Energy Efficient Soft Real-Time Computing through Cross-Layer Predictive Control

Guangyi Cao Arun A. Ravindran
Department of Electrical and Computer Engineering
University of North Carolina at Charlotte

Abstract

The next decade of computing workloads is expected to be dominated by soft-real time applications such as multimedia and machine vision. Such workloads are characterized by transient spikes requiring over provisioning of compute servers, adversely affecting the cost, energy usage, and environmental impact of data centers. In many of these applications, although deadlines need to be met to provide QoS guarantees, other quality parameters of the application (for example, visual quality in video processing) can be tuned in conjunction with hardware parameters (for example, DVFS) to give acceptable performance under overload conditions. In this paper, we experimentally demonstrate a predictive control approach for improving overload capacity and energy efficiency by incorporating control variables from both the hardware and the application layer. Further, we illustrate the impact of the choice of multiprocessor real-time scheduling algorithms on the performance of the controller for heterogeneous workloads.

1 Introduction

The next decade of computing is expected to be driven by the increasing pervasiveness of personal mobile computing devices and cyber physical systems. Many of the next generation applications in entertainment, human computer interaction, infrastructure, security and medical systems are computationally intensive, always-on, and are characterized by periodic tasks with soft real time (SRT) requirements. While failure to meet deadlines is not catastrophic in SRT systems, missing deadlines can result in an unacceptable degradation in the quality of service (QoS). To ensure acceptable QoS under dynamically changing operating conditions such as changes in the workload, energy availability, and thermal constraints, systems are typically designed for worst case conditions. Unfortunately, such overdesigning of systems increases costs and overall power consumption. A possible solution to this problem is run-time adaptation of the system to handle dynamically changing operating conditions. Previous research on cross-layer run-time adaptation has focused on open-loop control where the output has no effect on the system input and hence can only counteract against disturbances for which it has been designed [39, 10, 19]. In contrast in closed loop control, feedback is used to determine if real-time requirements are in met in the presence of unmodeled disturbances. However, existing research on closed loop control for real-time workloads have been limited to the use of control inputs derived from a single layer of the computing stack such as processor DVFS or scheduling policies.

In this paper we show that a higher overload capacity and better energy efficient operation of the system is possible if the control inputs are derived from all parts of the computing stack. We note that in many of the applications described above, although deadlines need to be met to provide QoS guarantees, other quality parameters of the application (for example, visual quality in video processing) can be tuned in conjunction with hardware parameters (for example, DVFS) to give acceptable performance under overload conditions. We formulate the real-time task execution as a Multiple-Input, Single-Output (MISO) optimal control problem involving tracking a desired system utilization set point with control inputs derived from across the computing stack. We assume that an arbitrary number of SRT tasks may join and leave the system at arbitrary times. The tasks are scheduled on multiple cores by a dynamic priority multiprocessor scheduling algorithm. Note that utilization above the set-point results in tasks missing deadlines while utilization under the set-point results in energy inefficient operation.

Our cross-stack control framework is shown in Figure 1. We use a model predictive controller (MPC) to realize optimal control. MPCs use an internal system

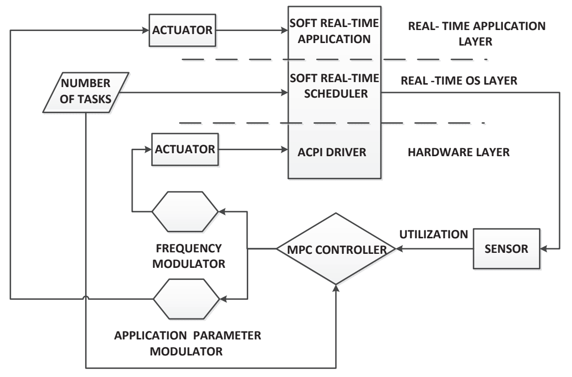


Figure 1: Schematic representation of our cross-layer control framework

model to predict the future trajectory of the output variables. Based on a history of past control moves, a constrained optimization is solved on-line to determine the future input trajectory such that the output variables track a reference trajectory over a receding horizon. MPCs are easy to tune, can handle multiple control variables, and constraints on both the dependent and independent variables.

