## Optimize Performance with Intel®VTune<sup>TM</sup> Profiler

Heinrich Bockhorst

HLRS, September 15<sup>th</sup>



## Agenda

- VTune overview + VTune hotspots
- How to use VTune e.g., in a Cluster environment
- Demo
- VTune HPC analysis
- VTune GPU analysis
- Demo with GROMACS

## Playbook for easy access to command lines

VTune_Playbook.txt • +	—		×				
File Edit View			ŝ				
VTune Playbook with templates for Intel tools usage							
The Playbook contains command lines starting with \$ Please change \$PRG, \$ARG into the path,name and parameters of your program!							
Please send feedback/questions to heinrich.bockhorst@intel.com							
0. Environment							
load environment by:							
<pre>\$ source <path oneapi="" to="">/setvars.sh</path></pre>							
Note: on Devcloud the environment is already loaded!							
default							
<pre>\$ source /opt/intel/oneapi/setvars.sh</pre>							
or load the local compiler and VTune modules							
Ln 29, Col 1 100% Windows (CRLF)	UTF-	8					

## Demo slides/life

- Presentations contain sample snapshots
- Demo content might not match exactly these snapshots

#### Analysis based on APS results

- APS is first step in analysis.
- Check for hints provided in APS results
- APS provides only data on whole app
- VTune will provide function/loops/source code/assembly level analysis

#### Optimize Performance Intel® VTune™ Profiler

#### Get the Right Data to Find Bottlenecks

- A suite of profiling for CPU, GPU, FPGA, threading, memory, cache, storage, offload, power...
- Application or system-wide analysis
- DPC++, C, C++, Fortran, Python\*, Go\*, Java\*, or a mix
- Linux, Windows, FreeBSD, Android, Yocto and more
- Containers and VMs

#### Analyze Data Faster

- Collect data HW/SW sampling and tracing w/o recompilation
- See results on your source, in architecture diagrams, as a histogram, on a timeline...
- Filter and organize data to find answers

#### Work Your Way

- User interface or command line
- Profile locally and remotely
- GUI (desktop or web) or command line



6

#### Find Answers Fast Intel<sup>®</sup> VTune<sup>™</sup> Profiler



Analysis Configuration Collection Log Summary Bottom-up Caller/Callee Top-down Tree Platform ▼ ≪ Grouping: Function / Call Stack CPU Time Overhead Time Function / Call Stack Effective Time by Utilization **v** Spin Time Idle Poor Ok Ideal Over Creation Scheduling Reduction Atomics grid intersect 4.0879 sphere intersect 3.748s 0s 0s 0s 0s 0s 3.748s 0s 0s grid intersect 0s 0s 0s intersect objects 3.580s 0s0s 0s0s 0s 0.168s 📙 ▶ < grid intersect.</p> 2.021s 0s 0s 0s 0s func@0x69e19df0 2.467s 0s 0s 0s 0s 0s 8.2s 8.4s 8.6s 8.8s 9s 9.4s 9.6s 9.8s 10s 10.2s 10.4s10.6s 0: /9.2s Thread mandamandamandamanda Running WinMainCRTStartup (TID: 2. LCPU Time OMP Master Thread #0 (TI) Spin and Overhe. OMP Worker Thread #2 (TI CPU Sample CPU Utilization 📥 CPU Time CPU Utilization Spin and Overhe FILTER 100.0% Any Pr Any Thread < Any Modu 🔻 Any U 🔻 User functi V Function • Show inli • Filter by Process Tuning Opportunities Shown in Pink. & Other Controls Hover for Tips

Hotspots Hotspots by CPU Utilization • ?

