ATLANTIS PRESS

Advances in Engineering Research, volume 86

2nd International Conference on Electrical, Automation and Mechanical Engineering (EAME 2017)

Adaptive Power Mode Tuning for QoS Guarantee in Cyber-Physical Systems

Wei Zhang, Lihui Pan and Zhaohui Yuan*

School of Software, East China Jiaotong University, P.R. China

*Corresponding author

***Abstract*—The Quality of Service (QoS) of Cyber-physical systems (CPS) is inevitably undermined by various physical uncertainties which include the stochastic environment noise and dynamics of the physical phenomenon. Both of those uncertainties might lead to task overload and the power overconsumption as task execution times are unpredictable. While recent feedback control scheduling have shown promise to meet the real-time requirement by adaptively adjusting the task loading rate, little work has focused on both the system service rate and the CPU execution modes as a whole to optimize the system QoS. This paper presents a feedback control based scheme that adaptively maintains desired task service rate for real-time requirement while minimizes the power consumption by assigning proper execution modes on processor for each node. The theoretic analysis, numerous simulations and experimentations demonstrate that the proposed scheme can provide robust real-time guarantees and reduce the power consumption when the task workload vary significantly at run-time.**

Keywords-cyber physical systems; QoS; adaptive control

I. INTRODUCTION

Quality of Service (QoS) guaranteed Cyber-physical systems (CPS) which deployed in open and unpredictable environment have been rapidly growing. However, a key challenge in such systems is providing critical QoS guarantees imposed by users while the workload cannot be accurately estimated as a known condition.

While classical real-time scheduling approaches concern with statically assured avoidance of undesirable effects such as task overload and deadline misses [11][16], employing control theoretic approaches have shown its effectiveness in providing real-time guarantees in unpredictable environments [8][9]. However, existing works have only focused on meeting the deadlines constraint on the assumptions that the workload is fluctuating, they did not account for the energy consumptions in the whole systems.

Reducing energy consumption for QoS guaranteed systems has been extensively studied. One of the effective way is to assign proper operation modes to CPUs according to the task workload at runtime. Dynamic voltage frequency scaling (DVFS) solutions are first proposed in [18, 10], supply voltage and clock frequency is typically scaled down to achieve energy reduction with performance trade off. However, the above operating mode assignment approaches and traditional adaptive DVFS only consider how to lessen the processor energy consumption, i.e. defining a method for choosing the different frequencies of each processor [5, 13, 16], they have not focus on real-time requirements of applications.

In this paper, we proposed a novel approach to address execution mode tuning problem for real-time tasks in workload fluctuating CPS. Specifically, the main contributions of this paper are listed as following:

- We propose a novel problem formulation for the operation mode tuning where a given upper bound on the CPU utilization is enforced as well as the system power consumption is minimized.
- We develop feedback control based Operation Mode Tuning Scheme (OMTS) that adaptively adjusts the CPU frequency level to bound the CPU utilization according to user requirement.
- Theory analysis, extensive simulated experiments have been conducted for demonstrating the stabilization and convergence of the designed controller. The experiment results show that OMTS can achieve robust real-time performance as well as lower power consumption in an unpredictable environment.

The rest of the paper is organized as follows. The preliminaries are presented in Section II. A formalized description is proposed for mode assignment problem in section III. The controller and algorithm are described in Section IV. The simulation experiment is presented in Section V.

II. PRELIMINARIES

A. System Model

A typical CPS consists of multiple wireless connected sensors, actuators and some base nodes (or cluster heads). We assume that sensors sample signals omitted from the objects in physical environment periodically and send the measurements to their cluster heads, which are responsible to process those data and trigger actuators for further operation. However, the workload on cluster head fluctuates as the data sent by its cluster members can vary dramatically. To avoid the data congestion caused by processing delay, the cluster head is usually equipped with a buffer which act as a reservoir to store the sequence data received by sensors. The processor continually take out the front data segment from the queue after completed the former processing.

FIGURE I illustrates the architecture of the system, applications of it are extensive in CPS.

