インテル® Code Modernization 2018

インテル® Xeon Phi™ プロセッサー向けプログラミング

エクセルソフト株式会社

インテル® Xeon Phi™ プロセッサー
開発コード名：Knights Landing（KNL）

HPC / スーパーコンピューティング向けメニーコア・プロセッサー

- これまでのインテル® プロセッサーとのバイナリー互換性
  - 一般的な OS を起動可能 / ソフトウェアを容易に移行
- コア当たり 4 スレッド、最大 72 コア
- AVX-512 命令セットをサポート (512 ビット幅のベクトル演算)
- 高帯域幅のメモリーを内蔵（MCDRAM, 16GB）
  - メモリー帯域幅がボトルネックとなるコードに効果的
- インテル® Omni-Path ファブリックを内蔵（対応モデルのみ）

インテル® Xeon Phi™ 製品ファミリー

Knights Corner†

最初のインテル® Xeon Phi™ 製品
（インテル® MIC アーキテクチャー）

22nm プロセス

コプロセッサー（PCIe の拡張カード）

50 を超えるインテル®
アーキテクチャー・コア

1 TFLOPS 以上の理論ピーク性能

Knights Landing†

最初のインテル® Xeon Phi™ プロセッサー製品
（Knights Landing マイクロアーキテクチャー）

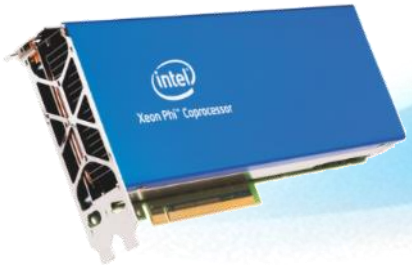
14nm プロセス

高帯域幅のオンパッケージ・メモリー

2D メッシュ・インターコネクト

3 TFLOPS 以上の理論ピーク性能

インテル® Omni-Path ファブリックの統合

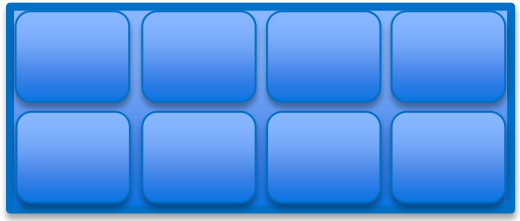


†開発コード名

インテル® Xeon® プロセッサーとの違い

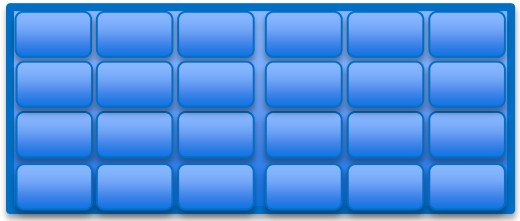
インテル® Xeon® プロセッサー

- 大きなコア：高クロック、単一スレッドの高速化



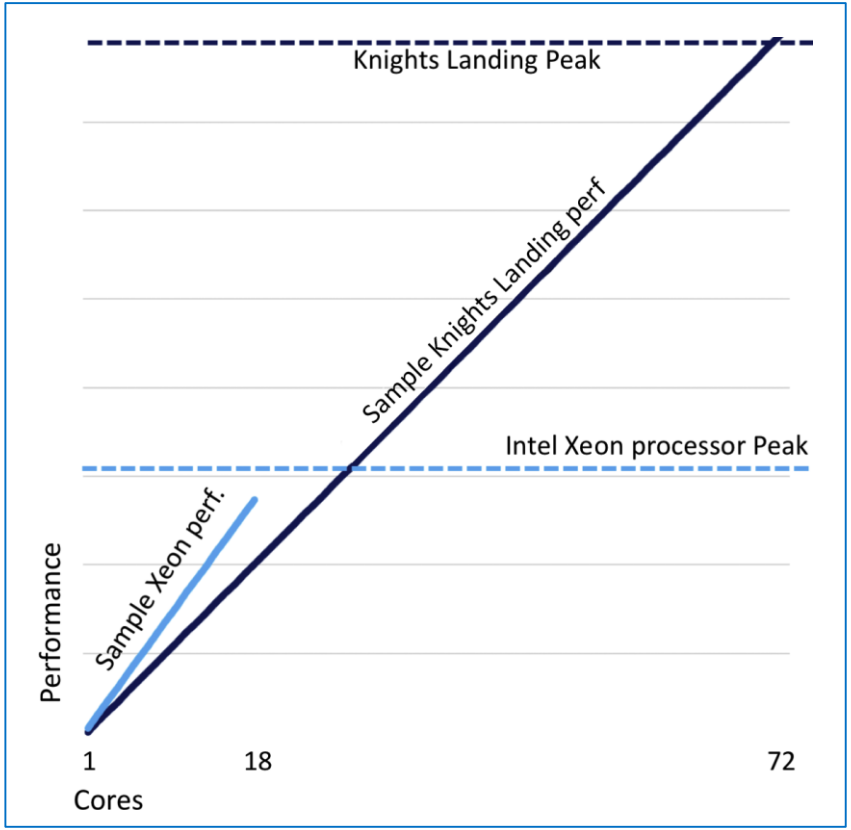
インテル® Xeon Phi™ プロセッサー

- 小さなコア：低消費電力化、幅の広いベクトル



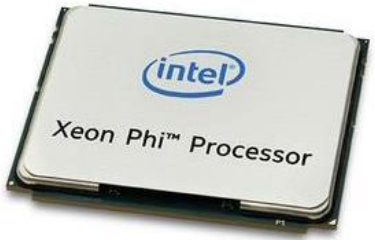
＋

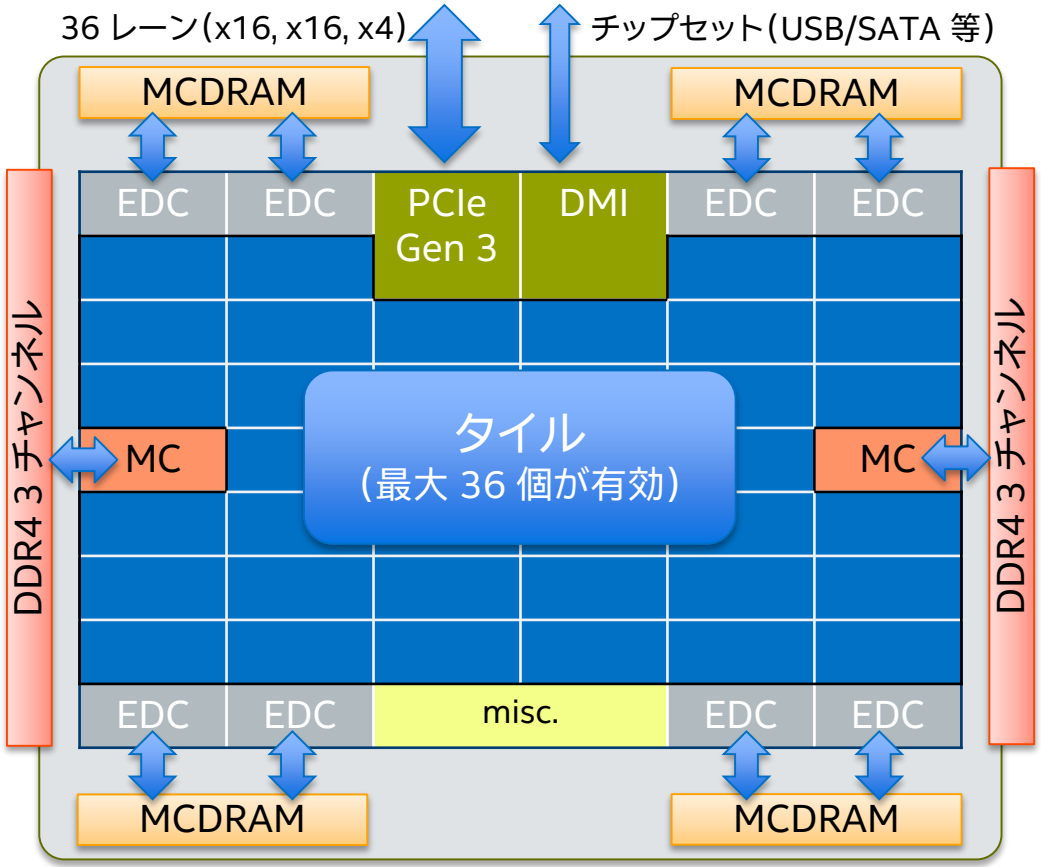
高帯域幅なメモリー
サブシステム



インテル® Xeon Phi™ プロセッサーの構造

- 最大 36 のタイル(72のコア)
- 二次元メッシュ構造の内部接続
- 内蔵の MCDRAM
  (16 GB / ~ 400 GB/s)
- DDR4 メモリーを 6チャンネル
  最大 384 GB まで接続
- 36 レーンの PCIe 3.0





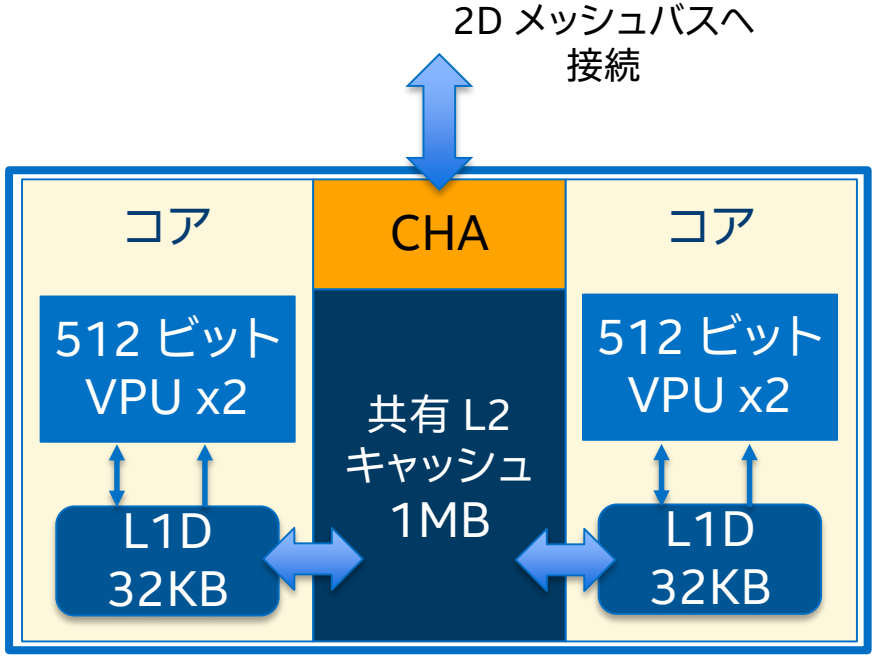
タイルとコア

タイル

- 2 つのコアと共有 L2 キャッシュ

多くの HPC 向け拡張を含む
インテル® Atom™ (Silvermont†)
プロセッサー・コア

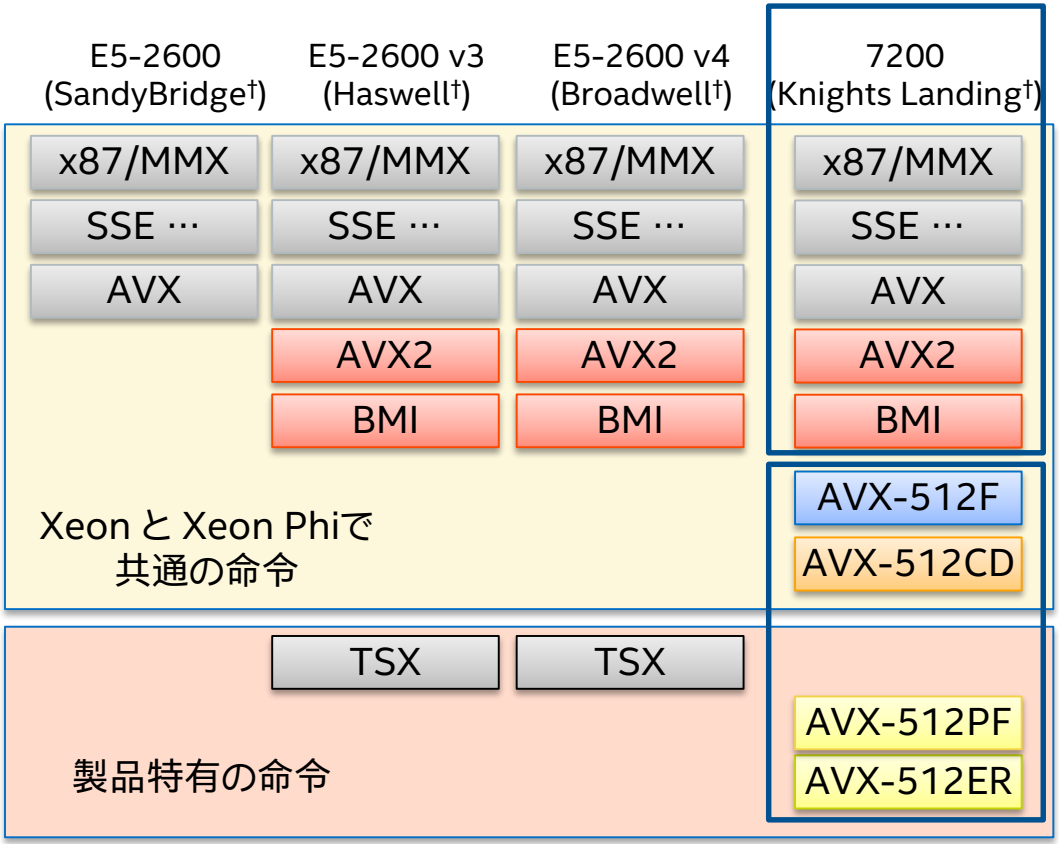
- アウトオブオーダー (72 エントリー)
- コア当たり 2 つの 512 ビット幅
  ベクトル演算器(VPU)
- コア当たり 4 つのハードウェア・スレッド
  (ハイパースレッディング)