C

v

## See Profile Data On Source / Asm

Double Click from Grid or Timeline

View Source /	Asm or both	CPU Time	Right	click	for inst	ruction	refere	nce manua
	Source Assembly	👬 🏕 🔐 Assembly gi	rouping: Address				Q	]
5	Source 64 tmax.x +- toeita.x; 65 curpos = nXp; 66 rVp n += relater no	CPU Ti	me: Total Addr 0x400 0x400 0x400	ess ▲ Sour cb9 570 cbf	Ass De UX4Udd49 < Block 44:	embly (BLOCK 53>	CPU Time: 1	
Quick Asm na	vigation:	0.004s	0x400 0x400 0x400	cc1 572 cc3	<u>z 0x40dd07 &lt;</u> 3lock 45:	<u>3lock 50&gt;</u>	0.0075	
Select source	to highlight Asr	0.265s	0x40d 0x40d	cc3 573 cc6 573	mov eax, dword mov ecx, dword	d ptr [esi+0x4] d ptr [edi+0x10]	0.053s 0.750s	
5 5	71 cur = g->cells[voxin 72 while (cur != NULL)	ndex]; { 0.007s	0x40d 0x40d	cc9 573 ccc 573	mov edx, dword mov eax, dword	d ptr [edi+0xc] d ptr [eax]	0.020s	
5	/3         if (ry->mbox[cur->           74         ry->mbox[cur->ob           75         cur->obi->method	<pre>&gt;obj-&gt;id] != r 2.058s oj-&gt;id] = ry-&gt; 0.604s ds-&gt;intersect( 0.687s</pre>	0x40d 0x40d 0x40d	cce 573 cd1 573 cd3	cmp dword ptr jz 0x40dce5 <1 Block 46:	[ecx+eax*4], edx <u>Block 48&gt;</u>	0.003s	
5	76 } 77 cur = cur->next;	0.423s	0x40d 0x40d	cd3 574 cd6 575	mov dword pt mov eax, dwoi	[ecx+eax*4], edx 1 ptr [esi+0x4]	0.604s	
5	<pre>78      } 79      curvox.z += step.z; 80      if (nur)moudiat &lt; tm</pre>	0.019s	0x40d 0x40d 0x40d	cd9 575 cda 575	push edi push eax	ntr [004+048]	0.005s 0.004s	
5	B1         break;           82         voxindex += step.z*q	g->xsize*g->ys 0.006s	0x400 0x40d 0x40d	cde 575 ce0 575	mov ecx, dwor mov eax, dwor call eax	ptr [ecx]	0.130s	
E				0	D1 1 47		0	

Scroll Bar "Heat Map" is an overview of hot spots

#### Click jump to scroll Asm

#### Bottom-up tab – most popular tab

- Grouping: different ordering of results –check out different choices
- Source view: double click on function or loop will open another window
   source must be compiled with "-g". Source code must be available
- Zoom and filter in timeline section. Grid will adapt your choice
- Filter process/user code/libraries/loops/system functions (e.g. libc)
- Values in pink: e.g. high overhead like barrier waiting or low cpu utilization.

#### How to start an analysis GUI vs CMD

- Start GUI by \$ vtune-gui on command line or by double clicking on Windows
- Click on "Configure Analysis ..."
- 3 Sections: WHERE, WHAT, HOW
- WHERE: local vs remote. We do local here
- WHAT : define your application with parameters and environment
- HOW: Analysis type like "Hotspots" or "APS" with additional parameters

Intel V	Tune Profiler		- D X
	P + 🗅 🕹	Welcome × Configure Analysis ×	
VI.	<b>⊸ 🖿</b> sam	Configure Analysis 🛍	INTEL VTUNE PROFILE
_	r000hs	WHERE	HOW
_	r001ue		
묩	r002ps		Holspols +
	r004so		Identify the most time consuming functions and drill down to see time spent on each line of source
$\triangleright$	r005so	WHAI	code. Focus optimization efforts on hot code for the greatest performance impact. <u>Learn more</u>
	r006so	E Launch Application 👻	User-Mode Sampling ⑦     Overhead
<b>U</b>	r007so		─ Hardware Event-Based Sampling ⑦
Г	r008so	Specify and configure your analysis target: an application or a script to execute. Follow <u>Prepare</u> Application for Analysis to compile your app for best analysis productivity.	
		Application:	
		C:\Users\hbockhor\OneDrive - Intel Corporation\Documents\Issues\2023\JIRA_VASP-2861	Show additional performance insights
		Application parameters:	
		ව ව	
			Collect CPU sampling data
		✓ Use application directory as working directory	With stacks -
		Advanced >	CPU sampling interval me
			10
			collect synchronize CODV command line here
			No
			Collect signalling API data
			No
<b>1</b> 50			
ŝ			
(?)			

## VTune and MPI

- To run VTune in an MPI job you may use the "-gtool" flag
- More convenient is the I\_MPI\_GTOOL environment variable. Example for HPC analysis:

\$ export I\_MPI\_GTOOL= "vtune -c hpc-performance -r HPC:0"

run your program, as usual, under MPI. The setting will collect data on rank #0. Use a list of ranks or :all for multi rank analysis.

