Predictive Thermal Control for Real-Time Video Decoding*

Mehmet H. Suzer
Harran University
msuzer@harran.edu.tr

Kyoung-Don Kang
Binghamton University
kang@binghamton.edu

Abstract

Multimedia applications with soft real-time constraints consume considerable power, incurring thermal problems. It is challenging to process multimedia data in real-time with the highest possible QoS, while avoiding potential thermal problems. To aggravate the problem, immediate dissipation of the accumulated heat is infeasible. In this paper, we design an empirical model to predict the power consumption and resulting processor temperature for video decoding. Based on the model, we develop a predictive method that avoids deadline misses due to thermal faults by adapting the video quality before the temperature exceeds the specified threshold. We implement our predictive control scheme and two baseline approaches by extending an open source implementation of the H.264 scalable video coding standard. Our approach controls the temperature to be below the specified threshold for most of the time unlike the tested baselines, while supporting similar or even better video quality.

1 Introduction

Important soft real-time applications, such as visual surveillance, traffic control, or gaming, need to deal with computationally demanding multimedia workloads, such as high definition (HD) video frames. As a result, the processor is subject to high power consumption and overheating when it runs a soft real-time multimedia application. If the CPU gets overheated, many multimedia deadlines (e.g., playtime deadlines of an HD video stream) can be missed due to hardware-triggered processor speed reductions, such as clock throttling for thermal control available in modern processors unaware of real-time constraints.

In this paper, we define thermal faults as any overheating events in which the CPU chip temperature exceeds the specified threshold, incurring potential deadline misses due to hardware-triggered CPU speed reductions. Given the definition, we investigate how to avoid CPU thermal faults subject to deadline misses in a predictive fashion, while supporting the highest possible HD video quality.

To handle thermal issues in real-time systems, reactive thermal management techniques, including [25, 3, 14, 9], have been developed. In reactive methods, the processor speed is usually reduced via, for example, dynamic voltage and frequency scaling (DVFS) once the temperature reaches or exceeds the threshold. Unfortunately, in reactive methods, users may suffer from many deadline misses and resulting quality of service (QoS) degradation due to the reduced processor speed. If the multimedia quality is adapted to match the decreased processor speed in a reactive manner, users may experience poor QoS for a long interval of time, since the accumulated heat cannot be dissipated immediately. Recently, QoS adaptation techniques have been applied to decrease the energy consumption of multimedia applications [15, 12]. However, relatively little work has been done to support predictive CPU temperature control that systematically adapts the HD video quality by a minimal degree to prevent thermal faults potentially causing deadline misses rather than relying on ad hoc approaches to QoS adaptation.

It is challenging to prevent the CPU from overheating and simultaneously support as high video quality as possible under the temperature constraint because of the high computational demand and dynamic workloads in real-time multimedia applications. For example, the complexity of the scenes in an HD video stream may vary from frame to frame and may not be known a priori. To address the challenge, we propose a systematic and cost-effective approach in this paper:

- We design a *new empirical model* that predicts the CPU temperature by directly considering the CPU thermal characteristics and multimedia application semantics.
- We periodically *update the model* at each control

*This work was supported, in part, by NSF grant CNS-117352.

point to capture the potentially time-varying relation between the provided video quality and the predicted CPU power consumption and temperature. Thus, our thermal control scheme is an adaptive feedforward controller consisting of both feedforward and feedback components.

- Using the predictive model, we adapt the video quality by a *minimal degree* within a specified range at each control point, if necessary, to avoid overheating the CPU in the next sampling period.

Via predictive video quality adaptation at each control point, our approach intends to avoid violating the specified temperature threshold in a predictive fashion, while supporting as high overall quality as possible under the temperature constraint.

Our approach can be considered a mirror image of feedback-based approaches in that we take an action at a control point, if necessary, to prevent overheat over the prediction horizon rather than reacting to a thermal error observed in the previous control period.[1] The notions of the online system modeling, predictive control, and minimal QoS adaptation of our approach are conceptually similar to model predictive control [16]. However, in model predictive control, the controlled system behavior is usually modeled via online black-box modeling based on the recursive least square method. In this paper, we instead perform white-box modeling that directly considers real-time multimedia workload and CPU thermal characteristics. More specifically, we derive and update our predictive model online based on the inherent relationship between the video quality and the resulting CPU cycle and power consumption, which result in temperature changes, rather than relying on blackbox modeling.

To manage the CPU temperature via predictive video quality adaptation, we use H.264 scalable video coding (H.264/SVC) jointly standardized by ISO/IEC MPEG and ITU-T VCEG [21]. Notably, the term "scalability" in SVC means the potential partial removal of the video bit stream for QoS adaptation. SVC supports temporal, spatial, and quality scaling that adapt the frame rate, spatial resolution, and quantization parameter (QP) affecting the peak signal to noise ratio (PSNR), respectively. In this paper, we adapt the video quality, if necessary, to avoid overheating the CPU in the next control period via quality scaling, which normally provides less disturbing visual effects than spatial or temporal scaling does [21]. Thus, the provided QP for HD video decoding is the QoS metric used in this paper.

