OpenCL
State of the Nation
Neil Trevett | Khronos President
NVIDIA Vice President Developer Ecosystem
OpenCL Working Group Chair
ntrevett@nvidia.com | @neilt3d
Toronto, May 2017
Topics

1. The Good
   The amazing progress of OpenCL

2. The Bad
   Lessons Learned from the first eight years

3. The Exciting
   Where do we go from here?
OpenCL 2.2 Finalized Here at IWOCL!

- **2011**: OpenCL 1.2
  - Becomes industry baseline

- **2013**: OpenCL 2.0
  - Enables new class of hardware
  - SVM Generic Addresses
  - On-device dispatch

- **2015**: OpenCL 2.1
  - SPIR-V 1.0
  - SPIR-V in Core
  - Kernel Language Flexibility

- **2017**: OpenCL 2.2
  - SPIR-V 1.2
  - OpenCL C++ Kernel Language
  - Static subset of C++14
  - Templates and Lambdas
  - SPIR-V 1.2
  - OpenCL C++ support
  - Pipes
    - Efficient device-scope communication between kernels
  - Code Generation Optimizations
    - Specialization constants at SPIR-V compilation time
    - Constructors and destructors of program scope global objects
    - User callbacks can be set at program release time

[Link to official website](https://www.khronos.org/opencl/)
New Open Source Engagement Model

• Khronos is open sourcing specification sources, conformance tests, tools
  - Merge requests welcome from the community (subject to review by OpenCL working group)

• Deeper Community Enablement
  - Mix your own documentation!
  - Contribute and fix conformance tests
  - Fix the specification, headers, ICD etc.
  - Contribute new features (carefully)

Source Materials for Specifications and Reference Documentation CONTRIBUTED Under Khronos IP Framework (you won’t assert patents against conformant implementations, and license copyright for Khronos use)

Spec Build System and Scripts

Spec and Ref Language Source

Redistribution under CC-BY 4.0

Contributions and Distribution under Apache 2.0

Khrnos builds and Ratifies Canonical Specification under Khronos IP Framework. No changes or re-hosting allowed

Spec and Ref Language Source and derivative materials. Re-mixable under CC-BY by the industry and community

Community built documentation and tools

Khrnos Adopters Program

Conformance Test Suite Source

Contributions and Distribution under Apache 2.0

Anyone can test any implementation at any time

Conformant Implementations can use trademark and are covered by Khronos IP Framework
Shout Out to University of Windsor

The Windsor Testing Framework, also released today, enables developers to quickly install and configure the OpenCL Conformance Test Suite on their own systems.
SPIR-V Ecosystem

Khronos has open sourced these tools and translators

https://github.com/KhronosGroup/SPIRV-Tools

SPIR-V
- Khronos defined and controlled cross-API intermediate language
- Native support for graphics and parallel constructs
  - 32-bit Word Stream
  - Extensible and easily parsed
  - Retains data object and control flow information for effective code generation and translation

Third party kernel and shader languages

glslang

MSL
HLSL
GLSL

SPIR-V Cross

Other Intermediate Forms

SPIR-V (Dis)Assembler

SPIR-V Validator

IHV Driver Runtimes

OpenCL C
Front-end

OpenCL C++
Front-end

LLVM

LLVM to SPIR-V Bi-directional Translator

SPIR-V (Dis)Assembler

Other Intermediate Forms

IHV Driver Runtimes

Khronos coordinating liaison with Clang/LLVM Community
E.g. discussing SPIR-V as supported Clang target
SYCL Ecosystem

• Single-source heterogeneous programming using STANDARD C++
  - Use C++ templates and lambda functions for host & device code
  - Layered over OpenCL

• Fast and powerful path for bring C++ apps and libraries to OpenCL
  - C++ Kernel Fusion - better performance on complex software than hand-coding
  - Halide, Eigen, Boost.Compute, SYCLBLAS, SYCL Eigen, SYCL TensorFlow, SYCL GTX
  - triSYCL, ComputeCpp, VisionCpp, ComputeCpp SDK …