Following initial design using MATLAB, we experimentally demonstrate the operation of our controller on a video encoder application (*x264*) and a computer vision application (*bodytrack*) executing on a dual socket quad-core Xeon processor with a total of 8 processing cores. All tasks including the application and the controller are scheduled by real-time dynamic task schedulers implemented in Linux using the Litmus-RT patch [13]. We evaluate the performance and energy saving of the cross-layer control for homogeneous workloads. Further, we illustrate the impact of the choice of multiprocessor real-time scheduling algorithms on the performance of the controller for heterogeneous workloads.

The rest of the paper is organized as follows - Section 2 gives a brief overview of related work. Section 3 describes our control framework and Section 4, the evaluation methodology. Section 5 presents the experimental setup and results. Section 6 concludes the paper.

2 Related Work

Most of the existing research on run-time power management targets exclusively either the hardware layer [33, 31, 37, 29, 38, 2, 6, 17], or the application layer [3, 32, 5, 21, 4, 28, 34, 35, 22, 25, 26, 20] or the system software layer [24, 27, 36, 11]. Among the cross-layer open-loop approaches , the Illinois Grace project [39] uses a hierarchical adaptation at all system layers including application (frame rate and dithering for video decoding), soft real time scheduling (CPU time allocation) and CPU (DVFS) for power optimization. The optimization problem involves maximizing quality and minimizing power with energy, processor utilization, frequency and quality of service as constraints. Bitirgen et al. proposed a framework that manages multiple shared multicore resources in a coordinated fashion to guarantee performance objectives[10]. They formulate global resource allocation as a machine learning problem. Hoffman, et al. [19] proposed PTRADE, a framework that coordinates a number of different hardware resource in a computing system. PTRADE consists of two interfaces which communicates an application's performance goals and current performance, and a runtime system which determines how to optimize system performance by tuning hardware resources. However PTRADE only works on the hardware layer. Cucinotta, et al. [14] have proposed an adaptive resource allocation mechanism organized in two feedback loops. The internal loop is responsible for updating the execution budget for soft real-time multimedia task so that timing constraints of the application are satisfied. The external loop operates on the QoS level of the applications and on the power level of the resources in open loop to strike a good trade-off between the global QoS and the energy consumption. We use a single closed loop model predictive controller to meet timing constraints while adapting power and QoS.

3 Cross-layer Control Framework

3.1 Soft Real-Time Schedulers

A multicore real-time system consists of *m* cores and *N* periodic tasks. Under a periodic task model, each task consists of a sequence of jobs which are released periodically. We use Global and Cluster Earliest Deadline First (G-EDF and C-EDF) scheduling algorithms to assign tasks to the cores. Previous research has shown that for SRT tasks scheduled by EDF executing on *m* processors, a utilization level of *m* is possible with bounded tardiness [16]. In EDF, tasks with earlier deadlines are selected to run at the next scheduling point. G-EDF uses a single unified task scheduler, ready queue, and release queue to handle all tasks. At any instant, the task scheduler picks at most *m* jobs with the earliest deadlines to execute on *m* processors. There is no restriction imposed on where a task can execute. In C-EDF, all cores that share a specific cache level (L2 or L3) are defined to be a cluster; tasks are allowed to migrate within a cluster, but not across clusters; tasks assigned to a cluster are scheduled globally within the cluster under the EDF algorithm. Typically, better load balancing is achieved with G-EDF while C-EDF promotes better data locality.

3.2 System Model

We model our system as a standard linearized, discrete-time, state space model in the form below [8]:

$$\begin{aligned} x(k+1) &= Ax(k) + B_u u(k) + B_v v(k) + B_d d(k) \\ y_m(k) &= C_m x(k) + D_{vm} v(k) + D_{dm} d(k) \end{aligned} \tag{1}$$

In Equation 1, $x(k)$ is the n_x-dimensional state vector of the plant, $u(k)$ is the n_u dimensional vector of manipulated variables, $v(k)$ is the n_v dimensional vector of measured disturbances, $d(k)$ is the n_d dimensional vector of unmeasured disturbances entering the system, and $y_m(k)$ is the n_y dimensional vector of measured outputs. The unmeasured disturbance $d(k)$ is modeled as the output of an LTI system

$$\begin{aligned} x_d(k+1) &= \bar{A} x_d(k) + \bar{B} n_d(k) \\ d(k) &= \bar{C} x_d(k) + \bar{D} n_d(k) \end{aligned} \tag{2}$$

Where $n_d(k)$ is the random Gaussian noise with zero mean and unit covariance matrix.

3.3 Model Predictive Control

The values of the set-points, measured disturbances, and constraints are specified over a finite prediction horizon P; the controller computes future inputs for a control horizon M ($1 \le M \le P$). Assuming that the estimates $x(k)$ and $x_d(k)$ are available at time k from state estimation, the future inputs at time k are obtained by solving the optimization problem [8]

$$\begin{aligned} \min_{\Delta u(k|k),\ldots,\Delta u(m-1+k|k),\varepsilon} \{ \sum_{i=0}^{p-1} [\sum_{j=1}^{n_y} |w^{y}_{i+1,j}(y_j(k+i+1|k) \\ - r_j(k+i+1))|^2 + \sum_{j=1}^{n_u} |w^{\Delta u}_{i,j} \Delta u_j(k+i|k)|^2] + \rho_\varepsilon \varepsilon^2 \} \end{aligned} \tag{3}$$

subject to the constraints,

$$\begin{aligned} & u_{jmin}(i) - \varepsilon V^{u}_{jmin}(i) \le u_j(k+i|k) \le u_{jmax}(i) + \varepsilon V^{u}_{jmax}(i) \\ & \Delta u_{jmin}(i) - \varepsilon V^{\Delta u}_{jmin}(i) \le \Delta u_j(k+i|k) \le \Delta u_{jmax}(i) + \varepsilon V^{\Delta u}_{jmax}(i) \\ & y_{jmin}(i) - \varepsilon V^{y}_{jmin}(i) \le y_j(k+i|k) \le y_{jmax}(i) + \varepsilon V^{y}_{jmax}(i) \\ & ,i = 0,\ldots,p-1 \\ & \Delta u(k+h|k) = 0,\ h = M,\ldots,P-1 \ \ \varepsilon \ge 0 \end{aligned} \tag{4}$$

Here, $r(k)$ is the value of the reference variable at time k, $w^{\Delta u}_{i,j}$, $w^{y}_{i,j}$ are non-negative weights for the corresponding variables. A smaller w indicates a lower importance of the corresponding variable in the overall cost function. $u_{j,min}$, $u_{j,max}$, $\Delta u_{j,min}$, $\Delta u_{j,max}$, $y_{j,min}$, and $y_{j,max}$ are the lower/upper bounds of the corresponding variables. The weight ρ_ε of the variable ε penalizes the violation of constraints. The relaxation vectors V^{u}_{min}, V^{u}_{max}, $V^{\Delta u}_{min}$, $V^{\Delta u}_{max}$, V^{y}_{in}, and V^{y}_{max} represent the penalty for relaxing the corresponding constraints; the larger the V, the softer the constraint. If all bounds are infinite and the slack variables are removed, the problem can be solved analytically; else a Quadratic Programming (QP) solver is used. Since the output constraints are always soft, the QP problem is never infeasible [8]. Note that only $\Delta u(k|k)$ is actually used to compute $u(k)$. The remaining samples $\Delta u(k+i|k)$ are discarded and a new optimization problem based on $y_m(k+1)$ is solved the the next sampling step $k+1$.

Since the states $x(k)$ and $x_d(k)$ are not directly measurable, predictions are obtained from a state estimator. The estimates are computed from the measured output $y_m(k)$ by the linear state observer

$$\begin{bmatrix} \hat{x}(k|k) \\ \hat{x}_d(k|k) \end{bmatrix} = \begin{bmatrix} \hat{x}(k|k-1) \\ \hat{x}_d(k|k-1) \end{bmatrix} + G(y_m(k) - \hat{y}_m(k)) \tag{5}$$

$$\hat{y}_m(k) = C_m \hat{x}(k|k-1) + D_{vm} v(k) + D_{dm} \bar{C} \hat{x}_d(k|k-1) \tag{6}$$

The gain G is designed using Kalman filtering techniques [8].

4 Evaluation Methodology

4.1 Benchmarks

We use two soft real-time benchmarks from the Parsec benchmark suite [9] - *x264* and *bodytrack*. Both benchmarks were modified to confirm to an implicit deadline periodic workload model. The *x264* application is an H.264/AVC (Advanced Video Coding) video encoder. The encoder grabs video frames periodically and encodes based on the MP4 video format specification. The video frame resolution level ranges from $\frac{1}{4}$ HD (230,400 Pixels per image) to full HD (921,600 Pixels per image), and is chosen as the application quality tuning knob determining the visual quality. The *bodytrack* computer vision application tracks a humans movement through an image sequence periodically from multiple cameras [7] [15]. An annealed particle filter is used to track the movement of a human through the scene.The graphic output of *bodytrack* generates conic cylinders to represent 10 body components including torso, head and limbs. The number of annealing layers ranging from 1 - 5, and the number of particles ranging from 100 - 4000, are chosen as the application quality tuning knobs. As a measure of visual quality, the relative mean square error in the magnitude of the position vectors of the body parts for different values of the tuning knobs is used [20]. For the values of the application quality tuning knobs, the relative mean square error is less than 56%. The benchmarks are modified such that the resolution can be updated on a per-frame basis using a global variable, with appropriate initializations redone each time the global variable is updated.

