

# INTEL® VTUNE™ PROFILER A QUICK INTRO FOR HPC DEVELOPERS

Vladimir Tsymbal Rev. 09/2020

## Agenda

- Intel<sup>®</sup> VTune<sup>™</sup> Profiler 2020 analysis types overview
- Basic analysis: Performance Snapshot and Hotspots
- HPC Performance Characterization analysis for hybrid MPI + OpenMP apps
- How to run VTune Profiler on MPI applications
- Memory Access analysis
- Accelerators (GPU and FPGA) analysis



## Intel® VTune™ Profiler 2020 Analysis Types



### Start with

- Command line
- Within a GUI

### Which type of analysis

- Performance Snapshot for initial assessment
- With little experience you know which one to use

### Simplest command line

- vtune -collect performance-snapshot ./my\_app
- Almost every GUI functionality is available in CL

Launching analysis in a cluster environment

The same CL, but wrapped into a mpi launcher commands

#### **Optimization Notice**



## VTune Performance Snapshot



\*Other names and brands may be claimed as the property of others

intel

## Performance Snapshot report in CL

vtune -collect performance-snapshot ./my\_app

Memory Bound: 77.5% of Pipeline Slots
The metric value is high. This can indicate that the significant fraction of
execution pipeline slots could be stalled due to demand memory load and
stores. Use Memory Access analysis to have the metric breakdown by memory
hierarchy, memory bandwidth information, correlation by memory objects.
L1 Bound: 0.0% of Clockticks
L2 Bound: 0.0% of Clockticks
L3 Bound: 60.2% of Clockticks
This metric shows how often CPU was stalled on L3 cache, or contended
with a sibling Core. Avoiding cache misses (L2 misses/L3 hits) improves
the latency and increases performance.
DRAM Bound: 13.1% of Clockticks
$\mid$ This metric shows how often CPU was stalled on the main memory (DRAM).
Caching typically improves the latency and increases performance.
DRAM Bandwidth Bound: 0.0% of Elapsed Time
Store Bound: 0.0% of Clockticks
NUMA: % of Remote Accesses: 20.9%
A significant amount of DRAM loads were serviced from remote DRAM.
Wherever possible, try to consistently use data on the same core, or at
least the same package, as it was allocated on.

#### Optimization Notice



## **HPC Performance Characterization Analysis**

### Show important aspects of application performance in one analysis

 Entry point to assess application efficiency with definition of the next steps to investigate events with significant performance cost

### Threading: CPU Utilization

- Serial vs. Parallel time
- Top OpenMP regions by potential gain
- Tip: Use Hotspot OpenMP region analysis for more detail

### Memory Access Efficiency

- Stalls by memory hierarchy
- Bandwidth utilization
- Tip: Use Memory Access analysis

### Vectorization: FPU Utilization

- FLOPS † estimates from sampling
- Tip: Use Intel Advisor for precise metrics and vectorization optimization

