SYCL* ベースの oneAPI プログラミング

2021 年 5 月

IA Software User Society (iSUS)
編集長　すがわら　きよふみ

内容

- oneAPI 概要
- SYCL* の概要
- コンパイラーの仕組み
- oneAPI ライブラリーの例
- 関連ドキュメント

oneAPI 業界イニシアチブ

独自のアーキテクチャーに代わる選択肢

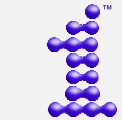
C++ および SYCL* 標準ベースのクロスアーキテクチャー言語

ドメイン固有の関数を高速化するように設計された強力なライブラリー

低水準ハードウェア抽象化レイヤー

コミュニティーと業界の協力を促進するオープンな環境

複数のアーキテクチャーとベンダーでコードを再利用可能



経済的および技術的な負担を伴う独自のプログラミング・モデルに代わって、高速なコンピューティングを実現する生産性に優れたスマートなパス

詳細は、oneapi.com (英語) を参照してください。

データ並列 C++ (DPC++)

標準ベースのクロスアーキテクチャー言語
DPC++ = ISO C++ と Khronos SYCL*

CPU とアクセラレーターに並列処理、生産性、パフォーマンスを提供

- ハードウェアの機能を活用して高速なコンピューティングを実現
- ターゲット・ハードウェア間でコードの再利用が可能、特定のアクセラレーター向けのカスタム・チューニングを行うことが可能
- 単一アーキテクチャー専用のソリューションに代わる、オープンな業界共通のソリューションを提供

C++ および SYCL* ベース

- C++ の生産性の利点を提供、一般的で使い慣れた C および C++ 構造を使用
- The 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

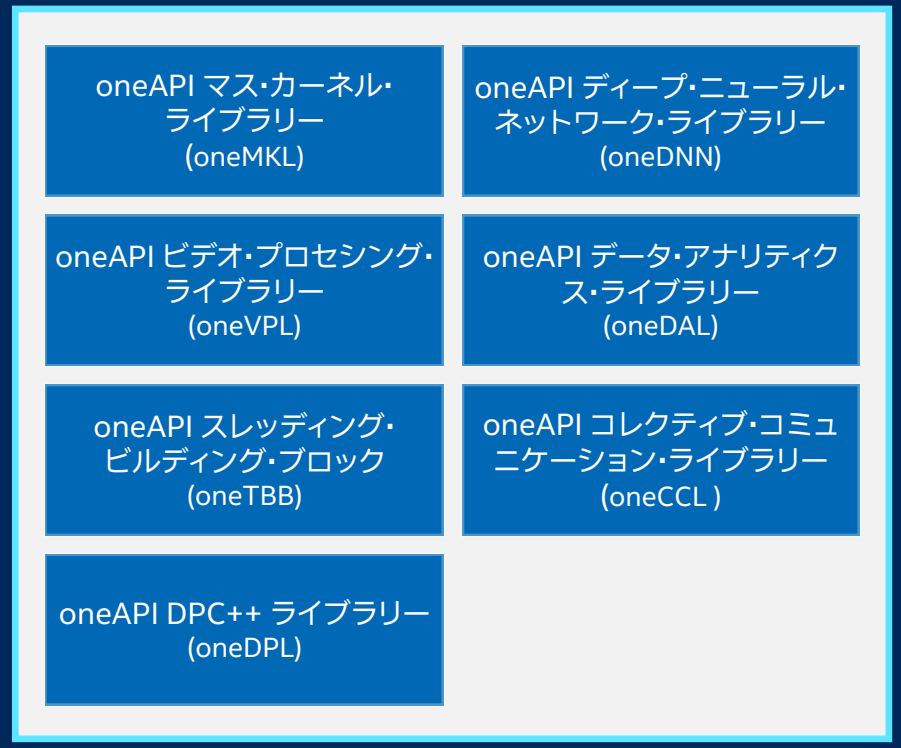
- データ並列プログラミングを簡素化する拡張機能を提供
- オープンな共同開発により継続的に進化

