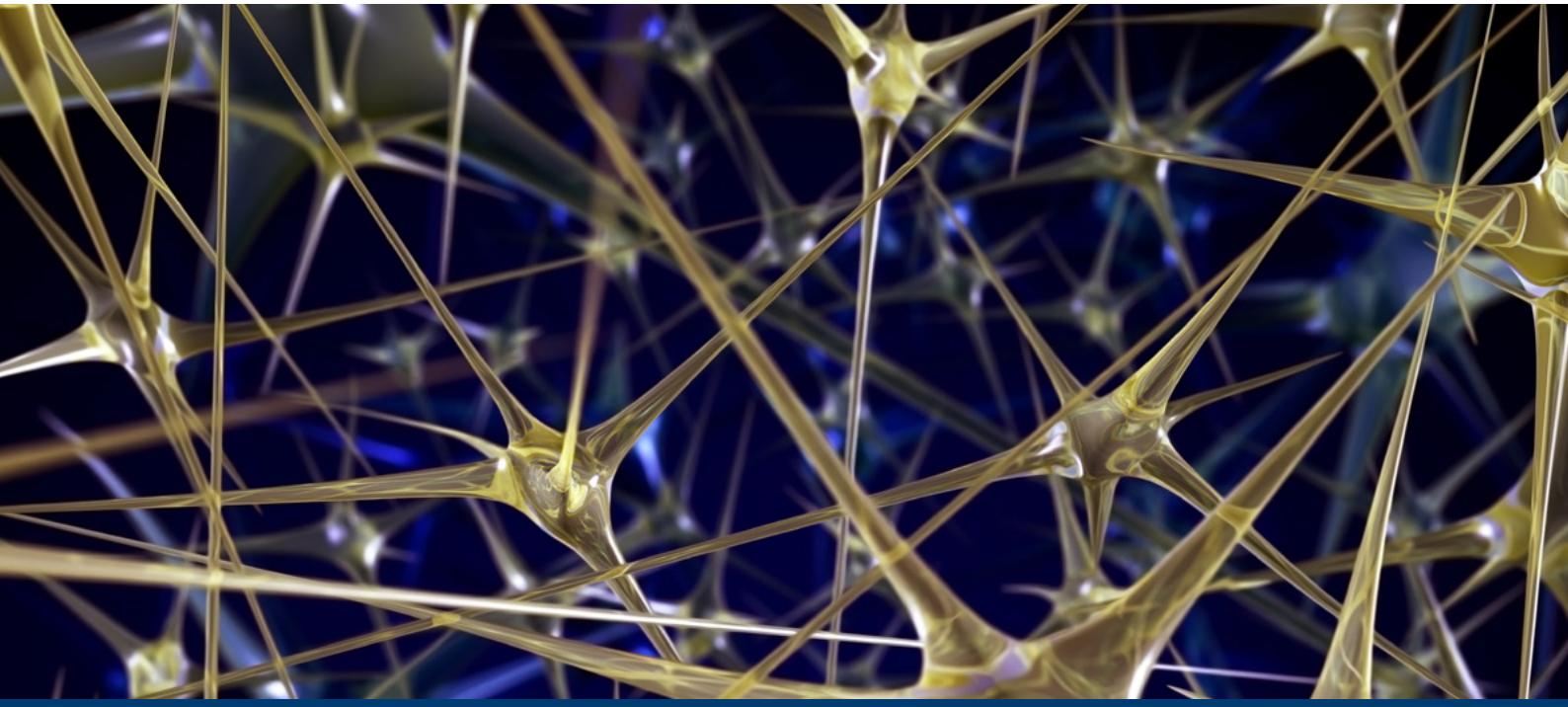


# 行列 - 行列乗算のパックのオーバーヘッドを減らす

## マルチコア / メニーコアのインテル® アーキテクチャーにおけるディープ・ニューラル・ネットワークのパフォーマンスの向上

Kazushige Goto、Murat Efe Guney、Sarah Knepper インテル コーポレーション ソフトウェア開発エンジニア

汎用行列 - 行列乗算 (GEMM) は、多くの科学、工学、マシンラーニング・アプリケーションで利用される基本操作であり、BLAS (Basic Linear Algebra Subprograms) ドメインの重要なルーチンの 1 つです。GEMM には 4 つの精度 (実数単精度、実数倍精度、複素数単精度、複素数倍精度) があります。この記事では、SGEMM (実数単精度) について取り上げます。

