インテル® MKL を使用した小行列乗算の高速化

インテル® MKL チーム

内容

- インテル® MKL の概要
- インテル® MKL の新機能
- 行列-行列乗算
- 小行列のパフォーマンスの課題
- 小行列のパフォーマンスを向上するインテル® MKL のソリューション
  - MKL_DIRECT_CALL
  - バッチ API
  - コンパクト API
  - パックド API
- パフォーマンスのヒントと測定
- サマリーおよびインテル® MKL 関連情報

インテル® マス・カーネル・ライブラリー (インテル® MKL)

- 科学、工学、金融、マシンラーニング・アプリケーションにおける計算を高速化
- 密/スパース線形代数 (BLAS、LAPACK、PARDISO)、FFT、ベクトル演算、サマリー統計、ディープラーニング、スプラインなどの主な機能を提供
- インテル® Parallel Studio XE と インテル® System Studio で利用可能
- 無料およびロイヤルティー・フリーで利用可能
- シングルコアのベクトル化とキャッシュ効率向けに最適化
- マルチコアとメニーコアの自動並列化
- クラスターにスケーリング

最適化された数値計算ビルディング・ブロック

線形代数
- BLAS
- LAPACK
- ScaLAPACK
- スパース BLAS
- PARDISO SMP
- クラスター直接法スパースソルバー
- 反復法スパースソルバー

高速フーリエ変換
- 多次元
- FFTW インターフェイス
- クラスター FFT

ベクトル演算
- 三角関数
- 双曲線
- 指数
- 対数
- 累乗
- 累乗根
- ベクトル RNG

ディープ・ニューラル・ネットワーク
- 畳み込み
- プーリング
- 正規化
- ReLU
- 内積

サマリー統計
- 尖度
- 中心積率
- 変化係数
- 順序統計量と分位数
- 最小/最大
- 分散/共分散
- ロバスト推定

その他
- スプライン
- 補間
- 信頼領域
- 高速ポアソンソルバー

チューニングされた ISA 固有のコードパスに自動ディスパッチ

コア数の増加 → スレッド数の増加 → ベクトル幅の増加

	インテル® Xeon® プロセッサー 64 ビット	インテル® Xeon® プロセッサー 5100 番台	インテル® Xeon® プロセッサー 5500 番台	インテル® Xeon® プロセッサー 5600 番台	インテル® Xeon® プロセッサー E5-2600 v2 製品ファミリー	インテル® Xeon® プロセッサー E5-2600 v3 製品ファミリー v4 製品ファミリー	インテル® Xeon® スケーラブル・プロセッサー[1]	インテル® Xeon Phi™ x200 製品ファミリー (開発コード名 Knights Landing)
最大コア数	1	2	4	6	12	18-22	28	72
最大スレッド数	2	2	8	12	24	36-44	56	288
SIMD 幅	128	128	128	128	256	256	512	512
ベクトル ISA	インテル® SSE3	インテル® SSE3	インテル® SSE4、インテル® SSE4.1	インテル® SSE4.2	インテル® AVX	インテル® AVX2	インテル® AVX-512	インテル® AVX-512

1. 発売済および出荷済製品の製品仕様は ark.intel.com を参照してください。

インテル® SSE: インテル® ストリーミング SIMD 拡張命令
インテル® AVX: インテル® アドバンスト・ベクトル・エクステンション

intel Software

インテル® MKL 2018 の新機能と最適化

- インテル® Xeon Phi™ プロセッサー (開発コード名 Knights Mill) 向けの最適化
  - DNN 畳み込み関数および内積関数の最適化 (インテル® MKL-DNN)
  - SGEMM の最適化 (AVX512_4FMAPS 向け)
  - BLAS3 実数および複素数単精度の最適化 (AVX512_4FMAPS 向け、インテル® MKL 2018.1)
  - 新しい整数 GEMM API (8 ビットまたは 16 ビット入力、32 ビット出力)
- BLAS および LAPACK
  - コンパクト BLAS および LAPACK 関数
  - LAPACK コレスキーおよび QR の直接呼び出しのサポート
  - ピボット選択なし LU 因数分解および逆関数
  - Aasen ベースの因数分解およびソルバー関数
  - 制限付き Bunch-Kaufman (rook) ピボット選択因数分解
- スパース BLAS
  - 前処理付き対称ガウス・ザイデル
  - スパース SYRK ルーチン
- FFT
  - Verbose モードのサポート
- ベクトル演算
  - 24 の新しい関数: v?Fmod、v?Remainder、v?Powr、v?Exp2、v?Exp10、v?Cospi、v?Sinpi、v?Tanpi など

インテル® MKL の主な拡張点

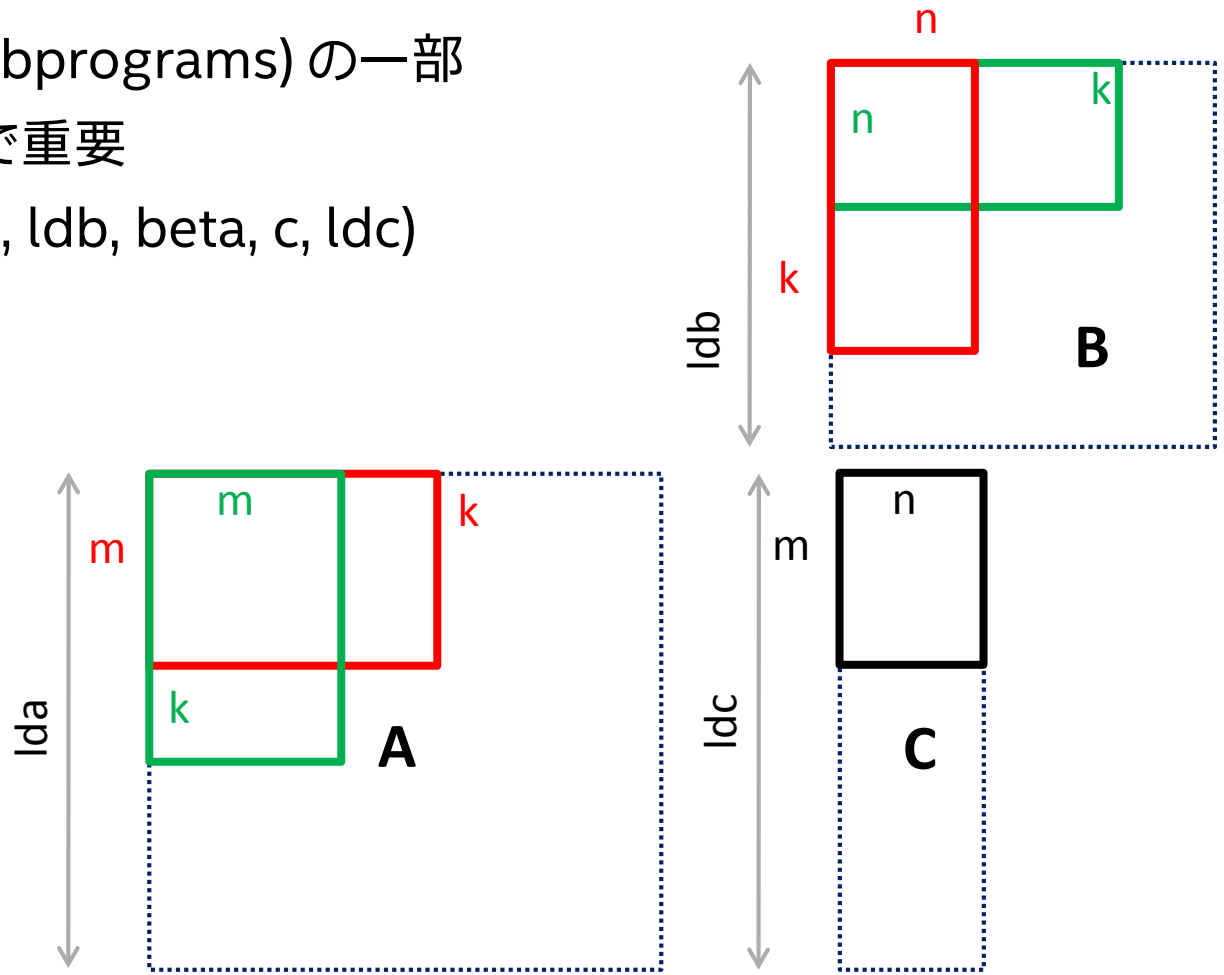
- 条件付き数値再現性 (CNR)
- インテル® スレッディング・ビルディング・ブロック (インテル® TBB) とのコンポーザビリティー
- インテル® Optimized High Performance Conjugate Gradient (HPCG) Benchmark
- スパース BLAS 検査-実行 API
- クラスターのサポートの拡張 (MPI ラッパーおよび macOS*)
- クラスター用並列直接法スパースソルバー
- 拡張固有値ソルバー
- ディープ・ニューラル・ネットワークの畳み込み、正規化、活性化、プーリング・プリミティブ
- **GEMM の拡張**
- **MKL_DIRECT_CALL、バッチ API およびパックド API**

