インテル® C++ コンパイラーを使用したパフォーマンスの最適化

yang-wang (Intel) 2014 年 10 月 9 日公開、2017 年 12 月 5 日更新

この記事は、2017 年 12 月 18 日時点の、インテル® デベロッパー・ゾーンに公開されている「Step by Step Performance Optimization with Intel® C++ Compiler」の日本語訳です。

一般に、インテル® C++ コンパイラーを使用したパフォーマンスの最適化には、次の 6 つのステップがあります。

1. 最適化なしでコンパイルする
2. 一般的な最適化を有効にする
3. プロセッサー固有の最適化を有効にする
4. IPO 最適化を使用する
5. PGO 最適化を使用する
6. 自動ベクトル化のチューニング

1. 最適化なしでコンパイルする

これは最初のステップです。パフォーマンス・チューニングは、適切なアプリケーションをベースにする必要があります。パフォーマンス・チューニングを開始する前に、アプリケーションの正当性を確認します。簡単にデバッグできるように、このステップでは通常、-O0 オプションを使用します。GNU* コンパイラーまたはインテル® C++ コンパイラーを使用してすでにアプリケーションを検証している場合は、次のステップから開始してください。

2. 一般的な最適化を有効にする

このステップでは、最も一般的なコンパイラーの最適化を有効にします。アプリケーションに応じて、以下のいずれかのオプションを選択してください。

-O1 および -Os (Linux*) または /O1 および /Os (Windows*)

このオプションは、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。また、このオプションは、一部の組込み関数のインライン展開を無効にします。O1 オプションが指定された場合、デフォルトで Os オプションが有効になり、コードサイ

ズが増えない最適化が行われます。O1 オプションを使用すると、コンパイラーの自動ベクトル化機能は無効になります。

-O2 (Linux*) または /O2 (Windows*)

このオプションは、コードの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。コンパイラーのベクトル化は O2 以上のレベルで有効になります。このオプションを使用すると、コンパイラーは、一部の基本的なループの最適化、組込み関数のインライン展開、イントラファイルのプロシージャー間の最適化、最も一般的なコンパイラーの最適化テクノロジーを実行します。

-O3 (Linux*) または /O3 (Windows*)

O2 の最適化を行い、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。O3 の最適化を使用すると、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。

-no-prec-div (Linux*) または /Qprec-div- (Windows*)

このオプションは、完全な IEEE 準拠の除算よりも多少精度が低い最適化を有効にします。この場合、コンパイラーは浮動小数点数の除算計算を分母の逆数による乗算に変えることがあります。例えば、A/B を A * (1/B) として計算して計算速度を上げます。アプリケーションで完全な IEEE 準拠の除算と同じ精度が必要ない場合は、このオプションを有効にします。

-ansi-alias (Linux*) または /Qansi-alias (Windows*)

このオプションは、プログラムが ISO C 標準の別名規則に準拠していると仮定するようにコンパイラーに指示します。プログラムがこれらの規則に準拠している場合、このオプションを指定することで、コンパイラーはさらに強力な最適化を実行します。プログラムがこれらの規則に準拠していない場合、コンパイラーは誤ったコードを生成する可能性があります。このオプションは、プログラムが ISO C 標準の別名規則に準拠している場合にのみ有効にしてください。

このステップの一般的なヒントは、O2 オプションと O3 オプションの両方でアプリケーションをコンパイルしてみて、パフォーマンスが高いオプションを使用することです。

3. プロセッサー固有の最適化を有効にする

アプリケーションが固有のプロセッサーをターゲットにしている場合は、**-x<code> (Linux*)** または **/Qx<code> (Windows*)** オプションを使用して、プロセッサー固有の最適化を有効にします。このオプショ

ンは、生成する命令セットと最適化を含む、ターゲット・プロセッサー機能をコンパイラーに指示します。また、インテルの機能固有の最適化と追加の最適化を有効にします。

生成されるプロセッサー専用コードは、インテル以外のプロセッサーでは実行できないことがあります。code 値を指定して生成した実行ファイルは、指定した命令セットをサポートしているインテル® プロセッサーでのみ動作します。code 値を指定して生成したバイナリーは、指定した機能をサポートしているインテル® プロセッサーで実行できます。

