クロスプラットフォーム・ソフトウェア開発にツールを活用

Windows* Subsystem for Linux* 2 と Visual Studio* Code を使用して Linux* と Windows* 向けに oneAPI をビルドする方法

Tony Mongkolsmai インテル コーポレーション テクノロジー・エバンジェリスト

皆さんこんにちは。インテルで 19 年間、パフォーマンス・ツール、並列化ツール、AI システム・プラットフォームのソフトウェア・エンジニアおよびソフトウェア・アーキテクトとして得た私の経験を皆さんと共有し、ソフトウェア、テクノロジー、インテルについて皆さんの考えをお聞かせいただければ嬉しく思います。最初に、現代のソフトウェア開発者が抱えるジレンマを解決しましょう。

物心ついたときから、私はプログラミングや問題を解くことが好きでした。プログラミングがまだそれほど一般的ではなかった 1985 年、小学 1 年生のときに BASIC を使ってプログラミングを始めました。幸か不幸か、これまで数十のプログラミング言語を学び、いくつかの IDE でプログラミングし、Windows* と Linux* の両方の環境でソフトウェアを開発する経験に恵まれました。また、ウェブ /UI フレームワーク、クラウド開発、ディープラーニングなど、急速に進化する分野で開発する必要がありました。

この間、一貫して言えることは、エンジニアはより生産性を高めるため、常にツールと戦っているということです。開発者として、私はより多くの仕事をこなすため、あるいはプライベートの時間を確保するため、できるだけ効率的でありたいと思っています。そのため、これは私個人にとって大きな意味を持ちます。

対象を絞り込む

現代のソフトウェア開発者は 1 つのツールやプラットフォームだけを理解していれば良い訳ではありません。マイクロサービスや Kubernetes* 向けの Go* コードの記述には、React*/Angular*/Vue*/JavaScript* コードの記述とは全く異なるツールチェーンを使用します。また、多くの場合、Windows*/Linux*/Mac* 間のクロスプラットフォーム・サポートが必要とされます。

幸い、クロスプラットフォームの IDE とサポートは進化しています。私は長年 Vi を使用していますが（Emacs* の方が強力なのは知っています）、最新の IDE が好きです。エンジニアにとって、さまざまなプラットフォームでワークフローを標準化することは効率の向上につながります。私が通常使用する言語（C/C++、Python*、JavaScript*、Go*）はすべて Microsoft* Visual Studio* Code（VSCode）内でうまく動作し、Microsoft* Windows* Subsystem for Linux*（WSL）の最近の更新によりクロスプラットフォーム開発にも対応できます。

ここでは、私のインテル® Core™ i9 プロセッサー搭載の Alienware* R13 システム（インテルからの支給品ではありません）で、いくつかのクロスプラットフォームの oneAPI サンプルを動かしてみたいと思います。

WSL と VSCode のセットアップ

WSL のセットアップは簡単です。Windows* のコマンドプロンプトを開いて、次のコマンドを実行するだけです。

```
> wsl --install
```

必要な再起動を行った後、WSL 用の Ubuntu* 20.04 がインストールされます。

問題が発生した場合は、Microsoft の「トラブルシューティング・ガイド」（英語）の説明に従って、システムを WSL 対応にするのに必要な手順を実行します。

次に、VSCode をセットアップします。Linux* のインストール手順に従って WSL で VSCode をセットアップするのではなく、Microsoft* の手順（英語）に従って VSCode で Remote WSL 拡張を有効にします。

