Proceedings
Work-in-Progress Session

of the 33rd IEEE Real-Time Systems Symposium (RTSS'12)

December 4-7, 2012
San Juan, Puerto Rico

Edited by Thomas Nolte

Message from the WiP Chair

Dear Colleagues,

Welcome to San Juan and to the Work-in-Progress (WiP) Session of the 33rd IEEE Real-Time Systems Symposium (RTSS'12). I am pleased to present 25 excellent WiP papers that describe innovative research contributions from the broad field of real-time and embedded systems and applications. The 25 accepted papers were selected from 38 submissions. These proceedings are also published online in the ACM SIGBED Review (available at http://sigbed.seas.upenn.edu/), and as a Technical Report at Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Västerås, Sweden (available at http://www.mrtc.mdh.se/).

The primary purpose of the WiP Session is to provide researchers with an opportunity to discuss their evolving ideas and gather feedback from the real-time and embedded systems and applications community at large. The presentation session is limited in duration, and can only provide a brief overview of each WiP paper. I hope however that members of the audience will find ideas presented particularly interesting and want to participate in longer discussions with the authors during the poster session that follows.

I would like to thank the WiP Program Committee members, listed below, for their hard work in reviewing the papers.

Jian-Jia Chen	Karlsruhe Institute of Technology, Germany
Shinpei Kato	Nagoya University, Japan
Linh T.X. Phan	University of Pennsylvania, USA
Harini Ramaprasad	Southern Illinois University at Carbondale, USA

Special thanks also goes to Chenyang Lu, Luis Almeida, Giorgio Buttazzo, Enrico Bini and Oleg Sokolsky for their support and assistance.

Thomas Nolte
Work-in-Progress Chair
33rd IEEE Real-Time Systems Symposium (RTSS'12)

Table of Contents

TEMPO: Performance Viewpoint for Component-Based Design of Real-Time Systems

Rafik Henia
THALES Research & Technology
F-91767 Palaiseau – France
Rafik.Henia@thalesgroup.com

Laurent Rioux
THALES Research & Technology
F-91767 Palaiseau – France
Laurent.Rioux@thalesgroup.com

Nicolas Sordon
THALES Research & Technology
F-91767 Palaiseau – France
Nicolas.Sordon@thalesgroup.com

The growing complexity of applications, combined with constant quality and time-to-market constraints, creates new challenges for performance engineering practices in the area of real-time embedded systems. It is namely expected that delivered products implement more and more complex features, while respecting strict real-time requirements. When developing such real-time systems according to a traditional application of the "V"-cycle, performance verification and validation activities start only when development and integration are completed. As a consequence, performance issues are not detected until the validation and verification stage starts. At this time, they are more difficult and expensive to fix. Thus, a reliable performance prediction at early design stages is essential to guarantee that the designed system meets its timing requirements before time and resources are invested for the system implementation. However, this is not an easy task due the lack of methodologies allowing to directly apply performance engineering activities to design models.

