Machine Learning-Based Quality-Aware Power and Thermal Management of Multistream HEVC Encoding on Multicore Servers

Arman Iranfar, *Student Member, IEEE,* Marina Zapater, *Member, IEEE,* and David Atienza, *Fellow, IEEE*

Abstract—The emergence of video streaming applications, together with the users' demand for high-resolution contents, has led to the development of new video coding standards, such as High Efficiency Video Coding (HEVC). HEVC provides high efficiency at the cost of increased complexity. This higher computational burden results in increased power consumption in current multicore servers. To tackle this challenge, algorithmic optimizations need to be accompanied by content-aware application-level strategies, able to reduce power while meeting compression and quality requirements. In this paper, we propose a machine learning-based power and thermal management approach that dynamically learns and selects the best encoding configuration and operating frequency for each of the videos running on multicore servers, by using information from frame compression, quality, encoding time, power, and temperature. In addition, we present a resolution-aware video assignment and migration strategy that reduces the peak and average temperature of the chip while maintaining the desirable encoding time. We implemented our approach in an enterprise multicore server and evaluated it under several common scenarios for video providers. On average, compared to a state-of-the-art technique, for the most realistic scenario, our approach improves BD-PSNR and BD-rate by 0.54 dB, and 8%, respectively, and reduces the encoding time, power consumption, and average temperature by 15.3%, 13%, and 10%, respectively. Moreover, our proposed approach enhances BD-PSNR and BD-rate compared to the HEVC Test Model (HM), by 1.19 dB and 24%, respectively, without any encoding time degradation, when power and temperature constraints are relaxed.

Index Terms—HEVC standard, multicore servers, encoding efficiency, encoding configuration, reinforcement learning.

1 Introduction

Video streaming services are expected to account for 80% of global traffic by 2019 [1], with services such as Netflix and YouTube accounting for over 50% of downstream traffic [2]. Due to the great variety of devices accessing media content as well as the users' demand for higher quality video, encoding has become a key application in current High Performance Computing (HPC). To satisfy the emerging large video resolutions, the High Efficiency Video Coding (HEVC) standard provides twice the compression of its predecessors [3] while maintaining the same video quality, at the price of increasing the encoder complexity by several times [4]. HEVC also brings more flexibility in terms of encoding parameters, which makes selecting the most appropriate encoding configuration more challenging.

The undeniable complexity of HEVC, together with the increase of video streaming users, poses an important challenge for power- and thermal-aware resource allocation and management of these applications when running on multicore servers. In particular, users daily upload more than 65 years of contents to YouTube servers [5]. In order to satisfy massive streaming to a wide variety of personal devices, video providers need to decode and encode the video into several formats (a process named *transcoding* [6]). This poses a high computational burden on the providers' server facilities, leading to the need of developing runtime quality-aware power and thermal management for multicore servers. Current research in the area is mostly focused on the software optimization of one or several blocks of the encoding algorithm. However, to address the challenge of power and thermal management for HEVC, application-level configuration and system-level parameters need to be jointly integrated on top of algorithmic optimization. To the best of our knowledge, such a holistic approach, that tackles encoding efficiency (i.e., quality and compression) and encoding time, together with power consumption and thermal control, for several videos running on multicore servers, has not been addressed in the literature.

In addition, when dealing with multiple encoding requests on a multicore server, a proper video (i.e., workload) assignment strategy is vital for reducing the thermal hot spots while maintaining the desirable encoding time. Indeed, the increased complexity of HEVC requires high-performance architectures and induces more frequent hot spots.

In this paper, we propose a machine learning (ML)-based approach for power and temperature management of multistream HEVC encoding on multicore servers, where multiple video streams need to be processed concurrently. The initial idea of this paper was presented in our previous work [7]. We build upon the observations that different encoding configurations and frame properties result in different combinations of power, temperature, video quality, compression, and encoding time. Consequently, our proposed ML-based approach, using Q-learning (QL) algorithm, learns and applies the best encoding configuration and per-core frequency for each of the videos being encoded on a multicore server. Our proposed methodology situates on top of any algorithmic optimization and is able to improve the target objectives beyond the given baseline application. In addition, we present a resolution-aware video assignment strategy for multicore servers, in order to reduce the thermal hot spots without encoding time degradation.

In particular, our main contributions are as follows:

- we propose an reinforcement learning (RL)-based approach to utilize both application- and system-level parameters in order to increase encoding efficiency

This work has been partially supported by the EC H2020 MANGO project (GA No. 671668), and the ERC Consolidator Grant COMPUSAPIEN (GA No. 725657).

while satisfying power and thermal constraints without encoding time degradation. Our approach is able to learn from the obtained encoding time, encoding efficiency, and server's power and temperature, and dynamically sets both encoding configuration and frequency during runtime for arbitrary video.
- we develop a resolution-aware video assignment strategy to reduce temperature while maintaining the desired encoding time.
- we evaluate how our approach is able to perform power and thermal management of multicore servers when multiple videos are processed at the same time. We accurately evaluate our solution in a multicore enterprise server, where we collect power and temperature traces, as well as application-specific parameters, at runtime. For the evaluation, we consider several possible scenarios on the servers of video providers. Specifically, we show how it benefits from resource availability to improve encoding efficiency.

The rest of paper is organized as follows. Related works are briefly reviewed in Section 2. Section 3 illustrates the motivation behind this work. In Section 4, we present a heuristic video assignment strategy. In Section 5, we present our ML-based runtime management in detail. Experimental setup and results are explained in Section 6 and 7, respectively. Finally, the main conclusions of this work are summarized in Section 8.

2 Related Work

Power and thermal management of multicore processors [8], [9] and research in multimedia applications [10], [11] have been extensively discussed in the literature. In particular, there are several works providing power reduction and/or encoding time enhancement for multimedia workloads most of which targeting the previous standards, such as H.264/AVC [12] (*e.g.*, [13], [14]). However, these works need modifications to conform with HEVC requirements due to its higher complexity.

On the HEVC side, although a few works tackle the decoder design and optimization (e.g., [15]), latest research has focused more on the encoder as it is approximately 100 times more complex [4]. In this context, the majority of the work includes algorithmic optimization with various objectives such as power and complexity reduction, encoding time enhancement, etc.

Several works focus on the time reduction and performance improvement of HEVC encoders [16], [17]. One approach to increase the throughput of the encoder is taking advantage of the tile feature in the HEVC standard. In this context, Shafique et al. [18] and Khan et al. [19] efficiently split a frame into different number of tiles and gain speedups since each tile can be processed independently and, hence, regarded as a thread.

Moreover, the HEVC encoder complexity is highly dependent on the depth levels of Coding Units (CUs) [3]. Thus, CU depth reduction has been addressed by recent works [20], [21], where it is shown that the reduction of the computational complexity of the encoder leads to a decreased energy or latency of the application.

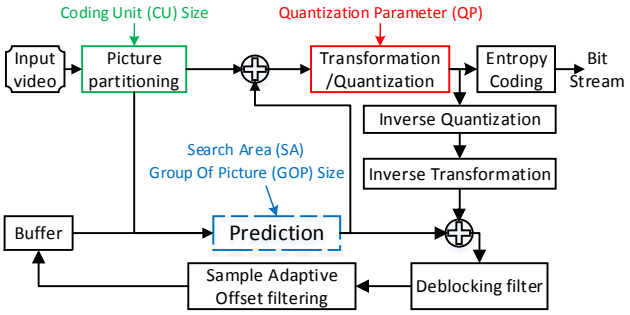


Figure 1. HEVC encoder block diagram and main configuration parameters

Finally, several works directly aim at power reduction of HEVC streaming by proper thread allocation [22], and memory bandwidth reduction [23]. On the decoder side, a power-aware streaming framework is proposed by [24], targeted at mobile devices. The number of reference frames also influences the power consumption since as it increases, more data must be transferred and processed. Some works [25] decrease the memory bandwidth and, hence, power consumption by reducing the reference frames.

In all of the above works, the encoder optimization does not consider temperature as an important issue of today's multicore servers. Nevertheless, a few works consider temperature constraints as well as encoding efficiency of next generation video encoders [26], [27], [28], [29]. In particular, Palomino et al. [28] employ an adaptive approximate computing method at both algorithm and data levels to optimize the thermal profile. Alternatively, the authors in [26], first, perform an offline analysis to explore the relation of video properties and encoding configuration with CPU temperature. Then, they propose an application-driven thermal management policy. Shafique and Henkel [29] address the complexity reduction of HEVC, use hardware accelerators for low-power HEVC, and apply a dynamic thermal management similar to [30].

The most similar work to ours is TONE [27]. Authors, first, extract Pareto optimal curves of the temperature associated with different encoding configurations. Then, a prediction model based on frame complexity is used to predict the temperature resulted from next frame. If a temperature threshold is exceeded, a new encoding configuration optimizing the encoding efficiency, is selected. To recap, TONE [27] provides a multi-objective solution for HEVC encoding with respect to video quality, compression, and temperature.

Nonetheless, none of these works ([26], [27], [28], and [29]) considers power consumption, temperature, encoding time, and encoding efficiency jointly. Moreover, when multiple videos are running at the same time on a multicore platform, power and thermal management of HEVC is more challenging and has not been addressed so far. As opposed to previous works, we propose an ML-based runtime power and thermal management approach which exploits configuration parameters in addition to CPU frequency for multi-stream HEVC encoding on multicore servers. In addition, we present a low-overhead video assignment strategy to reduce the temperature of the chip, while maintaining the encoding time requirements.