In this paper, we extend the JSVM (Joint Scalable Video Model) software [13], which is a reference implementation of the H.264/SVC standard [21], to implement our predictive CPU temperature management scheme and two baselines: a static approach and a reactive feedback controller that uses a fixed QP and dynamically adapts the QP based on formal control theory [17], respectively. We compare the CPU temperature and supported video quality of our approach to those of the baselines. The experimental results show that the temperature of the feedback-based baseline, which shows considerably better performance than the static baseline does, exceeds the threshold by 5°C − 9.4°C (9% − 17%) for 270 or more seconds out of the 10 minute experiment for the tested videos. In our approach, the temperature exceeds the threshold by no more than 0.4°C (0.007%). Also, a temperature overshoot exceeding the threshold has lasted for no more than 11s out of 10 minutes. The video quality supported by our approach is similar to or slightly better than the quality provided by the baselines for most of the time. Since our approach dynamically adapts the video quality in a predictive manner, it can avoid overheating the CPU. As a result, it can also avoid potentially severe QoS degradation due to the overheat for an extended time interval. In our user experience study, none of the participating users has noticed visible quality degradation due to potential QoS adaptation via our predictive thermal control scheme. Further, our approach is lightweight. According to our measurements, it consumes less than 3% CPU utilization.

The rest of the paper is organized as follows. Background information about H.264/SVC is given in Section 2. Our predictive temperature management scheme is described in Section 3. Section 4 presents the performance evaluation results. Related work is discussed in Section 5. Finally, Section 6 concludes the paper and discusses future work.

2 Overview of the H.264/SVC Standard

In this section, an overview of H.264/SVC that extends H.264 advanced video coding (H.264/AVC), which is one of the most common formats used to record, compress, and distribute HD videos, to support scalable video coding.

Traditional digital video coding, transmission, and storage techniques, e.g., standard-definition television (SDTV), high-definition television (HDTV), or common intermediate format (CIF), usually support a fixed

[1] In our approach, the length of the prediction horizon is 1 in that we adapt the video quality, if necessary, to prevent thermal faults in the next control period, similar to one-step look-ahead control [1].

spatial resolution and frame rate. However, fixed format videos cannot efficiently deal with diverse network connections and devices used by end-users with considerably different computational and display capabilities [21].

Figure 1. Quality scaling: Frames decoded using QP=16, 28, 36, and 40 are shown from top-left to bottom-right (Best seen in colors)

The H.264/SVC standard [21] is designed to substantially enhance the coding efficiency, while increasing the video scalability compared to the previous scalable video coding standards. In H.264/SVC, a raw video is compressed only once to get a high quality video stream that contains one or more subset bit-streams. One video can be coded with a combination of different temporal rates, spatial resolutions, and quantization parameters (QPs) at the coding time. Therefore, multiple bit-streams of the same source content, which differ in the coded video size, frame rate, and/or PSNR, can be serviced at the same time. Also, a decoder of a client with constrained resources can dynamically adapt the QoS by extracting and decoding only selected partial bit-streams from the compressed bit-stream, discarding the other unselected sub-bit-streams that improve the quality of the video. In this paper, we take advantage of selective video decoding for predictive CPU temperature control.

Specifically, H.264/SVC supports three types of scaling.

- In *spatial scaling*, a video is coded with several spatial resolutions. The original high resolution video is down-converted to new video streams with lower resolutions. The final bit-stream contains the video with all the coded resolutions. Thus, a compressed video can be decoded at different resolutions.
- In *temporal scaling*, all or a part of the frames of the original video is coded with different frame rates. Hence, a compressed video can be transmitted or decoded at different frame rates.
- In *quality scaling*, a substream provides the same spatial and temporal resolution as the complete bit-stream does, but supports lower fidelity. The fidelity of an image created by a lossy codec is usually represented by the PSNR, which indicates how close a reconstructed image is to the original (uncompressed) image [18]. Thus, a higher PSNR value normally indicates a better reconstruction of an image. A QP is used to selectively cancel some information from the original video. Its effect is similar to the effect of a low-pass filter [18, 21]. As human eyes are more sensitive to low frequencies than high frequencies, canceling high contrasts can be done progressively with a moderate impact on the visual perception. Generally, a smaller QP leads to higher fidelity and PSNR at the cost of a higher bit rate.

As shown in Figure 1, adapting the QP usually results in tolerable visual quality degradation. Although the QP is substantially increased from 16 to 40, the visible quality degradation is not pronounced. Usually, quality scaling leads to less noticeable quality degradation compared to spatial and temporal scaling that decreases the video resolution/size and induces choppy video playback due to the skipped frames, respectively [21]. Hence, we use quality scaling as the QoS adaption mechanism for thermal control with minimal QoS degradation. However, our approach is not limited to quality scaling. If required, it can be easily adapted to support temporal or spatial scaling for video decoding.

In H.264/SVC, each layer of a frame (i.e., a picture) is decoded in the following sequence [21]:

1. Initialize the slices (spatially distinct regions of a frame encoded separately from the other regions in the frame) and decoding parameters.
2. Parse the slices, analyze the bit-stream, and decode the entropy.
3. Decode the slices to reconstruct the picture.
4. Do an optional final processing using a loop filter and subsequently display a reconstructed picture.

