パート 6: oneAPI ライブラリーを VSCode の C++ プロジェクトにリンクする

この記事は 2023 年 8 月 16 日に Codeplay のウェブサイトで公開された「Linking oneAPI libraries to a Visual Studio Code C++ project」の日本語参考訳です。原文は更新される可能性があります。原文と翻訳文の内容が異なる場合は原文を優先してください。

本シリーズのこれまでの記事では、Ubuntu* 20.04 プラットフォーム上で Visual Studio* Code (VSCode) 開発環境をゼロからセットアップする方法を紹介しました。

本シリーズの記事:

- パート 1: DPC++ と Visual Studio* Code で SYCL* コードをデバッグ
- パート 2: Ubuntu* で C++ 開発向けに Visual Studio* Code を設定
- パート 3: Ubuntu* で SYCL* 開発向けに oneAPI、DPC++、Visual Studio* Code を設定
- パート 4: Ubuntu* で Visual Studio* Code を使用して DPC++ をデバッグ
- パート 5: DPC++ と Visual Studio* Code を使用した C++ プロジェクトの SYCL* への移行

はじめに

オープンソースのデータ並列 C++ (DPC++) SYCL* コンパイラーと oneAPI エコシステムのサポートにより、ベンダー・ニュートラルなアクセラレーション・ハードウェア向けのアプリケーションの開発とチューニングが容易になりました。エコシステムは、チューニング・ツールだけでなく、ハイパフォーマンスでデータセントリックなアプリケーションの開発を支援するランタイム・ライブラリーのコアセットも提供します。

このガイドでは、パート 5 に引き続き、VSCode IDE を使用して、oneAPI マス・カーネル・ライブラリー (oneMKL) を簡単な DPC++ プログラムにリンクする方法を紹介します。Microsoft* C/C++ 拡張機能パックを使用して、この DPC++ プログラムのビルドおよびデバッグセッションを設定します。

必要条件

VSCode 開発環境が DPC++ 開発用に設定済みであり、既存の DPC++ プロジェクトをコンパイルできる必要があります。そうでない場合は、パート 2 および 3 の手順に従って設定してください。また、このガイドで言及されているツールについて詳しく説明されている、パート 5 をお読みになることを推奨します。

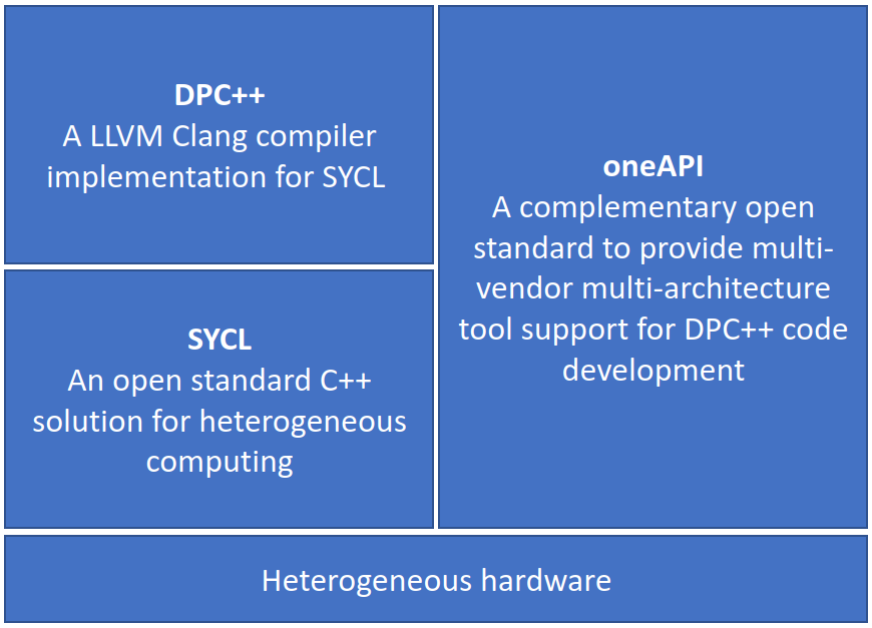

図 1: oneAPI エコシステム

ここでは、インテルの oneAPI サンプルプログラム **student_t_test** と oneMKL ライブラリーを使用し、サンプルプログラムを VSCode の Microsoft* C/C++ 拡張プロジェクトに変換します。サンプルプログラムはさまざまな場所から入手できますが、ここではインテルの VSCode 拡張である **Code Sample Browser for Intel® oneAPI Toolkits** を使用して取得します。

> **注:**
> インテルの **student_t_test** サンプル・プロジェクトは、インテル® oneAPI ベース・ツールキットの **oneAPI-samples/Libraries/oneMKL/student_t_test** ディレクトリーにあります。

