



# Using Intel® VTune™ Amplifier XE with Intel® SGX Enabled Applications on Microsoft\* Windows\*

## Scope

This paper describes how to use Intel® VTune™ Amplifier XE to gather and analyze performance data from Intel® Software Guard Extensions (Intel® SGX) enabled applications for Microsoft\* Windows\*. Intel VTune Amplifier XE is an application for software performance analysis of 32- and 64-bit x86 based machines. A basic knowledge of Intel SGX is assumed. General information on Intel SGX is provided on the Intel SGX portal at: <https://software.intel.com/sgx>.

## Introduction

This paper covers the following topics:

- Initial setup of a VTune project for an Intel SGX enabled application.
- Performing local data gathering for the application/enclave code.
- Understanding the VTune data to make useful observations that can be used to improve the performance of applications.

An Intel® SGX application is divided into two logical components:

- Trusted component — The code that accesses the secret resides here. This component is also called an enclave. More than one enclave can exist in an application.
- Untrusted component — The rest of the application including all its modules.

Enclave measurement is done using Intel VTune Amplifier XE. VTune can be used to measure the performance of enclave code *only* when the enclave has been launched as a Debug enclave. For details on build and debug configurations, see:

<https://software.intel.com/sites/default/files/managed/e5/d8/intel-sgx-build-configuration.pdf>.

Ultimately, it is the workload that drives applications to a steady state so that VTune can help understand where the most time is being spent. The developer:

- Decides the performance goals for the application (for example, X number of MBs per second for encryption throughput)
- Sets a realistic goal for application workload
- Uses VTune to analyze actual performance of the application while it runs

## Intel® Software Guard Extensions (Intel® SGX)

- Compares performance data with the goals for the application to understand actual performance and where the most time is being spent

For Intel SGX applications, developers will typically focus their analysis on the time spent executing Ecalls in the Enclave and will optimize their code as needed to achieve a performance goal. For an overview of performance considerations related to Intel SGX applications see: <https://software.intel.com/sites/default/files/managed/09/37/Intel-SGX-Performance-Considerations.pdf>.

## Setting up and running an analysis

The following SW must be installed:

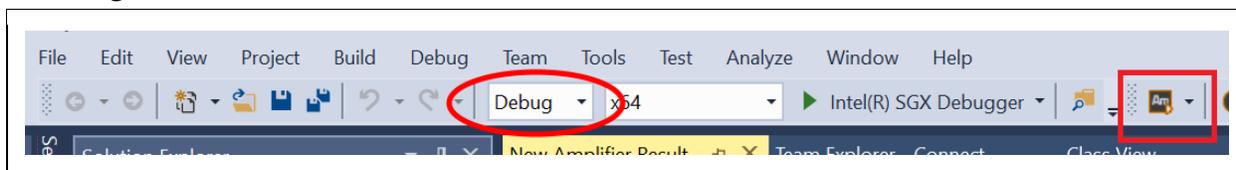
- Microsoft Visual Studio 2015 (Professional or higher)
- Intel SGX Platform Software v1.8 (or newer) for Windows\*
- Intel SGX SDK v1.8 (or newer) for Windows\*
- Intel VTune Amplifier XE Application 2017 — Update 5 or newer (and the required drivers): <https://registrationcenter.intel.com/forms/?ProductID=1503>
  - Installation steps for Intel VTune Amplifier XE are provided at: <https://software.intel.com/en-us/vtune-amplifier-install-guide-windows>.

When VTune XE is installed with Visual Studio 2015, it integrates into the Visual Studio toolset. You build your application and run it to make sure it works as intended (as you typically do). Then you can import Visual Studio project information into VTune to create an analysis target and run the analysis on the application to gather performance information.

**Note:** This paper focuses only on using VTune within the Visual Studio environment. You can also define and launch a VTune analysis outside Visual Studio. Information to do that is available in the VTune documentation.

Follow these steps to set up an Intel SGX project and launch an initial analysis in VTune:

1. Build and debug your application in Visual Studio until you are satisfied that it is functionally working as intended. (In this paper, screenshots from some simple applications are shown.) Make sure that you build your application in Debug mode, shown in Figure 1, so that the resulting enclave code can be read by VTune later.
2. Invoke VTune Amplifier by clicking on the  icon on the Visual Studio toolbar, as shown in Figure 1.

