Práctica 4. Paralelización heterogénea con GPUs

Computación de Altas Prestaciones

Carlos García Sánchez

22 de noviembre de 2021

- "Computer Architecture: A Quantitative Approach", J.L. Hennessy, D.A. Patterson, Morgan Kaufmann 2011
- "Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition", James Jeffers, James Reinders, Avinash Sodani

Outline

Objetivos

- Familiarizarse con la programación de GPUs por medio de directivas:
  - OpenACC o OpenMP
- Evaluar las mejoras/speedup

Directivas

- OpenACC: **pgcc**/**nvc** con *-acc=gpu -Minfo*
  - Manual de uso en `https://docs.nvidia.com/hpc-sdk/compilers/index.html`
  - Disponible la descarga en `https://developer.nvidia.com/hpc-sdk`
- OpenMP: **icx** *-fiopenmp -fopenmp-targets=spir64 -qopt-report*

Ejemplo 0

ejemplo0.c

```
#include <stdio.h>

#include <openacc.h>

#define N 1000

int main() {

   int a[N];
   int b[N];

   acc_init(acc_device_not_host);
   printf(" Compiling with OpenACC support \n");
   printf(" Hello World! \n ");

   // Compute on the host
   for (int i = 0; i < N; i++) {
      a[i] = i;
   }
   // Compute on the GPU if OpenACC/OpenMP support - host if not
   #pragma acc kernels copy(b[0:N])
   for (int i = 0; i < N; i++) {
      b[i] = i;
   }
   for (int i = 0; i < N; i++) {
      if (a[i] != b[i]) {
         printf("Something went wrong\n");
      return 1;
      }
   }
   acc_shutdown(acc_device_not_host);

  return 0;

}
```

OpenACC

- Uso del compilador PGI con soporte OpenACC: **pgcc**
  - Conocer las características de la GPU para poder compilar adecuadamente
- Preparados los **makefile**

Terminal #1

```
name@host:~/PracticaGPU/Ejemplo0$ make pgi
pgcc -Minfo -fast -acc -ta=nvidia -tp=nehalem hello.c -o hello.pgi.exe
main:
     26, Loop not fused: function call before adjacent loop
         Generated vector simd code for the loop
     31, Generating copy(b[:])
     36, Loop is parallelizable
         Generating Tesla code
         36, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
     39, Loop not vectorized/parallelized: potential early exits
```

OpenMP

- Uso del compilador ICX con soporte para *offloading*: **icx**
  - Soporte para de **gcc** OpenMP4.0 y OpenMP4.5 a partir de versión 7 y 8
  - *gcc-offload-nvptx - GCC offloading compiler to NVPTX*
- Preparados los **makefile**

Ejemplo 0

ejemplo0.c

```
#include <stdio.h>

#include <omp.h>

#define N 1000

int main() {

    int a[N];
    int b[N];

    printf(" Hello World! \n ");

    // Compute on the host
    for (int i = 0; i < N; i++) {
        a[i] = i;
    }
    // Compute on the GPU if OpenACC/OpenMP support - host if not
    #pragma omp target map(from:b[0:N])
    for (int i = 0; i < N; i++) {
        b[i] = i;
    }
    for (int i = 0; i < N; i++) {
        if (a[i] != b[i]) {
            printf("Something went wrong\n");
        return 1;
        }
    }

   return 0;

}
```

OpenMP

- Uso del compilador ICX con soporte OpenMP: **icx** (Compilador de Intel basado en tecnología LLVM)
  - Compilación y ejecución en la GPU de Intel
- Preparados los **makefile**

