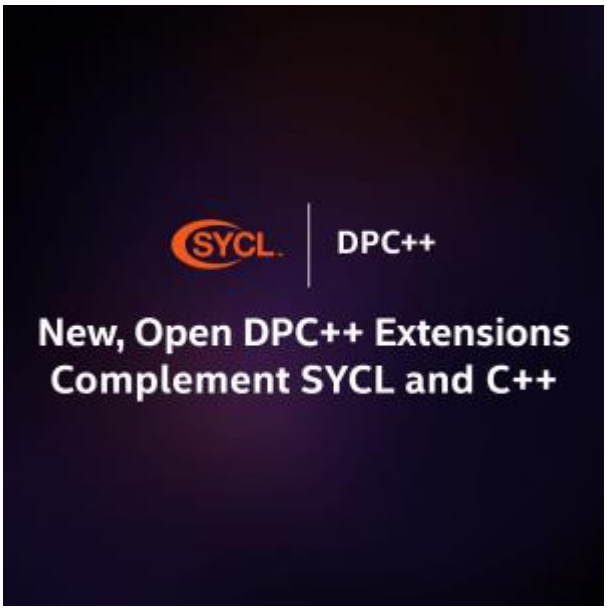
SYCL* と C++ を補完する新しいオープンな DPC++ 拡張

この記事は、Intel TECH.DECODED に公開されている「New, Open DPC++ Extensions Complement SYCL and C++」の日本語参考訳です。

ヘテロジニアス・アーキテクチャーで並列プログラミングを容易にする機能を追加



数十年前までは、「1 つのワークロードに 1 つのプロセッサー」という考え方が常識でした。しかし現在では、ヘテロジニアス・コンピューティングの台頭により、そのような考え方は過去のものとなっています。最新のテクノロジーは、ハイパフォーマンス・コンピューティング (HPC)、人工知能 (AI)、高度なレンダリング、モノのインターネット (IoT) システム時代を切り開くのに必要な卓越したスピードとスケーラビリティーを提供することができます。このような進化は、業界全体に大きなチャンスをもたらす一方で、アプリケーション開発に課題をもたらします。業界では、ソフトウェア開発の生産性を向上させると同時に、さまざまなアーキテクチャーに対するパフォーマンスを実現するため、oneAPI イニシアチブを採用しています。

データセントリックなワークロードとアプリケーションは、近年多様化の道を歩み続けています。ハードウェアの選択によって、ツールや開発言語のサイロ化が進んでおり、独自のソリューションと開発言語の多様性により、「再利用可能なコード」の量は制限されます。データ並列 C++ (DPC++) は、この問題に対応するため ISO C++ を進化させたものです。コーディングの生産性を向上させ、多様なアーキテクチャーでのプログラミングのパラダイムを変えることができます。oneAPI は、DPC++ (主要言語) により、そのプロセスを容易にします。

オープンな仕様は重要です。これがないと、開発者は言語の経年変化や将来の互換性を予測できません。したがって、イノベーションにはオープンな開発が必要なのです。「ウォールドガーデン」があると、イノベーションは停滞し、アーキテクチャー間のパフォーマンスが制限されます。C++ は広く普及している強固な言語ですが、主に CPU 向けに定義されているため、いくつかの制限があります。現在の標準規格では、何らかの機能追加なしではヘテロジニアスなプログラミングに対応できません。

Khronos Group が中心となって開発した OpenCL* は、複数のアーキテクチャーに対して低レベルで詳細なプログラミングを可能にしましたが、OpenCL* コンソーシアムのメンバーが独自の拡張機能を開発し始めたため、規格が事実上分裂し、独自の言語が幅を利かせるようになってしまいました。Khronos Group は、OpenCL* の経験を基に、SYCL* プログラミング・モデルでプロセスをさらに簡素化する革新を行いました。OpenCL* と同様に、SYCL* はヘテロジニアスなプログラミングをサポートしますが、標準の C++ 構文を使用し、ホストとアクセラレーターのコードをシングルソースで提供します。

