GRACEFUL QUALITY DEGRADATION FOR VIDEO DECODING SYSTEMS THROUGH PRIORITY SCHEDULING AND PROCESSOR POWER ADAPTATION

Brian Foo and Mihaela van der Schaar

University of California Los Angeles (UCLA), Dept. of Electrical Engineering (EE), Los Angeles, CA, 90095
Tel: +1-310-825-5843, Fax: +1-310-206-4685, Email: {bkungfoo , mihaela}@ee.ucla.edu

ABSTRACT

Voltage/frequency configurable processors can provide significant energy savings in video decoding systems due to their ability to dynamically adapting the frequency and voltage according to time-varying workloads. In this paper, we propose a joint voltage scaling and priority scheduling algorithm that decodes jobs in order of their importance (quality impact), such that by setting the processor to various power levels and decoding only the most important jobs, different quality and energy tradeoffs can be achieved. We demonstrate that our algorithm performs well in practical decoding scenarios, where reducing the power to 25% of the original power can lead to quality degradations of less than 1.0 dB PSNR.

Index Terms— *Multimedia Systems, Modeling, Complexity Theory, Queuing analysis*

1. INTRODUCTION

Dynamic voltage scaling (DVS) allows a processor to dynamically adjust its operating frequency and voltage to time-varying workloads, which enables the system to optimize energy-delay tradeoffs for tasks where jobs need to be completed by certain deadlines [1]. As a result, DVS is a popular solution for delay-sensitive multimedia applications running on energy-constrained systems [2]. Currently, most DVS algorithms are used in conjunction with earliest deadline first (EDF) job scheduling [2] [3]; however, such scheduling policies may not perform well when the workload is high, or the system is severely energy constrained. In this paper, we propose a quality-adaptive DVS algorithm based on priority scheduling, where jobs are decomposed based on their importance, such that more important jobs are processed first. In this way, the video stream can be decoded at various quality levels even if the system energy is insufficient for decoding all scheduled jobs before their deadlines. Based on the priority-scheduling mechanism, we introduce several DVS algorithms to achieve graceful quality degradation under low system energy.

2. PRIORITY-SCHEDULING QUEUING MODEL

In this section, we consider a motion compensation temporal filtering (MCTF) video coder which decomposes a video sequence into a hierarchy of transform frames based on their dependencies and contribution to the overall video quality. By setting the decoding of a transform frame as a job, the system can organize jobs into different priority classes and use priority scheduling to process jobs. However, in order to analyze the average quality of the video under various processor powers, we first need to introduce a queuing model to determine the probability that jobs of different priority classes will miss their deadlines.

2.1. Modeling Entropy Decoding Complexity as Memoryless "Arrivals"

Consider a buffer that streams jobs (or frames) to the decoder according to a deterministic process which corresponds to the frame rate of the video. Before operations such as inverse transform (IT), motion compensation (MC), and fractional pixel interpolation (FI) can be performed, entropy decoding (ED) must first be used to reconstruct the average and error frames. In this section, we construct a queuing theoretic model for the decoding process by treating ED complexity as an arrival process, and the total complexity associated with each unit of ED complexity as the service time.

In order to determine the complexity of decoding a particular frame, we collected job execution times (offline) from a set of 11 training sequences with 16 GOPs each, decoded at 7 different bit rates. Based on the data, we investigated the complexities contributed by different steps of a decoding process. The total complexity for decoding a class i job, $i = 1,...,I$ in sequence seq is given by C_i^{seq}, where:

$$C_i^{seq} = C_{i,\mathrm{ED}}^{seq} + C_{i,\mathrm{IT}}^{seq} + C_{i,\mathrm{MC}}^{seq} + C_{i,\mathrm{FI}}^{seq} \qquad (1)$$

where each $C_{i,\mathrm{op}}^{seq}$, $op \in \{ED, IT, MC, FI\}$, indicates the complexity associated with one type of decoding step for a job of class i. Note that in some cases, $C_{i,\mathrm{op}}^{seq} = 0$. For example, the complexity of a top level L-frame requires only entropy decoding and inverse transform, while the top level H-frame requires motion compensation to restore the next lower level frames. The entropy decoding complexity, however, exists for

each job and interestingly, can be modeled by a shifted and scaled Poisson distribution (shown in Figure 1):

