Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2012, Article ID 935724, 11 pages
doi:10.1155/2012/935724

Research Article

Towards an Automatic Parameter-Tuning Framework for Cost Optimization on Video Encoding Cloud

Xiaowei Li, Yi Cui, and Yuan Xue

Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37235, USA

Correspondence should be addressed to Xiaowei Li, xiaowei.li@vanderbilt.edu

Received 15 May 2011; Revised 27 September 2011; Accepted 11 October 2011

Academic Editor: Yifeng He

The emergence of cloud encoding services facilitates many content owners, such as the online video vendors, to transcode their digital videos without infrastructure setup. Such service provider charges the customers only based on their resource consumption. For both the service provider and customers, lowering the resource consumption while maintaining the quality is valuable and desirable. Thus, to choose a cost-effective encoding parameter, configuration is essential and challenging due to the tradeoff between bitrate, encoding speed, and resulting quality. In this paper, we explore the feasibility of an automatic parameter-tuning framework, based on which the above objective can be achieved. We introduce a simple service model, which combines the bitrate and encoding speed into a single value: encoding cost. Then, we conduct an empirical study to examine the relationship between the encoding cost and various parameter settings. Our experiment is based on the one-pass Constant Rate Factor method in x264, which can achieve relatively stable perceptive quality, and we vary each parameter we choose to observe how the encoding cost changes. The experiment results show that the tested parameters can be independently tuned to minimize the encoding cost, which makes the automatic parameter-tuning framework feasible and promising for optimizing the cost on video encoding cloud.

1. Introduction

The consensus of both academia and industry has been reached that cloud computing will be the next revolutionary progress in information technology, in which dynamically scalable and virtualized computation resources are provided as service over Internet. It has drawn extensive attention from IT giants, such as Amazon (EC2, Elastic Computing Cloud; S3, Simple Storage Service) [1], Google (GAE, Google Application Engine) [2], and Microsoft (Azure) [3]. Customers are charged based on the amount of computation resources they actually have consumed, including CPU time, memory, storage, and transferred bytes. For example, Amazon EC2 charges $0.125 per hour for standard on-demand CPU instances running Windows and $1.20 for extra-large high-CPU instances as for now [1].

Online video vendors like YouTube [4], Youku [5], and Tudou [6] have a huge number of videos of various formats uploaded by users every day, which need to be transcoded (format converted) into their desired intermediate formats, such as .flv, for storage and replay. Such transcoding work is computing-intensive and time-consuming. It also requires storage space and bandwidth for transfer, which pose extremely high demands on hardware resources. Many content owners (libraries, traditional media production units, and governments) have the need to digitize and distribute their contents, but no technical resources/expertise to do so. With the introduction of cloud computing, a new type of Internet service termed encoding cloud emerges, which builds on the cloud infrastructure (e.g., Amazon EC2) to provide value-added video encoding service to meet the aforementioned demands. A successful example is Flix Cloud [7], which combines On2's Flix Engine encoding software with Zencoder's web-based application to provide scalable online encoding service based on Amazon EC2. Another example is http://www.encoding.com/, which offers a similar web-based "pay-as-you-go" service.

For both encoding service providers and customers, a good trade-off between quality and speed is desirable for cost saving purposes. The difficulty originates from the inherent complexity of modern video encoding technologies, most of which inherit from the block-based hybrid coding

design. For example, x264, a widely adopted video encoding engine, exposes more than 100 parameters [8] for its users to balance a long list of factors including encoding speed, bitrate, objective/subjective quality, decoding buffer, and so forth. Evidently, it would be prohibitively expensive for an operator to figure out the most economically efficient "encoding recipes" for thousands of uploaded videos every day. Nevertheless, any endeavor towards this goal is imperative and essential, since the cost saving on the cloud resource consumption will simply add to the profits, whose amount would be quite considerable given the sheer volume of video being encoded. To this end, we believe that an automated mechanism for encoding parameter selection and tuning would be very valuable and timely.

In this paper, we commence our exploration towards this goal. To the best of our knowledge, there is no prior work on this front. Our study chooses H.264 as the targeted video coding standard, and specifically x264 as the targeted encoding software. The rationale of our choice is delayed to Section 2. Our empirical study makes the following important findings.

(i) Modern video encoding technology, represented by x264, is able to precisely and consistently achieve any specified numerical target on subjective/objective quality metric, such as PSNR (peak signal-to-noise ratio). This is a must-have property for any cost-optimization framework desiring to meet stringent encoding quality constraint.

(ii) While observing the quality constraint, we find that many key video encoding parameters (e.g., B-frame) are shown to be able to be independently tuned to adjust the encoding cost.

(iii) There are also strong indications that the impact of each parameter to the final encoding cost can be specified in a weighted sum fashion, in which the weight stays invariant to external factors such as cost coefficients. Combined with previous two observations, it suggests the feasibility of an automatic tuning framework over decomposable parameters, which can quickly find the optimal operation point and predict the encoding cost, if certain knowledge on encoding complexity of the video itself is available a priori.

The rest of this paper is organized as follows. Section 2 reasons our selection on x264 and reveals the internal relationship among bitrate, quality, and encoding speed. Section 3 defines a simple service model under the cloud computing environment. Section 4 presents the experimental results on key parameters in x264, such as B-frame, Ref, Subme, and Trellis. By examining critical observations made here, Section 5 discusses promises and hurdles towards an automatic cost tuning framework. Some related work are described in Section 6. Section 7 concludes this paper.

2. Related Work

Some research work [9–11] propose various algorithms, including mode ranking and learning theoretic, to optimize the R-D (rate-distortion) performance while constraining the computational complexity. Although they also address the ternary relationship between bitrate, encoding time, and quality, their approaches aim at improving the encoding technologies at the macroblock level. References [12, 13] also address the issues associated with handheld devices and power-constrained environments. The most related work to ours are the algorithms proposed by Vanam et al. [14, 15], which are employed to select the parameter settings that have comparable performances with those chosen by exhaustive search using a much fewer number of trials. Rhee et al. [16] propose a real-time encoder, which dynamically adjusts the predefined complexity level to meet the target time budget while maintaining the encoding quality and efficiency.

Both our objective and methodology are different from theirs. First, we are trying to establish a parameter-tuning framework, which can be used for optimizing the encoding cost (or resource consumption) based on a specific service model, while they aim at optimizing the R-D performance. Second, we do not touch the internal of the encoding software, but regard it as a black-box system, whose input is the encoding parameter settings and the original video. We would like to examine the correlation between different parameter settings and encoding performances regardless of the tested video, essentially how the encoding parameters affect the behavior of the encoding software. The parameter-tuning framework is established on the empirical study over a large number of videos. Given a specific video whose encoding complexity can be estimated, an optimal parameter setting which minimizes the encoding cost with satisfiable quality will be generated.

