ホワイトペーパー

インテル® Xeon Phi™ コプロセッサーで インテル® MKL 自動オフロードを利用する

はじめに

インテル® マス・カーネル・ライブラリー (インテル® MKL) に新たに追加された自動オフロード (AO) 機能により、ユーザーコードで呼び出される計算集約型のインテル® MKL 関数は、インテル® Xeon Phi™ コプロセッサーを自動かつ透過的に利用することができます。コードでインテル® MKL 呼び出しを変更したり、コンパイルやリンク方法を変更しなくても、コプロセッサーによって提供される計算リソースを活用できます。プログラマーは、馴染みのあるプログラミング・モデルにより、マルチコアからメニーコアへとパフォーマンスのスケーリングを達成できます。

コプロセッサーは PCI Express (PCIe) を介してホストシステムに接続されるため、自動オフロードは十分な大きさの問題を持つ関数と、データアクセスに比べて計算の割合が大きい関数でのみサポートされます。例えば、多くの LAPACK 関数やレベル 3 BLAS 関数のように、計算の複雑さが $O(n^3)$ よりも大きく、データアクセスが $O(n^2)$ のインテル® MKL 関数の場合、計算時間が非常に大きくデータ転送にかかる時間を償却できるため、問題全体の計算時間を短縮しパフォーマンスを向上できます。自動オフロードには、このような特性を持つ関数が適しています。インテル® MKL 11.1 では、次のレベル 3 BLAS 関数と LAPACK 関数が自動オフロードに対応しています。

- ?GEMM、?SYMM、?TRMM、および ?TRSM
- LU、QR、コレスキー分解

自動オフロードの有効化/無効化

自動オフロードを有効にするには、サポート関数を呼び出すか、または環境変数を設定します。コンパイラー・プラグマは不要です。また、通常どおりコンパイルとリンクを行うことができます。

FORTRAN あるいは C コードで、サポート関数を呼び出して自動オフロードを有効にする場合は、以下のように記述します。

```
rc = mkl_mic_enable( )
```

環境変数を設定する場合は、以下のように指定します。

```
MKL_MIC_ENABLE=1
```

自動オフロードが有効になると、必要に応じてインテル® MKL は 1 つ以上のインテル® Xeon Phi™ コプロセッサーに自動的に計算をオフロードします。

計算のオフロードは透過的に行われ、インテル® MKL ランタイムがオフロードする処理量を決定します。問題サイズとコプロセッサーの現在の状態に応じて、計算のすべてまたは一部がコプロセッサーで実行されるか、あるいはコプロセッサーでは全く実行されません。同様に、インテル® MKL は、すべてのコプロセッサーを利用するか、あるいは 1 つだけを利用するかを決定します。オフロードする処理量をプログラマーが制御できるように、インテル® MKL では、ホストとコプロセッサー間の処理量の配分を細かく調整するメカニズムを提供しています。詳細は、後述の「作業分割の制御方法」を参照してください。

コプロセッサーが利用不可の場合も同じ実行ファイルを使用できる、という点でもオフロードは透過的です。その場合、すべての計算はホストで通常どおり行われ、ペナルティーは発生しません。

自動オフロードを無効にするには、サポート関数 `mkl_mic_disable()` の呼び出しを挿入するだけです。再度有効にするには、`mkl_mic_enable()` を呼び出します。

作業分割の制御方法

サポートしているレベル 3 BLAS 関数 (?GEMM、?SYMM、?TRMM、および ?TRSM) に対して、インテル® MKL は、インテル® MKL ランタイムによって決定されるデフォルトの作業分割をオーバーライドし、現在の作業分割設定と利用可能なコプロセッサーの数を照会するサポート関数と環境変数を提供します。各サポート関数の詳細は、リファレンス・マニュアルを参照してください。これらの作業分割制御は、LAPACK 関数では動作が異なります。LAPACK 関数は、指定された値を使用しません。作業の割合に非ゼロ値を設定すると、コプロセッサーで自動オフロードモード用が有効になるだけです。

次の表に、ホストとコプロセッサー間の作業分割の設定と制御方法の例を示します。

例	注
mkl_mic_set_workdivision (MKL_TARGET_MIC, -1, MKL_MIC_AUTO_WORKDIVISION)	ランタイムシステムがオフロードする処理量と使用するカードを決定します。
mkl_mic_set_workdivision (MKL_TARGET_MIC, -1, 0.5)	計算の 50% をオフロードし、ランタイムシステムが使用するカードを決定します。
mkl_mic_set_workdivision (MKL_TARGET_MIC, 0, 0.5)	計算の 50% を最初のカード (カード 0) にオフロードします。
mkl_mic_set_workdivision (MKL_TARGET_HOST, 0, 0.5)	計算の 50% をホストで行い、残りをすべてのカードにオフロードします。(第 2 引数は無視されます)
mkl_mic_get_workdivision (MKL_TARGET_MIC, 0, &wd)	最初のカード (カード 0) に配分された作業量を照会します。
mkl_mic_get_device_count ( )	システムで利用可能なカードの数を照会します。

