejecución con colores las respuestas a las preguntas (b)-(e))**

```
$cat salida.txt
Target device: 1
Tiempo:0.141300917
Iteracción 0, en thread 0/32 del team 0/3
Iteracción 1, en thread 1/32 del team 0/3
Iteracción 2, en thread 2/32 del team 0/3
Iteracción 3, en thread 3/32 del team 0/3
Iteracción 4, en thread 4/32 del team 0/3
Iteracción 5, en thread 5/32 del team 0/3
Iteracción 6, en thread 6/32 del team 0/3
Iteracción 7, en thread 7/32 del team 0/3
Iteracción 8, en thread 8/32 del team 0/3
Iteracción 9, en thread 9/32 del team 0/3
```

```
Iteracción 10, en thread 10/32 del team 0/3
Iteracción 11, en thread 11/32 del team 0/3
Iteracción 12, en thread 12/32 del team 0/3
Iteracción 13, en thread 13/32 del team 0/3
Iteracción 14, en thread 14/32 del team 0/3
Iteracción 15, en thread 15/32 del team 0/3
Iteracción 16, en thread 16/32 del team 0/3
Iteracción 17, en thread 17/32 del team 0/3
Iteracción 18, en thread 18/32 del team 0/3
Iteracción 19, en thread 19/32 del team 0/3
Iteracción 20, en thread 20/32 del team 0/3
Iteracción 21, en thread 21/32 del team 0/3
Iteracción 22, en thread 22/32 del team 0/3
Iteracción 23, en thread 23/32 del team 0/3
Iteracción 24, en thread 24/32 del team 0/3
Iteracción 25, en thread 25/32 del team 0/3
Iteracción 26, en thread 26/32 del team 0/3
Iteracción 27, en thread 27/32 del team 0/3
Iteracción 28, en thread 28/32 del team 0/3
Iteracción 29, en thread 29/32 del team 0/3
Iteracción 30, en thread 30/32 del team 0/3
Iteracción 31, en thread 31/32 del team 0/3
Iteracción 32, en thread 0/32 del team 1/3
Iteracción 33, en thread 1/32 del team 1/3
Iteracción 34, en thread 2/32 del team 1/3
Iteracción 35, en thread 3/32 del team 1/3
```

Contestar las siguientes preguntas **justificando la respuesta usando el contenido del fichero salida.txt**:

(b) ¿Cuántos equipos (*teams*) se han creado y cuántos se han usado realmente en la ejecución?

RESPUESTA: Se han creado 3 equipos pero en la ejecucion se han usado 2

(c) ¿Cuántos hilos (*theads*) se han creado en cada equipo y cuántos de esos hilos se han usado en la ejecución?

RESPUESTA: Se han creado 32 hilos en cada equipo. En el equipo 0 se han usado todos los threads y en el equipo 1 se han usado solo 4

(d) ¿Qué número de iteraciones se ha asignado a cada hilo?

RESPUESTA: Se ha asignado una iteracion a cada hilo

(e) ¿Qué número de iteraciones se ha asignado a cada equipo?

RESPUESTA: Se han asignado 4 iteraciones al equipo 1 y 32 al equipo 3

2. Eliminar en `opp_offload.c num_teams(nteams)` y `thread_limit(mthreads)` y la entrada como parámetros de `nteams` y `mthreads`. Llamar al código resultante `opp_offload2.c`. Compilar y ejecutar el código para poder contestar a las siguientes preguntas:

(a) ¿Qué número de equipos y de hilos por equipo se usan por defecto?

