vbench: Benchmarking Video Transcoding in the Cloud

Andrea Lottarini[1,*] Alex Ramirez[2] Joel Coburn[2] Martha A. Kim[1]
Parthasarathy Ranganathan[2] Daniel Stodolsky[2] Mark Wachsler[2]
[1]Columbia University [2]Google

Abstract

This paper presents vbench, a publicly available benchmark for cloud video services. We are the first study, to the best of our knowledge, to characterize the emerging video-as-a-service workload. Unlike prior video processing benchmarks, vbench's videos are *algorithmically* selected to represent a large commercial corpus of millions of videos. Reflecting the complex infrastructure that processes and hosts these videos, vbench includes carefully constructed metrics and baselines. The combination of validated corpus, baselines, and metrics reveal nuanced tradeoffs between speed, quality, and compression.

We demonstrate the importance of video selection with a microarchitectural study of cache, branch, and SIMD behavior. vbench reveals trends from the commercial corpus that are not visible in other video corpuses. Our experiments with GPUs under vbench's scoring scenarios reveal that context is critical: GPUs are well suited for live-streaming, while for video-on-demand shift costs from compute to storage and network. Counterintuitively, they are not viable for popular videos, for which highly compressed, high quality copies are required. We instead find that popular videos are currently well-served by the current trajectory of software encoders.

ACM Reference Format:
Andrea Lottarini, Alex Ramirez, Joel Coburn, Martha A. Kim Parthasarathy Ranganathan, Daniel Stodolsky, and Mark Wachsler. 2018. vbench: Benchmarking Video Transcoding in the Cloud. In *Proceedings of 2018 Architectural Support for Programming Languages and Operating Systems (ASPLOS'18).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3173162.3173207

1 Introduction

Video sharing represents a growing fraction of internet traffic. For example, in the November 2016 Facebook earnings presentation, Mark Zuckerberg described Facebook's evolution into a "video first" company [12]. The 2016 Sandvine Global Internet Phenomena report [33] places audio and video at 71% of evening traffic in North America and projects that figure will grow to 80% by 2020. Video processing plays a pivotal role in virtual and augmented reality (Oculus Rift, HoloLens), video surveillance (Nest), cloud gaming (GeForce Now, PlayStation Now), and other emerging applications.

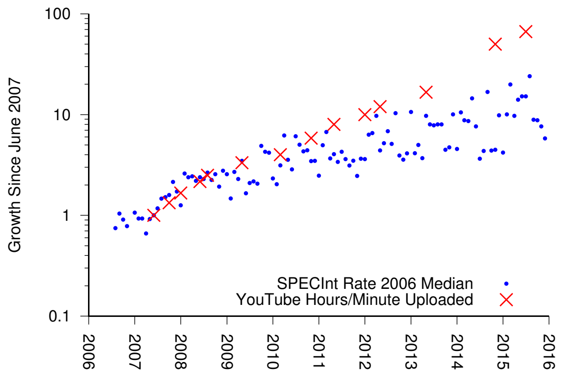

Figure 1. Many hours of video are uploaded to YouTube every minute [21]. The uploads are growing more rapidly than CPU performance (as measured on SPECRate2006), which creates a growing burden on sharing infrastructure.

To keep up with growing usage, video on demand providers such as Netflix, YouTube, and Facebook maintain large video serving infrastructures. All these services perform a large number of transcoding operations [40], i.e., decoding a compressed video into raw frames and re-encoding it in a new compressed format. Each uploaded video is transcoded at least once before it is sent to viewers. This ensures videos that are malformed are not distributed. Even more importantly, each upload must be converted to a range of resolutions, formats, and bitrates to suit varied viewer capabilities, i.e., screen resolution, codecs supported, and available network bandwidth. In every transcoding operation, there is a trade-off between compressed video size, fidelity to the original video, and transcoding time. For example, reducing video size may reduce visual quality but encourages smooth playback, thus potentially improving the overall quality of experience.

Within a transcode operation, the decoding step, which converts a compressed video stream into a sequence of frames

*The work was done when this author was an intern at Google.

to be displayed, is deterministic and relatively fast. In contrast, the encoding step has to make many decisions that can not be exhaustively explored, so encoders perform a heuristic search over this decision space. Increasing the space searched, also known as the *effort level*, increases the likelihood of finding a better transcode, i.e., a compressed video with less distortion, or lower bitrate.

Transcoding is ripe for optimization. As Figure 1 depicts, demand is outstripping CPU performance, and within Google, the cycles spent transcoding have grown by 10x in the last two years. Research is needed in order to advance areas like hardware video transcoding, evaluation of new video codecs, and related technologies. However, *there is no well-defined way to compare transcoding solutions*. In the video processing community, encoders are evaluated in terms of visual quality and bitrate. Large scale studies [7] show increases in compression rate without hurting video quality. Computation time, however, is not typically measured and thus also increases: as new codecs introduce new knobs and parameters, the search space grows. Our case study in Section 6.2 demonstrates this effect. In the architecture community, two widely used benchmark suites, SPEC [19] and PARSEC [2], include some video encoding use cases. However, the video content and settings are not representative of a video sharing infrastructure.

To establish some common ground, this paper presents a video transcoding benchmark, vbench, that reflects the transcode demands of a video sharing service such as YouTube. YouTube receives videos in thousands of combinations of resolution, framerate, and complexity (entropy). vbench uses clustering techniques to select 15 videos of 5 seconds each. This is a small enough number to allow detailed RTL or microarchitecture simulations, but wide enough to cover a significant cross section of the corpus of millions of videos. vbench also establishes a set of reference transcode operations against which other transcoder proposals can be compared. These operations are comparable with operations that are performed at providers like YouTube. This ensures a consistent and appropriate baseline. Finally, vbench includes five comparison metrics that derive from real-world transcoding scenarios. These metrics guide meaningful improvements by enforcing constraints associated with video transcoding at scale.

We demonstrate the value of vbench with four use cases. First, we show how the choice of input videos can change the apparent microarchitectural trends and qualitative conclusions from performance studies. Second, we quantify the limits of SIMD vectorization for transcoding: consistent with other studies [17], we find that SIMD instructions can provide limited improvements. Next, we evaluate current GPU support for video transcoding, finding non-intuitive tradeoffs in their behavior. While GPUs are a clear win for live transcoding tasks, they sacrifice compression and quality for video archival. Lastly, we find that while GPUs today cannot meet the strict quality and compression targets for popular videos, newer and more complex software encoders can. Collectively, these studies demonstrate the relevance of our benchmark and the importance of having a curated set of videos, meaningful baselines, and encoding scenarios to evaluate new transcoding solutions.

2 Background

This section provides some background on video transcoding techniques, how they are evaluated, and the video sharing infrastructures where they play a crucial role.

To understand the importance of transcoding, consider that a raw Full-HD frame (1920 * 1080 pixels) is roughly 3MB. Streaming uncompressed video at 30 frames/second would require 90 MB/s, or 700 Mb/s, exceeding the capabilities of most home broadband installations.

Since streaming raw video is not practical, it must always be compressed. A video *transcoder* is a device or program that decodes a compressed input video into a raw, uncompressed format and then re-encodes it in a new compressed format. It is equivalent to a decoder and encoder chained together. While video decoding simply follows the interpretation rules for the bitstream of the video format, video encoding has a number of degrees of freedom to decide how the raw video should be compressed. Video encoding formats, like H.264/AVC [38], H.265/HEVC [34], or VP9 [28], are usually referred to as *codecs*.

2.1 Video Transcoding

Video encoders exploit properties of human perception as well as spatial and temporal redundancy in the video content. Humans perceive changes in luminosity more than changes in color, therefore video processing is performed on the YUV color space, rather than in RGB. YUV separates luminosity signal (luma) from color information (chroma) allowing encoders to dedicate more bits for the luma plane than the chroma plane (a process called chroma subsampling). They also rely on the fact that blocks of pixels in a frame are usually similar to other blocks of pixels in previous (and future) frames. Encoders take advantage of this similarity by expressing part of a frame as a function of blocks in other *reference frames*.