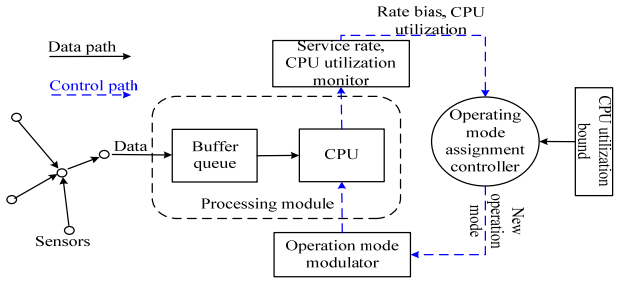


FIGURE I. THE ARCHITECTURE OF THE SYSTEM

B. CPU Power Model

By reducing processor supply voltage and clock frequency, we can reduce energy consumption at the cost of time-consuming performance. We assume each node supports m discrete execution modes. Each execution mode has different energy consumptions and corresponding execution times for the tasks when running at different frequency level.

TABLE I. XSCALE PXA270 POWER SPEC

Frequency	Active power	Idle power
624 MHz(208MHz system bus)	925mW	260mW
520 MHz(208MHz system bus)	747mW	222mW
416 MHz(208MHz system bus)	570mW	186mW
312 MHz(208MHz system bus)	390mW	154mW
312 MHz(104MHz system bus)	375mW	109mW
208 MHz(208MHz system bus)	279mW	129mW
104 MHz(104MHz system bus)	116mW	64mW
13MHz(CCCR[CPDIS]=1)	44.2mW	15.4mW

For example, as shown in Table I, when operating in different frequency levels, Intel Xscale PXA270 has different power requirements, and each mode may also have its own body bias setting to manage leakage power.

C. Task Model

The periodic task T_i is represented by a triple (r_i, e_i, d_i), where r_i is the start time of task T_i, e_i is the execution time, and d_i is the deadline of the task. The task assumed to be composed with a sequence of subtasks $\{T_i^j | 1 \le j \le n_i\}$, which can also be denoted by a quadruple $(r_i^j, e_i^j, d_i^j, m_i^j)$, where r_i^j is the start time of subtask T_i^j, e_i^j is the actual execution time of T_i^j, d_i^j is the relative deadline of the subtask, and m_i^j is the workload of T_i^j, which can be measured by times of standard float point operations, for example, m_i^j *mflops*. According to [9], the subtask T_i^j of a periodic task T_i is also periodic and T_i^j shares the same rate as task T_i. Obviously, we have $e_i = \sum_{j=1}^{n_i} e_i^j$, and $d_i = \sum_{j=1}^{n_i} d_i^j$.

TABLE II. SUMMARY OF NOTATIONS

Symbol	Definition
T_i / T_i^j	task i /the j^{th} subtask of task T_i
$r_i / e_i / d_i$	start/execution/relative deadline time of T_i
$r_i^j / e_i^j / d_i^j$	start/execution/relative deadline time of T_i^j
$Rate_T$/$Rate$	the loading/service rate of task
m_i^j	data volume of subtask T_i^j
$F(k)$	the frequent in the k^{th} sampling period
$U(k)$	the actual CPU utilization in the k^{th} sampling period

III. PROBLEM FORMULATION

The increasing complexity of workload in QoS guaranteed CPS lead to heavy variousness on system real-time and energy consumption which cannot easily be handled by means of the static characterization.

In an open application environment, the CPU working mode cannot easily be set as a fixed pattern. To avoid the task delay, the CPU utilization is imposed to indicate the system situation of workload. Suppose the CPU utilization can be calculated as follows [6];

$$U(k) = \sum_{s=1}^{n_k} e_s / d_s$$

where n_k is the number of processed tasks in k sampling period, e_s and d_s is execution time and deadline of tasks respectively. Normally, the expected utilization $U(k)$ cannot exceed than 100%, and should greater than a certain value δ (normally, δ are set to 75%) as to utilize most of the processing capabilities, while do not miss the deadline. Then, the energy-efficient operation mode assignment can be formulated as a constrained optimization problem:

Problem 1: Given a CPU utilization set point δ(δ ≤100), buffer size γ and a set of power mode $M_i = \{(t_i, p_i) | i = 1, ..., m\}$ of processor, where m is the number of modes, t_i is the execution time for mode i and p_i is the power consumption of CPU when running on mode i, p_{ki} is the power consumption of CPU on execution mode i at the k^{th} sampling period. The objective is to assign the proper execution mode for processor at runtime which minimize the whole power consumption of the system within n sampling period:

$$\min \sum_{k=1}^{n} p_{ki}, (i \in M_i) \qquad (1)$$

Subject to constraints:

$$(|Rate(k) - Rate_T|) \cdot T \le \gamma, k = 1, ..., n \qquad (2)$$

$$U(k) = \sum_{s=1}^{n_k} e_s / d_s \geq \delta \tag{3}$$

where T is the sampling period. In equation (2), the difference between the tasks loading rate *RateT* and the service rate of task $Rate(k)$ are bounded by buffer size γ such that the unprocessed data (delayed task) are not exceed the buffer size.

IV. ADAPTIVE MODE ASSIGNMENT SCHEME

A. Approach Overview

Feedback control has been widely employed to improve the adaptability of networked system. To address these challenges, we propose an Operation Mode Tuning Scheme (OMTS), which integrates feedback control theory. OMTS features the feed-back control loop to adaptively assign proper execution mode for tasks loading from the buffer, which is comprised of two parts, a proportional controller and a mode assignment algorithm.

B. Proportional Controller

We adapt a proportional controller as tactics to tune the task service rate to converge at the loading rate. In this controller, the system reference in the loop is $Rate_T$, the control variable is $Rate(k)$ and the input of the controller is the rate bias ΔR, which can be calculated as:

$$\Delta R(k) = Rate_T - Rate(k-1) \tag{4}$$

where *RateT* is the loading rate and *Rate(k – 1)* is the service rate at the former tuning period. According to control theoretic methodology, we first establish a dynamic model that characterizes the relationship between service rate $Rate(k)$ and the rate bias $\Delta R(k)$.

$$Rate(k) = Rate(k-1) + g * \Delta R(k) \tag{5}$$

where g is a constant, note that the exact value of g is unknown at first as the environment is unpredictable. However, its value may infect the convergence speed, and which can be determined by training when the controller is at work [6,8,9].

In the control system, there is a destabilization since unpredictable environment scenarios. The destabilization is integrated as a task workload $S(k)$, which is determined by the service rate $Rate(k-1)$ in last sampling period and environment variable (destabilization) e.

$$S(k) = e * Rate(k-1) \tag{6}$$

The destabilization will change the workload and also affect the best operation mode, since the CPU utilization should stabilizes around the set point δ. According to article [6], the relationship between utilization and frequency is:

$$U(k) = h * \frac{S(k)}{F(k)} \tag{7}$$

Based on the equation (7), the best work frequency can be calculated and we discuss details in next section.

C. Mode Assignment Algorithm

The main idea of mode assignment scheme is to choose a proper CPU frequency according to the task service rate which is adjusted by controller, the processing flow of the scheme can be described as follows:

Step 1. Calculate the workload in K^{th} sampling period by $S(k) = e * Rate(k-1)$.

Step2. Calculate baseline frequency of CPU by $F_1 = h\frac{S(k)}{100}$ and $F_2 = h\frac{s(k)}{U_0}(U_0 = \delta, F_2 > F_1)$.

Step 3. Choose optimal frequency of CPU (operating mode) by:

If $F_1 \geqslant F_{\max}$, then $F(k) = F_{\max}$, $U(k) = 100$.

Else if $F_2 \leqslant F_{\min}$, then $F(k) = F_{\min}, U(k) = h\frac{S(k)}{F\min} < U_0$.

Else $F(k) = 2h\frac{S(k)}{100 + U_0}$, the desired value F can be obtained by a combination of the supported discrete frequency levels to get optimal frequency $F(k)$, $U(k) = h * \frac{S(k)}{F(k)}$.

End if

Step 4. Calculate the service rate $Rate(k)$ by $Rate(k) = Rate(k-1) + g * \Delta(k)$.

Step 5. Report the new status information to the controller for further modulation.

