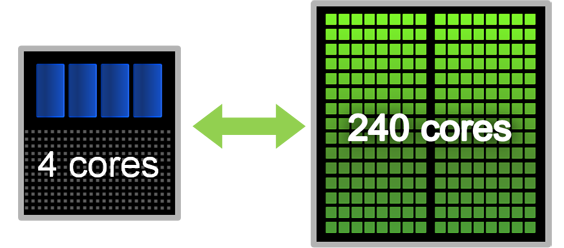
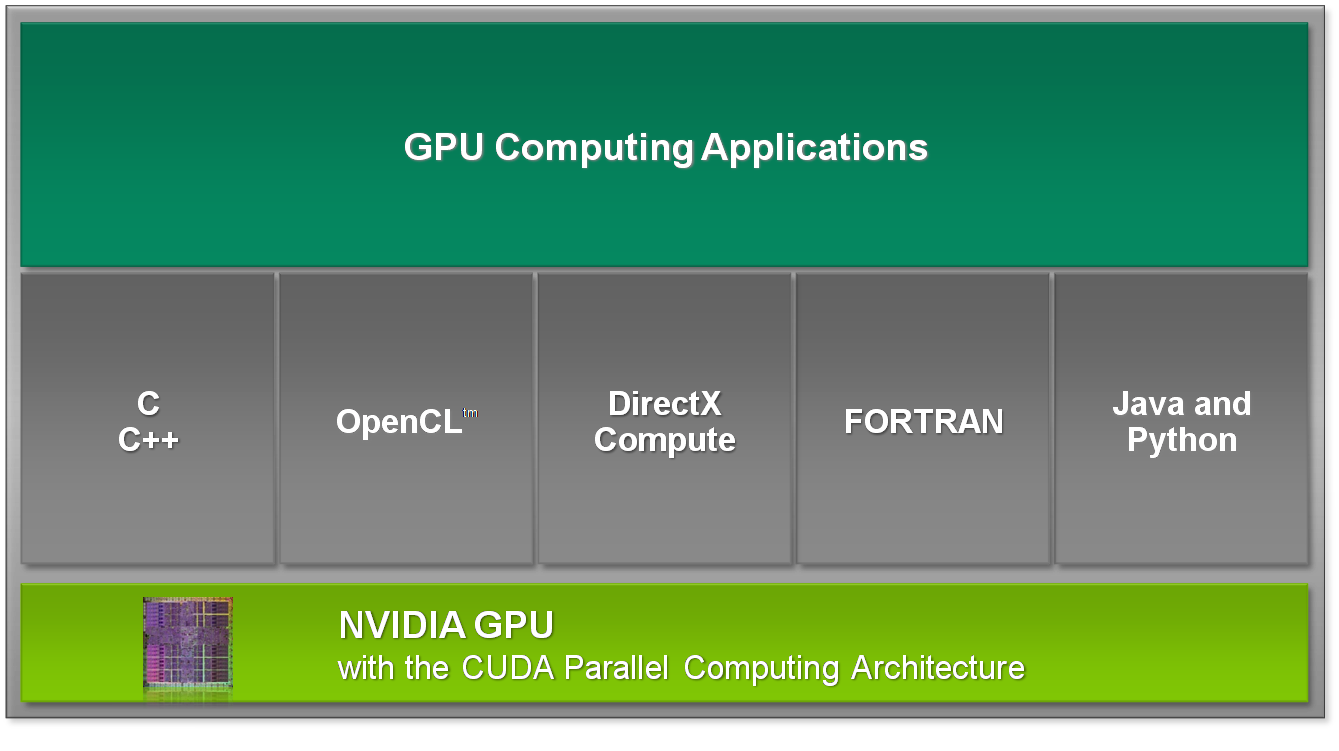
**<http://www.nvidia.com/object/GPU_Computing.html>**

# What is GPU Computing?

GPU computing is the use of a GPU (graphics processing unit) to do general purpose scientific and engineering computing.   
  
The model for GPU computing is to use a CPU and GPU together in a heterogeneous computing model. The sequential part of the application runs on the CPU and the computationally-intensive part runs on the GPU. From the user’s perspective, the application just runs faster because it is using the high-performance of the GPU to boost performance.   
  
