マルチプラットフォーム プログラミングの現状と可能性

oneAPI がもたらす未来

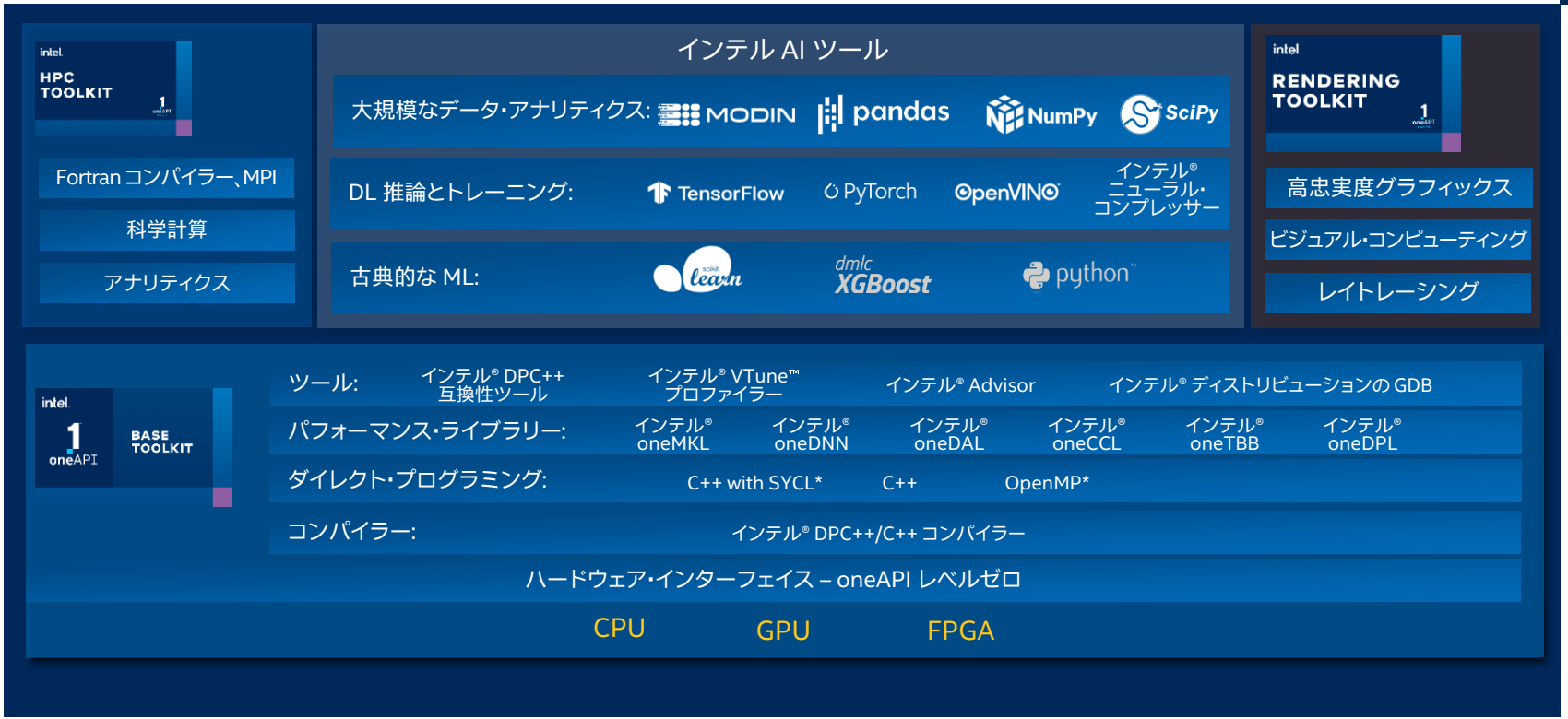
iSUS 編集部 すがわら

2025年9月11日

本日のメニュー

- oneAPI と SYCL*
- oneAPI を使用した GPU プログラミング
  - ✓ oneAPI for NVIDIA/AMD プラグイン
- TBB と oneTBB
- oneMath と oneMKL
- インテル® VTune™ プロファイラーに関するお知らせ
- 参考資料

インテル® ソフトウェア開発ツール powered by oneAPI



インテル® oneAPI ベース & HPC ツールキット

コンパイラーや実行環境

- インテル® Fortran コンパイラー
- インテル® oneAPI DPC++/C++ コンパイラー
- インテル® ディストリビューションの Python*

