インテル® Advisor 2020
ベクトル化アドバイザー 入門ガイド

エクセルソフト株式会社 テクニカル・サポート

1.0.0 - 2020/11/04

目次

はじめに

インテル® Advisor 製品は、並列プログラミングを行う上で重要なベクトル化とスレッド化を支援するためのアドバイザー・ツールです。インテル® Advisor のベクトル化アドバイザーは、アプリケーションに実装されているループ処理、および関数を調査して、主に下記の情報を提供します。

- ループごとにベクトル化の可否と消費した時間
- 使用された SIMD 拡張命令とベクトル化効率 (ベクトル化されている場合)
- ベクトル化されていない理由と、解決に向けた推奨事項の提案 (ベクトル化されていない場合)
- ループの繰り返し回数と呼び出し回数、1 ループ処理するために消費する時間
- コンパイラーの最適化情報 (ループアンロール、ブロッキング、アラインメント、ベクトル幅の拡張)
- データ構造によるメモリー・アクセス・パターン
- データ間の依存関係
- FLOPS 情報
- ルーフライン・グラフ

さらにこれらの情報をソースコードおよびアセンブリー・コードに対応付けることで、ベクトル化に関する情報をコードレベルで可視化します。

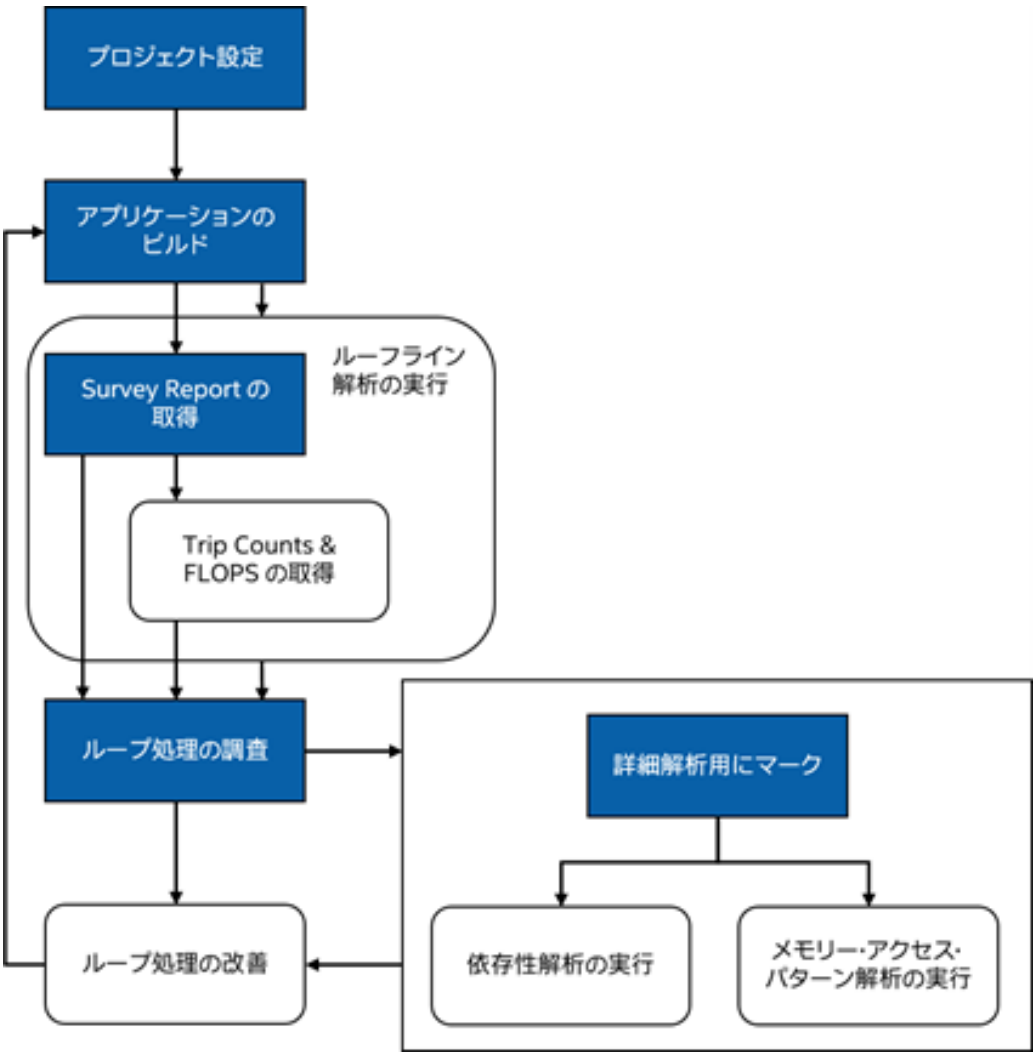
本ガイドはインテル® Advisor のベクトル化アドバイザー向けに提供されている C++ 言語のサンプルプログラムを使用して、ベクトル化アドバイザーから必要な情報を取得するための使用方法と、各画面の見方をチュートリアル形式で説明します。本チュートリアルでは以下のステップを実行します。また、本ガイドは Windows 環境を使用しますが、Linux 環境でも同様の手順にて使用できます。

