インテル® VTUNE™ AMPLIFIER XE を利用した PYTHON* コードの高速化

非常に低いオーバーヘッドで行レベルのプロファイリング情報を提供

Kevin O'Leary インテル コーポレーション ソフトウェア・テクニカル・コンサルティング・エンジニア

インテル® VTune™ Amplifier XE 2017 (**インテル® Parallel Studio XE** に含まれる) では、Python* スクリプトを解析してアプリケーションで時間が費やされている場所を正確に特定できるようになりました。この記事では、この新しいインテル® VTune™ Amplifier XE の Python* 機能の使用方法について説明します。

Python* の最適化が必要な理由

- Python* は、アプリケーションのパフォーマンスが重要なケースを含め、広範なソフトウェアで使用されています。例: Web サーバーコード、複雑な自動化スクリプト (ビルドシステムを含む)、科学計算。
- Python* はコードを素早く記述できますが、プロセスに十分なワークロードが与えられないと、コードが適切にスケーリングしません。

最適化が必要なコードの場所の特定

次のような方法があります。

- **コードの調査。** コードエディター以外のツールを必要としない、最も簡単な方法ですが、作業は簡単ではありません。特定のコードがどのように実行されるか正しく理解している必要があります。理解すべき情報が多すぎるため、非常に小さなコードベースを除いて適していません。
- **ログ収集。** タイムスタンプを記録する特別なバージョンのソースコードを作成する方法です。記録を確認してコードの遅い部分を調べます。この解析は骨の折れる作業であり、ソースの変更も必要です。
- **プロファイル。** プロファイルは、特定のワークロードの下でアプリケーションの動作を示すいくつかのメトリックを収集します。この記事では、CPU hotspot プロファイル (大量の CPU サイクルを費やしている関数の検索およびチューニング) について説明します。ロックの待機、メモリー消費などのプロファイルも可能ですが、典型的な Python* アプリケーションでは、CPU 使用率が問題になることが多く、CPU hotspot を調べることにより、メモリー消費ポイントを特定できます (特に大きな Python* シーケンス・オブジェクトを処理する場合、メモリーの割り当てと解放により CPU 使用率が非常に高くなります)。

アプリケーションをプロファイルするツールが必要になりますが、ツールの導入により得られる生産性の向上は十分に価値のあるものです。

既存のツールの概要

イベントベースのツールは、関数の入口 / 出口、クラスのロード / アンロードなど、特定のイベントでデータを収集します。ビルトイン Python* cProfile プロファイラー (および sys.setprofile /sys.settrace API) はイベントベース収集の例です。

インストルメンテーションベースのツールは、データを収集するためにターゲット・アプリケーションを変更します。この変更は、手動で行うか、コンパイラーやほかのツールを使用して行います。

統計ツールは、一定の間隔でデータを収集します。最も頻繁に実行された関数が、「関数別サンプル」分布の一番上になります。これらのツールはおおよその結果を提供し、プログラムの実行にあまり干渉しません (ターゲット・アプリケーションへの影響はわずかで、小さな関数の頻繁な呼び出しにも影響を受けません)。また、プロファイル・オーバーヘッドはワークロードにほとんど依存しません。

インテル® VTune™ Amplifier XE を利用した Python* コードのプロファイル

インテル® VTune™ Amplifier XE で、Python* コードをプロファイルできるようになりました。非常に低いオーバーヘッド (Grand Unified Python* Benchmark で約 1.1 倍から 1.6 倍) で、行レベルのプロファイル情報を取得できます。Windows® および Linux* をサポートしています。主な機能は次のとおりです。

- Python* 32 ビットおよび 64 ビット
- 2.7.x および 3.4.x ブランチ (2.7.10 および 3.4.3 でテスト)
- SSH によるリモート収集のサポート
- マルチスレッドのサポート、ズームおよびフィルター、ソースのドリルダウンを含む UI

次のワークフローをサポートしています。

- アプリケーションを開始して、終了を待機する。
- アプリケーションにアタッチし、すぐに短いプロファイルを行い、デタッチする。
- インタープリターの初期化および終了時に実行される多くの Python* コードを含む、すべての段階でコードをプロファイルする。