SGEMM の Fortran API は次のとおりです。

```
SGEMM(transa, transb, m, n, k, alpha, A, lda, B, ldb, beta, C, ldc)
```

この API は次の計算を実行します。

```
C := alpha * op(A) * op(B) + beta * op(C)
```

(`op(X)` は `X` または $X^T$、`transx` パラメーターの値に依存)

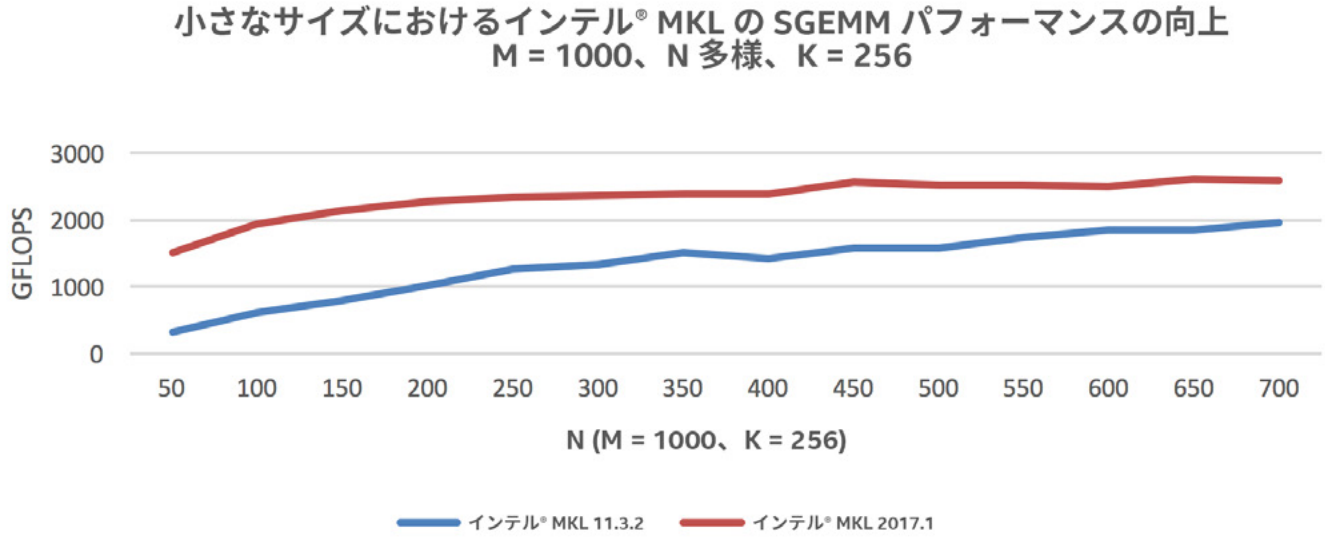
配列 A と B は入力で、C は入力および出力です。配列 A は **m** x **k** 行列、配列 **B** は **k** x **n** 行列、配列 C は **m** x **n** 行列をそれぞれ含みます。リーディング・ディメンジョン (**lda**、**ldb**、**ldc**) は、GEMM が大きな行列の一部を処理できるように、ある列から次の列へのストライドを決定します。リーディング・ディメンジョンは、後続列をキャッシュラインにアライメントしたり、8 ウェイ 1 次キャッシュの同じセットにマップすると、パフォーマンスにも影響します。

**インテル® マス・カーネル・ライブラリー (インテル® MKL)** は、ハイパフォーマンスな GEMM 実装を提供します。行列 - 行列乗算を最適化する一般的なアプローチは、オリジナルの入力行列のブロックを内部データ形式 (パックド形式など) に変換して、手書きのアセンブリー・カーネルで変換されたブロックを乗算した後、出力行列を更新します。[1] ブロックサイズはキャッシュとレジスターの使用率が最大になるように選択されます。パックする理由はさまざまです。

- A および B から多くのデータをキャッシュに格納できる (ブロックを大きくするとより多くのデータを再利用できる)。
- 連続し、アライメントされた、予測可能なアクセス (キャッシュおよびデータ・トランスレーション・ルックアサイド・バッファー (DTLB) のミスを最小限に抑えることができる)。
- ループのオーバーヘッドの軽減。