#### More information:

https://www.intel.com/content/www/us/en/develop/documentation/mpi-developer-reference-linux/top/commandreference/mpiexec-hydra/gtool-options.html



🔽 Intel VI	une Profiler						– 🗆 X
	P.+ 🗅 🕹	Welcome × hs-1.sprpvc ×					
VT.	<b>▼</b> ■ sam	Hotspots @ 🛍				INTE	EL VTUNE PROFILER
	r000hs	Analysis Configuration Collection Log Summary	Bottom-up Caller/	Callee	Top-down Tree Flame Graph Platform		
	r001ue	Grouping: Source Eunction / Eunction / Call Stack				Call Stacks	ъ ÷
8	r002ps	Source Function / Function / Call Stack	CPU Time 🔻 🚿	Module	Function (Full)	08.9% (34.219c of 34.612c)	
E	r003so	kmp launch thread	230.066s	Module	kmp launch thread	libromacs mpi so 8   nbnxm kernel ElecEw Vdwl J E 2xm	/
	r004so	nbnxm_kernel_ElecEw_VdwLJ_F_2xmm	34.612s		nbnxm_kernel_ElecEw_VdwLJ_F_2xmm(NbnxnPairlistCpu const*, nb	libgromacs_mpi.so.8   nbnxn kernel_cpu+0x1ef	
	r005so	▶ fft5d_execute	28.240s		fft5d_execute(fft5d_plan_t*, int, gmx_wallcycle*)	libiomp5.so ! [OpenMP dispatcher]+0x132 - kmp_runtime.cpp;	:7845
	r006so	_INTERNAL1311483b::kmp_wait_template <kmp_flag_< p=""></kmp_flag_<>	19.337s		_INTERNAL1311483b::kmp_wait_template <kmp_flag_64<(bool)0,< th=""><th>libiomp5 so I [OpenMP fork]+0x1a2 - kmp h:350</th><th></th></kmp_flag_64<(bool)0,<>	libiomp5 so I [OpenMP fork]+0x1a2 - kmp h:350	
	r007so	spread_on_grid.extracted.13	9.454s		spread_on_grid(gmx_pme_t const*, PmeAtomComm*, pmegrids_t co	libaromacs mpi so 8   nonbonded verlet todispatchNonbonde	edKernel+0x20d
<b>C</b> -1	r008so	gmx::lincs_matrix_expand	8.531s		gmx::lincs_matrix_expand(gmx::Lincs const&, (anonymous namespac	libgromacs mpi so 8 / do nb verlet+0x20b	
			5 788s		any "lines undate atoms (amy "lines* int float amy "Array Ref <float (<="" th=""><th>libgromacs mpi so 8 / do_force+0x28a5</th><th></th></float>	libgromacs mpi so 8 / do_force+0x28a5	
		ather f bsplines	5.362s		gather f bsplines(gmx pme t const*, float const*, bool, PmeAtomCo	libgromacs_mpi.so.8 / gmy::/ eggcySimulator::do_md+0y3aa1	
		<ul> <li>DftiComputeBackward</li> </ul>	4.984s		DftiComputeBackward	libgromacs_mpi.so.8   gmv::Mdrupper::mdrupper+0v5377	
		▶ compute_1d_small_fwd	4.781s		compute_1d_small_fwd	amy mai amy: any marun±0x102e	
		▶ spread_on_grid	3.471s		spread_on_grid(gmx_pme_t const*, PmeAtomComm*, pmegrids_t co	gmx_mpi ! gmxgmx_mdrun+0x192e	
		copy_fftgrid_to_pmegrid	2.685s		copy_fftgrid_to_pmegrid(gmx_pme_t*, float const*, float*, int, int, int)	gmx_mpi ! gmxgmx_marun+0x46	
		compute_mg_row_bwd	2.578s		compute_mg_row_bwd	libgromacs_mpi.so.8 ! gmx::CommandLineModuleManager::ru	IN+UX176
		▶ gmx::do_lincs	2.421s		gmx::do_lincs(gmx::ArrayRefWithPadding <gmx::basicvector<float> cr</gmx::basicvector<float>	gmx_mpi ! main+0x9b	
		amy"lines update atoms ind	2 368s		amy: lines_update_atoms_ind(amy: ArravRef <int const="">_amy: ArravR</int>	gmx_mpi ! _start+0x29 - start.S:120	
			1s	1.5s	2s 2.5s 3s 3.5s 4	is 4.5s 5s 5.5s 🔽 Thr	read 🗸
		ਲੂ OMP Primary Thread #0 (TI					Running
		OMP Worker Thread #85 (TI					CPU Time
		OMP Worker Thread #70 (TI					CPU Sample
		OMP Worker Thread #92 (TI					, J Utilization
		OMP Worker Thread #84 (TI					CPU Time
		OMP Worker Thread #107 (					Spin and Overhead…
		OMP Worker Thread #108 (					
<u>نې</u>		OMP Worker Thread #91 (TI					
		CPU Utilization					
?		FILTER 🝸 100.0% 🙀 Process Any Process	✓ Thread Any	Thread	✓ Module Any Module ✓ Any Utilization ✓	Only user functions V Functions only V Show	w inline functions 🗸

