

Intel® Software Guard Extensions (Intel® SGX)

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

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 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:

- 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.

File	Edit	View	Project	Build	Debug	Team	Tools	Test	Analyze	Window	Help	
i G	• 🕤	ੈੈ •	😩 💾 🖥	2	- 66	Debug	• x54		-	Intel(R) SG	X Debugger 🔻	🏓 🖕 🤅 🌆 👻 (
I Se L	a luation				- " ~	Now	molifier D	ocult	J X Toop	Evoloror (Connect	Class View

Figure 1. Invoking VTune from Visual Studio toolbar





Intel® Software Guard Extensions (Intel® SGX)

3. 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.

SampleEnclave - Microsoft Visual Studio (Administrator) File Edit View Project Build Debug Team Tools Test Analyzz Image: Solution of the state of t	Window Help Attach ▼ #
New Amplifier Result * X Choose Target and Analysis Type Analysis Target A nalysis Type Accessible targets Specify and configure your analys Imat Xeen Phi coprocessor (notive) Inhel Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Inhel Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Inhel Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Inhel Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocessor (notive) Imat Xeen Phi coprocese	Image:

Figure 2. VTune Analysis Target tab

4. 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.





Choose Target and Analys	is Type	INTEL VTUNE AMPLIFIER XE 20
Analysis Target A Analysis Type Agentian Analysis Agentian Analysis	SGX Hotspots Copy Analyze flotspots inside security enclaves for systems with the Intel Software Guard Extensions (Intel SGX) feature enabled. Lead more (F1) CPU sampling interval, ms: 1 CPU sampling interval, ms: 1 Analyze user tasks, events, and counters © Details Image: Provide the security of the	Choose Target
		Command Line

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.





BOX HOISPOIS HOISPO	ots viewpoint (<u>chan</u>	<u>ge</u>) 🥝			INTEL VIUNE AMPLIFIER XE
Θ Analysis Target 🔺 Analysis	s Type 🔛 Collection Log	g 🔋 Summary 🚳 Bottom-	p 🗞 Caller/Callee 🗞 Top-down Tree 🖭 P	Platform	
S Elapsed Time	: 6.870s				
Precise Clockticks [®]	7.244.010.866				
Total Thread Count:	15				
Paused Time [®] :	0s				
	ā				
This section lists the most	extive functions in your	application. Optimizing thes	hotspot functions typically results in improvir	ng overall application performance.	
Function	Module	Precise Clockticks [®]			
printf	Enclave.signed.dll	6,724,010,086			
[Outside any known modu	ile]	74,000,111			
func@0x1005ab70	sgx_urtsd.dll	66,000,099			
Thread32Next	kernel32.dll	34,000,051			
do_ocall	Enclave.signed.dll	18,000,027			
[Others]		328,000,492			
 Collection and 	Platform Info				
This section provides info	rmation about this colled	ction, including result set size	and collection platform data.		
Application Comman Environment Variable	d Line: C:\Program File s:	s (x86)\Intel\IntelSGXSDK\si	:\SampleEnclave\Debug\App.exe		
Operating System:	Microsoft Windo	ows 10			
Computer Name:	DESKTOP-39N	AU1M			
Result Size:	6 MB				
Collection start time:	21:38:40 04/10/	2017 UTC			
Collection stop time:	21:38:47 04/10/	2017 UTC			
					•
tput					

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.