有効にしたら、VSCode WSL 接続ウィンドウとして新しい WSL ウィンドウ（**図 1**）を選択するだけです。

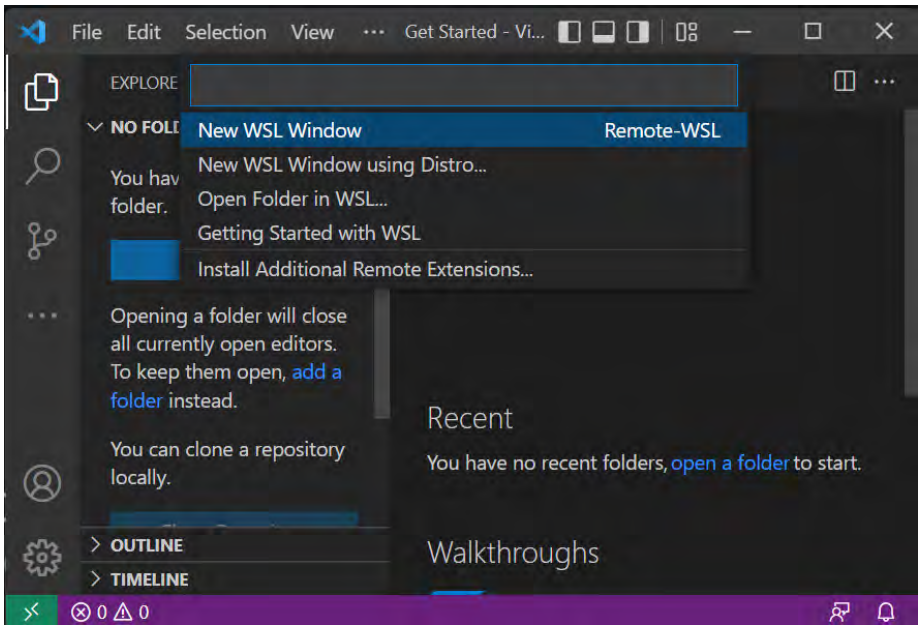

図 1. 新しい WSL ウィンドウを開く

インテル® oneAPI ツールキットのインストール

ここでは、システム上の Linux* と Windows* の両方で oneAPI ベースのコードが動くか確認するため、インテル® oneAPI ツールキットのリリースページ（英語）（**図 2**）からインストールするツールキットを選択し、指示に従ってインストールしました（**図 3**）。WSL ターミナルでは apt インストールを使用し、Windows* ではオンライン・インストーラーを使用しました。