3. Encoding in x264

3.1. Rationale. In this study, we choose H.264 as the video coding standard and use x264 as the encoding software. Our choice of H.264 roots on the simple fact that it has been recognized as the most pervasive video coding standard for web videos with the support of Flash Player 10 by its outstanding compression and efficiency. Currently, the encoding cloud services have integrated a large number of encoding/transcoding software, either open-source or commercial, including On2 Flix [7], Mencoder [17], and Zencoder [18]. Among burgeoning H.264 codec, x264 [19] is one of the best performers (supported by MSU Annual H.264 Codec Comparison Report [20]) and adopted by many widely popular applications including ffmpeg [21], Mencoder, and similar x264 implements almost all of the H.264 features, which means that users have a pretty large parameter configuration space to accommodate their specific encoding requirements, a particular merit of x264. Furthermore, experiments conducted by YUV Soft [22] have shown that using x264 could achieve either excellent compression quality or extremely fast encoding speed with great flexibility, depending on the parameter settings.

3.2. Encoding Model. We model a video for encoding using two parameters:

(i) N: the number of frames;

(ii) F: the frame rate (frame per second), which is the number of frames played in one second.

We use three metrics for evaluating the encoding configuration under H.264.

(i) V: the encoding speed (second/frame), which is the time required to encode one frame. We use encoding rate and encoding speed interchangeably in this paper;

(ii) R: the bitrate after compression with container overhead (kbps), which implies the final file size and the compression ratio;

(iii) *PSNR*: the peak signal-to-noise ratio, which represents the resulting objective quality. There are two variations, average PSNR and global PSNR. Usually global PSNR is a little lower than average PSNR and we choose average PSNR in this paper.

All the above metrics can be easily obtained from the output statistics of x264.

3.3. Rate Control Modes in x264. Modern video encoding software, combined with the flexibility promised by the encoding standard such as H.264, offers handy and functional rate control methods to determine how bits are allocated within streams and tune the delicate balance among bitrate (R), encoding speed (V), and objective quality (*PSNR*). There are four rate control modes in x264, which are used most frequently, namely, CQP (Constant Quantization Parameter), CRF (Constant Rate Factor), ABR (Average Bit Rate), and two-pass VBR (Variable Bit Rate).

CQP mode specifies the same quantization values for all P-frames, meanwhile fixing the relative ratios (i.e., ipratio and pbratio) of quantization values for I-frames and B-frames in the same GOP (group of pictures). CQP mode disables adaptive quantization, a technique aiming to keep constant visual quality by imposing larger quantization values on high-complexity scenes and lower ones on low-motion scenes, based on the assumption that humans are less perceptive to high-motion scenes.

CRF mode targets a certain level of "quality" by adaptively adjusting quantization values for frames with different complexity by setting a constant rate factor. A smaller rate factor means higher quality and higher bitrate. A constant rate factor aims at consistent visual quality, while it may sacrifice some objective quality (PSNR). Generally, CRF mode produces smaller file size than CQP mode and is the most recommended one-pass mode if the quality concern overrides the strictness of the desired bitrate.

ABR and VBR modes focus on achieving the desired bitrate rather than the objective or subjective quality. Reference [23] shows that ABR mode achieves lower PSNR than CRF mode and cannot achieve the exact final file size as specified in one pass. On the same note, two-pass VBR mode gives more accurate file size with possibly the highest quality. However, much more time is needed since the encoding is done practically twice, where the second pass achieves much more accurate bitrate by learning from statistics resulted from the first pass.

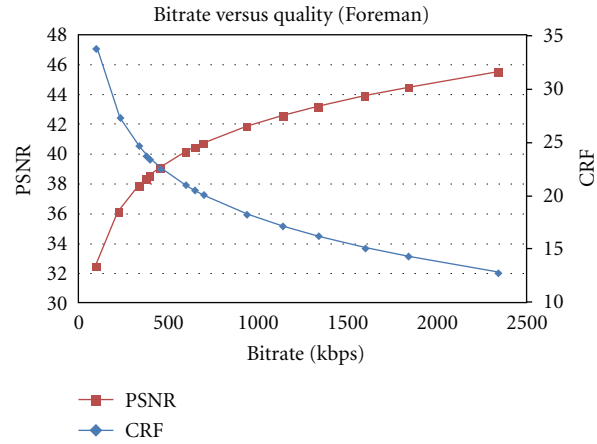

Figure 1: Bitrate versus PSNR and CRF (Foreman).

3.4. Bitrate, Encoding Speed, and Quality. In this section, we obtain a basic understanding of the relationship between the above three evaluation metrics. We choose the ABR mode using which we can target different bitrates and "foreman" as the example video file. Figure 1 shows the relationship between the bitrate and the resulting PSNR. We can see that the larger the bitrate, the better the quality, which is expected, although the quality improvement is becoming smaller with the escalation of bitrate. We also plot the final rate factor with varying bitrates, which is the average quantization value for all types of frames. We can see that the final rate factor is in near reverse proportional with PSNR. Since the rate factor is the major tuning parameter in CRF mode, using CRF mode can target relatively accurate video quality. Figure 2 shows the relationship between the bitrate and the encoding speed. We can see that the larger the bitrate, the longer it requires to encode a frame, since more bits are allocated. In summary, the better video quality requires larger bitrate (final file size) and longer encoding time.

4. Service Model

It is clear that changing encoding parameters simultaneously affects bitrate, encoding speed, and video quality. Thus the optimal selection of encoding parameters becomes a multiobjective optimization problem, where the tradeoffs and constraints among these objectives (e.g., bitrate, encoding speed, and video quality) are dependent on the user's preference and cost structure. Cloud service providers charge the customers based on the resource consumption. They naturally provide a unified cost structure, in which CPU usage time, memory, bandwidth, and storage expenses are represented by singular monetary value and combined as the final charges. The encoding service consumes all major resource types, including CPU usage time, memory, storage

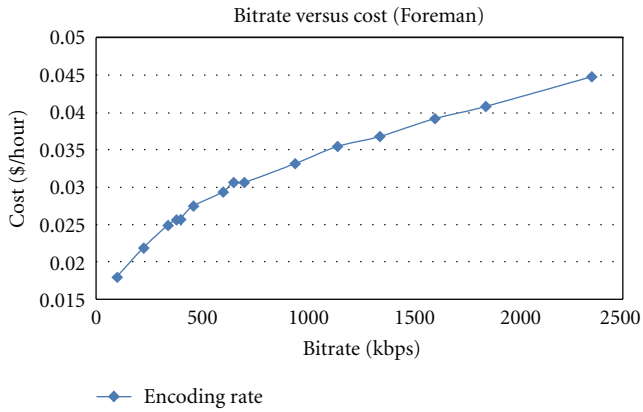

Figure 2: Bitrate versus Encoding speed (Foreman).

