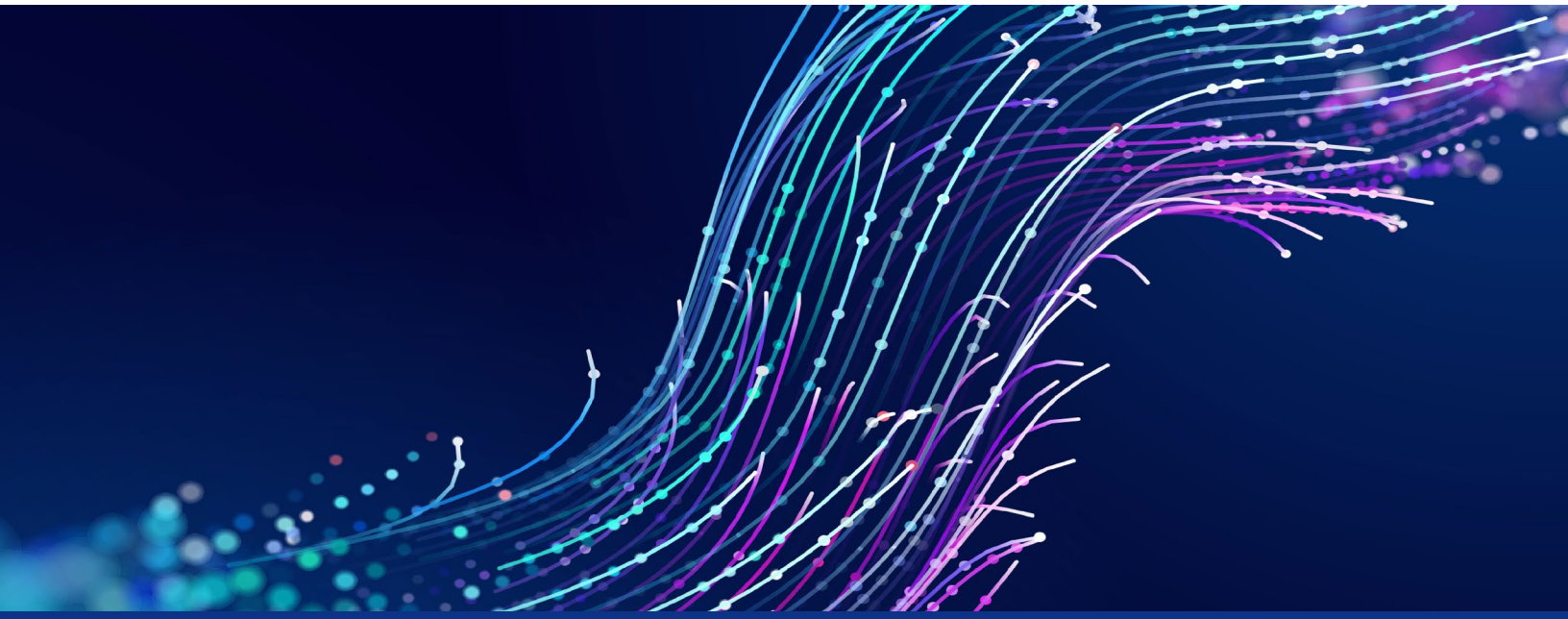

インテル® DPC++ 互換性ツールを使用した CUDA* から DPC++ への移行

独自仕様でないプログラミング言語への移行がさらに容易に

Subarnarekha Ghosal インテル コーポレーション コンパイラー・テクニカル・コンサルティング・エンジニア

将来のコンピューティング・システムはヘテロジニアスになることが明白になりつつあります。テネシー大学の **MAGMA**（英語）プロジェクトは、LAPACK に似ていますが、最新の CPU および GPU システムを含むヘテロジニアス・アーキテクチャー向けの密線形代数ライブラリーを開発しています。異なるアーキテクチャーで優れたパフォーマンスを提供するスパースソルバーのコードサンプルを探していたとき、MAGMA は有力な候補でした。この記事では、インテル® DPC++ 互換性ツール (DPCT) を使用して MAGMA の CUDA* コードをデータ並列 C++（DPC++）に移行する方法を説明します。