Terminal #1

```
name@host:~/PracticaGPU/Ejemplo0$ make omp_offload
icx -O3 -lm -lrt -std=c99 -fiopenmp -fopenmp-targets=spir64 -qopt-report hello.c -
o hello.icx.exe
Global loop optimization report for : main

LOOP BEGIN at hello.c (26, 2)
LOOP END

LOOP BEGIN at hello.c (41, 2)
LOOP END
================================================================

Global loop optimization report for : .omp_offloading.requires_reg
================================================================

Global loop optimization report for : main.DIR.OMP.PARALLEL.LOOP.5.split41.split

LOOP BEGIN at hello.c (35, 4)
    remark: OpenMP: Outlined parallel loop
LOOP END
================================================================

Global loop optimization report for : __omp_offloading_10303_3293fb__Z4main_l34
================================================================

Global loop optimization report for : openmp.descriptor_reg
================================================================

Global loop optimization report for : openmp.descriptor_unreg
================================================================


name@host:~/PracticaGPU/Ejemplo0$ ./hello.icx.exe
There are 1 devices
 Hello World!
 Done
```

Ejemplo1

axpy.c

```
...
int main (int argc, const char *argv[])
{
....
    // SAXPY
    t0 = get_time();
    #pragma .... // Data
    #pragma .... // Loop
    for(i=0; i<n; i++)
        y_acc[i] = a*x_acc[i] + y_acc[i];
....

    return(1);
}
```

Ejemplo2

- Jacobi
  - Método iterativo para resolución de ec. diferenciales
  - Ej: solución para la ecuación de Laplace 2D ($\nabla^2 f(x,y) = 0$)

$$A_{k+1}(i,j) = \frac{A_k(i-1,j)+A_k(i+1,j)+A_k(i,j-1)+A_k(i,j+1)}{4}$$

jacobi.c

```
...
while ( error > tol && iter < iter_max ){
       error = 0.0;

       for( int j = 1; j < n-1; j++){
          for( int i = 1; i < m-1; i++ ){
             Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1]
                                 + A[j-1][i] + A[j+1][i]);
             error = fmax( error, fabs(Anew[j][i] - A[j][i]));
          }
       }

       for( int j = 1; j < n-1; j++){
          for( int i = 1; i < m-1; i++ )
          {
             A[j][i] = Anew[j][i];
          }
       }

       if(iter % 100 == 0) printf("%5d, %0.6f\n", iter, error);

       iter++;
}
```

Ejemplo 3

- Mandelbrot: fractal conocido

$$\begin{cases} z_0 = 0 & \textit{término inicial} \\ z_{n-1} = {z_n}^2 + c & \textit{relación de indución} \end{cases}$$

- Si esta sucesión queda acotada, entonces se dice que **c** pertenece al conjunto de Mandelbrot, (sino queda excluido)
  - P.Ej: si $\mathbf{c = 1}$ genera la sucesión 0, 1, 2, 5, 26... que diverge.
    - c=1 no es un elemento del conjunto de Mandelbrot
  - P.Ej: si $\mathbf{c = -1}$ obtenemos la sucesión 0, -1, 0, -1,... (acotada)
    - c=-1 conjunto de Mandelbrot.

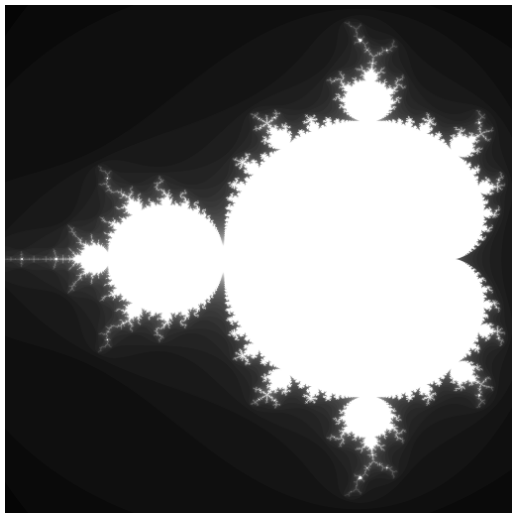
Ejemplo 3

main.cpp

```
...
 for(int y=0;y<HEIGHT;y++) {
    for(int x=0;x<WIDTH;x++) {
      image[y*WIDTH+x]=mandelbrot(x,y);
    }
...
```

