

**Recommendations:****Increase execution time:**

Application execution time is too short. Metrics data may be unreliable. Consider reducing the sampling interval or increasing your application execution time.

**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 (22.0%) 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 (40.5%). 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.

**Elapsed Time:** 0.018s

Application execution time is too short. Metrics data may be unreliable. Consider reducing the sampling interval or increasing your application execution time.

**CPU:**

**IPC:** 0.009

The IPC may be too low. This could be caused by issues such as memory stalls, instruction starvation, branch misprediction or long latency instructions. Explore the other hardware-related metrics to identify what is causing low IPC.

**DP GFLOPS:** 0.000

**x87 GFLOPS:** 0.011

**Average CPU Frequency:** 2.702 GHz

**GPU:****Time:** 114.6% (0.021s) of Elapsed time

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

**IPC Rate:** 1.547**Effective Logical Core Utilization:** 357.1% (28.569 out of 8)**Effective Physical Core Utilization:** 100.0% (4.000 out of 4)**Microarchitecture Usage:** 22.0% 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:** 22.0% of Pipeline Slots

A high fraction of pipeline slots was utilized by useful work. While the goal is to make this metric value as big as possible, a high Retiring value for non-vectorized code could prompt you to consider code vectorization. Vectorization enables doing more computations without significantly increasing the number of instructions, thus improving the performance. Note that this metric value may be highlighted due to Microcode Sequencer (MS) issue, so the performance can be improved by avoiding using the MS.

**Front-End Bound:** 2.3% of Pipeline Slots**Back-End Bound:** 71.9% 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:**

40.5% 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.

**Core Bound:**

31.4% 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:**

3.8% of Pipeline Slots

**Memory Bound:**

40.5% 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:**

1.0% of Clockticks

**L3 Bound:**

7.2% 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:** 2.8% of Clockticks  
**DRAM Bandwidth Bound:** 54.2% of Elapsed Time

The system spent much time heavily utilizing DRAM bandwidth. Improve data accesses to reduce cacheline transfers from/to memory using these possible techniques: 1) consume all bytes of each cacheline before it is evicted (for example, reorder structure elements and split non-hot ones); 2) merge compute-limited and bandwidth-limited loops; 3) use NUMA optimizations on a multi-socket system. Note: software prefetches do not help a bandwidth-limited application. Run Memory Access analysis to identify data structures to be allocated in High Bandwidth Memory (HBM), if available.

**Store Bound:** 0.2% 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:**

<b>SP FLOPs:</b>	0.0% of uOps
<b>Packed:</b>	0.0% from SP FP
<b>128-bit:</b>	0.0% from SP FP
<b>256-bit:</b>	0.0% from SP FP
<b>Scalar:</b>	0.0% from SP FP
<b>DP FLOPs:</b>	0.0% of uOps
<b>Packed:</b>	3.8% from DP FP
<b>128-bit:</b>	3.8% from DP FP
<b>256-bit:</b>	0.0% from DP FP
<b>Scalar:</b>	96.2% from DP FP

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

**x87 FLOPs:** 0.0% of uOps

**Non-FP:** 100.0% of uOps

**FP Arith/Mem Rd Instr. Ratio:** 0.018

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:** 0.033

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:** 114.6%  
**GPU Utilization when Busy:** 40.6%

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

**IPC Rate:** 1.547  
**EU State:** 40.6%  
**Active:** 40.6%  
**Stalled:** 30.1%

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 accesses.

**Idle:** 29.3%

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

**Occupancy:** 51.6% 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/HHI-VVC/decoder/vvdecapp "-b"  
"./bin/HHI-VVC/randomaccess\_faster.cfg/CLASS\_C/  
RaceHorses\_416x240\_30\_QP\_37\_HHI-VVC.bin"

**Operating System:** 5.4.0-72-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:** 22:35:59 18/04/2021 UTC

**Collection stop time:** 22:35:59 18/04/2021 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:** 30.000 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