作業分割は、環境変数で制御することもできます。次の表に例を示します。サポート関数のほうが常に環境変数よりも優先されることに注意してください。

例	注
MKL_MIC_WORKDIVISION=MIC_AUTO_WORKDIVISION	ランタイムシステムがオフロードする処理量と使用するカードを決定します。
MKL_MIC_WORKDIVISION=0.5	計算の 50% をオフロードし、ランタイムシステムが使用するカードを決定します。
MKL_MIC0_WORKDIVISION=0.5	計算の 50% を最初のカード (カード 0) にオフロードします。
MKL_HOST_WORKDIVISION=0.5	計算の 50% をホストで行い、残りをすべてのカードにオフロードします。

自動オフロードの使用に関するヒント

自動オフロードをよく理解し、最大限に活用するためのコーディングのヒントを紹介します。

自動オフロードは適切な行列サイズで最適に動作する

自動オフロードで良いパフォーマンスを得るには、行列サイズが重要です。実際、行列サイズが小さすぎると自動オフロードは行われません。これは、データ転送のオーバーヘッドのほうが、オフロードによって得られるパフォーマンスよりも大きいためです。行列が十分に大きい場合、一般に正方行列で最高のパフォーマンスがもたらされます。自動オフロードに対応した各インテル® MKL 関数の行列サイズのしきい値に関する最新情報は、ナレッジベースの記事を参照してください。

自動オフロードで使用する最大メモリー量をコプロセッサーで予約する

インテル® MKL は、自動オフロード計算を実行する各プロセスに追加のメモリーを割り当てます。バッファーの初期化とデータ転送のオーバーヘッドを軽減するため、プログラマーは自動オフロードで使用するコプロセッサーのメモリーを制限することができます。サポート関数 `mkl_mic_set_max_memory()` を呼び出すか、環境変数 `MKL_MIC_MAX_MEMORY` を設定します。例えば、次の呼び出しは最初のコプロセッサーのメモリーを 4GB に制限します。

```
rc = mkl_mic_set_max_memory(MKL_TARGET_MIC, 0, 4G)
```

関数 mkl_mic_free_memory() を呼び出して、特定のコプロセッサーで予約されたメモリーを解放できます。

ランタイム・オフロード・レポートから自動オフロードを把握する

自動オフロードは透過的なため、デフォルトでは、プログラマーは自動オフロードが行われているのかどうか、そしてどのように行われているのか分かりません。状況によっては (デバッグ時など)、何が行われているのかを把握したい場合もあるでしょう。そのような場合、インテル® MKL のランタイムは、プログラムの実行中に自動オフロードのプロファイル・レポートを生成できます。レポートを表示するには、実行前に環境変数 OFFLOAD_REPORT を 1 または 2 に設定します。この値はレポートの情報レベルを表し、値 2 は最も詳しい情報を出力します。

OFFLOAD_REPORT を設定すると、プログラマーは関数 mkl_mic_set_offload_report() を使って、動的にレポートをオン/オフにできます。この関数の引数は 1 つで、整数値を指定できます。非ゼロ値はレポートの生成を有効にし、ゼロ値で再度関数を呼び出すと、レポートの生成は無効になります。

同じプログラムで自動オフロードとコンパイラーによるオフロード支援を使用する

自動オフロードに加えて、インテル® MKL はコンパイラーによるオフロード支援 (CAO) をサポートしています。つまり、インテル® Fortran コンパイラー XE とインテル® C/C++ コンパイラー XE により提供されるコンパイラー・プラグマを用いて、オフロードを明示的に指定することができます。コンパイラーによるオフロード支援はプログラマーの負担を増やしますが、より細かな制御と柔軟性がもたらされます。洗練されたアプリケーションでは、1 つのアプリケーションで自動オフロードとコンパイラーによるオフロード支援の両方が必要になるかもしれません。インテル® MKL はこのような使用モデルをサポートしています。ただし、同じプログラムで自動オフロードとコンパイラーによるオフロード支援を使用する場合、サポート関数か環境変数を使って、自動オフロード対応関数の作業分割をプログラマーが明示的に指定する必要があります。そうでない場合、デフォルトの作業分割設定は、すべての処理をホストで実行し、オフロードは行いません。

オフロードに失敗したら実行の失敗を強制する

インテル® MKL の自動オフロードは、オフロードランタイムが利用可能なコプロセッサーを検出できない場合、またはコプロセッサーを適切に初期化できない場合、すべての計算をホストで行います。つまり、プログラマーは計算がコプロセッサーで行われたか、ホストで行われたか分かりません。このデフォルトの動作は、環境変数 MKL_MIC_DISABLE_HOST_FALLBACK を次のように設定して変更できます。

```
MKL_MIC_DISABLE_HOST_FALLBACK=1
```

MKL_MIC_DISABLE_HOST_FALLBACK を設定した場合、プログラムはオフロードを試みて失敗すると、「Could not enable Automatic Offload (自動オフロードを有効にできませんでした)」とい

うエラーメッセージを出力して終了します。

自動オフロードにおけるスレッド制御

ホストとコプロセッサーにおけるスレッド化は、2 つの OpenMP* 環境変数を使用して個別に制御できます。**自動オフロード**はホストとコプロセッサーの両方で計算をスポーンする可能性があるため、通常、パフォーマンスを最適化するには両方でこれらの環境変数を設定する必要があります。