インテルの oneAPI 製品担当シニア・ディレクターの Joe Curley は次のように述べています。

> 「高速なコンピューティングは、CPU、GPU、FPGA、AI 技術の進歩により、ここ数年で多様化が進んでいます。これにより、開発者が新しいハードウェアの可能性を実現し、開発コストや複雑さを最小限に抑え、ソフトウェア投資を最大限に再利用できる、オープンでクロスプラットフォームな言語の必要性が高まっています。」

現在、Khronos Group の SYCL* は、AMD、ARM Mali、NVIDIA、インテルなどのハードウェア・ベンダーをサポートするオープンソースの実装を提供しています。

ヘテロジニアス・アーキテクチャー向けのプログラミングを容易にする

DPC++ は、oneAPI 業界イニシアチブの一環として開発されたクロスアーキテクチャー言語であり、ヘテロジニアスの課題を解決するため、既存の取り組みを補完します。DPC++ は、クロスアーキテクチャー・システムを促進するため、C++、SYCL*、拡張機能の 3 つを組み合わせています。また、DPC++ は、SYCL* 仕様の将来の改訂版に新しい機能を導入する標準ベースの言語実装としても機能します。インテルは、この取り組みの主要な参加者として、oneAPI の実装を構築しています。

Joe Curley は、言語のオープン性への取り組みを強化しました。

> 「DPC++ は C++ と SYCL* というオープンスタンダードをベースにしていますが、拡張機能により迅速に実験と革新を行い、それらを開発し、Khronos SYCL などのオープンスタンダード団体への好循環を確立することができます。DPC++ は、開発者がデータ並列プログラミングを迅速に進化させるためのオープンなメカニズムを提供します。」

最近発表された Codeplay 社の NVIDIA* GPU 用 DPC++ コンパイラーに見られるように、DPC++ の採用はすでに勢いを増しています。

新しい DPC++拡張が SYCL* と C++ を解き放つ

DPC++ には、現在利用可能なツールを補強するための 30 近い新しい拡張機能 (英語) が用意されており、SYCL 2020 仕様にはその多くが組み込まれています。

- **統合共有メモリー (USM)** は、ポインターベースのメモリーアクセスと管理インターフェイスを定義します。USM は、ホストとデバイスの両方で可視化され、一貫したポインター値を持つ割り当てを可能にします。デバイスや実装のサポートの程度に応じて、さまざまな USM 機能レベルが定義されています。
- **インオーダーキュー**は、キューのセマンティクスを定義し、一般的なコーディング・パターンを効率化します。
- **オプションのラムダ名**を使用すると、カーネルを定義するラムダに手動で名前を付ける必要がなくなります。また、コーディングを簡素化し、ライブラリーとの互換性を実現します。デバッグや sycl::program オブジェクトとのインターフェイスなどでは、ユーザーはオプションでラムダに名前を付けることができます。
- **推論ガイド**は、C++ のクラス・テンプレート引数推論 (CTAD) を有効にすることで、一般的なコードパターンを簡素化し、コードの冗長性と長さを軽減します。

