NVIDIA® CUDA™ Architecture

Introduction & Overview

Version 1.1

April 2009

Introduction

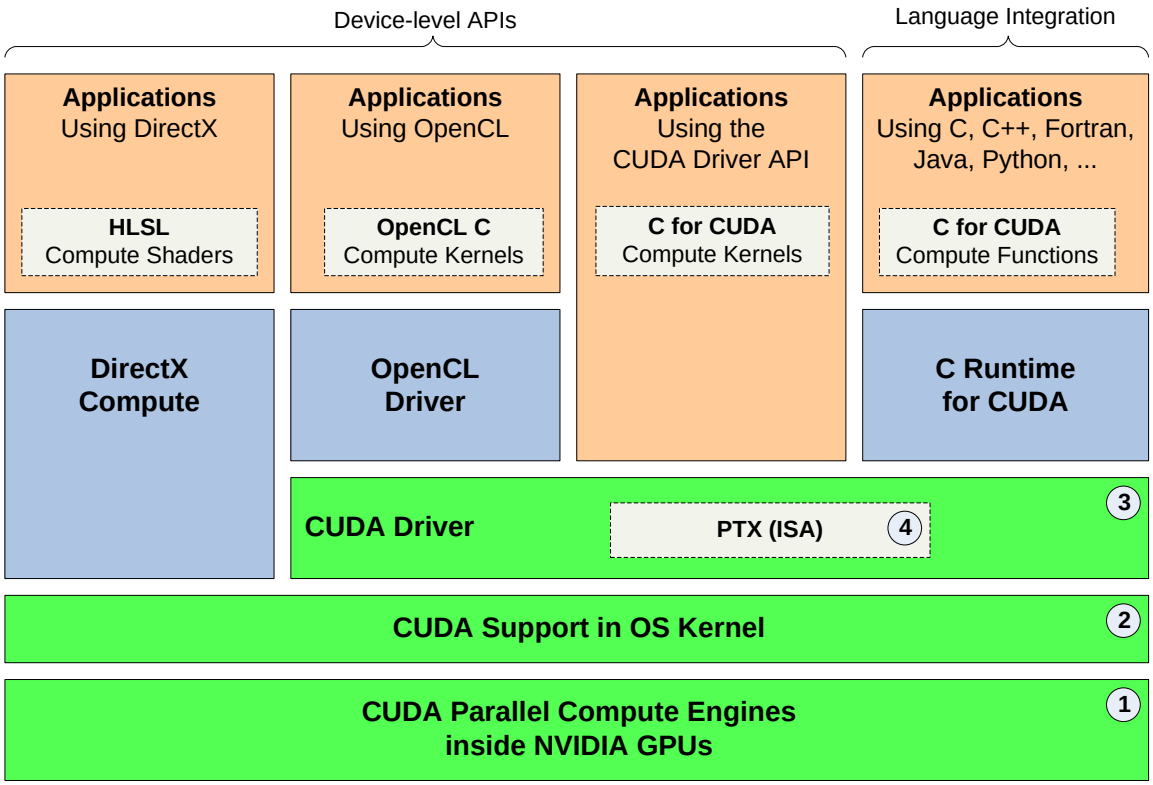
NVIDIA® CUDA™ technology leverages the massively parallel processing power of NVIDIA GPUs. The CUDA architecture is a revolutionary parallel computing architecture that delivers the performance of NVIDIA's world-renowned graphics processor technology to general purpose GPU Computing. Applications that run on the CUDA architecture can take advantage of an installed base of over one hundred million CUDA-enabled GPUs in desktop and notebook computers, professional workstations, and supercomputer clusters.

With the CUDA architecture and tools, developers are achieving dramatic speedups in fields such as medical imaging and natural resource exploration, and creating breakthrough applications in areas such as image recognition and real-time HD video playback and encoding. CUDA enables this unprecedented performance via standard APIs such as the soon to be released OpenCL™ and DirectX® Compute, and high level programming languages such as C/C++, Fortran, Java, Python, and the Microsoft .NET Framework.

