EFFICIENT MULTI-BITRATE HEVC ENCODING FOR ADAPTIVE STREAMING

Deepthi Nandakumar, Sagar Kotecha, Kavitha Sampath,
Pradeep Ramachandran, Tom Vaughan

MulticoreWare Inc.

ABSTRACT

Adaptive bitrate streaming is a critical feature in internet video that significantly improves the viewer experience by customizing video stream quality to the viewer device's capability and connectivity. Encoding the source content at multiple quality tiers or bitrates is extremely demanding for post-production houses, studios, and content delivery networks. This paper describes an intelligent multi-bitrate encoder, based on the High Efficiency Video Coding (HEVC)/H.265 standard that encodes a single title to multiple bitrates at significant performance gains and no compression efficiency loss, as compared to standalone single bitrate encoder instances.

We first describe the threading infrastructure of x265, and demonstrate its ability to dynamically adapt to varying degrees of parallelism in hardware. We then describe the key architectural design of a multi-bitrate encoder, including thread synchronization challenges across encoder instances. We also discuss the analysis data shared across different quality tiers, that is carefully chosen to eliminate loss of compression efficiency compared to a single bitrate encoder instance. Finally, we show the high performance gains achieved by the multi-encoder, and demonstrate the feasibility of simultaneous encoding to multiple bitrates with negligible loss of compression efficiency.

INTRODUCTION

Over-The-Top (OTT) content streaming is poised to grow exponentially in the coming years, driven by the consumer's need for a rich and high quality viewing experience. Recent projections indicate that Internet delivery of video will consume 80 to 90% of all Internet bandwidth by the year 2019 [1] [2].

Consequently, video streaming over the Internet has evolved tremendously over the past several years including advances in compression, and transmission technology. The Advanced Video Coding (AVC) standard has been the de-facto compression standard for some years now. The recently-proposed High Efficiency Video Coding (HEVC) standard was developed by the Joint Collaborative Group for Video Coding (JCT-VC) with the goal of achieving the same quality as that achieved by the AVC standard at 50% the bit-rate. Studies have verified this improvement in encoding efficiency can be realized at typical consumer video distribution quality levels [3]. In this paper, we explore using the HEVC standard to enable OTT content streaming while ensuring a significantly improved viewer experience. Since encoding for the HEVC standard is expected to be 5-10X computationally more intensive than AVC, we discuss critical aspects related to improving encoder performance.

HEVC codec solutions play a key role in enabling OTT content streaming by significantly reducing the bitrate required for defined visual quality levels. In addition to efficient compression of video, multimedia streaming over the internet has unique challenges that need to be addressed. The open internet is by definition an "unmanaged" network where end-user bandwidth for the OTT consumer cannot be guaranteed. Due to network congestion at times of peak demand, frames could take longer to reach, thus causing the playback to stall due to an empty buffer. A widely adopted technique to mitigate this is adaptive streaming, where the bitrate of the delivered video is dynamically adapted to changing network conditions. By encoding to multiple bitrates, and dynamically switching between the bitrate tiers, streaming media servers adapt to changing network conditions, significantly improving the viewer experience.

This discussion is guided by our experience developing x265 [4], an open-source software HEVC encoder that was developed using the x264 AVC encoder project as a reference. x265 is the world's most widely adopted HEVC encoder and is integrated into popular media processing applications and frameworks such as VLC, Handbrake, FFMPEG, and gstreamer. In a recent comparison of HEVC encoders conducted by the video experts at Moscow State University, x265 achieved the highest efficiency (the lowest bit rate at any target quality level) of any HEVC encoder tested [5]. We also describe the UHDkit multi-bitrate encoder that enables simultaneous and efficient encoding of multiple HEVC bit-streams at different bitrate tiers from a single video source. The multi-bitrate encoder is architected around x265, and shares analysis information from one bitrate instance to the others to enable a significantly faster encode with every little impact to encoding efficiency. Our results show that encoding to 4 bitrates with our multi-bitrate HEVC encoder results in a 2.5X speed-up for1080p streams, and a 2.1X speed-up for 4K streams.