space, and bandwidth for data transfer. Here we omit the memory usage in our model, considering the facts that an average modern computer in the cloud (e.g., Amazon EC2 instance) has abundant memory for encoding software and memory consumption alone is not charged by current cloud service vendors. Obviously, the encoding process and the encoding results would affect the usage of various types of resource. Among the remaining three, CPU usage is only related with the encoding of the video, which is a one-time job. The CPU usage can be represented using the encoding time (T), which is calculated as

$$T = N \times V \text{ (second)}. \tag{1}$$

Recall that N is the number of frames and V is the encoding speed as defined in Section 2. The storage and bandwidth (file transfer) usage depends on the compressed file size (S), calculated as

$$S = \frac{N}{F} \times \frac{R}{8} \text{ (byte)}. \tag{2}$$

Recall that R is the bitrate after compression and F is the frame rate (frame per second) as defined in Section 2. Different from CPU usage, the storage and transfer costs of a video can be recurring, and dependent on different encoding service models. We define the cloud encoding cost as COST and merge the encoding time, the storage, and the bandwidth usage into a singular monetary value: operational cost of encoding (briefly encoding cost) in the cloud: COST ($)

$$\text{COST} = \alpha \times T + \beta \times S = N \times \left(\alpha V + \frac{\beta R}{8F}\right), \tag{3}$$

where α and β are cost coefficients related with CPU usage and storage/bandwidth resources, respectively. Since different videos have different number of frames, we could normalize the above cost by eliminating N. We obtain the following normalized cost (COST′, $ per hour), which implies the cost to encode an hour long of this specific video

$$\text{COST}' = \text{COST} \times \frac{3600}{N/F} = 3600 \times \left(\alpha V F + \frac{\beta R}{8}\right). \tag{4}$$

The new cost value, irrelevant with the number of frames, is varied with different videos, since the encoding complexity of a specific video is different from others. The cost coefficients α can be easily set as the current hourly CPU price. For example, we use $0.34/hour here, which is the price for large standard on-demand instances of Amazon EC2. On the other hand, β is dependent on the service model. For example, in the encoding-only service model, the Cloud platform is only used for the computationally extensive encoding tasks. It downloads the encoded video to its local hosting platform for long-term video storage and video distribution. In this case, the storage cost is limited to short term and there is only one-time transfer cost. For example, for transfer cost, we use the Amazon EC2's receding staircase cost starting at $0.15/GB; for the storage cost, we use $0.15/GB/month of Amazon S3 for two weeks. In the full-fledge video service model, the Cloud platform is responsible for long-term storage and distribution in addition to encoding. In this case, the storage cost depends on the storage time and the transfer cost depends on the amount of video access. Since the lifetime of the video to remain in the cloud and its popularity (hence its distribution cost) will be hard to determine, albeit a separate issue. The difficulty of predicting the above values depends on the time horizon. Nevertheless, some simple intuition developed later in this paper can be useful when uncertainty has to be dealt with.

In this work, we consider the encoding-only service model and set the goal of encoding as to maintaining the encoded quality to be acceptable by users through service-level agreement (SLA). In this way, factors other than the encoding aspect of a video (such as its popularity) are excluded.

5. Experiment Study

In the following encoding experiments, we examine the effects of parameter settings on encoding time, bitrate, and objective quality (PSNR), the collection of which determines the final cost. The test sequences we utilized are downloaded from [24], a well-recognized video test set. They are all in CIF format, 25 frames/second, and resolution of 352 × 288. We have tested 12 videos in total, including Bus, Coastguard, Container, Foreman, Flower, Hall, Mobile, News, Silent, Stefan, Tempete, and Waterfall. In order to minimize the stochastic errors, every set of experiments is performed five times and the results are averaged.

To align with the necessity to maintain certain acceptable level of quality, we must choose rate control methods best at controlling quality, which leaves us CQP and CRF. If purely measured by objective quality, that is, PSNR, CQP is a better candidate than CRF. But CRF brings about significant benefits, which are (1) saving more bitrate than CQP and (2) maintaining constant subjective quality, a value much appreciated by end users. As such, we believe that CRF will be commonly preferred by encoding service providers as the primary rate control method, which is also our choice in this study.

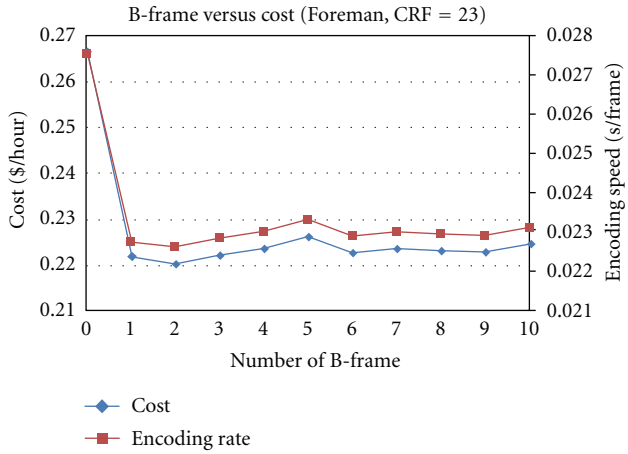

Figure 3: Cost and encoding speed with varying B-frame number (Foreman, crf = 23).

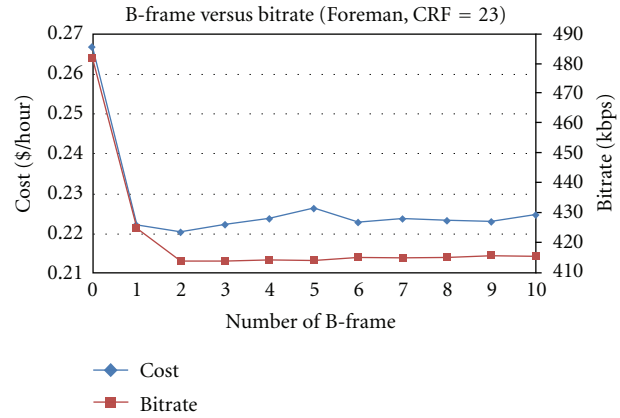

Figure 4: Bitrate and cost with varying B-frame number (Foreman, crf = 23).

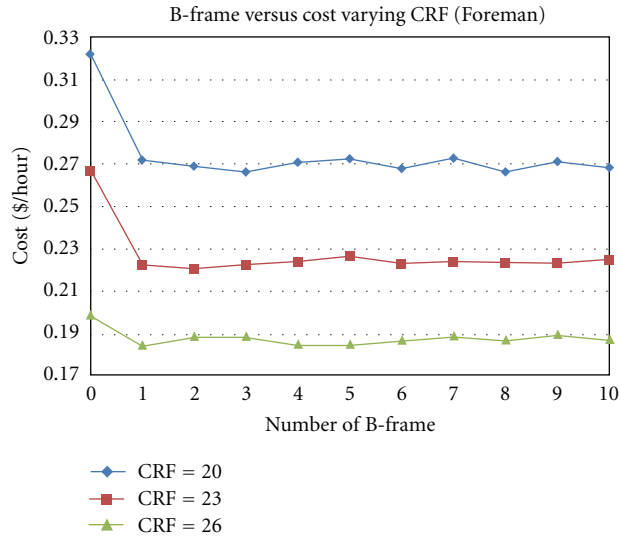

Figure 5: Encoding cost with varying B-frame number under different CRF settings (Foreman).

