# OpenCL

Taking the graphics processor beyond graphics.

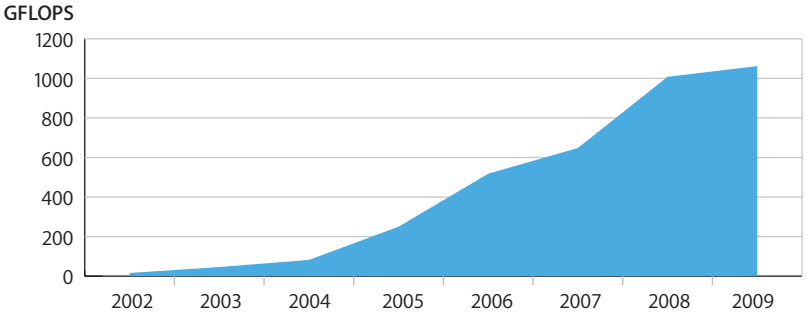

**Features**

- Hardware abstraction
- Familiar C-based language
- IEEE 754–based precision
- Optimized at runtime
- Works with OpenGL

Modern graphics processing units (GPUs) have evolved from single-purpose chips into flexible processors that offer levels of performance once reserved for room-sized supercomputers. OpenCL is a new API, language, and runtime in Mac OS X version 10.6 Snow Leopard that lets any application tap into the vast computing power of the GPU, opening up incredible performance opportunities.

## Supercomputer Performance

Each new generation of GPUs pushes the graphics-rendering envelope forward by delivering increased realism, fidelity, and resolution. As a result, today's GPUs are capable of rendering billions of pixels per second. Each pixel is the end-result of a complex set of mathematical operations. When viewed from the computational perspective, GPUs are performing operations at supercomputer performance levels, with the fastest GPUs performing around one trillion computations per second (1000 gigaFLOPS).




Source: NVIDIA

## Massive Parallelism

Behind the remarkable rise in GPU computing power is the dramatic increase in the amount of work a GPU performs at once. Because there are over a million pixels on a typical screen, the best way to rapidly render graphics is to process more than one pixel at a time. GPU designers now include large numbers of pixel processing elements on their chips. The more pixel processing elements a GPU has, the faster it can calculate all the pixels and produce the resulting graphics onscreen. The latest GPUs process over a hundred pixels simultaneously to fluidly render even the most complex 3D scenes.

## Moving Beyond Graphics

**More on graphics shaders**
Shaders are very specialized programs that allow specific processing steps in a GPU to be reprogrammed. Shaders allow common 3D graphics operations, such as vertex transformation and pixel color calculations, to be changed to suit the needs of the software developer without requiring a whole new graphics API.

Early GPUs were designed to specifically implement graphics programming standards such as OpenGL. The tight coupling between the language used by graphics programmers and the inner workings of the chips ensured good performance for most applications. However, this relationship limited the graphics-rendering realism to only that which was defined in the graphics language. To overcome this limitation, GPU designers eventually made the pixel processing elements customizable using specialized programs called graphics shaders.

Over time, developers and GPU vendors evolved shaders from simple assembly language programs into high-level programs that create the amazingly rich scenes found in today's 3D software. To handle increasing shader complexity, the vertex and pixel processing elements were redesigned to support more generalized math, logic, and flow control operations. This set the stage for a new way to accelerate computation.

## Harnessing the Power of the GPU

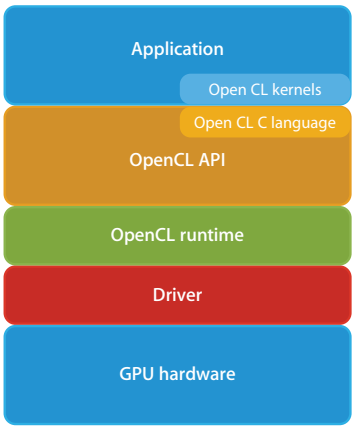
Apple realized that the trends in GPU designs offered an incredible opportunity to take the GPU beyond graphics. All that was needed was a nongraphics API that could engage the emerging programmable aspects of the GPU and access its immense power. OpenCL is that technology, delivering the means for any application to access the supercomputer-like performance of the modern GPU.