4.2 Experimental Setup

We experimentally demonstrate the operation of our cross-stack predictive control framework on a dual socket quad-core Intel Clovertown server. This server is equipped with an Intel Xeon processor X5365 with 8MB on-die L2 cache 1.333 GHz FSB and a 16 GB main memory. The processor supports four DVFS level: 3.0, 2.67, 2.33 and 2.0GHz. The operating system is Linux 2.6.36 kernel patched with Litmus-RT-2011. Litmus-RT implements several real-time multiprocessor scheduling and synchronization algorithms for Linux [13]. Each soft real-time task is mapped to a single thread and is independent of other tasks. In our work, the *x264* encoder grabs video frames periodically at 25 fps and *bodytrack* processes a new frame at 20 fps.

4.3 Sensor and Actuators

Actuators update the processor operational frequency and the application quality tuning knob values. Prior to applying the manipulated variables to the hardware and the application stack, the actuators filter these through a modulator to allow for fine-grained control. We use a first order delta-sigma modulator for the frequency actuator and a pulse width modulator for the application tuning knob actuator. Compared to the pulse width modulator, the first order delta-sigma modulator provides higher accuracy but incurs larger overhead due to oversampling. Hence first order delta-sigma modulators are more suitable for frequency actuators due to the small transition latency for DVFS (10 μs). However, the latency associated with the application tuning knobs may be up to 500 μs, precluding the use of oversampled techniques.

In each control period, the actuator reads the frequency and application tuning knob values from the controller, filter these through the modulator, and then writes the modulated values to the hardware and the application stack. Frequency actuator utilizes a Linux kernel subsystem called *cpufreq* to dynamically scale values of the operational frequency. Application quality actuator updates the tuning knobs, which are global variables protected by a real-time Flexible Multiprocessor Lock Protocol (FMLP) read-write lock [12]. FMLP is used to prevent deadlock and priority inversion in multiprocessor systems. In every control period, the MPC controller reads the system utilization from a sensor implemented as a custom Linux system call to calculate average per-core execution time over one control period.

4.4 System Identification

For both benchmarks, we obtain the system utilization for randomly generated combination of inputs including CPU frequency, application quality, and number of tasks for 400 control periods. The Hubble video [1] and the camera image sequence input from the Parsec benchmarks are used in executing *x264* and *bodytrack* respectively. We use the first half of working data for data modeling and the other half of data for validation. We use *n4sid* algorithm from the MATLAB System Identification Toolbox [23] to generate the state space models given in Equation 1. For our model, $n_x = 1$, $n_u = 2$ (frequency and application quality), $n_v = 1$ (number of tasks), $n_d = 1$ (job level variations in the execution time), and $n_y = 1$ (system utilization). The first-order model has a fit of 84.8% for *x264* and 87.4% for *bodytrack*.

4.5 Controller Design

The Model Predictive Controller is designed using the MATLAB MPC Toolbox [8]. The tunable parameters of the MPC controller include control horizon, prediction horizon, input and output weights, blocking modes, and disturbance model. The disturbance model is obtained by low-pass filtering a Gaussian white noise. We manually tune these parameters one at a time to obtain a good step response for the controller.

