using Intel[®] VTune and Intel[®] Advisor on GPUs:

Stephen Blair-Chappell

External Intel Certified oneAPI Instructor

7th June 2023



Introduction



About Me



Stephen Blair-Chappell is an

independent software consultant and is an Intel certified oneAPI instructor. He was formerly the Technical Director at Bayncore where he led a team of consultants providing HPC and AI training on Intel Architecture. For 18 years he was a Technical Consulting Engineer at Intel helping their strategic customers in software optimisation and code modernisation. He is author of the book "Parallel Programming with Intel Parallel Studio XE".

stephen-at-sbcnow-dot-co-dot-uk

Today's session is about getting visibility



Agenda

- Occupancy
- GPU Offload Modelling with Advisor
- GPU Offload Analysis with Intel Vtune
- GPU Roofline Analysis with Intel Advisor
- Experience with Lammps
- Q & A

Host with accelerator GPUs



Programmers' perspective: Three things to consider

Offload the code to the device

7

Choosing appropriate kernel code

Manage the transfer of Data

(a) Mitigating againstcommunication & memory latency(b) Re-use of data

Implement Parallelism

Maximising Occupancy



Programmers' perspective: Three things to consider

Offload the code to the device

Choosing appropriate kernel code

Manage the transfer of Data

(a) Mitigating againstcommunication & memory latency(b) Re-use of data

Implement Parallelism

Maximising Occupancy

Today's examples



GPU Occupancy



EXECUTION UNIT	EXECUTION UNIT			
THREAD	CONTROL			
THREAD STATE & REGISTER FILE	THREAD STATE & REGISTER FILE			
	FPU / INT			
SEND	SEND			
BRANCH	BRANCH			

Intel[®] Iris[®] X^e-LP GPU Compute Throughput Rates (Ops/clock/EU)

FP32	FP16	INT32	INT16	INT 8
8	16	8	16	32 (DP4A)

X^e-LP configuration

SHAR	RED F	UNCT	rions	5	CO	PY E	NGINE										MEDIA	ENGI	IE						
GEOMETRY				RASTER PIXEL DISPATCH																					
		50	BSLICE			SUR	SLICE	71-60		SUB	SLICE	191015	1		SUE	SLICE			5	JBSLICE			SU	RSLICE	
	IS	Th	IREAD D	ISPATCH	15	тн	READ DIS	PATCH	15	TH	READ DIS	БРАТСН	I	IS	TH	READ DIS	SPATCH	15		HREAD	SPATCH	IS	Т	IREAD DI	SPATCH
	EU	EU	EU	EU	EU	EU	EU	EU	EU	EU	EU	EU		EU		EU	EU	EU	EU	EU		EU			EU
Ш	EU	EU	EU	EU	EU	EU	EU	EU	EU	EU	EU	EU		EU				EU	EU	EU		EU			EU
Ĭ	EU	EU	EU	EU	EU	EU	EU	EU	EU	EU	EU	EU		EU	EU	EU		EU	EU	EU		EU			EU
ر م	EU		EU	EU	EU	EU	EU	EU	EU	EU	EU	EU	l	EU	EU	EU		EU	EU	EU		EU			EU
×	SAMPLER	AFRIA	SAMPLER	LOAD ' STORE	SAMPLER	MEDIA	SAMPLER	LOAD ' STORE	SAMPLER	MEDIA	SAMPLER	LOAD ' STORE	ľ	SAMPLER	MEDIA	SAMPLER	LOAD ' STORE	SAMPLER		MEDIA SAMPLER	LOAD STORE	SAMPLER		SAMPLER	LOAD
	L1)	TEX\$		SLM	L1/1	TEX\$	s	ilm	L1/1	EX\$	3	SLM	I	L1/	EX\$		SLM	L1,	TEX\$		SLM	L1,	TEX\$		SLM
				PIXEL E	ACKEN	۱D					P	IXEL E	A	CKEN	D			PIXEL BACKEND							
			_							_															
		1				50	20	ACU	E												G	AM			
							L3 C	ACH	E			1								GTI	Sec. U		(STI	

Xe-LP (TGL) GPU

	VEs	Threads	Operations	Maximum Work Group Size
Each X ^e -core	16	$7\times 16=112$	$112\times8=896$	512
Total	$16 \times 6 = 96$	$112\times 6=672$	$896\times 6=5376$	512

6 Cores

16 Vector Engines/core [Total 96]

7 Threads

16 FP16 ops/clock/VE

INTEL® GPU OCCUPANCY CALCULATOR



The GPU Occupancy Calculator gives a theoretical estimate of GPU Occupancy, actual occupancy on the hardware may be different. Note that higher occupancy does not always translate to higher performance.

Total Number of Work-Groups 512

Impact of varying Work-Group Size



Size of (262144) on Integrated GPU (Xe LP)							
Valid WG Sizes	SS Occupancy	WG per SS	GPU Occupancy				
512	100.00%	7	97.30%				
256	100.00%	14	97.30%				
128	100.00%	28	97.30%				
64	100.00%	56	97.30%				
32	100.00%	112	97.30%				
16	50.00%	112	49.32%				
8	25.00%	112	24.83%				
4	12.50%	112	12.46%				
2	6.25%	112	6.24%				
1	3.13%	112	3.12%				

VALID WORK-GROUP SIZES

The list below show all the valid Work-Group(WG)

sizes and GPU Sub-Slice(SS) Occupancy for Globa

100% V OCCUPANCY CONFIGURATION

This table shows ideal combination of Work-Group(WG) size, Sub-Group size and Shared Local Memory(SLM) usage limit that will get you 100% occupancy in the selected hardware Sub-Slice(SS).

Work-Group Size	Sub-Group Size	SLM Limit (Bytes)	í
512	32	9362	
448	32	8192	
256	32	4681	
224	32	4096	
128	32	2340	
112	32	2048	
64	32	1170	
56	32	1024	
32	32	585	
448	16	16384	
258	16	9362	
224	16	8192	
128	16	4681	

Maximum Occupancy Calculator

Intel. GPU Occupancy Calculator

https://github.com/oneapi-src/oneAPI-samples/tree/master/Tools/GPU-Occupancy-Calculator intel. ¹²

Language concepts for max occupancy

SYCL	OpenMP
Work-item	Thread
Sub-group	Team
Work-Group	League

oneAPI Debug Tools (intel.com)

Tool	Description
Environment variables	Gather diagnostic information from the OpenMP and SYCL runtimes at program execution with no modifications to your program.
Onetrace	From Profiling Tools Interfaces for GPU (PTI for GPU). Used to debug backend errors and for performance profiling on both the host and device.
Intercept Layer for OpenCL™ Applications	Used to debug backend errors and for performance profiling on both the host and device (has wider functionality comparing with onetrace).
Intel [®] Distribution for GDB*	Used for source-level debugging of the application, typically to inspect logical bugs, on the host and any devices you are using (CPU, GPU, FPGA emulation).
Intel [®] Inspector	locate and debug memory and threading problems, including those that can cause offloading to fail.
In-application debugging	printf; looking at output of apps etc, etc
Intel [®] Advisor	Use to ensure Fortran, C, C++, OpenCL™, and SYCL applications realize full performance potential on modern processors.
Intel [®] VTune TM Profiler	Use to gather performance data either on the native system or on a remote system.

Advisor Offload Modelling



Offload Modelling – doing it by hand

Run code on CPU and find hotspots





Examine results – decide which hotspots are suitable for offloading



Implement offload using SYCL or OpenMP



Run code on CPU and GPU and see if there is a speed up



Possibly no Potentially time-consuming return-on-investment

Offload Modelling – with Advisor



Run code on CPU using Advisor (multiple stage profiling collection)





Examine results – decide which hotspots are suitable for offloading



- Implement offload using SYCL or OpenMP
- (4)

Run code on CPU and GPU and see if there is a speed up





Demo – Our Example oneapi-cli



make CXX=icpx EXTRA_CFLAGS=-g

Using accuracy presets to control modelling

• Default (Medium Accuracy)

advisor --collect=offload --config=gen12_tgl
--project-dir=./cpu2gpu_offload_modeling -./release/Mandelbrot 1

• Low Accuracy

```
advisor --collect=offload -accuracy=low
--config=gen12_tgl --project-
dir=./cpu2gpu_offload_modeling --
./release/Mandelbrot 1
```

• Getting list of steps

advisor --collect=offload --dry-run --config=gen12_tgl --projectdir=./cpu2gpu_offload_modeling --./release/Mandelbrot 1

Low	Medium	High
5-10x overhead	15-50x overhead	50-80x overhead
Survey Trip Count Offload Modelling	Survey Trip Count Offload Modelling	Survey Trip Count Dependency analysis Offload Modelling
L1 Cache	L1 Cache + Host- Device data	L1 Cache + Host- Device data

Steps to Offload Projection with Advisor

1. Run a **Survey**: get a list of hotspots

advisor -collect survey ...

- SamplingBinary Static Analysis
- Compiler & debug info
- 2. Run a **Trip Count**: count loop iteration

advisor -collect=tripcounts -target-device=gen9_gt2

- Trip count Cache simulation
- 3. Perform a dependency analysis [optional for quick modelling]

advisor -collect dependencies . . .

4. Model the Performance

advisor -collect projection -no-assume-dependencies . . .

Generate HTML report

20

Check memory accesses

Loop selection heuristic

Expensive Steps

DRY RUN LOWACCURACY========= advisor: The 'offload' is a special batch mode for data collection. It runs several analyses one by one. advisor --collect=survey --auto-finalize --static-instruction-mix --project-dir=./cpu2gpu_offload_modeling -- ./03-run.sh LOW advisor --collect=tripcounts --flop --auto-finalize --target-device=gen12_tgl --projectdir=./cpu2gpu offload modeling -- ./03-run.sh advisor --collect=projection --no-assume-dependencies --config=gen12 tgl --projectdir=./cpu2gpu offload modeling -- ./03-run.sh DRY RUN MEDIUM ACCURACY======= advisor: The 'offload' is a special batch mode for data collection. It runs several analyses one by one. advisor --collect=survey --auto-finalize --static-instruction-mix --project-dir=./cpu2gpu_offload_modeling MEDIUN -- ./03-run.sh advisor --collect=tripcounts --flop --stacks --auto-finalize --cache-simulation=single --data-transfer=light --target-device=gen12_tgl --project-dir=./cpu2gpu_offload_modeling -- ./03-run.sh advisor --collect=projection --no-assume-dependencies --config=gen12_tgl --projectdir=./cpu2gpu offload modeling -- ./03-run.sh advisor: The 'offload' is a special batch mode for data collection. It runs several analyses one by one. advisor --collect=survey --auto-finalize --static-instruction-mix --project-dir=./cpu2gpu_offload_modeling -- ./03-run.sh HIGH advisor --collect=tripcounts --flop --stacks --auto-finalize --cache-simulation=single --datatransfer=medium --target-device=gen12_tgl --project-dir=./cpu2gpu_offload_modeling -- ./03-run.sh advisor --collect=dependencies --filter-reductions --loop-call-count-limit=16 -select=markup=gpu_generic --project-dir=./cpu2gpu_offload_modeling -- ./03-run.sh advisor --collect=projection --config=gen12_tgl --project-dir=./cpu2gpu_offload_modeling -- ./03-run.sh Jintel.

21

Summary

intel. AD	VISOR build: 610659 Project:	610659_gen1	2tgl_mediumnogpu Application: M	andelbrot
음 Perspective: O	ffload Modeling Accelerated	Regions •	Source View	
Top Metrics				
11.1X Speed Up for Accel	? erated Code	4.0x Amdahl's	Law Speed Up	33% Fraction of Accelerated C
Program Metrics				^
Original Accelerated	0.22s 0.05s			
	Program Time on Host After Acceleration	0.04s	Target Platform	XeLP GT2
	Time on Target	0.02s	Speed Up for Accelerated Code	11.1x
	Time in MPI calls	0s	Amdahl's Law Speed Up	4.0x
	Non-Accelerable Time	0s	Fraction of Accelerated Code	83%
	Data Transfer Tax	0s	Number of Offloads	1
	Kernel Launch Tax	<0.01s		

 \sim



Combined time spent on host and target after aceleration

Loop Offload (from Summary Page)

Top Five offload candidates



GPU-GPU modelling

Add the –gpu flag

Runs code on ACTUAL GPU and models against new GPU.

• Use when upgrading from one GPU to another one.

NB: use –accuracy=low

Comparison of CPU-GPU and GPU-GPU modelling

CPU-GPU	GPU-GPU
Survey	Survey (on GPU)
Trip Count	Trip Count
	(Characterization
	 num Floats and Integer
	operations)
Dependency Check	X
Model Offloading	Model Offloading

Re-modelling for a different GPU - once you already have a set of results



Adjust Values
 Save parameters
 Re-run modelling

e.g.: advisor -c=projection --custom-config=config.toml --config=gen12_tgl --project-dir=/...

Advantage: quicker than doing a completely new modelling

GPU Offload Analysis with Intel[®] VTuneTM



VTune GPU Profiling Recipe

Intel[®] VTune[™] Profiler Performance Analysis Cookbook

Profiling a SYCL* Application running on a GPU (intel.com)



- 1. Build and Compile a SYCL Application
- 2. Run GPU Offload Analysis on a SYCL Application
- 3. Analyze Collected Data
- 4. Run GPU Compute/Media Hotspots Analysis
- 5. Analyze Your Compute Task

STEP 1: Build and Compile

- cd matrix_multiply_vtune mkdir build –p cd build cmake ..
 make
- icpx -g -O3 -fsycl -Wno-write-strings -w -D_Linux -MD -MT CMakeFiles/matrix.dpcpp.dir/src/matrix.cpp.o
 -MF CMakeFiles/matrix.dpcpp.dir/src/matrix.cpp.o.d
 -o CMakeFiles/matrix.dpcpp.dir/src/matrix.cpp.o
 -c /home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU/matrix_multiply_vtune/src/matrix.cpp

The Starting point – Running a Performance Snapshot

Welcome × Configure Analysis ×		
Configure Analysis 🛍		INTEL VTUNE PROFILER
WHERE	HOW	
Local Host -	Performance Snapshot -	
what ■ Launch Application →	Performance Snapshot	
Specify and configure your analysis target: an application or a script to execute. Follow Prepare Application for Analysis to compile your app for best analysis productivity.		MICROARCHITECTURE
Application:	Hotspots Anomaly	Microarchitecture Memory Access
/home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU/03-run.sh	Detection (preview)	Exploration
Application parameters:	Mamory	
<u>छ</u>	Consumption	
✓ Use application directory as working directory		
Advanced >	PARALLELISM	I/O
	E II: Threading HPC Performance Characterization	LT Input and Output
	ACCELERATORS	PLATFORM ANALYSES
	GPU Offload GPU Compute/Media Hotspots (preview)	System GPU Rendering Overview (preview)
	CPU/FPGA Interaction	Platform Profiler
	Get a quick snapshot of your application performance and identify n	ext steps for deeper analysis. <u>Learn more</u>

Choose one of the Accelerator Analysis Types [GPU Offload is best to do first]

Performance Snapshot ③ Analysis Configuration Collection Lo	ជើ bg Summary		INTEL VTUNE PROF
Choose your next and Select a highlighted recommendation ALGORITHM Hotspots Hotspots PARALLELIS	Alysis type on based on your performance snapshot.	MICROARCHITECTURE	 ➢ Elapsed Time [©]: 0.817s ➢ CPU IPC [©]: 1.286 SP GFLOPS [©]: 0.002 DP GFLOPS [©]: 0.001 x87 GFLOPS [©]: 0.000 Average CPU Frequency [©]: 3.2 GHz ➢ GPU Time [©]: 18.6% (0.152s) ^k of Elapsed time
Threading HPC 15.3% Perform Character	i C niance rization	LT Input and Output	③ Logical Core Utilization [®] : 15.3% (1.221 out of 8) ▶
ACCELERATO	DRS	Microarchitecture Usage [☉] : 36.5% of Pipeline Slots	
GPU Offload GPU CPU/FPGA 18.6% Compute/Media Interaction Hotspots (preview)		System GPU Rendering Platform Profiler Overview (preview)	Memory Bound [⊕] : 16.5% of Pipeline Slots
Collection and Plat	form Info		Wectorization [∞] : 0.7% of Packed FP Operations
This section provides information about this collection, including result set size and collection platform data. Application Command Line: /home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU/03-run.sh			③ GPU Active Time [®] : 18.6% ▶
Operating System: Computer Name: Result Size: Collection start time: Collection stop time: Collector Type: Finalization mode: Fast. If processed samples to spo CPU	5.1.5.1.5.4.5.5.5.5.5.5.5.5.5.5.5.5.5.5.	TS"	

intel 32

GPU Offload Analysis



gpu-offload (GPU Offload Analysis)



- Explore code execution on various CPU and GPU cores on your platform.
- Correlate CPU and GPU activity.
- Identify whether your application is GPU or CPU bound.





vtune -collect gpu-offload

--app-working-dir=/home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU

-- /home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU/03-run.sh
Running a First Analysis -

Hottest CPU and GPU Tasks



Hottest Host Tasks

This section lists the most active tasks running on the host, sorted by the Task Time. Focus on performance-critical tasks first.

Host Task	Task Time 💿	% of Elapsed Time <a>>	Task Count (2)
zeModuleCreate	0.211s	7.6%	1
zeEventHostSynchronize	0.048s	1.7%	2
zeCommandListAppendMemoryCopyRegion	0.001s	0.0%	1
zeCommandQueueExecuteCommandLists	0.000s	0.0%	2
zeCommandListCreateImmediate	0.000s	0.0%	1
[Others]	0.000s	0.0%	9

*N/A is applied to non-summable metrics.

Hottest GPU Computing Tasks

This section lists the most active computing tasks running on the GPU, sorted by the Total Time. Focus on the computing tasks flagged as performance-critical.

Computing Task	Total Time (1)	Execution Time ③	% of Total Time $\ensuremath{}$	SIMD Width ③	Peak EU Threads Occupancy ③	EU Threads Occupancy ③	SIMD Utilization
Matrix1 <float></float>	0.049s	0.048s	97.1%	32	100.0%	98.8%	100.0
zeCommandListAppendBarrier N	0.000s	0s	0.0%				

*N/A is applied to non-summable metrics.

The Platform Tab

PU Offload GPU Offload	• ③		INTEL VTUNE PF	ROFILER
alysis Configuration Collectio	n Log Summary Graphics Platform			1
	s 0.2s 0.4s 0.6s 0.8s 1s 1.2s 1.4s 1.8s 1.8s 2.0665 ks 2.4s	2.6s ✓ GPU Computing Queue ✓ GPU Computing Task ✓ GPU Computing Q ✓ Thread ✓	Elapsed Time: 2.772s GPU Time, % of Elapsed time: 6.0% GPU Time, % of Elapsed time GPU Engine GPU Time GPU Time, % of Elapsed time	ତ ତ
[Unknown] matrix.dpcpp (TID: 30315)	nden un 114 Lauré Aktor Aktor (1844)	Running Running CPU Time User Tasks GPU Computing Task	Render and GPGPU 0.167s 6.0% *WA is applied to non-summable metrics.	
sh (TID: 30289) pinbin (TID: 30289)		GPU Execution Units EU Arrays C Active	Top Hotspots when GPU was idle	3
pinbin (TID: 30311)		 ✓ m Idle ✓ Stalled ✓ GPU Computing Threa 		
sh (TID: 30322)		Computing Thread		
pin (TID: 30289)		GPU Memory Access		
pinbin (TID: 30310)		✓ m Read ✓ Write		
sh (TID: 30315)		GPU Busy		
Thread (TID: 30319)		CPU Time		
sh (TID: 30321)		GPU Time		
[Unknown]		~ GPU Frequency		
ampixe-runss (TID: 30289)				
Thread (TID: 30313)				
Thread (TID: 30324)				
func@0x67da26 (TID: 30327)				
GPU Execution Units GPU Computing Threads Dis				
GPU Memory Access 46.028				
GPU Busy CPU Time				

GPU Offload 18.6%

The Graphics Tab

		littet v fulle P	Torner				
Welcome × r000ps × r001go >							
GPU Offload GPU Offload -	0						INTEL VTUNE PROFILE
Analysis Configuration Collection L	og Summary Graphic	Platform					
Grouping: GPU Computing Task / Host	Call Stack			✓ %	Call	Stacks	۲h :
		otal Time by Device Operation Typ	e V		Exe	cution (GPU Computing Task) ~	
GPU Computing Task / Host Call Stack	Alloca Host-to-Devic	e Tra Execu Device-to	-Host Tra	Instance Count	ost-to-C matr	ix.dpcpp multiply1+0x72a - multiply.cpp:42	2
Matrix1 <float></float>	49.418ms			1	matr	rix.dpcpp multiply1+0xe - multiply.cpp:163	
zeCommandListAppendBarrier	0.001ms			1	matr	rix.dpcpp ParallelMultiply+0x601 - matrix.c	pp:92
[Outside any task]				0	libc.	so.6 main+0xf2 - libc-start.c:308	
					matr	rix.dpcpp !libc_start_main+0x2d	
					matr	rix.dpcpp ! _start+0x2d	
Q: + - r	.≝ 0s 0.2s	0.4s 0.6s 0.8s 0.8	923s s 1.2s 1.4s	1.6s 1.i	ls 2s	2.2s 2.4s 2.6s	Thread
De matrix da ena (TID: 20215)	Life also also	and the state	La statistica in the statistica in the statistical statisticae statisticae statisticae statisticae sta	allowed a low build and the	Main In Lin	المعتقل المتعالمة وملته والمطعورة والمالة والمالية	Running
e matrix.dpcpp (TID: 30315)	Unit formation					28ModuleCreate 29	CPU Time
sh (TID: 30289)	NOTIN .						Spin and Overnead
pinbin (TID: 30289)							User Tasks
pinbin (TID: 30311)							GPU Computing Task
sh (TID: 30322)							GPU Execution Units
pin (TID: 30289)							EU Arrays
pinbin (TID: 30310)							
sh (TID: 30315)							🔽 📥 Stalled
Thread (TID: 30319)							GPU Computing Thread
sh (TID: 30321)							Computing Threads
[Unknown]							EU Inreads Occupa
ampixe-runss (TID: 30289)							Average Bandwidth, GB/sec
Thread (TID: 30313)							🗹 👞 Read
Thread (TID: 30324)							🔽 💼 Write
func@0x67da26 (TID: 30327)			Thread (TID: 30313)				GPU Busy
func@0x46bc76 (TID: 30328)			0.0%				GPU Busy
GPU Execution Units	1111		Spin and Overhead Time		1	1 🖉 🖞	CPU Time
GPU Computing Threads Dispatch	4.4.4.4.4	1	0.0%				Spin and Overhead
GPU Memory Access 4	8.327	MA.					GPU Frequency
GPU Busy							GPU Frequency
CPU Time							
GPU Frequency				and and a subsection of the section			×
]	
FILTER 7 100.0% - Pr	ocess Any Process	Thread Any Thread	✓ Module Any	Module	~	Call Stack Mode User functions + 1 V	

GPU Offload 18.6%

GPU Compute/Media Hotspots analysis



gpu-hotspots (GPU Compute/Media Hotspots analysis)

Explore GPU kernels with high GPU utilization, estimate the effectiveness of this utilization, identify possible reasons for stalls or low occupancy and options.

Explore the performance of your application per selected GPU metrics over time.

■ Analyze the hottest SYCL* standards or OpenCL[™] kernels for inefficient kernel code algorithms or incorrect work item configuration.

Configure Analysis

GPU Compute/Media Hotspots (preview) -

Analyze the most time-consuming GPU kernels, characterize GPU utilization identify performance issues caused by memory latency or inefficient kernel a instruction frequency per certain instruction types. Learn more

Characterization

Select a predefined GPU metric group to characterize code performance on Intel Graphics:

· Overview metrics to analyze sampler, general memory and cache accesses;

 Compute Basic metrics to track accesses to different types of GPU memory;

- Compute Extended to analyze additional metrics targeted
- only for Intel processor code name Broadwell and higher; Full Compute to combine Overview and Compute Basic event

Ins

- Dynamic Instruction Count to analyze GPU instruction
- frequency per certain instruction types.

Enter computing task of interest

Characterization ③ 0 J Overview Overview Compute Basic (with global/local memory accesses) **Full Compute Dynamic Instruction Count** רוומוצבט שטחטו ששמעש Source Analysis ③ Instance step



Running a GPU Compute/Media Hotspots analysis Characterization: Overview

vtune -collect gpu-hotspots

- --app-working-dir=/home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU
- -- /home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU/03-run.sh



Running a GPU Compute/Media Hotspots analysis Characterization: Compute Basic

vtune -collect gpu-hotspots

-knob characterization-mode=global-local-accesses

- --app-working-dir=/home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU
- -- /home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU/03-run.sh



Running a GPU Compute/Media Hotspots analysis Characterization: Full Compute

vtune -collect gpu-hotspots

-knob characterization-mode=full-compute

- --app-working-dir=/home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU
- -- /home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU/03-run.sh



0

intel⁴⁷

Running a GPU Compute/Media Hotspots analysis Characterization: Dynamic Instruction Count

vtune -collect **gpu-hotspots**

-knob characterization-mode=instruction-count

--app-working-dir=/home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU

-- /home/stephen/dv/LRZ-DEMOS/2-VTUNE-GPU/03-run.sh

GPU Hotspots - Summary

GPU Compute/Media Hotspots (preview)

Selapsed Time[™]: 2.912s

Hottest GPU Computing Tasks

This section lists the most active computing tasks running on the GPU, sorted by the Total Time.

Computing Task	Total Time 💿	Average Time 💿	Instance Count ③	GPU Instructions Executed
Matrix1 <float></float>	0.073s	0.073s	1	917,635,072

*N/A is applied to non-summable metrics.

Graphics Tab Instruction count – by Type

<i>a</i> .	recome a roops a roongo a roozyn a														
	GPU Compute/Media Hotspots (preview) ③													INTEL V	TUNE PROFILER
	Analysis Configuration Collection Log Summary Graphics														
_	O: + - F = 0 0.24 0.44	0.65	0.85		1.26	1.46	1.05	1.84	28	2.28	2.45	2.64	2.86	C Thread	~
8	matrix.dpcpp (TID: 31796)													Runn	ing Tasks
												and the second second		Com 🖂 Com	puting Task
~															
	rouping Computing Task / Function / Call Stack														V (K) (L)
	rouping Computing Task / Function / Call Stack	10			Computing Task			Data Tra		GPU Inst	ructions Exec	uted by Instruct	ion Type	(2)	▼ <u>≮</u> Q 2
c	rouping: Computing Task / Function / Cail Stack Computing Task / Function / Cail Stack	te Local	Total Time	Average Time	Computing Tass	k SIMD Width	SVM Usage Type	Data Tra Size	Control Fl	GPU Inst Se § Synci	ructions Exec	uted by instruct	ion Type <mark>I</mark> Int32 & SP Fi		♥ SIMD Utilization
	rouping Computing Task / Function / Cail Stack Computing Task / Function / Cail Stack	e Local 512 x 1	Total Time 72.933ms	Average Time 72.933ms	Computing Task Instance Count	k SIMD Width 32	SVM Usage Type	Data Tra Size	Control Fl 917,635,072	GPU Inst Se Synci	ructions Exec	uted by Instruct	ion Type Ind2 & SP FI		v x Q € SIMD Utilization 93.09
	rouping Computing Task / Function / Call Stack Computing Task / Function / Call Stack	te Local 512 x 1	Total Time 72.933ms	Average Time 72.933ms	Computing Task Instance Count	k SIMD Width 32	SVM Usage Type	Data Tra Size	Control Fl 917.635.072 3.112.960 262.144	GPU inst Se Synci	ructions Exec	uted by Instruct	Ion Type Int32 & SP Fi	() OL	♥ 🛠 Q 🕫 SIMD Utilization 99.09 100.05
	rouping Computing Task / Function / Call Stack Computing Task / Function / Call Stack	te Local 512 x 1	Total Time 72.933ms	Average Time 72.933ms	Computing Tass Instance Count	k SIMD Width 32	SVM Usage Type	Data Tra Size 0 B	Control Fl 917,535,072 3,112,960 262,144 524,248	GPU inst Se Synd	ructions Exec	uted by Instruct	ion Type Imi32 & SP FL	OL	✓ ≤ Q 5 SIMD Utilization 99.99 100.07 100.07 100.07
	rouping Computing Task / Function / Call Stack Computing Task / Function / Call Stack Metal Science > Market Science > 2Market 3/ Vielentiat Stain, Joog, ImpULI.mREI.mTEEZPASSD, Bacc > 2Market 3/ Vielentiat Stain, Joog, ImpULI.mREI.mTEEZPASSD, Bacc > 2Market 3/ Vielentiat Stain, Joog, ImpULI.mREI.mTEEZPASSD, Bacc	te Local 512 x 1	Total Time 72.933ms	Average Time 72.933ms	Computing Tass Instance Count	k SIMD Width 32	SVM Usage Type	Deta Tra Size 0 B	Control Fl 917, 635,072 3,112,960 262,144 524,283 333,216	GPU inst Se Synd	ructions Exec	uled by Instruct	ion Type Int32 & SP FI		▼ (<) (<) SIMD Utilization 99 99 100.07 100.07 100.07 100.07
	rouping Computing Task / Function / Call Stack Computing Task / Function / Call Stack Manar Column > Market Andread > Delayday / Medical Talam, Joog, Impd.J.am6ELm1EED2000; Bark > Delayday / Medical Talam, Joog / Jong Stack / Delayday / Delayd	te Local S12 x 1	Total Time 72.933ms	Average Time 72.933ms	Computing Tass Instance Count	k SIMD Width 32	SVM Usage Type	Data Tra Size 0 B	Control Fl 917.635.072 3.112.960 262,144 524,288 333.216 912.588.800	GPU Inst ∎ Se ∎ Synd	ructions Exec	uled by Instruct	ion Type Int32 & SP Fi	2 0L	▼(%) Q) 5 SIMD Utilization 100.07 100.07 100.07 100.07 99.99
	couping Computing Task / Function / Call Stack Computing Task / Function / Call Stack Metal Status > Market Scheduch > ZMAyed Sylvekatett Statin, Joog. Jmpd.L.mcRcIm (EEZPASS), Baco > ZMAyed SylVekatett Statin, Joog Jmpd.L.mcRcIm (EEZPASS), Baco > ZMAyed SylVekatett Statin, Joog Jmpd.L.mcRcIm (EEZPASS), Baco > ZMAyed SylVekatett Statin, Joog Jmpd.L.mcRcIm (EEZPASS), Baco	te Local 512 x 1	Total Time 72.953ms	Average Time 72.933ms	Computing Task Instance Count	k SIMD Wridth 32	SVM Usage Type	Data Tra Size 0 B	Control Fl 917, 635,072 3,112,960 262,144 524,288 393,216 912,588,800 753,664	GPU Inst § Se § Synd	ructions Exec hronizati	uted by Instructor	Jon Type Int32 & SP Fi		▼ ≪ Q == SIMD Utilization 100.07 100.07 100.07 100.07 100.07 100.07
	rouping Computing Task / Function / Call Stack Computing Task / Function / Call Stack Hours Folgue > Matrix House > ZMerged 3/Hosteal138m, loop_impl.LmGELm15EZPM30, Bacc > ZMerged 3/Hosteal138m, loop_impl.LmGELm15EZPM30, Bacc	te Local 512 x 1	Total Time 72.933ms 0.288ms	Average Time 72.033ms 0.288ms	Computing Tass Instance Count	k SIMD Wridth S22	SVM Usage Type	Data Tra Size 0 B	Control FL 917.655.072 262,144 524,288 393,216 912,588,800 753,664	GPU Inst § Se Ø Synd	ructions Exec hronizati	uled by Instruct	lon Type ∎int32 & SP Fi		▼ ≪ Q E SIMD Utilization 100.05 100.05 100.05 100.05 100.05

		GPU Instruction kernel shown in Control Flow Send Synchroniza	tion	a breakdo owing cat	wn of ir egories:	struction	s executed t	y the			
		Other	loat					-		~ *	2 0
		- Contractor Contractor		BPU Instruc	tions Ex	ecuted by	nstruction Typ	0	CL.	SIMD Utiliz	ation
T	Data Tra			Synchro	nizati	Int16 8	HP FL	RJZ a or ri			0.00
Te	Size	Control Fl	Se	Synchro	onizati	Int16 8	HP FL	R52 a 5F F		10	99.9%
T	Data Tra Size 0 B	Control Fl 917,635,072	Se	Synchro	onizati	Int16 8	HPFL. 1	11.52 a 51 T		10	0.0%
T	Data Tra Size 0 B	Control Fl 917,635,072 3,112,960	Se	Synchro	onizati	Int16 8	HP FL	11.52 a or 11		10	0.0%
I	Data Tra Size 0 B	Control Fl 917,635,072 3,112,960 262,144	Se	Synchro	onizati	Int16 8	HP FL	102 0 01 11		10 10 10 10	0.0% 0.0% 0.0% 0.0%
I	Data Tra Size 0 B	Control Fl 917,635,072 3,112,960 262,144 524,288	Se	Synchro	onizati	Int16 8	HP FL.	102 0 01 1		10 10 10 10 10 9	0.0% 0.0% 0.0% 0.0% 0.0%
I	Data Tra Size 0 B	Control Fl 917,635,072 3,112,960 262,144 524,288 393,216	8 Se	Synchro	onizati	Inti6 8	HP PL.			10 10 10 10 9 10	0.0% 0.0% 0.0% 0.0% 9.9% 0.0%
F	Data Tra Size 0 B	Control Fl 917,635,072 3,112,960 262,144 524,288 393,216 912,588,800	8 Se	Synchro	onizati	Intife 8	HP FL.			10 10 10 10 9 10	0.0% 0.0% 0.0% 0.0% 9.9% 0.0%
T	Data Tra Size 0 B	Control Fl 917,635,072 3,112,960 262,144 524,288 393,216 912,588,800 753,664	0 Se	Synchro	onizati	Intife 8	HP FL.			10 10 10 10 9 10	0.0% 0.0% 0.0% 0.0% 9.9% 0.0%
T	Data Tra Size 0 B	Control Fl 917,635,072 3,112,960 262,144 524,288 393,216 912,588,800 753,664	Se	Synchro	onizati	Intife 8				10 10 10 10 9 10	0.0% 0.0% 0.0% 0.0% 0.0%

GPU Roofline Analysis with Intel[®] Advisor



Roofline Summary (GPU)

Perspective: GPU Roofline Ins	sights - GPU	Roofline Regions								٩ ٩	3
Program Metrics											^
4.71s Program Elapsed Time		3.55s GPU Time			0.00s Data Transfer Time			1.16s CPU Time			
O GPU							CPU				
GFLOPS: 0.04		GINTOPS: 0.01		GTI Bandwidth: 1	.23 GB/s		GFLOPS: 0.11		GINTOPS: 0.06		
GFLOP: 0.13	FP AI (GTI): 0.03	GINTOP: 0.04	INT AI (GTI): 0.01	GTI Traffic: 4.35 GB			GFLOP: 0.13	FP AI: 0.64	GINTOP: 0.08	INT AI: 0	J.31
FPU Utilization: 86.2%		EU Threading Occupancy: 93.8%		EU IPC Rate: 1.8	9		Thread Count: 2				
											~
GPU							CPU				
ROOFLINE	0.1 the GTI (Memory) Bandwidth: [©]	1.23 1% of 76.00 GB/sec	1		DP Vecto	ELOAT INT r Add Peak: 57.58 GFLOPS LOP/Byte (Arithmetic Intensity)	ROOFLINE	? GB/sec ? 3 GB/sec ? 8 3.17 GB/sec ? 8 0.1 0.1	Int32 Vect Integer Sca compute and memory root 1 OPS Scalar Add Peak	FLOAT or Add Peak: 17.2.47 GINTO alar Add Peak: 17.71 GINTO Co U S? INTOP/Byte (Arithmetic Inter	PS PS PS mp te und ? nsity) 10
Top Hotspots											^
GPU Kernel ③ mandelbrot_kernel <float> mandelbrot_kernel<double></double></float>	Elapsed Time ③ 3.54s <0.01s	GFLOPS ⑦ 0 17.561	GINTOPS ()) 5.871	Global/Local ③ 16 x 50 x 600/16 x 2 16 x 50 x 600/16 x 2	Active/ 2 × 8 97.0/1.9 2 × 8 98.3/0.7	Stalled/Idle, % ③ /1.1 /1.1	CPU Function Call Sites (loop in verify at main [loop in tcf 0] [loop in GLOBAL [loop in khrlcdVendor [loop in khrlcdVsven	Self Elapsed Time Image: Organization 0.04s 0.01s 0.01s <0.01s 0s 0s	Self GFLOPS ③ 3.251 0 0 0 0 0	Self GINTOPS ③ 1.200 0 0 0 0 0 0 0 0 0	
Platform Information				\sim	Collection Informat	ion					~
ii Darfamanaa Charastaistiss											<u>^</u>
Performance Characteristics GPU							CPU				$\overline{\mathbf{v}}$
EU Array Active / EU Array Stalled /	EU Arrav Idle					97 0% / 1 9% / 1 1%	Time in 1 Vectorized Loops			0	0.01s

51

el

GPU Roofline differences to CPU Roofline

- The dots on the chart correspond to *OpenCL, OpenMP, Level Zero and SYCL kernels*, while in the CPU version, they correspond to individual loops.
- Some displayed information and controls (for example, thread/core count) are not relevant to GPU Roofline. For more information, see the table below.
- The GPU Roofline chart enables you to view arithmetic intensity of one kernel at multiple memory levels. To do so, double-click a dot representing this kernel or select it and press ENTER. The dots that appear on the Roofline chart correspond to different memory levels used to calculate arithmetic intensity. Hover over a dot to identify its arithmetic intensity. To show or hide certain dots from a chart, use the Memory Level drop-down filter.

A GPU Roofline



Cookbook Example

- Intel[®] Advisor Performance Optimization Cookbook
- Analyze a SYCL Application with GPU Roofline (intel.com)
 - 1. Prerequisites.
 - 2. Run GPU Roofline Insights perspective.
 - 3. View GPU Roofline results.
 - 4. Examine the Application Performance on GPU.
 - 5. Explore detailed GPU metrics with Intel Advisor Python* API.
 - 6. Alternative steps.

Example Results: https://cdrdv2.intel.com/v1/dl/getContent/720943

Lammps offloading

Stephen Blair-Chappell

stephen AT sbcnow DOT co DOT uk

Test Application

LAMMPS Molecular Dynamics Simulator



https://www.lammps.org/

Optimize Your GPU Application with the Intel® oneAPI Base Toolkit

The Workflow

Step 1: Choose Your GPU Hardware Access.

Step 2: Choose Your Sample Code.

Step 3: Assess Code for Offload Opportunities with Intel® Advisor.

Step 4: Offload and Optimize Code Using Intel® Compilers and Libraries.

Step 5: Evaluate Offload Efficiency with Intel Advisor.

Step 6: Review Overall Application Performance with Intel[®] VTune[™] Profiler.



https://www.intel.com/content/www/us/en/developer/tools/oneapi/training/gpu-optimization-workflow.html

Machine 1 Spec - Workstation

58345@s011-n001:~/ILDevCON/lamm	ps-offload-OneAPI/LAB3/1-Model-Offload\$ lscpu
rchitecture:	x86_64
PU op-mode(s):	32-bit, 64-bit
yte Order:	Little Endian
ddress sizes:	46 bits physical, 48 bits virtual
PU(s):	24
n-line CPU(s) list:	0-23
hread(s) per core:	2
ore(s) per socket:	12
ocket(s):	1
UMA node(s):	1
endor ID:	GenuineIntel
PU family:	6
odel:	85
odel name:	Intel(R) Core(TM) 19-10920X CPU @ 3.50GHz
tepping:	7
PU MHz:	1200.315
PU max MHz:	4800.0000
PU min MHz:	1200.0000
ogoMIPS:	6999.82
'irtualization:	VT-x
1d cache:	384 KIB
1i cache:	384 KIB
2 cache:	12 MiB
3 cache:	19.3 MiB
IUMA node0 CPU(s):	0-23
ulnerability Itlb multihit:	KVM: Mitigation: Split huge pages
ulnerability L1tf:	Not affected
ulnerability Mds:	Not affected
ulnerability Meltdown:	Not affected
ulnerability Spec store bypass:	Mitigation; Speculative Store Bypass disabled via prctl and seccomp
ulnerability Spectre v1:	Mitigation; usercopy/swapgs barriers and user pointer sanitization
ulnerability Spectre v2:	Mitigation; Enhanced IBRS, IBPB conditional, RSB filling
ulnerability Srbds:	Not affected
ulnerability Tsx async abort:	Mitigation; TSX disabled
lags:	fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acp
	i mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_per
	fmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64
	monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic
	movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cp
	uid fault epb cat 13 cdp 13 invpcid single ssbd mba ibrs ibpb stibp ibrs enhanced tpr s
	hadow vnmi flexpriority ept vpid ept ad fsgsbase tsc adjust bmi1 avx2 smep bmi2 erms in
	vpcid cgm mpx rdt a avx512f avx512dg rdseed adx smap clflushopt clwb intel pt avx512cd
	avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves cgm llc cgm occup llc cgm mbm total cg
	m mbm local dtherm ida arat pln pts hwp hwp act window hwp epp hwp pkg reg avx512 vnni
	red close fluck lid onch comphilition

Intel(R) Core(TM) i9-10920X CPU @ 3.50GHz

CPU(s):	24
On-line CPU(s) list:	0-23
Thread(s) per core:	2



Machine 2 Spec - Laptop

	Dititles
stephen@stephen-Swift-SF314-510G	:~/dv/OneAPI-Package-1/LAMMPS\$ lscpu
Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
Byte Order:	Little Endian
Address sizes:	39 bits physical, 48 bits virtual
CPU(s):	8
On-line CPU(s) list:	0-7
Thread(s) per core:	2
Core(s) per socket:	4
Socket(s):	1
NUMA node(s):	1
Vendor ID:	GenuineIntel
CPU family:	6
Model:	140
Model name:	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz

u58345@s011-n001:~/ILDevCON/lammps-offload-OneAPI/LAB3/1-Model-Offload\$ lspci | grep VGA 1c:00.0 VGA compatible controller: Intel Corporation Device 4905 (rev 01) 6a:00.0 VGA compatible controller: Intel Corporation Device 4905 (rev 01)

Graphics processor table					
PCI IDs	Name	Architecture	Codename		
4905	Intel® Iris® Xe MAX Graphics	Xe	DG1		





Analyze







https://www.intel.com/content/www/us/en/developer/tools/oneapi/gpu-optimization-workflow.html

Goal of Analysis





1. Find the 'hot spots'

2. if possible, predict benefit of offloading

Three Tools

APS – (Application Performance Snapshot)

Advisor - model offloading

VTune - Profiler

 \leftarrow

 \rightarrow С ① File C:/Users/STEPHE~1/ONEDRI~1/DOCUME~1/MobaXterm/slash/stephenblaircha_desktopikul9t5/RemoteFiles/133800_2_1/aps_report_20210923_065855.html

APS – (Application Performance Snapshot)

Tuning Potential

Application Performance Snapshot

× +



()

DP GFLOPS

SP GFLOPS

1.43 2.88 GHz

MPI Sendrecv

Average CPU Frequency

0.06 s gpu

Elapsed Time ------

GPU	Utilizat	ion w	hen	Busy
3 696				

EU State	% of EUs
Active	3.6%
Idle	83%
Stalled	13.4%

IPC Rate

9.1% of Peak Value

Memory Footprint

Resident 113.19 MB

Resident per Node 1811 MB

Virtual 2646.12 MB

Virtual Per Node 42338 MB

MPI Time 4.3 så 54.64% ▲ of Elapsed Time MPI Imbalance 2.38 s 30.28% ▶ of Elapsed Time TOP 5 MPI Functions % of Elapsed MPI Bcast MPI Wait MPI_Allreduce MPI_Init

	Memory Stalls 7.2% of Pipeline Slots
	Cache Stalls 19.2% of Cycles
	DRAM Stalls 7.5% of Cycles
psed Time	DRAM Bandwidth
28.84%	
9.86%	Average
7.070/	Peak
7.87%	Bound
5.4%∿	
1,69%	

Your application may underutilize the GPU.

MPI Time

Memory Stalls

Vectorization

GPU Utilization when Busy

Current run

54.64% ▲ <10% 7.2%

Target

<20%

0% ► >70%

3.6% > 80%

Vectorization 0% Instruction Mix SP FLOPs 0% of uOps DP FLOPs 6.9% of uOps Packed: 9 GB/s 0% from DP FP 8.9 GB/s 128-bit: 0% 0% 256-bit: 0% 512-bit: 0% Scalar: 100% from DP FP Non-FP 93.1% of uOps FP Arith/Mem Rd Instr. Ratio 0.21

Run a GPU Offload (Preview) or a GPU Compute/Media Hotspots (Preview) analysis with VTune Profiler to discover how to better utilize the GPU.

FP Arith/Mem Wr Instr. Ratio 0.73

LAB1

×



Steps to Offload Projection with Advisor

1. Run a **Survey**: get a list of hotspots

advisor -collect survey ...

- SamplingBinary Static Analysis
- Compiler & debug info
- 2. Run a **Trip Count**: count loop iteration Trip count

advisor -collect=tripcounts -target-device=gen9_gt2

- Cache simulation
- 3. Perform a dependency analysis [optional for quick modelling]

advisor -collect dependencies . . .

4. Model the Performance

advisor -collect projection -no-assume-dependencies . . .

Generate HTML report

Check memory accesses

Loop selection heuristic

Expensive Steps









https://www.intel.com/content/www/us/en/developer/tools/oneapi/gpu-optimization-workflow.html

intel. 6

TWO TYPES OF COMPILER



Deprecated and will be removed from product release in the second half of 2023



ifort

Offloading not supported

LLVM based [Totally new compilers]

- icx
 offloading supported
- ifx

oading ported All objects are binary compatible

One approach . . .

NB: icc deprecated mid 2023



... or alternatively



GPU Architecture




73

⁷³intel.

OpenMP GPU Offload and OpenMP Constructs

- OpenMP GPU offload support all "normal" OpenMP constructs
 - E.g. parallel, for/do, barrier, sections, tasks, etc.
 - Not every construct will be useful

- Full threading model outside of a single GPU subslice **not** supported
 - No synchronization among subslices
 - No coherence and memory fence between among subslice L1 caches

intel



⁷⁵intel.



#pragma omp target teams distribute parallel for

Distributes loop iterations across all EUs

76

76

⁷⁶intel.



```
Experiment in Lammps - Target
```

```
#if 1
     #pragma omp target map(to:ilist[iifrom:iito]) \
                        map(to:x[x min:x max]) \
                        map(tofrom:f[f_min:f max]) \
                        map(from:ev global[0:7]) \
                        map (tofrom:lj1,lj2,lj3,lj4,offset)
     #pragma omp teams distribute parallel for
 #endif
       ----- END SBC code ------
      for (int ii = iifrom; ii < iito; ii += iip) {</pre>
       const int i = ilist[ii];
       int itype, ptr off;
       const FC PACKED1 T * noalias ljc12oi;
       const FC PACKED2 T * noalias lj34i;
       if (!ONETYPE) {
         itype = x[i].w;
         ntr off = itvne * ntvnes:
```

78

intel

Experiment in Lammps –Indirect Indexes

Start here	*pair_lj_cut_intel.cpp 🗵	
222	// SBC code	
223	#if 1	
224	<pre>// find min an max of indirect indexes</pre>	
225	<pre>int i_min=0;</pre>	
226	<pre>int i max=0;</pre>	
227	<pre>int jlist min=0;</pre>	
228	<pre>int jlist max=0;</pre>	
229	<pre>int j min =0;</pre>	
230	<pre>int j max=0;</pre>	
231	<pre>int x min=0;</pre>	
232	<pre>int x max=0;</pre>	
233	<pre>int f min=0;</pre>	
234	<pre>int f max=0;</pre>	
235		
236	<pre>for (int ii = iifrom; ii < iito; ii += iip) {</pre>	
237	<pre>int i=0, itype=0, sbindex=0;</pre>	
238		
239	// get i min and i max	
240	i = ilist[ii]:	
241	<pre>i min=std::min(i.i min):</pre>	
242	i max=std::max(i,i max):	
243		
244		
245	<pre>const int * noalias const ilist = firstneigh[i];</pre>	
246	int inum = numneigh[i]:	
247		
247	// we also need the i min and i may as this is use	lily ni be
249	for (int ii = 0 · ii < inum · ii++) {	a in all'i
250	int i=0, itype=0, shindex=0.	
251 -	if (IONETYPE) {	
252	shinder = ilist[ii] >> SBBITS & 3	
252	i = ilist[ii] & NETGHMASK	
254	- } else	
255	i = ilist[ii].	
255] =](13([)]),	
257	i min=std::min(i i min):	
258	i max=ctd::max(i i max):	
250	l	
255)) // ii	
261	$y = f = f = ctd \cdot min(i = min)$	
262	$x = f = f = ctd \cdot max(i = min),$	
202	<pre>x_max = 1_max = stamax(1_max, j_max),</pre>	
203	#if 0	
204	#11 0	
205	<pre>print(X_MILT: %U, X_MIX: %U ,X_MILT,X_MIX); #ondif</pre>	
200	#CHULI #ondif	
207	#CHOTI	
200	<i>μ</i> ί <i>ξ</i> η	
209	#1 1 Hereans and torget men(to,ilist[iifram.iitel))	
2/0	<pre>#pragma omp target map(to:llist[litrom:llt0]) \</pre>	

```
// we also need the j_min and j_max as this
 for (int jj = 0; jj < jnum; jj++) {</pre>
   int j=0, jtype=0, sbindex=0;
   if (!ONETYPE) {
     sbindex = jlist[jj] >> SBBITS & 3;
     j = jlist[jj] & NEIGHMASK;
    } else
      j = jlist[jj];
    j min=std::min(j,j min);
    i max=std::max(j,j_max);
} // ii
x min = f min = std::min(i min,j min);
x max = f max = std::max(i max, j max);
```

intel

Debug











https://www.intel.com/content/www/us/en/developer/tools/oneapi/gpu-optimization-workflow.html

oneAPI Debug Tools (intel.com)

Tool	Description
Environment variables	Gather diagnostic information from the OpenMP and SYCL runtimes at program execution with no modifications to your program.
Onetrace	From Profiling Tools Interfaces for GPU (PTI for GPU). Used to debug backend errors and for performance profiling on both the host and device.
Intercept Layer for OpenCL™ Applications	Used to debug backend errors and for performance profiling on both the host and device (has wider functionality comparing with onetrace).
Intel [®] Distribution for GDB*	Used for source-level debugging of the application, typically to inspect logical bugs, on the host and any devices you are using (CPU, GPU, FPGA emulation).
Intel [®] Inspector	locate and debug memory and threading problems, including those that can cause offloading to fail.
In-application debugging	printf; looking at output of apps etc, etc
Intel [®] Advisor	Use to ensure Fortran, C, C++, OpenCL™, and SYCL applications realize full performance potential on modern processors.
Intel [®] VTune Profiler	Use to gather performance data either on the native system or on a remote system.

Example – offload on Laptop

BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
 RANK 5 PID 23953 RUNNING AT stephen-Swift-SF314-510G
 KILLED BY SIGNAL: 9 (Killed)

Runtime FAILS

Get Visibility

export LIBOMPTARGET_DEBUG=4
export LIBOMPTARGET_INFO=-1

error: double type is not supported on this platform in file: pair_lj_cut_intel.cpp:311

Get Visibility

export OverrideDefaultFP64Settings=1 export IGC_EnableDREmulation=1

The Solution

Tune









https://www.intel.com/content/www/us/en/developer/tools/oneapi/gpu-optimization-workflow.html

Run and collect VTune[™] data

vtune -collect gpu_hotspots -result-dir vtune_data a.out

Various types of profiling data can be collected like hotspots, memory-consumption, memory-access, threading ...

Use the command line help to find out more:

vtune --help -collect

Generate html report for collected VTune[™] data:

vtune -report summary -result-dir vtune_data -format html -report-output
\$(pwd)/summary.html

Various types of report can be generated like summary, top-down, callstacks ...

VTune Results - 1



- Reduced Core Utilization

🕑 Effective Physical Core Utilization 🤆 84.0% (6.723 out of 8) 临

Effective Logical Core Utilization [®]: **57.5%** (9.197 out **b** 16) **№** MPI Imbalance [®]: 0.003s (0.0%)

⊙ MPI Rank on the Critical Path ^③: [™]

MPI Busy Wait Time : 0.001s (0.0%)

- ⊙ Serial Time (outside parallel regions) ©: 4.675s (72.8%) 🕅 🐚
 - 💿 Top Serial Hotspots (outside parallel regions) 临

This section lists the loops and functions executed serially in the master thread outside of any OpenMF hotspot functions. Since the Serial Time metric includes the Wait time of the master thread, it may sign

Function	Module	Serial CPU Time 🛛
module_get_kallsym	vmlinux	0.534s 🎙
[Loop at line 76 in LAMMPS_NS::FixNVEIntel::initial_integrate]	Imp	0.259s 🏲
[Loop at line 633 in LAMMPS_NS::FixIntel::reduce_results <double>]</double>	Imp	0.226s 🏲
[Loop at line 844 in .omp_outlineddebug35]		0.195s 🏲
[Loop at line 2238 in LAMMPS_NS::Neighbor::check_distance]		0.191s 🎙
[Others]	N/A*	2.705s 🏲

*N/A is applied to non-summable metrics.

② Parallel Region Time ^③: 1.744s (27.2%) ¹/₁

💿 Effective CPU Utilization Histogram 🐚

SPU Utilization when Busy [©] (; 10.5%)

⊘ EU State [⊙]:

Active : 10.5%

Stalled : 18.0%

Idle 🛛: 71.6% 🎙

Occupancy [®]: 23.9% ► of peak value

🕑 Effective Physical Core Utilization 🤆 69.6% (5.566 out of 8) 🎙 🐚

Effective Logical Core Utilization 1: 35.0% (5.604 out of 16)

💿 Effective CPU Utilization Histogram 临

 ✓ GPU Utilization when Busy[®]: 27.9%
 ✓ EU State[®]: Active[®]: 27.9% Stalled[®]: 36.7% ℝ Idle[®]: 35.4% ℝ

Occupancy [©]: 53.1% ▶ of peak value

INCREASED GPU Utilization when busy

QUESTIONS?



Notices & Disclaimers

Performance varies by use, configuration, and other factors. Learn more at <u>www.Intel.com/PerformanceIndex</u>.

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.

Your costs and results may vary.

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

© 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.