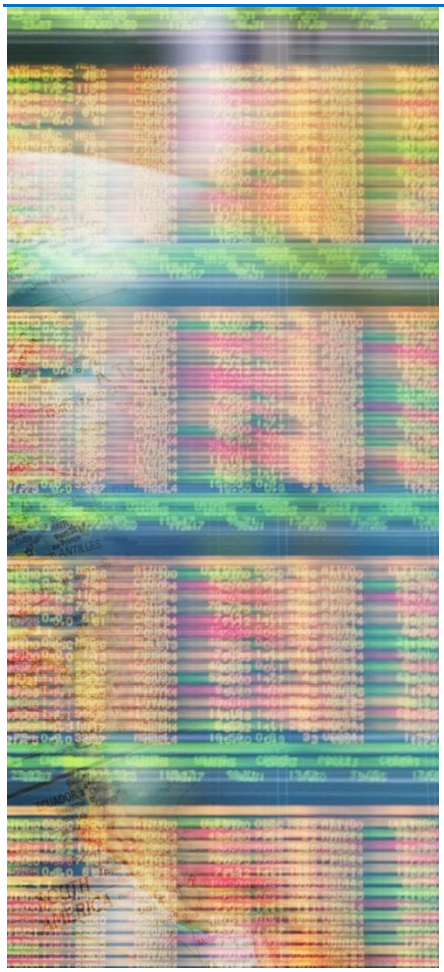
Optimization Guide

High Performance Computing
Intel® Xeon® Scalable Processors

SVT-HEVC Encoder Tuning Guide on 3rd Generation Intel® Xeon® Scalable Processors



Revision Record

Date	Rev.	Description
05/27/21	1.3	Added conclusion and feedback section
04/12/2021	1.2	Minor corrections plus adding section 6.4
04/06/2021	1.0	Initial public release.

1. Introduction

This guide is targeted towards users who are already familiar with the Scalable Video Technology for HEVC Encoder (SVT-HEVC) and provides pointers and setting for BIOS, OS, SVT-HEVC encoder, and system settings that will provide the best performance for most situations. However, please note that we rely on the users to carefully consider these settings for their specific scenarios, since SVT-HEVC can be deployed in multiple ways and this is a reference to one such use-case. SVT-HEVC is hosted on both Linux* and Microsoft Server* operating systems. The settings apply to running SVT-HEVC on a 2-socket Intel® Server System with 3rd Generation Intel® Xeon® Scalable Processors and Intel recommends using:

- Ubuntu* v18.04 or newer
- CentOS* 7.4 or newer
- Windows Server 2016 or newer

SVT-HEVC is an open source project and pull requests are managed by the git hub community. As of this writing, Intel recommends SVT HEVC 1.5.0. Generally, the newest master branch is recommended. To get the latest SVT-HEVC code, visit: https://github.com/OpenVisualCloud/SVT-HEVC .To learn more about SVT, visit: https://01.org/svt .

3rd Gen Intel® Xeon® Scalable processors deliver improved architecture, increased core counts and larger caches and provides a seamless performance foundation to help speed data's transformative impact, from the multi-cloud to the intelligent edge and back. Improvements of particular interest to media applications are:

- Increased DDR4 Memory Capacity & Speed
- Intel® Advanced Vector Extensions
- Intel® Security Essentials and Intel® Security Libraries for Data Center

2. BIOS Settings

<No special instructions for BIOS – use defaults>

3. System Set-up and Optimizations

3.1 Memory Configuration & Sizing

SVT utilizes multi-channel encoding to accelerate performance. Best results are achieved when each core has direct access to memory. Intel recommends that each of the 8 memory channels per socket be populated with DDR4 RAM DIMM modules rated at the maximum MT/S for the selected processor. The 3rd Generation Intel Xeon processors support a range from 2667 to 3200 MT/s.

In addition to populating each memory channel, the total amount of system memory is an important consideration. Table 1 below lists the recommended memory per stream type of a given resolution. For example, 6GB of memory is recommended for each FHD stream being processed. To process 64 FHD channel simultaneously, 384 GB (64x6GB) should be allocated.

