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

The Good

The amazing progress of OpenCL

The Bad

Lessons Learned from the first eight years

The Exciting

Where do we go from here?

1 2 3

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

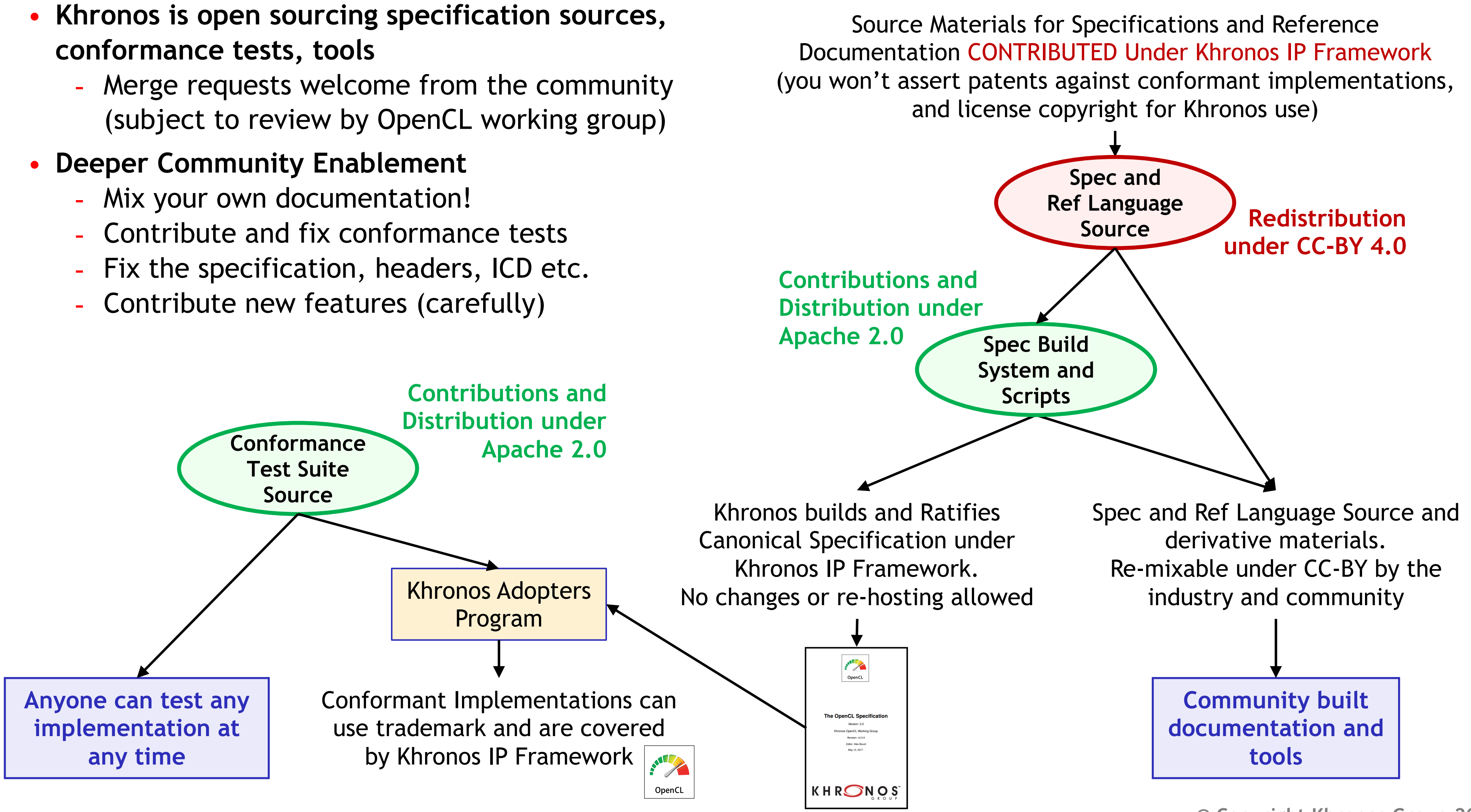
Code Generation Optimizations
- Specialization constants at SPIR-V compilation time
- Constructors and destructors of program scope global objects
- User callbacks can be set at program release time

https://www.khronos.org/opencl/

OpenCL 2.2 Reference Card
Page 1
OpenCL API Reference
The OpenCL Platform Layer
The OpenCL Runtime
©2017 Khronos Group - Rev. 0517
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 and Ref Language Source

Redistribution under CC-BY 4.0

Contributions and Distribution under Apache 2.0

Spec Build System and Scripts

Khronos 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

Conformance Test Suite Source

Contributions and Distribution under Apache 2.0

Khronos Adopters Program

Conformant Implementations can use trademark and are covered by Khronos IP Framework