次のハードウェア・プラットフォーム向けのソフトウェア
変更ではなく、次のイノベーションにスキルを活用できる

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。

強力な oneAPI ライブラリー

- 主要なドメイン固有の関数を高速化するように設計
- 最高のパフォーマンスが得られるように各プラットフォーム向けに事前に最適化



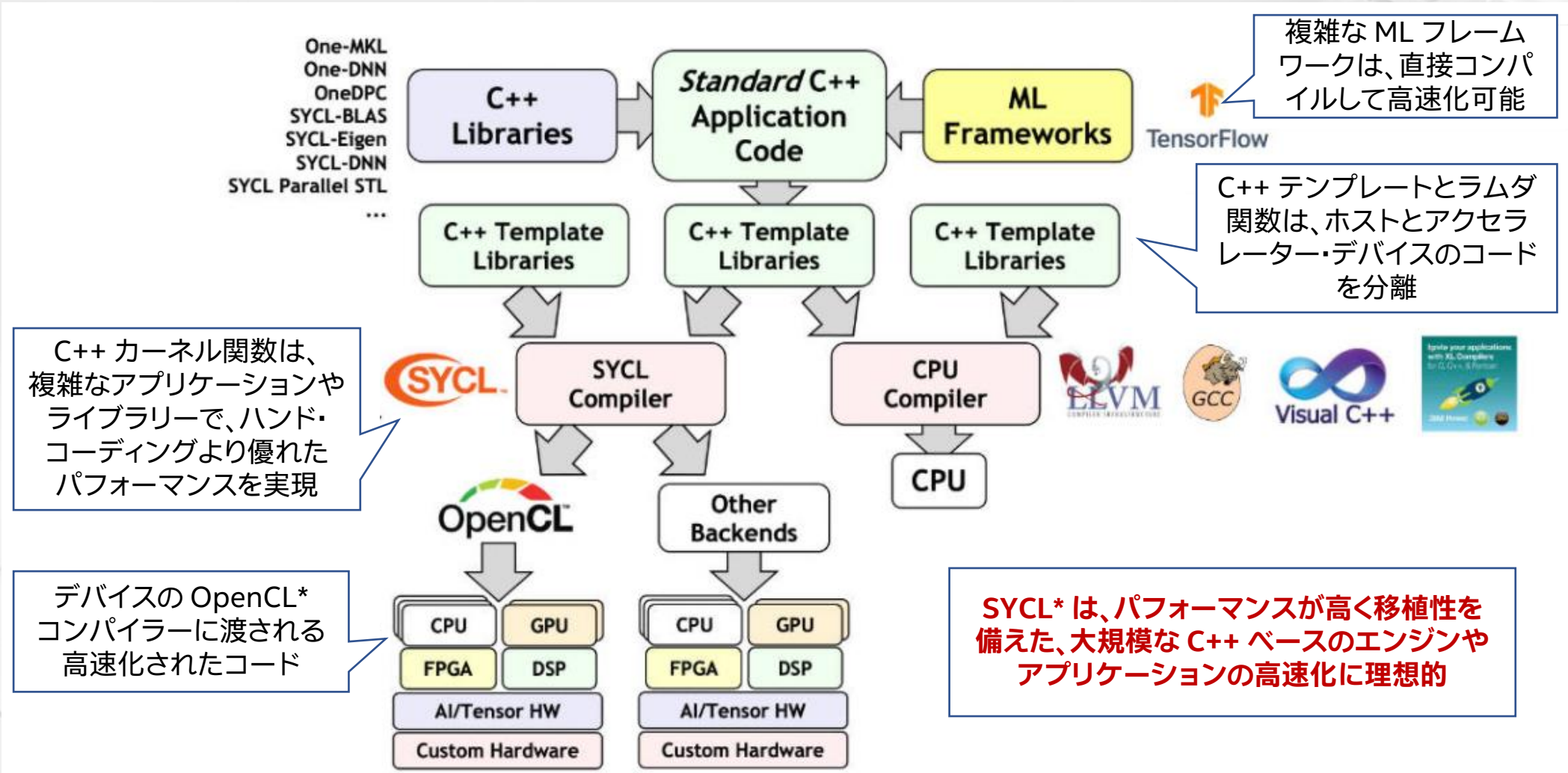
SYCL* 2020

SYCL* 2020 は、2020 年後半にワーキンググループによって承認され、2021 年初頭に The Khronos Group によって公開された SYCL* 仕様の最新リリースです

OpenCL* に基づく最後のバージョンである SYCL* 1.2.1 に続くものです。より一般化されたバックエンド・モデルへの移行と、進化する ISO C++ のリリースに合わせて、プロジェクトは年ベースで更新されます

