1. みなさん、こんにちは。私は Neil Trevett です。NVIDA で開発者エコシステムに取り組んでいます。Khronos グループの社長であり、OpenCL ワーキンググループの議長も務めています。並列プログラミングのための OpenCL オープンスタンダードの概要と更新について、数分かけて説明します

H i everyone, I am Neil Trevett, I work on developer ecosystems at NVIDA, I am President of the Khronos Group and I also chair the OpenCL Working Group. I am going to spend a few minutes giving you a summary and update on the OpenCL open standard for parallel programming.

2. OpenCL は、アプリケーションを高速化するために使用されるのですが、異種プロセッサ（CPU、GPU、DSP、FPGA などのさまざまな複数のタイプのプロセッサ）上でアプリケーションを実行する為に使われます。現在、「ムーアの法則の終焉」に対抗する為に、並列処理アクセラレーションに対する業界の関心が高まっています。これは、必要なパフォーマンスを提供する効果的な方法として並列プログラミングがプロセッサの周波数を向上させるという手法に取って代わった為です。クロノスには、並列計算処理向けに複数の標準規格が存在しています。ここにある SYCL と OpenVX は、高レベルのプログラミングフレームワークです。SYCL は標準の C ++を使用した並列プログラミングを可能にし、OpenVX は抽象化グラフを使用してビジョンと推論機能を表現します。 SYCL と OpenVX は、GPU をプログラミングするための Vulkan を含む低レベルの Khronos API と、ほぼすべてのタイプのプロセッサをプログラミングできる OpenCL によって高速化することができます。 OpenCL と Vulkan はどちらも、Khronos の SPIR-V という標準規格の中間表現に含まれるシェーダーとカーネルプログラムを処理することが可能です。これにより、後で説明するように、言語とコンパイラのエコシステムに大きな柔軟性がもたらされます。

OpenCL is used to accelerate applications by running them across heterogeneous processors - multiple processors of diverse types including CPUs, GPUs, DSPs, and FPGAs. There is increasing industry interest in parallel processing acceleration to combat the ‘End of Moore’s Law’ as processor frequency scaling gives way to parallel programming as an effective way to deliver needed performance. Khronos has multiple standards for parallel computation. SYCL and OpenVX shown here are higher-level programming frameworks: SYCL enables parallel programming using standard C++, and OpenVX uses a graph abstraction to expresses vision and inferencing functionality. SYCL and OpenVX MAY be accelerated by lower level Khronos APIs that include Vulkan for programming GPUs, and OpenCL that lets you program any almost type of processor. Both OpenCL and Vulkan can process the shaders and kernel programs in Khronos’ SPIR-V standard intermediate representation, which enables significant language and compiler ecosystem flexibility - as we will see later.

3. OpenCL は、ランタイムおよびプログラミングフレームワークで、システム上にある利用可能なあらゆる並列プロセッサ上でコンパイルおよび実行が可能な計算負荷のかかるカーネルを作成するための C および C ++ベースの言語を含むものです。 OpenCL は、アクセラレータデバイスを検出し、選択したデバイスのカーネルをコンパイルし、カーネルを実行し（カーネルの

配置、メモリの割り当て、同期を明示的に制御して）、最終的に結果を収集するための API を提供します。 OpenCL と Vulkan などの GPU API のどちを選択しようか考える際、グラフィックスレンダリングを必要としない多くの開発者は、OpenCL の方がよりシンプルなプログラミングモデルで、より軽量なランタイムで、より厳密に定義された数値の精度（多くのコンピューティングアプリケーションにとっては必須）を提供していることに気がついています。

OpenCL is a runtime and programming framework that includes C and C++-based languages for creating compute-intensive kernels that can be compiled and executed across any available parallel processors in a system. OpenCL provides APIs to discover accelerator devices, compile kernels for selected devices, execute the kernels - with explicit control over kernel placement, memory allocation and synchronization - and finally gather the results. When deciding between OpenCL, or GPU APIs such as Vulkan, many developers that don't need graphics rendering find that OpenCL provides a simpler programming model, a lighter weight runtime, and more rigorously defined numerical precision that can be critical for many compute applications.

