インテル® oneAPI マス・カーネル・ライブラリー (インテル® oneMKL)導入ガイド

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「Get Started with Intel® oneAPI Math Kernel Library」の日本語参考訳です。

バージョン 2021.1(最終更新日:2020 年 12 月 4 日)

インテル® oneAPI マス・カーネル・ライブラリー(インテル® oneMKL)は、CPU および GPU 向けに高度に最適化され、広範囲に並列化されたルーチンからなる数学計算ライブラリーにより、最高のパフォーマンスを実現するのを支援します。CPU 上のほとんどのルーチンには C と Fortran インターフェイスが用意されており、CPU と GPU 上の一部のルーチンでは DPC++ インターフェイスが用意されています。さまざまなインターフェイスで数学演算の包括的なサポートを提供しています。

CPU 上の C および Fortran

- 線形代数
- 高速フーリエ変換(FFT)
- ベクトル演算
- 直接法および反復法スパースソルバー
- 乱数ジェネレーター

CPU および GPU 上の DPC++(詳細は、『インテル® oneAPI マス・カーネル・ライブラリー - データ並列 C++ デベロッパー・リファレンス』(英語)を参照)

- 線形代数
  - BLAS
  - 一部のスパース BLAS 機能
  - 一部の LAPACK 機能
- 高速フーリエ変換(FFT)
  - 1D r2c FFT
  - 1D c2c FFT
- 乱数ジェネレーター
  - 単精度の一様分布、ガウス分布、対数正規分布
- 一部のベクトル演算機能

はじめに

既知の問題と最新情報は、「リリースノート」(英語)を参照してください。

動作環境は、「インテル® oneAPI マス・カーネル・ライブラリー の動作環境」(英語)を参照してください。

DPC++ コンパイラーの動作環境は、「インテル® oneAPI DPC++/C++ コンパイラー導入ガイド」(英語)を参照してください。

ステップ 1:インテル® oneMKL のインストール

インテル® oneAPI ベース・ツールキットからインテル® oneMKL(英語)をダウンロードします。

ステップ 2:関数またはルーチンの選択

問題に最適なインテル® oneMKL の関数またはルーチンを選択します。次のリソースを利用できます。

リソースのリンク	説明
インテル® oneMKL デベロッパー・ガイド Linux* 版(英語) Windows* 版(英語) macOS* 版(英語)	デベロッパー・ガイドには、次のようなトピックに関する詳細な情報があります。 • アプリケーションのコンパイルとリンク • カスタム DLL のビルド • スレッド化 • メモリー管理
インテル® oneMKL デベロッパー・リファレンス C 言語(英語) Fortran 言語(英語) DPC++ 言語(英語)	デベロッパー・リファレンス(C、Fortran、および DPC++ 版)には、すべてのライブラリー・ドメインの関数とインターフェイスの詳細な説明があります。
インテル® oneMKL 関数検索アドバイザー(英語)	LAPACK 関数検索アドバイザーを使用して、特定の問題に役立つ LAPACK ルーチンを調査できます。例えば、次のような条件で検索できます。 • Routine type(ルーチンタイプ:Computational(計算) • Computational problem(計算問題):Orthogonal factorization(直交因数分解) • Matrix type(行列タイプ):General(一般) • Operation(操作):Perform QR factorization(QR 分解を実行)

ステップ 3:コードのリンク

インテル® oneMKL リンク行アドバイザー(英語)を使用して、プログラムの特徴に応じてリンクコマンドを設定します。

いくつかの制限と追加の要件:

DPC++ 向けインテル® oneAPI MKL は、mkl_intel_ilp64 インターフェイス・ライブラリーとシーケンシャルまたは TBB スレッドのみをサポートしています。

Linux* 上で静的リンクを使用する DPC++ インターフェイス

```
dpcpp -fsycl-device-code-split=per_kernel -DMKL_ILP64 <typical user includes
and linking flags and other libs> ${MKLROOT}/lib/intel64/libmkl_sycl.a -Wl,
--start-group ${MKLROOT}/lib/intel64/libmkl_intel_ilp64.a
${MKLROOT}/lib/intel64/libmkl_<sequential|tbb_thread>.a
${MKLROOT}/lib/intel64/libmkl_core.a -Wl,--end-group -lsycl -lOpenCL
-lpthread -ldl -lm
```

例えば、ILP64 インターフェイスと TBB スレッドを使用する main.cpp をビルドして静的にリンクする場合:

```
dpcpp -fsycl-device-code-split=per_kernel -DMKL_ILP64 -I${MKLROOT}/include
main.cpp ${MKLROOT}/lib/intel64/libmkl_sycl.a -Wl,--start-group
${MKLROOT}/lib/intel64/libmkl_intel_ilp64.a
${MKLROOT}/lib/intel64/libmkl_tbb_thread.a
${MKLROOT}/lib/intel64/libmkl_core.a -Wl,--end-group -
L${TBBROOT}/lib/intel64/gcc4.8 -ltbb -lsycl -lOpenCL -lpthread -lm -ldl
```