#### >vtune -collect hpc-performance -data-limit=0 -r result\_dir ./my\_app

HPC Performance Chara	cterizatio	n	
🕘 Å Analysis Configuration 🔲 Co	ollection Log	🕯 Summary	🗞 Bottom-up
Elapsed Time <sup>(2)</sup> : 10.25	3s		
SP GFLOPS <sup>3</sup> : 129.32	5		
Seffective Physical Core	Utilizatio	on <sup>@</sup> : 52.7	% (23.181 out of 44) 🏲
Serial Time (outside parallel res	on : 52.3% (4	6.012 out of 88	
Parallel Region Time <sup>(2)</sup> : 10.116	(98.7%)	3 (1.376)	
Estimated Ideal Time <sup>(2)</sup> :	5.623s (54.8%	6)	
OpenMP Potential Gain <sup>(2)</sup> :	4.493s (43.89	(6) 🏲	
Top OpenMP Regions by Performance Perfo	otential Gain		
③ Effective CPU Utilization Histor	ogram		
Memory Bound <sup>®</sup> : 21. <sup>9</sup>	9% <b>►</b> of I	Pipeline Slo	ots
Cache Bound <sup>(1)</sup> :	25.1%	of Clockticks	
ORAM Bound <sup>®</sup> :	6.7%	of Clockticks	
NUMA: % of Remote Accesses	s <sup>@</sup> : 43.3%		
Bandwidth Utilization Histogram	am		
	% 🏲 街		
SP FLOPs per Cycle .	0.505 Out (	¥ 22	
Vector Capacity Usage <sup>(2)</sup> :	A significan	at fraction of floati	2
Section Mix: Control Contro	point arithr	metic instructions a	are
	scalar. Use possible rea not vectori	Intel Advisor to se asons why the cod zed.	e e was
% of Scalar FP Instr. <sup>(1)</sup> :	88.9% 2		
FP Arith/Mem Rd Instr. Ratio	D: 0.862		
FP Arith/Mem Wr Instr. Ratio	<sup>D</sup> : 2.459		
Ton Loons/Eunctions with EPI	Illeage by CPI	ITimo	

#### **Optimization Notice**

#### Optimization Notice Copyright © 2020, Intel Corporation. All rights reserved.

Copyright © 2020, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others.

## Performance Aspects: CPU Utilization (1/4)

### **CPU Utilization**

- % of "Effective" CPU usage by the application under profiling (threshold 90%)
  - Under assumption that the app should use all available logical cores on a node
  - Subtracting spin/overhead time spent in MPI and threading runtimes

### Metrics in CPU utilization section

- Average CPU usage
- Intel OpenMP scalability metrics impacting effective CPU utilization
- CPU utilization histogram

V			
HPC Performance Characterization HPC Performance Characterization v	iewpoint ( <u>change</u> )	0	INTEL VTUNE AMPLIFIER XE 2
🗷 Collection Log 🛛 🖶 Analysis Target 🙏 Analysis Type 📓 Summary 😽 Bottom-up			
CPU Utilization <sup>®</sup> : 76.4% Average CPU Usage <sup>®</sup> : 18.344 Out of 24 logical CPUs Serial Time <sup>®</sup> : 0.021s (0.3%) ♥ Parallel Region Time <sup>®</sup> : 7.745s (99.7%) Estimated Ideal Time <sup>®</sup> : 6.413s (82.2%) OpenMP Potential Gain <sup>®</sup> : 1.371s (17.6%) ♥ Top OpenMP Regions by Potential Gain This section lists OpenMP regions with the highest potential for performance improvement. The Potential	Gain metric shows the elap	osed time ti	hat could be saved if the region
was optimized to have no toad initiatance assuming no runtime overhead. OpenMP Region	OpenMP Potential Gain ∅	(%) 💿	OpenMP Region Time
conj_grad_\$omp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:517:695	1.308s 🎙	16.8% 🎙	7.526s
MAIN\$omp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:189:235	0.046s	0.6%	0.240s
MAIN\$omp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cq.f:365:369	0.016s	0.2%	0.017s
MAIN\$omp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:343:349	0.001s	0.0%	0.001s
MAIN\$omp\$parallel:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:267:273	0.000s	0.0%	0.000s
[Others]	N/A*	N/A*	0.000s

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

#### OPU Usage Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU usage value.





SP GFLOPS <sup>©</sup>: 14.041 → CPU Utilization <sup>©</sup>: 76.4% → Memory Bound <sup>©</sup>: 63.2% → FPU Utilization <sup>©</sup>: 1.3% ►

## Performance Aspects: CPU Utilization (2/4)

VTune Profiler OpenMP\* Analysis: answering on customers' questions about performance in the same language a program was written in





