Intel® oneAPI VTune™ Profiler 2021.1.1 Gold

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.

Clockticks:	207,000,000
Instructions Retired:	176,760,000
CPI Rate:	1.171

> The CPI may be too high. 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 high CPI.

MUX Reliability:	0.861
Retiring:	36.5% of Pipeline Slots
Light Operations:	34.2% of Pipeline Slots
FP Arithmetic:	0.0% of uOps
FP x87:	0.0% of uOps
FP Scalar:	0.0% of uOps
FP Vector:	0.0% of uOps
Other:	100.0% of uOps
Heavy Operations:	2.4% of Pipeline Slots
Microcode Sequencer:	6.7% of Pipeline Slots
Assists:	0.0% of Pipeline Slots
Front-End Bound:	34.4% 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.

Front-End Latency:	37.6% of Pipeline Slots

> This metric represents a fraction of slots during which CPU was stalled due to front-end latency issues, such as instruction-cache misses, ITLB misses or fetch stalls after a branch misprediction. In such cases, the front-end delivers no uOps.

ICache Misses:	0.0% of Clockticks
ITLB Overhead:	1.0% of Clockticks
Branch Resteers:	5.2% of Clockticks

> Issue: A significant fraction of cycles was stalled due to Branch Resteers. Branch Resteers estimate the Front-End delay in fetching operations from corrected path, following all sorts of mispredicted branches. For example, branchy code with lots of mispredictions might get categorized as Branch Resteers. Note the value of this node may overlap its siblings.

Mispredicts Resteers:	0.0% of Clockticks
Clears Resteers:	5.2% of Clockticks

> A significant fraction of cycles could be stalled due to Branch Resteers as a result of Machine Clears.

Unknown Branches:	0.0% of Clockticks
DSB Switches:	0.0% of Clockticks
Length Changing Prefixes:	0.0% of Clockticks
MS Switches:	0.0% of Clockticks

> Issue: A significant fraction of cycles was stalled due to switches of uOp delivery to the Microcode Sequencer (MS). Commonly used instructions are optimized for delivery by the DSB or MITE pipelines. Certain operations cannot be handled natively by the execution pipeline, and must be performed by microcode (small programs injected into the execution stream). Switching to the MS too often can negatively impact performance. The MS is designated to deliver long uOp flows required by CISC instructions like CPUID, or uncommon conditions like Floating Point Assists when dealing with Denormals. Note that this metric value may be highlighted due to Microcode Sequencer issue.

Front-End Bandwidth:	0.0% of Pipeline Slots
Front-End Bandwidth MITE:	29.2% of Clockticks
Front-End Bandwidth DSB:	0.0% of Clockticks
(Info) DSB Coverage:	28.3%
Bad Speculation:	3.1% of Pipeline Slots
Branch Mispredict:	0.0% of Pipeline Slots
Machine Clears:	3.1% of Pipeline Slots
Back-End Bound:	25.9% 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:	12.8% of Pipeline Slots
L1 Bound:	5.2% of Clockticks
DTLB Overhead:	1.2% of Clockticks
Load STLB Hit:	0.0% of Clockticks
Load STLB Miss:	1.2% of Clockticks
Loads Blocked by Store Forwarding:	0.0% of Clockticks
Lock Latency:	0.0% of Clockticks
Split Loads:	0.0% of Clockticks
4K Aliasing:	0.0% of Clockticks
FB Full:	0.0% of Clockticks
L2 Bound:	
L3 Bound:	10.4% of Clockticks
Contested Accesses:	
Data Sharing:	
L3 Latency:	
SQ Full:	0.0% of Clockticks
DRAM Bound:	
Memory Bandwidth:	2.6% of Clockticks
Memory Latency:	7.8% of Clockticks

Store Bound:	0.0% of Clockticks
Store Latency:	15.7% of Clockticks
False Sharing:	0.0% of Clockticks
Split Stores:	0.0% of Clockticks
DTLB Store Overhead:	0.6% of Clockticks
Store STLB Hit:	0.0% of Clockticks
Store STLB Hit:	0.6% of Clockticks
Core Bound:	13.1% 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).

Divider:	0.0% of Clockticks
Port Utilization:	13.3% of Clockticks
Cycles of 0 Ports Utilized:	29.2% of Clockticks
Serializing Operations:	5.2% of Clockticks
Mixing Vectors:	0.0% of uOps
Cycles of 1 Port Utilized:	6.3% of Clockticks
Cycles of 2 Ports Utilized:	18.8% of Clockticks
Cycles of 3+ Ports Utilized:	10.4% of Clockticks
ALU Operation Utilization:	10.4% of Clockticks
Port 0:	8.3% of Clockticks
Port 1:	8.3% of Clockticks
Port 5:	4.2% of Clockticks
Port 6:	20.9% of Clockticks
Load Operation Utilization:	4.2% of Clockticks
Port 2:	4.2% of Clockticks
Port 3:	4.2% of Clockticks
Store Operation Utilization:	0.0% of Clockticks
Port 4:	0.0% of Clockticks
Port 7:	0.0% of Clockticks
Vector Capacity Usage (FPU):	0.0%
Average CPU Frequency:	2.279 GHz
Total Thread Count:	9
Paused Time:	0s

Effective Physical Core Utilization: 44.6% (1.782 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.
```

Effective Logical Core Utilization: 35.6% (2.852 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.
```

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"

User Name:	root
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:	14.1 MB
Collection start time:	22:32:22 18/04/2021 UTC
Collection stop time:	22:32:22 18/04/2021 UTC
Collector Type:	Event-based sampling driver

CPU:

Name:	Intel(R) Processor code named Kabylake ULX
Frequency:	1.992 GHz
Logical CPU Count:	8

Cache Allocation Technology:	
Level 2 capability:	not detected
Level 3 capability:	not detected