



# State of the Union

## OpenCL Working Group

Neil Trevett

Khronos President

OpenCL Working Group Chair

NVIDIA VP Developer Ecosystems

[ntrevett@nvidia.com](mailto:ntrevett@nvidia.com) | [@neilt3d](https://twitter.com/neilt3d)



# Primary Developments Since IWOCL 2020

## OpenCL 3.0 Finalized and Released

Including subgroups and embedded processor extensions

## Multiple OpenCL 3.0 Implementations Shipping

With many more Adopters in the pipeline

## Conformance Testing Improvements

>250 commits to the OpenCL test suite since IWOCL 2020

## Regular Releases

Including OpenCL 3.0.7 here at IWOCL with new extensions!

## OpenCL Guide Released and SDK Enhanced!

Tutorial on SDK Layers here at IWOCL

## C++ for OpenCL gaining momentum

Interaction with LLVM community deepening

## Increased activity in layered implementations

Microsoft's OpenCLon12 in addition to Google's clspv

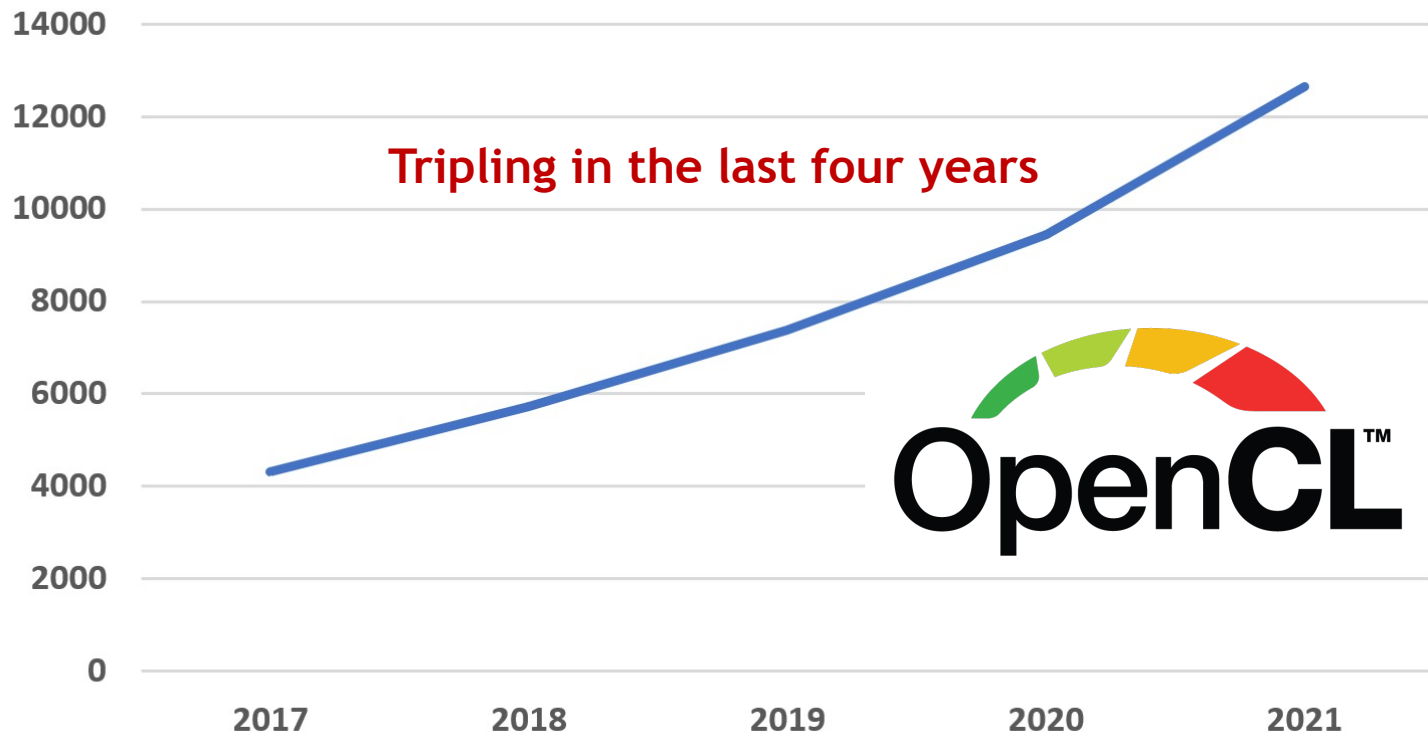
## Roadmap Discussions Underway

Building Advisory Panel Interactions



# OpenCL Open-Source Project Momentum

# OpenCL-based GitHub Repos



# OpenCL 3.0

## Increased Ecosystem Flexibility

All functionality beyond OpenCL 1.2 queryable  
Macros for optional OpenCL C language features  
Widely adopted extensions to be integrated into core