ツール/ライブラリー

- インテル® MPI ライブラリー
- インテル® oneAPI スレッディング・ビルディング・ブロック (インテル® oneTBB)
- インテル® oneAPI DPC++ ライブラリー (インテル® oneDPL)
- インテル® oneAPI コレクティブ・コミュニケーション・ライブラリー (インテル® oneCCL)
- インテル® oneAPI マス・カーネル・ライブラリー (インテル® oneMKL)
- 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 (インテル® oneDNN)
- インテル® oneAPI データ・アナリティクス・ライブラリー (インテル® oneDAL)
- インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP)
- インテル® DPC++ 互換性ツール
- インテル® oneAPI DPC++/C++ コンパイラー用 FPGA サポートパッケージ

解析/デバッグツール

- インテル® VTune™ プロファイラー
- インテル® Advisor
- インテル® ディストリビューションの GDB

■ インテル® **HPC** ツールキットにのみ含まれるもの

個別ダウンロードが必要なコンポーネントを含みます

GPU プログラミングの多様性

- CUDA*
  NVIDIA* が開発した GPU 向けの並列コンピューティング・プラットフォームとプログラミング・モデルです。GPU の並列処理能力を最大限に活用し、高速な計算処理を実現します。科学技術計算、機械学習、ビッグデータ解析など、大規模なデータ処理を必要とする分野で広く利用されています
- OpenCL*
  OpenCL* は、さまざまな種類のプロセッサーや計算リソースが混在する環境で並列処理を行うための規格です。CPU、GPU、DSP、FPGA など、多様なデバイスを連携させ、高速な計算を可能にするフレームワークを提供します
- ROCm*/HIP*
  ROCm* は、AMD の GPU を活用するためのオープンソースのソフトウェア・スタックです。主に AI や HPC ワークロード向けに最適化されています。低レベルのカーネルからエンドユーザー・アプリケーションまで、GPU プログラミングに必要なドライバー、開発ツール、API を提供します
- OpenGL*、C++ AMP*、DirectX*、Vulkan*、OpenACC*、OpenMP* …
- SYCL*

SYCL 10 周年

- SYCL は、開発者が多様なハードウェア・アーキテクチャーを活用できることを目標とした、オープンで標準ベースの抽象化レイヤーです。SYCL でアプリケーションを記述することで、複雑で時間のかかる移行作業や複数のソフトウェア・スタックを保守することなく、CPU、GPU、FPGA、あるいは革新的な新しい AI アクセラレーターなどのハードウェアで同時に実行できます
- SYCL は C++ で実装されており、使いやすさを前提に設計されています。Khronos Group が SYCL 仕様を管理しており、SYCL ワーキンググループを通じて業界に協力を呼びかけています
- SYCL は、プログラマーがヘテロジニアス・アプリケーションを開発しやすくするため、OpenCL 向けの高水準プログラミング・モデルを作成するという控えめな目標から始まりました
- SYCL の将来は非常に有望です。複数のオープンソース実装とさまざまな業界での採用拡大により、エコシステムは急速に拡大しています。ベンダーや Khronos が魅力的な拡張機能を開発中であり、強力な機能が導入されることで、幅広いヘテロジニアス・プラットフォームで動作する生産的で高性能なアプリケーションの開発がさらに容易になります

SYCL* の仕様 – 最新 SYCL* 2020 Rev 10

SYCL 2020 仕様 (英語) - Rev 1.0 は、2021年2月9日にリリースされ、大きな前進を示し、40 を超える新機能と改善点が含まれています。SYCL 2020 Rev 10 (2025年4月) が一般公開されました。変更点についてはリリースノート (英語) を参照してください

- 統合共有メモリー (USM) により、ポインターを含むコードがバッファーやアクセサーなしで自然に動作できるようになります
- 並列リダクションは、ビルトインのリダクション操作を追加し、定型的なコードを回避するのに役立ち、ビルトイン操作を備えたハードウェアに最大のパフォーマンスを提供します
- ワークグループとサブグループのアルゴリズムにより、ワーク項目間の効率的な操作が可能になります
- クラス・テンプレート引数推論 (CTAD) と推論ガイドにより、クラス・テンプレートのインスタンス化が簡単になります
- アクセサーの簡素化により、ビルトインのリダクション操作が追加され、定型コードの負担が軽減され、C++ パターンの簡素化が可能になります
- さまざまなバックエンドとの相互運用性が拡張され、OpenCL 以外のバックエンドのサポートが可能になりました
- アトミック操作を C++ のアトミック操作に近づけ、より自由な並列プログラミングが可能になりました