4. OpenCL は、低レベルの並列プログラミングのための最も普及している、クロスベンダーのオープンな標準規格です。 OpenCL は、GPU、DSP、FPGA ベンダーによって展開され、ますます多くのアプリケーション、エンジン、ライブラリで使用されています。また、ハードウェアアクセラレーションを手に入れる為に、移植が容易な API を必要とする言語、コンパイラ、機械学習スタックのバックエンドターゲットとしても機能します。これには Khronos の SYCL と OpenVX も含まれます。 特に、Arm、Qualcomm、Intel、VeriSilicon、Synopsis、TI、Xiaomi といった企業のほぼすべてのモバイル向けの推論エンジンは、アクセラレーションバックエンドとして OpenCL を使用しています。

OpenCL is the most pervasive, cross-vendor, open standard for low-level parallel programming. OpenCL is deployed by GPU, DSP and FPGA vendors and used by an 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 - including Khronos' own SYCL and OpenVX. In particular, almost all mobile inferencing engines from companies such as Arm, Qualcomm, Intel, VeriSilicon, Synopsis, TI, and Xiaomi use OpenCL as an acceleration backend.

5. OpenCL が業界全体でどれほど広く使用されているかを示すもう 1 つのものとして、現在、GitHub に OpenCL を使用したオープンソースプロジェクトが 12,000 以上もあることが挙げられます。これは過去 4 年間で 3 倍になりました。現在最もホットなトピックで重要なオープンソースプロジェクトの 1 つは、そうです、コロナウイルスの研究に使用されている Folding@Home です。 Folding@Home は、コンシューマーPC を繋いだ GPU アクセラレーションに OpenCL を使用しています。これにより、2.4 エクサフロップスという驚異的な数字を実現しています。これは従来のスーパーコンピューターの上位 500 台を組み合わせたものよりも高速です。

Another indication of how widely OpenCL is used in throughout the industry, is that there are now over

12,000 open-source projects on GitHub using OpenCL - that's a tripling in the last four 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 combined.

6. OpenCL の最新バージョンは 2020 年 9 月にリリースされた OpenCL3.0 です。OpenCL3.0 を使用すると、API クエリと言語マクロを通じて OpenCL 1.2 以降のすべての機能をオプションにすることで、ハードウェアベンダーは顧客が必要とするドライバー機能にリソースを集中できます。 OpenCL 3.0 はまた、仕様から OpenCL C ++言語を削除し、SPIR-V コードを生成する OpenCL オープンソースフロントエンドコンパイラーに C ++を使用することを推奨しています。 既存のアプリケーションは OpenCL3.0 に簡単に移行可能です。OpenCL1.2 アプリケーションは変更を必要とせず、OpenCL 2.X アプリケーションも、使用するすべての機能が存在することがわかっている場合はコードを変更する必要はありません。OpenCL 3.0 を使った実装への将来の移植性を確保するために、開発者の方々には使用するオプション機能を照会することをお勧め致します。

The latest version of OpenCL is OpenCL 3.0 that was released in September 2020. OpenCL 3.0 enables hardware vendors to focus their resources on driver functionality that their customers need, by making all functionality beyond OpenCL 1.2 optional through API queries and language macros. OpenCL 3.0 also removed the OpenCL C++ language from the specification and recommends use of the C++ for OpenCL open-source front-end compiler that generates SPIR-V code. Existing applications can be easily moved to OpenCL 3.0: OpenCL 1.2 applications require no changes, and OpenCL 2.X applications also require no code changes if all the functionality they use is known to be present. To ensure future portability to any OpenCL 3.0 implementation, developers are encouraged to query for any optional functionality that they use. **4:16**

7. OpenCL 3.0 とともに、DSP のような組み込みプロセッサが DMA トランザクションを介してグローバルメモリとローカルメモリ間でデータを転送できるようにする非同期 DMA 拡張機能など、いくつかの新しい拡張機能がリリースされました。 この新しい機能は、複雑なメモリ転送と、並行して実行される複数の非同期トランザクションの同期のためのフェンスの使用をサポートすることにより、以前の 1 次元操作を補完します。 DSP やその他の組み込みプロセッサ向けに OpenCL 機能を拡張し続けることに大きな関心が寄せられています。

Several new extensions were released alongside OpenCL 3.0, including Asynchronous DMA extensions that enable DSP-like embedded processors to transfer data between global and local memories via DMA transactions. This new functionality complements previous one-dimensional operations by supporting complex memory transfers and use of fences for synchronization of multiple asynchronous transactions running in parallel. There is significant interest in continuing to extend OpenCL functionality for DSPs and other embedded processors.