V. SIMULATION

A. Experimental Setup

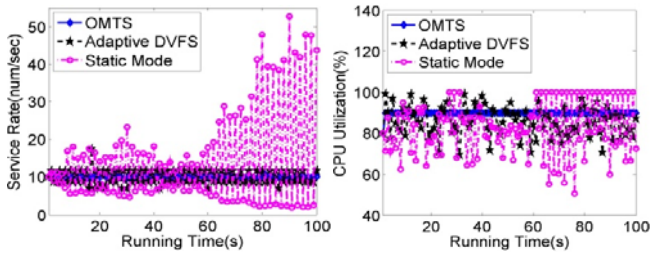
The simulation environment is composed of an event-driven simulator implemented in C++ and a controller implemented in MATLAB. Specifically, tasks are recognizing targets from sequential images captured by wireless cameras, data volume of tasks are generated with different pictures. According to the data sheet of Intel Xcale PXA 270, the entire parameters of operating modes of the processor are shown in Table I. We choose 8 operating modes from it to initialize the mode list M .

We employ two baselines for comparison. One is a typical adaptive DVFS based scheme which is similar to works in [5], [15] and [16]. More details can be known in them.

For comparing with the scheme with no adaptive control strategy, we also implement a static mode assignment scheme as the second baseline. In this baseline, system calculate the CPU utilization and missing rate after every sampling period,

and adjust the CPU mode to the neighbor level on the mode list according the utilization.

B. *Stability and Convergence*



(A)SERVICE RATE (B)CPU UTILIZATION

FIGURE II. STABILITY AND CONVERGENCE

For verifying the stability of the OMTS, we plot the evolution results of the task service rate which can be reached by different schemes on FIGURE II (a) , the x-axis represents the running time, sampling period is set as 1 second, and task loading rate is set to 10 tasks per second, CPU utilization set point is 90%. From the figure, we can see that OMTS and adaptive DVFS are stable at all time, while the static mode is unstable. The reason is that the static mode just tunes the execution mode heuristically, without any feed-back control strategy. However, adaptive DVFS can also achieve a relatively stable service rate as it employ adaptive control scheme.

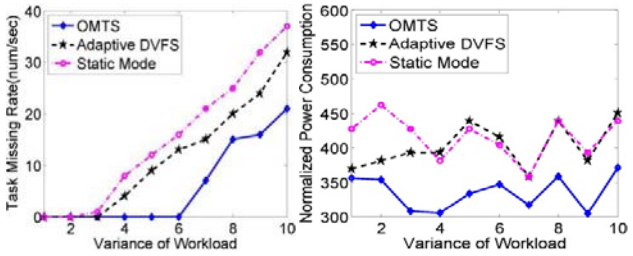
FIGURE II (b) plots the CPU utilization with running time when employing different schemes, the parameters are set as the same with simulations which plotted in FIGURE II (a). OMTS can stably maintain the CPU utilization to the set point, while the utilization of static mode assignment are fluctuated dramatically. However, some CPU utilization of adaptive DVFS is much higher than the set point, which will lead to the deadline missing as the task buffer is full while there is no time relaxation.

C. *Realtime Performance*

To verify the real-time performance of the OMTS, we compare the task missing rate of three schemes while the workload variance level changes from 1 to 10. The CPU utilization set points is 90%, and task loading rate is set to 10 per seconds. The variance of workload is generated by a standard Gaussian distribution data generating algorithm, we set the variance of the distribution from 1 to 10 and generate tasks with different volume as to simulate the fluctuation workload. FIGURE III (a) plots the average task missing rates of three schemes. From the figure, we can see that when the variance of workload is stable (variance from 0 to 3), three schemes have almost zero task missing rate, the reason is that the CPU utilization bound is set to 90% in the simulation, there is processing time relaxation. However, the task missing rate rising dramatically when the variance of workload increases, and OMTS has shown better performance than two baselines.