## Performance Aspects: CPU Utilization (3/4)

### Details in grid view: expansion by constructs with a barrier inside the region

### Imbalance distribution by loop, single, reduction, user, join barriers

Advanced Hotspots Hotspots viewpoint ( <u>change</u> ) Ø										Intel \	/Tune	Ampl	ifier XE	20		
📧 Collection Log 😝 Analysis Target 🛱 Analysis Type 🕅 Summary 🗞 Bottom-up 🍫 Caller/Callee 🗣	Top-down T	ree 🗷	Platform				In	nbal		e on	a lo	op	bar	rier		
Grouping: OpenMP Region ( OpenMP Barrier-to-Barrier Segment / Lunction / Call Stack													~		٩	
OpenMP Region / OpenMP Barrier-to-Barrier Segment / Function / Call Stack	Imbalance <del>-</del>	Lock Con	OpenMP F Creation	Potential G Scheau	ain Redu	Atomi	الله Other	Elapsed Time	Number of OpenMP threads	Ins Cou	OpenMP Loop Schedule Type	Open Loc Chu	nMP op unk	vg Open Loop Iteratior Count	ИР	
=conj_grad_\$omp\$parallel:24@/ho.ie/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:514:695	3.944	s Os	0.00 Us	0.002s	0.000s	Os	0.094s	11.095s	24	76						
Econj_grad_\$omp\$cop_barrier_segment	3.725	s <del>- 0.</del>	0.5	0.0005	0.	0.	0.0005	10.1155	21	-	Static	3125	5	75,0	000	
Econj_grad_\$omp\$loop_barrier_segment@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:683	0.149	s Os	Os	Os	Os	Os	0.004s	0.418s	24	-	Static	3125	5	75,0	000	
Econj_grad_\$omp\$loop_barrier_segment@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:625	0.033	s Os	Os	0.002s	0.000s	Os	0.002s	0.068s	24		Static	3125	5	75,0	000	
teconj_grad_\$omp\$loop_barrier_segment@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:650	0.015	s Os	Os	0.000s	0s	Os	0.001s	0.064s	24		Static	3125		75,0	000	
	0.011	5 05	05	0.0003	05	0.5	0.0015	0.0795			static	5125		, 3,		
Advanced Hotspots       Hotspots viewpoint (change)       Intel V lune Amplifier XE 2         Image: Collection Log       Analysis Target       Analysis Type       Summary       Bottom-up       Caller/Callee       Top-down Tree       Tasks and Frames         Grouping:       OpenMP Regio       / OpenMP Barrier-to-Barrier Segment       Function / Call Stack       V       V       V								4 AE 20								
		0	penMP Po	tential G	ain	<b></b>	OpenM	1P Poten	tial Gain	(% of C	ollection	т 🗹		Nu.		Open
OpenMP Region / OpenMP Barrier-to-Barrier Segment / Function / Call Stack	Imba.	Loc Con	k Cre	Schedul	ing Red	d Oth	Imba (%)	Lock Cont	Cre 5 (%)	Scheduli (%)	Red (%)	Oth (%)	Elap Time	. of Ope. thr	Co.	pe. Loop >o Sched hu. Type
=conj_grad_\$omp\$paral/el:24@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:514:6	95 0.20	0.0	0.0	3.1	27s 0.0	0.0	1.7%	0.0%	0.0%	25.9	9% 0.0%	0.0%	11.7	24		
Econj_grad_\$om\$\$loop_barrier@bome/utupe/work/apps/NPB/NPB3_3_1/NPB3_3_OMP/CG/cg f:57	2.580 0.00	0.0	0s	3.13	25s	0.0	0.1%	0.0%	0.07	25.9	9% 0.0%	0.0%	111	- 24	1	Dynam
Econj_grad_\$00p_barrier@/home/stune/work/apps/NPB/NPB3_3_1/NPB3_3_0MP/CG/cgf577 Econj_grad_\$00p_barrier@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-0MP/CG/cgf675	2·580 0.00 5:683 0.12		Os Os Os	3.13	25s 0s	0s 0.0	0.1% 1.1%	0.0%	0.0%	25.9	9% 0.0% 0% 0.0%	0.0%	11.1 0.41	. 24	1	Dynam l2. Static
Econj_grad_\$om\$\$loop_barrier@/home/stune/work/apps/NPB/NPB3_3_1/NPB3_3_OMP/CG/cgf577 Econj_grad_\$omp\$loop_barrier@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3.3-OMP/CG/cg.f:679 Econj_grad_\$omp\$loop_barrier@/home/vtune/work/apps/NPB/NPB3.3.1/NPB3_3-OMP/CG/cg.f:621	2:580 0.00 5:683 0.12 1:625 0.02		0s 0s 0s 0s	3.13	25s 0s 01s 0.0	05 0.0. 0s 0.0. 0 0.0.	0.1% 1.1% 0.2%	0.0%	0.0%	25.9 0.0 0.0	9% 0.0% 0% 0.0% 0% 0.0%	0.0% 0.0%	11.1 0.41 0.07	. 24 24 24	1 3: 3:	Dynam 12. Static 12. Static