The excellent performance of x264 is attributed to several factors, including motion estimation, frame-type decision, macroblock mode decision, and quantization. We first vary one parameter while keeping all others default, and then test multiple parameters to examine the composite effect. We note that what will frequently show up in the following figures is a parameter which is also called *CRF*. It sets the target to the CRF method to achieve the perceptual quality effect as if the same quantization value (default value is 23 in x264) is applied to the encoded video.

In the following experiments, if not clearly stated, all the test sequences exhibit similar curves as the example Foreman we will use below, under three quality levels, for concise purpose.

5.1. B-Frame. B-frame setting determines the maximum number of consecutive B-frames we can use. B-frame, different from P-frame, can use motion prediction from future frames, which enhances the compression efficiency.

Figure 3 shows the encoding cost and rate with varying B-frame numbers. We can see that the use of B-frame dramatically decreases the encoding cost, which fluctuates a little bit with higher B-frame setting. The frame-type decision algorithm in x264 for replacing a P-frame with B-frame is fast and flexible. Notice that the cost curve closely matches the encoding speed, which becomes the dominant factor using our chosen cost coefficients α and β, and the minimum cost may be achieved at different points if coefficients vary.

Figure 4 shows the bitrate and encoding cost with varying B-frame numbers. We can see that the resulting bitrate drops with the use of B-frame and down to stable level at the point of two B-frames. We speculate that the most effective number of B-frames for Foreman is 2, which is supported by the consecutive B-frame distributions extracted from x264 output statistics. The distribution of consecutive B-frames for Foreman changes from {13.4%, 86.6%} to {13.1%, 69.8%, 17.1%} if we vary the B-frame number from 1 to 2, which means with one B-frame setting, 13.4% of frames use no consecutive B-frame and 86.6% use one B-frame, while with two B-frame setting, 17.1% of frames use two consecutive B-frames. If we set a cutoff threshold for the cumulative distribution, say 90%, we can see the B-frame number is set to 2, which matches our observation. We also did the same experiments over other videos and conclude that it is appropriate for most videos to use 1 up to 3 B-frames to save significant bitrates as well as encoding time and cost while higher B-frame settings contribute little.

Figure 5 shows the encoding cost with different number of B-frames under varying CRF settings. We can see expected results that the encoding cost increases with CRF values and has marginal variations with higher B-frame numbers. Figure 6 shows the averaged encoding cost using 2 B-frames under different quality levels (CRF value) for all videos. We can see the cost rankings of videos under a specific CRF value are consistent for all the three quality levels (except Hall with CRF = 20). We speculate that it is the encoding complexity, the inherent characteristic of video, that leads to this phenomenon.

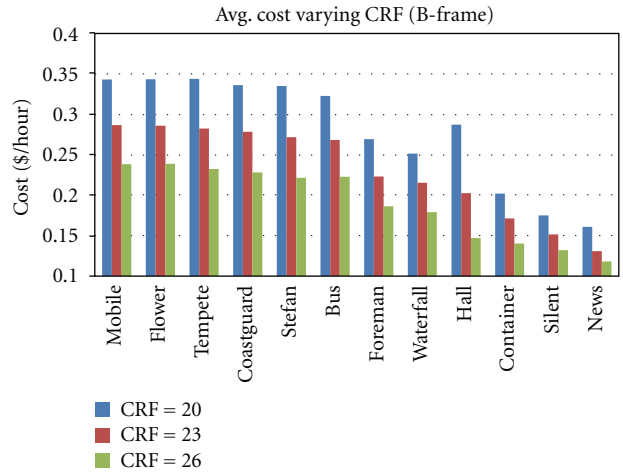

Figure 6: Averaged encoding cost for all videos.

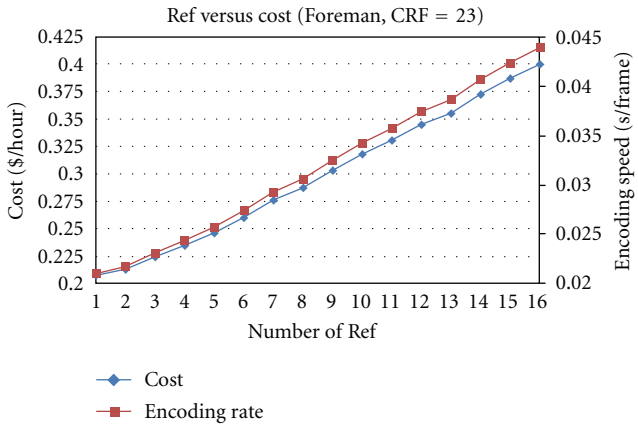

Figure 7: Encoding cost and rate with varying Ref number (Foreman, crf = 23).

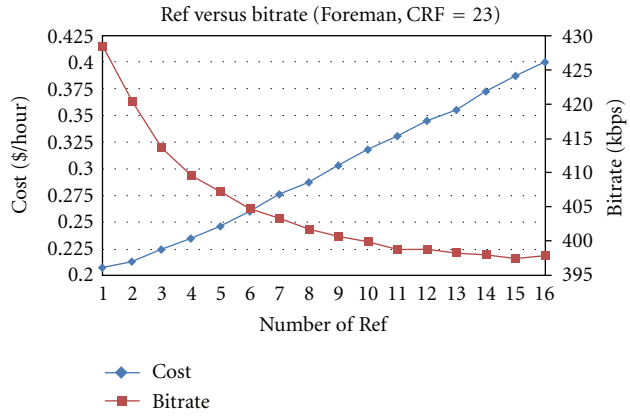

Figure 8: Bitrate and cost with varying Ref number (Foreman, crf = 23).

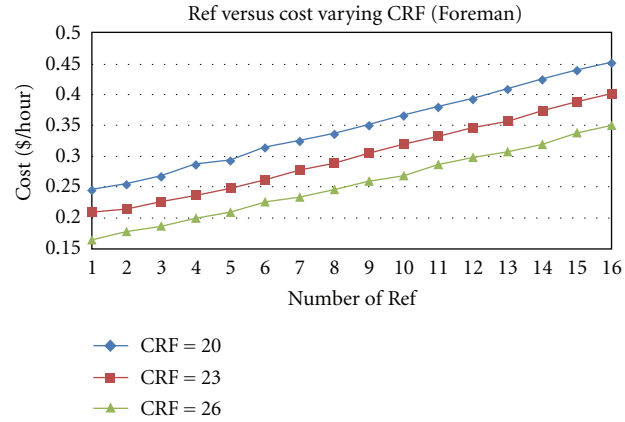

Figure 9: Encoding cost with varying Ref number under different CRF settings (Foreman).