Choose Target and	Description: My comments	^			INTEL VTUNE AMPLIFIER XE 2
Analysis Target A Analys	Properties	×	Г	Сору	O Start
Algorithm Analysis	Events and (as CDL late/D) Decrease and assess Club	ALL UT	L		
A Advanced Hotspots	Events conligured for CPU: intel(k) Processor code named skyl				Start Paused
A Concurrency A Locks and Walts Compute-Intensive Application A HPC Performance Charaction	NOTE: For analysis purposes, Intel Viune Amplitier XE 2017 ma on the value of the Duration time estimate option specified in t	y adjust the Sample Atter values in the table below by a multiplier. The multiplier depends the target configuration window.			Choose Target
Car Microarchitecture Analysis	Event Name Sample After Event Description	Add Events X			
A General Exploration		Available events:			
A TSX Exploration		Hiter (use * and ?, e.g. br*retired) : Clear Filter			
A SGX Hotspots		Event Name Event Description	the value of the	_	
Platform Analysis A CPU/GPU Concurrency		BR_MISP_RETIRED.NEAR_TAKEN Number of near bran	and runde of the		
A GPU Hotspots		BR_MISP_RETIRED.NEAR_TAKEN_PS Number of near brai		~	
Custom Analysis		CPU_CLK_THREAD_UNHALTED.ONE_THREAD_ACTIVE Core crystal clock cy			
- A SGX Hotspots 0	Analyze I/O waits	CPU_CLK_THREAD_UNHALTED.REF_XCLK Core crystal clock cy			
	Collect I/O ABI data	CPU_CLK_UNHALTED.ONE_THREAD_ACTIVE Core crystal clock cy			
		CPU_CLK_UNHALTED.REF_TSC Reference cycles whe			
	Contractors to be the second s	CPU_CLK_UNHALTED.REF_XCLK Core crystal clock cy			
	Stack size, in bytes:	CPU_CLK_UNHALTED.REF_XCLK_ANY Core crystal clock cy 🗸			
	Stack type:	k			
	L Estimate call counts	Check All Check None Explain			
	Estimate trip counts				
	Chipset events:	OK Cancel			
	Analyze memory bandwidth				
	Analyze PCIe bandwidth				
	Analyze user tasks, events, and counters				
	Analyze system-wide context switches				
	Capture transactional cycles				
	Collect precise clockticks	*			
	Command line name: sgx-hotspots-1	Analysis identifier: sgxhs1	a de la composición de la comp	in new Year	A Command Line
itout		Help OK Cancel	God	e Settines	to activate Windows 👻 🎙

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.





Analysis Target 🗛 Analysis Type 🔛	Collection Log	Sottom-up Sottom-up	lier/Callee 💁 Top-down Tree 🛋 Pi	lattorm			
Grouping: Function / Call Stack							<u>~</u> [%]Q["
Function / Call Stack	Precise Clockticks V	Module	Function (Full)	Source File	Start Address		
▶ printf	6,724,010,086	Enclave.signed.dll	printf	enclave.cpp	0x100329a0		
[Outside any known module]	74,000,111		[Outside any known module]		0		
func@0x1005ab70	66,000,099	sgx_urtsd.dll	func@0x1005ab70		0x1005ab70		
Thread32Next	34,000,051	kernel32.dll	Thread32Next		0x6b84fb30		
⊳ do_ocall	18,000,027	Enclave.signed.dll	do_ocall	trts_pic.asm	0x100387a7		
SwapContext	16,000,024	ntoskrnl.exe	SwapContext		0x140171670		
 MiDispatchFault 	10,000,015	ntoskrnl.exe	MiDispatchFault		0x14008e260		
 KiFlushRangeTb 	10,000,015	ntoskrnl.exe	KiFlushRangeTb		0x140132e50		
MiDeletePteRun	8,000,012	ntoskrnl.exe	MiDeletePteRun		0x1400951f0		
MilnsertPageInList	8,000,012	ntoskrnl.exe	MilnsertPageInList		0x140097790		
sgx_spin_lock	6,000,009	Enclave.signed.dll	sgx_spin_lock	spinlock.c	0x1000a2d0		
ExAcquirePushLockSharedEx	6,000,009	ntoskrnl.exe	ExAcquirePushLockSharedEx		0x1400635e0		
 MiAllocateWsle 	6,000,009	ntoskrnl.exe	MiAllocateWsle		0x140092d50		
<	> <						>
್ಷ∾ Q+ ೧–೧+ 0.5s	1s 1.5s 2s	2.5s 3s	3.5s 4s 4.5s	5s	5.5s 6s	6.5s 🗸	Thread
Thread (TID: 372)						^	🗹 🔲 Running
Thread (TID: 3684)		л					Precise (
Thread (TID: 7788)		- T					CPU Time
Thread (TID: 5292)		- E					Precise Clo
Thread (TID: 5884)		- C					
Thread (TID: 5320)		1					
Thread (TID: 7040)		1					
COLLT:						~	
CPU Time							
					1.0		
FILTER 100.0% 🙀 Any F	Process ~ Thread Any	y Thread V Modu	e Any Module ~ User function	ons + 1	 Show inline func 	tions ~ Function	s only ~
							▼ # .

