インテル® oneAPI ツールキット アップデート

バージョン 2023 の注目機能

内容

- インテル® oneAPI および AI ツール 2023 の新機能と最新情報
- インテル® oneAPI DPC++/C++ コンパイラーのアップデート
- NVIDIA* および AMD* GPU 向け CodePlay コンパイラー・プラグイン

インテル® oneAPI ツールキット 2023 登場

インテル® oneAPI ツールキット 2021
インテル® oneAPI ツールキット 2022

HPC & データセンター

AI & ビジュアライゼーション

組込みシステム & IoT

設計

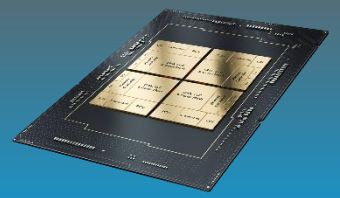
デバッグ

チューニング

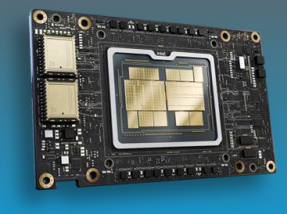
高い処理能力が求められる現代のアプリケーション

今日のパフォーマンス要件を満たすためには多様なアクセラレーターが必要
開発者の 48% が 2 種類以上のプロセッサーやコアを使用する
ヘテロジニアス･システムをターゲットにしている[1]

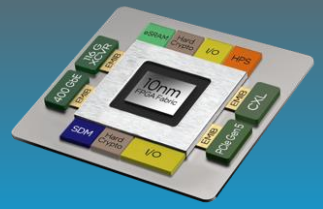
CPU



GPU



FPGA



その他のアクセラレーター

開発者の課題: 複数のアーキテクチャー、ベンダー、プログラミング･モデル

oneAPI

オープンな標準ベースのマルチアーキテクチャー･プログラミング

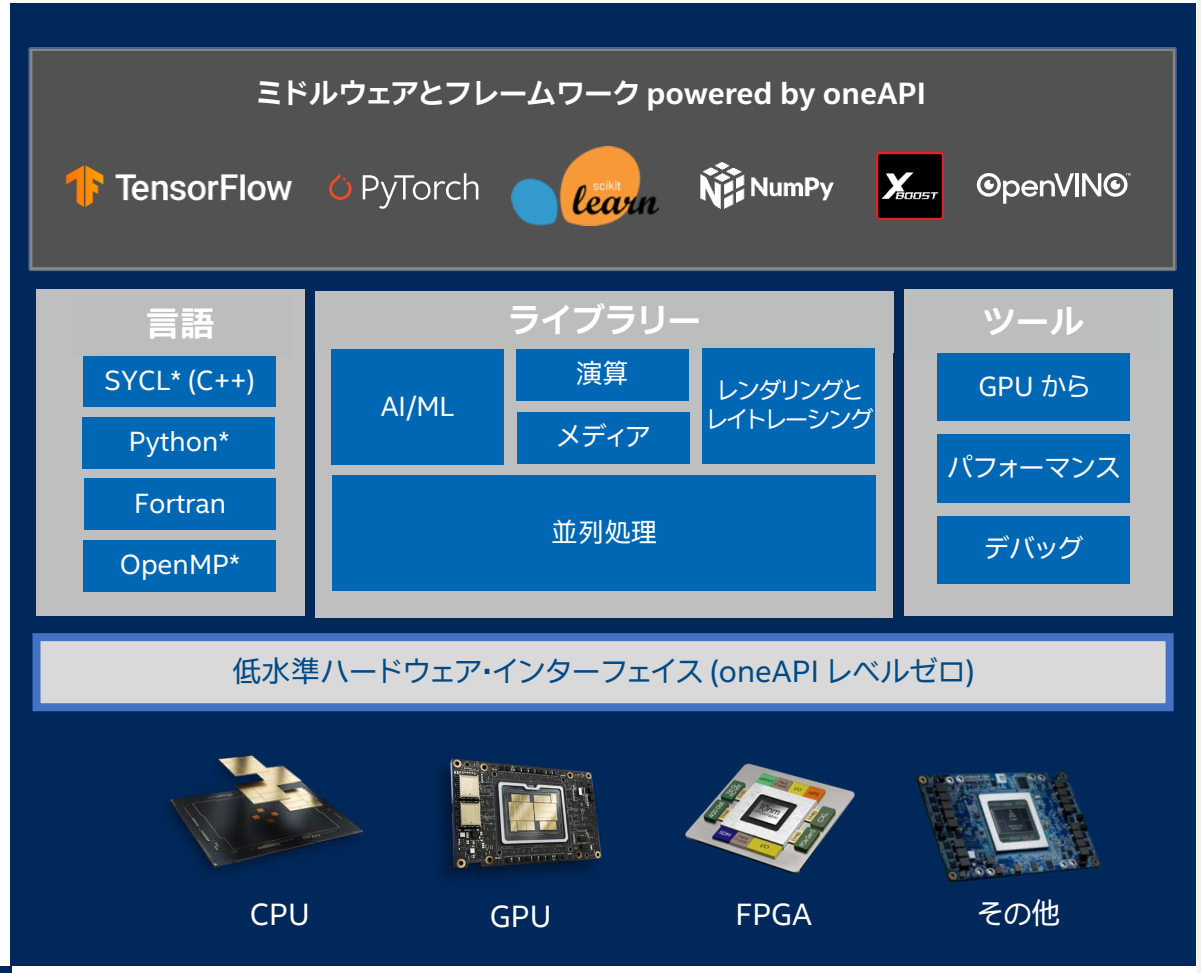
1. 2022年6月に Evans Data が実施した「Global Development Survey Report 22.1」

oneAPI をサポートするインテルの開発ツール

CPU からアクセラレーターまでカバーする優れたツールの完全なセット

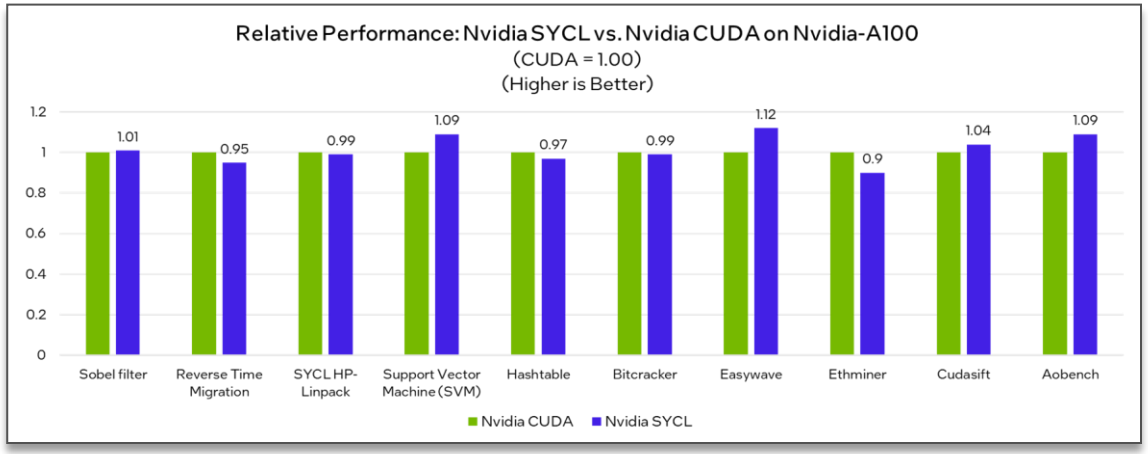
oneAPI

- 高度なコンパイラー、ライブラリー、解析ツール、デバッグツール、移行ツール
- C、C++ with SYCL*、Python*、Fortran、MPI、OpenMP* をフルサポート
- インテル® Advisor はコードを記述する前にデバイスターゲットの組み合わせを決定
- インテルのコンパイラーはコードを最適化してマルチアーキテクチャーでワークロード分散を最大限に活用
- インテル® VTune™ プロファイラーは hotspot を解析してコードのパフォーマンスを最適化
- インテル® AI ツールは主要な DL & ML フレームワークをサポート



SYCL* で高速化する
Khronos Group の標準仕様

- オープンで標準ベース
- マルチアーキテクチャー・パフォーマンス
- ベンダー依存からの解放
- NVIDIA* GPU 上のネイティブ CUDA* に匹敵するパフォーマンス
- 広く使用されている C++ 言語拡張
- オープンソースの SYCLomatic (英語) やインテル® DPC++ 互換性ツールによる迅速なコードの移行



Testing Date: Performance results are **based on testing by Intel as of August 15, 2022** and may not reflect all publicly available updates.