SYCL* の実装

SYCL 実装は、OpenCL に加えてさまざまなアクセラレーション API バックエンドのサポートを追加するなど、ますます多くのベンダーから提供されています:

- インテル® oneAPI - DPC++/C++ コンパイラーは、Khronos SYCL* 2020 仕様を完全にサポートします
- AdaptiveCpp (英語) - C++ ベースのヘテロジニアス・プログラミングのための独立したコミュニティー主導の最新プラットフォームです
- triSYCL (英語) - SYCL* 標準の仕様を実験する研究プロジェクト
- neoSYCL (英語) - SX-Aurora TSUBASA 用の SYCL* 実装
- SimSYCL (英語) - シミュレートされたハードウェアに対して SYCL* アプリケーションをテストするための、SYCL* 2020 のシングル・スレッド・ライブラリー実装

単一ソースのマルチプラットフォーム GPU プログラミングの提案

```
#include <sycl/sycl.hpp>
#include <iostream>
constexpr int num=16;
using namespace sycl;

int main() {
  auto R = range<1>{ num };
  buffer<int> A{ R };

  queue{}.submit([&](handler& h) {
    auto out =
      A.get_access<access::mode::write>(h);
    h.parallel_for(R, [=](id<1> idx) {
      out[idx] = idx[0];
    });
  });
  auto result = A.get_host_access();
  for (int i=0; i<num; ++i)
    std::cout << result[i] << "¥n";

  return 0;
}
```

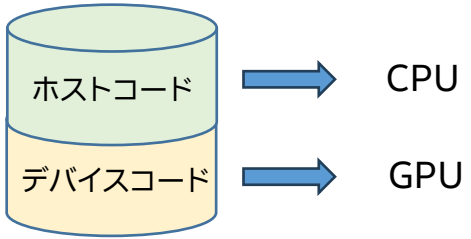
ホストコード

デバイスコード

ホストコード

```
$ icpx /fsycl sample.cpp -o sample
       /fsycl-targets=…
```

FAT バイナリー
.exe または a.out



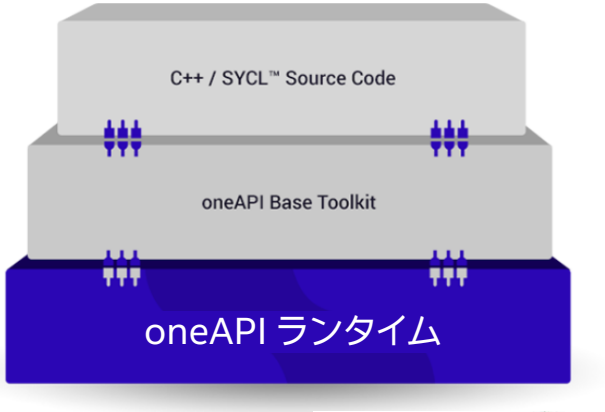
JIT コンパイル
spir64
nvptx64-nvidia-cuda
amdgcn-amd-amdhsa

oneAPI for NVIDIA*/AMD* GPU プラグイン

oneAPI とインテルの CPU/GPU

コンパイラーは中間コード (spir64) を生成

oneAPI ランタイムが spir64 をターゲットバイナリーにコンパイル



oneAPI とプラグイン

コンパイラーは中間コード (ptx64、amdgcn) を生成

NVIDIA*/AMD* バックエンドが ptx64、amdgcn をターゲットバイナリーにコンパイル

プラグインを入手 (英語)

https://github.com/intel/llvm/tree/sycl/sycl/plugins/unified_runtime/ur/adapters (英語)

インストールと環境設定

- 必須モジュール (Windows*/Ubuntu*)
  - ✓ インテル® oneAPI ベース・ツールキット
- インテル® ARC™ GPU を使用する場合 (Windows*/Ubuntu*)
  - ✓ インテル® ARC™ GPU ドライバー
- NVIDIA GPU を使用する場合 (Windows*/Ubuntu*)
  - ✓ NVIDIA GPU ドライバー
  - ✓ CUDA* ツールキット
  - ✓ Codeplay oneAPI for NVIDIA GPU プラグイン
- AMD GPU を使用する場合 (Ubuntu*)
  - ✓ AMD GPU ドライバー
  - ✓ ROCm ツールキット
  - ✓ Codeplay oneAPI for AMD GPU プラグイン

