OpenCL Working Group Update IWOCL 2025

Ben Ashbaugh, Intel

IWOCL 2025

13th International Workshop on OpenCL and SYCL

April 7-11, 2025
Heidelberg, Germany

AMD
FIREPRO
intel
Look Inside
Performance to boost productivity
PAUL & MILDRED BERG HALL
Accelerating SGEMM with Subgroups
EXIT

Khronos Connects Software to Silicon

Over 150 members worldwide
Any organization is welcome to join

Liaisons: Cooperation with industry associations and organizations

KHRONOS GROUP

Non-profit Standards Consortium creating open, royalty-free standards

Focused on runtime APIs and file formats for 3D, XR, AI, vision, and parallel compute acceleration

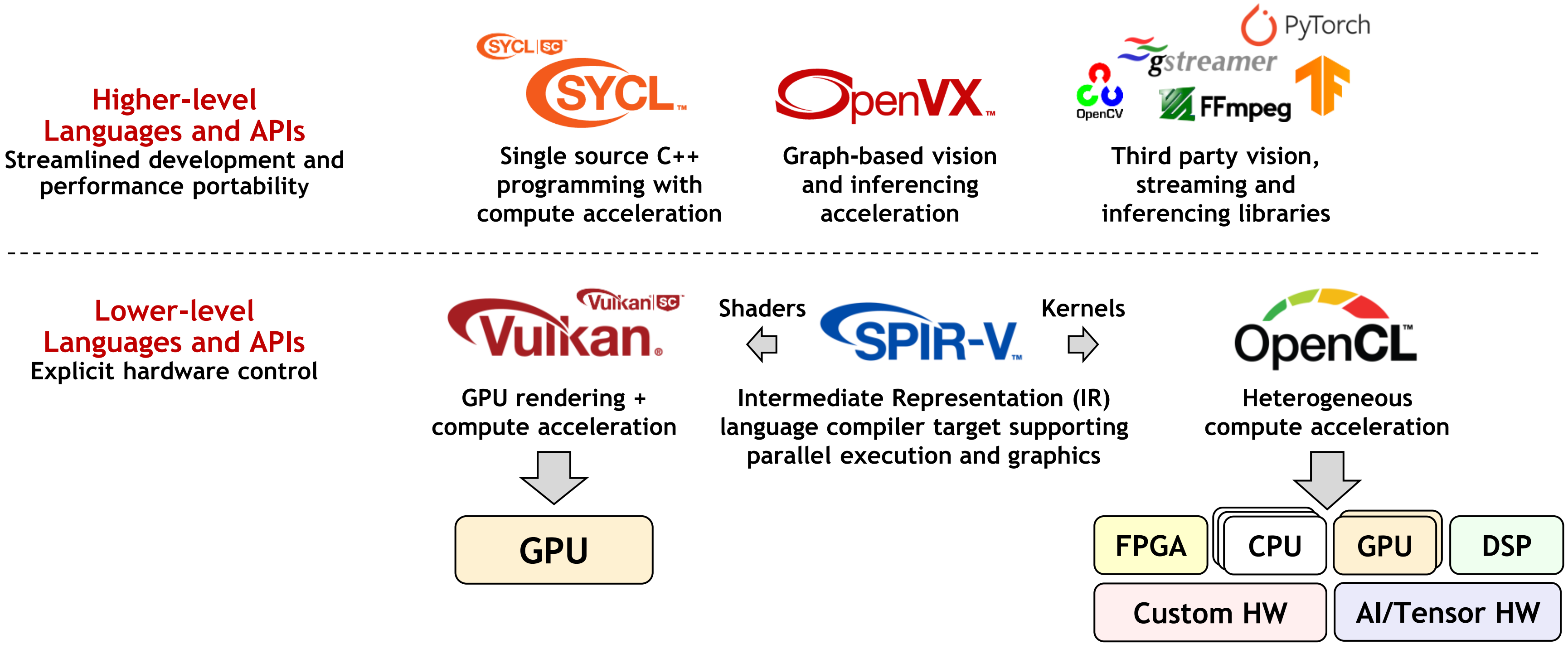
Member-driven, open to any company

Founded in 2000
~ 160 Members | ~ 40% US, 30% Europe, 30% Asia
ISO/IEC JTC 1 PAS Submitter

Khronos Compute Acceleration

Choice of programming models to meet the needs of diverse developers

Higher-level applications, libraries, and languages often access hardware acceleration through lower-level APIs



Higher-level Languages and APIs
Streamlined development and performance portability

SYCL SC
SYCL
Single source C++ programming with compute acceleration

OpenVX
Graph-based vision and inferencing acceleration

PyTorch
gstreamer
OpenCV
FFmpeg
Third party vision, streaming and inferencing libraries

Lower-level Languages and APIs
Explicit hardware control

Vulkan SC
Vulkan
GPU rendering + compute acceleration

GPU

Shaders
SPIR-V
Kernels
Intermediate Representation (IR) language compiler target supporting parallel execution and graphics

OpenCL
Heterogeneous compute acceleration

FPGA
CPU
GPU
DSP
Custom HW
AI/Tensor HW

OpenCL - Low-level Parallel Programing

Programming and Runtime Framework for Application Acceleration

Offload compute-intensive kernels onto parallel heterogeneous processors

CPUs, GPUs, DSPs, FPGAs, Tensor Processors

OpenCL C or C++ kernel languages

Platform Layer API

Query, select and initialize compute devices

Runtime API

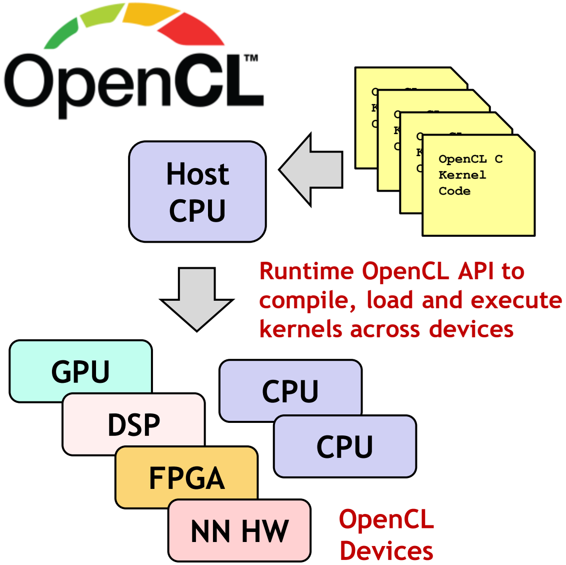
Build and execute kernels programs on multiple devices

Explicit Application Control

Which programs execute on what device

Where data is stored in memories in the system

When programs are run, and what operations are dependent on earlier operations



Runtime OpenCL API to compile, load and execute kernels across devices

OpenCL Devices

Complements GPU-only APIs

Simpler programming model

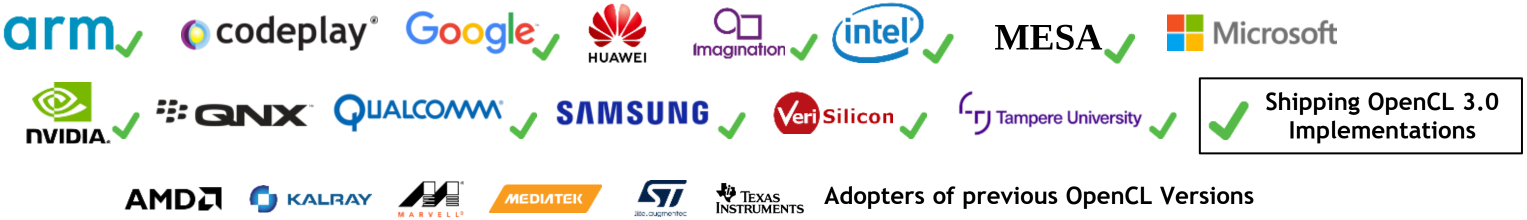
Relatively lightweight run-time

More language flexibility, e.g., pointers

Rigorously defined numeric precision

OpenCL 3.0 Growing Adoption and Roadmap

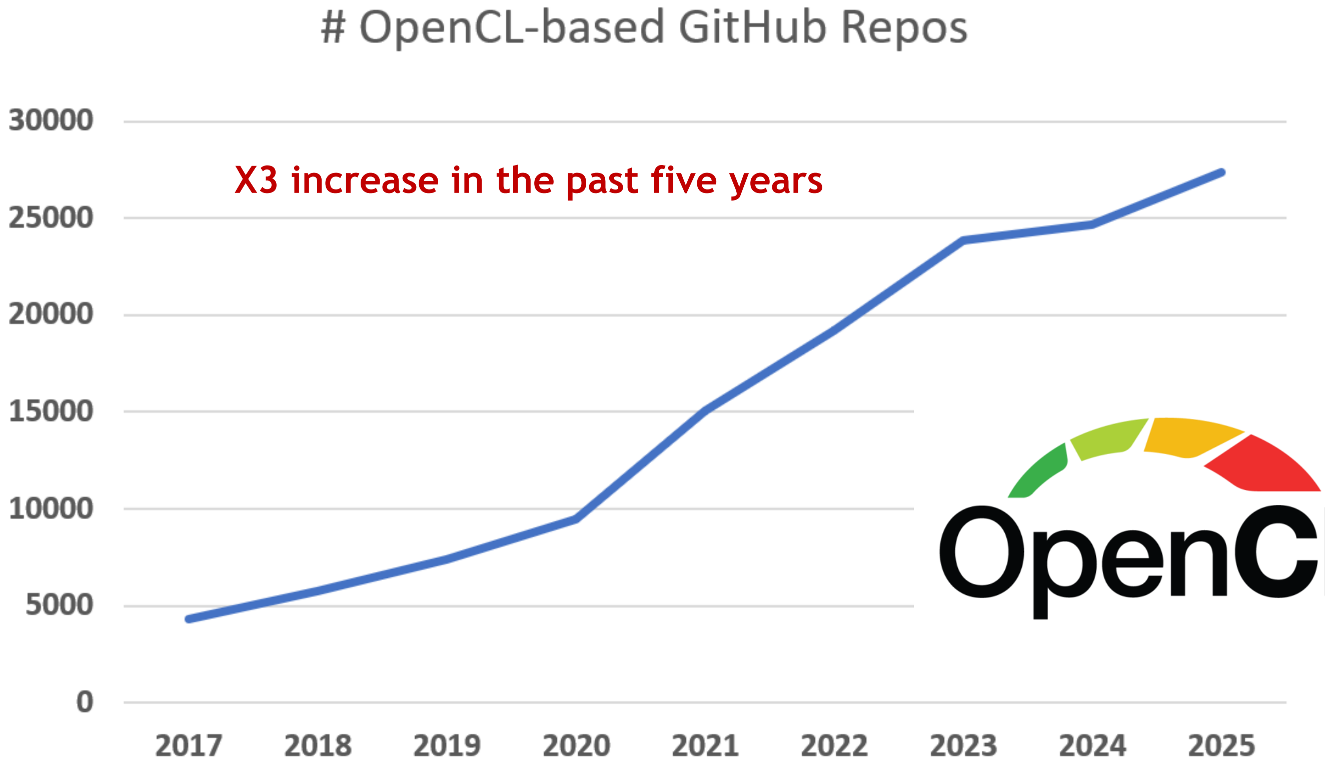
- **Active Extension Pipeline - driven by mobile, embedded and desktop markets**
  - Recordable Command Buffers, Mutable Command Buffers, Cooperative Matrix, AI Data Formats
  - Unified Shared Memory, Image Tiling Controls, External Memory Improvements, and more!
- **Regular Releases 3x-4x Per Year**
  - 3.0.18 released in April 2025 with new EXT extensions, clarifications, and bug fixes
- **Considerable Open-Source Activity**
  - Mesa Rusticl for Linux, PoCL, OneAPI Construction Kit
  - clang/LLVM compilation front-ends + SPIR-V LLVM back-end
  - Layered implementations over Vulkan, over DX12, and more!
- **Emerging acceptance of OpenCL as compute layer**
  - Especially for Machine Learning



OpenCL 3.0 growing adoption

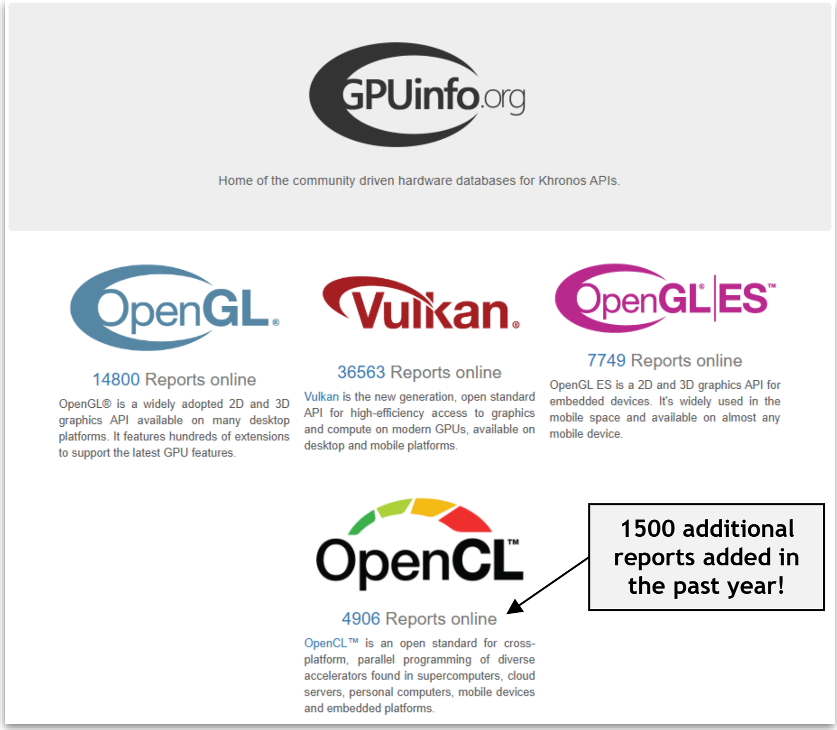
https://www.khronos.org/conformance/adopters/conformant-products/opencl

OpenCL Open-Source Project Momentum



X3 increase in the past five years

OpenCL on GPUInfo.org



GPUinfo.org

Home of the community driven hardware databases for Khronos APIs.