Ejemplo 3

- Ejecución *./madelbrot.host.exe*



Ejemplo 4: ecuación del calor

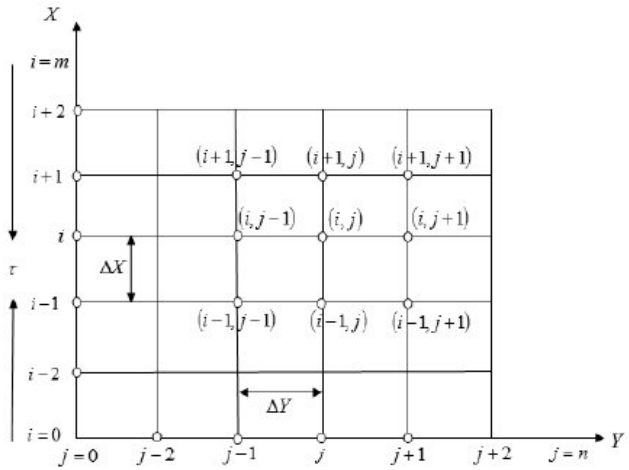
- Resolución Numérica de Problemas de Transmisión de Calor

Diferencias finitas

1. División del espacio considerado en una serie de elementos cuyas propiedades vienen representadas por un punto central (nodo)
2. Aplicación de balances de energía a cada elemento, obteniendo la ecuación característica para cada nodo.
3. Resolución simultánea de todos los balances, para obtener el perfil de temperaturas.
4. Si el caso lo requiere cálculo del flujo de calor con la ley de Fourier y el perfil de temperaturas.

Ejemplo 4: ecuación del calor

- Considerando la ecuación del calor en 2 dimensiones:
  $$Q = -kA\frac{\partial T}{\partial x} \cong -k\frac{\Delta x T}{\Delta x} = -(W\Delta y)\frac{T_{i+1,j} - T_{i,j}}{\Delta x}$$
  - donde Q se ha agregado como un término de generación de calor (positivo para la generación)



Ejemplo 4: ecuación del calor

- Para una celda (i, j) la aportación de calor $Q = Q_1 + Q_2 + Q_3 + Q_4$ (celdas vecinas) donde:
  - $Q_1 = \frac{T_{i,j+1} - T_{i,j}}{\Delta y / k(W \Delta x)}$
  - $Q_2 = \frac{T_{i+1,j} - T_{i,j}}{\Delta x / k(W \Delta y)}$
  - $Q_3 = \frac{T_{i-1,j} - T_{i,j}}{\Delta x / k(W \Delta y)}$
  - $Q_4 = \frac{T_{i,j-1} - T_{i,j}}{\Delta y / k(W \Delta x)}$
- En el estado estacionario $Q_1 + Q_2 + Q_3 + Q_4 = 0$
  - ... luego haciendo las sumas anteriormente descritas $T_{i,j} = \frac{T_{i,j+1} + T_{i,j-1} + T_{i+1,j} + T_{i-1,j}}{4}$