3 Analysis of the HEVC Encoder

In this work, we consider the Peak Signal-to-Noise Ratio (PSNR), measured in dB, as the quality; bitrate, measured in bits per second (bps), as the compression; and encoding time as the application output metrics. The goal is to achieve higher PSNR along with more compression (lower bitrate) while reducing the encoding time per frame. In this work, we have studied all configuration parameters available in the reference software HM 16.3 (more than 100) by conducting experiments using the experimental setup explained in Section 6. Since not all encoding parameters affect the output PSNR, bitrate, power, temperature, and encoding time considerably, we take into account those with the largest impact (shown in Table 2). Moreover, although some parameters can take a wide range of values, we limit this range based on our observation on the output PSNR and bitrate, as well as according to the Joint Collaborative Team on Video Coding (JCT-VC) documentation [31]. The idea behind constraining some of the encoding parameters is to avoid irrelevant ones, which will finally result in no gain with respect to the objectives, but with larger time required for the convergence of ML algorithm.

Figure 1 illustrates a simplified HEVC encoder block diagram. Each block contains several parameters to configure the encoder (i.e., configuration parameters). Prior to all the blocks, the largest CU (LCU) size is specified. Search range, prediction mode, GOP (group of picture) size, and reference frames are used in the prediction block. Also, the quantization parameter (QP) is used to control the level of quantization. All these parameters can be dynamically tuned frame-by-frame, except for the GOP size that can only be changed every several frames. Finally, when $GOP = 1$, frames are considered as I-frame, while when $GOP = 8$ *or* 16, only the first frame is considered as I-frame and the rest in the GOP structure are B-frames.

In contrast to the HEVC decoder, the diversity of the HEVC encoder parameters and their wide value range [4] requires a careful study of their respective impact on encoding efficiency and time, power consumption, and temperature. Although QL, as a model-free ML algorithm, does not depend on any characterization of single or multiple videos running on the server, the following study motivates why an ML-based approach is necessary for quality-aware power and thermal management of multistream HEVC encoding on multicore servers.

3.1 Impact of Video Content on Encoding Efficiency, Encoding Time, Power, and Temperature

Apart from the inherent and exclusive features of each video type, such as frame resolution, bit depth, etc., the contents of a video play a major role in the obtained encoding time, quality, compression, power consumption, and peak temperature, resulted from a specific encoding configuration.

For this section, all test sequences are encoded using the default *Main Intra* configuration provided by the reference software, HM 16.3 [32]. For all experiments throughout this work, data are gathered from the experimental setup specified in Section 6 with the maximum frequency otherwise noted. Figure 2 shows the encoding time, bitrate, and PSNR obtained when encoding seven different 8-bit test sequences, whose specifications are shown in Table 1.

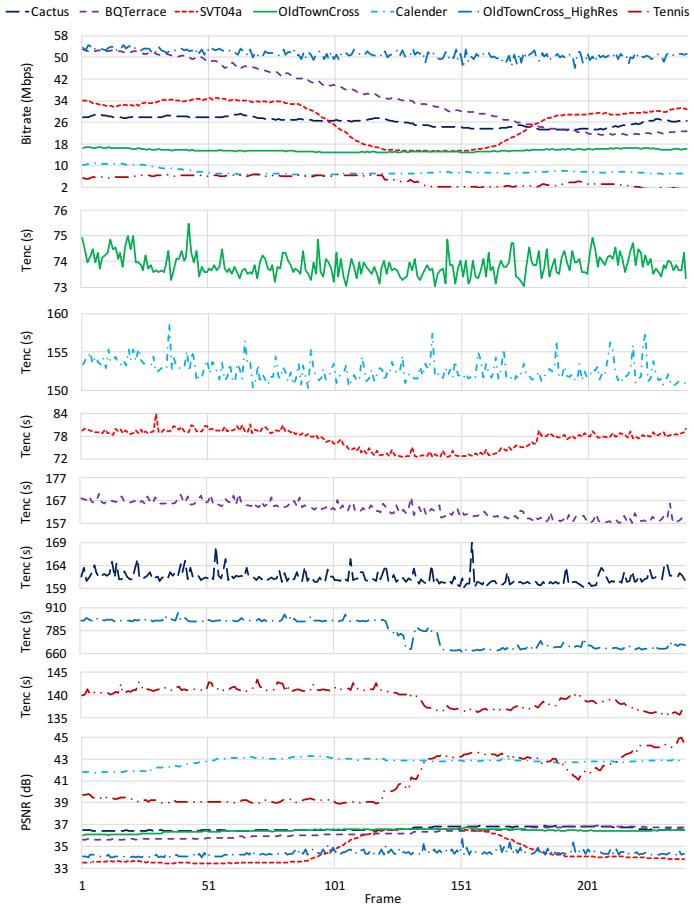


Figure 2. Per-frame bitrate, encoding time (T_{enc}), and PSNR for seven test sequences with *Main Intra* configuration

These videos cover a wide variety of frame-to-frame motion, and texture within a frame. While *Cactus* and *BQTerrace* are among the test sequences introduced by JCT-VC [31], the rest are being frequently used in benchmarking video coding applications [33], [34], [35].

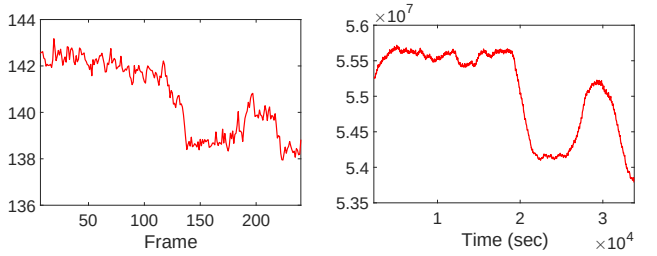
As shown in Figure 2, the obtained metrics not only do differ considerably from one video resolution to another (OldTownCross vs. OldTownCross_HR), but they also differ vastly from one video to another of the same resolution (Cactus vs. Calendar), and even, within a video (all videos).

Such variations in video contents impact the encoding time directly through affecting the memory sub-systems. In fact, while the CPU is fully utilized throughout the video encoding (tested on an Intel S2600GZ server with a E5-2620 CPU), the number of accesses/misses to/from L2 cache and Last Level Cache (LLC) change vastly as the contents vary from one frame to another. This behavior is consistent across different platforms (Intel Xeon X5650 and E5-2620, and AMD Opteron 6272 and 6300).

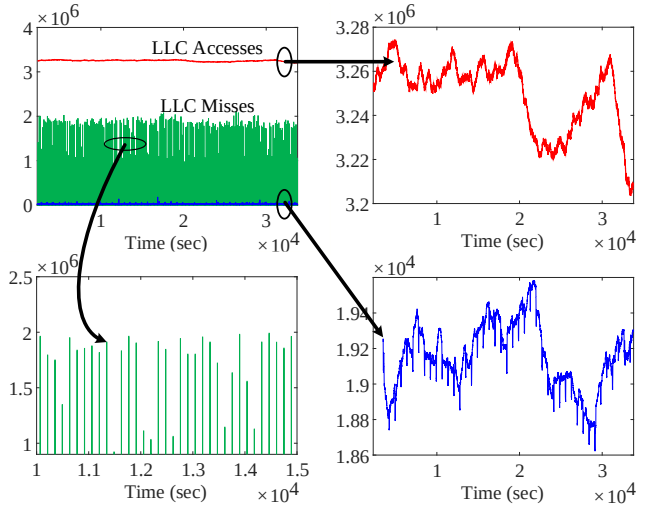
Figure 3 shows the number of accesses to L2 and LLC, and misses from LLC for test sequence *Tennis* every second collected by Oprofile [36]. Comparing the number of accesses to L2 and LLC with the encoding time per frame for the *Tennis* sequence clearly shows the importance of video contents on memory hierarchy and, hence, on the encoding time as the trend thoroughly coincides with this metric (the same trend is observed for misses from L2).

Table 1
Test sequences and their relevant features used in this work

	SVT04a	OldTownCross	Tennis	Cactus	Calendar	BQTerrace	OldTownCross_HR
Resolution	1280x720	1280x720	1920x1080	1920x1080	1920x1080	1920x1080	3840x2160
Frame rate (Hz)	50	50	24	50	50	60	50
Frame count	500	500	240	500	500	600	500
Target bitrate	4000kbps	4000kbps	4000kbps	6000kbps	6000kbps	6000kbps	8000kbps

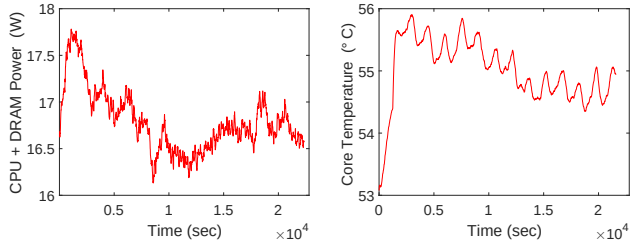


(a) Encoding time/frame (sec) (b) Number of L2 Accesses per second



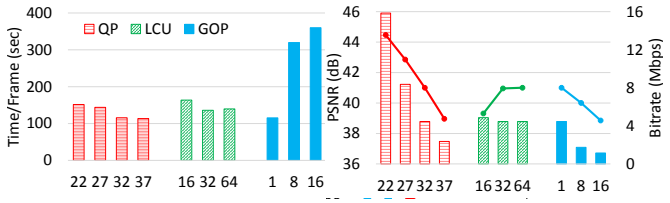
(c) Number of LLC Accesses and Misses per second

Figure 3. Number of accesses to L2, accesses to LLC, misses from LLC and encoding time/frame for *Tennis*



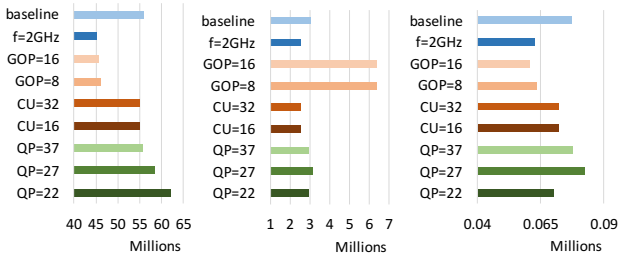
(a) Power consumption (b) Core Temperature