## OpenCL C++ for OpenCL

Open-source [C++ for OpenCL](#) front end compiler combines  
OpenCL C and C++17 replacing OpenCL C++ language spec

## Unified Specification

All versions of OpenCL in one specification for easier  
maintenance, evolution and accessibility

[Source](#) on Khronos GitHub for community feedback,  
functionality requests and bug fixes

## New Functionality

Subgroups with SPIR-V 1.3 in core (optional)  
Asynchronous DMA extension for embedded processors

## Easy OpenCL 3.0 migration for applications

OpenCL 1.2 applications - no change  
OpenCL 2.X applications - no code changes if all used  
functionality present  
Queries recommended for future portability

Released  
September  
2020



The OpenCL™ Specification

Khronos® OpenCL Working Group

Version v3.0.6, Fri, 18 Dec 2020 12:00:00 +0000: from git branch: master commit:  
e9a4d468b1a0a38c1e10b8af484bb2bb495e2b7

<https://www.khronos.org/registry/OpenCL/>

# OpenCL is Widely Deployed and Used

The industry's most pervasive, cross-vendor, open standard for low-level heterogeneous parallel programming

## Desktop Creative Apps



## Parallel Languages



## PyOpenCL

## Linear Algebra and FFT Libraries



## CLBlast

## VkFFT

## Machine Learning Libraries and Frameworks



## SYCL-DNN Caffe



## VeriSilicon

## Synopsis

## MetaWare EV

## TI DL Library (TIDL)

## Arm Compute Library

## Molecular Modelling Libraries



## Machine Learning Compilers



## GLOW



## Vision, Imaging and Video Libraries



## OpenVX



## Halide



## Math and Physics Libraries



## Wolfram Mathematica



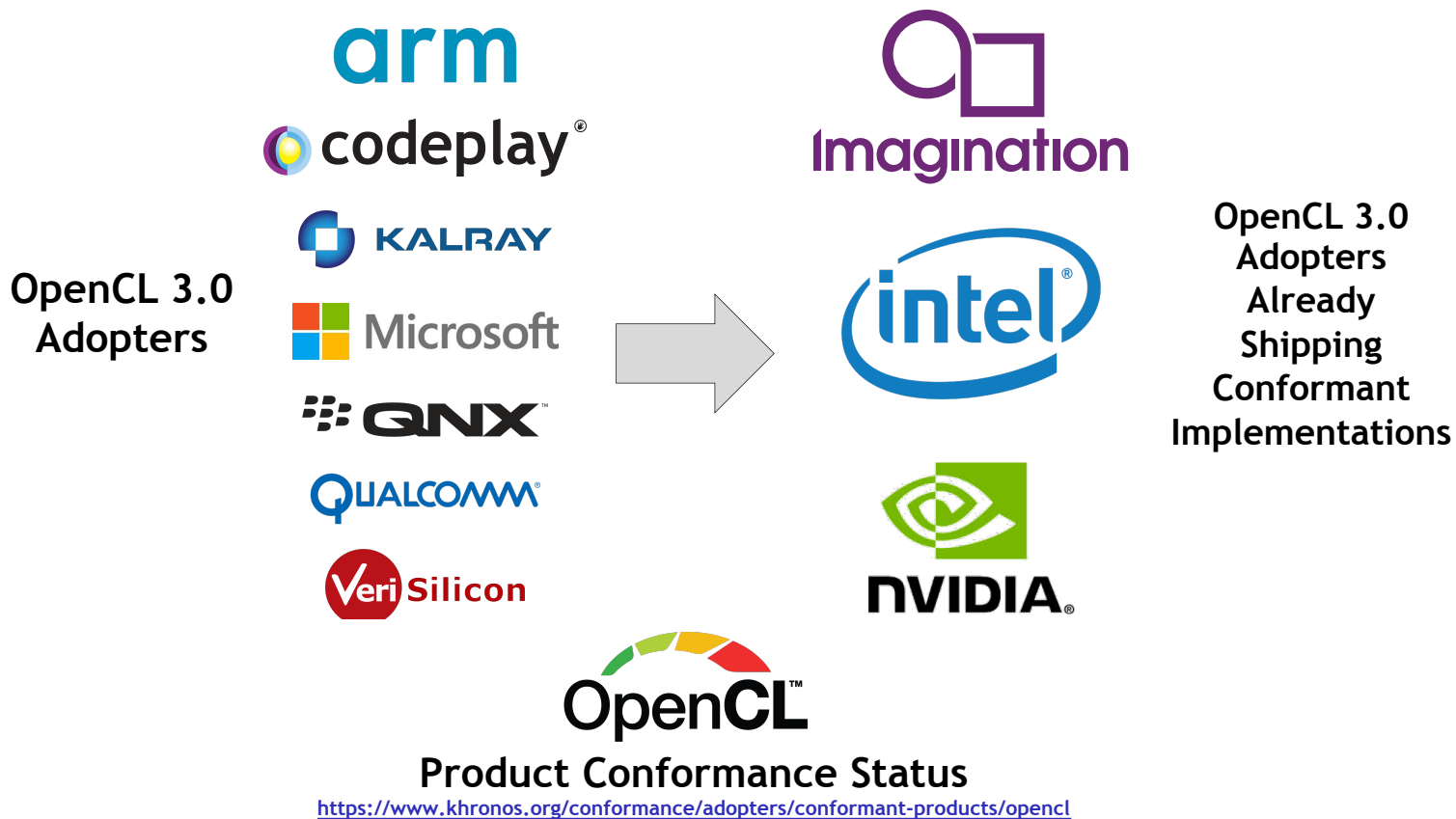
## Matlab



## Accelerated Implementations

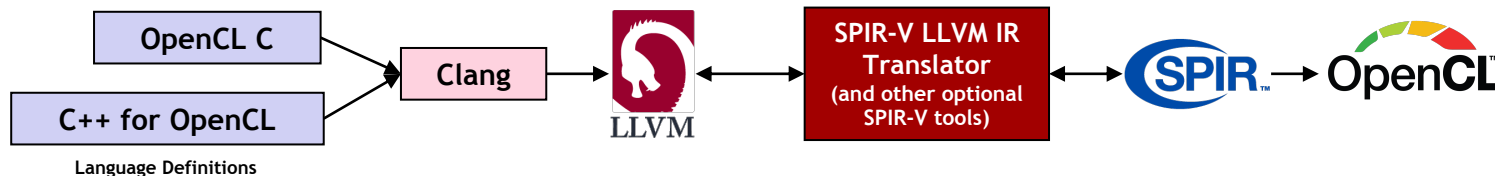
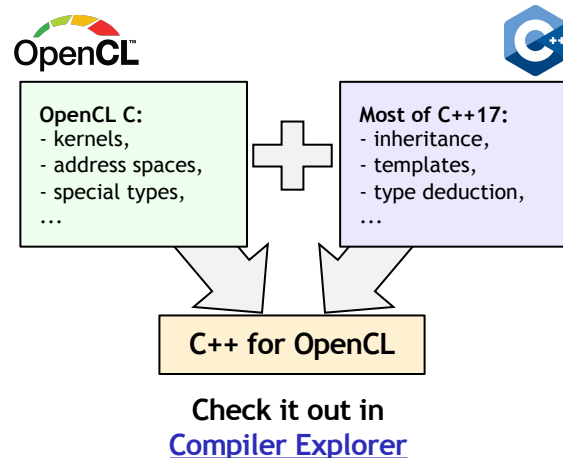
[https://en.wikipedia.org/wiki/List\\_of\\_OpenCL\\_applications](https://en.wikipedia.org/wiki/List_of_OpenCL_applications)

# OpenCL 3.0 Adoption



# C++ for OpenCL

- Open-Source Compiler Front-end
  - Replaces the OpenCL C++ kernel language spec
  - [Official releases](#) 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
  - [Language documentation](#)
- Supported in Clang since release 9.0
  - Generates SPIR-V 1.0 plus SPIR-V 1.2 where necessary
  - Full details are provided in [OpenCL-Guide](#)
- Online compilation via [cl\\_ext\\_cxx\\_for\\_openccl](#)