1. 必要条件の設定
2. インテル® Advisor の起動
3. プロジェクトの作成と設定
4. ベクトル化に関する情報を取得する
5. ループ情報の詳細を取得する
6. メモリー・アクセス・パターンを取得する
7. データ間の依存性を解析する

なお、インテル® Advisor の使用方法の説明を目的としており、ベクトル化の改善手法に関して詳しく説明しません。チュートリアルの前に、ベクトル化の概要と改善例について ベクトル化によるパフォーマンスの向上 をご確認ください。

1. ワークフロー

インテル® Advisor のベクトル化アドバイザーは下記左側のワークフローに従ってアプリケーションを調査します。右図の順番に解析をかけていき、必要な情報を取得してアプリケーションのパフォーマンスの改善につなげます。(白いボックスの操作は必須ではありません)



画像 1. ベクトル化アドバイザーのワークフロー

2. 必要条件の設定

1. インストールフォルダから Vector_Tutorial_Introduction.zip を任意の場所に解凍します。

   デフォルトの設定でインストールされている環境では、以下のフォルダに配置されています。

   ```
   C:\Program Files (x86)\IntelSWTools\Advisor 2020\samples\en\C++\Vector_Tutorial_Introduction.zip
   ```

   ここでは、デスクトップにコピーします。

2. Vector_Tutorial_Introduction フォルダを開き Visual Studio* のソリューションファイル (.sln) をクリックして Visual Studio* を起動します。
3. Visual Studio* の起動後、上部メニューから **ビルド › ソリューションのビルド** を選択して実行ファイルを作成します。
4. **デバッグ › デバッグなしで開始** を選択してプログラムを実行します。

   プログラムが正常に完了するか確認します。サンプルプログラムでは、複数のループ処理が記述されており、それぞれインテルコンパイラーによって自動ベクトル化が有効な設定でビルドされています。これらのループ処理は配列へのアクセス順序や、呼び出し方法が異なるように実装されており、ベクトル化アドバイザーを使用して、効率のよい実装はどれなのか確認します。