In this paper, we use P_c to denote the control period. For $k \geq 0$, the k^{th} predictive control point is equal to the time instant kP_c. At the k^{th} control point, we estimate the decoding cost of the H.264/SVC video frames that will be decoded in the $(k+1)^{th}$ control period, which is the time interval $[kP_c, (k+1)P_c)$, by performing only the first two steps to parse the slice data (also called metadata) of the frames without actually decoding them, since the last two steps dominate the decoding cost. Specifically, to design a lightweight

approach to predictive thermal control for HD video decoding, we only extract the QPs used to encode the frames that will be decoded in the next control period at each control point.

3 CPU Temperature Modeling and Control

In this section, the overall system structure and our predictive temperature control scheme are described.

3.1 Overall System Architecture

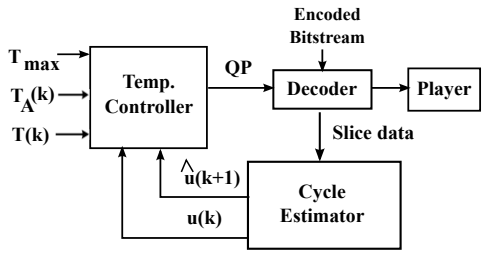

Figure 2. System Architecture

Figure 2 shows the overall system architecture. In our approach, the *controlled variable* is the CPU temperature. T_{max} in Figure 2 represents the specified CPU temperature threshold. In addition, $T(k)$ and $T_A(k)$ represent the current chip temperature and ambient temperature measured at the k^{th} control point, respectively.

At the k^{th} (≥ 1) control point, the cycle estimator in Figure 2 measures the number of the CPU cycles that have been consumed for video decoding in the k^{th} control period, $u(k)$. Also, it predicts the estimated CPU cycles required for video decoding in the $(k+1)^{th}$ control period, $\hat{u}(k+1)$.

In this paper, the QP is the *manipulated variable* adapted by our temperature controller, if necessary, to control the CPU temperature to be below T_{max}. Based on $\hat{u}(k+1)$ and the corresponding estimated power consumption, the smallest possible QP, $QP(k+1)$, is derived by the temperature controller. $QP(k+1)$ will be used to decode the frames in the $(k+1)^{th}$ control period with the highest possible quality under the temperature constraint, T_{max}.

The decoder in Figure 2 takes a coded SVC bit stream and decodes it according to $QP(k+1)$ in the $(k+1)^{th}$ control period. When the QP is increased, the PSNR normally decreases and the quality of the picture, which is reconstructed by decoding the video frame coded via lossy compression, decreases. Finally, the player displays the decoded frames.

Further, we assume that all deadlines for video decoding can be met using a uniprocessor real-time scheduling algorithm, if the CPU speed is not decreased by the hardware due to thermal faults. Thus, we focus on preventing thermal faults that can result in deadline misses.

3.2 Design of the Predictive Model

Our predictive temperature control scheme is summarized in the pseudo code in Algorithm 1 and discussed in this subsection.

Algorithm 1 Predictive CPU Temperature Control

1. At the k^{th} control point, update the model parameters.
2. At the k^{th} control point, estimate the maximal allowable power consumption, $p_{max}(k+1)$, in the $(k+1)^{th}$ control period without exceeding T_{max}.
3. At the k^{th} control point, based on $p_{max}(k+1)$, compute the smallest possible $QP(k+1)$ expected to support the highest possible quality, while avoiding to violate T_{max} in the $(k+1)^{th}$ control period.
4. Use $QP(k+1)$ for video decoding in the $(k+1)^{th}$ control period.
5. Repeat the steps above at each control point until all frames are decoded.

To design a predictive scheme for CPU temperature management, we use the widely accepted RC temperature model [23, 6] that captures the relation between the CPU temperature and power consumption in the continuous time domain as follows:

$$\frac{dT(t)}{dt} = -\frac{1}{RC}\left[T(t) - T_A(t)\right] + \frac{1}{C}p(t) \qquad (1)$$

where R and C represent the thermal resistance and capacitance values of the CPU chip. $T(t)$, $T_A(t)$, and $p(t)$ are the current chip temperature, ambient temperature, and the amount of the power consumed by the CPU at time t, respectively. In this paper, we assume that the ambient temperature does not change significantly between two sampling points. Our model can be easily extended to consider the ambient temperature for thermal control, if it can be read using, for example, an off-chip sensor.

We discretize the continuous time domain model in Eq. 1 to predict the CPU temperature in the $(k+1)^{th}$ control period, $x(k+1)$, at the k^{th} control point, similar

to [23, 6]:

$$x(k+1) = Ax(k) + B\hat{p}(k+1) \quad (2)$$

where $A = e^{-P_c/RC}$, $B = (1-A)R$, and $x(k) = T(k) - T_A(k)$. In addition, $\hat{p}(k+1)$ represents the predicted power consumption for video decoding in the $(k+1)^{th}$ control period.