- 統合共有メモリー (USM)、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機能できるようにします
- 並列リダクション、ビルトイン・リダクション操作の追加、定形コードセクションの回避を支援、ビルトイン操作を備えたハードウェアに最大限のパフォーマンスを提供
- ワークグループとサブグループのアルゴリズムは、ワーク項目間の効率的な操作を可能にします
- クラス・テンプレートのインスタンス化を可能にするクラス・テンプレート引数控除 (CTAD) および控除ガイド・ビルトイン・リダクション操作を追加するアクセサーの簡素化により、定形コードセクションの負担が軽減され、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
- 並列プログラミングをより自由にする C++ アトミックに類似したアトミック操作

SYCL* の概要



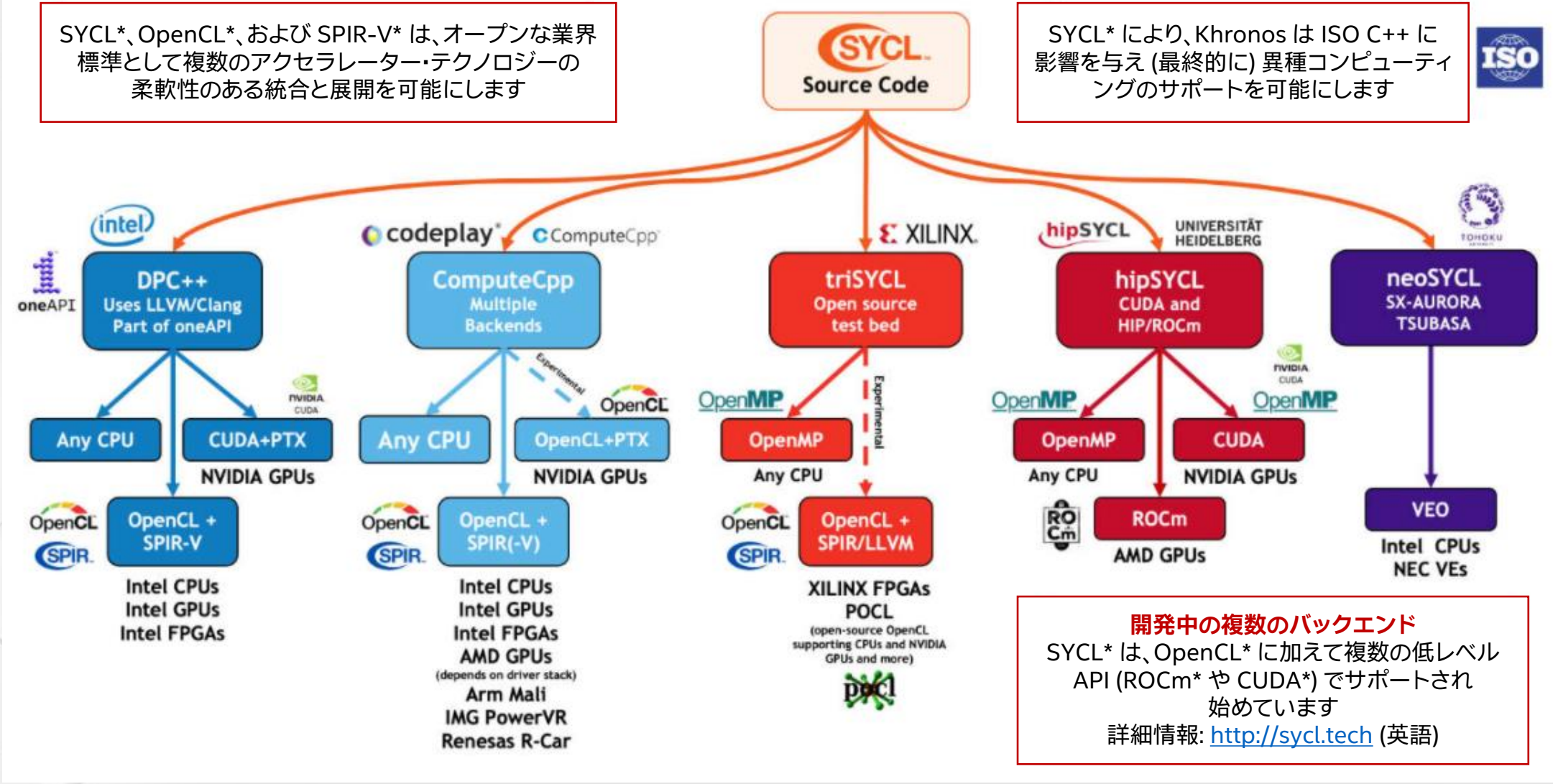
SYCL* 単一ソース C++ 並列プログラミング

出典: https://www.khronos.org/sycl/

SYCL* の実装

SYCL*、OpenCL*、および SPIR-V* は、オープンな業界標準として複数のアクセラレーター・テクノロジーの柔軟性のある統合と展開を可能にします

SYCL* により、Khronos は ISO C++ に影響を与え (最終的に) 異種コンピューティングのサポートを可能にします



開発中の複数のバックエンド
SYCL* は、OpenCL* に加えて複数の低レベル API (ROCm* や CUDA*) でサポートされ始めています
詳細情報: http://sycl.tech (英語)

出典: https://www.khronos.org/sycl/

コンパイラーの仕組み

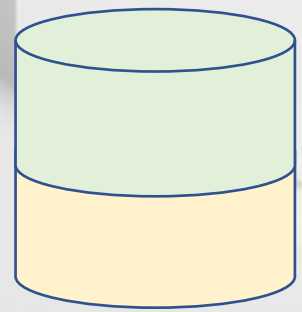
```
#include <CL/sycl.hpp>
#include <iostream>
constexpr int num=16;
using namespace cl::sycl;

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_access<access::mode::read>();
  for (int i=0; i<num; ++i)
    std::cout << result[i] << "¥n";

  return 0;
}
```

```
$ dpcpp sample.cpp -o sample
```

