intel Software

最適化クイック・リファレンス・ガイド

インテル® C++ 、Fortran コンパイラー 16.0 のオプション紹介
およびインテル® VTune™ Amplifier XE 2016 の概要

IA-32 プロセッサー、インテル® 64 プロセッサー、互換プロセッサー (非インテル製プロセッサー)、
インテル® Xeon Phi™ コプロセッサー、およびインテル® グラフィックス・テクノロジー向け。

製品と購入情報については、インテル® ソフトウェア開発ツールのサイトをご覧ください :
http://www.xlsoft.com/jp/products/category_intel.html

目次

§ § コンパイラーによるインテル® グラフィックス・テクノロジー・サポートは、オペレーティング・システムでのサポート状況に依存します。

§ インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーとインテル® Xeon Phi™ コプロセッサーは、インテル® Parallel Studio XE に含まれるコンパイラーではサポートされますが、インテル® System Studio に含まれるコンパイラーではサポートされません。

** これらのオプションは、インテル® マイクロプロセッサーおよびインテル製以外のマイクロプロセッサーの両方で利用可能ですが、インテル® マイクロプロセッサー向けのほうが、より多くの最適化が適用される場合があります。 ‡

インテル® コンパイラーでアプリケーションをチューニングする手順

パフォーマンス・チューニングを開始する前に、**/Od (-O0)** を使用して最適化を行わずにアプリケーションをビルドし、正常に動作することを確認してください。

1. 一般的な最適化オプション (Windows* では **/O1**、**/O2**、**/O3**。Linux* および OS X* では **-O1**、**-O2**、**-O3**) を使用してパフォーマンスを測定し、アプリケーションにとって最適なオプションを判断します。通常は、最初に **/O2 (-O2)** (デフォルト) を試してから、より高度な最適化を行うと効果的です
2. 次に、ループを多用するアプリケーションに対しては **/O3 (-O3)** を試します。**
3. **/Qx (-x)** や **/arch (-m)** のようなプロセッサー専用のオプションを使用してきめ細かな最適化を行います。
   例えば、第 4 世代インテル® Core™ プロセッサー・ファミリー向けには **/QxCORE-AVX2 (-xcore-avx2)** オプション、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3) 以上の命令セットをサポートするインテル製以外の互換プロセッサー向けには **/arch:SSE3 (-msse3)** オプションがあります。また、**/QxHOST (-xhost)** オプションを指定し、コンパイルするホストマシンに搭載されるプロセッサーで利用可能な最上位の命令セットを使用して最適化することもできます。**
4. プロシージャー間の最適化 (IPO) を行う **/Qipo (-ipo)** やプロファイルに基づく最適化 (PGO) を行う **/Qprof-gen** および **/Qprof-use** (**-prof-gen** と **-prof-use**) オプションを追加して、パフォーマンスを測定し、これらの最適化がアプリケーションにとって効果的かどうかを確認します。
5. インテル® Advisor とインテル® VTune™ Amplifier XE†† を使用して、シリアルあるいは並列処理におけるパフォーマンスの「hotspot」を識別し、アプリケーション・コードでさらにチューニングが必要な部分を特定することができます。コンパイラーの最適化レポート **/Qopt-report (-qopt-report)** オプションは、個々の最適化の可能性を特定するのを支援します。
6. また、C/C++ 向けのインテル® Cilk™ Plus 言語拡張や、**/Qopenmp-simd (-qopenmp-simd)**† による OpenMP* 4.0 の SIMD 機能を使用して、明示的なベクトル・プログラミングを行い、アプリケーションを最適化します。
7. マルチスレッド、マルチコア、そしてマルチプロセッサー・システム上で並列実行を最適化するため、自動並列化オプション **/Qparallel (-parallel)**、C/C++ 向けのインテル® Cilk™ Plus 言語拡張や **/Qopenmp (-qopenmp)** †
   オプションによる OpenMP* プラグマやディレクティブ、または、製品に含まれるインテル® パフォーマンス・ライブラリーを使用します。** インテル® Inspector を使用して、メモリーとスレッドのエラーを診断し、開発工程を早めてマルチスレッド・アプリケーションの市場投入までの時間を短縮します。

詳細については、製品ドキュメントのページをご覧ください: https://software.intel.com/intel-software-technical-documentation (英語)。
インテル® コンパイラーの『ユーザー・リファレンス・ガイド』には、インテル® MIC アーキテクチャーとインテル® グラフィックス・テクノロジー向けのアプリケーションのコンパイルに関する節が含まれています。
インテル® コンパイラーの『ユーザー・リファレンス・ガイド』の日本語版は、無償評価版登録ページ: https://www.xlsoft.com/jp/products/download/intelj.htmlより、インテル® Parallel Studio XE各エディションをお申し込みいただくことで入手いただけます。

** これらのオプションは、インテル® マイクロプロセッサーおよびインテル製以外のマイクロプロセッサーの両方で利用可能ですが、インテル® マイクロプロセッサー向けのほうが、より多くの最適化が適用される場合があります。‡

†OpenMP* は、インテル® Parallel Studio XE に含まれるコンパイラーではサポートされますが、インテル® System Studio に含まれるコンパイラーではサポートされません。