The CUDA Architecture

The **CUDA Architecture** consists of several components, in the green boxes below:

1. Parallel compute engines inside NVIDIA GPUs
2. OS kernel-level support for hardware initialization, configuration, etc.
3. User-mode driver, which provides a device-level API for developers
4. PTX instruction set architecture (ISA) for parallel computing kernels and functions



The CUDA Software Development Environment

The CUDA Software Development Environment provides all the tools, examples and documentation necessary to develop applications that take advantage of the CUDA architecture.

Libraries	Advanced libraries that include BLAS, FFT, and other functions optimized for the CUDA architecture
C Runtime	The **C Runtime for CUDA** provides support for executing standard C functions on the GPU and allows native bindings for other high-level languages such as Fortran, Java, and Python
Tools	NVIDIA C Compiler (nvcc), CUDA Debugger (cudagdb), CUDA Visual Profiler (cudaprof), and other helpful tools
Documentation	Includes the CUDA Programming Guide, API specifications, and other helpful documentation
Samples	SDK code samples and documentation that demonstrate best practices for a wide variety GPU Computing algorithms and applications

The CUDA Software Development Environment supports two different programming interfaces:

1. A *device-level programming interface*, in which the application uses DirectX Compute, OpenCL or the CUDA Driver API directly to configure the GPU, launch compute *kernels*, and read back results.
2. A *language integration programming interface*, in which an application uses the C Runtime for CUDA and developers use a small set of extensions to indicate which compute *functions* should be performed on the GPU instead of the CPU.

When using the *device-level programming interface*, developers write compute kernels in separate files using the kernel language supported by their API of choice. DirectX Compute kernels (aka “compute shaders”) are written in HLSL. OpenCL kernels are written in a C-like language called “OpenCL C”. The CUDA Driver API accepts kernels written in C or PTX assembly.

When using the *language integration programming interface*, developers write compute functions in C and the C Runtime for CUDA automatically handles setting up the GPU and executing the compute functions. This programming interface enables developers to take advantage of native support for high-level languages such as C, C++, Fortran, Java, Python, and more (see below), reducing code complexity and development costs through *type integration* and *code integration*:

- **Type integration** allows standard types as well as vector types and user-defined types (including structs) to be used seamlessly across functions that are executed on the CPU and functions that are executed on the GPU.
- **Code integration** allows the same function to be called from functions that will be executed on the CPU and functions that will be executed on the GPU.

When necessary to distinguish functions that will be executed on the CPU from those that will be executed on the GPU, the term **C for CUDA** is used to describe the small set of extensions that allow developers to specify which functions will be executed on the GPU, how GPU memory will be used, and how the parallel processing capabilities of the GPU will be used by the application.

CUDA Adoption

First introduced in March 2007, and with over 100 million CUDA-enabled GPUs sold to date, thousands of software developers are already using the free CUDA software development tools to solve problems in a variety of professional and home applications – from video and image processing and physics simulations, to oil and gas exploration, product design, medical imaging, and scientific research.

Applications written in C and C++ can use the C Runtime for CUDA directly. Applications written in other languages can access the runtime via native method bindings, and there are several projects that enable developers to use the CUDA architecture this way, including:

Fortran:
- Fortran wrapper for CUDA – http://www.nvidia.com/object/cuda_programming_tools.html
- FLAGON Fortran 95 library for GPU Numerics – http://flagon.wiki.sourceforge.net/
- PGI Fortran to CUDA compiler – http://www.pgroup.com/resources/accel.htm

Java:
- JaCuda – http://jacuda.wiki.sourceforge.net
- Bindings for CUDA BLAS and FFT libs – http://javagl.de/index.html

Python:
- PyCUDA Python wrapper – http://mathema.tician.de/software/pycuda

.NET languages:
- CUDA.NET – http://www.gass-ltd.co.il/en/products/cuda.net

Resources for other languages:
- SWIG – http://www.swig.org (generates interfaces to C/C++ for dozens of languages)

Developers can take advantage of great performance on the CUDA architecture today using rich APIs and a variety of high-level languages on 32-bit and 64-bit versions of Linux, MacOS, and Windows.

For more information about GPU Computing and the CUDA architecture, please visit www.nvidia.com/cuda.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com