Vulkan Guide

Khronos Group

Lina Liu Nov. 22

Logistics Overview

What is Vulkan?

1. Vulkan is a new generation graphics and compute API
2. Vulkan provides high-efficiency, cross-platform to modern GPUs used in a devices like pc/mobile/ embeded platforms
3. Vulkan provide a way for developers to program their modern GPU hardware
    1. Vulkan is a tool for developers to create hardware accelerated applications
4. The Khronos Group is created and maintains Vulkan.

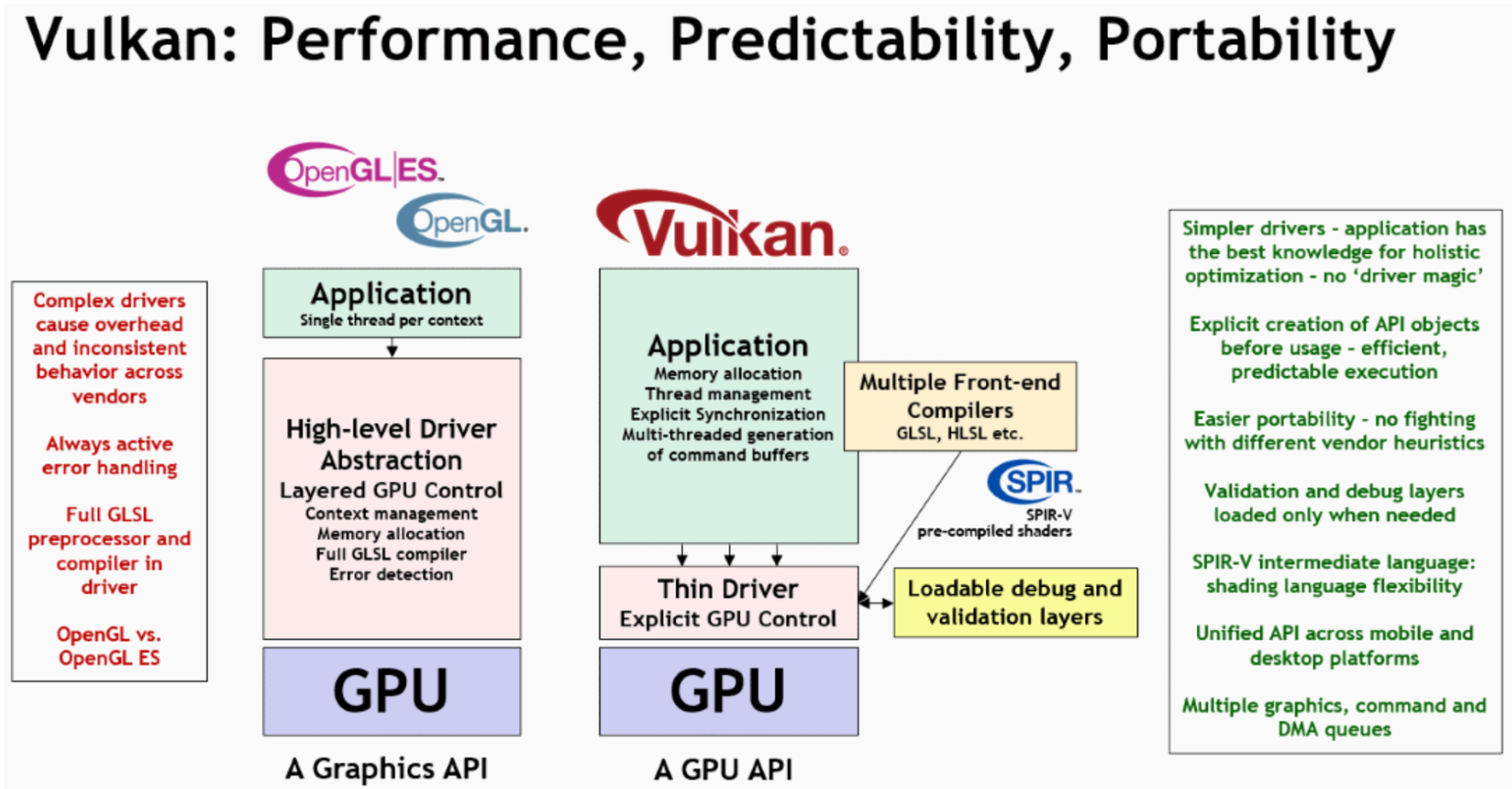
Vulkan and OpenGL

Difference

1. OpenGL is also a 3D Graphics API
2. Vulkan is not a replacement for OpenGL
3. Vulkan is an explicit API allows for more explicit control of the GPU

Feature	OpenGL ES	Vulkan
State management	Global state	State objects
API execution model	Synchronous	Asynchronous
API threading model	Single threaded	Multi-threaded
API error checking	Extensive runtime checks	Only via layers
Render pass abstraction	Inferred render passes	Explicit render passes
Memory allocation	Client-server pools	Shared memory pool
Memory usage	Typed allocations	Typed views

