インテル® DPC++ 互換性ツールを使った CUDA* から SYCL* への移行

ハイパフォーマンス・ソフトウェア・カンファレンス 2023

エクセルソフト株式会社

ご紹介したいこと

SYCL* で実装したコード開発にあたり CUDA* → SYCL* への移行を支援するツールのご紹介

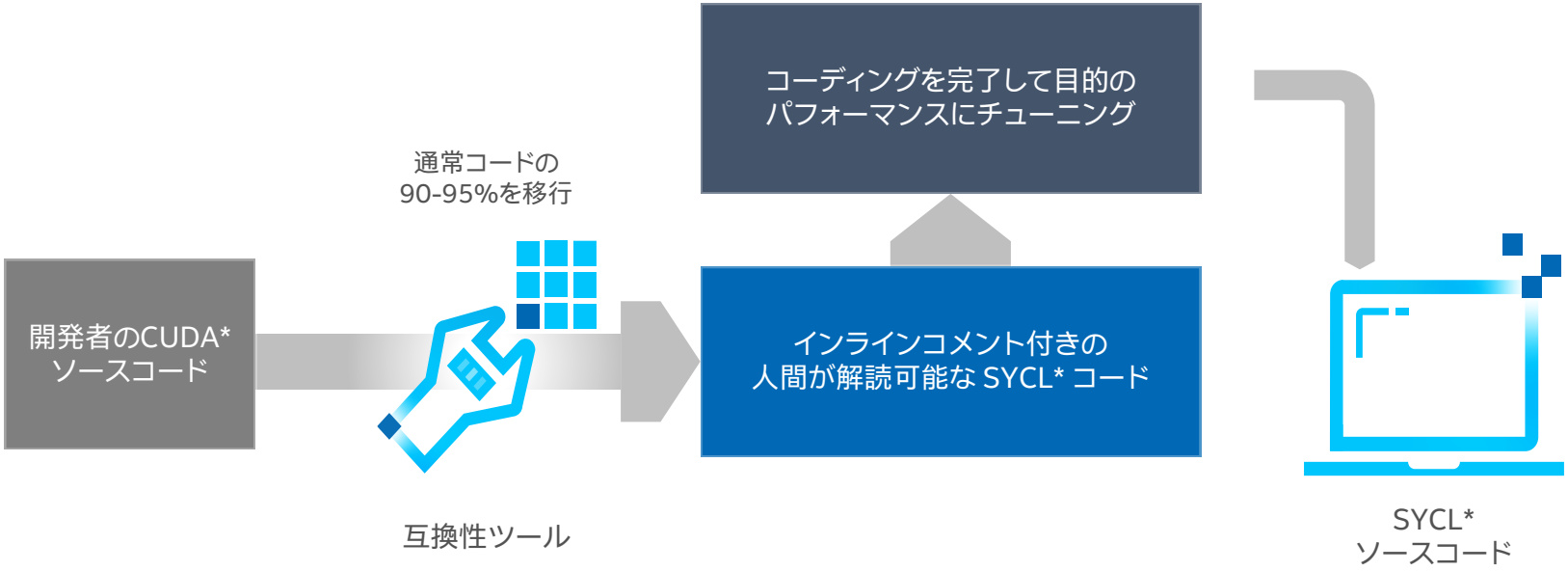
- インテル® DPC++ 互換性ツールの概要
- 移行の流れ
- ツール使用後の手続き

インテル® DPC++ 互換性ツール
CUDA* から SYCL* へのコード移行ツール

インテル® oneAPI ベース・ツールキットに含まれる

- SYCLomatic オープンソース・プロジェクトがベース
  GitHub* - oneapi-src/SYCLomatic (英語)
- 移行を支援する単独のツール
- CUDA* API を対応した SYCL* コードに変換
  +
  インラインコメントを提供

インテル® DPC++ 互換性ツール



現行 Ver 2023.1 時点

動作環境

- 対応 OS はインテル® oneAPI ツールキットと同じ
- ただし CUDA* ヘッダーファイルにアクセスが必要
  - デフォルトの参照先

```
/usr/local/cuda/include
/usr/local/cuda-x.y/include
```

CUDA* ツールキットの対応: 8.0、9.x、10.x、11.x、12.0
参照先の変更: -cuda-include-path=<path/to/cuda/include>