Python* コードを解析する手順は次のとおりです。

- インテル® VTune™ Amplifier XE プロジェクトを作成する。
- **Basic Hotspots 解析**を実行する。
- 結果データを解釈する。

プロジェクトの作成

インテル® VTune™ Amplifier XE でターゲットを解析するには、プロジェクトを作成して解析ターゲットおよびデータの収集方法を設定する必要があります。

a. **`amplxe-gui`** スクリプトを実行して、インテル® VTune™ Amplifier XE の GUI を起動します。

b.  メニューボタンをクリックし、**[New (新規)] > [Project... (プロジェクト ...)]** を選択します。**[Create a Project (プロジェクトの作成)]** ダイアログボックスが表示されます。

c. プロジェクト名を **`test_python`** にします。

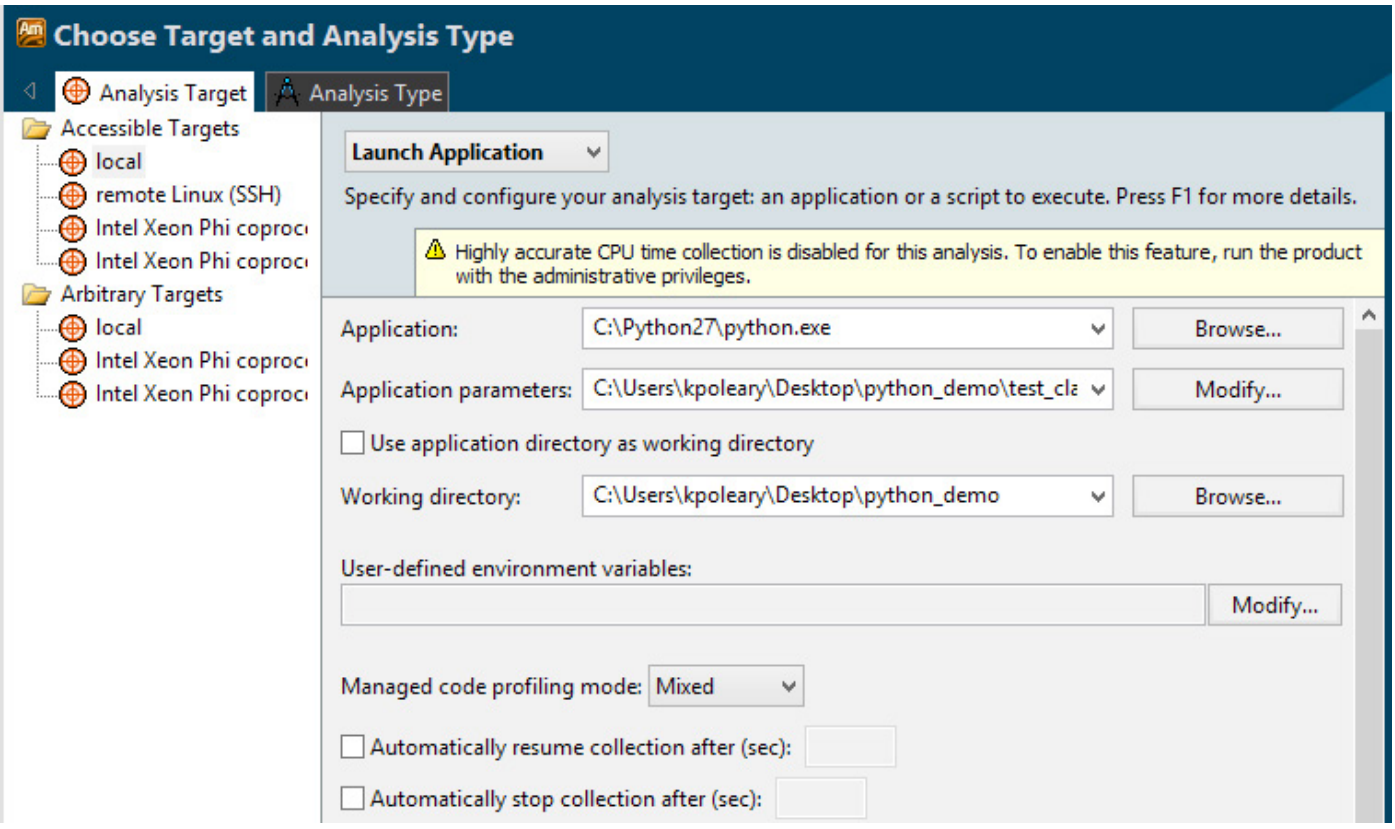
インテル® VTune™ Amplifier XE は、**`$HOME/intel/ampl/projects`** ディレクトリーの下に **`test_python`** プロジェクト・ディレクトリーを作成して、**[Choose Target and Analysis Type (ターゲットと解析タイプの選択)]** ウィンドウの **[Analysis Target (解析ターゲット)]** タブを開きます **(図 1)**。