Figure 4. Content-based power and temperature variation



(a) (b)

Figure 5. Impact of different encoding parameters on (a) encoding time, and (b) on PSNR and bitrate, for the test sequence *Tennis*



(a) L2 Accesses (b) LLC Accesses (c) LLC Misses

Figure 6. Average number of accesses to L2, accesses to LLC, and misses from LLC every second

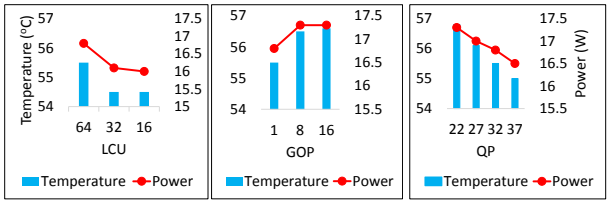


Figure 7. Application parameters' impact on CPU power and temperature for *Tennis* running on one core

As Figure 3a and Figure 3b show, there is a correlation between the number of L2 accesses and encoding time. In this sense, when the number of L2 (or LLC) accesses is small, lower encoding time/frame is observed. Although such a pattern seems to be absent in the trend of LLC misses in Figure 3c, this figure contains important information. In fact, LLC misses consist of two traces. The first one (bottom left) includes the spikes (in the range of $1 \sim 2 \times 10^6$) which represent misses due to loading a new frame from the main memory. The second trace (bottom right) appears around 2×10^4 and follows the same variations as that of the encoding time/frame. This paper focuses on the latter as it is more correlated to the obtained encoding time/frame trace.

The similarity between content variations (reflected in encoding time) and memory sub-system events demonstrates the importance of frame-by-frame power and thermal management, since such variations ultimately affect the power and temperature of the chip. Figure 4 shows the power and temperature variations with respect to content variations in the input test sequence *Tennis*.

3.2 Impact of Encoding and System Parameters on Encoding Efficiency, Time, Power, and Temperature

Figure 5 shows, on average, how different parameters affect encoding time, PSNR and bitrate. Although not shown in the figure, similar tradeoffs are observed for search range and reference frames, for all videos considered in this paper.

Similar to the video contents, encoding parameters affect the memory sub-systems considerably, resulting in significant change in encoding time, power consumption, and temperature. Figure 6 shows how different encoding configurations in addition to the operating frequency change the average number of L2 and LLC accesses, and LLC misses for test sequence *Tennis*. For this figure we use the default *main Intra* configuration (QP=32, GOP=1, CU=64) with the maximum frequency (2.4 GHz) as the *baseline*. Then, we provide the number of memory events by changing these system- and application-level parameters. Because these metrics are correlated with encoding time, as previously shown, improvements on encoding time at the CPU level will also have a beneficial impact on the power consumption of the memory subsystem, due to the reduced accesses to memory. Such observations imply the significance of application-level parameters in encoding efficiency and time, ultimately affecting power and temperature of the chip. Figure 7 shows how power and temperature are affected by LCU, GOP, and QP, which are the most important encoding parameters.

Finally, while CPU frequency does not affect the encoding efficiency, it plays a major role in encoding time, power consumption, and peak temperature. Therefore, frequency has to be considered as a major runtime parameter along with all application parameters.

3.3 Discussion

The great number of different combinations of configuration parameters, in addition to sudden content variations within a video and the substantial differences across videos, require a more exhaustive solution than those proposed by previous works.

In this paper, we propose a two-stage power- and thermal-aware video assignment strategy and runtime management for multistream HEVC encoding shown in Figure 8. When the users' encoding requests along with the corresponding videos are received, first, each video and, in particular, each frame, should be assigned to a core in the multicore server. Meanwhile, an ML-based runtime management takes care of adjusting the cores frequencies and tuning the application-level parameters. In particular, we adopt the QL algorithm, which is able to learn the states resulted from the taken actions. This approach suits well multicore servers where multiple videos are to be processed at the same time.

Our approach can be implemented on top of any HEVC implementation, regardless of its specific performance, such as x265[1], as a parallel implementation, as long as they implement a wide range of control parameters. In particular, although x265 supports frame-level parallelization, our observation shows the same behavior as for HM 16.3 in figures 2, 3, 4, 5, 6, and 7. Consequently, the runtime adaptation of encoding parameters is a promising solution for quality-aware power and thermal management of next generation encoders.

4 Heuristic Video Assignment and Migration

Video assignment plays an important role in power consumption, peak temperature, and encoding time. This is

1. http://x265.org/

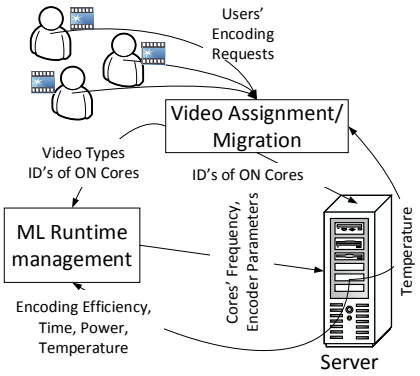


Figure 8. Proposed Approach

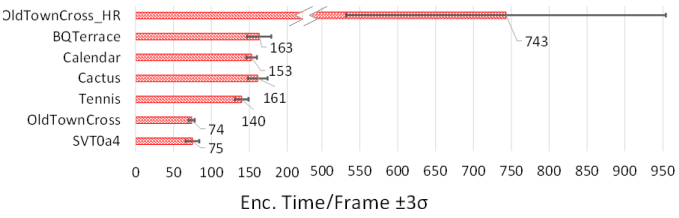


Figure 9. Average encoding time/frame for different videos when encoded by default *Main Intra* configuration

especially due to significant difference in thermal characteristics of different video types. Thus, a proper video assignment strategy aware of the exclusive features of videos such as resolution may provide reduced peak/average temperature. In addition, a proper video assignment strategy is able to affect the encoding efficiency. In other words, the temperature reduction resulted from the video assignment provides the ML-based runtime manager with more opportunities to increase the encoding efficiency and time by tuning the application- and system-level parameters.

As indicated in Section 3.2, higher frequency leads to higher temperatures and hot spots. On the other hand, Figure 9 shows the time spent on average for a frame to be processed for each of the seven different videos included in Table 1. These results refer to the use of the default *main intra* configuration (average and standard deviation are highly dependent on the encoding configuration and increase considerably if GOP is not one). Although video contents play a significant role on encoding time, the major driver of encoding time per frame is resolution. Consequently, use of higher frequencies is necessary for videos with higher resolution. However, when the power consumption is constrained by a power budget or a power cap, all the videos running concurrently on the target multicore server cannot benefit from the highest available frequency.

Based on such observation, we propose a video assignment and migration strategy which takes into account the resolution of the videos and current thermal profile of the chip, shown in the pseudo code of Algorithm 1. In this pseudo code, N_c is the total number of cores on the target multicore server, r_i is the resolution of the video running on the i^{th} core, where $i \in \{1, ..., N_c\}$, $\mathbf{C}$ shows the set of available cores, Θ represents the set of temperature values read from the available sensors, and r_{new} represents the

Algorithm 1: Video assignment strategy

Input : $N_c, floorplan, \Theta, \boldsymbol{R_s} = \{r_i\}, \boldsymbol{C} = \{c_k\}, r_{new}$
Output: n_c ; // Index of the assigned core
1 **forall** k **do**
2 $\quad M_k = \sum_{i}^{Adjacent_k} r_{i,k} + r_{new}$
3 **if** $Num(\underset{k}{\text{argmin}}(M_k)) > 1$ **then**
4 $\quad n_c \leftarrow \underset{j}{\text{argmin}}(\theta_j)$
5 **else**
6 $\quad n_c \leftarrow \underset{k}{\text{argmin}}(M_k)$

resolution of the unassigned video. Once a new video starts, the best core to process it is determined based on a merit function M (line 2). For each available core k, the value of the merit function, M_k, is calculated by summing the resolution of the videos running on the adjacent cores in addition to r_{new} (line 2). Thereafter, the core whose merit function value is the smallest will be selected as the destination core for the new video (line 6). If there is more than one core with the minimum M value, we choose the one with the lowest temperature (line 4).

In the proposed video assignment strategy, we first rely on the resolution of the assigned videos (r_i) and the new video (r_{new}), rather than on the instantaneous temperature, or even on the temperature history. Thus, the merit function considers the resolution of videos being processed on the adjacent cores of the available candidate, as well as the resolution of the new video. The higher this sum is, the higher the temperature becomes in the long term. In fact, as the current temperature could be strongly affected by a temporal high resolution video which does not exist anymore (i.e., its executions finished a few seconds ago), it is not possible to assign the new incoming video simply based on the temperature history as it does not guarantee to lower the average temperature in the future. Moreover, this strategy does not decide based on the instantaneous temperature, which is mainly the result of content variation of a video.

Since in multistream encoding on multicore servers after proper assignment of new videos other videos may leave the server, video migration from one core to the other is vital to satisfy encoding time and temperature requirements. Therefore, in this paper, we propose video migration every few seconds. In particular, if there is (are) any available resource(s), the video running on the hottest core should move to an available one. Since the resolution plays a major role, we use the same merit function as for the video assignment except for the change of the subscript from "new" to "hot" to indicate that this re-assignment is performed for the video experiencing a high temperature. Unlike the initial video assignment strategy, only those unoccupied cores whose current temperatures are lower than the hottest occupied core are considered as proper candidates. Here, once again, for each idle core, c_k, the value of the merit function, M_k is calculated by summing the resolution of the videos running on the adjacent cores, $c_{i,k}$, this time, in addition to r_{hot}.

