

HLRS Workshop – 15.09.2022

# Application Profiling with Intel® VTune™ and PTI-GPU

Dr. Heinrich Bockhorst



# Agenda

- PTI-GPU tools
- onetrace tool from PTI-GPU
- VTune overview
- Command lines – Playbook
- VTune GPU analysis using GROMACS
- Documentation
- Demo – Hands on

# Profiling Tools Interfaces for GPU (PTI for GPU)

- Light weight Open Source tools for GPU profiling (MIT license)
- Github initiated by Intel<sup>®</sup> engineers but with contributions from community:  
<https://github.com/intel/pti-gpu>
- Profiling for Intel<sup>®</sup> GPUs using OpenCL and Level Zero runtime
- OpenCL\*, SYCL\*, and OpenMP\* offloading supported
- Snapshots taken from Intel<sup>®</sup> DevCloud using Intel<sup>®</sup> Iris<sup>®</sup> Xe GPUs

# sysmon

- Linux static and dynamic information about GPU activity
- Can be used like Linux „top“

```
u48655@s011-n001: ~  
Every 1.0s: /data/comp/sysmon s011-n001: Fri May 20 11:23:16 2022  
=====  
GPU 0: Intel(R) Iris(R) Xe MAX Graphics [0x4905] PCI Bus: 0000:69:00.0  
Vendor: Intel(R) Corporation Driver Version: 1.2.21786 Subdevices: 0  
EU Count: 96 Threads Per EU: 7 EU SIMD width: 8 Total Memory(MB): 7714.0  
Core Frequency(MHz): 1550.0 of 1550.0 Core Temperature(C): unknown  
=====  
Running Processes: 2  
  PID, Device Memory Used(MB), Shared Memory Used(MB), GPU Engines, Executable  
  221409, 15.9, 0.0, 3D;DMA, /home/u48655/software/GROMACS/2022/gromacs-master-22.02.03/oneapi/2022.1.2SIMD8USM/bin/gmx  
  221449, 0.2, 0.0, UNKNOWN, /data/comp/sysmon  
=====  
GPU 1: Intel(R) Iris(R) Xe MAX Graphics [0x4905] PCI Bus: 0000:1b:00.0  
Vendor: Intel(R) Corporation Driver Version: 1.2.21786 Subdevices: 0  
EU Count: 96 Threads Per EU: 7 EU SIMD width: 8 Total Memory(MB): 7714.0  
Core Frequency(MHz): 300.0 of 1550.0 Core Temperature(C): unknown  
=====  
Running Processes: 1  
  PID, Device Memory Used(MB), Shared Memory Used(MB), GPU Engines, Executable  
  221449, 0.2, 0.0, UNKNOWN, /data/comp/sysmon
```

# onetrace

- Profile OpenCL\* and Level Zero backend.
- For OpenCL, SYCL\* and OpenMP\* offloading applications.

usage: onetrace [options] <application> <args>

Options:

--host-timing [-h]	host API calls statistics
--device-timing [-d]	kernel execution timing
--chrome-kernel-timeline	generates trace for chrome://tracing

...

# onetrace host statistics (GROMACS SYCL)

- \$ onetrace -h gmx <gmx args>

```

/cydrive/c/Users/hbockhor/OneDrive - Intel Corporation/Training/2022/22.05.30_ISC_MTA/RESULTS_onetrace

=== API Timing Results: ===

      Total Execution Time (ns):          8849359641
      Total API Time for LO backend (ns):  5083012588

== LO Backend: ==

      Function,           Calls,      Time (ns),  Time (%),   Average (ns),   Min (ns),     Max (ns)
zeCommandQueueSynchronize,  2009,    3697535106,  72.74,     1840485,       164,          7145738
zeModuleCreate,              7,      1030797983,  20.28,     147256854,    83120536,     227200191
zeCommandQueueExecuteCommandLists, 4226,    146236830,  2.88,      34604,        14481,        104637
zeCommandListAppendMemoryCopy, 3172,    87847894,  1.73,      27694,        11550,        840112
zeCommandListReset,         4226,    37004797,  0.73,      8756,         554,          36072
zeCommandListCreate,         8,      27893013,  0.55,     3486626,     226640,       13670348
zeCommandListAppendLaunchKernel, 2811,    13505254,  0.27,      4804,         926,          17791
zeKernelSetArgumentValue,   35215,    9806280,  0.19,      278,          31,           9071
zeEventCreate,              5983,    6979341,  0.14,     1166,         364,          54293
zeCommandListClose,         4226,    3455592,  0.07,      817,         140,          15018
zeMemAllocDevice,           11,     3054949,  0.06,     277722,     56756,        1191462
zeFenceReset,               4226,    3037219,  0.06,      718,         536,          24105
zeKernelSetGroupSize,      2811,    2867434,  0.06,     1020,        141,          4410