Ejemplo 4: ecuación del calor

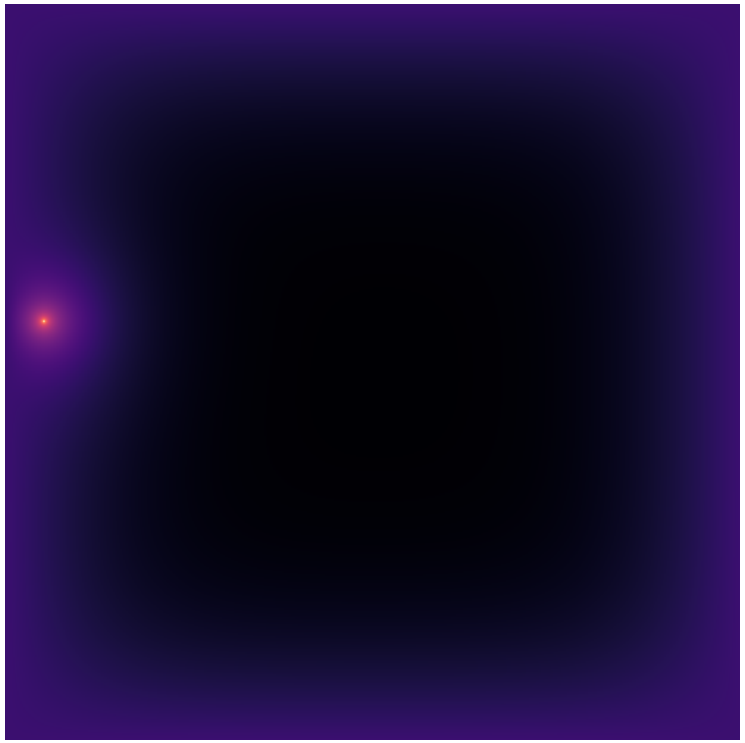
- Inicializa con un valor rand las coordenadas de la fuente de calor: `source_x, source_y`
- Inicializa las condiciones de contorno
- Ejecución del bucle principal `(it < MAX_ITERATIONS) && (t_diff > MIN_DELTA)`
- Salida en fichero .png que muestra el calor en una placa 2D

heat.c

```
for (unsigned int it = 0; (it < MAX_ITERATIONS) && (t_diff >
     MIN_DELTA); ++it) {
    step(source_x, source_y, current, next);
    t_diff = diff(current, next);
    if(it%(MAX_ITERATIONS/10)==0){
       printf("%u: %f\n", it, t_diff);
    }

    float * swap = current;
    current = next;
    next = swap;
}
```

Ejemplo 4: ecuación del calor

- Tras 20000 iteraciones



Ejemplo 4: ecuación del calor

- Las rutinas a prestar especial atención son **step** y **diff**
- **step**: esquema paralelización basado en **descomposición de dominios**
- **diff**: reducción del *maxdiff*

heat_core.c

```
static void step(unsigned int source_x, unsigned int source_y, const
     float * current, float * next) {

   for (unsigned int y = 1; y < N-1; ++y) {
      for (unsigned int x = 1; x < N-1; ++x) {
         if ((y == source_y) && (x == source_x)) {
            continue;
         }
         next[idx(x, y, N)] = (current[idx(x, y-1, N)] +
         current[idx(x-1, y, N)] +
         current[idx(x+1, y, N)] +
         current[idx(x, y+1, N)]) / 4.0f;
      }
   }
}

static float diff(const float * current, const float * next) {
   float maxdiff = 0.0f;
   for (unsigned int y = 1; y < N-1; ++y) {
      for (unsigned int x = 1; x < N-1; ++x) {
         maxdiff = fmaxf(maxdiff, fabsf(next[idx(x, y, N)] - current[
              idx(x, y, N)]));
      }
   }
   return maxdiff;
}
```

Ejemplo 4: ecuación del calor

- Las rutinas a prestar especial atención son **step** y **diff**
- Recomendación: ¡¡¡Minimiza la transferencia entre memorias!!!

Recuerda

- La memoria *no* está compartida entre el host y el device
- NOTA: el moviento de datos es un destructor de rendimiento
  - V100 tiene 900 GB/s de ancho de banda con memoria
  - El PCIe Host-Device logra 32 GB/s de pico
  - **¡¡¡Minimiza la transferencia entre memorias!!!**

Entorno desarrollo

- *pgcc/nvc* v20.7 para compilar los ficheros fuente en OpenACC y ejecución en la gráfica NVIDIA
- *icx* v2021.3.0 para compilar fichero fuente en OpenMP y ejecución en la gráfica de Intel

Entorno desarrollo

- NVIDIA Instalado es el driver 470.57.02

nvidia-smi

Terminal #1

```
garsanca@pixel:~$ nvidia-smi
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.66       Driver Version: 450.66       CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GeForce GTX 106...  Off  | 00000000:01:00.0 Off |                  N/A |
|  0%   31C    P8     5W / 120W |     81MiB /  6075MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      8070      G   /usr/lib/xorg/Xorg                 78MiB |
+-----------------------------------------------------------------------------+
```

