インテル® Inspector チュートリアル: C++ サンプル・アプリケーションのスレッドエラーを解析する (Linux* 版)

この記事は、インテル® デベロッパー・ゾーンに公開されている「Tutorial: Analyzing Threading Errors With Intel® Inspector and a C++ Sample Application for Linux*」の日本語参考訳です。

目次

インテル® Inspector は、Windows* および Linux* でシリアル・アプリケーションおよびマルチスレッド・アプリケーションを開発するユーザー向けの、メモリーとスレッドのエラーを動的にチェックするツールです。

このチュートリアルでは、C++ サンプル・アプリケーションを使用して、Linux* プラットフォームでインテル® Inspector を利用してスレッドエラーを解析する方法を示します。

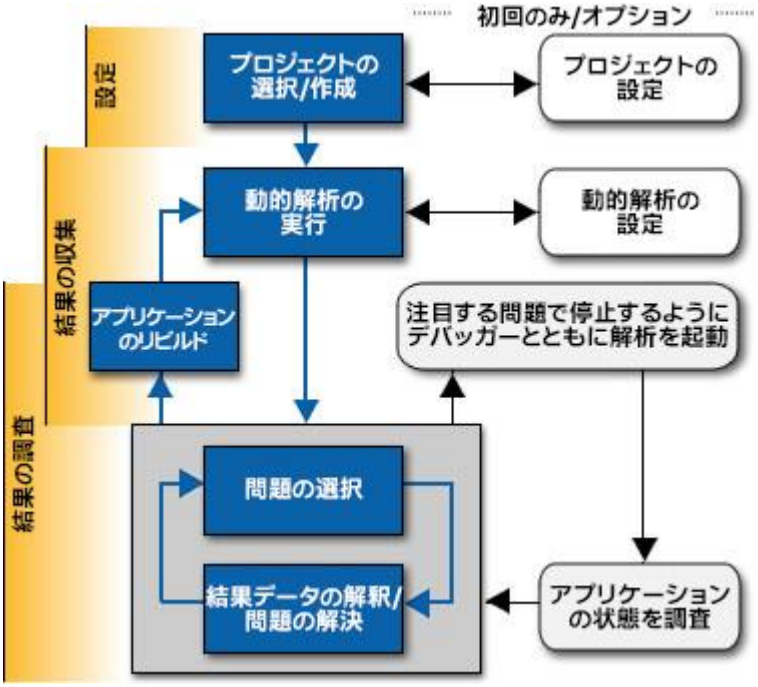
インテル® Inspector は、スタンドアロン製品として、または次の製品のコンポーネントとして利用できます。

- インテル® System Studio Professional Edition
- インテル® System Studio Ultimate Edition
- インテル® Parallel Studio XE Professional Edition
- インテル® Parallel Studio XE Cluster Edition
- ベータ版インテル® oneAPI HPC ツールキット
- ベータ版インテル® oneAPI IoT ツールキット

チュートリアルの概要	このチュートリアルでは、最終的に独自のアプリケーションに適用可能なエンドツーエンドのワークフローを示します。 1. 調査に最適なアプリケーションをビルドします。 2. アプリケーションを調査してスレッドエラーを見つけます。 3. アプリケーション・コードを編集してスレッドエラーを修正します。 4. アプリケーションをリビルドして再度調査します。 このチュートリアルは、バージョン 2017 以降向けに作成されています。バージョン 2018 以降では、スクリーンショットの背景色が異なります。
所要時間	10 - 15 分
目的	このチュートリアルでは、以下のトピックについて説明します。 • インテル® Inspector を使用してスレッドエラーを見つけて修正する手順をリストします。 • 主要なインテル® Inspector の用語を定義します。 • 最も正確で完全な解析結果を生成するコンパイラー/リンカーオプションを特定します。 • スレッドエラー解析を実行します。 • 解析の範囲と実行時間に影響を与えます。 • インテル® Inspector の結果ウィンドウを使用します。 • エラーを修正するため優先順位付けされた To Do リストを表示します。 • 特定のエラーを修正するためヘルプを利用します。 • エラーを修正するためソースコードにアクセスします。
関連情報	このチュートリアルの概念と手順は、プログラミング言語に関係なく適用されますが、別のプログラミング言語のサンプル・アプリケーションを使用したチュートリアルも用意されています。 • インテル® ソフトウェア・ドキュメント・ライブラリー (英語) • インテル® ソフトウェア製品のサンプルおよびチュートリアル (英語) これらのサイトでは、ほかのインテル® 製品のチュートリアルも提供されています。 さらに、以下のサイトから追加のリソースを利用できます。 • インテル® Inspector 入門 (英語) • インテル® Inspector

スレッドエラー解析の使用例

強力で柔軟性に優れたインテル® Inspector を活用する方法はたくさんあります。並列プログラムのスレッドエラーを見つけて修正する次のワークフローは、生産性を迅速かつ最大限に高める 1 つの方法です。



このチュートリアルは、次の方法でこのワークフローを実装します。

ステップ 1: サンプルの展開と解析用の設定	スレッドエラーを調査するためアプリケーションをビルドして、インテル® Inspector 環境を設定して、新しいインテル® Inspector プロジェクトを作成します。
ステップ 2: 結果の収集	• スレッドエラー解析を設定します。 • アプリケーションのスレッドエラー解析を実行します。
ステップ 3: 結果の調査	• 解析結果中の問題を選択します。 • 結果データを解釈します。 • 問題を解決します。
ステップ 4: 作業の確認	アプリケーションをリビルドして、スレッドエラー解析を再度実行します。

ナビゲーションのクイックスタート

インテル® Inspector 環境を設定する

次のいずれかの方法でインテル® Inspector 環境を設定します。

- 次のいずれかの `source` コマンドを実行します。
  - csh/tcsh ユーザーの場合: `source <inspector-install-dir>/inspxe-vars.csh`
  - bash ユーザーの場合: `source <inspector-install-dir>/inspxe-vars.sh`

  注
  インテル® oneAPI HPC ツールキットまたはインテル® oneAPI IoT ツールキットに含まれる
  インテル® Inspector では、このスクリプトの名前は `inspxe-vars` ではなく `env\vars` です。

  デフォルトのインストール先 `<inspector-install-dir>` は、次のディレクトリー以下にあります。

  - root ユーザーの場合: `/opt/intel/`
  - 非 root ユーザーの場合: `$HOME/intel/`
- `<inspector-install-dir>/bin32` または `<inspector-install-dir>/bin64` をパスに追加します。