intel 14

	P. + 🗅 🕹	Welcome × hs-1.sprpvc ×				
Т	<b>▼ ■</b> sam…	Hotspots ③ 山				INTEL VTUNE PROFILER
_	r000hs	Analysis Configuration Collection Log Summary Bottom-up Caller/Callee To	p-down Tree Flame Graph Platform	Customize C	Grouping	
	r001ue	Grouping: Source Function / Function / Call Stack		✓ ≪ Q ::	Call Stacks	ቤ 🗄
4	r002ps	Source Function / Call Stack			libiomp5 so l _ kmp launch thread _kmp runtime	opp
3	r003so	kmp launch thread	28.307s	kmp launch thread	libiomp5.so   INTERNAL7b50d17b::[OpenMB.work	vorl±0x200 z Lipux util o
	r004so	v nbnxm kernel ElecEw VdwLJ F 2xmm	7.187s	nbnxm kernel ElecEw	librothpo.so : _INTERNAL/DS0d1/D[Openin/P work	
~	r005so	▼ nbnxm kernel ElecEw VdwLJ F 2xmm	7.187s libgromacs mpi.so.8	nbnxm kernel ElecEw		
	r006so	▼ nbnxn_kernel_cpu	7.187s libgromacs_mpi.so.8	nbnxn_kernel_cpu(Pairl	IIDC.SO.6 ! CIONE+UX3e	
	r007so	▶ < [OpenMP dispatcher] ← [OpenMP fork] ← nonbonded_verlet_t::dispatchNonbor	7.103s libiomp5.so	kmp_invoke_task_fur		
	r008so	$\blacktriangleright$ [OpenMP fork] $\leftarrow$ nonbonded_verlet_t::dispatchNonbondedKernel $\leftarrow$ do_nb_verlet_t::dispatchNonbondedKernel	0.085s libiomp5.so	kmpc_fork_call		
		▶ fft5d_execute	4.870s	fft5d_execute(fft5d_plar		
		INTERNAL1311483b::kmp_wait_template <kmp_flag_64<(bool)0, (bool)1="">, (bool)1, (bo</kmp_flag_64<(bool)0,>	2.778s	_INTERNAL1311483b::		
		spread_on_grid.extracted.13	1.894s	spread_on_grid(gmx_pr	L Shows CP	Usample
		gmx::lincs_update_atoms	1.663s	gmx::lincs_update_aton		
		compute_1d_small_fwd	1.633s	compute_1d_small_fwd	noir	nte
		▶ DftiComputeBackward	1.362s	DftiComputeBackward		100
		▶ spread_on_grid	0.963s	spread_on_grid(gmx_pr		
		PairlistSet::constructPairlists.extracted.72	0.764s	PairlistSet::constructPai		
		▶ clearForceBuffer	0.649s	clearForceBuffer(nbnxn		
		hnnn atomdata add nhat f to f reduce	0.647s	nhnxn atomdata add r		
<b>3</b>	Direction       Direction					

#### Intel VTune Profiler

– 🗆 X

## More Resources

Intel<sup>®</sup> VTune<sup>™</sup> Profiler – Performance Profiler

- Product page overview, features, FAQs...
- Training materials <u>Cookbooks</u>, <u>User Guide</u>, <u>Processor Tuning</u> <u>Guides</u>
- Support Forum
- <u>Online Service Center</u> Secure Priority Support
- What's New?

#### Additional Analysis Tools

- Intel<sup>®</sup> Advisor Design code for efficient vectorization, threading, memory usage, and accelerator offload
- Intel® Inspector memory and thread checker/ debugger
- Intel® Trace Analyzer and Collector MPI Analyzer and Profiler

Additional Development Products

oneAPI: A new era of heterogenous computing





## Summary

- Hotspots is the most common analysis after APS
- After Summary you may try bottom up
- Check out grouping and filtering features
- Many tutorials + YouTube videos available

#### Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation. Learn more at intel.com or from the OEM or retailer.

Your costs and results may vary.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

**Optimization Notice:** Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804. <a href="https://software.intel.com/enus/articles/optimization-notice">https://software.intel.com/enus/articles/optimization-notice</a>

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. See backup for configuration details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

#