The migration overhead depends on the encoding configuration and the resolution of the video. The latter plays a more important role and induces a maximum performance overhead of 0.2, 0.3, and 1.2 seconds, respectively, for 1280x720, 1920x1080, and 3840x2160 videos studied in this work. Comparing these value with average encoding time for different frame resolutions demonstrates that for the current HEVC test model [32] migration is applicable. Video migration conditions are checked every 2 minutes, and applied if a proper candidate is found, resulting in less than 1% performance overhead in the worst case.

In the proposed video assignment and migration strategy, if all cores are occupied, any new incoming video (encoding request) needs to wait in the non-preemptive queue until an available resource exists. In particular, we assume a first-come first-served policy to assign each video to a core. Hence, from all queued encoding requests, at each decision time (once a new available core is found), we serve only the first one in the queue. Also, since the proposed strategy can be used in the servers of video providers such as YouTube and Netflix, we assume that no other simultaneous tasks are running on the cores and, hence, no scheduling conflict occurs [37].

5 ML-Based Runtime Management

Although each HEVC encoder block has its own model, the interaction of the application parameters and input video with the processing platform cannot be characterized by any already known model where it would be possible to apply more conventional power and thermal management strategies, while considering encoding efficiency and time. Moreover, when encoding multiple videos with different contents and different encoding parameters, it is truly challenging to provide a modeled environment with predictable results. Nonetheless, in this context, RL can be used as is the best learning method when the goal is to perform a multi-objective optimization in a very dynamic environment. This dynamism exists in power and thermal management of multicore servers for HEVC encoder with its hundreds of possible encoding parameters, as well as several video types and content variation within videos. In fact, RL algorithms cope with environment-dependant problems using a dynamic optimization programming approach [38]. Consequently, RL is able to figure out interactions of the application parameters, input contents, and output system- and application-level metrics, and choose accordingly the best strategy at each specific moment to maximize the defined objective or set of objectives. As a result, they are promising solutions for dynamic power and thermal management of multistream HEVC encoding.

As a result, in this work, we adopt a QL algorithm that dynamically determines the best possible encoding configuration and per-core frequency to increase video quality and compression, without encoding time degradation under power and temperature constraints. QL, as a model-free algorithm of RL, makes acting optimally in Markovian domains possible by learning from consequences of the previously taken actions. Compared to other well-known RL algorithms, QL is able to interact with more sophisticated industrial applications [39].

There are several works proposing alternative multi-objective RL algorithms [40], [41], [42]. In particular,

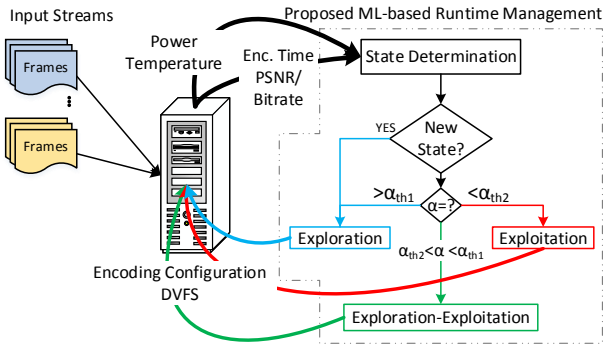


Figure 10. ML-based approach block diagram

Pareto Q-Learning (PQL) [42] provides an alternative multi-objective QL algorithm. However, the proposed PQL algorithm only suits problems with terminal states, where a sequence of actions at one episode of time leads to a final state. In our problem, however, at each decision time, only one action can be applied. Moreover, content variation within the videos, as well as the actions taken for other videos running on the multicore server, can considerably affect the current state of each video, which makes impractical the definition of a terminal state at runtime.

5.1 Q-Learning Background Concepts

The QL model is composed of a finite action set, **A**, a finite state space, **S**, and an agent. The agent acts based on a learned policy,π, which is a mapping from the state space to the action set while taking into account the reward value granted to each state-action pair. This value implies whether, given a state, an action is worth to apply. The QL agent maximizes this reward by storing a $Q^{\pi}(s,a)$ value to represent the quality of each state-action pair in a Q-table. This value demonstrates the most probable long-term reward, considering starting from state, s, applying action a, and following the policy π. The Q-values and the Q-table are updated as follows [38]:

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha(s_t,a_t) \times [R_{t+1} + \gamma \max Q_{t+1}(s_t,a) - Q_t(s_t,a_t)] \quad (1)$$

where $Q_t(s_t,a_t)$ and $Q_{t+1}(s_t,a_t)$ are, respectively, the current and updated Q-values corresponding to a_t action and s_t state, R_{t+1} is the reward observed after a_t is applied for state s_t, $\alpha(s_t,a_t)$ determines the learning rate, and γ is the discount factor and controls the significance of the history of the Q-values against the recently obtained reward.

5.2 Proposed QL-Based Quality-Aware Power and thermal Management

Figure 10 shows a general description of our proposed approach. We consider three phases for the QL algorithm including *exploration*, *exploration-exploitation*, and *exploitation*. In the exploration phase, once the first frame arrives, an action is selected randomly from an action pool, including all available configuration modes and operating frequencies. Therefore, the state transitions from an initial state to a new one observed by the ML agent. Thereafter, the new Q-value corresponding to the selected action and the initial state is calculated.

As the second frame arrives, once again, a random action is selected. However, this time, the action is selected from a subset of the initial action pool. The previous randomly selected action, in fact, blocks some of the configuration modes to be selected, for one or several incoming frames, since their reward is not observed instantly. For instance, when a configuration mode including GOP of size 8 is selected, observing the reward for this specific part of the whole selected configuration mode must be postponed until the 8^{th} frame is encoded.

The *exploration* phase for each pair of state-action continues until the learning rate decreases down to a predefined threshold. We define the learning rate as a function of number of state-action observations as:

$$\alpha(s_t,a_t) = \lambda / Num(s_t,a_t) \quad (2)$$

where λ is constant and set to 0.3, to increase the learning speed. In addition, α_{th1} and α_{th2} are experimentally set to 0.2 and 0.1 respectively. $\alpha(s_t,a_t)$ determines the learning rate and shows if the state-action pair of (s_t,a_t) has been sufficiently observed or not. This is important especially in stochastic environments where applying an action a_t at state s_t may not always result in a specific next state s_{t+1}. On one hand, if it is set to 1, the previous learning process (old data) is overridden and a new learning process starts. On the other hand, when it is set to zero, the agent does not learn. For fully deterministic environments, $\alpha(s_t,a_t) = 1$ provides optimal learning. However, stochastic problems (as discussed in [38]) require a decreasing-to-zero $\alpha(s_t,a_t)$ to achieve an optimal learning phase. Alternatively, $\alpha(s_t,a_t) = 0.1$ is a good compromise, as suggested by [38], and no variations of this parameter are needed. However, as discussed in the paper, to facilitate transitions between our proposed learning phases, we followed the varying learning rate approach, which depends on the number of observations of each state-action pair. Thus, we empirically explored the best values with different video encoding inputs and configurations, and we found $\lambda = 0.3$ and $\alpha(s_t,a_t) = 0.3/1$. These values satisfy convergence conditions [38] and provide quick learning, and are similar to those found by [43] in the literature.

In the *exploration-exploitation* phase, the agent keeps updating the Q-values, while selecting the best possible action.

In the *exploitation* phase, the learning process stops and the agent relies on the obtained rewards and available Q-table to choose the most appropriate action for each state. However, if a new state is observed, the *exploration* phase starts again, but only for this newly observed one. This is, in fact, one of the major advantages of our work when compared to TONE [27]. Since the Pareto optimal curves are extracted statically, TONE is not able to select the best encoding configuration if a different video type affects the system and application states differently. In contrast, learning from new states and actions is an inherent feature of machine learning.

In the following subsections we define the states and available actions, as well as the reward function.

5.3 State Definition

5.3.1 System-Level States

Power consumption. The instant total power consumption of the target multicore server, ranging from static power, P_{static}, and a user-defined power cap, P_{cap}, are split into n_{power} intervals to create the power state subset.

Temperature. Temperature varies from ambient temperature ($\theta_{ambient}$) to critical temperature. In this work, we control the peak temperature of the multicore server and define a temperature constraint (θ_{const}) below the critical temperature (θ_{critic}). The interval between $\theta_{ambient}$ and θ_{const} is divided into n_θ intervals.

Power consumption and per-core temperature data are measured directly from the multicore server (Section 6).

5.3.2 Application-Level States

PSNR. Usual PSNR values for lossy video compression range from 30 dB to 50 dB, for a bit depth of 8 [44]. This range is divided into n_{psnr} intervals to constitute the quality state subset.

Bitrate. The achievable bitrate varies from a few hundreds of kbps to several thousands of kbps. Nonetheless, a target bitrate (BR_{targ}) for each video type is defined based on the required link speed. Thus, the following subset is used in this work: ($\leq 0.75BR_{targ}$), ($0.75BR_{targ}$-BR_{targ}), (BR_{targ}-$1.25BR_{targ}$), and ($\geq 1.25BR_{targ}$).

Encoding time. The current version of reference software (HM 16.3) does not provide real-time encoding. As a result, for different video types and contents the encoding time varies considerably. Therefore, states must be defined regarding a unique reference for each individual video type.

The reference software [32] reports PSNR, bitrate, and time. We use these data to build the application-level states. Moreover, although the overall state set is extremely large, the ML agent does not have to explore all these states, since a great number of them do not occur. For instance, it is not possible that the highest PSNR and the lowest bitrate are observed at the same time for a specific frame.

5.4 Action Pool and Action Set Definition

The action pool proposed in this work consists of the most effective encoding configuration modes in conjunction with the available CPU frequencies. Table 2 shows the design parameters and the corresponding values considered as the available actions to the ML agent.

