A SCALABLE COMPLEXITY SPECIFICATION FOR VIDEO APPLICATIONS

Nicholas Mastronarde, Mihaela van der Schaar

University of California, Los Angeles (UCLA)

ABSTRACT

We propose a new complexity modeling framework for multimedia tasks. We characterize the traffic with five parameters that together we designate as a task's *complexity specification* (CSPEC). We extend this model to a *scalable* CSPEC, which can be used to characterize the many complexity- and quality-scalable operating points available to multimedia tasks. The proposed scalable CSPEC can be used by multimedia applications to match their resource requirements to available system resources.

Index Terms— Video decoding complexity, complexity scalability

1. INTRODUCTION

A majority of state-of-the-art video coders can create bit-streams that can be decoded at different quality- and complexity-levels. This is achieved by dividing the bit-stream into layers, partitions, video packets etc. that can be decoded successively in order to gradually improve the video quality at an increased computational cost. Fine complexity scalability can be achieved by simultaneously adjusting the number of decoded layers, l, and the average extracted bit-rate r [2].

The majority of work done on multimedia complexity analysis or prediction does not consider that multimedia applications can cooperate with the system they run on. Specifically, much of this work does not provide models that are appropriate for admission control and resource allocation scenarios. In [4], analytic models are developed based on coder-specific assumptions that cannot be easily generalized to other coding schemes or other coding/decoding configurations. Therefore, such models cannot capture the complexity-scalability available to video decoding applications. In [1], a statistical regression approach is used to model video decoding complexity. Complexity prediction is done using histograms of previous measurements and a linear prediction model. This technique can be used to meet an arbitrary percentage of deadlines (e.g. 95%), however, worst-case CPU allocations must be made in order to meet 100% of deadlines.

In this paper, we propose a complexity model that captures the complexity scalability of video applications and enables 100% of deadlines to be met without using worst-cast resource allocations.

2. ADAPTIVE WORKLOAD MODELING

2.1. Properties of Video Decoding Workloads

In this paper, we focus on MCTF-based temporal scalability. However, our approach is general and not limited by the particular layering/partitioning choice or video coder. Table 1 shows the decoding deadlines for the frames of an MCTF structure with $T = 4$ temporal layers (i.e. it is partitioned into $l = 1, 2, 3, 4$ temporal layers). In Table 1, t_0 denotes the display deadline of the pair of original frames $\{A_0, A_1\}$ and the deadline of each subsequent pair is 2Δ seconds after the previous pair's deadline. The value of Δ depends on the encoded frame-rate and, for our experiments, is set as $\Delta = 0.0333\,\mathrm{s}$ to correspond to a 30Hz encoded frame-rate. We note that a 5/3 Haar filter temporal decomposition is used in Table 1, but similar results can be obtained using other decompositions [3].

Table 1. Display deadlines for decoding frames partitioned into $T = 4$ temporal layers. Decoding l layers requires decoding the frames in columns $1, \ldots, l-1$ as well as the frames in column l.

Display Deadline	Number of Decoded Layers (l)			
	1	2	3	4
t_0	$L_{4,0}$, $\mathrm{H}_{4,0}$, $L_{3,0}$	$\mathrm{H}_{3,0}$, $L_{2,0}$	$\mathrm{H}_{2,0}$, $L_{1,0}$	$\mathrm{H}_{1,0}$, A_0, A_1
$t_0 + 2\Delta$	$L_{3,1}$	-	$L_{1,1}$, $L_{1,2}$	$\mathrm{H}_{1,1}$, A_2, A_3
$t_0 + 4\Delta$	-	$\mathrm{H}_{3,1}$	$\mathrm{H}_{1,2}$, A_4, A_5	$\mathrm{H}_{1,2}$, A_4, A_5
$t_0 + 6\Delta$	-	$L_{2,1}$, $L_{2,2}$	$\mathrm{H}_{2,2}$	$\mathrm{H}_{1,3}$, A_6, A_7
$t_0 + 8\Delta$	-	-	-	$\mathrm{H}_{1,4}$, A_8, A_9
$t_0 + 10\Delta$	-	$L_{2,3}$	$\mathrm{H}_{2,3}$, $L_{1,5}$, $L_{1,6}$	$\mathrm{H}_{1,5}$, A_{10}, A_{11}
$t_0 + 12\Delta$	-	-	-	$\mathrm{H}_{1,6}$, A_{12}, A_{13}
$t_0 + 14\Delta$	-	-	$L_{1,7}$	$\mathrm{H}_{1,7}$, A_{14}, A_{15}

To illustrate several key properties of video decoding workload traffic, which make it challenging to characterize and model, example traffic for a variety of scenarios is shown in Fig. 1. We will describe each plot in Fig. 1 individually.

