WIKIPEDIA

OpenCL

OpenCL (**Open Computing Language**) is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), field-programmable gate arrays (FPGAs) and other processors or hardware accelerators. OpenCL specifies programming languages (based on C99 and C++11) for programming these devices and application programming interfaces (APIs) to control the platform and execute programs on the compute devices. OpenCL provides a standard interface for parallel computing using task- and data-based parallelism.

OpenCL is an open standard maintained by the non-profit technology consortium Khronos Group. Conformant implementations are available from Altera, AMD, Apple (OpenCL along with OpenGL is deprecated for Apple hardware, in favor of Metal 2[7]), ARM, Creative, IBM, Imagination, Intel, Nvidia, Qualcomm, Samsung, Vivante, Xilinx, and ZiiLABS.[8][9]

OpenCL API

Original author(s)	Apple Inc.
Developer(s)	Khronos Group
Initial release	August 28, 2009
Stable release	3.0[1] / April 27, 2020
Written in	C with C++ bindings
Operating system	Android (vendor dependent),[2] FreeBSD,[3] Linux, macOS, Windows
Platform	ARMv7, ARMv8,[4] Cell, IA-32, POWER, x86-64
Type	Heterogeneous computing API
License	OpenCL specification license
Website	www.khronos.org/opencl/ (https://www.khronos.org/opencl/)

Contents

Overview
- Memory hierarchy

OpenCL C language
- Example: matrix-vector multiplication
- Example: computing the FFT

History
- OpenCL 1.0
- OpenCL 1.1
- OpenCL 1.2
- OpenCL 2.0
- OpenCL 2.1
- OpenCL 2.2
- OpenCL 3.0

Roadmap

Open Source implementations

Vendor implementations
- Timeline of vendor implementations

Devices

OpenCL C/C++

Khronos Conformance Test Suite
Conformant products
Version support
OpenCL 3.0 support
OpenCL 2.2 support
OpenCL 2.1 support
OpenCL 2.0 support
OpenCL 1.2 support
OpenCL 1.1 support
OpenCL 1.0 support
Portability, performance and alternatives
See also
References
External links

Paradigm	Imperative (procedural), structured, object-oriented (C++ only)
Family	C
Stable release	OpenCL C++ 1.0 revision V2.2-11[5] OpenCL C 3.0 revision V3.0.1[6] / April 27, 2020
Typing discipline	Static, weak, manifest, nominal
Implementation language	Implementation specific
Filename extensions	.cl
Website	www.khronos.org/opencl (https://www.khronos.org/opencl)
Major implementations	
AMD, Apple, freeocl, Gallium Compute, IBM, Intel Beignet, Intel SDK, Texas Instruments, Nvidia, pocl	
Influenced by	
C99, CUDA, C++14	

Overview

OpenCL views a computing system as consisting of a number of *compute devices*, which might be central processing units (CPUs) or "accelerators" such as graphics processing units (GPUs), attached to a *host* processor (a CPU). It defines a C-like language for writing programs. Functions executed on an OpenCL device are called "kernels".[10]:17 A single compute device typically consists of several *compute units*, which in turn comprise multiple *processing elements* (PEs). A single kernel execution can run on all or many of the PEs in parallel. How a compute device is subdivided into compute units and PEs is up to the vendor; a compute unit can be thought of as a "core", but the notion of core is hard to define across all the types of devices supported by OpenCL (or even within the category of "CPUs"),[11]:49–50 and the number of compute units may not correspond to the number of cores claimed in vendors' marketing literature (which may actually be counting SIMD lanes).[12]

In addition to its C-like programming language, OpenCL defines an application programming interface (API) that allows programs running on the host to launch kernels on the compute devices and manage device memory, which is (at least conceptually) separate from host memory. Programs in the OpenCL language are intended to be compiled at run-time, so that OpenCL-using applications are portable between implementations for various host devices.[13] The OpenCL standard defines host APIs for C and C++; third-party APIs exist for other programming languages and platforms such as Python,[14] Java, Perl[15] and .NET.[11]:15 An implementation of the OpenCL standard consists of a library that implements the API for C and C++, and an OpenCL C compiler for the compute device(s) targeted.

In order to open the OpenCL programming model to other languages or to protect the kernel source from inspection, the Standard Portable Intermediate Representation (SPIR)[16] can be used as a target-independent way to ship kernels between a front-end compiler and the OpenCL back-end.

More recently Khronos Group has ratified SYCL,[17] a higher-level programming model for OpenCL as single-source DSEL based on pure C++11 to improve programming productivity.

Memory hierarchy

OpenCL defines a four-level memory hierarchy for the compute device:[13]

- global memory: shared by all processing elements, but has high access latency (`__global`);
- read-only memory: smaller, low latency, writable by the host CPU but not the compute devices (`__constant`);
- local memory: shared by a group of processing elements (`__local`);
- per-element private memory (registers; `__private`).

Not every device needs to implement each level of this hierarchy in hardware. Consistency between the various levels in the hierarchy is relaxed, and only enforced by explicit synchronization constructs, notably barriers.

Devices may or may not share memory with the host CPU.[13] The host API provides handles on device memory buffers and functions to transfer data back and forth between host and devices.

OpenCL C language

The programming language that is used to write compute kernels is called OpenCL C and is based on C99,[18] but adapted to fit the device model in OpenCL. Memory buffers reside in specific levels of the memory hierarchy, and pointers are annotated with the region qualifiers `__global`, `__local`, `__constant`, and `__private`, reflecting this. Instead of a device program having a `main` function, OpenCL C functions are marked `__kernel` to signal that they are entry points into the program to be called from the host program. Function pointers, bit fields and variable-length arrays are omitted, and recursion is forbidden.[19] The C standard library is replaced by a custom set of standard functions, geared toward math programming.

OpenCL C is extended to facilitate use of parallelism with vector types and operations, synchronization, and functions to work with work-items and work-groups.[19] In particular, besides scalar types such as `float` and `double`, which behave similarly to the corresponding types in C, OpenCL provides fixed-length vector types such as `float4` (4-vector of single-precision floats); such vector types are available in lengths two, three, four, eight and sixteen for various base types.[18]:§ 6.1.2 Vectorized operations on these types are intended to map onto SIMD instructions sets, e.g., SSE or VMX, when running OpenCL programs on CPUs.[13] Other specialized types include 2-d and 3-d image types.[18]:10–11

Example: matrix-vector multiplication

The following is a matrix-vector multiplication algorithm in OpenCL C.

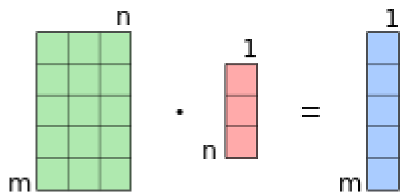
```
// Multiplies A*x, leaving the result in y.
// A is a row-major matrix, meaning the (i,j) element is at A[i*ncols+j].
__kernel void matvec(__global const float *A, __global const float *x,
                     uint ncols, __global float *y)
{
    size_t i = get_global_id(0);              // Global id, used as the
row index
    __global float const *a = &A[i*ncols];    // Pointer to the i'th row
    float sum = 0.f;                          // Accumulator for dot
```

```
    product
    for (size_t j = 0; j < ncols; j++) {
        sum += a[j] * x[j];
    }
    y[i] = sum;
}
```

The kernel function `matvec` computes, in each invocation, the dot product of a single row of a matrix A and a vector x:

$$y_i = a_{i,:} \cdot x = \sum_j a_{i,j} x_j.$$

To extend this into a full matrix-vector multiplication, the OpenCL runtime maps the kernel over the rows of the matrix. On the host side, the `clEnqueueNDRangeKernel` function does this; it takes as arguments the kernel to execute, its arguments, and a number of work-items, corresponding to the number of rows in the matrix A.



Each invocation (*work-item*) of the kernel takes a row of the green matrix (`A` in the code), multiplies this row with the red vector (`x`) and places the result in an entry of the blue vector (`y`). The number of columns n is passed to the kernel as `ncols`; the number of rows is implicit in the number of work-items produced by the host program.

