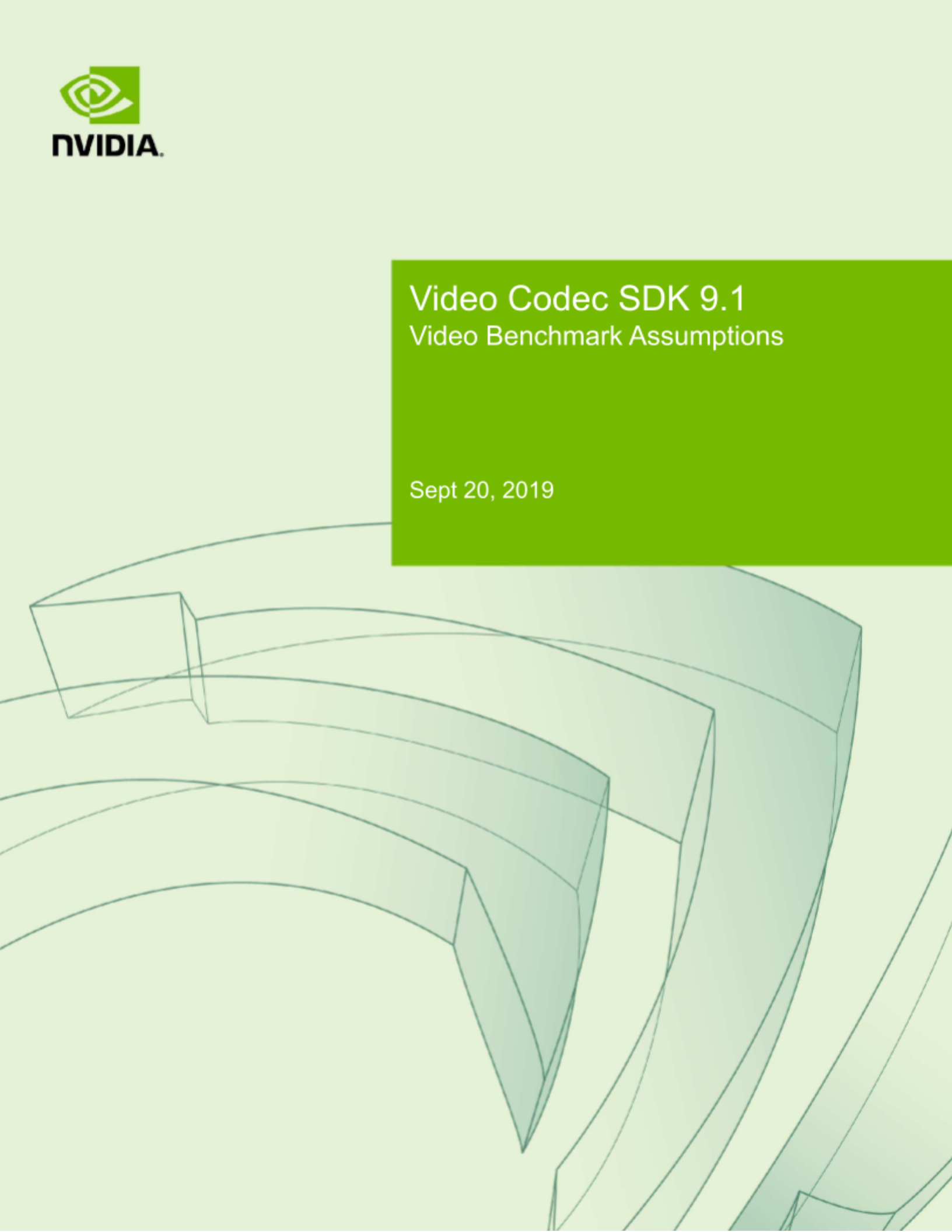

Video Codec SDK 9.1

Video Benchmark Assumptions

Sept 20, 2019

Table Of Contents

Motivation

Anyone familiar with video encoding knows that different encoders employ different strategies for achieving the bitrate vs quality targets. Depending on the platform specifics, optimization strategies may differ significantly. This causes peculiarities in the behavior of the encoder output and makes it difficult to perform an exact apples-to-apples comparison between two encoders.

A comparison of x264 with NVENC encoding has similar challenges. In this document, we provide details about the assumptions made in order to effectively compare x264 with NVENC.

1 Hardware Platform

The following table summarizes the details of the hardware used in our benchmarking setup.

Table 1. Details of hardware platforms used

System Cfg	**Tesla T4**	**Tesla P4**	**x264/x265**
CPU	Dual Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz		
GPU	Tesla T4 (TU104)	Tesla P4 (GP104)	N/A
RAM	128 GB	128 GB	128 GB
FFmpeg version	4.1	4.1	4.1
Driver	418.04	418.66	N/A

2 Content

Our encoding benchmark uses a large variety of video content from the following types of video footage:

1. Natural video - High motion (e.g. sports) and movie-type
2. Game captures - high motion and high texture
3. Synthetic content (e.g. animated movies)
4. Amateur video content (e.g. videos shot using camcorder)
5. Video conferencing

The library consists of several thousands of videos in resolutions 720p, 1080p, and 2160p (4K), with each video containing at least 600 frames.