• More information at http://sycl.tech
OpenCL Adoption

- 100s of applications using OpenCL acceleration
  - Rendering, visualization, video editing, simulation, image processing
- Over 4,000 GitHub repositories using OpenCL
  - Tools, applications, libraries, languages
  - Up form 2,000 two years ago
- Multiple silicon and open source implementations
  - Increasingly used for embedded vision and neural network inferencing
- Khronos Resource Hub
  https://www.khronos.org/opencl/resources/opencl-applications-using-opencl
OpenCL as Language/Library Backend

- **Caffe**: C++ based Neural network framework
- **Halide**: Language for image processing and computational photography
- **C++ AMP**: Accelerated Massive Parallelism with Microsoft Visual C++
- **SYCL**: MulticoreWare open source project on Bitbucket
- **aparapi**: Single Source C++ Programming for OpenCL
- **Java language extensions for parallelism**
- **Vision processing open source project**
- **Compiler directives for Fortran, C and C++**
- **Open source software library for machine learning**

**Hundreds of languages, frameworks and projects using OpenCL to access vendor-optimized, heterogeneous compute runtimes**
Safety Critical APIs

New Generation APIs for safety certifiable vision, graphics and compute e.g. ISO 26262 and DO-178B/C

OpenGL SC 1.0 - 2005
Fixed function graphics subset

OpenGL SC 2.0 - April 2016
Shader programmable pipeline subset

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Shader programmable pipeline

OpenVX SC 1.1 Released 1st May 2017
Restricted “deployment” implementation executes on the target hardware by reading the binary format and executing the pre-compiled graphs

Khronos SCAP
‘Safety Critical Advisory Panel’ Guidelines for designing APIs that ease system certification. Open to Khronos member AND industry experts

OpenCL SC TSG Formed
Working on OpenCL SC 1.2 Eliminate Undefined Behavior Eliminate Callback Functions Static Pool of Event Objects

Experience and Guidelines

OpenCL SC TSG Formed
Working on OpenCL SC 1.2 Eliminate Undefined Behavior Eliminate Callback Functions Static Pool of Event Objects
OpenCL Conformant Implementations

Vendor timelines are first conformant submission for each spec generation

- **Dec08**: OpenCL 1.0 Specification
- **Jun10**: OpenCL 1.1 Specification
- **Nov11**: OpenCL 1.2 Specification
- **Nov13**: OpenCL 2.0 Specification
- **Nov15**: OpenCL 2.1 Specification

Desktop
- 1.0 | May09
- 1.1 | May10
- 1.2 | Jun10
- 2.0 | Jul14
- 2.0 | Sep14

Mobile
- 1.0 | May09
- 1.1 | Jun10
- 1.2 | May12
- 1.2 | Jun12
- 2.0 | Jul14
- 2.0 | Apr17

Embedded
- 1.0 | Jan10
- 1.1 | Feb11
- 1.2 | Sep13
- 1.2 | Apr14
- 1.2 | Dec14
- 1.2 | May15

FPGA
- 1.1 | Aug12
- 1.1 | Nov12
- 1.1 | May13
- 1.2 | May15
- 1.2 | Aug15
- 1.0 | Jul13
OpenCL - 1000s Man Years Effort

**Single Source C++ Programming**
Full support for features in C++14-based Kernel Language

**API and Language Specs**
Brings C++14-based Kernel Language into core specification

**Portable Kernel Intermediate Language**
Support for C++14-based kernel language e.g. constructors/destructors

---

<table>
<thead>
<tr>
<th>Dec08</th>
<th>Jun10</th>
<th>Nov11</th>
<th>Nov13</th>
<th>Nov15</th>
<th>May17</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCL 1.0 Specification</td>
<td>OpenCL 1.1 Specification</td>
<td>OpenCL 1.2 Specification</td>
<td>OpenCL 2.0 Specification</td>
<td>OpenCL 2.1 Specification</td>
<td>OpenCL 2.2 Specification</td>
</tr>
<tr>
<td>18 months</td>
<td>18 months</td>
<td>24 months</td>
<td>24 months</td>
<td>18 months</td>
<td></td>
</tr>
</tbody>
</table>