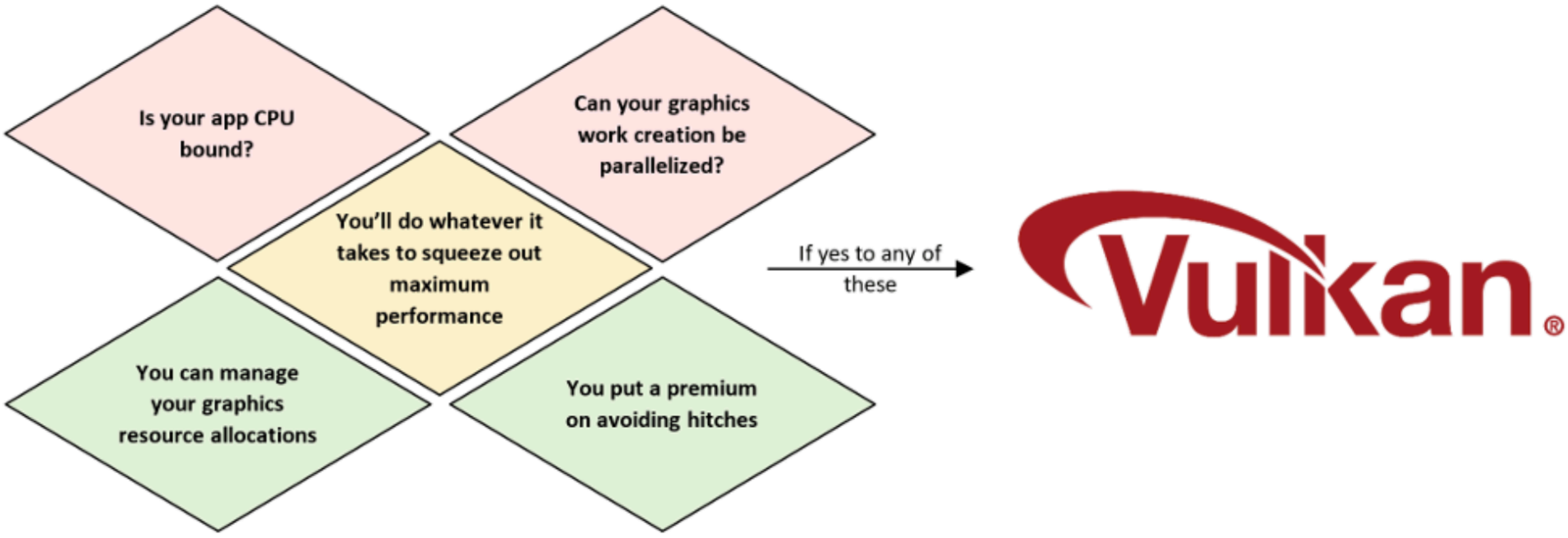
Vulkan and OpenGL



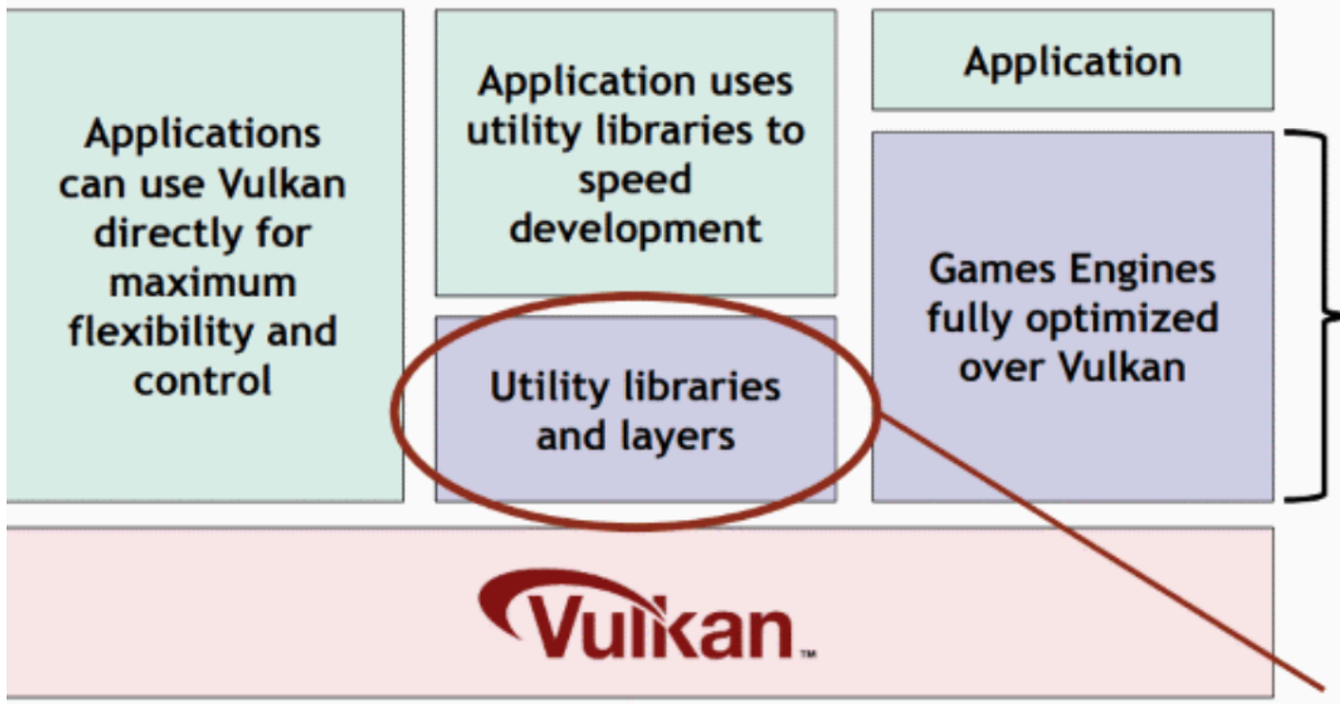
Vulkan and OpenGL

Vulkan puts more work and responsibility into application

1. Why Vulkan put more work and responsibility into application?
    1. For those who use Vulkan correctly can find power and performance improvements



The Power of a Three Layer Ecosystem



The industry's leading games and engine vendors are participating in the Vulkan working group

OXIDE · LUCASFILM Ltd · BLIZZARD ENTERTAINMENT · EA · unity · EPIC GAMES · VALVE

Developers can choose at which level to use the Vulkan Ecosystem

Rich Area for Innovation

- Many utilities and layers will be in open source
- Layers to ease transition from OpenGL
- Domain specific flexibility

The same ecosystem dynamic as WebGL

A widely pervasive, powerful, flexible foundation layer enables diverse middleware tools and libraries

What Vulkan Can Do?

