ベータ版インテル® oneAPI DPC++ コンパイラー・リリースノート

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「Intel® oneAPI DPC++ Compiler Release Notes (Beta)」の日本語参考訳です。

バージョン履歴

日付	バージョン	主な変更点
2020 年 3 月	2021.1-beta05	新機能、改善、および問題の修正
2020 年 2 月	2021.1-beta04	新機能、改善、および問題の修正
2019 年 11 月	2021.1-beta03	ベータ (3) 初期公開リリース

パッケージの入手

https://software.intel.com/en-us/oneapi (英語) からツールキットをダウンロードしてインストールしてください。

2021.1-beta05

新機能

- ポインターを逆参照する際に利用されるロード・ストア・ユニット (LSU) の特性を示す FPGA SYCL* デバイスの `__builtin_intel_fpga_mem` が追加されました。
- `intelfpga::uses_global_work_offset` 属性のサポートが追加されました。
- `-fsycl-device-code-split` オプションを使用して、単一のデバイス・コード・モジュールを複数に分割するようコンパイラーに指示できるようになりました (デバイスコードの分割機能)。
- CPU と GPU (GEN9) の事前コンパイルに関する手順は文書化されています。この機能を利用するには、『インテル® oneAPI ツールキットのインストール・ガイド』 (英語) に従って "ocloc" ツールをインストールしてください。詳細については、『インテル® oneAPI DPC++ コンパイラー・デベロッパー・ガイドおよびリファレンス』 (英語) の「Ahead of Time Compilation (AOT)」セクションをご覧ください。

改善点

DPC++ コンパイラー:

- `loop_unroll` 属性で非型テンプレート・パラメーターがサポートされました。
- カーネル引数の標準レイアウト要件が緩和されました。現在、デフォルトではトリビアルにコピー構築可能で、トリビアルなデストラクターを備えています。`sycl-std=1.2.1` ドライバーオプションは、標準レイアウト要件を「on」にします。
- デバイスコードでの `__float128` 型の使用に関する診断機能が追加されました。

- `intelfpga::max_private_copies` の名前が変わり `intelfpga::private_copies` になりました。
- Windows* で `-fsycl-link` の出力オプションの動作を修正しました。
- `-fsycl` のエラー診断の重複を解決しました。
- FPGA AOT コンパイル時の依存ファイル情報の処理が改善されました。
- 追加のデバイスコードで既存の FPGA 静的ライブラリーを使用する際の診断が追加されました。

DPC++ ランタイム:

- 並列実行可能なコマンドグループの数が非常に多いアプリケーションでリーフの増加を回避するため、実行グラフでリーフ数が制限できるようになりました。
- SYCL* 仕様に準拠するため、イメージアクセサーとローカルアクセサーに `get_range()` メソッドを追加しました。
- サポートされない型のアクセサー作成を診断できるようになりました。
- ホストデバイスでは、不正な USM メモリー割り当て (huge、0 など) に対し nullptr を返します。
- コマンド完了の待機が暗黙的もしくは明示的に呼び出される場合に、実行グラフの完了したコマンドノードをクリーンアップする機能が追加されました。
- `sycl::context` 引数と `sycl::device` 引数を受け入れる 2 つの `sycl::queue` コンストラクターが追加されました。

問題の修正

DPC++ コンパイラー:

- 制限された機能 (例外など) が、未評価のコンテキストでのみ参照される関数で使用されてもエラーを生成しません。
- 統合ヘッダーでただしくない 'typename' キーワードを生成する問題を解決しました。
- SYCL* のターゲットイメージ登録関数は、OpenMP* 登録関数との競合を避けるため名称が変更されました。
- 2 ステップのコンパイル中に発生する FPGA 最適化レポートの生成エラーを修正しました。
- データ競合を引き起こす、パイプを含むカーネル・エミュレーションの問題を修正しました。
- FPGA エミュレーターのアサーション・エラーを修正しました。
- Visual Studio* IDE で生成される FPGA 最適化レポートに表示されるエラーメッセージを修正しました。
- インテル® oneAPI DPC++ コンパイラー Beta04 でコンパイルされたハードウェアでプログラムされた FPGA デバイスの自動検出文字列のエラーを修正しました。
- `-o` または `/Fo` オプションでコンパイルされた FPGA レポートフローの問題を修正しました。

DPC++ ランタイム:

- `sycl::mad_sat` のホスト実装で最終結果の飽和を修正しました。
- 非 nullptr_t の値 0x0 が、`sycl::buffer::set_final_data` メソッドに渡されるとクラッシュする問題を修正しました。
- 異なるコンテキスト間でサブバッファーをコピーする際の問題を修正しました。
- ローカルアクセサーが一時オブジェクトである場合に生じる問題を解決しました。
- 相互運用性コンストラクターを使用してメモリー・オブジェクトを構築すると、イベントが保持されない問題を修正しました。

- メモリー・オブジェクトを破棄する際の実行グラフのクリーンアップを修正しました。
- ユーザーコードで double を使用するコンパイラーの組込みコードが原因で生じる、double 型をサポートしないデバイスでのアプリケーション実行の問題を修正しました。

既知の問題

- 最新のツールキットのインストーラは更新されているため、Beta05 をインストールする前にすべての Beta04 ツールキットをアンインストールすることを推奨します。
- SUCCESS または FAILED という名前のマクロを定義すると、すでに内部で使用されているため SYCL* ヘッダーが壊れる可能性があります。
- コンパイラーによって生成されるオブジェクト・ファイル形式は、バージョン間で異なる場合があります。アプリケーションをすべてリビルドしてください。
- SYCL* ライブラリーは API/ABI の安定性を保証しないため、古いバージョンの SYCL* ライブラリーでコンパイルされたアプリケーションは新しいバージョンでは動作しない場合があります。アプリケーションをすべてリビルドしてください。
- `cl::sycl::program` API を使用して別の翻訳単位で定義されたカーネルを参照すると動作は未定義となります。
- Microsoft* Visual Studio* 2019 16.3.0 よりも古いバージョンを使用して SYCL* アプリケーションをビルドすると、リンク時に次のエラーが発生することがあります。

  `error LNK2005: "bool const std::_Is_integral<bool>" (??$_Is_integral@_N@std@@3_NB) already defined.` (エラー `LNK2005: "bool const std::_Is_integral<bool>" (??$_Is_integral@_N@std@@3_NB)` は既に定義されています。)

  Visual Studio* でこのエラーが発生する場合、-std=c++17 オプションを使用することで問題を回避できます。

- Windows* ですべての FPGA ハードウェア・コンパイルは次の警告を発生します。

  `warning LNK4221: This object file does not define any previously undefined public symbols, so it will not be used by any link operation that consumes this library.` (警告 `LNK4221: このオブジェクト ファイルは、以前に未定義であったパブリック シンボルを定義していないため、このライブラリを使用するリンク操作では使用されません。`)

  Windows* ハードウェア・コンパイルで生成されたオブジェクトの使用はまだサポートされていないため、この警告は無視してかまいません。

- FPGA 向けにコンパイルする場合、ソースファイルが作業ディレクトリーにないと、FPGA 最適化レポートにソースコードは表示されません。この問題を回避するには、ソースコードが存在するディレクトリーでコンパイルします。
- FPGA 向けにコンパイルする場合、カーネル名をローカルで宣言すると、FPGA 最適化レポート (サマリー、FMAX II レポート、システムのエリア解析、グラフビューアー (ベータ版)、カーネル・メモリー・ビューアー、およびスケジュール・ビューアー (アルファ版)) にカーネル名が `const::kernel_name` と表示されます。カーネル名をグローバルで宣言することでこの問題を回避できます。

- Microsoft* Visual Studio* で FPGA エミュレーター・コードのデバッグを行う場合、デバッガーはカーネルコードに設定されたブレークポイントで停止しません。現在この問題を回避する方法はありません。
- Windows* システムでは、FPGA エミュレーターの実行ファイルをデバッグする際に、FPGA デバイス側のコードにアクセスできません。現在この問題を回避する方法はありません。
- Red Hat* Enterprise Linux* 8 システムで GPU オフロードを行うアプリケーションは、次のエラーメッセージを出力してランタイムエラーをスローします。

  ```
  terminate called after throwing an instance of
  'cl::sycl::runtime_error' what(): No device of requested type
  available. Please check https://software.intel.com/en-
  us/articles/intel-oneapi-dpcpp-compiler-system-requirements-beta 0
  (CL_SUCCESS) ('cl::sycl::runtime_error' のインスタンスをスローした後に
  terminate が呼び出されました。要求されたタイプのデバイスがありません。
  https://software.intel.com/en-us/articles/intel-oneapi-dpcpp-compiler-
  system-requirements-beta （英語） を確認してください。)
  ```