Anyone can test any implementation at any time

Shout Out to University of Windsor

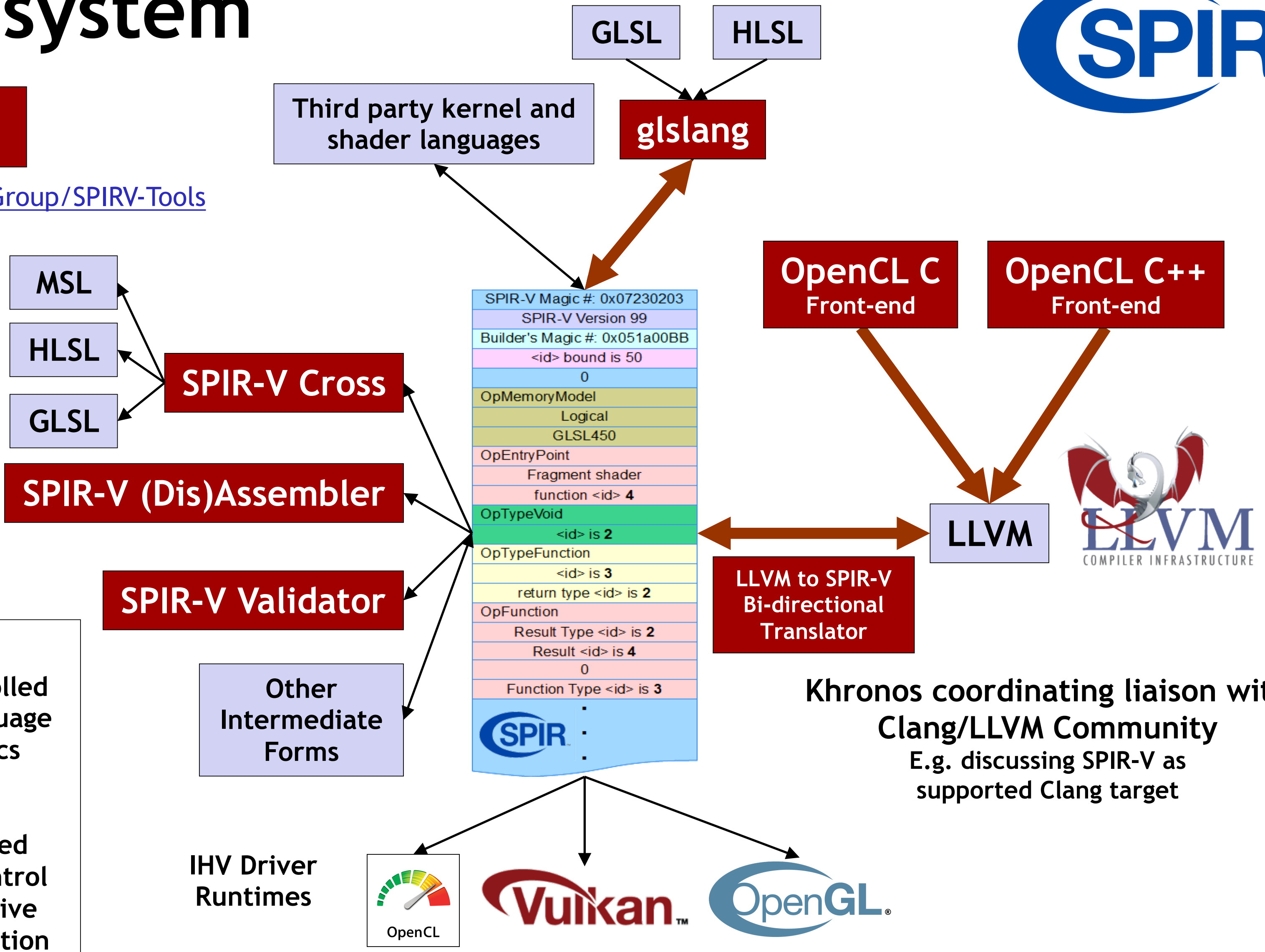
The Windsor Testing Framework, also released today, enables developers to quickly install and configure the OpenCL Conformance Test Suite on their own systems.

University of Windsor

SPIR-V Ecosystem

Khronos has open sourced these tools and translators

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



Khronos coordinating liaison with Clang/LLVM Community

E.g. discussing SPIR-V as supported Clang target

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

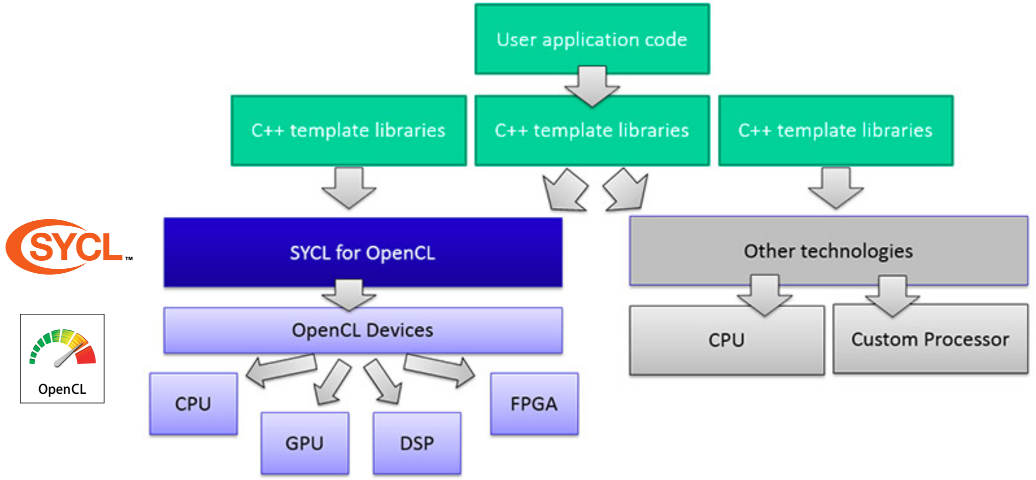
SYCL Ecosystem

- **Single-source heterogeneous programming using STANDARD C++**
  - Use C++ templates and lambda functions for host & device code
  - Layerered 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**