Example: computing the FFT

This example will load a fast Fourier transform (FFT) implementation and execute it. The implementation is shown below.[20] The code asks the OpenCL library for the first available graphics card, creates memory buffers for reading and writing (from the perspective of the graphics card), JIT-compiles the FFT-kernel and then finally asynchronously runs the kernel. The result from the transform is not read in this example.

```
#include <stdio.h>
#include <time.h>
#include "CL/opencl.h"

#define NUM_ENTRIES 1024

int main() // (int argc, const char* argv[])
{
    // CONSTANTS
    // The source code of the kernel is represented as a string
    // located inside file: "fft1D_1024_kernel_src.cl". For the details see the next listing.
    const char *KernelSource =
        #include "fft1D_1024_kernel_src.cl"
            ;

    // Looking up the available GPUs
    const cl_uint num = 1;
    clGetDeviceIDs(NULL, CL_DEVICE_TYPE_GPU, 0, NULL, (cl_uint*)&num);

    cl_device_id devices[1];
    clGetDeviceIDs(NULL, CL_DEVICE_TYPE_GPU, num, devices, NULL);

    // create a compute context with GPU device
    cl_context context = clCreateContextFromType(NULL, CL_DEVICE_TYPE_GPU, NULL, NULL, NULL);

    // create a command queue
    clGetDeviceIDs(NULL, CL_DEVICE_TYPE_DEFAULT, 1, devices, NULL);
    cl_command_queue queue = clCreateCommandQueue(context, devices[0], 0, NULL);

    // allocate the buffer memory objects
    cl_mem memobjs[] = { clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, sizeof(float) * 2 *
NUM_ENTRIES, NULL, NULL),
                         clCreateBuffer(context, CL_MEM_READ_WRITE, sizeof(float) * 2 * NUM_ENTRIES, NULL, NULL)
};
    // cl_mem memobjs[0] = // FIXED, SEE ABOVE
    // cl_mem memobjs[1] = // FIXED, SEE ABOVE
```

```
    // create the compute program
    // const char* fft1D_1024_kernel_src[1] = {  };
    cl_program program = clCreateProgramWithSource(context, 1, (const char **)& KernelSource, NULL, NULL);

    // build the compute program executable
    clBuildProgram(program, 0, NULL, NULL, NULL, NULL);

    // create the compute kernel
    cl_kernel kernel = clCreateKernel(program, "fft1D_1024", NULL);

    // set the args values

    size_t local_work_size[1] = { 256 };

    clSetKernelArg(kernel, 0, sizeof(cl_mem), (void *)&memobjs[0]);
    clSetKernelArg(kernel, 1, sizeof(cl_mem), (void *)&memobjs[1]);
    clSetKernelArg(kernel, 2, sizeof(float)*(local_work_size[0] + 1) * 16, NULL);
    clSetKernelArg(kernel, 3, sizeof(float)*(local_work_size[0] + 1) * 16, NULL);

    // create N-D range object with work-item dimensions and execute kernel
    size_t global_work_size[1] = { 256 };

    global_work_size[0] = NUM_ENTRIES;
    local_work_size[0] = 64; //Nvidia: 192 or 256
    clEnqueueNDRangeKernel(queue, kernel, 1, NULL, global_work_size, local_work_size, 0, NULL, NULL);
}
```

The actual calculation inside file "fft1D_1024_kernel_src.cl" (based on Fitting FFT onto the G80 Architecture (http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project6_report.pdf)):[21]

```
R"(
  // This kernel computes FFT of length 1024. The 1024 length FFT is decomposed into
  // calls to a radix 16 function, another radix 16 function and then a radix 4 function

  __kernel void fft1D_1024 (__global float2 *in, __global float2 *out,
                            __local float *sMemx, __local float *sMemy) {
    int tid = get_local_id(0);
    int blockIdx = get_group_id(0) * 1024 + tid;
    float2 data[16];

    // starting index of data to/from global memory
    in = in + blockIdx;  out = out + blockIdx;

    globalLoads(data, in, 64); // coalesced global reads
    fftRadix16Pass(data);      // in-place radix-16 pass
    twiddleFactorMul(data, tid, 1024, 0);

    // local shuffle using local memory
    localShuffle(data, sMemx, sMemy, tid, (((tid & 15) * 65) + (tid >> 4)));
    fftRadix16Pass(data);               // in-place radix-16 pass
    twiddleFactorMul(data, tid, 64, 4); // twiddle factor multiplication

    localShuffle(data, sMemx, sMemy, tid, (((tid >> 4) * 64) + (tid & 15)));

    // four radix-4 function calls
    fftRadix4Pass(data);      // radix-4 function number 1
    fftRadix4Pass(data + 4);  // radix-4 function number 2
    fftRadix4Pass(data + 8);  // radix-4 function number 3
    fftRadix4Pass(data + 12); // radix-4 function number 4

    // coalesced global writes
    globalStores(data, out, 64);
  }
)"
```

A full, open source implementation of an OpenCL FFT can be found on Apple's website.[22]

History

OpenCL was initially developed by Apple Inc., which holds trademark rights, and refined into an initial proposal in collaboration with technical teams at AMD, IBM, Qualcomm, Intel, and Nvidia. Apple submitted this initial proposal to the Khronos Group. On June 16, 2008, the Khronos Compute Working Group was formed[23] with representatives from CPU, GPU, embedded-processor, and software companies. This group worked for five months to finish the technical details of the specification for OpenCL 1.0 by November 18, 2008.[24] This technical specification was reviewed by the Khronos members and approved for public release on December 8, 2008.[25]

OpenCL 1.0

OpenCL 1.0 released with Mac OS X Snow Leopard on August 28, 2009. According to an Apple press release:[26]

> Snow Leopard further extends support for modern hardware with Open Computing Language (OpenCL), which lets any application tap into the vast gigaflops of GPU computing power previously available only to graphics applications. OpenCL is based on the C programming language and has been proposed as an open standard.

AMD decided to support OpenCL instead of the now deprecated Close to Metal in its Stream framework.[27][28] RapidMind announced their adoption of OpenCL underneath their development platform to support GPUs from multiple vendors with one interface.[29] On December 9, 2008, Nvidia announced its intention to add full support for the OpenCL 1.0 specification to its GPU Computing Toolkit.[30] On October 30, 2009, IBM released its first OpenCL implementation as a part of the XL compilers.[31]

OpenCL 1.1

OpenCL 1.1 was ratified by the Khronos Group on June 14, 2010[32] and adds significant functionality for enhanced parallel programming flexibility, functionality, and performance including:

- New data types including 3-component vectors and additional image formats;
- Handling commands from multiple host threads and processing buffers across multiple devices;
- Operations on regions of a buffer including read, write and copy of 1D, 2D, or 3D rectangular regions;
- Enhanced use of events to drive and control command execution;
- Additional OpenCL built-in C functions such as integer clamp, shuffle, and asynchronous strided copies;
- Improved OpenGL interoperability through efficient sharing of images and buffers by linking OpenCL and OpenGL events.

OpenCL 1.2

On November 15, 2011, the Khronos Group announced the OpenCL 1.2 specification,[33] which added significant functionality over the previous versions in terms of performance and features for parallel programming. Most notable features include:

- Device partitioning: the ability to partition a device into sub-devices so that work assignments can be allocated to individual compute units. This is useful for reserving areas of the device to reduce latency for time-critical tasks.
- Separate compilation and linking of objects: the functionality to compile OpenCL into external libraries for inclusion into other programs.
- Enhanced image support: 1.2 adds support for 1D images and 1D/2D image arrays. Furthermore, the OpenGL sharing extensions now allow for OpenGL 1D textures and 1D/2D texture arrays to be used to create OpenCL images.
- Built-in kernels: custom devices that contain specific unique functionality are now integrated more closely into the OpenCL framework. Kernels can be called to use specialised or non-programmable aspects of underlying hardware. Examples include video encoding/decoding and digital signal processors.
- DirectX functionality: DX9 media surface sharing allows for efficient sharing between OpenCL and DX9 or DXVA media surfaces. Equally, for DX11, seamless sharing between OpenCL and DX11 surfaces is enabled.
- The ability to force IEEE 754 compliance for single precision floating point math: OpenCL by default allows the single precision versions of the division, reciprocal, and square root operation to be less accurate than the correctly rounded values that IEEE 754 requires.[34] If the programmer passes the "-cl-fp32-correctly-rounded-divide-sqrt" command line argument to the compiler, these three operations will be computed to IEEE 754 requirements if the OpenCL implementation supports this, and will fail to compile if the OpenCL implementation does not support computing these operations to their correctly-rounded values as defined by the IEEE 754 specification.[34] This ability is supplemented by the ability to query the OpenCL implementation to determine if it can perform these operations to IEEE 754 accuracy.[34]

OpenCL 2.0

On November 18, 2013, the Khronos Group announced the ratification and public release of the finalized OpenCL 2.0 specification.[35] Updates and additions to OpenCL 2.0 include:

- Shared virtual memory
- Nested parallelism
- Generic address space
- Images
- C11 atomics
- Pipes
- Android installable client driver extension

OpenCL 2.1

The ratification and release of the OpenCL 2.1 provisional specification was announced on March 3, 2015 at the Game Developer Conference in San Francisco. It was released on November 16, 2015.[36] It introduced the OpenCL C++ kernel language, based on a subset of C++14, while maintaining support for the preexisting OpenCL C kernel language. Vulkan and OpenCL 2.1 share SPIR-V as an intermediate representation allowing high-level language front-ends to share a common compilation target. Updates to the OpenCL API include:

- Additional subgroup functionality
- Copying of kernel objects and states
- Low-latency device timer queries
- Ingestion of SPIR-V code by runtime
- Execution priority hints for queues
- Zero-sized dispatches from host

AMD, ARM, Intel, HPC, and YetiWare have declared support for OpenCL 2.1.[37][38]

OpenCL 2.2

OpenCL 2.2 brings the OpenCL C++ kernel language into the core specification for significantly enhanced parallel programming productivity.[39][40][41] It was released on May 16, 2017.[42] Maintenance Update released in May 2018 with bugfixes.[43]

- The OpenCL C++ kernel language is a static subset of the C++14 standard and includes classes, templates, lambda expressions, function overloads and many other constructs for generic and meta-programming.
- Uses the new Khronos SPIR-V 1.1 intermediate language which fully supports the OpenCL C++ kernel language.
- OpenCL library functions can now use the C++ language to provide increased safety and reduced undefined behavior while accessing features such as atomics, iterators, images, samplers, pipes, and device queue built-in types and address spaces.
- Pipe storage is a new device-side type in OpenCL 2.2 that is useful for FPGA implementations by making connectivity size and type known at compile time, enabling efficient device-scope communication between kernels.
- OpenCL 2.2 also includes features for enhanced optimization of generated code: applications can provide the value of specialization constant at SPIR-V compilation time, a new query can detect non-trivial constructors and destructors of program scope global objects, and user callbacks can be set at program release time.
- Runs on any OpenCL 2.0-capable hardware (only driver update required)

OpenCL 3.0

OpenCL 3.0 is in provisional Mode. OpenCL 1.2 is mandatory. All OpenCL 2.x Modules and new 3.0 modules are optional. [44][45]