行列-行列乗算

- インテル® MKL BLAS (Basic Linear Algebra Subprograms) の一部
- 科学、工学、マシンラーニング・アプリケーションで重要
- *GEMM(transa, transb, m, n, k, alpha, a, lda, b, ldb, beta, c, ldc)

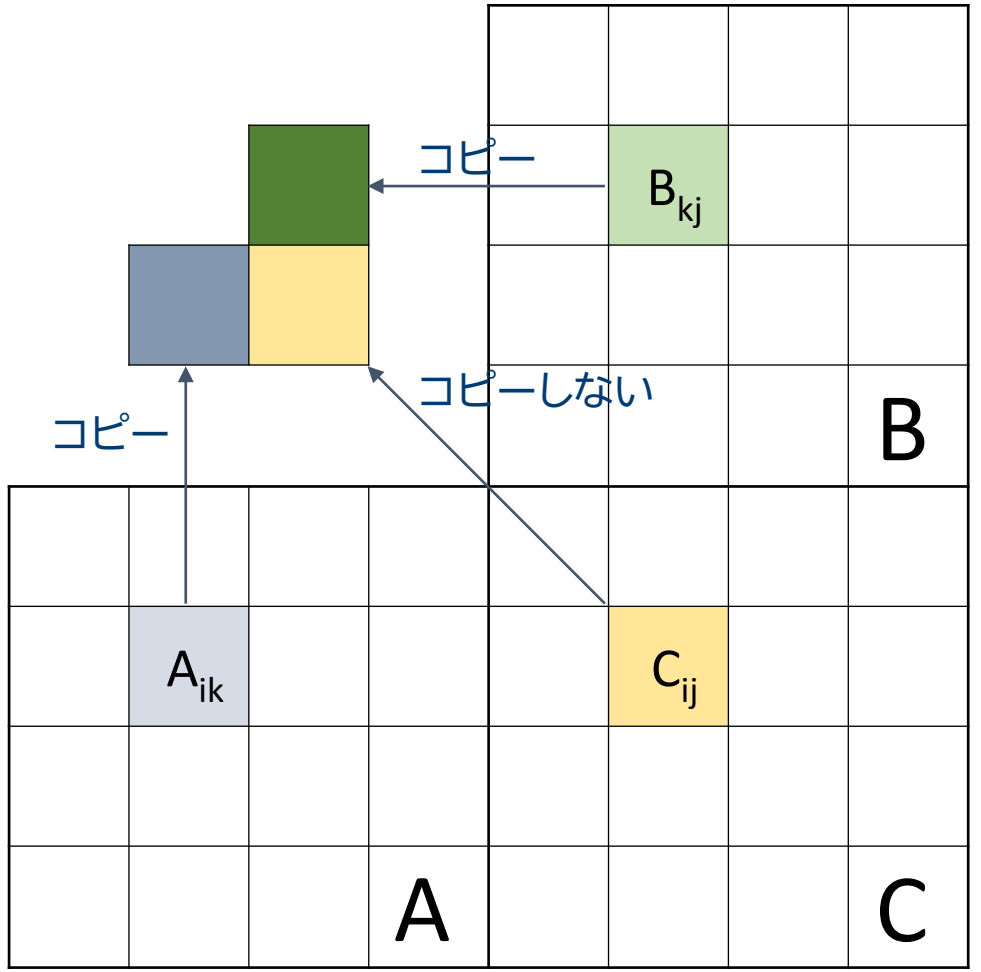
C = *alpha* op(A) * op(B) + *beta* C
op(X) = X または X^T

```
C = beta*C
DO i=1,M
   DO j=1,N
      DO kk=1,K
         C(i,j) += alpha*A(i,kk)*B(kk,j)
      END DO
   END DO
END DO
```