3. インテル® Advisor の起動

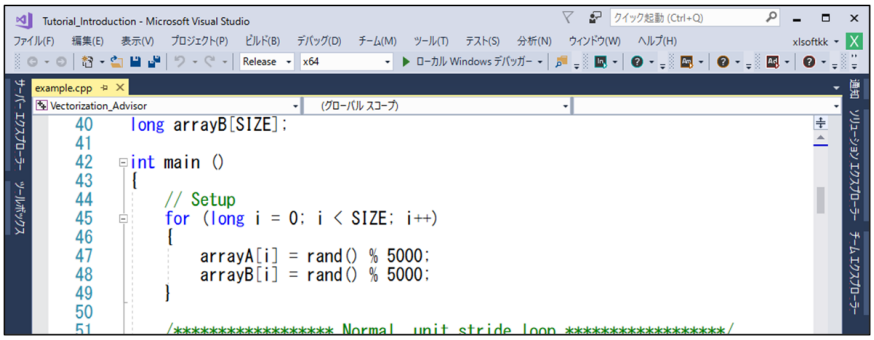
OS と開発環境にあわせて 3 パターンの方法で起動することができます。いずれかの方法から インテル Advisor を起動します。

- **スタートメニューからインテル® Advisor GUI を起動する (Windows* のみ)**

  [Intel Parallel Studio XE 2020] > [Advisor 2020] を開きます。

- **Visual Studio* へ統合されたインテル® Advisor GUI を起動する (Windows* のみ)**

  上部メニューのアイコンを選択してインテル Advisor を開きます。



画像 2. Visual Studio のメニューからインテル® Advisor を開く

- **コマンドプロンプトからインテル® Advisor GUI を起動する (Linux* 向け)**

  1. advixe-vars.sh を実行して GUI を起動するために必要な環境変数を設定します。

     ```
     $source <インストール・ディレクトリー>/advisor/bin/advixe-vars.sh
     ```

  2. インテル® Advisor GUI を起動します。

     ```
     $advixe-gui
     ```

4. プロジェクトの作成と設定

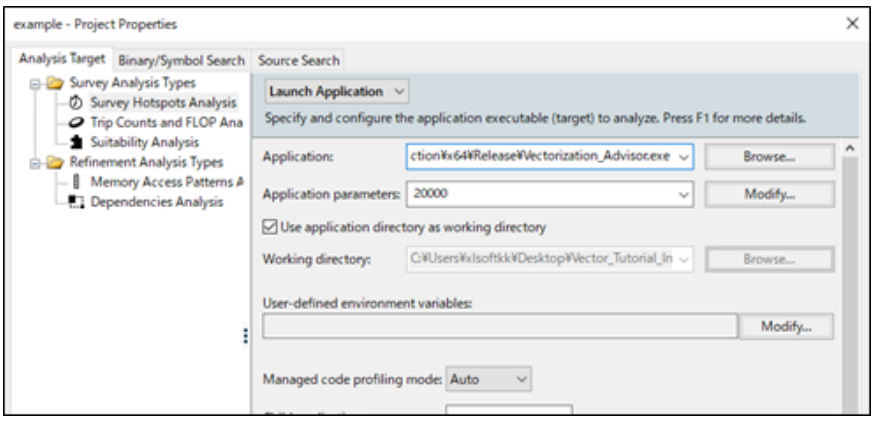
GUI を起動した場合にはプロジェクトの作成と設定を行います。統合環境から起動すると Visual Studio* のプロジェクト設定を継承するため、基本的にインテル® Advisor のプロジェクトを別途作成、設定する必要はありません。変更したい場合は、[プロジェクト] > [Intel Advisor 2019 Project Properties…] を選択します。インテル® Advisor GUI と同じプロジェクト設定の画面を確認することができます。

インテル® Advisor は 1 つのアプリケーションに対して、1 つのプロジェクトを作成して管理します。設定画面では解析対象の実行ファイルと、コンパイル時に生成されるシンボルファイル (オブジェクト・ファイル)、ソースファイルを必ず指定します。

1. インテル® Advisor のプロジェクトを作成するために を選択します。
2. [Project name] にプロジェクト名を入力して [Create Project] を選択します。