User can use Vulkan to develop application for many use cases

1. Graphics
    1. Developers can create 2D/3D hardware accelerated graphical applications
2. Compute
    1. Vulkan supports compute variation of VkQueues/VkPipelines, so Developer can use Vulkan for general computation
3. Ray Tracing
    1. What is Ray Tracing?
        1. Ray tracing is an alternative rendering technique, based around the concept of simulating the physical behavior of light
        2. Vulkan support VK_KHR_ray_tracing_pipeline
4. Vulkan Video
    1. Vulkan Video provide fine-grained control over video processing scheduling, synchronization, and memory utilization to the application.
5. Machine Learning
    1. Make Vulkan a first class API for exposing ML compute capabilities of modern GPUs.
6. Safety Critical
    1. Bring the graphics and compute capabilities of modern GPUs to safety-critical systems in the automotive, avionics, industrial and medical space.

Platforms

Vulkan runs on many platforms, each has small variations

`DVK_USE_PLATFORM_WIN32_KHX`

`DVK_USE_PLATFORM_MACOS_MVK`

`DVK_USE_PLATFORM_ANDROID_KHX`

`DVK_USE_PLATFORM_WAYLAND_KHR`

`DVK_USE_PLATFORM_XCB_KHR`

Checking for Vulkan Support

Platform support and Device Support

1. How to check if your platform is support Vulkan?
    1. Each platform uses a different mechanism to manage how the Vulkan Loader is implemented
        1. Android: run Vulkan Hardware Capability Viewer app developed by Sascha Willems.
        2. BSD Unix: run vulkaninfo in VulkanSDK.
        3. iOS: Vulkan Hardware Capability Viewer provided by LunarG.
        4. Linux: run Vulkaninfo in VulkanSDK
        5. macOS: run Vulkaninfo in VulkanSDK
        6. Windows: run Vulkaninfo.exe in VulkanSDK
    2. The loader is then in charge of determining if a Vulkan Driver is exposed correctly.

Checking for Vulkan Support

Platform Support and Device Support

2. How to Check if your device is support Vulkan?
    1. For device support, one will need to make sure a Vulkan Driver is available that full implements Vulkan.
    2. There are a few different variations of a Vulkan Driver
        1. Hardware Implementation
            1. A certain GPU might have the physical capabilities of running Vulkan, it still requires a driver to control it.
            2. The driver is in charge of getting the Vulkan calls mapped to the hardware in the most efficient way possible.
            3. Drivers, like software have different versions and update, so there can be many variations of drivers for the same physical device and platform.
                1. Vulkan Database https://vulkan.gpuinfo.org/ is the largest collection of recorded Vulkan implementation details.
        2. Null Driver
            1. The term “Null Driver” is given to any driver that accepts Vulkan API calls, but does not do anything with them.
            2. The term “Null Driver” is common for testing interactions with the driver without needing any working implementation backing it.
            3. Many uses cases such as creating CTS Tests for new features, testing the validation layers, and more rely on the idea of a null driver.
            4. Khronos provides Mock ICD as one implementation of a null driver that works on various platforms
    3. Software Implementation
        1. It is possible to create a Vulkan implementation that only runs on the CPU.
        2. Is useful to test Vulkan that is hardware independent, but unlike the null driver, also outputs a valid result
        3. SwiftShader is an example of CPU-based implementation.

Vulkan Versions

How many version types in Vulkan?

1. Instance-level version
    1. Applications use vkEnumerateInstanceVersion to check what version of a Vulkan instance is supported.
2. Device-level version
3. Loader version
4. SPIR-V version
    1. Every minor version of Vulkan maps to a version of SPIR-V that must be supported.
    2. the application to make sure that the SPIR-V in VkShaderModule is of a valid version to the corresponding Vulkan version