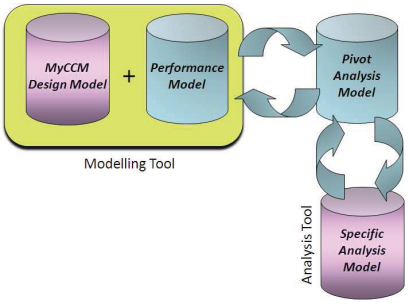


Figure 1 Performance Viewpoint for MyCCM design

At Thales, we are developing a performance Viewpoint, called TEMPO, for the integration and application of performance engineering activities as early as possible in the design process of real-time embedded systems, as a mean to shorten the design time and reduce risks of timing failures. The design of such systems is supported in Thales by a software framework, named MyCCM (Make your Component Container Model). MyCCM is a tailorable component based design approach that takes inspiration from the Lightweight Component Container Model (LwCCM) defined by the OMG. It implements the concept of functional components that encapsulate algorithms and are connected through communication ports to create a complete application. MyCCM models are described using UML modelers. The TEMPO Viewpoint for early performance estimation of MyCCM design models is represented in Figure 1. The first step consists in extending the MyCCM model with performance properties, i.e. timing and behavior characteristics of the application (e.g. execution times and activation frequencies for threads, communication protocols between threads) and execution characteristics of the hardware platform (e.g. scheduling policy). The OMG standard MARTE is key technology for this purpose. However, due to the complexity of its syntax, it may result in very complex and confusing diagrams and models. We have therefore adapted the MARTE syntax based on the Thales designers' feedbacks, thus allowing representing the performance properties in an easier and much more intuitive manner.

We have opted for scheduling analysis techniques for the performance estimation of the extended MyCCM models. These techniques are well adapted for this purpose, since they rely on an abstraction of the timing relevant characteristics and behaviors. From these characteristics, the scheduling analysis systematically derives worst-case scheduling scenarios, and timing equations safely bound the worst-case response times. However, we faced the problem that scheduling analysis is not directly applicable to extended MyCCM models due to the semantic mismatch between the later and the variety of analysis models known from the classical real time systems research and from the industrial scheduling analysis tools. For instance, in analysis models, a standard assumption is that a task writes its output data at the end of its execution. In contrast, in MyCCM models, a sender task may write data into the input FIFO of a receiver task and continue executing (case of asynchronous communication). In order to fill this large semantic gap and also to ensure a minimum of independence from modeling and analysis tools, we have decided to introduce a pivot analysis model in-between. For this purpose, we have specified a set of rule transforming extended MyCCM models into equivalent pivot analysis models and the later into the selected analysis tool models to which a dedicated MyCCM scheduling analysis, we have developed, is applicable. For instance, in the case of asynchronous communication mentioned above, the transformation consists in splitting the sender task in two tasks. Each of them inherits a subset of the sender task behavior: the first task terminates after writing data into the input FIFO of the receiver task, while the second task is activated immediately after the termination of the first task.

We are next planning to define rules adapting the analysis results back to the original MyCCM design model. This would allow a fully automated and to the user completely transparent scheduling analysis for MyCCM designs. Further details on our performance viewpoint for component-based design, including the model transformation rules and the MyCCM specific scheduling analysis will be addressed in more detail in the conference presentation.

SPM-Aware Scheduling for Nested Loops in CMP Systems

Zhi Chen, Meikang Qiu*
Dept. of Elec. and Comp. Engr., Univ. of Kentucky, Lexington, KY 40506, USA *

1. PROBLEM AND MOTIVATION

Chip multiprocessors (CMP) computing systems are usually employed to facilitate many specific applications including medical image processing, computer vision, and aerospace. In these computation-intensive applications, nested loops take the most significant section of computation cost and greatly affect system performance in terms of latency due to the frequent memory accesses. In order to enhance the parallelism of a nested loop, a critical work is to strategically map the iterative loops to processors so that we can exploit good parallelization of these loops and reduce the execution latency of the whole application. One of the most widely used method to do the iteration-to-processor mapping is pipelining, which enables each processor to perform the operations for the iteration mapped to it.

2. BACKGROUND AND RELATED WORK

Most prior work on nested loops either focuses on loop partitioning such as tiling, collapsing, and transformation, etc, or focuses on scheduling loops on different processors with hardware caches. However, hardware caches have shortcomings in size, power, and predictability and incur high penalties in cache misses. Statistically, caches are replacing the processors becoming the main energy consumer of computing systems. SPM, a software-controllable on-chip memory, has been widely utilized by many key manufacturers, due to two major advantages over a cache memory.