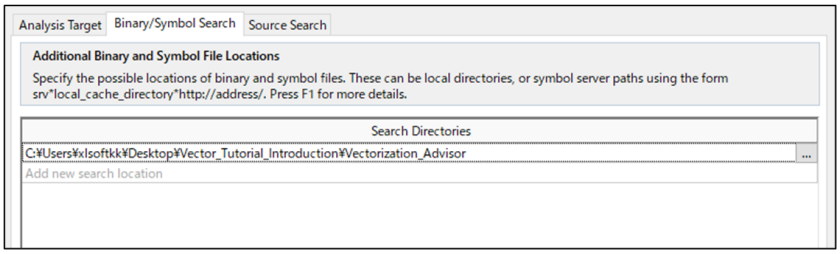
   本ガイドでは vec_intro をプロジェクト名に指定します。プロジェクト名には任意の名前を入力することが可能です。

3. インテル® Advisor のプロジェクト・プロパティー画面から必要な設定を追加します。



画像 3. Configure Analysis タブ

- [Application] に実行ファイルを指定します。サンプルコードの vec.exe または vec.out を指定します。
- [Binary/Symbol Search] タブに移動してシンボルファイルの配置先を指定します。



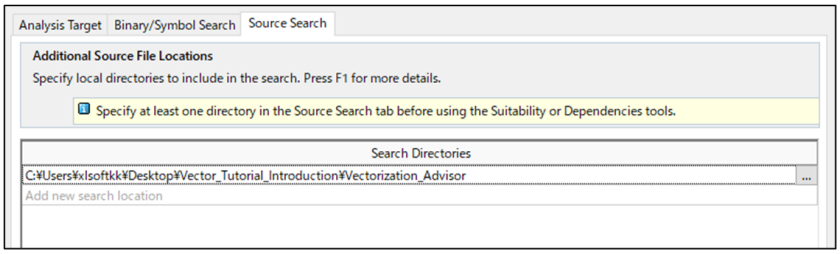
画像 4. シンボルの検索

サンプルコードのシンボルファイルが配置されているフォルダーを指定します。Linux の場合、実行ファイルの配置先ディレクトリを指定します。

シンボルファイルの配置先

```
Vector_Tutorial_Introduction\Vectorization_Advisor\x64\Release
```

- [Source Search] タブに移動してソースファイルの配置先を指定します。



画像 5. ソースファイルの検索

サンプルコードの example.cpp が配置されているフォルダーを指定します。

ソースファイルの配置先

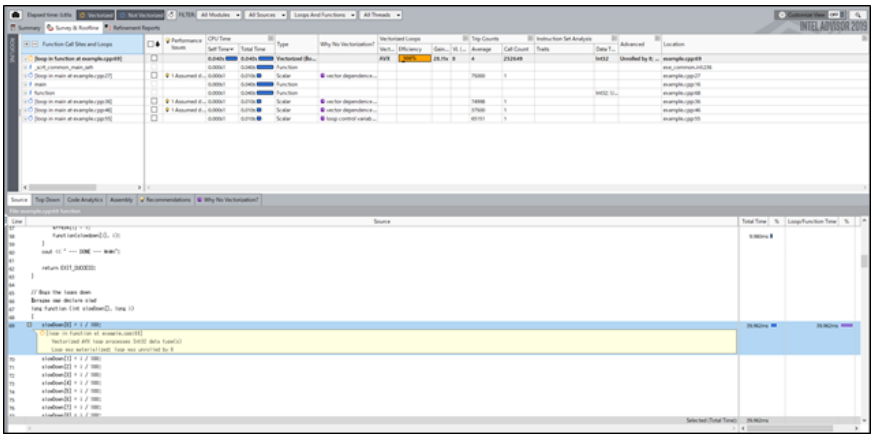
```
Vector_Tutorial_Introduction\Vectorization_Advisor
```

5. ベクトル化に関する情報を取得する

1. Survey Target を選択して、解析対象のプログラムの解析を開始します。

   解析を開始すると、解析対象のプログラムが実行され、インテル® Advisorがサンプリングを行います。プログラムの終了後、インテル® Advisor は収集した情報を解析して、Survey Report を表示します。