Windows 11 Pro 24H2
Ubuntu 24.04
oneAPI ベース・ツールキット 2025.1
oneAPI for NVIDIA/AMD プラグイン 2025.1
CUDA ツールキット 12.4 (Ubuntu)、13.0 (Win)
ROCm ツールキット 6.3

デバイスがシステムで認識されているか

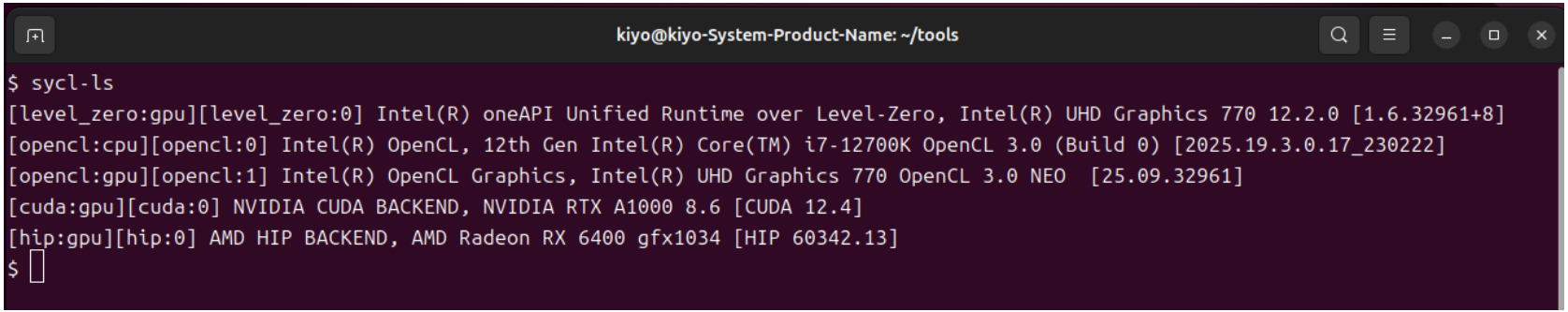
Linux* では、lspci コマンドで確認

```
$ lspci -k -d ::03xx
00:02.0 VGA compatible controller: Intel Corporation AlderLake-S GT1 (rev 0c)
        DeviceName: Onboard - Video
        Subsystem: ASUSTeK Computer Inc. AlderLake-S GT1
        Kernel driver in use: i915
        Kernel modules: i915, xe
01:00.0 VGA compatible controller: NVIDIA Corporation GA107GL [RTX A1000] (rev a1)
        Subsystem: NVIDIA Corporation Device 1878
        Kernel driver in use: nvidia
        Kernel modules: nvidiafb, nouveau, nvidia_drm, nvidia
08:00.0 VGA compatible controller: Advanced Micro Devices, Inc. [AMD/ATI] Navi 24 [Radeon RX 6400/6500 XT/6500M] (rev c7)
        Subsystem: Tul Corporation / PowerColor Navi 24 [Radeon RX 6400/6500 XT/6500M]
        Kernel driver in use: amdgpu
        Kernel modules: amdgpu
```