```

# onetrace device statistics (GROMACS SYCL)

- `$ onetrace --demangle -d gmx <gmx args>`

```

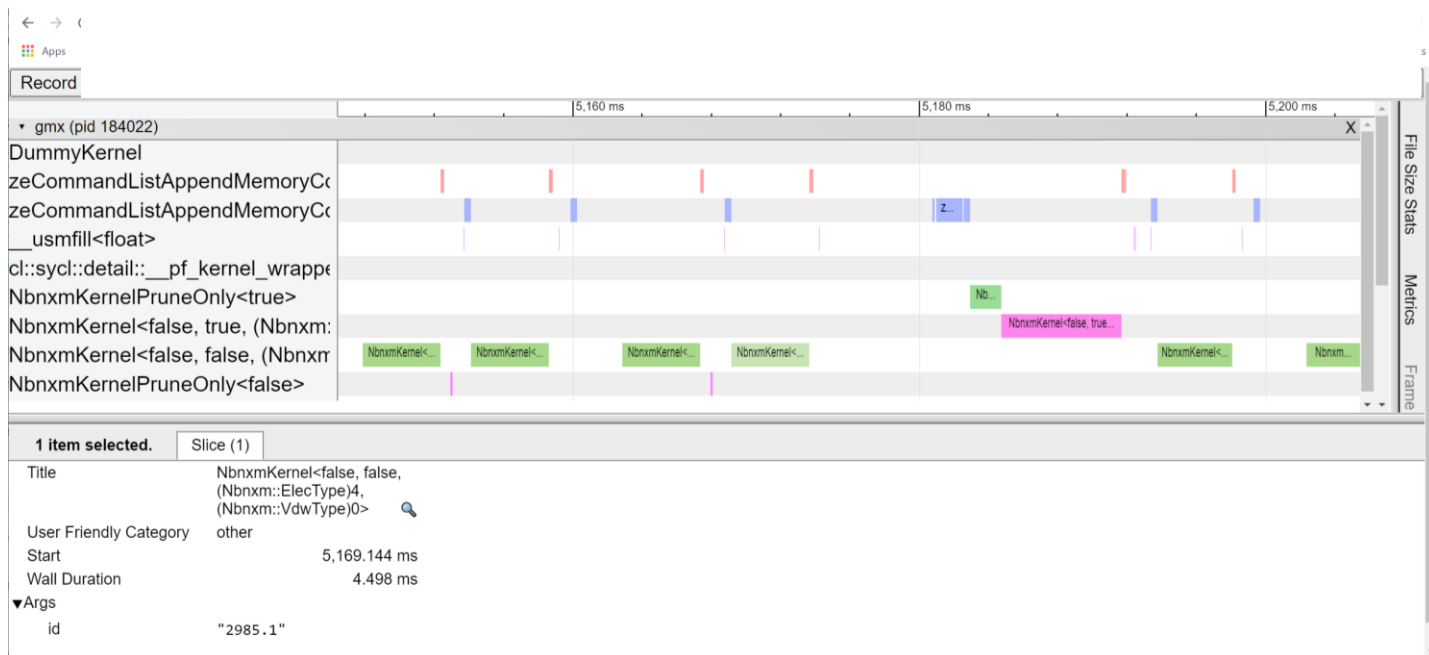
/cygdrive/c/Users/hbockhor/OneDrive - Intel Corporation/Training/2022/22.05.30_ISC_MTA/RESULTS_onetrace
=== Device Timing Results: ===
      Total Execution Time (ns):          9250282340
      Total Device Time for L0 backend (ns): 5144159294
== L0 Backend: ==
      Kernel,           Calls,           Time (ns),           Time (%),           Average (ns),           Min (ns),           Max (ns)
NbnxmKernel<false, false, (Nbnxm::ElecType)4, (Nbnxm::VdwType)0>,           990,           4374811057,           85.04,           4419001,           4290416,           4946458
      zeCommandListAppendMemoryCopy(M2D),           2047,           393601766,           7.65,           192282,           3229,           1283489
      zeCommandListAppendMemoryCopy(D2M),           1125,           183165569,           3.56,           162813,           520,           190104
NbnxmKernel<false, true, (Nbnxm::ElecType)4, (Nbnxm::VdwType)0>,           11,           77472702,           1.51,           7042972,           6985937,           7131562
      NbnxmKernelPruneOnly<false>,           490,           67430948,           1.31,           137614,           130416,           151979
      __usmfill<float>,           1307,           27828716,           0.54,           21292,           1666,           42500
      NbnxmKernelPruneOnly<true>,           11,           19808433,           0.39,           1800766,           1798125,           1804687
      cl::sycl::detail::__pf_kernel_wrapper<__usmfill<float> >,           1,           35833,           0.00,           35833,           35833,           35833
      DummyKernel,           1,           4270,           0.00,           4270,           4270,           4270