Roadmap

When releasing OpenCL 2.2, the Khronos Group announced that OpenCL would converge where possible with Vulkan to enable OpenCL software deployment flexibility over both APIs.[46][47] This has been now demonstrated by Adobe's Premiere Rush using the clspv[48] open source compiler to compile significant amounts of OpenCL C kernel code to run on a Vulkan runtime for deployment on Android.[49] OpenCL has a forward looking roadmap independent of Vulkan, with 'OpenCL Next' under development and targeting release in 2020. OpenCL Next may integrate extensions such as Vulkan / OpenCL Interop, Scratch-Pad Memory Management, Extended Subgroups, SPIR-V 1.4 ingestion and SPIR-V Extended debug info. OpenCL is also considering Vulkan-like loader and layers and a ‘Flexible Profile’ for deployment flexibility on multiple accelerator types.[50]

Open Source implementations

The International Workshop on OpenCL (IWOCL) held by the Khronos Group

OpenCL consists of a set of headers and a shared object that is loaded at runtime. An installable client driver (ICD) must be installed on the platform for every class of vendor for which the runtime would need to support. That is, for example, in order to support Nvidia devices on a Linux platform, the Nvidia ICD would need to be installed such that the OpenCL runtime (the ICD loader) would be able to locate the ICD for the vendor and redirect the calls appropriately. The standard OpenCL header is used by the consumer application; calls to each function are then proxied by the OpenCL runtime to the appropriate driver using the ICD. Each vendor must implement each OpenCL call in their driver.[51]

The Apple,[52] Nvidia,[53] RapidMind[54] and Gallium3D[55] implementations of OpenCL are all based on the LLVM Compiler technology and use the Clang compiler as its frontend.

MESA Gallium Compute

An implementation of OpenCL (actual 1.1 incomplete, mostly done AMD Radeon GCN) for a number of platforms is maintained as part of the Gallium Compute Project,[56] which builds on the work of the Mesa project to support multiple platforms. Formerly this was known as CLOVER.,[57] actual development: mostly support for running incomplete framework with actual LLVM and CLANG, some new features like fp16 in 17.3,[58] Target complete OpenCL 1.0, 1.1 and 1.2 for AMD and Nvidia. New Basic Development is done by Red Hat with SPIR-V also for Clover.[59][60]

BEIGNET

An implementation by Intel for its Ivy Bridge + hardware was released in 2013.[61] This software from Intel's China Team, has attracted criticism from developers at AMD and Red Hat,[62] as well as Michael Larabel of Phoronix.[63] Actual Version 1.3.2 support OpenCL 1.2 complete (Ivy Bridge and higher) and OpenCL 2.0 optional for Skylake and newer.[64][65] support for Android has been added to Beignet.,[66] actual development targets: only support for 1.2 and 2.0, road to OpenCL 2.1 and 2.2 is gone to NEO.

NEO

An implementation by Intel for Gen. 8 Broadwell + Gen. 9 hardware released in 2018.[67] This driver replaces Beignet implementation for supported platforms. NEO provides OpenCL 2.1 support on Core platforms and OpenCL 1.2 on Atom platforms.[68] Actual in 2020 also Graphic Gen 11 Ice Lake and Gen 12 Tiger Lake are supported.

ROCm

Created as part of AMD's GPUOpen, ROCm (Radeon Open Compute) is an open source Linux project built on OpenCL 1.2 with language support for 2.0. The system is compatible with all modern AMD CPUs and APUs (actual partly GFX 7, GFX 8 and 9), as well as Intel Gen7.5+ CPUs (only with PCI 3.0).[69][70] With version 1.9 support is in some points extended experimental to Hardware with PCIe 2.0 and without atomics. An overview of actual work is done on XDC2018.[71][72] Actual ROCm Version 2.0 supports Full OpenCL 2.0, but some errors and limitations are on the todo list.[73][74] Version 3.3 is improving in details.[75] Actual documentation is available at github.[76]

POCL

A portable implementation supporting CPUs and some GPUs (via CUDA and HSA). Building on Clang and LLVM.[77] With version 1.0 OpenCL 1.2 was nearly fully implemented along with some 2.x features.[78] Actual is Version 1.2 with LLVM/CLANG 6.0, 7.0 and Full OpenCL 1.2 support with all closed tickets in Milestone 1.2.[78][79] OpenCL 2.0 is nearly full implemented.[80] Version 1.3 Supports Mac OS X.[81] Version 1.4 includes support for LLVM 8.0 and 9.0.[82] Version 1.5 implements LLVM/Clang 10 support. [83]

Shamrock

A Port of Mesa Clover for ARM with full support of OpenCL 1.2,[84][85] no actual development for 2.0.

FreeOCL

A CPU focused implementation of OpenCL 1.2 that implements an external compiler to create a more reliable platform,[86] no actual development.

MOCL

An OpenCL implementation based on POCL by the NUDT researchers for Matrix-2000 was released in 2018. The Matrix-2000 architecture is designed to replace the Intel Xeon Phi accelerators of the TianHe-2 supercomputer. This programming framework is built on top of LLVM v5.0 and reuses some code pieces from POCL as well. To unlock the hardware potential, the device runtime uses a push-based task dispatching strategy and the performance of the kernel atomics is improved significantly. This framework has been deployed on the TH-2A system and is readily available to the public.[87] Some of the software will next ported to improve POCL.[78]

Vendor implementations

Timeline of vendor implementations

- December 10, 2008: AMD and Nvidia held the first public OpenCL demonstration, a 75-minute presentation at SIGGRAPH Asia 2008. AMD showed a CPU-accelerated OpenCL demo explaining the scalability of OpenCL on one or more cores while Nvidia showed a GPU-accelerated demo.[88][89]
- March 16, 2009: at the 4th Multicore Expo, Imagination Technologies announced the PowerVR SGX543MP, the first GPU of this company to feature OpenCL support.[90]
- March 26, 2009: at GDC 2009, AMD and Havok demonstrated the first working implementation for OpenCL accelerating Havok Cloth on AMD Radeon HD 4000 series GPU.[91]
- April 20, 2009: Nvidia announced the release of its OpenCL driver and SDK to developers participating in its OpenCL Early Access Program.[92]
- August 5, 2009: AMD unveiled the first development tools for its OpenCL platform as part of its ATI Stream SDK v2.0 Beta Program.[93]
- August 28, 2009: Apple released Mac OS X Snow Leopard, which contains a full implementation of OpenCL.[94]
- September 28, 2009: Nvidia released its own OpenCL drivers and SDK implementation.
- October 13, 2009: AMD released the fourth beta of the ATI Stream SDK 2.0, which provides a complete OpenCL implementation on both R700/R800 GPUs and SSE3 capable CPUs. The SDK is available for both Linux and Windows.[95]
- November 26, 2009: Nvidia released drivers for OpenCL 1.0 (rev 48).

- October 27, 2009: S3 released their first product supporting native OpenCL 1.0 – the Chrome 5400E embedded graphics processor.[96]
- December 10, 2009: VIA released their first product supporting OpenCL 1.0 – ChromotionHD 2.0 video processor included in VN1000 chipset.[97]
- December 21, 2009: AMD released the production version of the ATI Stream SDK 2.0,[98] which provides OpenCL 1.0 support for R800 GPUs and beta support for R700 GPUs.
- June 1, 2010: ZiiLABS released details of their first OpenCL implementation for the ZMS processor for handheld, embedded and digital home products.[99]
- June 30, 2010: IBM released a fully conformant version of OpenCL 1.0.[4]
- September 13, 2010: Intel released details of their first OpenCL implementation for the Sandy Bridge chip architecture. Sandy Bridge will integrate Intel's newest graphics chip technology directly onto the central processing unit.[100]
- November 15, 2010: Wolfram Research released Mathematica 8 with OpenCLLink (http://reference.wolfram.com/mathematica/OpenCLLink/tutorial/Overview.html) package.
- March 3, 2011: Khronos Group announces the formation of the WebCL working group to explore defining a JavaScript binding to OpenCL. This creates the potential to harness GPU and multi-core CPU parallel processing from a Web browser.[101][102]
- March 31, 2011: IBM released a fully conformant version of OpenCL 1.1.[4][103]
- April 25, 2011: IBM released OpenCL Common Runtime v0.1 for Linux on x86 Architecture.[104]
- May 4, 2011: Nokia Research releases an open source WebCL extension for the Firefox web browser, providing a JavaScript binding to OpenCL.[105]
- July 1, 2011: Samsung Electronics releases an open source prototype implementation of WebCL for WebKit, providing a JavaScript binding to OpenCL.[106]
- August 8, 2011: AMD released the OpenCL-driven AMD Accelerated Parallel Processing (APP) Software Development Kit (SDK) v2.5, replacing the ATI Stream SDK as technology and concept.[107]
- December 12, 2011: AMD released AMD APP SDK v2.6[108] which contains a preview of OpenCL 1.2.
- February 27, 2012: The Portland Group released the PGI OpenCL compiler for multi-core ARM CPUs.[109]
- April 17, 2012: Khronos released a WebCL working draft.[110]
- May 6, 2013: Altera released the Altera SDK for OpenCL, version 13.0.[111] It is conformant to OpenCL 1.0.[112]
- November 18, 2013: Khronos announced that the specification for OpenCL 2.0 had been finalized.[113]
- March 19, 2014: Khronos releases the WebCL 1.0 specification[114][115]
- August 29, 2014: Intel releases HD Graphics 5300 driver that supports OpenCL 2.0.[116]
- September 25, 2014: AMD releases Catalyst 14.41 RC1, which includes an OpenCL 2.0 driver.[117]
- January 14, 2015: Xilinx Inc. announces SDAccel development environment for OpenCL, C, and C++, achieves Khronos Conformance[118]
- April 13, 2015: Nvidia releases WHQL driver v350.12, which includes OpenCL 1.2 support for GPUs based on Kepler or later architectures.[119] Driver 340+ support OpenCL 1.1 for Tesla and Fermi.

