Using Intel® VTune™ Amplifier XE with Intel® SGX Enabled Applications on Linux*

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 the Linux* OS. 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:

- Setup verification and first use of VTune for an Intel SGX application/enclave.
- Performing local data collection for the application/enclave.
- Creating reports and understanding the VTune data to make useful observations that can be used to improve the performance of applications.

The main goal is to get you started using VTune with Intel SGX enabled applications/enclaves. Towards that goal, basic VTune command examples are included here. Details on VTune commands/options, however, are found in the VTune documentation.

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
White Paper

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 their 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.

Verifying setup and first use

The following SW must be installed:

- Intel SGX SDK v1.9 (or later) for Linux*
- Intel SGX Platform Software v1.9 (or later) for Linux*
  - The Intel SGX SDK and PSW can be downloaded, built, and installed by following the instructions on GitHub at: https://github.com/01org/linux-sgx.
- Intel VTune Amplifier XE Application 2017 — Update 5 or newer (and the required drivers): https://registrationcenter.intel.com/forms/?ProductID=2993
  - Installation steps for Intel VTune Amplifier XE are provided at: https://software.intel.com/vtune-amplifier-install-guide-linux.
  - Remember to source the path to VTune in your .bashrc file as described in the installation instructions.

Notes:

- When VTune is installed on Linux, it automatically installs the sampling driver kernel to support hardware events.
- This paper focuses on using the VTune CLI only. A GUI is also available for Linux. Information on installing/using the GUI is available in the VTune documentation.
- This paper describes local data gathering only. Remote data gathering is also possible. Details are provided in the VTune documentation.

Follow these steps to complete your first run with VTune on an Intel SGX application/enclave:

1. Build and debug your application in Visual Studio until you are satisfied that it is functionally working as intended. This sequence uses the Intel SGX SDK sample application SampleEnclave. Make sure that you build your application in Debug mode so that the resulting enclave code can be analyzed by VTune.

2. Figure 1 shows a VTune data collection session for the SampleEnclave application. CLI commands are shown in red. Application messages are shown in blue. The remaining messages/responses are from VTune or Linux. Since we will be discussing VTune reports later in this paper, the VTune messages have been shorted in this Figure.

3. Invoke the VTune CLI to collect hotspots for the application/enclave SampleEnclave. The basic command form is:
VTune launches the SampleEnclave application and starts gathering performance data. As the application executes, it requests the user to enter a character followed by <Enter>. The application displays the character on the screen and exits.

VTune completes its hotspot data collection, writes results to a local directory, displays Collection and Platform Info messages to the screen, then terminates. The Collection and Platform Info content provides a good starting point for analysis. Assuming the collection ran successfully, the messages include:

- Information regarding the processor, collection start time, and collection stop time.
- Hotspots data identifying the functions where the most time is spent.
- Elapsed time data providing the overall frame of reference for your analysis.

If issues occurred during the analysis that prevented results from being collected, examine the Collection and Platform Info messages to identify errors/conditions that prevented successful completion.

The default directory name for the first collection is r000sgxhs. VTune increments the three digits (000) in the directory name each time it performs a new collection.

4. View the current collection summary by invoking VTune with the -report action and the summary argument, as follows:

```
$ amplxe-cl -report summary
```
Figure 2 shows a collection summary for SampleEnclave. This is the same information as the Collection and Platform Info displayed during the default collection. The results include the directory name for the collection being displayed, in this case r000sgxhs.

```
$ amplxe-cl report summary
amplxe: Using result path `/home/user/projects/linux-sgx/SampleCode/SampleEnclave/r000sgxhs'
amplxe: Executing actions 75 % Generating a report
Collection and Platform Info
------------------------------------------
Parameter r000sgxhs
------------------------------------------
Application Command Line ./app
User Name user
Operating System 4.8.0-36-generic NAME="Ubuntu"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.3 LTS (Xenial Xerus)"
VERSION="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
Computer Name user-lab-sgx
Result Size 4544904
Collection start time 16:38:26 06/11/2017 UTC
Collection stop time 16:38:33 06/11/2017 UTC
CPU
--------------
Parameter r000sgxhs
--------------
Name Intel(R) Processor code named Kabylake
Frequency 4200027259
Logical CPU Count 8
TSX Hotspots
--------------
Parameter r000sgxhs
--------------
Precise Clockticks 21800327
Summary
-------
Elasped Time: 0.060
Event summary
--------------
Hardware Event Type Hardware Event Count:Self Hardware Event Sample Count:Self Events Per Sample
--------------
INST_RETIRED.TOTAL_CYCLES_PS 21800327 109 2000003
amplxe: Executing actions 100 % done
```