Workload depends on the sequence characteristics: To illustrate the impact of varying video source characteristics, ξ, on the decoding workload across a GOP, Fig. 1(a-b) show the upper- and lower-bounds on the workload traffic for GOPs four and five (frames 65-96) of the *Silent* and *Stefan* sequences, respectively (CIF resolution, 30Hz encoded frame-rate, and rate $r \in [200, 1536]$ Kb/s). Comparing Fig. 1(b) and Fig. 1(a), it is clear that decoding *Stefan* is significantly more computationally complex than decoding *Silent* for most choices of l. This observation is congruent with the intuition that *Stefan*'s intense motion characteristics should generally yield heavier peak and mean computational workloads at the CPU. Hence, the complexity depends on a particular video source's characteristics, ξ.

Workload is highly time-varying: From Table 1 it is clear that the number of frames that must be decoded at each deadline is not distributed evenly over the duration of the GOP. For example, when decoding all the layers, only 3 frames require decoding or reconstruction during the time-interval $(t_0 + 6\Delta, t_0 + 8\Delta]$ while 10 frames are decoded or reconstructed during the time-interval $(t_0 - 2\Delta, t_0]$. The time-varying workload curves in Fig.

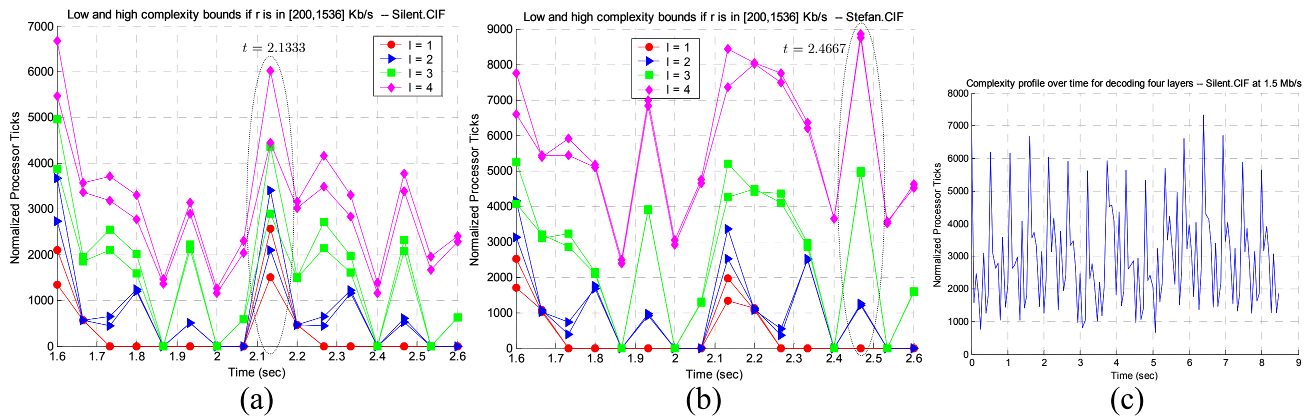

Fig. 1. (a,b) Typical complexity profile for decoding $l = 1,2,3,4$ temporal layers (with $T = 4$ temporal level MCTF structure) of one GOP of the *Silent* and *Stefan* sequences, respectively. Includes upper- and lower-bounds on the decoding complexity when decoding $l = 1,2,3,4$ layers with rate $r \in [200,1536]$ Kb/s. Notice that the y-axis scales are different for each sequence. (c) Typical complexity profile over many GOPs when decoding $l = 4$ layers of the *Silent* sequence.