Even with the constrained action set, there are 684 different combinations of encoding parameters and operating frequencies that can be applied for a single video at a specific state if static profiling approach instead of machine learning is used. In particular, when GOP size is one, QP, CU, and frequency can take all their available values of Table 2 (i.e., 4, 3, and 3, respectively) and there will be no choice for the rest, while if GOP is not one (eight or 16) action could be any combination of the encoding parameters and the available frequencies. Due to content variation within a single video, the application-level and system-level state changes constantly at runtime. This requires profiling every single frame, which means running each frame with all 684 different combinations of actions to figure out the best one. More importantly, configuring the encoding parameters is even more challenging when multiple videos running on a multicore server are taken into account. This is because power, temperature and encoding time objectives, which are the main optimization targets of our work, are considerably affected by operating frequency, and available power and temperature budget. Moreover, when several videos are running on the server, the outcome of one of those 684 different combinations of encoding parameters and frequencies will be strongly affected by the values chosen for other videos, thus an exhaustive profiling is required to include all possible combinations of encoding parameters and frequencies for all encoded videos in parallel. In addition, all combinations of frames that can potentially be running simultaneously should be also profiled carefully.

Table 2
Application and system parameters, and corresponding selected values

QP	22	27	32	37
Search Range	128	64	32	
LCU size	64	32	16	
GOP size	16	8	1	
# of reference frames	4	2	1	
Frequency (GHz)	2.4	2.0	1.8	

Different combinations of system and application parameters may ultimately result in a unique application- and system-level output. Although a few works such as [22] tried to partially model these outputs based on application parameters, these models are very platform-dependent and only take a few encoding parameters. Nevertheless, RL is able to consider any interaction between arbitrary encoding and system parameters on any arbitrary platform as their effect will cumulatively appear in the defined states.

5.5 Reward Function

The proposed reward function must provide a proper feedback from the selected action for a previous state. Since in this work we look for a solution to take bitrate, PSNR, power consumption, temperature and processing time into account, we propose a reward function composed of 5 sub-functions, one for each of these parameters.

The higher compression obtained by HEVC, in comparison with other video encoding standards, is one of the most important features to be maintained. However, the best achievable bitrate differs from one format to another. Thus, we assume a specific target bitrate for each video type. In particular, we propose the following reward function:

$$R_{br} = \begin{cases} -aBR^2 + bBR & BR < BR_{targ} \\ \frac{-c \times BR}{BR_{targ}} + d & BR > BR_{targ} \end{cases} \quad (3)$$

where BR shows the bitrate. The maximum reward is given to $BR = BR_{targ}$ and it degrades faster when the bitrate is larger than the target value. The quadratic part provides a larger difference between the granted rewards from point to point when the attained bitrate is far from the target. This difference decreases as the obtained bitrate approaches the target, letting the agent take into account other reward functions. $a = 1/BR_{targ}^2$ and $b = 2/BR_{targ}$ provide such behavior by making R_{br} local maximum at $BR = BR_{targ}$. The decreasing slope for the linear part is defined by c. We experimentally found c equal to 2 sufficient. Therefore, d is equal to 3 to provide continuity for the reward function.

The reward sub-function corresponding to PSNR is defined as:

$$R_{psnr} = a \times e^{(PSNR/PSNR_{max})} - b \quad (4)$$

where $PSNR_{max}$ is 50 dB as discussed in Section 5.3. Also, a and b are constants and defined such that the maximum value obtained from this sub-function is one while the minimum (assuming $PSNR_{min} = 30\ dB$) is zero. The exponential reward helps getting higher rewards as the PSNR approaches to the maximum value.

In this work, we seek for shorter encoding time and the reward sub-function is proposed as follows:

$$R_T = 1 - T/T_{ref} \quad (5)$$

where T is the encoding time of the frame, and T_{ref} is the reference time (see Section 5.3). T_{ref} is a user-defined value and varies depending on the frame resolution. In this work, we assume 75, 150, and 750 seconds, respectively, for 1280x720, 1920x1080, and 3840x2160 resolutions.

In order to meet the user-defined power cap, the reward function provides a negative value, which is large enough to cancel probable positive rewards attained by other sub-functions, if the constraint is not met (also applied for temperature reward). Higher Q-values are given to those state-action pairs leading to lower power consumption. Hence, the reward sub-function is:

$$R_{power} = \begin{cases} -4 & P > P_{cap} \\ P_{static}/P & P \leq P_{cap} \end{cases} \quad (6)$$

where P is the total power consumption of the multicore server.

The temperature reward sub-function must facilitate preventing any state-action pair resulting in temperature higher than the peak temperature constraint. Thus, similar to [43], we employ a reward sub-function defined as:

$$R_\theta = \begin{cases} -4 & \theta > \theta_{const} \\ e^{(\theta_{ambient} - \theta)} & \theta \leq \theta_{const} \end{cases} \quad (7)$$

When the temperature is below the constraint, the reward exponentially increases as it approaches towards the ambient temperature. Nonetheless, reaching the ambient temperature is ideal and not a goal of this work. Therefore, the values corresponding to this reward function are comparatively small and dominated by other reward functions.

The proposed reward functions for power and temperature only depend on the ambient temperature and the static power consumption which may differ for different systems and environments. However, it does not affect the validity of the proposed approach when the environment changes.

Figure 11 shows the individual reward functions (for $BR_{targ} = 5\ Mbps$ and $T_{ref} = 150\ sec$). Finally, Eq. (8) forms our total reward function.

$$R_{tot} = c_1 R_{br} + c_2 R_{psnr} + c_3 R_\theta + c_4 R_T + c_5 R_{power} \quad (8)$$

The proposed reward function simply sums all the sub-functions without considering interactions among them. First, we recall that although the output PSNR, bitrate, encoding time, as well as the system power and temperature, all vary with changes in the encoding configuration, they are also strongly dependant on the video contents. Hence, modeling the total reward function with the interactions among its sub-functions will only add to the complexity of the reward function, providing only minor gains. Moreover, the interrelation of power and temperature in SoCs, and particularly in multicore servers, is not straightforward, due to heat sharing. As a consequence, we consider both temperature and power independently in the reward sub-functions. Furthermore, the temperature reward function in our formulation affects the total reward function less significantly than what the power does. In fact, the main role of R_θ is taking care of the peak temperature of each individual component, while on the other hand, R_{power} is in charge of total power consumption of the chip. The maximum value of all reward sub-functions is normalized to 1. When the power and temperature constraints are violated, a sufficiently large negative reward is considered so that the corresponding action can be discarded from future decisions. Reward functions are depicted in Figure 11.

Coefficients c_1 to c_5 are introduced to manipulate the effect of each reward sub-function. These constants are in charge of tuning the total reward to emphasize more on a particular sub-function. For our setup, since we are using sub-functions with different behaviours for each reward based on our comprehensive study of the application, this objective is already fulfilled. Thus, we set all these constants equal. With these equal weights, as shown in Section 7.4, the outcome solution is only marginally outperformed by the optimal solution. Yet, we may weigh specific constants more than the others to emphasize a specific objective based on our requirements posed by the system and/or application.

6 Experimental Setup

6.1 Experimental Platform

The proposed runtime power and thermal management approach can be used for any platform architecture and HEVC implementation since it focuses on leveraging application-level parameters. As a result, the underlying architecture or platform, regardless of its type, needs to deal with an optimized application code, which ultimately leads to improved encoding efficiency and time. In this work, we perform the experiments on an Intel S2600GZ server running CentOS 6.5. The server includes a 6-core SandyBridge-EP processor. The platform supports per-core DVFS and a frequency range from 1.4 GHz to 2.0 GHz spaced by 100 MHz, as well as a turbo boost frequency of 2.4 GHz. Our server comes with 32KB instruction and data L1, 256KB private L2, and a 15MB shared L3. We use Intel's Running Average Power Limit (RAPL) to collect power measurements of CPU and DRAM, while Intelligent Platform Management Interface (IPMI) is used to gather CPU temperature sensor measurements, based on the same methodology used in [45].

Our server is equipped with the default PWM-based thermal management mechanism of commercial servers, which keeps fan speed at low speed until 75°C, and then increases it to keep the CPU temperature below this value. Since cooling power is a cubic function of fan speed, we set the CPU thermal constraint equal to 70°C, in order to provide more power saving. In addition, the CPU power constraint is set to 33 watts. Finally, given that the room temperature of our server is fixed at 24°C, we use the following temperature state subset: $(\theta_{ambient}, \frac{\theta_{const}+\theta_{ambient}}{2})$, $(\frac{\theta_{const}+\theta_{ambient}}{2}, \theta_{const})$, and $(\geq \theta_{const})$. We also split the range (P_{static}, P_{cap}) into 5 equal portions. Although all temperature data for the runtime management are gathered

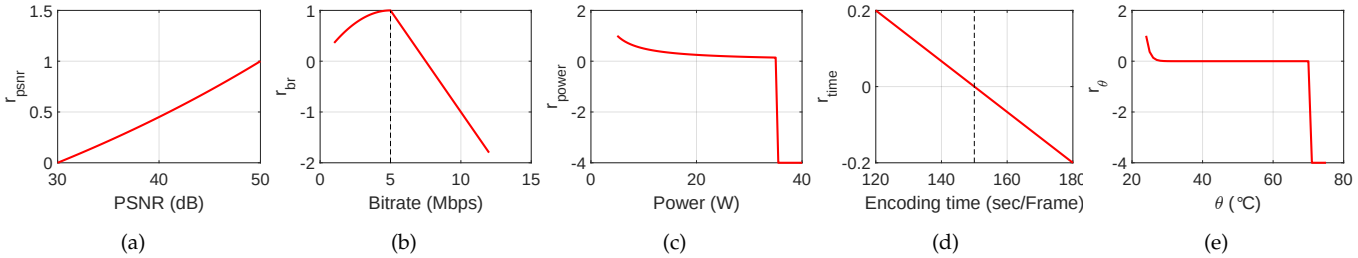


Figure 11. Reward functions of a) PSNR, b) bitrate, c) power, d) encoding time, and e) temperature

via the available per-core thermal sensors, in order to visualize the thermal profile of the chip, we use the power traces measured on the server and an approximate floorplan available online for 6-core SandyBridge processors[2] to feed the 3D-ICE [46] simulator.