OpenCL Offline Compiler Flow

# Asynchronous DMA Extensions

## OpenCL embraces a new class of Embedded Processors

Many DSP-like devices have Direct Memory Access hardware

## Transfer data between global and local memories via DMA transactions

Transactions run asynchronously in parallel to device compute enabling wait for transactions to complete

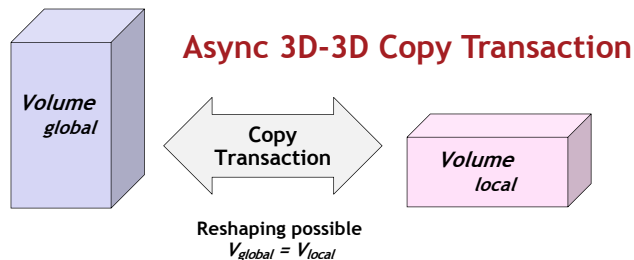
Multiple transactions can be queued to run concurrently or in order via fences

## OpenCL abstracts DMA capabilities via extended asynchronous workgroup copy built-ins

(New!) 2- and 3-dimensional async workgroup copy extensions support complex memory transfers

(New!) async workgroup fence built-in controls execution order of dependent transactions

New extensions complement the existing 1-dimensional async workgroup copy built-ins



## Async Fence controls order of dependent transactions

`async_copy1`  
`async_copy2`  
`async_fence`  
`async_copy3`



All transactions prior to `async_fence` must complete before any new transaction starts, without a synchronous wait

The first of significant upcoming advances in OpenCL to enhance support for embedded processors



# OpenCL 3.0.7 Release at IWOCL

## Second Maintenance release since OpenCL 3.0 in September 2020

Clarifications, formatting, bug fixes

Adds optional extensions

### **cl\_khr\_spirv\_extended\_debug\_info**

Enables SPIR-V modules to use the OpenCL.DebugInfo.100 extended instruction set

### **cl\_khr\_pci\_bus\_info**

Query PCI domain, bus, device, and function information for an OpenCL device

### **cl\_khr\_extended\_bit\_ops**

Adds OpenCL C built-in functions to insert, extract, and reverse bits in a bitfield

### **cl\_khr\_suggested\_local\_work\_size**

Adds a query for a suggested local work group size for a kernel running on an OpenCL device

### **cl\_khr\_spirv\_linkonce\_odr**

Enables LinkOnceODR SPIR-V link type to separately compile and link C++ programs

Specification  
available on the  
[OpenCL Registry](#)

# OpenCL SDK - In Development

- Bringing together all the components needed to develop OpenCL applications
  - OpenCL Headers (include/api)
  - OpenCL C++ bindings (include/cpp)
  - OpenCL Utility Libraries (include/utlis)
  - Build system and CI
- Other resources useful to OpenCL developers
  - OpenCL Guide
  - Code samples (samples/)
  - Documentation (docs/)
- Loader and Layers
  - Initial layers implemented
  - SDK and Layers Tutorial here at IWOCL
- Watch GitHub Repo for updates
  - Community contributions welcome!