Herramientas *nvidia*

- Herramientas de desarrollo en CUDA/OpenACC
- NVIDIA Nsight Visual Studio Edition | NVIDIA Developer
  - Eclipse: *nsight*
  - Depurador: *cuda-gdb*
  - Profiling: *nvprof* y *nvvp*

Herramientas *nvidia*

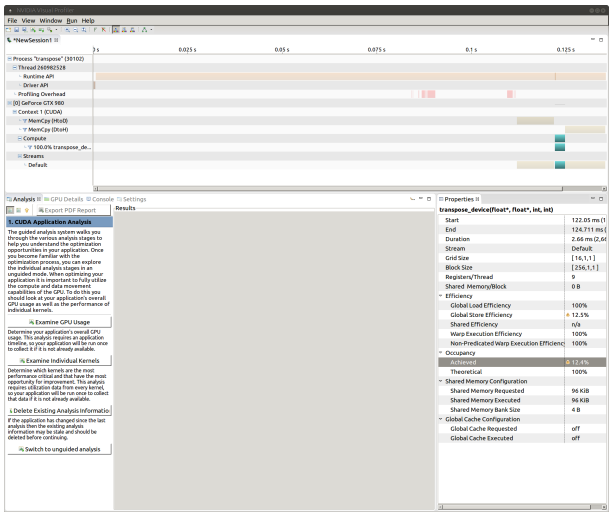
- CUDA profiler: **nvprof**

Terminal #1

```
name@host: nsys nvprof  ./hello.pgi.exe
WARNING: hello.pgi.exe and any of its children processes will be profiled.

Collecting data...
 Compiling with OpenACC support
 Hello World!
 Done
Processing events...
Saving temporary "/tmp/nsys-report-6c60-2866-ebde-e502.qdstrm" file to disk...
Creating final output files...

Processing [==========================================================100%]
Saved report file to "/tmp/nsys-report-6c60-2866-ebde-e502.qdrep"
Exporting 473 events: [==================================================100%]

Exported successfully to
/tmp/nsys-report-6c60-2866-ebde-e502.sqlite

Generating CUDA API Statistics...
CUDA API Statistics (nanoseconds)

Time(%)      Total Time       Calls         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  -------------
   65.7        14310695           1      14310695.0        14310695        14310695  cuMemHostAllo
   31.3         6805780           1       6805780.0         6805780         6805780  cuMemFreeHost
    2.2          477892           1        477892.0          477892          477892  cuMemAllocHos
    0.3           65431           2         32715.5            3402           62029  cuMemAlloc_v2
    0.3           56138           1         56138.0           56138           56138  cuModuleLoadD
    0.1           19250           1         19250.0           19250           19250  cuLaunchKerne
    0.1           14895           1         14895.0           14895           14895  cuMemcpyHtoDA
    0.0            8613           1          8613.0            8613            8613  cuStreamCreat
    0.0            6146           3          2048.7             785            4385  cuStreamSynch
    0.0            5524           1          5524.0            5524            5524  cuMemcpyDtoHA
    0.0            2116           1          2116.0            2116            2116  cuEventRecord
    0.0            1642           2           821.0             273            1369  cuEventCreate
    0.0             776           1           776.0             776             776  cuEventSynchr

Generating CUDA Kernel Statistics...
CUDA Kernel Statistics (nanoseconds)

Time(%)      Total Time   Instances         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  -------------
  100.0            1280           1          1280.0            1280            1280  main_41_gpu

Generating CUDA Memory Operation Statistics...
CUDA Memory Operation Statistics (nanoseconds)

Time(%)      Total Time  Operations         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  -------------
   55.7            1248           1          1248.0            1248            1248  [CUDA memcpy
   44.3             992           1           992.0             992             992  [CUDA memcpy

CUDA Memory Operation Statistics (KiB)

              Total      Operations              Average           Minimum              Maximum
------------------  --------------  ------------------  ----------------  ------------------
           3.906              1               3.906             3.906               3.906
           3.906              1               3.906             3.906               3.906

Generating NVTX Push-Pop Range Statistics...
NVTX Push-Pop Range Statistics (nanoseconds)
```