ホストコード

デバイスコード

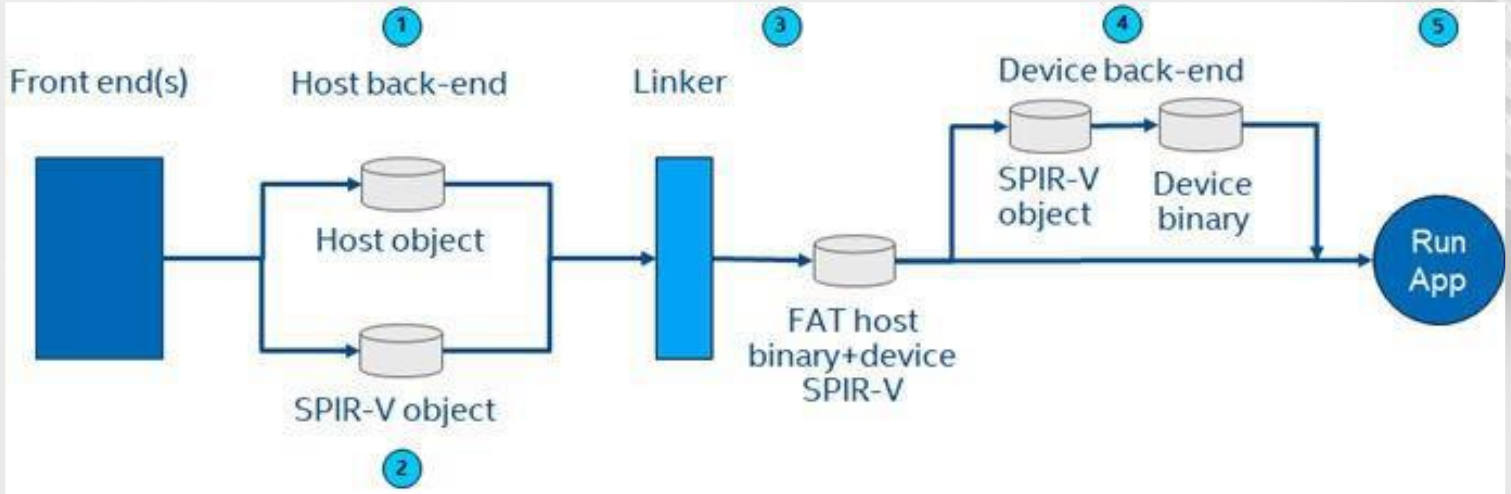
ホストコード



FAT バイナリー
.exe または a.out

JIT コンパイル
AOT コンパイル

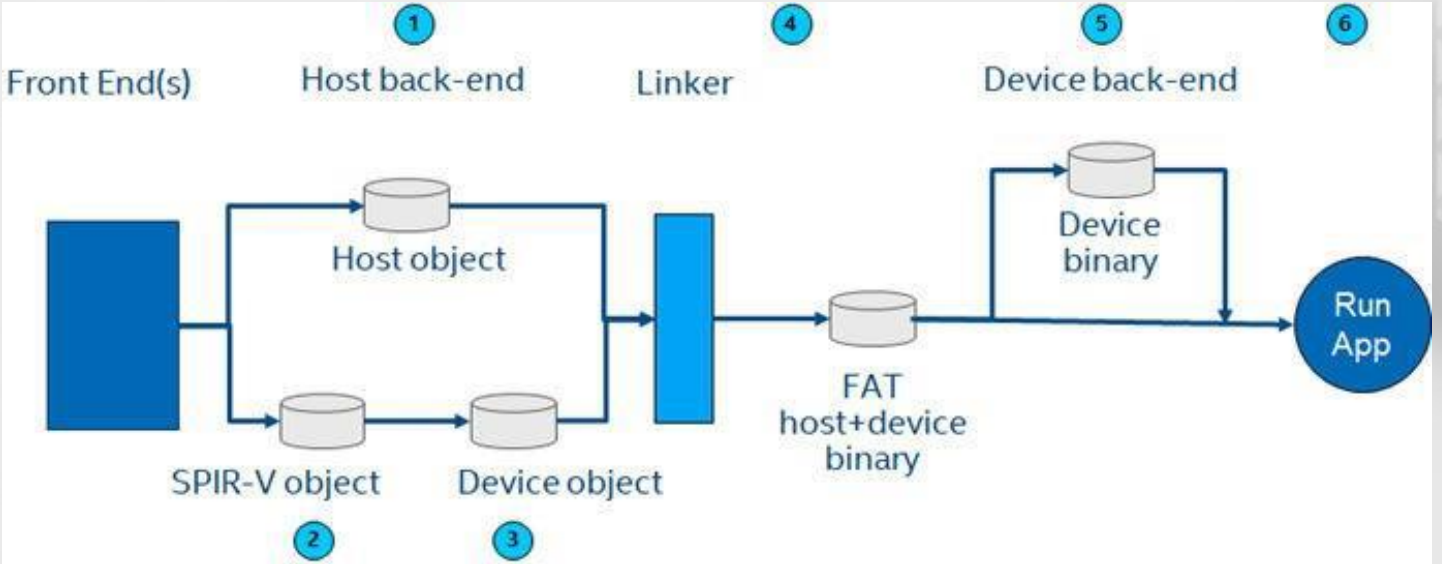
ジャストインタイム (JIT) コンパイル



1. ホストコードは、バックエンドでオブジェクト・コードに変換される
2. デバイスコードは SPIR-V* 形式に変換される
3. リンカーは、ホスト・オブジェクト・コードとデバイス SPIR-V* を組み合わせた、(SPIR-V* が埋め込まれた) ホスト実行コードを含むファットバイナリーを生成

4. 実行されると次のように処理される
   a. ホスト上のデバイスランタイムは、デバイスの SPIR-V* をデバイスのバイナリーに変換
   b. 変換されたデバイスバイナリーはデバイスへロードされる

事前 (AOT) コンパイル



1. ホストコードは、バックエンドでオブジェクト・コードに変換される
2. デバイスコードは SPIR-V* 形式 (SPIR-V.obj) に変換される
3. デバイスの SPIR-V* は、ユーザーが指定したデバイス向けのデバイス・コード・オブジェクトに変換される
4. リンカーは、ホスト・オブジェクト・コードとデバイス・オブジェクト・コードを組み合わせた、デバイスバイナリーが埋め込まれたホスト実行コードを含むファットバイナリーを生成
5. 実行時に、デバイスバイナリーはデバイスへロードされる

oneMKL の例: 行列乗算

バッファー API

```
using mkl::blas::gemm;
int64_t n = 32;

device dev({host, cpu, gpu}_selector());
queue Q(dev);

double *A = ..., *B = ..., *C = ...;

buffer<double, 1> A_buf{A, range<1>(n * n)},
                  B_buf{B, range<1>(n * n)},
                  C_buf{C, range<1>(n * n)};


gemm(Q, mkl::transpose::N, mkl::transpose::N,
        n, n, n, 1.0, A_buf, n, B_buf, n,
        0.0, C_buf, n);
```