First, SPM does not have the comparator and tag SRAM, since it is accessed by direct addressing. Therefore, it does not perform the complex decode operations to support the runtime address mapping for references. This property of SPM can save a large amount of energy. Second, SPM generally guarantees single-cycle access latency, while accesses to cache may suffer capacity, compulsory, and conflict misses that incur very long latency [2]. Given these advantages, caches are widely used in CMP systems including Motorola M-core MMC221, IBM CELL [1], TI TMS370CX7X, and NVIDIA G80. Therefore, it is important to study the loop scheduling with SPM support to reduce the latency and energy consumption of nested loops on CMP system.

Although the obvious advantages SPM offers, two major challenges remain: 1) what is the basic granularity of the loop scheduling? 2) how to schedule nested loops on different SPMs of each processor in a CMP systems? Since there are a large amount of previous work targets nested loop partitioning, we mainly focus on the iteration-to-processor mapping by using the partitioning results from the existing techniques. Also, we are targeting the type of applications that intensively consist of loops manipulating arrays.

3. APPROACH AND UNIQUENESS

Depending on the partitioning technique such as cyclic partitioning and block partitioning, the granularity of partitioned loops varies. We focus on the block-based loops which can be obtained by using tiling techniques. Based on the obtained blocks, we construct a directed graph to represent the dependencies between them. Communication overhead between dependent blocks and the number of accesses to each processor can be learned with the help of profiling tools. Then, based on profiling information, a communication matrix and a processor access matrix can be built. We also define an assignment matrix to represent the mapping of each loop block to the SPM of each processor. The dimension of the assignment matrix is $N \times M$, where N and M represent the number of partitioned loop blocks and the number of processors on the target system, respectively.

In this paper, we will consider three types of on-chip memory access for loop scheduling: local access, remote access, and main memory access. Different memory access has different energy consumption and latency. In addition, the communication cost between different blocks will also be factored into the total execution cost. Since the target system is a CMP system, where each processor is attached with an on-chip SPM and all processors share an off-chip memory, there exist totally four kinds of communication: intra-memory communication, inter-memory communication, on-chip/off-chip memory communication, and off-chip memory communication. The overhead of each type of communication varies significantly. Our goal is to map all loop blocks in an applications to the on-chip SPMs and the off-chip main memory so that the total cost (either energy or latency) can be minimized.

A naive method is to distribute the loop blocks across available on-chip SPM evenly. However, this is not the best to implement the iteration-to-processor mapping for most of applications because different iteration blocks usually take different number of execution cycles and different amount of power, due to the existence of branches in loops and locality of on-chip SPMs. This paper carefully considers the characteristics of each loop block and proposes a dynamic programming algorithm to optimally perform the iteration-to-processor mapping. We will formally define the cost of memory access and communication, and use the availability of on-chip SPM resources to constrain the allocation.

4. RESULTS

Extensive experiments will be conducted to verify our algorithms on target a CMP system with 2, 4, and 8 cores. A host of benchmarks, such as MiBench and SPEC2006, will be used to evaluate the proposed strategies. We will also compare the efficiency of our dynamic programming algorithm with several other methods proposed in the literature. All these algorithms will be implemented as stand-alone programs, which take memory traces as inputs. Memory parameters are obtained from CACTI tools. Since the objective for this paper is to reduce the total memory access cost with respect to energy and latency, an array of results will be achieved to show how much on-chip memory energy and time (average and WCET) our dynamic algorithm can save.

5. REFERENCES

[1] C. R. Johns, et al. Introduction to the cell broadband engine architecture. *IBM Journal of Research and Development*, 51(5):503–520, 2007.

[2] P. R. Panda, et al. Efficient utilization of scratch-pad memory in embedded processor applications. In *IEEE/ACM DATE*, pages 7–11, 1997.

*Meikang Qiu (mqiu@engr.uky.edu) is the corresponding author, supported by NSF CNS-1249223.

