Predictive Thermal Control for Real-Time Video Decoding

Mehmet H. Suzer and Kyoung Don Kang

Department of Computer Science
State University of New York at Binghamton

kang@binghamton.edu

Research Problem

- Important **soft** real-time/cyber physical applications, e.g., visual surveillance or transportation management, need to handle demanding multimedia workloads, e.g., HD video frames
- Processor is subject to high power consumption and overheating
- Accumulated heat cannot be dissipated immediately
- Dynamic & demanding workloads
- Thermal fault may cause playtime deadline misses due to, for example, clock throttling (Assume no deadline miss if no thermal fault)

Contributions

A new empirical model

Predict CPU temperature by directly considering CPU thermal characteristics and multimedia application semantics

Feedforward and Feedback Control (adaptive nonlinear control)

Periodically update the predictive model at each control point to capture time-varying relation between provided video quality and predicted CPU power consumption and temperature via feedback

Minimal QoS Adaptation

Adapt video quality by a minimal degree within a specified range to avoid overheating in the next control period

Overview of H.264/SVC Standard

- Substantially enhance the coding efficiency and increase video scalability
- One video can be coded with a combination of different temporal rates, spatial resolutions, and quantization parameters (QPs) at the coding time for clients with diverse network connections and devices (wired, wireless, large screen, mobile display, etc.)
- Provides spatial, temporal, and quality scaling to adapt resolution, frame rate, and PSNR (Peak Signal to Noise Ratio), respectively

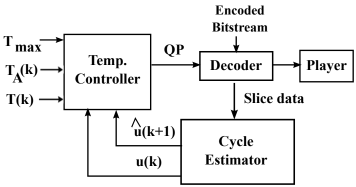
Video Quality Adaptation to Avoid CPU Overheating

Quality Scaling

- Applied to adapt video quality to prevent thermal faults in this paper
- Known to provide less QoS fluctuations compared to resolution/frame rate adaptation
- However, our approach is not limited to quality scaling

Quality scaling example: Frames decoded using QP=16, 28, 36, and 40 are shown from top-left to bottom-right

Overall System Archtecture



System Architecture

- CPU chip temperature: *Controlled variable*
- *QP: Manipulated variable*
- T_{max}: CPU temperature threshold
- $T(k)$ and $T_A(k)$: Chip and ambient temperature at the k^{th} control point (time kP_c where P_c is control period)
- $u(k)$: CPU cycles consumed for video decoding in k^{th} control period, i.e., time interval $[k(P_c - 1), kP)$
- $\hat{u}(k+1)$: estimated CPU cycles needed to decode video in $(k+1)^{th}$ control period

Procedure for 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 and avoid violating 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.

Predictive Model Design

RC temperature model (Skadron et. al. [4]):

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

- R: thermal resistance
- C: thermal capacitance
- $T(t)$: current chip temperature at time t
- $T_A(t)$: current ambient temperature
- p(t): CPU power consumption

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:

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

- $A = e^{-P_c/RC}$, $B = (1-A)R$, and $x(k) = T(k) - T_A(k)$
- $\hat{p}(k+1)$ represents the predicted power consumption for video decoding in the $(k+1)^{th}$ control period
- Assume ambient temperature does not significantly change between two consecutive control points

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

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

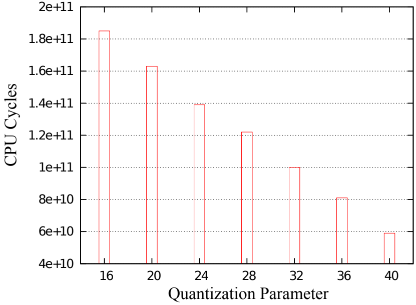
- P_{Idle}: idle power consumption
- P_{Idle} is measured offline when the CPU is idle
- $P_f(k)$: power factor is the gain capturing the relation between the number of the CPU cycles and power consumption for video decoding at the k^{th} control point

Our predictive model is **not** tied to the linear assumption between the CPU cycle and power consumption, because the power factor is continuously updated at every control point based on the RC thermal model.