CUDA* → SYCL* への変換

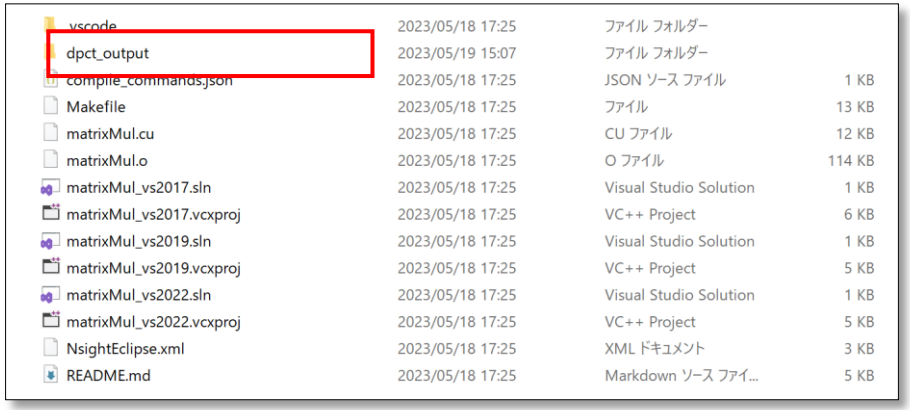
基本 : dpct <ソースファイル名>

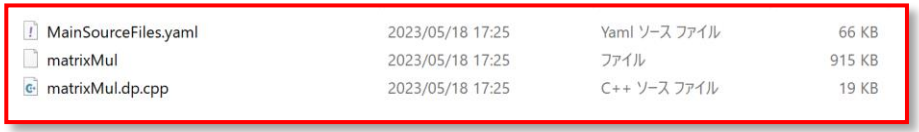
--in-root オプションにより指定ディレクトリー以下のソースファイルを対象

```
xlsoftkk@xlst2080ti:~/work/oneAPI-samples/Tools/Migration/vector-add-dpct$ dpct --in-root=. src/vector_add.cu
```

dpct_output ディレクトリーの生成と変換後のソースファイルを配置

--out-root オプションで変更可





移行後のコード

インテル® DPCT 名前空間の使用ガイド
インテル® DPC++ 互換性ツール (インテル® DPCT) デベロッパー・ガイド (isus.jp)

```
cudaMalloc(&d_A, VECTOR_SIZE*sizeof(float));
cudaMalloc(&d_B, VECTOR_SIZE*sizeof(float));
cudaMalloc(&d_C, VECTOR_SIZE*sizeof(float));

VectorAddKernel<<<1, VECTOR_SIZE>>>(d_A, d_B,
d_C);

float Result[VECTOR_SIZE] = { };

status = cudaMemcpy(Result, d_C,
VECTOR_SIZE*sizeof(float),
cudaMemcpyDeviceToHost);
if (status != cudaSuccess) {
    printf("Could not copy result to
host\n");
    exit(EXIT_FAILURE);
}

cudaFree(d_A);
cudaFree(d_B);
cudaFree(d_C);
```

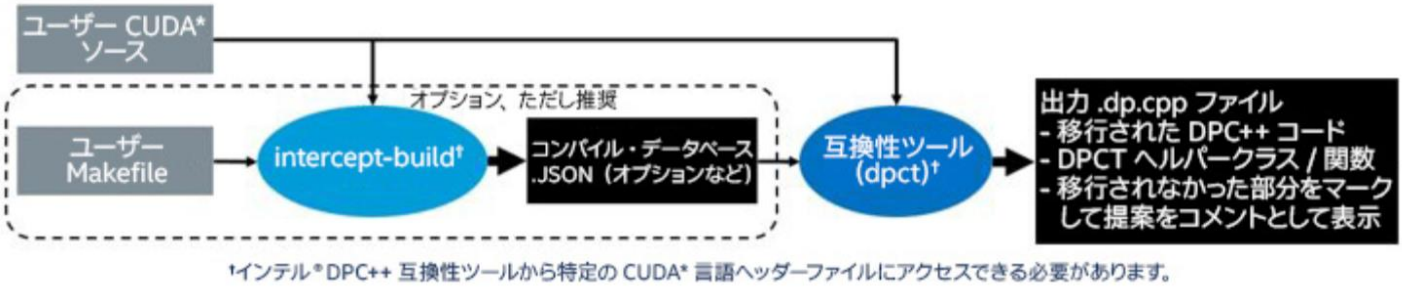
```
d_A = sycl::malloc_device<float>(VECTOR_SIZE, q_ct1);
d_B = sycl::malloc_device<float>(VECTOR_SIZE, q_ct1);
d_C = sycl::malloc_device<float>(VECTOR_SIZE, q_ct1);

q_ct1.parallel_for(sycl::nd_range<3>(sycl::range<3>(1, 1, VECTOR_SIZE),
                                     sycl::range<3>(1, 1, VECTOR_SIZE)),
                   [=](sycl::nd_item<3> item_ct1) {
                      VectorAddKernel(d_A, d_B, d_C, item_ct1);
                   });

float Result[VECTOR_SIZE] = { };

/*
DPCT1003:0: Migrated API does not return error code. (*, 0) is
inserted. You may need to rewrite this code.
*/
status = (q_ct1.memcpy(Result, d_C, VECTOR_SIZE * sizeof(float)).wait(), 0);

sycl::free(d_A, q_ct1);
sycl::free(d_B, q_ct1);
sycl::free(d_C, q_ct1);
```