ENCODER PARALLELISM AND PERFORMANCE

HEVC encoding is, on average, 5X more complex than encoding for the AVC standard, when targeting ultra-high definition (UHD) resolutions of 3840x2160 pixels for each frame. This complexity is further increased with 10-bit pixels, as opposed to traditional 8-bit pixels.

In x265, the paradigm of parallelism is baked fundamentally into the encoder to achieve high-performance. In this paper, we discuss the fundamental threading infrastructure in x265. While some of the features of parallelism have no impact on encoding efficiency, we also implement several features that trade-off encoding efficiency for heightened performance. Interested readers are referred to a recent publication on x265 that discusses the trade-offs between performance and efficiency in more detail [6].

Thread Categories in x265

x265 creates two main categories of threads that operate during the encoding process.

The first category are *frame encoder* threads that operate on multiple frames in parallel, enabling inter-frame parallelism during HEVC encoding. However, since there may be cross-frame dependencies for motion-compensated prediction, these frame encoder threads are orchestrated such that these constraints are not broken. While frame parallelism enables higher performance, it has an impact on the accuracy of rate prediction. This is because the accuracy of the rate-control information passed from a previous frame to the current frame is limited since the previous frame may be encoding in parallel with the current frame. Frame-level parallelism should therefore be carefully orchestrated to balance the improved performance with the impact to encoding efficiency.

Another category of threads are *worker* threads that are used to implement intra-frame parallelism when encoding a given frame. The HEVC standard introduces Wavefront Parallel Processing (WPP) as a key technique for parallel encoding and decoding. Through WPP, multiple rows of CTUs (coding blocks introduced by HEVC) may be encoded or decoded in parallel without significant impact on encode efficiency. The block-level restrictions on motion prediction limits the parallelism that can be exposed via WPP.

The combination of *frame encoder* and *worker* threads enable inter and intra-frame parallelism during HEVC encoding, resulting in close to optimum utilization on multi-core and multi-socket architectures.

Software Thread Pools and Mapping to Hardware Threads

x265 utilizes the concept of "thread pools" to better manage the worker threads and their mapping to hardware threads. A thread pool is a collection of worker threads whose affinity is set to the processors on one NUMA node in a multi-socket server; if the system has only one socket, the threads in a pool are associated with all the CPUs in the system by default. x265 uses the NUMA API calls exposed by various operating systems to set the affinity of the threads to the appropriate processors. By setting the affinity of these software threads to a limited number of CPUs, the mapping of the software threads to their hardware counter-parts ensures that there is limited movement of data between the caches, resulting in higher performance. x265 also has the ability to create multiple pools of threads that may each be restricted to one socket. Separating the worker threads in this fashion may result in limiting the amount of cross-socket traffic in a multi-socket server, enabling higher performance.

Impact of threading on Hardware Utilization

The overall threading design enables x265 to seamlessly adapt and utilize the underlying HW threads on any HW that it may execute on for HEVC encoding. **Error! Reference source not found.** shows how a 4k x265 encode for a typical ripping setting adapts itself to the varying number of hardware threads on three different systems, an i5-4500U mobile processor, i7-6700K (Skylake) desktop processor and a dual-socket E5-2666 v3 (Haswell) server. On the mobile and desktop systems, x265 is able to max out all available hardware threads due to its intelligent threading infrastructure. x265 however, saturates at around 24 threads on the server system due to fundamental limitations in the amount of parallelism that can be achieved with a single stream HEVC encode. Our work on the UHDkit multi-bitrate encoder discusses techniques to encode multiple parallel streams to overcome these bottle-necks and to achieve higher utilization and higher performance [7].