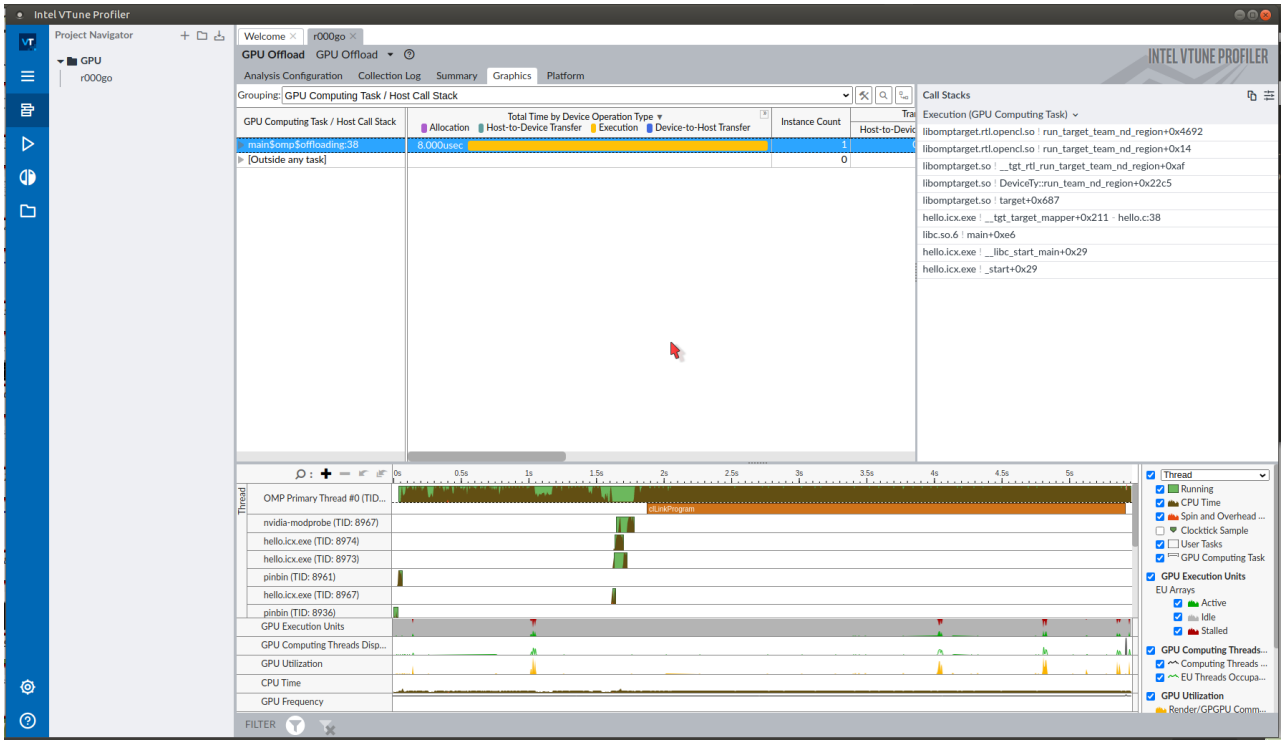
Herramientas *nvidia*

- CUDA profiler: **nvvp**
  - Aplicación gráfica con mayor precisión
- Muestra el uso de la GPU y los tiempos que conllevan los *memcpy*
  - Además muestra el *occupancy* de cada función y el uso de la memoria