The performance of the encoder is somewhat content-dependent. It is important to ensure that while averaging the performance, content of various types and resolutions is used to ensure that the results represent true behavior of the encoder.

3 Latency Tolerant and Latency Sensitive Encoding

There are two types of use-cases for which video encoding benchmark is executed

1. **Latency-tolerant:** Used in applications such as video archiving, streaming with high latency (> 0.5 seconds), video storage, web videos, video streaming (e.g. Netflix). This type of encoding typically has no restrictions on the encoding tools that can be used, subject to the complexity constraints. Features such as B-frames, look-ahead can be used.
2. **Latency-sensitive:** Used in latency-sensitive applications such as cloud gaming, game-streaming, game broadcasting. These applications cannot tolerate latency more than a couple of frames. Encoding tools such as B-frames, look-ahead cannot be used in this type of encoding. This type of encoding also puts a strict cap on frame-by-frame bit budget and expects strict HRD compliance at small VBV buffer size.

NVIDIA provides benchmarks in both the above use-cases.

Encoding

Each video from the library is encoded at 4 or 5 different bitrates, depending on the resolution, using libx264/libx265 and NVENC options within FFmpeg. For latency-sensitive and latency-tolerant use-cases, different performance/quality operating points are defined using FFmpeg command-line options. Specific command line options used for these operating points are provided below for reference.

Table 2. Latency-tolerant H.264 encoding parameters

Codec	**FFmpeg command line parameters**
NVENC H.264	-c:v h264_nvenc -preset fast/medium/slow -rc vbr -b:v BITRATE -profile:v high -bf 3 -b_ref_mode 2 -temporal-aq 1 -rc-lookahead 20 -vsync 0

libx264	-c:v libx264 -preset fast/medium/slow -b:v BITRATE -tune psnr -vsync 0 -threads 4

Table 3. Latency-sensitive H.264 encoding parameters

Codec	**FFmpeg command line parameters**
NVENC H.264	-c:v h264_nvenc -preset llhp/ll/llhq -rc cbr_ld_hq -b:v BITRATE -bufsize BITRATE/FRATE -profile:v high -g 999999 -vsync 0
libx264	-c:v libx264 -preset veryfast/faster/fast/medium/slow -b:v BITRATE -bufsize BITRATE/FRATE -maxrate BITRATE -g 999999 -x264opts no-sliced-threads:aq-mode=0:no-psy -tune zerolatency -threads 1 / 2 (720p / 1080p) -vsync 0

Table 4. Latency-tolerant HEVC encoding parameters

Codec	**FFmpeg command line parameters**
NVENC HEVC	-c:v hevc_nvenc -preset fast/medium/slow -rc vbr_hq -b:v BITRATE -profile:v 4 -bf 2 -rc-lookahead 20 -g 250 -vsync 0

libx265	-c:v libx265 -preset fast/medium/slow -b:v BITRATE -bf 2 -tune psnr -threads 4 -vsync 0

Table 5. Latency-sensitive HEVC encoding parameters

Codec	**FFmpeg command line parameters**
NVENC HEVC	-c:v hevc_nvenc -preset llhp/ll/llhq -rc cbr_ld_hq -b:v BITRATE -bufsize BITRATE/FRATE -maxrate BITRATE -profile:v 4 -bf 0 -g 999999 -vsync 0
libx265	-c:v libx265 -preset veryfast/faster/fast/medium/slow -b:v BITRATE -bufsize BITRATE/FRATE -maxrate BITRATE -g 999999 -tune zerolatency -x265-params no-sliced-threads:aq-mode=0:no-psy-rd -threads 1 / 2 (720p / 1080p) -vsync 0

7 Evaluation of Results

After encoding each video, metrics such as SSIM, PSNR and encoding performance are measured. To measure encoding performance, we measure the time taken to encode all frames at the application level. If multiple files are being encoded in parallel, then the aggregate number of frames in all parallel encoded videos are used to compute performance in frames/second.

In addition to these, the videos are visually inspected to confirm that there are no distortions or unexpected artifacts.

The rate-distortion characteristics of each encoded video are analyzed using PSNR and SSIM and we calculate metrics such a BD-PSNR, BD-SSIM and BD-BR for each video (BD = Bjontegaard metric). While calculating BD metrics, we use actual bitrate in latency tolerant encoding and target bitrate in latency sensitive

encoding. The reason for using different bitrates is as follows: For latency sensitive encoding, it is highly desirable that encoder generates output at bitrate as close to the target bitrate as possible because any bits saved by not reaching the target bits are “wasted”, because of real time transmission constraint in a bandwidth-limited channel. Therefore, for latency sensitive encoding, we use target bitrate. For latency tolerant encoding, the constraint of bandwidth limitation is not so strict and hence bit allocation strategy can be more flexible, and it is fine to use the actual bitrate generated by the encoder.

The BD metrics are averaged over all videos at a given resolution to generate charts such as those shown at http://developer.nvidia.com/nvidia-video-codec-sdk.