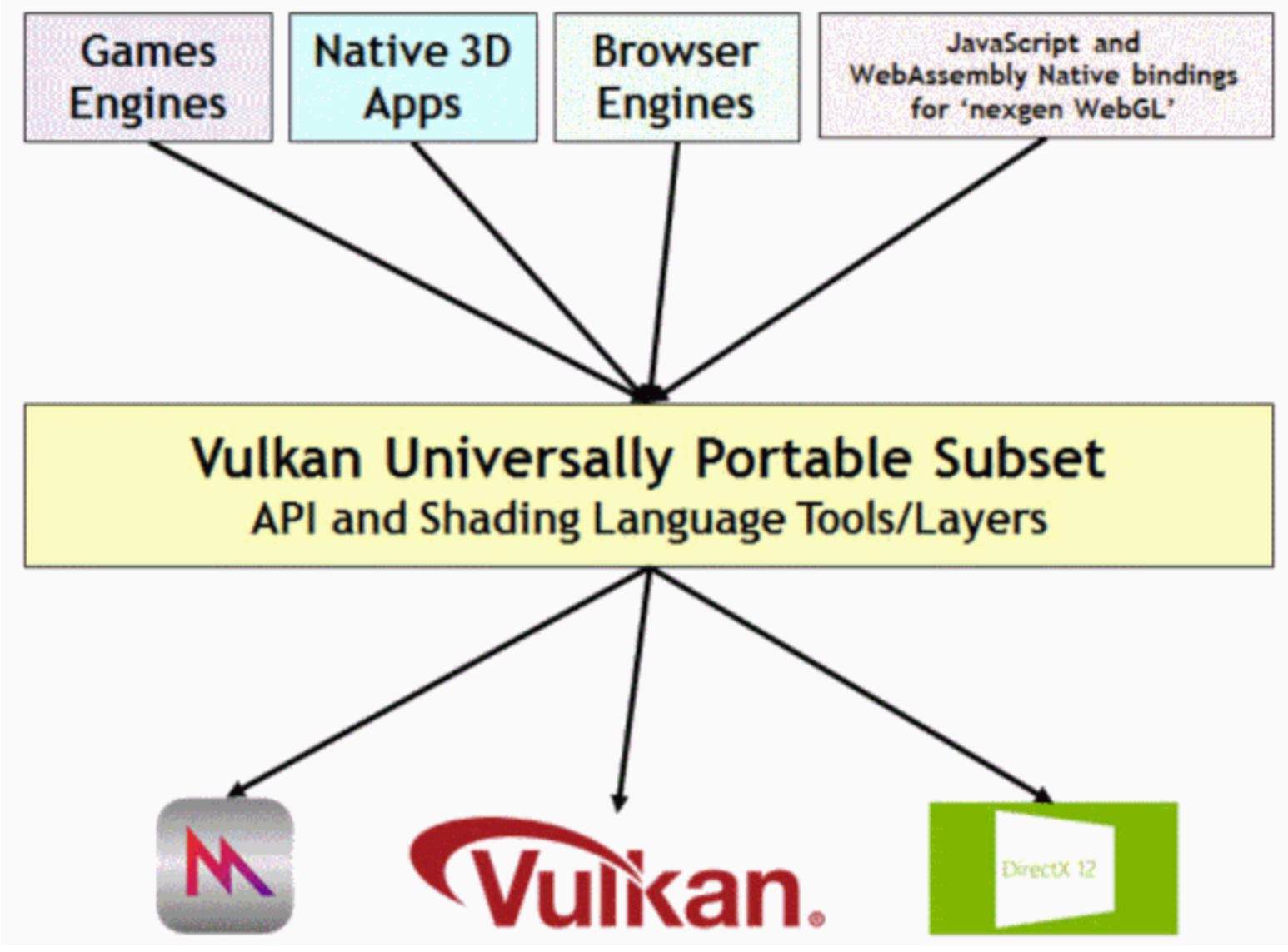
What is SPIR-V?

SPIR-V byte-format used in vkCreateShaderModule

1. Standard Portable Intermediate Representation Vulkan
2. SPIR-V supported capabilities Vulkan Requires:
    1. https://registry.khronos.org/vulkan/specs/1.3/html/vkspec.html#spirvenv-capabilities
3. Vulkan Interfaces with SPIR-V shaders
    1. https://registry.khronos.org/vulkan/specs/1.3/html/vkspec.html#interfaces
4. SPIR-V white paper
    1. https://registry.khronos.org/SPIR-V/papers/WhitePaper.pdf
5. Introduction to SPIR-V Shaders
    1. https://www.khronos.org/assets/uploads/developers/library/2016-vulkan-devday-uk/3-Intro-to-spir-v-shaders.pdf
6. Using SPIR-V in practice with SPIRV-Cross
    1. https://www.khronos.org/assets/uploads/developers/library/2016-vulkan-devday-uk/4-Using-spir-v-with-spirv-cross.pdf
7. GLSLang
    1. https://github.com/KhronosGroup/glslang
8. Shaderc
    1. A collection of tools, libraries and tests for shader compilation
    2. https://github.com/google/shaderc
    3. Glslc is command line compiler for GLSL/HLSL to SPIR-V
    4. Libshaderc is a library API for accessing glslc functionality.