RESPUESTA:Se usan 48 equipos y 1024 hilos por defecto

CAPTURA (que muestre el envío a la cola y el resultado de la ejecución)

```
$sbatch -pac4 -Aac --wrap "nvc -O2 -openmp -mp=gpu omp_offload2.c -o omp_offload2_gpu"
Submitted batch job 249734
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$ls
daxpbyz32_ompoff.c   omp_offload.c   omp_offload2_gpu  pi.c         slurm-249716.out
daxpbyz32_ompoff2.c  omp_offload2.c  omp_offload_gpu   salida.txt   slurm-249734.out
```

```
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac omp_offload2_gpu 36 > salida.txt
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$cat salida.txt
Target device: 1
Tiempo:0.146375179
```

```
Target device: 1
Tiempo:0.146375179
Iteracción 0, en thread 0/1024 del team 0/48
Iteracción 1, en thread 1/1024 del team 0/48
Iteracción 2, en thread 2/1024 del team 0/48
Iteracción 3, en thread 3/1024 del team 0/48
Iteracción 4, en thread 4/1024 del team 0/48
Iteracción 5, en thread 5/1024 del team 0/48
Iteracción 6, en thread 6/1024 del team 0/48
Iteracción 7, en thread 7/1024 del team 0/48
Iteracción 8, en thread 8/1024 del team 0/48
Iteracción 9, en thread 9/1024 del team 0/48
Iteracción 10, en thread 10/1024 del team 0/48
Iteracción 11, en thread 11/1024 del team 0/48
Iteracción 12, en thread 12/1024 del team 0/48
Iteracción 13, en thread 13/1024 del team 0/48
Iteracción 14, en thread 14/1024 del team 0/48
Iteracción 15, en thread 15/1024 del team 0/48
Iteracción 16, en thread 16/1024 del team 0/48
Iteracción 17, en thread 17/1024 del team 0/48
Iteracción 18, en thread 18/1024 del team 0/48
Iteracción 19, en thread 19/1024 del team 0/48
Iteracción 20, en thread 20/1024 del team 0/48
Iteracción 21, en thread 21/1024 del team 0/48
Iteracción 22, en thread 22/1024 del team 0/48
Iteracción 23, en thread 23/1024 del team 0/48
Iteracción 24, en thread 24/1024 del team 0/48
Iteracción 25, en thread 25/1024 del team 0/48
Iteracción 26, en thread 26/1024 del team 0/48
Iteracción 27, en thread 27/1024 del team 0/48
Iteracción 28, en thread 28/1024 del team 0/48
Iteracción 29, en thread 29/1024 del team 0/48
Iteracción 30, en thread 30/1024 del team 0/48
Iteracción 31, en thread 31/1024 del team 0/48
Iteracción 32, en thread 32/1024 del team 0/48
Iteracción 33, en thread 33/1024 del team 0/48
Iteracción 34, en thread 34/1024 del team 0/48
Iteracción 35, en thread 35/1024 del team 0/48
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$
```

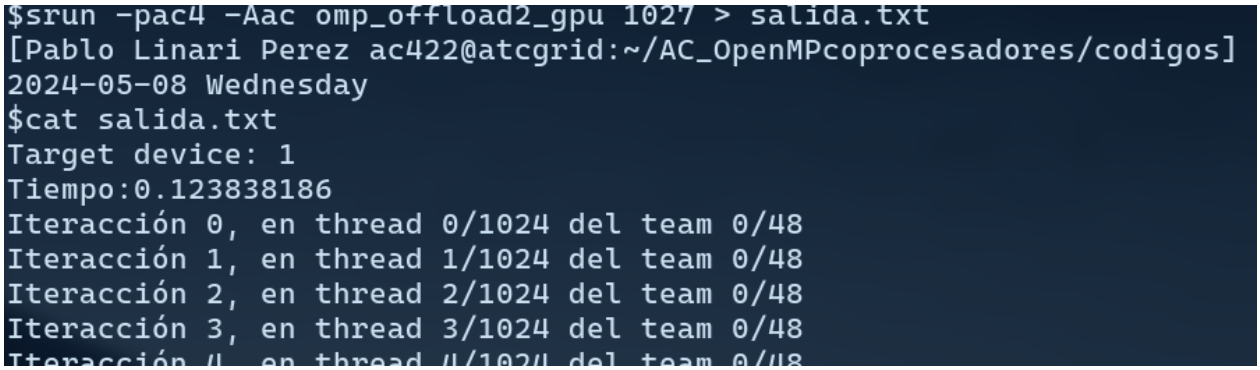
(b) ¿Es posible relacionar estos números con alguno de los parámetros, comentados en el seminario, que caracterizan al coprocesador que estamos usando? ¿Con cuáles?