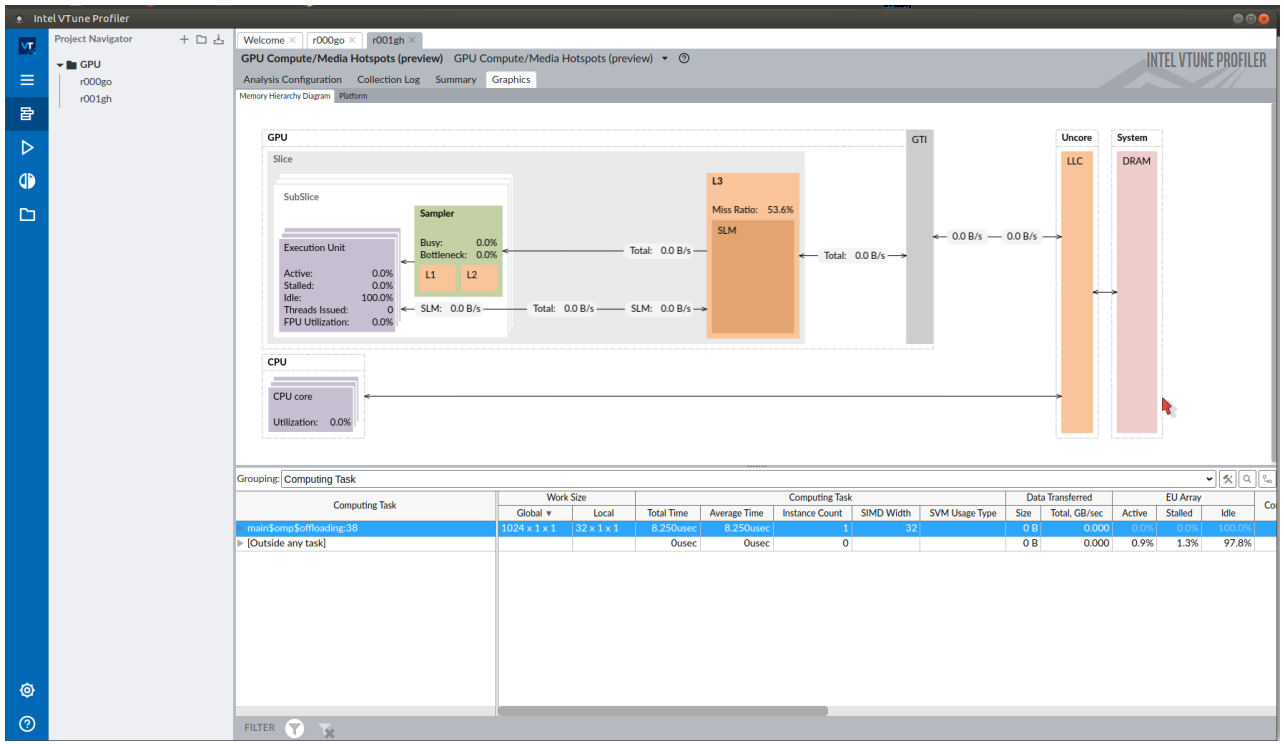
Herramientas Intel

- VTune con análisis GPU Offload



Herramientas Intel

- VTune con análisis GPU Compute/Media Hotspots
  - Utilización de las EU y uso de jerarquía de memoria



Material

- Training: `https://ngc.nvidia.com/catalog/containers/hpc:openacc-training-materials`
- Docker container

Terminal #1

```
user@computer:~$ docker run --gpus all -it --rm -p 8000:8000 nvcr.io/hpc/openacc-training-materials:20.7.1
```

Intel DevCloud

1. Conéctate al Intel DevCloud
2. ... pero vamos a trabajar con **OpenMP* Offload Basics**
   - `https://devcloud.intel.com/oneapi/get_started/hpcTrainingModules/`
3. Selecciona el cuaderno de jupyter **Module 1 Introduction to OpenMP Offload.**
4. Selecciona el cuaderno de jupyter **Module 2 Manage Device Data**
5. Selecciona el cuaderno de jupyter **Module 3 OpenMP Device Parallelism**