## OpenCL Up Close

**Also works with the CPU**
OpenCL is able to efficiently use multicore CPUs, which allows systems without an OpenCL-capable GPU to benefit from its capabilities.

OpenCL is designed from the ground up to accelerate application performance by using the GPU for general-purpose computations. It is a complete framework composed of an approachable C-based language with support for parallelism, and an API that allows applications to use one or more OpenCL devices (GPUs, CPUs, and so on) in the system.

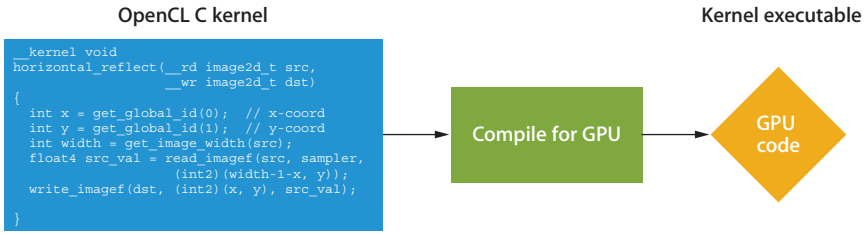
**The OpenCL architecture**



## OpenCL C

OpenCL defines OpenCL C, which is a variant of the familiar C99 language optimized for GPU programming. It incorporates changes necessary to adapt the C programming language for use with GPUs and to support parallel processing. OpenCL C includes comprehensive support for vector types to streamline data flow and increase efficiency. Well-defined numerical precision requirements (based on IEEE 754-2008) are specified to provide mathematical consistency across the GPU hardware of different vendors.

Developers use OpenCL C to rewrite just the performance- or data-intensive routines in their applications. During the rewrite, the routine is factored down to its most elemental state: a series of discrete operations that describe the computations that can be performed in parallel over a data set. The resulting code, which is similar to a traditional C function, is called an OpenCL *kernel*.




Unlike traditional C code, OpenCL kernels are incorporated into the application in an uncompiled state. They are compiled on the fly and optimized for the user's hardware before being sent to the GPU for processing.

## The OpenCL API

The OpenCL API provides functions that allow an application to manage parallel computing tasks. It enumerates the OpenCL-capable hardware in a system, sets up the sharing of data structures between the application and OpenCL, controls the compilation and submission of kernels to the GPU, and has a rich set of functions that manage queuing and synchronization.

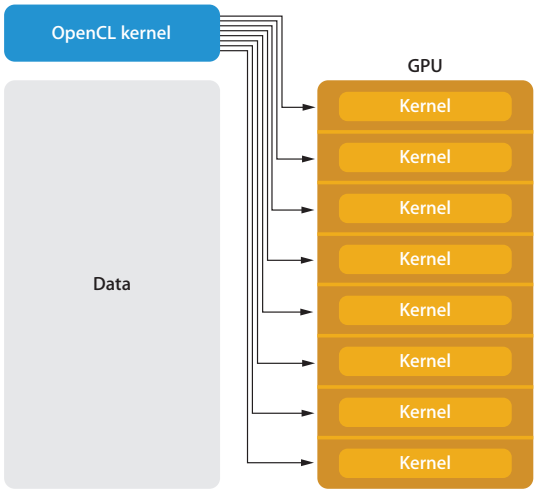
## OpenCL Runtime

The OpenCL runtime executes tasks submitted by the application via the OpenCL API. The runtime efficiently transfers data between main memory and the dedicated VRAM used by the GPU, and directs execution of the kernels on the GPU hardware. During execution, the OpenCL runtime manages the in-order or out-of-order dependencies between the kernels, and utilizes the GPU's processing elements in the most efficient manner.