Configuration Details and Workload Setup: Intel® Xeon® Platinum 8360Y CPU @ 2.4GHz, 2 socket, Hyper Thread On, Turbo On, 256GB Hynix DDR4-3200, ucode 0x000363. GPU: Nvidia A100 PCIe 80GB GPU memory. Software: SYCL open source/CLANG 15.0.0, CUDA SDK 11.7 with NVIDIA-NVCC 11.7.64, cuMath 11.7, cuDNN 11.7, Ubuntu 22.04.1. SYCL open source/CLANG compiler switches: -fsycl-targets=nvptx64-nvidia-cuda, NVIDIA NVCC compiler switches: -O3 --gencode arch=compute_80, code=sm_80. Represented workloads with Intel optimizations.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Performance varies by use, configuration, and other factors. Learn more at www.Intel.com/PerformanceIndex. Your costs and results may vary.

アーキテクチャー	インテル | NVIDIA | AMD CPU/GPU | RISC-V | ARM Mali* | PowerVR* | Xilinx

インテル® oneAPI ツールキット

oneAPI

インテル® oneAPI ベース・ツールキット

intel. 1 oneAPI BASE TOOLKIT

C/C++、SYCL*、OpenMP*、Python* アプリケーションを構築するためのハイパフォーマンス・ライブラリーとツールの基本セット

ドメイン固有のアドオン・ツールキット

intel. 1 oneAPI HPC TOOLKIT

HPC 開発者向け

インテル® oneAPI HPC ツールキット
スケーラブルで高速な Fortran、OpenMP* および MPI アプリケーションを開発

intel. 1 oneAPI IoT TOOLKIT

エッジ & IoT 開発者向け

インテル® oneAPI IoT ツールキット
ネットワークのエッジで実行する、効率的な、信頼性の高いソリューションを構築

intel. 1 oneAPI RENDERING TOOLKIT

映像クリエーター、科学者、エンジニア向け

インテル® oneAPI レンダリング・ツールキット
ハイパフォーマンスで高忠実度のビジュアライゼーション・アプリケーションを作成

ツールキット powered by oneAPI

intel. AI ANALYTICS TOOLKIT

AI 開発者とデータ・サイエンティスト向け

インテル® AI アナリティクス・ツールキット
最適化された DL & ML フレームワークとハイパフォーマンスの Python* ライブラリーでエンドツーエンドのマシンラーニングとデータサイエンスのパイプラインを高速化

OpenVINO™

ディープラーニング推論の開発者向け

インテル® ディストリビューションの OpenVINO™ ツールキット
エッジからクラウドまでハイパフォーマンスな推論とアプリケーションをデプロイ

ダウンロード: xlsoft.com/jp/products/intel/oneapi
または**インテル® デベロッパー・クラウド** (英語) で実行

インテル® oneAPI ツール 2023 とインテル® AI ツール 2023 – 特長

強力な新しいアーキテクチャーに最適化された標準ベースのサポートを提供

インテルの今後の CPU と GPU アーキテクチャーのポートフォリオに最適化されたサポートを提供

第 4 世代インテル® Xeon® スケーラブル・プロセッサー
インテル® アドバンスト・マトリクス・エクステンション、インテル® クイックアシスト・テクノロジー、インテル® AVX-512、bfloat16 などを搭載

インテル® Xeon® CPU マックス・シリーズ
高帯域幅メモリー搭載

インテル® データセンター GPU
ハードウェア AV1 エンコーダー内蔵のフレックス・シリーズと、データ型の柔軟性、インテル® Xe マトリクス・エクステンション、ベクトルエンジン、Xe リンク、その他の機能を搭載したマックス・シリーズ

コンパイラーと SYCL* サポート

インテル® oneAPI DPC++/C++ コンパイラー
CPU と GPU オフロードのパフォーマンスを向上、コードの移植性と生産性を高めるため SYCL* 言語サポートを拡大

インテル® oneAPI DPC++ ライブラリー (インテル® oneDPL) (英語)
ヒープとソート・アルゴリズムの追加により SYCL* カーネルでの C++ 標準ライブラリーのサポートを拡大し、OpenMP* を使用したスレッドレベルの並列処理に対応

インテル® DPC++ 互換性ツール (英語)
オープンソースの SYCLomatic プロジェクトをベースにしており、ランタイム、ドライバー、cuBLAS、cuDNN を含む、CUDA* ライブラリー API の移行を向上

インテル® Fortran コンパイラー
Co-Array を実装、MPI や OpenMP* などの外部 API の必要性を排除、OpenMP* 5.x のオフロード機能を拡張、DO CONCURRENT GPU オフロードを追加、ソースレベルのデバッグの最適化を向上

パフォーマンス・ライブラリー

インテル® oneAPI マス・カーネル・ライブラリー
BLAS および FFT の CUDA* ライブラリー関数 API の互換性を向上、第 4 世代インテル® Xeon® スケーラブル・プロセッサー (開発コード名 Sapphire Rapids) でインテル® XMX を活用して TF32、FP16、BF16、INT8 データ型の行列乗算計算を最適化、SYCL*、C/Fortran、OpenMP* オフロード・プログラミング向けインターフェイスを提供

インテル® oneAPI スレッディング・ビルディング・ブロック
parallel_sort の最新の C++ 標準のサポートと使用法を改善、同時に複数の task_arena を呼び出したときの競合を減らすように同期メカニズムを改善、Microsoft* Visual Studio* 2022 および Windows Server* 2022 をサポート

インテル® oneAPI ビデオ・プロセシング・ライブラリー
インテル® データセンター GPU フレックス・シリーズのハードウェア AV1 コーデックとインテル® Arc™ グラフィックスをサポート、OS のサポートが拡張され新たに RHEL9、CentOS* Stream 9、SLES15Sp4、Rocky 9 Linux* をサポート、sample_multi_transcode の並列エンコーディング機能を追加

インテル® oneAPI ツール 2023 とインテル® AI ツール 2023 – 特長

強力な新しいアーキテクチャーに最適化された標準ベースのサポートを提供

解析とデバッグ

インテル® VTune™ プロファイラー
アプリケーション・パフォーマンス・スナップショット機能により MPI のインバランス問題を特定、使用率、帯域幅の消費、その他の問題について、Xe リンクのクロスカード・トラフィックを可視化、第 4 世代インテル® Xeon® スケーラブル・プロセッサー (開発コード名 Sapphire Rapids)、インテル® データセンター GPU マックス・シリーズ (開発コード名 Ponte Vecchio)、第 13 世代インテル® Core™ プロセッサーをサポート

インテル® Advisor
メモリー、キャッシュ、計算のボトルネックを特定して優先度を設定し、原因を理解できるインテル® データセンター GPU マックス・シリーズの自動ルーフライン解析を追加、CPU から GPU へのオフロードのデータ転送再利用コストを最適化する実用的な推奨事項を提供

AI とアナリティクス

インテル® AI アナリティクス・ツールキット (英語)
分散トレーニングを除き、Windows* で Linux* と完全に同等でネイティブな実行を実現 (GPU のサポートは 2023年第一四半期を予定)

インテル® oneAPI ディープ・ニューラル・ネットワーク・ライブラリー (英語)
インテル® AMX、インテル® AVX-512、VNNI、bfloat16 を含む、第 4 世代インテル® Xeon® スケーラブル・プロセッサーの高度な機能を利用して、さらに優れた CNN パフォーマンスを提供

Modin 向けインテル® ディストリビューション (英語)
バックエンドで新しいヘテロジニアス・データ・カーネル (HDK) ソリューションと統合して、負荷の低い計算リソースから負荷の高いまたは分散計算リソースまで AI ソリューションを拡張

インテル® ディストリビューションの Python*
GPU への「計算-フォロー-データ」モデルの拡張、ライブラリーとフレームワーク間のデータ交換、NumPy* および Numba パッケージのデータ並列拡張などのベータ機能を含む

レンダリングとビジュアル・コンピューティング

インテル® oneAPI レンダリング・ツールキット
CPU で高速な SIMD パフォーマンスを実現するインテル® インプリシット SPMD プログラム・コンパイラー・ランタイム・ライブラリーを含む

インテル® オープン・ボリューム・カーネル・ライブラリー
VDB ボリュームのメモリーレイアウト効率を向上、インテル® AVX-512 の 8-wide の CPU デバイスモードを追加

インテル® OSPRay およびインテル® OSPRay Studio (英語)
メッシュ・ジオメトリー、プリミティブ、オブジェクトのマルチセグメント・デフォメーション・モーション・ブラー機能、メッシュおよびサブディビジョン・ジオメトリーの face-varying 属性、フォトメトリック・ライト・タイプなどの新しいライト機能、AI トレーニング向けのセグメンテーション画像を作成するインスタンス ID バッファーなどを追加