このプロジェクトを Microsoft* C/C++ 拡張のビルドおよびデバッグ構成を使用するように移行すると、機能豊富でインタラクティブな GUI を利用できます (変換手順は、パート 5 の「プロジェクト移行アプローチ」を参照)。インテルのサンプル・プロジェクトはすべて、CMake、Makefile、GNUmakefile ビルドシステムのいずれを使用していても VSCode IDE のフレームワーク内で使用できます。ただし、これらのビルドシステムでは、VSCode IDE をコードエディターとして使用することのみが可能であり、組み込みターミナルを使用して make コマンドを実行できます。シェルベースの環境に慣れた開発者にも、Microsoft* Visual Studio* などの GUI IDE に慣れた開発者にも、Microsoft* C/C++ 拡張機能とサードパーティー拡張機能は、同様の体験を提供します。

oneAPI ライブラリーのリンク

ほかのビルド構成ツールと同様に、VSCode の Microsoft* C/C++ 拡張機能は、プログラムのコンパイルの一部としてインクルードされる外部スタティック・ライブラリーとヘッダーファイルのパスを保持する必要があります。

以下の DPC++ コンパイラー・オプションで、特定のスタティック・ライブラリーの場所を指定して使用できます。

コンパイラー・オプション

- **-l**: ライブラリー・ファイルとリンクします。
- **-L**: ライブラリー・ファイルを検索するディレクトリーを指定します。

注:
VSCode では、コードテキスト内の変数を Ctrl キーを押しながらクリックすることで、シンボル定義を参照することができます。

外部プロジェクト・ライブラリーへのシンボリック・リンクを追加して、VSCode の [Explorer] ペインに表示できます。VSCode 拡張の **External libraries** をインストールして使用します。

oneAPI サンプルプログラム **t_test.cpp** を使用して、同等の Microsoft* C/C++ 拡張プロジェクトを作成します。次の操作を行います。

1. oneAPI サンプルプログラムをダウンロードします。
2. プロジェクト・フォルダーを新規作成して、oneAPI サンプルからコードファイルをコピーします。
3. パート 5 を参照して、新しいプロジェクトを作成してサンプルを複製します。
4. 新しいプロジェクト構造とプログラム名 (.cpp ファイルの名前と一致) に合わせて、設定ファイルを編集します。
5. パート 5 を参照して、Bear ツールを使用して**コンパイル・データベース・**ファイルを作成します。
6. オリジナルの MKL サンプル・プロジェクトのコンパイル・データベース情報と **make** ファイルを確認します。
7. 新しいプロジェクトの **task.json** ファイルを編集して (図 3 を参照)、プロジェクトにリンクするライブラリーに必要なコンパイラー・パスを追加します。

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "t_test MKL Debug C/C++: Intel icpx build active file",
            "command": "/opt/intel/oneapi/compiler/latest/linux/bin/icpx",
            "args": [
                "-fsycl",
                "-fno-limit-debug-info",
                "-DMKL_ILP64",
                "-fdiagnostics-color=always",
                "-fsycl-device-code-split=per_kernel",
                "-g",
                "-I/opt/intel/oneapi/mkl/latest/include",
                "-L/opt/intel/oneapi/mkl/latest/lib/intel64",
                "-lmkl_sycl",
                "-lmkl_intel_ilp64",
                "-lmkl_sequential",
                "-lmkl_core",
                "-O0",
                "${workspaceFolder}/src/${config:programName}.cpp",
                "-o",
                "${workspaceFolder}/bin/${config:programName}_d"
            ],
            "options": {
                "cwd": "${workspaceFolder}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": "build",
            "detail": "compiler: /opt/intel/oneapi/compiler/latest/linux/bin/icpx"
        },
        {
            "type": "cppbuild",
            "label": "t_test MKL Release C/C++: Intel icpx build active file",
```

```
            "command": "/opt/intel/oneapi/compiler/latest/linux/bin/icpx",
            "args": [
                "-fsycl",
                "-DNDEBUG",
                "-DMKL_ILP64",
                "-I/opt/intel/oneapi/mkl/latest/include",
                "-L/opt/intel/oneapi/mkl/latest/lib/intel64",
                "-lmkl_sycl",
                "-lmkl_intel_ilp64",
                "-lmkl_sequential",
                "-lmkl_core",
                "${workspaceFolder}/src/${config:programName}.cpp",
                "-o",
                "${workspaceFolder}/bin/${config:programName}"
            ],
            "options": {
                "cwd": "${workspaceFolder}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": "build",
            "detail": "compiler: /opt/intel/oneapi/compiler/latest/linux/bin/icpx"
        }
    ]
}
```

図 3: プロジェクトの tasks.json ビルド設定ファイル

8. **Ctrl + Shift + B** でプログラムをコンパイルします。
9. **Ctrl + Shift + D** の後に F5 キーを押して、プログラムを実行してデバッグします。
10. VSCode のドロップダウン・メニューからカーネル・アクセアレーション・デバイスを選択します。

まとめ

ここで紹介した知識とパート 1 ~ 4 を参照することで、VSCode IDE で DPC++ サンプルを変換して、さまざまな利点を得ることができます。

次のステップ

DPC++ プログラムを高速化する多くのターゲットデバイスが、リモートまたはクラウド・プラットフォームで利用できます。次のパート 7 では、VSCode で SSH を使用してリモート・プラットフォームとデバイス、特にインテル® デベロッパー・クラウド・プラットフォームに接続する方法を紹介します。既存の VSCode DPC++ サンプル・プロジェクトをインテル® デベロッパー・クラウドにコピーして、ターゲットノードを選択し、VSCode でコンパイルとデバッグを行います。
