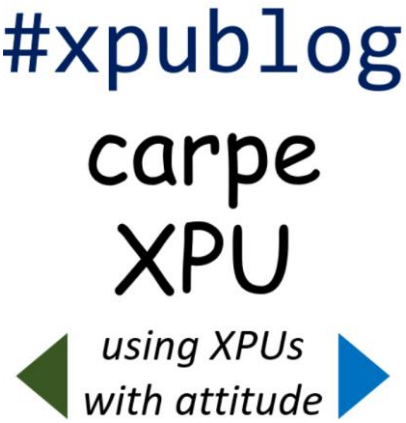
SYCL* と DPC++ へようこそ

この記事は、インテル® デベロッパー・ゾーンに公開されている「Hello SYCL and DPC++」の日本語参考訳です。



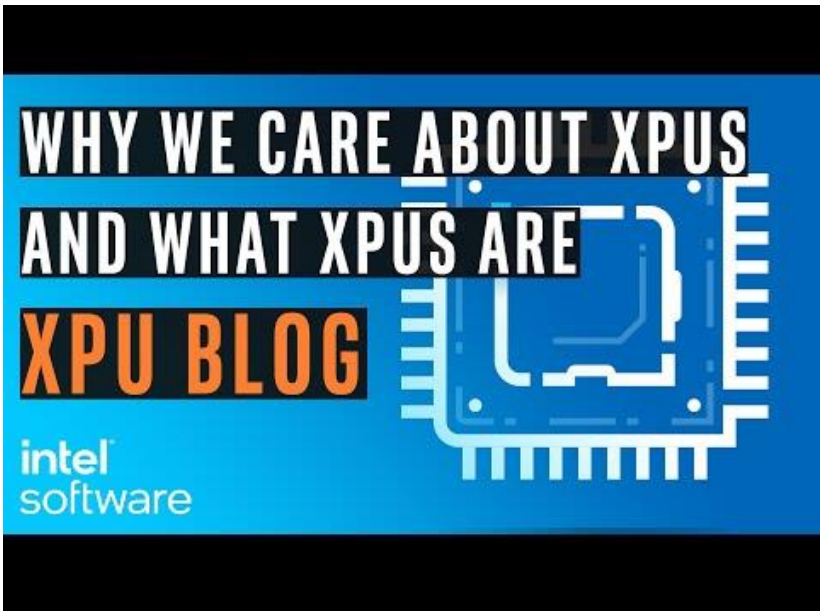
XPU 向けソフトウェア開発に関する本ブログへようこそ。第 1 回目である本記事に関して、記事の最後にあるリンクからフィードバックをお寄せください。ほかの読者のコメントを見ることもできます。フィードバックの内容は、今後の記事に役立てさせていただきます。

この #xpublog 記事では、次の 3 つの疑問に答えます。

1. XPU とは? XPU が重要な理由は?
2. SYCL* とは? 注目すべき理由は?
3. oneAPI 向けインテル® DevCloud (無料で簡単に利用可能) を試すには?

ほかの #xpublog (英語) 記事と同様に、本記事には動画の内容に関する、リンクを含む重要な詳細が含まれています。

XPU とは? XPU が重要な理由は?



この 20 年の間に、すべてのコンピューティングが並列コンピューティングになりました。これは、インターネット、クラウド、マルチコアなどがユビキタスになってきたからです。並列処理や並行処理は日常的に行われています。次の進化は、ヘテロジニアス・コンピューティング (XPU) への移行です。この変化は、順調に進んでいるとはいえ、まだ初期段階です。やがてそれは、すべてのコンピューティングを再構築し、すべてがヘテロジニアスになるでしょう。