**Figure 2. VTune report summary messages**

Note:

Intel VTune Amplifier XE has options to format/store reports to directories/files that you specify. See the VTune documentation for details.
Hotspot analysis

View the top hotspots by changing the report argument to hotspots, as follows:

```bash
amplxe-cl -report hotspots
```

Figure 3 shows the messages for this report, which allow you to compare time spent by each function. Information regarding hotspot analysis using VTune is provided at:

```bash
$ amplxe-cl -report hotspots
amplxe: Using result path `/home/user/projects/linux-sgx/SampleCode/SampleEnclave/r000sgxhs'
amplxe: Executing actions 75 % Generating a report

<table>
<thead>
<tr>
<th>Function</th>
<th>Precise Clockticks</th>
<th>Module</th>
<th>Function (Full)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[vmlinux]</td>
<td>[vmlinux]</td>
<td>128,000,192</td>
<td>__morestack</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0</td>
<td>40,000,060</td>
<td>libsgx_urts.so</td>
</tr>
<tr>
<td>do_ocall</td>
<td>0x6200</td>
<td>20,000,030</td>
<td>enclave.signed.so</td>
</tr>
<tr>
<td>CTrustThreadPool::get_bound_thread</td>
<td>0x9139</td>
<td>4,000,006</td>
<td>libsgx_urts.so</td>
</tr>
<tr>
<td>save_and_clean_xfeature_regs</td>
<td>0x91d0</td>
<td>4,000,006</td>
<td>enclave.signed.so</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0x8980</td>
<td>2,000,003</td>
<td>enclave.signed.so</td>
</tr>
<tr>
<td>[Outside any known module]</td>
<td>0</td>
<td>2,000,003</td>
<td>[Outside any known module]</td>
</tr>
<tr>
<td>__GI___pthread_mutex_lock</td>
<td>0x9d40</td>
<td>2,000,003</td>
<td>[Outside any known module]</td>
</tr>
<tr>
<td>memcmp_sse4_1</td>
<td>0x16e3f0</td>
<td>2,000,003</td>
<td>ld-2.23.so</td>
</tr>
<tr>
<td>_dl_fixup</td>
<td>0xf9f0</td>
<td>2,000,003</td>
<td>do_lookup_x</td>
</tr>
<tr>
<td>dl-runtime.c</td>
<td>0x9e70</td>
<td>2,000,003</td>
<td>do_ocall(int (<em>)(void const</em>), void*)</td>
</tr>
<tr>
<td>__GI___pthread_mutex_lock</td>
<td>0x9091</td>
<td>2,000,003</td>
<td>[Outside any known module]</td>
</tr>
<tr>
<td>enclave_entry</td>
<td>0x9a0</td>
<td>2,000,003</td>
<td>[Unknown]</td>
</tr>
<tr>
<td>enter_enclave</td>
<td>0x7fe0</td>
<td>2,000,003</td>
<td>[Unknown]</td>
</tr>
<tr>
<td>save_and_clean_xfeature_regs</td>
<td>0x3c20</td>
<td>2,000,003</td>
<td>sgx_thread_mutex_lock</td>
</tr>
<tr>
<td>sgx_get_backing</td>
<td>0x3c20</td>
<td>2,000,003</td>
<td>sgx_get_backing</td>
</tr>
<tr>
<td>sgx_thread_mutex_lock</td>
<td>0xcce0</td>
<td>2,000,003</td>
<td>sgx_thread_mutex_lock</td>
</tr>
</tbody>
</table>

$ amplxe: Executing actions 100 % done
$
$ amplxe-cl -report hotspots -group-by module
amplxe: Using result path '/home/user/projects/linux-sgx/SampleCode/SampleEnclave/r000sgxhs'
amplxe: Executing actions 75 % Generating a report

<table>
<thead>
<tr>
<th>Module</th>
<th>Precise Clockticks</th>
<th>Module Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmlinux</td>
<td>128,000,192</td>
<td>vmlinux</td>
</tr>
<tr>
<td>libsgx_urts.so</td>
<td>48,000,072</td>
<td>/usr/lib/libsgx_urts.so</td>
</tr>
<tr>
<td>enclave.signed.so</td>
<td>30,000,045</td>
<td>/home/user/projects/linux-sgx/SampleCode/SampleEnclave/enclave.signed.so</td>
</tr>
<tr>
<td>ld-2.23.so</td>
<td>4,000,006</td>
<td>/lib/x86_64-linux-gnu/ld-2.23.so</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>2,000,003</td>
<td>[Unknown]</td>
</tr>
<tr>
<td>isgx</td>
<td>2,000,003</td>
<td>isgx</td>
</tr>
<tr>
<td>libc-2.23.so</td>
<td>2,000,003</td>
<td>/lib/x86_64-linux-gnu/libc-2.23.so</td>
</tr>
<tr>
<td>libpthread-2.23.so</td>
<td>2,000,003</td>
<td>/lib/x86_64-linux-gnu/libpthread-2.23.so</td>