キャッシュライン幅 = 64 Byte

†開発コード名

命令レベルの互換性



KNL は既存の命令をすべて実装

- 再コンパイル不要で動作 (バイナリー互換)

※ インテル® Xeon Phi™ コプロセッサー用のバイナリーは再コンパイルが必要

インテル® AVX-512

- 512 ビットの浮動小数点/整数ベクトル (単精度 16 要素 / 倍精度 8 要素)
- 32 のレジスターおよび 8 つのマスクレジスター
- ギャザー / スキャッターのサポート
- ベクトルの競合検出

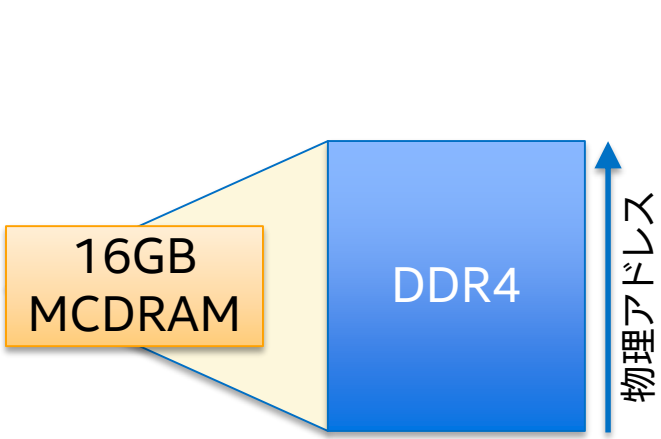
追加の拡張

- プリフェッチ / 指数と逆数

†開発コード名

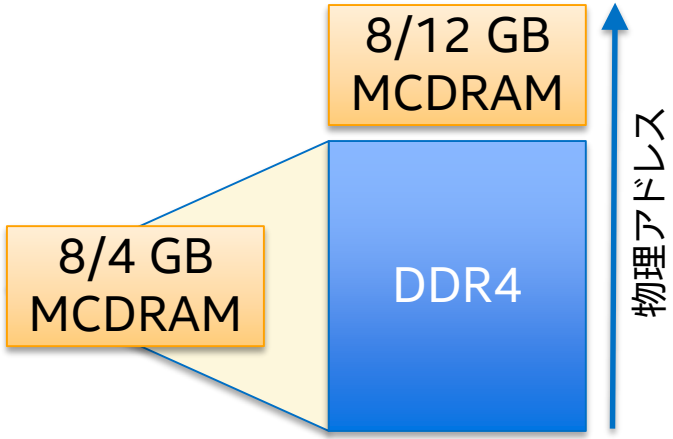
MCDRAM のメモリーモード

- 3つのタイプをシステム起動時に選択（BIOS 設定）



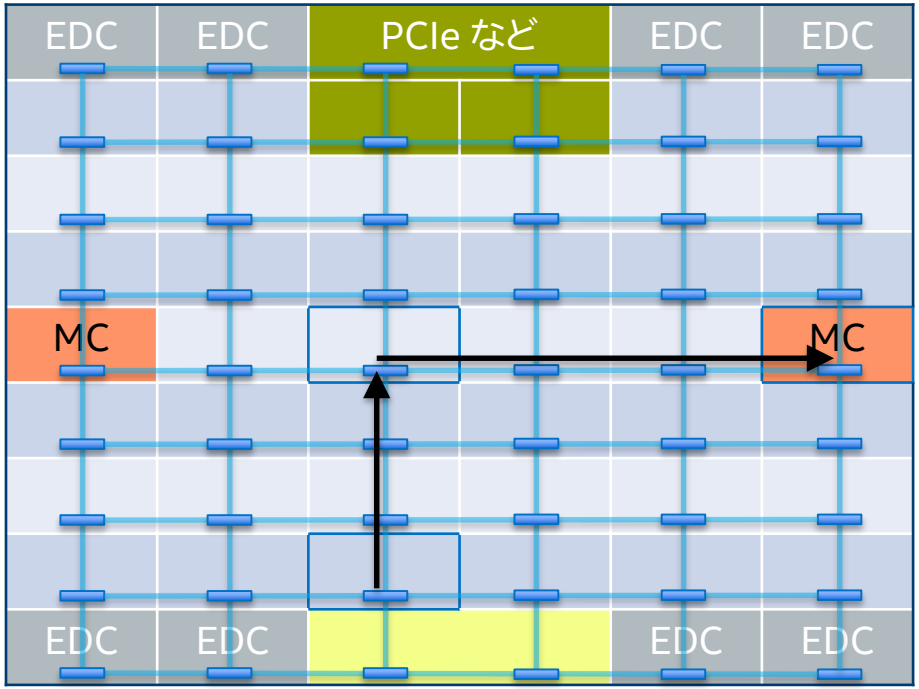
キャッシュモード
- MCDRAM は DDR4 のキャッシュとして自動的に利用される

フラットモード
- MCDRAM を割り当て可能なメモリーとする
- SW 側の対応が必要



ハイブリッドモード
- MCDRAM の一部をキャッシュ、一部を割り当て可能なメモリーとする

二次元メッシュ構造の内部接続



リングバスのメッシュ

- 各行および列が（ハーフ）リング
- YX ルーティング
  - データは Y（行方向）へ転送された後、X（列方向）へ向かう

キャッシュコヒーレントな接続

- キャッシュのタグディレクトリーは各タイルに分散配置されている

クラスターモード（BIOS 設定）

- 全対全、4分割（2分割）
- サブ NUMA クラスタリング (SNC)

numactl による MCDRAM の利用

- フラットモードにおける MCDRAM は NUMA ノードとして認識される
- numactl は NUMA システムを制御する標準的なコマンド
  - numactl --hardware でシステムで認識されている NUMA 設定を参照できる
  - numastat -m コマンドで各 NUMA ノードのメモリー利用情報を参照できる

- プログラムの総メモリー使用量が MCDRAM のサイズ(16GB)より小さい場合
  - numactl --membind=1 <実行するプログラム>

  プログラムにおけるすべてのメモリー確保が MCDRAM 上で行われる
- プログラムの総メモリー使用量が MCDRAM のサイズより大きい場合
  - numactl --preferred=1 <実行するプログラム>

  実際に使用量が MCDRAM サイズを超えた場合、残りは DDR4 が使用される

明示的な制御には、メモリー確保について別途プログラミングが必要

ソフトウェアの対応

- OS : RHEL/CentOS 7.2 以降、SLES 12 SP1 以降
  - RedHat* 社記事： https://access.redhat.com/articles/2423341
- 公開 Linux カーネルでは 3.15 以降 (インテル® AVX-512 のサポート追加)

- Intel® Xeon Phi™ Processor Software
  - KNL 用パッチ適用済み Linux カーネル (CentOS 7.2)
  - ユーティリティ
    - memkind ライブラリー (MCDRAM のプログラムからの利用に必要)
    - hwloc パッケージ

ダウンロード: https://software.intel.com/en-us/articles/xeon-phi-software

インテル® AVX-512 のコンパイラーの対応

- インテル® Fortran および C/C++ コンパイラー バージョン 15.0 以降
  - オプション: -xMIC-AVX512 (-axMIC-AVX512)
- GCC 4.9.1 (binutils 2.24) 以降
  - オプション: -mavx512f -mavx512cd -mavx512er -mavx512pf
- 対応プロセッサー以外での動作確認用エミュレーター

  「Intel® Software Development Emulator」
  https://software.intel.com/en-us/articles/intel-software-development-emulator

対応アプリケーションとパフォーマンス情報

- 様々な分野の 75 以上のアプリケーションにおいて、現行のインテル® Xeon® プロセッサーベースのシステムと比較して平均で 2 倍のパフォーマンス向上(※)を達成

※「Intel® Xeon Phi™ Processor(Codenamed Knights Landing) Software Performance Proof Points October 2017」p.5
https://software.intel.com/sites/default/files/managed/97/e8/Xeon_Phi_Marketing_Guide_Enabled_SW_Apps_Public6.pdf

使用されたシステム	Xeon (2 CPU)	Xeon Phi
プロセッサー名	インテル® Xeon® プロセッサー E5-2697 v4	インテル® Xeon Phi™ プロセッサー 7250
動作周波数	2.30 GHz	1.40 GHz
コア数 / スレッド数	18 コア / 36 スレッド	68 コア / 272 スレッド
消費電力	145 W	215 W
ソケット(CPU 数)	2 (合計 36 コア / 72 スレッド、290 W)	1
メモリー	DDR4 2400 MHz, 128 GB	MCDRAM 7.2 GT/s, 16 GB DDR4 2400 MHz, 96 GB
対応する命令セット	AVX2 (256 ビット, FMA)	AVX-512 (512 ビット, FMA)

詳細および最新情報は以下 Web ページよりご参照ください
https://software.intel.com/en-us/xeon-phi-apps

インテル® Xeon Phi™ プロセッサー x200 製品ファミリー

プロセッサー・ナンバー	コア数/スレッド数	動作周波数（GHz）	最大 TDP/消費電力（W）	内蔵メモリー	対応メモリー	L2 キャッシュ合計 (MB)	ファブリック
7290	72/288	1.50	245	16 GB 7.2 GT/s	DDR4-2400	36	なし
7250	68/272	1.40	215	16 GB 7.2 GT/s	DDR4-2400	34	なし
7230	64/256	1.30	215	16 GB 7.2 GT/s	DDR4-2400	32	なし
7210	64/256	1.30	215	16 GB 6.4 GT/s	DDR4-2133	32	なし
7290F	72/288	1.50	260	16 GB 7.2 GT/s	DDR4-2400	36	あり
7250F	68/272	1.40	230	16 GB 7.2 GT/s	DDR4-2400	34	あり
7230F	64/256	1.30	230	16 GB 7.2 GT/s	DDR4-2400	32	あり
7210F	64/256	1.30	230	16 GB 6.4 GT/s	DDR4-2133	32	あり

参照：https://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-processor-product-brief.html

インテル® Parallel Studio XE
最先端の並列コードを開発するためのツール

ハイパフォーマンスでスケーラブルな HPC、エンタープライズ、およびクラウド・アプリケーションを作成

- インテルのハードウェアとパフォーマンスの能力を最大限に利用
- インテル® Xeon® スケーラブル・プロセッサーとインテル® Xeon Phi™ プロセッサー向けの**インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX-512)** を使用して一貫したプログラミングを実現
- 最新のベクトル化、マルチスレッド、マルチノード、およびメモリーの最適化手法を利用して**簡単にコードを現代化**
- 最先端のコンパイラー、数値ライブラリー、パフォーマンス・プロファイラーにより**最新のハードウェア向けにソフトウェアを確実に最適化**
- **プライオリティー・サポート**によりインテルのエンジニアに技術的な質問を直接問い合わせ[1]