Heterothread: Hybrid Thread Level Parallelism on Heterogeneous Multicore Architectures

Chao Wang
School of Computer Science
Univ. of Sci. & Tech. of China
chao.wang@ieee.org

Xi Li
School of Computer Science
Univ. of Sci. & Tech. of China
llxx@ustc.edu.cn

Xuehai Zhou
School of Computer Science
Univ. of Sci. & Tech. of China
xhzhou@ustc.edu.cn

ABSTRACT
In this paper, we introduce middleware architecture to support hybrid thread level parallelism on heterogeneous multicore architectures, called Heterothread. Heterothread constructs a hierarchical level model for user programming and parallel task execution dataflow. Heterothread can provide unified programming interfaces which allow applications remain the same when the physical hardware (CPU, GPU, DSP and FPGA) is reconfigured or migrated. Consequently it can meet the real-time demands more efficiently than most state-of-the-art operating systems. A prototype has been built on FPGA to demonstrate Heterothread can achieve significant speedups against Linux kernels.

Categories and Subject Descriptors
D.4.1 [**Operating Systems**]: *Multiprocessing/multiprogramming/*

General Terms
Performance, Design

Keywords
Architectural support; middleware; dynamic reconfiguration;

1. INTRODUCTION
Heterogeneous architecture is dominating [1]. However, it still poses a significant challenge to build a moderate operating system or middleware to manipulate threads running on different architectures efficiently (e.g. CPU, GPU, DSP, and FPGA). Current ongoing projects like fos [2] and barrelfish [3] is still pursuing a fine approach. With respect to the real time demand on heterogeneous architectures, how to obtain sufficient timing predictability is still posing significant challenge. In order to tackle this problem, the major contribution of this paper is to propose a middleware hierarchical model to support hybrid MPSoC reconfigurations. Heterothread provides acceleration engines to diverse applications by substituting IP cores dynamically. After hardware is reconfigured, Heterothread will reorganize the task partitioning, mapping and scheduling strategies, which keep the APIs unchanged to users. The entire architectural model consists of both software and hardware thread level controller modules.

First, to provide an execution environment for tasks, Heterothread employs run-time libraries. These user libraries along with other kernel libraries provide well-structured application programming interfaces (APIs). APIs show a high-level abstract view of the internal implementations. When the APIs are defined, the user interfaces will be kept as persistent units, even the hardware (DSP, FPGA and GPU kernel) replacement is invisible to programmers.

Task partitioning and scheduling methods play a vital role in Heterothread. Before tasks are offloaded to acceleration engines, Heterothread must decide which task runs on which core, and also when the task is issued. For the sake of automatic thread level parallelism, we use out-of-order task execution engine to solve the inter-task data dependencies [4].

Finally, the communication interfaces layer is in charge of data transmission between Heterothread scheduling kernel and the diverse processing kernels. Generally there are three kinds of primitives: the unified software interface (USI), unified hardware interface (UHI), and unified reconfiguration interface (URI): USI primitive is employed when the information in transferred among microprocessors, including a series of function in libraries. UHI primitive is introduced to model the communication between microprocessor and hardware RTL implemented IP cores or DSP engines. Furthermore, an interrupt controller is employed to detect interrupt requests for synchronization. Finally, URI primitive is invoked only for IP reconfiguration to switch the pre-downloaded partial bitstream at run-time.

We have constructed our prototype OS using with Xilinx FPGA board. As a working-in-progress paper, we are still running large scale Benchmarks on both a Minicore system of Heterothread and Linux 2.4.18 operating system kernel. At the time of writing this paper, we have evaluated the Anubis, 10-step FIR, and 20-step FIR applications on the Minicore OS, which achieve the speedup from 4.5x to 170x, respectively. Preliminary results demonstrate Heterothread provides new insights to the real time operating system research paradigms on heterogeneous multicore architectures.

2. ACKNOWLEDGEMENT
This work was supported by the NSFC grants (No. 61272131, No. 61202053), Jiangsu Provincial NSF (No. SBK201240198).