</tr>
</tbody>
</table>

amplxe: Executing actions 100 % done

Figure 4. VTune report group-by-module messages

To see call sequences (stacks) measured during the collection, include top-down with the
-report command as follows:

    amplxe-cl -report top-down

Figure 5 shows the messages for this report.
$ amplxe-cl -report top-down
amplxe: Using result path `/home/user/projects/linux-sgx/SampleCode/SampleEnclave/r000sgxhs'
amplxe: Executing actions 75 % Generating a report

<table>
<thead>
<tr>
<th>Function Stack</th>
<th>Precise Clockticks:Total</th>
<th>Precise Clockticks:Self</th>
<th>Module</th>
<th>Function (Full)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Full)</td>
<td>Source File</td>
<td>Start Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
<td>-------------------------</td>
<td>---------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Unknown]</td>
<td>[Unknown]</td>
<td></td>
<td>[Unknown]</td>
<td>[Unknown]</td>
</tr>
<tr>
<td>[vmlinux]</td>
<td></td>
<td></td>
<td>vmlinux</td>
<td>[vmlinux]</td>
</tr>
<tr>
<td>[Unknown]</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>__morestack</td>
<td></td>
<td></td>
<td>18.349%</td>
<td>__morestack</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0x6200</td>
<td></td>
<td>40,000,060</td>
<td>libsgx_urts.so</td>
</tr>
<tr>
<td>do_ocall</td>
<td>9.174%</td>
<td></td>
<td>20,000,030</td>
<td>enclave.signed.so do_ocall</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0x9139</td>
<td></td>
<td>4,000,006</td>
<td>libsgx_urts.so</td>
</tr>
<tr>
<td>CTrustThreadPool::get_bound_thread</td>
<td>1.835%</td>
<td></td>
<td>4,000,006</td>
<td>enclave.signed.so</td>
</tr>
<tr>
<td>save_and_clean_xfeatureRegs [Outside any known module]</td>
<td>0.917%</td>
<td>2,000,003</td>
<td>[Unknown] [Outside any known module]</td>
<td></td>
</tr>
<tr>
<td>__GI__pthread_mutex_lock [Unknown]</td>
<td>0x91d0</td>
<td>4,000,006</td>
<td>enclave.signed.so</td>
<td></td>
</tr>
<tr>
<td>save_and_clean_xfeatureRegs [Unknown]</td>
<td>0x9890</td>
<td>2,000,003</td>
<td>[Unknown]</td>
<td></td>
</tr>
<tr>
<td>__morestack [Unknown]</td>
<td>0x6200</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

amplxe: Executing actions 100 % done

$ amplxe: Using result path '/home/user/projects/linux-sgx/SampleCode/SampleEnclave/r000sgxhs'
amplxe: Executing actions 75 % Generating a report

<table>
<thead>
<tr>
<th>Function Stack</th>
<th>Precise Clockticks:Total</th>
<th>Precise Clockticks:Self</th>
<th>Module</th>
<th>Function (Full)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Full)</td>
<td>Source File</td>
<td>Start Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
<td>-------------------------</td>
<td>---------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[vmlinux]</td>
<td></td>
<td></td>
<td>vmlinux</td>
<td>[vmlinux]</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0</td>
<td></td>
<td>128,000,192</td>
<td>vmlinux</td>
</tr>
<tr>
<td>__morestack</td>
<td>18.349%</td>
<td></td>
<td>40,000,060</td>
<td>libsgx_urts.so</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0x6200</td>
<td></td>
<td>2,000,003</td>
<td>enclave.signed.so __morestack</td>
</tr>
</tbody>
</table>

$ amplxe: Using result path '/home/user/projects/linux-sgx/SampleCode/SampleEnclave/r000sgxhs'
amplxe: Executing actions 75 % Generating a report

<table>
<thead>
<tr>
<th>Function Stack</th>
<th>Precise Clockticks:Total</th>
<th>Precise Clockticks:Self</th>
<th>Module</th>
<th>Function (Full)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Full)</td>
<td>Source File</td>
<td>Start Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
<td>-------------------------</td>