メタファーとしての XPU

コンピューティングの多くは「計算」を目的としているため、ここでは「処理ユニット」に注目します。処理ユニットの役割は、データを処理すること、つまり「計算」です。

一般的な処理ユニットとしては、CPU (中央演算処理ユニット) と GPU (グラフィックス演算処理ユニット) があります。これは XPU に一般化することができます。APU、BPU、CPU、DPU、EPU、FPU、GPU、… ZPU までの名前 (アルファベットの 26 文字だけを考慮) をインターネットで検索すると、YPU だけが利用可能です (Yarn Processing Unit がありますが、この場合 Unit は企業の一部門を意味します)。このため、「FPGA も DSP も ASIC も、すべて XPU だ」と言っても、「XPU は文字通りの定義ではなく、メタファーと考えるべきであり、名前空間を拡張しているだけだ」と認識していただけると思います。

すべての XPU を受け入れよう

今後 10 年以内に、すべてのプログラミングは XPU プログラミングになるでしょう。1 種類の処理ユニットですべての計算を行うことを想定してプログラムを書く時代 (ホモジニアス・コンピューティング) は終わります。

このような変革が必要とされているのは、多くの理由からコンピューティングが成熟し、非常に多様なコンピューティング技術が含まれるようになってきたからです。CACM* の最新号では、「The Decline of Computers as a General Purpose Technology (汎用技術としてのコンピューターの衰退)」と題した記事で、その傾向と根本的な原因を探っています。個人的には「コンピューティングの成熟」と考えていますが、おそらくこのタイトルでは雑誌があまり売れないでしょう。

(* The Decline of Computers as a General Purpose Technology, N. Thompson, and Svenja Spanuth, Communications of the ACM, March 2021, vol. 64, no. 3.)

SYCL* とは? DPC++ とは?

この動画では、簡単な SYCL* コードについて説明しています。

動画内で紹介しているインテル® DevCloud にサインアップする方法についてのビデオは、次のセクションにあります。まだインテル® DevCloud にサインアップしていない方は、このビデオを先にご覧になることをお勧めします。

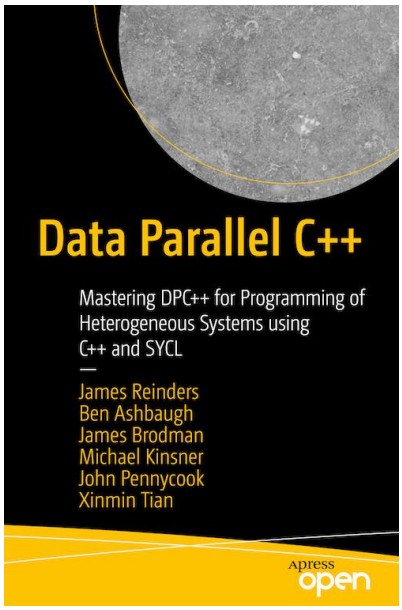
動画内で使用しているコードはすべて xpublog の GitHub* リポジトリー (英語) にありますが、インテル® DevCloud 上ですでにコードが入った Jupyter* Notebook を使って試すのが一番簡単です (手順は次のセクションを参照してください)。インテル® oneAPI ベース・ツールキット (英語) から DPC++ コンパイラーをダウンロードするか、LLVM ソース (英語) からこちらの手順 (英語) に従って DPC++ コンパイラーをビルドして、SYCL* プログラムを実行することもできます。

SYCL* は、ヘテロジニアス・ハードウェア (前述のとおり、私は XPU と呼んでいます) 向けのシングルソース C++ データ並列プログラミングのためのオープンスタンダードです。SYCL* は、ホストに C++ を使用しデバイスにドメイン固有のカーネル言語を使用するのではなく、システム上の複数のデバイスを対象に C++ でシングルソース・コンパイルを可能にします。

SYCL* は、C++ にカーネルスタイルのプログラミングと、システム内のアクセラレーターを検索、照会、使用するためのメカニズムをもたらします。カーネルベースのプログラミングは、OpenCL* や CUDA* でもサポートされており、データ並列性を活用するための重要なプログラミング・スタイルです。標準的な方法でアクセラレーターを列挙してアクセスする機能は、OpenCL* で導入されました。SYCL* は、OpenCL* の経験と、C++ プログラミングを強力にサポートしたいという要望から生まれた取り組みの結果です。SYCL* 標準は、Codeplay によって導入され、Khronos によって管理されています。