Using the GNU intelligent platform management interface (IPMI) tools [8], we measure $T(k)$ and $T_A(k)$ in Eq. 1 by taking on-chip temperature sensor readings at the k^{th} control point. Note that we build our predictive thermal control model based on direct sensor readings rather than using indirect indications of the CPU temperature, such as the fan speed adjusted by the hardware based on the CPU temperature sensor readings. By doing this, we not only decrease the modeling complexity but also directly capture real-time thermal behaviors of the CPU.[2]

At the k^{th} control point, we estimate $\hat{p}(k+1)$:

$$\hat{p}(k+1) = P_{Idle} + P_f(k)\hat{u}(k+1) \quad (3)$$

where P_{Idle} represents the idle power consumption. In this paper, P_{Idle} is measured offline when the CPU is idle. $P_f(k)$ in Eq. 3, called the power factor, is the gain that captures the relation between the number of the CPU cycles and the amount of the power consumed for video decoding at the k^{th} control point. Although Eq. 3 assumes a linear relation between the CPU cycle consumption, i.e., the number of the CPU cycles consumed for video decoding, and power consumption in a *single control period*, our predictive model is *not* tied to the linear assumption, because the power factor is continuously updated at every control point based on the RC thermal model. Thus, our piecewise linear approximation from control period to period can closely model the relation between the CPU cycle consumption and power consumption even when their relation is nonlinear over an extended time interval.

For $k \geq 1$, we derive $P_f(k)$ from Eq. 2, Eq. 3, the measured CPU cycle consumption, and the measured temperature value via some algebraic manipulations:

$$P_f(k) = \frac{T(k) - T(k-1) + A\left[T(k-2) - T(k-1)\right]}{B\left[u(k) - u(k-1)\right]} \quad (4)$$

where we measure the number of the CPU cycles used for video decoding in the k^{th} control period, $u(k)$, by reading the time stamp counter (TSC) that is a hardware counter readable through the IPMI at the k^{th} control point as follows:

$$u(k) = TSC(k) - TSC(k-1) \quad (5)$$

Therefore, $\hat{p}(k+1)$ can be derived, if $\hat{u}(k+1)$ in Eq. 3 is known.

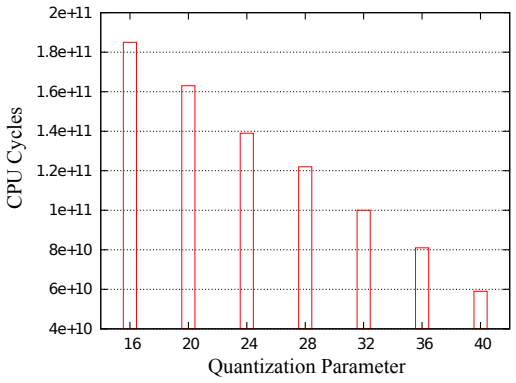

Figure 3. Number of the CPU cycles consumed for different quantization parameters

Chen et al. [5] have identified the linear relationship between the quality increment and bit rate of coded videos through a thorough mathematical analysis and extensive experiments. In addition, we have empirically verified the linearity for several videos heavily used in multimedia research [22, 24]. For example, Figure 3 shows the total number of the CPU cycles consumed to decode 600 frames of a test movie Harbor [24] with the 704×576 resolution and 60 frames per second (fps) frame rate for equidistant QPs in JSVM. As shown in Figure 3, the computational load increases almost linearly as the QP decreases and the video quality enhances consequently. Note that the linear relation between the QP and CPU cycle consumption is not limited to the tested videos but generally applicable to videos encoded following the popular H.264/SVC standard [5]. This is because the substreams in H.264/SVC have the same coding mechanism and configuration as the original bit stream of a coded video, decoding more substreams for quality enhancement increases the CPU cycle consumption in a linear fashion [5].

By letting $QP(k)$ represent the QP used in the k^{th} control period, we compute the utilization factor based on the linear relation between the QP and the CPU cycle consumption for video decoding at the k^{th} control point:

$$U_f(k) = \frac{u(k)}{QP(k)} \quad (6)$$

where $U_f(0)$ is calculated offline by decoding a few frames. Note that the utilization factor is also updated at every control point to closely keep track the relation

[2]Considering indirect/secondary hints of the CPU temperature too might enhance the accuracy of the model for additional overheads. A thorough investigation is reserved for future work.

between the QP and CPU cycle consumption that may vary in time.

Based on $U_f(k)$, we predict the estimated number of the CPU cycles that will be needed for video decoding in the $(k+1)^{th}$ control period at the k^{th} control point:

$$\hat{u}(k+1) = U_f(k) \cdot QP(k+1) \tag{7}$$

where the manipulated variable, $QP(k+1)$, is the smallest possible QP expected to not overheat the CPU over T_{max} in the $(k+1)^{th}$ control period.

To compute $QP(k+1)$, we first substitute $x(k+1)$ in Eq. 2 with $T_{max} - T_A(k)$, because the temperature is desired to converge to T_{max} to support the highest possible video quality under the temperature constraint in the $(k+1)^{th}$ control period. Also, we replace $\hat{p}(k+1)$ in Eq. 2 with $p_{max}(k+1)$, the estimated highest possible power consumption under the thermal constraint T_{max} in the $(k+1)^{th}$ control period, to derive the following equation:

$$p_{max}(k+1) = B^{-1}\left[T_{max} - T_A(k) - A(T(k) - T_A(k))\right] \tag{8}$$