<td>---------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[vmlinux]</td>
<td></td>
<td></td>
<td>vmlinux</td>
<td>[vmlinux]</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0</td>
<td></td>
<td>128,000,192</td>
<td>vmlinux</td>
</tr>
<tr>
<td>__morestack</td>
<td>18.349%</td>
<td></td>
<td>40,000,060</td>
<td>libsgx_urts.so</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0x6200</td>
<td></td>
<td>2,000,003</td>
<td>enclave.signed.so __morestack</td>
</tr>
</tbody>
</table>

Figure 5. VTune report top-down messages
An additional report reports on callstacks, shows a bottom-up view of performance data. Figure 6 shows messages for this view for SampleEnclave. To get this data, use -report callstacks in the command, as follows:

amplxe-cl -report callstacks

$ amplxe-cl -report callstacks
amplxe: Using result path '/home/user/projects/linux-sgx/SampleCode/SampleEnclave/r000sgxhs'
amplxe: Executing actions 75 % Generating a report

<table>
<thead>
<tr>
<th>Function</th>
<th>Source File</th>
<th>Start Address</th>
<th>Function Stack</th>
<th>Precise Clockticks:Total</th>
<th>Precise Clockticks:Self</th>
<th>Module</th>
<th>Function (Full)</th>
</tr>
</thead>
<tbody>
<tr>
<td>----------</td>
<td>-------------</td>
<td>---------------</td>
<td>----------------</td>
<td>--------------------------</td>
<td>-------------------------</td>
<td>---------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>[Unknown]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[vmlinux]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>vmlinux</td>
<td>[vmlinux]</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128,000,192</td>
<td>vmlinux</td>
</tr>
<tr>
<td>__morestack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40,000,060</td>
<td>libsgx_urts.so</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0x6200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2,000,003</td>
<td>enclave.signed.so __morestack</td>
</tr>
</tbody>
</table>

Figure 5. VTune report top-down messages
An additional report reports on callstacks, shows a bottom-up view of performance data. Figure 6 shows messages for this view for SampleEnclave. To get this data, use -report callstacks in the command, as follows:

amplxe-cl -report callstacks

$ amplxe-cl -report callstacks
amplxe: Using result path '/home/user/projects/linux-sgx/SampleCode/SampleEnclave/r000sgxhs'
amplxe: Executing actions 75 % Generating a report

<table>
<thead>
<tr>
<th>Function</th>
<th>Source File</th>
<th>Start Address</th>
<th>Function Stack</th>
<th>Precise Clockticks:Total</th>
<th>Precise Clockticks:Self</th>
<th>Module</th>
<th>Function (Full)</th>
</tr>
</thead>
<tbody>
<tr>
<td>----------</td>
<td>-------------</td>
<td>---------------</td>
<td>----------------</td>
<td>--------------------------</td>
<td>-------------------------</td>
<td>---------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>[Unknown]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[vmlinux]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>vmlinux</td>
<td>[vmlinux]</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128,000,192</td>
<td>vmlinux</td>
</tr>
<tr>
<td>__morestack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40,000,060</td>
<td>libsgx_urts.so</td>
</tr>
<tr>
<td>[Unknown]</td>
<td>0x6200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2,000,003</td>
<td>enclave.signed.so __morestack</td>
</tr>
</tbody>
</table>
VTune can report the number of hardware events for an enclave performance collection. Figure 7 shows the command to report the number of HW events for SampleEnclave. The command to generate this report is as follows:

```
amplxe-cl -report hw-events
```
There are two parameters (called *knobs* by VTune) that are supported with `-collect sgx-hotspots`. They are:

- **sampling-interval**: The default sampling interval is 1, which designates 1 millisecond. You can vary the sampling rate between 0.01 and 100 milliseconds using this knob. A sample command line is as follows:
  ```bash
  amplxe-cl -collect sgx-hotspots -knob sampling-interval=2 ./app
  ```

- **enable-user-tasks=true**: This knob allows you to analyze tasks, events, and counters specified in your application via the Task API. This option results in higher overhead and increases the result size. The default for this option is false. A sample command line is as follows:
  ```bash
  amplxe-cl -collect sgx-hotspots -knob enable-user-task=true ./app
  ```
Detailed information on using the VTune Task API are provided here: https://software.intel.com/en-us/vtune-amplifier-help-task-api.

Attaching VTune to a running application/process

Another method for profiling applications is to attach to an already running application/process. Intel SGX Applications are profiled by invoking the VTune Instrumentation and Tracing Technology (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.

Assuming that the SDK SampleEnclave application has already been launched in one terminal window, open an additional terminal window, navigate to the working directory for the application, and execute the basic form of the VTune command listed below:

```
amplxe-cl -collect sgx-hotspots -knob sampling-interval=2 --target-process=app
```

VTune attaches to the running application and gathers performance data. When the application exits, VTune writes the data it has collected to the screen and a results file. Figure 10 shows messages for the SampleEnclave application. Figure 11 shows the VTune command and summary messages for the SampleEnclave application. You can also see the VTune messages by using the -report summary command.

```
projects/linux-sgx/SampleCode/SampleEnclave$ ./app
Checksum(0x0x7fffd90ff5c0, 100) = 0xfffd4143
Info: executing thread synchronization, please wait...
Info: SampleEnclave successfully returned.
Enter a character before exit ...
a
~/projects/linux-sgx/SampleCode/SampleEnclave$
```

**Figure 10. Launching SampleEnclave application**

```
~/projects/linux-sgx/SampleCode/SampleEnclave$ amplxe-cl -collect sgx-hotspots -knob sampling-interval=2 --target-process=app
amplxe: Collection started. To stop the collection, either press CTRL-C or enter from another console window: amplxe-cl -r /home/user/projects/linux-sgx/SampleCode/SampleEnclave/r007sgxhs --command stop.
amplxe: Warning: To enable hardware event-base sampling, VTune Amplifier has disabled the NMI watchdog timer. The watchdog timer will be re-enabled after collection completes.
amplxe: Collection detached.
amplxe: Collection stopped.
amplxe: Using result path '/home/user/projects/linux-sgx/SampleCode/SampleEnclave/r007sgxhs'
amplxe: Executing actions 19 % Resolving information for 'vmlinux'
amplxe: Warning: %CantOpenFile
amplxe: Executing actions 75 % Generating a report
Collection and Platform Info
--------------------------------------
Parameter r007sgxhs
```

---

**Figure 11. VTune command and summary messages**
Figure 11. Attaching VTune to running application and summary messages

You can also attach VTune to a running process by changing --target-process to --target-pid and supplying the process ID name, as follows:

```
amplxe-cl -collect sgx-hotspots -knob sampling-interval=2 --target-pid=<processID>
```

Additional 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.
Intel® Software Guard Extensions (Intel® SGX)

Summary

Intel VTune Amplifier XE for Linux provides the capabilities to profile the performance of Intel SGX enabled applications. This paper gets you started by explaining how to verify the VTune settings, collect performance data, and run collection reports.

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 processes/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

Infomation 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