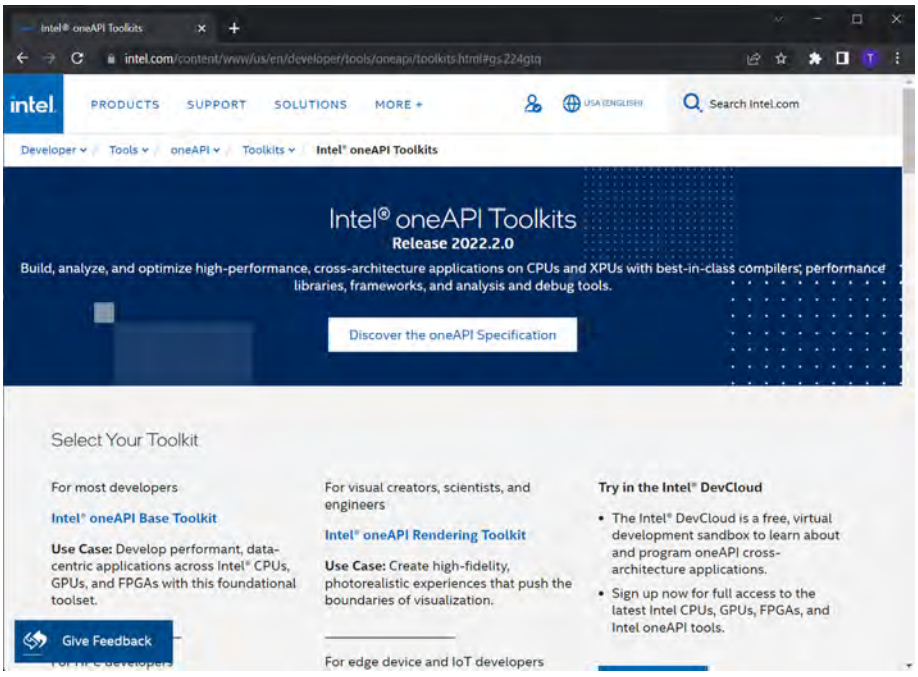

図 2. インテル® oneAPI ツールキットの選択

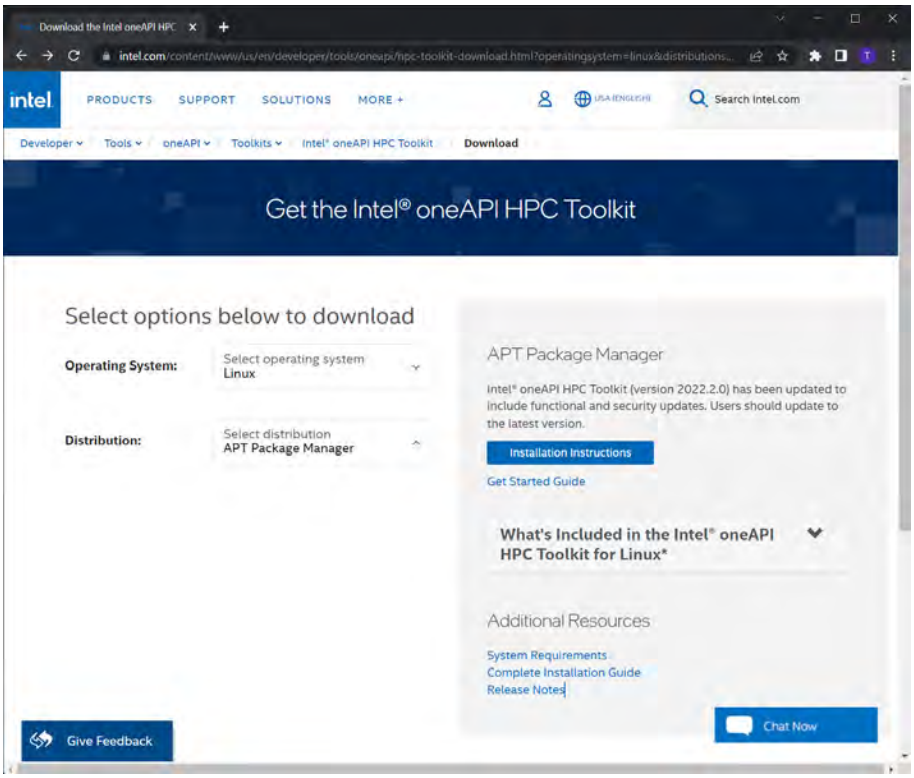

図 3. インテル® oneAPI ツールキットのインストール

コードの実行

開発環境の準備ができたら、GitHub* から oneAPI-samples リポジトリーをクローンしました。

このコードは Windows* と Linux* の両方に対応するように設計されているので、WSL の VSCode ウィンドウで [Libraries] > [oneMKL] > matrix_mul_mkl ディレクトリーをロードし、bash ターミナルを開き、以下のコマンドで oneAPI 環境をセットアップしてサンプルコードをビルドしました。

```
> source /opt/intel/oneapi/setvars.sh
> make
```