1(a-b) reflect this unbalanced workload distribution within a GOP. Fig. 1(c), on the other hand, shows how decoding workloads are also typically time-varying across several GOPs. This is the consequence of changes in motion and texture characteristics over the duration of the video sequence.

Workload is rate dependent: Fig. 1(a-b) also illustrates how rate affects the decoding complexity. Notice that adjusting the rate can have a significant impact on the complexity, particularly at the first decoding deadline in a GOP where the base-layer frames (i.e. $L_{T,0}, \mathrm{H}_{T,0}$ using the MCTF coder, or an I frame in H.264) containing most of the texture information are decoded. For example, in Fig. 1(a), at the circled complexity measurements at $t = 2.1333$ s (i.e. the first decoding deadline for GOP 4), adjusting the rate $r \in [200,1536]$ significantly impacts the decoding complexity. Hence, adjusting the rate r can increase a task's chances for admission into a system with limited resources. In other cases, when minimal residual texture information is decoded, rate-independent motion compensation operations dominate the complexity (see the circled complexity measurement at $t = 2.4667$ s in Fig. 1(b)).

Decoding different layers leads to complexity scalability: Fig. 1(a-b) also illustrate that significantly reduced peak and mean workloads can be achieved by decoding less layers. The wide range of complexity scalability enabled by decoding various layers is important in an admission control scenario where a task may not be allocated any processor resources if it cannot adapt its workload to the resources available to it.

Based on the above observations pertaining to the time-varying decoding workloads, a workload traffic model that captures all of these characteristics and can provide latency guarantees through an admission control process is highly desirable. We present such a model in the next subsection. Note that while the above observations were made for one particular coder, similar observations can be made for other coders such as H.264/AVC and MPEG-4 using the same methodology.

2.2. Characterizing Video Decoding Workload Traffic

In order to capture a video decoding task's time-varying and bursty resource requirements, we model the decoding workload traffic with a twin leaky bucket. We assume that the task decodes l layers at rate r. The important model parameters are the *Peak Workload* $P(l,r)$ (cycles/second), *Mean Workload* $\rho(l,r)$ (cycles/second), *Maximum Burst Size* $\sigma(l,r)$ (cycles), and *Delay* $d(l,r)$ (seconds). $d(l,r)$ is set based on the application requirements or user preferences. The remaining parameters can be determined using offline modeling, training, or profiling, followed by real-time classification [2] [4].

We would like to develop a model that considers both the Peak Workload and the Mean Workload for two reasons: considering just the Peak Workload $P(l,r)$ results in over conservative worst-case complexity estimates that inefficiently use the CPU bandwidth; conversely, considering only the Mean Workload $\rho(l,r)$ under allocates CPU bandwidth during time intervals where the workload exceeds the Mean Workload and therefore results in missed decoding deadlines and, consequently, frame drops. By enforcing the (small) Delay $d(l,r)$ on all display deadlines (i.e. the display deadlines in Table 1 become $t_0 + 2\Delta n + d(l,r)$ for $n = 0,\ldots,7$) the bursty workload can be smoothed to reduce the peak computational complexity.

This delay parameter was first introduced in [5] in order to reduce the peak computational capacity required by a device to decode a particular bit-stream. In this paper, we expand on the concept by also exploiting complexity-scalability which, given a fixed Delay parameter, allows a task to adapt its complexity to match available resources.

Based on a twin leaky bucket analysis, the *CPU Bandwidth Demand* for decoding l layers at rate r with Delay $d(l,r)$ is:

$$g(l,r) = \frac{P(l,r)}{1 + d(l,r)[P(l,r) - \rho(l,r)]\sigma^{-1}(l,r)}. \qquad (1)$$

Together, the token bucket parameters and the CPU Bandwidth Demand determine the task's *Complexity Specification* (CSPEC)

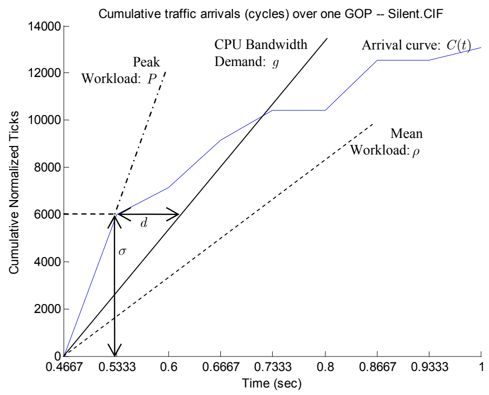

Fig. 2. Arrival curve for one GOP of the *Silent* sequence for decoding $l = 3$ layers with $T = 4$ and calculation of CPU Bandwidth Demand.

