

Integrated Power Management for Video Streaming to Mobile Handheld Devices

Radu Cornea, Shivajit Mohapatra, Nikil Dutt, Alex Nicolau & Nalini Venkatasubramanian

Dept. of Information & Computer Science

University of California, Irvine, CA 92697-3425

{radu,mopy,dutt,nicolau,nalini}@ics.uci.edu

Technical Report #03-19

Dept. of Information and Computer Science

University of California, Irvine, CA 92697, USA

May 2003

Abstract

Optimizing user experience for streaming video applications on handheld devices is a significant research challenge. In this paper, we propose an integrated power management approach that unifies low level architectural optimizations (CPU, memory, register), OS power-saving mechanisms (Dynamic Voltage Scaling) and adaptive middleware techniques (admission control, optimal transcoding, network traffic regulation). Specifically, we identify interaction parameters between the different levels and optimize them to significantly reduce power consumption. With knowledge of device configurations, dynamic device parameters and changing system conditions, the middleware layer selects an appropriate video quality and fine tunes the architecture for optimized delivery of video. Our performance results indicate that architectural optimizations that are cognizant of user level parameters (e.g. transcoded video quality) can provide energy gains as high as 57.5% for the CPU and memory. Middleware adaptations to changing network noise levels can save as much as 70% of energy consumed by the wireless network interface. Furthermore, we demonstrate how such an integrated framework, that supports tight coupling of inter-level parameters can enhance user experience on a handheld substantially.

Contents

1	Motivation	5
2	System Architecture	7
2.1	Modeling User Perception on Handheld Computers	8
3	Architectural Tuning for Improved Multimedia Streaming	10
3.1	Hardware-level Knobs for Handheld Devices	10
3.2	Quality-driven Cache Reconfiguration	12
3.3	Integrated Dynamic Voltage Scaling	14
4	Architecture-Aware Middleware Adaptation	15
4.1	Energy-Aware Admission Control & Stream Transformations	15
4.2	Network Traffic Regulation	16
5	Performance Evaluation	18
5.1	Experimental Setup	18
5.2	Experimental Results	19
6	Related Work	24
A	Other Experimental Results	27
A.1	Cache Exploration Results Before Applying DVS	27
A.1.1	<i>Action</i> Clips	27
A.1.2	<i>News</i> Clips	28
A.2	Cache Exploration Results After Applying DVS	29
A.2.1	<i>Action</i> Clip	29
A.2.2	<i>News</i> Clip	30

List of Figures

1	Conjunctive low-level and high-level adaptations for optimizing Power & Performance of Streaming Video to Mobile handheld computers	6
2	System Model	7
3	Main Components of a Handheld Device (a) and CPU Detail (b)	10
4	Internal Relative Power Distribution on the CPU During MPEG Decoding .	11
5	Cache Energy Variation on Size and Associativity	13
6	Admission Control	16
7	Wireless Network	17
8	Setup for Power Profiling	18
9	Cache Optimization for a High Quality, <i>Action</i> Clip	20
10	Cache Optimization for a High Quality, <i>News</i> Clip	20
11	Search Space for a Medium Quality <i>Action</i> Clip, no DVS	21
12	Search Space for a Medium Quality <i>Action</i> Clip, after DVS	21
13	Finding Optimal Burst Time	23
14	Optimal Burst Times	23
15	Power vs. Noise	23
16	Utility Factor over time	24
17	UF2	24
18	Cache Exploration Results for High Quality (Q1, Q2, Q3), <i>Action</i> Clips . .	27
19	Cache Exploration Results for Medium Quality (Q4, Q5, Q6), <i>Action</i> Clips	27
20	Cache Exploration Results for High Quality (Q1, Q2, Q3), <i>News</i> Clips . . .	28
21	Cache Exploration Results for Medium Quality (Q4, Q5, Q6), <i>News</i> Clips .	28
22	Cache/DVS Exploration Results for High Quality (Q1, Q2, Q3), <i>Action</i> Clips	29
23	Cache/DVS Exploration Results for Medium Quality (Q4, Q5, Q6), <i>Action</i> Clips	29
24	Cache/DVS Exploration Results for High Quality (Q1, Q2, Q3), <i>News</i> Clips	30
25	Cache/DVS Exploration Results for Medium Quality (Q4, Q5, Q6), <i>News</i> Clips	30

List of Tables

1	Energy-Aware Transformations for Compaq Ipaq 3650 with bright backlight, Cisco 350 Series Aironet WNIC card, for the Grand Theft Auto action video(encoded using MPEG-1), using Pocket Video player(CE, HTTP streaming) and VideoLan client(Linux, UDP streaming).	9
2	Architectural Configurations for Ideal Energy and Performance Gains (<i>Action Clip</i>)	22
3	Architectural Configurations for Ideal Energy and Performance Gains (<i>News Clip</i>)	22
4	Optimal network video burst lengths(in secs) and corresponding power gains for different quality and noise levels for the Grand Theft Auto action video, assuming sufficient buffer available at client and network packet size of 500KB	24

1 Motivation

Rapid advances in processor and wireless networking technology are ushering in a new class of multimedia applications (e.g. video streaming) for mobile handheld devices. These devices have modest sizes and weights, and therefore inadequate resources - lower processing power, memory, display capabilities, storage and limited battery lifetime as compared to desktop and laptop systems. Multimedia applications on the other hand have distinctive Quality of Service(QoS) and processing requirements which tend to make them extremely resource-hungry. In addition, the device specific attributes(e.g form factor) significantly influence the human perception of multimedia quality. Therefore, delivering high quality multimedia content to mobile handheld devices, while preserving their service lifetimes remain competing design requirements. This innate conflict introduces key research challenges in the design of multimedia applications, intermediate adaptations and low-level architectural improvements of the device.

Recent years have witnessed researchers aggressively trying to propose and optimize techniques for power and performance trade-offs for realtime applications. Several interesting solutions have been proposed at various computational levels - system cache and external memory access optimization, dynamic voltage scaling(DVS) [17, 14], dynamic power management of disks and network interfaces, efficient compilers and application/middleware based adaptations [15, 16]. However, an interesting disconnect is observed in the research initiatives undertaken at each level. Power optimization techniques developed at each computational level have remained seemingly independent of the other abstraction hierarchy levels, potentially missing opportunities for substantial improvements achievable through cross-level integration. Noticeably, the joint performance of an aggregation of techniques at various levels has received relatively little interest. The cumulative power gains observed by aggregating techniques at each stage can be potentially significant; however, it also requires a study of the trade-offs involved and the customizations required for unified operation. For example, decisive middleware/OS based adaptations are possible if low-level information(e.g optimal register file sizes & cache configurations) is made available; similarly low level architectural components(e.g CPU registers, caches etc.) can be optimized if the architecture is cognizant of higher-level details such as specific video encoding. Fig. 1 presents the different computation levels in a typical handheld computer and indicates the cross layer interactions for optimal power and performance deliverance.

The purpose of our study is to develop and integrate hardware based architectural optimization techniques with high level operating system and middleware approaches (Fig. 1), for improvements in power savings and the overall user experience, in the context of video streaming to a low-power handheld device. Multimedia applications heavily utilize the biggest power consumers in modern computers: the *CPU*, the *network* and the *display*(see Fig. 1). Therefore, we aggregate hardware and software techniques that induce power savings for these resources. To maximize power gains for a CPU architecture, we identify the predominant internal units of the architecture that contribute to power consumption. We use higher-level knowledge such as quality and encoding parameters of the video stream to

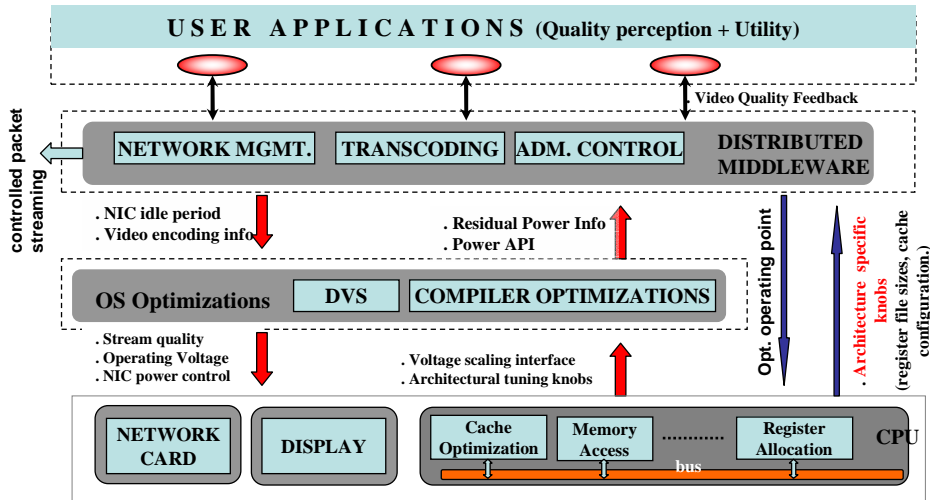


Fig. 1. Conjunctive low-level and high-level adaptations for optimizing Power & Performance of Streaming Video to Mobile handheld computers

optimize internal cache configurations, CPU registers and the external memory accesses. Further we study the trade-offs of using DVS alongside the other optimizations. Knowledge of the underlying architectural configuration is used by the compiler to generate code that compliments the optimizations at the low-level architecture.

