II. SOFTWARE CUDA

1. HERRAMIENTAS DE PROGRAMACIÓN CUDA:

CUDA es esencialmente una herramienta de programación. Es decir permite a los desarrolladores escribir, compilar, depurar y ejecutar código adaptado para las arquitecturas de las GPUs compatibles.

Es por ello por lo que se pone a disposición de los desarrolladores un paquete de herramientas adaptado para realizar su trabajo.

Este paquete de herramientas se descarga libremente del sitio Web de NVIDIA y se instala en el equipo que debe cumplir unos requisitos imprescindibles para trabajar con CUDA y son:

_ Sistema Operativo Compatible:

NVIDIA ofrece un listado de SO compatibles. Es de señalar que CUDA se ejecuta sobre varias distribuciones Linux y en entornos Windows y Mac. También se debe observar que es compatible tanto con plataformas de 32 como de 64 bits.

_ GPU que soporta la tecnología CUDA:

NVIDIA ofrece un listado actualizado de tarjetas que soportan la tecnología CUDA. En concreto las familias GeForce, Quadro y Tesla. La familia GeForce está dedicada a vídeo; la familia Quadro a visionado especialmente su gama alta en entornos muy exigentes y la famila Tesla está diseñada para computación. Eatas familias, aunque diseñadas para resolver unos problemas concretos, soportan la tecnología CUDA y por tanto pueden ser usadas para acelerar los procesos computacionales que son objeto de este proyecto.

CUDA ofrece también la posibilidad de realizar estas tareas sin disponer de tarjeta gráfica compatible mediante la herramienta de emulación.

_ Compilador C:

Debe estar previamente instalado el gcc y toolchain.

_ Software CUDA:

El software CUDA está disponible en la Web de NVIDIA. Se usó para la etapa de desarrollo la versión 2.3; que en su momento era la última estable, aunque en la etapa de pruebas se instaló la versión 3.1.

2. INSTALACIÓN DE LAS HERRAMIENTAS DE DESARROLLO CUDA

La Instalación de las herramientas en un sistema Linux se realiza siguiendo la guía ofrecida por NVIDIA en 5 pasos:

_ Verificación de que el sistema tiene una GPU compatible con la tecnología CUDA: En especial se verifica la compatibilidad del compilador C “gcc”. CUDA soporta la versión 3.4 y 4.x.

La versión instalada se comprueba con: **gcc -version**

_ Descarga de los drivers (controladores) NVIDIA y del Software CUDA: La tecnología CUDA exige el uso de drivers de la versión 190 o posterior. Los drivers están disponibles para descargar de la Web de NVIDIA.

Se puede comprobar con la orden: **/usr/bin/nvidia-settings**

Además del driver disponemos de:

- The CUDA Toolkit, caja de herramientas CUDA:

  Contiene las herramientas necesarias para compilar una aplicación CUDA compatible con la versión de driver. Incluye bibliotecas, archivos de cabecera y otros recursos.

  El Toolkit de CUDA es un entorno de desarrollo en lenguaje C diseñado para las GPUs compatibles con la tecnología CUDA. El entorno de desarrollo CUDA incluye:

  * El compilador de C nvcc

  * Bibliotecas FFT y BLAS de CUDA para la GPU

  * Analizador de rendimiento (Profiler)

  * Depurador gdb para la GPU.

  * Controlador CUDA de tiempo de ejecución.

  * Manual de programación CUDA

- Kit de desarrollo de Software o SDK (Software Developement Kit) de CUDA:

Incluye ejemplos de aplicaciones con código fuente. El contenido detallado es el siguiente:

* Ordenación bitónica en paralelo

* Multiplicación de matrices

* Transposición de matrices

* Análisis del rendimiento mediante temporizadores

* Suma de prefijos de arrays grandes (modelo Scan) en paralelo

* Convolución de imágenes

* DWT 1D mediante la transformada wavelet de Haar

* Ejemplos de interacción de gráficos OpenGL y Direct3D

* Ejemplos de uso de las bibliotecas BLAS y FFT en CUDA

* Integración de código C y C++ para CPU-GPU

* Modelo binomial de valoración de opciones (BOPM)

* Valoración de opciones con el modelo de Black-Scholes

* Valoración de opciones mediante el método Montecarlo

* Mersenne Twister paralelo (generación de números aleatorios)

* Cálculo de histogramas en paralelo

* Supresión de ruido en las imágenes

* Filtro Sobel para detección de bordes

* Plug-in MATLAB® de MathWorks

_ Instalación del driver NVIDIA

Se sale del entorno GUI mediante la orden (ctl-alt-backspace). En la línea de comandos se cierra el X Windows con el comando: **/sbin/init 3.**

Luego se ejecuta el programa del driver como superusuario o usuario administrador.

Se rearranca el entorno GUI con la orden “startx” o bien “init 5”.

En las propiedades del sistema se verifica que el driver ha sido instalado correctamente.

_ Instalación del Software CUDA

En caso de que exista una instalación previa se debe desinstalar el paquete.