††製品の一部の機能は、非インテル製マイクロプロセッサー上では利用できません。

一般的な最適化オプション**

Windows*	Linux*/OS X*	説明
/Od	-O0	**最適化なし**。アプリケーション開発の初期段階およびデバッグ時に使用します。
/Os /O1	-Os -O1	**コードサイズの最小化**。オブジェクトサイズを増加させる最適化を無効にします。多くの場合に、最も小さな最適化されたコードを生成します。 このオプションは、大きなコードサイズに起因するメモリーページングが問題となる大規模なサーバー/データベース・アプリケーションで有効です。
/O2	-O2	**実行速度の最大化**。デフォルト設定。ベクトル化を含む多くの最適化を有効にします。多くの場合に、/O1 (-O1) よりも高速なコードを生成します。
/O3	-O3	/O2 (-O2) で提供される最適化に加え、キャッシュとデータ・プリフェッチをより効率良く使用するため、スカラー置換、ループアンロール、分岐を排除するコードの複製、ループ・ブロッキングなど、より積極的なループとメモリーの最適化を有効にします。 /O3 (-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。これらの積極的な最適化は、アプリケーションの種類により /O2 (-O2) と比べて遅くなることがあります。
/Qopt-report[:*n*]	-qopt-report[*n*]	最適化レポートを生成します。デフォルトでは、レポートは *.optrpt* 拡張子を持つファイルに出力されます。*n* には、*0* (レポートなし) から *5* (最も詳しい) の詳細レベルを指定します。デフォルトは *2* です。
/Qopt-report-file:*name*	-qopt-report-file=*name*	最適化レポートを *stderr*、*stdout*、またはファイル *name* に出力します。
/Qopt-report-phase:*name1, name2, …*	-qopt-report-phase=*name1, name2,…*	最適化フェーズ *name1*、*name2* 固有の最適化レポートを生成します。 *name* 引数には、以下のキーワードを指定できます。 all – すべてのフェーズのすべての最適化レポート (デフォルト) loop – ループの入れ子とメモリーの最適化 vec – 自動ベクトル化と明示的なベクトル・プログラミング par – 自動並列化 openmp – OpenMP* によるスレッド化 cg – コード生成 ipo – インライン展開を含むプロシージャー間の最適化 pgo – プロファイルに基づく最適化 offload – インテル® MIC アーキテクチャーやインテル® グラフィックス・テクノロジーへのデータ/実行のオフロード
/Qopt-report-help	-qopt-report-help	上記の /Qopt-report-phase (-qopt-report-phase) の *name* に指定可能なすべてのキーワードを表示します。コンパイルは実行されません。
/Qopt-report-routine: *substring*	-qopt-report-routine=*substring*	*substring* (部分文字列) を含む関数やサブルーチンのみをレポートします。 デフォルトでは、すべての関数とサブルーチンがレポートされます。
/Qopt-report-filter: *"string"*	-qopt-report-filter=*"string"*	*"string"* (文字列) で指定したファイル、関数、サブルーチン、行番号の範囲のみレポートします。 例: "myfile、myfun、line1-line2"

** これらのオプションのいくつかは、インテル® マイクロプロセッサーおよびインテル製以外のマイクロプロセッサーの両方で利用可能ですが、インテル® マイクロプロセッサー向けのほうが、より多くの最適化が適用される場合があります。‡

推奨するプロセッサー固有の最適化オプション**

Windows*	Linux*/OS X*	説明
/Qx*target*	-x*target*	*target* で指定した命令セットをサポートするインテル® プロセッサー向けの専用コードを生成します。実行可能ファイルは、非インテル製プロセッサーや、下位の命令セットをサポートするインテル製プロセッサー上では動作しません。*target* に指定可能な命令セットは次のとおりです (上位から下位)。 **CORE-AVX512、MIC-AVX512、COMMON-AVX512、CORE-AVX2、AVX、SSE4.2、ATOM_SSE4.2、SSE4.1、ATOM_SSSE3、SSSE3、SSE3、SSE2** **注:** このオプションは、/arch または -m オプションで有効にならない最適化を有効にします。64 ビット OS X* では、**SSE3** と **SSE2** はサポートされません。
/arch: *target*	-m*target*	*target* で指定した命令セットをサポートするすべてのインテル® プロセッサーまたは、 非インテル製互換プロセッサー向けの専用コードを生成します。指定した命令セットをサポートしていないインテル製プロセッサーや、非インテル製互換プロセッサーで実行可能ファイルを実行すると、ランタイムエラーが発生する場合があります。 *target* に指定可能な値: **AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2、IA32** **注: IA32** オプションは、専用ではない、x86/x87 汎用コードを生成します。これは、IA-32 アーキテクチャーでのみサポートされます。OS X* ではサポートされません。
/QxHOST	-xhost	コンパイルを行うホストシステム上でサポートされる最上位の命令セットを利用するコードを生成します。インテル製プロセッサー上では **/Qx (-x)** オプションに相当します。非インテル製互換プロセッサー上では **/arch (-m)** オプションの **IA32**、**SSE2**、または **SSE3** に相当します。このオプションによる一部の最適化は、インテル® マイクロプロセッサーでのみ適用される場合があります。‡
/Qax*target*	-ax*target*	*target* で指定した命令セットをサポートするインテル® プロセッサー向けの専用コードとデフォルトコード (SSE2) を生成します。 *target* に指定可能な値: **CORE-AVX512、MIC-AVX512、COMMON-AVX512、CORE-AVX2、AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2** カンマで区切った複数の値を指定することで、同じ実行ファイルにほかのインテル製プロセッサー向けにチューニングされたコードを含めることができます。 例: **/QaxAVX,SSE4.2**。デフォルトのコードパスは、**SSE2** をサポートするすべてのインテル製または非インテル製互換プロセッサー上で実行できますが、**/Qx** (**-x**) や **/arch** (**-m**) オプションを使用して変更することができます。 例えば、第 4 世代インテル® Core™ プロセッサー・ファミリー向けに最適化されたコードパスと、**SSE3** をサポートするインテル製プロセッサーまたは非インテル製互換プロセッサー向けに最適化されたデフォルトのコードパスを生成するには、**/Qax CORE-AVX2 /arch:SSE3** (Linux* では **-axcore-avx2 -msse3**) を使用します。 アプリケーションは実行時にインテル® プロセッサーで実行されているか自動検出し、最適なコードパスを選択します。インテル製プロセッサーが検出されなかった場合、デフォルトのコードパスが選択されます。 **注:** 64 ビット OS X* では、**sse3** と **sse2** オプションはサポートされません。 このオプションによる一部の最適化は、インテル® マイクロプロセッサーでのみ適用される場合があります。‡