行列乗算サンプルがビルドされ、実行されました（**図 4**）。

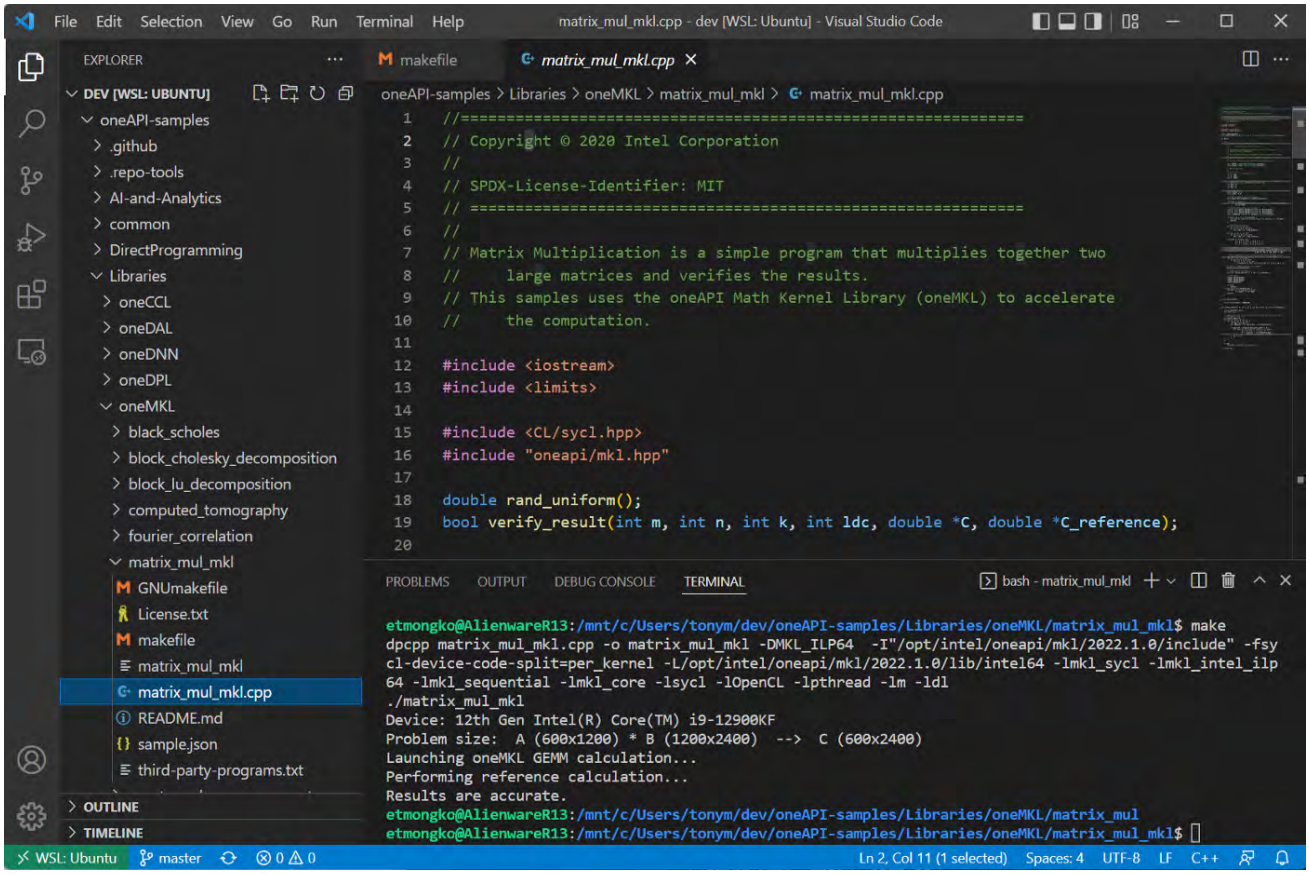

図 4. サンプルコードのビルドと実行

Windows* バイナリーを使用したサンプルの作成と実行は、新しいシステムであったため、少し手間がかかりました。Microsoft* Visual Studio* 2022 Community Edition と GNU* Make をインストールする必要がありました。その後は、VSCode をロードして実行するだけでした。

```
> C:\Program Files (x86)\Intel\oneAPI\setvars.bat
> make
```

思いがけない発見

多くの方はご存じかもしれませんが、WSL 上ではなく、Windows* 上の VSCode を操作していたら、分割ターミナルに WSL プロンプトを表示できることが分かりました（**図 5**）。これは私にとって驚くべき発見でした。これにより、同じ VSCode インスタンスで、Windows* と Linux* の両方のサンプルをビルドして実行することができました。1 つの IDE で、1 つのシステムで、クロスプラットフォーム開発ができるのです！クロスプラットフォーム開発の苦しみに何年も耐えてきた私にとって、これは言葉では言い表せないほどの感動でした。

図 5. 同じ VSCode インスタンスでの Windows* と Linux* の実行

まとめ

今回素晴らしかったことは、新しいシステムでわずか 2 時間ほどでセットアップできたことです。今後は、インテル® Arc™ GPU での検証、ほかの開発ワークフロー（Jupyter* Notebook など）、クラウド・コンピューティングや AI システム・プラットフォーム開発などのソフトウェアに関するトピックについて、テクノロジー業界のソフトウェア・エンジニアとしての経験を共有しながら、執筆する予定です。最後までお読みいただきありがとうございます。