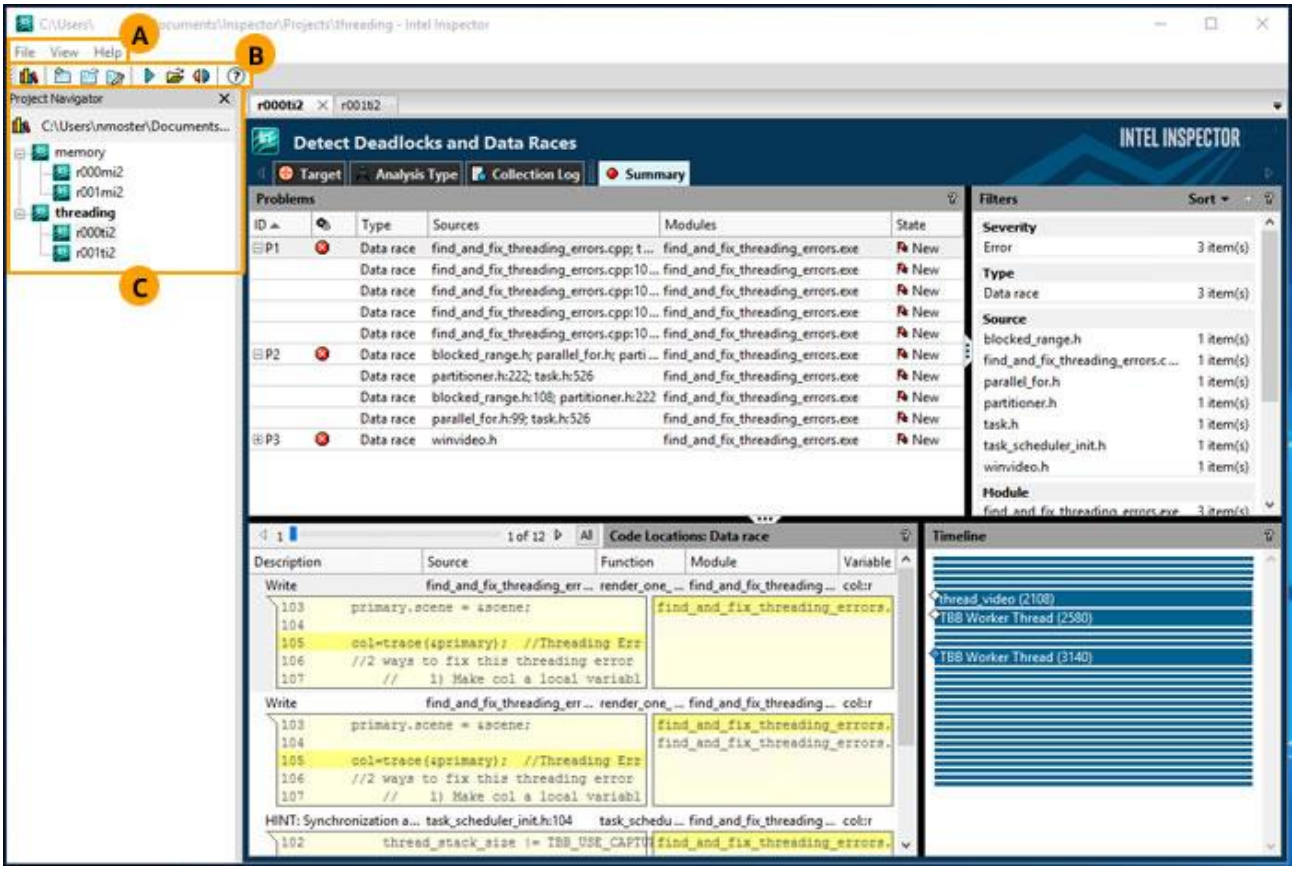
場合によっては、`<studio-install-dir>/psxevars.csh` または `<studio-install-dir>/psxevars.sh` コマンドを実行することもできます。デフォルトのインストール先 `<studio-install-dir>` は、次のディレクトリー以下にあります。

- root ユーザーの場合: `/opt/intel/`
- 非 root ユーザーの場合: `$HOME/intel/`

インテル® Inspector スタンドアロン GUI を開く

`inspxe-gui` コマンドを実行します。

インテル® Inspector スタンドアロン GUI を使用する



メニュー、ツールバー、および **[Project Navigator]** は、多くの同じ機能を実行する異なる方法を提供します。

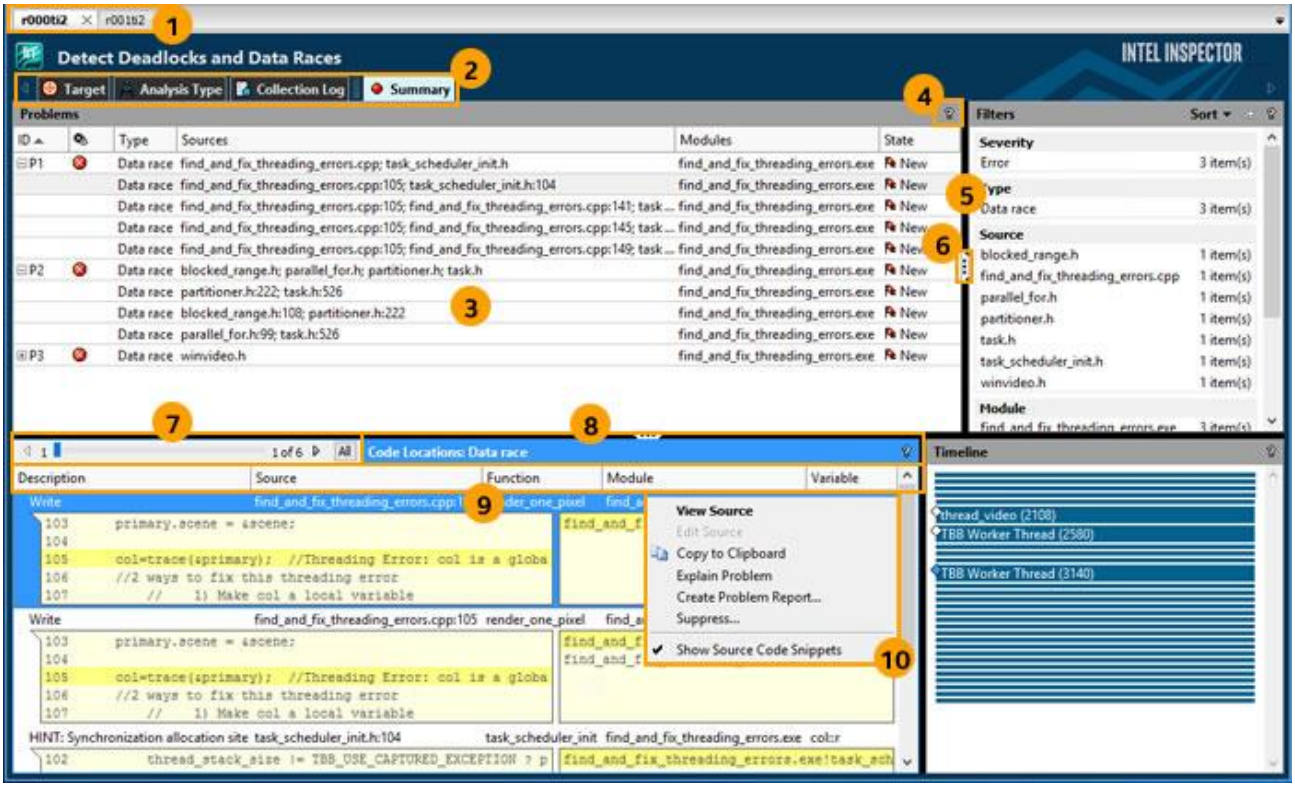
A メニューを使用して、プロジェクトや動的解析結果の作成、結果のアーカイブファイルやインテルのほかのエラー検出製品からの結果のインポート、プロジェクトや結果を開く、プロジェクトの比較、プロジェクトの設定、さまざまなオプションの設定、入門ページやヘルプへのアクセスを行います。

B ツールバーを使用して、入門ページを開く、プロジェクトの作成、設定、または開く、動的解析結果の作成、結果の比較を行います。

C **[Project Navigator]** を使用して、次の操作を行います。

- **ツリー**は、プロジェクトのディレクトリー構造に基づいてプロジェクトと結果を階層表示します。
- **コンテキスト・メニュー** (右クリックで表示) は、メニューやツールバーから利用可能な機能の実行、選択したプロジェクトや結果の削除または名前変更、開いている結果をすべて閉じる、さまざまなディレクトリー・パスのシステムのクリップボードへのコピーを行います。