Similarly, we utilize hardware/design level data(e.g cache config.) and user-level information(video quality perception) to optimize middleware and OS components for improved performance and power savings - through effective video transcoding, power-aware admission control and efficient network transmission. We reduce the power consumption of the network card by switching it to the “sleep” mode during periods of inactivity. An efficient middleware is used to control network traffic for optimal power management of the network interface. To maximize user experience, we conduct extensive tests to study video quality and power trade-offs for handheld computers. We use these results to drive our optimization efforts at each computing level.

Research Contributions

In this paper, we address the aforementioned challenge of integrating techniques at different levels(hardware, OS, middleware, “User”) through a multi-phase approach. Specifically, we (i) identify low-level architectural tuning knobs combined with compilation techniques for optimizing CPU performance; optimal operating points are then identified for video streams of specific quality levels using these knobs. we study the tradeoffs involved in performing DVS along with our low-level optimization methods. (ii) present a feedback-based middleware for power-aware admission control, quality and power-supported video transcoding.

(iii) extensively study power vs. quality tradeoffs in the context of handheld computers; we combine our extensive survey results on user perception of video quality on handheld computers with our experiences to drive our optimization decisions at each level, (iv) evaluate the power gains of the wireless network interface using an adaptive middleware technique for a typical network with multiple users(noise). Finally, we evaluate the performance of the integrated approach in improving the overall user experience(satisfaction) in the context of streaming video to handheld computers.

Our performance results indicate that when optimized for discrete quality levels, our architectural optimizations saved as much as 47.5% to 57.5% energy for the CPU and memory. Additionally, our adaptive middleware technique yielded 60%-78% savings in the power consumption of the network interface card. By integrating the above techniques we were able to enhance the user experience substantially.

2 System Architecture

We assume the system model depicted in Fig. 2. The system entities include a multimedia server, a proxy server that utilizes a directory service, a rule base for specific devices and a video transcoder, an ethernet switch, the wireless access point and users with low-power wireless devices. The circles represent the noise at the access point due to network traffic introduced by “other” users. The multimedia servers store the multimedia content and stream videos to clients upon receipt of a request. The users issue requests for video streams on their handheld devices. All communication between the handheld device and the servers are routed through the proxy server, that can transcode the video stream in realtime.

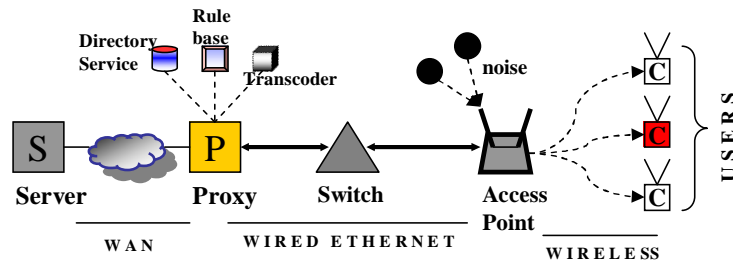


Fig. 2. System Model

The middleware executes on both the handheld device and the proxy, and performs two important functions. On the device, it obtains residual energy availability information from the underlying architecture and feeds it back to the proxy and relates the video stream parameters and network related control information to lower abstraction layers. On the proxy, it performs a feedback based power aware admission control and realtime transcoding of the video stream, based on the feedback from the device. It also regulates the video transmission over the network based on the noise level and the video stream quality. Additionally,

the middleware exploits dynamic global state information(e.g mobility info, noise level etc.) available at the directory service and static device specific knowledge (architecture, OS, video quality levels) from the static rule base, to optimally perform its functions. The rate at which feedbacks are sent by the device is dictated by administrative policies like periodic feedback etc.. Moreover we assume that network connectivity is maintained at all times.

2.1 Modeling User Perception on Handheld Computers

In order to achieve an optimal balance between power and performance, we introduce a notion of “*Utility Factor U_F* ” for a system, and try to optimize the U_F for the system. This approach precludes the system from aggressively optimizing for power at the expense of performance and vice-versa; thereby providing an optimized operating point for the system at all times. Under this strategy, the system tries to utilize all the available energy to maximize the user experience. U_F is a measure of “user satisfaction” and we specify it as follows: given the residual energy E_{res} on a handheld device, a threshold video quality level ($Q_{MAX} < Q_A < Q_{MIN}$), and the time of the video playback T , the U_F of the system is non-negative, if the system can stream the highest possible quality of video to the user such that the time, quality and the power constraints are satisfied; otherwise U_F is negative. Let P_{VID} denote the average power consumption rate of the video playback at the handheld and Q_{PLAY} be the quality of video streamed to the user by the system. Using the above notation, we define U_F as follows:

$$U_F = \begin{cases} Q_{MAX} - Q_{PLAY} & \text{IFF } P_{VID} * T < E_{RES} \\ Q_{PLAY} \geq Q_A & \\ -1 & \text{Otherwise} \end{cases}$$

In order to maintain an acceptable U_F for the system, it is important to understand the notion of video quality for a handheld device and its implications on power consumption. Though this is not the primary focus of our work, we extensively studied user perception of video quality for handheld devices. Video applications introduce the notion of human perception of video quality as an important measure of performance. Moreover, user perception of quality is significantly influenced by the environment and the viewing device(e.g PDA) [1]. These factors make objective assessment [2] of video quality extremely hard and subjective assessment [1] still remains the primary method of video quality appraisal. To validate our assessments, we conducted an extensive survey to subjectively assess the human perception of video quality on handhelds. We tried to follow the recommendations in [1] for our assessment techniques. We showed our subjects various videos(action, news, sports etc.) encoded at 12 different quality levels streamed onto a handheld(iPAQ). We also recorded the average overall energy consumption of the handheld for viewing the streams. The same subjects were also shown the same videos on a laptop. Based on our experiences and the results of our extensive survey, we present the following interesting observations and conclusions:

- *Almost 90% of the subjects were able to differentiate between close video quality levels*

Quality	Transformation Parameters	Avg. Power (Windows CE)	Avg. Power (Linux)
Like Original (No improvement required)	SIF, 30fps, 650Kbps	4.42 W	6.07 W
Excellent	SIF, 25fps, 450Kbps	4.37 W	5.99 W
Very Good	SIF, 25fps, 350Kbps	4.31 W	5.86 W
Good	HSIF, 24fps, 350Kbps	4.24 W	5.81 W
Fair	HSIF, 24fps, 200Kbps	4.15 W	5.73 W
Poor	HSIF, 24fps, 150Kbps	4.06 W	5.63 W
Bad	QSIF, 20fps, 150Kbps	3.95 W	5.5 W
Terrible (poorer quality not acceptable)	QSIF, 20fps,100kbps	3.88 W	5.38 W

Table 1. Energy-Aware Transformations for Compaq Ipaq 3650 with bright backlight, Cisco 350 Series Aironet WNIC card, for the Grand Theft Auto action video(encoded using MPEG-1), using Pocket Video player(CE, HTTP streaming) and VideoLan client(Linux, UDP streaming).

on laptop/desktop systems; however only about 20% were able to differentiate between close quality levels on a handheld computer.

- *It was hard to programmatically identify video quality parameters(a combination of bit rate, frame rate and video resolution) that produced a user perceptible change in video quality and/or a noticeable shift in power consumption.*
- *The manner of video display (auto, nominal, fit screen(stretching)) has a an impact on power consumption for the lowest quality streams but is insignificant for high quality streams. Almost all users preferred to watch the video in “auto” mode that preserved the video resolution.*
- *For all the video streams to handheld devices, it was enough to use just three standard intermediate formats(e.g SIF(320x240), Half SIF(340x160) and Quarter SIF(160x120)) for frame resolution values. Other resolutions did not produce a perceptible quality change or power uptake compared to the nearest SIF encoded video with similar bit and frame rates. For videos of resolution higher than SIF, the player automatically resized the video with the power consumption being similar to videos encoded with SIF and nearest frame rate and bit rates.*

Based on these conclusions, we identify the eight dynamic video stream transformation parameters (Table 1) for our proxy-based realtime transcoding and use the profiled average power consumption values to perform a high-level(coarse) power aware admission control for the system. Note that similar device specific transformations can be made for other portable computers. More importantly, we also optimize our low-level architecture based on these discrete video quality levels. This approach provides us with two significant advantages: (i) Real time stream quality transformations can be performed with no overheads of

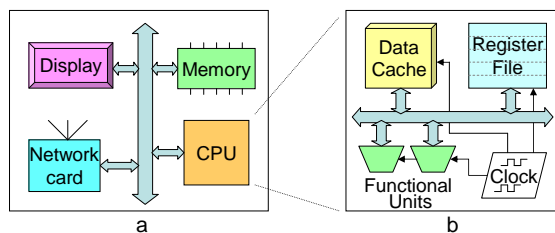


Fig. 3. Main Components of a Handheld Device (a) and CPU Detail (b)

dynamically determining quality degradation parameters, (ii) using the architectural tuning, optimized operating points can be pre-determined for a particular video stream quality. With the knowledge of the stream qualities, low-level cpu voltage scaling can also be optimized. The following sections describe the architectural and middleware optimizations that are integrated into our system.

3 Architectural Tuning for Improved Multimedia Streaming

Hardware design techniques that identify multiple modes of operation and dynamic reconfiguration can be employed to attain high power and performance gains at the CPU architecture level. In order to optimize the architecture for delivering optimal energy performance, we first identify the low level functional units that have maximal impact on power consumption. Furthermore, for video applications, some architectural components are more amenable for power improvements than others. We focus on these components for architectural level fine tuning, in the context of MPEG video applications; we identify "knobs" for these components that can be made available to the higher abstraction levels for dynamically tuning the hardware for MPEG video applications.