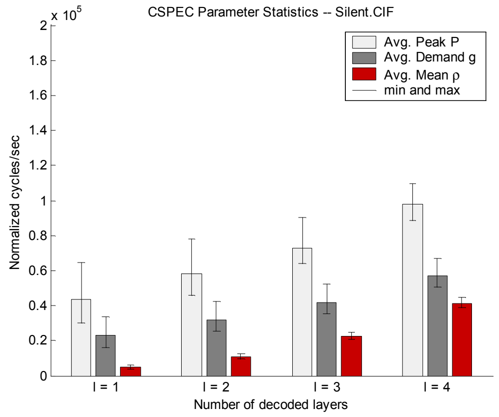

Fig. 3. CSPEC parameter statistics for various complexity strategies based on 16 GOP normalized computation workload with Delay $d(l,r) = 2\Delta$, Peak CPU Bandwidth $P(l,r)$, Mean CPU Bandwidth $\rho(l,r)$, and CPU Bandwidth Demand $g(l,r)$ for decoding $l = 1,2,3,4$ layers.

denoted as the set $\mathbf{g}(l,r) = \{g(l,r), P(l,r), \rho(l,r), d(l,r)\}$, where the Max Burst Size $\sigma(l,r)$ is omitted because it can be determined as

$$\sigma(l,r) = 2\Delta \cdot P(l,r) \text{ cycles.} \qquad (2)$$

Intuitively, $\sigma(l,r)$ can be expressed as in (2) because it corresponds to the maximum processor workload during any 2Δ second time interval. We note that the CPU Bandwidth Demand $g(l,r)$ in (1), corresponding to the smoothed workload, is the parameter that we believe should be used for resource allocation and admission control because it resolves the aforementioned issues with the Peak and Mean Workloads. Fig. 2 shows how the various CSPEC parameters are determined using the cumulative workload traffic arrival curve $C(t)$. Note that in Fig. 2, P, ρ, and g are the slopes of the respective lines.

Importantly, this CSPEC definition can be used to characterize the decoding workloads of other coders. The CSPEC can also be adapted to characterize video encoding workloads.

2.3. The Scalable Complexity Specification

Depending on the sequence characteristics, ξ_i, task i might deploy different decoding strategies in order to adapt its CSPEC and negotiate with the *Resource Manager* for its fair share of resources. Note that each video coder and task can implement its own decoding strategy set. Here, for illustration purposes, we define $\mathbf{a}_i = [l_i, r_i] \in \mathcal{A}_i$, $i = 1,\ldots,M$ as a *complexity strategy* vector in the feasible set of complexity strategies for task i, where $\mathcal{A}_i = \mathcal{A}_i^{\text{LAYER}} \times \mathcal{A}_i^{\text{RATE}}$ and $\mathcal{A}_i^{\text{LAYER}} = \{l_i^1,\ldots,l_i^{N_i^{\text{LAYER}}}\}$ and $\mathcal{A}_i^{\text{RATE}} = \{r_i^1,\ldots,r_i^{N_i^{\text{RATE}}}\}$ denote the decoding strategy spaces enabling spatio-temporal decoding tradeoffs and bit-rate adaptations (corresponding to the SNR scalability), respectively. We denote the cardinalities of the strategy spaces as $N_i^{\text{LAYER}} = |\mathcal{A}_i^{\text{LAYER}}|$ and $N_i^{\text{RATE}} = |\mathcal{A}_i^{\text{RATE}}|$. Note that the feasible complexity strategies and decoding strategy spaces for the i-th task depend on its video source characteristics, ξ_i, for the reasons described in Section 2.1. For notational simplicity, we do not explicitly indicate this dependence. By selecting the complexity strategy vector $\mathbf{a}_i = [l_i, r_i] \in \mathcal{A}_i$, a task operates at one of the $N_i = N_i^{\text{LAYER}} \cdot N_i^{\text{RATE}} = |\mathcal{A}_i|$ feasible complexity levels, which define its *Scalable* CSPEC. Formally, we define the i-th task's Scalable CSPEC as the set,