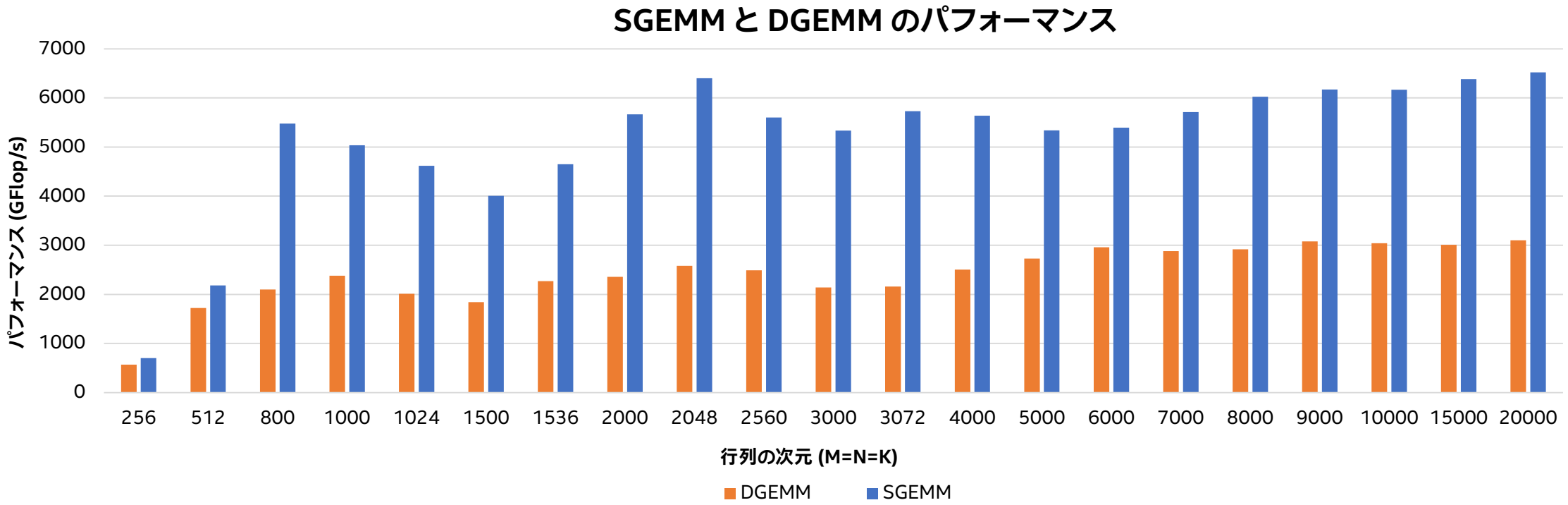


行列-行列乗算の最適化

- インテル® アーキテクチャー向けに高度にチューニング
  - 高スループットの SIMD 命令
  - マルチコア/メニーコア対応の並列アルゴリズム
  - タイリングによりキャッシュの再利用を最大化
- ほぼマシンのピーク・パフォーマンスで動作する高度にチューニングされたアセンブリー・カーネル
- 入力行列 A と B をバッファーにコピー
  - カーネル内部で連続するデータ・アクセス・パターン
  - キャッシュミスと TLB ミスを最小化
- コピーのオーバーヘッドは大きな行列では無視できる
  - 計算: $O(N^3)$、コピー: $O(N^2)$
- インテル® MKL は行列が小さい場合コピーをスキップする
  - 適切なリーディング・ディメンジョンが必要 (256 の倍数を避ける)
  - A と B が転置でないことが望ましい



インテル® Xeon® Platinum プロセッサー上での行列-行列乗算のパフォーマンス



システム構成: ハードウェア: インテル® Xeon® Platinum 8180 プロセッサー、2x28 コア、2.50GHz、376GB RAM。オペレーティング・システム: Ubuntu* 16.04 LTS。ソフトウェア: インテル® MKL 2018。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。 詳細については、www.intel.com/benchmarks (英語) を参照してください。ベンチマークの出典: 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

intel Software

行列サイズの分類とパフォーマンスの課題

- 小サイズ
  - M, N, K < 20
  - 課題: 関数呼び出しのオーバーヘッドが大きい、ベクトル化および並列化の効果が低い
  - ソリューション: バッチ API、コンパクト API、および MKL_DIRECT_CALL
- 中サイズ
  - 20 < M, N, K < 500
  - 課題: 並列化の効果が低い、コピーのオーバーヘッドが大きい
  - ソリューション: バッチ API およびパックド API
- 非対称サイズ
  - M < 500 で N が大きい
  - N < 500 で N が大きい
  - 課題: コピーのオーバーヘッドが大きい
  - ソリューション: パックド API
- 大サイズ
  - M, N, K > 5000
  - パフォーマンスはマシンの理論的なピークに近い

小、中、非対称サイズ向けのインテル® MKL のソリューション

- MKL_DIRECT_CALL
  - 小サイズ (M, N, K < 20) のパフォーマンスを向上
  - エラーチェックと関数呼び出しを省略してオーバーヘッドを軽減
  - 複数の関数で有効
    - BLAS: gemm、gemm3m、syrk、trsm、axpy、dot
    - LAPACK: potrf、getrf、getrs、getri、geqrf
- コンパクト API
  - 小サイズ (M, N, K < 20) のパフォーマンスを向上
  - データをコンパクト形式に変更することにより非常に小さな次元の行列のベクトル化が可能
  - 複数の関数で有効
    - BLAS: gemm、trsm
    - LAPACK: getrinp、getrfnp、potrf、geqrf
- バッチ API
  - 小-中サイズ (M, N, K < 500) のパフォーマンスを向上
  - 複数の独立した関数呼び出しをグループ化
  - gemm、gemm3m および trsm BLAS 関数で利用可能
- パックド API
  - 小-中 M または N サイズ (M または N < 500) のパフォーマンスを向上
  - 同じ入力行列の複数の GEMM 呼び出しでコピーのオーバーヘッドを軽減
  - sgemm および dgemm BLAS 関数で利用可能