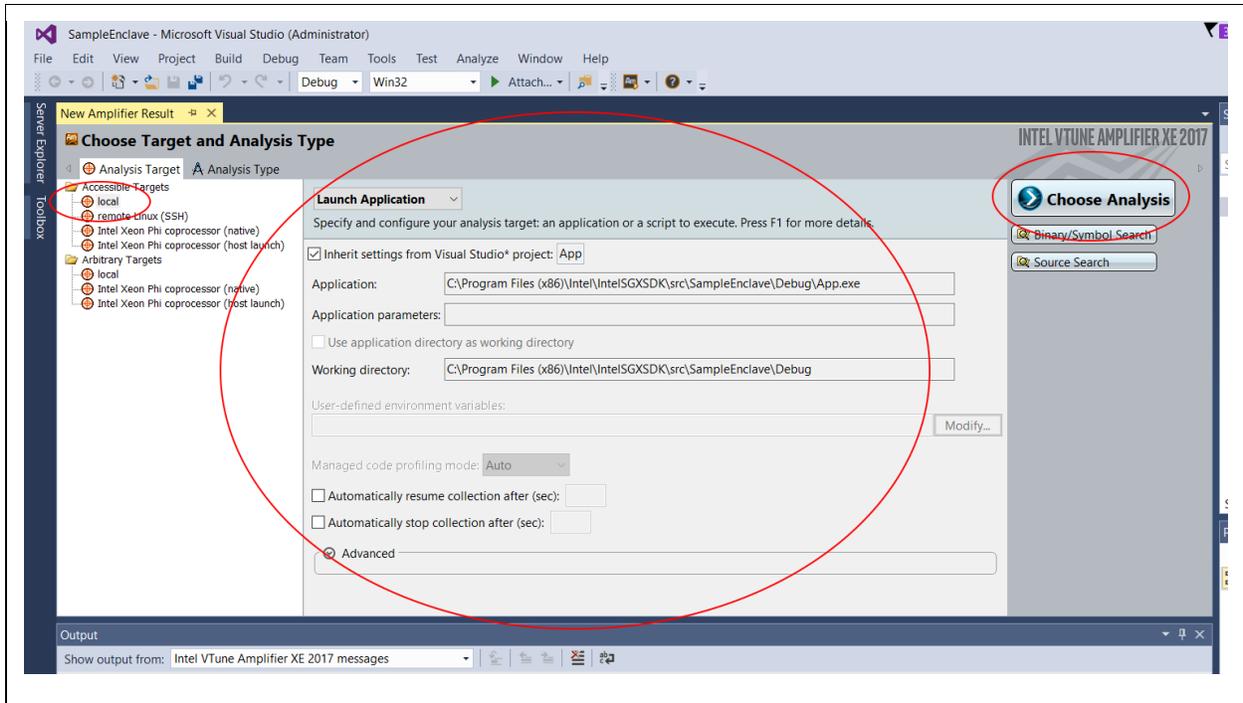


**Figure 1. Invoking VTune from Visual Studio toolbar**

## Intel® Software Guard Extensions (Intel® SGX)

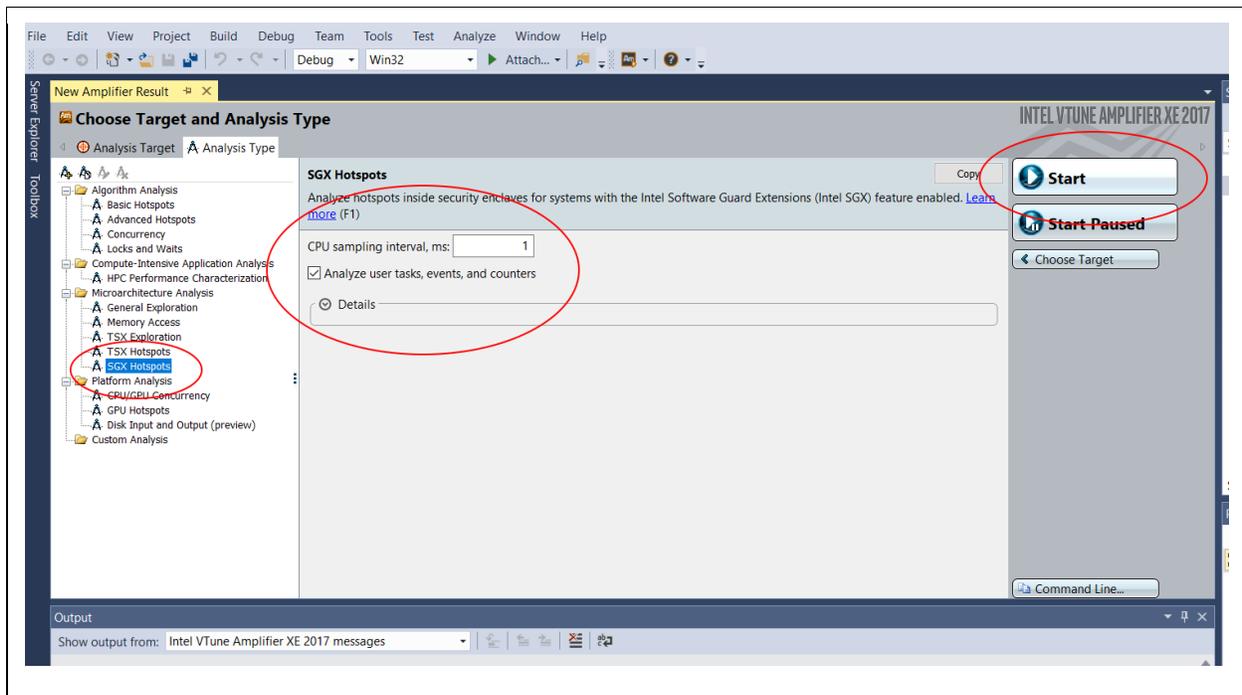
- VTune opens with the **Analysis Target** tab displayed, as shown in Figure 2. By default, VTune populates the fields on this tab with data from the open project in Visual Studio. Make sure to select **local** to run analysis on the local system. Then click **Choose Analysis** on the far right of the screen to move to the **Analysis Type** tab.