デバイス・セットアップ

行列の準備

C = A * B

USM API

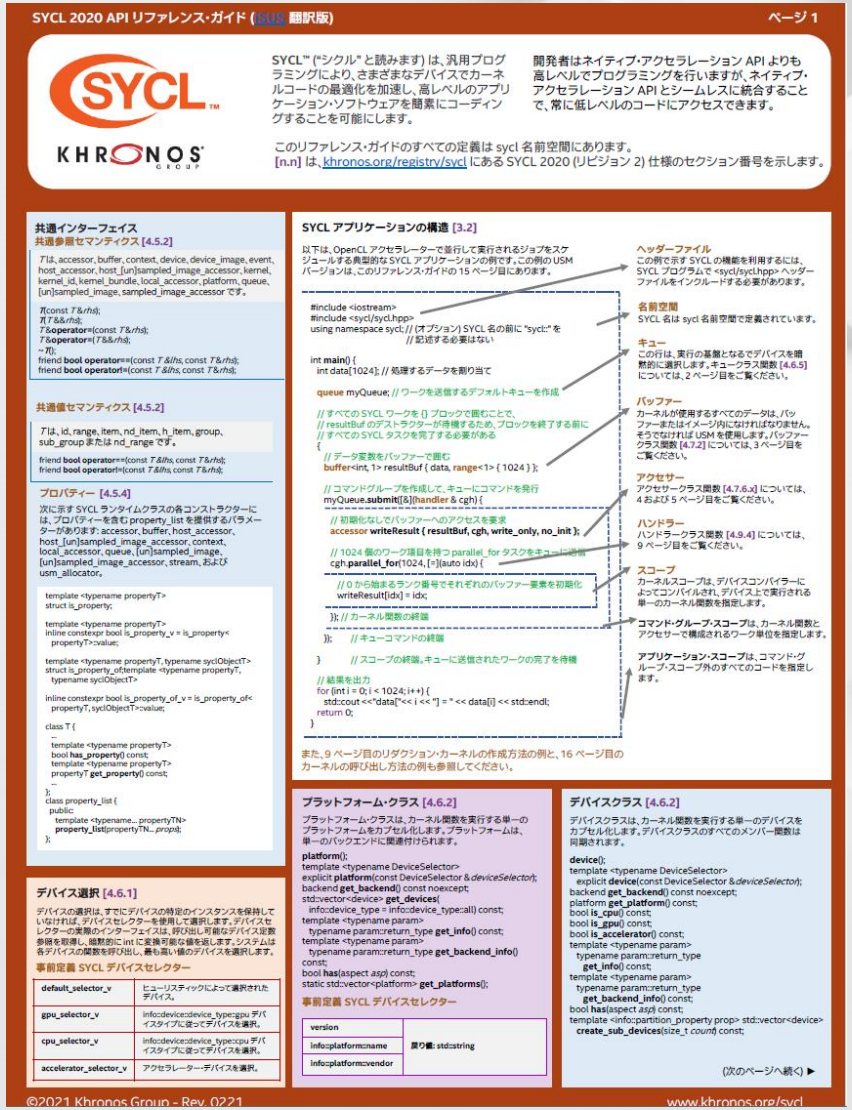
```
using mkl::blas::gemm;
int64_t n = 32;

device dev({host, cpu, gpu}_selector());
queue Q(dev);

size_t bytes = n * n * sizeof(double);
double *A = malloc_shared(bytes, Q,
                          Q.get_context());
double *B = malloc_shared(…);
double *C = malloc_shared(…);


gemm(Q, mkl::transpose::N, mkl::transpose::N,
     n, n, n, 1.0, A, n, B, n,
     0.0, C, n);

Q.wait_and_throw();
```

oneAPI の利点: まとめ

- 1 つのソースコードから CPU、GPU、FPGA、アクセラレーターに対応するバイナリーを生成することが可能
- 既存のインテル® ライブラリーを使用するアプリケーションは、容易に oneAPI 環境への移行が可能
- インテル® Advisor、インテル® VTune™ プロファイラー、インテル® Inspector など使い慣れたインテル® ソフトウェア開発製品を利用可能

ドキュメント

- SYCL* 2020 API リファレンス・ガイド
  https://www.isus.jp/isus-translated-documents/



ドキュメント

- SYCL* 2020 API リファレンス･ガイド
https://www.isus.jp/isus-translated-documents/

- SYCL* 2020 仕様 (リビジョン 3) 翻訳中
https://www.khronos.org/registry/SYCL/ (英語)