$$C_{i,ED}^{seq} \cong a_i^{seq} \hat{C}_{i,ED}^{seq} + b_i^{seq} \qquad (2)$$

where the normalized complexity distribution of $\hat{C}_{i,ED}^{seq}$ is:

$$p_{i,ED}^{seq}(n) = \frac{(\nu_i^{seq})^n e^{-\nu_i^{seq}}}{n!} \qquad (3)$$

where n is the Poisson bin number, $p_{i,ED}^{seq}(n)$ is the probability that the normalized complexity falls into bin n, and ν_i^{seq} is the shape parameter for the normalized complexity distribution. Figure 1 shows the normalized ED complexities $\hat{C}_{i,ED}^{seq}$ for various L-frames and H-frames averaged over all training sequences.

Due to the form for ED complexity distribution, we can model ED complexity as a pure Poisson distribution scaled by a constant number of cycles, which we call a "groups of cycles" (GOCs). It is a well-known fact that when ED GOCs "arrive" according to a memoryless process, a Poisson distributed number of ED GOCs will occur in any fixed time period [5]. Since frames arrive periodically according to a fixed frame rate, the ED GOCs form a memoryless arrival process.

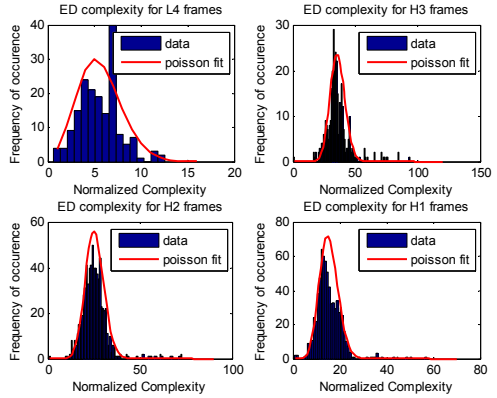

Figure 1: Normalized entropy decoding complexity for various L and H frames in a 4 temporal level MCTF GOP averaged over various training sequences.

2.2. GOC Service Time Modeling

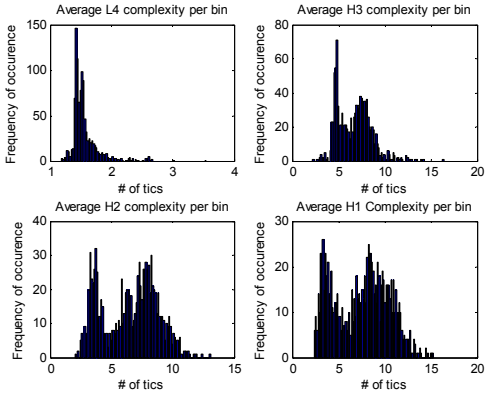

Figure 2: Example of total complexity per arriving ED GOC for various frames in a 4 temporal level MCTF GOP. The statistics are averaged over several sequences.

Since the decoding of each frame consists of more than just entropy decoding, for each arriving GOC, we need to approximate the distribution of the complexity of other steps (e.g. inverse transform or motion compensation) associated with the GOC. We modeled the service rate per GOC, by dividing the total complexity (in tics) associated with the decoding of each frame by the complexity of entropy decoding. Figure 2 shows examples of resulting service complexity distributions.

2.3. Non-Preemptive $M/G/1$ Priority Queuing and Delay Analysis

Based on the decomposition of jobs into arriving ED GOCs, we propose a DVS system that uses priority scheduling to process the incoming GOCs as packets. We model the system as a non-preemptive $M/G/1$ priority queuing system. Priority scheduling ensures that even if not all jobs can be processed before the display deadline, the higher priority jobs will be processed first, so that they are more likely to satisfy their deadline constraints. Effectively, this enables the system to gracefully adapt the quality to different amounts of available energy.