Table 1 Recommended System Memory by Stream Resolution

Stream Resolution	Memory recommendation per Stream
UHD8K	64 GB

Stream Resolution	Memory recommendation per Stream
UHD4K	16 GB
FHD	6 GB
HD/1080i	4 GB
SD	3 GB

3.2 Linux Configuration

There are several recommendations to improve performance on Linux.

It is common on modern Linux implementations to have scaling governors for managing CPU frequencies when using the intel_pstate driver (*powersave*, *performance*). Intel encourages setting the governors to *performance* as aggressive CPU frequencies will improve transcode throughput, using the following command:

```
echo "performance" | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

Intel recommends creating a Linux ramdisk file system for YUV files to avoid latency issues reading media files from disk. Ramdisks are preferred; Users may wish to explore the -nb switch documented in Chapter 4 for experiments. Note: Ramdisks should be configured to be at least the same size of the input media files.

```
sudo mount -t tmpfs -o size=<size in bytes> tmpfs <directory_name>
```

(Example: *sudo mount -t tmpfs -o size=64g tmpfs /media)*

4. SVT HEVC Encoder Settings

4.1 Downloading the Encoder

The SVT Encoder is found at the Open Visual Cloud Git Hub: https://github.com/OpenVisualCloud/SVT-HEVC. The README at this site includes instructions for building the encoder, and lists the Linux project's dependencies as cmake, YASM and GCC (Linux only).

4.2 AVX 512

While SVT-HEVC v1.5.0 includes functions that have been tuned for the AVX-512 instruction set, AVX512 is not enabled in the default build. Those interested in tuning their SVT-HEVC workloads for 3rd Gen Intel Xeon Scalable Processors (and beyond) should consider evaluating AVX-512 on their own pipelines.

To enable AVX-512, comment out the following line from Source/Lib/Codec/EbDefinitions.h and recompile using gcc 9.3 or newer:

```
#define NON_AVX512_SUPPORT
```

4.3 Command Line Switches for Performance

You'll find a complete list of the SVT-HEVC configuration switches at the github. Table 2 lists switches associated with runtime performance. -lp and -ss are part of pinning strategies discussed in Chapter 6.

Table 2 SVT HEVC Command Line Switches for Performance

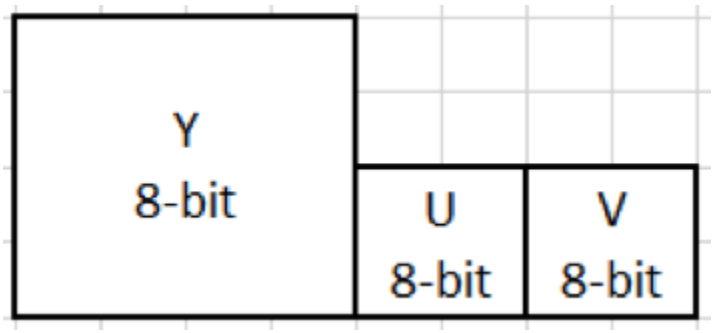
Optional HEVC parameter	Name	Description
-lp integer	logical processors	Sets the number of logical threads that will host encoder threads.
-ss integer	socket system	In a multi-socket team, use ss to set the socket that will host encoder threads
-nb integer	buffered input	Number of frames to pre-load to RAM before starting encode. This parameter eliminates the delay associated with reading from disk. Useful for experiments, Intel typically recommends the use of ramdisks as an alternative to -nb

5. Input Video Format

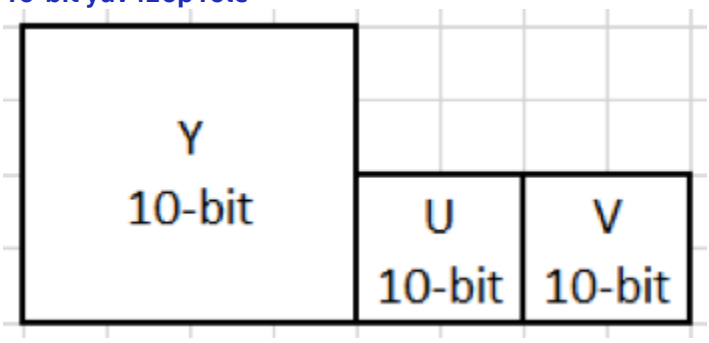
Input file format can have a significant impact on transcode performance. This section describes input file preparation techniques that can be used to improve performance.

The SVT-HEVC Encoder supports the following input formats:

8-bit yuv420p



10-bit yuv420p10le

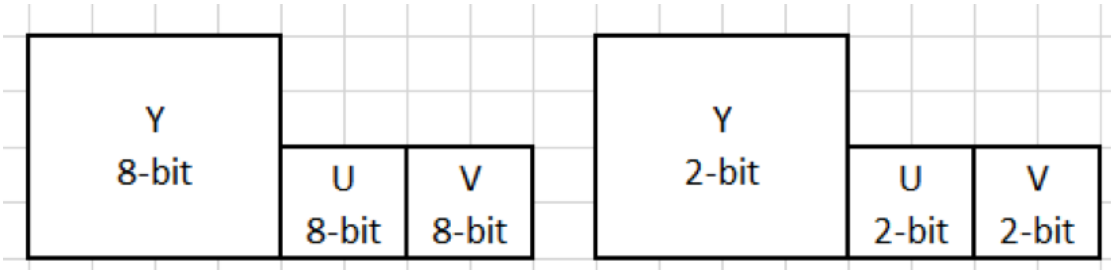


Compressed 10-bit format

In order to reduce the size of the input original YUV file, the SVT-HEVC Encoder uses a compressed 10-bit format allowing the software to achieve a higher speed and channel density levels. The conversion between the 10-bit yuv420p10le and the compressed 10-bit format is a lossless operation and is performed using the following steps.

Unpack the 10-bit picture

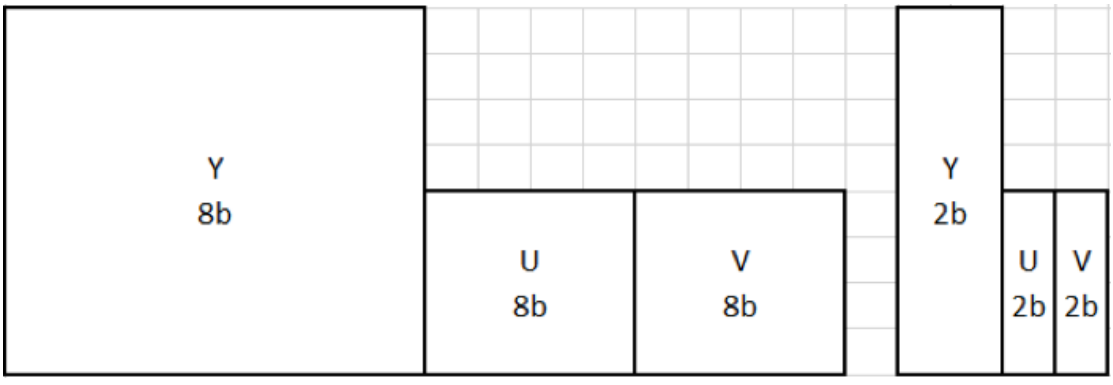
This step consists of separating the 10-bit video samples into 8-bit and 2-bit planes so that each 10-bit picture will be represented as two separate pictures as shown in the figure below. As a result of the operation, the 2 least significant bits of the 10-bits will be written into a full byte.



10-bit yuv420p10le unpacked

Compress the 2-bit Plane

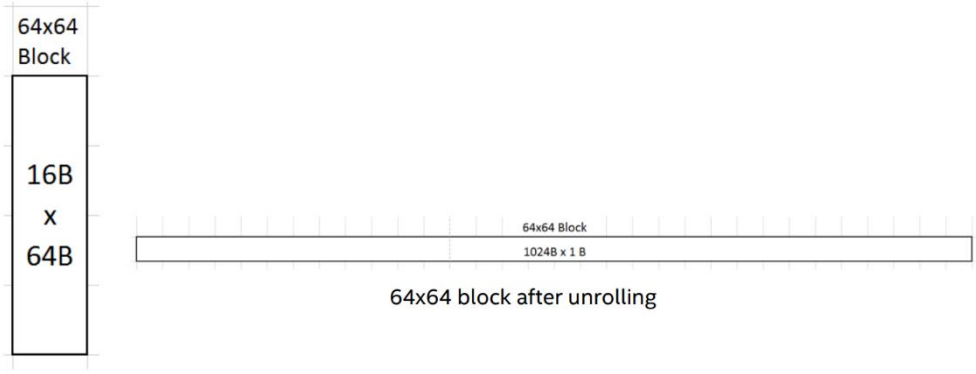