ドキュメント

- SYCL* 2020 API リファレンス･ガイド
https://www.isus.jp/isus-translated-documents/

- SYCL* 2020 仕様 (リビジョン 3) 翻訳中
https://www.khronos.org/registry/SYCL/ (英語)

- oneAPI 仕様リリース 1.0 Rev3 Part1
https://www.isus.jp/products/oneapi/oneapi-spec-japanese/

oneAPI™

oneAPI 仕様

リリース 1.0 リビジョン 3 日本語参考訳 Part 1 (1 – 311 ページ)

インテル

2020 年 11 月 23 日

注意事項:

1. この日本語マニュアルは、インテル コーポレーションのウェブサイト: https://spec.oneapi.com/versions/latest/versions.html (英語) で公開されている『oneAPI Specification Release 1.0-rev-3』の参考訳です。インテル社の許可を得て iSUS (IA Software User Society) が翻訳版を作成した iSUS の著作物です。原文はIntel Corporation の Copyright であり、日本語参考訳版にも適用されます。
2. 英語版と日本語版のページ番号は必ずしも同一ではありません。

ドキュメント

- SYCL* 2020 API リファレンス･ガイド
https://www.isus.jp/isus-translated-documents/

- SYCL* 2020 仕様 (リビジョン 3) 翻訳中
https://www.khronos.org/registry/SYCL/ (英語)

- oneAPI 仕様リリース 1.0 Rev3 Part1
https://www.isus.jp/products/oneapi/oneapi-spec-japanese/

- インテル® oneAPI プログラミング･ガイド
https://www.isus.jp/products/oneapi/oneapi-programming-guide-released/

ドキュメント

- SYCL* 2020 API リファレンス・ガイド
https://www.isus.jp/isus-translated-documents/

- SYCL* 2020 仕様 (リビジョン 3) 翻訳中
https://www.khronos.org/registry/SYCL/ (英語)

- oneAPI 仕様リリース 1.0 Rev3 Part1
https://www.isus.jp/products/oneapi/oneapi-spec-japanese/

- インテル® oneAPI プログラミング・ガイド
https://www.isus.jp/products/oneapi/oneapi-programming-guide-released/

- インテル® oneAPI ポーティング・ガイド
https://www.isus.jp/products/c-compilers/oneapi-porting-guide-japanese/

intel

インテル® oneAPI ポーティング・ガイド

DPCPP または ICX へ移行する ICC ユーザー向け

IntelCorporation

www.intel.com(英語)

著作権と商標について

注意事項:

この日本語マニュアルは、インテルコーポレーションのウェブサイト https://software.intel.com/content/www/us/en/develop/articles/porting-guide-for-icc-users-to-dpcpp-or-icx.html (英語)で公開されている『Porting Guide for ICC Users to DPCPP or ICX』(更新日:2020/12/09)の参考訳です。

インテル社の許可を得て iSUS(IA Software User Society)が翻訳版を作成した iSUS の著作物です。

原文は Intel Corporation の Copyright であり、日本語参考訳版にも適用されます。

ドキュメント

- SYCL* 2020 API リファレンス･ガイド
https://www.isus.jp/isus-translated-documents/
- SYCL* 2020 仕様 (リビジョン3) 翻訳中
https://www.khronos.org/registry/SYCL/ (英語)
- oneAPI 仕様リリース 1.0 Rev3 Part1
https://www.isus.jp/products/oneapi/oneapi-spec-japanese/
- インテル® oneAPI プログラミング･ガイド
https://www.isus.jp/products/oneapi/oneapi-programming-guide-released/
- インテル® oneAPI ポーティング･ガイド
https://www.isus.jp/products/c-compilers/oneapi-porting-guide-japanese/
- oneAPI GPU 最適化ガイド
https://www.isus.jp/products/oneapi/oneapi-gpu-optimization-guide-released/

ハイパフォーマンス・ソフトウェア・カンファレンス・オンライン 2021【春】

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。
製品および性能に関する情報: 性能は、使用状況、構成、その他の要因によって異なります。詳細については、http://www.intel.com/PerformanceIndex/ (英語) を参照してください。注意事項の改訂 #20201201
© 2021 Intel Corporation. 無断での引用、転載を禁じます。