- August 26, 2015: AMD released AMD APP SDK v3.0[120] which contains full support of OpenCL 2.0 and sample coding.
- November 16, 2015: Khronos announced that the specification for OpenCL 2.1 had been finalized.[121]
- April 18, 2016: Khronos announced that the specification for OpenCL 2.2 had been provisionally finalized.[40]
- November 3, 2016 Intel support for Gen7+ of OpenCL 2.1 in SDK 2016 r3[122]
- February 17, 2017: Nvidia begins evaluation support of OpenCL 2.0 with driver 378.66.[123][124][125]
- May 16, 2017: Khronos announced that the specification for OpenCL 2.2 had been finalized with SPIR-V 1.2.[126]
- May 14, 2018: Khronos announced Maintenance Update for OpenCL 2.2 with Bugfix and unified headers.[43]
- April 27, 2020: Khronos announced provisional Version of OpenCL 3.0

Devices

As of 2016, OpenCL runs on Graphics processing units, CPUs with SIMD instructions, FPGAs, Movidius Myriad 2, Adapteva epiphany and DSPs.

Khronos Conformance Test Suite

> To be officially conformant, an implementation must pass the Khronos Conformance Test Suite (CTS), with results being submitted to the Khronos Adopters Program.[127] The Khronos CTS code for all OpenCL versions has been available in open source since 2017.[128]

Conformant products

The Khronos Group maintains an extended list of OpenCL-conformant products.[4]

Synopsis of OpenCL conformant products[4]				
AMD SDKs (https://developer.amd.com/tools-and-sdks/) (supports OpenCL CPU and accelerated processing unit Devices), (GPU: Terascale 1: OpenCL 1.1, Terascale 2: 1.2, GCN 1: 1.2+, GCN 2+: 2.0+)	X86 + SSE2 (or higher) compatible CPUs 64-bit & 32-bit,[129] Linux 2.6 PC, Windows Vista/7/8.x/10 PC	AMD Fusion E-350, E-240, C-50, C-30 with HD 6310/HD 6250	AMD Radeon/Mobility HD 6800, HD 5x00 series GPU, iGPU HD 6310/HD 6250, HD 7xxx, HD 8xxx, R2xx, R3xx, RX 4xx, RX 5xx, Vega Series	AMD FirePro Vx800 series GPU and later, Radeon Pro
Intel SDK for OpenCL Applications 2013 (http://software.intel.com/en-us/vcsource/tools/opencl-sdk)[130] (supports Intel Core processors and Intel HD Graphics 4000/2500) actual 2017 R2 with OpenCL 2.1 (Gen7+), SDK 2019 in Beta,[131]	Intel CPUs with SSE 4.1, SSE 4.2 or AVX support.[132][133] Microsoft Windows, Linux	Intel Core i7, i5, i3; 2nd Generation Intel Core i7/5/3, 3rd Generation Intel Core Processors with Intel HD Graphics 4000/2500 and newer	Intel Core 2 Solo, Duo Quad, Extreme and newer	Intel Xeon 7x00,5x00,3x00 (Core based) and newer
IBM Servers with OpenCL Development Kit (http://www.alphaworks.ibm.com/tech/opencl) for Linux on Power running on Power VSX[134][135]	IBM Power 775 (PERCS), 750	IBM BladeCenter PS70x Express	IBM BladeCenter JS2x, JS43	IBM BladeCenter QS22
IBM OpenCL Common Runtime (OCR) (http://www.alphaworks.ibm.com/tech/ocr) [136]	X86 + SSE2 (or higher) compatible CPUs 64-bit & 32-bit;[137] Linux 2.6 PC	AMD Fusion, Nvidia Ion and Intel Core i7, i5, i3; 2nd Generation Intel Core i7/5/3	AMD Radeon, Nvidia GeForce and Intel Core 2 Solo, Duo, Quad, Extreme	ATI FirePro, Nvidia Quadro and Intel Xeon 7x00,5x00,3x00 (Core based)
Nvidia OpenCL Driver and Tools (http://developer.nvidia.com/opencl),[138] Chips: Tesla, Fermi : OpenCL 1.1(Driver 340+), Kepler, Maxwell, Pascal, Volta, Turing: OpenCL 1.2 (Driver 370+), OpenCL 2.0 beta (378.66)	Nvidia Tesla C/D/S	Nvidia GeForce GTS/GT/GTX,	Nvidia Ion	Nvidia Quadro FX/NVX/Plex, Quadro, Quadro K, Quadro M, Quadro P, Quadro with Volta, Quadro RTX with Turing

All standard-conformant implementations can be queried using one of the clinfo tools (there are multiple tools with the same name and similar feature set).[139][140][141]

Version support

Products and their version of OpenCL support include:[142]

OpenCL 3.0 support

None yet: Khronos Test Suite work in progress [143]

OpenCL 2.2 support

None yet: Khronos Test Suite ready, with Driver Update all Hardware with 2.0 and 2.1 support possible

- Intel NEO Compute: Work in Progress for actual products[144]

OpenCL 2.1 support

- (2018+) Support backported to Intel 5th and 6th gen processors (Broadwell, Skylake)
- (2017+) Intel 7th, 8th, 9th & 10th gen processors (Kaby Lake, Coffee Lake, Comet Lake, Ice Lake, Tiger Lake)
- Khronos: with Driver Update all Hardware with 2.0 support possible

OpenCL 2.0 support

- (2011+) AMD GCN GPU's (HD 7700+/HD 8000/Rx 200/Rx 300/Rx 400/Rx 500-Series), some GCN 1st Gen only 1.2 with some Extensions
- (2013+) AMD GCN APU's (Jaguar, Steamroller, Puma, Excavator & Zen-based)
- (2014+) Intel 5th & 6th gen processors (Broadwell, Skylake)
- (2015+) Qualcomm Adreno 5xx series
- (2018+) Qualcomm Adreno 6xx series
- (2017+) ARM Mali (Bifrost) G51 and G71 in Android 7.1 and Linux
- (2018+) ARM Mali (Bifrost) G31, G52, G72 and G76
- (2017+) incomplete Evaluation support: Nvidia Kepler, Maxwell, Pascal, Volta and Turing GPU's (GeForce 600, 700, 800, 900 & 10-series, Quadro K-, M- & P-series, Tesla K-, M- & P-series) with Driver Version 378.66+

OpenCL 1.2 support

- (2011+) for some AMD GCN 1st Gen some OpenCL 2.0 Features not possible today, but many more Extensions than Terascale
- (2009+) AMD TeraScale 2 & 3 GPU's (RV8xx, RV9xx in HD 5000, 6000 & 7000 Series)
- (2011+) AMD TeraScale APU's (K10, Bobcat & Piledriver-based)
- (2012+) Nvidia Kepler, Maxwell, Pascal, Volta and Turing GPU's (GeForce 600, 700, 800, 900, 10, 16, 20 series, Quadro K-, M- & P-series, Tesla K-, M- & P-series)
- (2012+) Intel 3rd & 4th gen processors (Ivy Bridge, Haswell)
- (2013+) Qualcomm Adreno 4xx series
- (2013+) ARM Mali Midgard 3rd gen (T760)
- (2015+) ARM Mali Midgard 4th gen (T8xx)

OpenCL 1.1 support

- (2008+) some AMD TeraScale 1 GPU's (RV7xx in HD4000-series)
- (2008+) Nvidia Tesla, Fermi GPU's (GeForce 8, 9, 100, 200, 300, 400, 500-series, Quadro-series or Tesla-series with Tesla or Fermi GPU)
- (2011+) Qualcomm Adreno 3xx series
- (2012+) ARM Mali Midgard 1st and 2nd gen (T-6xx, T720)

OpenCL 1.0 support

- mostly updated to 1.1 and 1.2 after first Driver for 1.0 only

Portability, performance and alternatives

A key feature of OpenCL is portability, via its abstracted memory and execution model, and the programmer is not able to directly use hardware-specific technologies such as inline Parallel Thread Execution (PTX) for Nvidia GPUs unless they are willing to give up direct portability on other platforms. It is possible to run any OpenCL kernel on any conformant implementation.