  この問題を解決するには、環境変数 SYCL_DEVICE_WHITE_LIST="" と SYCL_DEVICE_ALLOWLIST="" を設定します。

- FPGA がインストールされている DPC++ システムは、マルチプロセス実行をサポートしていません。コンテキストを作成すると関連するデバイスがオープンされ、そのプロセスがロックを取得します。その他のプロセスはデバイスを使用できません。また、`device.get_info<>()` を介したデバイスに関する一部のクエリーは、デバイスを照会してランタイムで情報を取得する必要があるため、デバイスをオープンしてそのプロセスがロックを取得します。以下はデバイスをロックするクエリーの例です。
  - `is_endian_little`
  - `global_mem_size`
  - `local_mem_size`
  - `max_constant_buffer_size`
  - `max_mem_alloc_size`
  - `vendor`
  - `name`
  - `is_available`

2021.1-beta04

新機能

- OCL CPU ターゲットのデバイスコードをプリコンパイルする OpenCL* 事前コンパイルツール (opencl-aot) がオープンソースになりました。このツールを使用するには、次のように -fsycl-targets=spir64_x86_64-unknown-unknown-sycldevice オプションを指定して dpcpp を実行します。

  ```
  dpcpp –fsycl-targets=spir64_x86_x64-unknownunknown-sycldevice
  syclApp.cpp –o app.out
  ```

- デバイスコードで `printf` をサポートしました。

  ```
  #ifdef __SYCL_DEVICE_ONLY__
      #define CONSTANT __attribute__((opencl_constant))
  #else
      #define CONSTANT
  #endif
  ```

```
Q.submit([](sycl::handler &CGH) {
    CGH.single_task([=]() {
        static const CONSTANT char Format[] = "%f\n";
        sycl::intel::experimental::printf(Format, 33.4f);
    });
});
```

- デバイスコードで、現在 `assert`、`std::complex`、および数学関数 (`std::cos`、`std::sin` など) をサポートするデバイス標準 C/C++ ライブラリー基盤を導入しました。
    - 関数宣言は標準ヘッダー (<assert.h> や <cassert> など) から取得され、対応するヘッダーをユーザーコードで明示的にインクルードする必要があります。

```
sycl::buffer<std::complex<float>, 1> Buf(Data, Size);
Q.submit([&](sycl::handler &CGH) {
    auto Acc = Buf.get_access<sycl_read_write>(CGH);
    CGH.single_task<class ComplexKernel>( [=]() {
        Acc[0] = std::sqrt(Acc[0]);
        assert(Acc0[0] == 1 && "Invalid value");
    });
});
```

現在の使用法では、特別なデバイス・ライブラリーを DPC++ プログラムにリンクする必要があります。ライブラリーは、プログラムのコンパイルに使用される C/C++ 標準ライブラリーと一致している必要があります。

Linux* で GNU* glibc を使用する場合:

```
dpcpp -c main.cpp -o main.o dpcpp main.o $(SYCL_INSTALL)/lib/libsycl-glibc.o -o a.out
```

Windows* の場合:

```
dpcpp -c main.cpp -o main.obj dpcpp
main.obj %SYCL_INSTALL%/lib/libsycl-msvc.o -o a.exe
```

- コンパイラー・ドライバーと IDE 統合プラグインでパフォーマンス・ライブラリーをサポートしました。
- インテル® FPGA デバイス向けの新しい属性が追加されました: `num_simd_work_items`、`bank_bits`、`max_work_group_size`、`max_global_work_dim`。

改善点

DPC++ コンパイラー:

- `-fsycl-unnamed-lambda` がデフォルトのコンパイラー・オプションになりました。
- `intelfpga::ivdep`、`intelfpga::ii`、および `intelfpga::max_concurrency` 属性の引数として整数テンプレート・パラメーターを使用できるようになりました。
- ファット静的ライブラリーで FPGA デバイス・ライブラリーをサポートしました。
- AOT で誤ったトリプルを使用した際の診断が改善されました。
- ラムダ関数に `cl::reqd_work_group_size` 属性と `cl::intel_reqd_sub_group_size` 属性を適用できるようになりました。
- `-Wno-error=sycl-strict` オプションを指定する場合、生ポインター引数または戻り値を持つ関数に SYCL_EXTERNAL を適用できるようになりました。
- `-fsycl-device-only` のエイリアスである `--sycl` オプションと SYCL* の仮想型の制限が排除されました。

DPC++ ランタイム:

- より「フラット」なカーネル転送 `cl::sycl::queue` メソッドが実装されました。USM ポインターと一緒に使用すると便利な queue メソッドを利用して、コマンドをキューに登録できます。

```
sycl::queue Q;
int *Array = (int *)sycl::malloc_device(N * sizeof(int), Q);
Q.single_task({e3}, [=]() {
for (int I = 0; I < N; ++I)
    Array[i]++;
});
```

- `queue::submit` は同期例外をスローするようになりました。
- デバイス許可リストでプラットフォーム名とプラットフォーム・バージョンをサポートしました。
- カーネルとプログラムのキャッシュがスレッドセーフになりました。
- SYCL* API を使用する際に cl 名前空間指定子を省略できるようになりました。`cl::sycl::buffer` の代わりに `sycl::buffer` を使用できます。
- ホスト・アクセサー・デストラクターの後に、保留中のタスクをキューに投入します。
- 次の診断が改善されました。
  - 無効なサブバッファーの作成
  - 単一コマンドグループでの複数アクションの実行
  - `cl::sycl::vec` オブジェクトを SYCL* ビルトインに引き渡す際の次元の不一致
- `cl::sycl::pipe` クラスは `cl::sycl::intel` 名前空間に移動され、デバイスが SYCL_INTEL_data_flow_pipes 拡張をサポートするかどうかを示す、新しいクエリー `info::device::kernel_kernel_pipe_support` が追加されました。
- `handler::copy` で任意の次元のアクセサーをコピーできるようになりました。
- ビルドオプションなしで、`cl::sycl::program::build_with_kernel_type` を介して作成されたプログラムのコンパイルとカーネル作成の結果をキャッシュできます。
- `cl::sycl::ordered_queue::single_task` に `single_task` と `parallel_for methods` メソッドが追加されました。これらは、カーネルを起動する代替方法を提供します。
- キュー参照による割り当てを行う USM 関数形式を追加しました。
- 以下の C++ 推論ガイドが追加されました。
  - `cl::sycl::range`
  - `cl::sycl::id`
  - `cl::sycl::nd_range`
  - `cl::sycl::vec`
  - `cl::sycl::multi_ptr`
  - `cl::sycl::buffer`
- 連続したコンテナーを引数として受け取る、新しいバッファー・コンストラクターが追加されました。
- `cl::sycl::intel::sub_group` クラスのメソッドをロードおよびストアする 1 バイト型がサポートされました。
- カーネルのキュー投入プロセスのエラーレポートが改良されました。
- ホストアクセサーを破棄した後も、メモリー・オブジェクトのインスタンスがホスト上で保持されるようになりました。
- SYCL* 実装から参照できるデバイスを指定する、SYCL_DEVICE_WHITE_LIST コントロールがサポートされました。

問題の修正

DPC++ コンパイラー:

- SYCL* カーネルに複数の属性をアタッチできない問題を修正しました。
- オブジェクトのような拡張子を持たないファイルをアンバンドルするようにドライバーを変更しました。
- `-fsycl` と `-lstdc++` が同時に指定された場合に、アボートする問題を修正しました。
- ホストとデバイスのコンパイルで __cplusplus の違いに対処するため、デバイスのデフォルトコンパイル設定 `-std=c++` が削除されました。
- 同一のシグネチャーを持つ 2 つの異なるラムダ関数が、同じマングル名になる問題を修正しました。
- デバイスのコンパイルで有効なオブジェクトが提供されない場合のデバイスリンク手順を削除しました。
- Windows* 上の FPGA ターゲットの AOT コンパイルの問題を修正しました。
- FPGA AOCX デバイスアーカイブを使用する際の FPGA AOT コンパイルの問題を修正しました。
- 同一ディレクトリーで 2 つのコンパイルを実行した際に生成される FPGA エミュレーターのエラーを修正しました。
- 無限ループでのループ属性に関連する FPGA 固有の問題を修正しました。

DPC++ ランタイム:

- 複数のスレッドがホストアクセサーを作成する際に、デッドロックが発生する可能性がある問題を修正しました。
- SYCL* ライブラリーと SYCL* アプリケーションが異なるバージョンの標準ヘッダーでコンパイルされた場合に生じる互換性の問題を修正しました。
- `-fsycl-unnamed-lambda` オプションが指定された場合の `handler::copy` のコンパイルエラーの問題を修正しました。
- ホストデバイスでローカルサイズがゼロのカーネルを送信するとクラッシュする問題を修正しました。
- SYCL* 仕様に合わせて `vec::load` と `vec::store` メンバー関数がテンプレート化されました。
- SYCL* メモリー・オブジェクトのデストラクターのコピーバック・ルーチンでスローされる例外が非同期になりました。
- 一部の状況でサブバッファーへのホストアクセサー作成でクラッシュする問題を修正しました。
- LLVM とそれにリンクされるアプリケーション/テストの RuntimeLibrary 値の不一致が原因で Windows* でビルドエラーが発生する問題を修正しました。
- 単一要素のスウィズルからスカラーに変換する際の問題の回避策を実装しました。
- 一部の状況で `sycl::stream` が複数のワーク項目の出力を混在させる UX の問題を修正しました。
- `-foffload-static-lib` オプションを指定すると、リンカーではなく、リンカ―オプションがバンドラーに渡される問題を修正しました。
- `cl::sycl::handler::copy` 内部で正しくないオフセットが使用される問題を修正しました。
- `cl::sycl::accessor` クラスの `get_count` メソッドと `get_size` メソッドが、作成された非ローカルアクセサー範囲に不正な値を返す問題を修正しました。
- 非同期例外がリターンしたイベントに関連付けられない問題を修正しました。
- ホストアクセサーがメモリー・オブジェクトへの排他アクセスを提供しないことがある問題を修正しました。
- `cl::sycl::property::buffer::use_host_ptr` プロパティーが無視される問題を修正しました。

既知の問題

- オブジェクト・ファイルのサイズは、オープンソースの LLVM におけるファット・オブジェクトのレイアウト変更により、古いバージョンでコンパイルされたオブジェクトに比べ増加しています。
- `cl::sycl::program` API を使用して別の翻訳単位で定義されたカーネルを参照すると動作は未定義となります。
- Visual Studio* 2019 16.3.0 よりも古いバージョンを使用して SYCL* アプリケーションをビルドすると、リンク時に次のエラーが発生することがあります。

  `error LNK2005: "bool const std::_Is_integral<bool>" (??$_Is_integral@_N@std@@3_NB) already defined.` (エラー `LNK2005: "bool const std::_Is_integral<bool>" (??$_Is_integral@_N@std@@3_NB)` は既に定義されています。)

  Visual Studio* でこのエラーが発生する場合、-std=c++17 オプションを使用することで問題を回避できます。

- FPGA がインストールされている DPC++ システムは、マルチプロセス実行をサポートしていません。コンテキストを作成すると関連するデバイスがオープンされ、そのプロセスがロックを取得します。その他のプロセスはデバイスを使用できません。また、`device.get_info<>()` を介したデバイスに関する一部のクエリーは、デバイスを照会してランタイムで情報を取得する必要があるため、デバイスをオープンしてそのプロセスがロックを取得します。以下はデバイスをロックするクエリーの例です。
  - `is_endian_little`
  - `global_mem_size`
  - `local_mem_size`
  - `max_constant_buffer_size`
  - `max_mem_alloc_size`
  - `vendor`
  - `name`
  - `is_available`
- Windows* ですべての FPGA ハードウェア・コンパイルは次の警告を発生します。

  `warning LNK4221: This object file does not define any previously undefined public symbols, so it will not be used by any link operation that consumes this library.` (警告 `LNK4221:` このオブジェクト ファイルは、以前に未定義であったパブリック シンボルを定義していないため、このライブラリを使用するリンク操作では使用されません。)

  Windows* ハードウェア・コンパイルで生成されたオブジェクトの使用はまだサポートされていないため、この警告は無視してかまいません。

- FPGA 向けにコンパイルする場合、ソースファイルが作業ディレクトリーにないと、FPGA 最適化レポートにソースコードは表示されません。この問題を回避するには、ソースコードが存在するディレクトリーでコンパイルします。
- FPGA 向けにコンパイルする場合、カーネル名をローカルで宣言すると、FPGA 最適化レポート (サマリー、FMAX II レポート、システムのエリア解析、グラフビューアー (ベータ版)、カーネル・メモリー・ビューアー、およびスケジュール・ビューアー (アルファ版)) にカーネル名が `const::kernel_name` と表示されます。カーネル名をグローバルで宣言することでこの問題を回避できます。

- FPGA エミュレーターをターゲットにする場合、コンパイラーが次のエラーを生成することがあります。

  `Assertion failed: DT.getRoots().size() == 1 "Only one entry block for post domfronts is expected!".` (アサートに失敗しました: `DT.getRoots().size() == 1 "domfront` 後のエントリーブロックは `1` つだけです。")