DPC++ は、いくつかの拡張機能を含む SYCL* を実装する LLVM プロジェクトです。現在 SYCL* 2020 に搭載されている多くの機能のプロトタイプ作成には DPC++ が使用されています。そのため、DPC++ は SYCL* 2020 の多くの機能を標準の策定前から実装していました。SYCL* 2020 仕様全体との整合性をとる作業が行われていますが、これらの作業はすべて非常に活発なオープンソース・リポジトリーで簡単に確認できます。DPC++ は、インテルの CPU、GPU、FPGA をターゲットとするためインテルによって提供されています。また、DPC++ は Codeplay によって NVIDIA* GPU をターゲットとして使用されています。hipSYCL を含むその他の取り組みでも DPC++ を利用することができます。SYCL* のサポートに LLVM を使用することは、ポピュラーなアイデアであり、今後も多くの開発が行われることでしょう。

書籍『Data Parallel C++』



私も執筆に参加した書籍『Data Parallel C++』の第 1 章では、SYCL* と DPC++ についてより詳しく紹介しています。今後の記事では、書籍の後半に出てくるトピックを紹介し、内容を補足したり、必要に応じて更新する予定です。

『Data Parallel C++』 - 無料ダウンロード (英語)

以下の資料は、特にこの書籍を読んだ後の参考になります。

SYCL* 2020 リファレンス・カード (16 ページ) 英語 | 日本語:
便利なリファレンスです。

オンラインの DPC++ リファレンス (英語):
インターフェイスの詳細を確認できます。

SYCL* 2020 言語仕様 (英語)

SYCL* 2020 が鍵

SYCL* 2020 仕様は、業界内の多くの献身的な人々による長年の仕様開発の成果です。SYCL* 2020 は、SYCL* 1.2.1 の機能をベースに、プログラマビリティーの向上、コードサイズの縮小、パフォーマンスの向上を実現しています。C++17 をベースにした SYCL* 2020 は、標準的な C++ アプリケーションの高速化を可能にし、ISO C++ のロードマップとの連携を強化しています。

Khronos Group は、SYCL* 2020 の発表 (英語) において、7 つの重要な SYCL* の新機能を強調しました。

- 統合共有メモリー (USM) により、ポインターを使用するコードがバッファーやアクセサーなしで自然に動作するようになります。
- 並列リダクションにより、組込みのリダクション操作が追加され、定型的なコードを回避し、リダクション操作の高速化に対応したハードウェアで最大限のパフォーマンスを実現できます。
- ワークグループとサブグループのアルゴリズムにより、ワークアイテム間の効率的な並列処理が追加されました。
- クラス・テンプレート引数推定 (CTAD) およびテンプレート推定ガイドにより、クラス・テンプレートのインスタンス化が簡素化されました。
- 簡略化されたアクセサーと組込みリダクション操作により、定型的なコードを軽減し、C++ ソフトウェアのデザインパターンを効率良く使用できます。
- 相互運用性の拡大により、多様なバックエンド・アクセラレーション API による効率的な高速化を実現します。
- SYCL* のアトミック操作が標準 C++ のアトミック操作に近づき、並列プログラミングの自由度が向上しました。

目標 = C++

最終的な目標は、C++ 標準に影響を与えることです。個人的には、一般的な慣行を標準化することに大賛成であり、C++ 標準化委員会の時間をかけて慎重に作業を進める姿勢を支持しています。なぜならば、いったん C++ 標準に採用されれば、すべてのコンパイラーはほぼ永久的にそれをサポートする必要があるからです。

探究し、学習し、達成することは楽しみであり、それが一般的な慣行につながります。SYCL* 2020 とその実装 (ArrayFire の Umar Arshad 氏が執筆した SYCL* 実装の概要 (英語) は一読に値します) は、それを現実のものにします。

『Data Parallel C++』 (英語) は、SYCL* 2020 標準のリリースよりも 3 カ月前に出版されましたが、新しい標準に対応しています。今後の記事で、いくつかの変更点と書籍の正誤表 (英語) について説明する予定です。