Finally, in order to perform per-core DVFS via the OS, we change the governor of the CPU frequency scaling to "*userspace*" via *cpupower* utility. We perform video assignment and migration through *taskset* utility. Moreover, since the predefined power/thermal budgets are lower than those defined in the CPU datasheet[3], the default shutdown mechanism or any other default power/thermal management schemes are not invoked by the OS. Hence, the OS does not intrude our proposed approach.

6.2 Compared Approaches

To make our solution as general as possible, we implement our ML-based approach and TONE [27] on top of the HEVC test model HM 16.3 [32]. HM 16.3 is the reference software for the HEVC standard introduced by Joint Collaborative Team on Video Coding (JCT-VC). It implements all the control parameters allowed by the standard, a feature which is not available in many other implementations. In order to provide a fair comparison with the reference software, with respect to power and temperature, we implement an RL-based power and thermal management approach [47], which outperforms other recent pro-active approaches (such as TAPE [9]), on top of the reference software, and call it HM*.

Although there are several works for power and thermal management of multimedia workloads, we choose TONE for comparisons as it is the only existing thermal management work for HEVC encoders that uses application-level parameters. As we aim at approaching towards a target bitrate, for a fair comparison, we adapt TONE [27] so that it avoids reducing the bitrate far below the target.

6.3 Scenario Definition

In order to provide a better insight in how our ML-based approach is able to dynamically manage the output encoding efficiency and time, we compare it with the HM 16.3 reference software. In this so-called Scenario 0, we study all test sequences while they are running alone with the maximum available core frequency. Since a single video running on our experimental platform does not violate the power and thermal constraints, power and thermal management over the HM is bypassed. Besides, the proposed ML-based approach can now only consider the reward functions of encoding efficiency and time.

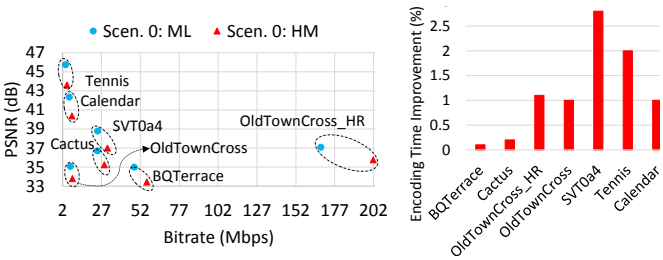


Figure 12. ML vs. default HM in Scenario 0

Thereafter, we evaluate the efficacy of the proposed approach against HM* (see Section 6.2) and TONE [27] in three scenarios. In the first scenario, we assume that all cores are fully utilized by receiving instances of the same video. In the second scenario, we assume a more realistic case where videos are randomly started on cores and finish. In this scenario we assume multiple instances of the same video. In the third scenario, we assume the same scenario as the second one but with different videos. In last two scenarios, in order to assign the videos to the cores we take advantage of the proposed video assignment strategy while DVFS is performed by our ML-based approach. Since in the last two scenarios videos randomly start and finish, we perform the experiments 100 times to obtain statistically significant results. At each run, we consider the average PSNR and bitrate for each instance separately. Finally, the average of all results over these 100 runs are reported as the final obtained gains. In all scenarios, to provide a fair comparison with HM, adaptive search range (ASR) and adaptive QP selection (AdaptiveQP) options are enabled, while the target bitrates (TargetBitrate) are set to those specified in Table 1.

7 Experimental Results

7.1 Evaluation of Encoding Efficiency and Time

In what follows, a scenario-wise discussion on the experimental results is presented. While Figure 12 shows the results of Scenario 0, Figure 13 and Figure 14 show the encoding efficiency and encoding time, respectively, for other scenarios.

2. http://www.anandtech.com/show/5091/intel-core-i7-3960x-sandy-bridge-e-review-keeping-the-high-end-alive

3. https://www.intel.com/content/www/us/en/motherboards/server-motherboards/server-board-s2600gl-gz.html

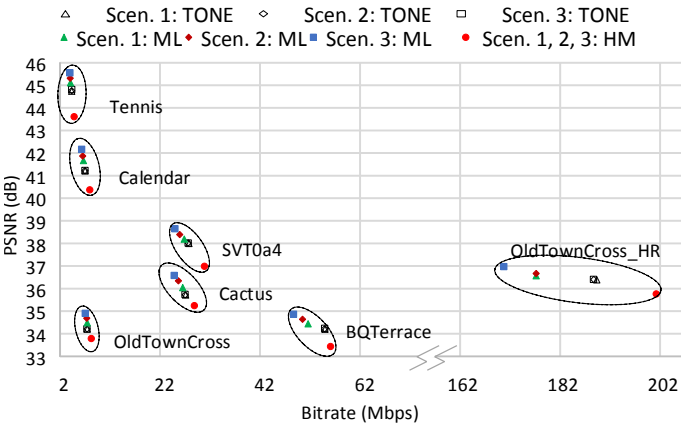


Figure 13. PSNR vs. bitrate achieved by ML, TONE, and HM* for all test sequences and scenarios

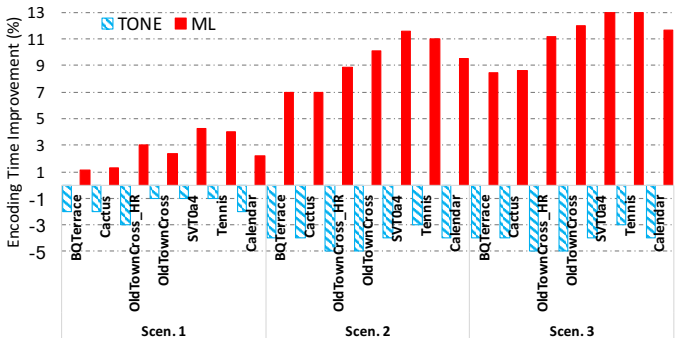


Figure 14. Encoding time of ML and TONE compared to HM*, for all test sequences and scenarios

7.1.1 Scenario 0

Figure 12a, shows the PSNR and bitrate obtained by the default HM and the ML-based approach for different test sequences. As shown in the figure, the proposed approach leads to larger improvements (i.e., shift to the upper left corner) in encoding efficiency when there is more frame-to-frame content variations, as in the case of *Tennis* and *SVT04a* (see Figure 2 to compare content variations within a single video). Such improvements come with a small encoding time enhancement, as shown in Figure 12b. This encoding time improvement comes from the more efficient and intelligent adaptation of encoding parameters compared to when ASR and AdaptiveQP options are enabled in HM. In fact, our ML-based approach is able to more efficiently find the most appropriate encoding parameters based on the contents of the video.

7.1.2 Scenario 1

In the first scenario, where all cores are occupied by instances of the same video, variations between frames result in a great opportunity for reducing the power consumption and, hence, the average temperature of the target multicore server, while improving the encoding efficiency in terms of video quality and compression. The improvement in encoding time compared to that of HM* in Scenario 1 lies in the fact that our ML-based power and thermal management uses DVFS and encoding parameters jointly, while the power and thermal management scheme of HM* leads

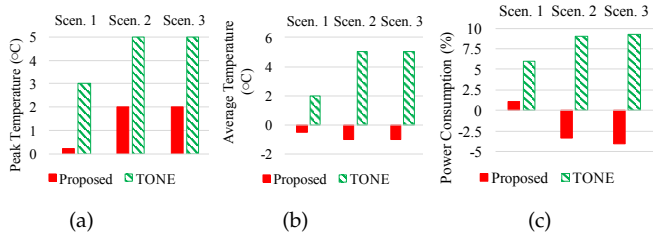


Figure 15. (a) peak temperature, (b) average temperature, and (c) power consumption of the proposed approach (ML) and TONE compared to HM*

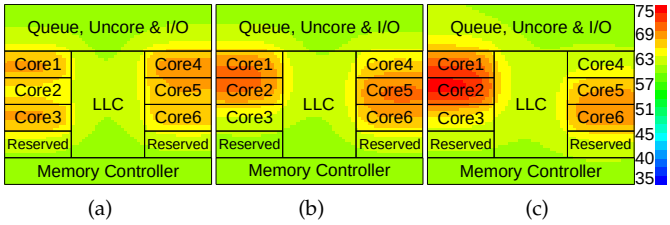


Figure 16. Thermal map (°C) of the third Scenario for (a) ML and video assignment (b) only ML (c) TONE

to application of lower core frequency for some frames. Thus, it is not able to further improve the encoding time by tuning the encoding parameters. TONE, on the other hand, suffers the most from encoding time degradation as neither does it take advantage of intelligent DVFS, nor tune the encoding parameters with respect to the output encoding time. Nonetheless, such an improvement provided by ML comes with the cost of less encoding efficiency compared to that obtained through Scenario 0, since the ML agent compensates the encoding time through adapting the computational complexity of the encoder which ultimately results in PSNR loss and less compression.

7.1.3 Scenarios 2 and 3

The main benefit of ML can be seen in the second and third scenarios, which are closer to the real cases of YouTube and Netflix servers. In Scenario 2, the HM* and TONE, unaware of the available potentials due to changes in the number of videos being processed, fail to improve encoding efficiency. On the contrary, our ML-based approach succeeds to increase the video quality (PSNR), and decreases the bitrate. More importantly, the proposed approach further improves the encoding time in comparison with HM* and TONE due to the same reasons already explained for Scenario 1. In particular, encoding time enhancement comes from more frequent opportunities of intelligently using higher operating frequencies, especially when some videos leave the server. Indeed, thanks to another video leaving the server, the increased available power budget allows other cores run with higher frequency and/or ML agent tunes the encoding parameters of the videos.