- **リダクション**は、明示的にコーディングすることなく、共通のリダクション・パターンを利用することで生産性を向上させます。リダクションを言語に組み込むことで、デバイス、ランタイム、リダクション・プロパティーの組み合わせに応じて最適化された実装が可能になります。
- **サブグループ**は、ワークグループ内のワークアイテムのグループ化を定義します。サブセット内のワークアイテムを同期するプロセスは、ほかのサブグループのワークアイテムとは独立して行うことができます。同時に、一般に SIMD ハードウェアにマッピングされるサブグループは、グループ内のワークアイテム間の通信操作を公開しています。
- **サブグループ・アルゴリズム**では、サブグループ内のワークアイテムを対象とした、サブグループでのみ利用可能な集合操作を定義します。また、サブグループの集約操作として、より一般的な「グループ・アルゴリズム」拡張のアルゴリズムも利用できるようにします。
- **エンキューされたバリア**は、いくつかの一般的なプログラミング・パターンにおいて、依存関係の作成と追跡を容易にします。この利点により、細粒度の依存関係を明示的に作成することなく、キュー内で粗粒度の同期が可能になります。
- **拡張アトミック**では、C++20 に準拠したアトミック操作が可能で、浮動小数点型や省略形演算子のサポートも含まれます。
- **グループ・アルゴリズム**では、ブロードキャスト、リダクション、スキャンなど、ワークアイテムのグループ間の集合操作を定義します。明示的なコーディングを必要としないアルゴリズムで生産性を向上させるとともに、デバイスとランタイムの組み合わせに応じて最適化された実装が可能です。
- **グループマスク**は、グループからのワークアイテムのセットを表すことができるタイプと、ballot や count など、そのタイプを作成または操作する集合操作を定義します。
- **全引数制限**は、カーネル (カーネルのラムダ定義を含む) に適用できる属性を定義します。これは、カーネルに渡される、またはカーネルによってキャプチャーされるポインター引数の間にメモリー・エイリアシングがないことを示します。開発者がカーネル引数について、コンパイラーが推論したり、安全に仮定できる以上のことを知っている場合、この最適化属性は最も有益です。
- **緩和されたデータレイアウト**により、ホストとデバイスで共有されるデータには C++ 標準のレイアウトタイプが必要ありません。レイアウトの互換性を検証するため、デバイス・コンパイラーが必要です。
- **キュー・ショートカット**は、キュークラスに直接カーネル呼び出し関数を定義します。追加のコマンド・グループ・スコープ内で依存関係やアクセサーを作成する必要がない場合、キュー・ショートカットはコードパターンを簡素化します。
- **必要なワークグループ・サイズ**は、ユーザー主導の追加情報に基づく最適化を有効にする属性です。特定のワークグループ・サイズのカーネル呼び出しを通知する、カーネルのラムダ定義を含むカーネル適用属性を定義します。
- **データ・フロー・パイプ**は、FPGA などの空間アーキテクチャー向けアルゴリズムの記述で一般的に使用されるメカニズムにおいて、DPC++ の効率良い先入れ先出し (FIFO) 通信を有効にします。

アルゴンヌ国立研究所のリーダーシップ・コンピューティングでコンパイラー・テクノロジー & プログラミング言語を担当する Hal Finkel 氏は次のように述べています。

> 「インテルと協力して、oneAPI を定義し、内部開発とテストに使用しています。oneAPI は、統合メモリーやリダクションのサポートなど、SYCL* 1.2.1 仕様よりも拡張された機能を提供しており、これらの機能は当組織にとって不可欠なものです。例えば、DPC++/oneAPI 用の Kokkos バックエンドの開発は、これらの追加機能を使用しています。このような開発で明ら

かになった特定のニーズに対応する DPC++ の重要な新機能が含まれる、SYCL* 仕様の更新を楽しみにしています。」

Joe Curley は次のように補足しています。

> 「DPC++ コミュニティーは、わずか数カ月の間に、言語設計、アーキテクチャー、実装の面で大きな進歩を遂げました。コミュニティーの皆さん、オープン・アクセラレーション・プログラミングの取り組みにぜひ参加してください。」

DPC++ を開始するため、開発者は 2 つの方法 (インテル® DevCloud またはインテルのリファレンス実装であるインテル® oneAPI ツールキット) で言語と API を利用できます。

法務上の注意書きと最適化に関する注意事項

インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。これには、インテル® ストリーミング SIMD 拡張命令 2、インテル® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、該当する製品のユーザー・リファレンス・ガイドを参照してください。

インテルのテクノロジーを使用するには、対応したハードウェア、ソフトウェア、またはサービスの有効化が必要となる場合があります。絶対的なセキュリティーを提供できる製品またはコンポーネントはありません。

実際の費用と結果は異なる場合があります。

インテルは、サードパーティーのデータについて管理や監査を行っていません。ほかの情報も参考にして、正確かどうかを評価してください。

インテルは、明示されているか否かにかかわらず、いかなる保証もいたしません。ここにいう保証には、商品適格性、特定目的への適合性、および非侵害性の黙示の保証、ならびに履行の過程、取引の過程、または取引での使用から生じるあらゆる保証を含みますが、これらに限定されるわけではありません。

本資料で説明されている製品には、エラッタと呼ばれる設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。現在確認済みのエラッタについては、インテルまでお問い合わせください。

OpenCL および OpenCL ロゴは、Apple Inc. の商標であり、Khronos の使用許諾を受けて使用しています。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

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、www.Intel.com/PerformanceIndex/ (英語) を参照してください。