**Note:** Remote data gathering is also possible. Details are provided in the VTune documentation.



**Figure 2. VTune Analysis Target tab**

- VTune displays the **Analysis Type** tab, as shown in Figure 3. Select **SGX Hotspots** under **Microarchitecture Analysis**. Set the CPU sampling interval as **1 ms** (the default) and check **Analyze user tasks, events, and counters**. Click **Start** to launch the application and run the performance analysis.



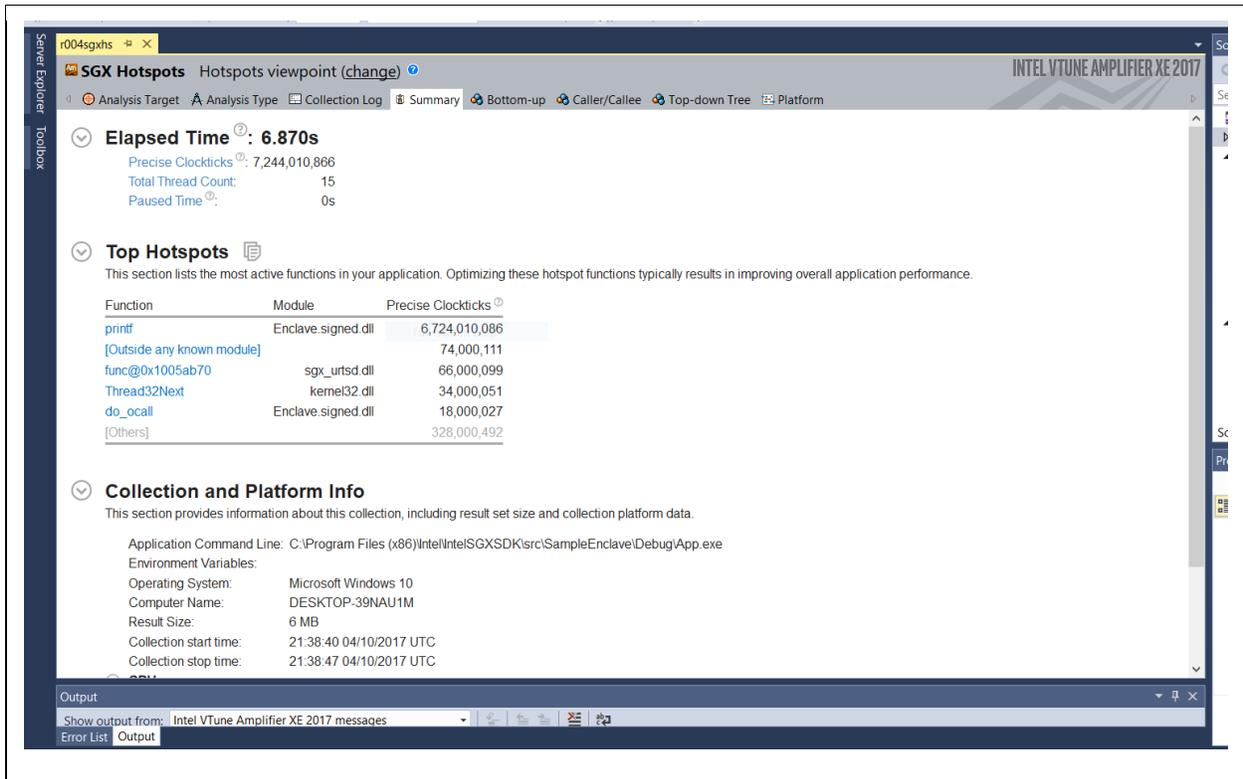
**Figure 3. VTune Analysis Type tab**

### VTune Results Tabs

VTune populates the **Collection Log** as it runs. When the analysis is complete, VTune opens the **Summary** tab to show overall results, as shown in Figure 4. The **Summary** tab provides a good starting point for analysis. Assuming that the analysis ran and collected results successfully, the **Summary** tab lists high-level information about the application analysis run and its performance, including the following:

- **Elapsed Time** data provides the overall frame of reference for your analysis.
- **Top Hotspots** data allows you to identify the functions where the most time is spent. You can click on a function name to see the source and code for that particular function.
- **Collection** and **Platform Info** tabs provide information regarding the processor, start time, and stop time.