However, performance of the kernel is not necessarily portable across platforms. Existing implementations have been shown to be competitive when kernel code is properly tuned, though, and auto-tuning has been suggested as a solution to the performance portability problem,[145] yielding "acceptable levels of performance" in experimental linear algebra kernels.[146] Portability of an entire application containing multiple kernels with differing behaviors was also studied, and shows that portability only required limited tradeoffs.[147]

A study at Delft University from 2011 that compared CUDA programs and their straightforward translation into OpenCL C found CUDA to outperform OpenCL by at most 30% on the Nvidia implementation. The researchers noted that their comparison could be made fairer by applying manual optimizations to the OpenCL programs, in which case there was "no reason for OpenCL to obtain worse performance than CUDA". The performance differences could mostly be attributed to differences in the programming model (especially the memory model) and to NVIDIA's compiler optimizations for CUDA compared to those for OpenCL.[145]

Another study at D-Wave Systems Inc. found that "The OpenCL kernel’s performance is between about 13% and 63% slower, and the end-to-end time is between about 16% and 67% slower" than CUDA's performance.[148]

The fact that OpenCL allows workloads to be shared by CPU and GPU, executing the same programs, means that programmers can exploit both by dividing work among the devices.[149] This leads to the problem of deciding how to partition the work, because the relative speeds of operations differ among the devices. Machine learning has been suggested to solve this problem: Grewe and O'Boyle describe a system of support-vector machines trained on compile-time features of program that can decide the device partitioning problem statically, without actually running the programs to measure their performance.[150]

- Project Coriander: Conversion CUDA to OpenCL 1.2 with CUDA-on-CL[151][152][153]

See also

- Advanced Simulation Library
- AMD FireStream
- BrookGPU
- C++ AMP
- Close to Metal
- CUDA

- DirectCompute
- GPGPU
- HIP
- Larrabee
- Lib Sh
- List of OpenCL applications
- OpenACC
- OpenGL
- OpenHMPP
- OpenMP
- Metal
- Renderscript
- SequenceL
- SIMD
- SYCL
- Vulkan
- WebCL

References

1. "Khronos OpenCL Registry" (https://www.khronos.org/registry/OpenCL/). Khronos Group. April 27, 2020. Retrieved April 27, 2020.
2. "Android Devices With OpenCL support" (https://docs.google.com/a/arrayfire.com/spreadsheets/d/1Mpzfl2NmLUVSAjIph77-FOsJeuyD9Xjha89r5iHw1hI/edit?pli=1#gid=0). *Google Docs*. ArrayFire. Retrieved April 28, 2015.
3. "FreeBSD Graphics/OpenCL" (https://wiki.freebsd.org/Graphics/OpenCL). FreeBSD. Retrieved December 23, 2015.
4. "Conformant Products" (https://www.khronos.org/conformance/adopters/conformant-products/opencl). Khronos Group. Retrieved May 9, 2015.
5. Sochacki, Bartosz (July 19, 2019). "The OpenCL C++ 1.0 Specification" (https://www.khronos.org/registry/OpenCL/specs/opencl-2.2-cplusplus.pdf) (PDF). Khronos OpenCL Working Group. Retrieved July 19, 2019.
6. Munshi, Aaftab; Howes, Lee; Sochaki, Barosz (April 27, 2020). "The OpenCL C Specification Version: 2.0 Document Revision: 33" (https://www.khronos.org/registry/OpenCL/specs/3.0-unified/pdf/OpenCL_C.pdf) (PDF). Khronos OpenCL Working Group. Retrieved April 27, 2020.
7. "OpenGL, OpenCL deprecated in favor of Metal 2 in macOS 10.14 Mojave" (https://appleinsider.com/articles/18/06/04/opengl-opencl-deprecated-in-favor-of-metal-2-in-macos-1014-mojave). *AppleInsider*. Retrieved July 3, 2018.
8. "Conformant Companies" (https://www.khronos.org/conformance/adopters/conformant-companies#opencl). Khronos Group. Retrieved April 8, 2015.
9. Gianelli, Silvia E. (January 14, 2015). "Xilinx SDAccel Development Environment for OpenCL, C, and C++, Achieves Khronos Conformance" (http://www.prnewswire.com/news-releases/xilinx-sdaccel-development-environment-for-opencl-c-and-c-achieves-khronos-conformance-300020285.html). *PR Newswire*. Xilinx. Retrieved April 27, 2015.
10. Howes, Lee (November 11, 2015). "The OpenCL Specification Version: 2.1 Document Revision: 23" (https://www.khronos.org/registry/cl/specs/opencl-2.1.pdf) (PDF). Khronos OpenCL Working Group. Retrieved November 16, 2015.
11. Gaster, Benedict; Howes, Lee; Kaeli, David R.; Mistry, Perhaad; Schaa, Dana (2012). *Heterogeneous Computing with OpenCL: Revised OpenCL 1.2 Edition*. Morgan Kaufmann.
12. Tompson, Jonathan; Schlachter, Kristofer (2012). "An Introduction to the OpenCL Programming Model" (https://web.archive.org/web/20150706143727/http://www.cs.nyu.edu/~lerner/spring12/Preso07-OpenCL.pdf) (PDF). New York University Media Research Lab. Archived from the original (http://www.cs.nyu.edu/~lerner/spring12/Preso07-OpenCL.pdf) (PDF) on July 6, 2015. Retrieved July 6, 2015.

13. Stone, John E.; Gohara, David; Shi, Guochin (2010). "OpenCL: a parallel programming standard for heterogeneous computing systems" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2964860). *Computing in Science & Engineering*. **12** (3): 66–73. Bibcode:2010CSE....12c..66S (https://ui.adsabs.harvard.edu/abs/2010CSE....12c..66S). doi:10.1109/MCSE.2010.69 (https://doi.org/10.1109%2FMCSE.2010.69). PMC 2964860 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2964860). PMID 21037981 (https://pubmed.ncbi.nlm.nih.gov/21037981).
14. Klöckner, Andreas; Pinto, Nicolas; Lee, Yunsup; Catanzaro, Bryan; Ivanov, Paul; Fasih, Ahmed (2012). "PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation". *Parallel Computing*. **38** (3): 157–174. arXiv:0911.3456 (https://arxiv.org/abs/0911.3456). doi:10.1016/j.parco.2011.09.001 (https://doi.org/10.1016%2Fj.parco.2011.09.001).
15. "OpenCL - Open Computing Language Bindings" (https://metacpan.org/pod/OpenCL). metacpan.org. Retrieved August 18, 2018.
16. "SPIR - The first open standard intermediate language for parallel compute and graphics" (https://www.khronos.org/spir/). Khronos Group. January 21, 2014.
17. "SYCL - C++ Single-source Heterogeneous Programming for OpenCL" (https://www.khronos.org/sycl/). Khronos Group. January 21, 2014.
18. Aaftab Munshi, ed. (2014). "The OpenCL C Specification, Version 2.0" (https://www.khronos.org/registry/cl/specs/opencl-2.0-openclc.pdf) (PDF). Retrieved June 24, 2014.
19. "Introduction to OpenCL Programming 201005" (https://web.archive.org/web/20110516092008/http://developer.amd.com/zones/OpenCLZone/courses/Documents/Introduction_to_OpenCL_Programming%20(201005).pdf) (PDF). AMD. pp. 89–90. Archived from the original (http://developer.amd.com/zones/OpenCLZone/courses/Documents/Introduction_to_OpenCL_Programming%20(201005).pdf) (PDF) on May 16, 2011. Retrieved August 8, 2017.
20. "OpenCL" (https://www.webcitation.org/66GmScoh5?url=http://s08.idav.ucdavis.edu/munshi-opencl.pdf) (PDF). SIGGRAPH2008. August 14, 2008. Archived from the original (http://s08.idav.ucdavis.edu/munshi-opencl.pdf) (PDF) on March 19, 2012. Retrieved August 14, 2008.
21. "Fitting FFT onto G80 Architecture" (http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project6_report.pdf) (PDF). Vasily Volkov and Brian Kazian, UC Berkeley CS258 project report. May 2008. Retrieved November 14, 2008.
22. "OpenCL on FFT" (https://developer.apple.com/mac/library/samplecode/OpenCL_FFT/index.html). Apple. November 16, 2009. Retrieved December 7, 2009.
23. "Khronos Launches Heterogeneous Computing Initiative" (https://web.archive.org/web/20080620123431/http://www.khronos.org/news/press/releases/khronos_launches_heterogeneous_computing_initiative/) (Press release). Khronos Group. June 16, 2008. Archived from the original (https://www.khronos.org/news/press/releases/khronos_launches_heterogeneous_computing_initiative/) on June 20, 2008. Retrieved June 18, 2008.
24. "OpenCL gets touted in Texas" (http://www.macworld.com/article/136921/2008/11/opencl.html?lsrc=top_2). MacWorld. November 20, 2008. Retrieved June 12, 2009.
25. "The Khronos Group Releases OpenCL 1.0 Specification" (https://www.khronos.org/news/press/the_khronos_group_releases_opencl_1.0_specification) (Press release). Khronos Group. December 8, 2008. Retrieved December 4, 2016.
26. "Apple Previews Mac OS X Snow Leopard to Developers" (https://www.webcitation.org/66GmLMR6B?url=http://www.apple.com/pr/library/2008/06/09Apple-Previews-Mac-OS-X-Snow-Leopard-to-Developers.html) (Press release). Apple Inc. June 9, 2008. Archived from the original (https://www.apple.com/pr/library/2008/06/09snowleopard.html) on March 19, 2012. Retrieved June 9, 2008.
27. "AMD Drives Adoption of Industry Standards in GPGPU Software Development" (https://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51_104_543~127451,00.html) (Press release). AMD. August 6, 2008. Retrieved August 14, 2008.