<https://github.com/KhronosGroup/OpenCL-Guide>

**More Information at**

<https://github.com/KhronosGroup/OpenCL-SDK>

# API Layering

Enabled by growing robustness of open-source compiler ecosystem



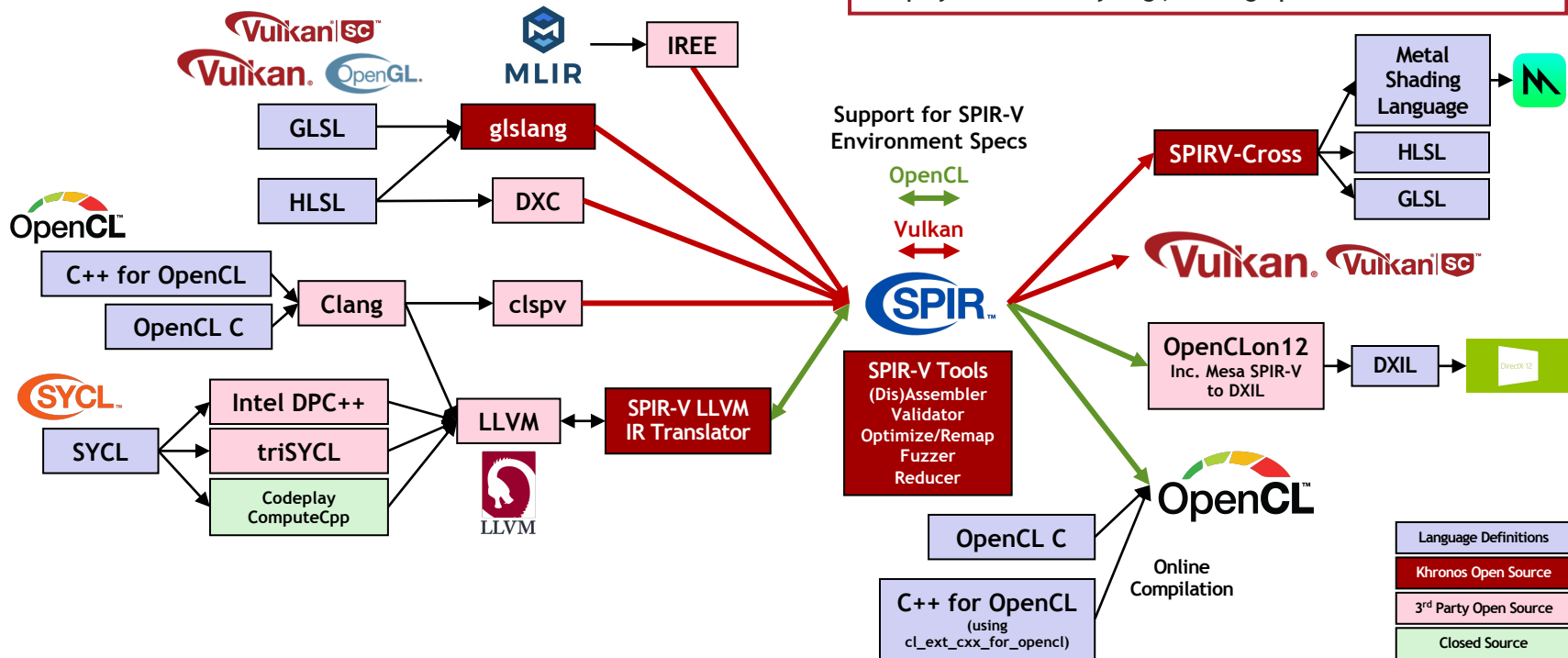
<i>Layers Over</i>	Vulkan	OpenGL	OpenCL	OpenGL ES	DX12	DX9-11
Vulkan		Zink	clspv clvk	GLOVE Angle	vkd3d-Proton vkd3d	DXVK WineD3D
OpenGL	gfx-rs Ashes			Angle		WineD3D
DX12	gfx-rs	Microsoft 'GLOn12'	Microsoft 'CLOn12'			Microsoft D3D11On12
DX9-11	gfx-rs Ashes			Angle		
Metal	MoltenVK gfx-rs			MoltenGL Angle		

**ROWS**  
Benefit  
Platforms by  
adding APIs  
Enable content  
without  
additional  
kernel level  
drivers

**COLUMNS** Benefit ISVs by making an API available everywhere  
Application deployment flexibility by fighting platform fragmentation  
Making an API available across multiple platforms even if no native drivers available

# SPIR-V Language Ecosystem

SPIR-V enables a rich ecosystem of languages and compilers to target low-level APIs such as Vulkan and OpenCL, including deployment flexibility: e.g., running OpenCL kernels on Vulkan