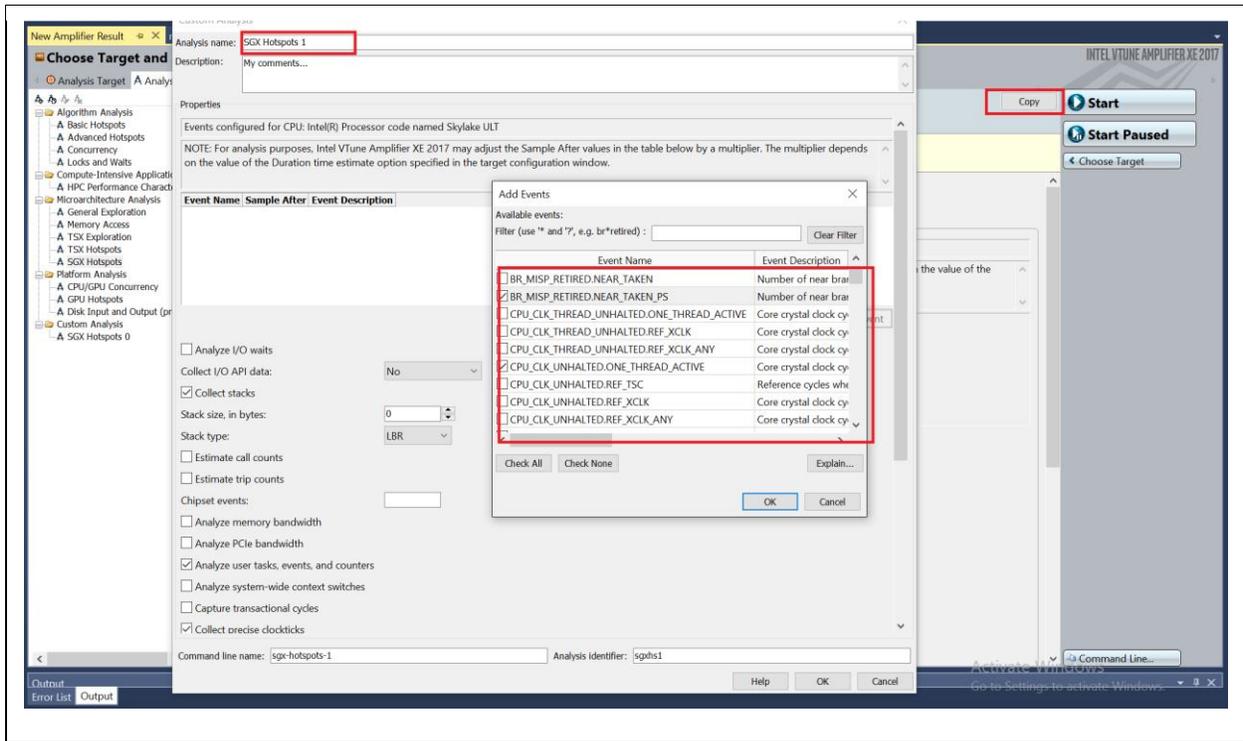
If issues occurred during the analysis that prevented results from being collected, examine the **Collection Log** tab to identify errors/conditions that prevented successful completion.



**Figure 4. VTune Summary tab including Top Hotspots**

### Hotspot analysis

You can click on any of the functions in **Top Hotspots** to see performance details for that function, as shown in Figure 5. Information regarding hotspot analysis using VTune is provided at: <https://software.intel.com/node/609046>.



**Figure 5. Details for a specific Hotspot**

Figure 6 shows the **Bottom-up** tab output, which allows you to compare behavior and performance of modules against each other.