#### **Optimization Notice**



## Performance Aspects: CPU Utilization (4/4)

Specifics for hybrid MPI + OpenMP apps

- MPI Imbalance metric for the node as average MPI busy wait time by rank (based on MPI progress sampling for MPICHbased MPIs (Intel MPI, CRAY MPI, ..\_)
  - Serial (outside any parallel region) and OpenMP Potenital Gain time for the MPI rank on critical path for the node (with minimal MPI busy wait per the node)
  - Worth to explore details on serial hotspots and OpenMP inefficiencies for the MPI rank on critical path in grid view







- Bandwidth utilization histogram

#### **Optimization Notice**



HPC Performance Characterization 🛛

## Performance Aspects: FPU Utilization

### FPU utilization

 % of FPU load (100% - FPU is fully loaded, threshold 50%)

### Metrics in FPU utilization section

- SP FLOPs per Cycle (vector code generation and execution efficiency)
- Vector Capacity Usage and FP Instruction Mix, FPArith/Mem ratios (vector code generation efficiency)
- Top 5 loops/functions by FPU usage
  - Dynamically generated issue descriptions on low FPU usage help to define the reason and next steps:

Non-vectorized, vectorized with legacy instruction set, memory bound limited loops not benefiting from vectorization etc.

# SP FLOPs per Cycle <sup>(0)</sup>: 0.211 Out of 16 <sup>(n)</sup> Vector Capacity Usage <sup>(0)</sup>: 48.3% <sup>(n)</sup> (○) FP Instruction Mix: (○) % of Packed FP Instr. <sup>(0)</sup>: 93.1% % of 128-bit <sup>(0)</sup>: 93.1% <sup>(n)</sup> % of 256-bit <sup>(0)</sup>: 0.0%

FPU Utilization 🗄: 1.3% 🏲

#### % of Scalar FP Instr.<sup>(2)</sup>: 6.9% FP Arith/Mem Rd Instr. Ratio<sup>(2)</sup>: 0.264 №

FP Arith/Mem Wr Instr. Ratio <sup>①</sup>: 6.298

#### ⊙ Top 5 hotspot loops (functions) by FPU usage

This section provides information for the most time consuming loops/functions with floating point operations.

Function	CPU Time ®	FPU Utilization <sup>®</sup>	Vector Instruction Set $^{}$	Loop Type 🖉
[Loop at line 575 in conj_grad_\$omp\$parallel@517]	126.149s	1.6% 🏲	SSE2(128) 🎙	Body
[Loop at line 678 in conj_grad_\$omp\$parallel@517]	5.004s	1.7%	SSE2(128)	Body
[Loop at line 575 in conj_grad_\$omp\$parallel@517]	2.678s	2.1%	[Unknown]	Remainder
[Loop at line 573 in conj_grad_\$omp\$parallel@517]	0.995s	4.0%	SSE2(128)	Body
[Loop at line 661 in conj_grad_\$omp\$parallel@517]	0.952s	1.3%	SSE(128); SSE2(128)	Body
[Others]	2.437s	N/A*	N/A*	N/A*
*N/A is applied to non-summable metrics.				



■ HPC Performance Characterization HPC Performance ■ Collection Log ● Analysis Traget ● Analysis Traget ● Summary Elapsed Time ©: 101.194s GFLOPS Upper Bound ©: 24.612 ○ CPU Utilization ©: 12.7% ► ● Back-End Bound ©: 87.8% ► FPU Utilization Upper Bound ©: 1.4% ►

## Performance Aspects: Process/Thread Affinity

### Process/Thread Affinity Collection and Reporting

- Visualization of affinity masks collected at the end of application
- Command Line Report with threads by cores affinity
- Experimental GUI report with affinity shown along with thread execution on CPU and remote accesses

### How to

```
>vtune -c h-p -knob collect-affinity=true -r my_result ./my_app
```

CL report:

>vtune -R affinity -r my\_result

GUI report (tech preview):

>vtune -R affinity -format=html -r my\_result







#### Optimization Notice



## Performance Aspects: Lustre Parallel File I/O

- Key Metrics:
  - I/O Wait Time, I/O Bandwidth, Packet Rate, Packet Size
- Availability:
  - Average on summary
  - Overtime on timeline with ability to
    - filter by time for context summary
- Details by Parallel File System Shares

Lustre File System IO Wait Time: 64.569s
 Bandwidth, B/s: 6,011,995.982
 Package Rate, pack/s: 25.750
 Average Package Size, KB: 233



 To enable set "AMPLXE\_EXPERIMENTAL=lustre", add "-knob parallel-fs-collection=true" to hpc-performance command line

$\odot$	Parallel File System detailed info:											
	Parallel File System	Read Bytes	Read Requests	Read Wait Time	Write Bytes	Write Requests	Write Wait Time	All Requests	Request Wait Time			
	/lfs/lfs09	6,161,204,090	26,379	48.513s	381,814	12	0.011s	27,086	49.799s			
	/lfs/lfs12	0	0	0s	0	0	0s	6,058	11.961s			
	/lfs/lfs13	0	0	0s	0	0	0s	963	2.020s			
	/lfs/lfs11	0	0	0s	0	0	0s	635	0.789s			

Optimization N

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

## HPC Performance Characterization – Command Line Reporting

- Generated after collection is done or with "-R summary" option of vtune
- Mimicking GUI metrics hierarchy

#### lapsed lime: 7.8055 P GFLOPS: 14.041 PU Utilization: 76.4%

The metric value is low, which may signal a poor logical GPU cores utilitation caused by load imbalance, threading runthes overhead, contended synchronization, or thread/process underutilization. Explore CPU Utilization sub-metrics to estimate the efficiency of FFI and OpenRD parallelism or run the locks and Waits analyzis to identify parallel bottlenecks for other parallel runtimes.

Serial Time: 0.021s (0.3%) Parallel Region Time: 7.784s (99.7%) Estimated Ideal Time: 6.4138 (82.2%) OpenMP Potential Gain: 1.371s (17.6%) | The time wasted on load imbalance or parallel work arrangement is | significant and negatively impacts the application performance and | scalability, Explore OpenMP regions with the highest metric values. | Make sure the workload of the regions is enough and the loop schedule emory Bound: 63.2% of Pipeline Slots The metric value is high. This can indicate that the significant fraction of stores. Use Memory Access analysis to have the metric breakdown by memory hierarchy, memory bandwidth information, correlation by memory objects. Cache Bound: 36.2% of Clockticks | caches are problematic and consider applying the same performance tuning as you would for a cache-missing workload. This may include reducing the | data working set size, improving data access locality, blocking or | partitioning the working set to fit in the lower cache levels, or exploiting hardware prefetchers. Consider using software prefetchers, but | note that they can interfere with normal loads, increase latency, and | increase pressure on the memory system. This metric includes coherence | penalties for shared data. Check General Exploration analysis to see if contested accesses or data sharing are indicated as likely issues. DRAM Bound: 28.94 of Clockticks | The metric value is high. This indicates that a significant fraction of | cycles could be stalled on the main memory (DRAM) because of demand loads | The code is memory bandwidth bound, which means that there are a | significant fraction of cycles during which the bandwidth limits of the

I sain memory are being teached and the code could stail. Review the Bandwidth Utilization fistogram to estimate the scale of the issue. I Consider improving data locality on NUMA multi-socket systems, which will reduce code memory bandwidth consumption.

NUMA: % of Remote Accesses: 13.5

#### Optimization Notice

Copyright © 2020, Intel Corporation. All rights reserved. \*Other names and brands may be claimed as the property of others. With issue descriptions that can be suppressed by "–report-knob show-issues=false" option

#### Elapsed Time: 7.805s SP GFLOPS: 14.041

CPU Utilization: 76.4% Average CPU Usage: 18.344 Out of 24 logical CPUs Serial Time: 0.021s (0.3%) Parallel Region Time: 7.784s (99.7%) Estimated Ideal Time: 6.413s (82.2%) OpenMP Potential Gain: 1.371s (17.6%) Memory Bound: 63.2% of Pipeline Slots Cache Bound: 36.2% of Clockticks DRAM Bound: 28.9% of Clockticks NUMA: % of Remote Accesses: 13.9% FPU Utilization: 1.3% SP FLOPs per Cycle: 0.211 Out of 16 Vector Capacity Usage: 48.3% FP Instruction Mix % of Packed FP Instr.: 93.1% % of 128-bit: 93.1% % of 256-bit: 0.0% % of Scalar FP Instr.: 6.9% FP Arith/Mem Rd Instr. Ratio: 0.264 FP Arith/Mem Wr Instr. Ratio: 6.298 Collection and Platform Info Application Command Line: ./cg.B.x User Name: vtune



## How to Run VTune on MPI Applications

Single or multiple node application launch:

<mpi\_launcher> - n N <vtune\_command\_line> ./app\_to\_run

- >aprun –n 48 -N 16 vtune –collect hotspots **-trace-mpi** –r result\_dir ./my\_mpi\_app
- >mpirun –n 48 -ppn 16 vtune –collect hotspots –r result\_dir ./my\_mpi\_app
- Encapsulates ranks to per-node result directories suffixed with hostname
  - result\_dir.hostname1 with 0-15, result\_dir.hostname2 with 16-31, result\_dir.hostname3 with 32-47
  - -trace-mpi option is mandatory to enable per-node result directories for MPIs not using PMI\_RANK (e.g. CRAY MPI)
- Works for software and driver-based/driverless collectors
  - Highly recommended to set /proc/sys/kernel/perf\_event\_paranoid to 0 for perf-based collections to avoid collection overhead on modern multi-/many-core systems



## Selective Rank profiling of MPI applications with VTune

To reduce data collected by VTune on big scale runs use selective rank profiling using MPMD (Multiple Program Multiple Data) runs:

Example: profile rank 1 from 0-15:

>mpirun -n 1 ./my\_app : -n 1 <vtune\_command\_line> ./my\_app : -n 14 ./my\_app

• In the case of Intel MPI launcher –gtool option can be used to simplify the launch configuration

Example: profile ranks 2, 3, 10-15 from 0-15:

>mpirun –gtool "vtune –collect hotspots –r result\_dir:2,3,10-15" ./my\_app



## Memory Access Analysis

Specialized analysis type "Memory Access" to focus on memory-related issues:

- Performance problems by memory hierarchy (e.g. L1-, L2-, LLC-, DRAM-bound)
- Bandwidth-limited accesses
  - Covers DRAM, QPI, and PMEM bandwidth
- NUMA problems
- Performance metrics by memory objects (data structures)

>vtune -collect memory-access ./my\_app

Memory Access Memory Usage     Collection Log O Analysis Target A Ai	viewpoint (change) 🔮 nalysis Type 🔹 Summary 🗟 Bottom-up 📧 Platform
S Flansed Time . 6	6895
CPU Time <sup>®</sup> :	25.121s
	44.4% Nof Pipeline Slots
L1 Bound <sup>®</sup> :	0.7% of Clockticks
L2 Bound <sup>®</sup> :	0.0% of Clockticks
L3 Bound <sup>®</sup> :	30.5% 🕅 of Clockticks
() DRAM Bound <sup>()</sup> :	8.0% of Clockticks
Loads:	17,604,528,120
Stores:	8,789,663,682
() LLC Miss Count <sup>®</sup> :	46,352,781

#### Bandwidth Utilization Histogram

Explore bandwidth utilization over time using the histogram and identify memory objects or functions with maximum contribution to the high bandwidth utilization.

Bandwidth Domain: DRAM, GB/sec • Bandwidth Utilization Histogram



## Memory Bandwidth Analysis

### DRAM and PMEM bandwidth

- Can be expanded per package/channel
- Split to Read/Write
- Automatic DRAM pick measurement on Xeon

### QPI bandwidth per-package

- Can be expanded per-link
- Split to Incoming/Outgoing and Data/Non-Data



#### Optimization Notice

## View Performance Metrics by Memory Objects

### Memory Access analysis configuration options for profiling memory objects:

- Analyze dynamic memory objects: enables the instrumentation of memory allocation/de-allocation and map hardware events to memory objects
- Minimal memory object size to track, in bytes: specify a minimal size of memory allocations to analyze. This option helps reduce runtime overhead of the instrumentation. Default is 1K.

>vtune -c memory-access -knob analyze-mem-objects=true -knob mem-object-size-min-thres=2048 -- <app>



#### **Optimization Notice**



## High Memory Bandwidth Analysis by Memory Objects

If memory object instrumentation is ON Bandwidth Utilization histogram is enriched with the table of Memory Objects by LLC Miss Count



#### **Optimization Notice**

## High Memory Bandwidth Analysis by Memory Objects

### Investigate the memory allocations inducing bandwidth

 "Bandwidth Domain/Bandwidth Utilization Type/Memory Object/Allocation Stack" grouping with expansion by "DRAM/High" and sorting by L2 Miss Count Count

Memory Access Memory Usage viewpoint (change) 🕐		INTEL VTUNE AMPLI	FIER XE 2017	💹 Me	mory Access Memory Usage viewpoint ( <u>change</u> ) ⑦
🔍 📟 Collection Log \varTheta Analysis Target 🚊 Analysis Type 🕅 Summary 🐼 E	Bottom-up 🛃 Platform 🖹 CS	CSRMatrix.h	Þ	⊲	Collection Log \varTheta Analysis Target 🙏 Analysis Type 📓 Summary 🗳 Bot
Cr°C+ C−Cr° 55 1/5 1/5 2/5 2/5 3/5 ¥ @ package_0 1/5 @ 7.0 # # # # # # # # # # # # #	355 ♥ DRAM Bandwidth, ♥ With Total, GB/sec ♥ ₩ Read, GB/sec ♥ Write, GB/sec ■ MCDRAM Flat Mod ■ MCDRAM Cache M	L2 Miss Count (Memory Allocation) Viewing ( 1 of 1 ): selected tack(s) 100.0% (20400612 of 20400612) miniFE xi anu cocraw, allocator/double2-allocate - new_allocator in miniFE xi anu cocraw, allocator/double2. Xi allocate and co- manife Xi with vector/double, dt/-allocator/double2. Xi allocate and co- manife Xi with vector/double, dt/-allocator/double2. Xi allocate and co- manife Xi with vector/double. Vector double and viewerse, stack(b) 10 - CE	v opy≤double*> SMatrix bop.93	Source So. ▲ 88 89 90	Assembly     Assembly     Assembly     Assembly     Sou     void reserve_space(unsigned nrows, unsigned ncols_per_row)     (     rows.resize(nrows);
Groupina: Bandwidth Doman / Bandwidth Likitation Type / Memory Object / Allocation Stack	CPU Time → →  →	minFE xlminFE: <u>denerate</u> matrix structure(minFE: <u>CSRMatrix<double< u="">, minFE xlminFE:<u>deneratouble</u>, int. inte<sup>-</sup>-0x186 - driver.hpp:181 minFE xlminFE:<u>directouble</u>, pp.178 minFE xl_<u>start</u>+0x28 - [unknown source file]</double<></u>	int int>>+0x1	91 92 93 94 95 96	<pre>tow_oilsets.resize(nrows+1); packed_ools.reserve(nrows * ncols_per_row); packed_coefs.reserve(nrows * ncols_per_row);  #pragma cmp parallel for for(MINIFE_GLOBAL_ORDINAL i = 0; i &lt; nrows; ++i) (</pre>
Bandwidth Domain / Bandwidth Utilization Type / Memory Object / Allocation Stack DRAM, GB/sec High	CPU Time 4517.651s 3053.636s	L2 Miss Count+ 311,009,330 259,007,770	^	97 98 99	rows[1] = 0; row_offsets[1] = 0; )
ID new_allocator.h104 (80 MB)         ID new_allocator.h104 (80 MB)           ID Uninown]         ID new_allocator.h104 (31 MB)           ID new_allocator.h104 (420 MB)         ID new_allocator.h104 (420 MB)           ID (Stack)         ID new_allocator.h104 (31 MB)           ID new_allocator.h104 (31 MB)         ID new_allocator.h104 (31 MB)		102,003,060 71,002,130 53,001,580 22,000,660 5,000,150 3,000,000	1		Focus on allocations inducing L2 misses
Unew_allocator.ht.194 (31 MB) @inew_allocator.ht.194 (51 MB) @Medum @DRAM.Read, GB/sec	754.1495 709.8665 4517.6515	2,000,000 1,000,030 5,0,001,500 2,000,060 311,009,330 211,009,330			Allocation stack shows the allocation place in

#### **Optimization Notice**

## **GPU** Application Analysis

### **GPU Compute/Media Hotspots**

- Visibility into both host and GPU sides
- HW-events based performance tuning methodology
- Provides overtime and aggregated views

### **GPU In-kernel Profiling**

- GPU source/instruction level profiling
- SW instrumentation
- Two modes: basic Block latency and Memory access latency

Identify GPU occupancy and which kernel to profile. Tune a kernel on a fine-grain level.







#### Optimization Notice

## **CPU/FPGA Interaction Analysis**

Now process data sources collected with AOCL Profiler (new mode) in addition to OpenCL Profiling API (legacy mode).

Extended with FPGA device-side metrics, like Stalls, Global Bandwidth and Occupancy, and mapping FPGA kernel performance data to the source code

🔅 CPU/FPGA Interaction (preview) 👻 🙆		
Preview feature - should we keep it, change it, or drop it? Send us your comments.		
Analyze the CPU/FPGA interaction issues via exploring OpenCL kernels running on the FPGA and identify the most time-consuming kernels. Learn more	11	FPG/
CPU sampling interval, ms	etrics	fft1
5	N N	inou
Collect stacks	EPC	·
FPGA profiling data source	11	outp
AOCL Profiler 👻	1.1	CPU
AOCL Profiler	1.5	0.0
OpenCL Profiling API	Ш	FILTER
Optimization Notice		

	FPGA Utilization		
etrics	fft1d		
-PGAN	input_kernel		fft1d Global Bandwidth, GB/s
	output_kernel		0 /sec Stalls (%)
	CPU Time		0.1% Occupancy (%)
F	ilter 🝸 100.0% 🦹	Any P ▼ Any Thre ▼	99.7%



## Legal Disclaimer & Optimization Notice

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. For more complete information visit <u>www.intel.com/benchmarks</u>.

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Copyright © 2018, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

#### **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