- **OpenCL C++ Kernel Language**
  - Static subset of C++14
  - Templates and Lambdas
  - SPIR-V 1.2 with C++ support
  - SYCL 2.2 single source C++
  - Pipes
  - Efficient device-scope communication between kernels
  - Multiple Code Generation Optimizations

- **OpenCL C++**
  - Single Source C++ Programming
  - Full support for features in C++14-based Kernel Language

- **API and Language Specs**
  - Brings C++14-based Kernel Language into core specification

- **Portable Kernel Intermediate Language**
  - Support for C++14-based kernel language e.g. constructors/destructors

---

- 3-component vectors
- Additional image formats
- Multiple hosts and devices
- Buffer region operations
- Enhanced event-driven execution
- Additional OpenCL C built-ins
- Improved OpenGL data/event interop

- Device partitioning
- Separate compilation and linking
- Enhanced image support
- Built-in kernels / custom devices
- Enhanced DX and OpenGL Interop

- Shared Virtual Memory
- On-device dispatch
- Generic Address Space
- Enhanced Image Support
- C11 Atomics
- Pipes
- Android ICD

- SPIR-V in Core
- Subgroups into core
- Subgroup query operations
- cCloneKernel
- Low-latency device timer queries

---

© Copyright Khronos Group 2017 - Page 12
Google Trends

Interest over time

- OpenCL
  Computer application
- CUDA
  Computer application
- OpenMP
  Computer application
- Apple Metal API
  Search term
- Vulkan
  API

Average

Jan 1, 2004
Aug 1, 2008
Mar 1, 2013
Embrace the Layered Ecosystem

OpenCL mixed providing low-level hardware access with ‘ease-of-use’

Didn’t make it clear that low-level performance portability is impossible

Did not focus on rapidly porting efficient libraries

Run-time abstraction hardware is needed:
- Software vendors can’t afford to port to every type/generation hardware
- Hardware vendors want to keep innovating under an abstraction

Middleware just needs direct access to hardware. Driver should ‘get out of the way’

Middleware can provide ease of use

Middleware has the system/domain context to try to provide performance portability
## Market Segments Need Deployment Flexibility

<table>
<thead>
<tr>
<th>Segment</th>
<th>Use Cases</th>
<th>Roadmap</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Desktop (actual and cloud)</strong></td>
<td>Video Image Processing, Gaming Compute, Rendering, Neural Network Training and Inferencing</td>
<td>Vulkan interop, dialable precision, pre-emption, collective programming and improved execution model, dynamic parallelism, pre-emption</td>
</tr>
<tr>
<td><strong>Mobile</strong></td>
<td>Photo and Vision Processing, Neural Network Inferencing</td>
<td>SVM, dialable precision for inference engine and pixel processing efficiency, pre-emption and QoS scheduling for power efficiency</td>
</tr>
<tr>
<td><strong>HPC</strong></td>
<td>Numerical Simulation, Neural Network Training, Virtualization</td>
<td>Enhanced streaming processing, enhanced library support</td>
</tr>
<tr>
<td><strong>FPGAs</strong></td>
<td>Network and Stream Processing</td>
<td>Enhanced execution model, self-synchronized and self-scheduled graphs, fine-grained synchronization between kernels, DSL in C++</td>
</tr>
<tr>
<td><strong>Embedded</strong></td>
<td>Vision, Signal and Pixel Processing, Neural Network and Inferencing</td>
<td>Arbitrary precision for power efficiency, hard real-time scheduling, async DMA</td>
</tr>
</tbody>
</table>