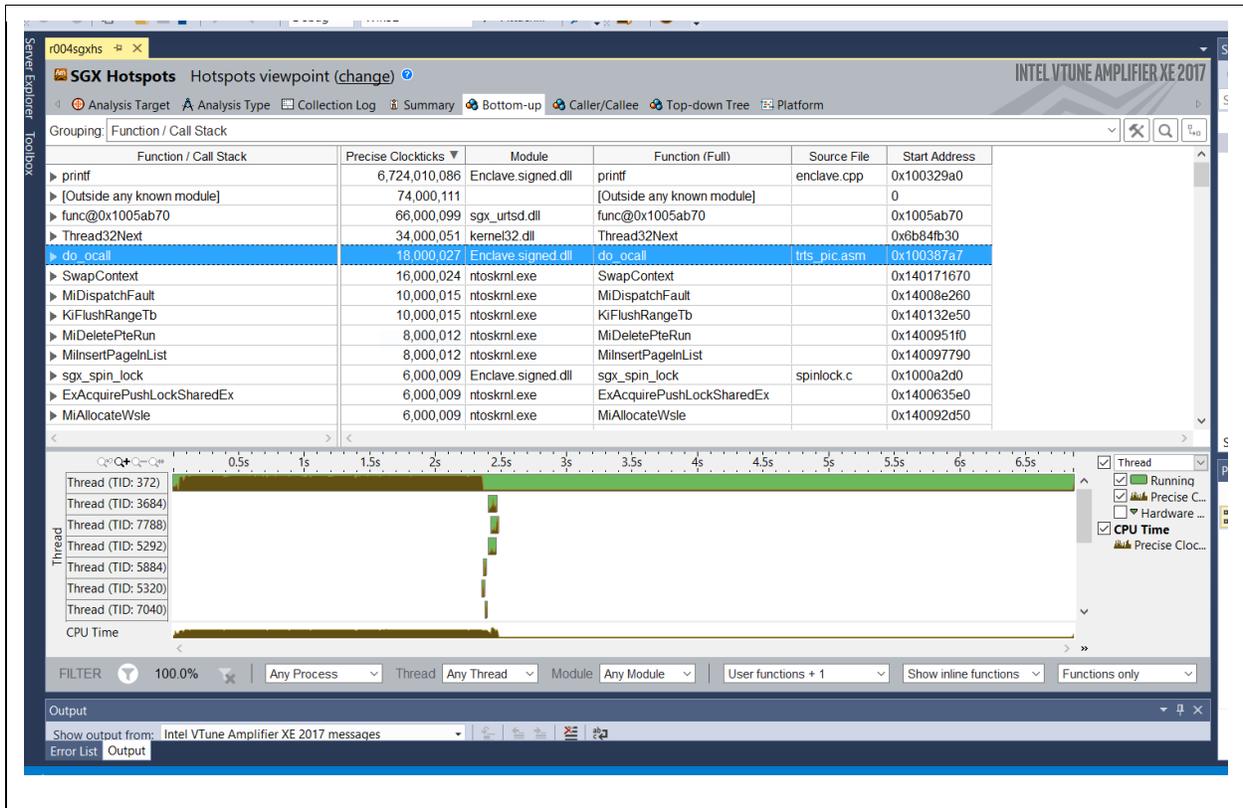


Figure 6. VTune Bottom-up analysis results

Enclave-specific profiling information can be viewed by using the **Module** filter on the **Bottom-up** tab, then choosing the enclave name. The output from enclave specific profiling is shown in Figure 7. In this case, the modules are for `sample_enclave.signed.dll`.