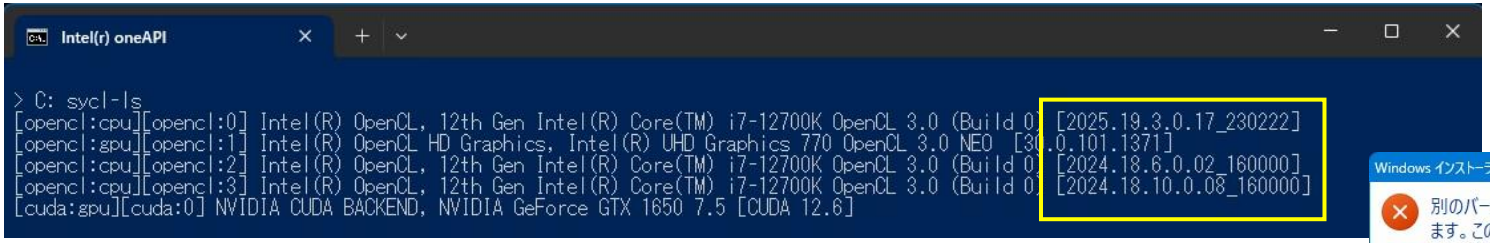
Windows* では、デバイス・マネージャーで確認

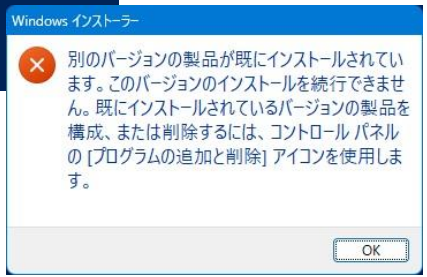
sycl-ls とランタイムのヒント

- インストールの成功を確認
- インストールされているランタイムバージョンの確認
- プログラムをオフロードするデバイスの選択

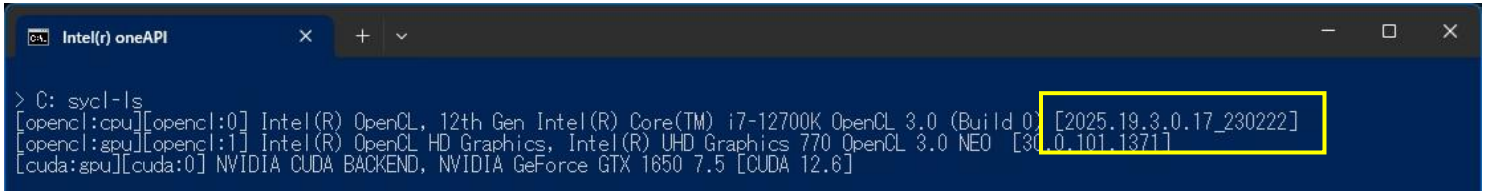


oneAPI ランタイムのバージョン管理



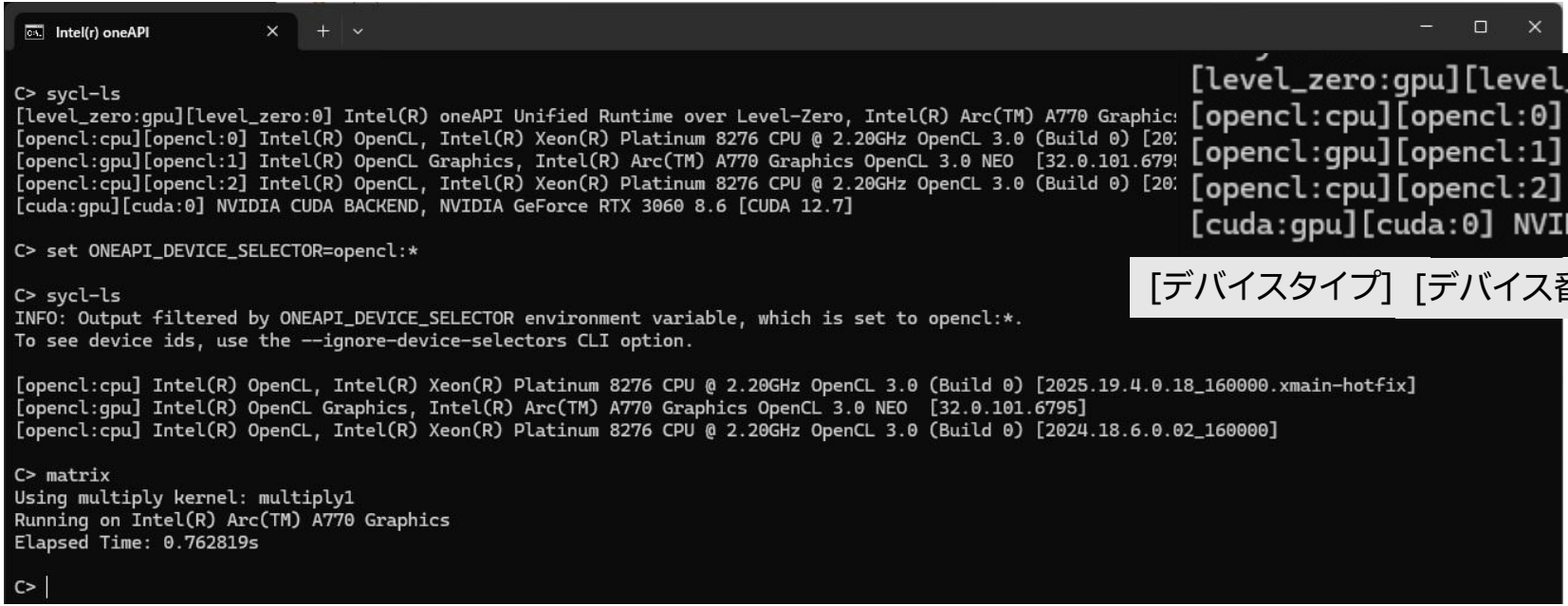


Intel® oneAPI Base Toolkit	Intel Corporation	2024/12/28	18.3 GB	2024.2
Intel® oneAPI Base Toolkit	Intel Corporation	2024/12/28	21.6 GB	2025
Intel® oneAPI Base Toolkit	Intel Corporation	2025/04/27	17.1 GB	2025.1
Intel® HPC Toolkit	Intel Corporation	2024/12/28	297 MB	2024.2
Intel® oneAPI HPC Toolkit	Intel Corporation	2024/12/28	219 MB	2025