C、C++、Fortran、および **Python*** ソフトウェア開発者を支援

標準規格に基づく並列モデルを使用: OpenMP*、MPI、インテル® TBB

[1]ライセンス購入者のみ、サポートサービス更新時には割引も提供。学生/アカデミック向けにインテル® ソフトウェア・ツールの割引または無料版 (英語) も用意

インテル® Parallel Studio XE のコンポーネント
包括的なソフトウェア開発ツールスイート

Composer Edition

ビルド
コンパイラーとライブラリー

インテル® C/C++ コンパイラー
最適化コンパイラー

インテル® MKL[2]

インテル® Fortran コンパイラー
最適化コンパイラー

インテル® IPP[3]
画像、信号、データ処理

インテル® TBB[1]
C++ スレッド・ライブラリー

インテル® DAAL[4]

インテル® Distribution for Python*
ハイパフォーマンスなスクリプト

Professional Edition

解析
解析ツール

インテル® VTune™ Amplifier
パフォーマンス・プロファイラー

インテル® Inspector
メモリー/スレッドの デバッガー

インテル® Advisor
ベクトル化の最適化と スレッドのプロトタイプ生成

Cluster Edition

スケール
クラスターツール

インテル® MPI ライブラリー
メッセージ・パッシング・インターフェイス・ライブラリー

インテル® Trace Analyzer & Collector
MPI チューニングと解析

インテル® Cluster Checker
クラスター診断エキスパート・システム

インテル® アーキテクチャー・ベースのプラットフォーム

オペレーティング・システム: Windows*、Linux*、macOS[5]*

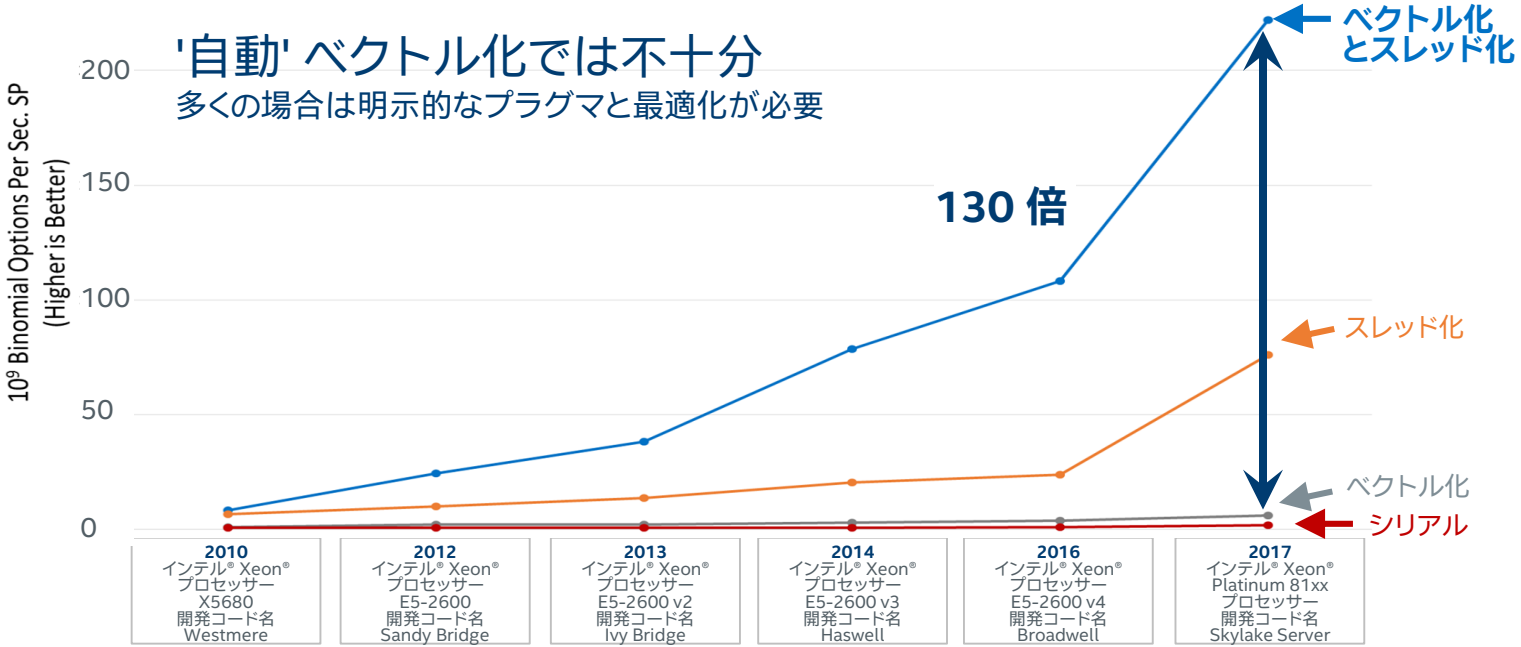
コードを強力に支援 - isus.jp/intel-parallel-studio-xe/

[1] インテル® スレッディング・ビルディング・ブロック
[2] インテル® マス・カーネル・ライブラリー
[3] インテル® インテグレーテッド・パフォーマンス・プリミティブ
[4] インテル® データ・アナリティクス・アクセラレーション・ライブラリー
[5] Composer Edition のみ提供

ベクトル化およびスレッド化するか、パフォーマンスを諦めるか

スレッド化 + ベクトル化はどちらか一方よりもはるかに高速

新しいハードウェア世代ごとにパフォーマンスが向上



2010 - 2016 のシステム構成については、補足資料の「ベクトル化およびスレッド化するか、パフォーマンスを諦めるか」にあるベンチマークを参照してください。ベンチマークの出典: インテル社内での測定値。
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。
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

インテル® Xeon Phi™ プロセッサー（Knights Landing† : KNL）向けプログラミング

- インテル® Xeon Phi™ プロセッサーの演算リソース
  - 512 ビット幅のベクトル演算ユニットを搭載した多数のコア

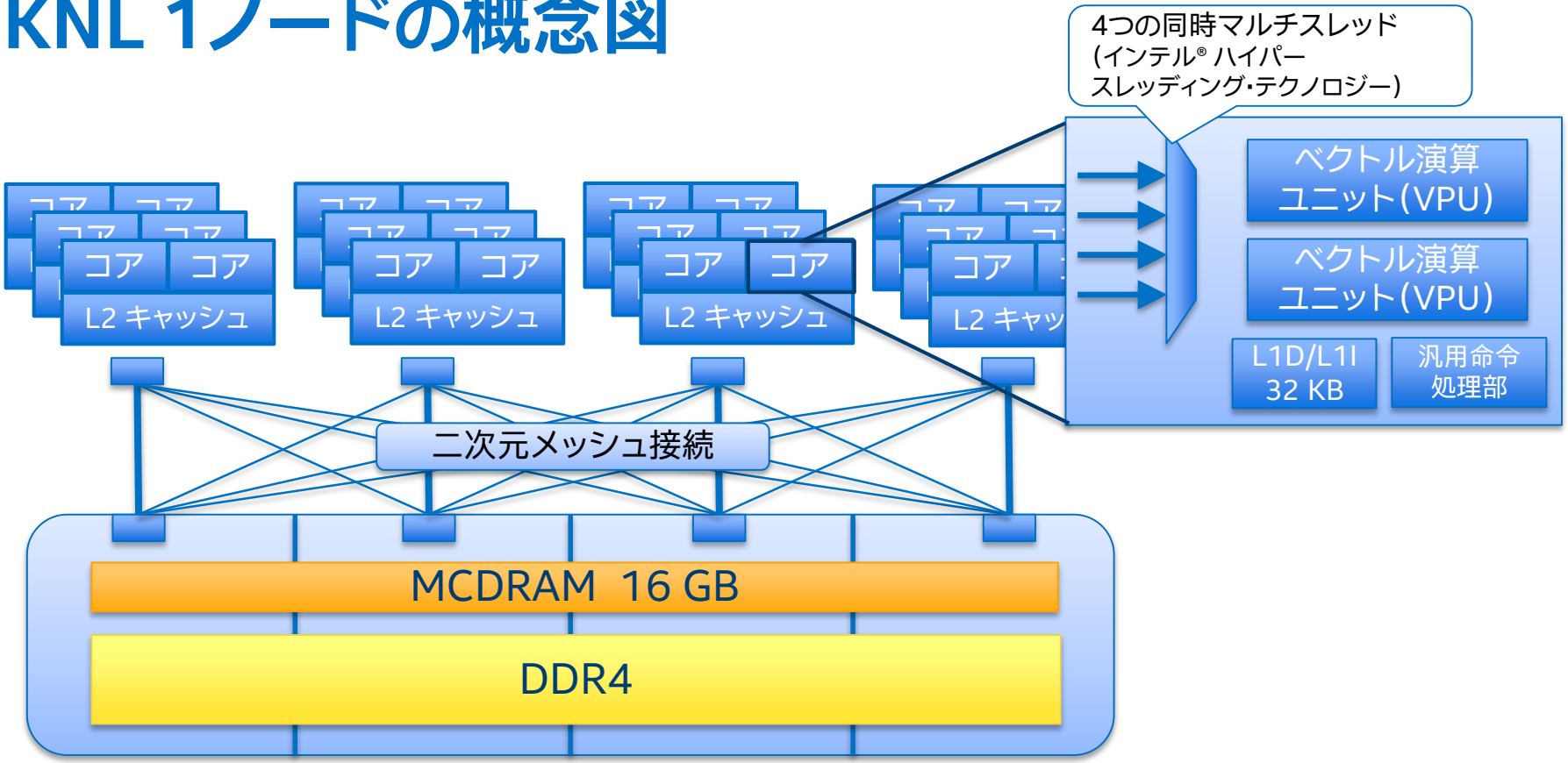
    並列度の高いアプリケーションに対応する高度な並列ハードウェア
    インテル® Xeon® プロセッサー向け最適化を再利用

- メモリーサブシステム
  - 高帯域幅のメモリー（MCDRAM）とクラスターモード

    アプリケーションの特性やチューニング度合いに応じて
    暗黙的（環境設定）または明示的（プログラミング）な方法を適用

† 開発コード名

KNL 1ノードの概念図

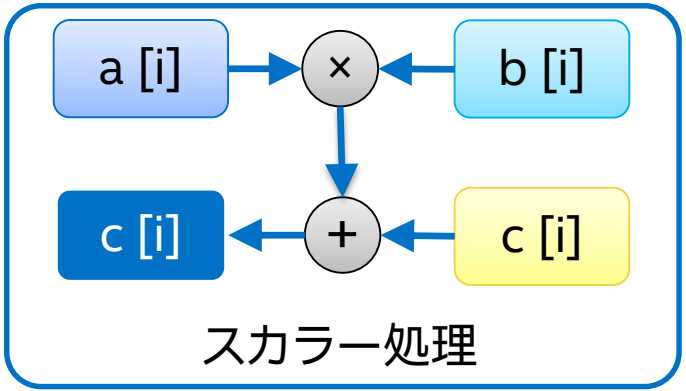


ベクトル / SIMD 演算

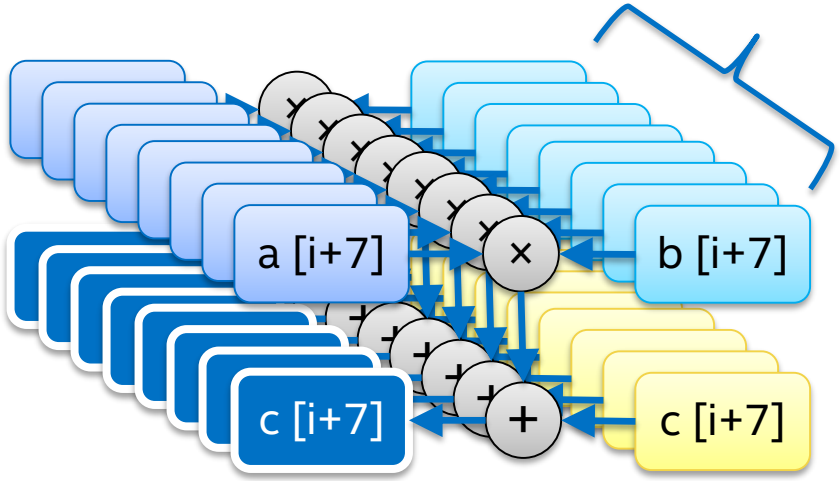
- 1回の演算で複数のデータ要素を同時に処理する

```
for (i=0; i>MAX; i++)
    c[i] += a[i] * b[i];
```