インテル® Inspector の結果タブを使用する



1	結果タブの名前で結果を区別します。
2	ナビゲーション・ツールバーのボタンをクリックしてウィンドウを切り替えます。
3	ウィンドウペインで結果データを表示および管理します。
4	ボタンをクリックしてウィンドウペインの使用法に関するヘルプページを表示します。
5	ウィンドウペインの境界をドラッグしてウィンドウペインのサイズを変更します。
6	、、、および をクリックしてウィンドウペインの表示/非表示を切り替えます。
7	ウィンドウペインのデータ・コントロールをクリックして、ペイン内 (および可能な場合は隣接するペイン) の結果データを調整します。
8	タイトルバーでウィンドウペインを識別します。
9	データの列見出し - データ列をドラッグして配置を変更します。左または右の境界をドラッグしてデータ列のサイズを変更します。列見出しをクリックして結果を昇順または降順に並べ替えます。
10	ウィンドウペインでデータを右クリックして、主要機能へのアクセスを提供するコンテキスト・メニューを表示します。

スタンドアロン GUI: アプリケーションのビルドと新規プロジェクトの作成

インテル® Inspector でスレッドエラーを調査可能なアプリケーションを作成します。

- ソフトウェア・ツールを取得する
- 最適なコンパイラー/リンカー設定を理解する
- アプリケーションをビルドする
- アプリケーションがインテル® Inspector の外部で実行されることを確認する
- インテル® Inspector 環境を設定する
- インテル® Inspector スタンドアロン GUI を開く
- 新しいプロジェクトを作成する

ソフトウェア・ツールを取得する

`tachyon_insp_xe` サンプル・アプリケーションを使用してチュートリアルの手順を実施するには、次のツールが必要です。

- インテル® Inspector のインストール・パッケージとライセンス
- `.tgz` ファイルの展開ユーティリティー
- サポートされるコンパイラー (詳細はリリースノートを参照)
- 編集者

インテル® Inspector を取得する

インテル® Inspector をまだインストールしていない場合は、https://www.isus.jp/intel-inspector-xe/ から評価版をダウンロードします。

インテル® Inspector サンプル・アプリケーションをインストールして設定する

1. `tachyon_inxp_xe.tgz` ファイルを `<install-dir>/samples/<locale>/C++/` ディレクトリーから書き込み可能なディレクトリーへコピーするか、システムで共有します。デフォルトの `<install-dir>` は `/opt/intel` です。
2. `.tgz` ファイルからサンプルを展開して、`tachyon_insp_xe` ディレクトリーを作成します。
3. EDITOR または VISUAL 環境変数が使用するテキストエディターに設定されていることを確認してください。

注

- サンプルは非決定的です。このチュートリアルで示すスクリーンショットは、実際の画面とは異なる場合があります。
- サンプルは、インテル® Inspector の機能を説明することを目的としており、コード作成のベスト・プラクティスを示すものではありません。

最適なコンパイラー/リンカー設定を理解する

インテル® Inspector は、デバッグモードとリリースモードの C++ および Fortran バイナリーでメモリーとスレッドのエラーを解析できますが、次の設定を使用してデバッグモードでコンパイル/リンクされたアプリケーションにおいて最も正確で完全な解析結果を生成します。

コンパイラー/リンカーのプロパティー	適切な C/C++ 設定	不適切な設定による影響
デバッグ情報	有効 (`-g`)	ファイル/行情報の欠落
最適化	無効 (`-O0`)	正しくないファイル/行情報
ダイナミック・ランタイム・ライブラリー	選択 (インテル® コンパイラーの場合は `-shared-intel`、GNU* コンパイラーの場合はデフォルトまたは `-Bdynamic`)	誤検出またはコード位置の欠落
基本ランタイム・エラー・チェック	無効 (`-fmudflap` は使用しない)	誤検出
コンパイラー/リンカーのプロパティー	**適切な Fortran 設定**	**不適切な設定による影響**
デバッグ情報	有効 (`-debug` または `-g`)	ファイル/行情報の欠落
最適化	無効 (`-O0`)	正しくないファイル/行情報
ダイナミック・ランタイム・ライブラリー	選択 (`-shared-intel`)	誤検出またはコード位置の欠落
基本ランタイム・エラー・チェック	なし (`-check:none`)	誤検出

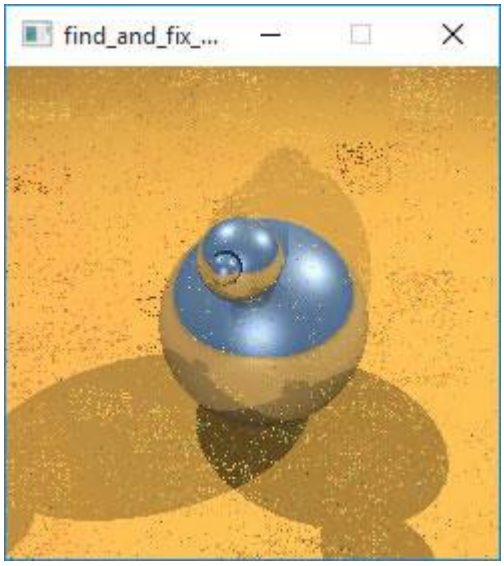
アプリケーションをビルドする

1. ターミナルセッションで、ディレクトリーを `tachyon_insp_xe` に変更します。
2. `make` を実行します。

アプリケーションがインテル® Inspector の外部で実行されることを確認する

1. 同じターミナルセッションで、次のコマンドを使用してサンプル・アプリケーションを実行します。
   `./tachyon.find_and_fix_threading_errors`

2. アプリケーションの非決定的な出力は次のようになります。



イメージ内に色落ちしたドットがあります。原因はスレッドエラーです。

インテル® Inspector 環境を設定する

次のいずれかの方法でインテル® Inspector 環境を設定します。

- 次のいずれかの `source` コマンドを実行します。
  - csh/tcsh ユーザーの場合: `source <inspector-install-dir>/inspxe-vars.csh`
  - bash ユーザーの場合: `source <inspector-install-dir>/inspxe-vars.sh`

  注
  インテル® oneAPI HPC ツールキットまたはインテル® oneAPI IoT ツールキットに含まれる
  インテル® Inspector では、このスクリプトの名前は `inspxe-vars` ではなく `env\vars` です。

  デフォルトのインストール先 `<inspector-install-dir>` は、次のディレクトリー以下にあります。
  - root ユーザーの場合: `/opt/intel/`
  - 非 root ユーザーの場合: `$HOME/intel/`
- `<inspector-install-dir>/bin32` または `<inspector-install-dir>/bin64` をパスに追加します。

場合によっては、`<studio-install-dir>/psxevars.csh` または `<studio-install-dir>/psxevars.sh` コマンドを実行することもできます。デフォルトのインストール先 `<inspector-install-dir>` は、次のディレクトリー以下にあります。

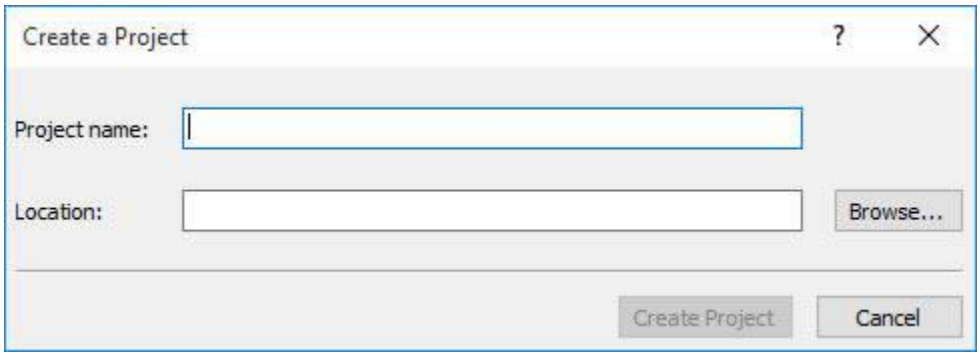
- root ユーザーの場合: `/opt/intel/`
- 非 root ユーザーの場合: `$HOME/intel/`

