PROYECTO FIN DE CARRERA

Arquitectura de Computadores y Sistemas Operativos

Instalación de Entornos de Desarrollo

Desarrollo CUDA en Java y Python

José Alejandro Pérez Sánchez

Consultor: Francesc Guim Bernat

Índice

Introducción

Este documento describe el proceso de instalación y puesta a punto de los entornos de desarrollo Java y Python para desarrollo en arquitectura CUDA.

Para ello se precisa instalar inicialmente, como base, el kit de desarrollo CUDA de NVIDIA, base para todas las demás herramientas.

Sin embargo para trabajar con PyCuda estaremos limitados a una versión más antigua del mismo, con la que es compatible.

También precisaremos, según el sistema operativo, un compilador y enlazador (*linker*) de C, Visual Studio 2008/2010 en Windows o gcc en Linux o Mac.

Finalmente, reseñar que, para poder ejecutar un programa CUDA se precisa una GPU CUDA, no existiendo, desde la versión de NVIDIA toolkit 2.9, un emulador CUDA oficial, aunque sí hay algunos no oficiales.

Reseñar que esta guía está orientada a la instalación en entornos Windows, que ha sido el utilizado en el desarrollo de los programas implementados durante la elaboración de este Proyecto. Aún así cabe reseñar que todas las herramientas de desarrollo CUDA estudiadas están disponibles también, como mínimo, para sistemas operativos Linux y Mac.

La única diferencia en el caso de estos sistemas operativos es la utilización de GNU GCC como compilador y linker C en vez de Visual Studio, que es la herramienta de referencia en Windows, si bien también se podría utilizar GCC en Windows mediante Cygwin o MinGW.

NVIDIA CUDA Toolkit & Driver

Para el desarrollo en arquitectura CUDA, NVIDIA proporciona este conjunto de herramientas, CUDA toolkit, compuesto por un compilador de lenguaje CUDA C a ejecutable CUDA, en formatos cubin o ptx, (formatos descritos en detalle en la memoria del PFC), una serie de librerías auxiliares, entre ellas las *Accelerated Libraries*, que integran aritmética matricial, números aleatorios así como otras muchas utilidades a ejecutar en la GPU, y también, según el IDE de desarrollo a utilizar, la herramienta CUDA NSight, que se integra con Eclipse (en Linux y Mac) y con Visual Studio (en Windows), y que permite el desarrollo, asistencia a la codificación y depuración de programas CUDA C.

También proporciona el CUDA Driver, normalmente integrado en los driver gráficos NVIDIA para PCs, y drivers especiales para tarjetas *no gráficas* como el caso de algunas TESLA. Este driver es el que comunica el ordenador con la GPU, y transmite código y datos entre ambos.

Con esto, los pasos para instalar estas herramientas en serían:

1. Instalación de NVIDIA Driver adaptado a la GPU instalada. Este paso debería ser automático en sistemas Windows/Linux recientes, si bien las últimas versiones también pueden encontrarse en:
   http://www.nvidia.com/Download/index.aspx?lang=es

2. Instalación de entorno de Desarrollo. En Windows Visual Studio (2008/2010). Recientemente se ha incorporado soporte a Visual Studio 2012.

3. Instalación de NVIDIA Cuda Toolkit. Hay que prestar atención a la versión correcta, pues en Windows hay versiones distintas, no totalmente compatibles, para ordenadores de sobremesa o servdores y otra versión para portátiles. En Linux también hay versiones optimizada para distintas distribuciones:
   https://developer.nvidia.com/cuda-downloads
   Tras ello se deberá incluir, para trabajar posteriormente con JCuda, PyCuda, etc.., en el *path* de ejecución la carpeta con el ejecutable *nvcc* (CUDA Compiler), y el *linker*, **cl.exe** en Windows, **gcc** en Linux.

4. Si bien no estrictamente preciso para el alcance del proyecto. Es recomendable para estudiar código de ejemplo e incluso para desarrollar código de *kernel* en CUDA C. Es el complemento al IDE **NSight**. Complemento para Visual Sutdio y Eclipse para desarrollar en CUDA C.
   http://www.nvidia.com/object/nsight.html

Inicialmente se ha trabajado con la versión 5.0 de CUDA Toolkit. Pero para trabajar con PyCuda tuvo que desinstalarse e instalar la versión 3.5 con la que es compatible.