2022年12月に発表された新機能と拡張機能 - 詳細: バージョン 2023 の機能

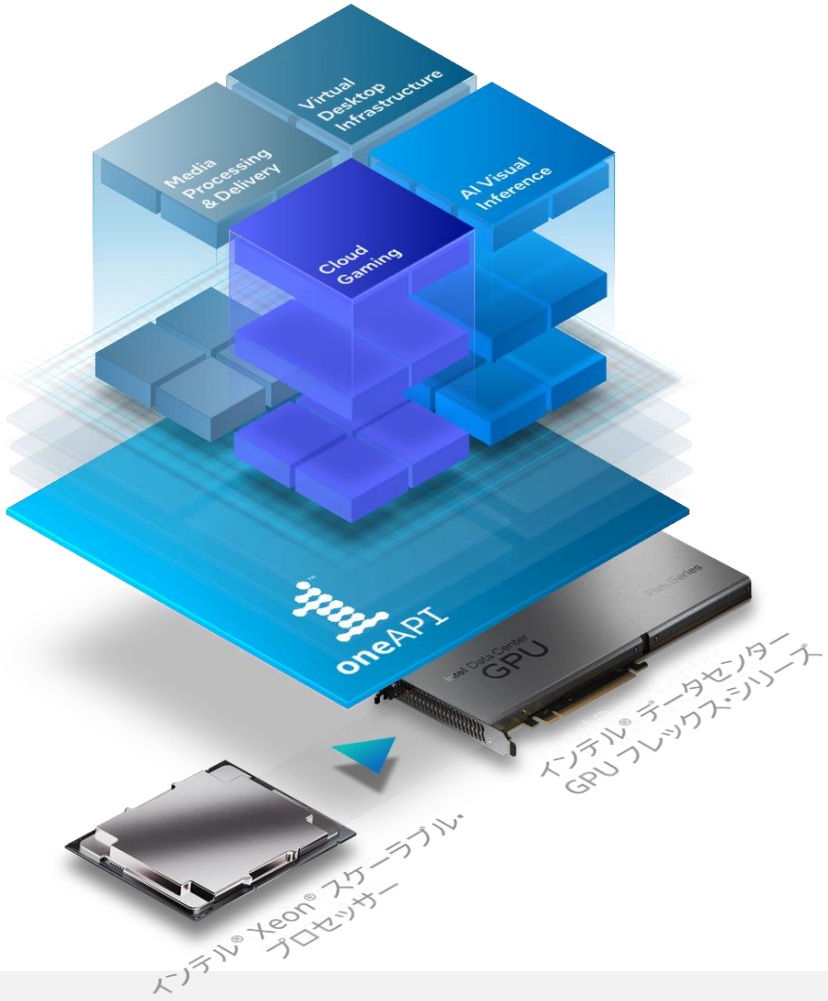
インテル® データセンター GPU フレックス・シリーズ向け開発ツール

メディア、視覚推論、およびクラウド・ワークロードをマルチアーキテクチャーで高速化

GPU には**インテル® oneAPI ビデオ・プロセシング・ライブラリー**、CPU 速度のメディア処理、配信、クラウド・ゲーム・ストリーミングには GStreamer + FFmpeg を使用

インテル® oneAPI DPC++/C++ コンパイラーと**インテル® oneVPL** (どちらもインテル® oneAPI ベース・ツールキットに含まれる) は、同じコードで AI 視覚推論ワークロードを CPU と GPU 間で最大 150TOPS INT8 にスケールアップ

インテル® VTune™ プロファイラーは、アプリケーションのパフォーマンスを解析し、最も時間のかかる GPU コード領域特定し、SYCL*、OpenCL* コード、Microsoft* DirectX*、OpenMP* オフロードコードの GPU オフロードスキーマとデータ転送を最適化して、計算集約型のタスクを高速化



インテル® グラフィックス・パフォーマンス・アナライザー (英語) は CPU と GPU 上でビジュアル・コンピューティングとグラフィックスのワークロードを最適化

インテル® ディストリビューションの OpenVINO™ ツールキット powered by oneAPI は TensorFlow* や PyTorch* などの一般的なディープラーニング・フレームワークをサポートしており、AI 視覚推論の効率を高め、迅速なデプロイを実現

インテル® AI アナリティクス・ツールキット powered by oneAPI は、最適化された DL/ML フレームワークとハイパフォーマンスの Python* ライブラリーにより、エンドツーエンドのマシンラーニングとデータサイエンスのパイプラインを高速化

インテル® データセンター GPU マックス･シリーズ向け開発ツール

HPC と AI で優れたパフォーマンスを発揮するマルチアーキテクチャー･アプリケーションを開発

HPC を高速化

インテル® oneAPI DPC++/C++ コンパイラー、インテル® Fortran コンパイラー、パフォーマンス･ライブラリー (インテル® oneMKL、インテル® oneDNN、インテル® MPI ライブラリー) は、インテル® Xe マトリクス･エクステンション (インテル® XMX) によるアクセラレーションとインテル® Xe リンクによる GPU 間の直接通信を利用して、複数の GPU で実行するアプリケーションを高速化

解析とデバッグツールは最も時間のかかる GPU コード領域を特定することで、開発者が計算集約型のタスクを解析して高速化し、ディスクリート GPU 上のアプリケーションの正当性を確保し、効率良くデバッグできるように支援。インテル® GPU 上で最適に動作するコードを設計するための実用的なアドバイスを提供

AI と DL 推論を高速化

インテル® oneAPI ベース･ツールキットに含まれる **インテル® oneDNN** はインテル® XMX を活用してディープラーニング･フレームワークの AI トレーニング/推論を高速化

インテル® OpenVINO™ ツールキット powered by oneAPI は AI 視覚推論を合理化し、迅速なデプロイを実現

どちらのツールも、TensorFlow* や PyTorch* などの一般的なディープラーニング･フレームワークをサポート。インテル® AI アナリティクス･ツールキットの各種ツールで近々 GPU をサポート予定

コード移行ツールでマルチアーキテクチャー･コードを効率良く作成

CUDA* コードを SYCL* に移行して、インテル® データセンター GPU マックス･シリーズを含む複数のベンダーのアーキテクチャーに容易に移植できるように単一のソースコード･ベースを作成

オープンソースの SYCLomatic (英語) をベースとする **インテル® DPC++ 互換性ツール** (英語) がほとんどのプロセスを自動化し、継続的なコードの保守にかかる時間を大幅に短縮

近日リリース予定 – ハイパフォーマンスかつ高忠実度のレイトレーシングとレンダリング

インテル® Embree (英語) は C++ with SYCL* に完全に移行され、ハードウェア･レイトレーシング･ユニットと拡張メモリー容量を活用して、大規模なデータセットのレンダリング･パフォーマンスを向上

インテル® オープン･イメージ･デノイズ (英語) は、インテル® XMX を利用して高品質でハイパフォーマンスなノイズ除去機能と AI 最適化を提供

インテル® oneAPI DPC++/C++ コンパイラー アップデート

データ並列 C++

SYCL* の oneAPI 実装

DPC++ = ISO C++ と Khronos SYCL* およびコミュニティーによる拡張

選択の自由: 将来に対応したプログラミング・モデル

- 複数のハードウェア・ターゲットでコードを再利用可能
- 特定のアクセラレーター向けのカスタム・チューニングが可能
- 専用の言語に代わる、オープンな業界全体の代替手段

DPC++ = ISO C++ と Khronos SYCL* およびコミュニティーによる拡張

- データ並列プログラミングの生産性を向上
- 標準 C++ と同等のネイティブな高水準言語パフォーマンスを提供
- Khronos Group の SYCL* を継承し、データ並列処理とヘテロジニアス・プログラミングをサポート

コミュニティー・プロジェクトを通じて言語の拡張を推進

- データ並列プログラミングを簡素化する拡張機能を提供
- オープンな共同開発により継続的に進化

オープンソースおよびインテルの DPC++/C++ コンパイラーはインテルの CPU、GPU、FPGA をサポートしています。
Codeplay* は、NVIDIA* GPU をターゲットとする DPC++ コンパイラー (英語) を発表しています。

インテル® oneAPI ツールキットとコンパイラー

重要な変更点

1. DPCPP コンパイラー･ドライバーが廃止されます

Intel(r) oneAPI Tools

```
C> dpcpp
icpx: warning: use of 'dpcpp' is deprecated and will be removed in a future release.
Use 'icpx -fsycl' [-Wdeprecated]
icpx: error: no input files
```

- makefile やコンパイル環境で設定されているコンパイラー･ドライバーを **icpx** に変更してください
- その際コンパイラー･オプションに **–fsycl** を追加することを忘れないでください