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.





(7)	1sgxhs r002ths	r003sgxhs	r004sgxhs X	_		
SGX Hotspots Hotspots viewpoint (<u>change</u>)	Ø					INTEL VIUNE AMPLIFIER XE 2017
🕘 💮 Analysis Target 🔺 Analysis Type 🔛 Collection Lo	ig 🗂 Summary 😪 Bo	ttom-up 🛛 😪 Caller	/Callee 🛛 🖓 Top-down Tree 📧	Platform		Þ.
Grouping: Function / Call Stack						~ 🔨 Q 🖓
Function / Call Stack	Precise Clockticks V	Module	Function (Full)	Source File	Start Address	
▶ sgx_thread_mutex_lock	200,000	sample_enclav	sgx_thread_mutex_lock	sethread	0x10017db4	
▶ relocate_enclave	100,000	sample_enclav	relocate_enclave	relocation	0x1002a5b7	
▶ do_ocall	100,000	sample_enclav	do_ocall	trts_pic.asm	0x10029547	
restore_xfeature_regs	100,000	sample_enclav	restore_xfeature_regs	trts_xsave	0x10001b3d	
enclave_entry	100,000	sample_enclav	enclave_entry	trts_pic.asm	0x1002a037	
▶intel_memcpy	100,000	sample_enclav	intel_memcpy		0x100063ac	
< > 	< 2s 2.5s 3s 3	.5s 4s 4.5s	5s [0.4%] ucrtbased.dll	^ 7.5s	8.13s 8.5s 9s	9.5s 10s 10.5s 11s V Thread V
Thread (TID: 4756)			[0.4 /o] (pertecut			
Thread (TID: 4756)			[0.4%] pereval [0.4%] combase.dll [0.3%] sample_enclave.s	igned.dll		A Construction of the second s
Thread (TID: 4756)			[0.4%] potestin [0.4%] combase.dll [0.3%] sample_enclave.s [0.2%] msvcrt.dll [0.1%] fileinf0.svs	igned.dll		 ✓ ■ Running ✓ ₩uts Precise Cloc ○ ♥ Hardware Even
Thread (TID: 4756)			[0.4%] pottstall [0.4%] combase.dll [0.3%] sample_enclave.s [0.2%] msvcrt.dll [0.1%] fileinfo.sys [0.1%] igdkmd64.	igned.dll module to filter dat	ta by its contribution.	 ✓ ■ Running ✓ ■ Running ✓ ■ Precise Cloc ○ ♥ Hardware Even. ✓ CPU Time
Thread (11D: 4756)			0.4%) combase dil (0.4%) combase dil (0.3%) sample_enclave si (0.2%) msvort dil (0.1%) fileinto.sys [0.1%) wdflor.sys (0.1%) wdflor.sys	igned.dll nodule to filter dat	ta by its contribution.	
Thread (TID: 4756)			0.4%) combase dil (0.4%) combase dil (0.3%) sample_enclave si (0.2%) msvct dil (0.1%) fileinto.sys (0.1%) igdkmd64. (0.1%) wdfilter.sys (0.1%) digmms2.sys	lgned.dll nodule to filter dat	ta by its contribution.	
Thread (TID: 4756)			0.4%) pervala 0.5%) sample_endaws 0.2%) sample_endaws 0.1%) fileinto.sys 0.1%) idento.sys 0.1%) wdfiter.sys 0.1%) wdfiter.sys 0.1%) idsp.sys 0.1%) idsp.sys 0.1%) idsp.sys	igned.dll	ta by its contribution.	
Thread (11D: 4756)			10 461 period 10 253 sample endower 10 253 sample endower 10 253 sample endower 10 1% file info.sys [Select: 10 1%] udd1000.sys 10 1%] udd1000.sys 10 1%] udd1000.sys 10 1%] udd1000.sys 10 1%] ksr.sys 10 1%] ksr.sys 10 1%] kernel32.dll 10 1%] kernel32.dll	igned.dll module to filter dat	ta by its contribution.	
Thread (TID: 4756)			0.447 (purchase dll 0.547 (particle critical and critical	igned.dll module to filter dat	ta by its contribution.	
Thread (TID: 4756)			(0.4%) combase dll (0.4%) combase dll (0.4%) feation syn (0.4%) feation syn (0.4%) fedimodal (<u>Sect.</u>) (0.4%) wdfiller syn (0.4%)	igned dil nodule to filter dat	ta by its contribution.	
Thread (TID: 4756)			10.4% Combase dil 10.4% Combase dil 10.5% Estippia mori di anti 10.7% filipino si gli comba 10.7% utili comba di anti 10.7%	igned.dll	ta by its contribution.	Activate Windows
CPU Time			10 491 (Joines all (10, 10, 10, 10, 10, 10, 10, 10, 10, 10,	igned.dll	ta by its contribution.	
CPU Time			(0.4%) combase dll (0.4%) combase dll (0.4%) feinfo syn (0.4%) feinfo syn (0.4%) feinfo syn (0.4%) feinfo syn (0.4%) wdllter syn (0.4%) wdllter syn (0.4%) wdllter syn (0.4%) wdllter syn (0.4%) word-4wn dll (0.4%) word-4wn dll (0	igned.dll	ta by its contribution.	Activate Windows Go to Settings to activate Windows.