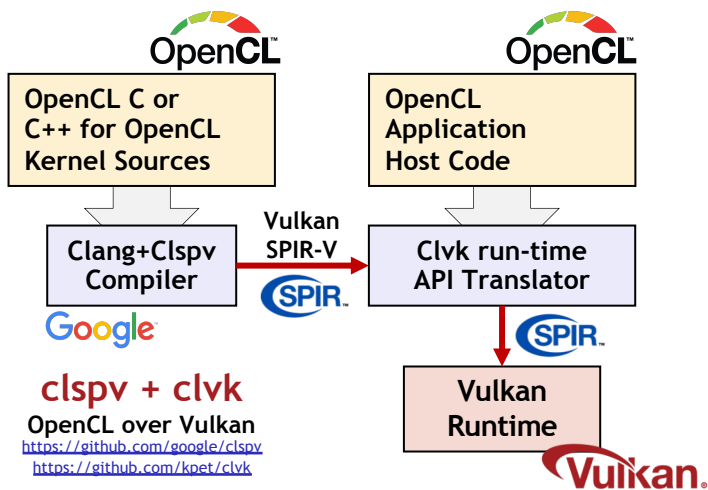
# Layered OpenCL Implementations

## clspv + clvk

clspv - Google's 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



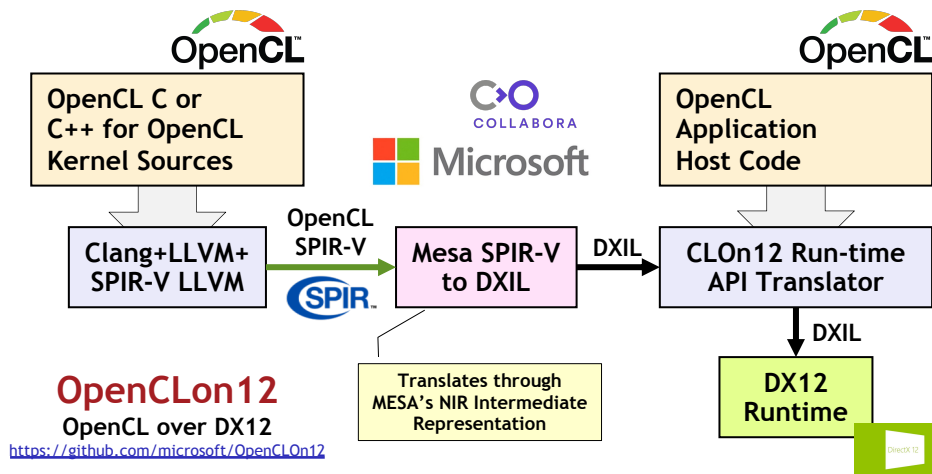
## OpenCLon12

Microsoft and COLLABORA

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

Leverages Clang/LLVM AND MESA

OpenCLon12 - OpenGL 3.3 over DX12 is already conformant



# OpenCL Roadmap

## Extensions in Ratification

Expected Public Release 2Q 2021

### cl\_khr\_integer\_dot\_product

Adds support for SPIR-V instructions and OpenCL C built-in functions to compute the dot product of vectors of integers

### External Sharing Extensions (Provisional)

#### cl\_khr\_external\_memory

Create OpenCL memory objects from OS-specific memory handles (similar to VK\_KHR\_external\_memory)

#### cl\_khr\_semaphore

Semaphore synchronization object that can be signaled and reset multiple times and signaled from outside OpenCL

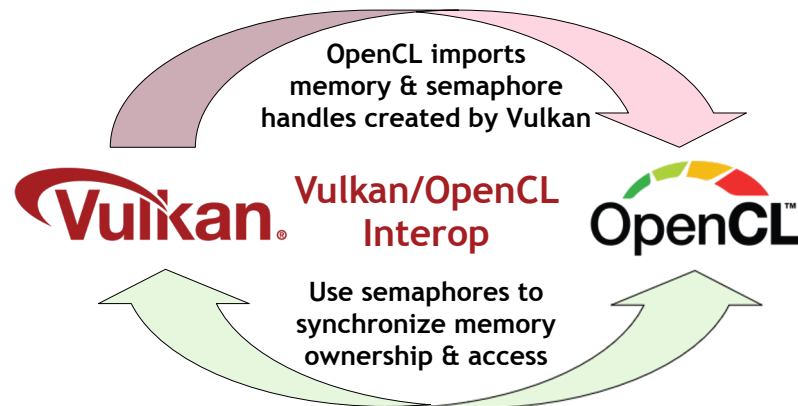
#### The cl\_khr\_external\_semaphore and cl\_khr\_external\_semaphore\_sync\_fd

Create OpenCL semaphore objects from OS-specific semaphore handles

#### cl\_khr\_vk\_sharing extension

Associate an OpenCL context with a Vulkan physical device

**The External Sharing Extensions are Provisional to enable developer feedback before finalization**



### External Sharing Extensions

Generic extensions to import external memory and semaphores exported by other APIs  
API-specific interop extensions e.g., Vulkan  
More flexible than previous interop APIs using implicit resources

# Longer Term Roadmap Discussions

Command Buffer Recording and Replay

Unified Shared Memory

Floating-point Atomics

Global Barriers

YUV Multi-planar Images

Generalized Image from Buffer

Indirect Dispatch

Collective Programming

Expect and Assume Optimization Hints

Required Subgroup Size

Machine Learning Operations

Extended Async Copies

2D and 3D Prefetch Built In Functions

## Developer Feedback Welcome!

What is your highest priority?