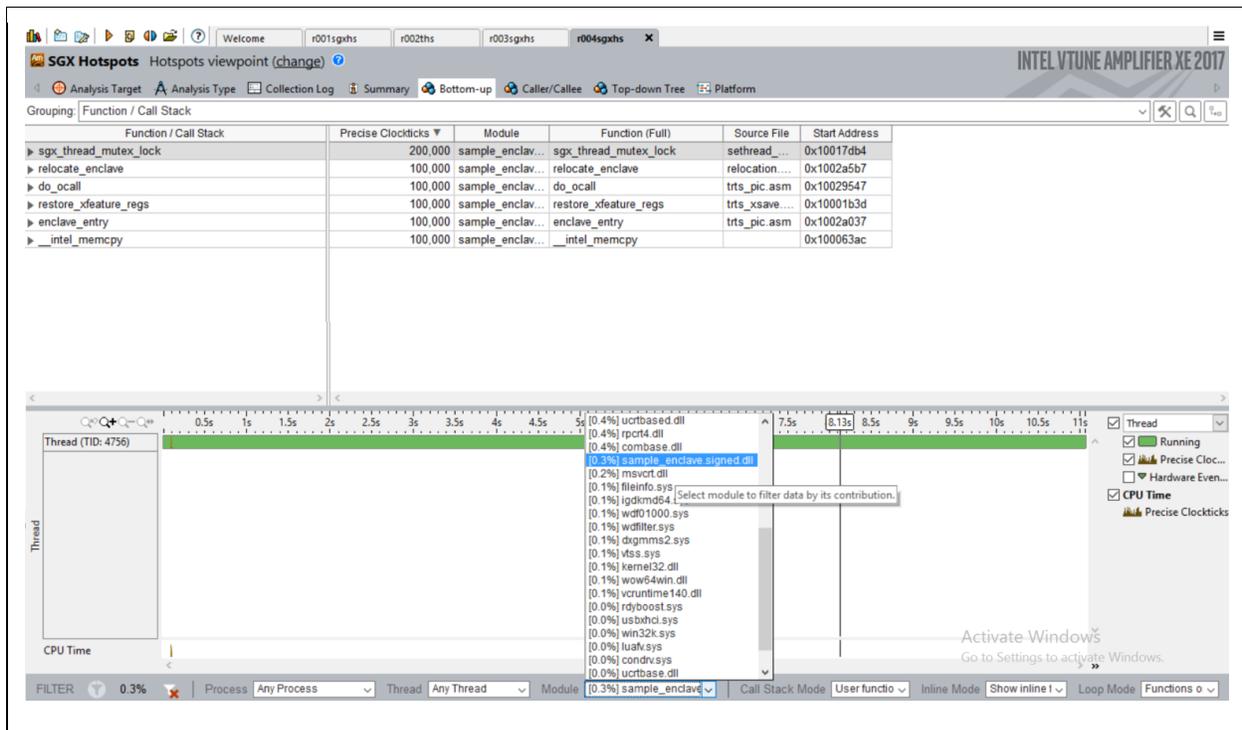
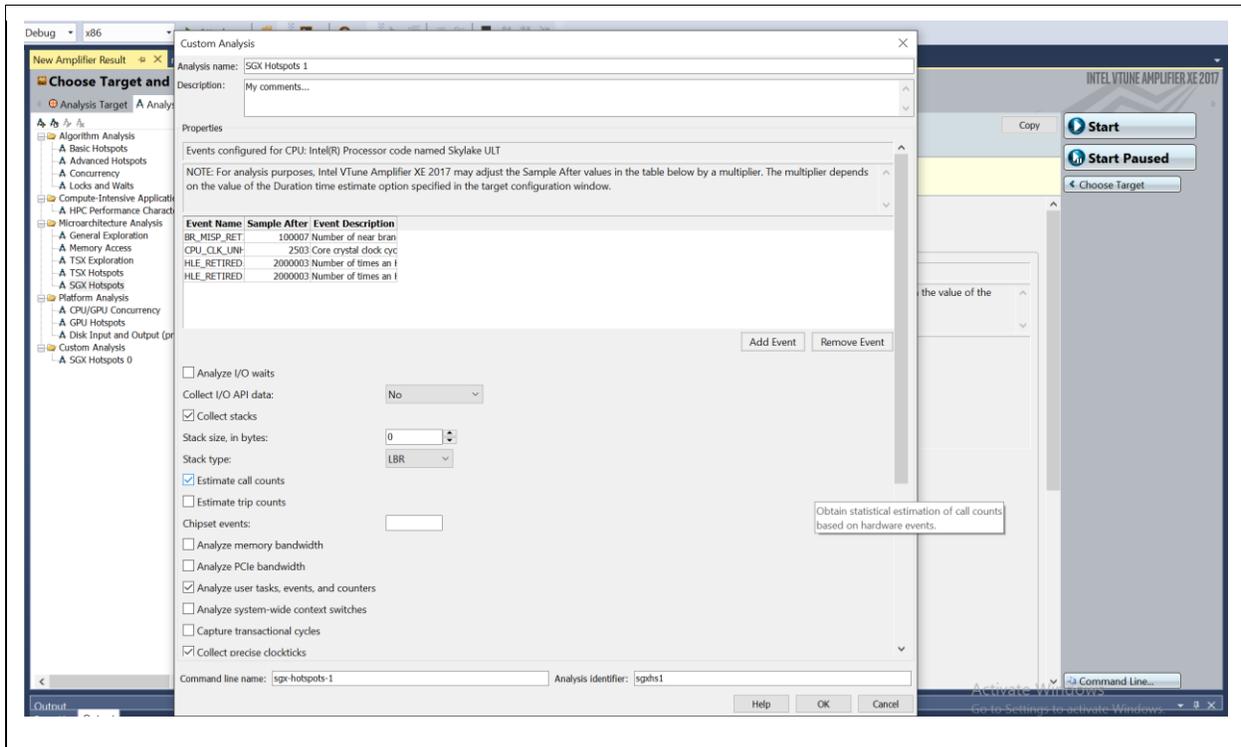


Figure 7. Analysis by Module

### Precise event-based sampling

Additional options like the **Collect stacks** checkbox can be enabled. Enclave code can also be profiled with the help of **Precise Event Based Sampling**. Events with `_PS` appended represent precise events and can be added to the collection to profile enclave code. To perform this, click the **Copy** button. This opens the dialog box in Figure 8. To collect more accurate call stacks for enclave code choose **LBR** from the **Stack type** drop-down menu as shown.