オンライン記事「インテル® SSE およびインテル® AVX 世代 (SSE2、SSE3、SSSE3、ATOM_SSSE3、SSE4.1、SSE4.2、ATOM_SSE4.2、AVX、AVX2) 向けのインテル® コンパイラー・オプションとプロセッサー固有の最適化: http://www.isus.jp/article/compileroptimization/performance-tools-for-software-developers-intel-compiler-options/」で、プロセッサー専用の推奨最適化オプションを参照してください。

** これらのオプションは、インテル® マイクロプロセッサーおよびインテル製以外のマイクロプロセッサーの両方で利用可能ですが、インテル® マイクロプロセッサー向けのほうが、より多くの最適化が適用される場合があります。‡

並列パフォーマンス**

Windows*	Linux*/OS X*	説明
/Qopenmp†	**-qopenmp**†	OpenMP* プラグマ/ディレクティブが記述されている場合にマルチスレッド化されたコードが生成されます。Fortran では、ローカル配列が自動になり、スタックサイズの増加が必要なことがあります。
/Qopenmp-simd†	**-qopenmp-simd**†	OpenMP* SIMD ディレクティブが記述されている場合に SIMD コードが生成されます。
/Qopenmp-stubs	**-qopenmp-stubs**	OpenMP* ディレクティブを無視し、OpenMP* ランタイム・ライブラリー関数への参照をシングルスレッド処理と仮定してスタブ (ダミー) 関数にリンクします。
/Qparallel	**-parallel**	自動並列化は、安全に並列実行できる構造のループ (DO CONCURRENT 構文とインテル® Cilk™ Plus の配列表記による暗黙のループを含む) を検出し、ループのマルチスレッド・コードを自動生成します。
/Qpar-threshold[:*n*]	**-par-threshold[*n*]**	パフォーマンス向上の可能性に基づいて、ループの自動並列化のしきい値を設定します。n=0 から 100 が指定でき、デフォルトは 100 です。 ***0*** – 計算量にかかわらずループを並列化します。 ***100*** – パフォーマンス上の利点があると思われる場合にのみループを並列化します。 **/Qparallel** (**-parallel**) オプションを併用する必要があります。
/Qpar-affinity: *name*	**-par-affinity= *name***	OpenMP* や自動並列化アプリケーション向けにスレッドとプロセッサーのアフィニティーを指定します。***name*** に指定できる値は、***none*** (デフォルト)、***scatter***、***compact*** などです。メインプログラムをコンパイルする場合にのみ効果があります。設定と詳細な情報については、インテル® コンパイラーの『ユーザー・リファレンス・ガイド』をご覧ください。
/Qopt-matmul[-]	**-q[no-]opt-matmul**	コンパイラーが生成する行列乗算 (matmul) ライブラリー呼び出しを有効 [無効] にします。行列乗算におけるループの入れ子 (該当する場合) を特定し、ループを matmul ライブラリー呼び出しに置換して、パフォーマンスを向上します。**/O3 (-O3)** と **/Qparallel** (**-parallel**) を指定すると、デフォルトで有効になります。このオプションは、**/O2 (-O2)** 以上を指定しない限り、効果はありません。
/Qcilk-serialize	**-cilk-serialize**	ヘッダーファイル cilk_stubs.h のインクルードし、コンパイラーにインテル® Cilk™ Plus のスレッド化キーワードを無視させ、シリアル実行コードを生成します (C/C++ のみ)。詳細は、インテル® コンパイラーの『ユーザー・リファレンス・ガイド』の「インテル® Cilk™ Plus プログラムのビルド、実行、デバッグ」をご覧ください。
/Qcoarray	**-coarray**	Fortran 2008 の Co-Array 機能を有効にします (Fortran のみ)。***shared***、***distributed***、***coprocessor***、***single*** オプションを利用できます。詳細は、インテル® コンパイラーの『ユーザー・リファレンス・ガイド』をご覧ください。
/Qmkl:*name*	**-mkl=*name***	インテル® マス・カーネル・ライブラリー (インテル® MKL) とのリンクを要求します。デフォルトはオフです。***name*** に設定可能な値は以下のとおりです。 ***parallel*** – スレッド化されたインテル® MKL をリンク (デフォルト) ***sequential*** – スレッド化されていないインテル® MKL をリンク ***cluster*** – MPI が実装されたシーケンシャルなインテル® MKL をリンク