OpenGL

14800 Reports online

OpenGL® is a widely adopted 2D and 3D graphics API available on many desktop platforms. It features hundreds of extensions to support the latest GPU features.

Vulkan

36563 Reports online

Vulkan is the new generation, open standard API for high-efficiency access to graphics and compute on modern GPUs, available on desktop and mobile platforms.

OpenGL ES

7749 Reports online

OpenGL ES is a 2D and 3D graphics API for embedded devices. It's widely used in the mobile space and available on almost any mobile device.

OpenCL

4906 Reports online

OpenCL™ is an open standard for cross-platform, parallel programming of diverse accelerators found in supercomputers, cloud servers, personal computers, mobile devices and embedded platforms.

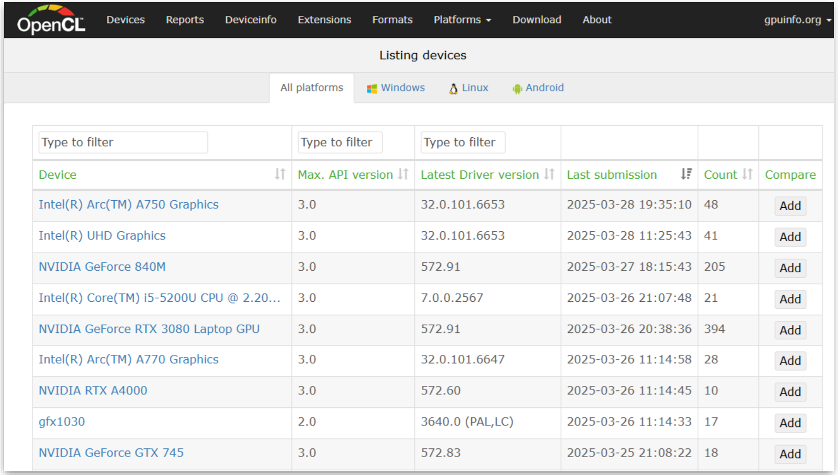
1500 additional reports added in the past year!

The online GPUinfo.org database is populated using the OpenCL Hardware Capability Viewer application

Available for Windows, Linux and Android

Reads and displays OpenCL information and uploads to the database

Please download and run to help populate the database!



Listing devices

All platforms | Windows | Linux | Android

Device	Max. API version	Latest Driver version	Last submission	Count	Compare
Intel(R) Arc(TM) A750 Graphics	3.0	32.0.101.6653	2025-03-28 19:35:10	48	Add
Intel(R) UHD Graphics	3.0	32.0.101.6653	2025-03-28 11:25:43	41	Add
NVIDIA GeForce 840M	3.0	572.91	2025-03-27 18:15:43	205	Add
Intel(R) Core(TM) i5-5200U CPU @ 2.20...	3.0	7.0.0.2567	2025-03-26 21:07:48	21	Add
NVIDIA GeForce RTX 3080 Laptop GPU	3.0	572.91	2025-03-26 20:38:36	394	Add
Intel(R) Arc(TM) A770 Graphics	3.0	32.0.101.6647	2025-03-26 11:14:58	28	Add
NVIDIA RTX A4000	3.0	572.60	2025-03-26 11:14:45	10	Add
gfx1030	2.0	3640.0 (PAL,LC)	2025-03-26 11:14:33	17	Add
NVIDIA GeForce GTX 745	3.0	572.83	2025-03-25 21:08:22	18	Add

API Layering

Enabled by growing robustness of open-source compiler ecosystem using SPIR-V

Layers Over	Vulkan	OpenGL	OpenCL	OpenGL ES	DX12	DX8-11
Vulkan		Zink	clspv + clvk/Angle Rusticl + Zink	Angle GLOVE	vkd3d-Proton vkd3d	DXVK WineD3D
OpenGL	Ashes			Angle		WineD3D
DX12	Dozen	Microsoft 'GLOn12'	Microsoft 'CLOn12'			Microsoft D3D11On12
DX9-11	Ashes			Angle		
Metal	MoltenVK			Angle MoltenGL		

ROWS Benefit Platforms by adding APIs

COLUMNS Benefit ISVs by making an API available everywhere

Layered OpenCL Implementations

clspv + clvk

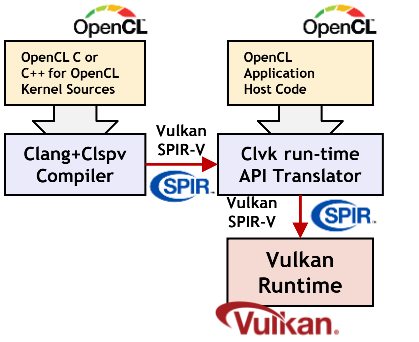
OpenCL over Vulkan

Google

clspv open-source OpenCL kernel to Vulkan SPIR-V compiler - tracks top-of-tree LLVM and Clang - not a fork

clvk - prototype open-source OpenCL to Vulkan run-time API translator

Used by shipping apps and engines on Android e.g., Adobe Premiere Rush video editor - 200K lines of OpenCL C kernel code


OpenCL C or C++ for OpenCL Kernel Sources
Clang+Clspv Compiler
Vulkan SPIR-V
OpenCL Application Host Code
Clvk run-time API Translator
Vulkan SPIR-V
Vulkan Runtime

clspv + Ancle

OpenCL over Vulkan

Samsung

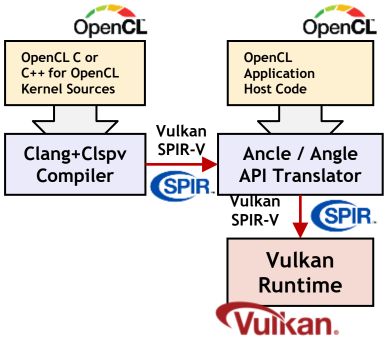
Integrates clspv and OpenCL runtime into Angle code base

Samsung Motivation

"OpenCL is widely used and deployed and is making a comeback thanks to ML"

"OpenCL is a favored high-level (front-end) compute language! Easier to write than Vulkan"

Ancle makes OpenCL a first-class citizen in Android by relying on Vulkan as its Native Driver"

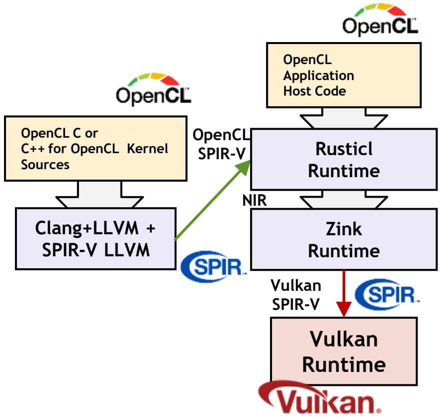

OpenCL C or C++ for OpenCL Kernel Sources
Clang+Clspv Compiler
Vulkan SPIR-V
OpenCL Application Host Code
Ancle / Angle API Translator
Vulkan SPIR-V
Vulkan Runtime

Rusticl over Zink

OpenCL over Vulkan

Mesa

The Zink Gallium driver emits Vulkan API calls and now supports OpenCL Kernels

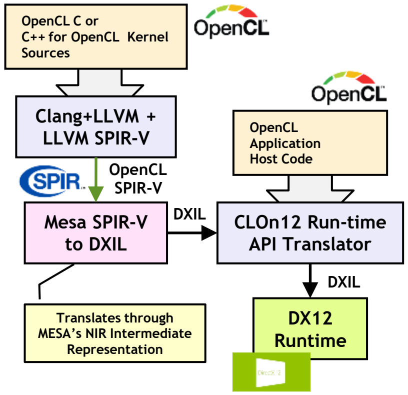

OpenCL Application Host Code
OpenCL C or C++ for OpenCL Kernel Sources
OpenCL SPIR-V
Rusticl Runtime
NIR
Clang+LLVM + SPIR-V LLVM
Zink Runtime
Vulkan SPIR-V
Vulkan Runtime

OpenCLOn12

OpenCL over DX12

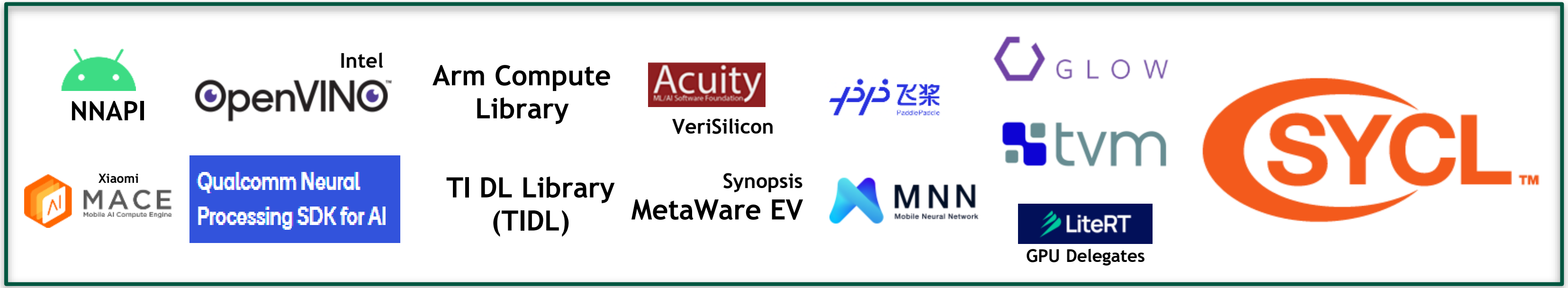
Microsoft

GPU-accelerated OpenCL on any DX12 PC and Cloud instance (x86 or Arm)


OpenCL C or C++ for OpenCL Kernel Sources
Clang+LLVM + LLVM SPIR-V
OpenCL SPIR-V
Mesa SPIR-V to DXIL
Translates through MESA's NIR Intermediate Representation
DXIL
OpenCL Application Host Code
CLOn12 Run-time API Translator
DXIL
DX12 Runtime

OpenCL and AI

- **OpenCL is often used as a backend for ML compilers and inference engines**
  - Especially in the embedded and mobile markets
- **OpenCL remains a popular backend API for SYCL**
  - SYCL support in ML frameworks means OpenCL support in ML frameworks


NNAPI

Intel

Arm Compute Library

VeriSilicon

Xiaomi

TI DL Library (TIDL)

Synopsis MetaWare EV

GPU Delegates

- **OpenCL has a robust pipeline of AI-related extensions**
  - Recordable command buffers, including mutable command buffers
  - Cooperative matrix, for standard access to dedicated matrix hardware
  - New AI data types, such as bfloat16 and fp8