Video encoders generally adhere to the following template: First, video frames are decomposed in square blocks of pixels called *macroblocks*. For each macroblock, the encoder searches temporally neighboring frames for similar macroblocks (*motion estimation*). This initial *motion estimation* is usually the most computationally onerous step [17]. Once a suitable reference block is found, the encoder computes the difference (the *residual block*) and stores only the relative location (the *motion vector*). Residual blocks are then encoded like a regular image [31]: A *discrete cosine transform*

(DCT) is used to convert blocks of pixels to the 2D spatial frequency domain. Then the matrix of coefficients is *quantized*, i.e. divided point-wise by another matrix (the *quantization matrix*) to introduce zeroes[1]. Quantization zeroes out the high frequency components (quick pixel transitions) which are less noticeable to the viewer. Quantization is the only lossy part of the process. The more zeroes introduced this way, the more effective the final compression step, in which each frame is losslessly compressed via *entropy encoding*, *e.g.* Context Adaptive Binary Arithmetic Coding (CABAC) or Context Adaptive Variable Length Coding (CAVLC) [26].

New codecs introduce new compression tools and algorithms, like the H.264 *deblocking filter*, which removes artifacts that can appear at the boundaries between macroblocks. Denoising is another optional operation that can be applied to increase video compressability by reducing high frequency components [23].

2.2 Encoding Effort

Video encoding requires the user to specify a target quality. If the user specifies a constant rate factor (CRF), the encoder will try to sustain the same quality level for all video frames, using as many bits as necessary. Alternatively, to make the video size predictable, the user can specify a target bitrate (bits per second); The encoder will try to fit the video in the allocated space, but may sacrifice quality to do so.

When encoding to a target bitrate, 2-pass encoding can optimize the allocation of bits to the more complex parts of the video. On the first pass, the encoder records how complex each frame is and uses that information in the second pass to budget fewer bits for simple frames, and more for complex frames.

The Rate Distortion Optimizer (RDO) decides how to gracefully degrade visual quality in order to meet the target bitrate. A sample RDO decision would be the post-DCT quantization strength. More complex decisions include how to decompose the frames into macroblocks or whether to perform sub-pixel motion estimation.

The difficulty of the RDO's job is input dependent. Videos with static images, such as slideshows or animations, are easily compressed since motion vectors describe most of the frames with precision. On the other hand, videos with high motion and frequent scene changes will require more time for motion search, and other optimizations to fit the frames in the allowed bitrate.

It is possible to specify an encoder effort level that affects the RDO decisions. RDO decisions at each stage of encoding entail difficult-to-predict tradeoffs between quality and bitrate. As a consequence, the whole encoding process resembles a heuristic search. Performing more computation, i.e. covering more combinations in the encoding space, ensures that better transcodes are found. The effort level restricts the parameters (motion search range, number of reference frames, etc.) used in the search for a better encoding. Higher effort will achieve higher quality at the same bitrate, at the expense of longer encoding time.

[1]Potentially the entire residual block can be discarded if equal to zero after quantization.

2.3 Transcoding Metrics

Video transcoding must be evaluated in three dimensions: visual quality, video size, and transcoding speed.

Visual quality is measured by comparing the original uncompressed frames with the same frames in the encoded version. *Peak signal-to-noise ratio* (PSNR) captures the ratio between the maximum error per pixel and the actual error per pixel, so larger values indicate higher quality. Given an initial raw frame F and its transcoded version T, both of $m \times n$ pixels, PSNR is obtained by computing the mean square error (MSE):

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (F(i,j) - T(i,j))^2$$

The MSE is then compared against the maximum pixel value of the frame, typically 255 for standard 8 bit pixels.

$$PSNR = 10 \log_{10}(\frac{255}{\sqrt{MSE}})$$

This process can be repeated for all planes, luma (Y) and chroma (Cb,Cr) of all frames and averaged to compute the average YCbCr PSNR. We use average YCbCr PSNR in the remainder of the paper as a measure for transcoding quality.

There are alternative "perceptual quality" metrics such as Structural Similarity (SSIM [37]), and those recently proposed by Netflix [24] and Google [6]. These metrics try to capture the specifics of human visual perception into an analytic method. They all assume that the original video is uncompressed. However, YouTube uploads generally arrive already encoded and thus potentially distorted. Furthermore, there is no consensus in the video processing community as to which one of these metrics works best. Therefore, we rely on the "objective" PSNR for the rest of this work.

Video size is usually measured by *bitrate*, the number of bits per second of video. While actual video file size depends on the length of the video, bitrate is a video-length-normalized metric. Decreasing the bitrate of a video stream decreases the likelihood of re-buffering events, i.e. video data packets not delivered on time for playback. To compare videos at different resolutions, we report bitrate normalized by the number of pixels in each frame (bits per pixel per second).

Transcoding speed, like bitrate, is normalized against the length of the video, and the resolution of each frame. We multiply the number of frames transcoded in a second by the number of pixels in a frame and report the number of pixels transcoded per second.

2.4 Evaluating a Transcoder

Since video quality is clearly related to the bitrate made available to the video encoder, the video community compares video transcoders using *PSNR curves* that plot video quality as a function of the video bitrate. Figure 2 (top) shows PSNR curves for three different software encoders on one HD video[2].

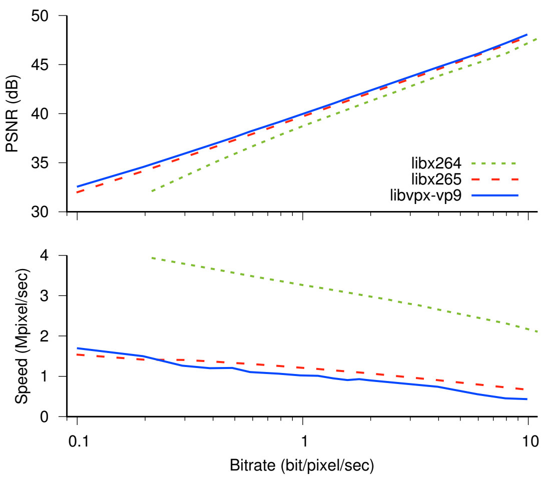

Figure 2. Video transcoding is usually compared on video quality vs. video bitrate, but that leaves out a critical third dimension: the transcoding speed.

The curves show that libvpx-vp9 achieves slightly better quality than libx265, and both achieve substantial improvements over libx264 for all target bitrates. That would indicate that libvpx-vp9 is a superior transcoder, since it can always provide better quality at the same video size, or smaller videos at the same quality.

However, when transcoding speed is factored in Figure 2 (bottom), we observe that the advantage of libvpx-vp9 over libx265 corresponds to a decrease in transcoding speed, and that both of them require 3-4x more computation than libx264. It is no longer obvious which one is the best encoder.

The answer depends on the use case. Sometimes a fast transcode is needed, *e.g.* when streaming a live event, so it is necessary to trade bitrate and/or quality to ensure that streaming is fluid. Conversely, when a video is expected to be played many times, it is worth using an advanced encoder, since the cost of producing a smaller video at equivalent perceptual quality is amortized, and the bitrate savings are multiplied, across the many playbacks. A video sharing infrastructure is designed to efficiently manage all these decisions.

[2]The first 1000 frames of Big Buck Bunny [3], used in SPEC 2017.

2.5 Video Sharing Service Architecture

A video streaming service such as Netflix, YouTube, or Facebook allows users to view, rate, upload, and comment on videos. They serve media to a wide variety of devices, from web browsers to mobile applications, with media content ranging from movies, television programs, music videos, video games, and documentaries, to animations, slideshows, and screen capture tutorials.

These services incur three primary costs: storage, network, and compute. The storage cost is proportional to the size of the corpus of videos stored in a central repository, including duplicates in various resolutions and formats, as well as replication across a Content Distribution Network (CDN) for faster service [4, 32]. The network cost is determined by the egress traffic from the central repository and/or CDN to users and, to a lesser extent, by the ingress traffic of new uploads. The compute cost is incurred each time a video is transcoded.