** これらのオプションは、インテル® マイクロプロセッサーおよびインテル製以外のマイクロプロセッサーの両方で利用可能ですが、インテル® マイクロプロセッサー向けのほうが、より多くの最適化が適用される場合があります。‡

プロシージャー間の最適化 (IPO) オプションとプロファイルに基づく最適化 (PGO) オプション

Windows*	Linux*/OS X*	説明
/Qip	-ip	現在のソースファイルを対象にしたインライン展開を含む、単一ファイルのプロシージャー間の最適化を行います。
/Qipo[*n*]	-ipo[*n*]	複数のソースファイルにまたがるインライン展開やその他のプロシージャー間の最適化を許可します。オプションの引数 ***n*** は、リンク時コンパイルの最大数 (オブジェクト・ファイル数) を制御します。デフォルトは、***n=0*** です (コンパイラーが判断します)。 注: 状況によってコンパイル時間とコードサイズが大幅に増えることがあります。
/Qipo-jobs[*n*]	-ipo-jobs[*n*]	プロシージャー間の最適化 (IPO) のリンクフェーズで、同時に実行するコマンド (ジョブ) の数を指定します。デフォルトは ***1*** です。
/Ob2	-finline-functions -finline-level=2	コンパイラーの判断により、現在のソースファイル内で関数のインライン展開を有効にします。**/O2** と **/O3** (**-O2** と **-O3**) では自動的に有効になります。 注: ファイルサイズが大きな場合、コンパイル時間とコードサイズが大幅に増えることがあります。**/Ob0** (Linux* と OS X* では **-fno-inline-functions**) で無効にできます。
/Qinline-factor:*n*	-finline-factor=*n*	インライン展開できる関数の合計サイズと最大サイズの係数を設定します。デフォルトは ***n=100*** で、100% またはスケーリング係数 1 を示します。
/Qprof-gen [*:kywd*]	-prof-gen [=kywd]	プロファイル情報を生成するためインストルメントを行います。 ***kywd= threadsafe*** は、スレッド化されたアプリケーションのプロファイル生成を可能にします。***kywd=srcpos*** と ***globdata*** は、関数やデータの並び替えに役立つ追加情報を収集します。
/Qprof-use	-prof-use	最適化でプロファイル情報を使用するようにします。
/Qprof-dir *dir*	-prof-dir *dir*	プロファイル結果の出力ファイル (*.dyn および *.dpi) を格納するディレクトリーを指定します。
PROF_DIR	PROF_DIR	プロファイル出力ファイルのディレクトリーを設定します (**/Qprof-dir** や **-prof-dir** で使用します)。
/Qprofile-functions	-profile-functions	各関数の実行時間を生成するようにインストルメント関数を追加します。
/Qprofile-loops	-profile-loops	関数をインストルメントし、各ループまたはループの入れ子のシリアルコードのプロファイルを生成します。プロファイルの表示方法やその他の詳細については、インテル® コンパイラーの『ユーザー・リファレンス・ガイド』で、「関数またはループの実行時間のプロファイル」をご覧ください。

プロファイルに基づく最適化 (Profile Guided Optimiztion、PGO) の使い方:

PGO は条件分岐によるペナルティを減少させるように、再コンパイル時に実行内容のプロファイルから動的な最適化を行います。

1. /Qprof-use (-prof-use) を指定してコンパイルします。
2. 生成されたバイナリーを実行します。実行するごとにプロファイル (拡張子 .dyn) が生成されます。
3. /Qprof-use (-prof-use) を指定してコンパイルします。複数のプロファイルがある場合、平均値が最適化に適用されます。
4. プロファイルに基づいて最適化されたバイナリーが生成されます。

浮動小数点演算オプション

Windows*	Linux*/OS X*	説明
/fp:*name*	-fp-model *name*	特定の最適化を制限することで、浮動小数点演算のパフォーマンス、精度、再現性を制御します。 *name* に指定可能な値は以下のとおりです。 **Fast[=1|=2]** – 精度と一貫性を多少低くすることにより、さらに強力な最適化が可能になります。(デフォルトは **fast=1**)。 **precise** – 精度に影響しない最適化のみ有効にします。 **double/extended/source** – 中間結果をそれぞれ倍精度、拡張精度、ソースの精度で丸めます。オーバーライドされない限り **precise** も適用されます。インテル® Fortran コンパイラーでは、**double** と **extended** はサポートされません。 **except** – 浮動小数点例外セマンティクスを強制します。 **strict** – **precise** と **except** オプションを有効にし、デフォルトの浮動小数点環境を仮定しません。コンパイラーが、FMA (Fused Multiplty Add) 命令を生成しないように強制します。 **推奨事項:** 浮動小数点の一貫性と再現性が必要とされる多くの状況では、**/fp:precise /fp:source (-fp-model precise -fp-model source)** が推奨されます。
/Qopt-dynamic-align[-]	-q[no-]opt-dynamic-align	実行時のデータ・アライメントに依存し、同じシリアル・アプリケーションを同じ入力データで繰り返し実行した際に、わずかな浮動小数点結果の違いを引き起こす可能性のある最適化を有効 [無効] にします。デフォルトでは、**/fp:precise** (**-fp-model precise**) が指定されない限り有効です。
/Qftz[-]	-ftz[-]	main プログラムや dll の main をこのオプションでコンパイルすると、プログラム (dll) 全体 で実行時にインテル® SSE やインテル® AVX 命令によるデノーマル結果がゼロにフラッシュされます。デフォルトでは、**/Od (-O0)** が指定されない限り有効です。
/Qimf-precision: *name*	-fimf-precision: *name*	算術ライブラリー関数の精度を設定します。デフォルトはオフです (コンパイラーは、デフォルトのヒューリスティックを使用します)。*name* には、*high*、*medium*、*low* を指定できます。精度を下げると、特にベクトル化されたコードのパフォーマンスが向上する可能性がありますが、その逆もあり得ます。
/Qimf-arch-consistency: *true*	-fimf-arch-consistency=*true*	算術ライブラリー関数が、同じアーキテクチャーの異なるインテル製プロセッサーまたは非インテル製互換プロセッサーにおいて一貫した結果を生成するようにします。これにより、ランタイム・パフォーマンスが低下することがあります。デフォルトは、 "*false*"(オフ) です。
/Qprec-div[-]	-[no-]prec-div	浮動小数点除算の精度を上げ [下げ] ます。パフォーマンスが低下 [向上] することがあります。
/Qprec-sqrt[-]	-[no-]prec-sqrt	平方根計算の精度を上げ [下げ] ます。パフォーマンスが低下 [向上] することがあります。
/Qprotect-parens[-] /assume:[no]protect_parens	-fprotect-parens[-] -assume [no]protect_parens	(C/C++ オプション) 括弧で指定した順序で式が評価されます。 (Fortran オプション) デフォルトでは、/fp:precise (-fp-model precise) が指定されない限りオフです。
/Qfma[-]	-[no]fma	コンパイラーが、FMA (Fused Multiplty Add) 命令を生成しないように強制します (ただし、ランタイム・ライブラリーでは FMA 命令が使用される可能性があります)。

http://www.isus.jp/article/compileroptimization/consistency-of-floating-point-results/ をご覧ください。