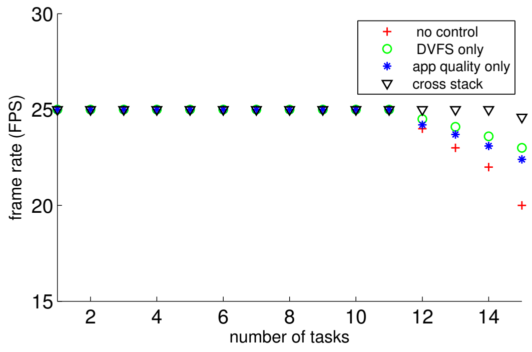
Table 1: Optimized parameter settings of the MPC controller

	x264	bodytrack
control horizon	2	4
prediction horizon	10	12
input weight	0, 0	0, 0
output weight	1	1
blocking	5	4
disturbance model	$\frac{1}{s+1}$	$\frac{1}{s+10}$

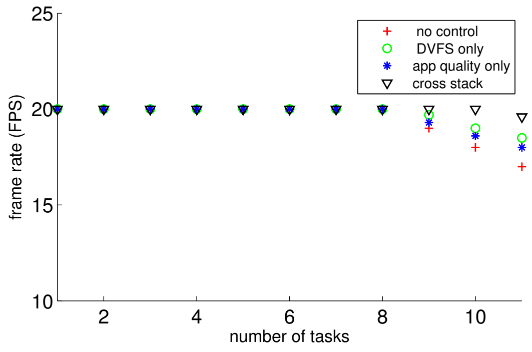
The optimized controller parameters for both *x264* and *bodytrack* are shown in Table 1. The control interval is chosen as 1 second, which is about 1/20-th of the open loop settling time. The closed loop poles of the unconstrained controller lie within the unit circle. The controller is implemented as a real-time task.

5 Experimental Results

Four our $m = 8$ core system, we set the task utilization set point arbitrarily to 4. Note that in a production system,



(a) x264



(b) bodytrack

Figure 2: Average FPS versus number of tasks - under no control, MPC control with DVFS-only, MPC control with application quality-only, and cross-layer MPC control

the choice of the set-point is dictated by the processor capacity required to run non-real time background tasks.

5.1 Homogeneous tasks

We demonstrate the need for the controller by measuring the average frame rate both with and without feedback control, as the workload is varied from light to heavy. A G-EDF scheduler is used to schedule all the tasks. As seen from Figure 2, in the absence of the controller, the frame rate drops beyond 11 tasks for *x264* and 8 tasks for *bodytrack*. For our 8 core system, the lowered frame rate indicates that the system is in overload. However, unlike the non-control case, the feedback controller is able to maintain a constant frame rate by automatically adjusting the processor frequency and the application quality. Figure 2 also shows the advantage of a cross-layer approach to feedback control as compared to deriving the control variables from a single layer of the computing stack. For both *x264* and *bodytrack* using DVFS-only or application quality-only as the control variable, results in a sharper drop in the frame rate with a heavier task load as compared to the cross-layer control.

For the *bodytrack* application, Figure 3 shows the controller′s step response to a change in the number of tasks from 5 to 9. As expected, the controller responds to a higher load by increasing the frequency while decreasing the visual quality to maintain the set-point within 5% with a peak overshoot of 29.7% and a settling time of 3.8 seconds. Since typical task change period is of the order of tens of seconds, this settling time suffices.

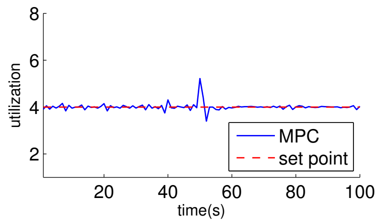
To evaluate the power savings obtained by our cross-layer control approach, we compare the average power consumption of the controller to the non-control case where the cores run at maximum frequency and the tasks operate at maximum application quality. The power savings are evaluated based on the power model described in [18], with the power proportional to the cube of the operating frequency. For a pseudo-random number of homogeneous input tasks, our model predictive control approach shows an average power saving of 31% compared to the non-control implementation for *x264* and an average power saving of 26% for *bodytrack*. The power saving is obtained at an average application quality of 70% for *x264* and 65% for *bodytrack*.

Table 2: Average FPS under C-EDF and G-EDF scheduler for a heterogeneous workload.

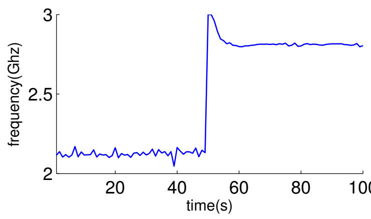
number of tasks		FPS of *x264*		FPS of *bodytrack*	
x264	*body-track*	C-EDF	G-EDF	C-EDF	G-EDF
2	2	25	25	20	20
2	8	25	25	15.8	20
10	2	20.1	25	20	20
8	6	25	23.1	20	18.3