8. もう一つ、活発に開発されている機能拡張のセットがあります。それは、外部メモリ共有

です。これは OpenCL が Vulkan のような他の API と相互運用できるようにするもので、メモリおよびセマフォオブジェクトにハンドルをインポートして、そこからセマフォを使用して待機し、他のランタイムに信号を送って共有メモリの使用を調整することで行います。 OpenCL と Vulkan の間の相互運用は、モバイルプラットフォーム、デスクトッププラットフォームの両方で広く使用され、将来的には Windows10 用の DirectX12 などの他のデスクトップ API にも拡張される予定です。

Another set of extensions being actively developed is External Memory Sharing to enable OpenCL to interop with other APIs such as Vulkan by importing handles to memory and semaphore objects, and then using semaphores to wait and signal the other runtime to coordinate use of the shared memory. Interop between OpenCL and Vulkan will be widely used on both mobile and desktop platforms and will be extended to other desktop APIs in the future such as DirectX 12 for Windows 10.

9. モバイル業界における OpenCL の使用の増加の例として、Google が最近、既存の OpenGL ES バックエンドを補完するために、TensorFlowLite の OpenCL への移植を発表しました。 OpenCL への移植により、同じデバイスにおける高速推論のパフォーマンスが 2 倍のになり、高速化されていない推論のパフォーマンスが 4 倍以上になりました。 Android プラットフォームの一部として公式に公開されていなくても、OpenCL はモバイル GPU のベンダーによって提供されることが多く、モバイル推論フレームワークやビデオおよびイメージのライブラリ用のアクセラレーションターゲットとしてますます使用されています。

As an example of OpenCL's growing usage in the *mobile* industry, Google recently announced a port of TensorFlow Lite to OpenCL, to complement its existing OpenGL ES backend. The OpenCL port delivers twice the accelerated inferencing performance on the same devices – and over four times the performance of unaccelerated inferencing. Even though not officially exposed as part of the Android platform, OpenCL is often provided by mobile GPU vendors - and it is increasingly being used as an acceleration target for mobile inferencing frameworks, as well as video and imaging libraries.

10. OpenCL は、推論アクセラレーションを利用するための多くの機械学習コンパイラのバックエンドターゲットとしても使用されます。 多くの企業が機械学習コンパイルスタックを研究開発しています。 たとえば、Amazon と TVM、Facebook と Glow、Google と XLA などです。 これらのコンパイラスタックはすべて、同様のパターンになっています。 まず、トレーニング済みのネットワークをインポートし、次にグラフレベルの最適化をネットワークに適用してから、ネットワークを実行可能命令に分解します。 これらすべての機械学習コンパイラーは、OpenCL コードを発行して、並列ハードウェアアクセラレーションに容易にアクセスできます。これは、OpenCL が機械学習ライブラリ、エンジン、コンパイラーにとって最も普及している「Close to Metal」(ハードウェアに近い)実行レイヤーであることを示しています。

OpenCL is also used as a backend target for many machine learning compilers to harness inferencing acceleration. Many companies are researching and developing machine learning compilation stacks. For

example, Amazon with TVM, Facebook with Glow and Google with XLA. All these compiler stacks follow a similar pattern. Firstly, import a trained network, then apply graph-level optimizations to the network, and then decompose the network into executable instructions. All these machine learning compilers can emit OpenCL code to portably access parallel hardware acceleration – illustrating how OpenCL is the most pervasive 'close-to-metal' execution layer for machine learning libraries, engines, and compilers.

11. GPU および並列アクセラレーションソリューションがコンパイラテクノロジにますます依存するようになるにつれて、SPIR-V はクロノスの標準規格エコシステムにおいてこれまで以上に重要な部分なってきています。 SPIR-V は、GPU とその他のプロセッサ用の命令に分解できる異種並列処理とグラフィックス操作を含む中間言語表現です。 SPIR-V は、Vulkan によって直接取り込まれる唯一の言語であり、その数が増えている多くの OpenCL 実装にインポートできます。SPIR-V を使用すると、ハードウェアとコンパイラのコミュニティが独立してイノベーションを起こすことができます。フロントエンドコンパイラは、さまざまなランタイム API によって取り込まれて実行される SPIR-V コードを簡単に生成できるためです。さらに、SPIR-V を取り込む API ランタイムにより、フロントエンドコンパイラを組み込みシステムにデプロイする必要がなくなります。