The unpacking steps separates the 10-bits into a group of 8-bits and a group of 2-bits, where the 2-bits are stored in a byte. In this step, every group of consecutive 4 bytes, each containing 2-bits from the unpacking step, are compressed into one byte. As a result, each 10bit picture will be represented as two separate pictures as shown in the figure below.



Unroll the 64x64

Now for a faster read of the samples, every 64x64 block of the 2-bit picture should be written into a one dimensional array. Therefore, the top left 64x64 sample block which is now written into 16 bytes x 64

bytes after the compression of the 2-bit samples, will be written into a 1024 bytes x 1 byte array as shown in the picture below.



64x64 block after 2 bit compression.

6. Invoking the Encoder

This section describes how to run the sample encoder application that uses the SVT-HEVC Encoder library. It describes the input video format, the command line input parameters and the resulting outputs.

The SVT-HEVC encoder achieves the best performance when restricting each channel to only one socket on either Windows* or Linux* operating systems. For example, when running four channels on a dual socket system, it's best to pin two channels to each socket and not split every channel on both sockets. Splitting a channel between sockets can lead to cross socket traffic, impacting the encoder performance. SVT-HEVC provides the -ss option to pin an encode instance to a specified socket. Refer to the following section for examples

The -lp option specifies the number of logical cores the encoder will utilize in execution. In cases where CPU utilization is limited, it is recommended to limit the number of logical cores using -lp. Limiting the number of logical cores reduces threading subscription overhead, resulting in improved performance.

6.1 Running a 6-stream 4kp60 simultaneous encode on Ubuntu

The example illustrates 6 concurrent 1:1 transcode from yuv to FHD4k/10bit/60fps on a dual socket system. Note that the –ss option is used in each command line to pin transcodes execution of that command to a specific socket. Commands 1-3 pin execution to socket 0, and commands 4-6 pin execution to socket 1. The –nb option is used to preload 500 frames into RAM before execution begins. This eliminates the impact of disk reading on encoding speed.