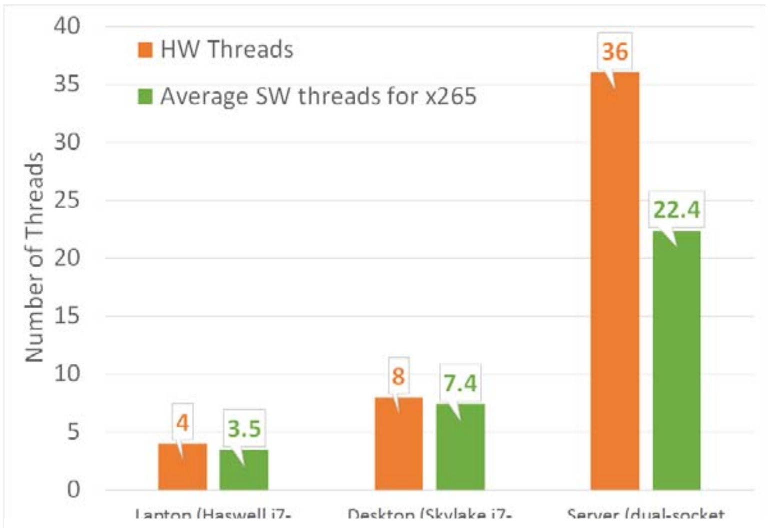

Figure 1: x265 dynamically adapts s/w threading levels to different machine configurations

ADAPTIVE STREAMING TECHNIQUES

The prevalent form of web-based media delivery using HTTP progressive download suffered from a serious drawback in that the encoding technology was limited to Constant Bitrate (CBR) or Variable Bitrate (VBR). CBR required that the number of bits per second of video always remain a constant, for simplified client playback. VBR was a substantial improvement over CBR, allowing for different sections of the video to be encoded at different bitrates based on frame complexity. However, progressive download had many drawbacks in terms of viewer experience, which included long start and seek times and, more importantly, the stop-start buffering phenomenon, as the playback client waited for its buffer to fill before playing it out.

Adaptive Bitrate (ABR) is designed to deliver the best streaming video quality, regardless of content, client bandwidth or client device. ABR video streaming allows streamed video quality to vary over the lifetime of a stream to match changing conditions on the network, thus trading video quality for continuous playback. For ABR systems using HTTP with pre-encoded content, the technology can be divided into 2 basic systems a) split the media file into several consecutive segments, and encode each segment at multiple bitrates and b) a monitoring system that determines network congestion and/or buffer capacity and requests each chunk at the appropriate bitrate from the streaming server. In a congested network, the streaming server sends smaller chunks from the lower bitrate tiers, to prevent the client playback buffer from emptying, thus avoiding a playback interruption.

In most systems, adaptive streaming is controlled by the client playback systems. The client initially receives a list of available bitrate tiers. The client will measure bandwidth and dropped frames continually during playback, and react to screen-size changes. When the server receives a request to change the bitrate, it will wait for the closest keyframe and switch to the requested bitrate tier.

Encoding Considerations for Adaptive Streaming

The most important design choice in encoding for adaptive streaming is the number of encoding tiers and the exact bitrate chosen for each encoding tier. The choice of bitrates will be governed by the nature of content, the resolution and the targeted playback device list. The larger the targeted range of devices and resolutions, the wider the number and range of bitrates needs to be. It is also imperative that the bitrate tiers are as close together as possible, so that switching across tiers does not cause noticeable fluctuations in quality. The decoder buffer constraints can also play an important role in determining response to bandwidth fluctuations.

Another important encoding consideration is the need for keyframes as synchronization points, where the streaming media server can switch to a different bitrate tier. For successful switching between bitrate tiers, it is necessary that open-GOP encoding is disabled. This prevents dependencies between frames following the keyframe and frames preceding the keyframe. The interval between keyframes plays a very important role in controlling streaming quality. Too many keyframes, and the overhead of increased bits is too large. Too few keyframes, and the streaming media server cannot respond fast enough to changing bandwidth conditions. A keyframe interval of 1-5 seconds is typically used in most streaming applications.

MULTI-BITRATE ENCODER ARCHITECTURE

The UHDkit multi-encoder, based on the open-source x265 HEVC encoder, enables efficient accelerated encoding of a single title to multiple bitrates for adaptive streaming. Encoding to multiple bitrates within a single application dramatically saves CPU cycles, thus enabling delivery to streaming formats like MPEG-DASH and HLS in less time or with less hardware.