5.2. Ref. Ref is a parameter referring to the number of previous frames that a P-frame can refer to, ranging from 1 to 16. Conceptually, each increment of Ref will bring about constant speed loss, and on the flip side, the benefit of lower bitrate, especially for highly animated videos. On the other hand, the number is dependent on both DPB (Decoded Picture Buffer) confined by ITU for specific playback devices and the preference over speed or bitrate.

Figures 7 and 8 show the encoding cost, rate, and bitrate with varying Ref numbers. As expected, a larger Ref number leads to increasing encoding time and cost, as well as decreasing bitrate. The encoding cost is mostly influenced by speed loss, and the deviation resulting from reduced bitrate is not prominent. Compared to P-frames, B-frames usually use one or two fewer reference frames. If we examine the bitrate more closely, the decreasing gradient and the relatively "stable" point for each video are different from each other.

There are two explanations. One is our choice of cost coefficients, which puts more weight on encoding speed (CPU cost) over bitrate (storage and bandwidth costs). On the same note, it also suggests that the speed loss overshadows the bitrate benefit. The encoding time increases by about two times when Ref varies from 1 to 16, while the bitrate only drops around 30 kbps from original 430 kbps. For some videos, the bitrate variance is within 10 kbps. As such, we may suggest using as few Ref as possible for most videos, except some with high motion or complexity.

Then we vary CRF and get expected result in Figure 9 that the encoding cost increases with the quality improvement. Figure 10 displays the average cost (in most cases also the median value due to approximately linear increase) when varying CRF for all videos. Like Figure 6, the ranking is consistent with Hall being the only exception.

5.3. Subme. Subme controls the subpixel estimation complexity. Since H.264 has a wide range of macroblock partitions for both intra- and interpredictions, mode decision is very time-consuming. Subme levels from 0 to 5 increase the level of refinement, while levels from 6 to 9 enable a hybrid decision of SATD (sum of absolute transformed differences) and RDO (rate distortion optimization) for corresponding frame types. We do not consider level 10 since

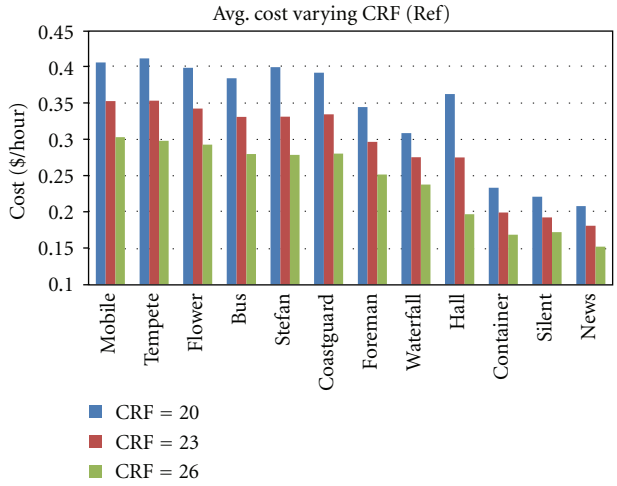

Figure 10: Averaged encoding cost for all videos.

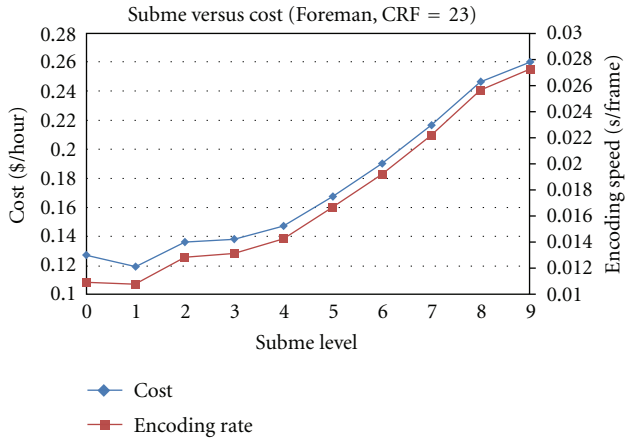

Figure 11: Encoding cost and rate with increasing Subme levels (Foreman, crf = 23).

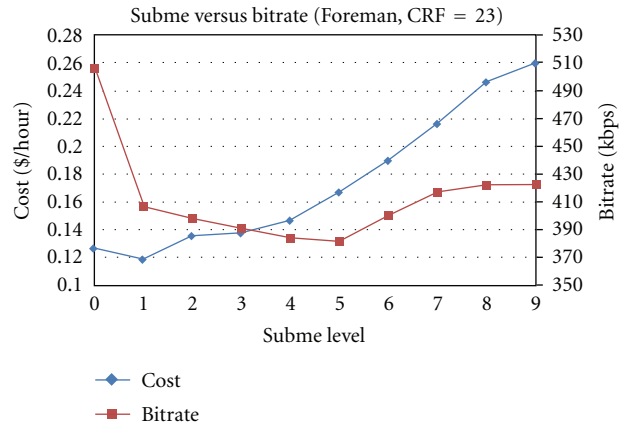

Figure 12: Bitrate and cost with increasing Subme levels (Foreman, crf = 23).

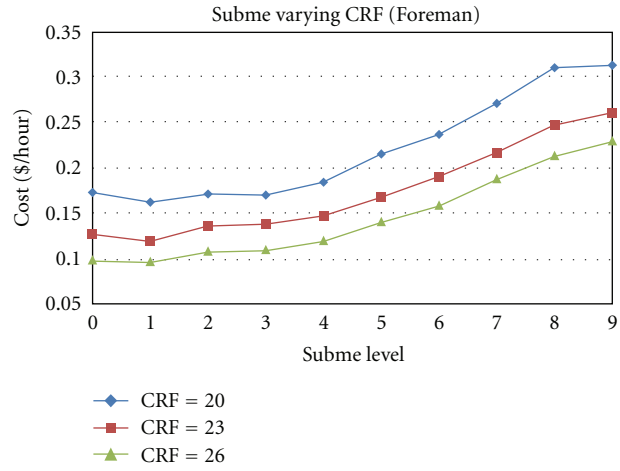

Figure 13: Encoding cost with increasing Subme levels under different CRF settings (Foreman).

it requires trellis option to be enabled, which only applies to extreme cases. Intuitively, higher Subme level leads to better compression efficiency at the price of encoding speed loss.