Further, we substitute $\hat{p}(k+1)$ and $\hat{u}(k+1)$ in Eq. 3 with $p_{max}(k+1)$ and $u_{max}(k+1)$, respectively. We then compute the estimated CPU cycle consumption required to support $QP(k+1)$ in the $(k+1)^{th}$ control period:

$$u_{max}(k+1) = \frac{p_{max}(k+1) - P_{Idle}}{P_f(k)} \tag{9}$$

To derive the manipulated variable, $QP(k+1)$, at the k^{th} control point, we replace $\hat{u}(k+1)$ in Eq. 7 with $u_{max}(k+1)$ to get the following equation:

$$QP(k+1) = \left\lceil \frac{u_{max}(k+1)}{U_f(k)} \right\rceil \tag{10}$$

After that, we ensure that $QP_{min} \leq QP(k+1) \leq QP_{max}$ where QP_{min} and QP_{max} are the minimum QP and maximum QP used to encode the frames to be decoded in the $(k+1)^{th}$ control period, respectively:

$$QP(k+1) = \begin{cases} QP_{min} & \text{if } QP(k+1) < QP_{min} \\ QP_{max} & \text{if } QP(k+1) > QP_{max} \\ QP(k+1) & \text{otherwise} \end{cases} \tag{11}$$

As quantization parameters are discrete, $QP(k+1)$ may not be equal to any QP available in the specified range $[QP_{min}, QP_{max}]$. In such a case, the QP that will be actually used for video decoding in the $(k+1)^{th}$ control period is set to the smallest QP among the QPs in the range that are larger than $QP(k+1)$.

Until every frame is decoded, our approach to CPU temperature control via predictive quality adaptation is repeatedly executed at each control point to prevent violating T_{max} in the next control period, while supporting as high video quality as possible.

4 Performance Evaluation

In this section, the system settings for our performance evaluation and the performance results are discussed.

4.1 System Settings

To evaluate the performance of our predictive temperature control scheme, we have built a micro-testbed using a Linux laptop with the 1.6 GHz Intel Pentium M processor and 512 MB of RAM. In our experiments, we decode and play one video at a time, since the laptop does not have enough resources to play two HD videos at the same time. For scheduling, we have used `SCHED_FIFO`. As video frames are decoded in sequence and their absolute deadlines are assigned accordingly, this approach is equivalent to EDF scheduling. One experimental run is 10 minutes long. After finishing an experimental run, we wait for the laptop to completely cool down before doing another experiment.

T_{max}	75 °C (55 °C for experiments)
T_{Amb}	27 °C
P_{Idle}	10.28 W
P_c	$1/r$ (1/frame rate)

Table 1. Thermal model parameters

We have extended JSVM [13] to support video quality adaptation using our predictive model. We read the TSC and probe the on-chip temperature sensors using the IPMI tools [8]. In this paper, we set the control period $P_c = 1/r$ where r is the frame rate of the video being decoded. As the QP can only be changed at the beginning of decoding a new frame, this is the shortest control period that we can choose to manage the CPU temperature for HD content decoding. The acceptable range of the QP, $[QP_{min}, QP_{max}]$, is set to [12, 44] where the difference between the two adjacent QPs is 4. As none of the 10 users participated in our user experience test noticed any visible quality degradation due to our thermal control scheme, we focus on the QP to describe the experimental results in terms of the video quality.

Table 1 shows the thermal model parameters of the processor used for our experiments. Although the threshold temperature T_{max} supported by the hardware is 75°C, we set $T_{max} = 55$°C in our experiments

to avoid any system instability issues or (hardware-triggered) processor speed reduction due to overheat, incurring playtime deadline misses. The table also shows the average ambient temperature T_{Amb} and the idle power consumption P_{Idle} of the laptop.

Our approach to predictive thermal control is lightweight. According to our measurement through the Linux `/proc` interface, it consumes less than 3% CPU utilization and much less than 1KB memory, because it only uses a compact set of mathematical equations composed of basic arithmetic operations.

4.2 Baselines

For performance comparisons, we implement two baselines: (1) a static approach and (2) a reactive feedback controller. Hardware-based approaches, such as DVFS or clock gating, are not considered as baselines, because solely decreasing the CPU speed or temporarily turning off a core without decreasing the amount of the bit-stream via video quality adaptation may result in many playtime deadline misses and severe QoS degradation. In the static open-loop approach, the QP is calculated offline using a set of training video frames and not adapted at run-time. For feedback control, we implement a PI controller−a variant of a popular PID (Proportional, Integral, and Differential) controller−to dynamically adapt the QP considering the temperature error, i.e., the difference between the measured temperature of the CPU and the specified temperature threshold, called the set-point, in control theory [17]. A P controller computes the control signal in proportion to the error. As a P controller by itself cannot cancel a steady state error of a feedback control system [17], we also use an I controller. We do not use a D controller, similar to a feedback-based reactive thermal control method [9].

We formulate the PI controller as an efficient form of a transfer function that characterizes the behavior of the feedback controller in the frequency domain [17]:

$$C(z) = \frac{G_1(z - G_2)}{z - 1} \tag{12}$$

In Eq. 12, $G_1 = K_P(K_I + 1)$ and $G_2 = \frac{1}{K_I+1}$ where K_P and K_I are the proportional and integral control gains. To support the stability of the closed-loop system, we tune K_P and K_I using the Root Locus method [17] in MATLAB to graphically locate the closed-loop poles inside the unit circle to support the stability of the closed-loop system. For more details about a PI controller design and tuning via the Root Locus method, readers are referred to [17]. Generally speaking, stability analysis of a predictive control system is an open issue [16]. In this paper, we strive to closely keep track the CPU thermal characteristics by continuously updating the model parameters, while leveraging multimedia workload semantics. A thorough analysis of the stability of our predictive control scheme is reserved for future work.