2. REFERENCES
[1] S. Singh, Computing without processors [J]. *Communications of ACM,* 2011. 54(8): p. 46-54. DOI= http://doi.acm.org/10.1145/1978542.1978558.

[2] D. Wentzlaff,A. Agarwal, Factored operating systems (fos): the case for a scalable operating system for multicores[J]. *ACM SIGOPS Operating Systems Review,* 2009. 43(2): p. 76-85. DOI= http://doi.acm.org/10.1145/1531793.1531805

[3] J. C. Mogul, A. Baumann, T. Roscoe, L. Soares. Mind the gap: reconnecting architecture and OS research. in Proceedings of the *13th USENIX conference on Hot topics in operating systems*.2011.

[4] G. Gupta,G. S. Sohi. Dataflow execution of sequential imperative programs on multicore architectures. *in Micro 44.* 2011.59-70. DOI= http://doi.acm.org/10.1145/2155620.2155628

Autonomic Computing Architecture for Real-Time Medical Application Running on Virtual Private Cloud Infrastructures

Yong woon Ahn, Albert Mo Kim Cheng
Department of Computer Science
University of Houston
4800 Calhoun Road, Houston, Texas, U.S.A.
+1-713-743-3350
{yahn, cheng}@cs.uh.edu

ABSTRACT

Cloud computing with virtualization technologies has become a huge trend which attracts academia and information technology industries because of its cost-efficiency. It has changed paradigms of development, release, and maintenance of diverse types of software and service. However, this big movement has not been applied to real-time applications yet because deploying real-time applications on the cloud infrastructures arouses many controversies because of numerous uncertainties from sharing physical resources. It is not trivial to apply conventional scheduling techniques for real-time systems to cloud infrastructures without further considerations. All virtual machines (VMs) must be controlled by the Virtual Machine Monitor (VMM) which is centralized and has a primary role to share physical resources fairly with all VMs in the same physical machine (PM). For best-effort applications, this fair resource sharing policy works well and end-to-end Quality of Service (QoS) is promised by the service level agreement (SLA) with reasonable delay windows. However, for real-time applications with aperiodic hard- and soft-real-time tasks, this mechanism has serious weaknesses. Although VMMs of most cloud infrastructures have auto-scaling and load-balancing mechanisms, these are unpredictably slow to accept urgent aperiodic-real-time tasks because of its fair resource sharing policy. Therefore, it is necessary to propose another approach to satisfy deadline constraints of real-time tasks transferred from remote locations. In this research, we focus on cloud medical applications processing sensitive patient data with deadline constraints. We propose feasible solutions to reserve computing and networking resources with an autonomic computing architecture in the virtual private cloud infrastructures.

Keywords

Auto-Scaling, Cloud Computing, Real-Time Systems, Medical Application, Virtualization, Autonomic-Computing

1. INTRODUCTION

When you run your medical application handling sensitive data on the public cloud infrastructure, your primary concern is how to protect your data from other computing components running with your instances because they have to share limited physical computing power, storage, and network bandwidth with their unknown neighbors. This problem can be resolved by using the virtual private cloud infrastructure [1] which has specialized mechanisms to support virtual private network protecting sensitive data. However, the virtual private network cannot resolve issues to support real-time applications with periodic and aperiodic tasks streamed from remote locations because a centralized VMM still needs to control VMs with its auto-scaling and load-balancing mechanisms satisfying its fair resource sharing policy. Therefore, a new approach is required to support reserving virtual resources on time.

*Supported in part by the National Science Foundation under Awards No. 0720856 and No. 1219082.