Developer Choice

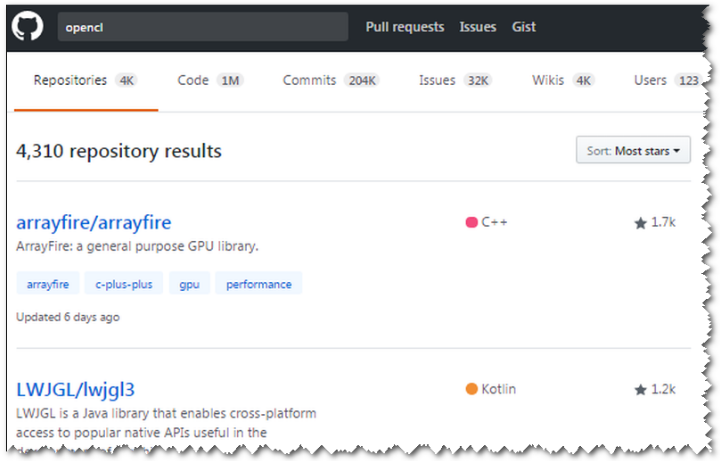
The development of the two specifications are aligned so code can be easily shared between the two approaches

C++ Kernel Language	Single-source C++
Low Level Control	Programmer Familiarity
'GPGPU'-style separation of device-side kernel source code and host code	Approach also taken by C++ AMP and OpenMP
OpenCL	SYCL™



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

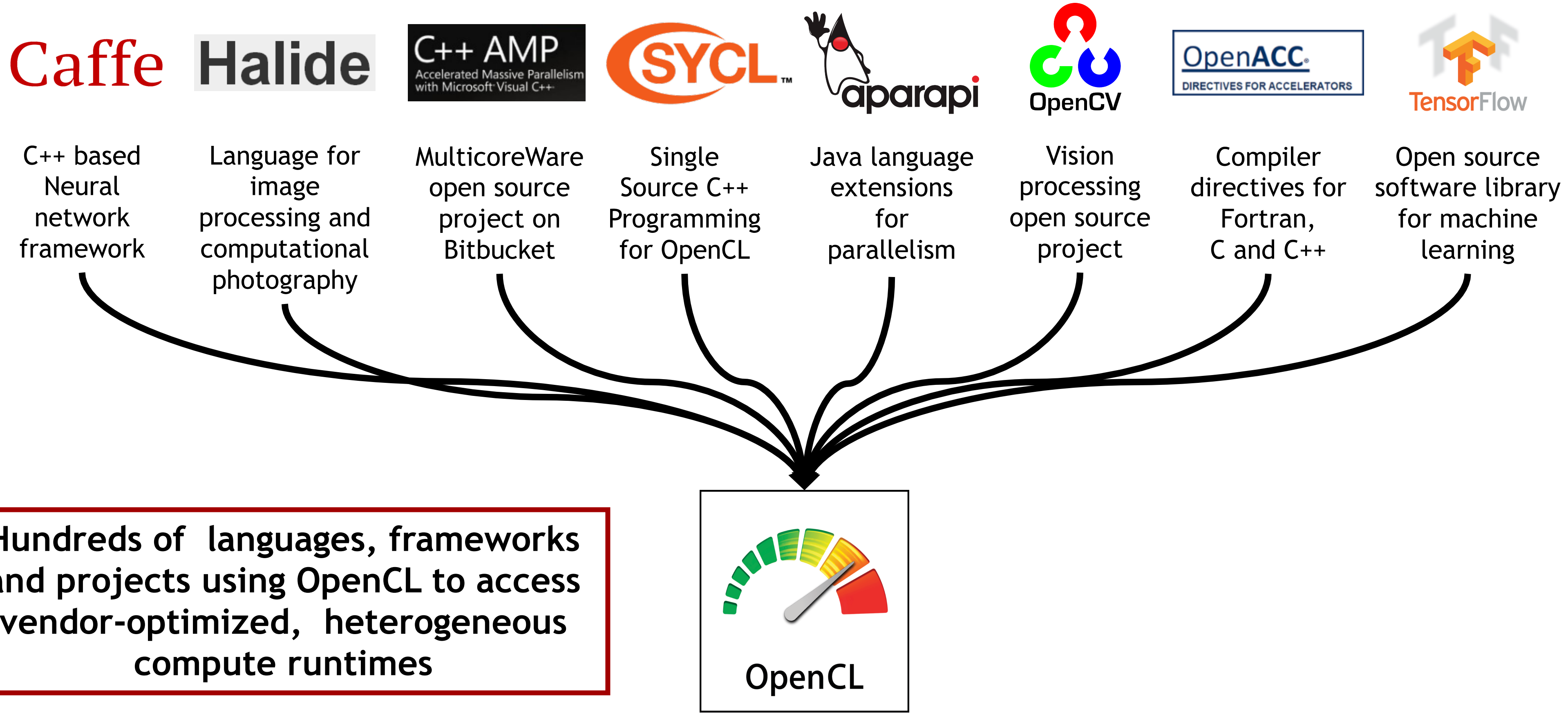

opencl
Pull requests Issues Gist
Repositories 4K Code 1M Commits 204K Issues 32K Wikis 4K Users 123
Sort: Most stars

4,310 repository results

arrayfire/arrayfire — C++ — ★ 1.7k
ArrayFire: a general purpose GPU library.
arrayfire c-plus-plus gpu performance
Updated 6 days ago