Portablity Initiative

Develop Vulkan capabilities to support all major platforms

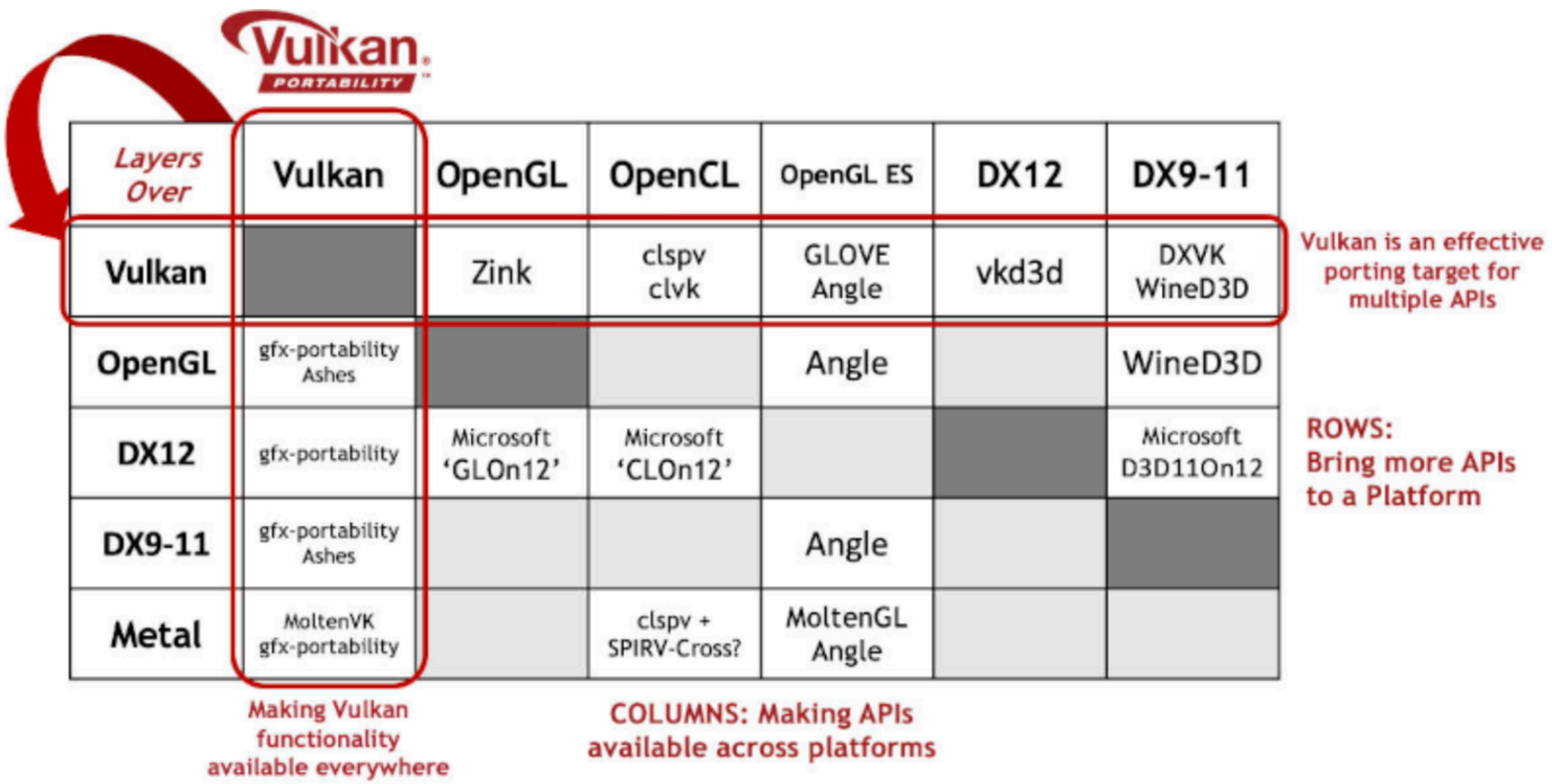


Vulkan Universally Portable Subset

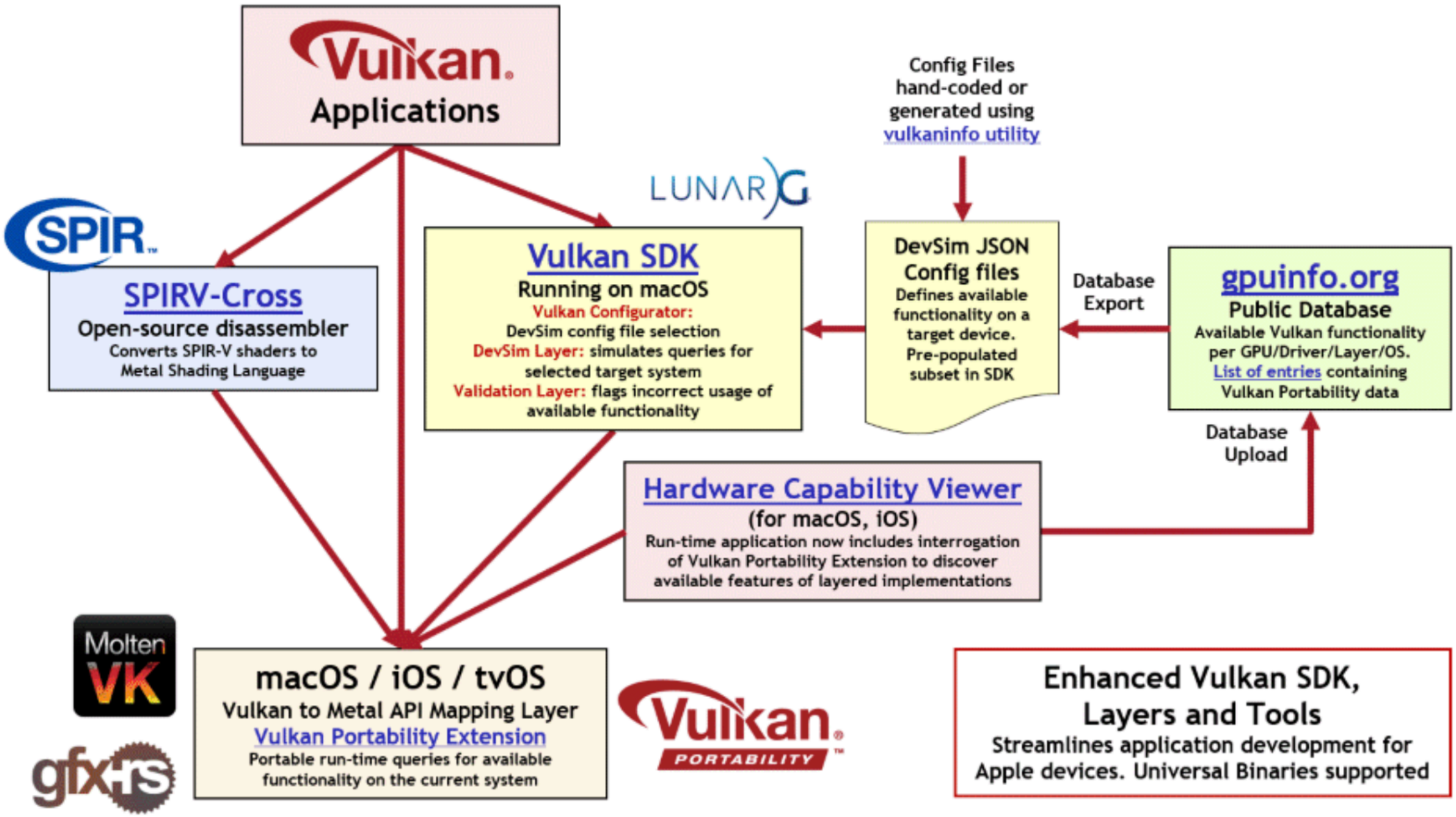
Translation layer

1. What's the advantage of layer implementations?
    1. Layered implementations used successfully ship production applications on multiple platforms.
    2. Layered implementation enable more applications to run on more platforms
    3. Vulkan can be used as a porting target to bring additional APIs to platforms without the need for additional kernel-level drivers.
    4. Vulkan provide translation layer to bring addition APIs to platform