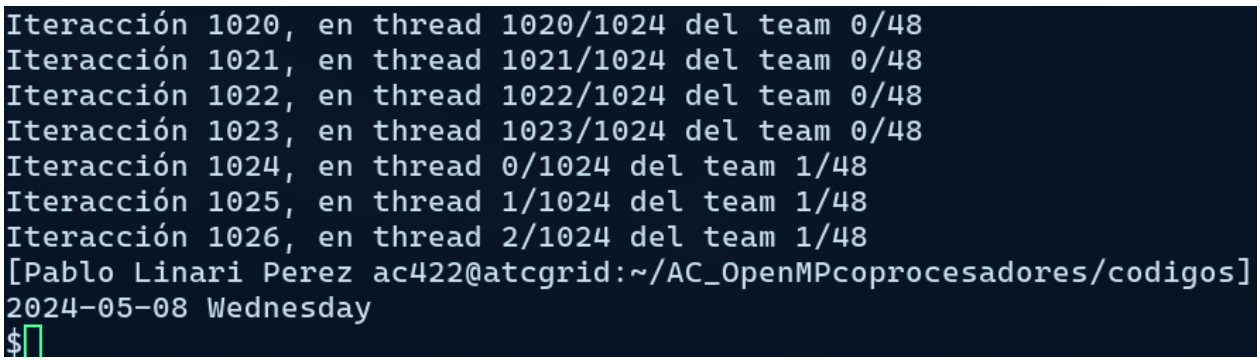
RESPUESTA:En la diapositiva 9 (Nvidia Quadro RTX 5000) aparecen los valores con los que se relacionan, concretamente con las filas Streaming Multiprocessor (SM) SIMT 48 y Máximo de threads por SM 1024. Estos son los valores que al ejecutar encontramos por defecto .

(c) ¿De qué forma se asignan por defecto las iteraciones del bucle a los **equipos** y a los **hilos** dentro de un equipo? Contestar además las siguientes preguntas: ¿a qué equipo y a qué hilo de ese equipo se asigna la iteración 2? Y ¿a qué

equipo y a qué hilo de ese equipo se asigna la iteración 1026, si la hubiera? (realizar las ejecuciones que se consideren necesarias para contestar a esta pregunta, en particular, ejecuciones con un número de iteraciones superior a 1026)

RESPUESTA: Por defecto se asigna una iteracion a cada hebra de un equipo , por tanto la ejecucion de la iteracion 2 se le asigna a la hebra 2 del equipo 0 , Para comprobar a que hebra y a que equipo se le asigna la iteracion 1026 se ha vuelto a ejecutar el programa pero esta vez con 1027 iteraciones por tanto la iteracion 1026 podemos ver que se le asigna a la hebra 2 del equipo 1 .





3. Ejecutar la versión original, `omp_offload`, con varios valores de entrada hasta que se pueda contestar a las siguientes cuestiones:

(a) ¿Se crean cualquier número de hilos (*threads*) por equipo que se ponga en la entrada al programa? (probar también con algún valor mayor que 3000) En caso negativo, ¿qué número de hilos por equipo son posibles?

RESPUESTA:

Tras probar varias entradas he podido comprobar que tal y como indica la diapositiva 9 el numero minimo de hebras que se pueden crear son 32 ya que se env'ian en paquetes de ejecucion de 32 hebras la unidad llamados wrap por tanto el numero minimo seran 32 aunque se le indiquen menos . Por tanto los numeros de hebras que se pueden asignar son multiplos de 32 hasta un maximo de 1024 ya que a partir de este numero da el error core dumped .