5.2 Heterogeneous tasks

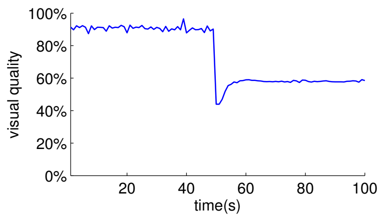
We also investigate the choice of the real-time scheduling algorithms on the performance of the the cross-layer feedback controller when the system hosts heterogeneous tasks from multiple applications (*x264* and *bodytrack*). In global-EDF scheduling, tasks from both the applications are scheduled globally across all 8 cores. In clustered-EDF scheduling, the applications run on separate clusters with tasks from a single application assigned to a cluster of 4 cores sharing the L2 cache. In both cases, separate controllers are designed for the two applications. Unfortunately, our hardware platform does not allow independent control of the core frequencies, limiting us to application quality as the only control variable for this experiment. Table 2 compares the average frame rate for G-EDF and C-EDF for different combination of number of tasks. For a balanced but light workload, both C-EDF and G-EDF achieve the targeted average FPS of 25 and 20 for *x264* and *bodytrack* respectively. For an



(a) Utilization versus time



(b) Processor operating frequency versus time



(c) Visual quality versus time

Figure 3: Experimental evaluation of model predictive control system step response for bodytrack. At t = 50s, the number of tasks changes from 5 to 9.

unbalanced workload, where the applications have dissimilar number of tasks, we note that G-EDF with its superior load balancing capability performs better. However, for a balanced but heavy workload with large number of tasks for both applications, load balancing is less of any issue. For this case, C-EDF with its better data locality performs better.

5.3 Non-linear tasks

Run-time characteristics of the two workloads depends on the video inputs. Since the *x264* controller was designed using a single video input, we investigate the performance of the controller for other types of video inputs. To characterize the similarity of two videos, we compare the probability distributions of the average execution times over a control period using the Kolmogorov-Smirnov (KS) [30] test. For 10 highly viewed test videos drawn from YouTube with content ranging from music, sports, movie clips, news reports, and documentaries, we observe that the controller performs well if the test video has a similar KS statistic to the input video. We note that for 8 videos out of 10 the controller shows acceptable performance. As a part of future work, we could implement a gain-scheduling approach, where separate controllers are designed for videos with significant difference in the KS statistic. A video classifier would periodically evaluate the video content and load the appropriate controller. The periodic classification approach could also be used to handle videos where the execution characteristic changes with time. A similar gain scheduling approach can also be applied to *bodytrack*.

5.4 Controller Overheads

The controller overheads include 1) computation cost of the MPC controller, 2) overheads due to DVFS and 3) real-time synchronization cost in modifying shared global variables in the application. The MPC controller uses a quadratic programming (QP) solver whose computation complexity is polynomial time in the product of number of control outputs and and prediction horizon. In one control period (1 second in our experiments), the core frequency is changed 20 times by the sigma-delta modulator. The overall DVFS overhead is measured using high resolution Linux timers and accumulated through all the subintervals within a control period. The synchronization occurs when the application parameters are updated. Our measurements indicate that for both *x264* and *bodytrack*, the overheads associated with DVFS dominates, and the total overhead is less than 0.4% of one control period.

6 Conclusions

In this paper we have demonstrated that a cross-layer approach to control enables the system to track large variations in the number of tasks allowing operation under heavily overloaded conditions while meeting timing requirements for soft real-time workloads. Moreover, the use of DVFS and application quality as control variables allows operation at a lower average power (and thereby lower energy for fixed run-time workloads) while meeting real-time constraints as compared to non cross-layer control approaches. Additionally, we show that the depending on the workload, the choice of the real-time scheduling algorithm impacts the performance of the controller. Also, the model predictive approach to control readily incorporates practical constraints on the output and control variables. Since the controllers are independent and have low overhead, the proposed framework scales well with the number of tasks and with multiple applications. Future extensions of our work would include gain scheduling for handling input dependent workload behavior and additional hardware and application parameters for a richer control of the system power-performance trajectories.

7 Acknowledgments

The authors would like to thank Dr. Tyrone Vincent for helpful suggestions.

References

[1] Hubble 20 years of discovery pt1-2 2010 nasa 1080 hd. `http://www.nasa.gov/mission_pages/hubble/science/hubble20th-img.html`, 2010.

[2] ABDELWAHED, S., BAI, J., SU, R., AND KANDASAMY, N. On the application of predictive control techniques for adaptive performance management of computing systems. *Network and Service Management, IEEE Transactions on 6*, 4 (2009), 212–225.