12.



Vulkan Portability Development on Apple

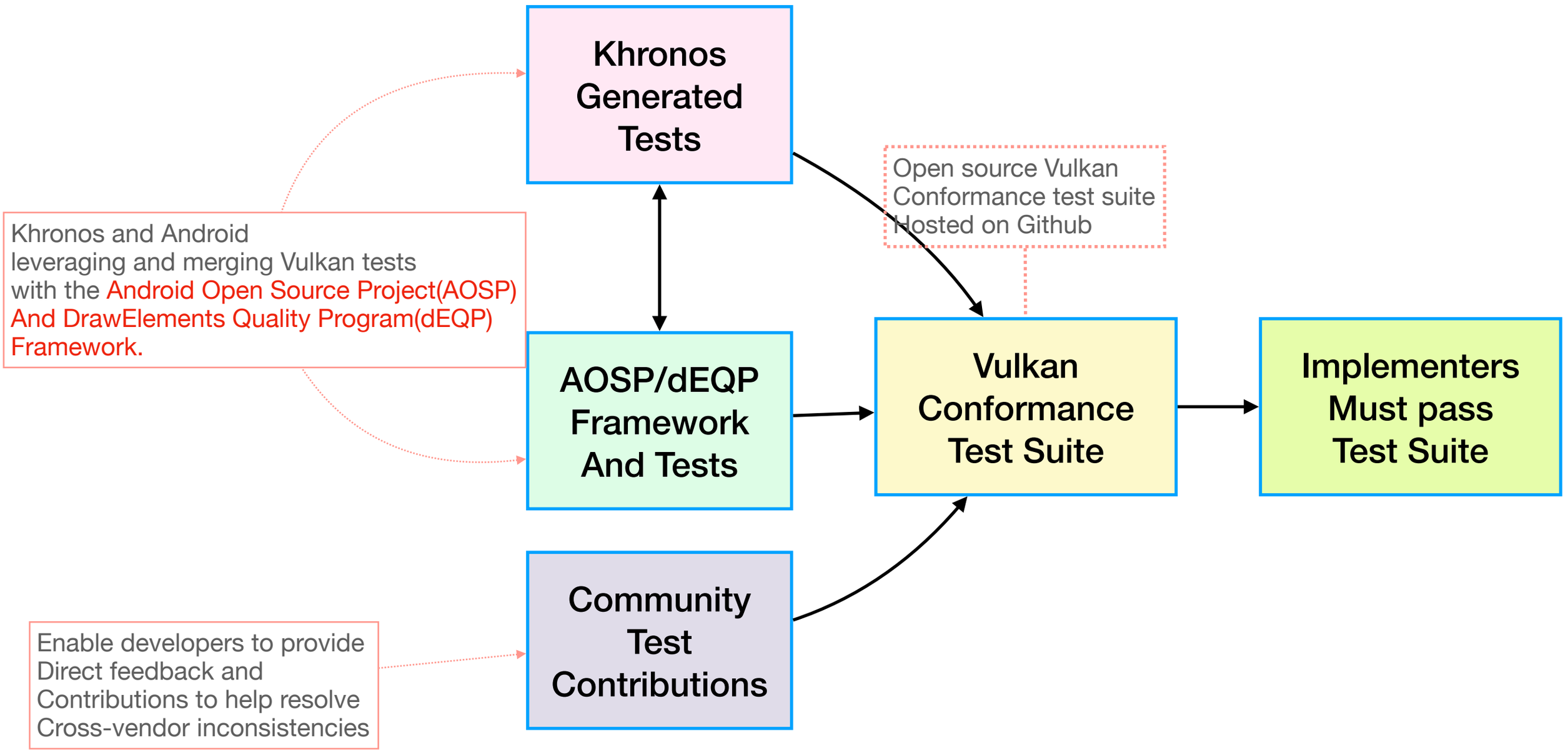


Vulkan CTS

Vulkan Conformance Tests Suites

1. What is Vulkan CTS?
    1. Vulkan conformance test suite
    2. Is a set of tests used to verify the conformance of an implementation.
    3. The conformant implementation shows that it has successfully passed CTS and it is an valid implementation of Vulkan.
4. Vulkan CTS source code is freely available
    1. Anyone is free to create and add new test
    2. Before add new test, the new test must follow the contributing wiki
        1. https://github.com/KhronosGroup/VK-GL-CTS/wiki/Contributing