D. *Power Consumption*

Minimizing power consumption is another objective in this paper. To evaluate the power consumption performance of OMTS and other schemes, we plot the normalized power consumption of three schemes under different level of workload variance. The workload variance is also simulated by the same setting claimed in Section IV-C. We plot the results in FIGURE III (b), in which the power consumption are calculated by the mode chosen by OMTS and baselines, according to the power specification of PXA 270 mode listed in Table I. Averaging the power consumption of several times we have the normalized results. From the figure, we can see that OMTS are more power efficient than two baselines. When the variance of workload increases, OMTS can choose lower work frequency (operation mode) for reduce energy consumption, so the OMTS is more energy efficient than two baselines.



(A)Real-Time Performance (B)Power Consumption Performanced

FIGURE III. QOS PERFORMANCE

VI. CONCLUSION

OMTS features a performance-critical feedback control algorithm to handle the workload fluctuation problems in cyber-physical systems. The controller adaptively maintains desired task service rate for real-time requirement while minimize the power consumption by assigning proper execution modes on processor for nodes. The theoretic analysis, numerous simulations demonstrate that the proposed algorithm can provide robust real-time guarantees and reduce the power consumption when the environment vary significantly at run-time.

ACKNOWLEDGMENT

The work described in this paper was supported by the National Nature and Science Foundation of China under Grants 61562028.

REFERENCES

[1] http://www.amd.com/.

[2] http://www.armkits.com/download/pxa270datasheet.pdf

[3] http://www.intel.com/design/intelxscale/.

[4] http://www.transmeta.com/.

[5] Ryan Cochran, Can Hankendi, Ayse K Coskun, and Sherief Reda. Pack & cap: adaptive dvfs and thread packing under power caps. In Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture, pages 175‐185. ACM, 2011.

[6] Yong Liao, Xu-Dong Chen, Guang-Ze Xiong, Qing-Xin Zhu, and Nan Sang. End-to-end utilization control for aperiodic tasks in distributed real-time systems. Journal of Computer Science and Technology, 22(1):135‐146, 2007.

[7] A Lopez-Bravo, J Diaz-Carmona, A Ramirez-Agundis, A PadillaMedina, and J Prado-Olivarez. Fpga-based video system for real time moving object detection. In Electronics, Communications and Computing (CONIELECOMP), 2013 International Conference on, pages 92‐97. IEEE, 2013.

[8] Chenyang Lu, Xiaorui Wang, and Christopher Gill. Feedback control real-time scheduling in orb middleware. In Real-Time and Embedded Technology and Applications Symposium, 2003. Proceedings. The 9th IEEE, pages 37‐48. IEEE, 2003.

[9] Chenyang Lu, Xiaorui Wang, and Xenofon Koutsoukos. Feedback utilization control in distributed real-time systems with end-to-end tasks. Parallel and Distributed Systems, IEEE Transactions on, 16(6):550‑561, 2005.

[10] A. Manzak and C. Chakrabarti. Variable voltage task scheduling algorithms for minimizing energy/power. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 11(2):270‑276, 2003.

[11] Ragunathan Rajkumar, Lui Sha, and John P Lehoczky. Real-time synchronization protocols for multiprocessors. In RTSS, pages 259‑269, 1988.

[12] T Semertzidis, K Dimitropoulos, A Koutsia, and N Grammalidis. Video sensor network for real-time traffic monitoring and surveillance. IET intelligent transport systems, 4(2):103‑112, 2010.

[13] Vasileios Spiliopoulos, Stefanos Kaxiras, and Georgios Keramidas. Green governors: A framework for continuously adaptive dvfs. In Green Computing Conference and Workshops (IGCC), 2011 International, pages 1‑8. IEEE, 2011.

[14] Jun Sun and Jane Liu. Synchronization protocols in distributed real-time systems. In Distributed Computing Systems, 1996., Proceedings of the 16th International Conference on, pages 38‑45. IEEE, 1996

[15] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I‑511. IEEE, 2001.

[16] Qiang Wu, Philo Juang, Margaret Martonosi, and Douglas W Clark. Voltage and frequency control with adaptive reaction time in multiple-clock-domain processors. In High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on, pages 178‑189. IEEE, 2005.

[17] Zichen Xu, Xiaorui Wang, and Yi-Cheng Tu. Power-aware throughput control for database management systems.

[18] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. Proceedings of Symp. Foundations of Computer Science, pages 374‑382, Oct 1995.