インテル® コンパイラー 18.0 を使用した最適化手法 (初級編)

久保寺　陽子

内容

- インテル® コンパイラーの紹介
- インテル® コンパイラーの新機能
- インテル® コンパイラーを用いた最適化例 (1)
- インテル® コンパイラーを用いた最適化例 (2)
- インテル® コンパイラー特有のオプション
- まとめ

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® コンパイラーの紹介

インテル® C++/Fortran コンパイラー 18.0

ソフトウェア開発者にとってのメリット

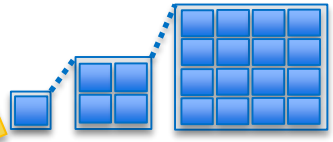
パフォーマンス

インテル® アーキテクチャー向けに最適化およびベクトル化されたコードを出力

生産性

最新の言語標準と OpenMP* 標準規格の利用、主要なコンパイラーおよび IDE と互換

ベクトル化、マルチスレッド、マルチノードをサポート



スケーラビリティー

インテル® Parallel Studio XE はソフトウェア開発者に最適な開発環境を提供

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® Parallel Studio XE – コンポーネント

	Composer Edition	Professional Edition	Cluster Edition
インテル® C++ コンパイラー	☑	☑	☑
インテル® Fortran コンパイラー	☑	☑	☑
インテル® Distribution for Python*	☑	☑	☑
インテル® マス・カーネル・ライブラリー (インテル® MKL)	☑	☑	☑
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)	☑	☑	☑
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)	☑	☑	☑
OpenMP*	☑	☑	☑
インテル® Advisor		☑	☑
インテル® Inspector		☑	☑
インテル® VTune™ Amplifier		☑	☑
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 (インテル® DAAL)		☑	☑
インテル® MPI ライブラリー			☑
インテル® Trace Analyzer & Collector			☑
ローグウェーブ IMSL* ライブラリー	バンドルとアドオン	アドオン	アドオン

インテル® コンパイラー 18.0 はインテル® Parallel Studio XE 2018 に含まれます。

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® コンパイラーの特徴

- インテル® Xeon® プロセッサーとインテル® Xeon Phi™ プロセッサーを含む、さまざまな インテル® アーキテクチャー向けに最適化およびベクトル化されたコードを開発
  - 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 512 (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

- 最新の言語標準と OpenMP* 標準規格の利用、主要なコンパイラーおよび IDE との互換性
  - C++14 の完全サポート、C++ 2017 (ドラフト) の初期サポート
  - Fortran 2008 の完全サポート、Fortran 2015 (ドラフト) の初期サポート
  - Python* 2.7/3.6 のサポート、OpenMP* 5.0 (ドラフト) の初期サポート
  - Microsoft* Visual Studio* 2017 との統合

- SIMD Data Layout Template (SDLT) ライブラリー
  - 標準の C++ 構造体配列コードをベクトル化

- 仮想関数をベクトル化
  - オブジェクト指向の C++ コードのパフォーマンスを向上

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

新しいオプションと機能

インテル® C++/Fortran コンパイラー 18.0

新しいオプションと機能

- Microsoft* Visual Studio* 2017 のサポート
- */GS* でスタック･セキュリティー保護を変更します
  - デフォルトでは、すべてのスタック･セキュリティー･レベルがチェックされます
  - 一部のプログラムではパフォーマンスに影響があるため、*/GS:partial* を使用します
- 制御フローの強制テクノロジー
  - *–[q|Q]cf-potection[[=|:]keyword]*
  - *keyword* には、*shadow_stack*、*branch_tracking*、*full* または *none* を指定します
  - 制御フローの強制テクノロジー (CET) 保護を有効にします
  - https://software.intel.com/sites/default/files/managed/4d/2a/control-flow-enforcement-technology-preview.pdf (英語)

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

新しいオプションと機能

- 関数分割
  - *–[f/Q]fnsplit[=n]*、ここで *0 <= n <= 100*
  - *n* より少ないか等しい実行確率の関数ブロックを分割します
  - GCC の *–freorder-blocks-and-partition* もサポートされます (動的なプロファイルが必要)
- SVML 呼び出し* をコンパイル時に割り当てます
  - *–[f/Q]imf-force-dynamic-target[[=|:]funclist]*
  - CPU 固有の SVML エントリーの呼び出しがデフォルトで実行されます
  - ランタイムで割り当ては行われません
- ベクトルとスカラーコードの FP 一貫性*
  - *–[f/Q]imf-use-svml*
  - スカラー数値計算で使用されている現在の LIBM に SVML の使用を強制します
  - 計算結果はビット単位で同等です (ベクトルとスカラー)

*18.0 Beta Update 1 でサポート

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

新しいオプションと機能

- バッファー操作関数のインライン展開制御
  - *–[m|Q]stringop-strategy[[=|:]keyword]*
  - *keyword* は、*const_size_loop*、*libcall* または *rep* です
- *-[q|Q][no-]opt-assume-safe-padding*
  - コンパイラーが、プログラムによって割り当てられた配列/変数に続く 64 バイトを完全にアクセスできることを仮定するのを許可します
  - KNC † 固有のオプションでしたが、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) 基本命令をサポートしているすべてのターゲット・プロセッサー向けに再導入されました

† 開発コード名 Knights Corner の略称

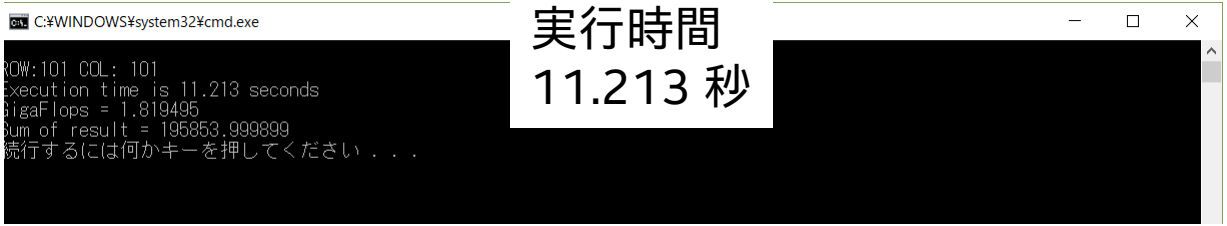
© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® コンパイラーを用いた最適化例 (1)

ベクトル化

性能のベースラインの確定 (最適化前)

- サンプルプログラム vec_samples の入手 (https://software.intel.com/en-us/product-code-samples (英語))
- インテル® コンパイラーを選択
- プロジェクトのプロパティーを設定 ([構成プロパティ] > [C/C++])
  - [Optimization (最適化)]: [Minimize Size (サイズの最小化)]
  - [Optimization [Intel C++] (最適化 [インテル(R) C++])] > [Interprocedural Optimization (プロシージャー間の最適化)]: [No (いいえ)]
  - [General [Intel C++] (言語 [インテル(R) C++])] > [Enable C99 Support (C99 サポートを有効にする)]: [Yes (はい)]
    - 可変長配列 (VLA) を使用するため
  - [Code Generation (コード生成)] > [Floating Point Model (浮動小数点モデル)]: [Fast (/fp:fast)]
    - /fp:fast オプションは、浮動小数点算術演算においてより強力な最適化を行うようにコンパイラーに指示
    - /fp:precise オプションや /fp:strict オプションは、自動ベクトル化の機会を制限することがある



実行時間
11.213 秒

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

ベクトル化レポートの生成

Qopt-report、Qopt-report-phase:vec

- **Qopt-report:0 ~ 5 までのレベル (0 はレポート出力なし)**
  - レベル 1 – ベクトル化されたループを出力
  - レベル 2 – ベクトル化されたループおよびされなかったループとその理由を出力
- **Qopt-report-phase:フェーズ名**
  - レポートを出力するフェーズを指定
  - フェーズの種類: cg (コード生成)、ipo (プロシージャー間の最適化)、loop (ループ入れ子構造の最適化)、offload (インテル® MIC アーキテクチャーまたはインテル® グラフィックス・テクノロジー向け)、openmp、par (自動並列化)、pgo (プロファイルに基づく最適化)、tcollect (トレース収集)、vec (ベクトル化)、all (デフォルト)
- **プロジェクトのプロパティーを設定 ([構成プロパティ] > [C/C++])**
  - [Diagnostics [Intel C++] (診断 [インテル(R) C++])] > [Optimization Diagnostic Level (最適化診断レベル)]: [Level 1 (レベル 1) (/Qopt-report:1)] または [Level 2 (レベル 2) (/Qopt-report:2)]
  - [Optimization Diagnostics Phase (最適化診断フェーズ)]: [Vectorization (ベクトル化) (/Qopt-report-phase:vec)]
  - [Optimization (最適化)]: [Maximize Speed (実行速度)] ← ベクトル化を実行
  - [Preprocessor (プリプロセッサー)] > [Preprocessor Definitions (プリプロセッサーの定義)] NOFUNCCALL を追加

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

ベクトル化レポートの出力例 (レベル 1)

vec_samples Property Pages

Configuration: Active(Release) Platform: Active(Win32) Configuration Manager...

- Configuration Properties
  - General
  - Debugging
  - Intel Debugging
  - Intel Performance Librari
  - VC++ Directories
  - C/C++
    - General
    - General [Intel C++]
    - Debug [Intel C++]
    - Optimization
    - Optimization [Intel C+
    - Preprocessor
    - Code Generation
    - Code Generation [Inte
    - Language
    - Language [Intel C++]
    - Precompiled Headers
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Diagnostics [Intel C++
    - Advanced
    - All Options
    - Command Line
  - Linker

Error Limit	
Disable Specific Diagnostics	
Emit Diagnostics to File	No
Diagnostics File	$(IntDir)$(TargetName).diag
Optimization Diagnostic Level	**Level 1 (/Qopt-report:1)**
Emit Optimization Diagnostics to File	No
Optimization Diagnostics File	$(IntDir)$(TargetName).rep
Optimization Diagnostic Phase	**Vectorization (/Qopt-report-phase:vec)**
Optimization Diagnostic Routine	Interprocedural Optimization (/Qopt-report-phase:ipo)
Guided Auto Parallelism Analysis	Profile Guided Optimization (/Qopt-report-phase:pgo)
Emit Guided Auto Parallelism Diagnostics	Loop Nest Optimization (/Qopt-report-phase:loop)
Guided Auto Parallelism Diagnostics File	Auto-Parallelization (/Qopt-report-phase:par)
Guided Auto Parallelism Code Selection O	Vectorization (/Qopt-report-phase:vec)
Generate Native Assembly Listing for Intel	Offload (/Qopt-report-phase:offload)
Native Assembly Listing for Intel(R) Graphi	OpenMP (/Qopt-report-phase:openmp)
	Code Generation (/Qopt-report-phase:cg)
	Intel® Trace Analyzer Collection (/Qopt-report-phase:tcollect)
	All optimizer phases

Optimization Diagnostic Phase
Specifies which optimizer phase to use when optimization reports are generated. (/Qopt-report-phase:<phase>)

OK キャンセル 適用(A)

Server Explorer Toolbox

Driver.c Multiply.h Multiply.c

vec_samples (Global Scope) matvec(int size1, int size

```
41  /*
42   * The pragma vector aligned below tells the compiler to assume that the data ir
43   * the loop is aligned on 16-byte boundary so the vectorizer can use
44   * aligned instructions to generate faster code.
45   */
46  #pragma vector aligned
47  #endif
48
    1 optimization note
    Multiversioned v1
      15300: LOOP WAS VECTORIZED (line: 49, column: 9)
49          for (j = 0;j < size2; j++) {
50              b[i] += a[i][j] * x[j];
51          }
52      }
53  }
54
```

100 %

Compiler Optimization Report

7 VEC | 0 PAR | 0 OpenMP | 0 PGO | 3 LNO | 0 Offload | 0 CG | Search Optimiza

	Description	Inlined Into	File	Line	Colum
4	Main loop	Driver.c (135, 5)	Driver.c	47	5
5	Main loop	Driver.c (136, 5)	Driver.c	62	5
7	Main loop	Driver.c (159, 5)	Driver.c	74	5

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

ベクトル化レポートの出力例 (レベル 2)

vec_samples Property Pages

Configuration: Active(Release) Platform: Active(Win32)

- Configuration Properties
  - General
  - Debugging
  - Intel Debugging
  - Intel Performance Librari
  - VC++ Directories
  - C/C++
    - General
    - General [Intel C++]
    - Debug [Intel C++]
    - Optimization
    - Optimization [Intel C+
    - Preprocessor
    - Code Generation
    - Code Generation [Inte
    - Language
    - Language [Intel C++]
    - Precompiled Headers
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Diagnostics [Intel C++
    - Advanced

Error Limit	
Disable Specific Diagnostics	
Emit Diagnostics to File	No
Diagnostics File	$(IntDir)$(TargetName).diag
Optimization Diagnostic Level	Level 2 (/Qopt-report:2)
Emit Optimization Diagnostics to File	No
Optimization Diagnostics File	$(IntDir)$(TargetName).rep
Optimization Diagnostic Phase	Vectorization (/Qopt-report-phase:vec)
Optimization Diagnostic Routine	
Guided Auto Parallelism Analysis	Disable
Emit Guided Auto Parallelism Diagnostics	No
Guided Auto Parallelism Diagnostics File	$(IntDir)$(TargetName).gap
Guided Auto Parallelism Code Selection O	
Generate Native Assembly Listing for Intel(	No
Native Assembly Listing for Intel(R) Graphi	$(IntDir)gpu-asm-dump.txt

C:¥WINDOWS¥system32¥cmd.exe

```
ROW:101 COL: 101
Execution time is 3.520 seconds
GigaFlops = 5.796023
Sum of result = 195853.999899
続行するには何かキーを押してください . . .
```

実行時間
3.520 秒

Server Explorer Toolbox

Driver.c Multiply.h Multiply.c

vec_samples (Global Scope)

```
47  #endif
48
49      for (j = 0;j < size2; j++) {
50          b[i] += a[i][j] * x[j];
51      }
52  }
53 }
54
```

4 optimization notes

- Multiversioned v1
  - 15300: LOOP WAS VECTORIZED (line: 49, column: 9)
- Remainder loop for vectorization, Multiversioned v1
  - 15335: remainder loop was not vectorized: vectorization possible but seems inefficient. Use vector always directive or /Qvec-threshold0 to override (line: 49, column: 9)
- Multiversioned v2
  - 15304: loop was not vectorized: non-vectorizable loop instance from multiversioning (line: 49, column: 9)
- Remainder, Multiversioned v2

ベクトル化されていない部分に関して説明

100 %

Compiler Optimization Report

15 VEC | 0 PAR | 0 OpenMP | 0 PGO | 0 LNO | 0 Offload | 0 CG | Search Optimization

	Description	Inlined Into	File	Line	Column	Pro
15	Remainder loop for vectorization		Driver.c	74	5	vec
16	Peeled loop for vectorization		Driver.c	62	5	vec
17	Main loop		Driver.c	62	5	vec

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

ポインター一義化により性能を向上

vec_samples Property Pages

Configuration: Active(Release) Platform: Active(Win32) Configuration Manager...

- Configuration Properties
  - General
  - Debugging
  - Intel Debugging
  - Intel Performance Librari
  - VC++ Directories
  - C/C++
    - General
    - General [Intel C++]
    - Debug [Intel C++]
    - Optimization
    - Optimization [Intel C+
    - Preprocessor
    - Code Generation
    - Code Generation [Inte
    - Language
    - Language [Intel C++]
    - Precompiled Headers
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Diagnostics [Intel C++
    - Advanced
    - All Options
    - Command Line
  - Linker

Preprocessor Definitions	VIN32;NDEBUG;_CONSOLE;%(PreprocessorDefinitions);NOALIAS
Undefine Preprocessor Definitions	
Undefine All Preprocessor Definitions	No
Ignore Standard Include Paths	No
Preprocess to a File	No
Preprocess Suppress Line Numbers	No
Keep Comments	No

Preproc
Defines

Server Explorer Toolbox

Driver.c Multiply.h Multiply.c

vec_samples (Global Scope) matvec(int size1, int size2, FTYPE a[][size2

```
42  * The pragma vector aligned below tells the compiler to assume that the data in
43  * the loop is aligned on 16-byte boundary so the vectorizer can use
44  * aligned instructions to generate faster code.
45  */
46 #pragma vector aligned
47 #endif
48
49     for (j = 0;j < size2; j++) {
           b[i] += a[i][j] * x[j];
       }
```

2 optimization notes

- *Main loop*
  - *15300: LOOP WAS VECTORIZED (line: 49, column: 9)*
- *Remainder loop for vectorization*
  - *15335: remainder loop was not vectorized: vectorization possible but seems inefficient. Use vector always directive or /Qvec-threshold0 to override (line: 49, column: 9)*

ポインターは一義化されたが、
まだベクトル化には十分でない

Compiler Optimization Report

12 VEC | 0 PAR | 0 OpenMP | 0 PGO | 0 LNO | 0 Offload | 0 CG

Search Optimization Report

	Description	Inlined Into	File	Line	Column	Project
11	Main loop		Driver.c	74	5	vec_samples
12	Remainder loop for vectorization		Driver.c	74	5	vec_samples

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

データのアライメントにより性能を向上

vec_samples Property Pages

Configuration: Active(Release) Platform: Active(Win32) Configuration Manager...

- Configuration Properties
  - General
  - Debugging
  - Intel Debugging
  - Intel Performance Librari
  - VC++ Directories
  - C/C++
    - General
    - General [Intel C++]
    - Debug [Intel C++]
    - Optimization
    - Optimization [Intel C+
    - Preprocessor
    - Code Generation
    - Code Generation [Inte
    - Language
    - Language [Intel C++]
    - Precompiled Headers
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Diagnostics [Intel C++
    - Advanced
    - All Options
    - Command Line
  - Linker

Preprocessor Definitions	EBUG;_CONSOLE;%(PreprocessorDefinitions);NOALIAS;ALIGNED
Undefine Preprocessor Definitions	
Undefine All Preprocessor Definitions	No
Ignore Standard Include Paths	No
Preprocess to a File	
Preprocess Suppress Line Numbers	
Keep Comments	

Preprocessor Definitions
Defines a preprocessing symbols for your sou

Server Explorer Toolbox

Driver.c Multiply.h Multiply.c

vec_samples (Global Scope) matvec(int size1, int size2, FTYPE a[][size2

```
41  /*
42   * The pragma vector aligned below tells the
43   * the loop is aligned on 16-byte boundary s
44   * aligned instructions to generate faster
45   */
46  #pragma vector aligned
47  #endif
48
49          for (j = 0;j < size2; j++) {
50              b[i] += a[i][j] * x[j];
51          }
52      }
53  }
54
```

2 optimization notes

- 15388: vectorization support: reference a[i][j] has aligned access (line: 50, column: 13)
- 15388: vectorization support: reference x[j] has aligned access (line: 50, column: 13)
- 15305: vectorization support: vector length 2 (line: 49, column: 9)
- 15399: vectorization support: unroll factor set to 4 (line: 49, column: 9)
- 15309: vectorization support: normalized vectorization overhead 0.594 (line: 49, column: 9)
- 15300: LOOP WAS VECTORIZED (line: 49, column: 9)
- 15448: unmasked aligned unit stride loads: 2 (line: 49, column: 9)
- 15475: --- begin vector cost summary --- (line: 49, column: 9)
- 15476: scalar cost: 10 (line: 49, column: 9)
- 15477: vector cost: 4.000 (line: 49, column: 9)
- 15478: estimated potential speedup: 2.410 (line: 49, column: 9)

データのアライメントが保証されているので、ベクトル化された

C:¥WINDOWS¥system32¥cmd.exe

```
ROW:101 COL: 102
Execution time is 3.248 seconds
GigaFlops = 6.281404
Sum of result = 195853.999899
続行するには何かキーを押してください . . .
```

実行時間
3.248 秒

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

プロシージャー間の最適化で性能を向上

vec_samples Property Pages

Configuration: Active(Release) Platform: Active(Win32)

- Configuration Properties
  - General
  - Debugging
  - Intel Debugging
  - Intel Performance Librari
  - VC++ Directories
  - C/C++
    - General
    - General [Intel C++]
    - Debug [Intel C++]
    - Optimization
    - Optimization [Intel C+
    - Preprocessor
    - Code Generation
    - Code Generation [Inte
    - Language
    - Language [Intel C++]
    - Precompiled Headers
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Diagnostics [Intel C++
    - Advanced
    - All Options
    - Command Line

Interprocedural Optimization	Multi-File (/Qipo)
Optimize For Windows Application	No
Flush Denormal Results To Zero	
Enable Matrix Multiply Library Call	Default
Loop Unrolling	
Parallelization	No
Use Intel Optimized Headers	No

C:¥WINDOWS¥system32¥cmd.exe

```
ROW:101 COL: 102
Execution time is 3.061 seconds
GigaFlops = 6.665142
Sum of result = 195853.999899
続行するには何かキーを押してください . . .
```

実行時間
3.061 秒

Driver.c | Multiply.h | Multiply.c

vec_samples | (Global Scope) | main()

```
149     #else
        3 optimization notes
        Main loop - Inlined From Multiply.c - (49, 9)
        Remainder loop for vectorization - Inlined From Multiply.c - (49, 9)
        Main loop - Inlined From Multiply.c - (37, 5)
150         matvec(size1,size2,a,b,x);
151     #endif
        1 optimization note
        Main loop
152         x[0] = x[0] + 0.000001;
153     }
154     endTime = clock_it();
155     execTime = endTime - startTime;
156
157     printf("Execution time is %2.3f seconds\n", execTime);
158     printf("GigaFlops = %f\n", (((double)REPEATNTIMES * (double)COL * (double)ROW * 2.0) /
        1 optimization note
        Main loop - Inlined From Driver.c - (74, 5)
159     printsum(COL,b);
160
```

100 %

プロシージャー matvec がインライン展開

プロシージャー printsum がインライン展開

Compiler Optimization Report

65 VEC | 0 PAR | 0 OpenMP | 0 PGO | 0 LNO | 0 Offload | 0 CG | Search Optimization Report

	Description	Inlined Into	File	Line	Columr	Project
1	Main loop	Driver.c (150, 9)	Multiply.c	49	9	vec_samples
2	Remainder loop for vectorization	Driver.c (150, 9)	Multiply.c	49	9	vec_samples
3	Main loop	Driver.c (150, 9)	Multiply.c	37	5	vec_samples
4	Main loop		Driver.c	152	9	vec_samples
5	Main loop	Driver.c (159, 5)	Driver.c	74	5	vec_samples
7	Main loop		Driver.c	62	5	vec_samples
9	Main loop		Driver.c	48	9	vec_samples
11	Main loop		Driver.c	47	5	vec_samples
12	Main loop		Driver.c	53	9	vec_samples

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

ベクトル化のまとめ

- インテル® コンパイラーのオプションを変更するだけ!
- 性能向上
  - 通常のアプリケーションを想定 11.213 秒
  - ベクトル化を入れた最適化 3.520 秒
  - ポインターの一義化、データのアライメント 3.248 秒
  - プロシージャー間の最適化 3.061 秒

3.7 倍の性能向上

高速化にはインテル® コンパイラーが最適

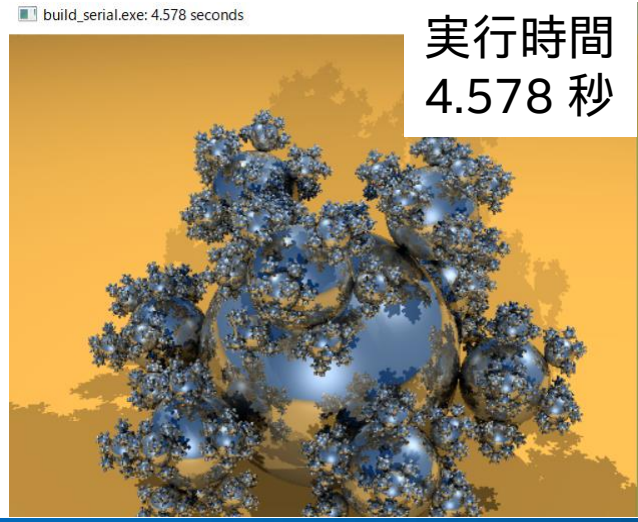
© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® コンパイラーを用いた最適化例 (2)

マルチスレッド化

性能のベースラインの確定 (最適化前)

- サンプルプログラム tachyon_samples_2017 の入手 (https://software.intel.com/en-us/product-code-samples (英語))
- build_serial プロジェクトをスタートアップ・プロジェクトに設定
- インテル® コンパイラーを選択
- プロジェクトのプロパティーを確認　([構成プロパティ] > [C/C++])
  - [Optimization (最適化)]: [Maximize Speed (/O2)]
  - [Optimization [Intel C++] (最適化 [インテル(R) C++])] > [Interprocedural Optimization (プロシージャー間の最適化)]: [Yes (はい)]
  - シングルスレッドとしては最適化されている

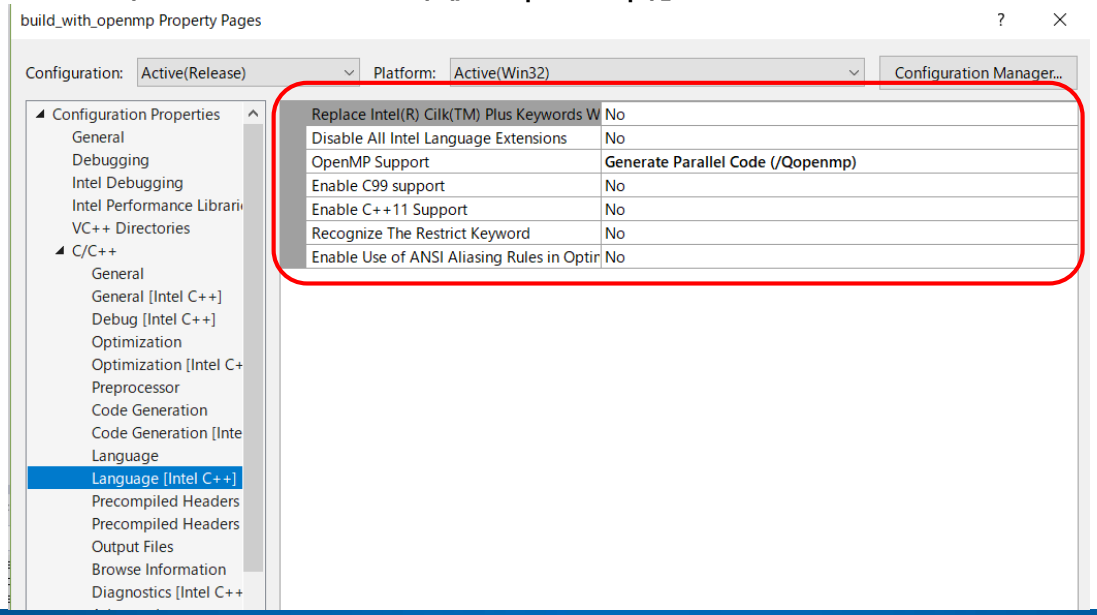


実行時間
4.578 秒

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

OpenMP* でマルチスレッド化

- build_with_openmp プロジェクトをスタートアップ･プロジェクトに設定
- プロジェクトのプロパティーを確認 ([構成プロパティ] > [C/C++])
  - [Language [Intel C++] (言語 [インテル(R) C++])] > [OpenMP Support (OpenMP* サポート)] > [Generate Parallel Code (並列コードの生成) (/Qopenmp)]
- コードの若干の変更



© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

OpenMP* でマルチスレッド化

変更前

```
static void parallel_thread(void)
{
   // thread-local storage
   unsigned int serial = 1;
   unsigned int mboxsize = sizeof(unsigned int)*(max_objectid() + 20);
   unsigned int * local_mbox = (unsigned int *)alloca(mboxsize);
   memset(local_mbox, 0, mboxsize);

   for (int y = starty; y < stopy; y++)
   {
      drawing_area drawing(startx, totaly - y, stopx - startx, 1);
      for (int x = startx; x < stopx; x++)
      {
         color_t c = render_one_pixel(x, y, local_mbox, serial, startx,
                              stopx, starty, stopy);
         drawing.put_pixel(c);
      }
      if (!video->next_frame()) return;
   }
}
```

各スレッドに共通な値は並列領域の外へ

有効性の削除

変更後

```
static void parallel_thread(void)
{
   unsigned int mboxsize = sizeof(unsigned int)*(max_objectid() + 20);
#pragma omp parallel for
   for ( int y = starty; y < stopy; y++ )
   {
      unsigned int serial = 1;
      unsigned int local_mbox[mboxsize];
      memset(local_mbox, 0, mboxsize);
      drawing_area drawing(startx, totaly - y, stopx - startx, 1);
      for ( int x = startx; x < stopx; x++ )
      {
         color_t c = render_one_pixel(x, y, local_mbox, serial, startx, stopx, starty, stopy);
         drawing.put_pixel(c);
      }
      video->next_frame();
   }
```

OpenMP* による並列領域

build_with_openmp_solution.exe: 2.562 seconds

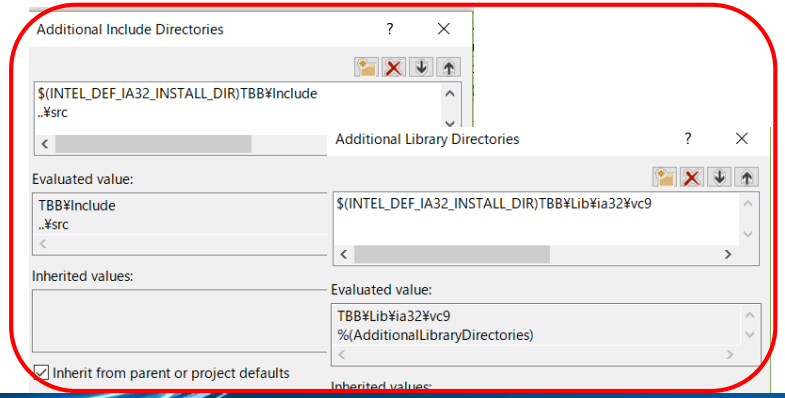
実行時間
2.562 秒

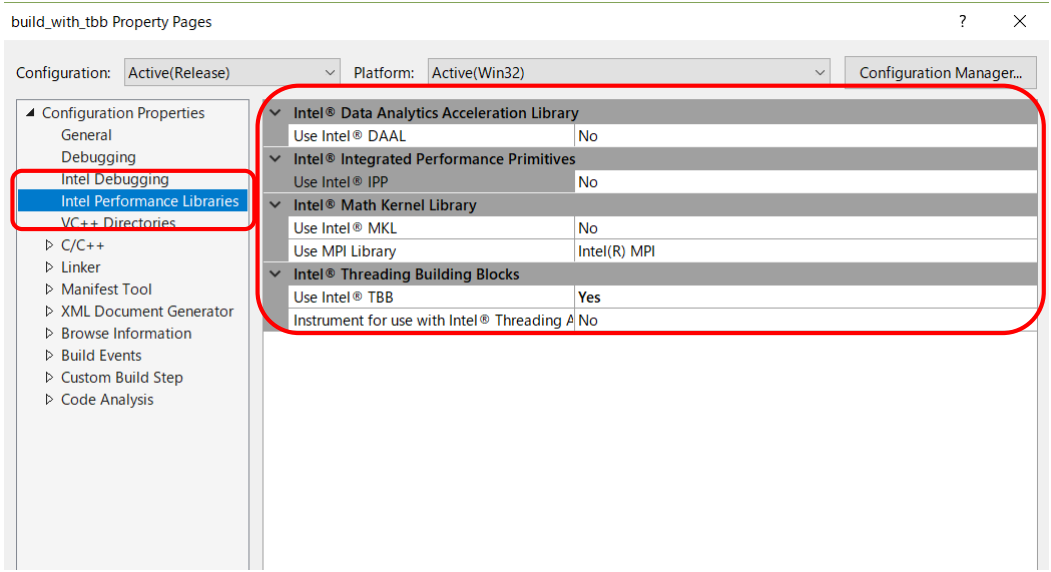
© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® TBB でマルチスレッド化

- build_with_tbb プロジェクトをスタートアップ・プロジェクトに設定
- プロジェクトのプロパティーを確認

  ([構成プロパティ] > [Intel Performance Libraries (インテル(R) パフォーマンス・ライブラリー)])
  - [Intel® Threading Building Blocks (インテル(R) TBB)] > [Use TBB (インテル(R) TBB を使用する)]
- インクルード、リンク先にインテル® TBB の各ディレクトリーがあることを確認
- コードの若干の変更





© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® TBB でマルチスレッド化

変更前

```
static void parallel_thread(void)
{
   // thread-local storage
   unsigned int serial = 1;
   unsigned int mboxsize = sizeof(unsigned int)*(max_objectid() +
20);
   unsigned int * local_mbox = (unsigned int *)alloca(mboxsize);
   memset(local_mbox, 0, mboxsize);

   for (int y = starty; y < stopy; y++)
   {
      drawing_area drawing(startx, totaly - y, stopx - startx, 1);
      for (int x = startx; x < stopx; x++)
      {
         color_t c = render_one_pixel(x, y, local_mbox, serial, startx,
                        stopx, starty, stop
         drawing.put_pixel(c);
      }
      if (!video->next_frame()) return;
   }
}
```

各スレッドに共通な値は並列領域の外へ

有効性の削除

実行時間
1.969 秒

変更後

```
static void parallel_thread(void)
{
   unsigned int mboxsize = sizeof(unsigned
int)*(max_objectid() + 20);
   tbb::parallel_for(starty, stopy, [mboxsize] (int y)
   {
      unsigned int serial = 1;
      unsigned int local_mbox[mboxsize];
      memset(local_mbox, 0, mboxsize);
      drawing_area drawing(startx, totaly - y, stopx - startx, 1);
      for ( int x = startx; x < stopx; x++ ) {
         color_t c = render_one_pixel(x, y, local_mbox, serial,
startx, stopx, starty, stopy);
         drawing.put_pixel(c);
      }
      video->next_frame();
   }
   );
```

インテル® TBB による並列領域

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

マルチスレッド化のまとめ

- インテル® コンパイラーのオプションを変更と若干のコード変更
- 性能向上
  - シングルスレッドのアプリケーション (最適化済み) 4.578 秒

1.8 倍の性能向上

- OpenMP* で マルチスレッド 2.562 秒

2.3 倍の性能向上

- インテル® TBB で マルチスレッド化 1.969 秒

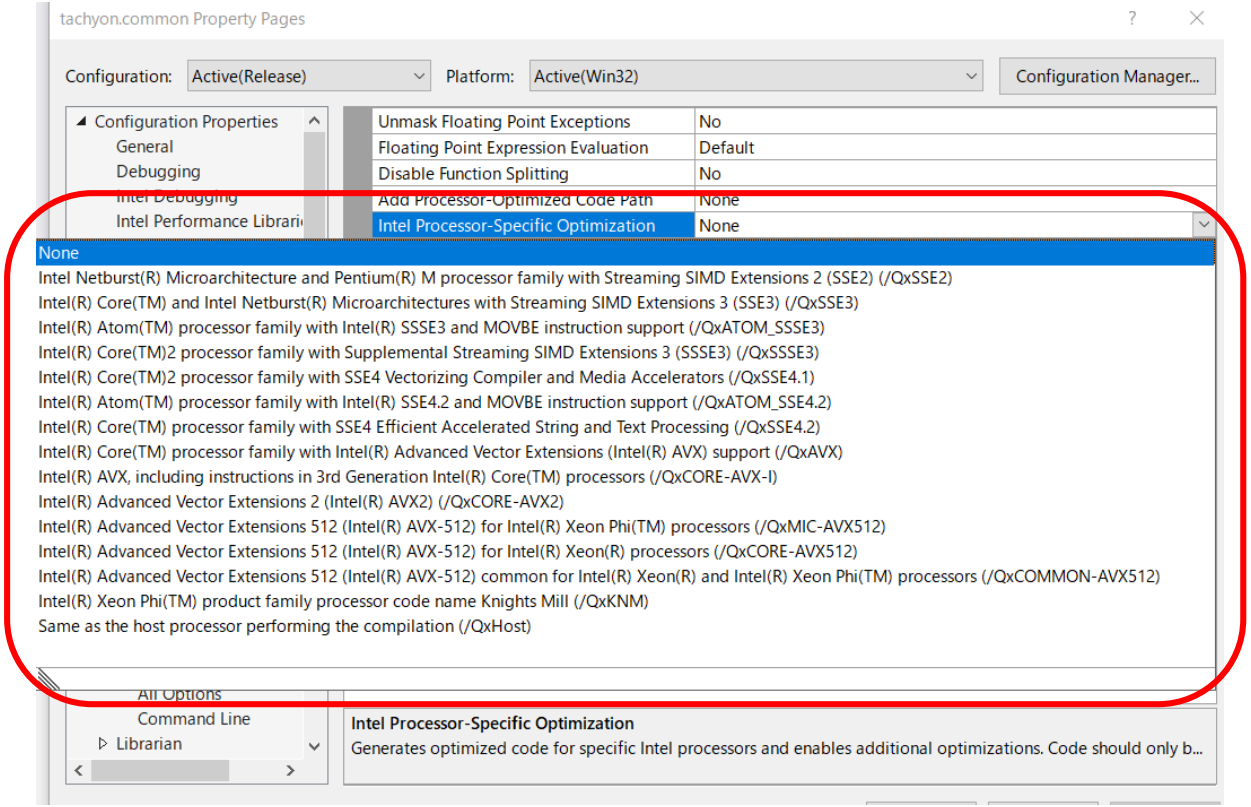
OpenMP* の移植性なども考慮

高速化にはインテル® コンパイラーが最適

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® コンパイラー特有のオプション

インテル® プロセッサー向けのオプション
(x/Qx と Ax/Qax オプション)

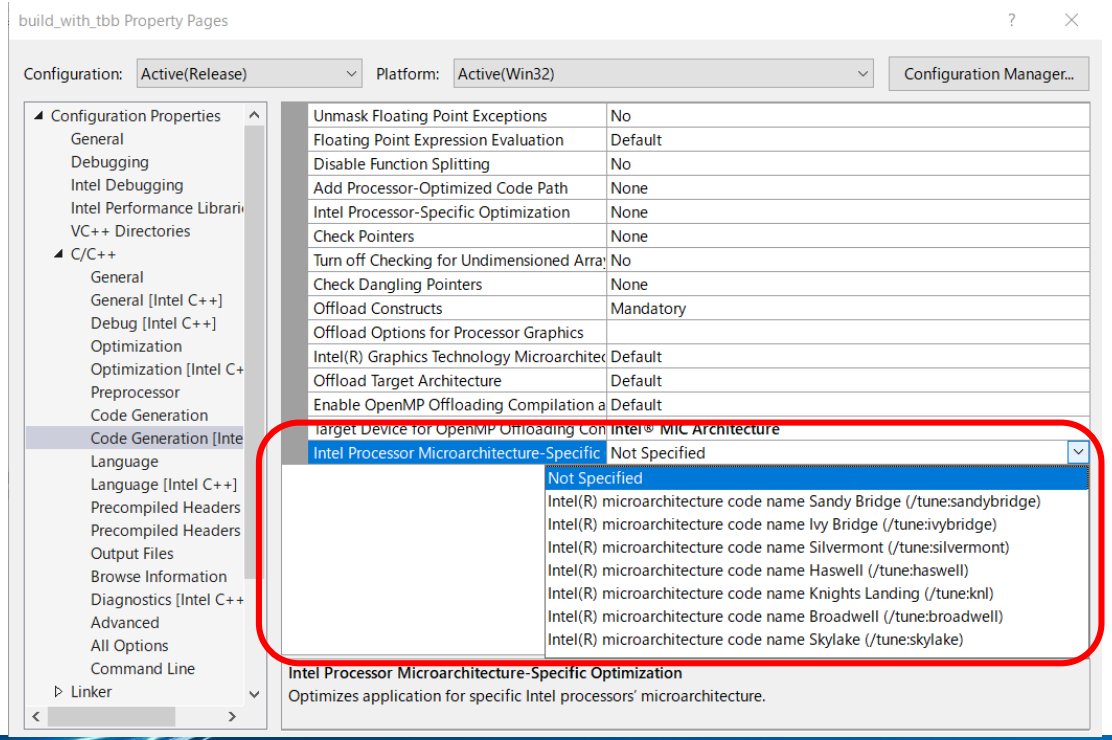


- インテル® プロセッサーごとに最適化
  - x/Qx: ターゲット・プロセッサーを限定
  - Ax/Qax: ターゲット以外のプロセッサーでも動作
  - Ax/Qax は複数のプロセッサーを指定可能