Vulkan Open Source Conformance Tests



dEQP

Draw Element Quality Program

1. What's dEQP contained?
    1. GPU testing suite
    2. Contains tests for several graphic APIs, including OpenGL ES, EGL, and Vulkan
    3. Vulkan CTS have been built on dEQP framework
    4. .qpa logs
        1. generated by the conformance tests
        2. Contain embedded PNG images of the results
        3. Can be viewed with scripts/qpa_image_viewer.html
            1. Using the Cherry tool

Running CTS

Cmd line options be used when running CTS

1. Cmd line options be used when running CTS
    1. Running multi test cases at a time:
        1. —deqp-caselist-file=/../../../mustpass/main/vk-default.txt
        2. —deqp-log-images=disable
        3. —deqp-log-shader-sources=disable
2. Run ./deqp-vk —help to get all the cmd line option and it’s meanings

Conformance Submission Package Requirements

1. What's the requirements of the Conformance Submission Package?
    1. Full test logs
        1. TestResult.qpa in the same folder as ./deqp-vk
    2. Run git status git log from vulkancts source directory
    3. Run git cherry-pick for apply bug fixes

Conformance test results

1. Pass
2. NotSupported
3. QualityWarning
4. CompatibilityWarning
5. Waiver

Cherry GUI

Vulkan test module can be used with Cherry

1. Cherry: GUI for test execution and analysis
    1. https://android.googlesource.com/platform/external/cherry

Shader Optimizer

Vulkan CTS can be optionally run with the shader optimizer enabled

1. Experimental feature: Vulkan CTS can run with the shader optimizer enabled.
    1. —deqp-optimization-recipe=<>
2. Experimental feature: Vulkan CTS can run with spir-v optimizer
    1. —deqp-optimize-spirv=enable
        1. Maybe useful in finding new bugs in driver or the optimizer itself.

Shader Cache

Why use Shader Cache?

1. Why use Shader Cache in Vulkan-CTS?
    1. To speed up the running of the CTS
    2. Skipping shader compilation can significantly reduce runtime, especially repeat runs.
    3. Default is shader cache enabled, but truncated at the the start of the CTS run

Shader Cache

How to use Shader Cache

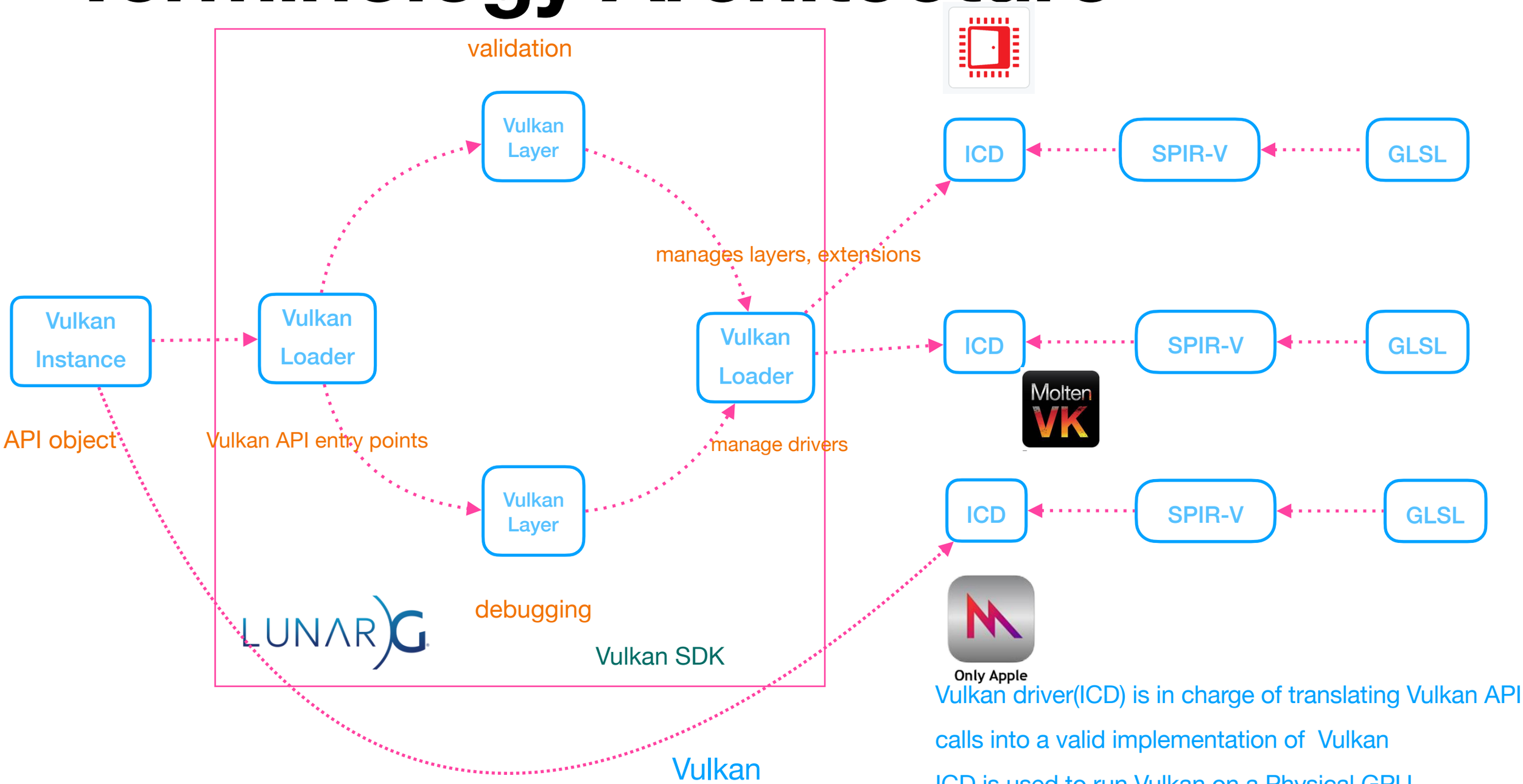
1. —deqp-shadercache=disable
2. —deqp-shadercache-ipc=enable
    1. Enable the use of inter-process communication primitives to allow several instances of CTS to share a single cache file.
    2. All the instances must use the same shader cache filename.
    3. if one instance should crash while holding the cache file lock, the other instances will hang