2. AUTONOMIC COMPUTING

In order to design a cloud infrastructure to run real-time applications on it, the most important consideration is how to scale up virtual resources to process real-time tasks, and scale down them to save costs. With conventional VMMs, it would be hard to monitor and control VMs which possibly are located on different physical machines. Therefore, we consider a distributed VM monitoring method with an autonomic computing architecture [2] for virtual private cloud infrastructures. An autonomic computing architecture has four phases to process tasks input from managed resources as shown in Figure 1. Also it can adjust its own system parameters to optimize processing performance and internal resource usages. We focus on the fact that these managed resources can be other autonomic computing architectures to build proper orchestration. In our research they can be multiple VMs. Moreover, we design each VM to be greedy and accept more tasks to process, and it always tries to steal real-time tasks from its child VMs, if its parent VM can process the same type of task. As a result of our approach, each VM and eventually each group of VMs can be optimized to process real-time tasks with lower costs. An auto-scaling mechanism also can be implemented with this approach. If one VM cannot handle tasks from its managed resources, it can invoke a new child VM. Since, the parent VM is greedy, the child VM is terminated when it has few jobs.

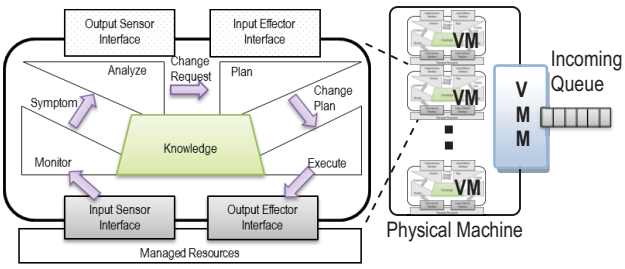


Figure 1. Autonomic computing architecture for VMs

3. CONCLUSION

In this research, we focus on running real-time medical applications on the private cloud infrastructures. In order to design and implement a scalable and reliable auto-scaling mechanism, we apply an autonomic computing concept to our system.

4. REFERENCES

[1] Miyamoto, T., Hayashi, M., Nishimura, K. Sustainable Network Resource Management System for Virtual Private Clouds. IEEE CloudCom, pp. 512-520. 2010.

[2] IBM, "Autonomic computing: IBM's perspective on the state of information technology," IBM Corporation, 2001.

Applying Language-based Static Verification in an ARM Operating System

Work-in-Progress

Matthew Danish Hongwei Xi Richard West
Boston University
Computer Science Department
111 Cummington Mall
Boston, MA 02215
{md,hwxi,richwest}@cs.bu.edu

1. EXTENDED ABSTRACT

In recent years, we have seen a proliferation of small, embedded, electronic devices controlled by computer processors as powerful as the ARM®. These devices are now responsible for tasks as varied as flying a plane, talking on a cellphone, or helping to perform surgery. Some of these tasks have severe consequences for a mistake caused by faulty programming or missed deadlines. The best defense against these mistakes is to prevent them from happening in the first place.

Advances in programming languages and type theory have lead to the creation of functional programming languages such as ATS [3] which are designed to combine theorem proving with practical systems programming. This allows programmers to bring the rigor of mathematical verification to important properties of their programs. We believe that the usage of languages such as ATS can lead to better assurance that the programs running on an embedded device are correct, responsive, and safe.

A key feature of ATS is that it allows an incremental approach to verification, where some parts of the program are more heavily checked than others, making it easier to make changes and focus proof-writing efforts on the most important portions. Also, the ATS compiler produces C code, it uses native data-layouts, and the ATS language itself is designed to be extremely easy to integrate with existing C code. This makes ATS much better suited than other, similar high-level languages, for systems programming.

Terrier [1] is a new operating system written to run on some of the ARM-based SoC boards from Texas Instruments®: the OMAP3530 and the OMAP4460. These boards are available to developers as the BeagleBoard and the PandaBoard ES respectively. In addition, Terrier supports the Cortex-A9 MPCore symmetric multiprocessor architecture which is used by the OMAP4460. The focus of Terrier is to be a small (currently about 5000 lines of source) and lightweight operating system for real time applications, by using some statically verified algorithms for scheduling and resource management.