a. 左ペインから **[local (ローカル)]** ターゲットシステム、右ペインから **[Application to Launch (起動するアプリケーション)]** ターゲットタイプを選択します。

b. 右の設定ペインが更新され、選択したターゲットタイプに適用可能な設定が表示されます。

c. 次のようにターゲットを指定して設定します。

- **[Application (アプリケーション)]** フィールドに、Python* 実行ファイルを指定します。
- **[Application parameters (アプリケーションのパラメーター)]** フィールドに、Python* スクリプトを入力します。
- プログラムの作業ディレクトリーを指定します。
- **[Managed code profiling mode (マネージドコードのプロファイリング・モード)]** プルダウンを使用して、**[Mixed (混在)]** を指定します。



1 インテル® VTune™ Amplifier XE でプロジェクトを作成する

Basic Hotspots 解析の実行

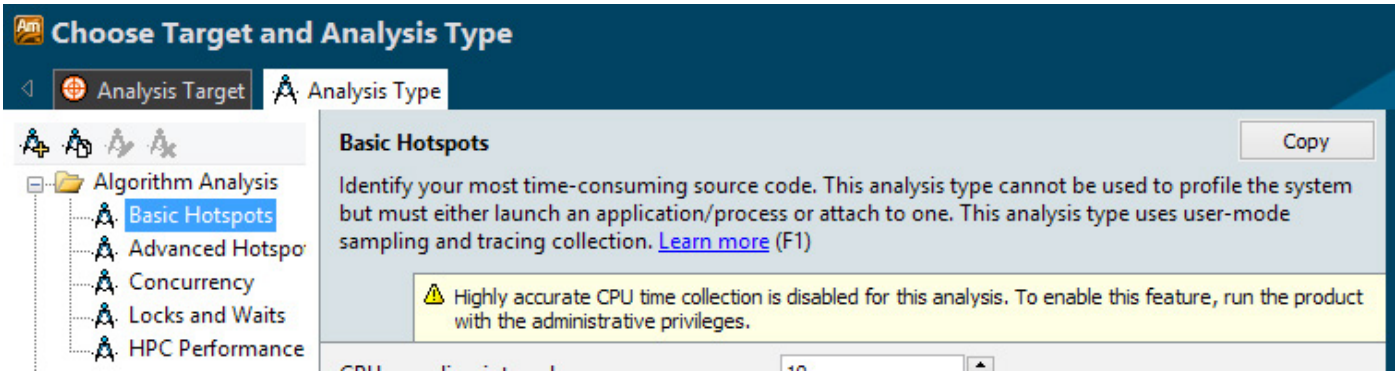
a. 右の **[Choose Analysis (解析の選択)]** ボタンをクリックして、**[Analysis Type (解析タイプ)]** タブに切り替えます。

b. **[Choose Target and Analysis Type (ターゲットと解析タイプの選択)]** ウィンドウで、**[Analysis Type (解析タイプ)]** タブに切り替えます。

c. 左の解析ツリーから、**[Algorithm Analysis (アルゴリズム解析)] > [Basic Hotspots (基本 hotspot)]** を選択します。

d. 右ペインが更新され、Basic Hotspots 解析のデフォルトオプションが表示されます。

e. 右のコマンドバーの **[Start (開始)]** ボタンをクリックして解析を実行します。



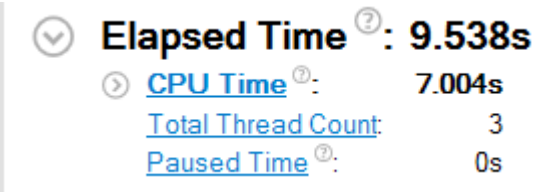
結果データの解釈

サンプル・アプリケーションが終了すると、インテル® VTune™ Amplifier XE は結果をファイナライズして、**hotspot** を表示します。ウィンドウまたはペインには、多くの CPU 時間を費やしたコード領域が表示されます。サンプルコードのデータのパフォーマンスを解釈するには、次の操作を行います。