インテル® DevCloud について

『Data Parallel C++』(英語) に加えて、SYCL* を学び、DPC++ を使用するための素晴らしいオンラインリソースが用意されています。

インテルでは、「DPC++ の要点を学ぶ」(英語) という、学習とハードウェアを使った実践 (オンラインアクセス) を組み合わせたトレーニングを提供しています。oneAPI 向けインテル® DevCloud は、DPC++ を使用して SYCL* をサポートする CPU、GPU、FPGA へのアクセスを提供する無料のオンラインリソースです。Jupyter* Notebook を使用した革新的なトレーニングで、DPC++ を簡単かつ生産的に学ぶことができます。DPC++ コンパイラーをダウンロードして、本の例に従うという「従来の方法」で学ぶこともできます。DPC++ コンパイラーのバイナリーの入手方法や、ソースコードからビルドする方法などの説明は、『Data Parallel C++』を参照してください。

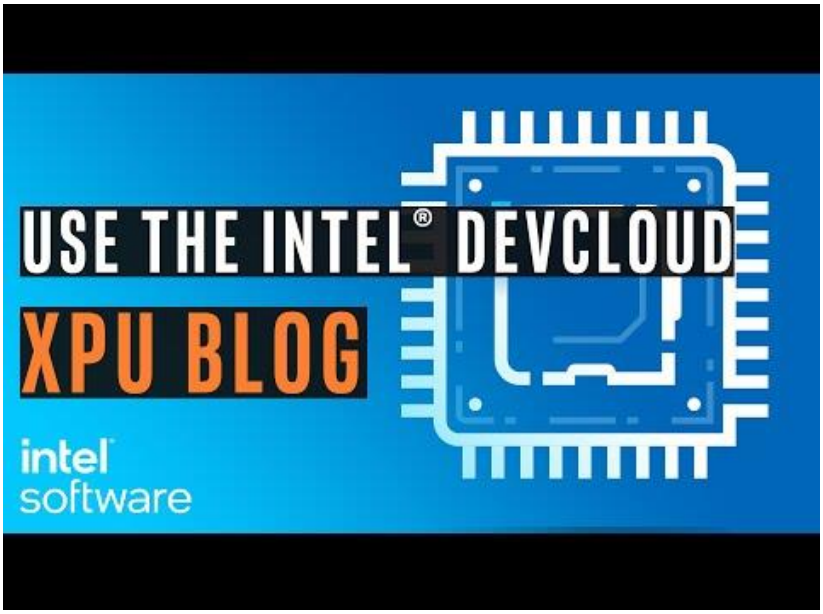
Codeplay (英語) では、コミュニティー・エディションのコンパイラー、SYCL* Academy (英語)、多くのコード例 (英語)、ブログ、その他の学習に役立つ資料を提供しています。ぜひ活用してください。

Khronos は、SYCL* の学習と使用に役立つ追加のリソース (英語) を管理しています。

2020 年に開催された IWOCL/SYCLcon 2020 (英語) カンファレンスの講演内容は、すべてオンラインで公開されており、豊富な情報が得られます。同じく 2020 年に開催された、Intel® oneAPI Developer Summit 2020 (英語) の講演内容もすべてオンラインで公開されており、その多くは DPC++ で SYCL* を使用することに注目しています。

IWOCL/SYCLcon 2021 (英語) とプレカンファレンスの oneAPI サミット (英語) (「at IWOCL21」) では、OpenCL* や SYCL* に興味のある方を対象に、講演やパネル・ディスカッション、ネットワーキングが実施されます。もちろん私は参加する予定です。バーチャルで皆さんにお会いできることを楽しみにしています。

無料で簡単な oneAPI 向けインテル® DevCloud を今すぐ試してみる



この動画では、インテル® DevCloud 上でコードを実行するための手順を説明します。動画中のすべてのリンクはこのセクションに記載されています。

インテル® DevCloud は、最新のインテルのハードウェアとソフトウェアで構成されたクラスター上で、ワークロードを無料で開発、テスト、実行できる場所です。私自身、DPC++ 書籍の執筆中に使い始めました。