28. "AMD Backs OpenCL, Microsoft DirectX 11" (http://www.eweek.com/c/a/Desktops-and-Notebooks/AMD-Backing-OpenCL-and-Microsoft-DirectX-11/). eWeek. August 6, 2008. Retrieved August 14, 2008.
29. "HPCWire: RapidMind Embraces Open Source and Standards Projects" (https://web.archive.org/web/20081218113648/http://www.hpcwire.com/topic/applications/RapidMind_Embraces_Open_Source_and_Standards_Projects.html). HPCWire. November 10, 2008. Archived from the original (http://www.hpcwire.com/topic/applications/RapidMind_Embraces_Open_Source_and_Standards_Projects.html) on December 18, 2008. Retrieved November 11, 2008.
30. "Nvidia Adds OpenCL To Its Industry Leading GPU Computing Toolkit" (http://www.nvidia.com/object/io_1228825271885.html) (Press release). Nvidia. December 9, 2008. Retrieved December 10, 2008.
31. "OpenCL Development Kit for Linux on Power" (http://www.alphaworks.ibm.com/tech/opencl). alphaWorks. October 30, 2009. Retrieved October 30, 2009.
32. "Khronos Drives Momentum of Parallel Computing Standard with Release of OpenCL 1.1 Specification" (https://web.archive.org/web/20160302231506/https://www.khronos.org/news/press/releases/khronos-group-releases-opencl-1-1-parallel-computing-standard/). Archived from the original (https://www.khronos.org/news/press/releases/khronos-group-releases-opencl-1-1-parallel-computing-standard/) on March 2, 2016. Retrieved February 24, 2016.
33. "Khronos Releases OpenCL 1.2 Specification" (https://www.khronos.org/news/press/khronos-releases-opencl-1.2-specification). Khronos Group. November 15, 2011. Retrieved June 23, 2015.
34. "OpenCL 1.2 Specification" (https://www.khronos.org/registry/cl/specs/opencl-1.2.pdf) (PDF). Khronos Group. Retrieved June 23, 2015.
35. "Khronos Finalizes OpenCL 2.0 Specification for Heterogeneous Computing" (https://www.khronos.org/news/press/khronos-finalizes-opencl-2.0-specification-for-heterogeneous-computing). Khronos Group. November 18, 2013. Retrieved February 10, 2014.
36. "Khronos Releases OpenCL 2.1 and SPIR-V 1.0 Specifications for Heterogeneous Parallel Programming" (https://www.khronos.org/news/press/khronos-releases-opencl-2.1-and-spir-v-1.0-specifications-for-heterogeneous). Khronos Group. November 16, 2015. Retrieved November 16, 2015.
37. "Khronos Announces OpenCL 2.1: C++ Comes to OpenCL" (http://www.anandtech.com/show/9039/khronos-announces-opencl-21-c-comes-to-opencl). AnandTech. March 3, 2015. Retrieved April 8, 2015.
38. "Khronos Releases OpenCL 2.1 Provisional Specification for Public Review" (https://www.khronos.org/news/press/khronos-releases-opencl-2.1-provisional-specification-for-public-review). Kronos Group. March 3, 2015. Retrieved April 8, 2015.
39. "OpenCL Overview" (https://www.khronos.org/opencl/). Khronos Group. July 21, 2013.
40. "Khronos Releases OpenCL 2.2 Provisional Specification with OpenCL C++ Kernel Language for Parallel Programming" (https://www.khronos.org/news/press/khronos-releases-opencl-2.2-provisional-spec-opencl-c-kernel-language). Khronos Group. April 18, 2016.
41. Trevett, Neil (April 2016). "OpenCL – A State of the Union" (http://www.iwocl.org/wp-content/uploads/iwocl-2016-opencl-state-union.pdf) (PDF). *IWOCL*. Vienna: Khronos Group. Retrieved January 2, 2017.
42. "Khronos Releases OpenCL 2.2 With SPIR-V 1.2" (https://www.khronos.org/news/press/khronos-releases-opencl-2.2-with-spir-v-1.2). Khronos Group. May 16, 2017.
43. "OpenCL 2.2 Maintenance Update Released" (https://www.khronos.org/blog/opencl-2.2-maintenance-update-released). *The Khronos Group*. May 14, 2018.
44. https://www.phoronix.com/scan.php?page=article&item=opencl-30-spec&num=1
45. https://www.khronos.org/news/press/khronos-group-releases-opencl-3.0

46. "Breaking: OpenCL Merging Roadmap into Vulkan | PC Perspective" (https://web.archive.org/web/20171101062642/https://www.pcper.com/reviews/General-Tech/Breaking-OpenCL-Merging-Roadmap-Vulkan). *www.pcper.com*. Archived from the original (https://www.pcper.com/reviews/General-Tech/Breaking-OpenCL-Merging-Roadmap-Vulkan) on November 1, 2017. Retrieved May 17, 2017.
47. "SIGGRAPH 2018: OpenCL-Next Taking Shape, Vulkan Continues Evolving - Phoronix" (https://www.phoronix.com/scan.php?page=article&item=siggraph-2018-khr&num=2). *www.phoronix.com*.
48. *Clspv is a prototype compiler for a subset of OpenCL C to Vulkan compute shaders: google/clspv* (https://github.com/google/clspv), August 17, 2019, retrieved August 20, 2019
49. "Vulkan Update SIGGRAPH 2019" (https://www.khronos.org/assets/uploads/developers/library/2019-siggraph/Vulkan-01-Update-SIGGRAPH-Jul19.pdf) (PDF).
50. Trevett, Neil (May 23, 2019). "Khronos and OpenCL Overview EVS Workshop May19" (https://www.khronos.org/assets/uploads/developers/library/2019-embedded-vision-summit/1b%20Khronos-and-OpenCL-Overview-EVS-Workshop_May19.pdf) (PDF). *Khronos Group*.
51. "OpenCL ICD Specification" (https://www.khronos.org/registry/cl/extensions/khr/cl_khr_icd.txt). Retrieved June 23, 2015.
52. "Apple entry on LLVM Users page" (http://llvm.org/Users.html#Apple). Retrieved August 29, 2009.
53. "Nvidia entry on LLVM Users page" (http://llvm.org/Users.html). Retrieved August 6, 2009.
54. "Rapidmind entry on LLVM Users page" (http://llvm.org/Users.html). Retrieved October 1, 2009.
55. "Zack Rusin's blog post about the Gallium3D OpenCL implementation" (http://zrusin.blogspot.com/2009/02/opencl.html). February 2009. Retrieved October 1, 2009.
56. "GalliumCompute" (http://dri.freedesktop.org/wiki/GalliumCompute/). dri.freedesktop.org. Retrieved June 23, 2015.
57. "Clover Status Update" (https://www.x.org/wiki/Events/XDC2013/XDC2013TomStellardCloverStatus/XDC2013TomStellardCloverStatus.pdf) (PDF).
58. "mesa/mesa - The Mesa 3D Graphics Library" (https://cgit.freedesktop.org/mesa/mesa/log/?qt=grep&q=clover). *cgit.freedesktop.org*.
59. "Gallium Clover With SPIR-V & NIR Opening Up New Compute Options Inside Mesa - Phoronix" (https://www.phoronix.com/scan.php?page=news_item&px=Gallium-Clover-NIR-SPIR-V-XDC18). *www.phoronix.com*.
60. https://xdc2018.x.org/slides/clover.pdf
61. Larabel, Michael (January 10, 2013). "Beignet: OpenCL/GPGPU Comes For Ivy Bridge On Linux" (https://www.phoronix.com/scan.php?page=news_item&px=MTI3MTU). *Phoronix*.
62. Larabel, Michael (April 16, 2013). "More Criticism Comes Towards Intel's Beignet OpenCL" (https://www.phoronix.com/scan.php?page=news_item&px=MTM1MzM). *Phoronix*.
63. Larabel, Michael (December 24, 2013). "Intel's Beignet OpenCL Is Still Slowly Baking" (https://www.phoronix.com/scan.php?page=news_item&px=MTU1MjA). *Phoronix*.
64. "Beignet" (https://freedesktop.org/wiki/Software/Beignet/). freedesktop.org.
65. "beignet - Beignet OpenCL Library for Intel Ivy Bridge and newer GPUs" (https://cgit.freedesktop.org/beignet/). *cgit.freedesktop.org*.
66. "Intel Brings Beignet To Android For OpenCL Compute - Phoronix" (https://www.phoronix.com/scan.php?page=news_item&px=Intel-Beignet-Android). *www.phoronix.com*.
67. "01.org Intel Open Source - Compute Runtime" (https://01.org/compute-runtime). February 7, 2018.
68. "NEO GitHub README" (https://github.com/intel/compute-runtime/blob/master/README.md). March 21, 2019.