CAPTURAS (que justifiquen la respuesta)

```
$srun -pac4 -Aac omp_offload_gpu 20 3 20 > salida.txt
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos]
2024-05-08 Wednesday
$cat salida.txt
Target device: 1
Tiempo:0.111284018
Iteracción 0, en thread 0/32 del team 0/3
Iteracción 1, en thread 1/32 del team 0/3
Iteracción 2, en thread 2/32 del team 0/3
Iteracción 3, en thread 3/32 del team 0/3
Iteracción 4, en thread 4/32 del team 0/3
Iteracción 5, en thread 5/32 del team 0/3
Iteracción 6, en thread 6/32 del team 0/3
Iteracción 7, en thread 7/32 del team 0/3
Iteracción 8, en thread 8/32 del team 0/3
Iteracción 9, en thread 9/32 del team 0/3
Iteracción 10, en thread 10/32 del team 0/3
Iteracción 11, en thread 11/32 del team 0/3
Iteracción 12, en thread 12/32 del team 0/3
Iteracción 13, en thread 13/32 del team 0/3
Iteracción 14, en thread 14/32 del team 0/3
Iteracción 15, en thread 15/32 del team 0/3
Iteracción 16, en thread 16/32 del team 0/3
Iteracción 17, en thread 17/32 del team 0/3
Iteracción 18, en thread 18/32 del team 0/3
Iteracción 19, en thread 19/32 del team 0/3
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos]
2024-05-08 Wednesday
$
```

```
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac omp_offload_gpu 34 3 3000 > salida.txt
srun: error: atcgrid4: task 0: Aborted (core dumped)
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac omp_offload_gpu 34 3 1500 > salida.txt
srun: error: atcgrid4: task 0: Aborted (core dumped)
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac omp_offload_gpu 34 3 1056 > salida.txt
srun: error: atcgrid4: task 0: Aborted (core dumped)
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac omp_offload_gpu 34 3 1024 > salida.txt
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$
```

(b) ¿Es posible relacionar el número de hilos por equipo posibles con alguno o algunos de los parámetros, comentados en el seminario, que caracterizan al coprocesador que se está usando? Indicar cuáles e indicar la relación.

RESPUESTA: Si , en el seminario en la diapositiva 9 aparece que el numero maximo de threads por SM es 1024 como hemos podido comprobar antes.

4. Eliminar las directivas `teams` y `distribute` en `omp_offload2.c`, llamar al código resultante `opp_offload3.c`. Compilar y ejecutar este código para poder contestar a las siguientes preguntas:

(a) ¿Qué número de equipos y de hilos por equipo se usan por defecto?

```
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPc$sbatch -pac4 -Aac --wrap "nvc -O2 -openmp -mp=gpu omp_offloa
d3.c -o omp_offload3_gpu"
Submitted batch job 249767
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac omp_offload3_gpu 36 > salida.txt
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$
$srun -pac4 -Aac omp_offload3_gpu 1050 > salida.txt
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$cat salida.txt
Target device: 1
Tiempo:0.137840986
Iteracción 0, en thread 0/1024 del team 0/1
Iteracción 1, en thread 0/1024 del team 0/1
Iteracción 2, en thread 1/1024 del team 0/1
Iteracción 3, en thread 1/1024 del team 0/1
Iteracción 4, en thread 2/1024 del team 0/1
Iteracción 5, en thread 2/1024 del team 0/1
Iteracción 6, en thread 3/1024 del team 0/1
Iteracción 7, en thread 3/1024 del team 0/1
Iteracción 8, en thread 4/1024 del team 0/1
Iteracción 9, en thread 4/1024 del team 0/1
Iteracción 10, en thread 5/1024 del team 0/1
Iteracción 11, en thread 5/1024 del team 0/1
Iteracción 12, en thread 6/1024 del team 0/1
Iteracción 13, en thread 6/1024 del team 0/1
Iteracción 14, en thread 7/1024 del team 0/1
Iteracción 15, en thread 7/1024 del team 0/1
```

RESPUESTA:Como se puede observar en la foto por defecto se usan 1024 hilos y un solo equipo , si se ejecutan mas iteraciones que hilos se asigna a los hilos mas iteraciones empezando desde el hilo 0 como podemos observar en la imagen

(b) ¿Qué tanto por ciento del número de SM se están utilizando? Justificar respuesta.

