Intel® oneAPI VTune™ Profiler 2021.1.1 Gold

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 (1.6%) 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 (50.5%). 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: 0.031s

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

CPU:

IPC: 0.024

> 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: 0.001

Average CPU Frequency: 2.342 GHz

GPU:

Time: 46.2% (0.014s) of Elapsed time

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

IPC Rate: 1.292

Effective Logical Core Utilization: 102.9% (8.231 out of 8)

Effective Physical Core Utilization: 100.0% (4.000 out of 4)

Microarchitecture Usage: 1.6% 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: 1.6% of Pipeline Slots

Front-End Bound: 4.6% of Pipeline Slots

Back-End Bound: 93.0% 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: 50.5% 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: 42.5% 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: 0.9% of Pipeline Slots

Memory Bound: 50.5% 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: 36.4% of Clockticks

> This metric shows how often machine was stalled without missing the L1 data cache. The L1 cache typically has the shortest latency. However, in certain cases like loads blocked on older stores, a load might suffer a high latency even though it is being satisfied by the L1.

L2 Bound: 0.2% of Clockticks
L3 Bound: 0.0% of Clockticks
DRAM Bound: 8.5% of Clockticks
DRAM Bandwidth Bound: 0.0% of Elapsed Time
Store Bound: 0.5% of Clockticks

Vectorization: 0.2% 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:** 12.1% from DP FP
    - **128-bit:** 12.1% from DP FP

> A significant fraction of floating point arithmetic vector instructions is executed with a partial vector load. Make sure you compile the code with the latest instruction set or use Intel Advisor for vectorization help.

    - **256-bit:** 0.0% from DP FP
  - **Scalar:** 87.9% 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:** 0.1% of uOps
- **Non-FP:** 99.9% of uOps

FP Arith/Mem Rd Instr. Ratio: 0.011

> 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: 0.017

> 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: 46.2%

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

GPU Utilization when Busy: 22.1%

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

IPC Rate: 1.292
EU State: 22.1%
Active: 22.1%
Stalled: 38.6%

> 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: 39.3%

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

Occupancy: 36.8% 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_22_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: 22:32:43 18/04/2021 UTC

Collection stop time: 22:32:43 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:** 11.000 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

Intel® oneAPI VTune™ Profiler 2021.1.1 Gold

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 (10.9%) 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 (29.5%). 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: 0.051s

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

CPU:

IPC: 0.424

> 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.

SP GFLOPS: 0.008

DP GFLOPS: 0.000

x87 GFLOPS: 0.002

Average CPU Frequency: 909.536 MHz

GPU:

Time: 21.5% (0.011s) of Elapsed time

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

IPC Rate: 1.646

Effective Logical Core Utilization: 129.2% (10.334 out of 8)
Effective Physical Core Utilization: 100.0% (4.000 out of 4)

Microarchitecture Usage: 10.9% 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: 10.9% of Pipeline Slots
Front-End Bound: 22.4% 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: 62.6% 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: 29.5% 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: 33.0% 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: 4.1% of Pipeline Slots

Memory Bound: 29.5% 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: 7.8% of Clockticks

> This metric shows how often machine was stalled without missing the L1 data cache. The L1 cache typically has the shortest latency. However, in certain cases like loads blocked on older stores, a load might suffer a high latency even though it is being satisfied by the L1.

L2 Bound: 0.4% of Clockticks
L3 Bound: 8.6% 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: 0.5% of Clockticks

Vectorization: 16.0% 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: 0.1% of uOps
Packed: 29.0% from SP FP
128-bit: 3.7% from SP FP
256-bit: 25.3% from SP FP
Scalar: 71.0% from SP FP

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

DP FLOPs: 0.0% of uOps
Packed: 9.2% from DP FP
128-bit: 9.2% from DP FP

> A significant fraction of floating point arithmetic vector instructions is executed with a partial vector load. Make sure you compile the code with the latest instruction set or use Intel Advisor for vectorization help.

256-bit: 348.7% from DP FP
Scalar: 90.8% 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: 0.1% of uOps

Non-FP: 99.8% of uOps

FP Arith/Mem Rd Instr. Ratio: 0.003

> 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: 0.007

> 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: 21.5%

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

GPU Utilization when Busy: 51.4%

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

IPC Rate: 1.646
EU State: 51.4%
- **Active:** 51.4%
- **Stalled:** 18.7%
- **Idle:** 29.9%

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

Occupancy: 56.9% 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_22_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: 07:52:29 19/04/2021 UTC

Collection stop time: 07:52:29 19/04/2021 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: 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