Reflections, Things I’ve Learned, and OpenCL Working Group Activities

Innovation vs. Standardization

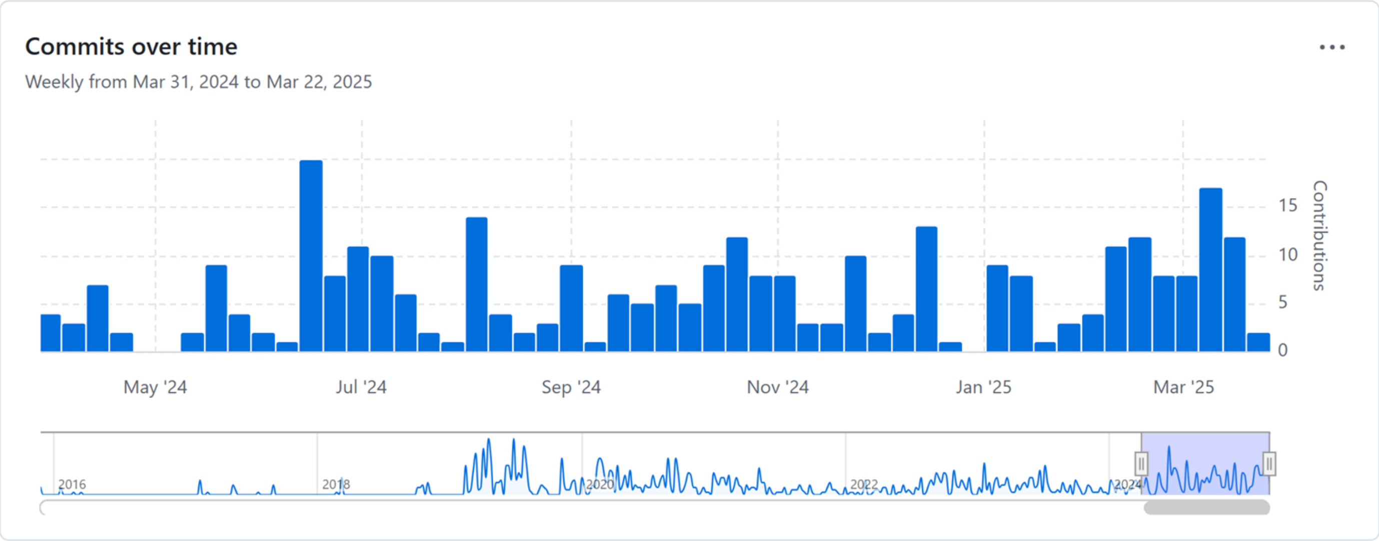
- **The OpenCL working group is balancing conflicting goals:**

- **We need to be thoughtful when adding features to the core standard**
  - Is the feature useful for applications? Is it implementable?
  - Does the feature integrate well with the existing standard?
- **We also want to encourage innovation and new features**
  - Reward first movers, explore emerging use-cases
- **How can we gain experience with a feature before adding it to the standard?**
  - Extensions!
  - Key change in 2025: more extension development is occurring in public

Innovate via Extensions, Standardize Best Practice

Tests Are Super Important!

- “A Standard Is Only As Good As Its Tests”
- Enhancing the OpenCL Conformance Test Suite (CTS) is a focus



Commits over time

Weekly from Mar 31, 2024 to Mar 22, 2025

- Almost 300 commits in the past year, over 40 contributors
- Reminder: Almost all CTS development occurring in public!
  - https://github.com/KhronosGroup/OpenCL-CTS

OpenCL Usage is Incredibly Diverse

- **Many applications use OpenCL and OpenCL C directly**
  - We will hear about many of them here at IWOCL
  - We will continue to support them and support them well
- **Many applications are also using OpenCL indirectly!**
  - Using OpenCL as a generic Hardware Abstraction Layer (HAL)

Today:

SYCL™

Vendor Inference Engines

OpenVX™

Future Possibilities?

SYCL SC™

chipStar

OpenMP®

Meet Applications Where They Are

- **Many applications will not be re-architected to use OpenCL**
  - Features that enable integration into existing codebases are valuable!
- **Examples in 2025:**
  - **cl_khr_unified_svm**: represent memory with pointers vs. cl_mem handles
  - **cl_khr_external_memory**: zero copy image and buffer sharing across APIs
  - **cl_khr_external_semaphore**: synchronization across APIs
  - **SPIR-V**: compiler target for alternative source languages

Open Standards, Open Source, and OpenCL

- **Open Source and Open Standards go great together!**

- **Open Source OpenCL Implementations**
  - Rusticl, PoCL, OneAPI Construction Kit, clvk, OpenCLOn12
- **OpenCL Ecosystem Projects**
  - OpenCL SDK, OpenCL CTS, SPIR-V Backend in LLVM
- **Applications, High-Level Languages, and Libraries**
  - Too many to list! Attend the rest of the conference :-)

Great for Users: Reduces Fragmentation, Increases Confidence

Great for Implementers: Reduces Cost of Implementation!

Summary and Wrap-Up

OpenCL State of the Union

- **OpenCL 3.0 adoption remains strong!**
- **Open-source activity is significant and increasing**
  - Mesa Rusticl for Linux, SPIR-V Backend in LLVM
- **OpenCL layered implementations are shipping and conformant - especially over Vulkan**
- **OpenCL remains a popular substrate layer for higher-level models - especially for SYCL**
- **Extension pipeline remains active - driven by mobile, embedded, and desktop markets**
  - Recordable Command Buffers, Mutable Command Buffers
  - AI Data Formats, Cooperative Matrix
  - Unified Shared Memory, Image Tiling Controls, …

Developer and Users - Please Provide Your Feedback!

- **Give us your feedback on the OpenCL spec GitHub**
  - What could be added to the OpenCL ecosystem to make you more productive?
  - What API and Language features do you most need?
  - https://github.com/KhronosGroup/OpenCL-Docs
- **Or, engage via the Khronos Discord Server**
  - https://www.khr.io/khrdiscord
- **Please download and run the GPUinfo OpenCL Hardware Capability Viewer**
  - https://opencl.gpuinfo.org/download.php

OpenCL Resources

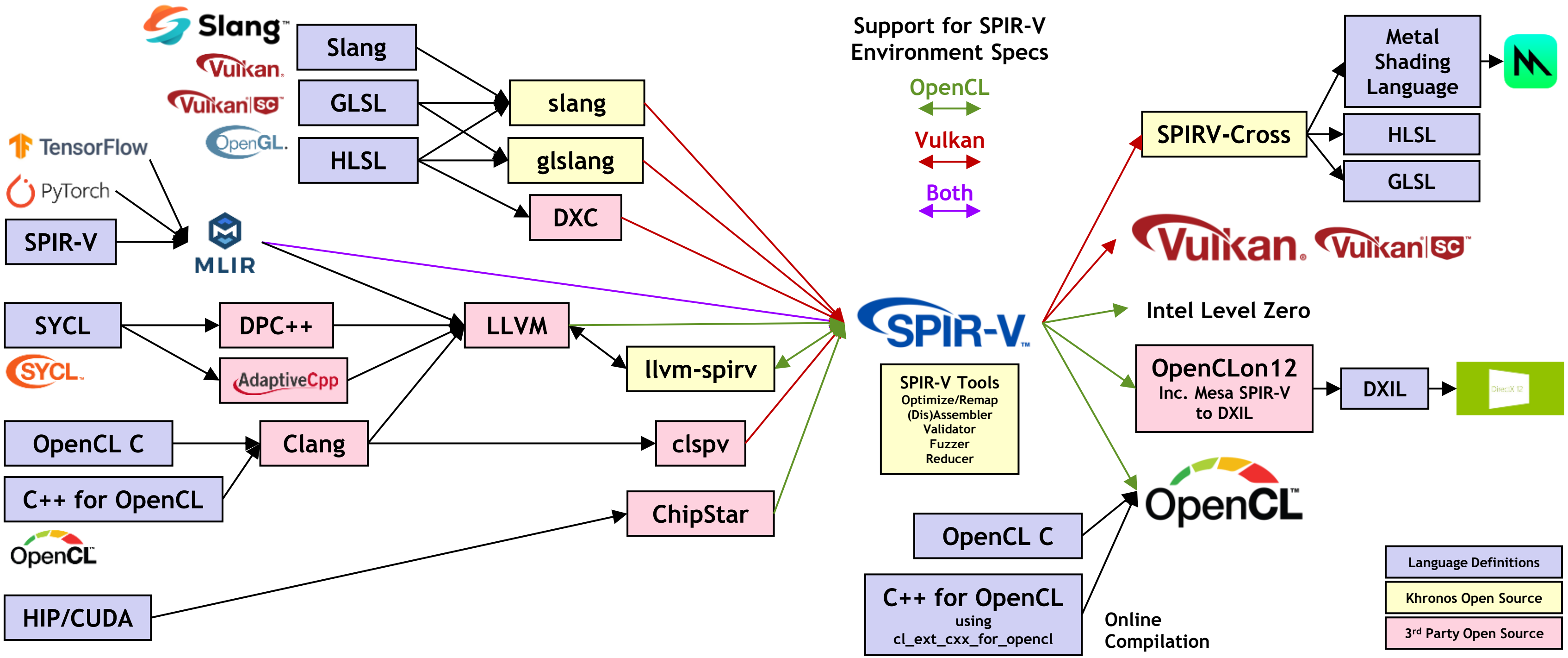
- OpenCL Home Page
  - https://www.khronos.org/opencl/
- OpenCL Registry for OpenCL Core and Extension Specifications
  - https://www.khronos.org/registry/OpenCL/
- OpenCL SDK
  - https://github.com/KhronosGroup/OpenCL-SDK
- OpenCL Specification Source
  - https://github.com/KhronosGroup/OpenCL-Docs
- OpenCL Conformant Products
  - https://www.khronos.org/conformance/adopters/conformant-products/opencl
- Open Source Implementations
  - https://gitlab.freedesktop.org/mesa/mesa (Mesa, including Rusticl)
  - https://github.com/pocl/pocl
  - https://github.com/uxlfoundation/oneapi-construction-kit
  - https://github.com/google/clspv (OpenCL C to Vulkan SPIR-V)
  - https://github.com/kpet/clvk (OpenCL over Vulkan)
  - https://github.com/microsoft/OpenCLOn12 (OpenCL over Direct3D 12)

Backup

OpenCL Random Slide Collection

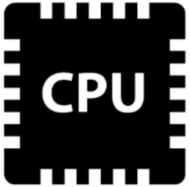
March 2025

SPIR-V Ecosystem



MLIR is part of the LLVM compiler infrastructure

The Need for Parallel Processing

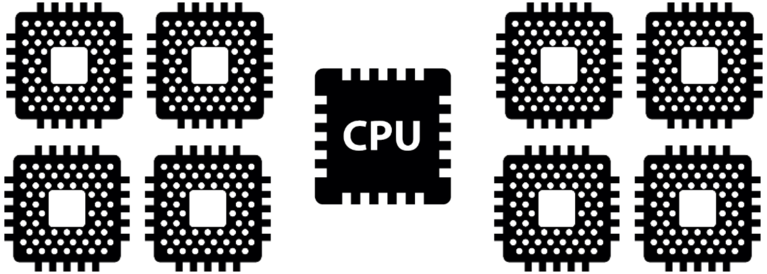


Single Processor

Simple to program *but* may not provide enough performance

especially

as Moore's Law frequency/power scaling is slowing



Multi-Processor

Additional processors can process expanded workloads *but a*dds complexity to system design and programming:

(i) Divide workload into kernel programs for distribution across available processors

(ii) Synchronize use of compute and memory resources

(iii) Communicate intermediate data and results

Open standard APIs and languages can help manage this complexity

What is an Open Interoperability Standard?

Open Standards

INTEROPERABILITY is precisely specified COMMUNICATION

E.g., software to hardware, client to server

OPEN standard specifications are created through multi-company cooperation under an agreed IP framework

Open standard specifications PLUS conformance tests enable MULTIPLE CONSISTENT IMPLEMENTATIONS to meet the needs of diverse markets, price points, and use cases

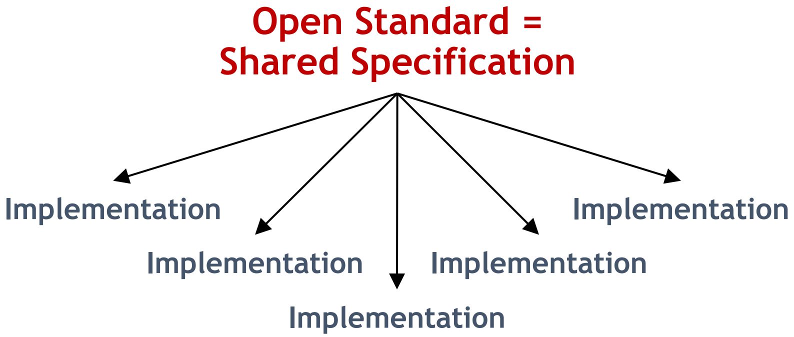
Open Source

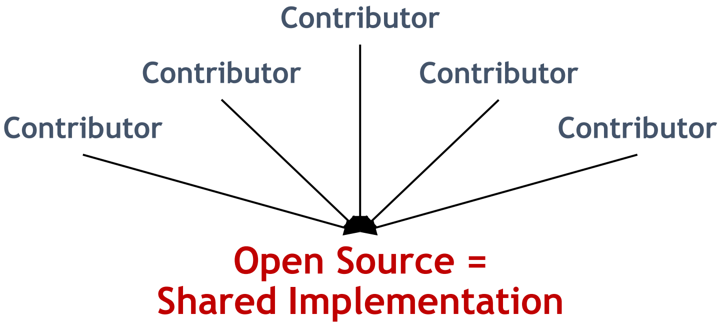
Open-source projects are created through multi-company cooperation and software effort via a contribution license

Design governance ranges from narrow to broad

Depending on project's history and purpose

Open *standards* often use open *source* to share the development effort for sample implementations, tools, samples, conformance tests, validators, etc.