RESPUESTA: En las transparencias hemos visto que el coprocesador GPU se compone de 48 streaming multiprocessors, cada uno con múltiples núcleos CUDA. Como en total hay 3072 núcleos de procesamiento, cada SM tiene 3072/48 = 64 núcleos CUDA (SP).

Al estar usando un solo equipo (SM), y por haber más hilos (1024) que procesadores (SP), estaremos utilizando todos los procesadores. En resumen, estamos utilizando 64 núcleos de 3072, luego 64/3072 x 100 = 2.083% del total de núcleos.

5. En el código `daxpbyz32_ompoff.c` se calcula (a y b son escalares, x, y y z son vectores):

$$z = a \cdot x + b \cdot y$$

Se han introducido funciones `omp_get_wtime()` para obtener el tiempo de ejecución de las diferentes construcciones/directivas `target` utilizadas en el código.

1) t2-t1 es el tiempo de `target enter data`, que reserva espacio en el dispositivo coprocesador para `x`, `y`, `z`, `N` y `p`, y transfiere del host al coprocesador aquellas que se mapean con `to` (`x`, `N` y `p`).
2) t3-t2 es el tiempo del primer `target teams distribute parallel for` del código, que se ejecuta en paralelo en el coprocesador del bucle:
   ```
   for (int i = 0; i < N; i++) z[i] = p * x[i];
   ```
3) t4-t3 es el tiempo de `target update`, que transfiere del host al coprocesador `p` e `y`.
4) t5-t4 es el tiempo del segundo target `teams distribute parallel for` del código, que ejecuta en paralelo en el coprocesador del bucle:
   ```
   for (int i = 0; i < N; i++) z[i] = z[i] + p * y[i];
   ```
5) t6-t7 es el tiempo que supone `target exit data`, que transfiere los resultados de las variables con `from` y libera el espacio ocupado en la memoria del coprocesador.

Compilar `daxpbyz32_off.c` para la GPU y para las CPUs de atctrid4 usando:

```
srun -pac4 -Aac nvc -O2 -openmp -mp=gpu daxpbyz32_ompoff.c -o daxpbyz32_ompoff_GPU
srun -pac4 -Aac nvc -O2 -openmp -mp=multicore daxpbyz32_ompoff.c -o daxpbyz32_ompoff_CPU
```

En `daxpbyz32_off_GPU` el coprocesador será la GPU del nodo y, en `daxpbyz32_off_CPU`, será el propio host. En ambos casos la ejecución aprovecha el paralelismo a nivel de flujo de instrucciones del coprocesador. Ejecutar ambos para varios valores de entrada usando un número de componentes `N` para los vectores entre 1000 y 100000 y contestar a las siguientes preguntas.