Linux* 上で動的リンクを使用する DPC++ インターフェイス

```
dpcpp -DMKL_ILP64 <typical user includes and linking flags and other libs>
-L${MKLROOT}/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -
lmkl_<sequential|tbb_thread> -lmkl_core -lsycl -lOpenCL -lpthread -ldl -lm
```

例えば、ILP64 インターフェイスと TBB スレッドを使用する main.cpp をビルドして動的にリンクする場合:

```
dpcpp  -DMKL_ILP64 -I${MKLROOT}/include main.cpp -L${MKLROOT}/lib/intel64
-lmkl_sycl -lmkl_intel_ilp64 -lmkl_tbb_thread -lmkl_core -lsycl -lOpenCL
-ltbb -lpthread -ldl -lm
```

Windows* 上で静的リンクを使用する DPC++ インターフェイス

```
dpcpp -fsycl-device-code-split=per_kernel -DMKL_ILP64 <typical user includes
and linking flags and other libs> "%MKLROOT%"\lib\intel64\mkl_sycl.lib
mkl_intel_ilp64.lib mkl_<sequential|tbb_thread>.lib mkl_core_lib sycl.lib
OpenCL.lib
```

例えば、ILP64 インターフェイスと TBB スレッドを使用する main.cpp をビルドして静的にリンクする場合:

```
dpcpp  -fsycl-device-code-split=per_kernel -DMKL_ILP64 -I"%MKLROOT%\include"
main.cpp"%MKLROOT%"\lib\intel64\mkl_sycl.lib  mkl_intel_ilp64.lib
mkl_tbb_thread.lib mkl_core.lib sycl.lib OpenCL.lib tbb.lib
```

Windows* 上で動的リンクを使用する DPC++ インターフェイス

```
dpcpp -DMKL_ILP64 <typical user includes and linking flags and other libs>
"%MKLROOT%"\lib\intel64\mkl_sycl_dll.lib mkl_intel_ilp64_dll.lib
mkl_<sequential|tbb_thread>_dll.lib mkl_core_dll.lib tbb.lib sycl.lib
OpenCL.lib
```

例えば、ILP64 インターフェイスと TBB スレッドを使用する main.cpp をビルドして動的にリンクする場合:

```
dpcpp  -fsycl-device-code-split=per_kernel -DMKL_ILP64 -I"%MKLROOT%\include"
main.cpp "%MKLROOT%"\lib\intel64\mkl_sycl_dll.lib mkl_intel_ilp64_dll.lib
mkl_tbb_thread_dll.lib mkl_core_dll.lib tbb.lib sycl.lib OpenCL.lib
```

OpenMP* オフロードをサポートする C/Fortran インターフェイス

インテル® oneMKL の C/Fortran インターフェイスを使用して GPU への OpenMP* オフロード機能を利用できます。

詳細は、「GPU への OpenMP* オフロード導入ガイド」(英語)を参照してください。

次の変更をインテル® oneMKL の C/Fortran コンパイル/リンク行に適用して GPU への OpenMP* 機能を有効にします。

- 追加のコンパイル/リンクオプション:`-fiopenmp -fopenmp-targets=spir64 -mllvm -vpo-paropt-use-raw-dev-ptr -fsycl`
- 追加のインテル® oneMKL ライブラリー:インテル® oneMKL DPC++ ライブラリー

例えば、Linux* 上で ilp64 インターフェイスと OpenMP* スレッドを使用する main.cpp をビルドして静的にリンクする場合:

```
icx -fiopenmp -fopenmp-targets=spir64 -mllvm -vpo-paropt-use-raw-dev-ptr
-fsycl -DMKL_ILP64 -m64 -I$ (MKLROOT)/include main.cpp
L${MKLROOT}/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_intel_thread
-lmkl_core -liomp5 -lsycl -lOpenCL -lstdc++ -lpthread -lm -ldl
```

その他のサポートされるすべての設定は、「インテル® oneMKL リンク行アドバイザー」(英語)を参照してください。

関連情報

リソース	説明
チュートリアル: インテル® oneMKL を使用した行列乗算 • C 言語(英語) • Fortran 言語(英語)	このチュートリアルでは、インテル® oneMKL を使用して行列乗算を行い、行列乗算のパフォーマンスを測定して、スレッドを制御する方法を説明します。
インテル® oneMKL リリースノート(英語)	リリースノートには、新機能と変更された機能を含む、最新バージョンのインテル® oneMKL に関する情報が記載されています。リリースに関連する主要なオンラインリソースへのリンクも含まれています。次の情報が得られます。 • 新機能 • 製品の内容 • テクニカルサポートの利用方法 • ライセンス定義
インテル® oneMKL (英語)	インテル® oneMKL 製品ページサポートとオンライン・ドキュメントに関する情報を入手できます。

法務上の注意書きと最適化に関する注意事項

性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサー用に最適化されていることがあります。SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。さらに詳しい情報をお知りになりたい場合は、http://www.intel.com/benchmarks/(英語)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。

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。

注意事項の改訂 #20110804

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。

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。

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