4.3 Experimental Results

In the static approach, we have tried to find an appropriate QP value to support acceptable video quality without overheating the CPU using a training set of SVC video frames. Unfortunately, this attempt was unsuccessful. As different videos may have largely different computational requirements, finding a single QP that can both avoid overheating the CPU and support acceptable quality for different videos is very hard, if at all possible. Even for a single video, we have found that one QP value is not enough to support both the temperature constraint and highest possible quality under the constraint due to the relatively intensive and dynamic workload that may change from scene to scene. Although we have manually tried various QP values, the tested QP values have resulted in either too low quality or excessive CPU temperature in the static approach. In general, the static approach has failed to balance between the temperature constraint and video quality. From these results, we observe that it is required to dynamically adapt the video quality, if necessary, to support the temperature constraint, while providing as high video quality as possible. Thus, we only present the performance results of the reactive feedback control scheme and our predictive approach.

In the rest of this section, we first present the experimental results for a sample movie, called Elephant Dream [22], which supports 1280×720 resolution at 24fps and has 15691 frames in total, in detail. We then present the experimental results for six HD videos in Table 2 [22, 24] frequently used to evaluate the performance of multimedia codecs. In our experiments, for the reactive PI controller, the QP is initialized as 24 that is at the end of the lower half of the QP range used in our experiments. Our predictive scheme derives the highest possible QP at the beginning of an experiment based on the slice data of the first frame and initial model parameters derived offline, e.g., P_{Idle} and $U_f(0)$, as described in Section 3. In addition, we note that the required offline modeling in our predictive thermal control scheme is relatively minimal compared to the PI controller that heavily relies on offline modeling of the controlled system [17], e.g., a multimedia system.

Figure 4 shows the temperature and QP curves for

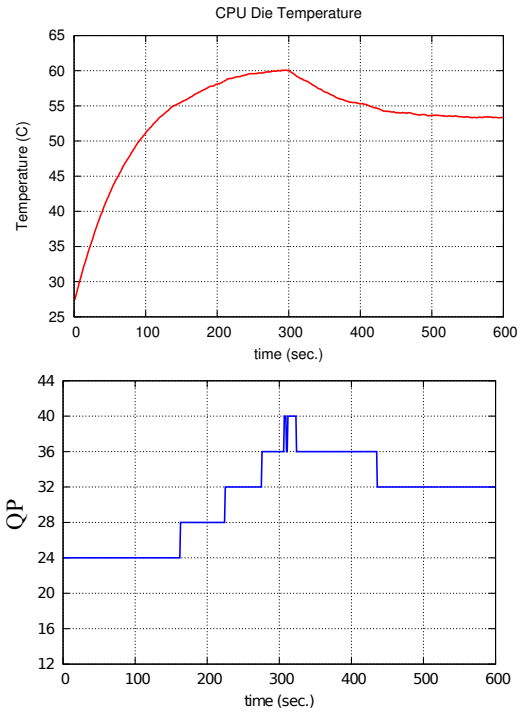

Figure 4. CPU temperature and QP of the PI controller for Elephant Dream

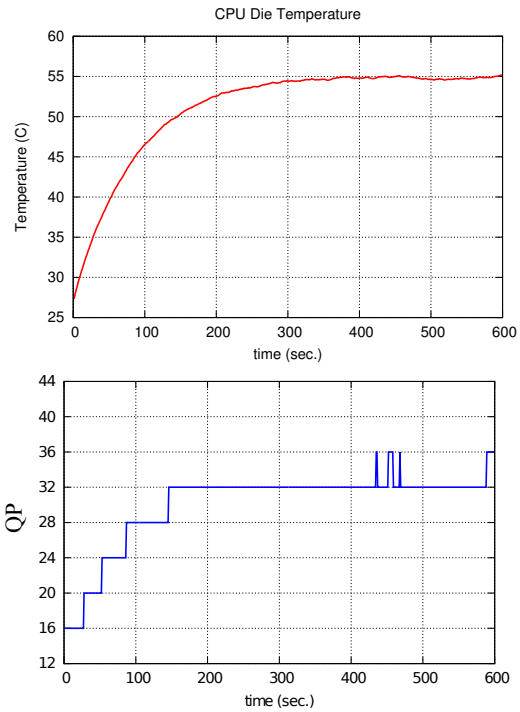

Figure 5. CPU temperature and QP of the predictive controller for Elephant Dream

the PI controller. The temperature begins to exceed $T_{max} = 55°C$ at 140s. The highest temperature is 60°C as shown in the figure. Since the PI controller is reactive, it starts to degrade QP after the CPU temperature exceeds T_{max}. As a result, it cannot cancel thermal overshoots until 400s. Since the heat cannot be dissipated immediately, the PI controller has to continuously degrade the video quality by increasing the QP from 24 to 40 between 140s and 310s. From 310s to 390s, the temperature converges down to the vicinity of T_{max} due to the QoS degradation in *reaction* to the thermal errors measured in the feedback loop. When the feedback controller detects a temperature undershoot after 420s, it starts to roll back to a higher quality level by decreasing QP back to 32 as shown in Figure 4. Due to the reactive nature and non-immediate heat dissipation, the PI controller suffers from the temperature overshoots that exceed T_{max} and QoS degradation for a relatively long time. In fact, many deadlines could have been missed due to thermal faults incurring hardware-triggered CPU speed reductions, if T_{max} had been set to the hardware-specified maximum value. Thus, in this paper, we favor the PI controller by setting T_{max} to a lower value than the threshold specified by the hardware.