ハイパフォーマンス・コンピューティングの従来のサイズでは、このパックベースのアプローチは適切に動作します。一般に、**m** および **n** は比較的大きく、**k** は中程度 (外積) または同様に比較的大きい (正方) ため、入力行列のパックで費やされる時間は計算カーネルで費やされる時間に比べるとわずかです。しかし、一部のマシンラーニング・アプリケーションで一般的な、**m** または **n** のサイズが比較的小さい場合は、パックのオーバーヘッドが大きく影響します。その結果、明示的なパックに依存しない GEMM 実装のほうが、従来のパックベースの GEMM 実装よりもパフォーマンスが高くなることがあります。インテル® MKL 11.3 Update 3 には、**インテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2)**、インテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512)、第 2 世代**インテル® Xeon Phi™** プロセッサー向けに、これらのいくつかのサイズを最適化した {S,D}GEMM カーネルが含まれています。以降のインテル® MKL バージョンでも、これらのカーネルは引き続き改良されています。

これらの新しいカーネルが有効な例を示すため、**図 1** では、マシンラーニングで利用されるサイズで、インテル® MKL 2017 Update 1 とインテル® MKL 11.3 Update 2 の SGEMM のパフォーマンスを比較しています。ここで、**m** と **k** はそれぞれ 1000 と 256 に固定されていますが、**n** は可変です。パフォーマンスは GFLOPS (1 秒間に 10 億回の浮動小数点演算を実行) で表されるため、数値が高いほうが優れています。




構成情報 – バージョン: インテル® MKL 11.3.2 およびインテル® MKL 2017.1。ハードウェア: インテル® Xeon® プロセッサー E5-2699 v4、2 x 22 コア CPU (55MB LLC、2.20GHz)、64GB RAM。オペレーティング・システム : RHEL 7.2 GA x86_64。

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。* 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。ベンチマーク出典 : インテル コーポレーション

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

1 小さなサイズにおけるインテル® MKL の SGEMM パフォーマンスの向上

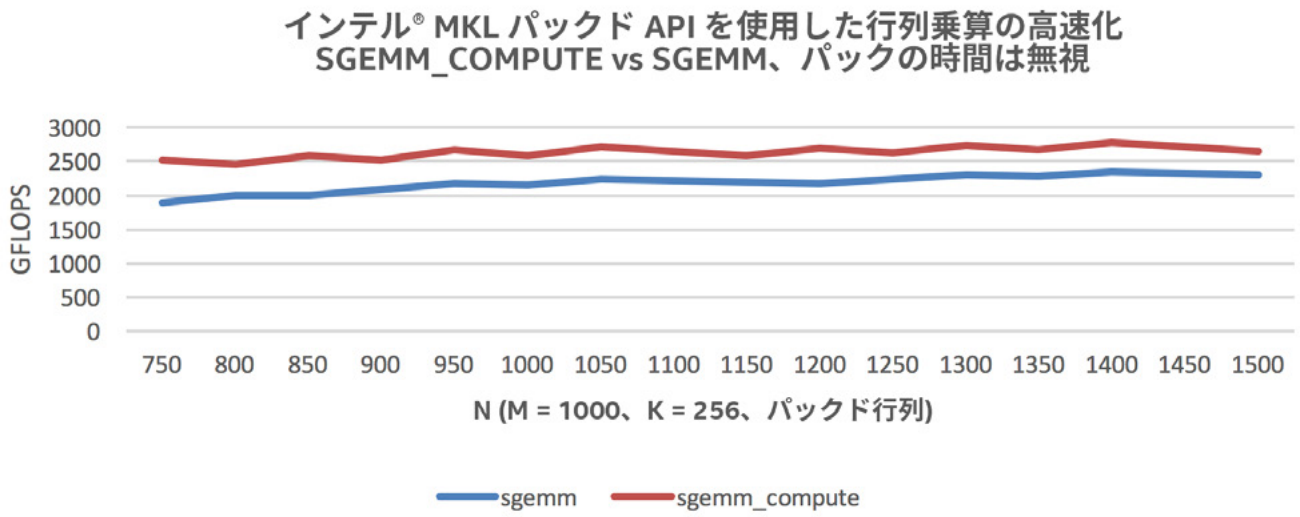
SGEMM 呼び出しの情報 (プロセッサーの種類とスレッド数、問題サイズとリーディング・ディメンジョン、転置パラメーターを含む) に基づいて、インテル® MKL は従来のカーネルと新しいパックフリー・カーネルのどちらを使用するか決定します。インテル® MKL の SGEMM パフォーマンスを利用するディープラーニング・フレームワークは、フレームワークの修正を行うことなく、これらの最適化の恩恵を受けることができます。