You can choose the analysis name then add an event of your choice. Once the event is added and the **Collect stacks** check box is enabled, the **Custom Analysis** window is displayed. Click **OK**. The analysis is now done based on the event selected.



**Figure 8. Setting up for event-based analysis**

### Loop analysis

You can also analyze the loops in the code by selecting **Loops only** in the lower dropdown menu as shown in Figure 9. This filter shows the elapsed time for a loop. Double clicking on the desired loop shows the source and assembly code. You can also filter for **Loops and functions**. VTune shows the performance and the elapsed time for each block and loop so you can spot areas for performance improvement.

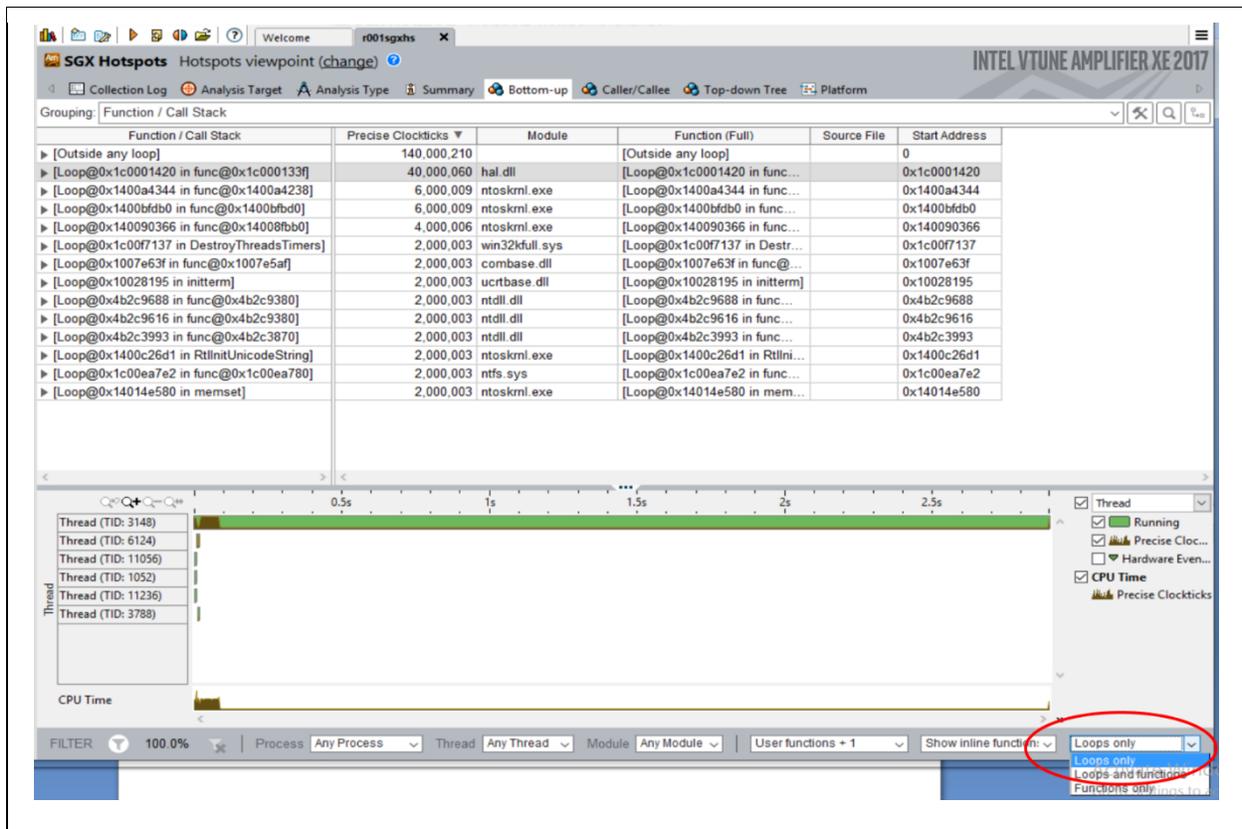


Figure 9. Analysis for loops present in the code.