Let $D_{i,k}$ be the delay of processing a GOC of class i, and define $\Pr\{D_{i,k} > T_i\}$ to be the probability that a GOC arriving at time t can not be processed before deadline $t + T_i$. Note that in reality, all GOCs of the same job have the same hard deadline regardless of their arrival times t, so the delay bound T_i would not be fixed for every GOC of a job. However, considering that GOCs of the same class need to be processed in FIFO order to complete the job, the deadlines for the first GOCs in the job may be set earlier to accommodate the processing time delay induced on later GOCs. For the purpose of analysis, we approximate the delays T_i tolerated by all GOCs within the same class to be approximately equal. In order to determine the probability of violating the delay deadline for a non-preemptive priority queuing system, we first define the load on the system induced by priority class i with service time $S_{i,k}$ as:

$$\rho_{i,k} = \lambda_i E[S_{i,k}] \qquad (4)$$

Let $\sigma_{i,k} = \sum_{j=1}^{i} \rho_{j,k}$ be the total load of traffic coming from priority classes 1 to i, and let $\mu_{i,k}$ be the average service rate for a class i job in processor operating mode k. The average waiting time in the queue for priority class i GOCs can then be expressed as [8]:

$$\mathrm{E}[W_{i,k}] = \frac{1}{2(1-\sigma_{i-1,k})(1-\sigma_{i,k})} \cdot \sum_{j=1}^{I} \frac{\rho_{j,k}}{\mu_{j,k}} \qquad (5)$$

From the average waiting time, we can obtain an approximation for the probability that the waiting time exceeds some time t. We use the waiting time tail approximation to estimate the tail of the delay:

$$\begin{aligned} \Pr\{D_{i,k} > T_i\} &= \Pr\{W_{i,k} + S_{i,k} > T_i\} \\ &\approx \rho_k \exp\left(-\frac{\rho_k T_i}{\mathrm{E}[W_{i,k}] + E[S_{i,k}]}\right) \end{aligned} \qquad (6)$$

Note finally that the fraction of busy time in an $M/G/1$

queuing system is $\sigma_{I,k}$.

3. DVS ALGORITHMS FOR GRACEFUL QUALITY DEGRADATION

In this section, rather than servicing jobs according to their deadlines, we service jobs based on priority levels, such that a lower quality level can be achieved even if not all frames can be decoded. (See Figure 3 for an example concerning 3 temporal level MCTF.) Based on this decomposition, we formulate and analyze a number of DVS optimization problems based on probabilistic delay constraints. We begin with a simple optimization problem, where a processor determines different fractions of time α_k to operate at different power levels P_k, $k = 1,...,K$.

Optimization Problem 1: Minimize the Average Active Power given an Average Video Quality

$$\begin{aligned} &\min_{\alpha=(\alpha_1,...,\alpha_K)} \sum_{k=1}^{K} \alpha_k P_k \\ &s.t. \sum_{k=1}^{K} \alpha_k Q_k \geq Q_{\text{avg}} \\ &\sum_{k=1}^{K} \alpha_k = 1 \end{aligned} \tag{7}$$

where:

$$Q_k = \sum_{i=1}^{I} \frac{\lambda_i}{\lambda} \Delta_i \Pr\{D_{i,k} \leq t_i\} \tag{8}$$

is the average quality of the decoded sequence at power level P_k. Here, α is a vector with components that are the fraction of time the processor is set to operate at power level P_k, and Δ_i is the quality slope parameter for priority i GOCs (i.e. the average quality contributed to video by a priority i GOC.) as introduced in [4]. Note that λ_i / λ is the fraction of GOCs of priority i received from the bitstream. Thus, the first constraint requires that the average quality of the video is at least Q_{avg}. This problem turns out to be a linear programming problem, since P_k and Q_k are constants. We can thus solve this via the simplex method. However, an even simpler closed-form solution exists if we explicitly consider the properties of power with respect to quality.

Proposition 1: *If quality is a concave increasing function of ED complexity, and there are a finite number of power/frequency levels, the optimal solution to Optimization Problem 1 is to run the processor always at a single power level, or to perform time sharing between two adjacent power levels.*

Proof: Let $\hat{Q}$ be a discrete random variable which takes on quality levels Q_k with probability α_k. Since power is a convex function of frequency [7] and complexity (and thus the processor frequency) is a convex function of quality [6], power is a convex function of the required average quality. For a convex quality to power function $P(q)$, the distribution of $\hat{Q}$ with $E[\hat{Q}] = Q_{\text{avg}}$ that minimizes the expected value of the function $E[P(\hat{Q})]$ is $\hat{Q} = Q_{\text{avg}}$ with probability 1 if $P(Q_{\text{avg}}) \in \{P_1,...,P_K\}$, or else:

$$\hat{Q} = \begin{cases} Q_{k^*} & \text{with prob. } \frac{Q_{\text{avg}} - Q_{k^*}}{Q_{k^*+1} - Q_{k^*}} \\ Q_{k^*+1} & \text{with prob. } \frac{Q_{k^*+1} - Q_{\text{avg}}}{Q_{k^*+1} - Q_{k^*}} \end{cases}, \tag{9}$$

where $Q_{k^*} < Q_{\text{avg}} < Q_{k^*+1}$. $\hat{Q}$ then minimizes $E[P(\hat{Q})]$, which gives us the solutions α_k to Optimization Problem 1. ■