JCuda

Se tratan de librerías Java con enlace nativo JNI. Por tanto disponer de JRE y JDK es prerrequisito básico. En este PFC se ha trabajado con las versiones 1.6 y 1.7. JCuda se puede obtener como una serie de librerías las cuáles se han de referenciar durante la programación para disponer de capacidad CUDA. Éstas son, *JCuda, JCublas, JCurand, JCufft, JCusparse* y *JCuda_utils.*

Al tener enlace nativo estas librerías están normalmente optimizadas y son compatibles con una versión específica de CUDA toolkit, aunque pueda darse cierta retrocompatibilidad. En este PFC se ha utilizado la versión 0.5.0a de JCuda, compatible con CUDA 5.0.

Las librerías se descargan como binarios Java, código fuente, documentación y biblioteca nativa, *dll* en Windows y *so* en Linux/Mac.

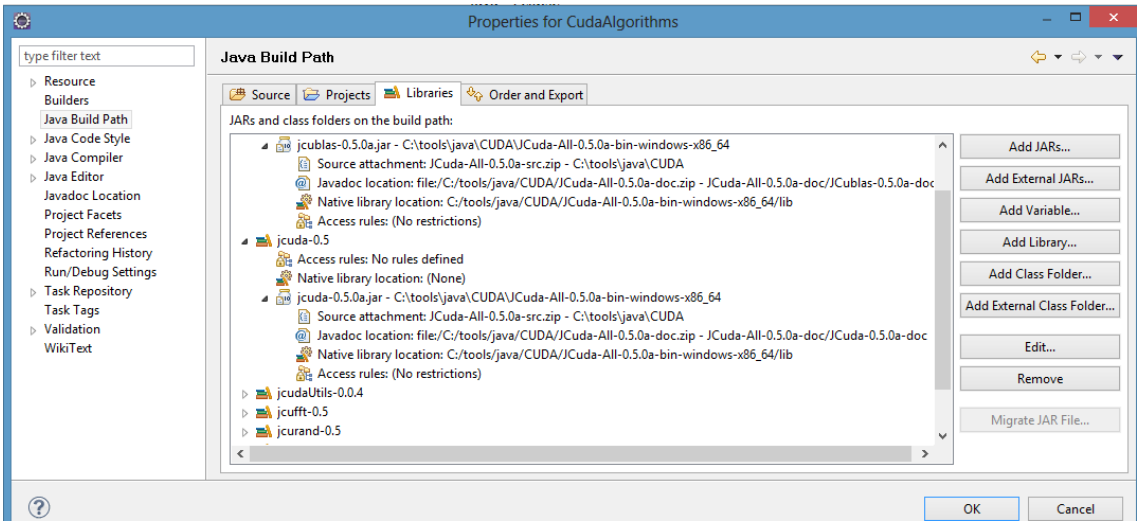
Los pasos de instalación serían:

1. Instalación de JDK/JRE.

2. Descarga de JCuda compatible con CUDA toolkit instalado.
   http://www.jcuda.org/downloads/downloads.html

3. Añadir al *path* del sistema operativo las librerías nativas de JCuda. (dll o so).

4. Para utilizar en un IDE, Eclipse, por ejemplo, se pueden configurar como librerías de usuario añadiéndolas al *build path* del proyecto.



A partir de este momento se pueden utilizar las librerías JCuda como librerías Java normales, usando su API para desarrollo CUDA.