The multi-encoder maintains encode efficiency by sharing cached analysis data across all bitrate instances. The highest bitrate instance, and therefore the one with the best visual quality, is referred to as the *master encoder*, and the other instances are referred to as *slave encoders*. The multiple encoders are synchronized such that the slave encoders start encoding a frame only after the master encoder has completed that frame encode. This prevents data races and ensures all the shared structures are available to the slave instances.

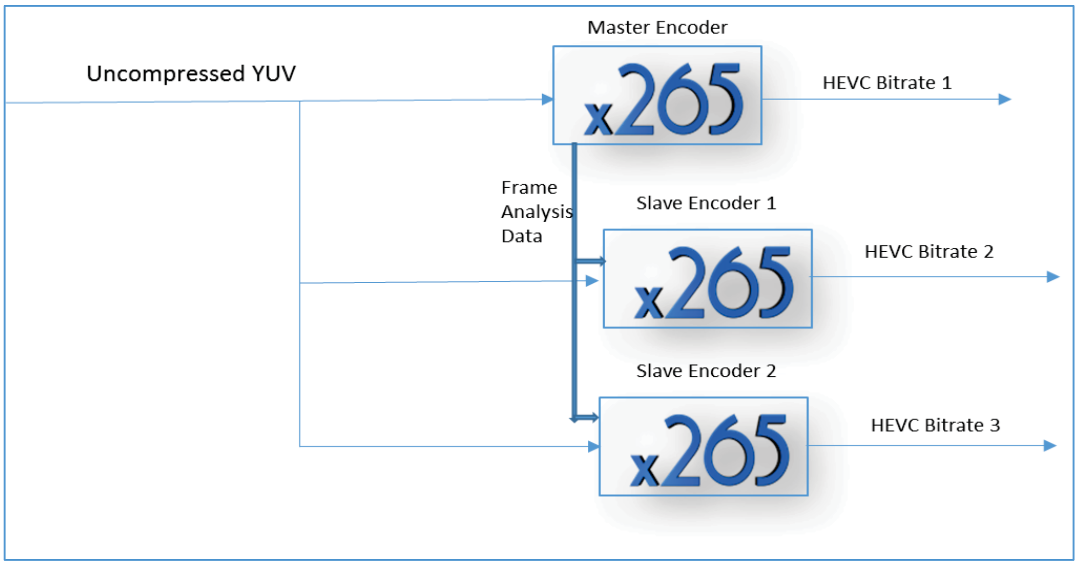

Figure 2: UHDkit multi-bitrate HEVC encoder top-level architecture

The x265 HEVC encoder implements an important module called lookahead, which performs pre-analysis of input frames to make several key decisions such as slicetype decision and adaptive quantization. As the UHDkit multi-encoder shares input frames across all encoder instances, lookahead pre-processing is also shared across all encoder instances and is managed on a separate thread. This is crucial for adaptive bitrate streaming, as each of the encoder instances must share keyframes at synchronization points. Adaptive quantization is another encode feature that is implemented in the lookahead and shared across all encoder instances. The encoder assigns QP offsets to blocks depending on source picture complexity. The basic ratecontrol feature is a key differentiating module across the multiple encoder instances, since their target bitrates and hence, base frame QPs, will be significantly different. The QP offsets are then applied over the base frame QPs.

The shared data structures are of 2 categories, intra-frame data and inter-frame data. Intra-frame data includes information about the CU partition size and the luma and chroma intra mode that was chosen by the master encoder. Inter-frame data includes information about the CU partition size, reference frame and motion vector that was chosen by the master encoder. For a weighted inter-frames, this also includes weighting parameters. The slave encoders further refine the mode and motion vectors to choose the best encode settings for the determined block QP. For the slave encode instance, the ability to refine key encode parameters prevents any drop in efficiency, but provides significant savings in CPU cycles.

RESULTS

Most encoder performance gains are a major encode efficiency trade off, in that large performance gains from algorithmic improvements typically cause a significant drop in encode efficiency and vice versa. With our carefully designed architecture, the UHDkit multi-bitrate encoder achieves substantial performance gains with little to no drop in efficiency.