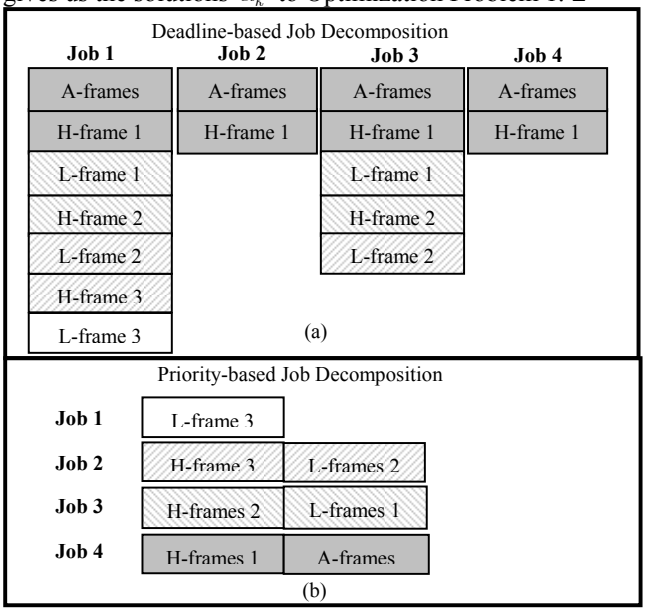

Figure 3: (a) Deadline-based job decomposition and (b) Priority-based job decomposition for 3 temporal level MCTF. The temporal level is indicated by the number beside the frame type.

If we now consider the case where the processor may shut down during idle times and expend essentially zero energy, we have a different optimization problem.

Optimization Problem 2: Minimize the Average Power given an Average Video Quality

$$\begin{aligned} &\min_{\alpha=(\alpha_1,...,\alpha_K)} \sum_{k=1}^{K} \alpha_k \rho_k P_k \\ &s.t. \sum_{k=1}^{K} \alpha_k Q_k \geq Q_{\text{avg}} \\ &\sum_{k=1}^{K} \alpha_k = 1 \end{aligned} \tag{10}$$

This problem is no longer convex. However, given that the optimal mode of operation should keep the system nonempty with high probability, the processor power should hover between at most a few power levels. If the solution is to run the processor at a nearly constant power level, we can determine a near optimal fixed power under complexity $O(K \cdot I)$ given an average desired video quality.

Optimization Problem 3: Choose a minimum fixed power

$$\begin{aligned} &\min P_k \\ &s.t. Q_k \geq Q_{\text{avg}} \end{aligned} \tag{11}$$

We now propose several simple priority scheduling and power

scheduling algorithms for DVS. The first algorithm chooses a constant power based on the arrival rate and service time statistics by solving Optimization Problem **3** with various levels of Q_{avg}. The second algorithm is the same as the first, but periodically purges the queue of expired jobs, thereby reducing the average waiting time for different classes. Finally, we present a combined DVS and job scheduling algorithm using priority scheduling with queue purging along with a last second power increase. Whenever a job in a class i is within δ seconds of being expired, the system will increase the processor power according to the job's priority by some $\Upsilon(i)$, thereby increasing the chance of that job being decoded on time.

Algorithm 1: Priority scheduling with last second power increase

```
1.  Solve Optimization Problem 3 for Q_avg, P_init.
2.  While jobs are available,
3.    For the highest priority class i,
      such that the deadline of a job in class i
                will expire in less than δ time
4.          Set P = P_init + Υ(i).
5.    end
6.    Process highest priority job in FIFO
            order. Record service time s.
7.    Subtract deadline of all other jobs by s.
8.    If deadline of a job j is less than 0,
              then purge job j.
9.  end
```

4. SIMULATIONS AND RESULTS

Table 1: Comparisons of performances of various priority scheduling algorithms in terms of the percentage of deadlines missed for various priority classes for 4 temporal level decomposition (f_0 indicates the minimum processor power.).