Figures 11 and 12 show the encoding cost, rate, and bitrate with increasing Subme levels. We can see that the encoding cost increases as Subme does, except a little decrease when the level goes from 0 to 1. However, in Figure 12, there is a valley at level 5 on the bitrate curve, suggesting that the application of SATD and RDO in fact increases the bitrate. This is mainly due to the rate control method (CRF) of our choice, which focuses on perceptual quality rather than the objective quality PSNR, which these two techniques target to improve. Similarly, Figure 13 shows the average encoding cost under different quality levels.

5.4. Trellis. Trellis quantization is used to optimize residual DCT coefficients by reducing the size of some coefficients while recovering others to take their place. This technique can effectively find the optimal quantization for each macroblock to optimize the PSNR relative to bitrate. There are three options in x264: level 1 applies conventional uniform deadzone; level 2 uses trellis only on the final encode of macroblock, while level 3 does them all. In addition, trellis quantization in x264 requires CABAC (Context Adaptive Binary Arithmetic Coder), instead of the less-expensive and more-popular CAVLC (Context Adaptive Variable Length Coder). Thus, we introduce a level 0, which repeats level 1, only replacing CABAC with CAVLC.

Figures 14 and 15 show the encoding cost, rate, and bitrate with varying trellis levels. Trellis is similar to the case of Subme (Figures 11 and 12), in which the encoding cost grows as the level increases, but the bitrate fluctuates. Again, this suggests that the objective of trellis quantization (optimizing objective quality PSNR) does not always agree with the objective of the CRF method, which is the perceptual quality. This is further confirmed by our later observation on PSNR. We also see that CABAC does not cause additional speed loss while reducing bitrate by about 10%, which implies that CABAC should always be turned on. Not surprisingly, applying trellis quantization increases encoding time, especially for the level 3. In Figure 16, the encoding

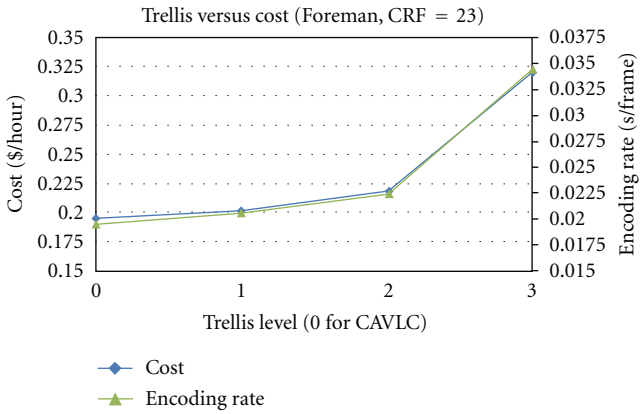

Figure 14: Encoding cost and rate with varying Trellis levels (Foreman, crf = 23).

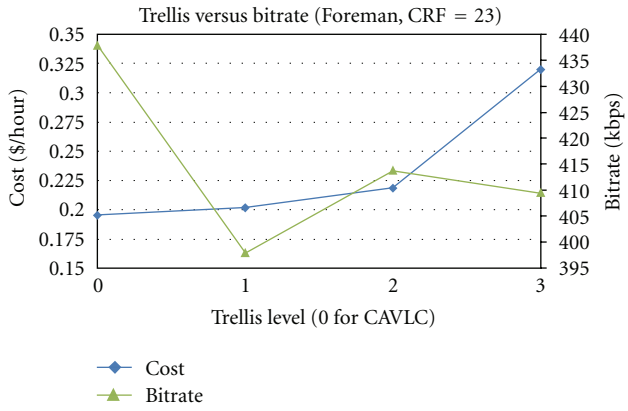

Figure 15: Bitrate and cost with varying Trellis levels (Foreman, crf = 23).

cost increases with the quality improvement, matching observations obtained from all previous experiments.

5.5. CRF versus PSNR. Although targeting at perceptual quality, CRF is known to be able to still achieve relatively constant objective quality level, that is, PSNR. To quantify the deviation, we observe PSNR in the above four parameter settings and obtain the following results in Table 1 (take Foreman, e.g.). The deviation is defined as the percentage of difference between lowest and highest PSNR to the highest value. We can see that during B-frame, Ref and Trellis tests, such change is within 1% range, while for Subme it is still around 3%. This makes us more confident to use CRF as the indication for quality level. Averaging all PSNR values from all tests gets us the correspondence between PSNR and CRF, as shown in Figure 17. The three quality levels show consistent rankings for all 12 videos, which implies that the encoding complexity varies with different videos.

5.6. Composite Effect of Multiple Parameters. In previous experiments, we only vary one parameter and leave all others default. Default setting uses 3 B-frames, 3 Refs, Subme level 7, and Trellis level 1. We plot the encoding costs from four parameters together in Figure 18. Approximately, the

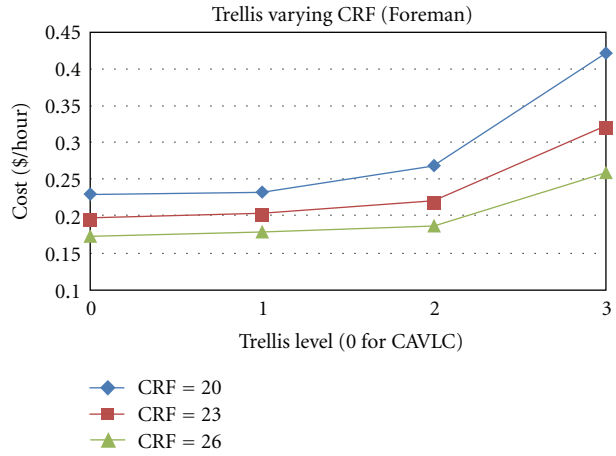

Figure 16: Encoding cost with varying Trellis levels under different CRF settings (Foreman).

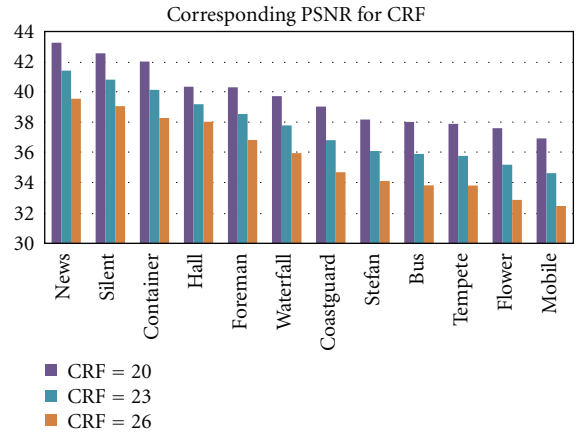

Figure 17: Correspondence between PSNR and CRF setting.

same costs are achieved with their respective default values. Subme leads to relatively lower costs while Ref renders linearly increasing costs with Ref number. The encoding costs incurred by all the settings, except for the cases of Trellis level 2, more than 3 Refs and Subme level 8 and 9, are bounded by the costs in B-frame tests. We also test for CRF value of 20 and 26, which show similar patterns. Therefore, we can define the baseline cost for Foreman under different quality levels as the averaged encoding cost in the B-frame tests. By this definition, our B-frame test (Figure 3) actually shows the baseline costs for all videos.