インテル® Inspector スタンドアロン GUI を開く

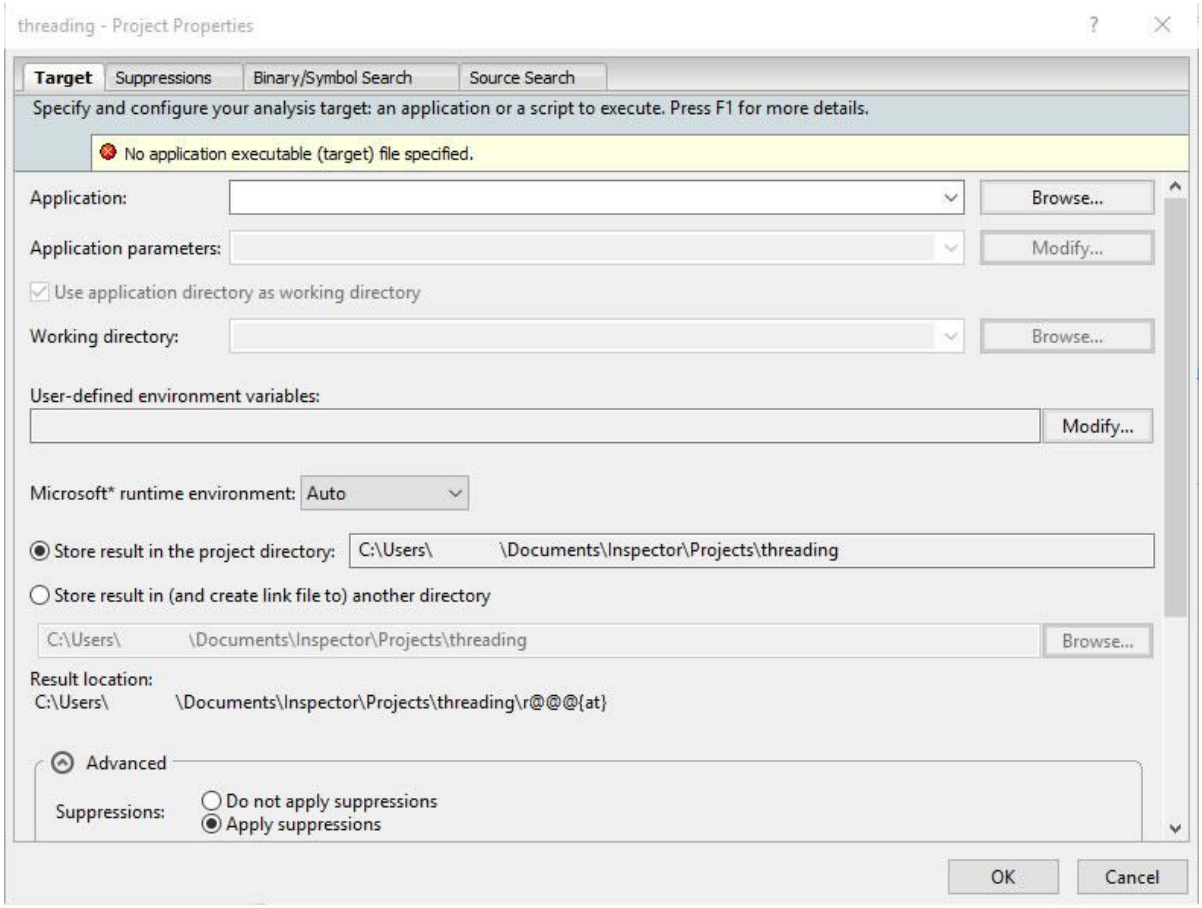
同じターミナルセッションで、`inspxe-gui &` コマンドを使用してインテル® Inspector スタンドアロン GUI をバックグラウンドで実行します。

新しいプロジェクトを作成する

1. **[File]** > **[New]** > **[Project...]** を選択して、次のようなダイアログボックスを表示します。



2. **[Project name]** フィールドに `threading` と入力します。**[Create project]** ボタンをクリックして、`~/intel/inspxe/projects/threading/` ディレクトリー (デフォルトの場所) に `config.inspxeproj` ファイルを生成します。次のようなダイアログボックスが表示されます。



3. **[Application]** フィールドの横にある **[Browse...]** ボタンをクリックして、`tachyon_insp_xe/tachyon.find_and_fix_threading_errors` アプリケーションを選択します。インテル® Inspector によって **[Working directory]** フィールドに値が自動入力されます。**[OK]** ボダンをクリックして [Welcome] ページに戻ります。開いたプロジェクトの名前がタイトルバーと **[Project Navigator]** ペインに表示されます(必要に応じて、**[View]** > **[Project Navigator]** を選択して **[Project Navigator]** を表示します)。

解析の設定

インテル® Inspector には、解析の範囲とコストの制御を支援するため、事前定義済みのさまざまなスレッド解析タイプが用意されています。範囲が最も狭い解析タイプは、システムの負荷、および解析の実行に必要な時間とリソースが最小になりますが、最も少ないエラーセットを検出し、最小限の情報のみ提供します。範囲が最も広い解析タイプは、システムの負荷、および解析の実行に必要な時間とリソースが最大になりますが、最も多くのエラーセットを検出し、最も詳細な情報を提供します。

スレッドエラー解析を設定します。

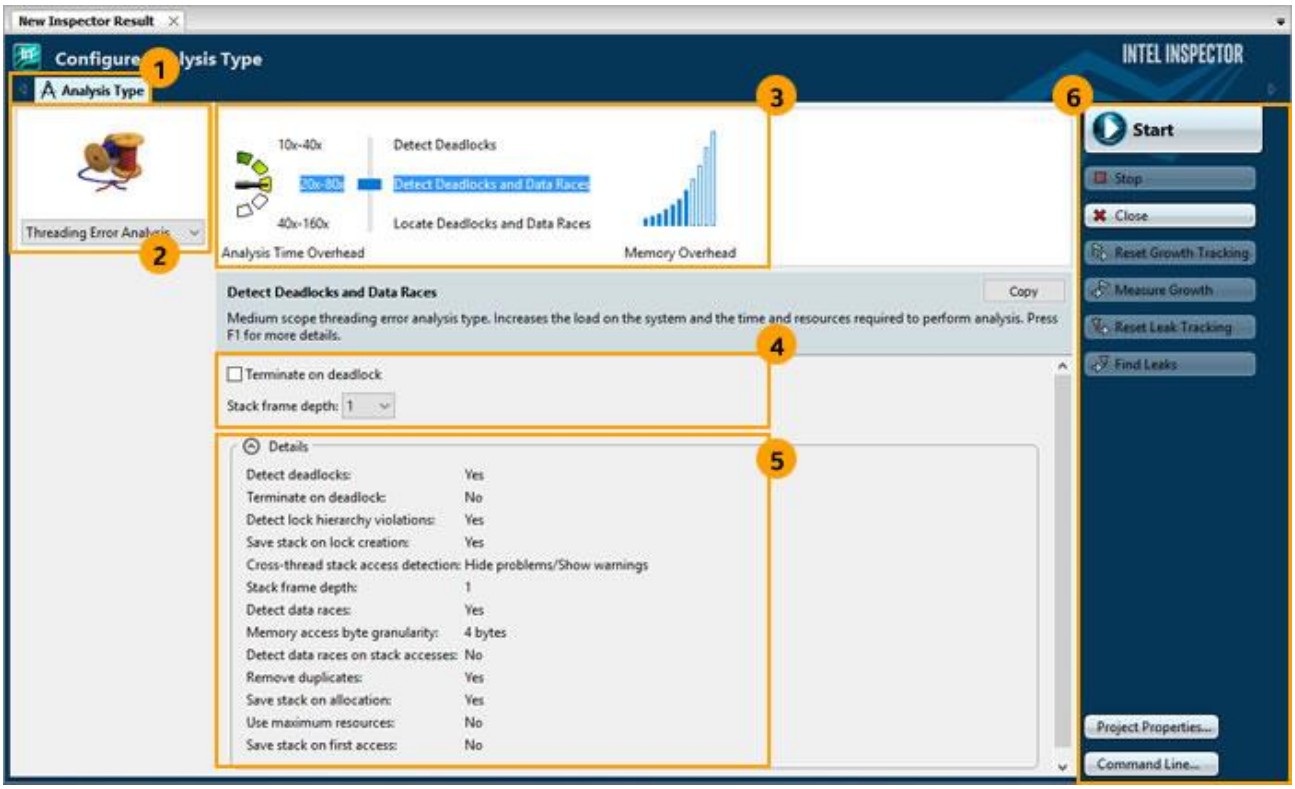
- [Analysis Type] ウィンドウを理解する
- スレッドエラー解析タイプを選択する