2. [Survey & Roofline] タブの情報を確認します。



画像 6. Survey Report の表示

[Survey Target] の実行により、主に下記の情報を確認することが可能です。

- ループ処理のベクトル化の可否
- ループ処理で消費した時間
- 使用された SIMD 拡張命令とベクトル化効率
- ベクトル化されていない理由と、解決に向けた推奨事項の提案
- コンパイラーの最適化情報

6. ループ処理の詳細を取得する

1. [Find Trip Counts and FLOP] 解析の [Trip Counts] と [FLOP] をチェックします。
2. [Find Trip Counts and FLOP] を選択して解析を開始します。

   Survey Target 同様にプログラムが起動します。インテル® Advisor の解析が完了するまで待機します。

3. [Survey & Roofline] タブの情報を確認します。

[Find Trip Counts and FLOP] の解析により主に下記の情報を確認できます。

- ループ処理の繰り返し回数、呼び出し回数

7. ループ処理を詳細解析の対象にする

「メモリー・アクセス・パターンを解析する」、「データ間の依存性を解析する」は解析にかかるオーバーヘッドが大きく、複雑なアプリケーションではすべてのループ処理を対象にすべきではありません。解析機能に応じて、詳細解析 (Deeper Analysis) のフラグを設定して、適切なループ処理から必要な情報を取得することを推奨します。 解析によって得られる情報は異なるため、詳細解析対象となりえる典型的なループ処理として下記のパターンが考えられます。

- **共通される事項**

  Self Time が大きく、呼び出し回数が多い

  注意

  > 本チュートリアルは解析結果の紹介を目的としており、プログラムの実行時間が短いため、Self Time と呼び出し回数は考慮していません。

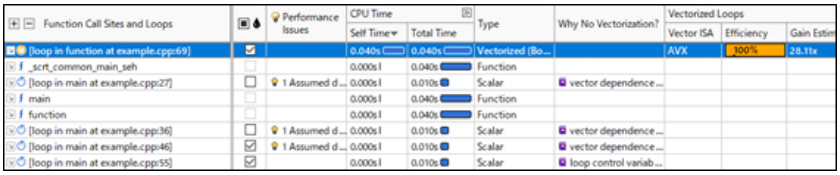
- **メモリー・アクセス・パターン解析**

  ベクトル化されているが効率性 (Efficiency) が低いく、ベクトル化されていない理由に、効率に関するメッセージが表示されている

- **依存性解析**

  ベクトル化されていないループ処理であり、ベクトル化されていない理由として依存性が含まれる場合があるとのメッセージが表示されている。

1. 詳細解析の対象とするためには、[Survey & Roofline] タブに表示される 項目をチェックします。



画像 7. 詳細解析の対象を選択

8. メモリー・アクセス・パターンを解析する

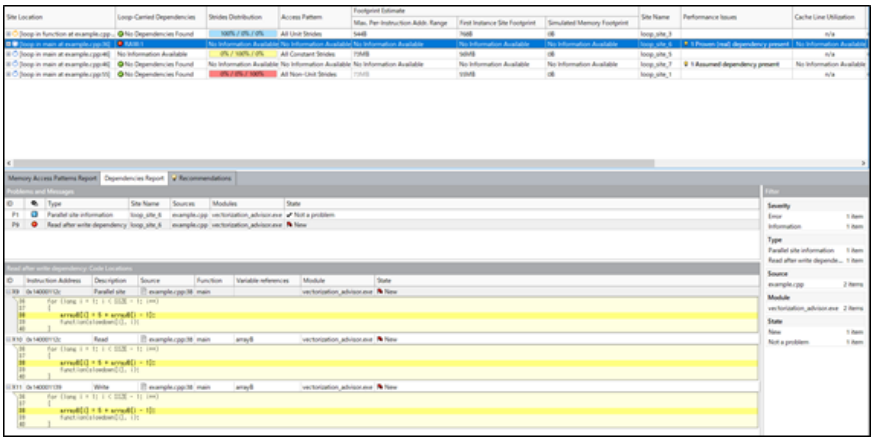
1. example.cpp:46、example.cpp:55、および example.cpp:69 の 項目をチェックします。example.cpp:69 は効率良くベクトル化されているループ処理の例として確認します。