Often used for HARDWARE APIs to enable competition between diverse implementations without fragmentation

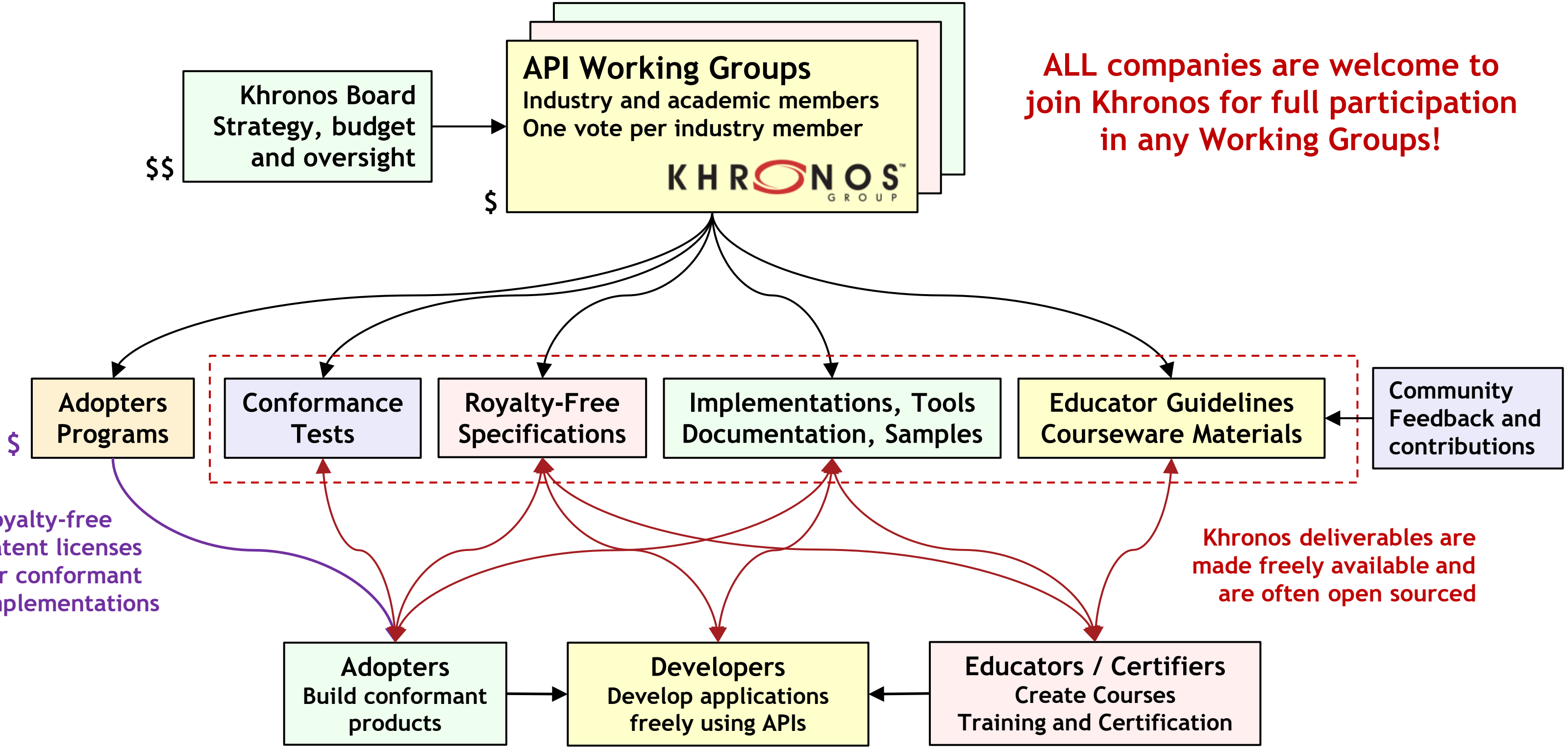
Often used for SOFTWARE libraries and languages to share effort AND gain consistency through a single implementation

Benefits of Open Interoperability Standards

- **Proven solutions - often available royalty free**
  - Leveraging significant industry effort and industry expertise
- **Benefits for hardware and software developers**
  - Cross-platform application portability and reusability
  - Industry-wide ecosystem of tools and libraries
- **Benefits for embedded markets**
  - Decoupled software and hardware for streamlined development, integration

Why Open Standards?	**Expand Commercial Opportunity** Network effect of compatible products & services	**Reduce Costs** Share design effort and drive increased volume
	Avoid Market Friction Reduce fragmentation and confusion	**Speed Time to Market** Leverage proven functionality and testing
When?	**When Technologies are Proven** Avoid R&D by standards committee	**Consensus Need** Downsides of no available standard widely obvious
How?	**Multi-company Governance to Build Trust** Avoid single-company control or dependency	**Well-defined IP Rights Policy** Royalty-free standards drive wide adoption
	Innovation through Flexible Extensibility Extensions meet timely customer & market needs	**Innovation through Careful Abstraction** Freedom to innovate implementation details

Khronos Cooperative Framework



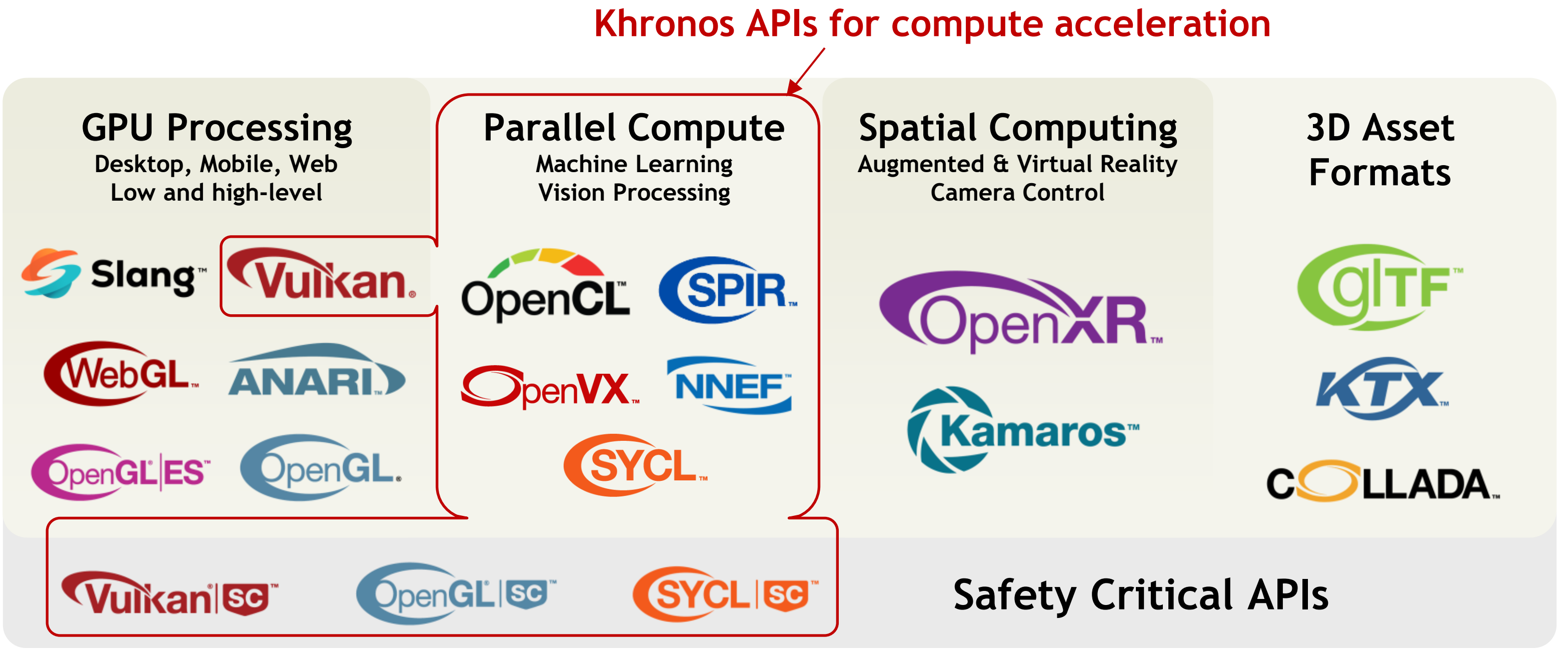
Khronos Board
Strategy, budget and oversight
$$
API Working Groups
Industry and academic members
One vote per industry member
KHRONOS GROUP
$
Adopters Programs
$
Conformance Tests
Royalty-Free Specifications
Implementations, Tools Documentation, Samples
Educator Guidelines Courseware Materials
Community Feedback and contributions
Adopters
Build conformant products
Developers
Develop applications freely using APIs
Educators / Certifiers
Create Courses
Training and Certification

ALL companies are welcome to join Khronos for full participation in any Working Groups!

Royalty-free patent licenses for conformant implementations

Khronos deliverables are made freely available and are often open sourced

Khronos Active Standards



Khronos APIs for compute acceleration

GPU Processing
Desktop, Mobile, Web
Low and high-level

Parallel Compute
Machine Learning
Vision Processing

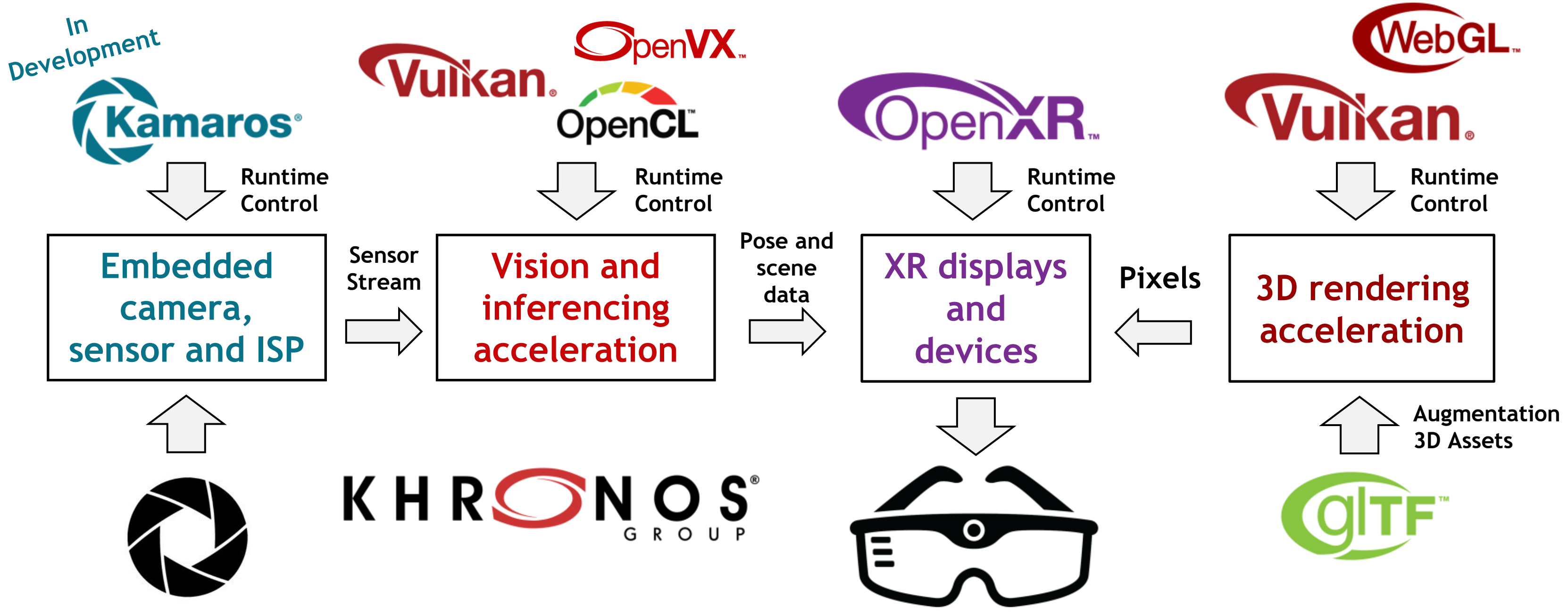
Spatial Computing
Augmented & Virtual Reality
Camera Control