  この問題を回避するには、カーネル C++ コードの switch 文を if/else 文に置き換えます。

- Visual Studio* IDE から FPGA 最適化レポートを作成すると、コンパイル時に次のメッセージが表示されます:「Warning! Your browser is not supported to view the report (警告! レポートの表示をサポートしないブラウザーです)」。このメッセージを無視して、`project_directory/reports/report.html` ファイルを表示できます。
- Windows* システムでは、FPGA レポートフローは `-o` または `/Fo` オプションを無視します。`-o` オプションを使用すると (例えば、`dpcpp-cl -fintelfpga -Xshardware -fsycl-link -o foo.a bar.cpp`)、生成される出力は bar-x86-64.a.W となります。`/Fo` オプションを使用すると (例えば、`dpcpp-cl -fintelfpga -Xshardware -fsycl-link /Fo foo.a bar.cpp`)、コンパイラーは「error: no such directory 'foo.a' (エラー: 'foo.a' ディレクトリーはありません)」というエラーを出力します。この問題により、Windows* でメモリー属性 FPGA サンプルコードが失敗する可能性があります。回避策として、サンプルコードのコンパイルで `-j1` オプションを `ninja` コマンドに渡します。
- Microsoft* Visual Studio* で FPGA エミュレーター・コードのデバッグを行う場合、デバッガーはカーネルコードに設定されたブレークポイントで停止しません。現在この問題を回避する方法はありません。
- FPGA デバイスがインテル® oneAPI DPC++ コンパイラー Beta04 でコンパイルされたハードウェアで構成されている場合、デバイスで Beta03 を使用しようとすると、次のエラーが生じることがあります。

  「`failed to read auto discovery string at byte 2: expected version is 20, found 22 error: the currently programmed/flashed;design is no longer supported in this release. Please recompile the design with the present version of the sdk and re program/flash the board.` (バイト `2` で自動文字列検出の読み取りに失敗しました: 期待されるバージョンは `20`、検出されたエラー `22`: 現在フラッシュまたはプログラムされた設計は、このリリースではサポートされていません。現行バージョンの `SDK` で設計を再コンパイルして、ボードを再プログラム/フラッシュしてください)」

  この問題を解決するには、`aocl program $AOCL_BOARD_PACKAGE_ROOT/bringup/pac_a10/default.aocx` を実行します。

- CPU オフロードと FPGA エミュレーションのデバッグ機能は、Beta04 では正しく動作しません。CPU オフロードと FPGA エミュレーションを利用する開発者は、Beta04 へアップグレードせず、Beta03 を使用することを推奨します。この問題は次のベータ版リリースで修正される予定です。
- デフォルトのセレクターを使用する DPC++ アプリケーションでは、GPU なしの Red Hat* Enterprise Linux* 8.1 と SUSE* 15 システムでランタイムエラーが発生します。この問題を回避するには、デフォルトセレクターの代わりに CPU セレクターを使用します。
- Windows* と Linux* 上では、GPU をターゲットとする事前コンパイル (AOT) は、ocloc ツールがないためこのリリースでは動作しません。これは、次のベータ版リリースで対処される予定です。

- Gen11 プロセッサー・グラフィックスを含む、一部のデスクトップとラップトップ GPU では、倍精度データ型はサポートされません。さらに、コンパイラーの問題により、Gen11 プロセッサー・グラフィックスではコードが実行できず、コードに倍精度データ型が含まれていない場合でも倍精度データ型に関するエラーが出力されます。

2021.1-beta03

このリリースの新機能 (英語)

法務上の注意書きと最適化に関する注意事項

本資料は、(明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず) いかなる知的財産権のライセンスも許諾するものではありません。

本資料には、開発中の製品、サービス、およびプロセスについての情報が含まれています。ここに記載されているすべての情報は、予告なく変更されることがあります。インテルの最新の製品仕様およびロードマップをご希望の方は、インテルの担当者までお問い合わせください。

インテル® テクノロジーの機能と利点はシステム構成によって異なり、対応するハードウェアやソフトウェア、またはサービスの有効化が必要となる場合があります。実際の性能はシステム構成によって異なります。絶対的なセキュリティーを提供できるコンピューター・システムはありません。詳細については、各システムメーカーまたは販売店にお問い合わせいただくか、http://www.intel.co.jp/ を参照してください。

本資料で説明されている製品およびサービスには、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。

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。

Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation またはその子会社の商標です。

Microsoft および Windows は、米国 Microsoft Corporation の、米国およびその他の国における登録商標または商標です。

OpenCL および OpenCL ロゴは、Apple Inc. の商標であり、Khronos の許諾を得て使用しています。

* 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。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。