メモリー・アクセス・パターンの解析は詳細解析の対象になるため、[Survey & Roofline] タブの画面から解析対象のループ処理をマークします。詳細解析の対象は「ループ処理を詳細解析の対象にする」を確認してください。

1. [Check Memory Access Patterns]を選択して解析を開始します。

   インテル® Advisor の解析が完了して画面が更新されるまで待機します。

2. [Refinement Reports] を確認します。



画像 8. Refinement Reports

[Check Memory Access Patterns] 解析では主に下記の情報を確認することが可能です。

- 変数ごとのメモリー・アクセス・パターン

9. データ間の依存性を解析する

1. メモリー・アクセス・パターンを解析する でチェックしたループ処理に追加して example.cpp:36 へチェックします。

   データ間の依存性の解析は詳細解析の対象になるため、[Survey & Roofline] タブの画面から解析対象のループ処理をマークします。詳細解析の対象とする処理は「ループ処理を詳細解析の対象にする」を確認してください。

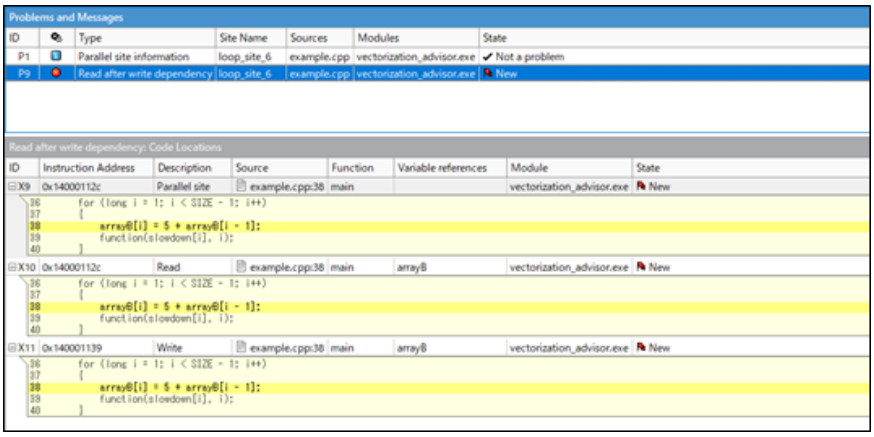
2. [Check Dependencies] の を選択して解析を開始します。

   インテル® Advisor のファイナライズが完了して画面が更新されるまで待機します。

   注意

   > ベクトル化されたループを選択しているため、警告が表示されることがあります。表示された場合は [Continue] を選択してそのまま進めます。

3. [Refinement Reports] を確認します。



画像 9. Refinement Reports

[Check Dependencies] 解析では主に下記の情報を確認することができます。

- 依存関係を持つ変数のリストとソースコード

データ間の依存関係に関する情報は、「ループをベクトル化するための条件」や、「インテル® C++ コンパイラーのベクトル化ガイド」の「4.2データ依存」項目を確認してください。

ベクトル化を適用した場合、ベクトル化の対象となる演算処理の実行順序は固定されません。一般的に、ある値を計算する時に使用するデータは計算前に決定されている必要があるため、前のループで演算した値を利用する処理は基本的にベクトル化できません。依存関係を持つ処理をベクトル化するためにはアルゴリズム・レベルでの変更が必要となるケースがあります。

10. お問い合わせ

本ドキュメントに関する不明点は以下よりご連絡ください。

お問い合わせ窓口

URL: https://www.xlsoft.com/jp/services/xlsoft_form.html

インテル®ソフトウェア製品のパフォーマンス/ 最適化に関する詳細は、Optimization Notice (最適化に関する注意事項) を参照してください。