To optimize these costs, video streaming services make multiple video transcoding passes on each video, as illustrated in Figure 3. Videos are uploaded in a wide variety

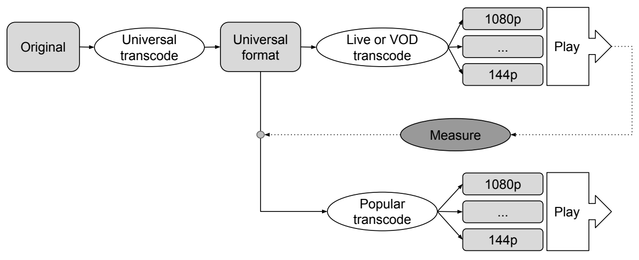

Figure 3. Video transcoding passes in a video sharing infrastructure. Higher effort is invested in popular videos watched many times.

of combinations of codec, container, color space, resolution, frame rate, etc. [11]. To apply a uniform process, all originals are first transcoded to an universal format that functions as an intermediate representation for the system.

From there, videos are transcoded to a wide variety of formats and resolutions to suit the capabilities of the network and the different client platforms. Depending on whether the video is being forwarded directly to clients, i.e. live streaming, or transcoded offline and stored to be viewed later, i.e. video on demand or VOD, this encoding can be single pass for low latency, or two pass for higher quality and smaller size. Newly uploaded videos must be available for playback as soon as possible, especially for live streaming, so the latency of these transcoding steps must be bounded.

Video popularity follows a power law distribution with exponential cutoff [5]: most of the watch time concentrates in a few popular videos, while there is a long tail of rarely watched videos. It would be wasteful to invest much compute effort on the long tail, but when a video is observed to be popular, services will spend the extra compute. Those videos are transcoded a second time at higher effort levels to produce high quality compressed replicas that optimize user experience, storage and network costs. The extra compute time is amortized across many playbacks of the video, while the savings are multiplied across playbacks.

3 Prior Work

Innovation in transcoding is both relevant and pressing. Figure 1 shows how the growth of transcoding demand greatly outpaces computational improvements in today's CPUs.

Benchmark suites have been developed for many classes of datacenter workloads, such as personal assistants [18], web applications [44], big data analytics [36] as well as datacenter workloads at large [13]. However, there is no suite specifically targeting video transcoding.

When transcoding does appear in popular CPU benchmarks, neither videos, nor the transcoding operations are representative of a video sharing infrastructure. For example, SPEC 2006 [19] includes the H.264 reference encoder with two low-resolution videos. SPEC 2017 uses the libx264 encoder and two segments of a HD video. PARSEC [2] includes a pthread parallelized version of the libx264 encoder and one input video. With such a limited set of videos and use cases, the complexity of video transcoding at scale can't be captured.

Netflix released a dataset of 9 videos from an internal suite of 34 video clips from popular TV and movies from their catalog [24]. This curated data set has been used in a study proposing a perceptual quality metric, as opposed to signal fidelity metrics like PSNR. The Alliance for Open Media is using Derf's HD video collection from Xiph.org [41] for the development of the AV1 codec [15]. The Derf collection contains 41 videos from 480p to 4K resolution. In both cases the rationale for inclusion is either absent (Xiph) or follows qualitative criteria such as diversity of subjects, lighting conditions, etc. (Netflix). An additional drawback is that these are only datasets, with no associated transcoding operations, or scoring metrics.

HD-VideoBench [1] is a notable previous attempt at benchmarking video transcoding that however lacks the diversity in both input sequences (only 4 videos obtained from the MPEG-Test Sequences archive) and scenarios (single pass constant quality encode only) that characterize a video sharing infrastructure. All their video sequences are included in the Xiph.org collection.

Prior architectural work in video transcoding does not use rigorously selected videos [17, 25, 30], or compares against unoptimized encoders [17, 19] that underperform compared to state of the art software solutions. As a consequence, it is difficult to translate their insights for video sharing infrastructures, since the implications on the quality of user experience, storage, and network costs can not be predicted from the reported results. As an example, Fouladi et al. recently implemented a system [14] to perform low latency transcodes using AWS Lambda. Their evaluation was performed using two movies from the Blender Foundation [3] (Sintel and Tears of Steel) and no rationale for this choice is stated. Zhang et al. investigate how to reduce video decoding energy consumption in mobile devices [43]. A mix of videogames and movie trailers video sequences are used for the evaluation. Again, vbench would be more representative of the content that mobile devices receive from video sharing infrastructures.

There has been recent work documenting how large video sharing infrastructures operate [20] or optimize for popular videos [35]. Results obtained using vbench should also apply to these systems.

Magaki et al. explore the possibility of introducing ASICs into the datacenter to process large scale workloads with a lower Total Cost of Ownership (TCO) [25]. Their analysis names video transcoding as a possible candidate. However, before building an ASIC, a performance analysis of the workload to accelerate is paramount.

Overall, the state of the art is not conducive to controlled comparisons between transcoding solutions. This lack of data and common benchmarks hampers innovation in this critical area.

4 Transcoding Benchmark

In this section, we describe vbench, our video transcoding benchmark consisting of input videos (Section 4.1), scoring functions (Section 4.2), and reporting rules (Section 4.3). All of the videos, reference data, and scripts are publicly available on the vbench website *http://vbench.net*.

4.1 Video Selection

The input videos must achieve a complex trade-off between representativeness and coverage. They must be representative so that the results match what is observed on a production system, but provide coverage to expose trends and not ignore corner cases that may become increasingly important in the future. Moreover, the number of videos must be constrained to facilitate adoption: while real machines can easily transcode many videos, that is not feasible for microarchitectural or RTL simulation.

Video feature selection.

From the many features describing a video, we determined three to have the greatest impact on transcoding:

- *resolution*, because higher resolution frames will require a higher bitrate to encode, and will also require more time to encode,
- *framerate*, because high framerate videos ($>$ 30 frames/s) will require a higher bitrate to encode, and

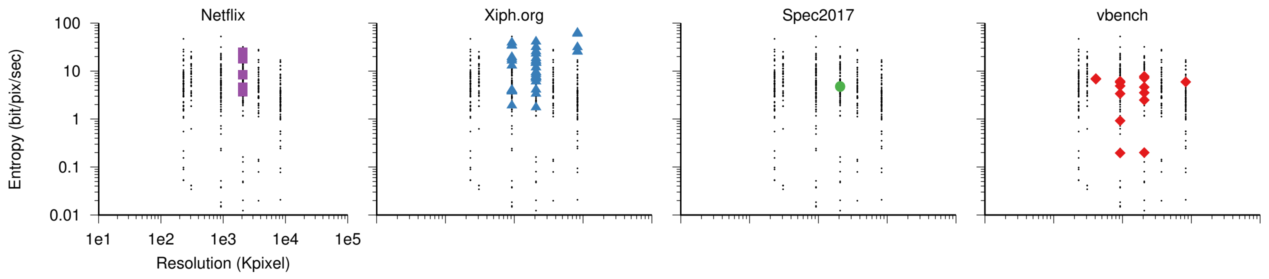

Figure 4. Black dots are a uniform sample of videos in the various resolutions and complexities uploaded to YouTube. Colored dots show how public video sets cover only a fraction of the space.

- *entropy*, because videos with high motion, or frequent scene transitions will require a higher bitrate to encode and higher effort (for motion search and other tools), or will incur quality losses.

While resolution and framerate are simple to understand and measure, entropy requires some explanation. Throughout the paper, we use bits/pixel/second when encoded using libx264 at *visually lossless* quality (Constant Rate Factor CRF 18) as a measure for video entropy[3]. As described in Section 2.2, when an encoder is asked to generate a fixed target quality, it will use as many bits as needed to do so, and thus the number of bits used by the encoder in this setting reflects the inherent entropy of the video content.

From these three characteristics, we define a video *category* as the set of videos that have the same resolution, measured in Kpixels/frame ($\frac{width \times height}{1000}$, rounded to integer), framerate (frames per second, rounded to integer), and entropy (bits per pixel per second when encoded using libx264 at constant quality – constant rate factor 18 – rounded to one decimal place).

Selecting video categories.

From logs of all the video transcoding operations at YouTube from January to June 2017, we accumulate the total transcoding time spent on each video category. This yields over 3500 video categories with significant weights (40+ resolutions and 200+ entropy values).