Intel® oneAPI HPC Toolkit	Intel Corporation	2025/04/27	225 MB	2025.1



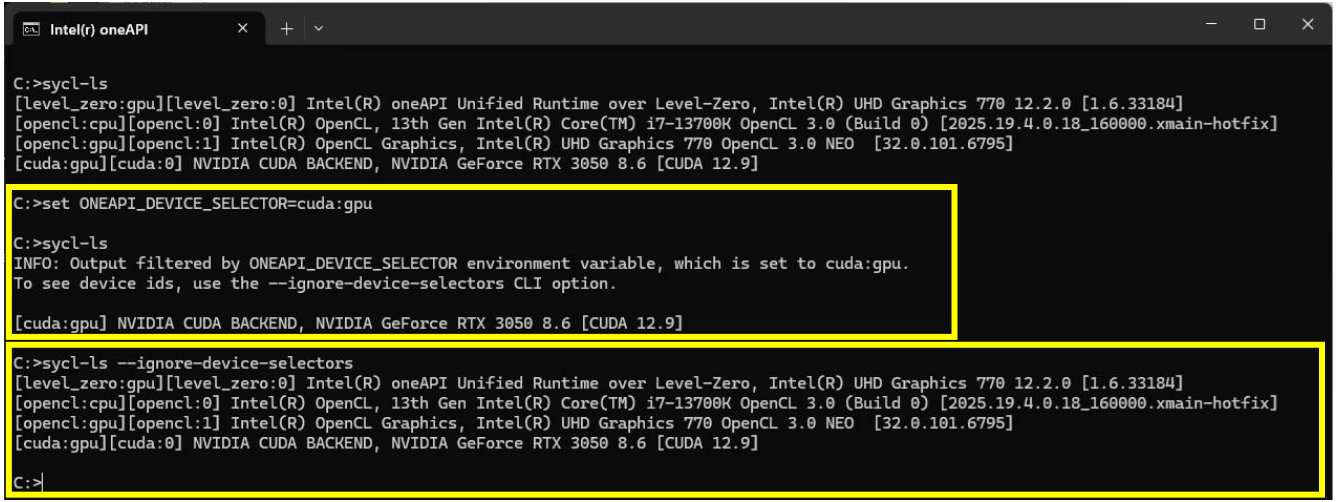
Intel® oneAPI Base Toolkit	Intel Corporation	2025/04/27	10.7 MB	2025.1
Intel® oneAPI HPC Toolkit	Intel Corporation	2025/04/27	225 MB	2025.1

オフロードするデバイスを選択



任意のデバイスをデバイスタイプで、または単独の番号で指定できます

sycl-ls と ONEAPI_DEVICE_SELECTOR



指定例:

“opencl:*”	すべての opencl デバイス。
“*:gpu”	すべての gpu デバイス。
“cuda:*”	すべての cuda デバイス。
“opencl:0”	デバイス番号 0 の opencl デバイス。

oneAPI for NVIDIA*/AMD* GPU プラグインを使用

- oneAPI for NVIDIA* GPU プラグインを使用するには CUDA* ツールキットが必要
- oneAPI for AMD* GPU プラグインを使用するには ROCm* ソフトウェア・スタックが必要