元になるプログラム



スカラー処理



ベクトル処理

ハードウェアの対応：SIMD 拡張命令セット、
ベクトル演算ユニット / ベクトルレジスター

ベクトル演算の活用

- インテル® コンパイラーによるベクトル化
  - ループの自動ベクトル化 (-x または -ax オプション)
  - 最適化レポート(-qopt-report オプション)でベクトル化状況を確認
  - OpenMP 4.0 以降の SIMD 構文による明示的なベクトル化指定

- インテルのパフォーマンス・ライブラリーを適用
  - インテル® マス・カーネル・ライブラリー (インテル® MKL)
    線形代数、線型方程式、高速フーリエ変換、統計処理、補間など
  - インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP)
    画像処理、信号処理

インテル® AVX-512

- インテル® AVX-512F : 基本命令
  - 多くのインテル® AVX2 命令の 512 ビット・レジスター拡張
  - マスク付き操作、ギャザー / スキャッターをサポート
- インテル® AVX-512CD : 競合検出命令
  - ベクトルレジスター内の競合検出

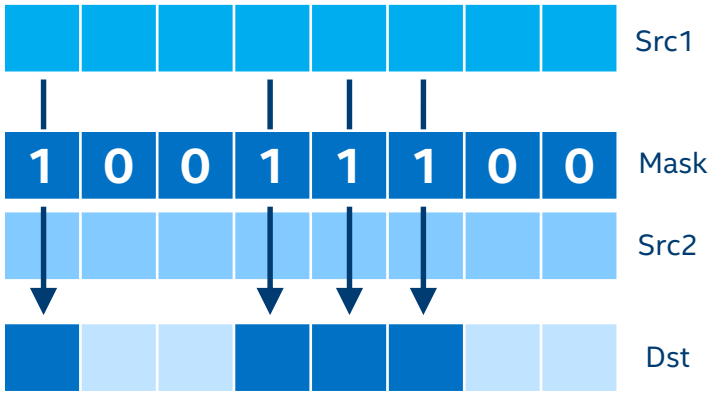
- インテル® AVX-512ER : 指数及び逆数命令
  - 指数 (exp2)、逆数 (1/x)、平方根の逆数 (1/√x) のベクトルバージョン
- インテル® AVX-512PF : プリフェッチ命令
  - ギャザー / スキャッター用のプリフェッチ

ベクトル化に役立つ命令機能 (1)

インテル® AVX-512F：将来のインテル® Xeon® プロセッサーでも有効

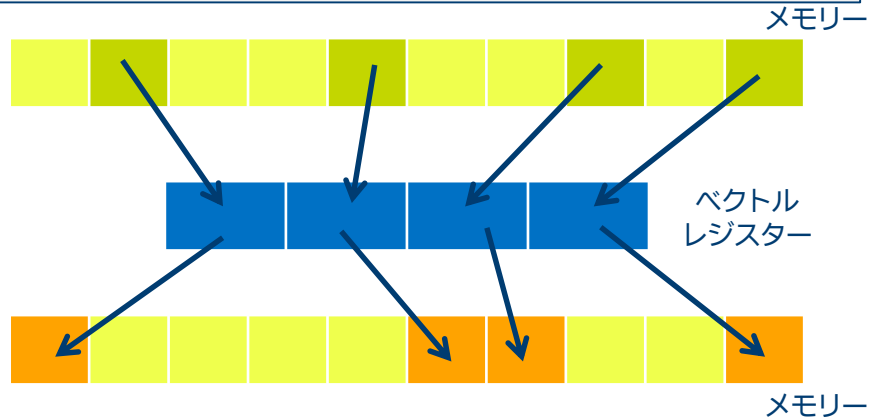
- マスクレジスター

```
for (i=0; i<N; i++){
        if ( x[i] > 0 ){  x[i] +=  … ;  }
}
```



- ギャザー / スキャッター

```
for (i=0; i<N; i++){
        tmp[i]  =  x[ idx[i] ];   //  ギャザー
        y[ idx[i] ]  = a*tmp[i] + b;  // スキャッター
}
```



ベクトル化に役立つ命令機能 (2)

インテル® AVX-512CD：将来のインテル® Xeon® プロセッサーでも有効

- 依存関係が曖昧なループのベクトル化

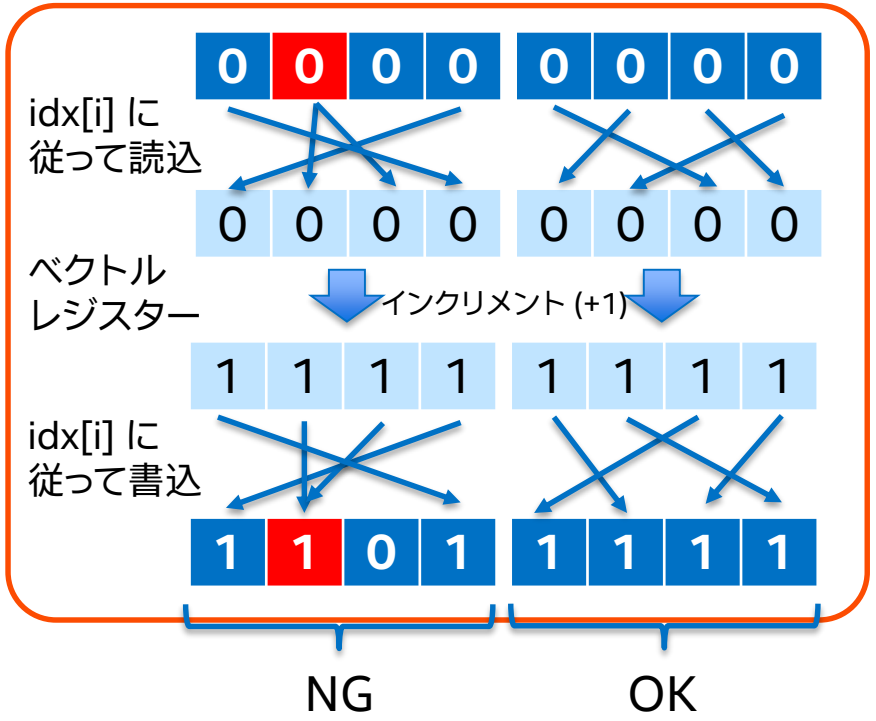
```
for (i=0; i<N; i++){
      x[ idx[i] ] = x[ idx[i] ] + 1;
}
```

idx[i] (i=0~7) | 3 | 1 | 1 | 0 | 6 | 4 | 7 | 5 |

計算前の x[] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

計算後の x[] | 1 | 2 | 0 | 1 | 1 | 1 | 1 | 1 |



インテル® AVX-512 はベクトル化を促進
幅の広いベクトルをできる限り活用

- コンパイラーレポート (一部)
  - インテル® AVX2 によるベクトル化 (オプション -xCORE-AVX2)

リマーク #15344: ループ はベクトル化されませんでした: ベクトル依存関係がベクトル化を妨げています。...
リマーク #15346: ベクトル依存関係: FLOW の依存関係が h[ih] (17:9) と h[ih] (17:9) の間に仮定されました。

  - インテル® AVX-512 によるベクトル化 (オプション -xMIC-AVX512)

リマーク #15300: ループがベクトル化されました。
...
リマーク #15482: ベクトル化された算術ライブラリーの呼び出し: 2
リマーク #15499: ヒストグラム: 1

```
/* sin(x) のヒストグラムを配列 h に取る */

for (i=0; i<n_event; i++) {     // C/C++
   y    =  sinf( x[i] * twopi );
   ih   =  floor( (y-bot)*invbinw );
   ih   =  ih > 0    ? Ih : 0;
   ih   =  ih < n_bin ? ih : n_bin;
   h[ih]  =  h[ih] + 1;
}
```

コードの主要な計算ループ

参考 : インテル® AVX-512 で向上したベクトル化のパフォーマンス
https://www.isus.jp/products/psxe/pu27-04-vectorization-opportunities/

ヒストグラムループのスピードアップ

- インテル® AVX2 によるベクトル化
  (オプション -xCORE-AVX2)

計算時間: およそ 35.5 秒

- インテル® AVX-512 によるベクトル化
  (オプション -xMIC-AVX512)

計算時間: およそ 3.72 秒

```
/* sin(x) のヒストグラムを配列 h に取る */

for (i=0; i<n_event; i++) {     // C/C++
   y    =  sinf( x[i] * twopi );
   ih   =  floor( (y-bot)*invbinw );
   ih   =  ih > 0    ? Ih : 0;
   ih   =  ih < n_bin ? ih : n_bin;
   h[ih]  =  h[ih] + 1;
}
```

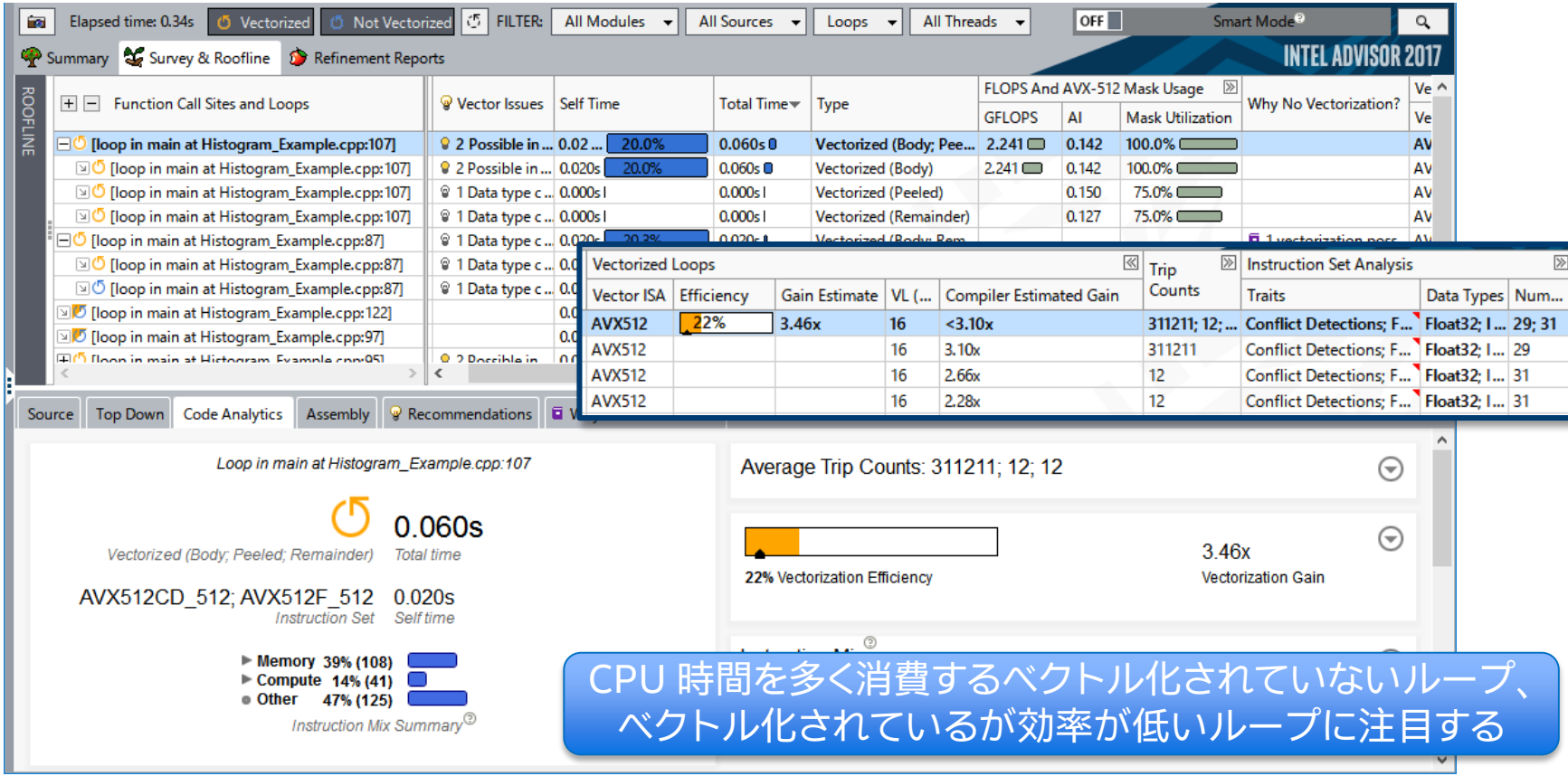
コードの主要な計算ループ

パフォーマンスは 9.54 倍に

ih と h[] は int (32bit 整数)、y と x[] は float (32bit 浮動小数点数)
n_event = 50,000,000 、n_bin = 200、10 回繰り返し