3D Asset Formats

Safety Critical APIs

Spatial Computing APIs

Cameras are now the *only* part of the spatial computing pipeline not yet well-served by Khronos open API standards



In Development

Kamaros → Runtime Control → Embedded camera, sensor and ISP

Sensor Stream →

Vulkan, OpenVX, OpenCL → Runtime Control → Vision and inferencing acceleration

Pose and scene data →

OpenXR → Runtime Control → XR displays and devices

← Pixels

WebGL, Vulkan → Runtime Control → 3D rendering acceleration

glTF → Augmentation 3D Assets

Executing OpenCL Programs

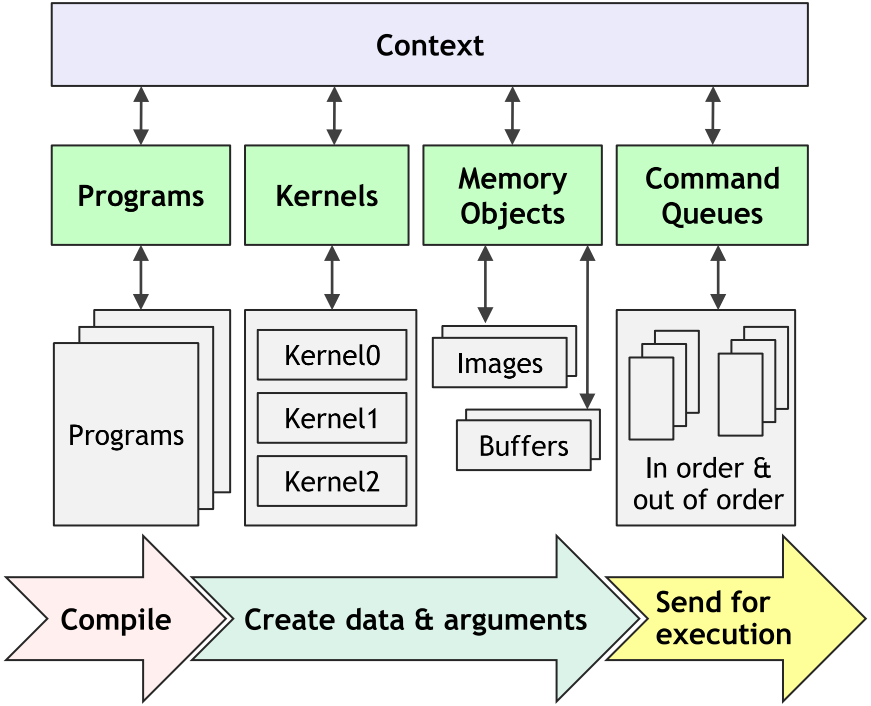
A ***kernel*** program is the basic unit of executable code (similar to a C function)

An OpenCL ***program*** is a collection of kernels and functions

An OpenCL ***command queue*** is used by the host application to send kernels and data transfer functions to a device for execution.

By *enqueueing* commands into a command queue, kernels and data transfer functions may execute asynchronously and in parallel with application host code

As an open standard, OpenCL is a well proven design, available from many silicon vendors with an extensive ecosystem of available tools, compilers, libraries and educational materials



C++ for OpenCL

Open-Source Compiler Front-end

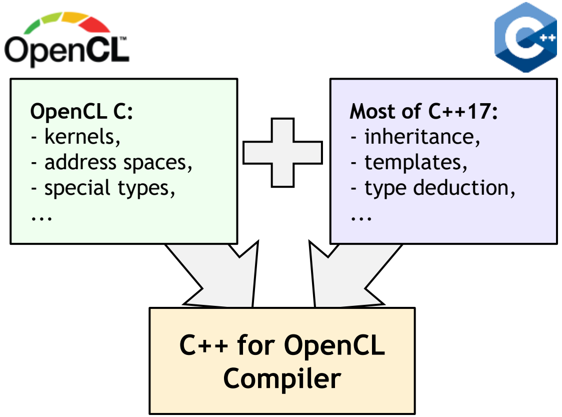
Replaces the OpenCL C++ kernel language spec
Official release published in OpenCL-Docs repo

Enables full OpenCL C and most C++17 capabilities

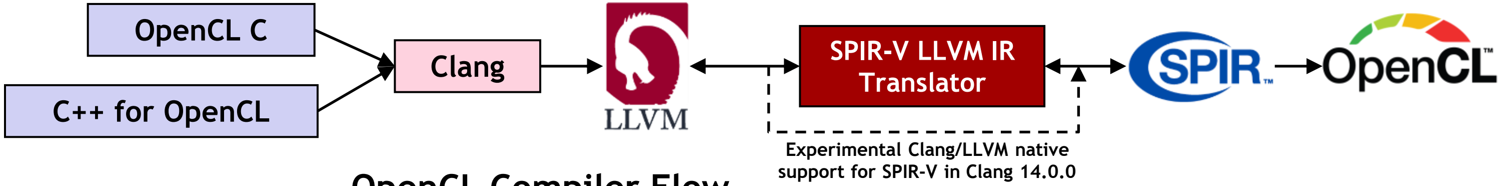
OpenCL C code is valid and fully compatible
Enables gradual transition to C++ for existing apps

Supported in Clang since release 9.0

Generates SPIR-V 1.0 plus SPIR-V 1.2 where necessary
Online compilation via cl_ext_cxx_for_opencl extension



Check it out in
Compiler Explorer



OpenCL C
C++ for OpenCL
Clang
LLVM
SPIR-V LLVM IR Translator
SPIR
OpenCL
Experimental Clang/LLVM native support for SPIR-V in Clang 14.0.0

OpenCL Compiler Flow

OpenCL SDK Upgrades

Open-source OpenCL SDK includes all components to develop OpenCL applications

OpenCL Headers (include/api)
OpenCL C++ bindings (include/cpp)
OpenCL Utility Libraries (include/utils)
Build system and CI

Documentation and Sample Code

OpenCL Guide
Code samples (samples/)
Documentation (docs/)

Loader and Layers

SDK and Layers Tutorial

Khronos funds SDK upgrades

Community contributions also welcome!

OpenCL Guide

This guide is written to help developers get up and running quickly with the Khronos® Group's OpenCL™ programming framework. It is an introductory read that covers the background and key concepts of OpenCL, but also contains links to more detailed materials that developers can use to explore the capabilities of OpenCL that interest them most.

Overview and Introduction

- What is OpenCL?
- How does OpenCL Work?
- How does OpenCL Compare to Other Khronos Standards?
- Programming OpenCL Kernels
- OpenCL Programming Model
- C++ for OpenCL
- OpenCL 3.0
- Tools for Offline Compilation of OpenCL Kernels
- Additional Resources

Spring 2022 SDK Updates

More details in the SDK Blog

Enhanced Cmake-based build system
Subprojects and components

Binary releases
Tagged SDK versions

Enhanced SDK documentation
In OpenCL Guide

OpenCL 3.0 Samples
C, C++, Python and Ruby

Utility Libraries
For loading kernel source and binary files

Coming Soon!

Upstream to Kitware's FindOpenCL.cmake
Enhances OpenCL:: namespace

Packaging and Distribution Support
Build packages from the SDK
Package newer versions of OpenCL
Ease cross-platform installation, including PPAs

Enhanced SDK Validation Layers
Object lifetime, Input parameters, SPIR-V

OpenCL State-of-the Union

- **OpenCL 3.0 adoption is strong and growing**
  - 14 OpenCL 3.0 Adopters, second only to OpenCL 1.2 (Vulkan 1.3 has 13 Adopters)
- **Significant open-source activity**
  - Mesa Rusticl for Linux
  - clang/LLVM compilation front-ends
  - Layered implementations clspv and Ancle over Vulkan, OpenCLon12 over DX12
- **OpenCL is a popular substrate layer for higher-level models, especially SYCL**
  - The second most common offload path, after CUDA, but before SYCL, Vulkan, HIP
- **Emerging acceptance of OpenCL as compute layer over Vulkan**
  - Especially for ML, simpler programming model, more language flexibility, e.g., pointers
- **Regular (roughly) quarterly Releases with new unified specification format!**
  - 3.0.16 is released for IWOCL 2024 with External Memory and Semaphores finalized
- **Active extension pipeline - driven by mobile, embedded and desktop markets**
  - Recordable Command Buffers, Cooperative Matrix, Unified Shared Memory, YUV Images, Tiling Controls…

Shout out to Ben Ashbaugh @ Intel!

Specification editor

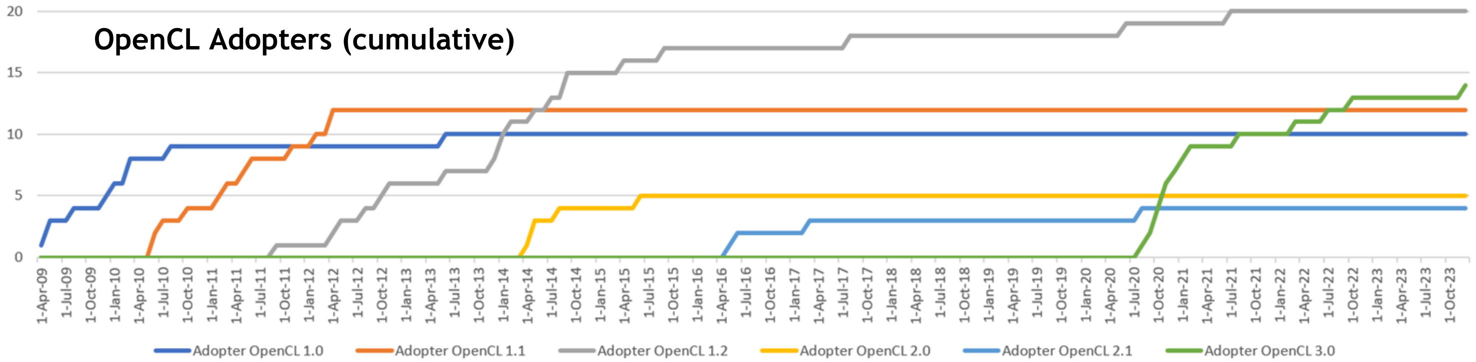
Significant guidance to working group technical direction

Also Nikhil Joshi @ NVIDIA

Memory TSG chair to offload significant detailed discussions from the main working group

OpenCL™

OpenCL 3.0 Adoption



Currently 14 OpenCL 3.0 Adopters, 9 already submitted conformant products - *second only to OpenCL 1.2*

https://www.khronos.org/conformance/adopters/conformant-products/opencl

arm ✓ codeplay Google ✓ HUAWEI Imagination ✓ intel ✓ MESA ✓ Microsoft

NVIDIA ✓ QNX QUALCOMM ✓ SAMSUNG ✓ VeriSilicon ✓ Tampere University

✓ Shipping OpenCL 3.0 Conformant Implementations

AMD KALRAY MARVELL ST life.augmented TEXAS INSTRUMENTS Adopters of previous OpenCL Versions

Apps, Libraries and Engines using OpenCL

Pervasive, cross-vendor, open standard for low-level heterogeneous parallel programming

https://en.wikipedia.org/wiki/List_of_OpenCL_applications

Desktop Creative Apps

F.
Adobe
otoy
REDCINE-X PRO
acdsee
DASSAULT SYSTEMES
Modo
SONY
SideFX
darktable
AUTODESK
CHAOSGROUP
Vegas Pro
Capture One
SILHOUETTE
CyberLink
ptc
GIMP
ArcSoft
RADEON PRORENDER
Blackmagicdesign
REALFLOW
LuxCoreRender OPEN SOURCE PHYSICALLY BASED RENDERER

Parallel Languages

SYCL
OpenACC DIRECTIVES FOR ACCELERATORS
aparapi
PyOpenCL

Machine Learning Libraries and Frameworks

Intel OpenVINO
Intel clDNN
Xiaomi MACE Mobile AI Compute Engine
SYCL-DNN
Caffe
NNAPI
Acuity ML/AI Software Foundation
VeriSilicon
Qualcomm Neural Processing SDK for AI
Synopsis MetaWare EV
TI DL Library (TIDL)
Arm Compute Library
飞桨 PaddlePaddle
MNN Mobile Neural Network

Molecular Modelling Libraries

siremol.org
FOLDING @HOME
ForceBalance
OpenMM
CHARMM Chemistry at HARvard Macromolecular Mechanics
GROMACS FAST. FLEXIBLE. FREE.