Se borran los archivos contenidos en **/usr/local/cuda** y en **$(HOME)/NVIDIA_CUDA_SDK/** donde habrá sido instalado por defecto.

Se instala el kit de herramientas como administrador ejecutando el archivo "*.run". El kit se instalará en el repertorio por defecto **/usr/local/cuda**. Las variables del PATH deben ser especificados y deben incluir la ruta **/usr/local/cuda/bin**. Además **LD_LIBRARY_PATH** debe contener **/usr/local/cuda/lib64** para un sistema Linux 64 bits.

Para modificar estos valores se usan los siguientes comandos:

export PATH=/usr/local/cuda/bin:$PATH

export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH

Para hacer que esos cambios sean permanentes se deben colocar esas órdenes en el documento **~/.bash_profile**.

La SDK se instala también ejecutando "*.run". Sin embargo se recomienda instalarlo como usuario habitual y no como administrador. En el script por defecto se localizará la carpeta de instalación en **(HOME)/NVIDIA_GPU_Computing_SDK**.

En un sistema con varios usuarios se recomienda instalarlo, sin embargo, como "root" en un repertorio accesible para todos los usuarios en modo sólo lectura. Esto permitirá restaurar la copia siempre que sea necesario.

_ Verificación de la instalación mediante la ejecución de alguno de los programas proporcionados en la SDK.

Para verificar la instalación del kit CUDA se ejecuta la orden **nvcc -V** en la línea de comandos. "nvcc" es el comando para ejecutar el controlador que compila los programas CUDA. Para su correcto funcionamiento realiza una llamada al compilador C "gcc" que es el compilador NVIDIA PTX para el código CUDA.

En el repertorio **NVIDIA_GPU_Computing_SDK/C** se encuentran los ejemplos que se compilarán mediante la orden "make". Los binarios serán instalados en la carpeta **NVIDIA_GPU_Computing_SDK/C/bin/linux/release**.

Lo binarios son ejecutados realizando llamadas a las bibliotecas contenidas en **LD_LIBRARY_PATH**.

Tras la compilación y en el repertorio **NVIDIA_GPU_Computing_SDK/C/bin/linux/release** situado en la carpeta del usuario, se debe ejecutar la orden **deviceQuery**. La salida informa de que el dispositivo ha sido encontrado que coincide con el hardware instalado y que ha superado el test.

Mediante la orden **bandwidthTest** podemos comprobar que el sistema y el dispositivo se pueden comunicar correctamente.

Compilación en modo emulación de Hardware:

Como apunte adicional se debe señalar que en el caso de carecer de una GPU que soporte la tecnología CUDA, se puede compilar, depurar y ejecutar código CUDA usando la opción de emulación. Sin embargo la velocidad de ejecución diferirá con el sistema real ya que la ejecución en realidad se realiza enteramente en la CPU.

Para ello se usará la orden **make emu=1**; los binarios generados serán situados en el repertorio **bin/linux/emurelease** en el directorio de instalación de la SDK;

3. BIBLIOTECA CUBLAS y CUFFT

Dos herramientas importantes para el cálculo numérico proporcionadas por CUDA en forma de bibliotecas son: CUBLAS y CUFFT.

La biblioteca CUBLAS de CUDA es una implementación optimizada de los Subprogramas Básicos de Álgebra Lineal o BLAS (Basic Linear Algebra Subprograms) sobre CUDA. Permite acceder a los recursos computacionales de la GPU NVIDIA. La biblioteca está auto contenida en el nivel API, por tanto no es necesario interactuar directamente con el driver CUDA. CUBLAS se vincula a una única GPU y por tanto no se auto-paraleliza sobre varias GPUs.

CUBLAS es la versión CUDA BLAS que es un estándar para una interfaz de programación de aplicaciones para bibliotecas que realizan operaciones básicas de álgebra lineal como multiplicación de vectores y matrices. Fueron publicadas en 1979 y son usadas para construir paquetes más complejos como LAPACK. Son muy usadas en computación de alto rendimiento. Versiones optimizadas de BLAS han sido desarrolladas por los fabricantes de Hardware como INTEL y AMD.

CUBLAS contiene versiones de las rutinas BLAS para operandos reales y complejos con precisiones float y double

Las Funcionalidades de CUBLAS coinciden con las definidas para BLAS y se dividen en 3 niveles:1

Nivel 1: En este nivel se encuentran las funciones que realizan operaciones con vectores de la forma: $y \leftarrow \alpha x+y$, al igual que productos escalares y el cálculo de normas vectoriales.

Nivel 2: Este nivel contiene las operaciones Matriz-Vector del tipo: $y \leftarrow \alpha Ax+\beta y$, además de rutinas para resolver sistemas del tipo: T*x=y donde T es triangular.

Nivel 3: Este nivel contiene la operaciones Matriz-Matriz con forma: $C \leftarrow \alpha AB+\beta C$ y la resolución de sistemas bajo la forma $B \leftarrow \alpha T^{-1}B$ donde T es triangular.

[1] Convenio de definición:
Escalar: letra griega.
Vector: letra latina minúscula.
Matriz: letra latina mayúscula
<-: El resultado de la operación de la derecha se almacena en el operando de izquierda.