例えば、再帰型ニューラル・ネットワークのように入力行列 (A または B) が複数の行列乗算で再利用される場合、別の方法でパックのオーバーヘッドを最小限に抑えます。この方法では、再利用される行列を 1 回パックし、複数の SGEMM 計算でパックドバージョンを使用します。インテル® MKL 2017 では、複数の行列乗算でパックのオーバーヘッドを相殺できる、{S,D}GEMM のパックド API が追加されました。単精度の場合、新しいパックド API は次の 4 つです。

```
dest = sgemm_alloc (identifier, m, n, k)

sgemm_pack (identifier, trans, m, n, k, alpha, src, ld, dest)

sgemm_compute (transa, transb, m, n, k, A, lda, B, ldb, beta, C, ldc)

sgemm_free (dest)
```

パラメーターは SGEMM とほぼ同じで、パックする行列 (A または B) を示すパラメーター **identifier** が追加されています。**sgemm_compute** の **transa** および **transb** パラメーターは、通常どおり、“T” (転置する) または “N” (転置しない) に設定します。値 **p** は、対応する行列が内部パックド形式であることを示します。パックド API は、入力行列が複数回使用される場合に恩恵が得られます。そのため、**sgemm_alloc** と **sgemm_pack** をそれぞれ呼び出して、メモリーの割り当てと行列のパックを行い内部パックド形式にした後、**sgemm_compute** を複数回呼び出して、パックド行列を入力行列の 1 つとして渡します。最後に、**sgemm_free** を呼び出してメモリーを解放します。(詳細は、**インテル® MKL デベロッパー・リファレンス** (英語) を参照してください。)

**図 2** は、**sgemm** と **sgemm_compute** のパフォーマンスの比較を示しています (A 行列はパックド形式、パックの時間は無視)。




構成情報 - バージョン : インテル® MKL 2017.1。ハードウェア : インテル® Xeon® プロセッサー E5-2699 v4、2 x 22 コア CPU (55MB LLC、2.20GHz)、64GB RAM。オペレーティング・システム : RHEL 7.2 GA x86_64。

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。* 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。ベンチマーク出典 : インテル コーポレーション

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

2 インテル® MKL パックド API を使用した行列乗算の高速化

GEMM 呼び出し間で大きな A および B 入力行列を再利用するアプリケーションは、パックド API の恩恵を最も得られる可能性があります。`m` と `n` パラメーターがともに大きな場合、あるいは 2 つの小さな入力行列を再利用する場合は、パックド API に変更して得られるパフォーマンスの向上がプログラミングの労力と釣り合わない可能性があります。このため、アプリケーションのコードを変更してパックド API を使用する前に、特定のユースケースでのパフォーマンスを測定することを推奨します。

これまで説明したように、特に 1 つ以上の行列のサイズが小さい場合、従来の GEMM 実装で行われている内部パック操作では大きなオーバーヘッドが発生します。このオーバーヘッドは、入力行列を明示的にパックしないカーネルを使用するか、複数の GEMM 計算でパックのコストを相殺することにより減らすことができます。インテル® MKL 11.3 Update 3 から、{S,D}GEMM は、最初に内部バッファーにパックしないで入力行列を直接操作するカーネルを選択できるようになりました。使用するカーネルは、問題の特性とプロセッサー情報に基づいて実行時に決定されます。別の方法として、インテル® MKL 2017 では、入力行列の 1 つまたは両方を明示的にパックした後、複数の行列 - 行列乗算で再利用できるパックド API が追加されました。これらの 2 つのアプローチは、特にディープ・ニューラル・ネットワークで使用されるサイズを利用するときに、マルチコア / メニーコアのインテル® アーキテクチャーで高い GEMM パフォーマンスを達成するのに役立ちます。

1. Kazushige Goto and Robert A. van de Geijn. 2008. “Anatomy of High-Performance Matrix Multiplication.” ACM Transactions on Mathematical Software, 34, 3, Article 12, May 2008.