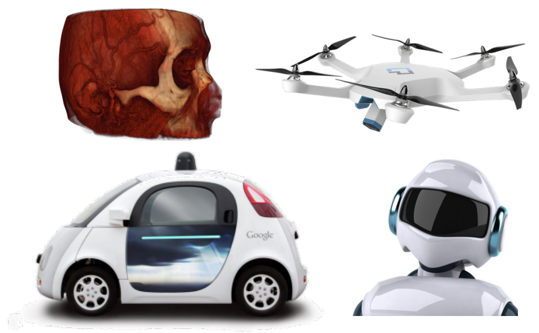
LWJGL/lwjgl3 — Kotlin — ★ 1.2k
LWJGL is a Java library that enables cross-platform access to popular native APIs useful in the

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++
MulticoreWare open source project on Bitbucket
SYCL
Single Source C++ Programming for OpenCL
aparapi
Java language extensions for parallelism
OpenCV
Vision processing open source project
OpenACC
DIRECTIVES FOR ACCELERATORS
Compiler directives for Fortran, C and C++
TensorFlow
Open source software library for machine learning
OpenCL

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

Safety Critical APIs



OpenGL SC 1.0 - 2005
Fixed function graphics subset

OpenGL ES 1.0 - 2003
Fixed function graphics

OpenGL ES 2.0 - 2007
Shader programmable pipeline

OpenGL SC 2.0 - April 2016
Shader programmable pipeline subset

Experience and Guidelines

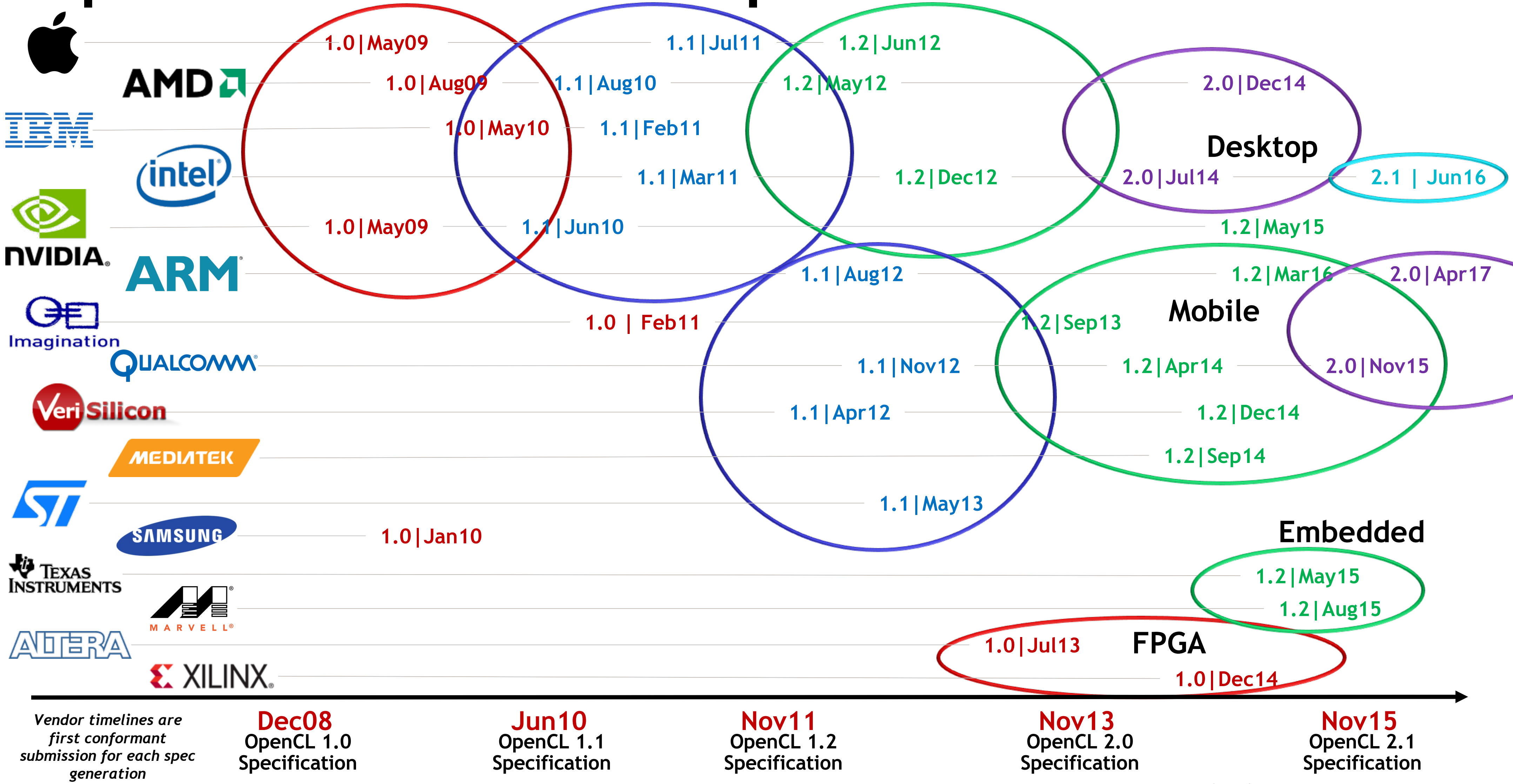
New Generation APIs for safety certifiable vision, graphics and compute
e.g. ISO 26262 and DO-178B/C