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® プロセッサー向けのオプション (/tune オプション)

- 拡張命令セットを使用せずに、特定のプロセッサーに対して最適化を実行



- 下位互換性を持ち、特定のプロセッサーに対して最適化されたコードを生成
- 以下のようなプロセッサーも指定可能
  - core-avx2
  - core-avx-i
  - corei7-avx
  - corei7
  - atom
  - core2, etc

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

まとめ

インテル® ソフトウェア開発製品

インテル® Parallel Studio XE – コンポーネント

	Composer Edition	Professional Edition	Cluster Edition
インテル® C++ コンパイラー	☑	☑	☑
インテル® Fortran コンパイラー	☑	☑	☑
インテル® Distribution for Python*	☑	☑	☑
インテル® マス・カーネル・ライブラリー (インテル® MKL)	☑	☑	☑
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)	☑	☑	☑
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)	☑	☑	☑
OpenMP*	☑	☑	☑
インテル® Advisor		☑	☑
インテル® Inspector		☑	☑
インテル® VTune™ Amplifier		☑	☑
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 (インテル® DAAL)		☑	☑
インテル® MPI ライブラリー			☑
インテル® Trace Analyzer & Collector			☑
ローグウェーブ IMSL* ライブラリー	バンドルとアドオン	アドオン	アドオン

インテル® コンパイラー 18.0 はインテル® Parallel Studio XE 2018 に含まれます。

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

まとめ

インテル® コンパイラーを使用して
アプリケーションを快適に動作

無料の 30 日間の
インテル® Parallel Studio XE 評価版を
ダウンロード:

makebettercode.com/parallelstudioxe-eval (英語) ›

皆さんのフィードバックは重要です!!

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

問い合わせ?

Q/A

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

法務上の注意書きと最適化に関する注意事項

本資料に掲載されている情報は現状のまま提供され、いかなる保証もいたしません。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を許諾するための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。

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴ、Xeon、Intel Xeon Phi、VTune は、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。

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® 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

インテル® コンパイラーの最適化オプション

O (Linux* および OS X*)	O2 を指定した場合と同じです。
O0 (Linux* および OS X*)	すべての最適化を無効にします。
	このオプションは、ほかのオプションも設定します。設定されるオプションは、使用しているオペレーティング・システムやアーキテクチャーに応じてコンパイラーにより決定され、バージョンによって異なる場合があります。
O1	速度の最適化を有効にし、コードサイズを大きくするだけで高速化に影響を与えるような一部の最適化を無効にします。
	コードのサイズを制限するために、このオプションは次のことを行います。
	・グローバルな最適化を有効にします。データフローの解析、コードの移動、ストレングス・リダクションとテストの置換、スプリット・ライフタイムの解析、および命令スケジュールが含まれます。
	・一部の組込み関数のインライン展開を無効にします。
	このオプションは、ほかのオプションも設定します。設定されるオプションは、使用しているオペレーティング・システムやアーキテクチャーに応じてコンパイラーにより決定され、バージョンによって異なる場合があります。
	O1 オプションは、分岐が多く、実行時間がループ内のコードに支配されない、非常に大きなコードサイズのアプリケーションでパフォーマンスを向上させます。
O2	速度の最適化を有効にします。一般的に推奨される最適化レベルです。 ベクトル化は O2 以上のレベルで有効になります。
	IA-32 アーキテクチャー・ベースのシステム: 分散、プレディケーション、交換、マルチバージョニング、スカラー置換などの一部の基本的なループの最適化が行われます。
	このオプションにより以下も有効になります。
	・組込み関数のインライン展開
	・次の項目を含む、イントラファイルのプロシージャー間の最適化:
	・・インライン展開、定数の伝播、前方代入、変数アドレスの解析、不要なスタティック関数の排除、未参照変数の削除など
	・パフォーマンスの向上のための機能:
	・・定数の伝播、コピーの伝播、不要コードの排除、グローバルレジスター割り当て、ループアンロール、コード選択の最適化など
	このオプションは、ほかのオプション、特にコードの速度を最適化するオプションも設定します。設定されるオプションは、使用しているオペレーティング・システムやアーキテクチャーに応じてコンパイラーにより決定され、バージョンによって異なる場合があります。
	Linux* システムでは、最適化 (オプション -O2 以上) とデバッグ (オプション -g) を有効にしてコンパイルすると、デフォルトで -debug inline-debug-info オプションが有効になります。
	共有ライブラリーの多くのルーチンは、互換マイクロプロセッサーよりもインテル製マイクロプロセッサーでより高度に最適化されます。
O3	O2 最適化を行い、融合、アンロールとジャムのブロック、IF 文の折りたたみなど、より強力なループ変換を有効にします。
	このオプションは、ほかのオプションも設定します。設定されるオプションは、使用しているオペレーティング・システムやアーキテクチャーに応じてコンパイラーにより決定され、バージョンによって異なる場合があります。
	O3 オプションを -ax または -x (Linux*)、/Qax または /Qx (Windows*) と組み合わせて使用すると、O2 よりも詳細にデータの依存性を分析します。そのため、コンパイル時間が長くなる場合があります。
	O3 の最適化は、ループ変換およびメモリーアクセス変換が行われない限り、パフォーマンスが向上しないことがあります。場合によっては、O2 の最適化よりも遅くなります。
	O3 オプションは、浮動小数点演算を多用するループや大きなデータセットを処理するループを含むアプリケーションに推奨します。
	共有ライブラリーの多くのルーチンは、互換マイクロプロセッサーよりもインテル製マイクロプロセッサーでより高度に最適化されます。

© 2017 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴは、アメリカ合衆国および / またはその他の国における
Intel Corporation の商標です。* 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。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。