Figure 3 shows the time taken to encode a single input stream to 4 output streams at different bitrates using the UHDkit multi-bitrate encoder application. The reported time is an average of the time taken to encode three 1080p and three 4k streams and the average time taken plotted. This is compared to independent x265 encodes, which generates a single output stream at a time. On average, the simultaneous multi-bitrate encoder achieves 2.5X speedup for 1080p streams, and 2.1X speedup for 4k streams.

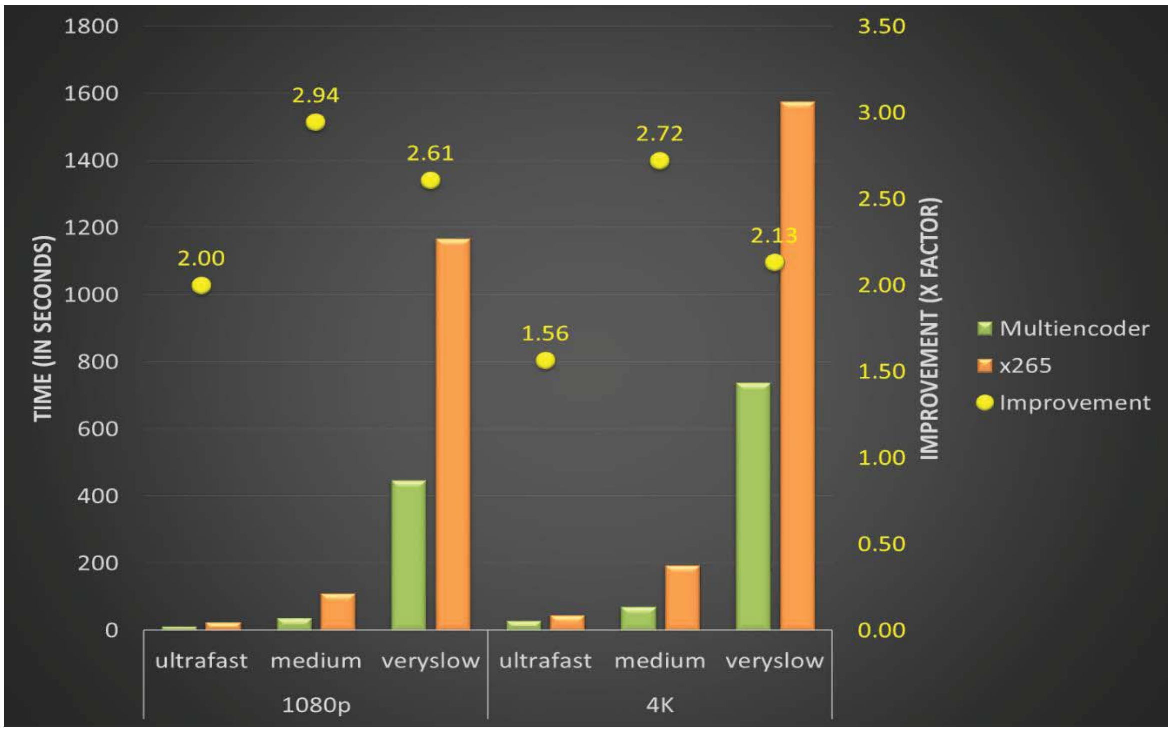

Figure 3: Encode time reduction from a 4 instance multi-bitrate encoder compared to a single instance x265 (encoding to 4 bitrates sequentially). The measurements were done on an E5-2699 v3 (18cores/socket)

Figure 4 compares the encode efficiency of the UHDkit multi-bitrate encoder, represented by BD-SSIM, using independent x265 encodes as the reference. As expected, the UHDkit multi-bitrate encoder shows a slight drop in efficiency particularly for the slave encoders, since certain modes and partition sizes are eliminated from evaluation based on the master encoder’s CU decisions. However, the average BD-SSIM drop is contained at 0.04 dB for 1080p videos and at 0.06 dB for 4k videos. This translates to almost no difference in visual quality across the UHDkit multi-bitrate encoder and independent x265 encodes.