[Analysis Type] ウィンドウを理解する

次のような **[Analysis Type]** ウィンドウを表示します。

1. **[File]** > **[New]** > **[Analysis...]** を選択します。
2. 解析タイプ・ドロップダウン・リストから **[Threading Error Analysis]** を選択します。
3. スライダーを使用して **[Detect Deadlocks and Data Races]** 解析タイプを選択します。

注
ウィンドウに 3 つのインタラクティブなデバッガー・ラジオボタンが表示される場合があります。



1	**ナビゲーション**・ツールバーを使用してインテル® Inspector のウィンドウを切り替えます。ツールバーのボタンは、表示されるウィンドウにより異なります。
2	解析タイプ・ドロップダウン・リストから解析タイプカテゴリーを選択します。**[Memory Error Analysis]**、**[Threading Error Analysis]**、または **[Custom Analysis Types]** を選択できます。 **注** このチュートリアルでは、スレッドエラー解析タイプについて説明します。この解析タイプでは、データ競合、デッドロック、ロック階層違反、スレッド間スタック・アクセス・エラーを検出できます。メモリーエラー解析は、リソースリーク、不正な memcpy 呼び出し、無効な割り当て解除、無効なメモリーアクセス、無効な部分メモリーアクセス、メモリー拡張、メモリーリーク、割り当て解除されていないメモリー、割り当て/割り当て解除の不一致、割り当てられていないメモリー、初期化されていないメモリーアクセス、初期化されていない部分メモリーアクセスなどのエラーを検出します。
3	スライダーを使用して、事前定義済みの解析タイプと対応するゲージを選択し、コストを評価します。スライダーの最上部の解析タイプは範囲が最も狭く、最下部の解析タイプは範囲が最も広いです。 **[Analysis Time Overhead]** ゲージは、選択した解析タイプで結果の収集にかかる時間を素早く予測するのに役立ちます。収集時間は、通常のアプリケーション実行時間の倍数で表されます。例えば、10x - 40x は、通常のアプリケーション実行時間の 10 ~ 40 倍の時間がかかることを示します。つまり、通常のアプリケーション実行時間が 5 秒の場合、予測される収集時間は 50 ~ 200 秒になります。 **[Memory Overhead]** ゲージは、選択した解析タイプでインテル® Inspector がエラー検出に使用するメモリー量を素早く予測するのに役立ちます。メモリー使用量に応じて、バーが青く塗りつぶされます。 **注** 解析中、アプリケーションの実行に使用されたメモリー量はゲージに含まれません。
4	チェックボックス、ラジオボタン、およびドロップダウン・リストを使用して、事前定義済みの解析タイプの一部の設定を細かく調整します (すべての設定を調整できるわけではありません)。 **注** 解析タイプの設定をさらに細かく調整する必要がある場合は、別の解析タイプを選択するか、カスタム解析タイプを作成します。
5	**[Details]** 領域は、この解析タイプのすべての設定を表示します。
6	**コマンド**ツールバーは、解析の実行を制御したり、その他の機能を実行します。例えば、**[Project Properties...]** ボタンは、デフォルトの結果ディレクトリーの場所を変更したり、解析をスピードアップする可能性があるパラメーターを設定したり、プロジェクトの設定に関するその他の機能を実行できる **[Project Properties]** ダイアログボックスを表示します。**[Command Line...]** ボタンは、インテル® Inspector コマンドライン・インターフェイス (`inspxe-cl` コマンド) で同じ解析を実行するコマンドを確認して、必要に応じてクリップボードにコピーできる **[Corresponding inspxe-cl Command Options]** ダイアログボックスを表示します。

スレッドエラー解析タイプを選択する

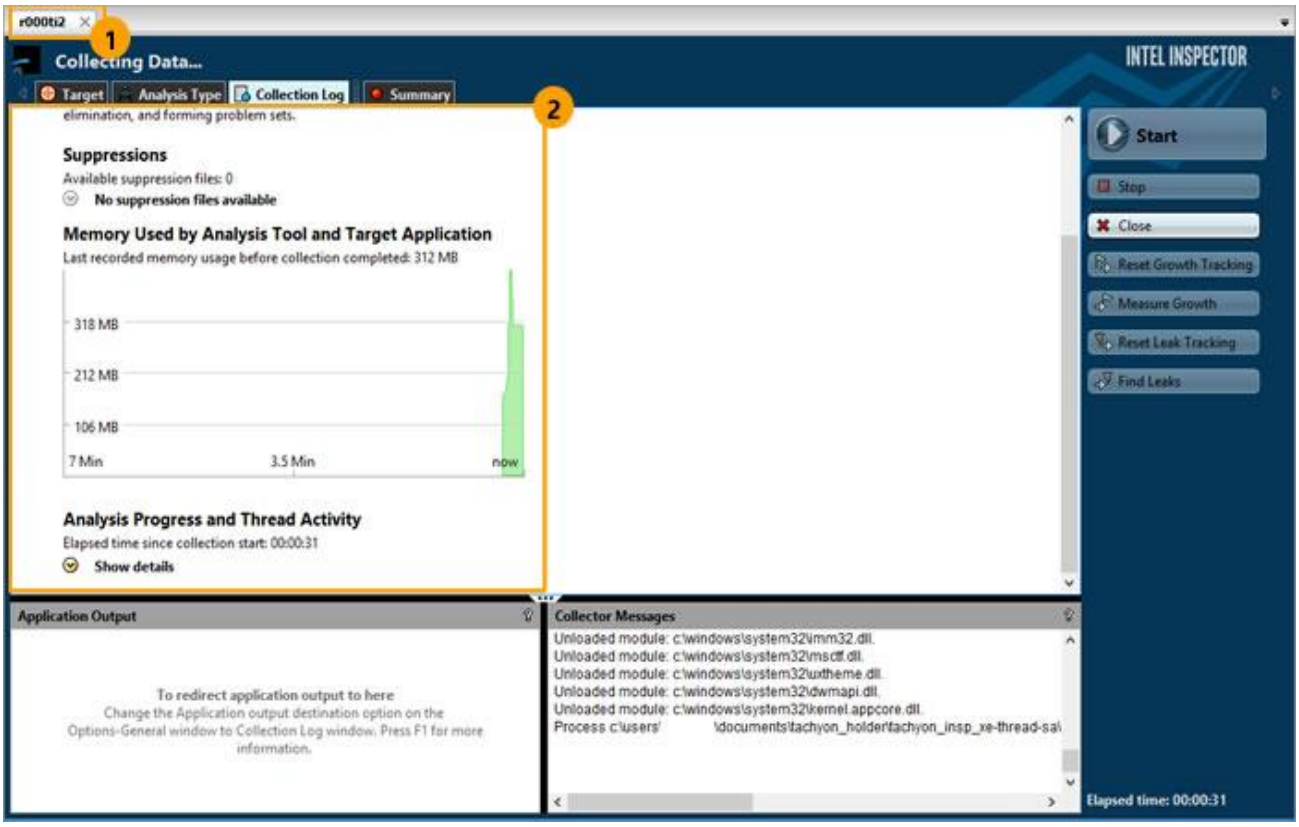
次のステップに進む前に、設定内容が上記のスクリーンショットと一致していることを確認してください。インタラクティブなデバッガー・ラジオボタンが表示されている場合は、**[Analyze without Debugger]** ラジオボタンがオンになっていることを確認してください。