プラグインをインストールして GPU が利用できることを確認したら次のコマンドでソースファイルをコンパイル

```
$ icpx -fsycl -fsycl-targets=nvptx64-nvidia-cuda sycl-app.cpp -o sycl-app

$ icpx -fsycl -fsycl-targets=amdgcn-amd-amdhsa -Xsycl-target-
backend=amdgcn-amd-amdhsa --offload-arch=gfx1030 sycl-app.cpp -o sycl-app
```

https://www.isus.jp/products/oneapi/oneapi-for-nvidia-gpu-get-started/

https://www.isus.jp/products/oneapi/oneapi-for-amd-gpu-get-started/

複数のターゲットデバイスで実行可能な SYCL* アプリケーションを作成

`-fsycl-targets` に複数のターゲットを指定することで、単一バイナリーで複数のターゲットデバイスで実行可能なファイルを作成できます

```
$ icpx -fsycl -fsycl-targets=nvptx64-nvidia-cuda,spir64 sycl-app.cpp -o sycl-app
```

`-fsycl-targets=nvptx64-nvidia-cuda,spir64` オプションでは、NVIDIA* GPU と SPIR64 (インテル® GPU など) で実行可能な SYCL* アプリケーションを生成できます

AMD* GPU もターゲットにする場合、`-fsycl-targets=nvptx64-nvidia-cuda,spir64,amdgcn-amd-amdhsa` と `-Xsycl-target-backend=amdgcn-amd-amdhsa -offload-arch=gfx1030` を追加します

デバイスのアーキテクチャーを確認

```
$ rocminfo
ROCk module version 6.10.5 is loaded
=====================
HSA System Attributes
=====================
Runtime Version:         1.14
Runtime Ext Version:     1.6
System Timestamp Freq.:
Sig. Max Wait Duration:
Machine Model:
System Endianness:
Mwaitx:
DMAbuf Support:

  ISA Info:
*******
Agent 2
*******
  Name:                    gfx1034
  Uuid:                    GPU-XX
  Marketing Name:          AMD Radeon RX 6400
  Vendor Name:             AMD
  Feature:                 KERNEL_DISPATCH
```

NVIDIA*/AMD* GPU 向けの oneAPI アプリの最適化

- NVIDIA* GPU と AMD* GPU の両方に適用される一般的な SYCL* 最適化については、oneAPI for NVIDIA* GPU ガイドの一般的な最適化を参照してください
- インテル® GPU 固有のパフォーマンス最適化については、GPU 最適化ガイドのパフォーマンス・ガイドを参照してください
- GPU コードのパフォーマンスに影響する主な要因を重要度の高い順に考慮します (グローバル・メモリー・アクセス、ローカルメモリーのバンク競合、条件文による命令の発散)
- カーネルのパフォーマンスを評価するには**占有率**を考慮します

GPU アクセラレーター向けのチューニングについてはこちらの PDF を参照

TBB と oneTBB

- TBB は、C++ で並列プログラムを作成する強力なソリューションであり、この言語の並列プログラミングに対する最もポピュラーで包括的なサポートを提供してきました
- 2019年、TBB は、進化した C++ 標準に組み込まれた機能のサポートを廃止することで、提供内容を合理化する大胆な決断を下しました
- oneTBB にブランド名が変更されても、競合ではなく最新の C++ 基盤に依存していることを除けば、元の TBB と基本的な違いはありません

インターフェイス名と名前空間は `tbb` のままで、`oneapi::tbb` が `tbb` のエイリアスとして定義されているため、一般には引き続き TBB と呼ばれます

TBB から oneTBB への改善点

- 最新の C++ との整合性
  - **例: `tbb::atomic` の削除 (`tbb atomic` から C++ `atomic` へ)**

  名前空間 `tbb` を `std` に変更し、`<atomic>` ヘッダーをインクルード

  `compare_and_swap` -> `compare_exchange_weak` / `compare_exchange_strong`
- インターフェイスの改善
  - **冗長または問題のあるインターフェイスの削除**