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

OpenCL Conformant Implementations



Vendor timelines are first conformant submission for each spec generation

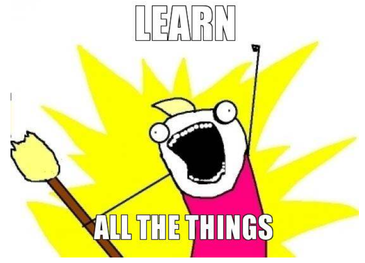
OpenCL - 1000s Man Years Effort

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

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



Dec08 OpenCL 1.0 Specification

18 months

Jun10 OpenCL 1.1 Specification
- 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

18 months

Nov11 OpenCL 1.2 Specification
- Device partitioning
- Separate compilation and linking
- Enhanced image support
- Built-in kernels / custom devices
- Enhanced DX and OpenGL Interop

24 months

Nov13 OpenCL 2.0 Specification
- Shared Virtual Memory
- On-device dispatch
- Generic Address Space
- Enhanced Image Support
- C11 Atomics
- Pipes
- Android ICD

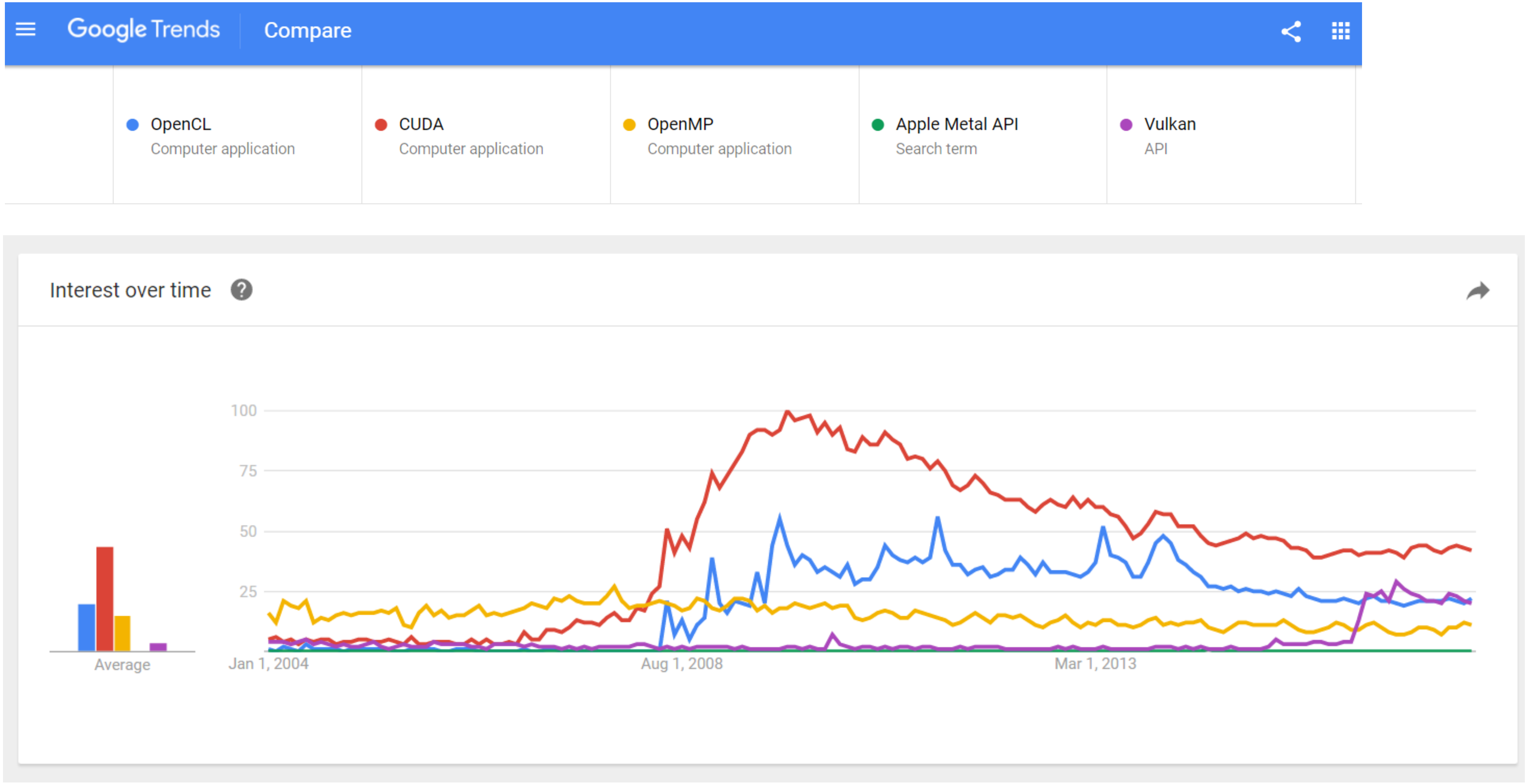
24 months

Nov15 OpenCL 2.1 Specification
- SPIR-V in Core
- Subgroups into core
- Subgroup query operations
- clCloneKernel
- Low-latency device timer queries