CAPTURAS DE PANTALLA (que muestren la compilación y las ejecuciones):

```
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$sbatch -pac4 -Aac --wrap "nvc -O2 -openmp -mp=multicore daxpbyz32_ompoff.c -o daxpbyz32_ompoffcpu"
Submitted batch job 249776
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$sbatch -pac4 -Aac --wrap "nvc -O2 -openmp -mp=gpu daxpbyz32_ompoff.c -o daxpbyz32_ompoffgpu"
Submitted batch job 249777
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$ls
daxpbyz32_ompoff     daxpbyz32_ompoffgpu  omp_offload3.c    salida.txt        slurm-249774.out
daxpbyz32_ompoff.c   omp_offload.c        omp_offload3_gpu  slurm-249716.out  slurm-249775.out
daxpbyz32_ompoff2.c  omp_offload2.c       omp_offload_gpu   slurm-249734.out  slurm-249776.out
daxpbyz32_ompoffcpu  omp_offload2_gpu     pi.c              slurm-249767.out  slurm-249777.out
```

```
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffgpu 1000 2 3
Target device: 1
 *      Tiempo: ((Reserva+inicialización) host 0.000006914) + (target enter data 0.146625996) + (target1 0.000370026) + (host actualiza 0.000000000) + (target data update 0.000032187) + (target2 0.000034809) + (target exit data 0.000061989)= 0.147131920    / Tamaño Vectores:1000 / alpha*x[0]+beta*y[0]=z[0](2.000000*100.000000+3.000000*100.000000=500.000000) / / alpha*x[999]+beta*y[999]=z[999](2.000000*199.899994+3.000000*0.100000=400.099976) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffgpu 20000 2 3
Target device: 1
 *      Tiempo: ((Reserva+inicialización) host 0.000078917) + (target enter data 0.129266977) + (target1 0.000520945) + (host actualiza 0.000042200) + (target data update 0.000042915) + (target2 0.000041962) + (target exit data 0.000097036)= 0.130090952    / Tamaño Vectores:20000        / alpha*x[0]+beta*y[0]=z[0](2.000000*2000.000000+3.000000*2000.000000=10000.000000) / / alpha*x[19999]+beta*y[19999]=z[19999](2.000000*3999.899902+3.000000*0.100000=8000.099609) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffgpu 50000 2 3
Target device: 1
 *      Tiempo: ((Reserva+inicialización) host 0.000261068) + (target enter data 0.160562992) + (target1 0.000593901) + (host actualiza 0.000132084) + (target data update 0.000077009) + (target2 0.000048876) + (target exit data 0.000198126)= 0.161874056    / Tamaño Vectores:50000        / alpha*x[0]+beta*y[0]=z[0](2.000000*5000.000000+3.000000*5000.000000=25000.000000) / / alpha*x[49999]+beta*y[49999]=z[49999](2.000000*9999.900391+3.000000*0.100000=20000.101562) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffgpu 75000 2 3
Target device: 1
 *      Tiempo: ((Reserva+inicialización) host 0.000315905) + (target enter data 0.131973028) + (target1 0.000514030) + (host actualiza 0.000165939) + (target data update 0.000088215) + (target2 0.000043869) + (target exit data 0.000229120)= 0.133330107    / Tamaño Vectores:75000        / alpha*x[0]+beta*y[0]=z[0](2.000000*7500.000000+3.000000*7500.000000=37500.000000) / / alpha*x[74999]+beta*y[74999]=z[74999](2.000000*14999.900391+3.000000*0.100000=30000.101562) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffgpu 100000 2 3
Target device: 1
 *      Tiempo: ((Reserva+inicialización) host 0.000384092) + (target enter data 0.127925873) + (target1 0.000385046) + (host actualiza 0.000216961) + (target data update 0.000105143) + (target2 0.000044823) + (target exit data 0.000288963)= 0.129350901    / Tamaño Vectores:100000       / alpha*x[0]+beta*y[0]=z[0](2.000000*10000.000000+3.000000*10000.000000=50000.000000) / / alpha*x[99999]+beta*y[99999]=z[99999](2.000000*19999.900391+3.000000*0.100000=40000.101562) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$
```

```
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffcpu 1000 2 3
Target device: 0
 *      Tiempo: ((Reserva+inicialización) host 0.000009060) + (target enter data 0.000000954) + (target1 0.001559019) + (host actualiza 0.000000954) + (target data update 0.000000000) + (target2 0.000005960) + (target exit data 0.000000000)= 0.001575947        / Tamaño Vectores:1000 / alpha*x[0]+beta*y[0]=z[0](2.000000*100.000000+3.000000*100.000000=500.000000) / / alpha*x[999]+beta*y[999]=z[999](2.000000*199.899994+3.000000*0.100000=400.099976) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffcpu 20000 2 3
Target device: 0
 *      Tiempo: ((Reserva+inicialización) host 0.000061035) + (target enter data 0.000000000) + (target1 0.001623869) + (host actualiza 0.000072956) + (target data update 0.000000000) + (target2 0.000030994) + (target exit data 0.000000000)= 0.001788855        / Tamaño Vectores:20000        / alpha*x[0]+beta*y[0]=z[0](2.000000*2000.000000+3.000000*2000.000000=10000.000000) / / alpha*x[19999]+beta*y[19999]=z[19999](2.000000*3999.899902+3.000000*0.100000=8000.099609) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffcpu 50000 2 3
Target device: 0
 *      Tiempo: ((Reserva+inicialización) host 0.000144958) + (target enter data 0.000000000) + (target1 0.002074957) + (host actualiza 0.000161171) + (target data update 0.000000000) + (target2 0.000073910) + (target exit data 0.000000000)= 0.002454996        / Tamaño Vectores:50000        / alpha*x[0]+beta*y[0]=z[0](2.000000*5000.000000+3.000000*5000.000000=25000.000000) / / alpha*x[49999]+beta*y[49999]=z[49999](2.000000*9999.900391+3.000000*0.100000=20000.101562) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffcpu 75000 2 3
Target device: 0
 *      Tiempo: ((Reserva+inicialización) host 0.000285864) + (target enter data 0.000000000) + (target1 0.002248049) + (host actualiza 0.000296116) + (target data update 0.000000000) + (target2 0.000106812) + (target exit data 0.000000000)= 0.002936840        / Tamaño Vectores:75000        / alpha*x[0]+beta*y[0]=z[0](2.000000*7500.000000+3.000000*7500.000000=37500.000000) / / alpha*x[74999]+beta*y[74999]=z[74999](2.000000*14999.900391+3.000000*0.100000=30000.101562) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac daxpbyz32_ompoffcpu 100000 2 3
Target device: 0
 *      Tiempo: ((Reserva+inicialización) host 0.000355959) + (target enter data 0.000000000) + (target1 0.002136946) + (host actualiza 0.000427961) + (target data update 0.000000000) + (target2 0.000180006) + (target exit data 0.000000000)= 0.003100872        / Tamaño Vectores:100000       / alpha*x[0]+beta*y[0]=z[0](2.000000*10000.000000+3.000000*10000.000000=50000.000000) / / alpha*x[99999]+beta*y[99999]=z[99999](2.000000*19999.900391+3.000000*0.100000=40000.101562) /
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$
```