We use k-means clustering to select a small set of categories – particular triplets (resolution, framerate, entropy) – from that 3-dimensional space. Prior to clustering, we linearize resolution using the base two logarithm. This ensures that the large distance between standard resolutions does not bias the clustering algorithm. We also use the base two logarithm of the entropy to quantify the relative difference between videos: videos of entropy 1 and 2 bit/pixel/s are much more different than videos of entropy 20 and 21 bit/pixel/s. Lastly, we normalize all dimensions to a [-1, +1] range. We then apply weighted k-means clustering to find a pre-defined number of centroids, with weights determined by the time spent transcoding for each category of videos. Since each centroid covers multiple categories, we select the category with the highest weight in the cluster – i.e., the mode – as the cluster representative.

This process achieves both representativeness, since we select the mode as cluster representative, and coverage, since all videos must be associated with a cluster.

Selecting actual videos.

The k-means clustering defines a reduced set of ideal video categories. We then select a random video from the YouTube corpus belonging to each selected category. To ensure our benchmark is redistributable, we restrict the selection pool to videos that were uploaded with a Creative Commons Attribution 3.0 (CC BY) license [10].

Finally, we split the full-length original videos into non-overlapping 5-second chunks, and select the chunk with the bitrate that best matches the average bitrate of the video. We limit videos to 5 seconds since it has been observed to be the optimal duration for subjective video quality assessment [27]. We verify that removing the creative commons restriction creates no significant difference in our results or insights. The videos that compose vbench are summarized in Table 2.

Coverage.

Our process ensures that the chosen videos are representative, with each sequence covering a significant fraction of the entire corpus. However, not all categories can be covered. We therefore compare our benchmark with an internal YouTube coverage set that collects 11 uniformly distributed entropy samples from the combination of the top six resolutions and the top eight framerates. These 36 resolution and framerate combinations account for more than 95% of the Youtube uploads. Figure 4 shows one black dot for each video in this set, and overlays the different public video sets – plus our own, vbench – on top to evaluate coverage.

Note that the entropy range is four order of magnitude wide, from still images and slideshows (entropy < 1) to high

[3]libx264 Constant Rate Factor (CRF) goes from 0 to 51. CRF 0 is lossless compression, CRF 23 is the default value, and CRF 18 is generally considered *visually lossless* [29].

motion videos with frequent scene transitions (entropy > 10). In contrast, the Netflix and Xiph datasets focus only on high entropy videos (entropy ≥ 1) as they are intended for visual analysis. Furthermore, the Netflix dataset contains a single resolution (1080p). As we show in Section 5.1, the lack of low entropy videos introduces significant bias in the results using this video set. SPEC'06 and the latest SPEC'17 contain only two video sequences. This is clearly not enough for a video benchmark. Moreover, the resolution of SPEC'06 videos is not representative (too small). This is improved in SPEC'17, however the two videos used in this case have almost identical entropy (Figure 4) as they are obtained from the same animation.

vbench achieves better coverage in both resolution and entropy than all of these other alternatives, and has fewer and shorter videos than Xiph.org, facilitating adoption.

4.2 Transcoding Scenarios

To capture the nuances of the various video processing pipelines outlined in Section 2.5, vbench distinguishes five scoring scenarios. Each scenario reflects the *constraints* and *priorities* of its corresponding real-world scenario: (1) uploading a new video, (2) live streaming, (3) archiving for video on demand, (4) optimizing popular videos, and (5) optimizing the hardware platform.

For each scenario we provide reference measurements, namely speed (in Mpixel/sec), bitrate (in bits/pixel/sec), and quality (in dB), all normalized to video resolution and duration to allow comparison across videos. The measurements for each of these scenarios are taken using ffmpeg with libx264 on a Intel Core i7-6700K CPU @ 4.00GHz with 4 cores and 8 threads. Each of these reference transcoding operations is a measuring stick, grounded in real-world video sharing infrastructure, with which to compare transcoding solutions. All ffmpeg parameters used are reported in the vbench website (*http://vbench.net*).

The Upload reference is single pass with a constant quality target, allowing the encoder to use as many bits as needed to maintain the quality of the original. The Live reference is single pass, low latency, with a fixed bitrate target; the encoder effort is lower for higher resolution videos to ensure that the latency constraints are met. The VOD reference is the average case and is the same as the Platform reference: two-pass encoding with a fixed bitrate target. Finally, the Popular reference is high-effort two-pass encoding. The reference measurements are **scientifically essential**. They ensure that vbench results reported by different groups are directly comparable, and that the baseline is meaningful.

Users of the benchmark will try to improve on the reference transcoding operations provided. vbench uses ratios (speedups) between a new system and a reference transcode to indicate improvement. Values greater than 1 indicate the new solution is *better* in that dimension.

$$S = \frac{Speed_{new}}{Speed_{ref}} \quad B = \frac{Bitrate_{ref}}{Bitrate_{new}} \quad Q = \frac{Quality_{new}}{Quality_{ref}}$$

Since video transcoding entails a trade-off between speed, size, and quality, it is unlikely that a new solution will Pareto dominate the reference transcodes on all three dimensions. In each vbench scoring function, one dimension is eliminated via a strict Quality of Service constraint, which is reflective of the particular transcoding scenario targeted, leaving ratios for the remaining two dimensions. These can be analyzed directly or it is possible to condense each video down to a score by multiplying the two ratios, similar to an energy-delay product [16]. These scores, summarized in Table 1, are easy to compare yet reflective of nuanced real-world constraints and trade-offs.

Scenario	Constraint	Score
Upload	when $B > 0.2$	$S \times Q$
Live	when $S_{new} \geq outputMpixel/s$	$B \times Q$
VOD	when $Q \geq 1$ or $Q_{new} \geq 50dB$	$S \times B$
Popular	when $B, Q \geq 1, S \geq 0.1$	$B \times Q$
Platform	when $B, Q = 1$	S

Table 1. vbench scoring functions and constraints.

The **Upload** transcoding pass requires speed and quality: the video should be available for further processing as soon as possible, while not degrading the quality of the uploaded original. On the other hand, bitrate can be almost arbitrarily large because it is only a temporary file. We therefore require the bitrate be no larger than 5x the reference ($B > 0.2$) when reporting Upload scores of $S \times Q$.

Live streaming must happen in real time, so transcode must not lag behind the pixels per second of the output video. The Live score is then $B \times Q$.

In the **VOD** scenario, one cannot degrade quality compared to the reference, as this would have negative effects on user experience. However, provided quality is maintained ($Q \geq 1$) or the transcode is visually lossless ($Q_{new} \geq 50dB$) one can report a VOD score of $S \times B$.

High-effort optimizations for **Popular** videos should always produce smaller videos of higher quality. Improvements on visual quality and reduction in network bandwidth will improve user experience, while extra compute cost of re-transcoding popular content will be amortized across many playbacks of these popular videos. In this case, we report bitrate and quality: $B \times Q$ (if $B \geq 1$ and $Q \geq 1$). While speed is not critical in this scenario, it should still be bounded to a 10x slowdown ($S \geq 0.1$).

The final vbench score captures the case where the encoding algorithm and settings are constant and only the **Platform** changes. Innovations evaluated in this scenario are the same as SPEC benchmarks: compilers (icc vs gcc), architecture (x86 vs PPC), and microarchitecture (cache size, branch prediction, etc.). The scoring function assumes that bitrate and quality will be unaffected, and thus the two platforms can be compared by reporting S (if $B = 1$ and $Q = 1$).

4.3 Reporting Results

For each scenario, a complete run of the benchmark requires a transcode operation for each input video sequence that is compared against the reference. Each transcode operation results in three values – speed, bitrate, and quality – reported individually. For each video, if the constraints specific to the scenario are satisfied, scoring metrics described in the previous section can be computed. Given the diversity of the videos, results should not be aggregated into averages as significant information would be lost. Each video reflects some segment of the video sharing workload, so that providers, who know the specifics of their corpus and service costs can weigh the information accordingly, similar to what is done today for SPEC.

We demonstrate how benchmark results should be reported in the next section.

5 Bridging the Performance Gap for VOD

In this section we analyze the performance of different video transcoding solutions on the VOD scenario, looking for opportunities to improve performance, and understanding the tradeoffs that they represent. Throughout the section, we will use the coverage corpus described in Section 3 as a golden reference, comparing the trends, correlations, and insights obtained with it to those of vbench.