ほとんどの組込みアプリケーション開発ではターゲット・プロセッサーが固定されているため、最高のパフォーマンスが得られるように、このオプションを使用するべきです。例えば、組込みデバイスが Intel Atom® プロセッサー・ベースで、アプリケーションをこのデバイス上でのみ実行する場合は、–xSSSE3_ATOM オプションを使用すると最高のパフォーマンスを得ることができます。

Silvermont† マイクロアーキテクチャー・ベースの Intel Atom® プロセッサーの場合は、-xATOM_SSE4.2 オプションを使用すると Silvermont† コア向けの最適化が行われます。

(†: 開発コード名)

インテル® Quark™ SoC X1000 向けにインテル® C++ コンパイラーを使用する場合は、次の 2 つのコンパイラー・オプションを利用できます。

-mia32 は、IA-32 アーキテクチャー向けにコードを生成するようにコンパイラーに指示します。
-falign-stack=assume-4-byte は、スタックが 4 バイト境界でアライメントされていることを仮定するように指示します。コンパイラーは、必要に応じて動的にスタックを 16 バイトにアライメントできます。このオプションを使用すると、ルーチンの呼び出しに必要なデータサイズが減ります。

-x オプションでサポートしている <code> 値の詳細は、次のリンクを参照してください。

https://www.isus.jp/products/c-compilers/performance-tools-for-software-developers-intel-compiler-options/

4. IPO 最適化を使用する

プロシージャー間の最適化 (IPO) はマルチステップの自動処理で、コンパイラーがコードを解析してどの最適化が有効であるかを判断できるようにします。IPO オプションを使用すると、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において多くの最適化が行われます。

IPO 最適化を有効にする前に、オリジナルのアーカイブ "ar" およびリンカー "ld" を "xiar" および "xild" に変更してください。詳細は、GNU* コンパイラーからインテル® C++ コンパイラーへの移植方法に関する次の記事を参照してください。

http://software.intel.com/en-us/articles/using-intel-c-compiler-for-embedded-system (英語)

IPO はファイル間の最適化です。通常は、**-ipo** オプションを使用すると IPO 最適化が有効になります。コンパイラーは、生成されるオブジェクト・ファイルに中間データを追加します。リンク段階で、コンパイラーは、オブ

ジェクト・ファイルの中間データに基づいてマルチファイルの最適化を行います。IPO 最適化では、いくつかの最適化を行うことができます。

IPO 最適化を有効にすると、メモリー要件が増加し、コンパイル時間が長くなります。特に、非常に大きなアプリケーションでは、コンパイル時間が大幅に長くなります。

IPO 最適化の詳細は、『インテル® C++ コンパイラー・デベロッパー・ガイドおよびリファレンス』の「主な機能」>「プロシージャー間の最適化 (IPO)」を参照してください。

5. PGO 最適化を使用する

プロファイルに基づく最適化 (PGO) では、命令キャッシュの問題を軽減させるコードの再構成、コードサイズの減少、分岐予測ミスの減少などによりアプリケーション・パフォーマンスの向上を図ることができます。PGO を行うと、アプリケーションで最も頻繁に実行される領域に関する情報がコンパイラーに伝えられます。この領域を知ることで、コンパイラーは、より慎重かつ明確にアプリケーションの最適化を行います。

PGO には、3 つの基本フェーズ (ステップ) があります。

ステップ 1 は、プログラムをインストルメントします。このフェーズでは、コンパイラーはソースコードおよび特別なコードからインストルメント済みプログラムを作成し、リンクします。このステップでは、"-prof-gen" オプションを有効にして、コンパイラーがインストルメント済みバイナリーを生成するようにします。

ステップ 2 は、インストルメント済み実行ファイルを実行します。インストルメント済みコードが実行されるたびに、最終コンパイルで使用される動的情報ファイルが生成されます。

ステップ 3 は、最終コンパイルを実行します。2 度目のコンパイルでは、生成した動的情報がマージされて 1 つのサマリーファイルができます。このプロファイル情報の概要を使用して、コンパイラーはプログラム内で最も頻繁に使用するパスの実行の最適化を図ります。このステップでは、"-prof-use" オプションを有効にして、コンパイラーがステップ 2 で生成したプロファイラーを使用するようにします。