利用システム: インテル® Xeon Phi™ プロセッサー 7210 1.30GHz 64 コア
CentOS* 7.3、インテル® コンパイラー 18.0.1.163 (インテル® Parallel Studio XE 2018 Update 1)

インテル® Advisor によるループ/ベクトル化解析

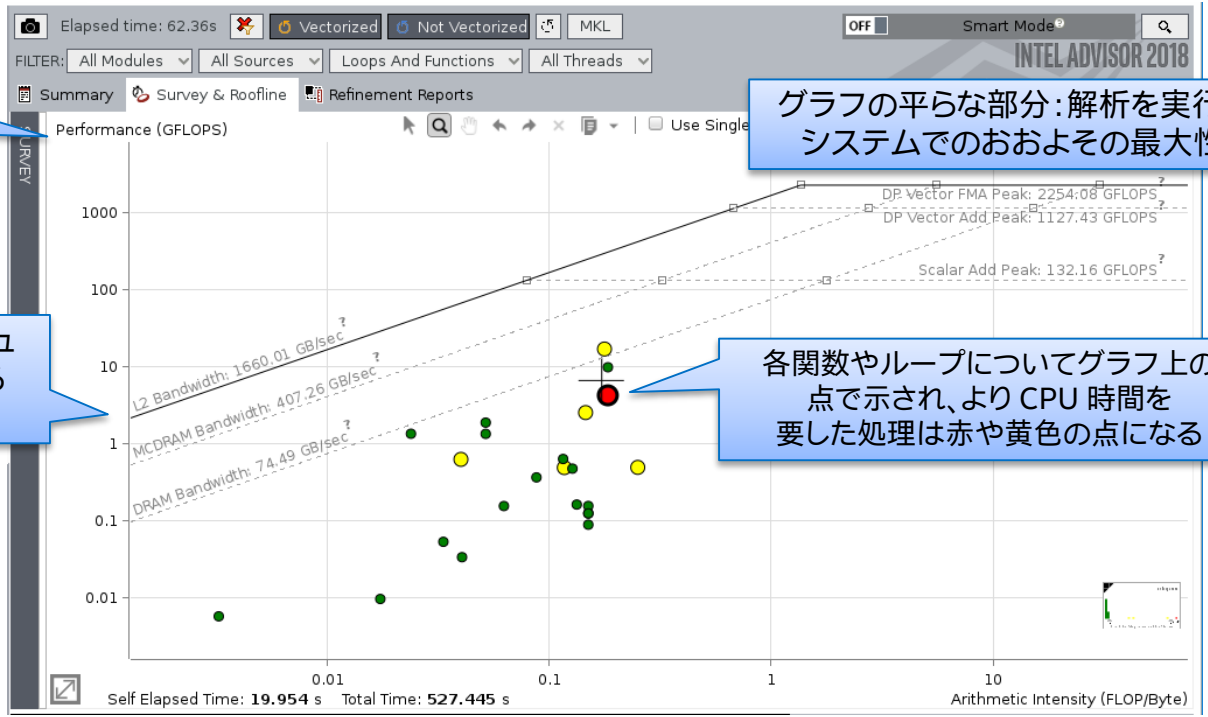


CPU 時間を多く消費するベクトル化されていないループ、
ベクトル化されているが効率が低いループに注目する

インテル® Advisor 2018 : ルーフライン解析
パフォーマンスを視覚化

縦軸: GFLOPS
時間当たりの計算回数

グラフの傾いた部分: キャッシュ
またはメモリーの帯域幅による
制限を考慮した最大性能



グラフの平らな部分:解析を実行した
システムでのおおよその最大性能

各関数やループについてグラフ上の
点で示され、より CPU 時間を
要した処理は赤や黄色の点になる

横軸: 演算強度(AI):計算回数とメモリー転送バイト数の比率

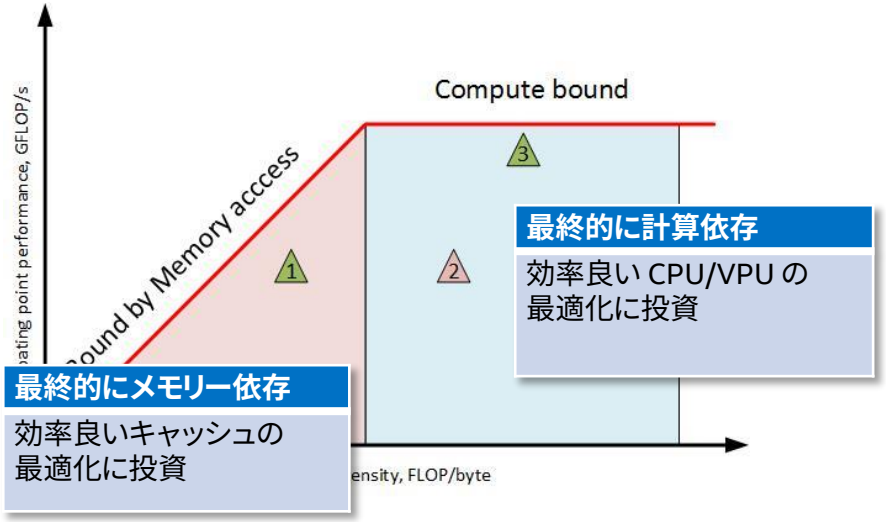
ルーフライン・データを解釈する

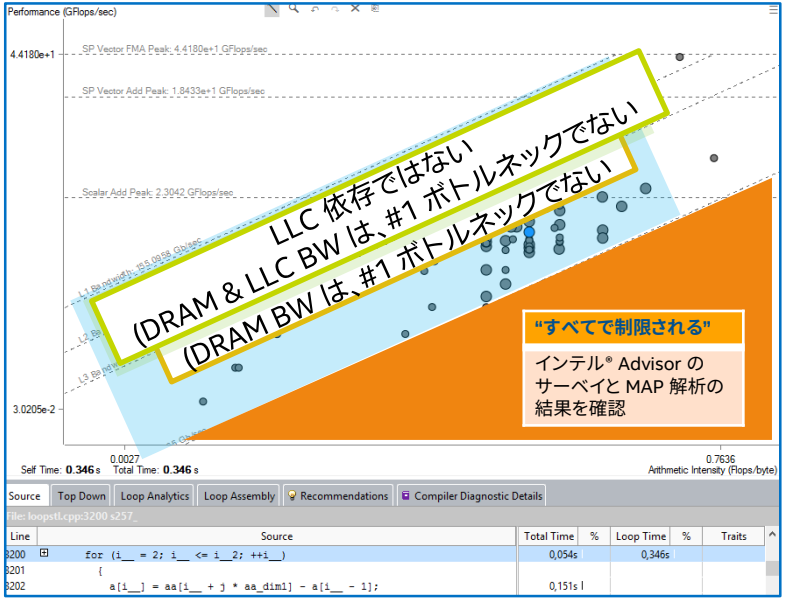
最終的な限界

(完全な最適化を想定)
長期に渡る ROI、最適化の方針

現在の限界

(現状のボトルネックは何か)
次のステップ、最適化の対策



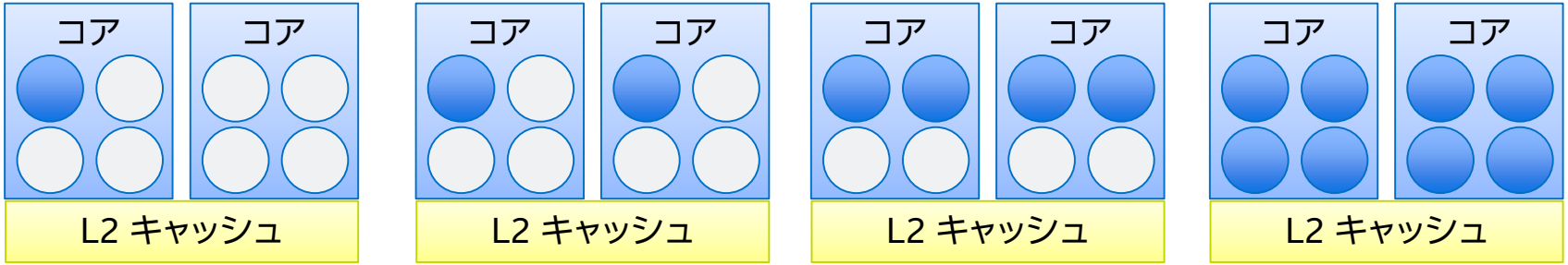


並列化は引き続き重要

- (前世代に比べてパフォーマンスは向上したが、)
  単一コアは依然としてインテル® Xeon® プロセッサーのコアよりも低速
- 実装には既存の並列プログラミング・モデルが適用可能
  - Pthread、OpenMP*、インテル® TBB、MPI、CoArray Fortran など
- 並列化パターン
  - 「構造化並列プログラミング—効率良い計算を行うためのパターン」
    著者: マイケル・マックール / アーク・D・ロビソン / ジェームス・レインダース(共著)、訳者: 菅原 清文 / エクセルソフト株式会社(共訳)、2013年、株式会社カットシステム、ISBN 978-4-87783-305-3

スレッドの制御

- コア当たり4つまでのスレッド(または MPI プロセス)を割り当て可能
  - コア当たりの最適な割り当て数はアプリケーションに依存
- インテルの OpenMP 実装およびインテル® MPI ライブラリーでは実行時の割り当て指定が可能



スレッドの制御(続き)

	インテルの OpenMP での環境設定 (全体で64コアの場合)
タイル/1スレッド	OMP_PLACES="threads(256)" OMP_NUM_THREADS=32 ./a.out
コア/1スレッド	KMP_HW_SUBSET=64c,1t KMP_AFFINITY=compact ./a.out
コア/2スレッド	KMP_HW_SUBSET=64c,2t KMP_AFFINITY=compact ./a.out
コア/4スレッド	KMP_HW_SUBSET=64c,4t KMP_AFFINITY=compact ./a.out

	インテル® MPI での環境設定 (全体で64コアの場合)
タイル/1プロセス	mpirun -n 32 ./a.out
コア/1プロセス	mpirun -n 64 ./a.out
コア/2プロセス	mpirun -n 128 ./a.out
コア/4プロセス	mpirun -n 256 ./a.out

環境変数に I_MPI_DEBUG=4 以上を
指定することで割り当て状況を確認できる

参考:「インテル® MPI ライブラリー: インテル® Xeon® プロセッサー、インテル® Xeon Phi™ コプロセッサー、インテル® Xeon Phi™ プロセッサー間の互換性」
https://www.isus.jp/products/c-compilers/mpi-compatibility-among-xeon-xeon-phi-x100/

スレッド配置を制御する環境変数(OpenMP)