```
./SvtHevcEncApp -encMode 11 -w 3840 -h 2160 -bit-depth 10  -compressed-ten-bit-format
1 -i in.yuv -rc 1 –tbr 10000000 -fps 60 -b out1.bin -ss 0 -n 5000 –nb 500 &

./SvtHevcEncApp -encMode 11 -w 3840 -h 2160 -bit-depth 10  -compressed-ten-bit-format
1 -i in.yuv -rc 1 –tbr 10000000 -fps 60 -b out2.bin -ss 0 -n 5000 –nb 500 &

./SvtHevcEncApp -encMode 11 -w 3840 -h 2160 -bit-depth 10 -compressed-ten-bit-format
1 -i in.yuv -rc 1 –tbr 10000000 -fps 60 -b out3.bin -ss 0 -n 5000 –nb 500 &

./SvtHevcEncApp -encMode 11 -w 3840 -h 2160 -bit-depth 10 -compressed-ten-bit-format
1 -i in.yuv -rc 1 –tbr 10000000 -fps 60 -b out3.bin -ss 1 -n 5000 –nb 500 &

./SvtHevcEncApp -encMode 11 -w 3840 -h 2160 -bit-depth 10 -compressed-ten-bit-format
1 -i in.yuv -rc 1 –tbr 10000000 -fps 60 -b out4.bin -ss 1 -n 5000 –nb 500 &
```

```
./SvtHevcEncApp -encMode 11 -w 3840 -h 2160 -bit-depth 10 -compressed-ten-bit-format
1 -i in.yuv -rc 1 –tbr 10000000 -fps 60 -b out5.bin -ss 1 -n 5000 –nb 500
```

6.2 Running a stream on one Logical Core on Ubuntu

This example illustrates running a single encode instance from yuv to FHD4k/10bit/60fps on a dual socket system, forcing execution to one logical core. Note that the –lp option is used to specify the number of logical cores for an instance.

```
./SvtHevcEncApp -encMode 1 -w 3840 -h 2160 -bit-depth 10 -compressed-ten-bit-format 1
-i in.yuv -rc 1 –tbr 10000000 -fps 60 -b out3.bin -ss 1 -n 5000 –nb 500 -lp 1
```

6.3 Running a 2-stream 8kp50 simultaneous encode on Windows

The example illustrates 2 concurrent 1:1 transcode from yuv to FHD8K/10bit/50fps on a dual socket system. The –ss option is used in each command line to pin transcodes execution of that command to a specific socket. Command 1 pins execution to socket 0 while command 2 pins execution to socket 1. Also, the example uses –nb to preload 500 frames into RAM before execution begins. This eliminates the impact of disk reading on encoding speed.

```
SvtHevcEncApp.exe -encMode 11 -w 7680 -h 4320 -bit-depth 10 -compressed-ten-bit-
format 1 -i in.yuv -rc 1 –tbr 20000000 -fps 50 -b out1.bin -n 5000 –nb 500 –ss 0

start /node 1 SvtHevcEncApp.exe -encMode 11 -w 7680 -h 4320 -bit-depth 10 -
compressed-ten-bit-format 1 -i in.yuv -rc 1 –tbr 20000000 -fps 50 -b out1.bin -n 5000
–nb 500 –ss 1
```

6.4 Advanced Scheduling Flexibility with Process Affinity

If the advanced user desires ultimate flexibility with pinning workloads to specific CPU threads, this can be done with taskset or numactl in place of the -lp and -ss flags. Pinning workloads to specific threads can give the assurance that thread contention with other instances is not occurring and may offer increased performance. In the example below, the first instance is pinned to cores 0-12 and the second instance is pinned to cores 13-24. It should be noted that supporting multiple platforms with different CPU core counts can make pinning more challenging.

```
numactl -C +0,1,2,3,4,5,6,7,8,9,10,11 ./SvtHevcEncApp -encMode 11 -w 3840 -h 2160 -
bit-depth 10 -i in.yuv -rc 1 –tbr 10000000 -fps 60 -b out1.bin -n 5000 –nb 500 &

numactl -C +12,13,14,15,16,17,18,19,20,21,22,23 ./SvtHevcEncApp -encMode 11 -w 3840 -
h 2160 -bit-depth 10 -i in.yuv -rc 1 –tbr 10000000 -fps 60 -b out2.bin -n 5000 –nb
500
```

7. Conclusion

We understand every application is unique. We shared many of our experiences with SVT-HEVC Encoder hoping that some of our learnings could be applied to your specific application. SVT-HEVC has been well tested on Intel platforms. With 3rd Generation Intel® Xeon® Scalable processor, Intel takes it even further by optimizing the platform as a whole -- CPU, memory, storage, and networking working together for the best user experience.

8. Feedback

We value your feedback. If you have comments (positive or negative) on this guide or are seeking something that is not part of this guide, please reach out to us here:

https://community.intel.com/t5/Software-Tuning-Performance/bd-p/software-tuning-perf-optimization