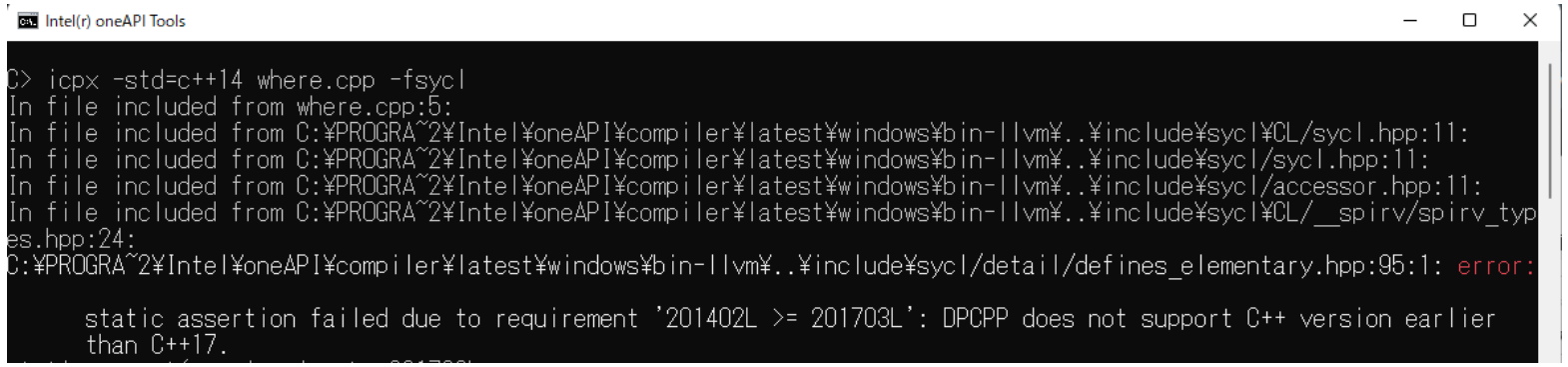
Intel(r) oneAPI Tools

```
C> icpx -v
Intel(R) oneAPI DPC++/C++ Compiler 2023.0.0 (2023.0.0.20221201)
```

インテル® oneAPI ツールキットとコンパイラー

重要な変更点

2. コンパイラーは C++17 をデフォルトの C++ 言語として使用するようになりました
   - 言語バージョンのデフォルトが変更されたことで、表示される警告を非表示にしたい場合、**-std=c++14** オプションを追加します
   - ただし、2023 バージョンのコンパイラーでは、C++17 以前の言語モードでの SYCL* コンパイルが廃止されました



3. このリリースは、以前のリリースと下位互換性がありません。既存の SYCL* アプリケーション は再コンパイルしないと新しいランタイムで動作しません

インテル® oneAPI ツールキットとコンパイラー

重要な変更点

4. device と platform のデフォルトが default_selector_v になりました

```
Intel(r) oneAPI Tools
C> icpx -fsycl device_old.cpp
device_old.cpp:18:28: warning: 'default_selector' is deprecated: Use the callable sycl::default_selector_v instead.
      [-Wdeprecated-declarations]
  output_dev_info( device{ default_selector{}}, "default_selector" );
                           ^
```

host_selector が廃止されました。使用するとエラーとなります

```
Intel(r) oneAPI Tools

C> sycl-ls
[opencl:acc:0] Intel(R) FPGA Emulation Platform for OpenCL(TM), Intel(R) FPGA Emulation Device 1.2 [2022.15.12.0.01_
081451]
[opencl:cpu:1] Intel(R) OpenCL, 12th Gen Intel(R) Core(TM) i7-12700K 3.0 [2022.15.12.0.01_081451]
[opencl:gpu:2] Intel(R) OpenCL HD Graphics, Intel(R) UHD Graphics 770 3.0 [31.0.101.4091]
[opencl:gpu:3] Intel(R) OpenCL HD Graphics, Intel(R) Arc(TM) A380 Graphics 3.0 [31.0.101.4091]
[ext_oneapi_level_zero:gpu:0] Intel(R) Level-Zero, Intel(R) UHD Graphics 770 1.3 [1.3.25143]
[ext_oneapi_level_zero:gpu:1] Intel(R) Level-Zero, Intel(R) Arc(TM) A380 Graphics 1.3 [1.3.25143]
```

デバイスセレクターを操作する例

```
#include <CL/sycl.hpp>
#include <iostream>
#include <string>

using namespace sycl;

void output_dev_info( const device& dev, const std::string& selector_name) {
  std::cout << selector_name << “: Selected device: ” <<
     dev.get_info<info::device::name>() << "¥n";
  std::cout << "                    -> Device vendor: " <<
     dev.get_info<info::device::vendor>() << "¥n";}
```

output_dev_info(device{ default_selector_v}, "default_selector");

```
int main() {
output_dev_info( device{ default_selector{}}, "default_selector" );
output_dev_info( device{ host_selector{}}, "host_selector" );
output_dev_info( device{ cpu_selector{}}, "cpu_selector" );
output_dev_info( device{ gpu_selector{}}, "gpu_selector" );
output_dev_info( device{ accelerator_selector{}}, "accelerator_selector" );  return 0;}
```

インテル® oneAPI DPC++/C++ コンパイラー 2023.0 リリースノート

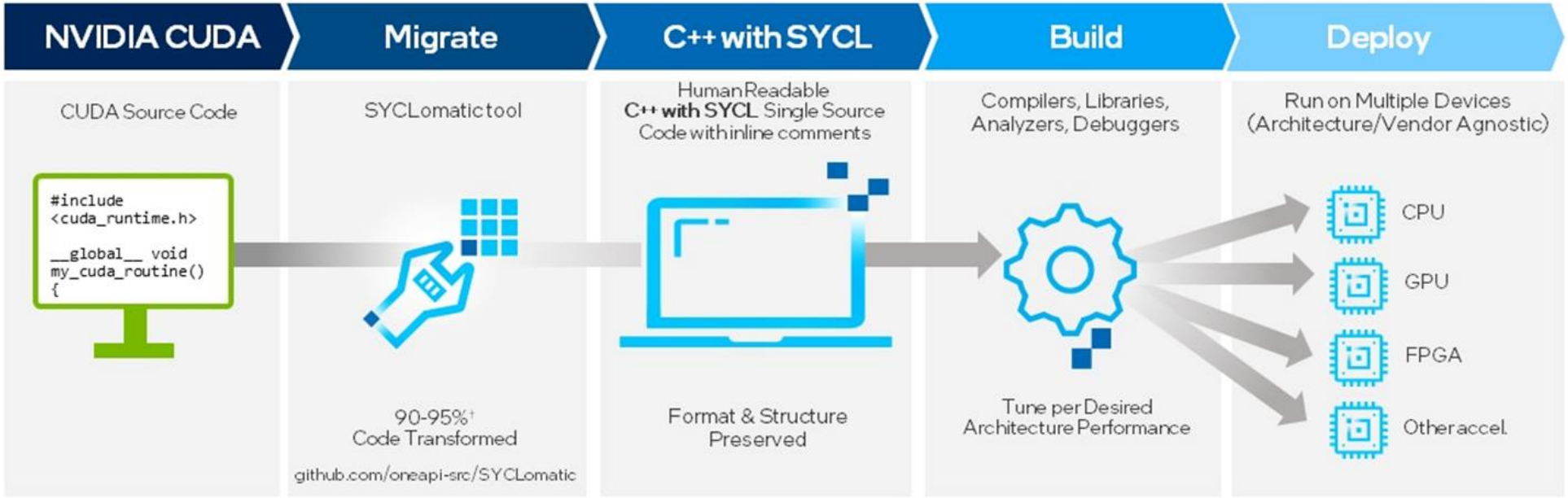
https://jp.xlsoft.com/documents/intel/compiler/dpc/23.0/intel-oneapi-dpc-c-compiler-release-notes.pdf

新機能、変更された機能、注意事項、および製品ドキュメントに記述されていない既知の問題については、リリースノートを参照してください

NVIDIA* および AMD* GPU 向け CodePlay コンパイラー・プラグイン

CUDA* から SYCL* への移行が容易に

オープンソースの SYCLomatic ツールによりコード移行時間を短縮



すでに CUDA* で記述されているコードを C++ with SYCL* に移行する開発者を支援、可能な場合は**人間が解読可能な**コードを生成

通常はコードの 90-95% を自動的に移行[1]

開発者がアプリケーションの移行を完了できるように支援するインラインコメントを提供

インテル® DPC++ 互換性ツールはインテル® oneAPI ベース･ツールキットに含まれるインテルの実装

[1]出典: 2021年9月現在のインテルによる推定。Rodinia、SHOC、PENNANT など、70 種類の HPC ベンチマークとサンプルの測定結果に基づいています。結果は異なることがあります。

NVIDIA* GPU および AMD* GPU 向け Codeplay コンパイラー・プラグイン

インテル® oneAPI ベース・ツールキットに NVIDIA* GPU と AMD* GPU サポートを追加

NVIDIA* GPU および AMD* GPU 向け oneAPI

- インテル® DPC++/C++ コンパイラーへの最新のバイナリープラグインを Codeplay から無料でダウンロード可能
  - NVIDIA* GPU
  - AMD* GPU (ベータ)
- インテル® oneAPI ベース・ツールキットのリリースと同時に利用可能
- プラグインは四半期ごとに oneAPI と同期して更新される

優先サポート

- インテル® DPC++/C++ コンパイラーに対する有効なインテルの優先サポートの保有者のみ、インテル、Codeplay、およびチャネルから追加で購入可能
- インテルが受け付け、Codeplay がサポートを提供
- 古いバージョンのプラグインへアクセス可能

NVIDIA* GPU プラグイン (英語)