What is missing?

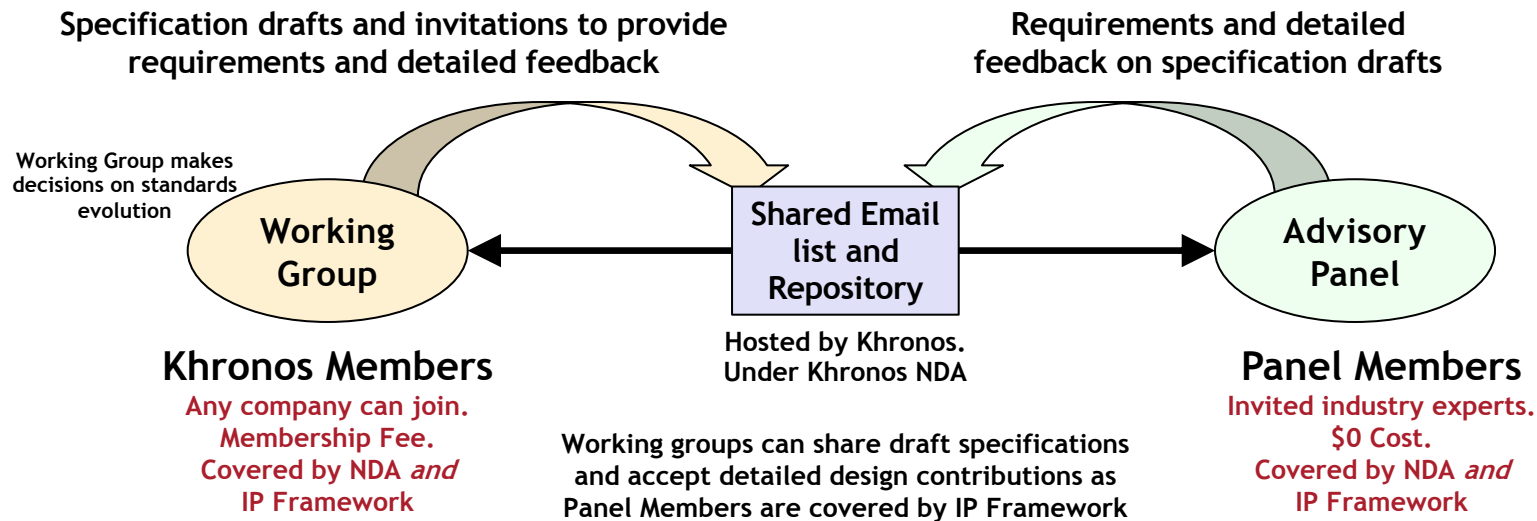
Requirements and use cases

See 'Extensions Feedback' issue on GitHub

<https://github.com/KhronosGroup/OpenCL-Docs/issues/604>

**New functionality is proven as extensions before being added to core**

# OpenCL Advisory Panel



**Chaired by Máté Ferenc Nagy-Egri at StreamHPC**

OpenCL Advisory Panel meeting here at IWOCL

Regular meetings to give feedback on roadmap and draft specifications

Please reach out to [opencl-chair@lists.khronos.org](mailto:opencl-chair@lists.khronos.org) if you wish to apply



# Developers - Please Give Us Feedback!

- How is your transition to using OpenCL 3.0?
  - Are you encountering any issues?
- Which optional features do you expect to use in your application or library?
  - Usage data drives which optional features should be made mandatory in future
- What new features do you most need?
  - What roadmap extensions would you prioritize, and are there any gaps?
  - <https://github.com/KhronosGroup/OpenCL-Docs/issues/604>
- Consider applying to join the OpenCL Advisory Panel!
  - Email [opencl-chair@lists.khronos.org](mailto:opencl-chair@lists.khronos.org)

**More OpenCL information!**

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

**Feedback Welcome!**

<https://github.com/KhronosGroup/OpenCL-Docs>