Figure 5 shows the CPU temperature and QP curves for our predictive controller. In our approach, the highest temperature observed for decoding Elephant Dream is only 55.2°C. Thus, it exceeds T_{max} by no more than 0.2°C. Different from the feedback controller, the QP does not exceed 36 as shown in Figure 5. Further, the QP is 36 only briefly. By comparing Figures 4 and 5, we observe that the predictive controller begins to increase the QP earlier than the reactive PI controller does. As soon as our approach predicts a temperature overshoot in the next control period, it immediately begins to gracefully degrade the video quality to avoid overheating. As a result, it closely supports T_{max}, while providing the video quality comparable to or slightly higher than the video quality supported by the PI controller.

Notably, our predictive approach estimates the maximum power consumption expected to increase the CPU temperature to T_{max} in the next control period. In practice, this is somewhat *pessimistic*, because the actual chip temperature may not increase so fast within one sampling period due to the physical thermal characteristics and any potential cooling. Being aggressive, however, our predictive model requires QoS adaptation to prevent thermal faults well in advance. As a result, it avoids relatively severe QoS degradation (or potential deadline misses) observed after thermal faults occur in the reactive method. A less aggressive model could

support higher QoS, missing no deadline or only a few deadlines due to small, controlled thermal overshoots. This is reserved for future work.

Tested Video	Control Method	Temperature Overshoot	Settling Time
B.B. Bunny	PIC	61.3 °C	286s
	PRE	55.4 °C	8s
Bridge (Close)	PIC	60.7 °C	278s
	PRE	55.2 °C	6s
Bridge (Far)	PIC	60.9 °C	274s
	PRE	55.3 °C	6s
E. Dream	PIC	60.0 °C	270s
	PRE	55.2 °C	5s
Highway	PIC	63.2 °C	297s
	PRE	55.3 °C	11s
Paris	PIC	61.0 °C	281s
	PRE	55.1 °C	3s

Table 2. Performance of controllers for different videos (PIC: PI Controller, PRE: Predictive Approach)

The performance results of the tested approaches for the other videos are similar to Figures 4 and 5. Table 2 shows the highest temperature overshoot and the total length of the time intervals in which the temperature exceeds the threshold, called the settling time in control theory [17], for the feedback control and predictive schemes. As summarized in the table, our predictive method exceeds T_{max} by no more than 0.4°C, i.e., 0.007%. Further, the settling time is between 3s − 11s only. In contrast, the PI controller exceeds T_{max} by 5°C − 9.4°C, i.e., 9% − 17%. Also, its temperature settling time is 270s or longer.

5 Related Work

In this section, a high level overview of previous work on thermal control in real-time systems is given.

Reactive Methods. A number of reactive approaches to CPU temperature control in real-time systems have been developed. Wang et al. [25] analyze the schedulability of hard real-time tasks under a reactive thermal control scheme. Chantem et al. [3] adapt the processor speed via DVFS once the CPU chip reaches the temperature threshold. Kumar and Thiele [14] compute the delay bound by analyzing a variable stream of jobs when the temperature constraint is supported via reactive speed scaling. Quan et al. [19] develop a reactive control algorithm, in which the CPU voltage is decreased when the temperature exceeds the specified threshold. Fu et al. [9] design a two-level nested feedback controller to manage the CPU temperature by manipulating the CPU utilization. However, reactive temperature control for real-time multimedia applications is very hard, because the accumulated heat cannot be dissipated immediately upon a thermal overshoot. Thus, approaches that react to thermal overshoots occurred in the previous control period may suffer from deadline misses and QoS degradation due to thermal faults for a relatively long interval of time. As a result, the reliability of the circuit could be diminished too.

Predictive Approaches. In their proactive setting, Wang et al. [25] devise a processor speed scheduler at design time to support thermal constraints in real-time systems. The feasibility of leakage-aware periodic tasks under thermal constraints is analyzed in [19]. Hettiarachchi et al. [11] have developed a predictive technique to allow a system designer to precisely quantify the hard real-time performance degradation due to thermal events *a priori*. In real-time multimedia applications, however, it is very hard to derive appropriate QoS levels, which are neither too pessimistic nor too optimistic under a temperature constraint at design time. Instead, at each control point, we update the predictive model and dynamically adapt the QP, if necessary, to avoid overheating via minimal QoS adaptation in the next control period. In addition, temperature management schemes are developed for multiprocessor real-time systems [26, 20, 4, 2, 7, 10].

However, most of the approaches discussed in this section rely on DVFS or power gating, an offline analysis of the worst case execution times for the known input, or static speed assignment at design time. DVFS or power gating without video quality adaptation may severely degrade the QoS as discussed before. Also, approaches based on an offline analysis of the task execution times for the known input are not directly applicable to thermal control for different videos, because the workload may vary considerably from frame to frame even in one video.

