DPC++ と Visual Studio® Code で SYCL* コードをデバッグ - パート 1

この記事は、Codeplay Blogs に公開されている「Debugging SYCL™ code with DPC++ and Visual Studio® Code」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原文を優先してください。

2023 年 3 月 1 日

この記事は、すでに最新の C++ のスキルを習得しており、Khronos の SYCL* オープン・スタンダード、インテル® oneAPI ツールキット、データ並列 C++ の SYCL* 実装を初めて使用する方に、最新の Microsoft* クロスプラットフォーム開発環境 (IDE) である Visual Studio* Code で、SYCL* を使ってハイパフォーマンス・コンピューティング (HPC) アプリケーションを開発するのに必要な情報を提供します。

SYCL* は、最新の C++ プログラムがヘテロジニアス・コンピューティング・アーキテクチャーを効率良く利用できるようにする、オープン・スタンダードの C++ API です (SYCL* 1.2.1 は C++11 ベース、SYCL* 2020 は C++17 ベースであるため、DPC++ には C++ 17 以降が必要です)。主な利点の 1 つは、コードを変更することなく、さまざまなヘテロジニアス・アーキテクチャーをターゲットにできることです。ヘテロジニアス・アーキテクチャーは、GPU、FPGA、マルチコア CPU、DSP など、あらゆる種類と数のアクセラレーター・デバイスで構成されています。

DPC++ は SYCL* 標準の実装であり、オープンソースの LLVM Clang コンパイラー・プロジェクトをベースとしています。DPC++ プログラムは、gdb-one API デバッガーを使用して、ホストデバイス (通常は CPU) から、行列ベクトルモデル (GPU) または空間パイプライン・モデル (FPGA) デバイスに移動して実行し、結果を返すコードとデータの検証およびチューニングを可能にします。

oneAPI は、ヘテロジニアス HPC システムで DPC++ プログラムを効率良く実行できるように、パフォーマンス・ライブラリーやチューニング・ツールなどを含むツールキットを実装しています。インテルは、oneAPI の習得を容易にするため、多数のビデオ、記事、サンプルプログラム、複数のクロスプラットフォーム開発環境の設定方法に関する情報を ウェブサイトで提供しています。

ほとんどの DPC++ プログラムのコンパイルには、Make や CMake ファイル、シェルスクリプトなど、従来のコマンドライン・ベースの C/C++ 設定およびビルドシステムが使用されています。Microsoft * Visual Studio* IDE などの GUI 指向の開発環境を使用している場合、DPC++ プログラミングは、特にプログラムのデバッグに関して、馴染みのある環境とはやや異なります。そのため、GUI IDE のプログラマーは、プログラミングを開始する前に、コマンドライン開発環境の使い方を理解する必要があります。また、機能豊富な GUI IDE とは異なる、基本的なデバッグツールでの作業にも慣れる必要があります。

Visual Studio* Code IDE と Microsoft の C/C++ 拡張機能を使用することで、コマンドラインでの開発アプローチと比較して以下のようなメリットが得られます。

- IntelliSense コードを強調表示したり、変数、関数、クラスにホバーして詳細を表示したり、それらの定義にジャンプできます。
- 複数のウィンドウや分割ビューでコードファイルを表示できます。
- インタラクティブなデバッグにより、関数のステップインまたはステップアウト、ブレークポイントの動的な設定または削除、スタックスコープの表示、リアルタイムでの変数の監視が可能です。
- IDE を使用しながら、ターミナルを使用してデバッガーコマンドを実行できます。
- IDE でプロジェクト構成を設定できます。
- デバッグ中にほかのプロジェクト・ファイルに即座にアクセスできます。
- IDE インターフェイスを通じて、プロジェクトを Git リポジトリーにアタッチできます。

このシリーズでは、Visual Studio* Code IDE を使用して、HPC プラットフォーム上で SYCL* プログラミングと DPC++ によるデバッグを行うため、Visual Studio* Code IDE を設定して利用する手順を説明します。

背景

私は C++ プログラマーとして、長年 Microsoft* Visual Studio* を使用して、PC や PlayStation* 向けアプリケーションやゲームツールの開発に従事してきました。これまで、Linux* で開発する機会はほとんどありませんでした。DPC++ のドキュメントやコードの多くは、Linux* または Windows* 向けです。Visual Studio* Code IDE は、DPC++プログラムを開発し、調査するためのオプションとして言及されていますが、IDE の使用は基本的なものであり、デバッグ機能については言及されていません。以前 Microsoft* Visual Studio* で行ったように、DPC++ 開発向けに Visual Studio* Code を設定して、GUI 環境でビルドおよびデバッグを行う方法に関するドキュメントを見つけるのは容易ではありませんでした。このシリーズでは、これらの課題に対応します。

はじめに

このシリーズでは、以下の手順を説明します。

- Visual Studio* Code IDE を使用して、DPC++ をコンパイルする環境を素早く設定します。
- DPC++ を使用する SYCL* 開発向けに Ubuntu* を準備します。
- 複数のデバイスで DPC++ コードをデバッグします。

以下の手順では、Ubuntu* 20.04.5 LTS と Visual Studio* Code バージョン 1.72.2 を使用します。ほかのプラットフォームを使用する場合、ここで説明する Visual Studio* Code 要素はそのまま適用します。その他の要素については、使用するプラットフォームで動作するように変更する必要があります。

このシリーズは、インストールしたばかりの Ubuntu* マシンに Visual Studio* Code をインストールし、DPC++ 開発向けに設定を行う必要があったため、アイデアを思いつきました。同じような状況にある方は、Visual Studio* Code IDE が使えるようになるまで、インテル® oneAPI ツールキット、および OpenCL* やその他の計算やグラフィックス・ドライバーをインストールしないことをお勧めします。それらの手順は、次のセクションで説明します。これにより、問題が DPC++ にあるのか、それとも C++ ツールチェーンにあるのかの見極めが不要になります。

C++ 環境の設定

C++ プロジェクトの開発環境をまだ設定していない場合、Visual Studio* Code の設定に必要なことを説明している以下のガイドを参照してください。

Ubuntu* で C++ 開発向けに Visual Studio* Code を設定 - パート 2

すでにこの設定を行い、Visual Studio* Code での C++ プロジェクトの操作に精通している場合は、この設定をスキップして次のステップに進んでください。

oneAPI と DPC++ を使用して SYCL* を設定

C++ 開発環境の準備ができたら、oneAPI と DPC++ を使用して SYCL* を設定します。

DPC++ と Visual Studio* Code による SYCL* のデバッグ - パート 3
