OpenMP coprocesadores

# Consideraciones previas

* Se usará el compilador nvc de Nvidia, que se puede descargar de su página web. En atcgrid está instalado en el nodo atcgrid4.
* 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

# Ejercicios basados en los ejemplos del seminario

1. **(a)** Compilar el ejemplo omp\_offload.c del seminario en el nodo atcgrid4::

sbatch –pac4 –Aac --wrap “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 35 3 32 > salida.txt

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

|  |
| --- |
|  |

Contestar las siguientes preguntas:

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

**RESPUESTA**:

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

**RESPUESTA**:

**(d)** ¿Qué número máximo de iteraciones se ha asignado a un hilo?

**RESPUESTA**:

**(e)** ¿Qué número mínimo de iteraciones se ha asignado a un equipo y cuál es ese equipo?

**RESPUESTA**:

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. Compilary 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**:

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

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

**RESPUESTA**:

**(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 1025, si la hubiera? (realizar las ejecuciones que se consideren necesarias para contestar a esta pregunta, en particular, alguna ejecución con un número de iteraciones de al menos 1025)

**RESPUESTA**:

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**:

**CAPTURAS (que justifiquen la respuesta)**

**(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**:

4. Eliminar las directivas teams y distribute en omp\_offload2.c, llamar al código resultante opp\_offload3.c. Compilary ejecutar este código para poder contestar a las siguientes preguntas:

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

**RESPUESTA**:

**(b)** ¿Qué tanto por ciento del número de núcleos de procesamiento paralelo de la GPU se están utilizando? Justificar respuesta.

**RESPUESTA**:

5. En el código daxpbyz32\_ompoff.c se calcula (a y b son escalares, x, y y z son vectores):

z = a  x + b  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 de espacio en el dispositivo coprocesador para x, y, z, N y p, y transfiere del host al coprocesador de 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];

1. t4-t3 es el tiempo de target update, que transfiere del host al coprocesador p e y.
2. 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];

1. 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:

sbatch –pac4 –Aac --wrap “nvc -O2 -openmp -mp=gpu daxpbyz32\_ompoff.c -o daxpbyz32\_ompoff\_GPU”

sbatch –pac4 –Aac --wrap “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):**

1. ¿Qué construcción o directiva target supone más tiempo en la GPU?, ¿a qué se debe?

**RESPUESTA**:

1. ¿Qué construcciones o directivas target suponen más tiempo en la GPU que en la CPU?, ¿a qué se debe?

**RESPUESTA**:

# 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 la GPU. Generar dos ejecutables, uno que use como coprocesador la CPU y otro que use la GPU. Comparar los tiempos de ejecución obtenidos en atcgrid4 con la CPU y la GPU, indicar cuáles son mayores y razonar los motivos.

**CAPTURA CÓDIGO FUENTE**: pi-ompoff.c

|  |
| --- |
|  |

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

**RESPUESTA**: