Intel® oneAPI VTune™ Profiler 2021.1.1 Gold

Recommendations:

Hotspots: Start with Hotspots analysis to understand the efficiency of your algorithm.

> Use Hotspots analysis to identify the most time consuming functions. Drill down to see the time spent on every line of code.

Microarchitecture Exploration: There is low microarchitecture usage (19.8%) of available hardware resources. of Pipeline Slots

> Run Microarchitecture Exploration analysis to analyze CPU microarchitecture bottlenecks that can affect application performance.

Memory Access: The Memory Bound metric is high (25.7%). A significant fraction of execution pipeline slots could be stalled due to demand memory load and stores. of Pipeline Slots

> Use Memory Access analysis to measure metrics that can identify memory access issues.

Threading: There is poor utilization of logical CPU cores (26.8%) in your application.

> Use Threading to explore more opportunities to increase parallelism in your application.

Elapsed Time: 0.160s

CPU:

IPC: 1.787

DP GFLOPS: 0.000

x87 GFLOPS: 0.001

Average CPU Frequency: 1.971 GHz

GPU:

Time: 1.1% (0.002s) of Elapsed time

> GPU utilization is low. Consider offloading more work to the GPU to increase overall application performance.

IPC Rate: 1.335

Effective Logical Core Utilization: 26.8% (2.145 out of 8)

> The metric value is low, which may signal a poor logical CPU cores utilization. Consider improving physical core

> utilization as the first step and then look at opportunities to utilize logical cores, which in some cases can improve processor throughput and overall performance of multi-threaded applications.

Effective Physical Core Utilization: 47.1% (1.883 out of 4)

> The metric value is low, which may signal a poor physical CPU cores utilization caused by:
>
> - load imbalance
> - threading runtime overhead
> - contended synchronization
> - thread/process underutilization
> - incorrect affinity that utilizes logical cores instead of physical cores
>
> Explore sub-metrics to estimate the efficiency of MPI and OpenMP parallelism or run the Locks and Waits analysis to identify parallel bottlenecks for other parallel runtimes.

Microarchitecture Usage: 19.8% of Pipeline Slots

> You code efficiency on this platform is too low.
>
> Possible cause: memory stalls, instruction starvation, branch misprediction or long latency instructions.
>
> Next steps: Run Microarchitecture Exploration analysis to identify the cause of the low microarchitecture usage efficiency.

Retiring: 19.8% of Pipeline Slots
Front-End Bound: 22.0% of Pipeline Slots

> Issue: A significant portion of Pipeline Slots is remaining empty due to issues in the Front-End.
>
> Tips: Make sure the code working size is not too large, the code layout does not require too many memory accesses per cycle to get enough instructions for filling four pipeline slots, or check for microcode assists.

Back-End Bound: 52.3% of Pipeline Slots

> A significant portion of pipeline slots are remaining empty. When operations take too long in the back-end, they introduce bubbles in the pipeline that ultimately cause

> fewer pipeline slots containing useful work to be retired per cycle than the machine is capable to support. This opportunity cost results in slower execution. Long-latency operations like divides and memory operations can cause this, as can too many operations being directed to a single execution port (for example, more multiply operations arriving in the back-end per cycle than the execution unit can support).

Memory Bound: 25.7% 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.

Core Bound: 26.7% of Pipeline Slots

> This metric represents how much Core non-memory issues were of a bottleneck. Shortage in hardware compute resources, or dependencies software's instructions are both categorized under Core Bound. Hence it may indicate the machine ran out of an OOO resources, certain execution units are overloaded or dependencies in program's data- or instruction- flow are limiting the performance (e.g. FP-chained long-latency arithmetic operations).

Bad Speculation: 5.8% of Pipeline Slots

Memory Bound: 25.7% 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: 1.4% of Clockticks
L2 Bound: 0.4% of Clockticks
L3 Bound: 6.0% 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: 4.0% of Clockticks
DRAM Bandwidth Bound: 0.0% of Elapsed Time
Store Bound: 2.8% of Clockticks

Vectorization: 0.0% of Packed FP Operations

> A significant fraction of floating point arithmetic instructions are scalar. Use Intel Advisor to see possible reasons why the code was not vectorized.

Instruction Mix:
SP FLOPs: 0.0% of uOps
Packed: 0.0% from SP FP
128-bit: 0.0% from SP FP
256-bit: 0.0% from SP FP
Scalar: 0.0% from SP FP

DP FLOPs: 0.0% of uOps
Packed: 0.0% from DP FP
128-bit: 0.0% from DP FP
256-bit: 0.0% from DP FP
Scalar: 100.0% from DP FP

> This code has floating point operations and is not vectorized. Consider using Intel Advisor to vectorize the loops.

x87 FLOPs: 0.0% of uOps

Non-FP: 100.0% of uOps

FP Arith/Mem Rd Instr. Ratio: 0.001

> The metric value is low. This can be a result of unaligned access to data for vector operations. Use Intel Advisor to find possible data access inefficiencies for vector operations.

FP Arith/Mem Wr Instr. Ratio: 0.001

> The metric value is low. This can be a result of unaligned access to data for vector operations. Use Intel Advisor to

> find possible data access inefficiencies for vector operations.

GPU Active Time: 1.1%

> GPU utilization is low. Consider offloading more work to the GPU to increase overall application performance.

GPU Utilization when Busy: 36.3%

> The percentage of time when the EUs were stalled or idle is high, which has a negative impact on compute-bound applications.

IPC Rate: 1.335
EU State: 36.3%
Active: 36.3%
Stalled: 28.1%

> A significant portion of GPU time is lost due to stalls. For compute-bound code, this could indicate that performance is limited by memory or sampler acesses.

Idle: 35.5%

> A significant portion of GPU time is spent idle. This is usually caused by imbalance or thread scheduling problems.

Occupancy: 44.4% of peak value

> Low value of the occupancy metric may be caused by inefficient work scheduling. Make sure work items are neither too small nor too large.

Collection and Platform Info:
Application Command Line: ./codecs/hm/decoder/TAppDecoderStatic "-b" "./bin/hm/encoder_intra_main.cfg/CLASS_A/Kimono_1920x1080_24_QP_32_hm.bin"

Operating System: 5.4.0-65-generic DISTRIB_ID=Ubuntu DISTRIB_RELEASE=18.04 DISTRIB_CODENAME=bionic DISTRIB_DESCRIPTION="Ubuntu 18.04.5 LTS"

Computer Name: eimon

Result Size: 3.7 MB

Collection start time: 09:31:31 10/02/2021 UTC

Collection stop time: 09:31:31 10/02/2021 UTC

Collector Type: Event-based sampling driver,Event-based counting driver

CPU:

- **Name:** Intel(R) Processor code named Kabylake ULX
- **Frequency:** 1.992 GHz
- **Logical CPU Count:** 8
- **Max DRAM Single-Package Bandwidth:** 10.000 GB/s
- **Cache Allocation Technology:**
  - **Level 2 capability:** not detected
  - **Level 3 capability:** not detected

GPU:

- **Name:**
  Display controller: Intel Corporation Device 22807
- **Vendor:** Intel Corporation
- **EU Count:** 24
- **Max EU Thread Count:** 7
- **Max Core Frequency:** 1.150 GHz