- Python* hotspot 解析により提供される**基本的なパフォーマンス・メトリックを理解します**。
- **最も時間を費やしている関数と CPU 使用率を解析します**。
- **スレッドごとのパフォーマンスを解析します**。

Python* hotspot のメトリックを理解する

[Summary (サマリー)] ウィンドウで解析を開始します。データを解釈するには、? アイコンの上にマウスのカーソルを移動して、ポップアップ・ヘルプを読み、各パフォーマンス・メトリックの意味を確認します。



サンプル・アプリケーションの **[CPU Time (CPU 時間)]** は 7.004 秒です。これはすべてのアプリケーション・スレッドの CPU 時間の合計です。**[Total Thread Count (スレッドの総数)]** は 3 で、サンプル・アプリケーションはマルチスレッド化されています。

[Top Hotspots (上位の hotspot)] セクションには、最も時間が費やされた関数 (hotspot 関数) のデータが CPU 時間でソートして表示されます。

Top Hotspots

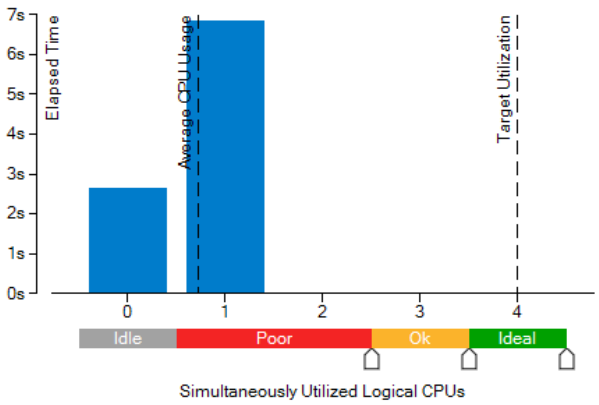
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
func@0x1e0eda10	python27.dll	2.894s
do_work	test_class_sample.py	1.580s
PyObject_GetAttr	python27.dll	1.229s
func@0x1e0eccd0	python27.dll	0.610s
func@0x1e08f4a0	python27.dll	0.280s
[Others]	N/A*	0.411s

*N/A is applied to non-summable metrics.

CPU Usage Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU usage value.



BLOG HIGHLIGHTS

インテル® Xeon Phi™ プロセッサー (開発コード名: Knights Landing) 向けデベロッパー・アクセス・プログラム

MICHAEL P >

インテルは、インテル® Xeon Phi™ プロセッサー (開発コード名 : Knights Landing) の一般販売に先立ち、このプロセッサー向けのデベロッパー・アクセス・プログラム (DAP) を開始しました。DAP は、インテル® Xeon Phi™ プロセッサー・ベースのシステムを購入するための、世界中の開発者を対象とした早期アクセスプログラムです。開発者が、コードの開発、アプリケーションの最適化、パフォーマンスの確認を開始できるように、単一のブート可能なインテル® Xeon Phi™ プロセッサー (開発コード名 : Knights Landing) を搭載した、スタンドアロンの筐体が用意されています。

この記事の続きはこちら (英語) でご覧になれます。>

混在モード解析を実行する

Python* はインタープリター型言語で、コンパイラーを使用しないでバイナリー実行コードを生成します。Cython も (拡張子は C ですが) インタープリター型言語で、ネイティブコードをビルドできます。インテル® VTune™ Amplifier XE 2017 は、ホットな関数のレポートにおいて Python* コードと Cython コードの両方をサポートしています。

まとめ

チューニングを行うと、コードのパフォーマンスを大幅に向上できます。インテル® VTune™ Amplifier XE で、Python* コードをプロファイルできるようになりました。Python* コードの解析に加えて、Python*/Cython の混在コードの解析を行うこともできます。インテル® VTune™ Amplifier には、パフォーマンス・ボトルネックを素早く特定できるように支援する強力な機能が用意されています。