組込み環境の場合は、"-prof-dir=<val>" オプションを使用して、実行中にプロファイラーを生成するパスを指定します。"<val>" はターゲットシステムのフォルダー名です。プログラムを終了した後、このフォルダーにプロファイラーが生成されていることを確認します。

生成されたプロファイラー (.dyn ファイル) をターゲットからホストマシンにコピーし、ホストマシンのフォルダーに追加した後、"-prof-dir" オプションの値を変更して、プロファイラーを格納したホストマシンのフォルダーを指定します。

組込みシステムで PGO 最適化を有効にするために必要な基本的なステップを次に示します。

1. "-prof-gen" オプションと "-prof-dir=<val>" オプションを有効にします ("<val>" はターゲットマシンのフォルダー名です)。

2. コンパイル後、ターゲットマシンでプログラムを実行します。"-prof-dir" オプションで指定したフォルダーから .dyn ファイルをホストマシンにコピーします。

3. "-prof-gen" オプションを "–prof-use" オプションに変更し、"–prof-dir" オプションの値を変更してプロファイラーを格納したホストマシンのフォルダーを指定して、アプリケーションを再コンパイルします。

プロファイラーを生成するには、プログラムを終了できることを確認する必要があります。組込みシステムで、終了ポイントなしでプログラムを実行している場合、プロファイラーが生成できるように追加の作業を行う必要があります。

1. プログラムに終了コードを手動で追加します。
2. プロファイラーをダンプするため、ソースコードに PGO API _PGOPTI_Prof_Dump_All() を追加します。
3. 通常のダンプを行う環境を使用します。例えば、5000 ミリ秒ごとに 1 つのファイルをすべてダンプする場合は、次の環境を使用します。

```
export INTEL_PROF_DUMP_INTERVAL 5000
export INTEL_PROF_DUMP_CUMULATIVE 1
```

PGO 最適化の詳細は、『インテル® C++ コンパイラー・デベロッパー・ガイドおよびリファレンス』の「主な機能」>「プロファイルに基づく最適化 (PGO)」を参照してください。

6. 自動ベクトル化のチューニング

自動ベクトル化はインテル® コンパイラーのコンポーネントです。コンパイラーは、インテル® ストリーミング SIMD 拡張命令 (インテル® SSE)、インテル® SSE2、インテル® SSE3、およびインテル® SSE4 ベクトル化コンパイラーとメディア・アクセラレーター、インテル® ストリーミング SIMD 拡張命令 3 補足命令 (インテル® SSSE3)、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) 命令を自動的に使用します。ベクトル化機能は、並列に実行できるプログラム内の演算を検出し、並列化します。例えば、データ型によって、2、4、8 または最大 16 までの要素を処理するシーケンシャルな SIMD 命令を 1 つの並列操作に変換します。

コンパイラーによるベクトル化は、大きなデータ処理ループを含むほとんどのデータ処理アプリケーションで重要なポイントです。コンパイラーが生成する詳細なレポートは、ベクトル化されたループとベクトル化されなかったループおよびその理由を知る際に役立ちます。コンパイラーの自動ベクトル化機能を理解し、ベクトル化が可能なループを作成する際に役立つ、オンラインのホワイトペーパーが提供されています。詳細は、次のリンクを参照してください。

インテル® C++ コンパイラーのベクトル化ガイド

まとめ

上記のステップに従って、次のインテル® C++ コンパイラーのオプション (Linux*) を指定すると、アプリケーションのパフォーマンスを向上することができます。

-O2/O3 -no-prec-div -x<code> -ipo -prof-gen/-prof-use -prof-dir=<val>

インテル® C++ コンパイラーは、アプリケーションのパフォーマンスをチューニングする、多くの高レベルで固有の機能を提供しています。この記事で説明した手順は、ほとんどのアプリケーションに適用できます。このほ

かにも、コンパイラーには、パフォーマンスの最適化を詳細に行うことができる多くのオプションが用意されています。詳細は、『インテル® コンパイラー・デベロッパー・ガイドおよびリファレンス』を参照してください。

インテル® VTune™ Amplifier を使用すると、アプリケーションのパフォーマンス hotspot やアーキテクチャーのパフォーマンス問題を調査することができます。パフォーマンス解析とパフォーマンス・チューニングには、インテル® VTune™ Amplifier を使用することを推奨します。

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。