Como se ha apuntado, BLAS sirve de soporte para el desarrollo de bibliotecas más complejas como LAPACK (Linear Algebra Package) que es una biblioteca de software libre para la resolución de problemas de álgebra lineal de manera numérica.

Contiene rutinas para la resolución de sistemas de ecuaciones lineales, valores propios, descomposición y factorización, mínimos cuadrados lineales. También incluye factorización de matrices como LU, QR y Cholesky y descomposición de Schur. LAPACK fue originalmente escrito en Fortran 77, actualmente está codificado en Fortran 90. Las rutinas manejan tanto matrices reales como complejas de precisión simple y doble.

LAPACK está basado en BLAS y así aprovecha eficientemente las cachés de las arquitecturas de computadores modernas. De momento todavía no existe una Biblioteca "CULAPACK", aunque nada impide su desarrollo usando como bloques constructivos los elementos proporcionados por CUBLAS.

El modelo básico por el cual las aplicaciones usan CUBLAS es mediante la creación de matrices y vectores objeto en el espacio de memoria de la GPU, su relleno con datos, llamada a una secuencia de funciones CUBLAS, y, finalmente, la transferencia de los resultados de la memoria de la GPU a la del Host.

Para ello, CUBLAS proporciona funciones auxiliares para la creación y destrucción de objetos en el espacio de la GPU y para la escritura y recuperación de datos contenidos en estos objetos.

Existen dos convenios para el almacenamiento de la matrices: Según fila o fila mayor y según columna o columna mayor.

Para ilustrar las diferencias, supongamos que tenemos la matriz A siguiente:

$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}$$

Según el convenio fila mayor; sería dispuesta en la memoria como:

$$[a_{11}\ a_{12}\ a_{13}\ a_{21}\ a_{22}\ a_{23}\ a_{31}\ a_{32}\ a_{33}]$$

Mientras que según el convenio de columna mayor sería:

$$[a_{11}\ a_{21}\ a_{31}\ a_{12}\ a_{22}\ a_{32}\ a_{13}\ a_{23}\ a_{33}]$$

Para una completa compatibilidad con entornos FORTRAN existentes, CUBLAS usa almacenamiento según el convenio de "columna mayor" e indexación basada en 1. Como C y C++ usan almacenamiento según el convenio de "fila mayor", las aplicaciones no pueden usar las semánticas de array nativas para los array bidimensionales. En lugar de ello las macros o las funciones dentro de línea deben ser definidos para implementar matrices sobre arrays unidimensionales.

Para código Fortran portado a C de una manera mecánica, se puede elegir mantener la indexación basada en 1 para evitar transformar bucles. En ese caso, el índice array de un elemento matricial situado en la fila i y la columna j puede ser computado mediante la macro:

```
#define IDX2F(i,j,ld) ((((j)-1)*(ld))+((i)-1))
```

Donde, ld se refiere a la dimensión principal de la matriz tal y como se designó, que en el caso de almacenamiento según la columna mayor es el número de filas.

Para código C y C++ nativo, se debe elegir indexación basada en 0 (comienza en cero), en cuyo caso la macro de indexación toma la forma:

```
#define IDX2C(i,j,ld) (((j)*(ld))+(i))
```

Debido a que las funciones CUBLAS de núcleo (a diferencia de las funciones auxiliares) no devuelven directamente el estado de error (por razones de compatibilidad con bibliotecas BLAS existentes), CUBLAS proporciona funciones independientes para ayudar en la depuración que recuperan el último error guardado.

CUFFT son las iniciales de la biblioteca de la transformada rápida de Fourier para CUDA (CUDA Fast Fourier Transform). La FFT es un algoritmo tipo "divide y vencerás"("divide and conquer") para la computación de transformadas discretas de Fourier con valores reales o complejos, y es uno de los algoritmos numéricos más extendidos con aplicaciones en física y especialmente en tratamiento de señales.

La biblioteca CUFFT proporciona una interfaz sencilla para la computación en paralelo de la FFT en una GPU NVIDIA, lo cual permite aprovechar la capacidad computacional de una GPU sin necesidad de desarrollar una implementación FFT particular.

Las bibliotecas FFT suelen diferir en cuanto al tamaño de las transformadas soportadas y el tipo de datos. Por ejemplo algunas bibliotecas sólo implementan FFTs Radix-2, restringiendo el tamaño de la transformada a un valor de potencias de 2, mientras que otras implementan soporte para tamaños arbitrarios.

La biblioteca CUFFT tiene las siguientes características:

- Transformadas 1D, 2D, y 3D de valores reales y complejos.
- "Batch execution" o Ejecución de lotes, para realizar múltiples transformadas 1D en paralelo.
- Tamaño de transformadas de cualquier dimensión 2D y 3D en el rango de [2, 16384]
- Transformada 1D con un tamaño de hasta 8 millones de elementos
- Transformadas "In place" y "out of place" para valores reales y complejos.
- Transformada con Doble precisión en Hardware compatible (GT200 y posteriores).