69. "ROCm" (https://web.archive.org/web/20161008220038/https://radeonopencompute.github.io/). *GitHub*. Archived from the original (https://radeonopencompute.github.io/) on October 8, 2016.
70. "RadeonOpenCompute/ROCm: ROCm - Open Source Platform for HPC and Ultrascale GPU Computing" (https://github.com/RadeonOpenCompute/ROCm). GitHub. March 21, 2019.
71. "A Nice Overview Of The ROCm Linux Compute Stack - Phoronix" (https://www.phoronix.com/scan.php?page=news_item&px=ROCm-Compute-Stack-Overview). *www.phoronix.com*.
72. "XDC Lightning.pdf" (https://drive.google.com/file/d/1ePlNzxYryveh6iFL-cqJO7ycSsYJ4LbC/view?usp=embed_facebook). *Google Docs*.
73. "Radeon ROCm 2.0 Officially Out With OpenCL 2.0 Support, TensorFlow 1.12, Vega 48-bit VA - Phoronix" (https://www.phoronix.com/scan.php?page=news_item&px=Radeon-ROCm-2.0-Arrives). *www.phoronix.com*.
74. "Taking Radeon ROCm 2.0 OpenCL For A Benchmarking Test Drive - Phoronix" (https://www.phoronix.com/scan.php?page=article&item=radeon-rocm-20&num=1). *www.phoronix.com*.
75. https://github.com/RadeonOpenCompute/ROCm/blob/master/AMD_ROCm_Release_Note
76. https://rocm-documentation.readthedocs.io/en/latest/
77. Jääskeläinen, Pekka; Sánchez de La Lama, Carlos; Schnetter, Erik; Raiskila, Kalle; Takala, Jarmo; Berg, Heikki (2016). "pocl: A Performance-Portable OpenCL Implementation". *Int'l J. Parallel Programming*. **43** (5): 752–785. arXiv:1611.07083 (https://arxiv.org/abs/1611.07083). Bibcode:2016arXiv161107083J (https://ui.adsabs.harvard.edu/abs/2016arXiv161107083J). doi:10.1007/s10766-014-0320-y (https://doi.org/10.1007%2Fs10766-014-0320-y).
78. "pocl home page" (http://portablecl.org/). *pocl*.
79. "GitHub - pocl/pocl: pocl: Portable Computing Language" (https://github.com/pocl/pocl). March 14, 2019 – via GitHub.
80. "HSA support implementation status as of 2016-05-17 — Portable Computing Language (pocl) 1.3-pre documentation" (http://portablecl.org/docs/html/hsa_status.html#opencl-2-0-atomics-and-hsa-memory-scope). *portablecl.org*.
81. http://portablecl.org/pocl-1.3.html
82. http://portablecl.org/pocl-1.4.html
83. http://portablecl.org/pocl-1.5.html
84. "About" (https://git.linaro.org/gpgpu/shamrock.git/about/). *Git.Linaro.org*.
85. Gall, T.; Pitney, G. (March 6, 2014). "LCA14-412: GPGPU on ARM SoC" (https://s3.amazonaws.com/connect.linaro.org/lca14/presentations/LCA14-412-%20GPGPU%20on%20ARM%20SoC%20session.pdf) (PDF). *Amazon Web Services*. Retrieved January 22, 2017.
86. "zuzuf/freeocl" (https://github.com/zuzuf/freeocl). *GitHub*. Retrieved April 13, 2017.
87. Zhang, Peng; Fang, Jianbin; Yang, Canqun; Tang, Tao; Huang, Chun; Wang, Zheng (2018). *MOCL: An Efficient OpenCL Implementation for the Matrix-2000 Architecture* (https://jianbinfang.github.io/files/2018-03-15-mocl.pdf) (PDF). Proc. Int'l Conf. on Computing Frontiers. doi:10.1145/3203217.3203244 (https://doi.org/10.1145%2F3203217.3203244).
88. "OpenCL Demo, AMD CPU" (https://www.youtube.com/watch?v=sLv_fhQlqis). December 10, 2008. Retrieved March 28, 2009.
89. "OpenCL Demo, Nvidia GPU" (https://www.youtube.com/watch?v=PJ1jydg8mLg). December 10, 2008. Retrieved March 28, 2009.
90. "Imagination Technologies launches advanced, highly-efficient POWERVR SGX543MP multi-processor graphics IP family" (http://www.imgtec.com/News/Release/index.asp?NewsID=449). Imagination Technologies. March 19, 2009. Retrieved January 30, 2011.
91. "AMD and Havok demo OpenCL accelerated physics" (https://web.archive.org/web/20090405072046/http://www.pcper.com/comments.php?nid=6954). PC Perspective. March 26, 2009. Archived from the original (http://www.pcper.com/comments.php?nid=6954) on April 5, 2009. Retrieved March 28, 2009.

92. "Nvidia Releases OpenCL Driver To Developers" (https://www.webcitation.org/66GmM7jsd?url=http://www.nvidia.com/object/io_1240224603372.html). Nvidia. April 20, 2009. Archived from the original (http://www.nvidia.com/object/io_1240224603372.html) on March 19, 2012. Retrieved April 27, 2009.
93. "AMD does reverse GPGPU, announces OpenCL SDK for x86" (http://arst.ch/5te). Ars Technica. August 5, 2009. Retrieved August 6, 2009.
94. Moren, Dan; Snell, Jason (June 8, 2009). "Live Update: WWDC 2009 Keynote" (http://www.macworld.com/article/140897/2009/06/keynote.html). *MacWorld.com*. MacWorld. Retrieved June 12, 2009.
95. "ATI Stream Software Development Kit (SDK) v2.0 Beta Program" (https://web.archive.org/web/20090809065559/http://developer.amd.com/GPU/ATISTREAMSDKBETAPROGRAM/Pages/default.aspx). Archived from the original (http://developer.amd.com/GPU/ATISTREAMSDKBETAPROGRAM/Pages/default.aspx#one) on August 9, 2009. Retrieved October 14, 2009.
96. "S3 Graphics launched the Chrome 5400E embedded graphics processor" (https://web.archive.org/web/20091202065250/http://www.s3graphics.com/en/news/news_detail.aspx?id=44). Archived from the original (http://www.s3graphics.com/en/news/news_detail.aspx?id=44) on December 2, 2009. Retrieved October 27, 2009.
97. "VIA Brings Enhanced VN1000 Graphics Processor]" (https://web.archive.org/web/20091215090119/http://www.via.com.tw/en/resources/pressroom/pressrelease.jsp?press_release_no=4327). Archived from the original (http://www.via.com.tw/en/resources/pressroom/pressrelease.jsp?press_release_no=4327) on December 15, 2009. Retrieved December 10, 2009.
98. "ATI Stream SDK v2.0 with OpenCL 1.0 Support" (https://web.archive.org/web/20091101061303/http://developer.amd.com/gpu/atistreamsdk/pages/default.aspx). Archived from the original (http://developer.amd.com/gpu/ATIStreamSDK/Pages/default.aspx) on November 1, 2009. Retrieved October 23, 2009.
99. "OpenCL" (http://www.ziilabs.com/opencl). ZiiLABS. Retrieved June 23, 2015.
00. "Intel discloses new Sandy Bridge technical details" (http://news.cnet.com/8301-13924_3-20016302-64.html). Retrieved September 13, 2010.
01. "WebCL related stories" (https://www.khronos.org/news/categories/C251). Khronos Group. Retrieved June 23, 2015.
02. "Khronos Releases Final WebGL 1.0 Specification" (https://web.archive.org/web/20150709134803/https://www.khronos.org/news/press/releases/khronos-releases-final-webgl-1.0-specification). Khronos Group. Archived from the original (https://www.khronos.org/news/press/releases/khronos-releases-final-webgl-1.0-specification) on July 9, 2015. Retrieved June 23, 2015.
03. "Community" (https://developer.ibm.com/community/).
04. "Welcome to Wikis" (https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/Wbf059a58a9b9_459d_aca4_493655c96370/page/OpenCL+Common+Runtime). *www.ibm.com*. October 20, 2009.
05. "Nokia Research releases WebCL prototype" (https://www.khronos.org/news/permalink/nokia-research-releases-webcl-prototype). Khronos Group. May 4, 2011. Retrieved June 23, 2015.
06. KamathK, Sharath. "Samsung's WebCL Prototype for WebKit" (https://web.archive.org/web/20150218105743/https://github.com/SRA-SiliconValley/webkit-webcl). Github.com. Archived from the original (https://github.com/SRA-SiliconValley/webkit-webcl) on February 18, 2015. Retrieved June 23, 2015.
07. "AMD Opens the Throttle on APU Performance with Updated OpenCL Software Development" (https://www.amd.com/us/press-releases/Pages/app-sdk-2011aug08.aspx). Amd.com. August 8, 2011. Retrieved June 16, 2013.
08. "AMD APP SDK v2.6" (http://forums.amd.com/forum/messageview.cfm?catid=390&threadid=157108). Forums.amd.com. March 13, 2015. Retrieved June 23, 2015.