The encoding cost is influenced by the comprehensive impacts of multiple parameters. In order to understand how separate parameters collaborate with each other, in one experiment we vary all the four parameters, that is, B-frame from 1 to 3, Ref from 3 to 12, Subme level from 5 to 7, and Trellis level 0 and 1. There are 180 test points in total. The results are shown in Figure 19. We can see that the encoding cost shows a recurring pattern, which implies that the aggregated cost can probably be decomposed into separate weighted factors attributable to specific parameters.

Table 1: PSNR versus CRF (foreman).

Foreman	crf = 20		crf = 23		crf = 26	
	PSNR mean	Deviation (%)	PSNR mean	Deviation (%)	PSRN mean	Deviation (%)
B-frame	40.38	0.51	38.60	0.09	36.88	0.33
Ref	40.45	0.79	38.73	1.08	37.03	1.21
Subme	40.05	2.6	38.25	3.01	36.49	3.26
Trellis	40.22	0.73	38.46	0.71	36.74	0.77

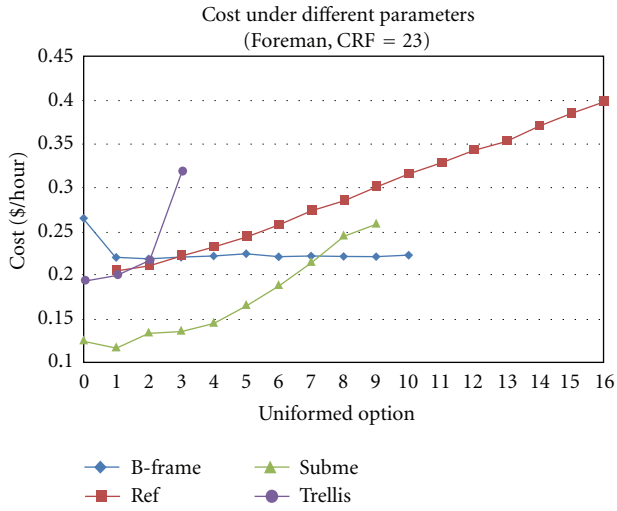

Figure 18: Encoding cost with four parameters setting together (Foreman, crf = 23).

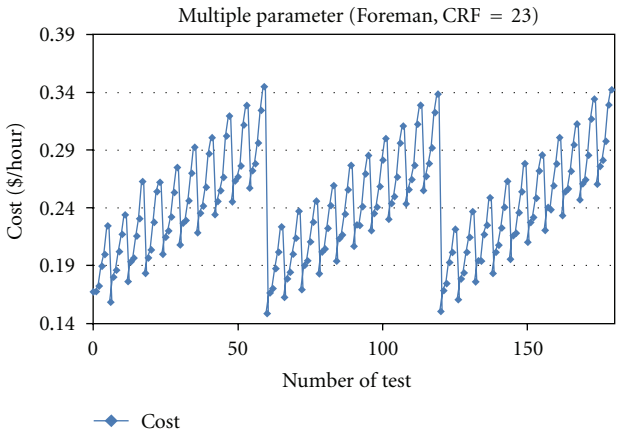

Figure 19: Encoding cost under combinations of four parameters (Foreman, crf = 23).

We also find that the average PSNR follows the similar pattern in Figure 20. PSNR ranges from 38.07 to 38.854, with the deviation of less than 2%. On a closer look, Trellis, Subme, and Ref consistently produce the expected impacts on PSNR.

5.7. Other Parameters. We have tested all major numerical parameters in x264, at least all that applicable to the CRF method. We here list other important parameters not included in our tests as below.

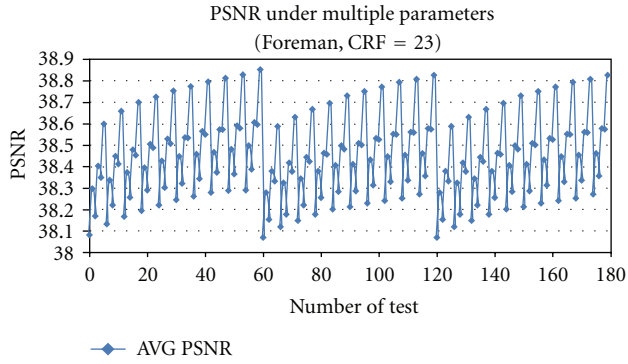

Figure 20: PSNR under combinations of four parameters (Foreman, crf = 23).

(i) ME determines the full-pixel motion estimation methods, and there are four choices in x264, namely, DIA (diamond), HEX (hexagon), UMH (Uneven Multi Hexagon), and ESA (Exhaustive Search). Generally, UMH provides good speed without sacrificing significant quality, while HEX is a better trade-off. We are using the default option HEX.

(ii) Partition controls the split up of 16 × 16 macroblock. Partitions are enabled per frame type and for intra- and interprediction, respectively. The default setting in x264 is "p8 × 8, b8 × 8, i8 × 8, i4 × 4." The p4 × 4 is not defaulted due to the high speed cost with respect to quality gain.

The above two parameters have also essential impacts on the encoding speed and quality. The reason why they do not fall into the scope of our consideration is that there is no apparent and absolute ranking among those options, which makes them hardly fit into our quantified parameter tuning framework. Besides, the empirical guidance about using the above two parameters are quite clear.

6. Discussion

Based on all the experiment results in the previous section, we claim the following observations.

(1) For all the tests conducted, if grouped by targeted CRF values, the resulting PSNR values in each group remain largely equal with a maximum deviation of 3% (Table 1).

(2) Each tested video, when trialed with different quality levels (CRFs), has its PSNR values shown with consistent ranking (Figure 17).

(3) Tests conducted regarding each parameter shows at least one period of asymptotic growth of encoding cost (e.g., when Ref ranges from 1 to 16), and one clear optimal point to minimize the cost (e.g., when Ref = 1). This observation also holds true across different quality levels tested.

The first two observations collectively confirm the effectiveness of CRF as a rate control method to accurately obtain consistent results on measurable and objective quality, that is, PSNR. Combined with the third observation, we can claim that our primary goal set at the end of Section 3 can be achieved. Conversely, with a clearly defined service model and desired quality level set by the user, our framework can find the optimal parameter setting to minimize the encoding cost.