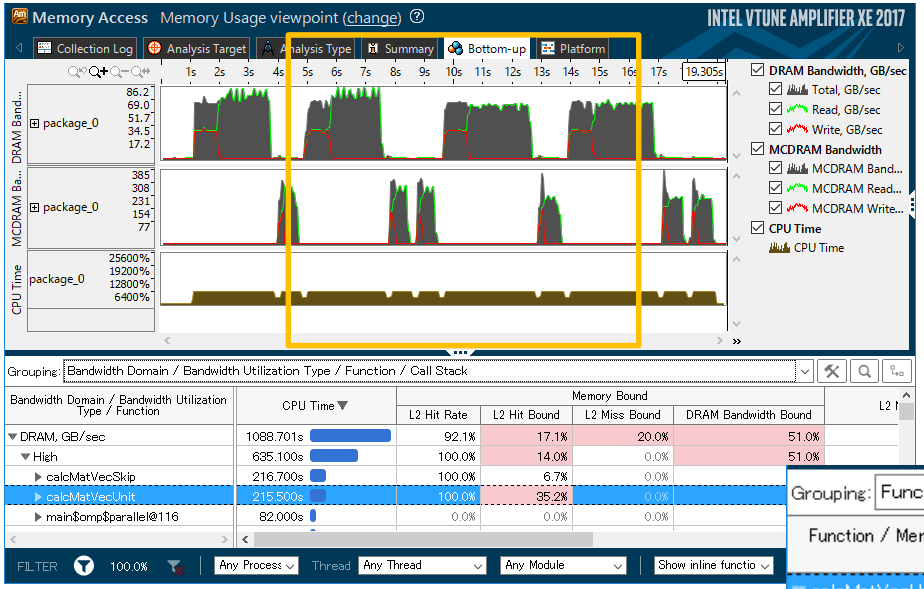
インテル の OpenMP 固有

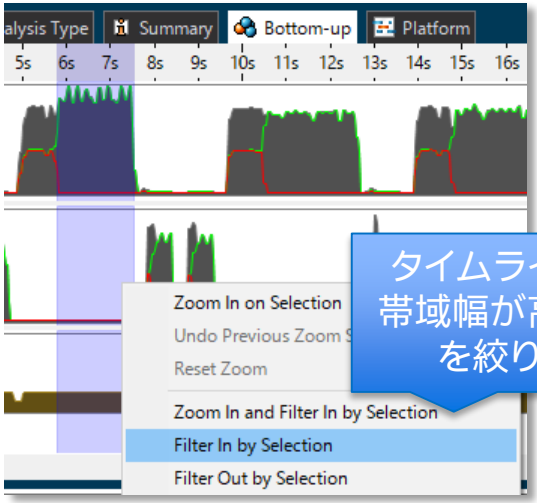
- KMP_HW_SUBSET (v17.0 以降、旧 KMP_PLACE_THREADS)
  - 実行に使用するコア数と、コア当たりのスレッド数を指定
  - 例: 32 コア、コア当たり 2 スレッド
    KMP_HW_SUBSET=32c,2t ./a.out

- KMP_AFFINITY
  - スレッドの割り当てルールを指定する
    (KNL のデフォルトは scatter : なるべく異なるコアへスレッドを割り当てる)
  - KMP_AFFINITY=verbose を指定すると割り当て状況が報告される

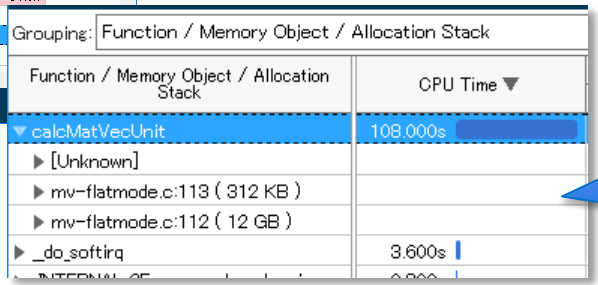
参考:「インテル® C++ コンパイラー 17.0 デベロッパー・ガイドおよびリファレンス ー スレッド割り当ての制御」
https://jp.xlsoft.com/documents/intel/compiler/17/cpp_17_win_lin/GUID-08389D3D-1619-403B-8A75-0965D24219A1.html

インテル® VTune™ Amplifier によるメモリー解析
DDR4/MCDRAM の帯域幅利用状況を確認





タイムラインから
帯域幅が高い時間
を絞り込む



帯域幅を使用した
関数や、アクセス
された配列が分かる

メモリーモードとクラスターモード

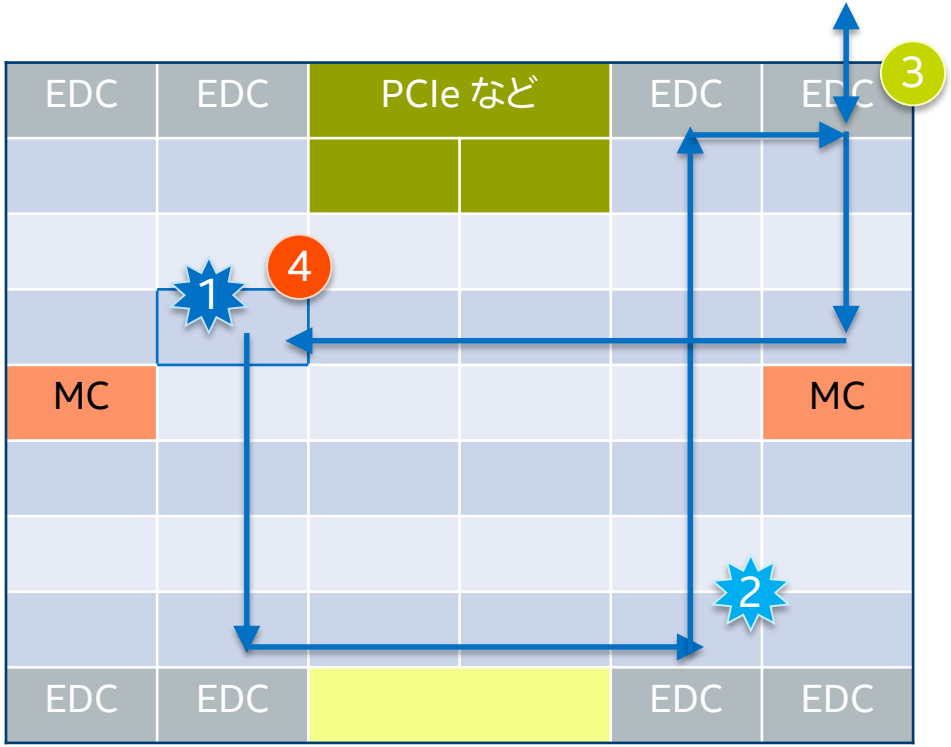
	キャッシュモード (MCDRAM はキャッシュ)	フラットモード (MCDRAM は独立のメモリー)
全対全 または 4分割	フラットな並列プロセッサーと 16 GB のキャッシュ	MCDRAM と DDR4 の 使い分けが必要
SNC4	NUMA な並列プロセッサー (4 CPUのシステムに似る)	NUMA ノードそれぞれで 区分された MCDRAM と DDR4

- ✓ より大きな問題(16 GB 以上のメモリー利用)へ適用する場合は
  MCDRAM をキャッシュとして扱う

- ✓ NUMA (非対称メモリーアクセス):
  共有メモリー型(スレッド)並列プログラムは
  対称メモリーアクセス(UMA)を想定するため、対応が複雑化する

参考:「NUMA 向けのアプリケーションの最適化」http://www.isus.jp/hpc/optimize-for-numa/

クラスターモード: 全対全(All-to-All)

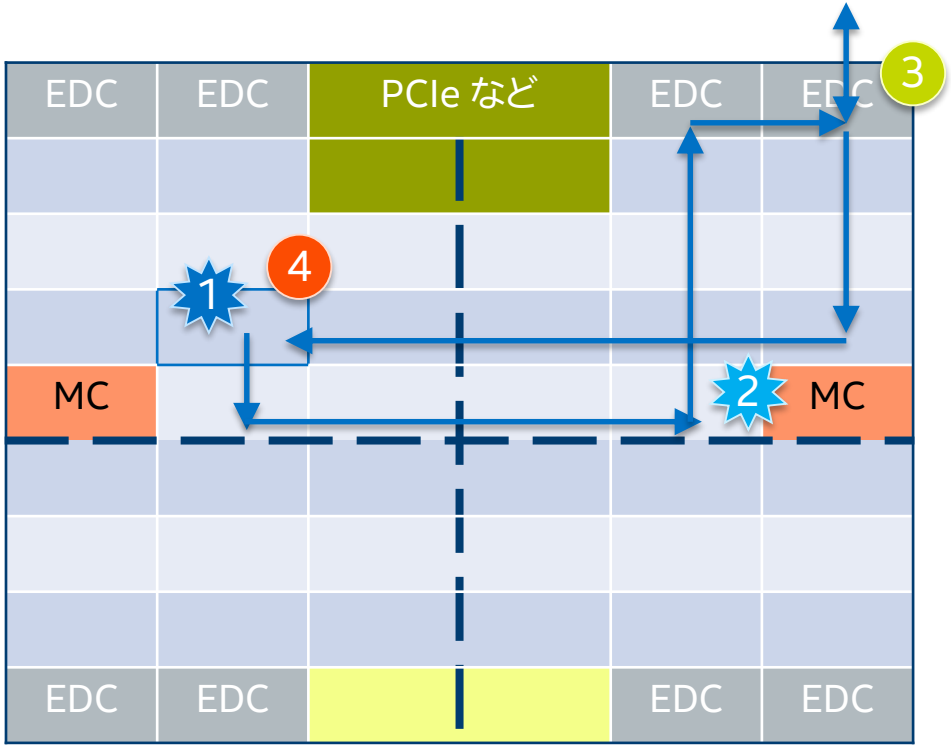


アドレスは一様にハッシュされ、
特定のコア、タグ・ディレクトリーと
メモリーの関連付けは無い

あるタイルからのメモリー要求
(読込/書込)に対する振る舞い

1. L2 キャッシュミスが発生
2. 該当するアドレスを管理するタイルへ、
   キャッシュ状況を問い合わせる
3. どのタイルにもキャッシュされて
   いなければ、メモリーからデータ読込
4. メモリーを要求したタイルへ
   データが送られる

クラスターモード: 4分割(Quadrant)

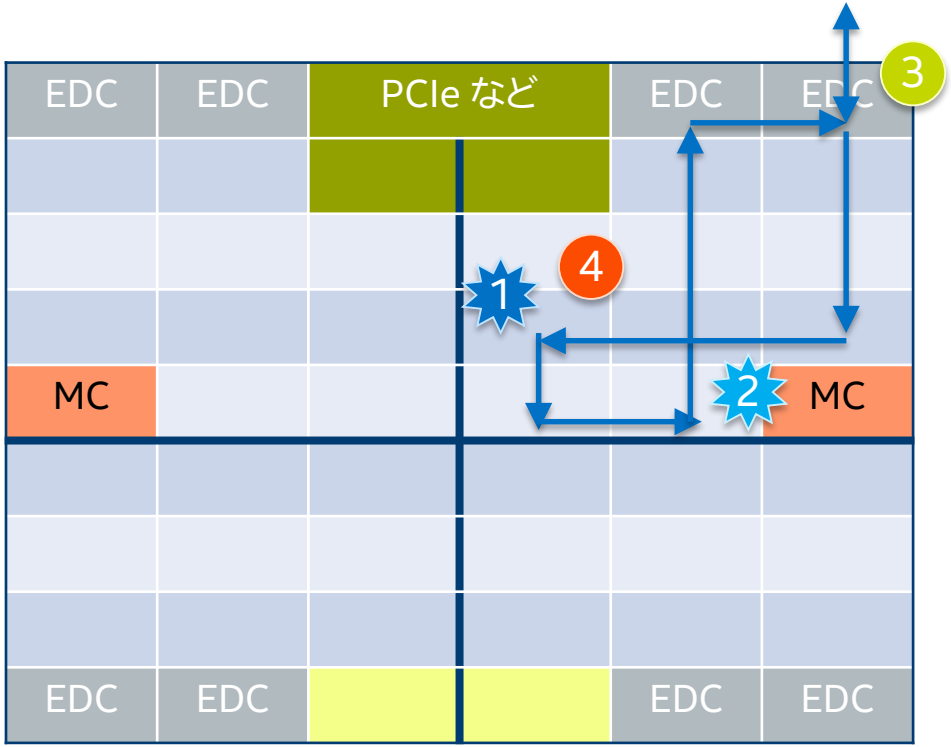


プロセッサーを4つに区切る
(ソフトウェアからは見えない)

あるアドレスについての
タグ・ディレクトリーとメモリーが
同じクアドラント上に置かれる

全対全に比べて、平均的に
低いレイテンシー/高い帯域幅で
メモリーへアクセス可能

クラスターモード:サブ NUMA クラスタリング (SNC4)