  `task_scheduler_init` で利用されるアクセス・コントロール

  `parallel_do` の削除

  `pipeline` クラスの削除

  最下位レベルのタスク/スケジューラー API の削除

タスクアリーナの NUMA やコアタイプへの制限

- `task_arena` クラスは、スレッドがタスクを共有/実行可能な場所を示します
- `task_arena` のインスタンスは、新しい TBB アリーナを作成したり、既存のアリーナへの軽量ハンドルとして機能します。アリーナは、異なる数のスロット、マスタースレッド用に予約された異なる数のスロットで作成できます。
  また、スレッドを NUMA ノードまたは E-core や P-core などの特定のプロセッサー・タイプに固定して作成することもできます

```
void constrain_for_core_type() {
    std::vector<tbb::core_type_id> core_types = tbb::info::core_types();
    tbb::task_arena arena(
        tbb::task_arena::constraints{}.set_core_type(core_types.back())
    );

    arena.execute([] {
        tbb::parallel_for(0, N, [](int) { f(w); });
    });
}
```

oneMKL と oneMath

oneMath は、oneMath 仕様のオープンソース実装であり、複数のライブラリー (バックエンド) を使用することで、複数のデバイスで動作します。oneMath プロジェクトは、以前は oneMKL インターフェイスと呼ばれていました

ユーザー・インターフェイス	oneMath レイヤー	サードパーティー・ライブラリー	ハードウェア・バックエンド
oneMath `oneapi::math::blas`	`oneapi::mkl::blas` oneMath セレクター	インテル® MKL	x86 CPU、iGPU
		NVIDIA cuBLAS …	NVIDIA GPU
		NETLIB LAPACK	x86、aarch64 CPU
		Arm パフォーマンス・ライブラリー	aarch64 CPU
		Arm OpenRNG	x86、aarch64 CPU
		AMD rocBLAS …	AMD GPU

出典: https://github.com/uxlfoundation/oneMath (英語)

インテル® VTune™ プロファイラー 日本語パッケージの紹介

- 日本語版パッケージは iSUS にて独自に日本語化したものです
  インテル社が提供する日本語版ではないため予めご了承ください
- それぞれ各バージョンのパッケージを提供しています
  - ✓ インテル® VTune™ プロファイラー 2025.4.0 (Windows* 版)
  - ✓ インテル® VTune™ プロファイラー 2025.3.0 (Windows* 版)
  - ✓ インテル® VTune™ プロファイラー 2025.0.1 (Windows* 版)
- 導入用の bat ファイルを実行します
  もしくは手動でインストール･フォルダーにコピーします
  1. コマンドプロンプトを管理者権限で開きます
  2. profiler2025_install.bat を実行します

https://www.isus.jp/products/vtune/vtune_profiler_jp/

注: インテル® VTune™ プロファイラー 2025.4 のパッケージから、レガシー･プロセッサーがサポートされなくなりました。詳細はこちらで

参考資料

- oneAPI GPU
  - ✓ oneAPI GPU 最適化ガイド 2025.2
- oneAPI for NVIDIA/AMD プラグイン
  - ✓ oneAPI for NVIDIA* GPU ガイド 2025.2
  - ✓ oneAPI for AMD* GPU ガイド 2025.2
- oneTBB
  - ✓ oneTBB 導入ガイド (2025.2)
  - ✓ oneTBB デベロッパー・ガイドおよび API リファレンス (2025.2)
  - ✓ クックブック
- oneMKL
  - ✓ oneMKL 導入ガイド (2025.0)
  - ✓ oneMKL データ並列 C++ デベロッパー・リファレンス (2025.0)
  - ✓ インテル® oneAPI マス・カーネル・ライブラリー・デベロッパー・ガイド (2025.0)
  - ✓ クックブック

https://www.isus.jp/isus-translated-documents/

新しい書籍

TBB は長年開発が続けられてきたため、TBB に関する以前の 2 冊の書籍を含め、大量の古い資料がオンライン上に存在します。古いリソースの大部分は正確ですが、微妙な違いが勘違いの原因となる場合があります

特に、進化する C++ 標準に準拠するため変更が反映されている場合、その傾向が強くなります

2006年に TBB がデビューした頃は C++03 が標準でしたが、この言語は大幅に進化しており、アップデートされた資料を参照することが重要になっています

SYCL SC

- 2023年3月、Khronos Group は、広く採用されている SYCL 2020 規格を活用し、セーフティー・クリティカル・システム向けの高水準コンピューティング規格を策定する、SYCL SC ワーキンググループの設立を発表しました
- SYCL SC は、Vulkan* SC などの低水準 API と C++ 高水準言語間のギャップを埋め、必要に応じて Vulkan SC の設計原則を活用し、ソフトウェア・スタックのあらゆるレベルでシステム安全性認証を効率化できるオープンスタンダードの実現を目指します

https://www.khronos.org/syclsc (英語)

ありがとうございました