AMD* GPU プラグイン (英語)

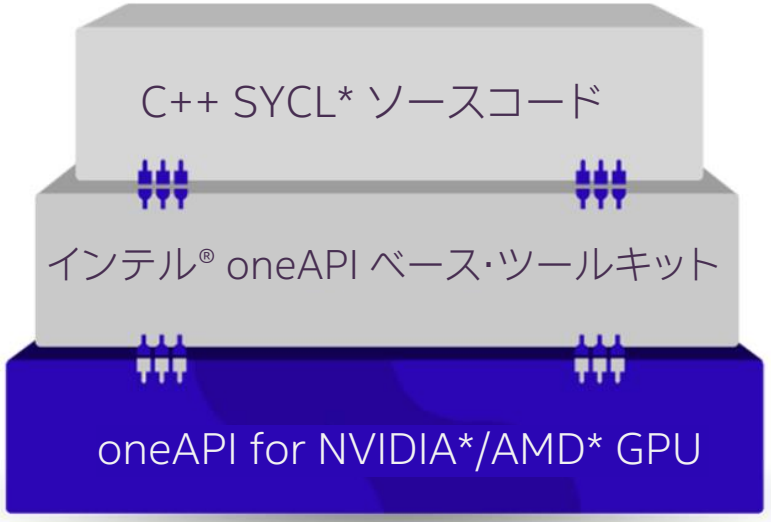
Codeplay のブログ (英語)

Codeplay のプレスリリース (英語)

oneAPI for NVIDIA*/AMD* GPU とは

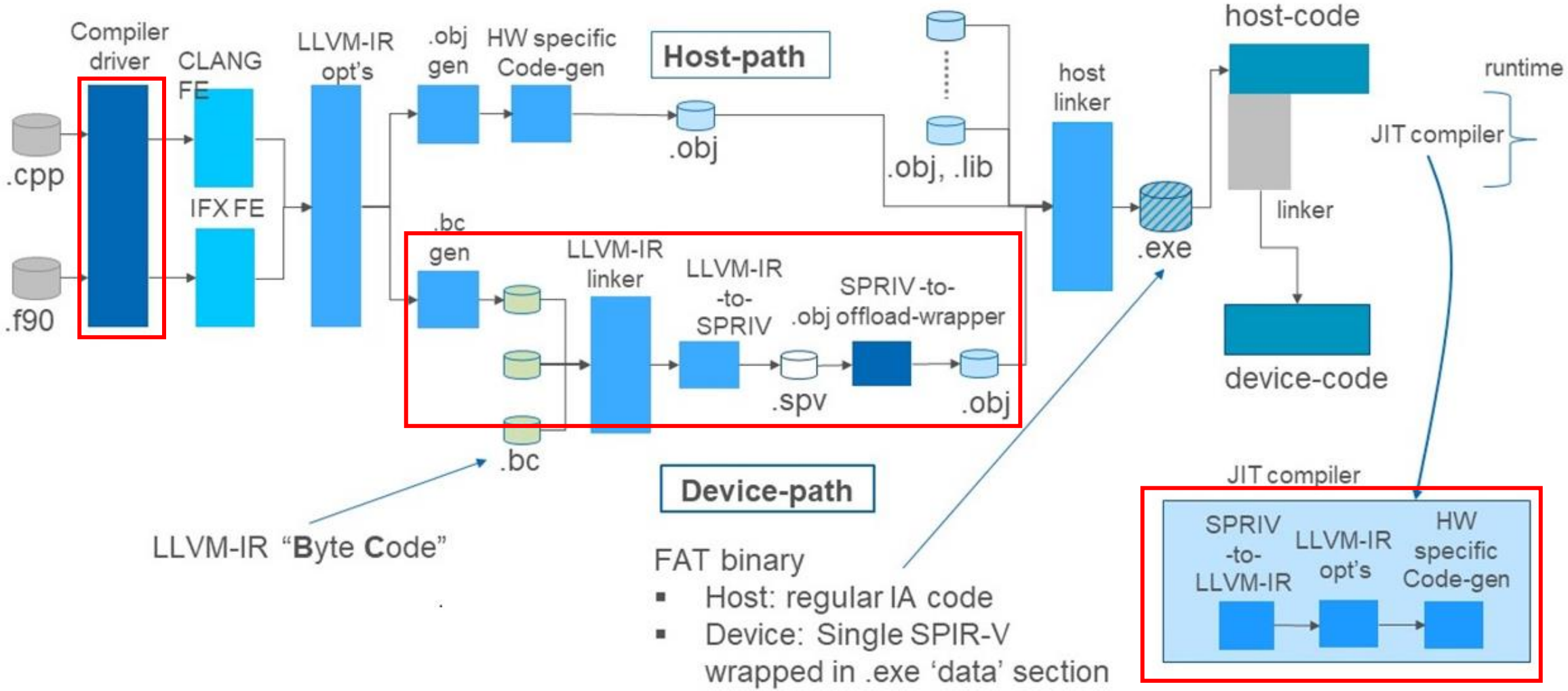
oneAPI for NVIDIA*/AMD* GPU は、開発者が DPC++/SYCL* を利用して oneAPI アプリケーションを作成し、それらを NVIDIA*/AMD* GPU 上で実行できるようにするインテル® oneAPI ツールキット向けのプラグインです

- oneAPI for NVIDIA*/AMD* GPU を使用すると、CUDA*/HIP をサポートする NVIDIA*/AMD* ベースの GPU をターゲットにすることができます
- このプラグインは、インテル® oneAPI DPC++/C++ コンパイラーを含む、既存のインテル® oneAPI ツールキットと併用して、SYCL* コードをビルドし互換性のある NVIDIA*/AMD* GPU で実行できます



† AMD* GPU 向けはベータ版です

ジャストインタイム (JIT) コンパイル



動作の仕組み

NVIDIA* GPU および AMD* GPU ターゲットの DPC++ の実装には 2 つの主要コンポーネントがあります

1. いくつかの clang++ コンパイラーのドライバーパッチ

コンパイラー・ドライバーのパッチは、ターゲットトリプルを識別し、LLVM プロジェクトの既存の CUDA* コンパイラーを使用して、デバイスイメージをビルドするアクションをトリガーすることで、DPC++ フロントエンドが NVIDIA* GPU 向けのビルドをできるようにします

2. ランタイムプラグイン

ランタイムプラグインは、SYCL* ランタイムが NVIDIA* プラットフォームと AMD* プラットフォーム上のネイティブ API (それぞれ CUDA* と HIP ドライバー) を呼び出せるようにするものです。DPC++ コンパイラーは、oneAPI ディストリビューションの SYCL* ランタイムと自動的にリンクし、NVIDIA* プラグインや AMD* プラグインが利用できる場合、ランタイムがそれらを選択して実行します

サポートされるプラットフォーム

このリリースは、次のプラットフォームで検証されています

GPU ハードウェア	GPU アーキテクチャー	検証済みのオペレーティング・システム	CUDA*/ ROCm*	GPU ドライバー
NVIDIA* GeForce RTX* 2060	Turing* - sm_75	Ubuntu* 20.04.5 LTS	11.7	515.65.01
NVIDIA* A100 (PCIe*) 40GB	Ampere - sm_80	Ubuntu* 20.04.4 LTS	11.7	515.86.01
AMD Radeon* Pro W6800	gfx1030	Ubuntu* 20.04.5 LTS	4.5.2	21.40.2.40502

- このリリースは各種 NVIDIA* GPU/CUDA* と AMD*/HIP バージョンで動作するはずですが、CodePlay は評価されていないプラットフォームでの正常な動作を保証するものではありません
- Windows* 上の oneAPI では使用できませんが、Windows* 向けのパッケージは今後リリース予定

oneAPI for NVIDIA* GPU を使用する

前提条件

次のツールキットとツールがインストールされていること:

1. Linux* の C++ 開発ツールの cmake、gcc、g++、make および pkg-config
2. インテル® oneAPI ベース・ツールキット 2023.0
3. NVIDIA* GPU ドライバーと CUDA* ソフトウェア・スタック
4. oneAPI for NVIDIA* GPU をダウンロードしてインストール

C++ 開発ツールのインストール

Ubuntu*

```
$ sudo apt update
$ sudo apt -y install cmake pkg-config build-essential
```

Red Hat* と Fedora*

```
$ sudo yum update
$ sudo yum -y install cmake pkgconfig
$ sudo yum groupinstall "Development Tools"
```

```
$ which cmake pkg-config make gcc g++
/usr/bin/cmake
/usr/bin/pkg-config
/usr/bin/make
/usr/bin/gcc
/usr/bin/g++
```