Machine Learning Compilers

MLIR
GLOW
tvm
plaidML

Vision, Imaging and Video Libraries

FAST
OpenVX
Halide
VisionCpp
Metashape
OpenCV
FFmpeg

Math and Physics Libraries

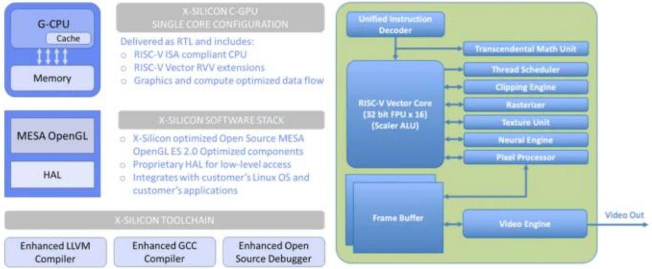
BULLET PHYSICS LIBRARY
GNU Octave
Wolfram *Mathematica*
ArrayFire C, C++, Fortran
Matlab

Linear Algebra Libraries

SYCL-BLAS
ViennaCL
CLBlast

Recent AI/RISC-V Use of Khronos Standards

- **SiM.ai Chip Startup Raises $70 Million to Quicken AI on Cars and Robots**
  - https://www.msn.com/en-us/money/other/chip-startup-raises-70-million-to-quicken-ai-on-cars-and-robots/ar-BB1l48bI
  - SiMa.ai is one of a growing number of startups trying to perfect hardware for a future where AI is mainstream. The startup has enlisted more than 50 customers for its first chip, which mainly targeted computer vision, and is now working on a second generation. The new chip is scheduled for release in the first quarter of next year. SiMa.ai’s products support various types of open standards including Linux and OpenCL
- **Axelera Uses oneAPI Construction Kit to Rapidly Enable Open Standards Programming for the Metis AIPU**
  - https://www.edge-ai-vision.com/2024/04/axelera-uses-oneapi-construction-kit-to-rapidly-enable-open-standards-programming-for-the-metis-aipu/
  - At Axelera, we therefore believe that the answer to the question of how to best bushwhack through the accelerator jungle is to embrace open standards, such as OpenCL and SYCL. OpenCL and SYCL are open standards defined by the Khronos Group. They define an application programming interface (API) for interacting with all kinds of devices as well as programming languages for implementing compute kernels to run on these devices.
- **New RISC-V microprocessor can run CPU, GPU, and NPU workloads simultaneously leveraging Khronos OpenGL**
  - https://www.tomshardware.com/pc-components/cpus/former-silicon-valley-vets-create-risc-v-microprocessor-that-can-run-cpu-gpu-and-npu-workloads-simultaneously



Delivered as RTL and includes:

- RISC-V ISA compliant CPU
- RISC-V Vector RVV extensions
- Graphics and compute optimized data flow

- X-Silicon optimized Open Source MESA OpenGL ES 2.0 Optimized components
- Proprietary HAL for low-level access
- Integrates with customer’s Linux OS and customer’s applications

Layered Open-Source OpenCL

	clspv+clvk	clspv+Angle	clspv on Apple	Rusticl on Zink	CLon12
OS	Android Chrome OS	Android Linux in test Other OS coming	macOS/iOS Prototype	Linux Windows Raspberry Pi OS	Windows
Compiler	Clang+clspv	Clang+clspv	Clang+clspv	Clang+LLVM + llvm-spirv	Clang+LLVM + llvm-spirv + Mesa SPIR-V to DXIL
IR Paths	OpenCL SPIR-V to Vulkan SPIR-V (inc LLVM IR all instances..)	OpenCL SPIR-V to Vulkan SPIR-V	OpenCL SPIR-V to Vulkan SPIR-V	OpenCL SPIR-V to NIR to Vulkan SPIR-V	OpenCL SPIR-V to DXIL
API Translation	clvk	Angle	clvk		CLon12
Intermediate API			MoltenVK	Zink	
HAL API	Vendor Vulkan	Vendor Vulkan	Metal	Vendor Vulkan	Vendor DX12
SPIR-V Ingestion	Yes	Not yet	Yes	Yes	Yes

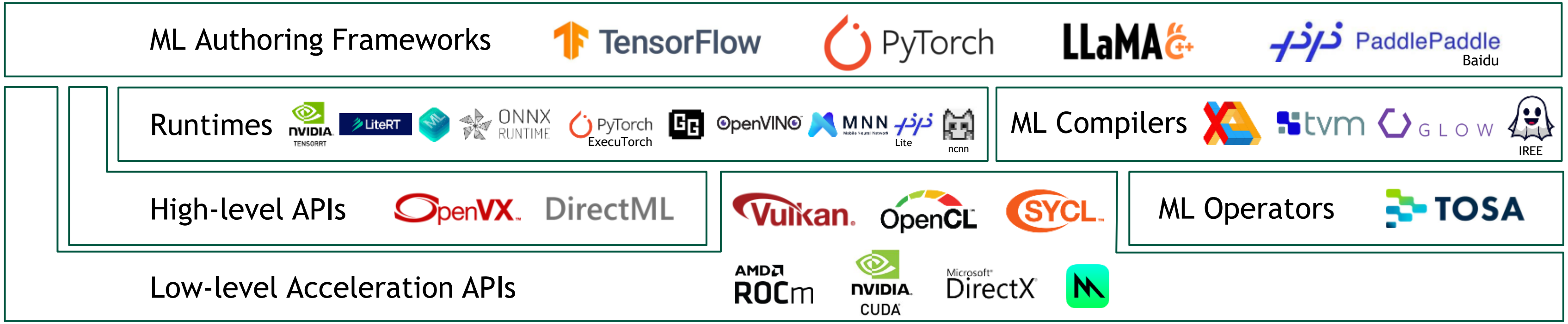
Open-Source OpenCL

UNDER CONSTRUCTION

	POCL	**Rusticl**	Intel Neo	AMD drivers	OneAPI Construction Kit
OS	Android Chrome OS	Linux Windows			
Compiler	Clang + clspv	Clang+LLVM + llvm-spirv			
HAL API	Vendor Vulkan	Iris - Intel Nouveau - NVIDIA RadeonSI/R600 - AMD Panfrost - Arm Mali Asahi - Apple Silicon			
Supported Device Types					
SPIR-V Ingestion	Yes	Yes			

Accelerated ML Industry Discussions

- Do intermediate runtimes or compilers deliver best performance?
- Do high-level or low-level acceleration APIs deliver the best performance?
- What functionality should APIs provide for effective tensor acceleration?
- What is the most effective way to balance inferencing and other loads on a GPU?
- How can APIs provide acceleration across diverse hardware - such as GPUs and NPUs?
- Should the industry agree on a standard tensor operator set such as Arm’s TOSA?



ML Authoring Frameworks

Runtimes

ML Compilers

High-level APIs

ML Operators

Low-level Acceleration APIs

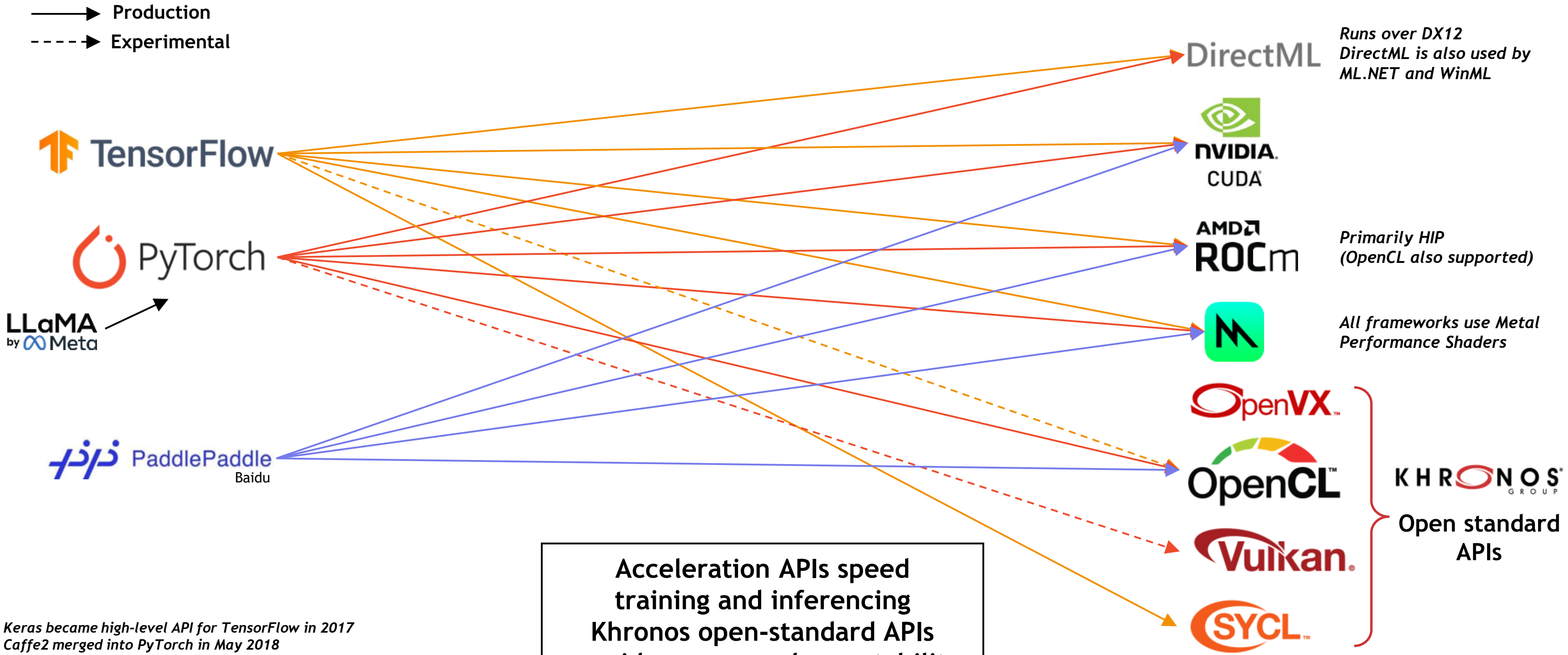
Native machine learning stack

Similar discussions happening in the JavaScript stack for machine learning in the Web

Machine Learning Acceleration APIs

Open-Source Frameworks

Acceleration APIs



Runs over DX12
DirectML is also used by ML.NET and WinML

Primarily HIP
(OpenCL also supported)

All frameworks use Metal Performance Shaders

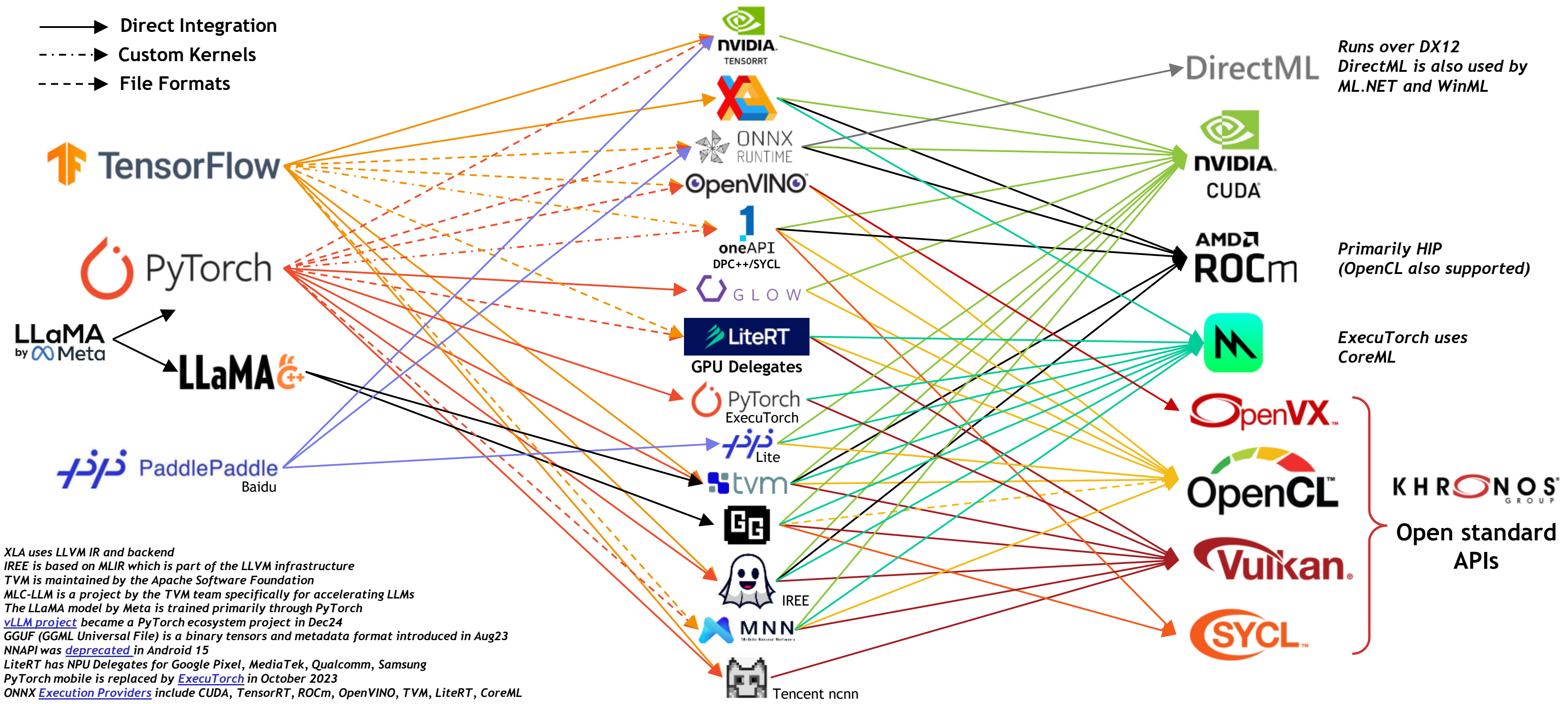
Open standard APIs

Acceleration APIs speed training and inferencing
Khronos open-standard APIs provide cross-vendor portability