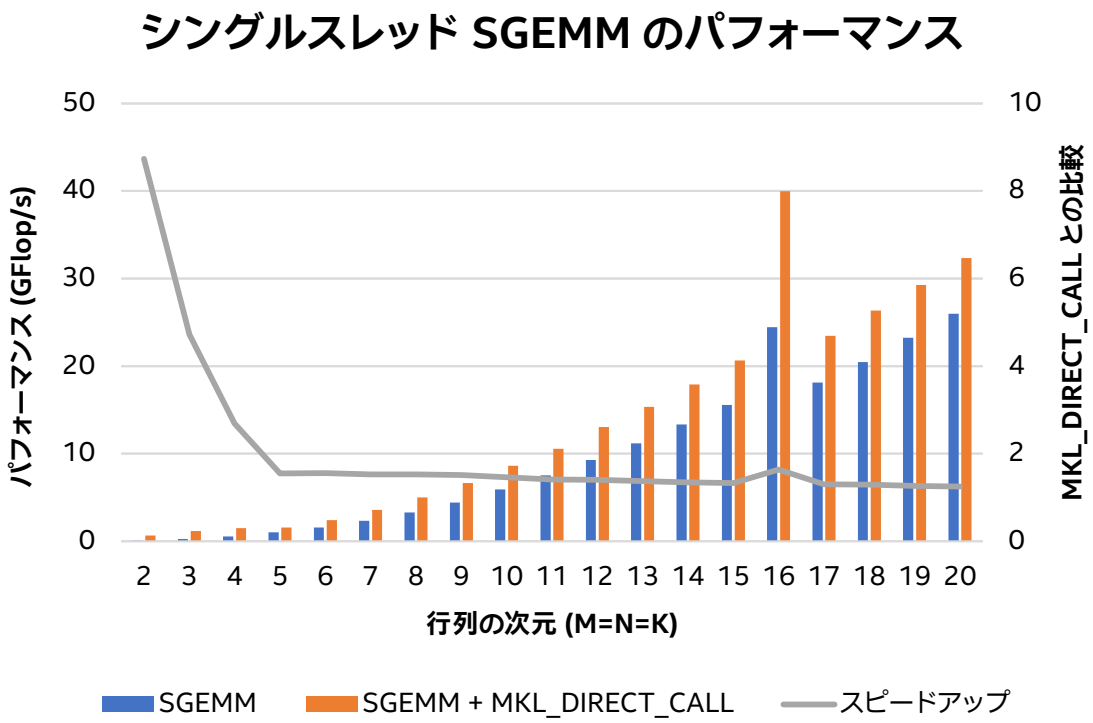
MKL_DIRECT_CALL コンパイラー・オプション

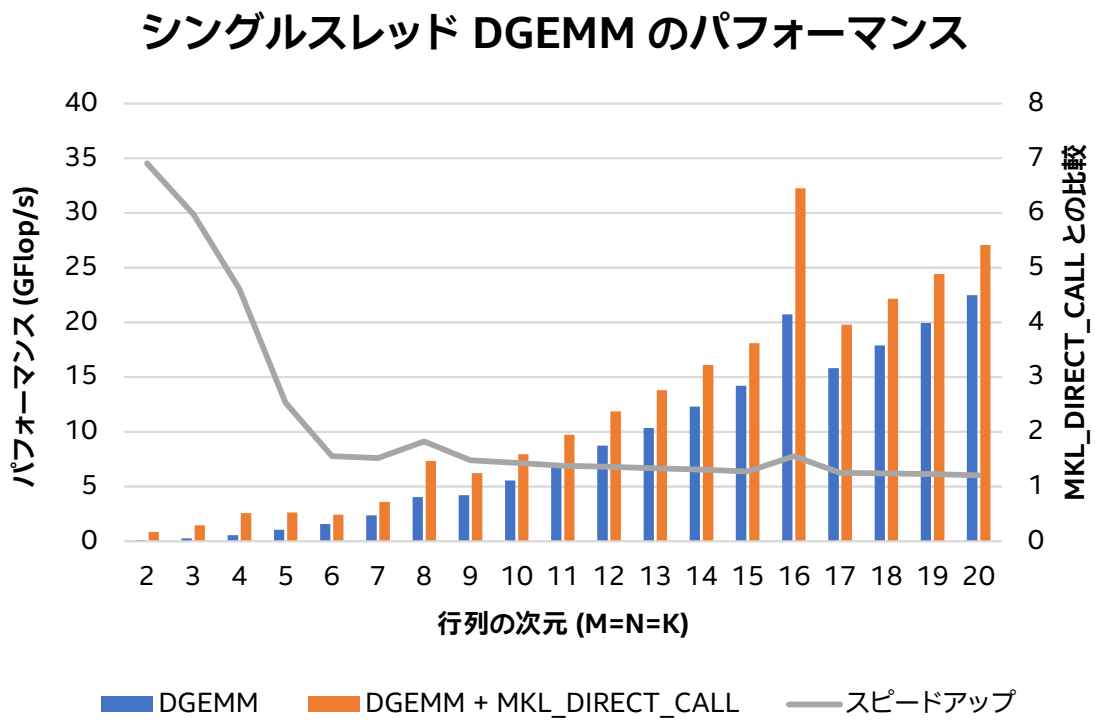
- プリプロセッサー・マクロ MKL_DIRECT_CALL を定義
  - スレッド化が必要ない場合は MKL_DIRECT_CALL_SEQ を使用
- 小サイズ (M, N, K < 20) のパフォーマンスを向上
  - ライブラリー関数を呼び出す代わりに C 実装を使用
  - インテル® MKL 2018.1 以降、DGEMM インテル® AVX2 以降向けのコンパイラー組込み関数カーネル
- インテル® MKL はオーバーヘッドを回避できる
  - エラーチェックなし
  - MKL_VERBOSE のサポートなし
  - CNR (条件付き数値再現性) のサポートなし
- 最小限の変更が必要、プリプロセッサー・マクロとヘッダーファイルを追加:

```
// icc でコンパイル -DMKL_DIRECT_CALL …
#include <mkl.h>
void main(void) {
    dgemm(…);
}
```

```
! ifort でコンパイル -DMKL_DIRECT_CALL -fpp …
#     include “mkl_direct_call.fi”
      program DGEMM_MAIN
      DGEMM(…)
```

インテル® Xeon® Platinum プロセッサー上での MKL_DIRECT_CALL のパフォーマンス





システム構成: ハードウェア: インテル® Xeon® Platinum 8180 プロセッサー、2x28 コア、2.50GHz、192GB RAM。オペレーティング・システム: Red Hat* Enterprise Linux* 7.2 LTS。ソフトウェア: インテル® MKL 2018。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。 詳細については、www.intel.com/benchmarks (英語) を参照してください。ベンチマークの出典: 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

バッチ API

- 1 つの関数呼び出しで複数の独立した汎用行列乗算 (GEMM) 操作を同時に実行
- 操作間でデータ依存性がないことを保証
- 小-中サイズ (M, N, K < 500) でもすべてのコアを活用
- ライブラリーのオーバーヘッドを最小化
- 同じサイズの行列をグループ化するコード変更が必要

$$\mathbf{C}^1 = \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^1) \cdot \mathrm{op}(\mathbf{B}^1) + \mathit{beta} \cdot \mathbf{C}^1$$

$$\mathbf{C}^2 = \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^2) \cdot \mathrm{op}(\mathbf{B}^2) + \mathit{beta} \cdot \mathbf{C}^2$$

$$\mathbf{C}^3 = \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^3) \cdot \mathrm{op}(\mathbf{B}^3) + \mathit{beta} \cdot \mathbf{C}^3$$