The PSNR and compression obtained in these scenarios are higher than those in Scenario 1, although smaller than in Scenario 0. In contrast, as shown in Figure 13, the obtained PSNR and bitrate points by TONE are almost overlapping each other meaning that very marginal improvements in encoding efficiency are obtained.

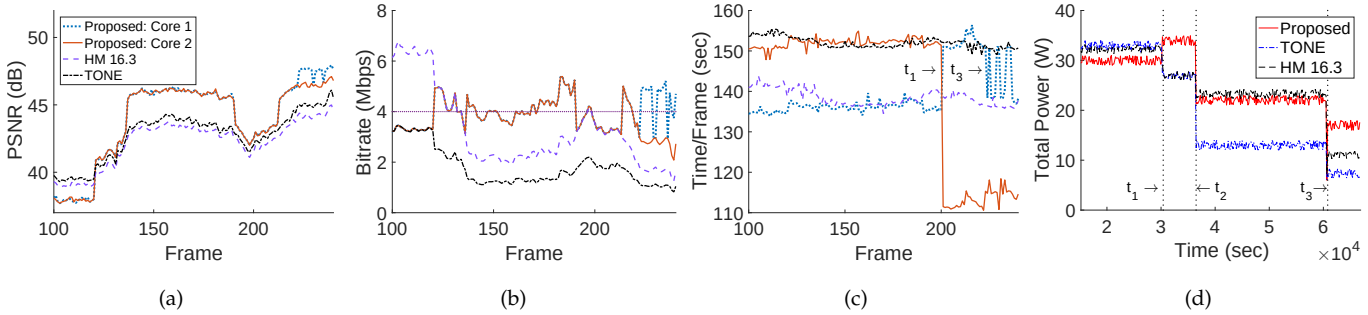


Figure 17. Frame-by-frame results for Tennis: proposed ML-based approach (Core 1 and Core 2) versus TONE and HM 16.3 (the best core)

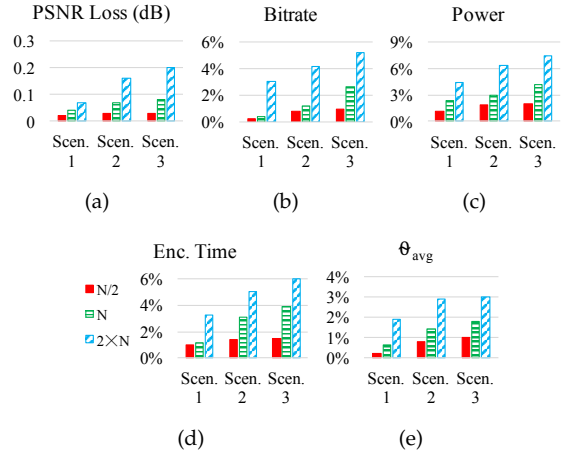


Figure 18. a) PSNR loss (dB), and increase (%) in b) bitrate, c) power, d) encoding time, and e) average temperature when scaling up the decision interval compared to when using N/N interval

The same trend is observed in Scenario 3, where different videos with different contents and resolutions result in more opportunities of well-tuning the encoding parameters. In other words, when the number of videos and their contents running on the multicore server changes, system state changes and, hence, proper decision taken by our ML-based approach leads to improving encoding efficiency and time. However, similar to the previous scenarios, HM* and TONE are unaware of such variations, thus, they fail to achieve any improvement in encoding efficiency.

7.2 Discussion on Power/Thermal Awareness

Figure 15 shows power consumption, average temperature, and peak temperature achieved through our proposed ML-based approach compared to HM*, and to TONE [27], for scenarios 1 to 3. Our proposed ML-based approach is able to reduce the average temperature for the second and the third scenarios compared to HM* and TONE. However, while peak temperature decreases compared to TONE, the peak temperature reduction is not as large as the reduction in the average temperature. This is because the peak temperature occurs mostly due to a rapid change in the video contents, or mainly in the second and the third scenarios, because of releasing an additional video on the target multicore server, resulting in a spike in the temperature.

Finally, Figure 16 shows the thermal profile of the chip at specific time for the third Scenario. At this given moment, the total power consumption obtained from TONE, ML-based approach with, and without the proposed video assignment are all at the maximum level. As shown in Figure 16, the best thermal profile, in terms of lower temperature and less thermal gradients is achieved when the proposed ML-based approach is accompanied by the resolution-aware video assignment. This implies that although runtime management of multistream encoding on the multicore server is necessary, a proper thermal-aware video assignment is also vital. In particular, our proposed video assignment and migration strategy leads to peak temperature reduction. Such a thermal profile eventually provides more opportunities for the QL agent to increase encoding time efficiency.

7.3 Frame-by-Frame Evaluation of the ML-based Approach

Figure 17 illustrates the second Scenario, frame-by-frame, for *Tennis* where some videos leave the server. For the sake of clarity, the behavior of only 2 cores is shown for our approach, while only the curve related to the core with the best behavior is depicted for TONE and HM 16.3. At time t_1 (see Figure 17d), one video leaves the server. TONE and HM 16.3 suffer from significant PSNR loss and deviation from target bitrate (Figure 17a and b). Our approach, however, instantly reacts and keeps the same power state by increasing the frequency of other cores to increase the encoding time (Figure 17c). In addition, it finds an opportunity to exploit other more efficient encoding configurations. The same happens at t_2 and t_3, respectively. Our ML-based approach improves the encoding time on core 1 and 2 during the whole encoding process by 10% and 27%, respectively. As these results show, our ML-based solution outperforms TONE [27], especially when the temperature is below its constraint. This is due to the fact that TONE [27] starts seeking for an appropriate encoding configuration only if the temperature exceeds the threshold. However, when only few cores are active, temperature drops, hence, granting an opportunity to our approach for increasing the frequency and/or using another appropriate encoding configuration.

7.4 Overhead and performance of the ML-based approach

In this work, although we deal with a multi-objective problem, we use a scalarization function over the predefined reward sub-functions. Despite its simplicity, scalarization works well for our specific target problem and the defined

reward functions, and our solution is very close to the optimal one. As an example, for the *Tennis* test sequence in Scenario 2, our ML-based solution provides the optimal solution for 226 frames out of 240 total frames. Even for the rest 14 frames, the Euclidean distance of our solution from the optimal one is in order of 10^{-3}. Moreover, similar results are obtained in our experiments with all the other considered test sequences and scenarios.

The proposed ML-based approach does not have to be applied on a per-frame basis due to the fact that video contents, in spite of their constant variation, are rarely changing rapidly between two consequent frames. Hence, based on the frame rate of each running video, our proposed ML-based approach can be applied at different intervals to achieve less runtime overhead without degrading the fulfillment of our goals. Figure 18 shows how scaling the intervals of applying our proposed ML-based approach influences the encoding efficiency, time, power consumption, and average temperature during the exploitation phase compared with the per-frame basis (N/N). In this Figure, N is the frame rate when denoted as N Hz (such as 24 Hz, thus, the evaluated intervals are equal to 12, 24, and 48). As shown in Figure 18, there is slight degradation in the achieved PSNR, deviation from bitrate, power saving, average temperature, and encoding time when using larger decision interval. However, this degradation differs between the scenarios. For the first scenario, where the changes in the defined states only depend on the variation of the video content, increasing the decision intervals to $2 \times N$ leads to negligible degradation. On the contrary, in the second scenario, the defined states change as a result of changes in the number of running videos. In addition, the different videos used in the third scenario add to the dynamism of the states. Thus, more noticeable degradation is observable for these two scenarios. For a fair comparison, we did not apply our proposed video assignment strategy for the second and third scenarios when evaluating the impact of decision intervals.

Our ML-based approach is limited by the sensor polling frequency, the availability of the application statistics, the readings from the Q-table, and updating it in *exploration* and *exploration-exploitation* phases. Updating the Q-table, however, consists of simple arithmetic operations, resulting in negligible overhead. Moreover, the proposed ML-based approach is used as a runtime power and thermal manager once it reaches the *exploitation phase*. Thus, these two phases are performed offline except when a new state is observed in the *exploitation* phase. As explained in Section 5, the ML agent has to select a random action in order to further explore this state. On average, 820 frames are required for each resolution (since we keep one Q-table per video resolution, as it has its own target bitrate and reference encoding time) after which the optimal decision is known for more than 96% of the observable states.

8 Discussion and Conclusion

In this work, we have presented a comprehensive quality-aware power and thermal management approach for next generation video coding which takes video quality, encoding time and compression into account. Our ML-based approach is the first work that utilizes application parameters together with DVFS, based on frame-to-frame variations between different videos and within a video for multicore servers. We also proposed a resolution-aware video assignment strategy which allowed for reduced encoding time as a result of reduced hot spots.

We evaluated the proposed approach against the state-of-the-art [27] on an enterprise multicore server, under different scenarios simulating YouTube and Netflix servers. Overall, the proposed approach outperformed TONE [27] mainly due to its awareness of the content variation within and across videos, and inaccuracy of the temperature prediction proposed in TONE for multicore servers. All in all, although our ML-based approach considered five objectives and constraints, two more than TONE, since in all scenarios all objectives have improved, the results would be consistently better if fewer objectives (the same as TONE) are considered. In such a case, we expect that our proposed approach further outperforms TONE. On average, for the most realistic scenario, compared to TONE [27], our ML-based approach improved BD-PSNR and BD-rate [48] by 0.54 dB, and 8%, respectively, and reduced the encoding time, power consumption, and average temperature by 15.3%, 13%, and 10%, respectively. Moreover, our approach improved BD-PSNR and BD-rate compared to HM 16.3 by 1.19 dB and 24%, respectively, without any encoding time degradation, when power and temperature constraints were relaxed.