18 months

May17 OpenCL 2.2 Specification

Google Trends

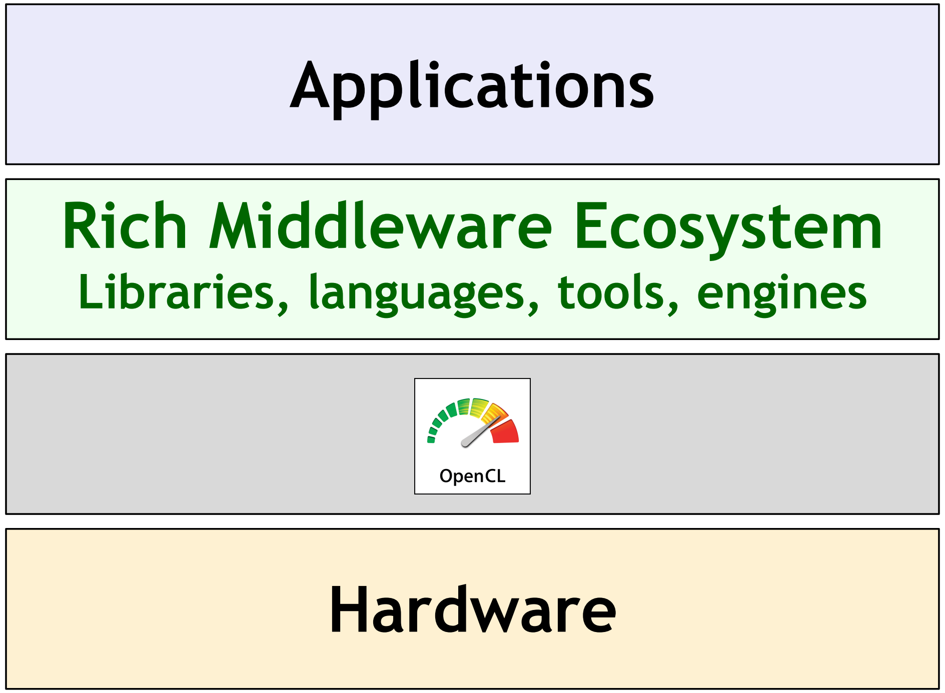


Embrace the Layered Ecosystem

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


Applications
Rich Middleware Ecosystem
Libraries, languages, tools, engines
OpenCL
Hardware

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

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

Market Segments Need Deployment Flexibility

OpenCL

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

Desktop (actual and cloud)

Use cases: Video Image Processing, Gaming Compute, Rendering, Neural Network Training and Inferencing

Roadmap: Vulkan interop, dialable precision, pre-emption, collective programming and improved execution model, dynamic parallelism, pre-emption

Mobile

Use case: Photo and Vision Processing, Neural Network Inferencing

Roadmap: SVM, dialable precision for inference engine and pixel processing efficiency, pre-emption and QoS scheduling for power efficiency

HPC

Use case: Numerical Simulation, Neural Network Training, Virtualization

Roadmap: enhanced streaming processing, enhanced library support

FPGAs

Use cases: Network and Stream Processing

Roadmap: enhanced execution model, self-synchronized and self-scheduled graphs, fine-grained synchronization between kernels, DSL in C++

Embedded

Use cases: Vison, Signal and Pixel Processing, Neural Network and Inferencing

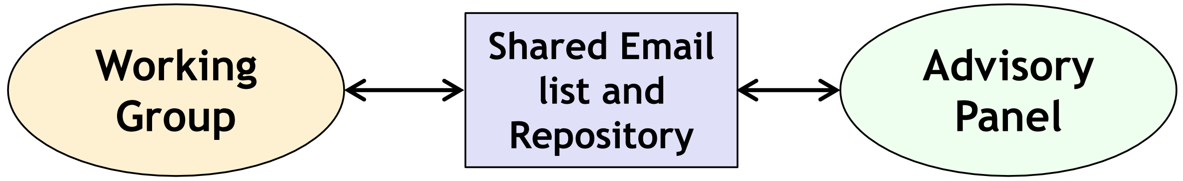
Roadmap: arbitrary precision for power efficiency, hard real-time scheduling, asynch DMA

Other Lessons

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

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'