移行手順と必要な変更

移行する方法は 2 つあります。1 つ目の方法は、ファイルからファイルへの手動移行で、わずかなファイルを移行する場合に適しています。2 つ目の方法は、**make** または **cmake** を使用するプロジェクト向けの **.json** ファイルを作成することです。MAGMA には **makefile** が用意されているため、JSON アプローチを利用します。

入力プロジェクト・ファイルのビルドオプション（インクルード・パス、マクロ定義など）は、**intercept-build make** コマンドを使用して **.json** ファイルに格納します（Make 4.0 以降を使用していることを確認してください）。次に、**dpct** コマンドを実行します。このコマンドには、移行に役立つオプションが用意されています。MAGMA を移行するためのコマンドラインは次のとおりです。

```
dpct -p=compile_commands.json --out-root=<dpct_output_path> --in-
root=<path_to_application>
--keep-original-code --process-all
```

表 1. オプションと説明

オプション	説明
-p=compile_commands.json	compile_command.json ファイルを指定します。
--keep-original-code	互換性ツールにより変更されたコード行を確認するには、このオプションを使用します。このオプションを使用すると、移行後のファイルで、オリジナルコードの後に、インテル® DPC++ 互換性ツールにより変更されたコードが続きます。
--in-root=<path to input folder>	移行するソースツリーのルートのディレクトリー・パスを指定します。
--out-root=<path to dpct_output folder>	出力ディレクトリーのカスタムパスを指定します。
--process-all	ファイルに移行する必要がある構文や型が含まれていない場合、インテル® DPC++ 互換性ツールはファイルをスキップします。スキップされたファイルは out-root フォルダーで利用できません。 このオプションは、オリジナル・アプリケーションの out-root フォルダーの構造を複製します。
--cuda-include-path=<dir>	CUDA* ヘッダーがパスに存在しない場合、または同じパスに複数のバージョンの CUDA* ヘッダーが存在する場合は、このオプションを使用して、使用するバージョンの CUDA* のパスを指定します。

(その他のオプションは、**コマンドライン・オプション・リファレンス**（英語）を参照してください。)

次のステップは、アプリケーションで **dpct** を実行した後の出力を解釈することです。**dpct** は、コードを DPC++ 準拠にする、または構文的に正しくするため変更が必要な可能性のある場所に注釈を付けます。

```
/path/to/file.hpp:26:1: warning: DPCT1003:0: Migrated API does not return
error code. (*,0) is inserted.
You may need to rewrite this code.
// source code line for which the warning was generated
^
```

大規模なプロジェクトの場合、移行ログをファイルにリダイレクトすることを推奨します。ツールでレポートされるエラーコード / 診断の詳細は、**こちら**（英語）を参照してください。

図 1 と **図 2** は、MAGMA ライブラリーからのカーネル呼び出しの正常な移行を示しています。**--keep-original-code** オプションを使用しているため、オリジナルコードも移行後のファイルに存在します（**図 2**）。

```
{
    dim3 grid( magma_ceildiv( m, BLOCK_SIZE ) );
    magma_int_t threads = BLOCK_SIZE;
    cgeellmv_kernel_shift<<< grid, threads, 0, queue->cuda_stream() >>>
                ( m, n, nnz_per_row, alpha, lambda, dval, dcolind, dx,
                                    beta, offset, blocksize, addrows, dy );


    return MAGMA_SUCCESS;
}
```

