Presented at IWOCL & SYCLcon 2020

by Neil Trevett. Khronos President and OpenCl Working Group Chair.

1. Hi everyone – it's a pleasure to be here at Virtual IWOCL. I am Neil Trevett and I work on developer ecosystems at NVIDA, I am President of the Khronos Group and also the OpenCL Working Group Chair. I am going to briefly update you on the OpenCL 3.0 launch. This is going to be a lightning tour – but there are many more details on the Khronos website if you are interested..

2. Let's start by reminding ourselves why this launch is so significant to the industry. OpenCL is the most widely used and deployed, cross-vendor, open standard for low-level parallel programming. And this is at a time when Moore's Law is slowing and so parallel programming and acceleration offload to heterogeneous processors is becoming ever more essential for everything from High Performance Computing through desktops applications to embedded and mobile platforms.

OpenCL is widely deployed by GPU vendors and used by an ever-increasing number of applications, engines and libraries – as well as acting as a backend target for languages, compilers and machine learning stacks that need a portable API to reach into hardware acceleration. And that includes Khronos' own SYCL of course.

3. It's become an IWOCL tradition to plot the growth in open source projects using OpenCL each year – and I am glad to report that the curve is accelerating - and we have now broken through the 9000-project barrier – that's a doubling in under three years.

One of the most topical and significant open source projects right now of course is Folding@Home being used to research the Corina Virus. Folding@Home uses OpenCL for GPU acceleration in its network of consumer PCs - which is now delivering an amazing 2.4 exaflops – faster than the top 500 traditional supercomputers in the world combined – thanks largely to OpenCL.

4. So let's talk about OpenCL 3.0 which was launched yesterday here at IWOCL. This is a unique launch as it doesn't add lots of new functionality to the API – it is an ecosystem realignment to enable OpenCL to thrive and move forward to reach even more developers and devices. To understand OpenCL 3.0 let's look at OpenCL's evolution - much of which has unfolded here each year at the annual IWOCL conference. This slide includes missteps as well as successes – as THAT is often where the most valuable lessons are learned.

On the left, we see how significantly OpenCL has influenced the Khronos ecosystem, by spawning SPIR which evolved into SPIR-V which became a foundational building block for Vulkan. Also, SYCL started as a layer over OpenCL – and is now SYCL is significantly impacting the industry and being made available over multiple API backends.

Looking back at OpenCL's evolution – it is remarkable how quickly the combination of OpenCL 1.2 and OpenCL C became a widely adopted baseline that successfully addressed the needs of a wide number of developers and platforms.

OpenCL 2.X added significant new functionality, but a series of monolithic specifications became increasingly onerous for vendors to implement when much of that functionality was not applicable to THEIR customers – and so the adoption rate for OpenCL 2.X decreased significantly. So. this leads directly to the most important aspect of OpenCL 3.0. In a precise and carefully designed way – OpenCL 3.0 makes all 2.X functionality beyond 1.2 optional. This enables vendors to focus on shipping the functionality they need for their customers – and also, somewhat paradoxically, resets the opportunity to carefully raise the bar on core functionality that can become as pervasive as OpenCL 1.2.

The philosophy behind OpenCL 3.0 also sets the stage for extended optionality in a possible future Flexible Profile to enable diverse embedded processors to deploy the OpenCL runtime framework pervasively and cost-effectively.

OpenCL 3.0 also embraces cooperation with the open source LLVM/Clang community to build effective kernel language solutions. OpenCL 3.0 does not include the OpenCL C++ specification. OpenCL 3.0 implementations are encouraged to use the 'C++ for OpenCL' open source front-end compiler that enables full OpenCL C to be mixed with much of C++ 17 to generate SPIR-V kernels.

Lastly – we want to put to rest one blind alley that was discussed at previous IWOCLs – the idea to somehow merge OpenCL and Vulkan. We now know that both APIs will be successful in their own right –

with Vulkan focused on GPU-only acceleration and mixed compute and rendering. OpenCL will continue to have more evolved compute capabilities and can be significantly easier to implement and program.

5. OpenCL 3.0 ships a new unified API specification with carefully designed, queryable optionality for all OpenCL 2.X functionality. In addition, the announcement yesterday included a significant extension for DSP-like processors to asynchronously and flexibly transfer 2D and 3D data between global and local memories via DMA transactions – this is the first in a series of upcoming advances in OpenCL to enhance support for embedded processors.

The released OpenCL 3.0 specification is provisional so that we can get developer feedback before we finalize, but once we do, we expect OpenCL 3.0 to gain rapid industry adoption. Current applications are fully compatible with any OpenCL 3.0 device that supports the functionality they use – and we recommend developers to use the new 2.X-level functionality queries for future cross-vendor portability. Implementers simply need to add the new queries for all 2.x functionality - whether missing or present – to their existing implementations.

6. Looking forward, OpenCL will ship new functionality first as extensions and only when they are fully implemented, and then ensuring they are proven and widely adopted before considering folding them into future core specifications. There are quite a few extensions in the drafting pipeline: including extended subgroups, Vulkan interop and accelerated machine learning primitives.

As well as defining specifications, the OpenCL Working Group is working hard to build out a Khronos OpenCL SDK - and to encourage an ever-stronger ecosystem of domain-specific libraries.

As well as considering a Flexible Profile for embedded processors, the OpenCL Working Group is considering a profile to precisely specify layered OpenCL implementations – more on that in a second – but in general, we see Profiles as a vital tool to harness OpenCL 3.0's flexibility – enabling us to strike the appropriate balance between implementation flexibility and application portability for each of our target markets.

7. Next, I want to talk about the growing industry trend of layering APIs over other APIs – in particular, the cross-compilation of shading languages by an increasingly robust open source compiler ecosystem.

For developers, a layered API can enable their application on platforms that don't support that API natively. Good examples include the MoltenVK Vulkan implementation over Metal and the Google clspv compiler layering OpenCL over Vulkan - which is being used to bring demanding OpenCL applications – such as Adobe's video editing apps - to Android.

Clspv is a great example of layered APIs also being invaluable to platform vendors that can enable content on their platform without incurring the support load of an additional kernel-level driver. A good example of this is the newly announced OpenCLon12 open source project by Microsoft that will enable GPU-accelerated OpenCL applications on any system with DX12.

8. Khronos' SPIR-V is at the center of this growing ecosystem of open source kernel language compilers, including Clang and LLVM, that can generate SPIR-V kernels - either for direct ingestion by OpenCL or Vulkan - OR for further translation into shaders to run on other APIs such as Metal – which would enable OpenCL applications on Apple platforms without needing to use native OpenCL drivers.

Also, Microsoft's OpenCLon12 project translates LLVM-generated SPIR-V kernels using a MESA conversion pipeline to DXIL, the DirectX intermediate language - for execution on DX12.

Enabling language compilers to innovate independently from run-times was always the primary dream of SPIR-V and it is great to see this vision come to fruition.

9. So - that's OpenCL 3.0 in a nutshell – we warmly request your feedback to help us finalize the specification, while we also complete the conformance test suite and prepare the first wave of implementations over the next few months. The Working Group has placed the source of the specifications and conformance tests IN open source – to enable and accelerate that feedback process.

Thank you for your time – again all these slides and more are on the Khronos website - and we have many folks here on the panel that have been instrumental in creating OpenCL 3.0 - and we look forward to answering your questions!