OpenCL

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

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

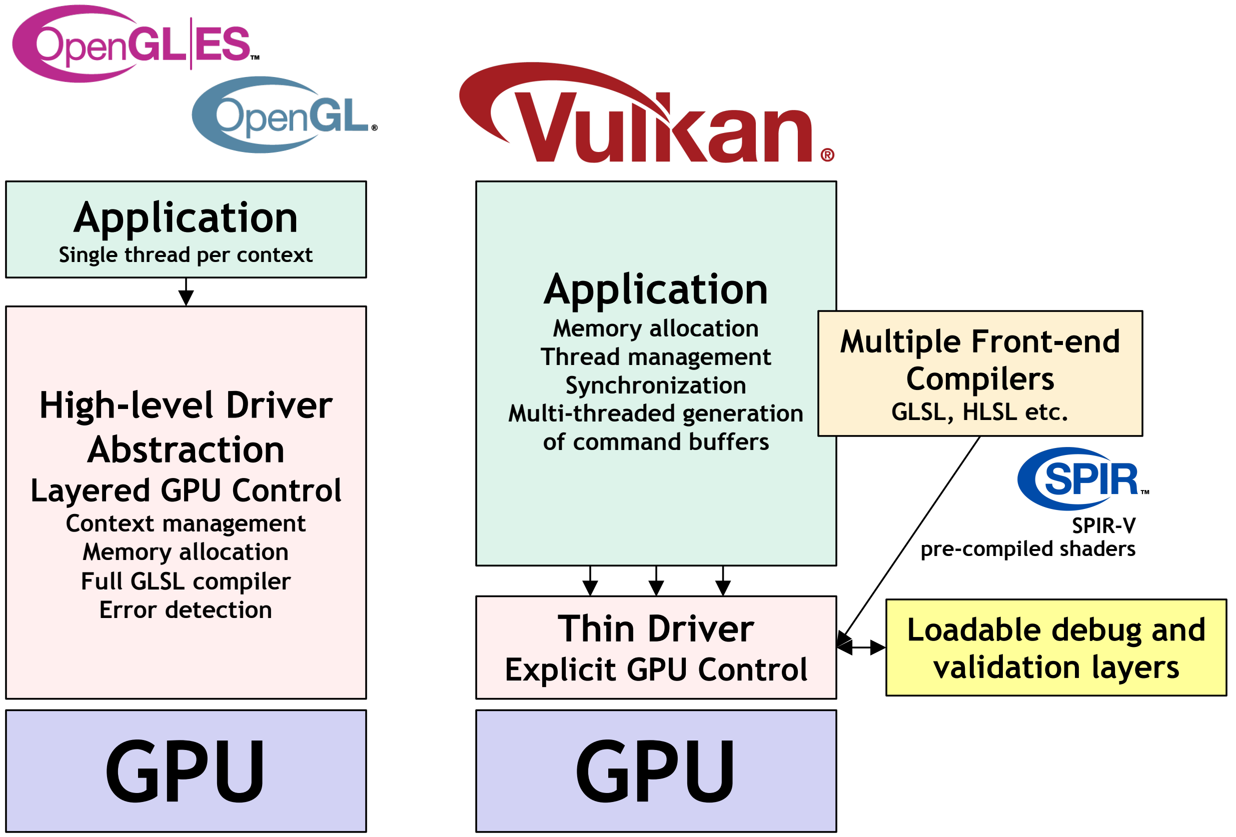
Vulkan Explicit GPU Control

Complex drivers cause overhead and inconsistent behavior across vendors

Always active error handling

Full GLSL preprocessor and compiler in driver

OpenGL vs. OpenGL ES



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

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 Adoption

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

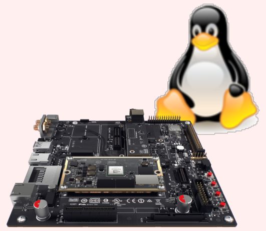
AMD ARM Imagination intel NVIDIA QUALCOMM VeriSilicon