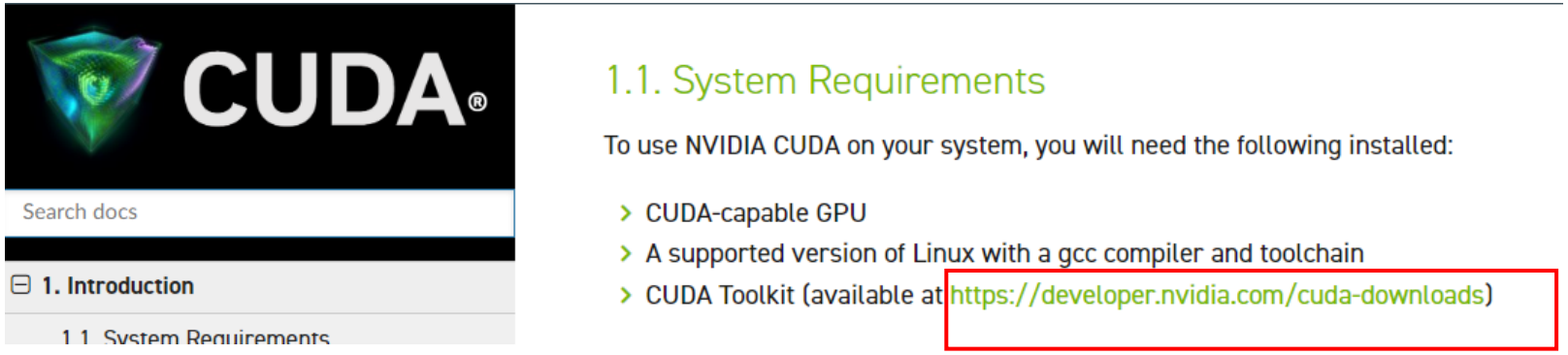
SUSE*

```
$ sudo zypper update
$ sudo zypper --non-interactive install cmake pkg-config
$ sudo zypper --non-interactive install pattern devel_C_C++
```

NVIDIA* GPU ドライバーと CUDA* ソフトウェア･スタック

「Linux* 向けの NVIDIA* CUDA* インストール･ガイド」(英語) の手順に従って、NVIDIA* GPU ドライバーと CUDA* ソフトウェア･スタックをインストールします

https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html (英語)



CUDA* ツールキットの入手

Operating System	Linux, Windows
Architecture	x86_64, ppc64le, arm64-sbsa, aarch64-jetson
Distribution	CentOS, Debian, Fedora, KylinOS, OpenSUSE, RHEL, Rocky, SLES, Ubuntu, WSL-Ubuntu
Version	18.04, 20.04, 22.04
Installer Type	deb (local), deb (network), runfile (local)

Download Installer for Linux Ubuntu 20.04 x86_64

The base installer is available for download below.

Base Installer

Installation Instructions:

```
$  wget https://developer.download.nvidia.com/compute/cuda/12.0.1/local_installers/cuda_12.0.1_525.85.12_linux.run
$  sudo sh cuda_12.0.1_525.85.12_linux.run
```

CUDA* ツールキットのインストール

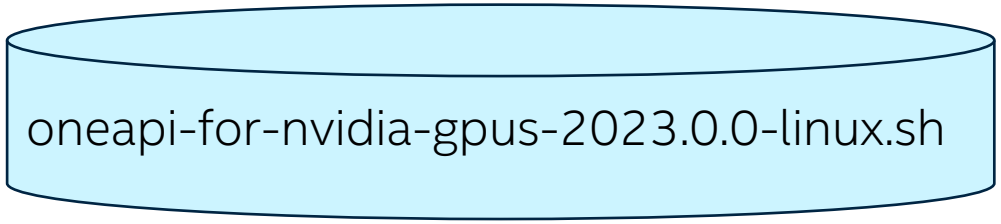
```
kiyo@kiyo-Virtual-Machine: ~

kiyo@kiyo-Virtual-Machine:~$ sudo sh cuda_12.0.1_525.85.12_linux.run
===========
= Summary =
===========

Driver:   Installed
Toolkit:  Installed in /usr/local/cuda-12.0/

Please make sure that
 -   PATH includes /usr/local/cuda-12.0/bin
 -   LD_LIBRARY_PATH includes /usr/local/cuda-12.0/lib64, or, add /usr/local/cud
a-12.0/lib64 to /etc/ld.so.conf and run ldconfig as root

To uninstall the CUDA Toolkit, run cuda-uninstaller in /usr/local/cuda-12.0/bin
To uninstall the NVIDIA Driver, run nvidia-uninstall
Logfile is /var/log/cuda-installer.log
kiyo@kiyo-Virtual-Machine:~$
```

PATH と **LD_LIBRARY_PATH** に CUDA* ツールキットのインストール先を追加

oneAPI for NVIDIA* GPU のインストーラーをダウンロード

https://developer.codeplay.com/products/oneapi/nvidia/download/ (英語)

- CodePlay 社の oneAPI for NVIDIA* GPU インストーラーのダウンロード・サイトにアクセスするとユーザー登録 (無料) が求められる場合があります
- ダウンロード・ページに移動したら、インテル® oneAPI ツールキットのバージョンとプラットフォームを選択します
- 利用許諾に同意すると自己解凍型のインストーラーがダウンロードされます



oneAPI for NVIDIA* GPU のインストール

```
kiyo@kiyo-Virtual-Machine:~$ sudo ./oneapi-for-nvidia*.sh

oneAPI for NVIDIA GPUs 2023.0.0 installer

Found oneAPI DPC++/C++ Compiler 2023.0.0 in /opt/intel/oneapi/.

By installing this software, you accept the oneAPI for NVIDIA GPUs License Agree
ment.

The package will be installed in /opt/intel/oneapi/, Proceed? [Yn]:
Y
* CUDA plugin library installed in /opt/intel/oneapi/compiler/2023.0.0/linux/lib
/.
* License installed in /opt/intel/oneapi/licensing/2023.0.0/.
* Documentation installed in /opt/intel/oneapi/compiler/2023.0.0/documentation/e
n/oneAPI_for_NVIDIA_GPUs/.

Installation complete.
```

インテル® oneAPI ツールキットのインストール先とプラグインのインストール先を確認してください

インストールの成否を確認

- DPC++ CUDA* プラグインのインストールを確認するには、DPC++ の sycl-ls ツールを使用して、SYCL* で利用可能な NVIDIA* GPU があることを確認します
- NVIDIA* GPU が利用できる場合、sycl-ls の出力に次のような情報が表示されます

```
$ [ext_oneapi_cuda:gpu:0] NVIDIA CUDA BACKEND, TITAN RTX 0.0 [CUDA 11.0]
```

```
kiyo@Ubuntu-Z10PE-D8-WS: ~/デスクトップ/Tools
$ sycl-ls
[opencl:acc:0] Intel(R) FPGA Emulation Platform for OpenCL(TM), Intel(R) FPGA Emulation Device 1.2 [2022.15.12.0.01_081451]
[opencl:cpu:1] Intel(R) OpenCL, Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz 3.0 [2022.15.12.0.01_081451]
[ext_oneapi_cuda:gpu:0] NVIDIA CUDA BACKEND, NVIDIA GeForce GTX 1060 6GB 0.0 [CUDA 12.0]
$
```

さあコンパイルしてみましょう

コンパイルする前に

- oneAPI for NVIDIA* GPU プラグインを使用するには、インテル® oneAPI ベース・ツールキットに同梱される `bin-llvm` ディレクトリーにある clang++ コンパイラーを使用します

- `bin-llvm` はデフォルトで PATH に設定されていないため、`setvars.sh` スクリプトに `--include-intel-llvm` オプションを追加して source します