3.1 Hardware-level Knobs for Handheld Devices

As shown on Fig. 3(a), there are three major sources of power consumption in a handheld device (e.g. iPaq): display (around 1W for full backlight), network hardware (1.4W) and CPU/memory (1-3W, with the additional board circuits). Each of these subsystems expose ways for controlling the power dissipation. In case of the display (LCD), the main energy drain comes from the backlight, which is a predefined user setting and therefore has a limited degree of controllability by the system (without affecting the final utility). The network interface allows for efficient power savings if cognizant of the higher level protocol's behavior and will be explored in a subsequent section. Out of the three components mentioned above, the CPU coupled with the memory subsystem poses the biggest challenge. Its intrinsic high dependence on the input data to be processed, the quality of the code generated by

the compiler and the organization of its internal architecture make predicting its power consumption profile very hard in general; nevertheless, very good power saving results can be obtained by utilizing the knowledge of the application running on it and through extensive profiling of a representative data input set from the application’s domain. Over the rest of this section, we focus our attention on the possible optimizations at the CPU level for a multimedia streaming application (MPEG).

We identified the subcomponents of the CPU (Fig. 3(b)) that consume the most power and observed the power distribution inside the CPU for MPEG decoding. Using extensive experimentation (Fig. 4) we conclude that:

- *The relative power contribution of the internal units of the CPU do not vary significantly with the nature or quality of the video played.* A possible reason for this is the symmetrical and repetitive nature of MPEG decoding, whose processing is done on fixed size blocks or macroblocks.
- *The units that show an important contribution to the overall power consumption and are amenable for power optimization are: caches, register files, functional units.* Cache behavior greatly affects the memory performance and hence power consumption, so we optimize the entire memory subsystem in an integrated way.

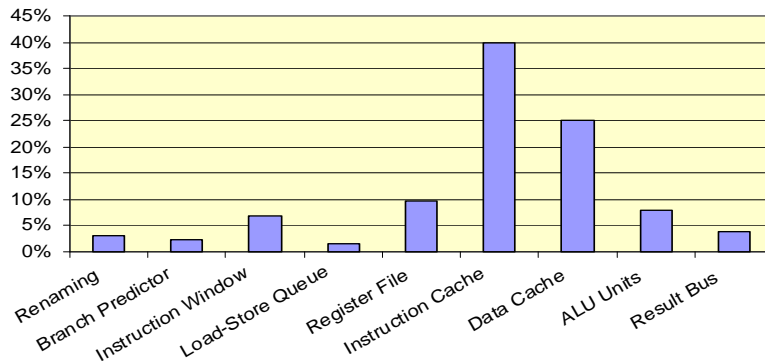


Fig. 4. Internal Relative Power Distribution on the CPU During MPEG Decoding

We briefly discuss the components identified above and suggest some additional improvements as a part of future work.

- **Caches/Memory:** cache configurations are determined by their size, number of sets, and associativity. The size specifies how large a cache should be, while the associativity/number of sets control its internal structure. We identify that most power gains for MPEG are possible through cache reconfiguration, more specific the data cache; cache optimizations influence memory traffic, so they are studied in an integrated way.

- **Compiler:** An efficient compiler can automatically set knobs as it generates the code. One example is the register file size. Each functions in the code has its own processing and storage requirements and therefore the compiler can choose a minimal register set to be used at runtime; this allows for the rest of the registers to be turned off, therefore saving power. We have experimented with this technique and noticed that the performance improvements in the case of MPEG decoding were just marginal and did not justify further evaluation.
- **Frame Traversal:** Decompressing MPEG video in its implied order does not leave space for exploiting the limited locality existent between dependent macroblocks. By just changing the frame traversal order algorithm based on the existing locality, faster decompression rates and significant power saving are achieved via reduced memory accesses [8]. Our proxy-based approach allows for a transparent on-the fly traversal reordering at transcoding time, giving an advantage over previous work where this was done at the device, incurring unacceptable frame decoding delays. However, we cite this as future work and this is not included in our results.

We should mention that while current processors (including the ARM core on iPaq) in general do not exhibit such aggressive architectural reconfigurations, except for special purposes, there are many research projects on this and eventually the techniques will be incorporated into more future processors.

Another knob, independent of the above is power management through the use of dynamic voltage scaling of the processor(DVS). DVS provides significant savings for MPEG streaming as it allows tradeoffs for transforming the frame decoding slack time (CPU idle time) into important power savings. We discuss DVS in a subsequent section and investigate the implications of DVS on other knobs in the system. All these knobs when fine-tuned for a specific video quality, will provide the best operating point(for power and performance) for a specific video stream.

3.2 Quality-driven Cache Reconfiguration

There are various techniques pertinent to cache optimizations. Power consumption for the cache depends on the runtime access counts: while hits result in only a cache access, misses add the penalty of accessing the main memory (external). Fortunately, in most applications the inherent locality of data means that cache miss rate is relatively low and so are accesses to external memory. However, MPEG decoding exhibits a relatively poor data locality, which, when combined with the large data sets exercised by the algorithm, leads to an increase in the cache memory-traffic.

The best configuration of the cache is not easily predictable; in fact, it may even be counter-intuitive. Decreasing the cache size, or the associativity, yields a reduction in cache power consumption; on the other hand it will generate more memory traffic due to the more frequent misses and line cache replacements, increasing power uptake. Finding the best

solution point is only achievable through an extensive simulation and profiling with data that is representative for the video domain.

Internal CPU caches are characterized by their size(\mathbf{S}), number of sets(\mathbf{NS}), line size(\mathbf{LS}) and associativity(\mathbf{A}). The relation between them is given by the formula $\mathbf{NS} \times \mathbf{LS} \times \mathbf{A} = \mathbf{S}$. We maintain the line size constant $\mathbf{LS} = c$, as it is typically harder to change line size through reconfiguration. As a result, we can fully describe the configuration of a cache by a pair (\mathbf{S}, \mathbf{A}) . Over the next paragraphs, by cache we refer to data cache (not instruction cache, which is not the scope of our optimizations).

Our cache reconfiguration goal is optimizing energy consumption for a particular video quality level \mathbf{Q}_k . In general, cache power consumption for a particular configuration and video quality is given by the function $\mathbf{E}_{cache,k}(\mathbf{S}, \mathbf{A})$. By profiling this function for the entire search space (\mathbf{S}, \mathbf{A}) of available cache configurations, we generate a cache energy variation graph shown in Fig. 5. Depending on the video quality \mathbf{Q}_k played, there will be one optimal operating point for that video quality: $(\mathbf{S}_k^{opt}, \mathbf{A}_k^{opt})$.

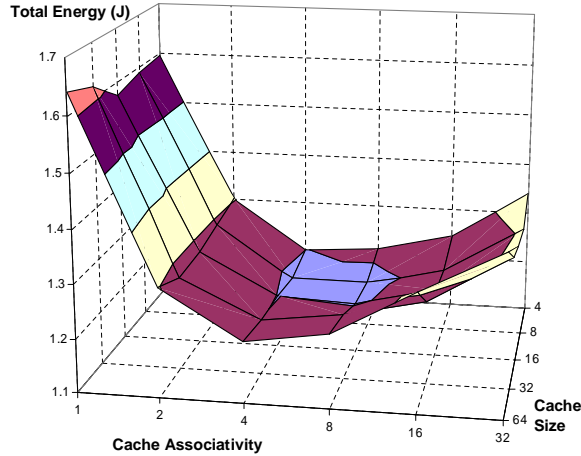


Fig. 5. Cache Energy Variation on Size and Associativity

We found out that for all video qualities an optimized operating point exists and it improves cache power consumption by up to 10-20% (as opposed to a suboptimized configuration). This technique effectively fine-tune the organization of the cache so that it perfectly matches the application and the data sets to be processed, yielding important power savings.

Cache behavior (especially misses) greatly affect memory performance (in terms of number of accesses). As we are interested in the total system power consumption and memory power is significant at this level; therefore we include it in our analysis. A memory access is performed (i) in case of a cache miss, to fetch an entire cache line and (ii) when a line needs to be replaced and is marked dirty (has been modified and needs to be written back to the memory). Hence the number of accesses to the memory can be computed as $\mathbf{accesses} = \mathbf{misses} + \mathbf{writebacks}$. Because we are not modifying the line size when reconfiguring the cache, energy consumption for a single memory access is constant \mathbf{E}_{access} . With this, the memory energy consumption is given by the formula $\mathbf{E}_{memory,k}(\mathbf{S}, \mathbf{A}) =$

$accesses_k(\mathbf{S}, \mathbf{A}) \times E_{access}$. By including the memory energy, the optimal operating point may change, as shown by the total energy consumption graph, as a function of cache parameters. Our results are along the line suggested in previous work [19].

3.3 Integrated Dynamic Voltage Scaling

The previous section shows that significant power savings can be achieved by optimally reconfiguring the cache to match the video quality. The savings can be further increased when this is combined with dynamic voltage scaling (DVS). A processor normally operates at a specific supply voltage V and clock frequency f . The dynamic power dissipated by the CPU (and any other CMOS circuits due to switching activity, in addition to the static component) varies linearly with frequency and circuit capacitance, and quadratically with voltage: $P \propto C \times f \times V^2$. The disadvantage of applying dynamic voltage scaling is its power-performance tradeoff. The values of P and f are not independent, the supply voltage determines the circuit delay τ ($\tau \propto V/(V - V_T)^\alpha$, where V_T is the threshold voltage and α is a velocity saturation index). Lowering the voltage causes an increase in the circuit delay and so a decrease in the frequency supported. As a consequence, DVS is only applied where such a tradeoff can be made.

In MPEG decoding, frames are processed in a fraction of the frame delay ($F_d = 1/frame_rate$). The actual value for the frame decoding time D depends on the type of MPEG frame being processed ($\mathbf{I}, \mathbf{P}, \mathbf{B}$) and also on the cache configuration (\mathbf{S}, \mathbf{A}) and DVS setting (f, V). In this study, we assume a buffered based decoding, where the decoded frames are placed in a temporary buffer and are only read when the frame is displayed. This allows us to decouple the decoder from the displaying; decoding time it still different for different frame, but we can assume an average D for a particular video stream/quality. The difference between the average frame delay and actual frame decoding time gives us the slack time $\theta = F_d - D$. When we perform DVS, we slow down the CPU so as to decrease the slack time to a minimum.

Let us assume that the initial configuration for the CPU before applying DVS is (f_0, V_0) . Frame decoding time is dependent on the cache configuration and the frequency at which the processor was running during profiling $D(\mathbf{S}, \mathbf{A}, f_0)$. The equation for the frame decoding slack time can be written as: $\theta_0 = F_d - D(\mathbf{S}, \mathbf{A}, f_0)$. After applying voltage scaling, the new values for frequency and voltage are (f_{new}, V_{new}) . The frame decoding time changes: $D_{new} = F_d - D(\mathbf{S}, \mathbf{A}, f_0) * f_0/f_{new}$ and the slack time is $\theta_{new} = F_d - D(\mathbf{S}, \mathbf{A}, f_0) * f_0/f_{new}$. The optimal solution is attained when the slack time θ_{new} is closest but not less than zero. For a particular quality level Q_k the frame delay F_d is a constant (known). The value for f_{new} is optimal when it minimizes θ_{new} (the slack time).

Having the best DVS setting for each cache configuration and quality level, we can look at the effect of the integrated approach on the power consumption. The DVS is not totally independent of the cache reconfiguration technique (cache configurations with a largest slack time allow for higher DVS based power reductions) and as a result it effectively reshapes the total power consumption. Through simulation, we find the best operating points for the DVS/cache reconfiguration combined approach in a manner similar to the one applied in the

previous section.

4 Architecture-Aware Middleware Adaptation

As seen in the previous section, architectural level optimizations can lead to substantial power and performance improvements. The gains can be further amplified if the low-level architecture is cognizant of the exact characteristics of the streamed video. An adaptive middleware framework at a proxy can dynamically intercept and doctor a video stream to exactly match the video characteristics for which the target architecture has been optimized. Additionally, it can regulate the network traffic to induce maximal power savings in a network interface. In this section, we introduce middleware techniques that compliment the architectural optimization approach.

4.1 Energy-Aware Admission Control & Stream Transformations

The middleware on the proxy utilizes the feedback from the device, to continuously monitor the U_f of the system. It performs an energy aware admission control initially to identify whether a request can be scheduled. Subsequently, it monitors the residual energy at the device and streams the highest quality video (performing realtime video transcoding) that meets the energy budget at the device and maintains an acceptable U_f . We characterize the admission control and stream transformations using the following parameters:

- T_{start}, T_{cur} : The start time of the video streaming and the current time respectively.
- T : duration of the entire video.
- E_{res} : Residual energy of the device.
- $Q_1 .. Q_N$: The 'N' video quality levels with the associated video transformation (Table 1) and architectural optimization parameters.
- $P_1 .. P_N$: The average power consumption (e.g. Table 1) of the corresponding quality levels of the video.
- I_f : Interval between successive feedbacks. This is decided by administrative policies.
- Q_a : Lowest user acceptable video quality for the request.
- R_i, F : The initial request and the feedback from the device respectively. The initial request contains E_{res}, Q_a and device specific details such as (NIC model, CPU arch etc). The feedback (F) simply contains values for E_{res} and Q_a . Note that R_i is used for initial admission and F is used for maintaining an acceptable U_f .

Fig 6 outlines the high level admission control algorithm employed by the proxy middleware. An initial admission control is performed to check whether the video can be streamed at the user requested quality level for the entire length of the video. Subsequently, the stream quality is adjusted using the periodic feedbacks from the device.

```

WHILE (TRUE)
{
  IF ( $R_i$  OR  $F$  received)
  {
    Determine  $i$ , such that  $Q_i = Q_a$ ;
    IF ( $(T - (T_{cur} - T_{start})) * P_i \leq E_{res} \&\& U_f == acceptable$ )
      Find Highest  $Q_i$  that satisfies the above
      inequality. Set the corresponding transformation
      parameters for the transcoder.
    ELSE
      REJECT the request OR (Negotiate video quality)
  }
}

```

Fig. 6. Admission Control

4.2 Network Traffic Regulation

In this section, we develop a proxy-based traffic regulation mechanism to reduce energy consumption by the device network interface. Our mechanism (a) dynamically adapts to changing network (e.g. noise) and device conditions. (b) accounts for attributes of the wireless access points (e.g. buffering capabilities) and the underlying network protocol (e.g. packet size). (c) uses the proxy to buffer and transmit optimized bursts of video along with control information to the device. However, even though packets are transmitted in bursts by the proxy, the device receives packets that are skewed over time Fig. 7; this cuts power savings, as the net *sleep* time of the interface is reduced. The skew is caused due to the ethernet access protocol (e.g. CSMA/CD) and/or the fair queueing algorithms implemented at the AP. Our mechanism optimizes the stream, such that the optimal video bursts sizes are sent for a given noise level, thus maximizing energy savings without performance costs.

Wireless network interface (WNIC) cards typically operate in four modes: *transmit*, *receive*, *sleep* and *idle*. We estimated the power consumption of the Cisco Aironet 350 series WLAN card to have the following power consumption characteristics: *transmit* (1.68W), *receive* (1.435W), *idle* (1.34W) and *sleep* (0.184W) which agree with the measurements made by Havinga et al. in [12, 20]. This observation [6] suggests that significant energy savings can be achieved by transitioning the network interface from *idle* to *sleep* mode during periods of inactivity. The use of bursty traffic was first suggested by Chandra [6, 7] and control information was used for adaptation in [18].

We analyze the above power saving approach using a realistic network framework (Fig. 7), in the presence of noise and AP limitations. The proxy middleware buffers the transcoded video and transmits I seconds of video in a single burst along with the time $\tau = I$ for the next transmission as control information. The device then uses this control information to switch the interface to the active/idle mode at time $\tau + \gamma \times D_{EtoE}$, where γ is an estimate between zero and one and D_{EtoE} is the end-to-end network delay with no noise [18].

Let B be the average video transmission bit-rate, F the video frame rate, S_N the packet

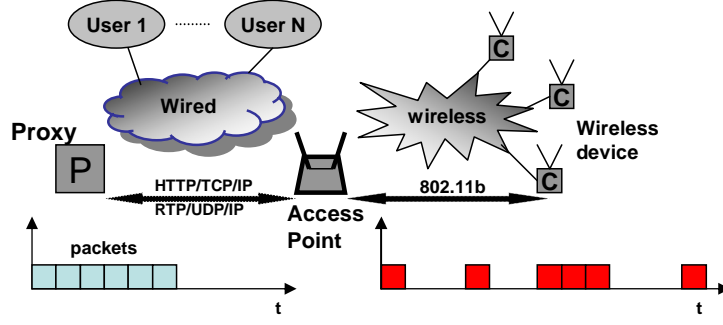


Fig. 7. Wireless Network

size used by the underlying network protocol. Let there be N users in the system. We model each “other” user as a Poisson process that injects packets into the network with the packet inter-arrival times following an exponential distribution with rate λ , with a density function of $f(t) = \lambda \cdot e^{-\lambda t}$. Therefore, the number of packets introduced into the network by each user in an interval ‘ t ’ has an expected value $E(t) = \lambda \cdot t$. Assume that the AP employs a simple round robin service for transmitting packets and let L_{AP} be the buffer length (in number of packets) available at the AP for downstream traffic. Let BW_s, BW_d be the bandwidth available to the wireless device for transmitting and receiving data; T_{AP}, P_d be the queueing transmission delay per packet of the AP and propagation delay of the wireless network respectively. Using the above characterization, the number of packets per frame $\alpha = \frac{B}{8 \times S_N \times F}$ and the total number of packets transmitted in one burst $P_b = \alpha \times F \times I$.

To simplify the analysis, we further assume that there are no loss of packets at the AP due to weak signal strengths or collisions. However, packets do get dropped if the AP buffer capacity (L_{AP}) is less than the number of arriving packets. A queueing theory analysis is used to predict packet loss rates at the AP. We omit the details due to space constraints. If T_b is the time taken to transmit the burst by the proxy, then the total expected number of packets received at the AP in that interval is $\sigma = P_b + \sum_{k=1}^{N-1} E_k(T_b)$, where E_k is the expected value of noise from by user k .

The worst case expected transmission delay experienced by the last packet in the burst is $D = \sigma \times T_{AP}$. Using the above approach of bursty transmissions, the total “sleep” time (δ) of the network interface can be calculated as $\delta = \tau - (D + \gamma \times D_{EtoE})$, neglecting the propagation delay of the final packet. As significant power gains are only achieved when the network interface is in the sleep mode, the total power savings are $P_{saved} = \delta * (P_{IDLE} - P_{SLEEP})$. Observe that large values of I can result in packet losses at the access point and/or buffer overflows at the device. We acknowledge that a QoS aware preferential service algorithm at the access point can impact power management significantly. The above analysis can be used by an adaptive middleware to calculate an optimal I (burst length) for any given video stream and noise level.

In the previous section, we demonstrated how low level architecture can be optimized

using high level information. In this section, we presented two middleware techniques that can be used to compliment the low-level hardware optimizations, lower energy consumption of the NIC and improve the overall utility of the system. We now present the performance results.

5 Performance Evaluation

We adopted a multi-phase methodology to achieve our results. First, through extensive experimentation and profiling we identified eight determinate video quality levels for a handheld. This determined our dynamic video transcoding parameters. Next, we optimized the low-level architecture to perform optimally with the above video streams. We also identified the best network transmission characteristics for video streams encoded at the above quality levels. Using these operating points for architecture and network transmission, we used our proxy admission control algorithm to optimally stream videos to the iPAQ, and measured the Utility Factor(U_F) for the system.

5.1 Experimental Setup

For video quality measurements, we used the setup shown in Fig. 8. All our measurements were made for a Compaq iPAQ 3650, with a 206Mhz Intel StrongArm processor, with 16MB ROM, 32MB SDRAM. The iPAQ used a Cisco 350 Series Aironet 11Mbps wireless PCMCIA network interface card for communication. The batteries were removed from the iPAQ during the experiment. We used a National Instruments PCI DAQ board to sample voltage drops across a resistor and the iPAQ, and sampled the voltages at 200K samples/sec. We calculated the instantaneous and average power consumption of the iPAQ using the formula $P_{iPAQ} = \frac{V_R}{R} \times V_{iPAQ}$ (Fig. 8). A number of videos ranging from high, medium and low action conter

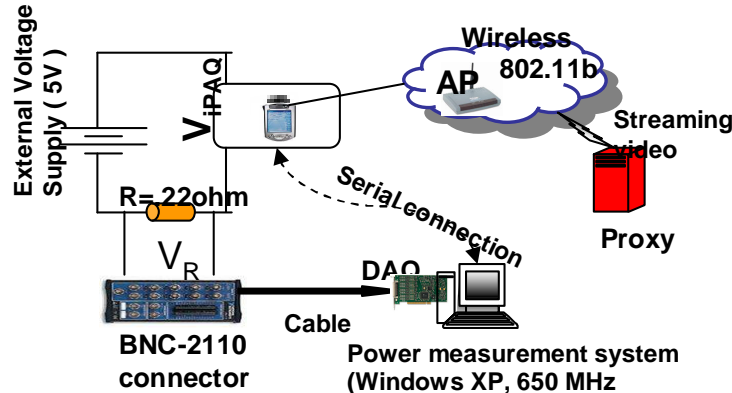


Fig. 8. Setup for Power Profiling

The CPU architecture simulation was implemented using the Wattch/SimpleScalar [5] power simulator. We configured our simulated CPU to resemble a typical Intel XScale processor (widely used in today’s mobile devices, mostly due to their excellent MIPS/Watt performance): ARM core, 400 MHz, 1.3V, 0.18um, 32k instruction cache, 32k data cache, single issue. The MPEG decoder from Berkeley MPEG Tools was used. Video transcoding was done using the commercially available TMPGEnc transcoder. As input video for the decoding, we used traces from various video clips from low action(e.g. “news”) like content to high action (e.g. GTA) fast scene changing streams. For each of these clips, we extract a sequence to be simulated and we encode it at noticeable different levels of quality (Table 1). The decoding program is then simulated through Wattch and statistics are extracted from the simulator output.

While the external memory is not simulated in Wattch, we estimate the power for memory accesses based on values from memory catalogs and cache statistics that translate in memory accesses (read and write misses). Including the external memory in our experiments is very important for accurate results, as is has been shown that the cache-memory traffic is very inefficient for MPEG video decoding [19]. In our experiments, we vary the cache size from 4Kb to 64Kb (4, 8, 16, 32, 64) and the associativity from 1 (direct mapped) to 32 (1, 2, 4, 8, 16, 32). This translates in a $5 \times 6 = 30$ point search space. The same exploration is done for each video quality (Q1 through Q8). Level 1 correspond to the best possible quality: 320x240 frame size, 30fps framerate, 650kbps bitrate. From 1 to 8, each level differs from the previous one by at least one parameter (frame size, frame rate or bitrate). This way, we have a clear (observable) degradation in quality between each quality level. We collect the total energy consumption for the duration of the clips (10 seconds).

We simulated the overall system using the admission control, network traffic regulation for measuring the *Utility factor*(\mathbf{U}_F) for the system. First, for each video quality(Q1-Q8), we varied the video burst time(100ms to 20sec) and the network noise level(N=1 to N=15), the network packet size (200bytes to 700 bytes) with the constant end-to-end network delay of 400ms. The wired and wireless ethernet bandwidths were set to 10Mbps and 8Mbps(effective B/W, 802.11b is capable of higher throughput) and γ was set to a 0.85. The transmission delay of the wireless access point was also fixed at 400 μ s per packet. Using the search space, we determined the ideal video burst size for a particular noise level and packet size, for each quality level. Using these operating points, we measured the overall \mathbf{U}_F of the system for video streams of various playback times.

5.2 Experimental Results

In this section, we first analyze the performance of our architectural and middleware optimizations. Later, we present the improvements in the overall utility of the system achieved in the system with the integrated approach.

Architectural Optimizations

We profile short (10sec) video clips through our power simulator, for all combinations of cache parameters. Two examples of the total (CPU/memory) energy variation is shown in Fig. 9 and Fig. 10 (high quality MPEG clips). For all video quality levels, we were able to determine a cache configuration that minimizes energy consumption. Moreover, through cache reconfiguration, we obtained power savings in the range 10-15%, depending on the nature of the video.

From the two graphs we make the following observations:

- The type of video content has a great impact on the shape of the energy consumption vs cache parameters. Looking at the two figures, we can observe that the first one (representing an *action* video) has a steeper shape, while the second (*news* video) is more flat.
- Cache associativity produces the largest shift in energy consumption, extreme values proving very inefficient (especially for direct mapped caches).
- The best cache configurations were in most of the cases [32Kb, 2-way set-associative] and [8Kb, 8-way set associative]. This reflects the internal storage requirements for different frame sizes and organization of the decoding algorithm.

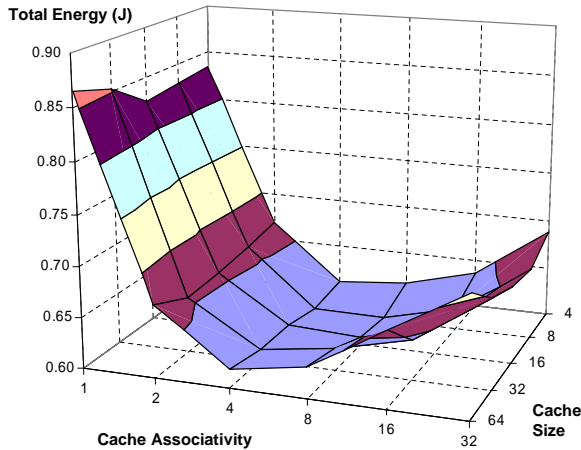


Fig. 9. Cache Optimization for a High Quality, Action Clip

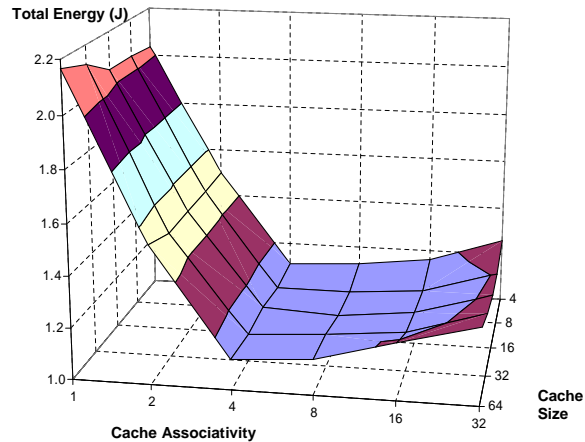


Fig. 10. Cache Optimization for a High Quality, News Clip

Optimized Knobs after Applying DVS

We combined dynamic voltage scaling technique with cache reconfiguration for an increased overall effect on power consumption. Fig 11 and Fig 12 plot energy consumption for a medium quality level of the same video used in Fig. 9, before and after DVS. The combined

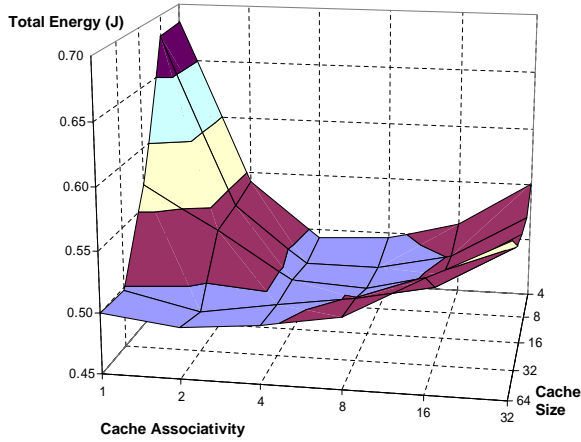


Fig. 11. Search Space for a Medium Quality Action Clip, no DVS

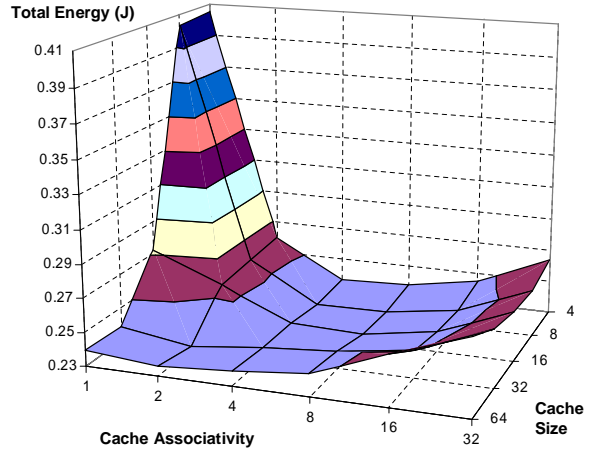


Fig. 12. Search Space for a Medium Quality Action Clip, after DVS

approach gave us up to 60% in energy savings as compared with the initial architecture. We repeated the same procedure for all the video quality levels. We make the following observations:

- DVS effectively lowers the floor on the energy surface, for most cache configurations, with the exception of small caches, with low associativity and small sizes. The reason is that the small, directed caches are already very inefficient for MPEG decoding and cannot accept the extra performance degradation caused by DVS.
- Comparing the high & medium quality video energy graphs for the same clip, we can see that lowering the video quality level, not only decreases the energy consumption, but also shifts the optimized point in regions very inefficient previously (e.g. 32k, 2-way cache vs 8k, 8-way).
- In most of the cases we were able to run the CPU at a significant lower voltage (100MHz, 66MHz), mainly due to the initial high speed of the simulated XScale processor (400MHz) and the quality of the code (highly optimized). A real device may not be able to scale the frequency to such a low level, due to other required computations except MPEG decoding (network drivers, OS, image rendering).

The overall energy savings we obtained after both above techniques and the knob values for the optimized configuration are shown in Table 2 and Table 3, for *action* and *news* video clips.

Middleware Optimization for Video Bursts

Fig. 15 shows the power optimization space for an “*action video*” for the eight video quality streams in the presence of noise. The case(N=1) represents the case with no noise(recall

Video Quality	Cache Size	Cache Associativity	Clock Frequency	Voltage	Original Energy	Optimized Energy	Savings
Q1	8	8	100	1	1.29	0.76	47.5%
Q2	8	8	100	1	1.09	0.64	47.8%
Q3	8	8	100	1	0.95	0.56	48.0%
Q4	32	2	66	0.9	0.54	0.26	57.6%
Q5	32	2	66	0.9	0.48	0.23	57.8%
Q6	32	2	33	0.9	0.42	0.20	58.0%
Q7	8	8	33	0.9	0.29	0.14	57.3%
Q8	8	8	33	0.9	0.24	0.11	57.5%

Table 2. Architectural Configurations for Ideal Energy and Performance Gains (Action Clip)

Video Quality	Cache Size	Cache Associativity	Clock Frequency	Voltage	Original Energy	Optimized Energy	Savings
Q1	8	8	100	1	1.19	0.70	47.7%
Q2	8	8	100	1	1.02	0.60	48.1%
Q3	8	8	100	1	0.94	0.55	48.4%
Q4	32	2	66	0.9	0.54	0.26	57.7%
Q5	32	2	66	0.9	0.48	0.23	57.9%
Q6	32	2	66	0.9	0.43	0.21	58.2%
Q7	32	2	33	0.9	0.31	0.15	57.4%
Q8	32	2	33	0.9	0.26	0.12	57.7%

Table 3. Architectural Configurations for Ideal Energy and Performance Gains (News Clip)

number of users introducing noise is $N-1$ in our model). It was seen that the power saved at the NIC, is the least for the highest quality video and the most for the lowest quality video. Also, as expected the power gains diminish with noise. Fig. 14 shows the optimal burst time search space for the video streams. As expected the highest quality video has a very small burst time compared to the lowest quality video. The ideal burst times were ascertained such that none of the frames missed deadlines at the device. With a small tolerance to missed frames the power savings can be improved even further. Table 4 shows the ideal burst times and the corresponding power savings for the same video stream encoded at the eight quality levels. Fig. 13 shows the percentage of packets dropped in the network for different burst times and noise levels for the highest quality stream. Clearly very small burst times yield no gains. The vertical line indicates the point at which packets start getting dropped. Interestingly, for every additional user in the system, a new optimal burst time

exists.

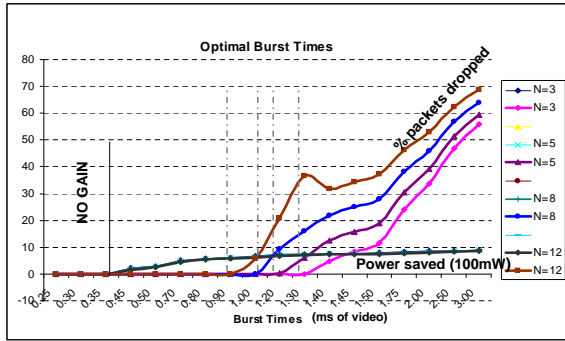


Fig. 13. Finding Optimal Burst Time

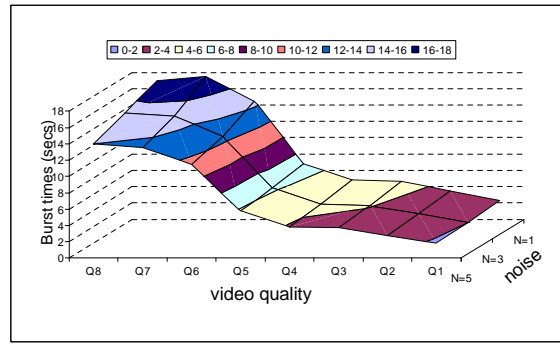


Fig. 14. Optimal Burst Times

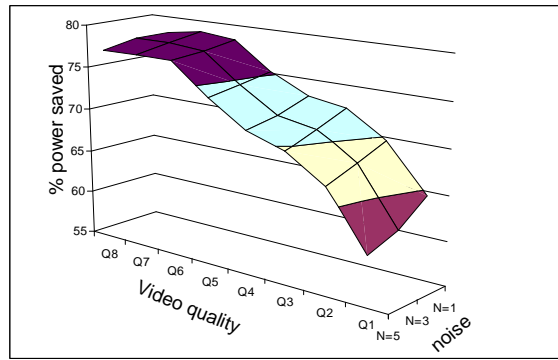


Fig. 15. Power vs. Noise

Utility Factor of the Integrated Framework

We finally evaluate the performance of the integrated framework with the architectural and middleware optimizations in place. Here we observe how the *Utility Factor* (U_F), which is a measure of "user satisfaction" is improved by the system. Fig. 16 plots the Utility Factor versus time for a two hour video sequence with varying initial residual energy values at the device. The horizontal lines indicate the U_F without the optimizations. Clearly, the savings allow for a significant improvement in the U_F , and hence substantially improves the user experience. Fig. 16 shows the same video requested but this time with a random user induced energy cost (due to other processes, backlight etc.). Clearly, the utility factor is improved with the integrated approach even in the presence of noise. Note that for us "Q1" is the highest quality and "Q8" is the lowest quality, therefore a higher utility factor indicates better user experience.

Quality Level	Burst Length (N=1, secs)	Power Saved (N=1,Watts)	Burst Length (N=3, secs)	Power Saved (N=3,Watts)	Burst Length (N=5, secs)	Power Saved (N=5, Watts)
Q1	2.3	.925	2.0	0.89	1.8	0.87
Q2	3.5	1.0	3.05	0.98	2.76	0.96
Q3	4.6	1.04	4.05	1.02	3.68	1.0
Q4	4.85	1.05	4.2	1.03	3.85	1.02
Q5	6.8	1.08	6.25	1.07	5.75	1.06
Q6	14.5	1.12	12.5	1.11	11.5	1.11
Q7	17.5	1.13	15.0	1.12	13.5	1.11
Q8	17.0	1.12	15.4	1.12	14.0	1.11

Table 4. Optimal network video burst lengths(in secs) and corresponding power gains for different quality and noise levels for the Grand Theft Auto action video, assuming sufficient buffer available at client and network packet size of 500KB

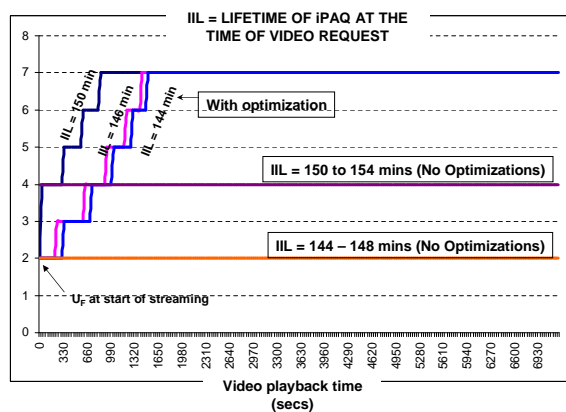


Fig. 16. Utility Factor over time

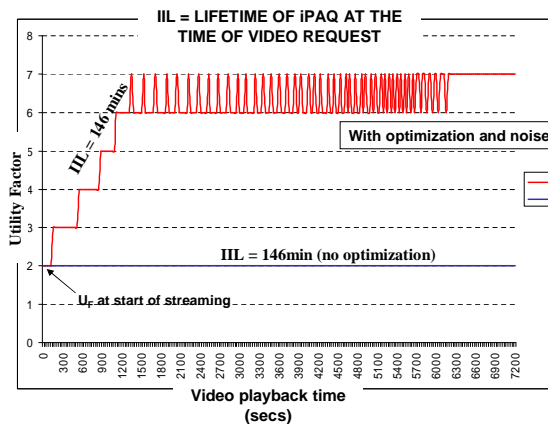


Fig. 17. UF2

6 Related Work

To provide acceptable video performance at the hardware level, efforts have concentrated on analyzing the behavior of the decoder software and devising either architectural enhancements or software improvements for the decoding algorithm. Until recently it was believed that caches can bring no potential benefit in the context of MPEG (video) decoding. In fact, due to the poor locality of the data stream, many MPEG implementations viewed video data as “un-cacheable” and completely disabled the internal caches during playback. However, Soderquist and Leeser [19] show that video data has sufficient locality that can be exploited to reduce cache-memory traffic by 50 percent or more through simple architectural changes. Dynamic Voltage Scaling [14, 9] for MPEG streams have been widely researched. A different way of improving cache performance by reordering frame traversal was proposed in [8].