```

More information available by adding `-v` (verbose) option

# onetrace kernel timeline

- `$ onetrace --demangle --chrome-kernel-timeline <gmx> <args>`





# PTI-GPU advantages

- No additional drivers necessary
- Low overhead < 3%
- Traces in json format to be loaded by Google Chrome:  
`chrome://tracing`

# Optimize Performance

Intel® VTune™ Profiler

## Get the Right Data to Find Bottlenecks

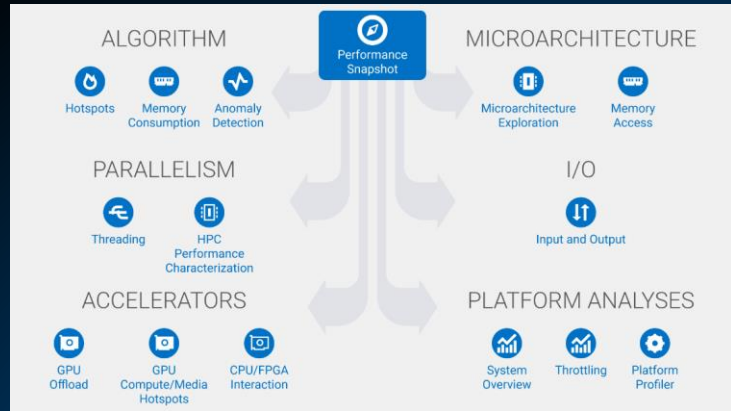
- A suite of profiling for CPU, GPU, FPGA, threading, memory, cache, storage, offload, power...
- DPC++, C, C++, Fortran, Python\*, Go\*, Java\*, or a mix
- Linux, Windows, FreeBSD, Android, Yocto and more

## Analyze Data Faster

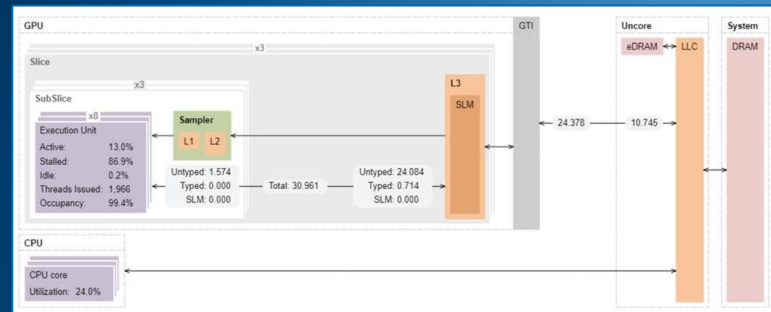
- See data on your source, in architecture diagrams, as a histogram, on a timeline...
- Filter and organize data to find answers