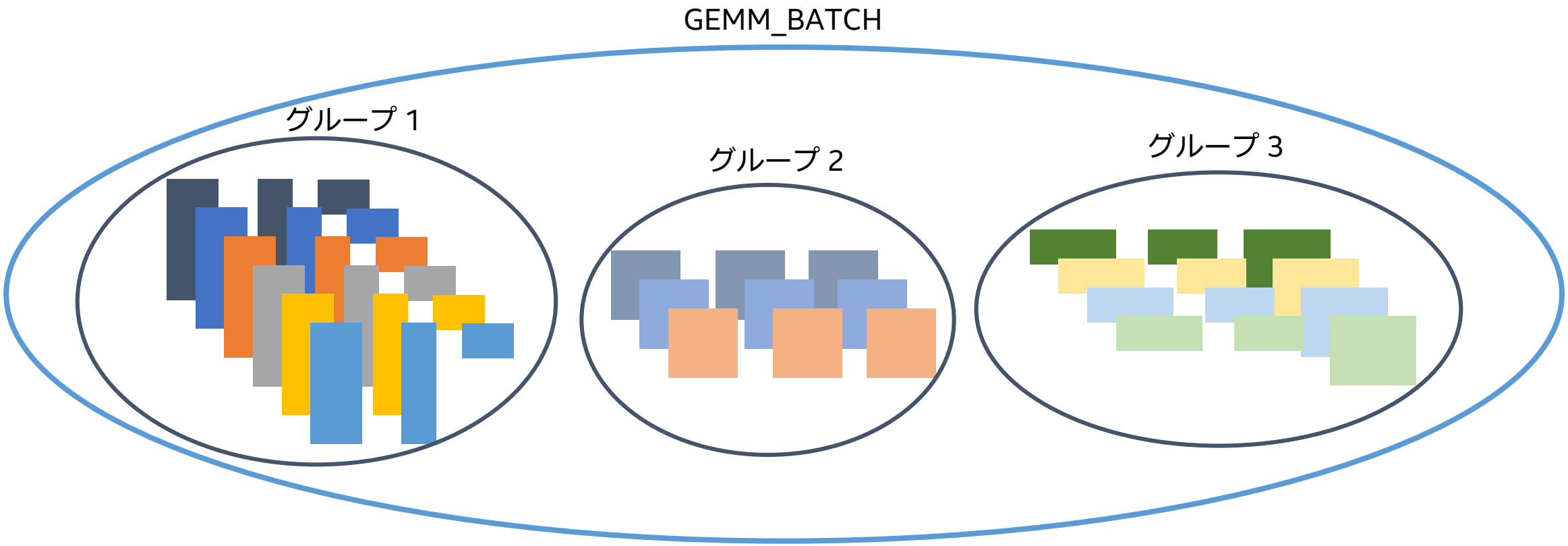
ポインター・エイリアシングがないと仮定して並列で実行

$$\mathbf{C}^2 = \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^4) \cdot \mathrm{op}(\mathbf{B}^4) + \mathit{beta} \cdot \mathbf{C}^2$$

$\mathbf{C}^2$ への前の書き込みを待つ

バッチ API のグループコンセプト

- グループ: 同じ入力パラメーターの GEMM 操作のセット (異なる行列ポインターを含む)
  - 転置、サイズ、リーディング・ディメンジョン、アルファおよびベータ
- 1 つの GEMM_BATCH 呼び出しで複数のグループを制御できる

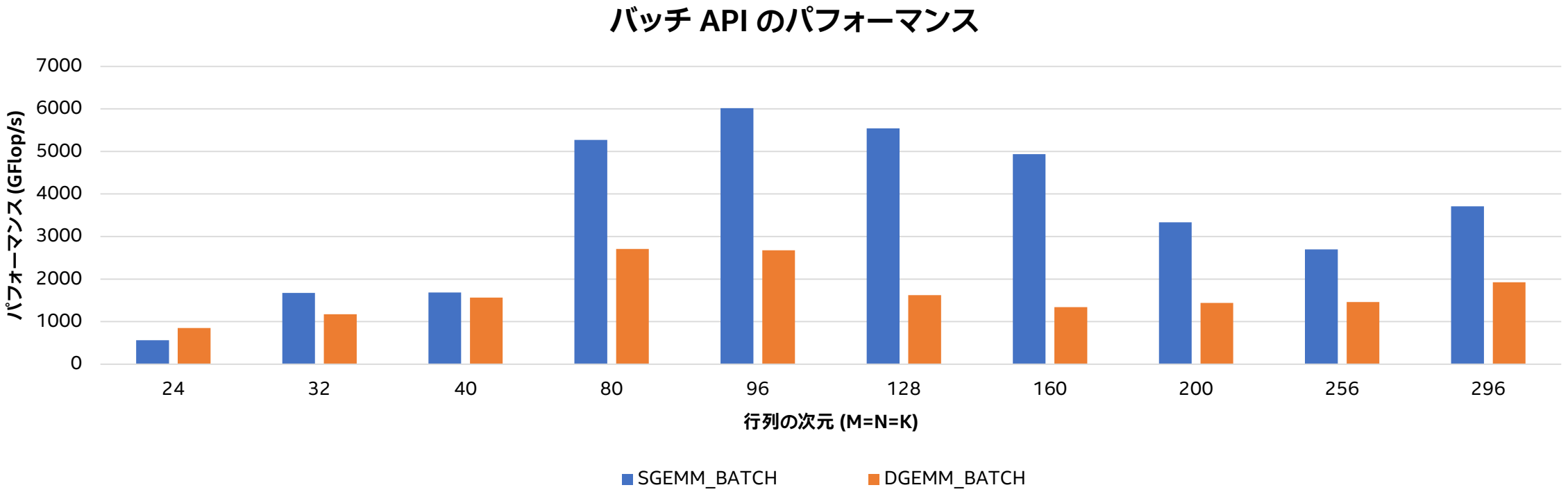


バッチ API の使用例

- 同じパラメーターの GEMM 呼び出しをグループ化
- GEMM 呼び出しの 2 つのグループの例:

```
#include <mkl.h>
int group_count = 2;
// group_count の配列サイズを作成して GEMM 引数に格納
CBLAS_TRANSPOSE transA[] = {CblasNoTrans, CblasNoTrans};
CBLAS_TRANSPOSE transB[] = {CblasTrans, CblasNoTrans};
MKL_INT   m[] = {4, 3};
MKL_INT   k[] = {4, 6};
MKL_INT   n[] = {8, 3};
MKL_INT   lda[] = {4, 6};
MKL_INT   ldb[] = {4, 6};
MKL_INT   ldc[] = {8, 3};
double    alpha[] = {1.0, 1.0};
double    beta[] = {0.0, 2.0};
MKL_INT size_per_grp[] = {20, 30};

// cblas_dgemm_batch を呼び出して 50 の GEMM 操作を実行
cblas_dgemm_batch(CblasRowMajor, transA, transB, m, n, k,
                  alpha, a_array, lda, b_array, ldb,
                  beta, c_array, ldc, group_count, size_per_group);
```

インテル® Xeon® Platinum プロセッサー上での バッチ API のパフォーマンス



システム構成: ハードウェア: インテル® Xeon® Platinum 8180 プロセッサー、2x28 コア、2.50GHz、376GB RAM。オペレーティング・システム: Ubuntu* 16.04 LTS。ソフトウェア: インテル® MKL 2018。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。 詳細については、www.intel.com/benchmarks (英語) を参照してください。ベンチマークの出典: インテル コーポレーション