きめ細かなチューニング (すべてのプロセッサー)

Windows*	Linux*/OS X*	説明
/Qunroll[*n*]	-unroll[*n*]	ループアンロール回数の上限を設定します。 ***n=0*** はループアンロールを無効にします。デフォルトは、デフォルトのヒューリスティックを使用する **/Qunroll (-unroll)** です。
/Qopt-prefetch:*n*	-qopt-prefetch=*n*	さまざまなレベルのソフトウェア・プリフェッチを有効にします。***n*** は、***0*** (プリフェッチなし) から ***4*** (最大限のプリフェッチ) の任意の値です。**/O3 (-O3)** を指定した場合、***n*** のデフォルトは ***2*** です。警告: 過度のプリフェッチは、リソースの競合によりパフォーマンスが低下することがあります。
/Qopt-block-factor:*n*	-qopt-block-factor=*n*	デフォルトのヒューリスティックをオーバーライドするブロッキング係数 ***n*** (ブロック内のループ反復回数) を指定します。ループ・ブロッキングは、**/O3** (**-O3**) で有効になり、キャッシュのデータを再利用するように設計されています。
/Qopt-streaming-stores:*mode*	-qopt-streaming-stores *mode*	ストリーミング・ストアの生成を有効/無効にします。***mode*** に指定可能な値は次のとおりです。 **always** -アプリケーションのメモリー再利用が少ないと想定し、キャッシュをバイパスするストリーミング・ストアの生成を促進します。 **never** - ストリーミング・ストアの生成を無効にします。 **auto** - ストリーミング・ストアの生成にデフォルトのコンパイラーのヒューリスティックを使用します。
/Qrestrict[-]	-[no]restrict	**restrict** キーワードによるポーンターの一義化を有効 [無効] にします。 デフォルトはオフです。(C/C++ のみ)。
/Oa	-fno-alias	プログラムにエイリアシングがないことを想定します。デフォルトはオフです。
/Ow	-fno-fnalias	関数内にエイリアシングがないことを想定します。デフォルトはオフです。
/Qalias-args[-]	-fargument-[no]alias	関数の引数のエイリアス化を有効 [無効] にします。デフォルトはオンです。(C/C++ のみ)。**-fargument-noalias** は、配列引数を持つ関数呼び出しを含むループのベクトル化に役立ちます。
/Qansi-alias[-]	-[no-]ansi-alias	ANSI および ISO C 標準の別名規則を有効 [無効] にします。 デフォルト: Windows* では無効、Linux* と OS X* では有効。
/Qopt-class-analysis[-]	-q[no-]opt-class-analysis	C++ クラス階層情報を使用して、コンパイル時に C++ 仮想関数の呼び出しを解決します。C++ アプリケーションに標準的ではない C++ 構造 (ポインターのダウンキャスティングなど) が含まれている場合、動作が異なることがあります。デフォルトではオフですが、**/Qipo** (Windows*) や **-ipo** (Linux* と OS X*) オプションを指定すると、C++ の最適化を高めるためオンになります。(C++ のみ)。
/Qvec-threshold:*n*	-vec-threshold=*n*	パフォーマンス向上の可能性に基づいて、ループの自動並列化のしきい値 ***n*** を設定します。デフォルトは ***n=100*** です。 ***0*** – 計算量にかかわらずループをベクトル化します。 ***100*** – パフォーマンス上の利点が確実である場合にのみループをベクトル化します。
/Qvec[-]	-[no-]vec	ベクトル化を有効 [無効] にします。デフォルトは、**/O2** (**-O2**) で有効です。
/align: array*nn*byte	-align array*nn*byte	アライメントされたロードとベクトル化を支援するため、***nn*** で割り切れるメモリーアドレスに配列の先頭を配置するように指示します。(Fortran のみ)。
/assume [no] buffered_io	-assume [no] buffered_io	I/O 効率を向上するため、連続するシーケンシャルな読み書きデータをバッファリングします。デフォルトでは、バッファリングしません。(Fortran のみ)。
	-f[no-] exceptions	**-fexceptions** は、例外処理テーブルの生成を有効にします (C++ のデフォルト)。 **-fno-exceptions** (C と Fortran のデフォルト) は、コードサイズが小さくなります。 C++ では、例外指定は解析されますが、無視されます。 構造化例外処理 (try ブロックや throw 文) は、呼び出しに -fno-exceptions でコンパイルされた関数が含まれている場合、エラーになります。

インテル® Xeon Phi™ コプロセッサー x100 製品ファミリー向けの最適化§
(開発コード名 Knights Corner)

Windows*	Linux*	説明
/Qmic	-mmic	インテル® Xeon Phi™ コプロセッサー x100 製品ファミリーでネイティブに実行するアプリケーションをビルドします。(デフォルトはオフです)。
/Qopt-streaming-cache-evict:*n*	-qopt-streaming-cache-evict=*n*	ストリーミング・ストアの後にキャッシュ・エビクション (追い出し) 命令を生成するかどうかを制御します。*n=0* 追い出しなし、*n=1* L1 のみ追い出し、*n=2* L2 のみ追い出し (デフォルト)、*n=3* L1 と L2 を追い出し。
/Qopt-assume-safe-padding	-qopt-assume-safe-padding	ユーザープログラムと同様に、コンパイラーが配列や動的に割り当てられたオブジェクトの終端から最大 64 バイトを超えて安全にアクセスできることを示します。パディングの追加はユーザーの責任です。デフォルトはオフです。
/Qopt-threads-per-core:*n*	-qopt-threads-per-core=*n*	物理コアあたり *n* スレッド向けに最適化するようにコンパイラーに知らせます。*n* は、1、2、3、または 4 です。
/Qopt-prefetch:*n*	-qopt-prefetch=*n*	ソフトウェア・プリフェッチのレベルを *n=0* から *4* で指定します。最適化レベル -O2 以上のデフォルトは *n=3* です。
/Qimf-domain-exclusion:*n*	-fimf-domain-exclusion=*n*	数学関数が IEEE 標準に準拠する必要がない、特殊なケースの引数を指定します。n のビットは以下に相当します。 *0* – 極値 (非常に大きい、非常に小さい、特異値に近いなど)、*1* – NaN、*2* – 無限大、*3* – デノーマル値、*4* – ゼロ。
/Qopt-gather-scatter-unroll	-qopt-gather-scatter-unroll	集約 (Gather) / 分散 (Scatter) ループの代替アンロールシーケンスを指定します。
/align: array64byte	-align array64byte	アライメントされたロードとベクトル化を支援するため、64 で割り切れるメモリーアドレスに配列の先頭を配置するように指示します。(Fortran のみ)。
__MIC__	__MIC__	インテル® MIC アーキテクチャー向けのプログラミングで利用できる事前定義済みマクロです。 ※本オプションで使用される ”__” は、アンダーバー ”_” 2つ分となります。