However, this only works well in the best-effort way, that is, one can claim confidently that the encoding cost is indeed minimized, but never know how much the cost is until the job is done. This leads to the predictive power of the optimization framework, that is, can we predict the encoding cost of a video before it is encoded? This feature, if at all achievable, will be significantly valuable to the budget planning (hence projection of price and profit margin) for any service providers, especially appreciated when unclear service models are present, typically when it comes to the cloud-based distribution service whose storage and bandwidth costs are hard to determine. We further claim the following observations from our tests.

(4) By the functioning principles of block-based hybrid video coding, the tested parameters (e.g., Ref, Trellis, etc.) operate largely independent from each other. Our tests establish that, regarding each tested parameter, its asymptotic growth window (mentioned in claim (3)) is not affected by the changing of other parameters.

(5) Aside from independence, it is very likely that these parameters can be decomposed in a weighted sum fashion to qualify their contribution to the final encoding cost. Furthermore, such weights are not subject to the changing of the cost coefficients α and β.

(6) The B-frame test as the baseline cost (Figure 18) provides a worst-case envelope for the cost estimation if the prospect of claim (5) gets gloomy.

All these claims sound promising, even to a self-improving framework, which, with the growth of its sampling set, provides increasingly accurate weight estimation.

However, we must not ignore the elephant in the room. In Section 3, we mentioned the encoding complexity, which originates from the intrinsic characteristics of the video content itself. Evidently, Figure 21 shows the resulting bitrates of all 12 videos under the default setting (crf = 23) of x264, where the most complex video is encoded with

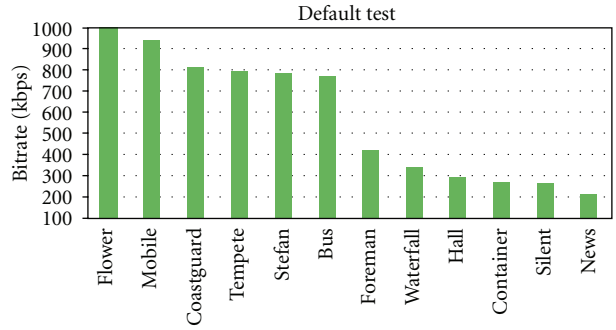

Figure 21: Resulting bitrates under the default setting of x264 for all videos.

bitrate 5 times the one of the lowest complexity. Figures 6 and 10 further confirm that this ranking holds across different quality levels, with the Hall as the only exception, indicating that the encoding complexity is largely rooted in the content itself.

There is no silver bullet to this problem. In order to learn the encoding complexity of a video, one has to encode it to know it, which paradoxically defeats the very propose of saving encoding cost. Fortunately, common practices of the modern encoding service largely alleviate this problem. First, most of the original content is already in the digital format (usually encoded by block-based hybrid video coding technology) awaiting transcoding to be suitable to play on the new platform. Second, the diversity of end-user terminals requires each video to be encoded multiple times to different qualities, resolutions, and formats. As such, we can easily obtain statistics from metadata of the originally encoded file to obtain the first-hand knowledge of the encoding complexity.

7. Conclusion

In this paper, we conduct an empirical study in the cloud-computing environment, targeting an optimization framework to save H.264 encoding cost by tuning the key parameters. Our study shows convincing results, in which the rate control method (CRF) demonstrates remarkable ability to precisely achieve the targeted quality level, and the tested parameters can be independently tuned to minimize the encoding cost. We have also discussed the encoding complexity as a major hurdle to the predictive power of our framework and how it can be alleviated by leveraging the diversity of end-user platforms.

Towards future study, the most urgent task is to quantify the contributing weight of key parameters to encoding cost, through more extensive experiments. Also we will incorporate important but nonnumerical parameters (such as motion estimation method) and examine its fitness to the current framework.

The final note is on the objective of our optimization framework, which is to minimize encoding cost, meanwhile maintaining certain quality level. It is not impossible for service providers to reverse the equation to give a cost budget,

try to maximize the encoded quality. In light of such a change, we should switch to rate control methods best at controlling bitrate, that is, ABR and VBR. Study of this flavor will be especially valuable when distribution cost becomes the overwhelming concern. Whether a similar parameter-tuning framework could be established on this regard could be only answered by another empirical study.

References

[1] Amazon Web Services, http://aws.amazon.com/.

[2] Google Web Application Engine, http://code.google.com/appengine/.

[3] Microsoft Azure, http://www.microsoft.com/windowsazure/.

[4] http://www.youtube.com.

[5] http://www.youku.com.

[6] http://www.tudou.com.

[7] http://www.zencoder.com/flixcloud/.

[8] http://mewiki.project357.com/wiki/X264_Settings.

[9] A. Jagmohan and K. Ratakonda, "Time-efficient learning theoretic algorithms for H.264 mode selection," in *Proceedings of the International Conference on Image Processing*, pp. 749–752, October 2004.

[10] T. A. da Fonseca and R. L. de Queiroz, "Complexity-constrained H.264 HD video coding through mode ranking," in *Proceedings of the Picture Coding Symposium*, pp. 329–332, 2009.

[11] E. Akyol, D. Mukherjee, and Y. Liu, "Complexity control for real-time video coding," in *Proceedings of the IEEE International Conference on Image Processing*, pp. I77–I80, September 2007.

[12] X. Li, M. Wien, and J.-R. Ohm, "Rate-complexity-distortion optimization for hybrid video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 7, pp. 957–970, 2011.

[13] L. Su, Y. Lu, F. Wu, S. Li, and W. Gao, "Complexity-constrained H.264 video encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 4, pp. 477–490, 2009.

[14] R. Vanam, E. A. Riskin, S. S. Hemami, and R. E. Ladner, "Distortion-complexity optimization of the H.264/MPEG-4 AVC encoder using the GBFOS algorithm," in *Proceedings of the Data Compression Conference*, pp. 303–312, March 2007.

[15] R. Vanam, E. A. Riskin, and R. E. Ladner, "H.264/MPEG-4 AVC encoder parameter selection algorithms for complexity distortion tradeoff," in *Proceedings of the Data Compression Conference*, pp. 372–381, March 2009.

[16] C.-E. Rhee, J.-S. Jung, and H.-J. Lee, "A real-time H.264/AVC encoder with complexity-aware time allocation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1848–1862, 2010.

[17] http://www.mplayerhq.hu.

[18] http://www.zencoder.com/.

[19] http://www.videolan.org/developers/x264.html.

[20] MSU Graphics & Media Lab (Video Group), "Fourth annual msu mpeg-4 avc/h.264 video codec comparison," http://compression.ru/video/codec comparison/mpeg-4 avc h264 2007 en.html.

[21] http://www.ffmpeg.org/.

[22] YUV Soft Inc., http://www.yuvsoft.com.

[23] L. Merritt and R. Vanam, "Improved rate control and motion estimation for H.264 encoder," in *Proceedings of the International Conference on Image Processing*, vol. 5, pp. 309–312, 2007.

[24] http://media.xiph.org/video/derf/.