  
The application developer has to modify their application to take the compute-intensive kernels and map them to the GPU. The rest of the application remains on the CPU. Mapping a function to the GPU involves rewriting the function to expose the parallelism in the function and adding “C” keywords to move data to and from the GPU.   
  
GPU computing is enabled by the massively parallel architecture of NVIDIA’s GPUs called the CUDA architecture. The CUDA architecture consists of 100s of processor cores that operate together to crunch through the data set in the application.  
  
The Tesla 10-series GPU is the second generation CUDA architecture with features optimized for scientific applications such as IEEE standard double precision floating point hardware support, local data caches in the form of shared memory dispersed throughout the GPU, coalesced memory accesses and so on.

"GPUs have evolved to the point where many real-world applications are easily implemented on them and run significantly faster than on multi-core systems. Future computing architectures will be hybrid systems with parallel-core GPUs working in tandem with multi-core CPUs."  
  
**Prof. Jack Dongarra**  
Director of the Innovative Computing Laboratory  
The University of Tennessee

History of GPU Computing  
Graphics chips started as fixed function graphics pipelines. Over the years, these graphics chips became increasingly programmable, which led NVIDIA to introduce the first GPU or Graphics Processing Unit. In the 1999-2000 timeframe, computer scientists in particular, along with researchers in fields such as medical imaging and electromagnetics started using GPUs for running general purpose computational applications. They found the excellent floating point performance in GPUs led to a huge performance boost for a range of scientific applications. This was the advent of the movement called **GPGPU** or General Purpose computing on GPUs.  
  
The problem was that GPGPU required using graphics programming languages like OpenGL and Cg to program the GPU. Developers had to make their scientific applications look like graphics applications and map them into problems that drew triangles and polygons. This limited the accessibility of tremendous performance of GPUs for science.  
  
NVIDIA realized the potential to bring this performance to the larger scientific community and decided to invest in modifying the GPU to make it fully programmable for scientific applications and added support for high-level languages like C and C++. This led to the **CUDA architecture** for the GPU.

CUDA Parallel Architecture and Programming Model  
The CUDA parallel hardware architecture is accompanied by the CUDA parallel programming model that provides a set of abstractions that enable expressing fine-grained and coarse-grain data and task parallelism. The programmer can choose to express the parallelism in high-level languages such as C, C++, Fortran or driver APIs such as OpenCL™ and DirectX™-11 Compute.  
  
  
  
The first language support NVIDIA provided is for the C language. A set of [C for CUDA software development tools](http://www.nvidia.com/cuda) enable the GPU to be programmed using C with a minimal set of keywords or extensions. Support for Fortran, OpenCL, et cetera will follow soon.  
  
The CUDA parallel programming model guides programmers to partition the problem into coarse sub-problems that can be solved independently in parallel. Fine grain parallelism in the sub-problems is then expressed such that each sub-problem can be solved cooperatively in parallel.

The CUDA GPU architecture and the corresponding CUDA parallel computing model are now widely deployed with 100s of applications and nearly a 1000 published research papers

**Bandwidth**

The bandwidth is particularly important for [I/O](http://www.webopedia.com/TERM/b/I_O.html) devices. For example, a fast [disk drive](http://www.webopedia.com/TERM/b/disk_drive.html) can be hampered by a [bus](http://www.webopedia.com/TERM/b/bus.html) with a low bandwidth. This is the main reason that new buses, such as [AGP](http://www.webopedia.com/TERM/b/AGP.html), have been developed for the [PC](http://www.webopedia.com/TERM/b/PC.html)..

latency

***(1)*** In general, the period of time that one component in a [system](http://www.webopedia.com/TERM/l/latency.html) is spinning its wheels waiting for another component. Latency, therefore, is wasted time. For example, in accessing data on a [disk](http://www.webopedia.com/TERM/l/disk.html), latency is defined as the time it takes to position the proper [sector](http://www.webopedia.com/TERM/l/sector.html) under the [read/write head](http://www.webopedia.com/TERM/l/head.html).