インテル® Xeon Phi™ コプロセッサー向け環境変数§

変数	説明
OFFLOAD_REPORT=<*n*>	オフロード・アプリケーションのランタイムレポートを生成します。 *n*=1 ホストとコプロセッサー上の実行時間をレポートします。 *n*=2 ホストとコプロセッサー間のデータ転送もレポートします。 *n*=3 デバイスの初期化と個々の変数の転送を含む詳細をレポートします
OFFLOAD_DEVICES= <*n1,n2,…*>	ホスト上のプロセスが使用する物理コプロセッサーを *n1*、*n2* などに制限します。コプロセッサーの番号は *0* から始まります。
MIC_STACKSIZE=<*n*>M	オフロード・アプリケーション向けにコプロセッサー上の最大スタックサイズを設定します。この例では、*n* はメガバイトです。
MIC_ENV_PREFIX=<name>	オフロード・アプリケーション向けに、ホストとコプロセッサー上の環境変数を区別するためプリフィックスを指定します。例えば、*name*=MIC の場合、MIC_OMP_ NUM_THREADS がコプロセッサー上の OpenMP* スレッドの数を制御します。
MIC_USE_2MB_ BUFFERS=<*n*>M	実行時に *n* MB を越えるオフロードポインター変数は、2MB のラージページに割り当てられます。

詳細は、http://www.isus.jp/article/mic-article/low-precision-optimizations/ をご覧ください。インテル® MIC アーキテクチャーについては、http://www.isus.jp/article/idz/mic-developer/ と http://www.isus.jp/article/mic-article/xeon-phi/ をご覧ください。
インテル® コンパイラーの『ユーザー・リファレンス・ガイド』の日本語版は、無償評価版登録ページ: https://www.xlsoft.com/jp/products/download/intelj.htmlより、インテル® Parallel Studio XE各エディションをお申し込みいただくことで入手いただけます。

§ インテル® MIC アーキテクチャーとインテル® Xeon Phi™ コプロセッサーは、インテル® Parallel Studio XE に含まれるコンパイラーではサポートされますが、インテル® System Studio に含まれるコンパイラーではサポートされません。

オフロード向けコンパイル

Windows*	Linux*	説明
/Qoffload[-] /Qoffload[:*kywd*]	-q[no-]offload -qoffload=*kywd*	インテル® MIC アーキテクチャーやインテル® グラフィックス・テクノロジーへのオフロード言語構文を制御します。*kywd* には次の値を指定できます。 *none*: オフロード構文は無視され、すべてのコードはホスト上でのみ実行するようにコンパイルされます。**/Qoffload-** (**-qno-offload**) と同じです。 *mandatory*: status 句が指定されておらずターゲットが利用できない場合、プログラムは失敗しオフロードはスキップされます。 *optional*: ターゲットが利用できない場合、すべてのコードはホストで実行されます。 デフォルトは、**/Qoffload:mandatory** (**-qoffload=mandatory**) です。
/Qoffload-option, *target, tool, "option-list"*	-qoffload-option, *target, tool, "option-list"*	ターゲット向けのコンパイルにのみ使用するオプションを指定します。*target* には、*mic* (インテル® MIC アーキテクチャー) または *gfx* (インテル® グラフィックス・テクノロジー) を指定できます。*tool* には、*compiler*、*ld*、*link*、または *as* を指定できます。
/Qoffload-attribute-target: *target-name*	-qoffload-attribute-target =*target-name*	ファイルスコープ関数やデータ・オブジェクトにオフロード属性 target(*target-name*) のフラグを付けます。*target-name* には、*mic* (インテル® MIC アーキテクチャー) または *gfx* (インテル® グラフィックス・テクノロジー) を指定できます。
/Qopt-report-phase:*offload*	-qopt-report-phase *offload*	ホストとコプロセッサー、もしくはホストとプロセッサー・グラフィックス間でコピーされる変数のレポートをコンパイル時に生成します。
__INTEL_OFFLOAD	__INTEL_OFFLOAD	ホスト上のオフロード・プログラミングで利用できる事前定義マクロです。

インテル® グラフィックス・テクノロジー向けコンパイル§§

Windows* (32 および 64 ビット)	Linux* (64 ビットのみ)	説明
/Qgpu-arch:*arch*	-mgpu-arch=*arch*	*arch* で指定したインテル® マイクロアーキテクチャー開発コード名 *ivybridge*、*haswell*、または *broadwell* 上のプロセッサー・グラフィックス向けのネイティブ命令を生成します。デフォルトは、JIT エンジンで翻訳される仮想命令です。
__GFX__	__GFX__	インテル® グラフィックス・テクノロジー向けのプログラミングで利用できる事前定義済みマクロです。