解析の実行

修正が必要なスレッドエラーを検出するには、**[Analysis Type]** ウィンドウの **[Start]** ボタンをクリックします。インテル® Inspector は、次の処理を実行します。

- `tachyon.find_and_fix_threading_errors` アプリケーションを実行します。
- 対応が必要なスレッドエラーを識別します。
- `intel/inspxe/projects/threading/` 以下のディレクトリーに結果を収集します。
- 結果をファイナライズします。

解析中、インテル® Inspector は次のような **[Collection Log]** ウィンドウを表示します。



1 結果の名前がタブに表示されます。この例では、結果の名前は **r000ti2** です。結果の名前は、次の規則に沿って付けられます。

- `r` = 定数
- `000` = 次に利用可能な番号
- `ti` = スレッドエラー解析タイプ
- `2` = 中程度の範囲の事前定義済み解析タイプ

注
インテル® Inspector は、**[Project Navigator]** で結果へのポインターも提供します。

2	**[Collection Log]** ペインは、解析の進行状況と成果を示します。 **[Summary]** ボタンをクリックして、解析 (収集とファイナライズ) が完了する前に結果の管理を開始できますが、このチュートリアルではその方法について取り上げません。

注
このチュートリアルでは、インテル® Inspector スタンドアロン GUI から解析を実行する方法を説明します。インテル® Inspector コマンドライン・インターフェイス (`inspxe-cl` コマンド) を使用して解析を実行することもできます。

解析が完了すると、**[Summary]** ウィンドウが自動的に表示されます。

問題の選択

検出されたスレッドエラーの調査を開始します。

- 主要な用語を理解する
- [Summary] ウィンドウ内の各種ペインを理解する
- 問題を選択する

主要な用語を理解する

code location (コード位置):「書き込み」コード位置など、インテル® Inspector が観測したソースコードの場所。以前は observation (観測位置) と呼ばれていました。

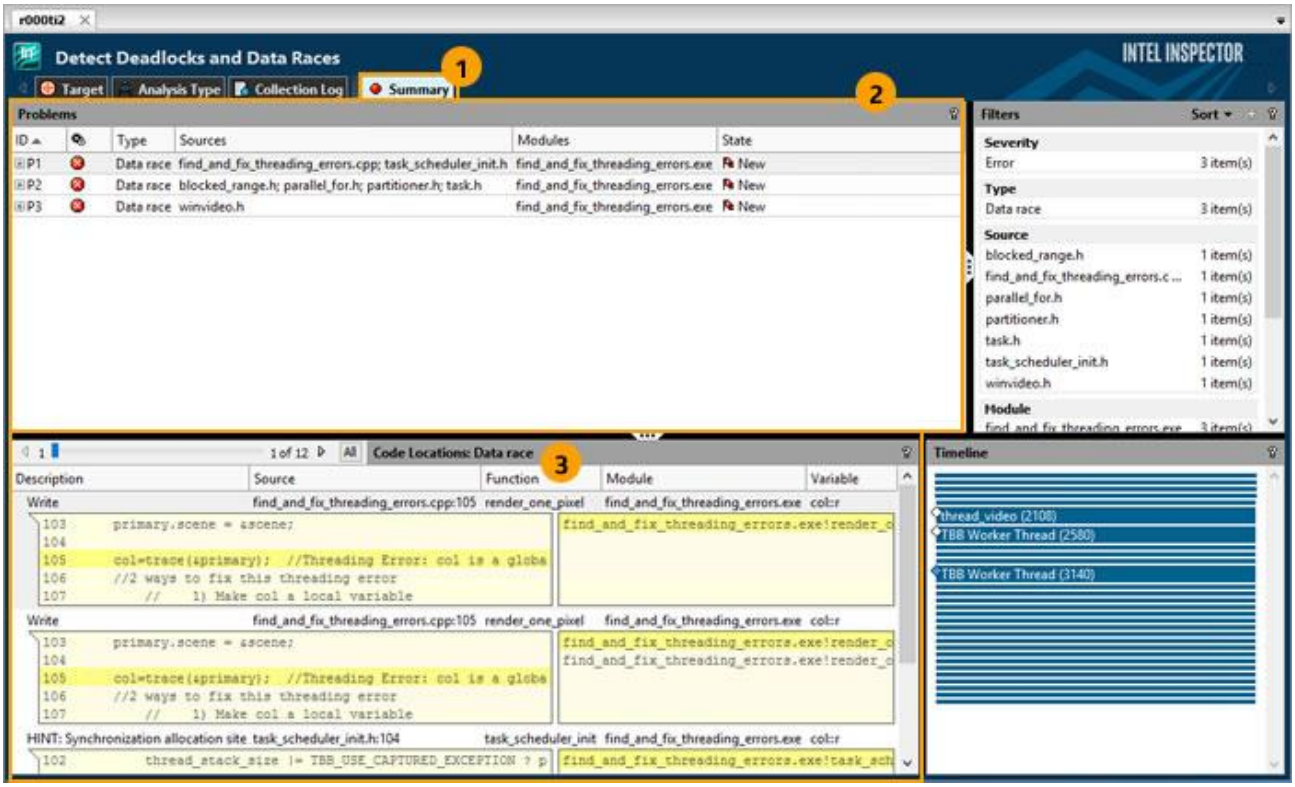
problem (問題): 初期化されていないメモリーアクセスなど、1 回または複数回検出された問題。問題が複数回検出された場合、コールスタックは同じですが、スレッドやタイムスタンプは異なります。問題とそれぞれの検出に関する情報を確認できます。

problem set (問題セット): アプリケーション実行中のオブジェクトの割り当て解除が早すぎることに起因する問題セットなど、解決策が共通している可能性がある、共通の問題タイプと共有コード位置を持つ問題のグループ。問題セットは、解析が完了した後にのみ表示されます。

その他の主要な用語については、「インテル® Inspector の用語集」(英語) を参照してください。

[Summary] ウィンドウ内の各種ペインを理解する

解析が完了すると、次のような **[Summary]** ウィンドウが自動的に表示されます。



1	**[Summary]** ウィンドウから結果データの管理を開始します。問題は問題セットにグループ化され、重要度とサイズで優先順位付けされます。
2	**[Problems]** ペインは、「To Do」リストと考えることができます。最上部から開始し、下へ進みます。 ⊞ アイコンをクリックして、問題セットに含まれる問題を確認します。
3	**[Code Locations]** ペインは、**[Problems]** ペインで選択した問題が検出されたすべてのコード位置について、コード位置の概要、周囲のソースコード、およびコールスタック情報を表示します。 スライダーの各種コントロールを使用して、選択した問題の異なる検出の情報を確認します。