システム全体

```
$ . /opt/intel/oneapi/setvars.sh --include-intel-llvm
```

ローカル

```
$ . ~/intel/oneapi/setvars.sh --include-intel-llvm
```

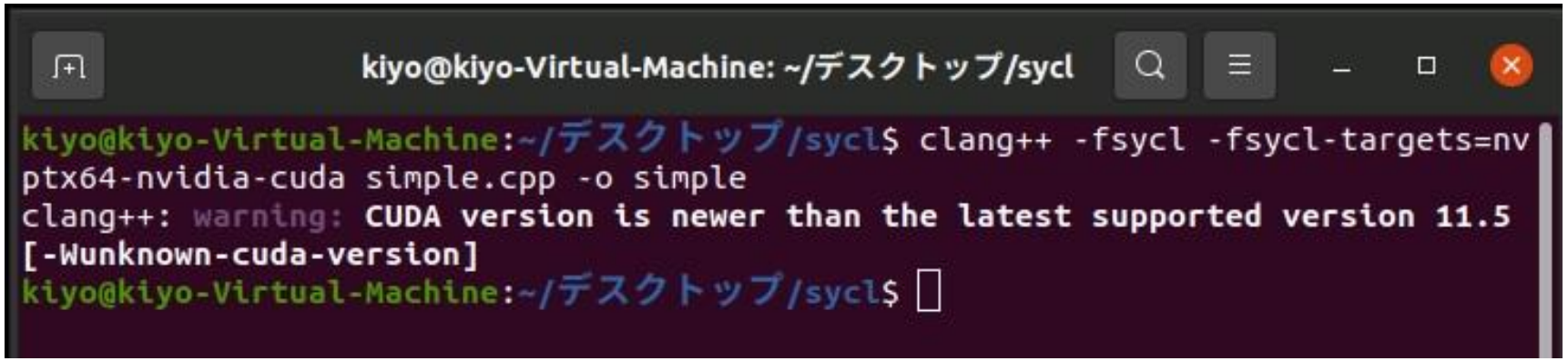
サンプルコード

```
#include <sycl/sycl.hpp>
int main() {
  // カーネルコード内で使用する 4 つの int バッファーを作成
  sycl::buffer<sycl::cl_int, 1> Buffer(4);
  // SYCL* キューを作成
  sycl::queue Queue;
  // カーネルのインデックス空間サイズ
  sycl::range<1> NumOfWorkItems{Buffer.size()};
  // キューへコマンドグループ（ワーク）を送信
  Queue.submit([&](sycl::handler &cgh) {
    // デバイス上のバッファーへの書き込み専用アクセサーを作成
    auto Accessor = Buffer.get_access<sycl::access::mode::write>(cgh);
    // カーネルを実行
    cgh.parallel_for<class FillBuffer>(
        NumOfWorkItems, [=](sycl::id<1> WIid) {
          // インデックスでバッファーを埋めます
          Accessor[WIid] = (sycl::cl_int)WIid.get(0);
        });
  });
```

```
  // ホスト上のバッファーへの読み取り専用アクセサーを作成。
  // キューのワークが完了するのを待機する暗黙のバリア
  const auto HostAccessor = Buffer.get_access<sycl::access::mode::read>();
  // 結果をチェック
  bool MismatchFound = false;
  for (size_t i = 0; i < Buffer.size(); ++i) {
    if (HostAccessor[I] != i) {
      std::cout << "The result is incorrect for element: " << i
                << " , expected: " << i << " , got: " << HostAccessor[i]
                << std::endl;
      MismatchFound = true;
    }
  }
  if (!MismatchFound) {
    std::cout << "The results are correct!" << std::endl;
  }
  return MismatchFound;
}
```

アプリケーションをコンパイル

```
$ clang++ -fsycl -fsycl-targets=nvptx64-nvidia-cuda simple.cpp -o simple
```



インストールされている CUDA* のバージョンによっては、上記のような警告が表示されることがありますが、これは無視してもかまいません

重要なコンパイラー・オプション

- **-fsycl**: C++ ソースファイルを SYCL* モードでコンパイルするようにコンパイラーに指示します。このフラグは暗黙的に C++ 17 を有効にし、SYCL* ランタイム・ライブラリーを自動でリンクします
- **-fsycl-targets=nvptx64-nvidia-cuda**: NVIDIA* GPU をターゲットとして、SYCL* カーネルをビルドすることをコンパイラーに指示します

また、次のフラグを使用して、特定の NVIDIA* アーキテクチャー向けの SYCL* カーネルをビルドすることができます

- -Xsycl-target-backend=nvptx64-nvidia-cuda --cuda-gpu-arch=sm_80

コンパイルのヒント

- CUDA* ツールキットがインストールされていれば、oneAPI for NVIDIA* GPU プラグインがインストールされていなくても、CUDA* ターゲット・オプションを指定してコンパイルできます

```
Intel(r) oneAPI Tools

C> nmake

Microsoft(R) Program Maintenance Utility Version 14.26.28806.0
Copyright (C) Microsoft Corporation.  All rights reserved.

        clang++ -fsycl simple.cpp -o simple86.exe
        clang++ -fsycl -fsycl-targets=nvptx64-nvidia-cuda -DCUDA simple.cpp -o simpleNV.exe
clang++: warning: CUDA version is newer than the latest supported version 11.5 [-Wunknown-cuda-version]
```

上記は、Windows* 10 環境に CUDA* ツールキット 12.0 とインテル® oneAPI ツールキット 2023 のみをインストールしてコンパイルした例です

アプリケーションを実行

- NVIDIA* ターゲット向けに SYCL* アプリケーションをコンパイルしたら、ランタイムが SYCL* デバイスとして NVIDIA* GPU を選択しているか確認する必要があります
- 通常、デフォルトのデバイスセレクターを使用すると、利用可能な NVIDIA* GPU の 1 つが選択されます。しかし、場合によっては、SYCL* アプリケーションを変更して、GPU セレクターやカスタムセレクターなど、より正確な SYCL* デバイスセレクターを設定することもあります

DPC++ CUDA* プラグインでサポートされる CUDA* デバイスのみを許可するには、次のように設定します

```
$ export SYCL_DEVICE_FILTER=cuda
```

実行のヒント

- プラットフォームに NVIDIA* GPU と CUDA* ツールキットがインストールされている環境では、CUDA* ターゲットのバイナリーの実行を開始できます

Intel(r) oneAPI Tools

```
C> nmake run

Microsoft(R) Program Maintenance Utility Version 14.26.28806.0
Copyright (C) Microsoft Corporation.  All rights reserved.

        simple86
The results are correct!
        simpleNV
 No CUDA device found
 No CUDA device found
 No CUDA device found
 No CUDA device found
NMAKE : fatal error U1077: '.¥simpleNV.EXE' : リターン コード '0xc0000409'
Stop.
```

上記の例は、後述する CUDA* デバイスの判定を組み込んだコードを Windows* 環境でビルドして実行した例です

デバイスの識別の例

DPC++ コンパイラーは、oneAPI ディストリビューションの SYCL* ランタイムと自動的にリンクし、NVIDIA* プラグインや AMD* プラグインが利用できる場合、ランタイムがそれらを選択して実行します。アプリケーションが SYCL* デバイスセレクターを使用すると、以下のように NVIDIA* (または AMD*) デバイスを選択できます

```
31    auto CUDASelector = [](sycl::device const &dev) {
32      if (dev.get_platform().get_backend() == sycl::backend::ext_oneapi_cuda) {
33        std::cout << " CUDA device found " << std::endl;
34        return 1;
35      } else {
36        return -1;
37      }
38    };
39    sycl::queue myQueue{CUDASelector};
```

CUDA* デバイス検出されるとメッセージを表示する CUDA* デバイスセレクターの例。
CUDA* デバイスが見つからないと、SYCL* キューはエラーをスローします

実行結果

```
kiyo@Ubuntu-Z10PE-D8-WS: ~/デスクトップ/Tools
$ make
clang++ -fsycl simple.cpp -o simple86
clang++ -fsycl -fsycl-targets=nvptx64-nvidia-cuda -D CUDA simple.cpp -o simpleNV
clang++: warning: CUDA version is newer than the latest supported version 11.5 [-Wunknown-cuda-version]
$ make run
./simple86
The results are correct!
./simpleNV
 CUDA device found
The results are correct!
$
```

Ubuntu 22.04 (インテル® Xeon® E5-2699 v4 + Nvidia GeForce GTX 1060 (Pascal 世代 sm_61 アーキテクチャー)) 環境で実行

SYCL* アプリケーションのデバッグ

SYCL* アプリケーションのホストコードは、C++ アプリケーションとしてデバッグできますが、カーネルデバッグのサポートやツールは、ターゲットデバイスによって異なる可能性があります

インテルの OpenCL* CPU デバイスでのデバッグ

- インテルの OpenCL* CPU デバイスを使用した DPC++ アプリケーションのデバッグについては、『インテル® oneAPI プログラミング・ガイド』の「DPC++ と OpenMP* オフロードプロセスのデバッグ」の節を参照してください

CUDA* デバッガーによるデバッグ

- CUDA* ツールキットには、CUDA* アプリケーションの NVIDIA* GPU カーネルのデバッグをサポートする cuda-gdb デバッガーが付属しています
  cuda-gdb は、DPC++ ext_oneapi_cuda バックエンド用にコンパイルされたカーネルのデバッグにも使用できます

トラブルシューティングと既知の問題 (1)

CUDA* デバイスが搭載されていない環境で、oneAPI for NVIDIA* GPU で生成したバイナリーを実行すると、当然ながら実行に失敗します

```
kiyo@kiyo-Virtual-Machine: ~/デスクトップ/sycl
kiyo@kiyo-Virtual-Machine:~/デスクトップ/sycl$ ./simple
terminate called after throwing an instance of 'sycl::_V1::runtime_error'
  what():  Native API failed. Native API returns: -42 (PI_ERROR_INVALID_BINARY)
 -42 (PI_ERROR_INVALID_BINARY)
Aborted (core dumped)
kiyo@kiyo-Virtual-Machine:~/デスクトップ/sycl$
```

```
kiyo@kiyo-Virtual-Machine: ~/デスクトップ/sycl
$ icpx simple.cpp -fsycl -o simple86
$ ./simple86
The results are correct!
$ clang++ -fsycl -fsycl-targets=nvptx64-nvidia-cuda simple.cpp -o simpleNV
clang++: warning: CUDA version is newer than the latest supported version 11.5
[-Wunknown-cuda-version]
$ ./simpleNV
terminate called after throwing an instance of 'sycl::_V1::runtime_error'
  what():  Native API failed. Native API returns: -42 (PI_ERROR_INVALID_BINARY)
 -42 (PI_ERROR_INVALID_BINARY)
Aborted (core dumped)
$
```