References

[1] Cisco Systems, Inc., "Cisco visual networking index: Forecast and methodology 2015-2020. cisco whitepaper." 2016.

[2] Sandvine, Inc., "Global internet phenomena report," 2013.

[3] J. V. Team, "Advanced video coding for generic audiovisual services," *ITU-T Rec. H*, vol. 264, pp. 14 496–10, 2003.

[4] F. Bossen, B. Bross *et al.*, "HEVC complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, 2012.

[5] (2017) DMR youtube report. [Online]. Available: http://expandedramblings.com/index.php/youtube-statistics/#

[6] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal processing magazine*, vol. 20, no. 2, pp. 18–29, 2003.

[7] A. Iranfar, M. Zapater, and D. Atienza, "A machine learning-based approach for power and thermal management of next-generation video coding on mpsocs," in *CODESS+ISSS*, 2017.

[8] A. Bartolini, M. Cacciari *et al.*, "Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 170–183, 2013.

[9] T. Ebi, A. Faruque *et al.*, "Tape: thermal-aware agent-based power economy for multi/many-core architectures," in *Proceedings of the 2009 International Conference on Computer-Aided Design*. ACM, 2009, pp. 302–309.

[10] C. Jiang and S. Nooshabadi, "Parallel multiview video coding exploiting group of pictures level parallelism," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 8, pp. 2316–2328, 2016.

[11] X. Li, M. A. Salehi *et al.*, "Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[12] J. V. Team, "Advanced video coding for generic audiovisual services," *ITU-T Recommendation H. 264 & ISO/IEC 14496-10*, 2005.

[13] H. Kim and Y. Altunhasak, "Low-complexity macroblock mode selection for h. 264-AVC encoders," in *Image Processing. ICIP'04. Int. Conf. on*, vol. 2. IEEE, 2004, pp. 765–768.

[14] D. S. Turaga, M. van der Schaar, and B. Pesquet-Popescu, "Complexity scalable motion compensated wavelet video encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 8, pp. 982–993, 2005.

[15] E. Raffin *et al.*, "Low power HEVC software decoder for mobile devices," *Journal of Real-Time Image Processing*, pp. 1–13, 2015.

[16] G. Correa, P. Assuncao *et al.*, "Complexity scalability for real-time HEVC encoders," *Journal of Real-Time Image Processing*, vol. 12, no. 1, pp. 107–122, 2016.

[17] G. Tian and S. Goto, "Content adaptive prediction unit size decision algorithm for HEVC intra coding," in *Picture Coding Symp.* IEEE, 2012, pp. 405–408.

[18] M. Shafique, M. U. K. Khan, and J. Henkel, "Power efficient and workload balanced tiling for parallelized high efficiency video coding," in *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1253–1257.

[19] M. U. K. Khan, M. Shafique, and J. Henkel, "Software architecture of high efficiency video coding for many-core systems with power-efficient workload balancing," in *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2014, p. 219.

[20] G. Corrêa, P. Assuncao *et al.*, "Complexity control of high efficiency video encoders for power-constrained devices," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1866–1874, 2011.

[21] G. Correa, P. Assuncao *et al.*, "Dynamic tree-depth adjustment for low power HEVC encoders," in *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*. IEEE, 2012, pp. 564–567.

[22] M. U. K. Khan, M. Shafique, and J. Henkel, "Power-efficient workload balancing for video applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2089–2102, 2016.

[23] D. Zhou, L. Guo *et al.*, "Reducing power consumption of hevc codec with lossless reference frame recompression," in *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2120–2124.

[24] Y. He, M. Kunstner *et al.*, "Power aware hevc streaming for mobile," in *Visual Communications and Image Processing (VCIP), 2013*. IEEE, 2013, pp. 1–5.

[25] Z. Ma and A. Segall, "Frame buffer compression for low-power video coding," in *IEEE Int. Conf. on Image Processing*. IEEE, 2011, pp. 757–760.

[26] D. Palomino, M. Shafique *et al.*, "HEVCDTM: Application-driven dynamic thermal management for high efficiency video coding," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE, 2014, pp. 1–4.

[27] ——, "TONE: Adaptive temperature optimization for the next generation video encoders," in *Proceedings of the 2014 international symposium on Low power electronics and design*. ACM, 2014, pp. 33–38.

[28] ——, "Thermal optimization using adaptive approximate computing for video coding," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*. IEEE, 2016, pp. 1207–1212.

[29] M. Shafique and J. Henkel, "Low power design of the next-generation high efficiency video coding," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*. IEEE, 2014, pp. 274–281.

[30] M. Shafique, B. Molkenthin, and J. Henkel, "An hvs-based adaptive computational complexity reduction scheme for h. 264/AVC video encoder using prognostic early mode exclusion," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2010, pp. 1713–1718.

[31] F. Bossen and H. Common, "test conditions and software reference configurations, jct-vc doc," *L1100, Jan*, 2013.

[32] P. Bordes, P. Andrivon *et al.*, "Joint collaborative team on video coding (JCT-VC) of itu-t sg 16 wp 3 and iso/iec jtc 1/sc 29/wg 11," 2016. [Online]. Available: https://HEVC.hhi.fraunhofer.de

[33] F. De Simone, L. Goldmann *et al.*, "Performance analysis of vp8 image and video compression based on subjective evaluations," in *Applications of Digital Image Processing XXXIV*, vol. 8135. International Society for Optics and Photonics, 2011, p. 81350M.

[34] A. S. Motra, A. Gupta *et al.*, "Fast intra mode decision for hevc video encoder," in *Software, Telecommunications and Computer Networks (SoftCOM), 2012 20th International Conference on*. IEEE, 2012, pp. 1–5.

[35] T. K. Tan, R. Weerakkody *et al.*, "Video quality evaluation methodology and verification testing of hevc compression performance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 76–90, 2016.

[36] J. Levon and P. Elie, "Oprofile: A system profiler for linux," 2004.

[37] V. K. Adhikari, Y. Guo *et al.*, "Measurement study of netflix, hulu, and a tale of three cdns," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 6, pp. 1984–1997, 2015.

[38] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[39] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[40] P. Vamplew, J. Yearwood *et al.*, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2008, pp. 372–378.

[41] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Hypervolume-based multi-objective reinforcement learning," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2013, pp. 352–366.

[42] K. Van Moffaert and A. Nowé, "Multi-objective reinforcement learning using sets of pareto dominating policies," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3483–3512, 2014.

[43] A. Iranfar, S. N. Shahsavani *et al.*, "A heuristic machine learning-based algorithm for power and thermal management of heterogeneous MPSoCs," in *Low Power Electronics and Design (ISLPED), 2015 IEEE/ACM International Symposium on*. IEEE, 2015, pp. 291–296.

[44] S. T. Welstead, *Fractal and wavelet image compression techniques*. SPIE Optical Engineering Press, Bellingham, WA, 1999.

[45] J. C. Salinas-Hilburg, M. Zapater *et al.*, "Unsupervised power modeling of co-allocated workloads for energy efficiency in data centers," in *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, 2016, pp. 1345–1350.

[46] A. Sridhar, A. Vincenzi *et al.*, "3D-ICE: Fast compact transient thermal modeling for 3d ics with inter-tier liquid cooling," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2010, pp. 463–470.

[47] T. Ebi, D. Kramer *et al.*, "Economic learning for thermal-aware power budgeting in many-core architectures," in *Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. ACM, 2011, pp. 189–196.

[48] G. Bjontegarrd, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.

Arman Iranfar (S'17) received the B.S. degree in Electrical engineering from Isfahan University of Technology, Iran, in 2013 and the M.S. degree in Electrical Engineering, circuits and systems from the University of Tehran, Iran. He is currently pursuing the Ph.D. degree in Electrical Engineering in Swiss Federal Institute of Technology Lausanne (EPFL). His research interests include reliability and power and temperature management of multicore servers.

Marina Zapater (M'16) is currently is a Post-Doctoral researcher in the Embedded Systems Laboratory (ESL) at the Swiss Federal Institute of Technology in Lausanne (EPFL). She was non-tenure track Assistant Professor in the Computer Architecture Department at Universidad Complutense de Madrid, Spain, in the academic year 2015-2016. She received her Ph.D. degree in Electronic Engineering from Universidad Politecnica de Madrid in 2015, an M.Sc. in Telecommunication Engineering degree and a M.Sc. in Electronic Engineering degree, both from the Universitat Politecnica de Catalunya, in 2010. Her research interests include proactive and reactive thermal and power optimization of complex heterogeneous systems, energy efficiency in data centers, ultra-low power architectures and embedded systems. In this area, she has co-authored over 30 publications in top-notch international conferences and journals, and she has participated in several national and international research projects. She is a member of IEEE and CEDA and has served as TPC member of several conferences, including VLSI-SoC and MCSoC.

David Atienza (M'05-SM'13-F'16) is associate professor of electrical and computer engineering, and director of the Embedded Systems Laboratory (ESL) at the Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland. He received his PhD in computer science and engineering from UCM, Spain, and IMEC, Belgium, in 2005. His research interests include system-level design and thermal-aware optimization methodologies for 2D/3D high-performance multi-processor system-on-chip (MPSoC) and ultra-low power system architectures for wireless body sensor nodes. He is a co-author of more than 250 papers in peer-reviewed international journals and conferences, several book chapters, and seven patents. Dr. Atienza received an ERC Consolidator Grant in 2016, the IEEE CEDA Early Career Award in 2013, the ACM SIGDA Outstanding New Faculty Award in 2012, and a Faculty Award from Sun Labs at Oracle in 2011. He served as DATE 2015 Program Chair and DATE 2017 General Chair. He is an ACM Distinguished Member and an IEEE Fellow.