- *Keras became high-level API for TensorFlow in 2017*
- *Caffe2 merged into PyTorch in May 2018*
- *PyTorch is maintained by the Linux Foundation*
- *Experimental PyTorch direct Vulkan integration*

Machine Learning Acceleration APIs

Open-Source Frameworks | **Compilers, Runtimes and Libraries** – For acceleration flexibility, customization and optimization - some in open source | **Acceleration APIs**



Runs over DX12
DirectML is also used by ML.NET and WinML

Primarily HIP (OpenCL also supported)

ExecuTorch uses CoreML

- *XLA uses LLVM IR and backend*
- *IREE is based on MLIR which is part of the LLVM infrastructure*
- *TVM is maintained by the Apache Software Foundation*
- *MLC-LLM is a project by the TVM team specifically for accelerating LLMs*
- *The LLaMA model by Meta is trained primarily through PyTorch*
- *vLLM project became a PyTorch ecosystem project in Dec24*
- *GGUF (GGML Universal File) is a binary tensors and metadata format introduced in Aug23*
- *NNAPI was deprecated in Android 15*
- *LiteRT has NPU Delegates for Google Pixel, MediaTek, Qualcomm, Samsung*
- *PyTorch mobile is replaced by ExecuTorch in October 2023*
- *ONNX Execution Providers include CUDA, TensorRT, ROCm, OpenVINO, TVM, LiteRT, CoreML*

Machine Learning Acceleration APIs

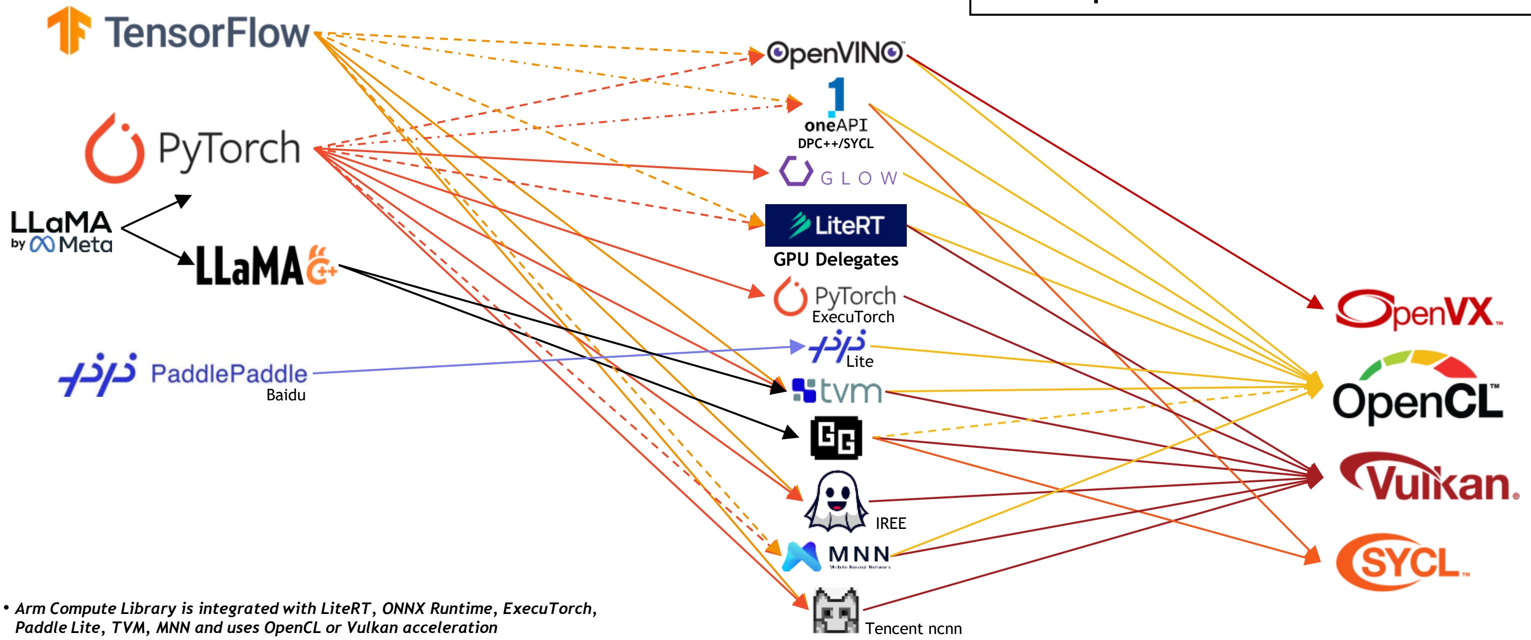
Open-Source Frameworks | **Compilers, Runtimes and Libraries** For acceleration flexibility, customization and optimization - some in open source | **Acceleration APIs**

───▶ Direct Integration

-·-·-▶ Custom Kernels

- - - -▶ File Formats

Using open standard APIs to access cross-vendor acceleration can reduce ecosystem complexity and development costs for machine learning developers



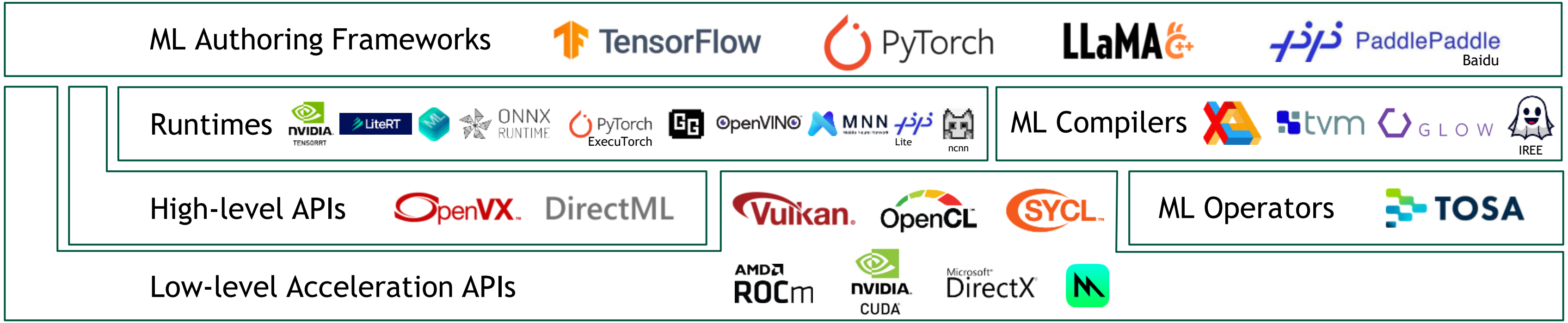
- *Arm Compute Library is integrated with LiteRT, ONNX Runtime, ExecuTorch, Paddle Lite, TVM, MNN and uses OpenCL or Vulkan acceleration*

Framework	Developer	Primary Focus	Hardware APIs	Key Strengths	Target Hardware	Notable Features
TensorRT	NVIDIA	GPU inference optimization	CUDA cuDNN TensorRT API	High performance on NVIDIA GPUs Layer fusion Precision calibration	NVIDIA GPUs	Dynamic Tensor Core utilization Automatic mixed precision Graph optimization
ONNX Runtime	Microsoft/Community	Cross-platform inference	CUDA OpenCL | OpenVINO TensorRT | CoreML | DirectML	Framework agnostic Wide hardware support Extensible architecture	CPU GPU Specialized hardware	Custom operator support Automatic performance tuning Multiple execution providers
Apache TVM	Apache Foundation	End-to-end compilation	CUDA | OpenCL Vulkan | Metal ROCm | OneAPI	Auto-tuning Hardware flexibility Optimization scheduling	CPU GPU Accelerators	Automated optimization Hardware abstraction Rich IR system
IREE	Google	MLIR-based compilation	Vulkan CUDA | Metal | ROCM	End-to-end compilation MLIR integration Multiple backend support	CPU GPU Accelerators	Vulkan/CUDA support MLIR dialects Lightweight runtime
ncnn	Tencent	Mobile inference	Vulkan ARM NEON	ARM optimization Lightweight No dependencies	Mobile CPUs Mobile GPUs	NEON optimization Vulkan support Small footprint
MNN	Alibaba	Mobile deployment	OpenCL | Vulkan Metal ARM NEON	Multi-backend support Model encryption Memory optimization	Mobile/Edge devices	Metal/OpenCL support Quantization Profile tools
GGML	Georgi Gerganov	CPU-first LLM inference	CUDA Metal CPU SIMD (AVX/NEON)	Memory efficiency CPU optimization Quantization focus	CPUs	4-bit quantization Memory mapping SIMD optimization
Glow	Meta	Neural network compilation	OpenCL CUDA Habana API	Memory planning Graph optimization Hardware flexibility	Multiple accelerators	AOT/JIT compilation Operator fusion Custom hardware support
OpenVINO	Intel	Intel hardware optimization	OpenCL oneAPI / Intel DPC++ OpenVX	Intel hardware support Vision workload focus Comprehensive tooling	Intel CPUs, GPUs Intel VPUs, FPGAs	Model optimization INT8 quantization Vision acceleration
PPLite	MEGVII	High-performance inference	CUDA ARM NEON x86 SIMD	Kernel optimization Auto-tuning Memory management	Multiple platforms	Operator fusion Dynamic shapes Profile tools
oneAPI	Intel	Unified programming	SYCL/DPC++ Level Zero OpenCL	Cross-architecture Comprehensive libraries Single codebase	Intel hardware ecosystem	DPC++ language Domain libraries Hardware abstraction

Embedded Machine Learning Acceleration

Name	Type	Authoring Frameworks	Acceleration APIs
Cadence Xtensa Neural Network Compiler (XNNC)	Compiler	TensorFlow, PyTorch, ONNX	OpenCL OpenVX
CEVA Deep Neural Network compiler (CDNN)	Compiler	TensorFlow, PyTorch, Caffe, ONNX	OpenCL OpenVX
Synopsis MetaWare EV	Runtime	TensorFlow, PyTorch, Caffe, ONNX	OpenCL OpenVX
Texas Instruments DL Library (TIDL)	Runtime	TensorFlow, PyTorch, Caffe, ONNX	OpenCL OpenVX
VeriSilicon Acuity	Runtime	TensorFlow, PyTorch, Caffe, ONNX	OpenCL OpenVX
Xiaomi Mace	Runtime	TensorFlow, Caffe, ONNX	OpenCL Vulkan
Xilinx Vitis AI	Runtime	TensorFlow, PyTorch	OpenCL Native

OpenCL and OpenVX are the open standard APIs of choice for inferencing acceleration in embedded (and often mobile) devices



ML Authoring Frameworks

Runtimes

ML Compilers

High-level APIs

ML Operators

Low-level Acceleration APIs

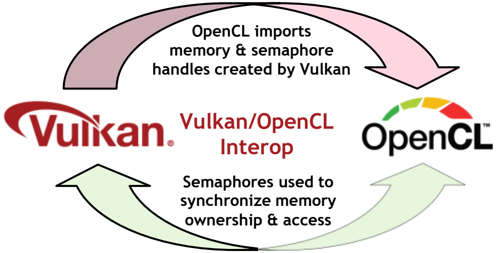
OpenCL Specification Releases and Roadmap

OpenCL 3.0.16 shipped on April 4th, 2024

Continues the regular release cadence for new functionality and bug fixes

External memory objects and semaphores for external sharing and Interop finalized

Kernel Clock extension provisional release


OpenCL imports memory & semaphore handles created by Vulkan
Vulkan
Vulkan/OpenCL Interop
OpenCL
Semaphores used to synchronize memory ownership & access

OpenCL Extension Pipeline

Provisional, EXT and Vendor extensions - candidates for final ratification

We are listening to your input!