OpenMP* のスレッド制御は、プログラムの実行中に使用可能な最大スレッド数とスレッド・アフィニティーを設定する柔軟な方法をプログラマーに提供します。次の環境変数を使用します。

ホスト	コプロセッサー
OMP_NUM_THREADS	MIC_OMP_NUM_THREADS
KMP_AFFINITY	MIC_KMP_AFFINITY

通常、コプロセッサーの環境変数にはプレフィックス "MIC_" が付いています。このプレフィックスはカスタマイズ可能で、環境変数 `MIC_ENV_PREFIX` で異なるプレフィックスを指定できます。これらの環境変数は、プログラマーによってホスト側で設定されます。自動オフロードでは、インテル® MKL ランタイムがこれらの環境変数を解釈し、自動的にコプロセッサー環境へ渡します。

コプロセッサー側では、1 つのコアは常にマイクロオペレーティング・システム (OS) の実行に使用されます。データ転送やハウスキーピング処理のようなタスクもこのコアで実行されます。パフォーマンス上の理由から、マイクロ OS を実行しているコアはオフロード実行に使用すべきではありません。このコアを除くすべてのコアを使用するように、コプロセッサーで (`MIC_KMP_AFFINITY` により) スレッド・アフィニティーを明示的に設定することを推奨します。ホストで実行する作業領域では、コア間でスレッドの移動が発生せず、キャッシュの局所性が失われないように `KMP_AFFINITY` を設定することを推奨します。

これまで説明した概念を明確にするため、次の例について考えてみましょう。以下の環境変数の設定例は、60 コアのコプロセッサー (コアあたり 4 スレッド) を搭載した 8 コアのシステム (コアあたり 1 スレッド) において適切なスレッド制御の設定です。

```
OMP_NUM_THREADS=8
KMP_AFFINITY=granularity=fine,compact,1,0
MIC_OMP_NUM_THREAD=236
MIC_KMP_AFFINITY=explicit,granularity=fine,proclist=[1-236:1]
MIC_ENV_PREFIX=MIC
```

まとめ

自動オフロードは、インテル® Xeon Phi™ コプロセッサーを搭載したシステムでインテル® MKL を使用する最も簡単な方法です。コードを変更することなく、ホストとコプロセッサー両方の利点を活用することができます。コプロセッサーが存在する場合、インテル® MKL はオフロードする価値のある処理およびホストとコプロセッサー間の適切な作業分割を決定します。そして、データ転送とリモート実行の管理を透過的に処理します。利用可能なコプロセッサーがない場合、オフロード処理はペナルティーなしでホスト上で実行されます。

インテル® MKL は、プログラマーが自動オフロードをより細かく制御できるように、ホストとコプロセッサー間の作業分割を制御するサポート関数と環境変数を提供します。

また、同じアプリケーションで自動オフロードとコンパイラーによるオフロード支援を利用することができます。

インテル® MKL は、インテル® Xeon Phi™ 製品において高度な並列アプリケーションで画期的なパフォーマンスを引き出します。これは、自動オフロード機能により、インテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーで一貫したプログラミング・モデルを保証することで、スムーズに達成できます。同じコード、使い慣れたツールとライブラリーにより、マルチコアからメニーコアへパフォーマンスをスケーリングすることができます。

関連情報

次のドキュメントは、この記事で触れたサポート関数と環境変数について詳しく説明しています。

- インテル® マス・カーネル・ライブラリー・リファレンス・マニュアル (英語)
- インテル® マス・カーネル・ライブラリー・ユーザーズガイド
- Intel MKL Automatic Offload Enabled Functions for Intel Xeon Phi Coprocessors (英語)
- Intel MKL Compiler Assisted Offload and Automatic Offload Example on Intel Xeon Phi Coprocessor (英語)

謝辞

この記事は、インテル® MKL のアーキテクトであり主任エンジニアである Greg Henry (greg.henry@intel.com) のテクニカル・プレゼンテーションを基にしています。

この記事のレビューに協力してくれた Shane Story (shane.story@intel.com)、Todd Rosenquist (todd.rosenquist@intel.com)、Ying Song (ying.song@intel.com)、Larry McGlinchy (larry.j.mcglinchy@intel.com)、Roger Herrick (roger.i.herrick.jr@intel.com) に感謝します。

著者紹介

Zhang Zhang (zhang.zhang@intel.com) は、インテル® ソフトウェア開発ツールのテクニカル・コンサルティング・エンジニアで、インテル® アーキテクチャーにおいてインテル® MKL やほかのソフトウェア開発製品を活用できるように ISV を支援しています。

著作権と商標について

最適化に関する注意事項

最適化に関する注意事項

インテル® コンパイラーは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。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® SSE2)、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。インテル® マイクロアーキテクチャーに非固有の特定の最適化は、インテル製マイクロプロセッサー向けに予約されています。この注意事項の適用対象である特定の命令セットの詳細は、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。

改訂 #20110804