1 **オリジナル CUDA* コード**

```
{
/* DPCT_ORIG     dim3 grid( magma_ceildiv( m, BLOCK_SIZE ) );*/
    sycl::range<3> grid(1, 1, magma_ceildiv(m, BLOCK_SIZE));
    magma_int_t threads = BLOCK_SIZE;
/* DPCT_ORIG     cgeellmv_kernel_shift<<< grid, threads, 0, queue->cuda_stream()
   >>> ( m, n, nnz_per_row, alpha, lambda, dval, dcolind, dx, beta, offset,
   blocksize, addrows, dy );*/
    /*
    DPCT1049:2224: The workgroup size passed to the SYCL kernel may exceed the
    limit. To get the device limit, query info::device::max_work_group_size.
    Adjust the workgroup size if needed.
    */
 queue->cuda_stream()->submit([&](sycl::handler &cgh) {
  cgh.parallel_for(sycl::nd_range<3>(grid * sycl::range<3>(1, 1, threads),
                                     sycl::range<3>(1, 1, threads)),
                   [=](sycl::nd_item<3> item_ct1) {
                    cgeellmv_kernel_shift(m, n, nnz_per_row, alpha, lambda,
                                          dval, dcolind, dx, beta, offset,
                                          blocksize, addrows, dy, item_ct1);
                   });
 });

    return MAGMA_SUCCESS;
}
```

2 **移行後の DPC++ コード**

ツールが移行できない関数は手動で移行する必要がありますが、その際、ツールにより生成される注釈が参考になります。

```
/*
DPCT1050:3634: The template argument of the image_accessor_ext could
not be deduced. You need to update this code.
*/
```

CUDA* コードの 1 つの行に対して複数の注釈が生成されることがあります。

```
/* DPCT_ORIG             zmgesellptmv_kernel_1_3D_tex<true><<< grid, block, 0,
   queue->cuda_stream() >>> ( m, n, num_vecs, blocksize, alignment, alpha,
              dval, dcolind, drowptr, texdx, beta, dy );*/
            /*
            DPCT1049:1518: The workgroup size passed to the SYCL kernel may
            exceed the limit. To get the device limit, query
            info::device::max_work_group_size. Adjust the workgroup size if
            needed.
            */
            /*
            DPCT1050:3634: The template argument of the image_accessor_ext could
            not be deduced. You need to update this code.
            */
```

一部の CUDA* ライブラリーには、oneAPI ライブラリー（oneMKL、oneDNN、oneVPL など）と同等の機能があります。インテル® DPC++ 互換性ツールは、多くの CUDA* ライブラリー関数を同等の oneAPI 関数に移行できます。直接移行できない関数は明示されます。多くの場合、同じ機能を手動で実装できます。例えば、CUDA* **`cusparseDcsrmv`** 関数は、oneMKL **`mkl::sparse::gemv`** 関数と **`mkl::sparse::set_csr_data`** 関数を組み合わせて手動で移行できます。

インテル® DPC++ 互換性ツールは、ユーザーからのフィードバックにより日々進化しています。ツールの既知の問題は、**リリースノート**（英語）の「既知の問題と制限事項」セクションに記載されています。

インテル® DPC++ 互換性ツールは、CUDA* アプリケーションを DPC++ に移行するために必要な時間と労力を軽減します。ツールで移行されないコードのセクションで必要な手動の作業を最小限に抑えるために、役立つ注釈と警告を提供します。MAGMA の CUDA* から DPC++ への移行は、このツールがなければ、うんざりする作業だったでしょう。

CUDA* アプリケーションの DPC++ への移行に興味があるユーザー向けに、**オンライン・トレーニング**（英語）も提供されています。インテル® DPC++ 互換性ツールは、**インテル® oneAPI ベース・ツールキット**（英語）に含まれています。

NEWS HIGHLIGHTS

ニジニ・ノヴゴロドのロバチェフスキー大学が oneAPI を使用して量子プロセスの研究を高速化

ニジニ・ノヴゴロドのロバチェフスキー大学（UNN）が、oneAPI クロスアーキテクチャー・プログラミングにより、CPU、GPU、その他のアクセラレーターを利用して現代物理学の研究を促進する新たな oneAPI 研究拠点（CoE）の開設を発表しました。新しいセンターは、ヘテロジニアス・アーキテクチャー上でハイパフォーマンス・コンピューティング（HPC）を必要とする研究課題に取り組むとともに、次世代の科学者を育成するためのソフトウェア・カリキュラムを拡充することを目的としています。

この記事の続きはこちら(英語)でご覧になれます。>