問題を選択する

確認が終わったら、次の操作を行います。

1. `find_and_fix_threading_errors.cpp` ソースファイルの **[Data race]** 問題セットの横にある ⊞ アイコンをクリックして、問題セットに含まれるすべての問題を確認します。
2. 問題セット内の最初の **[Data race]** 問題のデータ行をダブルクリックして、**[Sources]** ウィンドウでエラーの原因に関する詳細を確認します。

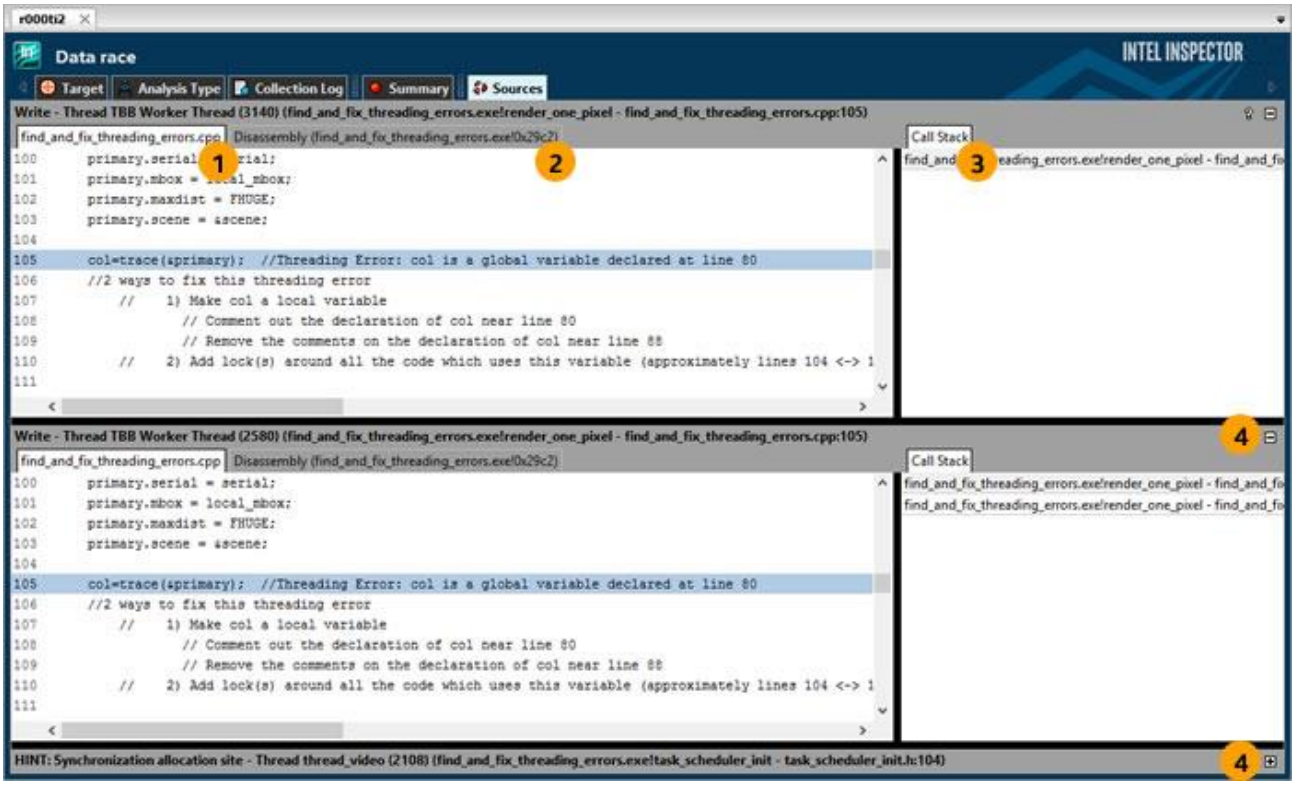
結果データの解釈

検出されたスレッドエラーの原因を特定します。

- [Sources] ウィンドウ内の各種タブを理解する
- 問題の解釈と解決に関する詳細情報にアクセスする

[Sources] ウィンドウ内の各種タブを理解する

問題セット内の最初の **[Data race]** 問題のデータ行をダブルクリックすると、次のような **[Sources]** ウィンドウが表示され、エラーの原因に関する詳細な情報を確認できます。



1	**ソース**タブは、**[Data race]** 問題のコード位置を含むソースコード・ファイル全体を表示します(サンプルは非決定的であるため、メモリー**書き込み**またはメモリー**読み取り**のコード位置が表示されます)。
2	**[Disassembly]** タブは、**[Data race]** 問題のコード位置に対応するアセンブリー命令を表示します。
3	**ソース**タブは、**[Data race]** 問題のコード位置に対応する完全なコールスタックを表示します。
4	この領域は、**[Data race]** 問題の別のコード位置のソース、逆アセンブリー、コールスタック情報を表示します(サンプルは非決定的であるため、メモリー**書き込み**またはメモリー**読み取り**のコード位置が表示されます)。 ⊞/⊟ アイコンを使用して、**[Data race]** 問題の各コード位置のソース、逆アセンブリー、コールスタック情報を展開/折りたたみます。

問題の解釈と解決に関する詳細情報にアクセスする

1. **ソース**タブまたは **[Disassembly]** タブ内の任意の場所を右クリックします。
2. **[Explain Problem]** を選択して、**[Data race]** 問題タイプに関するインテル® Inspector のヘルプを表示します。

問題の解決

検出されたスレッドエラーを修正します。

- 問題を調査する
- インテル® Inspector からエディターに直接アクセスする
- ソースコードを変更する

問題を調査する

`find_and_fix_threading_errors.cpp` サンプルファイルに埋め込まれているコメントから、データ競合の原因は、複数のスレッドが同時にグローバル変数 `col` にアクセスしているためだと分かります。この問題を修正する 1 つの方法は、グローバル変数をローカル変数に変更することです。

エディターにアクセスする

ソースタブでコードの 80 行目付近にスクロールして `color col;` 行をダブルクリックし、エディターで `find_and_fix_threading_errors.cpp` ソースファイルを開きます。

ソースコードを変更する

1. `color col;` をコメントアウトして、88 行目付近の `//color col;` のコメントを外し、変数 `col` を関数 `render_one_pixel` のローカル変数にします。
2. 編集内容を保存します。
3. 結果タブをクリックして **[Sources]** ウィンドウに戻ります。