[3] ABDELZAHER, T. F., STANKOVIC, J. A., LU, C., ZHANG, R., AND LU, Y. Feedback performance control in software servicess. *Control Systems 23*, 3 (2003), 74–90.

[4] ABENI, L., AND BUTTAZZO, G. Hierarchical qos management for time sensitive applications. In *Real-Time Technology and Applications Symposium, 2001. Proceedings. Seventh IEEE* (2001), IEEE, pp. 63–72.

[5] ABENI, L., CUCINOTTA, T., LIPARI, G., MARZARIO, L., AND PALOPOLI, L. Qos management through adaptive reservations. *Real-Time Systems 29*, 2-3 (2005), 131–155.

[6] AYDIN, H., DEVADAS, V., AND ZHU, D. System-level energy management for periodic real-time tasks. In *Real-Time Systems Symposium, 2006. RTSS'06. 27th IEEE International* (2006), IEEE, pp. 313–322.

[7] BALAN, A. O., SIGAL, L., AND BLACK, M. J. A quantitative evaluation of video-based 3d person tracking. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on* (2005), IEEE, pp. 349–356.

[8] BEMPORAD, A., MORARI, M., AND RICKER, N. L. *Model Predictive Control Toolbox*. The MathWorks, 2005.

[9] BIENIA, C., KUMAR, S., SINGH, J. P., AND LI, K. The parsec benchmark suite: characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques* (2008), ACM, pp. 72–81.

[10] BITIRGEN, R., IPEK, E., AND MARTINEZ, J. F. Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach. In *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture* (2008), IEEE Computer Society, pp. 318–329.

[11] BLOCK, A., BRANDENBURG, B., ANDERSON, J. H., AND QUINT, S. An adaptive framework for multiprocessor real-time system. In *Real-Time Systems, 2008. ECRTS'08. Euromicro Conference on* (2008), IEEE, pp. 23–33.

[12] BLOCK, A., LEONTYEV, H., BRANDENBURG, B. B., AND ANDERSON, J. H. A flexible real-time locking protocol for multiprocessors. In *Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007. 13th IEEE International Conference on* (2007), IEEE, pp. 47–56.

[13] CALANDRINO, J. M., LEONTYEV, H., BLOCK, A., DEVI, U. C., AND ANDERSON, J. H. Litmusˆ rt: A testbed for empirically comparing real-time multiprocessor schedulers. In *Real-Time Systems Symposium, 2006. RTSS'06. 27th IEEE International* (2006), IEEE, pp. 111–126.

[14] CUCINOTTA, T., PALOPOLI, L., ABENI, L., FAGGIOLI, D., AND LIPARI, G. On the integration of application level and resource level qos control for real-time applications. *Industrial Informatics, IEEE Transactions on 6*, 4 (2010), 479–491.

[15] DEUTSCHER, J., AND REID, I. Articulated body motion capture by stochastic search. *International Journal of Computer Vision 61*, 2 (2005), 185–205.

[16] DEVI, U. C., AND ANDERSON, J. H. Tardiness bounds under global edf scheduling on a multiprocessor. *Real-Time Systems 38*, 2 (2008), 133–189.

[17] FU, X., KABIR, K., AND WANG, X. Cache-aware utilization control for energy efficiency in multi-core real-time systems. In *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on* (2011), IEEE, pp. 102–111.

[18] FU, X., AND WANG, X. Utilization-controlled task consolidation for power optimization in multi-core real-time systems. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2011 IEEE 17th International Conference on* (2011), vol. 1, IEEE, pp. 73–82.

[19] HOFFMANN, H., MAGGIO, M., SANTAMBROGIO, M. D., LEVA, A., AND AGARWAL, A. A generalized software system for accurate and efficient management of application performance goals. In *International Conference on Embedded Software (EMSOFT 2013)* (2013).

[20] HOFFMANN, H., SIDIROGLOU, S., CARBIN, M., MISAILOVIC, S., AGARWAL, A., AND RINARD, M. Dynamic knobs for responsive power-aware computing. In *ACM SIGPLAN Notices* (2011), vol. 46, ACM, pp. 199–212.

[21] KATO, S., RAJKUMAR, R., AND ISHIKAWA, Y. Airs: Supporting interactive real-time applications on multicore platforms. In *Real-Time Systems (ECRTS), 2010 22nd Euromicro Conference on* (2010), IEEE, pp. 47–56.