For $k \geq 1$, we derive $P_f(k)$:

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

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), which is a hardware counter readable through the IPMI (Intelligent Platform Management Interface)



QP vs. CPU Cycles
(test movie Harbor [5] with
704×576 resolution and 60 fps)

- Linear relationship between the quality increment and bit rate of coded videos (Chen et. al. [1])
- We empirically verified the linearity for several videos heavily used in multimedia research [3, 5]
- CPU cycle consumption increases almost linearly as QP decreases, and video quality enhances accordingly

Compute the utilization factor based on linear relation between QP and CPU cycle consumption for video decoding:

$$U_f(k) = \frac{u(k)}{QP(k)} \tag{5}$$

- $U_f(0)$ is calculated offline by decoding a few frames
- Utilization factor is also updated at every control point to closely keep track the relation between the QP and CPU cycle consumption that may vary in time

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

$$\hat{u}(k+1) = U_f(k) \cdot QP(k+1) \tag{6}$$

- manipulated variable, $QP(k+1)$, is the smallest possible QP expected to support highest video quality without overheating the CPU in the $(k+1)^{th}$control period

After some substitutions and algebraic manipulations, the estimated utilization needed to support $QP(k+1)$ is:

$$u_{max}(k+1) = \frac{p_{max}(k+1) - P_{Idle}}{P_f(k)} \quad (7)$$

From it, $QP(k+1)$ is:

$$QP(k+1) = \left\lceil \frac{u_{max}(k+1)}{U_f(k)} \right\rceil \quad (8)$$

Finally, ensure $QP_{min} \leq QP(k+1) \leq QP_{max}$:

$$QP(k+1) = \begin{cases} QP_{min} & \text{if } QP(k+1) < QP_{min} \\ QP_{max} & \text{if } QP(k+1) > QP_{max} \\ QP(k+1) & \text{otherwise} \end{cases} \tag{9}$$

For more mathematical details, please refer to our paper available at:
`http://www.cs.binghamton.edu/~kang/ecrts14.pdf`

Performance Evaluation

- Micro-testbed built upon a Linux laptop with the 1.6 GHz Intel Pentium M processor and 512 MB of RAM using sample videos [3, 5]
- Thermal Control Parameters

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

- Baselines:
  - Static Approach: Use a fixed QP derived offline
  - Reactive feedback controller, PI controller similar to Fu et. al. [2]: Adapt the QP in reaction to a thermal error

Expeirmental Results

Static Approach

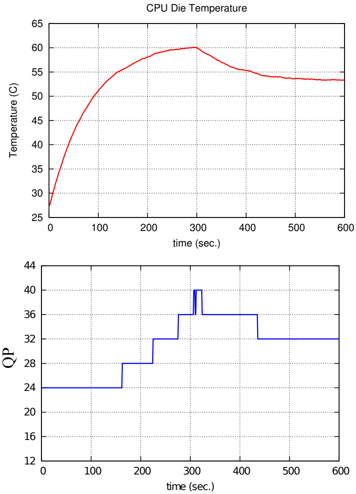
Suffers from either a large temperature overshoots or poor video quality. Thus, results are not presented.

Reactive PI Controller

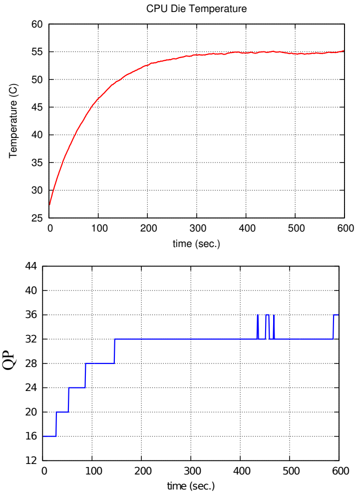
Shows much better performance than static approach does. However, it experiences temperature overshoots due to the reactive nature undesirable for thermal control

Proposed Feedforward + Feedback Approach

Temperature exceeds the threshold (55°C) by no more than 0.4°C for similar or lower QP (higher video quality)



CPU temperature and QP of the PI controller for Elephant Dream [3]



CPU temperature and QP of the predictive controller for Elephant Dream [3]

Performance Results for Other Videos

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: Performance of PIC (PI Controller) and PRE (Predictive Approach)

Conclusions and Future Work

- A new empirical model to predict CPU temperature by directly considering CPU thermal characteristics and multimedia application semantics
- Feedforward and feedback control (adaptive nonlinear control) to periodically update the predictive model at each control point
- Adapt video quality by a minimal degree within a specified range to avoid overheating in the next control period
- Closely support the CPU temperature threshold, while supporting similar or better video quality compared to the tested baselines
- In the future, multicore thermal control issues will be explored: Assign frames or slices of the same frame to the cores? Thermal/workload balancing between cores?

Thanks! Questions?

Xu Chen, Ji hong Zhang, Wei Liu, Yong sheng Liang, and Ji 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.

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

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

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.

Sample Video Sequences.
ftp://ftp.tnt.uni-hannover.de/pub/svc/testsequences/.