Terrier includes an implementation of a Liu and Layland-based [2] rate-monotonic fixed priority scheduler, written in the functional programming style of ATS. The code is annotated with static, dependent and linear types to describe several of the invariants associated with the algorithm. The scheduler written in ATS integrates directly with the rest of the kernel which is written mostly in C. Performance of the scheduler implementation in ATS is essentially equivalent to an implementation in C because the output of the ATS compiler is C code that does not need any additional runtime support.

We are working on extending the rate-monotonic scheduler with stronger proofs and more features. One goal is to show that the algorithm respects some form of temporal isolation. Another goal is to adopt some more modern scheduling algorithms and prove useful properties for them. And finally, we are working on some applications which would potentially test the scheduling capabilities of the system while also demanding strong correctness guarantees.

2. REFERENCES

[1] Matthew Danish. Terrier Homepage. `http://www.github.com/mrd/terrier`.
[2] C. L. Liu and James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM*, 20(1):46–61, January 1973.
[3] Hongwei Xi. ATS Homepage. `http://www.ats-lang.org`.

Improving Schedulability and Energy Efficiency for Real-Time Systems with (m,k)-Guarantee

Linwei Niu
Department of Computer and Electrical Engineering and Computer Science
California State University Bakersfield
Bakersfield, CA
lniu@csub.edu

Kuai Xu
School of Mathematical and Natural Sciences
Arizona State University
Glendale, AZ
kuai.xu@asu.edu

ABSTRACT

In this paper, we explore improving the schedulability and energy performance for real-time systems with (m,k)-constraints, which require that at least m out of any k consecutive jobs of a task meet their deadlines. The preliminary results demonstrated that our proposed approach is very promising in achieving these dual goals.

1. INTRODUCTION

Many real-time applications, such as multimedia processing and real-time communication systems, can tolerate occasional deadline misses, which can be modeled using (m,k)-model [1]. In this paper, we explore improving the schedulability and energy efficiency for such kind of systems based on EDF scheme.

Two widely used mandatory/optional partitioning techniques for (m,k)-model are R-pattern and E-pattern [3]. It is known that E-pattern has much better schedulability than R-pattern due to its even distribution of mandatory jobs [3]. In [4], Quan *et al.* showed that the schedulability of E-pattern could be improved by shifting the E-pattern. However, there still exists a large number of task sets that can not be scheduled with any shifted patterns. For example, for the task set in Figure 1(a), however we shift the E-pattern such as in Figure 1(b), the task set can not be schedulable. Also in [4], a genetic algorithm was introduced to find the general (m,k)-pattern for task sets based on fixed-priority scheme. However, their approach is not applicable to EDF case. In this paper, we developed a Simulated Annealing (SA) algorithm to solve the problem for systems based on EDF scheme, which is based on the following concept:

DEFINITION 1. ***(Intensity)*** *For a real-time task set whose mandatory jobs are determined by the given (m,k)-pattern, the intensity in the time interval $[t_s,t_f]$ is defined to be*

$$I(t_s,t_f) = \frac{\sum_i M_i(t_s,t_f) \times C_i}{t_f - t_s} \quad (1)$$

where C_i is the worst case execution time of task τ_i, and $M_i(t_s,t_f)$ is the number of mandatory jobs of τ_i which are released at or after t_s, and have deadlines less than or equal to t_f.

With the definition of intensity, our SA algorithm works by iteratively adjust the mandatory/optoinal job patterns such that the highest intensity of the system could be minimized. For example, if we apply our SA algorithm to the task set in Figure 1(a), after some iterations, the job patterns will eventually converge to the configurations similar to that in Figure 1(c) or Figure 1(d). And both (with equal highest intensities of 0.92) are schedulable. Moreover, since energy consumption is a convex function of the processor speed, which is closely related to the highest intensity of the system, the energy efficiency of the resulting task set could also be improved.