(a) ¿Qué construcción o directiva target supone más tiempo en la GPU?, ¿a qué se debe?
RESPUESTA:Las imagenes corresponden respectivamente a los tiempos de gpu (la de arriba)y cpu(la de abajo) La directiva rarget enter data supone mas teimpo en la gpu , esto es por que se debe de copiar en la memoria de los coprocesadores las variables indicadas por lo que debe mapearlas , crear las tablas de paginacion y todo lo que ello conlleva .

(b) ¿Qué construcciones o directivas target suponen más tiempo en la GPU que en la CPU?, ¿a qué se debe?
RESPUESTA:: Las directivas target que suponen más tiempo en la GPU que en la CPU son target enter data, target data update y target exit data. Esto se debe a que en la CPU no se usa la memoria de la GPU, sino la del host, que es la de la CPU, por lo que no tarda nada en crear un nuevo ámbito de variables en la CPU, al igual que al actualizarlas y borrarlas como se explica en la diapositiva 11 .

3. Resto de ejercicios

6. A partir del código secuencial que calcula PI, obtener un código paralelo basado en las construcciones/directivas OpenMP para ejecutar código en coprocesadores. El código debe usar como entrada el número de intervalos de integración y debe imprimir el valor de PI calculado, el error cometido y los tiempos (1) del cálculo de pi y (2) de la trasferencia hacia y desde el coprocesador. Generar dos ejecutables, uno que use como coprocesador la CPU y otro que use la GPU. Comparar **la precisión** del resultado y los **tiempos de ejecución total, cálculo y comunicación** obtenidos en atcgrid4 con la CPU y la GPU, indicar cuál arquitectura son mejores y razonar los motivos.