Mobile, Embedded and Console Platforms Supporting Vulkan

Android 7.0

Nintendo Switch

Android TV

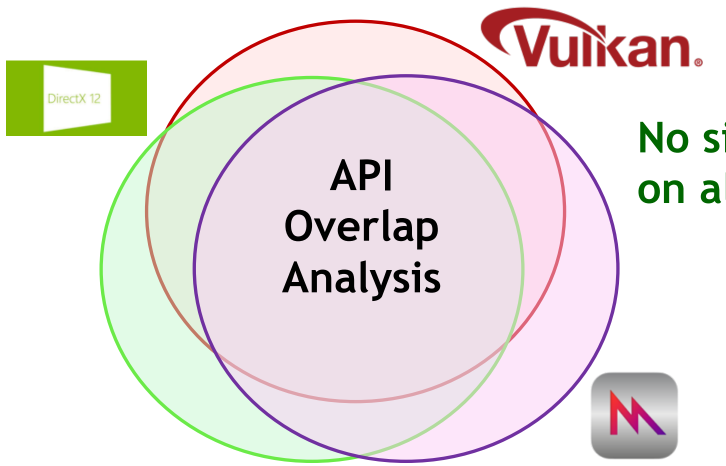

Embedded Linux

Vulkan®

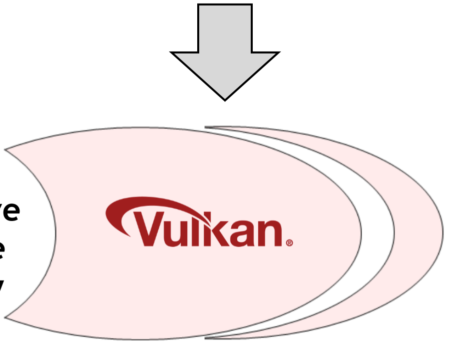
Cross Platform

Windows 7 Windows 8 Windows 10 SteamOS ubuntu redhat TIZEN NINTENDO SWITCH 7

GPU Portability - Call For Participation



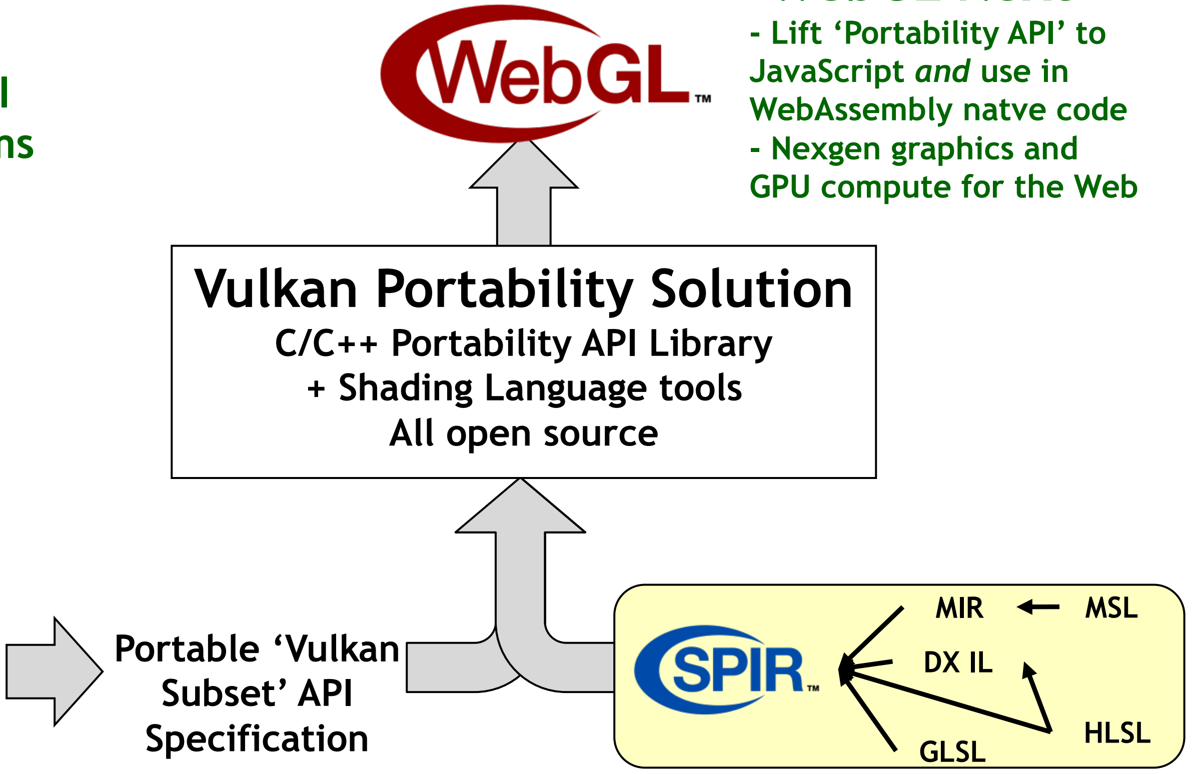
No single API on all systems



Use Feature Sets to remove non-portable functionality

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

Portable 'Vulkan Subset' API Specification



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

'WebGL Next'
- Lift 'Portability API' to JavaScript *and* use in WebAssembly natve code
- Nexgen graphics and GPU compute for the Web

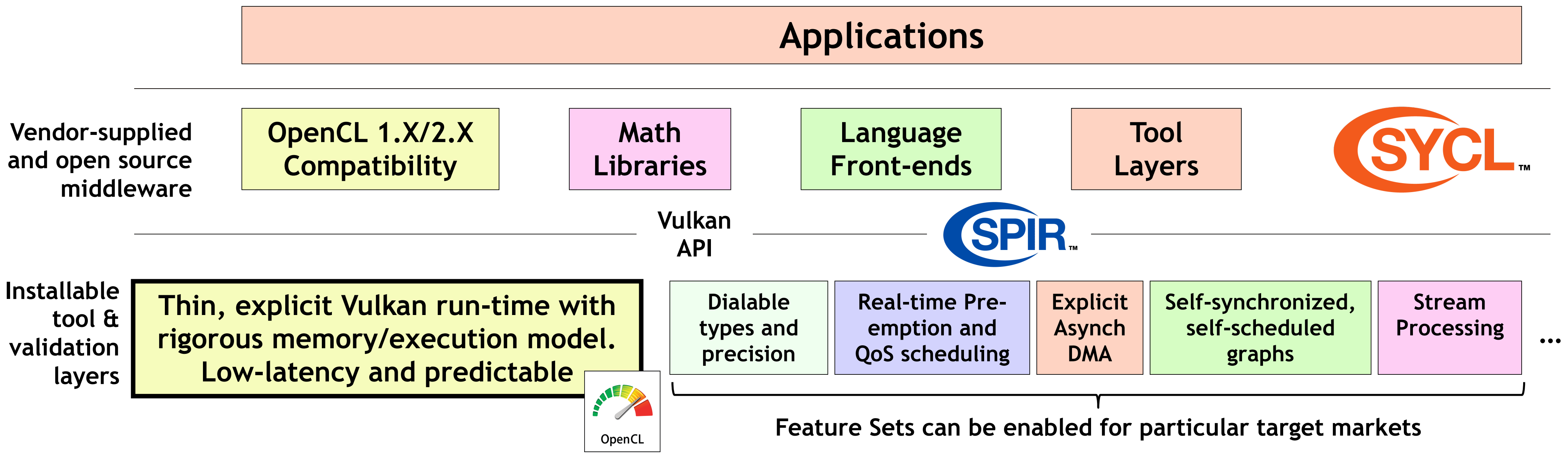
Open source compilers/translators for shading and intermediate languages

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

A Portability Solution needs to address APIs *and* shading languages

‘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
SYCL
Vulkan API
SPIR
Installable tool & validation layers
Thin, explicit Vulkan run-time with rigorous memory/execution model. Low-latency and predictable
OpenCL
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

OpenCL Evolution Discussions

C++ AMP
Accelerated Massive Parallelism with Microsoft Visual C++

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

NVIDIA CUDA

The C++ Programming Language

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

SYCL 1.2
C++11 Single source programming

SYCL 2.2
C++14 Single source programming

OpenCL

2011
OpenCL 1.2

2015
OpenCL 2.1
SPIR-V in Core

2017
OpenCL 2.2
C++ Kernel Language

OpenCL 1.2++?
Focus on embedded imaging, vision and inferencing
Make FP32 optional for DSPs and general power efficiency

Vulkan

'OpenCL-V'
Converge Vulkan and OpenCL

Your Input!

Get Involved!

- **OpenCL is driving to 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