5.1 CPU Performance

First, we examine how video transcoding exercises the CPU microarchitecture. We found that the microarchitectural profile of video transcoding is very sensitive to the input video, which reinforces the need of a validated benchmark. Furthermore, its performance on general purpose CPUs is better than the typical datacenter workload, e.g. websearch, with respect to retiring rate, frontend stalls and bad speculation stalls [22].

Figure 5 shows how L1 instruction cache misses, branch mispredictions, and last level cache misses correlate with video entropy[4]. Each plot shows two sets of data: black dots for the coverage corpus and colored dots for the various benchmark suites.

Our results show that transcoding of complex videos incurs more icache misses, and more branch mispredictions per kilo instructions. As videos become more complex, the encoder needs to use more advanced compression tools in order to meet the bitrate constraint without degrading quality. This requires exercising more code, which leads to worse front-end performance. At the same time, complex videos incur a lower LLC miss rate. The memory footprint of a video depends only on its resolution, not on video entropy. Transcoding more complex videos will execute more instructions on the same data, leading to higher temporal reuse, hence the lower cache miss rate. In all cases, the trend observed using the vbench video suite matches the trend exposed by the much larger coverage corpus.

Figure 5 also reveals how the choice of the video set can lead to different microarchitecture trends. Since these trends are most visible if low entropy videos are present, the high entropy of the videos in Netflix and Xiph.org biases results. The Xiph.org set shows the opposite trend on icache misses, while the Netflix set shows no correlation between icache MPKI and video entropy. A similar error appears on LLC misses, where the Xiph.org set shows no correlation between them and video entropy.

Figure 6 translates these microarchitecture event counters to performance using the Top-Down methodology [42]. For all video sets, we show boxplots (minimum, maximum, first and third quartiles, and median values) for the % of time spent on front-end (FE), bad speculation (BAD), waiting for memory (BE/Mem), waiting for the back-end (BE/Core), or retiring instructions (RET).

Our results show that for all sets 15% of the time is spent on front-end stalls, 10% on bad speculation, 15% of the time is spent waiting for memory (decreasing for higher entropy videos, lower LLC misses), with the remaining 60% spent retiring instructions or waiting for functional units. Except for maximum and minimum values, the results observed with vbench closely match those obtained with the coverage corpus.

5.2 SIMD Analysis

The high fraction of time retiring instructions or waiting for back-end resources indicates an opportunity for performance improvement by increasing the number of functional units and their data width. Exploiting data-level parallelism with wider SIMD functional units does both at the same time.

Media processing was a major reason for the introduction of SIMD in consumer processors in the 1990s. Indeed, video transcoding is amenable to SIMD acceleration because most of its kernels (DCT, quantization, motion estimation, etc.) are performed on blocks of adjacent pixels. However, not all the video transcoding process can be vectorized. Figure 7 shows the fraction of scalar instructions, and of AVX2 vector instructions as a function of video entropy.

Scalar code represents close to 60% of the instructions on all videos, regardless of their entropy. Focusing only on videos with entropy greater than 1, we observe a slight increase in the scalar fraction as entropy increases and a corresponding reduction of AVX2 instructions. Non-vectorizable

[4]Measurements for Figures 5 to 7 are reported on a Google corporate machine different from the vbench reference: a Xeon E5-1650v3 with 32 GB of DDR4. This was necessary to not distribute user data that was not Creative Commons.

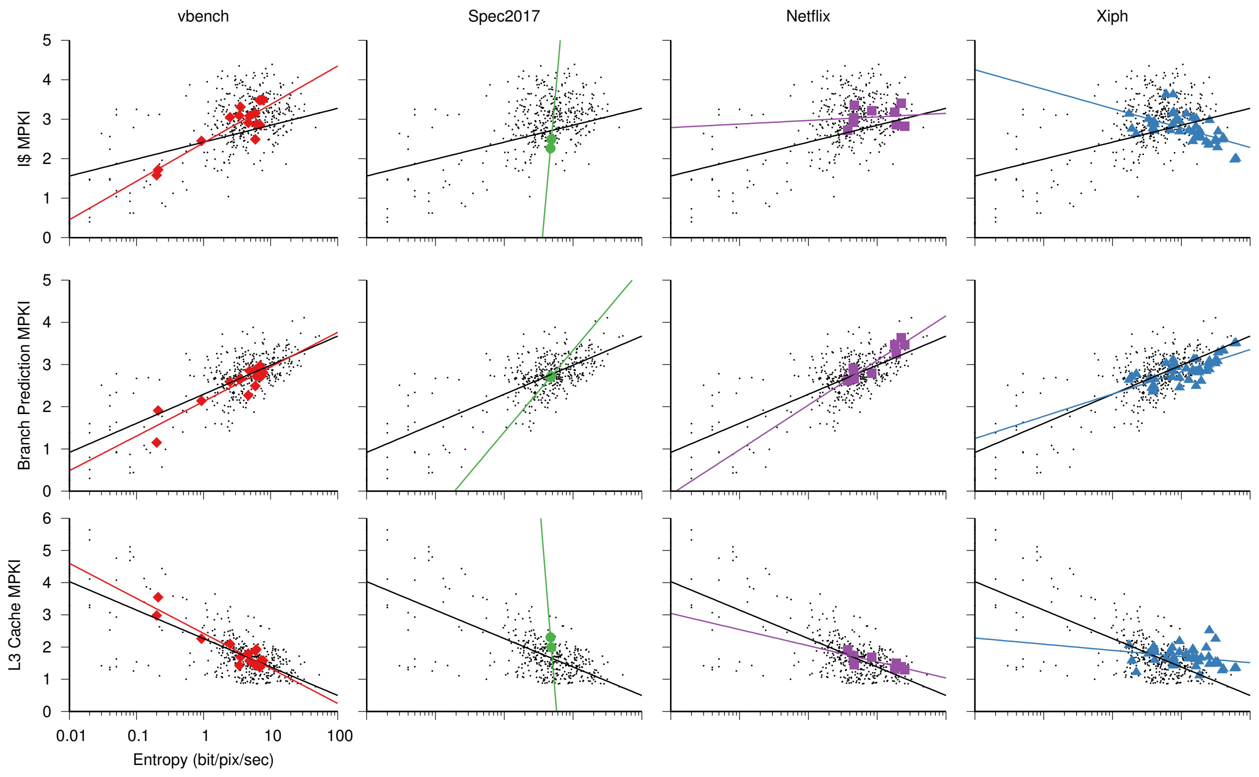

Figure 5. These plots overlay the videos from the different benchmarks (colored dots) on the coverage set (black dots). In both the coverage set and vbench, videos with higher entropy have worse front-end behavior, and reduced last level cache miss rates. The lack of low-entropy videos in the Xiph.org and Netflix datasets leads to different microarchitecture performance trends: lower branch misprediction for high entropy videos, no correlation between video entropy and LLC misses. Logarithmic interpolation ($y = a * \log(x) + b$) is used to obtain trends.

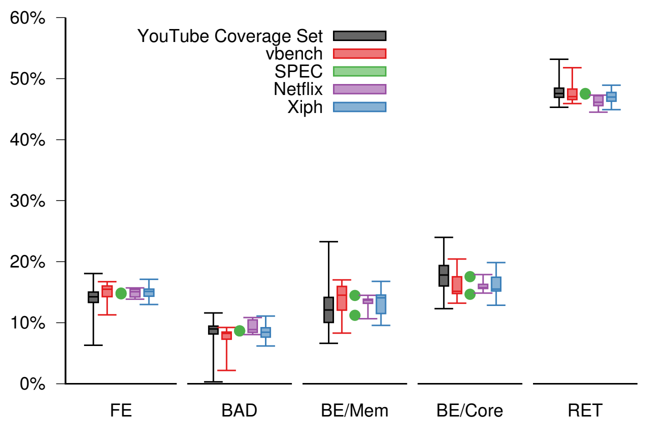

Figure 6. Statistical distribution of the fraction of time spent on front-end, bad speculation, memory, back-end, and retiring instructions. 60% of the time is either retiring instructions or waiting for the back-end functional units.

functions include all the decision making logic, e.g. the frame reference search for motion estimation which averages 9% of the time, or functions that are strictly sequential and control dominated, e.g. entropy encoding which averages 10% of the time.