   注
   [Sources] ウィンドウのデータは、解析時のソースコードであるため、変更内容は反映されません。

4. **[Summary]** ボタンをクリックして **[Summary]** ウィンドウを再度表示します。

リビルドと解析の再実行

ソースコードの変更により **[Data race]** 問題が修正されたかどうか確認します。

- 変更したソースコードでアプリケーションをリビルドする
- 解析を再度実行する

アプリケーションをリビルドする

ターミナルセッションで、次の操作を行います。

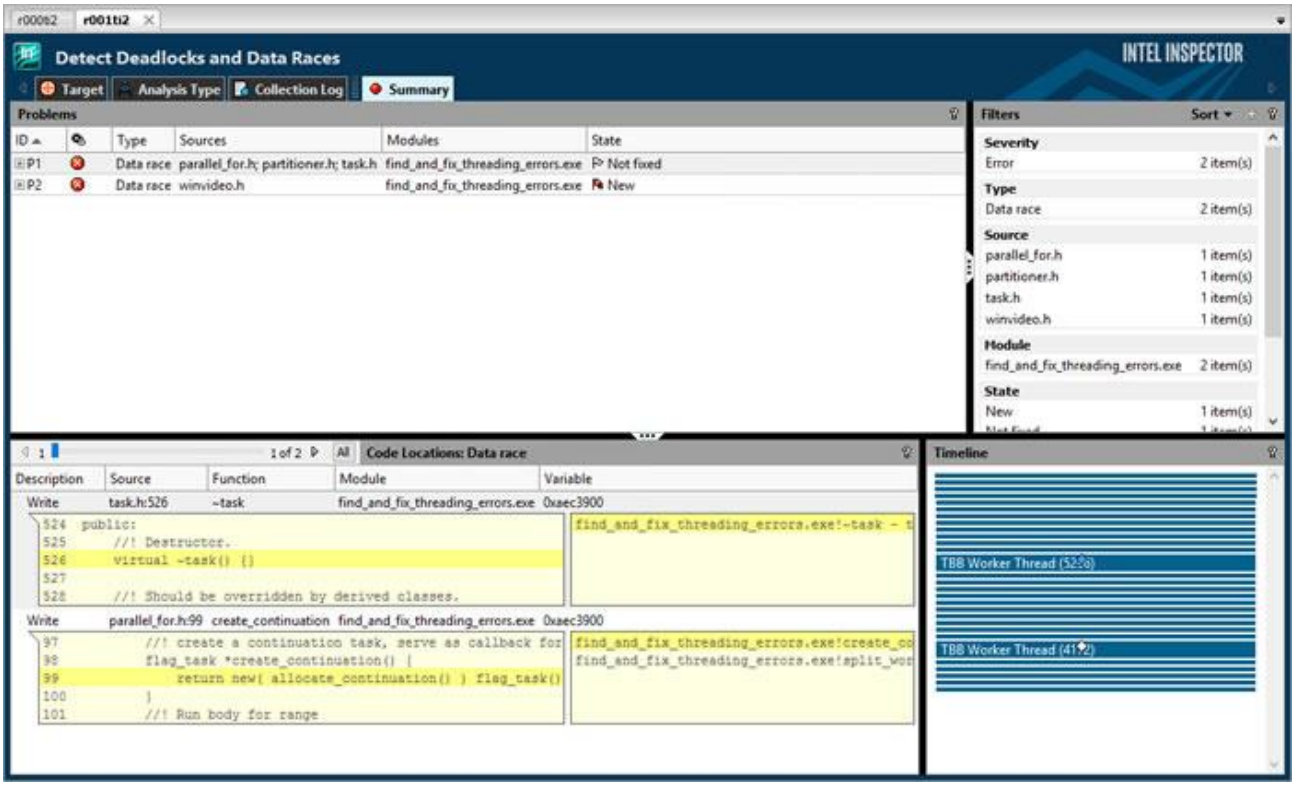
1. ディレクトリーを `tachyon_insp_xe/` に変更します。
2. make を実行します。

解析を再度実行する

同じ解析タイプの別の解析を実行するには、インテル® Inspector スタンドアロン GUI のメニューから、**[File]** > **[New]** > **[Threading Error Analysis / Detect Deadlocks and Data Races]** を選択します。

アプリケーションの出力が適切に表示されるようになります。

解析 (収集とファイナライズの両方) が完了すると、**[Summary]** ウィンドウが自動的に表示されます。



次のことに気付くでしょう。

- 新しい結果タブ「**r002ti2**」が作成されます。
- `find_and_find_threading_errors.cpp` ソースファイルで **[Data race]** 問題が検出されなくなりました。

まとめ

このチュートリアルでは、最終的に独自のアプリケーションに適用可能なエンドツーエンドのワークフローを示しました。

ステップ	チュートリアルの内容	要点
1. 設定	最適なコンパイラー/リンカー設定でアプリケーションをビルドして、インテル® Inspector の外部で実行されることを確認し、インテル® Inspector 環境を設定して、解析結果を保持するプロジェクトを作成します。	次のオプションでデバッグモードでコンパイル/リンクされたアプリケーションは、最も正確で完全な結果を生成します: `-g`、`-O0`、`-shared-intel` (インテル® コンパイラー)、およびデフォルトまたは `-Bdynamic` (GNU* コンパイラー) を指定し、`-fmudflap` を指定しない。
2. 結果の収集	解析タイプを選択して解析を実行します。解析中、インテル® Inspector は次の処理を行います。 • アプリケーションを実行して、対応が必要なエラーを識別し、結果を収集して結果タブに表示します。 • **[Project Navigator]** に結果へのポインターを追加します。	• インテル® Inspector には、解析の範囲とコストの制御を支援するため、事前定義済みのさまざまな解析タイプが用意されています。解析の範囲を広くすると、システムの負荷、および解析の実行に必要な時間とリソースが増えます。 • **[File]** メニュー、ツールバー、またはコマンドライン (`inspxe-cl` コマンドを使用) からエラー解析を実行します。

ステップ	チュートリアルの内容	要点
3. 結果の調査	検出された問題を調査し、結果データを解釈して、インテル® Inspector からエディターに直接アクセスしてソースコードを変更します。	• 主要な用語: 「code location (コード位置)」は、インテル® Inspector が観測したソースコードの場所。「problem (問題)」は、1 回または複数回検出された問題。「problem set (問題セット)」は、解決策が共通している可能性がある、共通の問題タイプと共有コード位置を持つ問題のグループ。 • **[Summary]** ウィンドウの **[Problems]** ペインを「To Do」リストと考えて、最上部から開始して下へ進みます。 • **[Summary]** ウィンドウでコード位置または問題をダブルクリックして **[Sources]** ウィンドウに移動します。**[Sources]** ウィンドウの **[Summary]** ボタンをクリックして **[Summary]** ウィンドウに戻ります。 • **[Summary]** ウィンドウまたは **[Sources]** ウィンドウ内を右クリックしてコンテキスト・メニューを表示し、**[Explain Problem]** を選択して問題の解釈と解決に関する詳細情報にアクセスします。 • **[Sources]** ウィンドウでコード位置をダブルクリックしてエディターを開きます。
4. 作業の確認	アプリケーションをリビルド/再リンクして、再度調査します。	

次のステップ: 解析用に独自のアプリケーションを準備して、インテル® Inspector でエラーを見つけて修正します。

法務上の注意書き

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。

インテル® テクノロジーの機能と利点はシステム構成によって異なり、対応するハードウェアやソフトウェア、またはサービスの有効化が必要となる場合があります。実際の性能はシステム構成によって異なります。絶対的なセキュリティーを提供できる製品またはコンポーネントはありません。詳細については、各システムメーカーまたは販売店にお問い合わせいただくか、http://www.intel.co.jp/ を参照してください。

インテルは、明示されているか否かにかかわらず、いかなる保証もいたしません。ここにいう保証には、商品適格性、特定目的への適合性、および非侵害性の黙示の保証、ならびに履行の過程、取引の過程、または取引での使用から生じるあらゆる保証を含みますが、これらに限定されるわけではありません。

本資料には、開発中の製品、サービスおよびプロセスについての情報が含まれています。本資料に含まれる情報は予告なく変更されることがあります。最新の予測、スケジュール、仕様、ロードマップについては、インテルの担当者までお問い合わせください。

本資料で説明されている製品およびサービスには、設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。現在確認済みのエラッタについては、インテルまでお問い合わせください。

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 またはその子会社の商標です。

* 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。

本ソフトウェアおよび関連ドキュメントは、インテルが著作権を有する著作物であり、その使用には付随する明示的なライセンス (「**ライセンス**」) が適用されます。ライセンスで特に明記されていない限り、インテルから書面による許可を得た場合を除き、本ソフトウェアまたは関連ドキュメントを使用、改変、複製、公表、配布、公開することはできません。

本ソフトウェアおよび関連ドキュメントは現状のまま提供され、ライセンスに明記されているものを除き、明示されているか否かにかかわらず、いかなる保証もいたしません。

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。