## Work Your Way

- User interface or command line
- Profile locally and remotely
- Install as an application
- Install as a server accessible with a web browser



Source		Assembly		GPU Instructions Executed by Instruction T...	
				Control Flow	Send & Wait
				Int32 & SP Float	Int64 & DP Float
				Other	
158	<code>dx = ptr[j].pos[0] - ptr[i].pos[0];</code>	75,002,500			
159	<code>dy = ptr[j].pos[1] - ptr[i].pos[1];</code>	12,500,000			
160	<code>dz = ptr[j].pos[2] - ptr[i].pos[2];</code>	12,500,000			



# VTune Playbook – for this session

- Easy access to command lines
- Nbody Example for DevCloud – please try

```
~/INCOMING/22.02.15-EuroCC
Playbook for using vTune tool on devcloud or other clusters
-----
Note: command lines start with "$" prompt.
-----
1. Log into DevCloud
-----
$ ssh devcloud
Alternative: open a jupyter notebook and start the terminal
-----
2. Clone samples GitHub
-----
$ git clone https://github.com/oneapi-src/oneapi-samples.git
-----
3. Start interactive session on a node with DG1 Xe GPU (iris_xe_max)
-----
(People with NDA accounts may use ATS-P gpu)

It is better to compile on compute node because login node has very limited memory etc.
$ qsub -I -l nodes=1:iris_xe_max:ppn=2
-----
4. Check properties
-----
$ sycl-ls --verbose

prints out all backends (GPU device + low level driver level_zero or opencl)
Level Zero shows:
Platform [#4]:
  Version : 1.2
  Name    : Intel(R) Level-Zero
  Vendor  : Intel(R) Corporation
  Devices : 1
  Device [#0]:
```

Playbooks on DevCloud:  
[/data/comp/workshop/vtune\\_playbook.txt](/data/comp/workshop/vtune_playbook.txt)

# How to start an Analysis

- VTune offers different analysis types with additional “knobs”
- Application Performance Snapshot (VTune or standalone)
- APS does first analysis and guides to the right VTune analysis or a different tool available in the oneAPI toolkits.
- APS analyses HW counters, OpenMP and MPI
- APS can be used for MPI and/or OpenMP only to avoid too much data being collected

# APS usage

- Help menu

```
$ aps -help
```

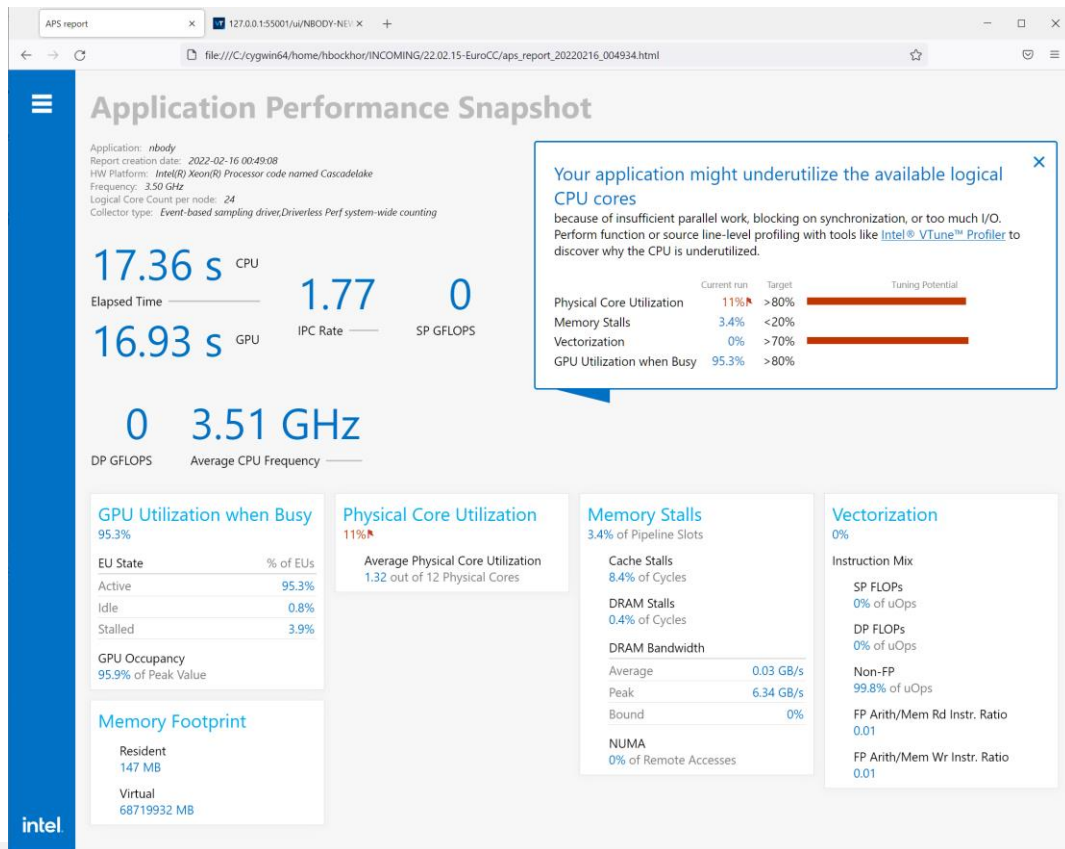
- Run with APS

```
$ aps <application> <app paramter>
```

- Run with MPI (Intel MPI or compatible)

```
$ mpirun -n <N> aps -- <application> < app parameter>
```

# APS HTML output (nbody)



Intel® VTune™ Profiler  
GROMACS  
SYCL version

# VTune GPU analysis

- Host API and GPU kernels
- Detailed source view on Kernels
- Bandwidth data with Hierarchical Memory Diagram
- Estimate for timing per source line (basic block timing)
- Estimate for memory latency per source line



# SYCL version of GROMACS

- Clone master branch:

\$ git clone <https://gitlab.com/gromacs/gromacs.git>

- Workload: <https://www.mpinat.mpg.de/632209/benchMEM.zip>

```
SRC=$HOME/gromacs  
PRE=$HOME/GROMACS/
```

```
cmake $SRC  
-DCMAKE_INSTALL_PREFIX=$PRE \\  
-DGMX_OPENMP=ON \\  
-DGMX_OPENMP_MAX_THREADS=128 \\  
-DGMX_GPU=SYCL \\  
-DGMX_FFT_LIBRARY=mk1 \\  
-DCMAKE_EXE_LINKER_FLAGS= \\  
-DCMAKE_C_COMPILER=icx \\  
-DCMAKE_CXX_COMPILER=icpx
```

Please check: <https://manual.gromacs.org/>

- Run:

```
${EXE_DIR}/gmx mdrun -ntmpi $RANKS -ntomp $OMP_NUM_THREADS -nb $MODE -pme cpu -s benchMEM.tpr -nsteps $steps
```



# Application Performance Snapshot

Application: gmx  
 Report creation date: 2022-02-16 05:09:15  
 OpenMP threads: per Process: 12  
 HW Platform: intel(R) Xeon(R) Processor code named CascadeLake  
 Frequency: 3.50 GHz  
 Logical Core Count per node: 24  
 Collector type: Event-based sampling driver: Driverless Perf system-wide counting

**24.12 s** CPU

Elapsed Time

**0.92**

IPC Rate

**7.81**

SP GFLOPS

**0.13**

DP GFLOPS

**3.59 GHz**

Average CPU Frequency

## GPU Utilization when Busy

58.2%

EU State	% of EUs
Active	58.2%
Idle	19.1%
Stalled	22.7%

GPU Occupancy

80.6% of Peak Value

## Vectorization

56.1%

### Instruction Mix

#### SP FLOPs

7% of uOps  
 Packed: 55.8% from SP FP  
 128-bit: 11.6%   
 256-bit: 31.8%   
 512-bit: 12.3%   
 Scalar: 44.2% from SP FP

#### DP FLOPs

0.2% of uOps  
 Packed: 66% from DP FP  
 128-bit: 3.5%  
 256-bit: 62.6%   
 512-bit: 0%  
 Scalar: 34% from DP FP

#### Non-FP

92.7% of uOps

FP Arith/Mem Rd Instr. Ratio

0.26

FP Arith/Mem Wr Instr. Ratio

0.94

## Serial Time

20.12 s

83.42% of Elapsed Time

## Memory Footprint

Resident

882 MB

Virtual

34362707 MB

Your application has significant serial time that can prevent application scalability.  
 Use OpenMP profiling tools like [Intel® VTune™ Profiler](#) to explore serial hotspots and opportunities on parallelization.

	Current run	Target	Tuning Potential
Serial Time	83.42%	<15%	
OpenMP Imbalance	1.08%	<10%	
Memory Stalls	20.6%	<20%	
Vectorization	56.1%	>70%	
GPU Utilization when Busy	58.2%	>80%	

## Memory Stalls

20.6% of Pipeline Slots

### Cache Stalls

14.8% of Cycles

### DRAM Stalls

5.8% of Cycles

### DRAM Bandwidth

Average 3.18 GB/s

Peak 14.62 GB/s

Bound 0%

### NUMA

0% of Remote Accesses

# Command lines used

- Basic gpu analysis:

```
vtune -collect gpu-hotspots -r <your-result-dir> -- <executable><args>
```

- Full instrumentation:

```
vtune -c gpu-hotspots -knob characterization-mode=instruction-count \  
-r <your-result-dir> -- <executable><args>
```

- Source Instrumentation with timing of basic blocks:

```
vtune -c gpu-hotspots -knob profiling-mode=source-analysis \  
-r <your-result-dir> -- <executable><args>
```

- Source Instrumentation with only memory inst. timed:

```
vtune -c gpu-hotspots -knob profiling-mode=source-analysis \  
-knob source-analysis=mem-latency -r ...
```

# DEMO: Memory Hierarchy vtune -c gpu-hotspots

APS report | 127.0.0.1:55001/ui/NBODY-NEV x | +

https://127.0.0.1:55001/ui/GROMACS/gh

Project... + | Welcome x | Configure Analysis x | gh x | gh x

## GPU Compute/Media Hotspots (preview) ©

Analysis Configuration | Collection Log | Summary | Graphics

Memory Hierarchy Diagram | Platform

**GPU**

**GT**

Slice x6

SubSlice x16

Execution Unit x16

Active: 69.0%  
Stalled: 26.1%  
Idle: 4.9%  
Threads Issued: 1,408,535  
Occupancy: 94.8%

Sampler

Busy: 0.0%  
Bottleneck: 0.0%

L1 L2

38.3 GB/s

2.9 GB/s

SLM

L3

1.6 GB/s

323.0 MB/s

Traffic Router

VRAM

PCIe

Uncore

LLC

System

DRAM

**GPU**

Grouping: Computing Task

Computing Task	Work Size		Computing Task				Data Transferred		EU Array			EU Threads Occupancy	Com
	Global	Local	Total Time	Average Time	Instance Count	SIMD Width	SVM Usage Type	Size	Total, GB/sec	Active	Stalled		
▶ NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	4 x 4 x 7129	4 x 4 x 1	1.226s	0.006s	198	8	0 B	0.000	69.0%	26.1%	4.9%	94.8%	
▶ NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	4 x 4 x 7149	4 x 4 x 1	0.614s	0.006s	99	8	0 B	0.000	68.8%	26.2%	4.9%	94.8%	
▶ NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	4 x 4 x 7147	4 x 4 x 1	0.614s	0.006s	99	8	0 B	0.000	68.8%	26.2%	4.9%	94.8%	
▶ NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	4 x 4 x 7137	4 x 4 x 1	0.614s	0.006s	99	8	0 B	0.000	68.9%	26.1%	5.0%	94.7%	
▶ NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	4 x 4 x 7140	4 x 4 x 1	0.614s	0.006s	99	8	0 B	0.000	68.8%	26.2%	4.9%	94.8%	
▶ NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	4 x 4 x 7117	4 x 4 x 1	0.614s	0.006s	99	8	0 B	0.000	68.8%	26.2%	5.0%	94.8%	
▶ NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	4 x 4 x 7130	4 x 4 x 1	0.614s	0.006s	99	8	0 B	0.000	68.7%	26.3%	5.0%	94.7%	
▶ NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	4 x 4 x 7162	4 x 4 x 1	0.613s	0.006s	99	8	0 B	0.000	68.8%	26.1%	5.2%	94.5%	
▶ NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	4 x 4 x 7135	4 x 4 x 1	0.612s	0.006s	99	8	0 B	0.000	68.9%	26.1%	5.0%	94.7%	

FILTER

# DEMO: Platform (zoomed)

INTEL VTUNE PROFILER



Grouping: Source Computing Task

Source Computing Task	Computing Task			Data Transferred		EU Array			EU Threads Occupancy	Computing Threads Started	L3 Bandwidth, GB/sec	Shared Local Mem	
	Total Time	Average Time	Instance Count	Size	Total, GB/sec	Active	Stalled	Idle				Read	Write
NbnxmKernel<(bool)0, (bool)0, (Nbnxm::ElecType)4, (Nbnxm::Vdw	34.799ms	5.800ms	6	0 B	0.000	68.9%	26.1%	5.0%	94.7%	41,256	188.241	38.920	3.1
zeCommandListAppendMemoryCopy	3.320ms	0.184ms	18	14.2 MB	4.290	0.0%	0.0%	100.0%	0.0%	0	0.000	0.000	0.1
NbnxmKernelPruneOnly<(bool)0>	0.539ms	0.180ms	3	0 B	0.000	56.0%	17.5%	26.4%	71.9%	5,346	58.006	16.002	0.1
_usmfill<float>	0.206ms	0.034ms	6	0 B	0.000	0.0%	0.0%	100.0%	0.0%	0	0.000	0.000	0.1
[Outside any task]	0ms					0.2%	1.1%	98.7%	0.4%	23,844	1.083	0.000	0.1

DEMO:  
characterization-mode  
=instruction-count

APS report x vt 127.0.0.1:55001/ui/NBODY-NEV x +

https://127.0.0.1:55001/ui/GROMACS/ghi

GPU Compute/Media Hotspots (preview) @

Analysis Configuration Collection Log Summary Graphics nbnxm\_sycl\_kernel.cpp x

Source Assembly

Source Line ▲	Source	GPU Instructions Executed by Instruction Type
847	// cutoff & exclusion check	
848		
849	const bool notExcluded = doExclusionForces ? (nonSelfInteraction   (ci != cj))	
850	: (wexcl & maskJI);	
851		
852	// SYCL-TODO: Check optimal way of branching here.	
853	if ((r2 < rCoulombSq) && notExcluded)	15,247,802,964
854	{	
855	const float q1 = Xq1[3];	
856	int atomTypeI; // Only needed if (!props.vdwComb)	
857	float sigma, epsilon;	
858	Float2 c6c12;	
859		
860	if constexpr (!props.vdwComb)	
861	{	
862	/* LJ 6*C6 and 12*C12 */	
863	atomTypeI = sm_atomTypeI[i * c_clSize + tidxi];	3,158,901,217
864	c6c12 = a_nbf[numTypes * atomTypeI + atomTypeJ];	89,057,962,872
865	}	
866	else	
867	{	
868	const Float2 ljCombI = sm_ljCombI[i * c_clSize + tidxi];	
869	if constexpr (props.vdwCombGeom)	
870	{	
871	c6c12 = Float2(ljCombI[0] * ljCombJ[0], ljCombI[1] * ljCombJ[1]);	
872	}	
873	else	
874	{	
875	static_assert(props.vdwCombLB);	
876	// LJ 201/61stima and 12comb12	

GPU Instructions Executed by Instruction Type

Control Fl... Se... Synchronizat... Int16 & HP Fl... Int32 & SP F

# DEMO: profiling-mode=source- analysis

The screenshot displays the Intel VTune Profiler interface. The top navigation bar shows the project name 'APS report' and the URL 'https://127.0.0.1:55001/ui/NBODY-NEV'. The main window is titled 'GPU Compute/Media Hotspots (preview)'. The left sidebar shows a project tree with folders like 'ggg', 'NBody', and 'NBO...'. The central pane shows the source code for 'nbnxm\_sycl\_kernel.cpp'. The right pane shows a table of GPU hotspots.

Source Line	Source	Estimated GPU Cycles
852	// SYCL-TODO: Check optimal way of branching here.	
853	if ((r2 < xCoulombSq) && !notExcluded)	1.1%
854	{	
855	const float qi = xqi[3];	
856	int atomTypeI; // Only needed if (!props.vdwComb)	
857	float sigma, epsilon;	
858	Float2 c6c12;	
859		
860	if constexpr (!props.vdwComb)	
861	{	
862	/* LJ 6*C6 and 12*C12 */	
863	atomTypeI = sm_atomTypeI[i * c_clSize + tidxi];	1.4%
864	c6c12 = a_nbf[numTypes * atomTypeI + atomTypeU];	17.3%
865	}	
866	else	
867	{	
868	const Float2 ljCombI = sm_ljCombI[i * c_clSize + tidxi];	
869	if constexpr (props.vdwCombGeom)	
870	{	
871	c6c12 = Float2(ljCombI[0] * ljCombJ[0], ljCombI[1] * ljCombJ[1]);	
872	}	
873	else	
874	{	
875	static_assert(props.vdwCombLB);	
876	// LJ 2^(1/6)*sigma and 12*epsilon	
877	sigma = ljCombI[0] + ljCombJ[0];	
878	epsilon = ljCombI[1] * ljCombJ[1];	
879	if constexpr (doCalcEnergies)	
880	{	
881	c6c12 = convertSigmaEpsilonToC6C12(sigma, epsilon);	
882	}	

DEMO:  
source-analysis=mem-  
latency

APS report x 127.0.0.1:55001/ui/NBODY-NEV x +

https://127.0.0.1:55001/ui/GROMACS/ghl

GPU Compute/Media Hotspots (preview)

Analysis Configuration Collection Log Summary Graphics nbnxm\_sycl\_kernel.cpp

Source Assembly

Source Line	Source	Average Latency, Cycles	Estimated GPU Cycles
848			
849	const bool notExcluded = doExclusionForces ? (nonSelfInteraction   (ci != cj))		
850	: (wexcl & maskJI);		
851			
852	// SYCL-TODO: Check optimal way of branching here.		
853	if ((r2 < rCoulombSq) && notExcluded)		
854	{		
855	const float qi = xqi[3];		
856	int atomTypeI; // Only needed if (!props.vdwComb)		
857	float sigma, epsilon;		
858	Float2 c6c12;		
859			
860	if constexpr (!props.vdwComb)		
861	{		
862	/* LJ 6*C6 and 12*C12 */		
863	atomTypeI = sm_atomTypeI[i * c_clSize + tidxi];	58 7.1%	
864	c6c12 = a_nbf[numTypes * atomTypeI + atomTypeJ];	185 22.6%	
865	}		
866	else		
867	{		
868	const Float2 ljCombI = sm_ljCombI[i * c_clSize + tidxi];		
869	if constexpr (props.vdwCombGeom)		
870	{		
871	c6c12 = Float2(ljCombI[0] * ljCombJ[0], ljCombI[1] * ljCombJ[1]);		
872	}		



# Documentation

VTune cookbooks:

<https://www.intel.com/content/www/us/en/develop/documentation/vtune-cookbook/top.html>

Vtune on DevCloud:

<https://www.intel.com/content/www/us/en/develop/documentation/vtune-cookbook/top/configuration-recipes/using-vtune-server-with-vs-code-intel-devcloud.html>

GPU profiling:

<https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/analyze-performance/accelerators-group/gpu-compute-media-hotspots-analysis.html>

# Notices & Disclaimers

Performance varies by use, configuration and other factors. Learn more at [www.Intel.com/PerformanceIndex](https://www.Intel.com/PerformanceIndex).

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

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

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

© 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.

intel®