$$\begin{aligned}\mathcal{G}_i(\mathcal{A}_i) &= \{\mathbf{g}_i(\mathbf{a}_i) \mid \mathbf{a}_i \in \mathcal{A}_i\} \\ &= \{\{g(\mathbf{a}_i), P(\mathbf{a}_i), \rho(\mathbf{a}_i), d(\mathbf{a}_i)\} \mid \mathbf{a}_i \in \mathcal{A}_i\},\end{aligned} \qquad (3)$$

where $g_i(\mathbf{a}_i)$, rewritten using the complexity strategy notation, is determined by (1) using the corresponding Peak Workload $P_i(\mathbf{a}_i)$, Mean Workload $\rho_i(\mathbf{a}_i)$, Max Burst Size $\sigma_i(\mathbf{a}_i)$ determined by (2), and the Delay $d_i(\mathbf{a}_i)$, which relaxes the decoding/display deadlines.

3. EVALUATION

3.1. Scalable CSPEC statistics

Fig. 3 illustrates example Scalable CSPEC statistics for the first 256 frames of the *Silent* sequence (CIF resolution, 30 Hz encoded frame-rate). The Delay $d(\mathbf{a}_i)$ is set to 2Δ. The example operating points, $\mathbf{g}_i(\mathbf{a}_i) \in \mathcal{G}_i(\mathcal{A}_i)$, from which the statistics are gathered, are defined by the $N_i = 52$ complexity strategies $\mathbf{a}_i = [l_i, r_i] \in \mathcal{A}_i$ with the vector components $l_i \in \mathcal{A}_i^{\text{LAYER}} = \{1,2,3,4\}$ and $r_i \in [200, 1536]$ Kb/s, with $N_i^{\text{LAYER}} = 4$ and $N_i^{\text{RATE}} = 13$

The solid bars in Fig. 3 are the averaged value of the corresponding parameter (i.e. $P(\mathbf{a}_i)$, $g(\mathbf{a}_i)$, or $\rho(\mathbf{a}_i)$) over 13 measurements taken for bit-rates between 200Kb/s and 1.5Mb/s. The error bars show the maximum and minimum value of the corresponding parameter for the 13 measurements.

The solid bars in Fig. 3 illustrate that, by selecting $l_i \in \mathcal{A}_i^{\text{LAYER}}$, a task's CPU Bandwidth Demand $g(\mathbf{a}_i)$ can be scaled to below half of its maximum. Additionally, observing the error bars, it is clear that adjusting $r_i \in \mathcal{A}_i^{\text{RATE}}$ yields finer complexity scalability by approximately 10-40% of the maximum CPU Bandwidth Demand for a fixed value of l_i. Clearly, the feasible

Table 2. Deadline miss percentage for different deadline/priority classes for two bandwidth allocation strategies based on either meeting 95% of all deadlines or only providing the mean CPU bandwidth required by a task. Comparison of PSNRs for different resource allocations.

Sequence	Allocated CPU Bandwidth	Deadline / Priority Class Deadline Miss %							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Silent	95%	43.75	0	0	0	0	0	0	0
	Mean	100.0	68.75	62.50	50.00	0	81.25	0	0
Stefan	95%	43.75	0	0	0	0	0	0	0
	Mean	100.0	87.50	75.00	0	0	87.50	0	0

Sequence	PSNR (Proposed)	PSNR (95%)	PSNR (Mean)
Silent	38.25 dB	37.14 dB	31.37 dB
Stefan	35.03 dB	32.06 dB	26.46 dB

set of complexity strategies defines a wide range of complexity scalability. Therefore, complexity strategies are essential when tasks negotiate for limited system resources with the resource manager.