ランタイムエラーではなく、アプリケーションでデバイスを識別してエラーメッセージを表示することもできます

トラブルシューティングと既知の問題 (2)

現在のバージョンの icpx には、NVIDIA* または AMD* ターゲット向けにコンパイルする際に既知の問題があり、次のエラーが出力ます

```
$ make simple_NV
icpx -fsycl -fsycl-targets=nvptx64-nvidia-cuda simple.cpp -o simpleNV
icpx: warning: CUDA version is newer than the latest supported version 11.5 [-Wunknown-cuda-version]
LLVM ERROR: Bitcode output disabled because proprietary optimizations have been performed.
make: *** [makefile:12: simple_NV] Error 1
```

```
LLVM ERROR: Bitcode output disabled because proprietary optimizations have been
performed.
```

これは次のリリースで修正される予定ですが、それまでは NVIDIA* または AMD* GPU をターゲットとする場合、icpx 実行ファイルの代わりに、このガイドで説明されている clang++ 実行ファイルを直接使用します

トラブルシューティングと既知の問題 (3)

sycl-ls の出力にデバイスが表示されない

1. システムに互換性のあるバージョンの CUDA* または ROCm* ツールキット (それぞれ CUDA* と HIP プラグイン向け)、および互換性のあるドライバーがインストールされていることを確認します
2. nvidia-smi または rocm-smi がデバイスを正しく認識できることを確認します
3. プラグインが正しくロードされていることを確認します
   これは、環境変数 SYCL_PI_TRACE に 1 を設定して、sycl-ls を再度実行することで分かります

```
$ SYCL_PI_TRACE=1
$ sycl-ls

SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_opencl.so [
PluginVersion: 11.15.1 ]
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_level_zero.so [
PluginVersion: 11.15.1 ]
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_cuda.so [
PluginVersion: 11.15.1 ]
[ext_oneapi_cuda:gpu:0] NVIDIA CUDA BACKEND, NVIDIA A100-PCIE-40GB 0.0 [CUDA 11.7]
```

トラブルシューティングと既知の問題 (3) 続き …

- CUDA* プラグインには、CUDA* SDK で提供される libcuda.so と libcupti.so が必要です
- HIP プラグインには、ROCm* の libamdhip64.so が必要です

CUDA* または ROCm* のインストールと、環境が適切に設定され、LD_LIBRARY_PATH が上記のライブラリーを検出できる場所を指しているか確認してください

4. SYCL_DEVICE_FILTER または SYCL_DEVICE_ALLOWLIST などのデバイスフィルター環境変数が設定されていないことを確認します (SYCL_DEVICE_FILTER が設定されていると、sycl-ls は警告を表示します)。

トラブルシューティングと既知の問題 (4)

コンパイラー・エラー: “cannot find libdevice (libdevice が見つかりません)”

CUDA* SDK がデフォルトの位置にインストールされていないと、clang++ が SDK を検出できず、コンパイル中にこのエラーが発生することがあります

```
clang-16: error: cannot find libdevice for sm_50; provide path to different CUDA installation via
'--cuda-path', or pass '-nocudalib' to build without linking with libdevice
```

この問題を解決するには、--cuda-path オプションで CUDA* SDK のインストール・パスを指定します

トラブルシューティングと既知の問題 (5)

コンパイラーのエラー: “needs target feature (ターゲットの機能が必要です)”

DPC++ ランタイムで使用される一部の nvptx ビルトインは、コンパイルに最小限の計算機能を要求します
プログラムが使用するビルトインに対し十分な計算機能をターゲットにしていない場合 (コンパイル引数 -Xsycl-target-backend --cuda-gpu-arch=sm_xx を使用)、次のエラーが報告されます

```
error: '__builtin_name' needs target feature (sm_70|sm_72|..),...
```

- このようなエラーを回避するには、十分な計算機能を持つデバイスをターゲットにしてコンパイルしていることを確認してください
- -Xsycl-target-backend に 32 ビット・トリプル nvptx-NVIDIA-cuda を渡していないか確認してください。この場合 nvptx64-NVIDIA-cuda に変更します

トラブルシューティングと既知の問題 (6)

Linking Error with nvvm-reflect-ftz (nvvm-reflect-ftz とのリンクエラー)

現在、DPC++ のバグにより、CUDA* プラグインをターゲットにすると FTZ を有効にできません
また、FTZ は -ffast-math や -Ofast によって有効になる場合があります
この場合、次のようなエラーが発生します

```
error: linking module flags 'nvvm-reflect-ftz': IDs have conflicting override values
in '/tmp/lit-tmp-045kklh1/clang++-b050f4/libsycl-crt-a42c81.cubin' and 'llvm-link'
```

- FTZ を無効にして問題を解決する必要があります
  -ffast-math や -Ofast を使用する場合、-Xclang -fdenormal-fp-math=ieee を追加することで対応できます

- さらに、FTZ が -fcuda-flush-denormals-to-zero、-fdenormal-fp-math=preserve-sign または、-fdenormal-fp-math=positive-zero などのフラグで有効にされていないことを確認してください

oneAPI for NVIDIA*/AMD* GPU 今後の予定

- 2023年中は、最新の NVIDIA* と AMD* の GPU と SDK リリースを引き続きサポートすることで機能を追加し、テストおよびサポートするプラットフォームを拡大していきます

- oneAPI マス・カーネル・ライブラリー (oneMKL) や oneAPI ディープ・ニューラル・ネットワーク・ライブラリー (oneDNN) などの oneAPI 仕様コンポーネントのハードウェア・サポートを充実させていきます。これらのライブラリーについては、cuBLAS、cuDNN、hipBLAS および hipDNN のサポートをとおして正式リリースを準備し、適切にサポートされるようにし、「優先サポート」で重要な問題に対処できるようにします

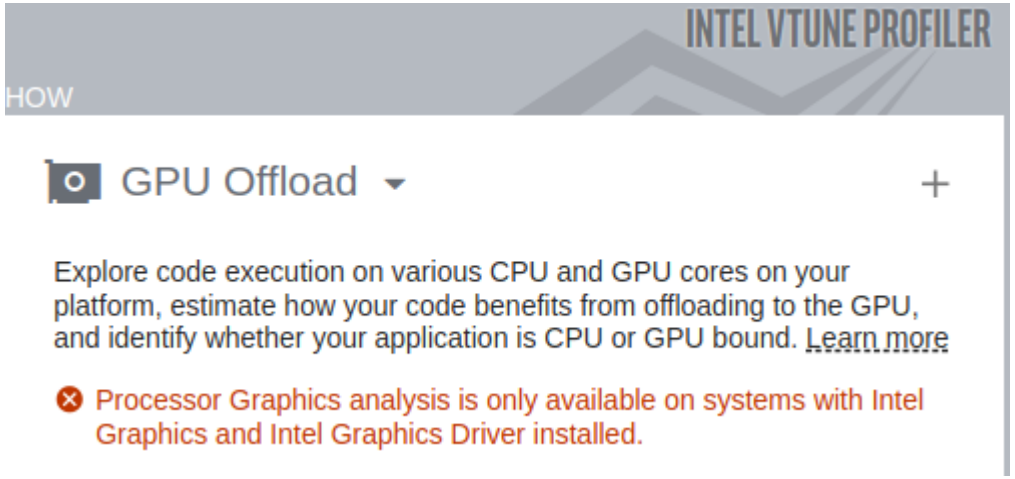
- 将来的には、oneAPI DPC++ ライブラリー (oneDPL) などの oneAPI ライブラリーで複数ベンダーのハードウェアをサポートし、NVIDIA* や AMD* ハードウェアを利用する開発者が SYCL* にシームレスに移行できるように支援したいと考えています

参考資料

- oneAPI for NVIDIA* GPU 2023.0.0 ガイド
- ベータ版 oneAPI for AMD* GPU 2023.0.0 ガイド
- oneAPI で NVIDIA と AMD のサポートを実現
- oneAPI DPC++ 導入ガイド
- oneAPI GPU 最適化ガイド日本語版
- oneAPI DPC++ コンパイラーとランタイム･アーキテクチャーの設計 (英語)

最後に一つ

インテル® VTune™ プロファイラーは、NVIDIA* GPU のパフォーマンス・データを収集して表示することができますが、GPU 関連のパフォーマンス・データ (OpenCL* インターフェイスを使用した) を収集するには、インテル® グラフィックス・ドライバーがシステムにインストールされている必要があります



Intel、インテル、Intel ロゴ、その他のインテルの名称やロゴは、Intel Corporation またはその子会社の商標です。
* その他の社名、製品名などは、一般に各社の商標または登録商標です。
© 2023 Intel Corporation. 無断での引用、転載を禁じます。