As GPU and parallel acceleration solutions become increasingly dependent on compiler technology, SPIR-V is becoming an ever more important part of the Khronos standards ecosystem. SPIR-V is an intermediate language representation that includes heterogeneous parallel processing and graphics operations that can be broken down into instructions for GPUs and other processors. SPIR-V is the only language directly ingested by Vulkan and it can be imported into a growing number of OpenCL implementations. SPIR-V enables the hardware and compiler communities to innovate independently as front-end compilers can simply generate SPIR-V code to be ingested and executed by a wide range of run-time APIs. In addition, API runtimes that ingest SPIR-V eliminate the need for front-end compilers to be deployed in embedded systems.

12. 次に、SPIR-V を活用してますます堅牢になるオープンソースのコンパイラエコシステムによる、シェーディング言語のクロスコンパイルを通じて、他の API の上に API を階層化するのが進む業界の傾向についてお話ししたいと思います。 開発者にとって、階層化された API を使用すると、その API をネイティブにサポートしていないプラットフォームでもアプリケーションが実行できます。例えば、Google clspv コンパイラを使用すると、OpenCL カーネルを Vulkan ランタイムでコンパイルして実行できます。これにより、clvk ランタイム API トランスレータとともに、OpenCL アプリケーションを Vulkan 上で実行できます。 clspv は、Android 向けのパフォーマンスを要求される OpenCL アプリケーションを出荷する為に既に使用されています。たとえば、Adobe の Rush ビデオエディターは OpenCL の C で書かれた 200,000 行を超えるコードがフル GPU アクセラレーションを備えた Vulkan 上で実行されています。

Next, I want to talk about the growing industry trend of layering APIs over *other* APIs through the cross-compilation of shading languages by the increasingly robust open-source compiler ecosystem leveraging SPIR-V. For developers, a layered API can enable their application to run on platforms that don't support

that API natively. For example, the Google clspv compiler enables OpenCL kernels to be compiled and executed on a Vulkan runtime, which, together with the clvk run-time API translator, enables OpenCL applications to run over Vulkan. clspv is already being used to ship demanding OpenCL applications on Android, such as Adobe's Rush video editor that has over 200,000 lines of OpenCL C running on Vulkan with full GPU acceleration. **8:32**

13. 階層化された API は、追加のカーネルレベルのドライバーのサポート負荷を発生させることなく、プラットフォーム上でコンテンツを有効にするために、プラットフォームベンダーや開発者にとって非常に価値のあるものとなります。たとえば、Microsoft は、新しく発表された OpenCLon12 というオープンソースプロジェクトに取り組んでおり、Microsoft の DX12 API 上に OpenCL をレイヤー化しています。 OpenCLon12 は MESA オープンソースグラフィックフレームワークを活用しており、Microsoft は COLLABORA と協力して、Arm ベースの PC やクラウドを含む、Windows10 を対応するプラットフォームで OpenCL を有効にしています。

Layered APIs can also be invaluable to platform vendors, as well as developers, to enable content on a platform without incurring the support load of an additional kernel-level driver. For example, Microsoft is working on the newly announced OpenCLon12 open-source project to layer OpenCL over Microsoft's DX12 API. OpenCLon12 leverages the MESA open-source graphics framework, and Microsoft is working together with COLLABORA, to enable OpenCL wherever Windows 10 ships, including on Arm-based PCs and in the cloud.

14. 以上が、OpenCL の簡単な紹介となります。こちらのご紹介がお役に立てば幸いです。最後に、OpenCL に関連する情報へのリンクをいくつか載せておきました。スライドやその他のものは、Khronos の Web サイトで入手可能です。 たとえば、ワーキンググループからは仕様と適合性テストを GitHub のオープンソースに置いて、コミュニティからのバグ修正と提案が行えるようにしています。フィードバックは Khronos フォーラムでも受け付けています。 お時間をいただきありがとうございます！

So, that's OpenCL in a nutshell, I hope you have found this overview useful! I have put some links to OpenCL resources here, and all these slides - and more - are available on the Khronos website. For example, the Working Group has placed the specifications and conformance tests in open source on GitHub to enable bug fixes and suggestions from the community, and any feedback is also welcome on the Khronos forums. Thank you for your time! **9:40**