Jobs decoded	Frequency Levels	f_0	$2f_0$	$3f_0$	$4f_0$	$5f_0$	$6f_0$
Priority	class 1	99.7	100	100	100	100	100
	class 2	67.3	99.9	100	100	100	100
	class 3	0	98.4	99.9	100	100	100
	class 4	0	0	0	99.0	99.9	100
	class 5	0	0	0	0	0	0.01
Priority with Queue Purging	class 1	99.6	100	100	100	100	100
	class 2	68.1	99.9	100	100	100	100
	class 3	13.5	99.7	100	100	100	100
	class 4	0	14.3	57.5	99.6	99.9	100
	class 5	0	0	6.46	26.9	41.2	66.8
Algorithm 1	class 1	99.9	100	100	100	100	100
	class 2	91.2	99.9	100	100	100	100
	class 3	31.9	99.4	100	100	100	100
	class 4	0	14.0	58.4	99.6	99.9	100
	class 5	0	0	6.63	26.9	41.2	66.8

Based on various average power levels for the processor, we compared the probability of dropping jobs of different classes based on the strict priority scheduling policy, a priority scheduling policy with queue-purging of expired jobs, and Algorithm 1. Table 1 includes averaged results from many sequences encoded by 4 temporal level MCTF based on different processor operating frequencies. For Algorithm 1, we used $\Upsilon(1) = 1.5\Upsilon(2) = 3\Upsilon(3) = 3f_0$ for the first 3 priority classes. Table 2 compared the frame rates, energies, and quality levels achieved under different energy constraints, where E denotes a normalized unit of energy consumed. The results show that there is only a loss in quality of about 1.0 dB when the power is scaled down by 75%, which demonstrates that our algorithm gracefully adapts the quality to varying energy consumption.

Table 2: Comparisons of quality-energy adaptation points achieved by algorithm 3 for the *Coastguard* and *Stefan* sequences.

Alg	**Frame rate (fps):**		**Energy consumed:**		**PSNR (dB):**	
Seq	*Cstgrd*	*Stefan*	*Cstgrd*	*Stefan*	*Cstgrd*	*Stefan*
EDF	30.00	30.00	**2.63E**	**2.41E**	**33.24**	**27.35**
1	26.48	23.67	2.15E	2.15E	32.98	27.01
1	20.04	18.05	1.26E	1.25E	32.51	26.70
1	16.17	15.23	**0.65E**	**0.65E**	**32.23**	**26.48**
1	14.53	10.08	0.28E	0.29E	32.05	25.94

5. CONCLUSIONS

In this paper, we proposed an adaptive architecture combining both power and job scheduling to obtain scalable energy-quality tradeoffs. Our results indicated that priority-scheduling based DVS algorithms can save a significant amount of energy with only a small reduction to the quality level. This work may be extended to multiple tasks or multiple processor environments for future research.

6. REFERENCES

[1] L. Benini, G. De Micheli. *Dynamic Power Management: Design Techniques and CAD Tools.* Kluwer Academic Publishers, Norwell, MA, 1997.

[2] W, Yuan, K. Nahrstedt, S. Adve, D. Jones, R. Kravets. "GRACE: Cross-layer Adaptation for Multimedia Quality and Battery Energy," *IEEE Transactions on Mobile Computing*, 2006.

[3] A. Reddy, J. Wyllie, K. Wijayaratne. "Disk scheduling in a multimedia I/O system," *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2005.

[4] A. Ortega, K. Ramchandran. "Rate-distortion Methods for Image and Video Compression," *IEEE Signal Processing Mag.*, vol. 15, issue 6, Nov, 1998.

[5] R.G. Gallager, *Discrete Stochastic Processes*, Kluwer, Dordrecht, 1996.

[6] B. Foo, Y. Andreopoulos, M. van der Schaar. "Analytical Complexity Modeling of Wavelet-based Video Coders." ICASSP '07, to appear.

[7] T. Ishihara, H. Yasuura. "Voltage Scheduling Problem for Dynamically VariableVoltage Processors," in Proc. ACM ISLPED, 1998, pp. 197-202.

[8] D. Gross and C. Harris, Fundamentals of Queueing Theory, New York: Wiley-Interscience, 1997.