Figure 8 shows the fraction of the execution time in the different instruction sets as we progressively enable newer SIMD extensions in libx264. Our results show that the scalar fraction has been stable since the introduction of SSE2. New ISA extensions have accelerated the already vectorized fraction of time, but have not increased code coverage. Moreover the performance improvement from SSE2 – an ISA introduced more than fifteen years ago – is only 15%.

Furthermore, our results show that AVX2, which doubles vector width to 256 bits with respect to AVX, only partially replaces it and represents only 15% of the runtime. The remaining vectorized code does not benefit from 256-bit wide SIMD registers due to the width of macroblocks being smaller than the AVX2 vector length. Amdahl's Law limits the potential impact of a 2x wider SIMD extension to less than 10%, even if we assume that time spent in AVX2 instructions scales perfectly with vector size.

We conclude that performance of video transcoding on CPUs is limited by the scalar fraction of the code not suitable for data-level parallelism. To achieve significant speedups,

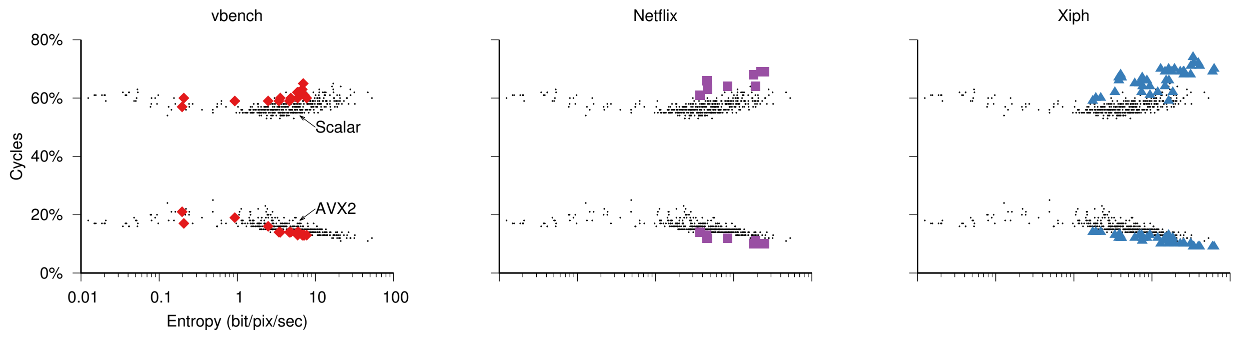

Figure 7. Fraction of time spent in scalar (non-vector) and AVX2 (long vectors) instructions as a function of video entropy. Over half the executed code is not suitable for SIMD acceleration, and less than 20% of the code would benefit from longer vectors. The high entropy videos in Netflix and Xiph show slightly higher ratios of scalar code.

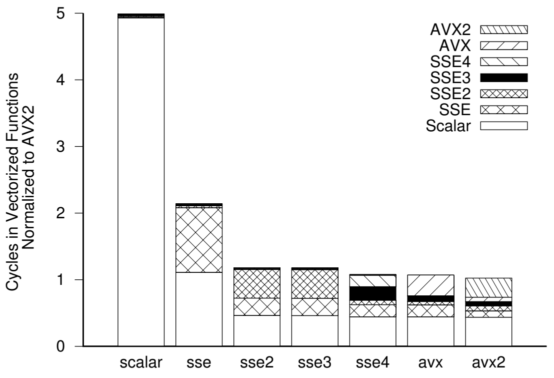

Figure 8. Time breakdown for H.264 transcoding across different SIMD instruction sets. The fraction of time spent in scalar code remains constant and becomes increasingly dominant.

processors could be enhanced with special functional units targeting increased code coverage [17, 30], and 2-dimensional SIMD extensions that exploit data-level parallelism across the entire macroblock [8]. Otherwise, we must resort to full implementations of video transcoding in hardware.

Notice that suites other than vbench, since they contain only high entropy videos, have a slightly larger fraction of time spent in scalar code, and a lower fraction in AVX2 code than what we observe in the coverage corpus. In both Xiph and Netflix sets, only 11% of the time is spent in AVX2 code compared to 14% and 15% for vbench and the coverage set, respectively. This predicts an even lower benefit from vectorization.

5.3 Hardware Accelerators

Contrary to SIMD extensions, end-to-end video transcode solutions are not limited by Amdahl's Law because they cover the entire algorithm, including the control flow and bitstream manipulation. In addition, they can exploit functional level partitioning and parallelism across different stages of the algorithm.

Our results show that hardware encoders provide significant improvements in terms of speed at the cost of increased bitrate. Hardware transcoders need to be selective about which compression tools to implement, in order to limit area and power. For example, enabling sub-pixel precision in motion search or increasing the motion search range will greatly increase the area of an implementation while providing only marginal compression improvements.

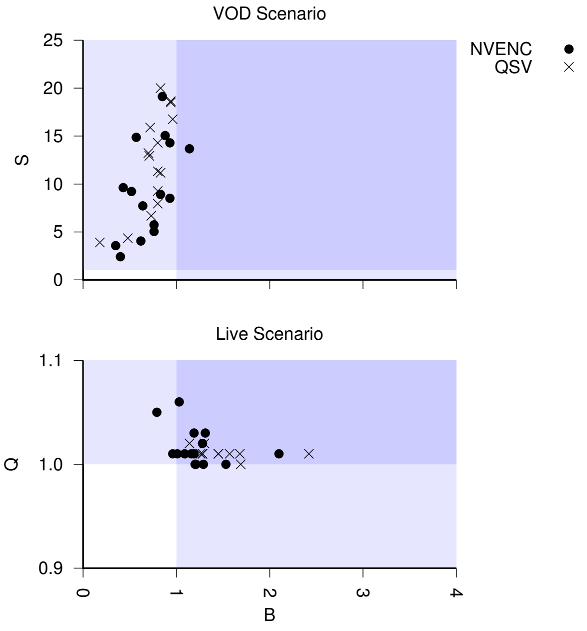

Figure 9. NVIDIA NVENC and Intel QSV results on the `VOD` and `Live` scenarios[5]. The shaded areas indicate gains. While GPU adoption for `VOD` entails tradeoffs (speedups offset by losses in compression) it is an unqualified win for `Live` transcoding.

Resolution (Kpixel)	Name	Entropy (bit/pix/sec)
854x480 (410 Kpixel)	cat	6.8
	holi	7.0
1280x720 (922 Kpixel)	desktop	0.2
	bike	0.9
	cricket	3.4
	game2	4.9
	girl	5.9
	game3	6.1
1920x1080 (2074 Kpixel)	presentation	0.2
	funny	2.5
	house	3.6
	game1	4.6
	landscape	7.2
	hall	7.7
3840x2160 (8M)	chicken	5.9

Table 2. vbench videos' description.

NVENC			QSV		
S	B	VOD Score	S	B	VOD Score
5.74	0.76	4.36	9.27	0.80	7.38
5.04	0.76	3.83	7.95	0.80	6.38
2.41	0.40	0.96	3.90	0.18	0.72
4.05	0.62	2.52	6.68	0.73	4.91
8.91	0.83	7.39	13.22	0.70	9.32
7.72	0.64	4.97	12.94	0.71	9.20
8.51	0.93	7.88	14.29	0.80	11.46
9.22	0.52	4.81	11.32	0.80	9.05
3.58	0.35	1.24	4.35	0.48	2.09
9.63	0.43	4.10	11.17	0.83	9.30
14.29	0.93	13.34	16.75	0.96	16.02
14.87	0.57	8.50	15.89	0.72	11.42
15.05	0.88	13.26	18.50	0.94	17.36
13.68	1.14	15.58	18.64	0.94	17.51
19.12	0.85	16.31	20.00	0.83	16.58

Table 3. VOD score improves with video resolution for NVIDIA NVENC and Intel QSV on the VOD scenario.