9. "The Portland Group Announces OpenCL Compiler for ST-Ericsson ARM-Based NovaThor SoCs" (http://www.anandtech.com/show/5607/the-portland-group-announces-opencl-compiler-for-stericsson-armbased-novathor-socs). Retrieved May 4, 2012.
10. "WebCL Latest Spec" (https://web.archive.org/web/20140801133503/https://cvs.khronos.org/svn/repos/registry/trunk/public/webcl/spec/latest/index.html). Khronos Group. November 7, 2013. Archived from the original (https://cvs.khronos.org/svn/repos/registry/trunk/public/webcl/spec/latest/index.html) on August 1, 2014. Retrieved June 23, 2015.
11. "Altera Opens the World of FPGAs to Software Programmers with Broad Availability of SDK and Off-the-Shelf Boards for OpenCL" (https://web.archive.org/web/20140109220211/http://newsroom.altera.com/press-releases/altera-opens-the-world-of-fpgas-to-software-programmers-with-broad-availability-of-sdk-and-off-the-shelf-boards-for-opencl.htm). Altera.com. Archived from the original (http://newsroom.altera.com/press-releases/altera-opens-the-world-of-fpgas-to-software-programmers-with-broad-availability-of-sdk-and-off-the-shelf-boards-for-opencl.htm) on January 9, 2014. Retrieved January 9, 2014.
12. "Altera SDK for OpenCL is First in Industry to Achieve Khronos Conformance for FPGAs" (https://web.archive.org/web/20140109083533/http://newsroom.altera.com/press-releases/nr-altera-sdk-opencl-conformance.htm). Altera.com. Archived from the original (http://newsroom.altera.com/press-releases/nr-altera-sdk-opencl-conformance.htm) on January 9, 2014. Retrieved January 9, 2014.
13. "Khronos Finalizes OpenCL 2.0 Specification for Heterogeneous Computing" (https://www.khronos.org/news/press/khronos-finalizes-opencl-2.0-specification-for-heterogeneous-computing). Khronos Group. November 18, 2013. Retrieved June 23, 2015.
14. "WebCL 1.0 Press Release" (https://www.khronos.org/news/press/khronos-releases-webcl-1.0-specification). Khronos Group. March 19, 2014. Retrieved June 23, 2015.
15. "WebCL 1.0 Specification" (https://www.khronos.org/registry/webcl/specs/1.0.0/). Khronos Group. March 14, 2014. Retrieved June 23, 2015.
16. "Intel OpenCL 2.0 Driver" (https://archive.is/20140917185715/https://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&DwnldID=24245). Archived from the original (https://downloadcenter.intel.com/Detail_Desc.aspx?agr=Y&DwnldID=24245) on September 17, 2014. Retrieved October 14, 2014.
17. "AMD OpenCL 2.0 Driver" (http://support.amd.com/en-us/kb-articles/Pages/OpenCL2-Driver.aspx). *Support.AMD.com*. June 17, 2015. Retrieved June 23, 2015.
18. "Xilinx SDAccel development environment for OpenCL, C, and C++, achieves Khronos Conformance - khronos.org news" (https://www.khronos.org/news/permalink/xilinx-sdaccel-development-environment-for-opencl-c-and-c-achieves-khronos). *The Khronos Group*. Retrieved June 26, 2017.
19. "Release 349 Graphics Drivers for Windows, Version 350.12" (https://a248.e.akamai.net/f/248/10/10/us.download.nvidia.com/Windows/350.12/350.12-win8-win7-winvista-desktop-release-notes.pdf) (PDF). April 13, 2015. Retrieved February 4, 2016.
20. "AMD APP SDK 3.0 Released" (http://developer.amd.com/community/blog/2015/08/26/introducing-app-sdk-30-opencl-2/). *Developer.AMD.com*. August 26, 2015. Retrieved September 11, 2015.
21. "Khronos Releases OpenCL 2.1 and SPIR-V 1.0 Specifications for Heterogeneous Parallel Programming" (https://www.khronos.org/news/press/khronos-releases-opencl-2.1-and-spir-v-1.0-specifications-for-heterogeneous). Khronos Group. November 16, 2015.
22. "What's new? Intel® SDK for OpenCL™ Applications 2016, R3" (https://software.intel.com/en-us/whats-new-code-builder-2016-r3). Intel Software.
23. "NVIDIA 378.66 drivers for Windows offer OpenCL 2.0 evaluation support" (https://www.khronos.org/news/permalink/nvidia-378.66-drivers-for-windows-offer-opencl-2.0-evaluation-support). Khronos Group. February 17, 2017.
24. Szuppe, Jakub (February 22, 2017). "NVIDIA enables OpenCL 2.0 beta-support" (https://streamhpc.com/blog/2017-02-22/nvidia-enables-opencl-2-0-beta-support/).
25. Szuppe, Jakub (March 6, 2017). "NVIDIA beta-support for OpenCL 2.0 works on Linux too" (https://streamhpc.com/blog/2017-03-06/nvidia-beta-support-opencl-2-0-linux/).

26. "The Khronos Group" (https://www.khronos.org/news/permalink/khronos-releases-opencl-2.2-with-spir-v-1.2). *The Khronos Group*. March 21, 2019.
27. "The Khronos Group" (https://www.khronos.org/adopters). *The Khronos Group*. August 20, 2019. Retrieved August 20, 2019.
28. "KhronosGroup/OpenCL-CTL: The OpenCL Conformance Tests" (https://github.com/KhronosGroup/OpenCL-CTS). GitHub. March 21, 2019.
29. "OpenCL and the AMD APP SDK" (https://web.archive.org/web/20110804010819/http://developer.amd.com/documentation/articles/pages/OpenCL-and-the-AMD-APP-SDK.aspx). *AMD Developer Central*. developer.amd.com. Archived from the original (http://developer.amd.com/documentation/articles/pages/OpenCL-and-the-AMD-APP-SDK.aspx) on August 4, 2011. Retrieved August 11, 2011.
30. "About Intel OpenCL SDK 1.1" (http://software.intel.com/en-us/articles/opencl-sdk/). *software.intel.com*. intel.com. Retrieved August 11, 2011.
31. "Intel® SDK for OpenCL™ Applications - Release Notes" (https://software.intel.com/en-us/articles/intel-sdk-for-opencl-applications-release-notes). *software.intel.com*. March 14, 2019.
32. "Product Support" (http://software.intel.com/en-us/articles/opencl-sdk-frequently-asked-questions/#12). Retrieved August 11, 2011.
33. "Intel OpenCL SDK – Release Notes" (https://web.archive.org/web/20110717054302/http://software.intel.com/en-us/articles/opencl-release-notes/). Archived from the original (http://software.intel.com/en-us/articles/opencl-release-notes/) on July 17, 2011. Retrieved August 11, 2011.
34. "Announcing OpenCL Development Kit for Linux on Power v0.3" (http://www.ibm.com/developerworks/forums/thread.jspa?messageID=14600651&tstart=0). Retrieved August 11, 2011.
35. "IBM releases OpenCL Development Kit for Linux on Power v0.3 – OpenCL 1.1 conformant release available" (https://www.ibm.com/developerworks/mydeveloperworks/blogs/80367538-d04a-47cb-9463-428643140bf1/entry/ibm_releases_opencl_development_kit_for_linux_on_power_v0_3_opencl_1_1_conformant_release_available6?lang=en). *OpenCL Lounge*. ibm.com. Retrieved August 11, 2011.
36. "IBM releases OpenCL Common Runtime for Linux on x86 Architecture" (https://www.ibm.com/developerworks/mydeveloperworks/blogs/80367538-d04a-47cb-9463-428643140bf1/entry/ibm_releases_opencl_common_runtime_for_linux_on_x86_architecture4?lang=en). October 20, 2009. Retrieved September 10, 2011.
37. "OpenCL and the AMD APP SDK" (https://web.archive.org/web/20110906045531/http://developer.amd.com/documentation/articles/pages/OpenCL-and-the-AMD-APP-SDK.aspx). *AMD Developer Central*. developer.amd.com. Archived from the original (http://developer.amd.com/documentation/articles/pages/OpenCL-and-the-AMD-APP-SDK.aspx) on September 6, 2011. Retrieved September 10, 2011.
38. "Nvidia Releases OpenCL Driver" (http://www.tomshardware.com/news/Nvidia-Cuda-OpenCL-SDK,7596.html). April 22, 2009. Retrieved August 11, 2011.
39. "clinfo by Simon Leblanc" (https://github.com/simleb/clinfo). Retrieved January 27, 2017.
40. "clinfo by Oblomov" (https://github.com/Oblomov/clinfo). Retrieved January 27, 2017.
41. "clinfo: openCL INFOrmation" (https://sourceforge.net/projects/clinfo/). Retrieved January 27, 2017.
42. "Khronos Products" (https://www.khronos.org/conformance/adopters/conformant-products#opencl). *The Khronos Group*. Retrieved May 15, 2017.
43. https://github.com/KhronosGroup/OpenCL-CTS/tree/master/test_conformance
44. "compute-runtime" (https://01.org/compute-runtime). *01.org*. February 7, 2018.
45. Fang, Jianbin; Varbanescu, Ana Lucia; Sips, Henk (2011). *A Comprehensive Performance Comparison of CUDA and OpenCL*. Proc. Int'l Conf. on Parallel Processing. doi:10.1109/ICPP.2011.45 (https://doi.org/10.1109%2FICPP.2011.45).

46. Du, Peng; Weber, Rick; Luszczek, Piotr; Tomov, Stanimire; Peterson, Gregory; Dongarra, Jack (2012). "From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming". *Parallel Computing*. **38** (8): 391–407. CiteSeerX 10.1.1.193.7712 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.193.7712). doi:10.1016/j.parco.2011.10.002 (https://doi.org/10.1016%2Fj.parco.2011.10.002).
47. Dolbeau, Romain; Bodin, François; de Verdière, Guillaume Colin (September 7, 2013). "One OpenCL to rule them all?". *2013 IEEE 6th International Workshop on Multi-/Many-core Computing Systems (MuCoCoS)*. pp. 1–6. doi:10.1109/MuCoCoS.2013.6633603 (https://doi.org/10.1109%2FMuCoCoS.2013.6633603). ISBN 978-1-4799-1010-6.
48. Karimi, Kamran; Dickson, Neil G.; Hamze, Firas (2011). "A Performance Comparison of CUDA and OpenCL". arXiv:1005.2581v3 (https://arxiv.org/abs/1005.2581v3) [cs.PF (https://arxiv.org/archive/cs.PF)].
49. A Survey of CPU-GPU Heterogeneous Computing Techniques, ACM Computing Surveys, 2015.
50. Grewe, Dominik; O'Boyle, Michael F. P. (2011). *A Static Task Partitioning Approach for Heterogeneous Systems Using OpenCL*. Proc. Int'l Conf. on Compiler Construction. doi:10.1007/978-3-642-19861-8_16 (https://doi.org/10.1007%2F978-3-642-19861-8_16).
51. "Coriander Project: Compile CUDA Codes To OpenCL, Run Everywhere" (http://www.phoronix.com/scan.php?page=news_item&px=CUDA-On-CL-Coriander). Phoronix.
52. Perkins, Hugh (2017). "cuda-on-cl" (http://www.iwocl.org/wp-content/uploads/iwocl2017-hugh-perkins-cuda-cl.pdf) (PDF). IWOCL. Retrieved August 8, 2017.
53. "hughperkins/coriander: Build NVIDIA® CUDA™ code for OpenCL™ 1.2 devices" (https://github.com/hughperkins/coriander). GitHub. May 6, 2019.

External links

- Official website (https://www.khronos.org/opencl/)
- Official website (https://www.khronos.org/webcl/) for WebCL
- International Workshop on OpenCL (https://www.iwocl.org/) (IWOCL) sponsored by The Khronos Group