OpenCL has been over-monolithic

E.g. DSP inferencing should not be forced to ship IEEE FP32

Solution: feature sets - enabling toggling capabilities within a coherent framework without losing conformance

© Copyright Khronos Group 2017 - Page 15
## Other Lessons

<table>
<thead>
<tr>
<th>Lessons</th>
<th>How We Learned Them</th>
<th>How We Do Better!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language flexibility is good! Enable language innovation!</td>
<td>OpenCL WG spent way too long designing OpenCL C and C++</td>
<td>Ingest SPIR-V! BUT Vendors need to support it!</td>
</tr>
<tr>
<td>Lack of tools and libraries</td>
<td>Assumption that the Working Group’s job is done once the specification is shipped</td>
<td>‘Hard launches’ i.e. simultaneous availability of spec, libraries, implementations and engines</td>
</tr>
<tr>
<td>Needs to be adopted/available on key platforms</td>
<td>Apple are focused on Metal, OpenCL/RenderScript Confusion NVIDIA not pushing to 2.0</td>
<td>Add value to key platforms and/or develop viable portability solutions</td>
</tr>
<tr>
<td>Middleware and application insights and prototyping are essential during standards design</td>
<td>The OpenCL Working Group has lacked active software developer participation</td>
<td>Encourage ISVs to join Khronos to help steer the industry! AND OpenCL Advisory Panels</td>
</tr>
</tbody>
</table>
Khronos Advisory Panels

The Working Group invites input and shares draft specifications and other WG materials

Members
Pay membership Fee
Sign NDA and IP Framework
Directly participate in working groups

Advisors
Pay $0
Sign Advisors Agreement = NDA and IP Framework
Provide requirements and feedback on specification drafts to the working group

Advisory Panel membership is ‘By Invitation’ and renewed annually.
No ‘minimum workload’ commitment - but we love input and feedback!
Please reach out if you wish to participate!
Requirements for ‘OpenCL Next’

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Fulfillment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-level explicit API as Foundation of multi-layer ecosystem</td>
<td>✓</td>
</tr>
<tr>
<td>Features set for Market Deployment Flexibility</td>
<td>✓</td>
</tr>
<tr>
<td>SPIR-V Ingestion for Language flexibility</td>
<td>✓</td>
</tr>
<tr>
<td>Widely Adopted No market barriers to deployment</td>
<td>✓</td>
</tr>
<tr>
<td>Installable tools architecture for Development flexibility</td>
<td>✓</td>
</tr>
<tr>
<td>Low-latency, multi-threaded dispatch For fine-grained, high-performance</td>
<td>✓</td>
</tr>
<tr>
<td>At least OpenCL 2.X-class compute capabilities</td>
<td>✗</td>
</tr>
<tr>
<td>Support for diverse processor types</td>
<td>✗</td>
</tr>
</tbody>
</table>

Working Group Decision!
Converge with and leverage Vulkan design!
Expand on Vulkan supported processors types and compute capabilities
Vulkan Explicit GPU Control

Complex drivers cause overhead and inconsistent behavior across vendors

Always active error handling

Full GLSL preprocessor and compiler in driver

OpenGL vs. OpenGL ES

High-level Driver Abstraction
Layered GPU Control
Context management
Memory allocation
Full GLSL compiler
Error detection

Application
Single thread per context

Application
Memory allocation
Thread management
Synchronization
Multi-threaded generation of command buffers

Multiple Front-end Compilers
GLSL, HLSL etc.