NVENC			QSV		
Q	B	Live Score	Q	B	Live Score
1.01	1.09	1.09	1.02	1.14	1.16
1.00	1.21	1.21	1.01	1.28	1.29
1.06	1.03	1.09	1.88	0.16	0.30
1.03	1.31	1.35	1.25	0.48	0.59
1.00	1.29	1.29	1.01	1.14	1.16
1.00	1.20	1.20	1.02	1.30	1.32
1.01	1.16	1.17	1.01	1.45	1.47
1.01	0.96	0.97	1.01	1.28	1.29
1.05	0.79	0.83	1.34	0.31	0.42
1.01	1.01	1.02	1.00	1.69	1.69
1.00	1.53	1.54	1.01	1.68	1.70
1.03	1.19	1.22	1.01	1.57	1.59
1.01	1.19	1.21	1.01	1.26	1.27
1.02	1.28	1.31	1.01	1.45	1.46
1.01	2.10	2.12	1.01	2.42	2.44

Table 4. Intel QSV and NVIDIA NVENC achieve real-time performance on the Live scenario.

LIBVPX-VP9			LIBX265		
Q	B	Pop. Score	Q	B	Pop. Score
1.00	1.47	1.48	1.02	1.17	1.19
1.00	1.06	1.06	1.01	1.12	1.13
1.01	0.67		1.00	0.87	
1.00	1.06	1.06	1.01	1.11	1.12
1.01	0.97		1.02	0.86	
1.00	1.33	1.33	1.01	1.03	1.04
1.01	1.06	1.06	1.02	0.81	
1.01	1.09	1.10	1.01	0.80	
1.00	1.86	1.86	1.00	1.13	1.13
1.00	1.37	1.37	1.00	1.06	1.06
1.01	1.06	1.07	1.01	0.97	
1.00	1.20	1.20	1.00	1.28	1.28
1.01	1.47	1.48	1.02	1.30	1.32
1.01	1.49	1.51	1.01	1.11	1.13
1.01	1.57	1.58	1.01	1.17	1.19

Table 5. libx265 and libvpx-vp9 can reduce bitrate significantly while being iso-Quality on the Popular scenario.

Table 3 reports the speed (S) ratios, bitrate (B) ratios, and VOD scores for two GPUs: the Intel QuickSync Video (QSV) [9] featured in the Intel core i7-6700K CPU, the NVIDIA NVENC [39] found in the GTX 1060 GPU. To obtain these results we used the highest effort settings on both GPUs and varied the target bitrate using a bisection algorithm until results satisfy the quality constraints by a small margin. The results show that the QSV scores are generally higher than the NVENC scores; this is mostly due to the higher speed ratios since bitrate ratios are comparable in the two platforms. Unfortunately, we cannot offer much deeper explanation for the difference as the GPUs do not allow software to inspect intermediate results, effectively creating a black box.

Both GPUs show higher speed improvements for higher resolution videos, since they better amortize the data transfer overheads, and enable higher parallelism across the macroblocks. They also show higher speedups for more complex videos, since they perform all the operations in parallel, while the software needs to run for a longer time to apply the more complex encoding tools: Having a curated video set is important here. Using Xiph or Netflix dataset – both containing only high resolution, high entropy videos – would overestimate the benefits of GPU transcoding.

Both GPUs show significant speed benefits that compensate the losses in video bitrate. Their higher speed would allow a significant downsizing of the transcoding fleet at a video sharing infrastructure. However, they need to compensate with an increase in storage and network costs. The precise balance between compute costs, storage, and network will depend on the specifics of the service, reinforcing the need to report metrics separately in addition to the score.

Given the speedups achieved by these hardware implementations, future hardware video transcoders might implement more advanced encoding tools to trade slower speed for higher video quality at lower bitrate, enabling service providers to tune them to their specific use cases.

6 Live and Popular Analysis

That was the VOD scenario. We now evaluate how GPUs perform on the Live and Popular scenarios. Both are key use cases in services like YouTube and Facebook.

6.1 Live Streaming

While speed is important for VOD, it is critical for the Live scenario. GPUs here shine as low latency transcoding is their intended application, while software encoders have to significantly decrease effort levels in order to meet the real-time speed constraint. In fact, to meet the real time constraint our reference transcodes have an effort level that is inversely proportional to the resolution of the input video. This explains the positive GPU Live results seen in Table 4. When real time speed is required, software encoders degrade the transcode quality much more than hardware.

There are a number of configurations for these GPUs that would have met the Live scenario constraints. For this experiment, we chose to maintain reference quality, which creates an interesting comparison with VOD. Contrary to what we have observed in the VOD scenario, using GPUs in this case generally incurs no tradeoffs. Our results show that hardware encoders achieve the same quality as our reference while **also** reducing the transcode bitrate. The only exceptions being low entropy videos, for which the GPUs struggle to degrade quality and bitrate gracefully. Had the benchmark not included low entropy videos and different scenarios, such insights would not be visible.

Even on the `Live` scenario, where hardware encoders do not incur sacrifices in bitrate and quality, we find that hardware encoders exceed real-time and thus are potentially faster than they need to be. As with `VOD` this again raises the possibility that the excess speedups seen on GPUs might be better spent finding higher quality transcodes.

6.2 Popular Videos

The `Popular` scenario deals with very high effort transcoding for videos that receive the most playbacks. As we saw in the `VOD` scenario, the hardware video transcoders require additional bitrate to match the quality of the reference transcodes in the `VOD` scenario. Given that the reference quality of the `Popular` scenario is higher than `VOD`, it was impossible for either of the GPUs to produce a single valid transcode for this scenario. GPUs are valuable in the `VOD` and `Live` scenarios because of their speed. However, speed is the least valuable metric in the `Popular` scenario. Software encoders are the best option today for optimizing bitrate and quality on highly popular videos, where the effort will be amortized across many playbacks.

Table 5 shows the benchmark scores for the recent libx265 and libvpx-vp9 encoders when transcoding `Popular` videos. For both, we selected a fixed effort level such that all videos can be encoded within the speed constraint[6]. Our results confirm what was seen earlier in Figure 2: both libx265 and libvpx-vp9 encoders are superior to the reference libx264 when transcoding speed is not considered. Both encoders are capable of significantly reducing bitrate for most videos at no quality loss, especially for HD and higher resolutions. Notice that the reference transcode operations in this scenario use the highest quality setting in libx264. This reflects how newer codecs (H.265, VP9) and relative encoders keep improving video compression, a trend that is expected to continue with the release of the AV1 codec by the end of the year [15].

7 Conclusions

We have described vbench, a video transcoding benchmark that has an algorithmically curated set of representative videos, metrics, and reporting guidelines that reflect complex real world tradeoffs between quality, compression, and speed. vbench quantitatively selected dataset improves on existing video collections assembled based on qualitative visual properties. With the same methodology we can update vbench videos over time to reflect changes in upload trends. The reference transcoding operations reflect those of large video sharing infrastructures and are thus a useful baseline. The metrics guide improvement over this baseline in meaningful directions.

Our studies of video transcoding reveal a microarchitectural sensitivity to inherent video entropy. There are limits on the benefits of vectorization for this workload while the viability of hardware support for transcoding depends strongly on the context; current GPUs are well suited for live streaming yet unable to meet the quality and compression demands of popular videos. We expect this benchmark and the insights it will produce will promote much needed innovation in video transcoding, a key warehouse scale workload that is at the core of all video sharing infrastructures.