Vulkan Development Tools

Tools From Khronos and LunarG

1. Khronos
    1. Vulkan Samples
    2. A collection of code
    3. A collection of tutorials
    4. The samples/code/tutorials that demonstrates API usage and explains the implementation of performance best practices.
2. LunarG
    1. The curator for the Vulkan Loader and Vulkan Validation Layers Khronos Group repositories
    2. Delivers the Vulkan SDK
    3. Develop Key tools:
        1. Vulkan Configurator
        2. GFXReconstruct

Terminology Architecture



Vulkan driver(ICD) is in charge of translating Vulkan API calls into a valid implementation of Vulkan

ICD is used to run Vulkan on a Physical GPU

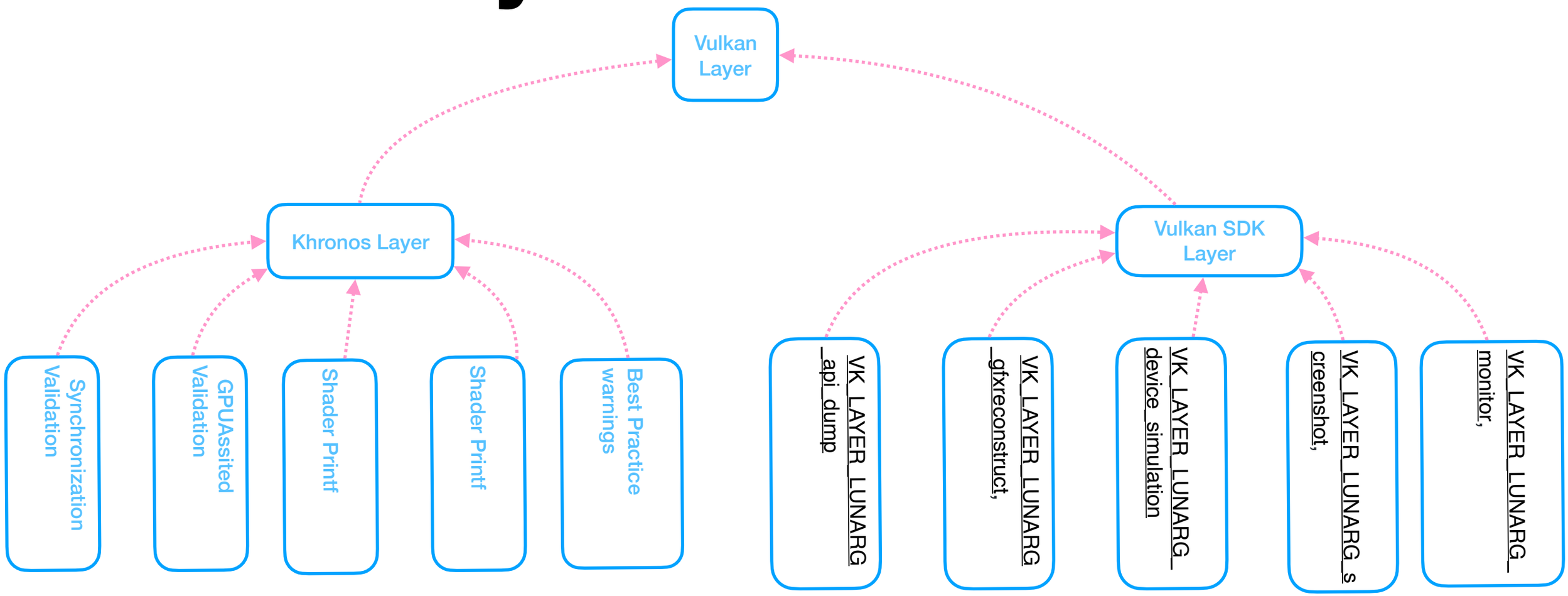
Vulkan SDK includs a MoltenVK runtime library for macOS

Vulkan Developing Tools

Vulkan Layers

1. What can Vulkan Layers do?
    1. Vulkan Layers are optional components that augment the Vulkan system.
    2. Vulkan Layers can intercept, evaluate, and modify existing Vulkan functions from the application down to the hardware.
    3. Layers are implemented as libraries
    4. Layers can be enabled and configured using Vulkan Configurator.

Vulkan Layer



Vulkan Developing Tools

What’s Vulkan Layer contained?

1. What’s Vulkan Layer contained?
    1. Khronos Layers
        1. Validation layer
            1. For debug use
                1. Synchronization validation: identify resources access conflicts due to missing or incorrect synchronization operation between actions(draw, copy, dispatch, blit) reading or writing the same region of memory.
                2. GPU-Assisted Validation: Instrument shader code to perform run-time checks for error conditions produced during shader execution
                3. Shader printf: Debug shader code by printing any values of interest to the debug callback or stdout
                4. Best Practices warnings: highlights potential performance issues, questionable usage patterns, common mistakes.
        2. Synchronization layer
            1. Extensions, disable by default.
    2. Vulkan SDK Layers