インテル® グラフィックス・テクノロジー向け環境変数§§

変数	説明
GFX_CPU_BACKUP=1	ターゲットが利用できない場合、オフロードコードはホストで実行されます (デフォルト)。*0* の場合、ターゲットが利用できないとアプリケーションは失敗します。
GFX_MAX_THREAD_COUNT	ループの入れ子を並列化する際のターゲットスレッドの最大数を制御します。 デフォルトは *-1* (システムのデフォルト) です。
GFX_OFFLOAD_TIMEOUT=*n*	オフロードタスクは *n* 秒でタイムアウトします (デフォルトは *n=60*)。このオプションを有効にするには、システム回復タイムアウトを無効にするか、増やす必要があるかもしれません。
GFX_SHOW_TIME=1	実行の最後にオフロードのタイミング情報を出力します。デフォルトは *0* (出力しない) です。
GFX_LOG_OFFLOAD=*n*	オフロードのログを生成します。*n* には、*0* (ログなし) から *3* (最も詳しい) の詳細レベルを指定します。デフォルトは *0* です。

§§ コンパイラーによるインテル® グラフィックス・テクノロジー・サポートは、オペレーティング・システムでのサポート状況に依存します。

詳細は、https://software.intel.com/articles/getting-started-with-compute-offload-to-intelr-graphics-technology (英語) にある入門ガイドや、インテル® コンパイラーの『ユーザー・リファレンス・ガイド』をご覧ください。

インテル® Cilk™ Plus を使用した並列パフォーマンス

スレッド化のキーワード	説明 (C/C++ のみ)
cilk_spawn	インテル® Cilk™ Plus のランタイムによって動的にスケジュールされた場合、スポーンされた関数を呼び出し元と並列に実行できるようにします (強制はしません)。
cilk_sync	バリアを定義し、スポーンされたすべての子が完了するまで関数を待機させます。
cilk_for	反復を並列に実行できる (強制はしません) for ループを定義します。

レデューサー: 合計の累積など、安全に並列実行できるリダクション操作を許可します。

例: **cilk::reducer< cilk::op_add<unsigned int> >** は符号なし整数を合計するレデューサーを定義。

ホルダー: cilk::holder テンプレート・クラスは、便利でスレッドセーフなタスク・ローカル・ストレージを提供します。

配列表記 (アレイ・ノーテーション): 分かりやすく明示的なデータ並列処理の C/C++ 言語拡張で、最適化レベル **-O2** 以上でベクトル化による SIMD 並列コードの生成を可能にし、依存関係がないことを明示します。

構文: **array[<lower bound>:<length>:<stride>]**

例: **bb[:][:] = 0** は、2 次元配列 bb 全体にゼロを代入します (コンパイラーがサイズと型を認識できなければなりません)。
c[j:len] = sqrt(c[k:len:2]) は、c[k] から始まる 1 つ飛びの要素の len 個の平方根を求め、c[j] から始まる要素に格納します。また、(j<k であれば) ベクトル化しても安全であることをコンパイラーに知らせます。

リダクション関数が用意されています。例: **__sec_reduce_add(a[:]** は配列 a のすべての要素の合計を返します。※本関数で使用される "__" は、アンダーバー "_" 2つ分となります。

SIMD 対応関数: 関数をスカラーまたは SIMD モードで呼び出し、関数呼び出しを含むループを効率良くベクトル化できるようにする言語拡張。コンパイラーは、1 つ以上のスカラー引数がベクトルに置き換えられた代替関数を生成します。

C/C++ 構文: **__declspec (vector(*clauses*)) func_name(arguments)** (または __attribute__)
※上記構文の**__declspec**、__attribute__ で使用される "__" は、アンダーバー "_" 2つ分となります。

Fortran 構文: **!DIR$ ATTRIBUTES VECTOR: (clauses) :: func_name**

オプションの clauses (節)には、*uniform*、*linear*、*mask*、*processor*、*vectorlength* を指定できます。

ベクトルバージョンの関数は、配列表記を使用して直接呼び出すか、ループなどから間接的に呼び出されます。

```
a[:] = func_name(b[:],c[:],d,..) ;
for (int i=0; i<n; i++)    a[i] = func_name(b[i],c[i],d,…) ;
DO J=1,N;    A(J) = FUNC_NAME(B(J),C(J),D,…) ;    ENDDO
```

同様の機能が、OpenMP* 4.0† の **DECLARE SIMD** でもサポートされます。SIMD 対応関数を含むループのベクトル化を確実にするため、SIMD プラグマやディレクティブが必要になることがあります。

SIMD プラグマ (C/C++) とディレクティブ (Fortran) を使用した明示的なベクトル・プログラミング

SIMD 命令を使用してループをベクトル化するようにコンパイラーに指示します。プログラマーは、正当性 (明示的にプライベート変数やリダクションを指定するなど) に関して責任があります。セマンティクスは、OpenMP* の #pragma omp parallel for (C/C++) と !$OMP PARALLEL DO (Fortran) 宣言と同じです。

コンパイラーは、OpenMP* 4.0† の SIMD 句で同様の機能をサポートします。

C/C++ 構文: **#pragma simd [clauses]**　　Fortran 構文: **!DIR$ SIMD [clauses]**

節	説明
private(*var1,var2,…*)	競合を避けるため、ループの各反復でプライベートにする変数を指定します。
reduction(oper:var1,var2,…)	変数 *var1, var2, …* にオペレーター oper を使用するベクトル・リダクション操作を行うようにコンパイラーに指示します。
linear(var1:step1,…)	スカラーループの各反復で、変数 var1 は step1 ずつインクリメントされます。

ほかにも、firstprivate、lastprivate、[no]assert、vectorlength、vectrolengthfor、vecremainder 節がサポートされます。