Register file reconfiguration was applied in [4]. At the application and middleware levels, the primary focus has been to optimize network interface power consumption [10, 6, 7]. A thorough analysis of power consumption of wireless network interfaces has been presented in [10]. Chandra et al. [6] have explored the wireless network energy consumption of streaming video formats like Windows Media, Real media and Apple Quick Time. In [7], they have explored the effectiveness of energy aware traffic shaping closer to a mobile client. In [18], Shenoy suggests performing power friendly proxy based video transformations to reduce video quality in real-time for energy savings. They also suggest an intelligent network streaming strategy for saving power on the network interface. We have a similar approach, but we model a noisy channel. Caching streams of multiple qualities for efficient performance has been suggested in [11]. The GRACE project [23] professes the use of cross-layer adaptations for maximizing system utility. They suggest both coarse grained and fine grained tuning of parameters for optimal gains. In [22], a resource aware admission control and adaptation is suggested for multimedia applications for optimal CPU gains. Dynamic transcoding techniques have been studied in [3] and objective video quality assessment has been studied in [21, 13].

Conclusions & Future Work

In this paper, we integrated low-level hardware optimizations with high level middleware adaptations for enhancing the user experience for streaming video onto handheld computers. We identified and fine tuned low level hardware to perform optimally with video streams at discrete quality levels. We then used a higher level middleware approach to intercept and doctor the stream to compliment the architectural optimizations. A proxy based adaptive network transmission mechanism was developed to minimize the power consumption of the network interface card. Finally, all the above techniques were integrated into a system, and the overall system utility in terms of user satisfaction was measured. Significant improvements were observed in the requested video stream quality, enhancing the user experience substantially. We are currently exploring further architectural and middleware adaptations for improving the power consumption of displays, storage devices etc. and integrating them into the framework. In mobile multimedia environments, composition of additional middleware services like domain security levels and service protocols pose interesting challenges, that we plan to investigate.

References

- [1] "ITU-R Recommendation BT-500.7, Methodology for the subjective assessment of the quality of television pictures". In *ITU Geneva Switzerland*, 1995.
- [2] Sarnoff Corporation white paper, JND: a human vision system model for objective picture quality measurements. In <http://www.sarnoff.com>, 2001.
- [3] S. Acharia and B.C.Smith. Compressed Domain Transcoding of MPEG. In *ICMCS*, 1998.
- [4] A. Azevedo, R. Cornea, I. Issenin, R. Gupta, N. Dutt, A. Nicolau, and A. Veidenbaum. Architectural and compiler strategies for dynamic power management in the copper project. In *IWIA*, 2001.

- [5] D. Brooks, V. Tiwari, and M. Martonosi. Wattach: A framework for architectural-level power analysis and optimizations. June 2000.
- [6] S. Chandra. Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats. In *MMCN*, January 2002.
- [7] S. Chandra and A. Vahdat. Application-specific Network Management for Energy-aware Streaming of Popular Multimedia Formats. In *Usenix Annual Technical Conference*, June 2002.
- [8] W. chi Feng and S. Sechrest. Improving data caching for software mpeg video decompression. In *IS&T/SPIE Digital Video Compresssion: Algorithms and Technologies*, February 1996.
- [9] K. Choi, K. Dantu, W.-C. Chen, and M. Pedram. Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder. In *ICCAD 2000*, 2002.
- [10] L. Feeney and M. Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an ad hoc Networking Environment. In *IEEE Infocom*, April 2001.
- [11] J. Flinn and M. Satyanarayanan. Energy-Aware Adaptations for Mobile Applications. In *SOSP*.
- [12] P. J. M. Havinga. *Mobile Multimedia Systems*. PhD thesis, University of Twente, Feb 2000.
- [13] J. Jansen, T. Coppens, and D. D. Vleeschauwer. Quality Assessment of Video Streaming in the Broadband Era. In *ACIVS*, 2002.
- [14] M. Mesarina and Y. Turner. A Reduced Energy Decoding of MPEG Streams. In *MMCN*, January 2002.
- [15] S. Mohapatra and N. Venkatasubramanian. PARM: Power-Aware Reconfigurable Middleware. In *ICDCS-23*, 2003.
- [16] B. D. Noble, M. Satyanarayanan, D.Narayanan, J.E.Tilton, and J. Flinn. Agile Application-Aware Adaptation for Mobility. In *SOSP*, October 1997.
- [17] P. Pillai and K. G. Shin. Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. In *SOSP*, 2001.
- [18] P. Shenoy and P. Radkov. Proxy-Assisted Power-Friendly Streaming to Mobile Devices. In *MMCN*, 2003.
- [19] P. Soderquist and M. Leiser. Optimizing the data cache performance of a software MPEG-2 video decoder. In *ACM Multimedia*, pages 291–301, 1997.
- [20] M. Stemm and R. Katz. Measuring and Reducing energy consumption of network interfaces in hand-held devices. In *IEICE*, August 1997.
- [21] S. Winkler. Issues in vision modeling for perceptual video quality assessment. In *Signal Processing 78(2)*, 1999., 1999.
- [22] W. Yuan and K. Nahrstedt. A Middleware Framework Coordinating Processor/Power Resource Management for Multimedia Applications. In *IEEE Globecom*, Nov 2001.
- [23] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets. Design and Evaluation of a Cross-Layer Adaptation Framework for Mobile Multimedia Systems. In *MMCN*, January 2003.