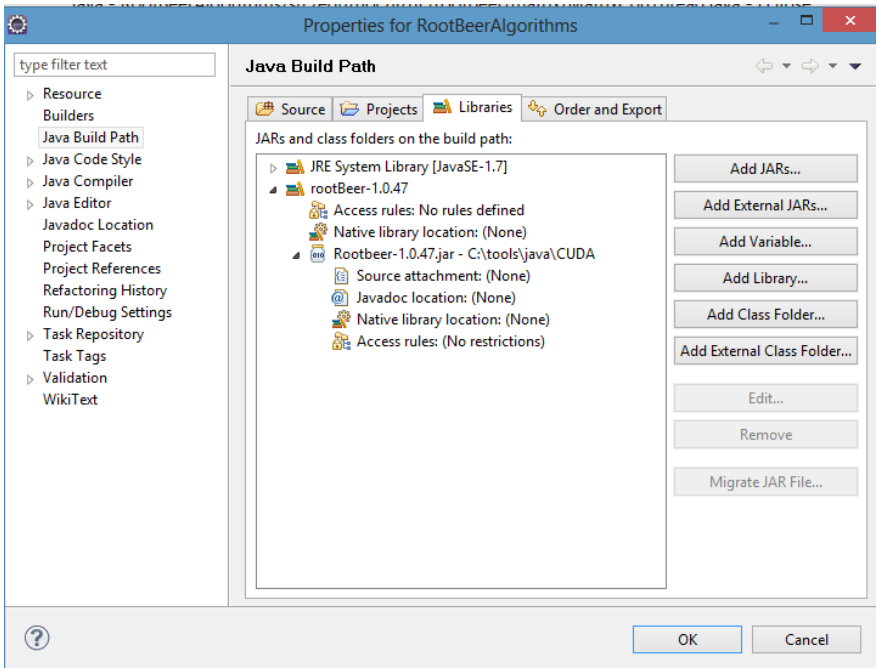
Rootbeer

Se trata de otra librería/precompilador Java con lo que precisa también del JDK/JRE. Se obtiene en forma de un único fichero *jar* . A partir de ahí se añade al *build path* del proyecto de forma similar a JCuda, y se hace uso de su API para interactuación con CUDA Driver y librerías de *toolkit.* Cabe indicar que lógicamente Rootbeer también tiene librerías de enlace nativo, *dll* y *so,* compatibles Windows y Linux, en el directorio del fichero *jar* "runtime2\native".

Los pasos de instalación serían:

1. Obtener RootBeer. En el PFC se han utilizado las versiones 1.0.46 y 1.0.47.
   http://rbcompiler.com/download.html

2. Añadir librería a proyecto Eclipse.



   A partir de aquí se utilizaría como librería normal Java.

3. Los programas Rootbeer incluidos en el PFC se han desarrollado de forma que el código de núcleo se pre-genera y compila durante la ejecución del mismo. Pero también se puede separar en clases aparte y pre-generar el código CUDA C con un Script independiente si este código no varía:

```
ant jar
java -jar ../../Rootbeer_1.0.47.jar QuickSort.jar QuickSort-GPU.jar
```

PyCuda

Esta herramienta es compatible con Python 2.7, y debe utilizarse conjuntamente con las librerías NumPy y Scipy. Como se indicó, es la única que no es fácilmente compatible con las últimas versiones de NVIDIA Cuda tookit y Visual Studio, y ha sido preciso instalar versiones anteriores de estas herramientas para ponerla en marcha.

Así, los pasos de instalación han sido:

1. Instalar NVIDIA toolkit 3.5 y Visual Studio 2008.

2. Instalar Python 2.7. Todas las plataformas:
   http://www.python.org/download/

3. Instalar librerías Python requeridas, NumPy 1.3.0 y Scipy 0.7.1
   http://www.lfd.uci.edu/~gohlke/pythonlibs

4. Instalar PyCuda.
   https://pypi.python.org/pypi/pycuda

5. Instalar Boost
   http://www.boost.org/doc/libs/1_46_1/more/getting_started/windows.html

6. Configurar **siteconf.py** (creado inicialmente con configure.py):
   BOOST_INC_DIR = ['C:\\Program Files\\boost\\boost_1_38_0']
   BOOST_LIB_DIR = ['C:\\Program Files\\boost\\boost_1_38_0\\stage\\lib']
   BOOST_COMPILER = 'msvc'
   BOOST_PYTHON_LIBNAME = ['boost_python-vc90-mt-1_38']
   BOOST_THREAD_LIBNAME = ['boost_thread-vc90-mt-1_38']
   CUDA_TRACE = False
   CUDA_ROOT = 'c:\\cuda'
   CUDA_ENABLE_GL = False
   CUDADRV_LIB_DIR = [r'C:/CUDA/lib']
   1CUDADRV_LIBNAME = ['cuda']
   CXXFLAGS = ['/EHsc']
   LDFLAGS = ['/FORCE']

7. Compilar configuración:
   python setup.py build
   python setup.py install

Anaconda Accelerate

Se trata de una herramienta con soporte comercial con gran facilidad de instalación. El complemento *Accelerate* básicamente añade capacidad CUDA a la distribución Python Anaconda, de la misma compañía, Continuum Analytics.

Esta distribución es la única compatible con *Accelerate* y no se puede usar directamente la distribución de referencia Python, (cython).

Los pasos de instalación serán:

1. Instalación de distribución Python Anaconda. Se facilitan instaladores para Linux, Mac y Windows.
   http://continuum.io/downloads

2. Instalación de *plug-in* Accelerate para soporte CUDA. Hay que tener presente que la distribución Python Anaconda es gratuita, pero no así este complemento. Para utilizarlo hay que registrarse para obtener una licencia temporal, 30 días, o bien comprar la licencia por 129$ a precio de Junio de 2013.
   https://store.continuum.io/cshop/accelerate

3. En el caso de utilizar un IDE Python, como PyCharm, establecer la versión de Python a utilizar en el proyecto para poder hacer uso de las librerías *Accelerate*.