一部の CUDA* API に対して移行を支援するヘルパー関数とマクロを提供します

Cmake を利用しているプロジェクトの移行

1. CMakeLists.txt から Makefile を設定して生成
2. intercept-build make コマンドの実行
3. -p=compile_commands.json を指定

```
dpct -p= compile_commands.json --in-root=.. --out-root=dpct_output
```



compile_commands.json

```
[
    {
        "command": "nvcc -c  -I../../../Common -m64 --std=c++11 -o matrixMul.o -D__CUDACC__=1 matrixMul.cu",
        "directory": "/home/xlsoftkk/Work/cuda-samples-11.8/Samples/0_Introduction/matrixMul",
        "file": "/home/xlsoftkk/Work/cuda-samples-11.8/Samples/0_Introduction/matrixMul/matrixMul.cu"
    },
    {
        "command": "nvcc  -m64 -o matrixMul matrixMul.o -D__CUDACC__=1",
        "directory": "/home/xlsoftkk/Work/cuda-samples-11.8/Samples/0_Introduction/matrixMul"
    }
]
```

表示されるコメント

- 得られる警告への対応を確認

```
/home/xlsoftkk/work/oneAPI-samples/Tools/Migration/vector-add-dpct/src/vector_add.cu:32:14: warning: DPCT1003:0: Migra
ted API does not return error code. (*, 0) is inserted. You may need to rewrite this code.
    status = cudaMemcpy(Result, d_C, VECTOR_SIZE*sizeof(float), cudaMemcpyDeviceToHost);
             ^
```

- 例: DPCT1003

  CUDA* API が返すエラーコードは SYCL* では利用できません

  SYCL* は例外を使用するため (*, 0) 演算子を挿入します

  期待したリターンコードではない場合はコードを変更します

Diagnostics Reference (英語) でコメントのリストと詳細を確認

SYCL* 移行後

- ヘテロジニアス・システムでの運用
- インテル製のハードウェア以外への対応は Codeplay* プラグインの利用を検討
- CUDA* と同等のパフォーマンスはコード最適化が必要な場合も
  - インテル® VTune™ プロファイラーや NVIDIA* Nsight などの CUDA* 向けプロファイラーも利用

```
xlsoftkk@xlst2080ti:~/Downloads/cuda-samples-11.8/Samples/0_Introduction/matrixMul/dpct_output_old$ ./matrixMul
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "Turing" with compute capability 7.5

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 1372.21 GFlop/s, Time= 0.096 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
```

```
xlsoftkk@xlst2080ti:~/Downloads/cuda-samples-11.8/Samples/0_Introduction/matrixMul/dpct_output_old$ SYCL_PI_TRACE=1 ./cpu
[Matrix Multiply Using CUDA] - Starting...
MatrixA(320,320), MatrixB(640,320)
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_opencl.so [ PluginVersion: 12.21.1 ]
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_cuda.so [ PluginVersion: 12.21.1 ]
SYCL_PI_TRACE[all]: Requested device_type: info::device_type::automatic
SYCL_PI_TRACE[all]: Requested device_type: info::device_type::automatic
SYCL_PI_TRACE[all]: Requested device_type: info::device_type::automatic
SYCL_PI_TRACE[all]: Selected device: -> final score = 1300
SYCL_PI_TRACE[all]:   platform: Intel(R) OpenCL
SYCL_PI_TRACE[all]:   device: 13th Gen Intel(R) Core(TM) i5-13400
Computing result using CUDA Kernel...
done
Performance= 259.55 GFlop/s, Time= 0.505 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
```

```
xlsoftkk@xlst2080ti:~/Downloads/cuda-samples-11.8/Samples/0_Introduction/matrixMul/dpct_output_old$ SYCL_PI_TRACE=1 ./nsycl
[Matrix Multiply Using CUDA] - Starting...
MatrixA(320,320), MatrixB(640,320)
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_opencl.so [ PluginVersion: 12.21.1 ]
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_cuda.so [ PluginVersion: 12.21.1 ]
SYCL_PI_TRACE[all]: Requested device_type: info::device_type::automatic
SYCL_PI_TRACE[all]: Requested device_type: info::device_type::automatic
SYCL_PI_TRACE[all]: Requested device_type: info::device_type::automatic
SYCL_PI_TRACE[all]: Selected device: -> final score = 1500
SYCL_PI_TRACE[all]:   platform: NVIDIA CUDA BACKEND
SYCL_PI_TRACE[all]:   device: NVIDIA GeForce RTX 2080 Ti
Computing result using CUDA Kernel...
done
Performance= 1021.58 GFlop/s, Time= 0.128 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
```

まとめ

- インテル® DPC++ 互換性ツールは新しいオープンソース・プロジェクト SYCLomatic をベースに CUDA* から SYCL* へのコード移行を支援します
- CUDA* から SYCL* への移行は思うほど難しいものではありません
- インテル、NVIDIA、AMD などの既存のアクセラレーターやプロセッサーをターゲットにしつつ将来のハードウェアへの対応を検討してください

インテル® oneAPI ツールキットの有償製品のサポートには
CUDA* → SYCL* への移行における技術サポートも含みます

参考資料

- デベロッパー・ガイドおよびリファレンス (英語)
- インテル® DPC++ 互換性ツール導入ガイド (英語)

日本語情報

- デベロッパー・ガイドおよびリファレンス
- インテル® DPC++ 互換性ツールのベスト・プラクティス

お問い合わせはこちらまで

https://www.xlsoft.com/jp/qa