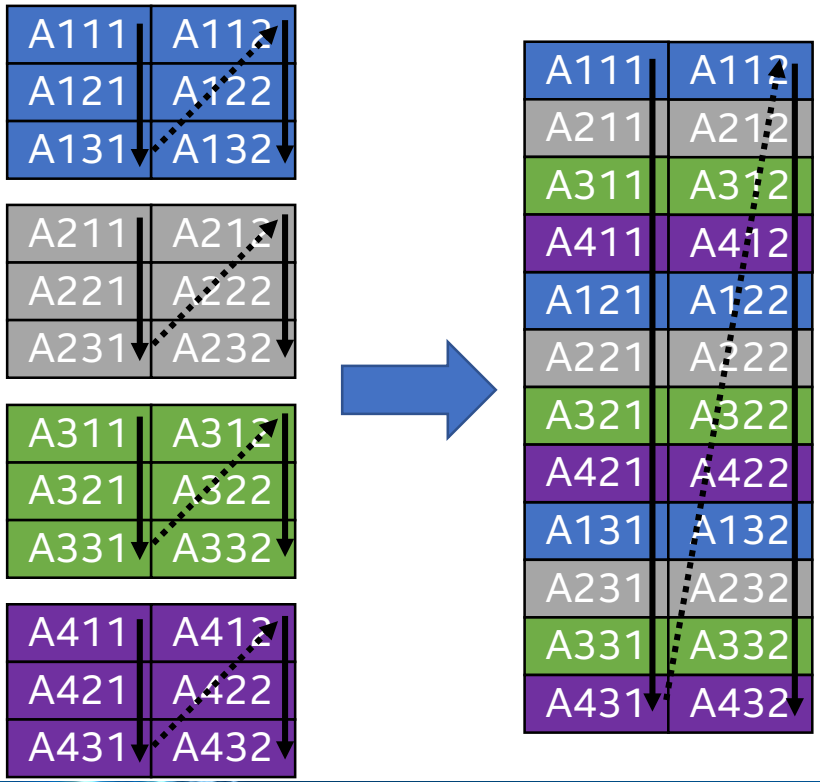
最適化に関する注意事項: インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。これには、インテル® ストリーミング SIMD 拡張命令 2、インテル® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® 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

intel Software

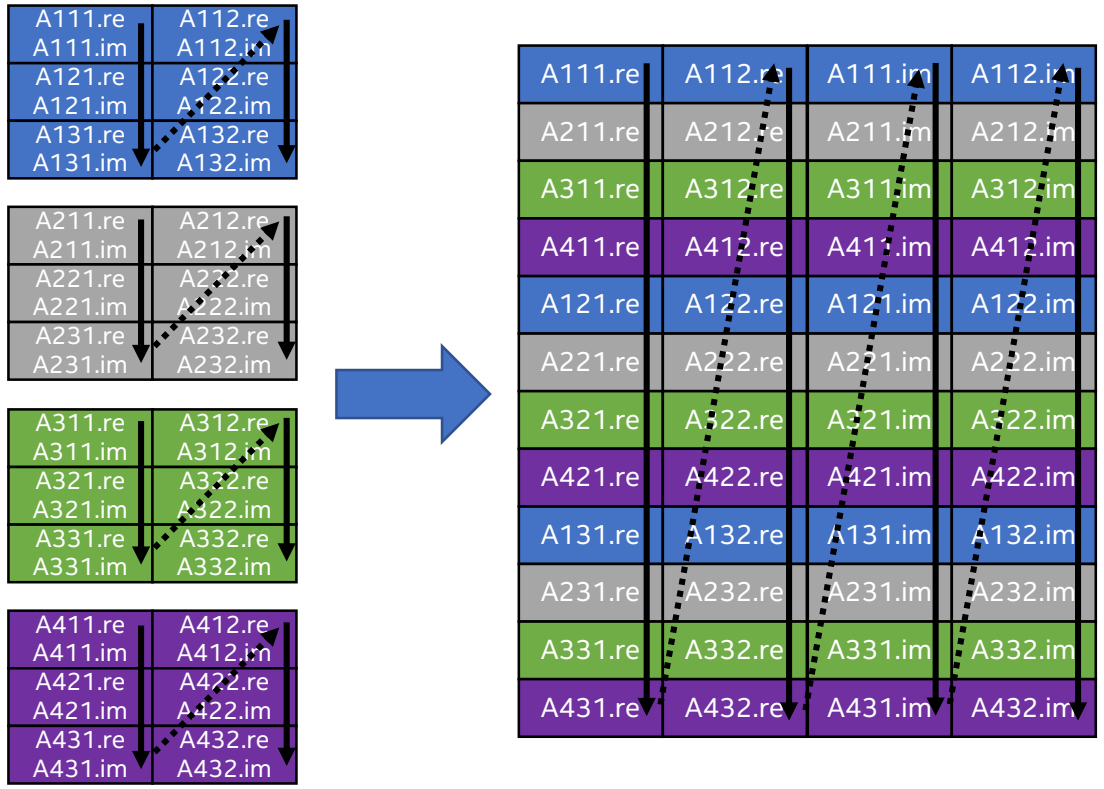
コンパクト API

- 大量の同じサイズの行列のベクトル化を利用する新しいデータ形式
- 同じインデックスの行列要素はメモリーでインターリーブ
- サブグループのサイズは SIMD 長 (SIMD 命令を活用するため)
- サブグループのサイズ = 4 で 3x2 の行列を変更した例:

実数データ型



複素数データ型

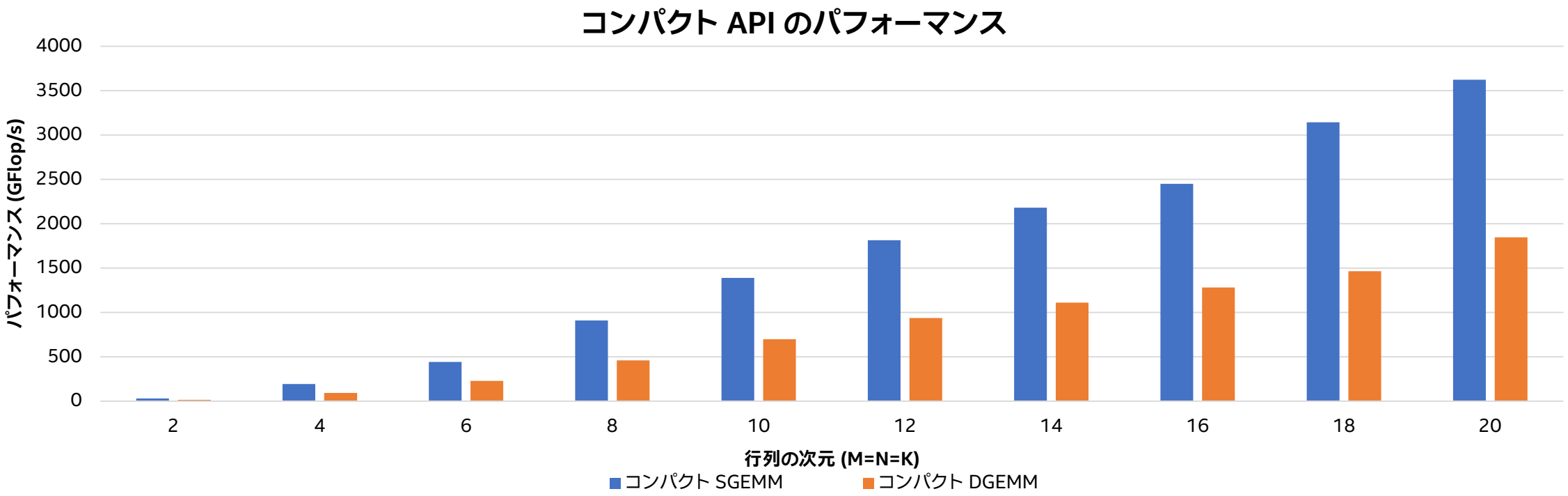


コンパクト API の使用例

- 一部のコード変更が必要な非標準 BLAS API
- 同じサイズの小行列 (M, N, K < 20) のグループのパフォーマンスを大幅に向上
- インテル® MKL ユーティリティー関数により列/行優先で行列を変換、コンパクト形式

```
#include <mkl.h>

// アーキテクチャーの最適な形式を照会
MKL_COMPACT_PACK compact_format = mkl_get_format_compact();

// コンパクト形式のメモリー割り当て
a_size = mkl_dget_size_compact(lda, k, compact_format, num_matrix);
b_size = mkl_dget_size_compact(ldb, n, compact_format, num_matrix);
c_size = mkl_dget_size_compact(ldc, n, compact_format, num_matrix);

// データをコンパクト形式に変換
mkl_dgepack_compact(layout, m, k, a_array, lda, a_c, lda, compact_format, num_matrix);
mkl_dgepack_compact(layout, k, n, b_array, ldb, b_c, ldb, compact_format, num_matrix);
mkl_dgepack_compact(layout, m, n, c_array, ldc, c_c, ldc, compact_format, num_matrix);

// コンパクト形式で複数の dgemm 操作を実行
mkl_dgemm_compact(layout, transa, transb, m, n, k, alpha, a_c, lda, b_c, ldb, beta, c_c, ldc, compact_format, num_matrix);

// コンパクト形式から標準 BLAS 形式に変換
mkl_dgeunpack_compact(layout, m, n, c_array, ldc, c_c, ldc, compact_format, num_matrix);
```

インテル® Xeon® Platinum プロセッサー上でのコンパクト API のパフォーマンス



システム構成: ハードウェア: インテル® Xeon® Platinum 8180 プロセッサー、2x28 コア、2.50GHz、376GB RAM。オペレーティング・システム: Ubuntu* 16.04 LTS。ソフトウェア: インテル® MKL 2018。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。 詳細については、www.intel.com/benchmarks (英語) を参照してください。ベンチマークの出典: 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

intel Software

パックド API

- 同じ入力行列の複数の GEMM 呼び出しにおけるコピー (パック) 操作を最小化
- コピー (パック) したデータを多くの GEMM 呼び出しで再利用
- 入力行列の再利用により中または非対称サイズ (M または N < 500) のパフォーマンスを向上

$$\mathbf{C}^1 = \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^1) \cdot \mathrm{op}(\mathbf{B}^1) + \mathit{beta} \cdot \mathbf{C}^1$$

$$\mathbf{C}^2 = \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^1) \cdot \mathrm{op}(\mathbf{B}^2) + \mathit{beta} \cdot \mathbf{C}^2$$

$$\mathbf{C}^3 = \mathit{alpha} \cdot \mathrm{op}(\mathbf{A}^1) \cdot \mathrm{op}(\mathbf{B}^3) + \mathit{beta} \cdot \mathbf{C}^3$$