A Other Experimental Results

A.1 Cache Exploration Results Before Applying DVS

A.1.1 Action Clips

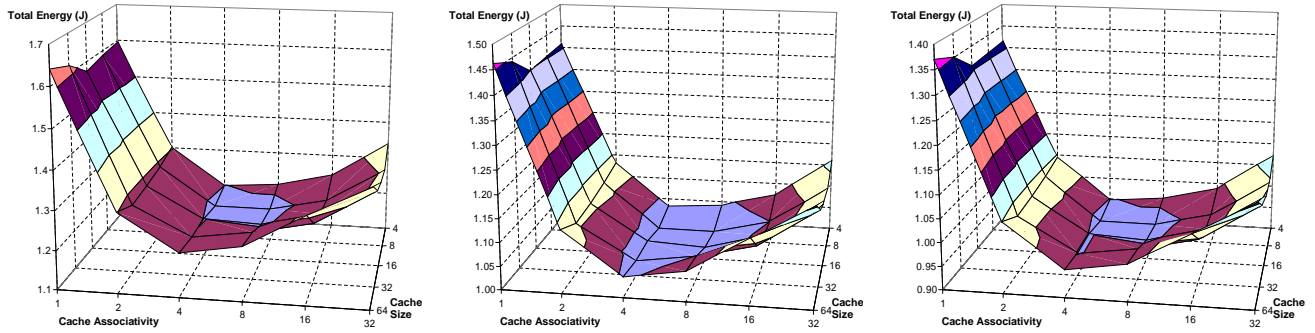


Fig. 18. Cache Exploration Results for High Quality (Q1, Q2, Q3), Action Clips

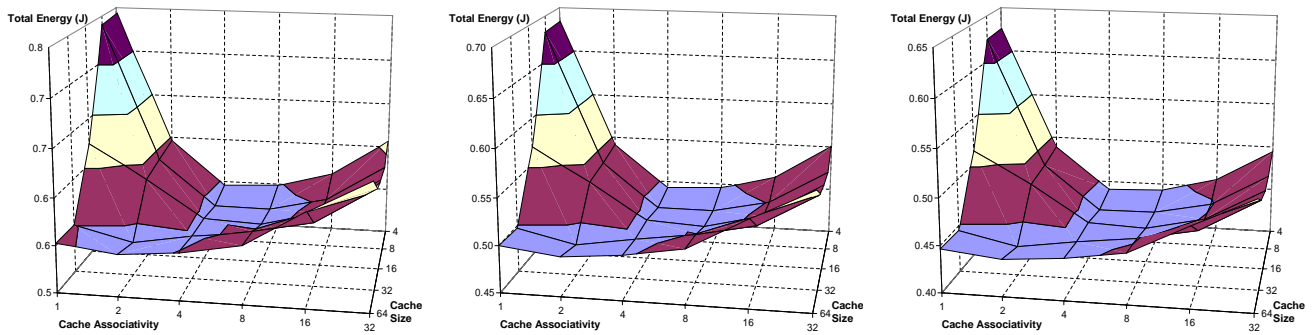


Fig. 19. Cache Exploration Results for Medium Quality (Q4, Q5, Q6), Action Clips

A.1.2 News Clips

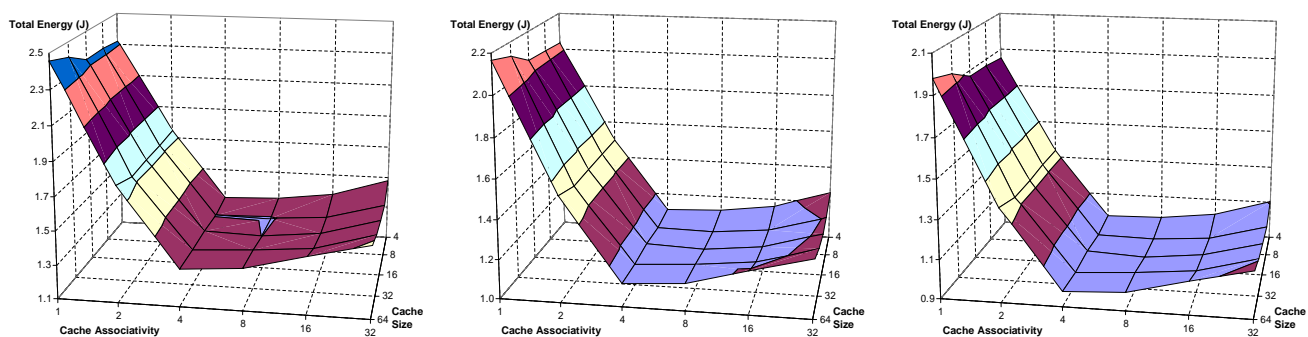


Fig. 20. Cache Exploration Results for High Quality (Q1, Q2, Q3), News Clips

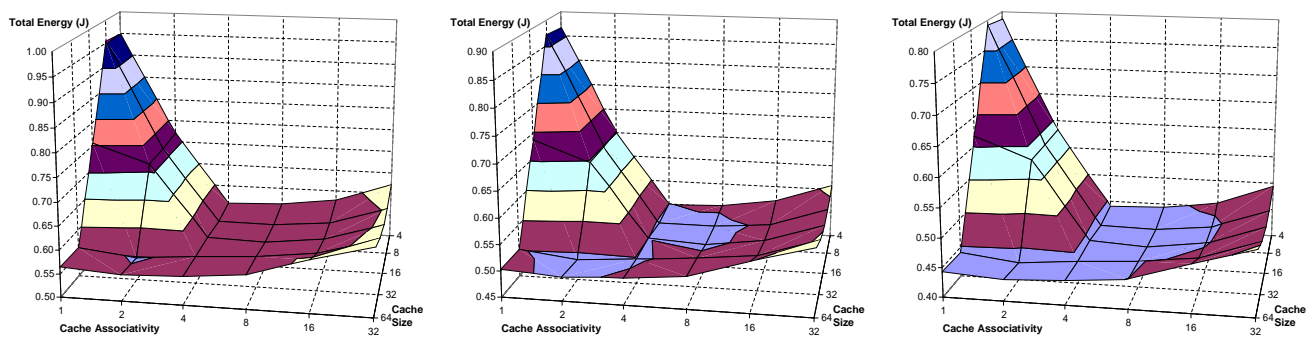


Fig. 21. Cache Exploration Results for Medium Quality (Q4, Q5, Q6), News Clips

A.2 Cache Exploration Results After Applying DVS

A.2.1 Action Clip

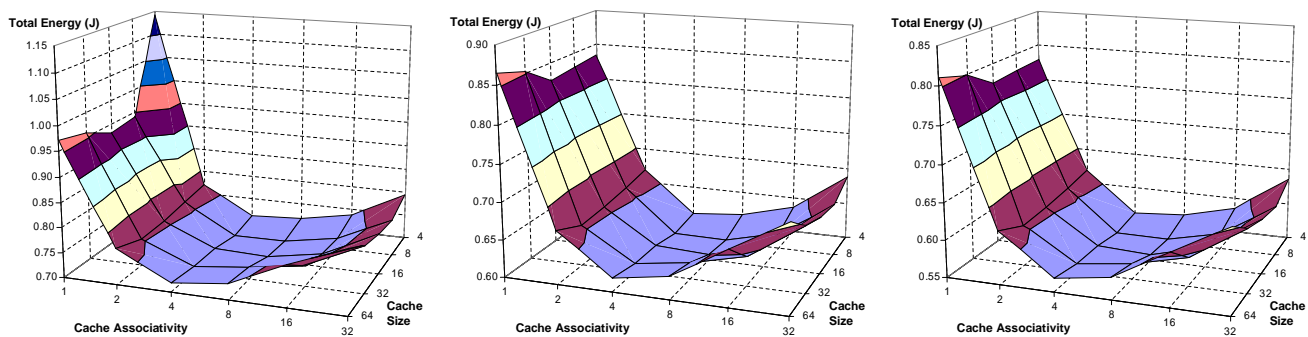


Fig. 22. Cache/DVS Exploration Results for High Quality (Q1, Q2, Q3), Action Clips

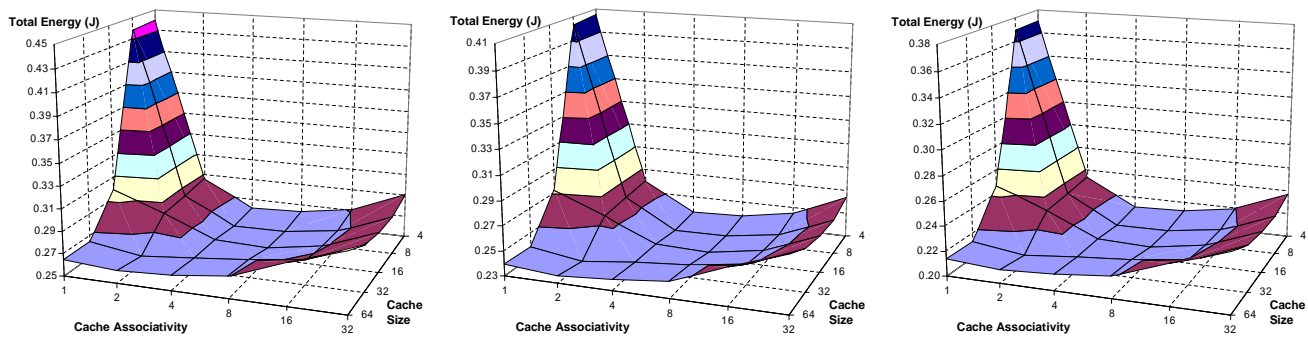


Fig. 23. Cache/DVS Exploration Results for Medium Quality (Q4, Q5, Q6), Action Clips

A.2.2 News Clip

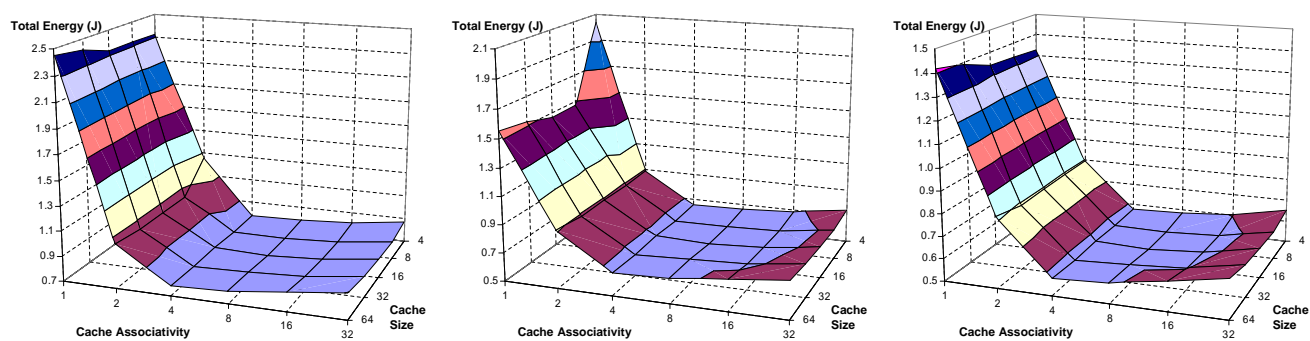


Fig. 24. Cache/DVS Exploration Results for High Quality (Q1, Q2, Q3), News Clips

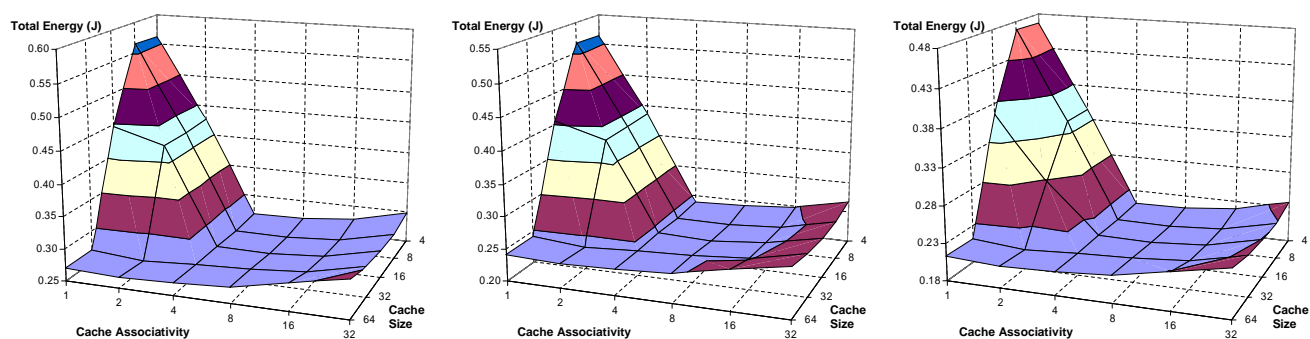


Fig. 25. Cache/DVS Exploration Results for Medium Quality (Q4, Q5, Q6), News Clips