「Hello SYCL」動画では、Jupyter* Notebook 内で簡単なプログラムをビルドして実行する方法を紹介しました。

以下は、ビデオで紹介した初回**アクセスの方法**です。以下のリンクは、動画内で使用したものです。

1. oneAPI 向けインテル® DevCloud 申し込みフォーム (英語) から申し込みます。
2. UUID が記載されたメールが届きます。サインイン時に入力を求められたら、この UUID (長い英数文字列) をコピーしてペーストします。
3. メインのトレーニング・ページ (英語) 上にある Sign In (英語) リンクをクリックします。
4. メインのトレーニング・ページをスクロールダウンして、「Launch Jupyter Notebook」をクリックします。
5. 「Server not running」ページが表示された場合は、「Launch Server」をクリックします。
6. 起動されるまで待ちます (通常は 1 分未満です)。
7. 「Other」の下にある「Terminal」をクリックします。必要なファイルをコピーするには、次のコマンドを実行します。
   /data/oneapi_workshop/get_jupyter_notebooks.sh

8. (上記の動画で示すように) ナビゲーション・ウィンドウで /xpublog に移動します。
9. Welcome.ipynb をダブルクリックして開きます。
10. 後で簡単に戻れるようにこのページをブックマークします。

戻るには:

1. ブックマークしたリンクをクリックします。トレーニング・ページにリダイレクトされた場合 (あるいは何らかの理由で Welcome ページが表示されなかった場合)、Sign In (英語) をクリックして、再度 UUID を指定します (ブラウザーが UUID を記憶している場合は UUID ボックスでプルダウンから選択できます)。
2. メインのトレーニング・ページ (英語) をスクロールダウンして、「Launch Jupyter Notebook」をクリックします。
3. 「Server not running」ページが表示された場合は、「Launch Server」をクリックします。このページは、サーバーが実行されていない場合にのみ表示されます。
4. 起動されるまで待ちます (通常は 1 分未満です)。
5. 今回はファイルをコピーする必要はありません。
6. (上記の動画で示すように) ナビゲーション・ウィンドウで /xpublog に移動します。
7. Welcome.ipynb をダブルクリックして開きます。

しばらくしたら、xpublog にアクセスするためのページを作成する予定です。作成したらこの手順と動画を更新します。

oneAPI については今後のブログでもっと取り上げる予定ですが、すでに膨大な量の素晴らしいトレーニングが用意されています。インテル® DevCloud 上の Jupyter* Notebook は、対話機能を使ってトピックを説明してくれるため、活用することをお勧めします。サンプルプログラムを編集して、自分の理解度を確認したり、新しいことを試したりすることができます。詳細は、インテル® oneAPI DevCloud のトレーニング・ページ (英語) を参照してください。最初に、インテル® oneAPI ベース・ツールキットの「View Training Modules」(英語) をクリックして開始することをお勧めします。

今後のトピック

今後のトピックに関する提案を含め、皆さんからのフィードバックをお待ちしております。ドキュメントからは読み取れない、経験や開発者同士の共有から生まれる視点や意味のある詳細を提供していく予定です。Paul Harvey 氏が 1 年かけてラジオの定番番組にした「The Rest of the Story」のソフトウェア・エンジニアリング版を提供したいと思っています。

私のブログでは、oneAPI、SYCL*、DPC++、C++、パフォーマンス、並列プログラミングなど、XPU (ヘテロジニアス・コンピューティング) 向けのソフトウェア開発について取り上げる予定です。Python* や Fortran についても触れるかもしれません。どちらも素晴らしいツールです。皆さんが興味を持ち、私がお役に立てるトピックを掘り下げていきたいと思っています。

皆さんのご意見、ご感想、そして楽しい議論をお待ちしています。

コメントをぜひお寄せください

#xpublog の感想、フィードバック、提案などのコメントを community.intel.com James-Reinders-Blog (英語) からぜひ投稿してください。

製品とパフォーマンス情報

[1] 実際の性能は利用法、構成、その他の要因によって異なります。詳細については、www.Intel.com/PerformanceIndex (英語) を参照してください。