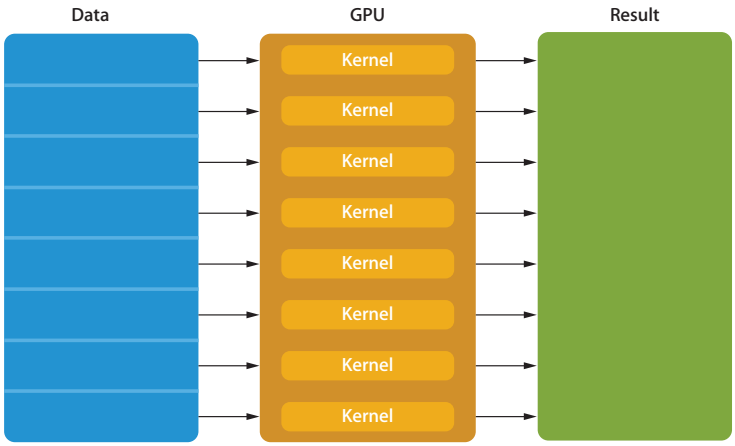
## OpenCL at Work

The following describes how an application interacts with OpenCL to perform GPU-accelerated computations.

At startup, the application calls the OpenCL API to determine which GPUs are available in the system. It then selects the appropriate GPUs and creates command queues. The application loads and compiles the OpenCL C kernels it will use.




When the application is ready to execute a kernel, it calls the OpenCL API to specify the data and the number of parallel kernel instances required. The OpenCL runtime moves the data required by the kernel up to the GPU's VRAM. The GPU then executes the kernel simultaneously on its processing elements.




This massively parallel execution of the kernel is the key to incredible performance, and differentiates OpenCL from other programming techniques such as multithreading on traditional processors. OpenCL using a modern 200-processing-element GPU performs 1000-element computation in only five iterations as 200 computations are performed in parallel with each iteration.

During execution, OpenCL manages the in-order or out-of-order dependencies between the kernels, so truly complex tasks composed of multiple OpenCL kernels can be scheduled to run efficiently across all of a GPU's processing elements.

OpenCL also performs its calculations asynchronously; the application can continue to run its main thread on the CPU while the GPU is executing kernels.

## OpenCL Performance Possibilities

By executing massive numbers of calculations in parallel, OpenCL can dramatically improve the speed and responsiveness of a wide-variety of applications such as games and scientific software. OpenCL also lets applications efficiently use very complex algorithms to deliver new functionality or tackle large processing tasks. Possibilities include realtime facial recognition, advanced video noise reduction and accelerated media transcoding.

## The OpenCL Standard

**The Khronos OpenCL Working Group**
3DLABS, Activision Blizzard, AMD, Apple, ARM, Broadcom, Codeplay, Electronic Arts, Ericsson, Freescale, Fujitsu, GE, Graphic Remedy, HI, IBM, Intel, Imagination Technologies, Los Alamos National Laboratory, Motorola, Movidia, Nokia, NVIDIA, Petapath, QNX, Qualcomm, RapidMind, Samsung, Seaweed Systems, S3, STMicroelectronics, Takumi Technology, Texas Instruments, and Toshiba.

While initially developing OpenCL, it became clear to Apple that the technology offered an opportunity for the industry to work together to define a standard for parallel programming. With the support of AMD, Intel, and NVIDIA, Apple proposed OpenCL to the Khronos Group consortium as the basis for a new standard. Demonstrating the strength of the proposal, OpenCL was expanded to include digital signal processors (DSPs) and other specialized processor architectures. It was ratified as an open, royalty-free open standard in December 2008.

## Conclusion

Mac OS X Snow Leopard ushers in a new generation of computing performance with OpenCL. Using this powerful new technology, Mac developers can easily access the incredible performance potential of the GPU for more than just graphics tasks. The comprehensive approach that OpenCL brings to parallel computation can accelerate a wide range of applications, from entertainment software to scientific solutions to image and video processing. With such huge potential, OpenCL is poised to become a pivotal technology not only for Mac developers, but for the entire computer industry.

## For More Information

For more information about Mac OS X v10.6 Snow Leopard, visit www.apple.com/macosx.