[22] LI, B., AND NAHRSTEDT, K. A control-based middleware framework for quality-of-service adaptations. *Selected Areas in Communications, IEEE Journal on 17*, 9 (1999), 1632–1650.

[23] LJUNG, L. *System Identification Toolbox*. The MathWorks, 1995.

[24] MAGGIO, M., HOFFMANN, H., SANTAMBROGIO, M. D., AGARWAL, A., AND LEVA, A. Controlling software applications via resource allocation within the heartbeats framework. In *Decision and Control (CDC), 2010 49th IEEE Conference on* (2010), IEEE, pp. 3736–3741.

[25] MESARINA, M., AND TURNER, Y. Reduced energy decoding of mpeg streams. *Multimedia Systems 9*, 2 (2003), 202–213.

[26] NOBLE, B. D., SATYANARAYANAN, M., NARAYANAN, D., TILTON, J. E., FLINN, J., AND WALKER, K. R. Agile application-aware adaptation for mobility. In *ACM SIGOPS Operating Systems Review* (1997), vol. 31, ACM, pp. 276–287.

[27] PADALA, P., SHIN, K. G., ZHU, X., UYSAL, M., WANG, Z., SINGHAL, S., MERCHANT, A., AND SALEM, K. Adaptive control of virtualized resources in utility computing environments. In *ACM SIGOPS Operating Systems Review* (2007), vol. 41, ACM, pp. 289–302.

[28] PARK, S. I., RAGHUNATHAN, V., AND SRIVASTAVA, M. B. Energy efficiency and fairness tradeoffs in multi-resource, multi-tasking embedded systems. In *Proceedings of the 2003 international symposium on Low power electronics and design* (2003), ACM, pp. 469–474.

[29] PILLAI, P., AND SHIN, K. G. Real-time dynamic voltage scaling for low-power embedded operating systems. In *ACM SIGOPS Operating Systems Review* (2001), vol. 35, ACM, pp. 89–102.

[30] PRESS, W. H. *Numerical recipes 2rd edition: The art of scientific computing*. Cambridge university press, 2002.

[31] QU, G., AND POTKONJAK, M. Energy minimization with guaranteed quality of service. In *Proceedings of the 2000 international symposium on Low power electronics and design* (2000), ACM, pp. 43–49.

[32] RAJKUMAR, R., JUVVA, K., MOLANO, A., AND OIKAWA, S. Resource kernels: A resource-centric approach to real-time and multimedia systems. In *Photonics West'98 Electronic Imaging* (1997), International Society for Optics and Photonics, pp. 150–164.

[33] ROY, A., RUMBLE, S. M., STUTSMAN, R., LEVIS, P., MAZIÈRES, D., AND ZELDOVICH, N. Energy management in mobile devices with the cinder operating system. In *Proceedings of the sixth conference on Computer systems* (2011), ACM, pp. 139–152.

[34] RUSU, C., MELHEM, R., AND MOSSÉ, D. Maximizing the system value while satisfying time and energy constraints. *IBM Journal of Research and Development 47*, 5.6 (2003), 689–702.

[35] SEGOVIA, V. R., ÅRZÉN, K.-E., SCHORR, S., GUERRA, R., FOHLER, G., EKER, J., AND GUSTAFSSON, H. Adaptive resource management framework for mobile terminals-the actors approach. In *Proceedings of the First International Workshop on Adaptive Resource Management (WARM), Stockholm, Sweden* (2010).

[36] SEO, E., JEONG, J., PARK, S., AND LEE, J. Energy efficient scheduling of real-time tasks on multicore processors. *Parallel and Distributed Systems, IEEE Transactions on 19*, 11 (2008), 1540–1552.

[37] SNOWDON, D. C., LE SUEUR, E., PETTERS, S. M., AND HEISER, G. Koala: A platform for os-level power management. In *Proceedings of the 4th ACM European conference on Computer systems* (2009), ACM, pp. 289–302.

[38] SRIKANTAIAH, S., KANDEMIR, M., AND WANG, Q. Sharp control: controlled shared cache management in chip multiprocessors. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture* (2009), ACM, pp. 517–528.

[39] YUAN, W., NAHRSTEDT, K., ADVE, S. V., JONES, D. L., AND KRAVETS, R. H. Grace-1: Cross-layer adaptation for multimedia quality and battery energy. *Mobile Computing, IEEE Transactions on 5*, 7 (2006), 799–815.