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.





	Custom Analysi	s				×		
fier Result 🕞 🗙 👔	Analysis name: S	GX Hotspots 1						INTEL UTIME AND USED
se Target and	Description: N	ly comments				0		INTEL VIUNE AMPLIFIEK
thm Analysis	Properties						Сору	Start
ic Hotspots	Events configu	red for CPU: Intel(R) Proce	ssor code named Skylake ULT			^		O Charle Dannel
vanced Hotspots ncurrency ks and Walts ute-Intensive Application C Performance Charaction	NOTE: For ana on the value o	lysis purposes, Intel VTune f the Duration time estima	Amplifier XE 2017 may adjust the te option specified in the target co	Sample After values in the table below I nfiguration window.	by a multiplier. The multip	lier depends		Choose Target
architecture Analysis	Event Name	Sample After Event Descr	iption					
neral Exploration mory Access	BR_MISP_RET	100007 Number of ne 2503 Core costal d	ar bran					
K Exploration	HLE_RETIRED	2000003 Number of tim	ies an F					
X Hotspots X Hotspots	HLE_RETIRED	2000003 Number of tim	ies an F				al a state	
rm Analysis						the	value of the	
U Hotspots							~	
k Input and Output (pr					Add Event	Remove Event		
X Hotspots 0								
	Analyze I/O	waits						
	Collect I/O API	data:	No ~					
	Collect stack	(S						
	Stack size, in by	/tes:	0					
	Stack type:		LBR ~					
	Estimate cal	counts						
	Estimate trip	counts					6.0	
	Chipset events:					based on hardware events.	or cail counts	
	Analyze me	mory bandwidth						
	Analyze PCI	e bandwidth						
	Analyze use	r tasks, events, and counte	rs					
	Analyze syst	tem-wide context switches						
	Capture tran	nsactional cycles						
	Collect prec	ise clockticks				~		
	Command line na	me: sgx-hotspots-1		Analysis identifier: sgxhs1			Activato M	Command Line
					Help	OK Cancel	Activate w	ta astirata Mindaura 🔻

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.