CAPTURA CÓDIGO FUENTE: `pi-ompoff.c`

```
int main(int argc, char **argv)
{
  register double width;
  double sum;
  register int intervals, i;
  double t1=0.0,t2=0.0,t3=0.0,t4=0.0;//para tiempo

  //Los procesos calculan PI en paralelo
  if (argc<2) {printf("Falta número de intevalos");exit(-1);}
  intervals=atoi(argv[1]);
  if (intervals<1) {intervals=1E6; printf("Intervalos=%d",intervals);}
  width = 1.0 / intervals;
  sum = 0;

  t1 = omp_get_wtime();
#pragma omp target enter data map(to:width,intervals,sum)
t2 =omp_get_wtime();
#pragma omp target teams distribute parallel for reduction(+:sum)
  for (i=0; i<intervals; i++) {
    register double x = (i + 0.5) * width;
    sum += 4.0 / (1.0 + x * x);
  }
t3=omp_get_wtime();
#pragma omp target exit data map(delete: intervals, width) map(from: sum)
  sum *= width;
  t4 = omp_get_wtime();
  printf("Iteraciones:\t%d\t. PI:\t%26.24f\t. Tiempo:\t%8.6f\n", intervals,sum,t3-t1);
  printf("Error:\t%8.6f\n", fabs(M_PI-sum));
  printf("(Target enter data): \t%8.6f\n (Target teams distribute parallel for): \t%8.6f\
n (Target exit data): \t%8.6f\n",t2-t1,t3-t2,t4-t3);

  return(0);
}
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
```

CAPTURAS DE PANTALLA (mostrar la compilación y la ejecución para 10000000 intervalos de integración en atcgrid4 – envío(s) a la cola):

```
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$sbatch -pac4 -Aac --wrap "nvc -O2 -openmp -mp=gpu pi-ompoff.c -o pi-ompoffgpu"
Submitted batch job 249813
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$sbatch -pac4 -Aac --wrap "nvc -O2 -openmp -mp=multicore pi-ompoff.c -o pi-ompoffcpu"
Submitted batch job 249814
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$ls
daxpbyz32_ompoff.c    omp_offload.c      omp_offload3_gpu  pi-ompoffgpu      slurm-249814.out
daxpbyz32_ompoff2.c   omp_offload2.c     omp_offload_gpu   pi.c
daxpbyz32_ompoffcpu   omp_offload2_gpu   pi-ompoff.c       salida.txt
daxpbyz32_ompoffgpu   omp_offload3.c     pi-ompoffcpu      slurm-249813.out
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$
```

```
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac pi-ompoffgpu 10000000
Iteraciones:    10000000        . PI:   3.14159265358979448265593. Tiempo:       1.462231
Error:  0.000000
(Target enter data):    1.460812
 (Target teams distribute parallel for):        0.001419
 (Target exit data):    0.000053
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$srun -pac4 -Aac pi-ompoffcpu 10000000
Iteraciones:    10000000        . PI:   3.14159265358979782901946137. Tiempo:       0.012781
Error:  0.000000
(Target enter data):    0.000000
 (Target teams distribute parallel for):        0.012781
 (Target exit data):    0.000000
[Pablo Linari Perez ac422@atcgrid:~/AC_OpenMPcoprocesadores/codigos] 2024-05-08 Wednesday
$
```

RESPUESTA:

Vemos que el tiempo de la GPU es mayor, esto se debe a por lo que
habíamos visto en el ejercicio anterior, pues en la CPU (host) las variables ya están en memoria pero cuando se usa la GPU no, por lo que se consume tiempo en la creación de un nuevo ámbito de variables. Además, en la GPU se hace uso de un mayor número de núcleos que la CPU, y la cláusula reduction implica comunicación, que también añade retardo (se vio en clases de teoria), y cuesta más comunicar un gran número de núcleos que una cantidad menor.