プロセッサーを4つに区切り、NUMA ノードとする
(4 CPU のインテル® Xeon® プロセッサーベースのサーバーに似た構造となる)

タイル、タグ・ディレクトリー、メモリーそれぞれが関連付けされた状態

同じNUMAノード内のメモリーアクセスであれば、他のモードよりレイテンシーを低く保つことが可能となる

ソフトウェアで NUMA 用の調整が必要

参考:「NUMA 向けのアプリケーションの最適化」http://www.isus.jp/hpc/optimize-for-numa/

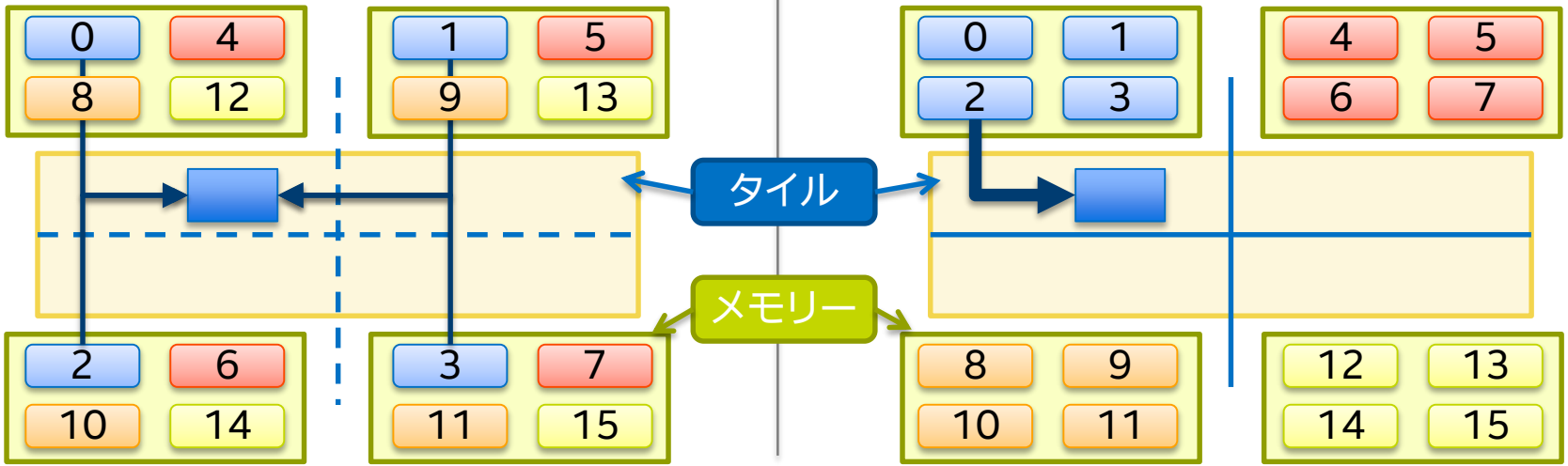
クラスターモードのメモリーアドレス配置

連続した
データブロック

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15

4分割：メモリーアドレスはインターリーブ

SNC4：メモリーアドレスは
各 NUMA ノード上でインターリーブ



メモリーモードとクラスターモードの確認

- numactl --hardware (または -H) コマンド

キャッシュモード + 4分割

```
$ numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 … 253 254 255
node 0 size: 49026 MB
node 0 free: 45821 MB
node distances:
node   0
  0:  10
```

フラットモード + 4分割

```
$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 … 253 254 255
node 0 size: 49026 MB
node 0 free: 45049 MB
node 1 cpus:
node 1 size: 16384 MB
node 1 free: 15879 MB
node distances:
node   0   1
  0:  10  31
  1:  31  10
```

システム情報:
インテル® Xeon Phi™ プロセッサー 7210 (64 コア)
搭載メモリー(DDR4) : 48 GB (8 GB x 6)
CentOS 7.2 + XPPSL 1.4.2

続き

キャッシュモード + SNC4

```
$ numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 1 2 … 205 206 207
node 0 size: 12162 MB
node 0 free: 11065 MB
node 1 cpus: 16 17 … 221 222 223
node 1 size: 12288 MB
node 1 free: 11662 MB
node 2 cpus: 48 49 … 253 254 255
node 2 size: 12288 MB
node 2 free: 11845 MB
node 3 cpus: 32 33 … 237 238 239
node 3 size: 12288 MB
node 3 free: 11267 MB
node distances:
node   0   1   2   3
  0:  10  21  21  21
  1:  21  10  21  21
  2:  21  21  10  21
  3:  21  21  21  10
```

フラットモード + SNC4

```
$ numactl --hardware
available: 8 nodes (0-7)
node 0 cpus: 0 1 2 … 205 206
207
node 0 size: 12162 MB
node 0 free: 11157 MB
node 1 cpus: 16 17 18 … 222 223
node 1 size: 12288 MB
node 1 free: 11844 MB
node 2 cpus: 48 49 50 … 254 255
node 2 size: 12288 MB
node 2 free: 11567 MB
node 3 cpus: 32 33 34 … 238 239
node 3 size: 12288 MB
node 3 free: 11268 MB
node 4 cpus:
node 4 size: 4096 MB
node 4 free: 3994 MB
```

~ 右上に続く

```
node 5 cpus:
node 5 size: 4096 MB
node 5 free: 3994 MB
node 6 cpus:
node 6 size: 4096 MB
node 6 free: 3994 MB
node 7 cpus:
node 7 size: 4096 MB
node 7 free: 3992 MB
node distances:
node   0   1   2   3   4   5   6   7
  0:  10  21  21  21  31  41  41  41
  1:  21  10  21  21  41  31  41  41
  2:  21  21  10  21  41  41  41  31
  3:  21  21  21  10  41  41  31  41
  4:  31  41  41  41  10  41  41  41
  5:  41  31  41  41  41  10  41  41
  6:  41  41  41  31  41  41  10  41
  7:  41  41  31  41  41  41  41  10
```

MCDRAM の利用方法(1)

- numactl コマンドによる割り当て
  --membind または -m : 常に割り当て
  --preferred または -p : 可能ならば割り当て

```
numactl  --membind  1  ./a.out
```

```
numactl  --preferred  1  ./a.out
```

- AutoHBW ライブラリーによる半自動化 (malloc() 呼び出しのフック)

✓ AutoHBW ライブラリーからのメッセージ表示(-1, 0, 1, 2, 3)
  AUTO_HBW_LOG=1

✓ MCDRAM から確保するメモリー量の下限値/上限値のバイト数
  AUTO_HBW_SIZE=min_size[:max_size]

```
export AUTO_HBW_LOG=1
export AUTO_HBW_SIZE=1024k
LD_PRELOAD=libautohbw.so  ./a.out
```

AutoHBW ライブラリーについて
https://software.intel.com/en-us/articles/using-autohbw-with-jemalloc-and-memkind-library

MCDRAM の利用方法(2)

- hbw_malloc (memkind ライブラリー)を使用する

```
#include <hbwmalloc.h>
unsigned int size = sizeof(float) * 1024;
float *ptr, *ptrA;    int ret;

ptr = hbw_malloc(size);
ret = hbw_posix_memalign(&ptrA, 64, size);

hbw_free(ptr);   hbw_free(ptrA);
```

```
icc -lmemkind  foo.c
```

インテル® Fortran コンパイラーでは FASTMEM 指定で同じことが可能

```
real, allocatable :: array
!dir$ attributes fastmem :: array

allocate (array(1024))
```

```
ifort -lmemkind  foo.f90
```

memkind ライブラリーについて(英語) : http://memkind.github.io/memkind/

MCDRAM の利用 : インテル® MKL

- インテル® MKL の関数は、可能であれば自動的に MCDRAM からメモリーを確保する

```
#include <mkl.h>
unsigned int size = sizeof(float) * 1024;
float *ptr;

ptr = (float *)mkl_malloc(size, 64);

mkl_free(ptr);
```

MCDRAM からのメモリー確保を試みる
確保できない場合には、
代わりに DDR4 から確保する

参考 :「インテル® MKL 2017 for Linux* デベロッパー・ガイド — インテル® MKL での高帯域メモリーの使用」
https://jp.xlsoft.com/documents/intel/mkl/2017/mkl2017_dev_guide_lnx/GUID-FFF068F5-CADB-463B-AD0A-CB8C34EC9811.html

MCDRAM の利用 : インテル® MPI ライブラリー

- I_MPI_HBW_POLICY 環境変数

  書式 : I_MPI_HBW_POLICY=<Policy1>[,[Policy2][,Policy3]]
  - Policy1: ユーザープログラムのメモリーを MCDRAM から確保するか
  - Policy2: MPI ライブラリー内部のメモリーを MCDRAM から確保するか
  - Policy3: MPI_Win_allocate_shared/MPI_Win_allocate API により確保されるメモリーを MCDRAM から確保するか

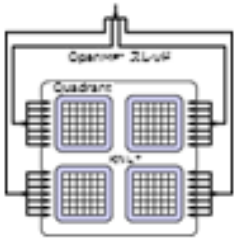
Policy に指定できる値	説明
hbw_preferred	MCDRAM から優先的にメモリーを確保する
hbw_bind	MCDRAM からのみメモリーを確保する
hbw_interleave	MCDRAM と DDR4 から交互にメモリーを確保する

クラスターモードの利用法

スレッド・アフィニティーとメモリー/ディレクトリーのアフィニティーを一致させる

入れ子並列処理 (OpenMP*)

```
1  #pragma omp parallel
2  {
3    // ...
4  #pragma omp parallel
5    {
6      // ...
7    }
8  }
```




```
user@knl% OMP_NUM_THREADS=4,72
user@knl% OMP_NESTED=1
```

MPI + OpenMP*

```
1  stat = MPI_Init();
2  // ...
3  #pragma omp parallel
4  {
5      // ...
6  }
7  // ...
8  MPI_Finalize();
```


```
user@knl% mpirun -host knl \
>  -np 4 ./myparallel_app
```

colfaxresearch.com/knl-webinar　クラスターモードを利用するプログラミング　© Colfax International, 2013–2016

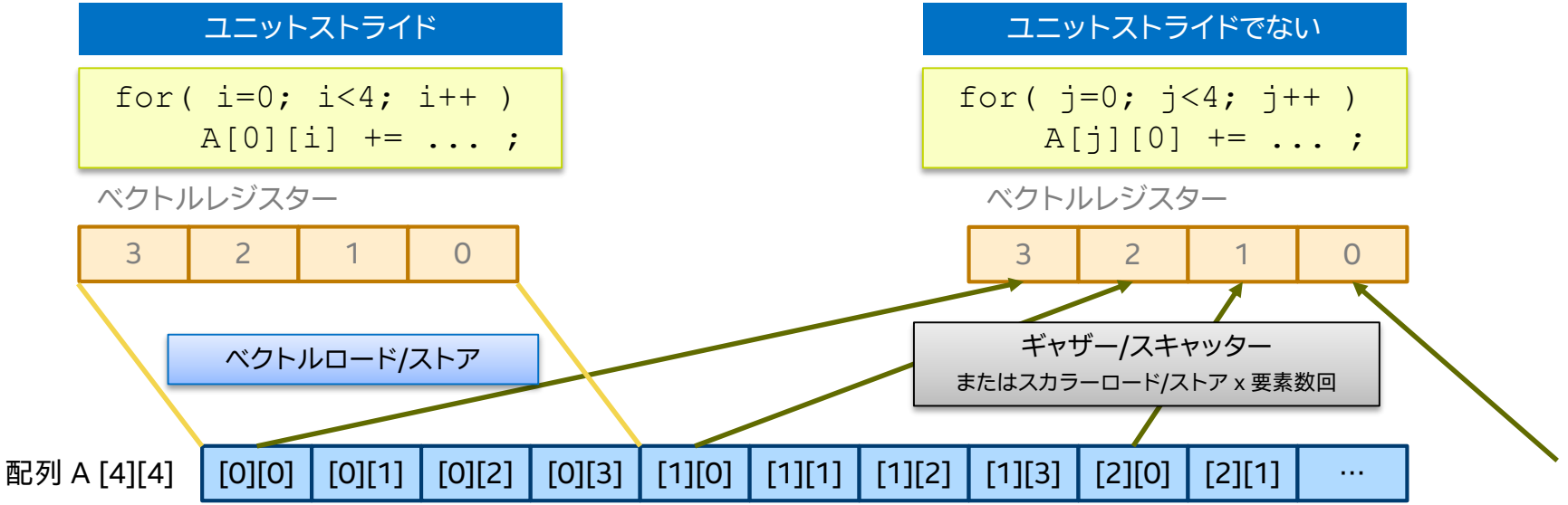
「開発者用 Knights Landing ガイド」p.55
https://jp.xlsoft.com/documents/intel/phi/colfax/Colfax-Programmers-Guide-to-KNL.pdf