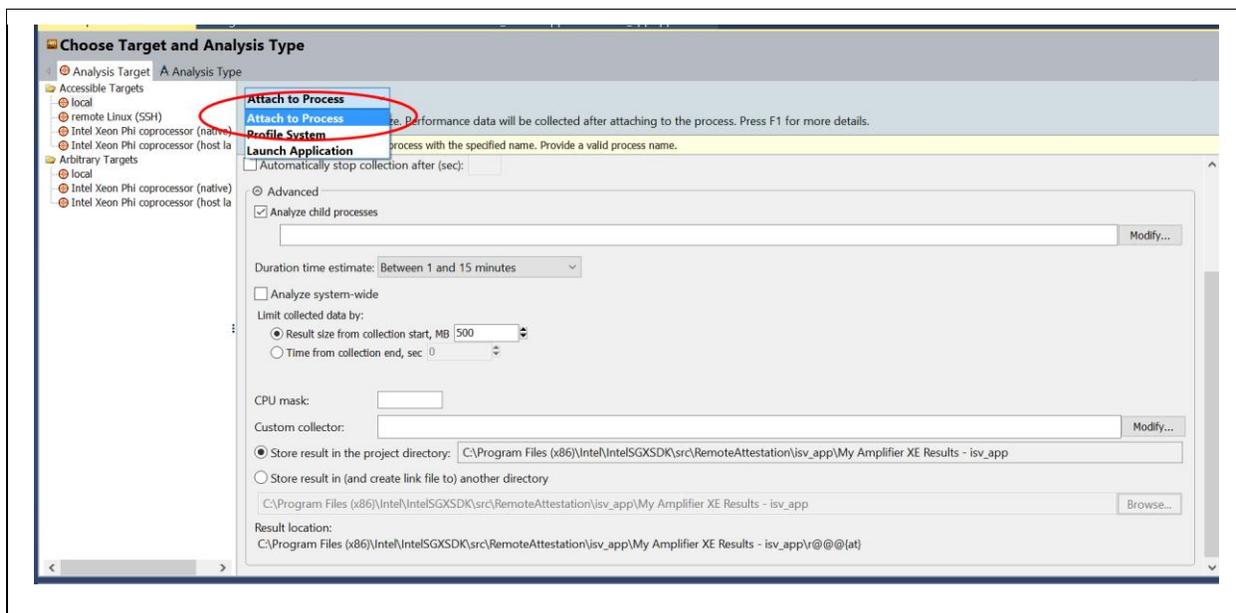
## Using VTune to attach to an enclave project

Another method for profiling applications is to attach to an already running process/application.

Before using VTune to attach to a currently running enclave application, a VTune specific environmental variable must be set by the user to indicate the intention to profile the enclave application. This is an expected behavior of VTune and is the standard way for setting up VTune to attach to a running process. The environmental variable is defined as follows:

```
Windows:
32bit:
INTEL_LIBITNOTIFY32 = <VTune Installation Dir>\bin32\runtime\itnotify_collector.dll
64bit:
INTEL_LIBITNOTIFY64 = <VTune Installation Dir>\bin64\runtime\itnotify_collector.dll
```

Figure 10 shows the menus used to attach VTune to a running process/application.



**Figure 10. Attaching VTune Analysis to an enclave project**

Intel SGX Applications are profiled by invoking VTune's ITT API in the uRTS that passes information about the enclave to VTune. This is done after the enclave has been loaded. When the user attaches VTune to the application after invoking VTune's ITT API, module information about the enclave is cached in the ITT shared library and is used by VTune application during attachment.

Information on available options when attaching VTune to a process are provided at: <https://software.intel.com/en-us/vtune-amplifier-help-local-attach-to-process-target-type>.

## Summary

Intel VTune Amplifier XE, integrated with Microsoft Visual Studio 2015, provides the capabilities to profile the performance of Intel SGX enabled applications. This paper gets you started by explaining VTune project setup, configuration of analysis targets, and launching of an analysis.

VTune supports Hotspots analysis of Intel SGX applications to help examine the modules consuming the most processor cycles. VTune also allows you to analyze specified events or designated loops.

Developers should compare VTune results with their knowledge of the application and their performance goals to understand where performance improvements may be needed.



## References

1. Getting Started with Intel VTune Amplifier 2017 for Systems (Windows Host) — 2017 update 5 — Intel Corporation. <https://software.intel.com/node/596477>.
2. Intel Software Guard Extensions SDK Developer Guide for Windows OS — 2017 Intel Corporation. <https://software.intel.com/sgx-sdk/documentation>.
3. Intel Software Guard Extensions Forums. <https://software.intel.com/en-us/forums/intel-software-guard-extensions-intel-sgx>.



## Intel® Software Guard Extensions (Intel® SGX)

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL® ASSUMES NO LIABILITY WHATSOEVER AND INTEL® DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL® AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL® OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL® PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

No computer system can provide absolute security under all conditions. Built-in security features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

Intel®, the Intel® Logo, Intel® Inside, Intel® Core™, Intel® Atom™, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and/or other countries. Other names and brands may be claimed as the property of others.

\* Other names and brands may be claimed as properties of others.

Copyright © 2017 Intel® Corporation