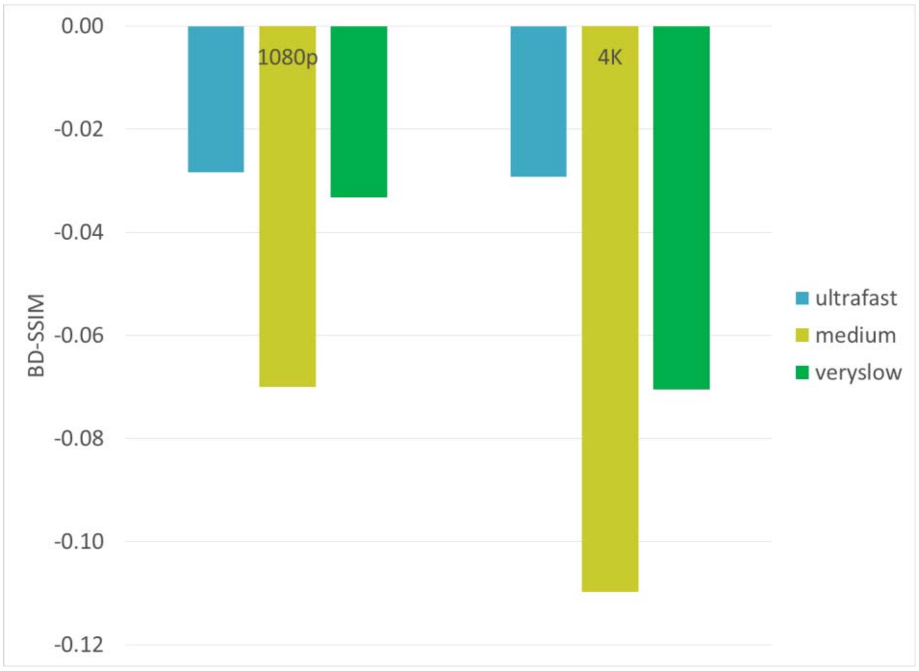

Figure 4: Averaged BD-SSIM of the UHDkit multi-bitrate encoder compared against independent x265 encodes.

CONCLUSION AND FUTURE WORK

In this paper, we described the parallelization challenges within the framework of the widely used HEVC open-source encoder x265, and how the threading design of x265 dynamically adapts to the target hardware architecture. We also described our work on the widely used HEVC open-source encoder x265, and how UHDkit, a multi-bitrate encoder based on x265 enables efficient adaptive streaming of OTT content. A possible future extension of this work includes enabling encoding to multiple resolutions, in addition to multiple bitrates.

As adoption of HEVC ramps up, we continue to explore new algorithms to achieve more efficient video encoding and higher performance. These include more efficient sharing of inter-frame motion structures and the ability to refine shared data across encoder instances.

REFERENCES

[1] Cisco, “Cisco Visual Networking Index: Forecast and Methodology, 2014-2019 White Paper,” Cisco, 2014. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html.

[2] Sandvine, “Global Internet Phenomena Report,” 2014. [Online]. Available: https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/2h-2014-global-internet-phenomena-report.pdf.

[3] BBC, “H.265/HEVC vs H.265/AVC 50% Bit-rate Savings Verified,” BBC, 2016. [Online]. Available: http://www.bbc.co.uk/rd/blog/2016-01-h-dot-265-slash-hevc-vs-h-dot-264-slash-avc-50-percent-bit-rate-savings-verified.

[4] Multicoreware, “x265 - An Open-Source HEVC Encoder,” Multicoreware Inc, 2013. [Online]. Available: http://x265.org.

[5] MSU, “HEVC Video Codecs Comparison,” Moscow State University, 2015. [Online]. Available: http://compression.ru/video/codec_comparison/hevc_2015/.

[6] T. Vaughan, D. Nandakumar, P. Ramachandran and J. Ramachandran, “Efficiency vs Performance Trade-offs in the Design of an HEVC Encoder,” in *National Association of Broadcasters Conference*, Las Vegas, 2016.

[7] Multicoreware Inc, “MulticoreWare Announces UHDkit Advanced HEVC Video Encoding Library, powered by x265,” 2016. [Online]. Available: http://www.prnewswire.com/news-releases/multicoreware-announces-uhdkit-advanced-hevc-video-encoding-library-powered-by-x265-300252630.html.