The Delay $d(\mathbf{a}_i) = 2\Delta$ significantly reduces the processing rate required to meet all decoding and display deadlines. Specifically, compared to the Peak CPU Bandwidth $P(\mathbf{a}_i)$ that is required to meet all task deadlines when $d(\mathbf{a}_i) = 0$, the CPU Bandwidth Demand $g(\mathbf{a}_i)$ for $d(\mathbf{a}_i) = 2\Delta$ is lower by ~30% on average. In scenarios where a device's limited processing capacity preclude the admission of one or more tasks with high peak requirements, even small delays can improve the number of admitted tasks. We note that the Delay $d(\mathbf{a}_i)$ can be increased further to achieve greater reduction in the CPU Bandwidth Demand, however, this requires larger memory buffers [5].

3.2. Benefits of Using the CPU Bandwidth Demand

In this subsection, we evaluate the use of the CSPEC's CPU bandwidth demand parameter g, used for admission control, against an existing solution in the literature [1] and also against the mean CPU bandwidth requirement .

Let t_χ be the first deadline of the χ-th GOP, then we say that all frames with their deadlines at $t_\chi + 2n\Delta$, for $n = 0,\ldots,7$ (as in Table 1) belong to the n-th deadline/priority class. Smaller values of n correspond to frames of higher priority because future frames depend on them. In an H.264/AVC based coder, for example, I frames can be classified as having $n = 0$, P frames $n = 1$, and B frames $n = 2$.

Table 2 illustrates the distribution of missed deadlines in a priority class for two cases: first, the CPU bandwidth assigned is statistically determined in order to meet 95% of the video decoding task's deadlines [1]; second, the CPU bandwidth is assigned as the task's Mean Bandwidth requirement ρ. In the latter case, many deadlines are missed across several classes because the mean CPU bandwidth does not guarantee that the instantaneous deadline-to-deadline bandwidth requirements are satisfied. The former case has many less missed deadlines, however, they all occur in the highest priority class which can adversely affect the quality of subsequent frames. The bottom half of Table 2 shows the quality impact of the Mean Bandwidth (labeled "Mean"), 95% deadline (labeled "95%"), and CPU Bandwidth Demand (labeled "Proposed") based resource allocations. Based on the Peak-Signal-to-Noise Ratios in dB (PSNR) in Table 2, it is clear that allocating resources to meet an arbitrary percent (e.g. 95%) of a task's deadlines significantly impacts the PSNR.

In the top half of Table 2 we do not include the case when the CPU bandwidth demand g is assigned to a task, and each priority class's deadlines are increased to $t_\chi + 2n\Delta + d$, because no deadlines are missed.

4. CONCLUSION

In this paper, we have introduced the scalable CSPEC, which characterizes multiple complexity-levels available to a video decoding application. We have shown that, by introducing a small latency when processing highly time-varying multimedia workloads, we are able to meet all application deadlines. Additionally, we are able to smooth the bursty workload, thereby reducing the CPU bandwidth required to decode a sequence.

5. REFERENCES

[1] W. Yuan, K. Nahrstedt, S.V. Adve, D.L. Jones, R.H. Kravets, "GRACE-1: cross-layer adaptation for multimedia quality and battery energy," IEEE Trans. on Mobile Computing, vol. 5, no. 7, pp.799-815, July 2006.

[2] M. Horowitz, A. Joch, F. Kossentini, A. Hallapuro, "H.264/AVC Baseline Profile Decoder Complexity Analysis," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 704-716, July 2003.

[3] J. R. Ohm, M. van der Schaar, and J. W. Woods, "Interframe wavelet coding – Motion picture representation for universal scalability," Signal Processing: Image Communication, vol. 19. no. 9, pp. 877-908, Oct. 2004.

[4] Z. He, Y. Liang, L. Chen, I. Ahmad and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," IEEE Trans. Circuits and Syst. for Video Technol., vol. 15, no. 5, pp. 645-658, May 2005.

[5] S.L. Regunathan, P.A. Chou, J. Ribas-Corbera, "A generalized video complexity verifier for flexible decoding," Proc. 2003 International Conference on Image Processing, 2003. ICIP 2003, vol.3, pp. III- 289-92 vol.2, 14-17 Sept. 2003