						~ % Q
Function / Call Stack	Precise Clockticks V	Module	Function (Full)	Source File	Start Address	
[Outside any loop]	140,000,210		[Outside any loop]		0	
[Loop@0x1c0001420 in func@0x1c000133f]	40,000,060	hal.dll	[Loop@0x1c0001420 in func		0x1c0001420	
[Loop@0x1400a4344 in func@0x1400a4238]	6,000,009	ntoskrnl.exe	[Loop@0x1400a4344 in func		0x1400a4344	
[Loop@0x1400bfdb0 in func@0x1400bfbd0]	6,000,009	ntoskml.exe	[Loop@0x1400bfdb0 in func		0x1400bfdb0	
[Loop@0x140090366 in func@0x14008fbb0]	4,000,006	ntoskml.exe	[Loop@0x140090366 in func		0x140090366	
Loop@0x1c00f7137 in DestroyThreadsTimers]	2,000,003	win32kfull.sys	[Loop@0x1c00f7137 in Destr		0x1c00f7137	
[Loop@0x1007e63f in func@0x1007e5af]	2,000,003	combase.dll	[Loop@0x1007e63f in func@		0x1007e63f	
Loop@0x10028195 in initterm]	2,000,003	ucrtbase.dll	[Loop@0x10028195 in initterm]		0x10028195	
Loop@0x4b2c9688 in func@0x4b2c9380]	2,000,003	ntdll.dll	[Loop@0x4b2c9688 in func		0x4b2c9688	
Loop@0x4b2c9616 in func@0x4b2c9380]	2,000,003	ntdll.dll	[Loop@0x4b2c9616 in func		0x4b2c9616	
Loop@0x4b2c3993 in func@0x4b2c3870]	2,000,003	ntdll.dll	[Loop@0x4b2c3993 in func		0x4b2c3993	
Loop@0x1400c26d1 in RtllnitUnicodeString]	2,000,003	ntoskml.exe	[Loop@0x1400c26d1 in Rtllni		0x1400c26d1	
Loop@0x1c00ea7e2 in func@0x1c00ea780]	2,000,003	ntfs.sys	[Loop@0x1c00ea7e2 in func		0x1c00ea7e2	
		at a stand server	[] (2014014-590 in man		0+14014-590	
[Loop@0x14014e580 in memset]	2,000,003	ntoskmi.exe	Loopgux 140 14e300 in mem		00140146560	
[Loop@0x14014e580 in memset]	2,000,003	ntoskmi.exe	Loopgux 140 146500 in mem		2.55	→ Thread
Loop@0x14014e580 in memset] ~~~~~~~+ Thread (TD: 3148) Thread (TD: 6124)	2,000,003	ntoskmi.exe	Loopgux 140 146500 in mem		2.55	Thread
دمەر@0x14014e580 in memset]	2,000,003	ntoskmi.exe	Loopgux 140 144500 in mem		2.55	Thread
د در ۲۰۰۵ د. ۲۰۰۵ در ۲۰۰۵ د. ۲۰۰۰ د. ۲۰۰۵ د.	2,000,003	ntoskmi.exe	Loopgux 140 146500 in mem		2.55	
Loop@0x14014e580 in memset]	2,000,003	i i i i	Loopgux 140 146500 in mem		2.55	



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_LIBITTNOTIFY32 = <VTune Installation Dir>\bin32\runtime\ittnotify_collector.dll
64bit:
INTEL_LIBITTNOTIFY64 = <VTune Installation Dir>\bin64\runtime\ittnotify_collector.dll
```

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





Jocal Jocal Jocal Termote Linux (SSH) Intel Xeon Phi coprocessor (native Intel Xeon Phi coprocessor (host i vrbitrary Targets Jocal Jocal	Attach to Process reperformance data will be collected after attaching to the process. Press F1 for more details.					
	st la Launch Application rocess with the specified name. Provide a valid process name.					
	Automatically stop collection after (sec):					
Intel Xeon Phi coprocessor (nat	(ve) ⊙ Advanced					
Intel Xeon Phi coprocessor (ho	ost a ✓ Analyze child processes					
		Modify				
	Duration time estimate: Between 1 and 15 minutes V					
	Analyze system-wide					
	Limit collected data by:					
	• Result size from collection start, MB 500					
	O Time from collection end, sec 0 v					
	CPU mask:					
	Custom collector:	Modify				
	Store result in the project directory: C:\Program Files (x86)\Intel\Intel\SGXSDK\src\RemoteAttestation\isv.app\My Amplifier XE Results - isv.app					
	O Store result in (and create link) file to) another directory					
	Stole result in (and cleate link file to) another directory					
	C\Program Files (x86)\Intel\IntelSGXSD(\src\RemoteAttestation\isv_app\My Amplifier XE Results - isv_app	Browse				

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.

WHITE PAPER



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