Support C++ for OpenCL (EXT)
Command Buffer Record/Replay (provisional)
Unified Shared Memory
Floating Point Atomics
Required Subgroup Size
Generalized Image from buffer
Image Tiling Controls

YUV Multi-planar Images
Cross-workgroup Barriers
Cooperative Matrices
Timeline Semaphores
32 and 64-length vectors
Indirect Dispatch
ML Operations

Strategic Discussion Topics #1

Fragmentation

- **Reduce desktop fragmentation - primarily lack of universal SPIR-V ingest**
  - Preventing usage of newer features in many applications
  - Layered implementations to the rescue?
  - Promote the idea of SPIR-V ingestion front-end to LLVM
  - Leverage Microsoft's SPIR-V in LLVM?
- **Provide more support and encouragement for layered OpenCL implementations?**
  - Clspv/Ancle, Microsoft OpenCLon12, Rusticl/Zink
  - Does Rusticl over Zink on MoltenVK work for OpenCL on Apple?
- **Work to close the SPIR-V shader/kernel 'schism'**
  - 'Schism Summit' for solution pooling and brainstorming for resolving differences?
  - Traverse Research (Jasper Bekkers, CTO), Embark Studios, Nicolai Hähnle from LLVM

Machine Learning

- **Tensorflow and PyTorch are not directly supporting OpenCL (just TensorFlow Lite)**
  - How can we improve this?
  - Increased investment in TVM as an open source path to other stacks?
- **Strengthen support for ML operations**
  - Cooperative matrix, Subgroup requirements for wavefront/warp size, Built-in Kernels
- **Effectiveness as target layer e.g., for SYCL and OpenMP**
  - Approach OpenMP for cooperation once we have SPIR-V backend in LLVM

Strategic Discussion Topics #2

- **Need for new core release?**
  - OpenCL 3.0 was released in September 2020 - need to show momentum
  - Move to subscription Adoption model?

Release cadence

- **Embedded market initiatives**
  - Market demand for Safety Critical Profile?
  - Backend for SYCL SC
  - OpenCL IS already being deployed in SC markets
  - OpenCL on Pi – maybe through Rusticl over Zink/Vulkan?
  - Leverage RISC-V and AutoSAR interest

Embedded

- **Managing GitHub reviews and working group sign offs is a bottleneck**
  - Cross Working Group trusted maintainer could be effective?

Logistics

- **Developer Engagement**
  - Re-invigorate Advisory Panel
  - 2024 Developer Survey? (see 2022 survey)

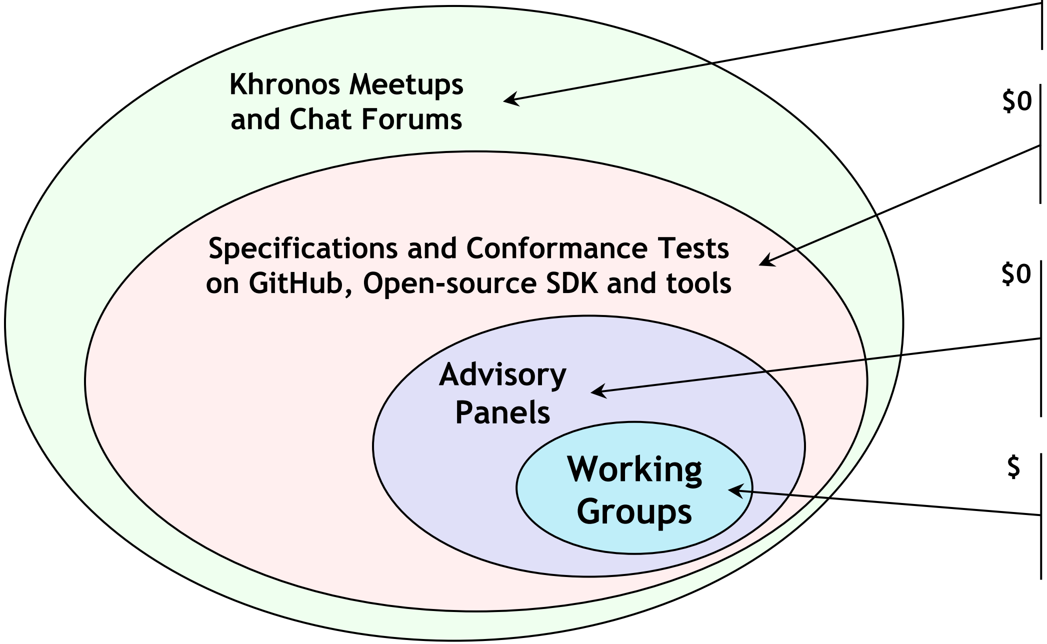
Developer Outreach

Discussion Topics

- **How can we reduce desktop fragmentation**
  - Need of universal SPIR-V ingest
  - Promote the idea of SPIR-V ingestion front-end to LLVM?
  - Leverage Microsoft's SPIR-V in LLVM?
  - Layered implementations may help?
- **Provide more support and encouragement for layered OpenCL implementations?**
  - Clspv/Ancle, Microsoft OpenCLon12, Rusticl/Zink
  - Does Rusticl over Zink on MoltenVK work for OpenCL on Apple?
  - OpenCL on Pi – maybe through Rusticl over Zink/Vulkan?
- **How encourage Tensorflow and PyTorch direct support for OpenCL (not just TensorFlow Lite)**
  - Increased investment in TVM as an open source path to other stacks?
  - Strengthen operations for ML: coop matrix, Subgroup requirements for wavefront/warp size, Built-in Kernels?
- **How increase effectiveness as target layer e.g., for SYCL and OpenMP**
  - Approach OpenMP for backend cooperation once we have SPIR-V backend in LLVM?
- **Market demand for OpenCL Safety Critical Profile?**
  - OpenCL IS already being deployed in SC markets
  - Backend for SYCL SC?

Your input and feedback is welcome!

Khronos Ecosystem Engagement



Khronos Meetups and Chat Forums

Specifications and Conformance Tests on GitHub, Open-source SDK and tools

Advisory Panels

Working Groups

$0 | Open to all!
https://community.khronos.org/
www.khr.io/slack

$0 | Spec fixes and suggestions. Open source contributions under repo’s CLA, typically Apache 2.0
https://github.com/KhronosGroup

$0 | Advisors under the Khronos NDA and IPR Framework can comment and contribute to requirements and draft specifications
https://www.khronos.org/advisors/

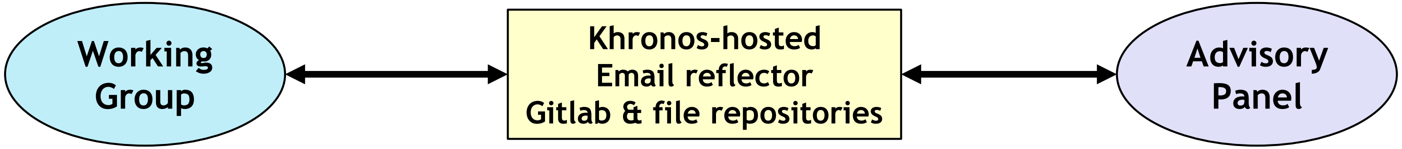
$ | Khronos members under Khronos NDA and IPR Framework participate and vote in working group meetings
https://www.khronos.org/members/

Khronos creates specifications and tools without an NDA as far as possible
BUT *hardware APIs* often need discussion of confidential technology roadmaps
This makes an NDA and IPR framework essential

Khronos Advisory Panel Structure

Working Group shares draft specifications and makes final decisions on specification design

Advisory Panel provides requirements, recommendations, detailed feedback, and design contributions



Khronos Members

- Any organization can join
- Membership Fee
- Covered by NDA and IP Framework

Under Khronos NDA

Working group can share draft specifications and accept detailed design contributions as Panel Members are covered by IP Framework

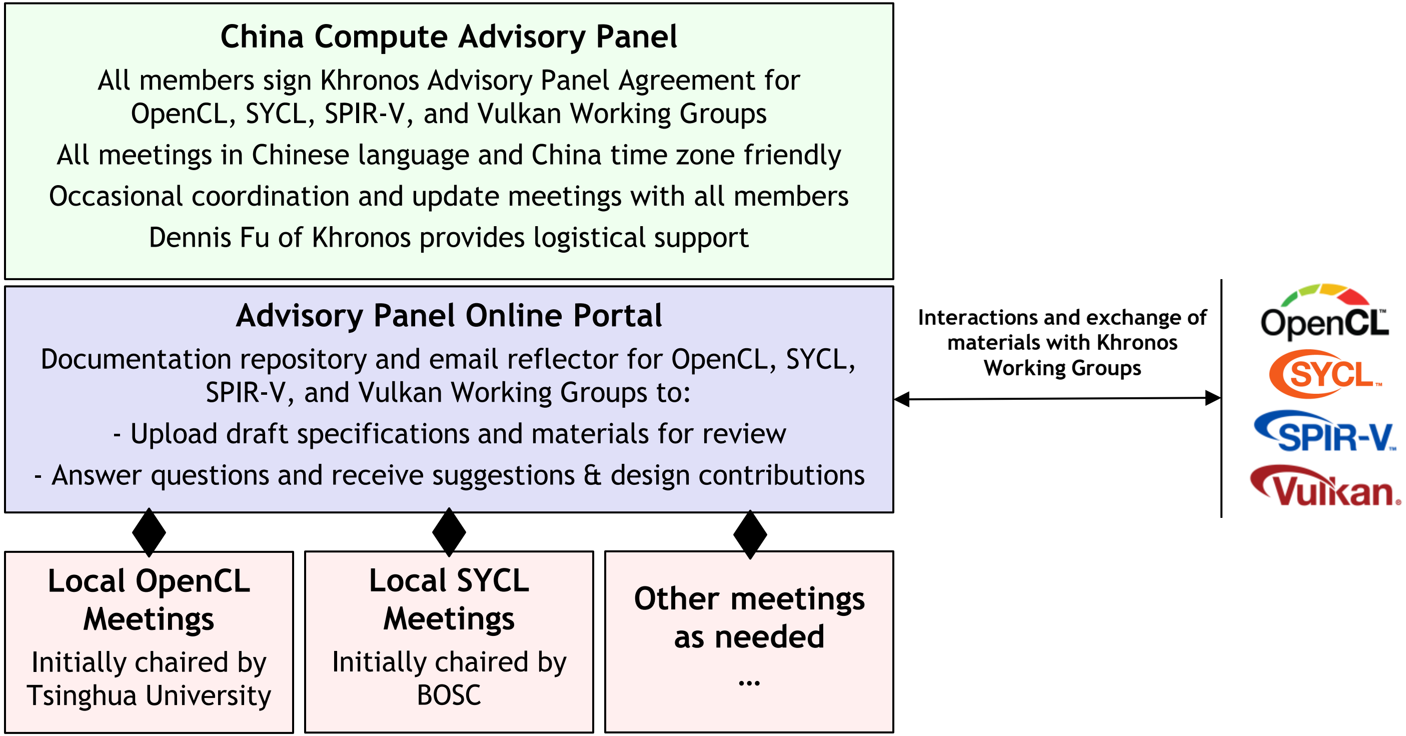
Panel Members

- Invited industry experts
- No membership fee
- Covered by NDA and IP Framework

Advisory Panel members are welcome to ‘upgrade’ to full Khronos membership at any time

Please reach out to opencl-chair@lists.khronos.org if you wish to apply

China Compute Advisory Panel Structure



China Compute Advisory Panel

All members sign Khronos Advisory Panel Agreement for OpenCL, SYCL, SPIR-V, and Vulkan Working Groups

All meetings in Chinese language and China time zone friendly

Occasional coordination and update meetings with all members

Dennis Fu of Khronos provides logistical support

Advisory Panel Online Portal

Documentation repository and email reflector for OpenCL, SYCL, SPIR-V, and Vulkan Working Groups to:

- Upload draft specifications and materials for review
- Answer questions and receive suggestions & design contributions

Interactions and exchange of materials with Khronos Working Groups

Local OpenCL Meetings

Initially chaired by Tsinghua University

Local SYCL Meetings

Initially chaired by BOSC

Other meetings as needed

…

Advisory Panel members can attend any local meetings
Working Group Chairs will occasionally join to assist in coordination