[6]cpu-used 0 for libvpx-vp9 and -preset veryslow for libx265. A empty score indicates that either the bitrate or quality constraints are not met (highlighted in red)

Acknowledgments

We thank the anonymous reviewers as well as Jim Bankoski, David Brooks, Milad Hashemi, Anil Kokaram, Heiner Litz, Scott Silver, Clint Smullen, Michael Taylor, and Drew Walton for their comments and feedback on early versions of this work. Lottarini's academic work was partially supported by Google, NSF (CCF-1065338) and C-FAR, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

[1] M. Alvarez, E. Salami, A. Ramirez, and M. Valero. 2007. HD-VideoBench: A Benchmark for Evaluating High Definition Digital Video Applications. In *IEEE 10th International Symposium on Workload Characterization (IISWC).*
[2] Christian Bienia. 2011. *Benchmarking Modern Multiprocessors.* Ph.D. Dissertation. Princeton University.
[3] Blender Foundation. 2002. *Blender - a 3D modelling and rendering package.* http://www.blender.org
[4] Rajkumar Buyya, Mukaddim Pathan, and Athena Vakali. 2008. *Content Delivery Networks* (1st ed.). Springer Publishing Company.
[5] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. 2009. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on Networking (TON)* 17, 5 (2009).
[6] Chao Chen, Mohammad Izadi, and Anil Kokaram. 2016. A Perceptual Quality Metric for Videos Distorted by Spatially Correlated Noise. In *Proceedings of the ACM Multimedia Conference.*
[7] Jan De Cock, Aditya Mavlankar, Anush Moorthy, and Anne M. Aaron. 2016. A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications. In *Applications of Digital Image Processing XXXIX.*
[8] Jesus Corbal, Roger Espasa, and Mateo Valero. 1999. MOM: A Matrix SIMD Instruction Set Architecture for Multimedia Applications. In *Proceedings of the ACM/IEEE Conference on Supercomputing (SC).*
[9] Intel Corp. 2017. Intel Quick Sync Video. https://www.intel.com/content/www/us/en/architecture-and-technology/quick-sync-video/quick-sync-video-general.html. (2017).
[10] CreativeCommons.org. 2007. Creative Commons Attribution 3.0 License. https://creativecommons.org/licenses/by/3.0/legalcode. (2007).
[11] Youtube Engineering and Developers Blog. 2016. A look into YouTube's video file anatomy. https://youtube-eng.googleblog.com/2016/04/a-look-into-youtubes-video-file-anatomy.html. (2016).
[12] facebook 2016. Facebook Inc. Third Quarter 2016 Earnings Call (transcript). https://seekingalpha.com/article/4018524-facebook-fb-q3-2016-results-earnings-call-transcript.

(2016).

[13] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. 2012. Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. In *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[14] Sadjad Fouladi, Riad S. Wahby, Brennan Shacklett, Karthikeyan Vasuki Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. 2017. Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[15] Mozilla & The Xiph.Org Foundation. 2017. Progress in the Alliance for Open Media. https://people.xiph.org/~tterribe/pubs/lca2017/aom.pdf. (2017).

[16] R. Gonzalez and M. Horowitz. 1996. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits* 31, 9 (1996).

[17] Rehan Hameed, Wajahat Qadeer, Megan Wachs, Omid Azizi, Alex Solomatnikov, Benjamin C. Lee, Stephen Richardson, Christos Kozyrakis, and Mark Horowitz. 2010. Understanding Sources of Inefficiency in General-purpose Chips. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)*.

[18] Johann Hauswald, Michael A. Laurenzano, Yunqi Zhang, Cheng Li, Austin Rovinski, Arjun Khurana, Ronald G. Dreslinski, Trevor Mudge, Vinicius Petrucci, Lingjia Tang, and Jason Mars. 2015. Sirius: An Open End-to-End Voice and Vision Personal Assistant and Its Implications for Future Warehouse Scale Computers. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[19] John L Henning. 2006. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News* 34, 4 (2006).

[20] Qi Huang, Petchean Ang, Peter Knowles, Tomasz Nykiel, Iaroslav Tverdokhlib, Amit Yajurvedi, Paul Dapolito, IV, Xifan Yan, Maxim Bykov, Chuen Liang, Mohit Talwar, Abhishek Mathur, Sachin Kulkarni, Matthew Burke, and Wyatt Lloyd. 2017. SVE: Distributed Video Processing at Facebook Scale. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*.

[21] Tubular Insights. 2015. 500 hours of video uploaded to YouTube every minute. http://tubularinsights.com/hours-minute-uploaded-youtube/. (2015).

[22] Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks. 2015. Profiling a Warehouse-scale Computer. In *Proceedings of the 42nd International Symposium on Computer Architecture (ISCA)*.

[23] Anil Kokaram, Damien Kelly, Hugh Denman, and Andrew Crawford. 2012. Measuring Noise Correlation For Improved Video Denoising. In *IEEE International Conference on Image Processing*.

[24] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward A Practical Perceptual Video Quality Metric. http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html. (2016).

[25] Ikuo Magaki, Moein Khazraee, Luis Vega Gutierrez, and Michael Bedford Taylor. 2016. ASIC Clouds: Specializing the Datacenter. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*.

[26] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand. 2003. Context-based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003).

[27] Felix Mercer Moss, Ke Wang, Fan Zhang, Roland Baddeley, and David R. Bull. 2016. On the Optimal Presentation Duration for Subjective Video Quality Assessment. *IEEE Transactions on Circuits and Sysystems for Video Technology* 26, 11 (2016).

[28] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jinghing Han, John Koleszar, Paul Wilkins, Yaowu Xu, and Ronald Bultje. 2013. The latest open-source video codec VP9 - An overview and preliminary results. In *Picture Coding Symposium (PCS)*.

[29] The FFmpeg project. 2017. Encode/H.264. https://trac.ffmpeg.org/wiki/Encode/H.264. (2017).

[30] Wajahat Qadeer, Rehan Hameed, Ofer Shacham, Preethi Venkatesan, Christos Kozyrakis, and Mark A. Horowitz. 2013. Convolution Engine: Balancing Efficiency & Flexibility in Specialized Computing. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*.

[31] Majid Rabbani and Paul W. Jones. 1991. *Digital Image Compression Techniques* (1st ed.). Society of Photo-Optical Instrumentation Engineers (SPIE).

[32] Albert Rafetseder, Florian Metzger, David Stezenbach, and Kurt Tutschku. 2011. Exploring YouTube's Content Distribution Network Through Distributed Application-layer Measurements: A First View. In *Proceedings of the International Workshop on Modeling, Analysis, and Control of Complex Networks (CNET)*.

[33] Sandvine Intelligent Broadband Networks. 2016. 2016 Global Internet Phenomena: Latin America & North America. https://www.sandvine.com/trends/global-internet-phenomena/. (2016).

[34] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012).

[35] Linpeng Tang, Qi Huang, Amit Puntambekar, Ymir Vigfusson, Wyatt Lloyd, and Kai Li. 2017. Popularity Prediction of Facebook Videos for Higher Quality Streaming. In *USENIX Annual Technical Conference*.

[36] Lei Wang, Jianfeng Zhan, Chunjie Luo, Yuqing Zhu, Qiang Yang, Yongqiang He, Wanling Gao, Zhen Jia, Yingjie Shi, Shujie Zhang, Chen Zheng, Gang Lu, Kent Zhan, Xiaona Li, and Bizhu Qiu. 2014. BigDataBench: A big data benchmark suite from internet services. In *20th IEEE International Symposium on High Performance Computer Architecture (HPCA)*.

[37] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004).

[38] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003).

[39] Martin Alexander Wilhelmsen, Håkon Kvale Stensland, Vamsidhar Reddy Gaddam, Asgeir Mortensen, Ragnar Langseth, Carsten Griwodz, and Pål Halvorsen. 2014. Using a commodity hardware video encoder for interactive video streaming. In *IEEE International Symposium on Multimedia (ISM)*.

[40] Jun Xin, Chia-Wen Lin, and Ming-Ting Sun. 2005. Digital Video Transcoding. *Proc. IEEE* 93, 1 (2005).

[41] Xiph.org. 2016. Derf's Test Media Collection. (2016). https://media.xiph.org/video/derf/

[42] Ahmad Yasin. 2014. A Top-Down method for performance analysis and counters architecture. *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*.

[43] Haibo Zhang, Prasanna Venkatesh Rengasamy, Shulin Zhao, Nachiappan Chidambaram Nachiappan, Anand Sivasubramaniam, Mahmut T. Kandemir, Ravi Iyer, and Chita R. Das. 2017. Race-to-sleep + Content Caching + Display Caching: A Recipe for Energy-efficient Video Streaming on Handhelds. In *Proceedings of the 50th International Symposium on Microarchitecture (MICRO)*.

[44] Yuhao Zhu, Daniel Richins, Matthew Halpern, and Vijay Janapa Reddi. 2015. Microarchitectural Implications of Event-driven Server-side Web Applications. In *Proceedings of the 48th International Symposium on Microarchitecture (MICRO)*.