6 Conclusions and Future Work

In a number of applications, e.g., visual surveillance or traffic control, real-time systems need to deal with computationally demanding and dynamic multimedia workloads, e.g., HD video streams. In these systems, CPU thermal faults can result in many deadline misses. In this paper, we derive a CPU temperature prediction model and design a predictive approach to avoiding temperature overshoots based on the model, while supporting as high video quality as possible under the temperature constraint. By extending an open-source

H.264/SVC framework, we have compared the performance of our approach to a static approach and a reactive feedback controller representing the state of the art. In our approach, the highest CPU temperature does not exceed the threshold by more than 0.007% and a temperature overshoot is canceled in no more than 11s. Further, the video quality supported by our approach is similar to or slightly better than that provided by the baselines, which fail to control the temperature to be below the threshold for 270 seconds or more in 10 minute experiments. In the future, we will continue to enhance our approach, while exploring other research issues, such as thermal control for multicore real-time systems.

References

[1] D. Bertsekas. Dynamic Programming and Suboptimal Control: A Survey from ADP to MPC. *European Journal of Control*, 11:301–334, 2005.

[2] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. In *Design, Automation and Test in Europe*, 2008.

[3] T. Chantem, X. S. Hu, and R. P. Dick. Online Work Maximization under a Peak Temperature Constraint. In *International Symposium on Low power Electronics and Design*, 2009.

[4] J.-J. Chen, C.-M. Hung, and T.-W. Kuo. On the Minimization of the Instantaneous Temperature for Periodic Real-Time Tasks. In *IEEE Real- Time and Embedded Technology and Applications Symposium*, 2007.

[5] X. Chen, J. hong Zhang, W. Liu, Y. sheng Liang, and J. qiang Feng. H.264/SVC parameter optimization based on quantization parameter, MGS fragmentation, and user bandwidth distribution. *EURASIP Journal on Advances in Signal Processing*, 2013(10), 2013.

[6] A. P. Ferreira, D. Mossé, and J. C. Oh. Thermal Faults Modeling Using a RC Model with an Application to Web Farms. In *Euromicro Conference on Real-Time Systems*, pages 113–124, 4-6 2007.

[7] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele. Thermal-Aware Global Real-Time Scheduling on Multicore Systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2009.

[8] GNU FreeIPMI Project. `http://www.gnu.org/software/freeipmi/`.

[9] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. Koutsoukos, and H. Wang. Feedback Thermal Control for Real-time Systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010.

[10] Y. Fu, N. Kottenstette, C. Lu, and X. D. Koutsoukos. Feedback thermal control of real-time systems on multicore processors. In *ACM International Conference on Embedded Software*, 2012.

[11] P. Hettiarachchi, N. Fisher, M. Ahmed, L. Y. Wang, S. Wang, and W. Shi. The Design and Analysis of Thermal-Resilient Hard-Real-Time Systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2012.

[12] P.-C. Hsiu, C.-H. Lin, and C.-K. Hsieh. Dynamic Backlight Scaling Optimization for Mobile Streaming Applications. In *IEEE/ACM International Symposium on Low Power Electronics and Design*, 2011.

[13] JSVM Software Manual - Google Code. `stream-svc.googlecode.com/files/SoftwareManual.pdf`.

[14] P. Kumar and L. Thiele. Timing Analysis on a Processor with Temperature-Controlled Speed Scaling. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2012.

[15] C.-H. Lin, P.-C. Hsiu, and C.-K. Hsieh. Dynamic Backlight Scaling Optimization: A Cloud-Based Energy-Saving Service for Mobile Streaming Applications. *IEEE Transactions on Computers*, 63(2):335–348, 2014.

[16] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

[17] C. Phillips and H. Nagle. *Digital Control System Analysis and Design*. Prentice Hall, 1995.

[18] Peak signal-to-noise ratio. `http://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio`.

[19] G. Quan, Y. Zhang, W. Wiles, and P. Pei. Guaranteed Scheduling for Repetitive Hard Real-Time Tasks under the Maximal Temperature Constraint. In *IEEE/ACM/IFIP International Conference on Hardware/Software Co-Design and System Synthesis*, pages 267–272, 2008.

[20] L. Schor, I. Bacivarov, H. Yang, and L. Thiele. Worst-Case Temperature Guarantees for Real-Time Applications on Multi-Core Systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2012.

[21] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103 –1120, September 2007.

[22] P. Seeling and M. Reisslein. Sample Video Sequences. `http://trace.eas.asu.edu/yuv/index.html`.

[23] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-Aware Computer Systems: Opportunities and Challenges. *IEEE Micro*, 23(6):52–61, 2003.

[24] Sample Video Sequences. `ftp://ftp.tnt.uni-hannover.de/pub/svc/testsequences/`.

[25] S. Wang and R. Bettati. Reactive Speed Control in Temperature-Constrained Real-Time Systems. *Real-Time Systems*, 39(1-3):73–95, 2008.

[26] Y. Wang, K. Ma, and X. Wang. Temperature-Constrained Power Control for Chip Multiprocessors with Online Model Estimation. In *International Symposium on Computer architecture*, 2009.