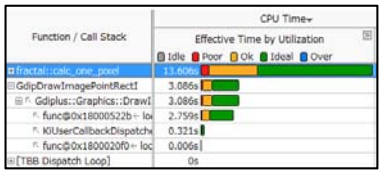
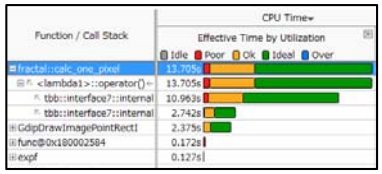
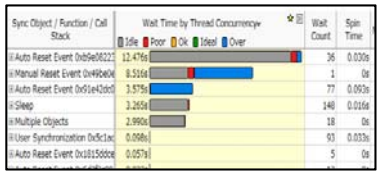
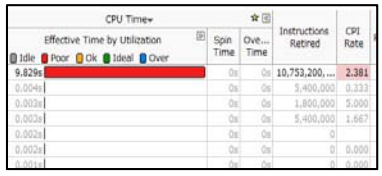
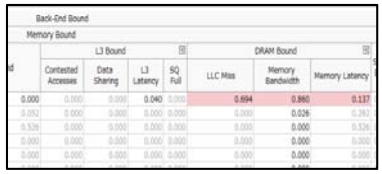
_Simd と **_Reduction** キーワードは、**#pragma simd reduction(...)** の代替手段を提供します。

詳細は、http://www.isus.jp/article/intel-cilk-plus/ とインテル® コンパイラーの『ユーザー・リファレンス・ガイド』をご覧ください。

インテル® VTune™ Amplifier XE を使用したパフォーマンス解析

Hotspot: コードの中で CPU を使用してる時間が最も多い関数またはループの箇所です。
ボトルネック: コードの中で全体の最大の実行速度に影響する、最も遅い箇所です。
ボトルネックの原因: ジャンプするメモリー参照や、分岐予測ミスなどの処理が遅延の原因になります。

インテル® VTune™ Amplifier XE を用いてパフォーマンス解析を行うことで、
1. Hotspot を発見し、
2. その箇所がボトルネックであるかを検出し、
3. ボトルネックの原因を特定して修正することで、パフォーマンスを改善させることができます。

主要な解析タイプ	説明
Basic Hotspots (コマンドライン解析タイプ: hotspots)	Hotspot (CPU 時間を要しているソースコードの箇所) を特定します。 実際の CPU 時間とともに、表の一番上に最も CPU 時間を要している Hotspot が表示されます。 
Concurrency (コマンドライン解析タイプ: concurrency)	プログラムのマルチスレッド動作を解析します。 実時間 (Elapsed Time) の内、プロセッサーのすべてのコアを同時に利用できていた時間が長いほど、Ideal (望ましい) と分類されます。 
Locks & Waits (コマンドライン解析タイプ: locksandwaits)	待ち状態による CPU 時間の浪費状況を解析します。 マルチスレッドにおけるクリティカルセクションでの同期や、入出力待ちなどが性能に影響しているかどうかを確認できます。 
Advanced Hotspots (コマンドライン解析タイプ: advanced-hotspots)	プログラムの各場所での CPI 値を求めます。 CPI とは、実行に要した CPU サイクル数と、実行された CPU 命令数の比であり、プログラムの実行効率を表す基本的な評価基準です。 
General Exploration (コマンドライン解析タイプ: general-exploration)	キャッシュミスや分岐予測ミスなど、CPU の動作に関連する詳細な性能情報やボトルネックを取得します。 解析結果において性能劣化の原因と見られるボトルネックを特定します。ボトルネックの原因となる項目や値はピンク色で強調表示されます。 

デバッグオプション

Windows*	Linux*/OS X*	説明
/Zi /debug /debug:full /debug:all	-g -debug -debug full -debug all	最適化されていないコードの完全なシンボリック・デバッグのため、一般的な開発環境のデバッガー向けにデバッグ情報を生成します。最適化オプション **/O2 (-O2)** (またはほかの **O** オプション) を指定した場合に有効になります。デバッグシンボルの生成は、一般的にオブジェクト・モジュールのサイズを増加させ、最適化されたコードのパフォーマンスをわずかに低下させることがあります。
/debug:none	-debug none	デバッグ情報は生成されません。(デフォルト)
/debug: minimal	-debug minimal	ローカルシンボルではなく、デバッグ用の行番号情報を生成します。
/debug:inline-debug-info	-debug inline-debug-info	このオプションを指定すると、インライン展開される関数のシンボルは、呼び出し元ではなく、呼び出される関数のソースに関連付けられます。 **-O2** を指定しない限り、**/debug:full** (**-debug full**) だけでは有効になりません。
	-debug extended	最適化されたコードのシンボリック・デバッグを改善するため、追加情報を生成します。(Linux* のみ。**debug full** では有効になりません。)
	-debug parallel	スレッド化されたコードのデバッグのため、追加のシンボルとインストルメント・コードを生成します。(Linux* のみ。**debug full** では有効になりません。)
/Qsox[-]	-[no-]sox (Linux* のみ)	オブジェクト・ファイル (Windows* と Linux*) と実行可能ファイル (Linux*) に、コンパイラーのバージョンとオプションを文字列で埋め込みます。デフォルトはオフです。
/Qtraceback	-traceback	ランタイム時に致命的なエラーが発生したとき、ソースファイルのトレースバック情報を表示できるように、オブジェクト・ファイル内に補足情報を生成します。最適化されたコードで利用します。(Fortran のみ)
/Qinit:snan, arrays	-init=snan, arrays	実行時に初期化されていない浮動小数点変数を検出しやすくするため、特定の浮動小数点変数をシグナル型 NaN に初期化します。(Fortran のみ)。浮動小数点例外のマスクを解除するには、/fpe:0 (-fpe0) を指定します。

‡ 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

製品と購入情報については、インテル® ソフトウェア開発ツールのサイトをご覧ください:
http://www.xlsoft.com/jp/products/category_intel.html