補足資料

データ配置、アクセスパターンについて(1)

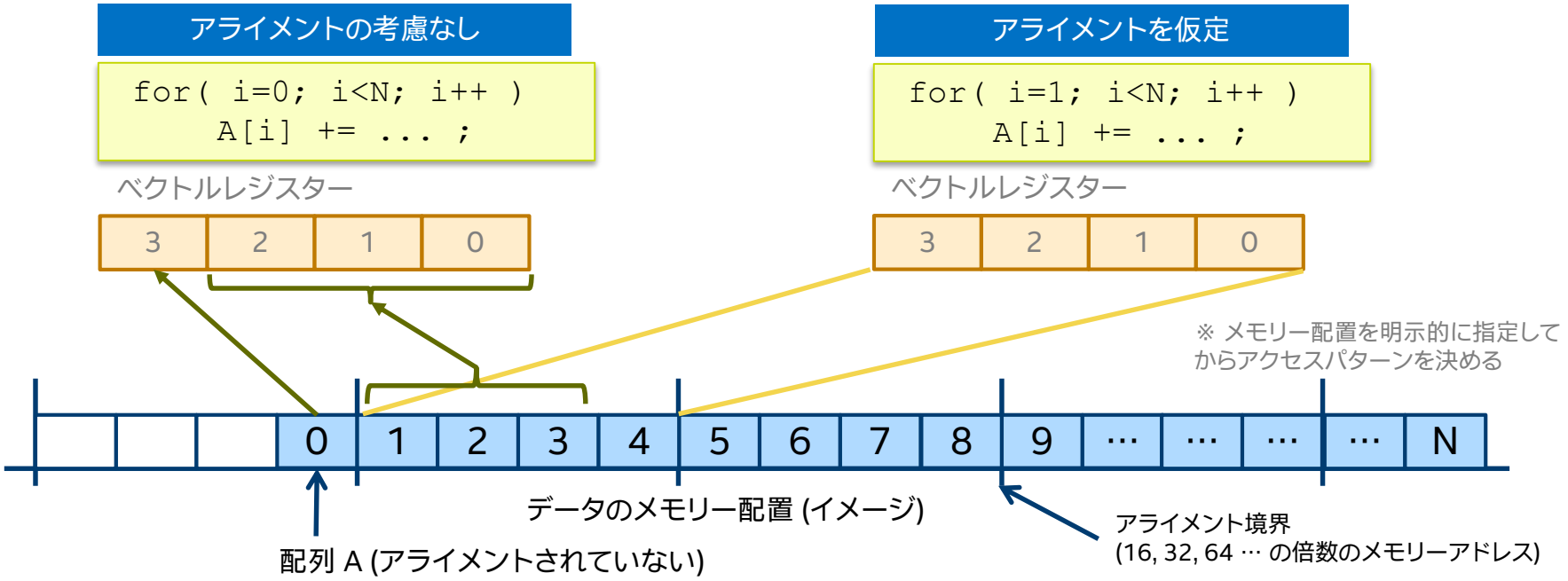
- SIMD 命令は、一般に連続したデータを取り扱う
  隣り合わない要素のアクセスは非効率
  (ユニットストライドでないデータアクセス)

※ Fortran は配列の左の添え字、
C/C++ は配列の右の添え字がユニットストライド



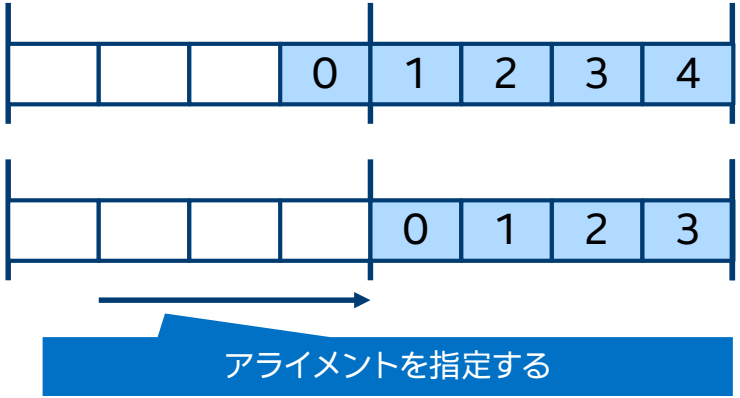
データ配置、アクセスパターンについて(2)

- アライメントは効率の良いベクトル演算のために重要

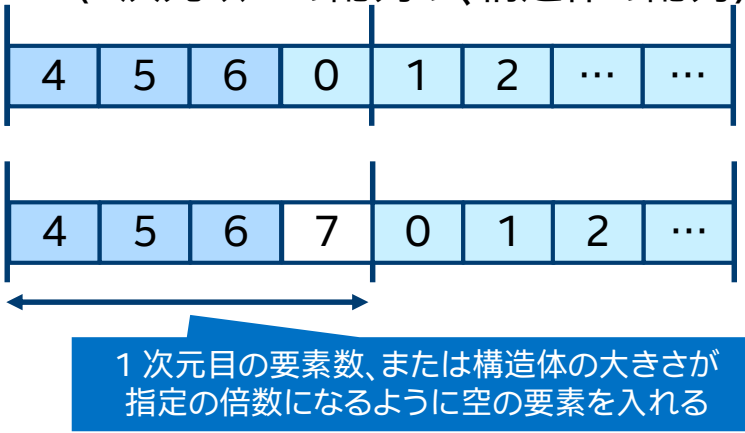


明示的なメモリー・アライメント

- データ(配列)の先頭



- データの中間点
  (2次元以上の配列や、構造体の配列)



	インテル® SSE 128 ビット/16 バイト	インテル® AVX 256 ビット/32 バイト	インテル® AVX-512 512 ビット/64 バイト
整数/単精度 (4バイト)	4 要素	8 要素	16 要素
倍精度 (8バイト)	2 要素	4 要素	8 要素

メモリー・アライメントの指定方法

Fortran

- コンパイラーオプション
  - -align array64byte
- 変数への指定 (!dir$)
  - real, allocatable :: A(:)
    !dir$ attributes align : 64 :: A
- コンパイラーへの通知
  - !dir$ assume_aligned A : 64

C/C++

- 変数への指定
- __attribute__((aligned(n)))
  (変数定義の前または最後に追加)
- メモリー確保関数
- void* __mm_malloc (site_t size, size_t align)
- void __mm_free (void *p)
- コンパイラーへの通知
- __assume_aligned(val, 64);

参考:「ベクトル化の可能性を高めるデータ・アライメント」
http://www.isus.jp/article/mic-article/data-alignment-to-assist-vectorization/

参考資料1 (Web)

- インテル® AVX-512 でベクトル化のパフォーマンスを向上する

https://www.isus.jp/products/c-compilers/improve-vectorization-performance-using-avx-512/

- インテル® Advisor のルーフライン解析

https://www.isus.jp/products/advisor/pu27-05-intel-advisor-roofline-analysis/

- オンライン・トレーニング : キャッシュを考慮したルーフライン解析を使用して ベクトル化とメモリーの最適化を詳しく調査する

https://www.isus.jp/products/advisor/roofline_webinar/

- Intel® Advisor Release Notes and New Features (リリースノートおよび新機能)

https://software.intel.com/en-us/articles/intel-advisor-xe-release-notes

参考資料2 (Web)

- オンライン・トレーニング: インテル® コンパイラーによる OpenMP* 入門

https://www.isus.jp/products/psxe/compiler-openmp-introduction/

- オンライン・トレーニング: OpenMP* 4.x による新しいレベルの並列化

https://www.isus.jp/products/psxe/parallel-openmp/

- Knights Landing† プロセッサーにおける 高帯域幅メモリー (HBM) としての MCDRAM: 開発者ガイド

https://www.isus.jp/hpc/colfax_knl_mcdram_guide/

- Code Sample: Allocate Memory Efficiently on an Intel® Xeon Phi™ Processor

https://software.intel.com/en-us/articles/mcdram-high-bandwidth-memory-on-knights-landing-analysis-methods-tools

「インテル® Xeon Phi™ プロセッサーの高帯域幅メモリーを活用するコードの作成」

https://jp.xlsoft.com/documents/intel/parallel/2017/code-for-speed-with-high-bandwidth-memory-on-intel-xeon-phi-processors.pdf

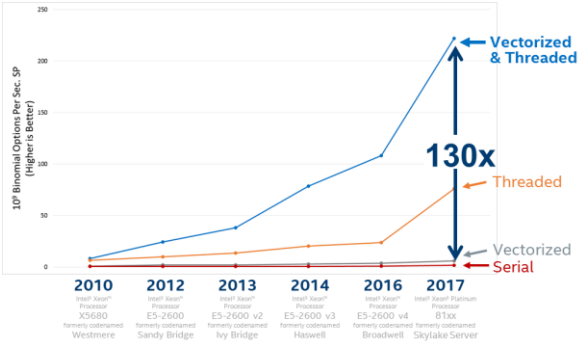
- Heart-Demo: Analyzing the Performance of an OpenMP* and MPI Application

https://software.intel.com/en-us/itac-vtune-mpi-openmp-tutorial-lin

https://software.intel.com/en-us/videos/have-a-heart-love-your-hybrid-programs

ベクトル化およびスレッド化するか、パフォーマンスを諦めるか

スレッド化 + ベクトル化はどちらか一方よりもはるかに高速



2010 年 ~ 2017 年のベンチマークのシステム構成

システム構成

	プラットフォーム	スケーリングされていないコアクロックの周波数	コア/ソケット	ソケット数	L1 データキャッシュ	L2 キャッシュ	L3 キャッシュ	メモリー	メモリー周波数	メモリーアクセス	H/W プリフェッチ有効	HT 有効	ターボ有効	C ステート	OS	カーネル	コンパイラー
WSM†	インテル® Xeon® プロセッサー X5680	3.33GHz	6	2	32K	256K	12MB	48MB	1333MHz	NUMA	Y	Y	Y	無効	Fedora* 20	3.11.10-301.fc20	icc 17.0.2
SNB†	インテル® Xeon® プロセッサー E5 2690	2.90GHz	8	2	32K	256K	20MB	64GB	1600MHz	NUMA	Y	Y	Y	無効	Fedora* 20	3.11.10-301.fc20	icc 17.0.2
IVB†	インテル® Xeon® プロセッサー E5 2697 v2	2.70GHz	12	2	32K	256K	30MB	64GB	1867MHz	NUMA	Y	Y	Y	無効	RHEL 7.1	3.10.0-229.el7.x86_64	icc 17.0.2
HSW†	インテル® Xeon® プロセッサー E5 2600 v3	2.20GHz	18	2	32K	256K	46MB	128GB	2133MHz	NUMA	Y	Y	Y	無効	Fedora* 20	3.15.10-200.fc20.x86_64	icc 17.0.2
BDW†	インテル® Xeon® プロセッサー E5 2600 v4	2.30GHz	18	2	32K	256K	46MB	256GB	2400MHz	NUMA	Y	Y	Y	無効	RHEL 7.0	3.10.0-123.el7.x86_64	icc 17.0.2
BDW†	インテル® Xeon® プロセッサー E5 2600 v4	2.20GHz	22	2	32K	256K	56MB	128GB	2133MHz	NUMA	Y	Y	Y	無効	CentOS* 7.2	3.10.0-327.el7.x86_64	icc 17.0.2
SKX†	インテル® Xeon® Platinum 81xx プロセッサー	2.50GHz	28	2	32K	1024K	40MB	192GB	2666MHz	NUMA	Y	Y	Y	無効	CentOS* 7.3	3.10.0-514.10.2.el7.x86_64	icc 17.0.2

ベンチマークの出典: インテル社内での測定値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。
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

† 開発コード名

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。

改訂 #20110804


intel
Software