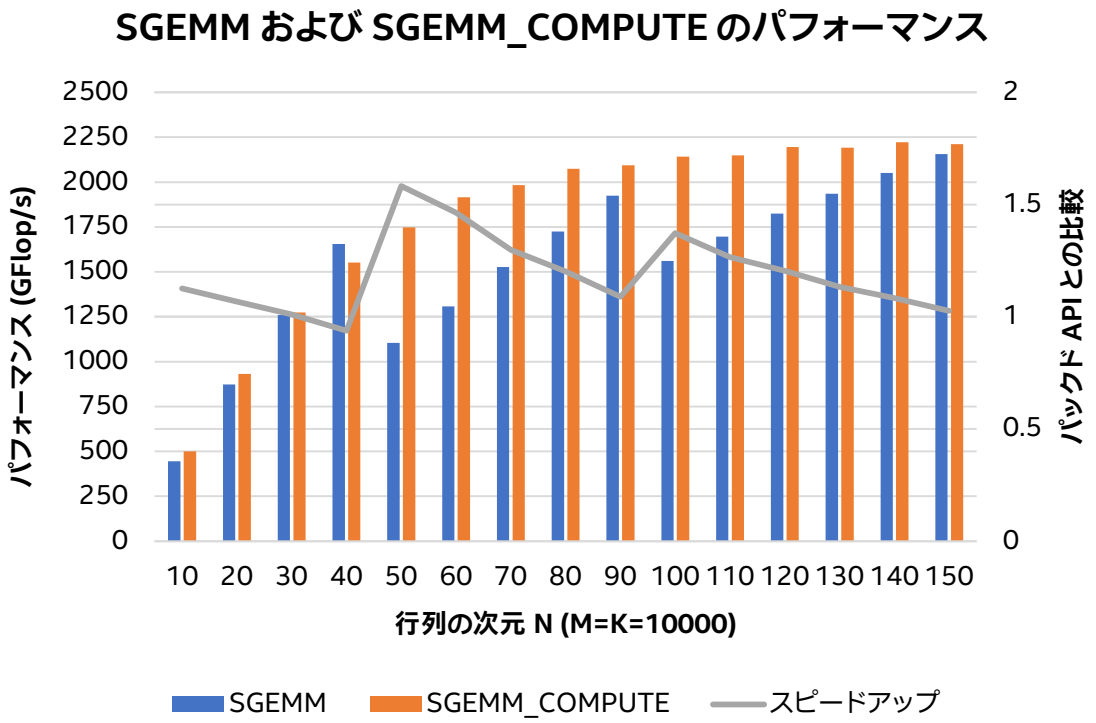
入力行列 A^1 は 3 つの GEMM 呼び出しで共有される

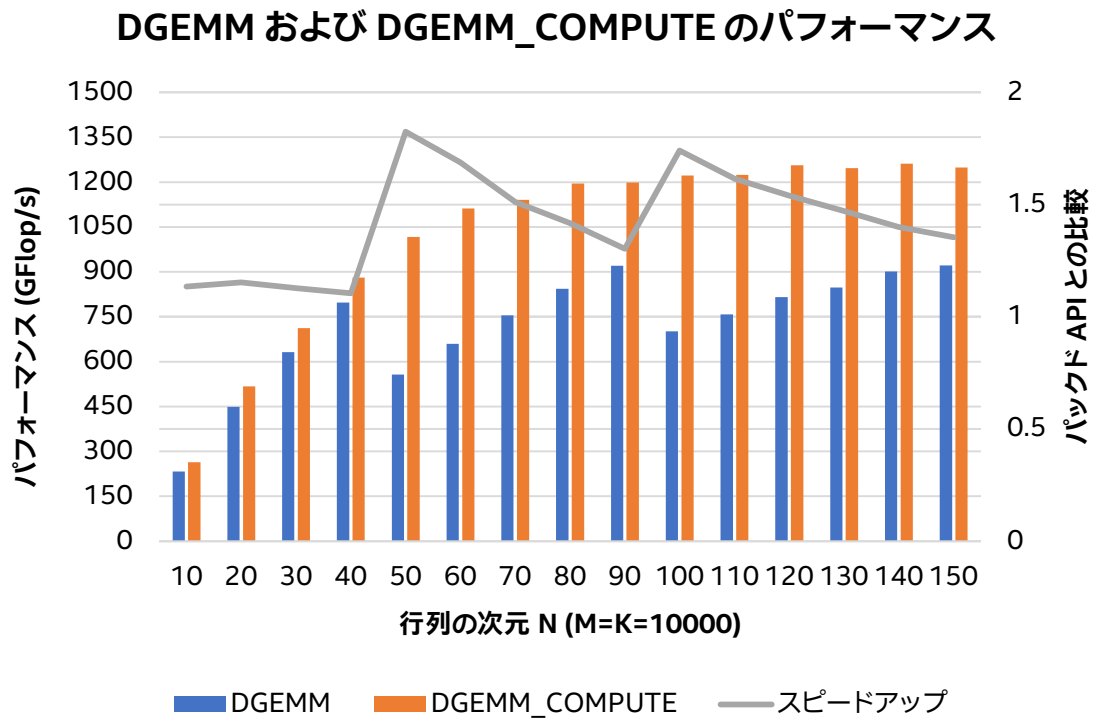
パックド API の使用例

- GEMM 呼び出しを GEMM_PACK + GEMM_COMPUTE に変換するコード変更が必要
- 行列 A を共有する 3 つの SGEMM 呼び出しをパックド API で計算する場合の例:

```
#include <mkl.h>

float *Ap;
Ap = sgemm_alloc(“A”, &m, &n, &k);

// A をパックド形式に変換
sgemm_pack(“A”, “T”, &m, &n, &k, &alpha, A, &lda, Ap);

// 行列 A のパックド形式 Ap を使用して SGEMM 計算を実行
sgemm_compute(“P”, “N”, &m, &n, &k, Ap, &lda, B1, &ldb1, &beta, C1, &ldc1);
sgemm_compute(“P”, “N”, &m, &n, &k, Ap, &lda, B2, &ldb2, &beta, C2, &ldc2);
sgemm_compute(“P”, “N”, &m, &n, &k, Ap, &lda, B3, &ldb3, &beta, C3, &ldc3);

// Ap のメモリーを解放
sgemm_free(Ap);
```

インテル® Xeon® プロセッサー E5-2699 v4 上でのパックド API のパフォーマンス





システム構成: ハードウェア: インテル® Xeon® プロセッサー E5-2699 v4、2x22 コア、2.20GHz、64GB RAM。オペレーティング・システム: Red Hat* Enterprise Linux* 7.2。ソフトウェア: インテル® MKL 2018。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。 詳細については、www.intel.com/benchmarks (英語) を参照してください。ベンチマークの出典: 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

intel Software

インテル® MKL の重要なパフォーマンスのヒント

- KMP_AFFINITY を設定してスレッド・マイグレーションを回避する
  - インテル® ハイパースレッディング (HT) テクノロジー有効:
    - Linux*/macOS*: export KMP_AFFINITY=compact,1,0,granularity=fine
    - Windows*: set KMP_AFFINITY=compact,1,0,granularity=fine
  - インテル® ハイパースレッディング (HT) テクノロジー無効:
    - Linux*/macOS*: export KMP_AFFINITY=compact
    - Windows*: set KMP_AFFINITY=compact
- リーディング・ディメンジョンが 256 の倍数になるのを避ける
  - ldim % 256 = 0 の場合は ldim に 16 を加算
- ページ境界でメモリーをアライメントする
  - メモリーの割り当てと解放には mkl_malloc および mkl_free を使用する
- ローカルメモリーのアクセスを最大化
- インテル® MKL が内部バッファーに高帯域幅メモリーを使用するようにする
  - memkind ライブラリーをインストールする
  - numactl
- 詳細は『インテル® MKL デベロッパー・ガイド』の「パフォーマンスとメモリーの管理」を参照

インテル® MKL 関数のパフォーマンスの評価

- 重要: 小行列向けのソリューションを使用する前にユースケースのパフォーマンスを評価する
- インテル® MKL 関数の最初の呼び出しに必要な時間を含めない
- 安定した結果が得られるようにパフォーマンスのヒントに従う
- 対象関数をループの内部に配置する
  - 小さなサイズでは多くのループ反復が必要

```
#include <mkl.h>
#define LOOP_COUNT 20

// 最初の呼び出し、スレッド/バッファーを初期化
DGEMM(“N”, “N”, &m, &n, &k, &alpha, A, &lda, B, &ldb, &beta, C, &ldc);

// 最初の GEMM 呼出しの後に開始
double time_st = dsecnd();
for (i=0; i<LOOP_COUNT; ++i){
    DGEMM("N", "N", &m, &n, &k, &alpha, A, &lda, B, &ldb, &beta, C, &ldc);
}
double time_end = dsecnd();
double time_avg = (time_end - time_st)/LOOP_COUNT;
double gflop = (2.0*m*n*k)*1E-9;
printf("Average time: %e seconds", time_avg);
printf("GFlop/sec   : %.5f  n," gflop/time_avg);
```

小さなサイズ向けのインテル® MKL のソリューション

- インテル® MKL は小行列演算のパフォーマンスを向上するさまざまなソリューションを提供
- インテル® MKL の活用分野

インテル® MKL のソリューション	問題サイズ	機能	対応関数
MKL_DIRECT_CALL	M, N, K < 20	関数呼び出しのオーバーヘッドを最小化、エラーチェックなし、最小限のコード変更	gemm、gemm3m、syrk、trsm、axpy、dot、potrf、getrf、getrs、getri、geqrf
コンパクト API	M, N, K < 20	ベクトル化有効、関数呼び出しのオーバーヘッドを最小化	gemm、trsm、getrinp、getrfnp、potrf、geqrf
バッチ API	M, N, K < 500	並列処理を利用	gemm、gemm3m、trsm
パックド API	M または N < 500	コピーのオーバーヘッドを最小化	sgemm、dgemm

インテル® MKL 関連情報

- インテル® MKL デベロッパー・リファレンス:
  https://software.intel.com/en-us/articles/mkl-reference-manual (英語)
- インテル® MKL デベロッパー・ガイド
  - Linux*: https://www.xlsoft.com/jp/products/intel/tech/documents.html#doc-mkl
  - Windows*: https://www.xlsoft.com/jp/products/intel/tech/documents.html#doc-mkl
  - macOS*: https://software.intel.com/en-us/mkl-macos-developer-guide (英語)
- インテル® MKL 2018 リリースノート:
  https://www.xlsoft.com/jp/products/intel/perflib/mkl/2018/release_note/index.html
- インテル® MKL フォーラム: https://software.intel.com/en-us/forums/intel-math-kernel-library/ (英語)
- インテル® MKL の無料オプション: https://www.isus.jp/products/psxe/free_mkl/
- インテル® MKL-DNN: https://github.com/01org/mkl-dnn (英語)

法務上の注意書きと最適化に関する注意事項

本資料の情報は、現状のまま提供され、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。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品性に関する保証、第三者の特許権、著作権、その他、知的財産権の侵害への保証を含む) をするものではありません。

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。

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


intel
Software