Thin Driver
Explicit GPU Control

Loadable debug and validation layers

Resource management offloaded to app:
low-overhead, low-latency driver

Consistent behavior:
no ‘fighting with driver heuristics’

Validation and debug layers loaded only when needed

SPIR-V intermediate language: shading language flexibility

Multi-threaded command creation. Multiple graphics, command and DMA queues

Unified API across all platforms with feature set flexibility

Vulkan 1.0 provides access to
OpenGL ES 3.1 / OpenGL 4.X-class GPU functionality
but with increased performance and flexibility
Vulkan Adoption

All Major GPU Companies shipping Vulkan Drivers - for Desktop and Mobile Platforms

Mobile, Embedded and Console Platforms Supporting Vulkan

Android 7.0
Nintendo Switch
Android TV
Embedded Linux

Cross Platform
SteamOS
ubuntu
redHat
TIZEN
Nintendo Switch
Android 7
GPU Portability - Call For Participation

No single API on all systems

Use Feature Sets to remove non-portable functionality

Use Extensions to add functionality e.g. security and robustness for the Web

Vulkan is non-proprietary and is already designed to be portable

A Portability Solution needs to address APIs and shading languages

‘WebGL Next’
- Lift ‘Portability API’ to JavaScript and use in WebAssembly native code
- Nexgen graphics and GPU compute for the Web

Vulkan Portability Solution
C/C++ Portability API Library
+ Shading Language tools
All open source

Portable ‘Vulkan Subset’ API Specification

Open source compilers/ translators for shading and intermediate languages

MIR → MSL
DX IL
GLSL
HLSL

Vulkan is non-proprietary and is already designed to be portable

API Overlap Analysis

Use Feature Sets to remove non-portable functionality

Use Extensions to add functionality e.g. security and robustness for the Web
‘OpenCL-V’ - OpenCL and Vulkan Convergence

- Converge OpenCL roadmap over time with Vulkan API and run-time
  - Support more processor types, e.g. DSPs and FPGAs (graphics optional)
- Layered ecosystem for backwards-compatibility and market flexibility
  - Feature sets for target market agility
- Single runtime stack for graphics and compute
  - Streamline development, adoption and deployment for the entire industry

Applications

- Vendor-supplied and open source middleware
  - OpenCL 1.X/2.X Compatibility
  - Math Libraries
  - Language Front-ends
  - Tool Layers

Installable tool & validation layers

Thin, explicit Vulkan run-time with rigorous memory/execution model. Low-latency and predictable

Vulkan API

- Dialable types and precision
- Real-time Pre-emption and QoS scheduling
- Explicit Asynch DMA
- Self-synchronized, self-scheduled graphs
- Stream Processing

Feature Sets can be enabled for particular target markets

© Copyright Khronos Group 2017 - Page 22
OpenCL Evolution Discussions

Industry working to bring Heterogeneous compute to standard ISO C++
C++17 Parallel STL hosted by Khronos
Executors - for scheduling work
“Managed pointers” or “channels” - for sharing data
Hoping to target C++ 20 but timescales are tight

‘OpenCL-V’
Converge Vulkan and OpenCL

Your Input!

OpenCL Evolution

- **2011**
  - OpenCL 1.2
  - Focus on embedded imaging, vision and inferencing
  - Make FP32 optional for DSPs and general power efficiency

- **2015**
  - OpenCL 2.1
  - SPIR-V in Core

- **2017**
  - OpenCL 2.2
  - C++ Kernel Language

Single source C++ programming.
Great for supporting C++ apps, libraries and frameworks

Industry working to bring Heterogeneous compute to standard ISO C++
C++17 Parallel STL hosted by Khronos
Executors - for scheduling work
“Managed pointers” or “channels” - for sharing data
Hoping to target C++ 20 but timescales are tight

‘OpenCL-V’
Converge Vulkan and OpenCL

Your Input!
Get Involved!

- OpenCL is driving to a new level of community engagement
  - Learning from the Vulkan experience
  - We need to know what you need from OpenCL
  - IWOCL is the perfect opportunity to find out!

- Any company or organization is welcome to join Khronos
  - For a voice and a vote in any of these standards
  - www.khronos.org

- If joining is not possible - ask about the OpenCL Advisory Panel
  - Free of charge - enables design reviews, requirements and contributions

- Neil Trevett
  - ntrevett@nvidia.com
  - @neilt3d