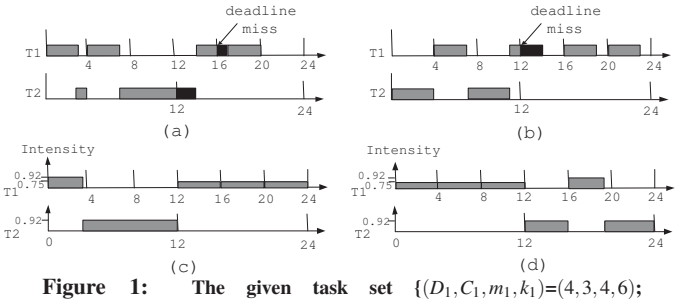


Figure 1: The given task set $\{(D_1,C_1,m_1,k_1)$=$(4,3,4,6)$; (D_2,C_2,m_2,k_2)=$(12,8,1,2)\}$: (a) NOT schedulable with E-pattern; (b) NOT schedulable with shifted E-pattern; (c) and (d) schedulable with Non-evenly distributed patterns achieved by our SA algorithm.

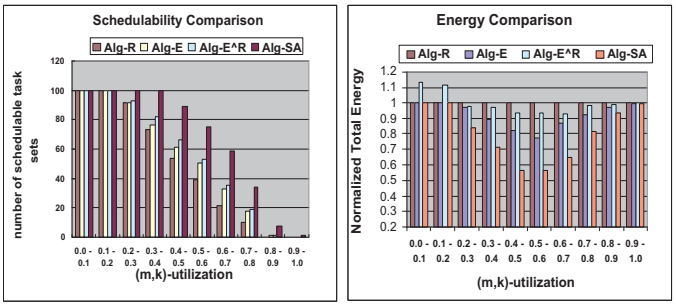


Figure 2: (a) shedulability comparison; (b) energy comparison.

2. PERFORMANCE AND CONCLUSIONS

Our experiments are based on the same task model in [3] and processor model in [2]. From Figure 2(a), for all (m,k)-utilization intervals, our SA algorithm, *i.e.*, **Alg-SA** has much better schedulablity than the approaches based on existing patterns, *i.e.*, **Alg-R** (based on R-pattern), **Alg-E** (based on E-pattern), and **Alg-E^R** [3]. Moreover, from Figure 2(b), compared to the previous approaches, significant energy reduction could be achieved by our approach.

It is also noted that SA is inherently a sequential approach and hence can be slow for problems with large search spaces. As part of our current and future work, we are exploring the parallel simulated annealing techniques on multi-core platforms to further improve the timing and energy performance of our proposed SA approach.

3. REFERENCES

[1] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m,k)-firm deadlines. *IEEE Transactions on Computers*, 44:1443–1451, Dec 1995.

[2] INTEL-XSCALE, 2003.

[3] L. Niu and G. Quan. Energy minimization for real-time systems with (m,k)-guarantee. *IEEE Trans. on VLSI*, pages 717–729, July 2006.

[4] G. Quan and X. Hu. Enhanced fixed-priority scheduling with (m,k)-firm guarantee. In *RTSS*, pages 79–88, 2000.

Online OLED Dynamic Voltage Scaling for Video Streaming Applications on Mobile Devices

Mengying Zhao
City University of Hong Kong

Xiang Chen
University of Pittsburgh

Yiran Chen
University of Pittsburgh

Chun Jason Xue
City University of Hong Kong

ABSTRACT

This work proposes an online DVS approach for OLED-based mobile video applications to reduce display power consumption. A time-efficient representative-region based DVS scheme is developed and applied in MPEG video streaming. Based on the proposed scheme, flexible DVS solutions can be adaptively derived according to timing constraints. Experimental results show a 26.9% power reduction while keeping 99.3% frames displayed in high quality.

1. PROBLEM ANALYSIS

As a basic component of mobile devices, display panel dominates the total power consumption. OLED (Organic Light Emitting Diode) is believed to be the replacement of LCD (Liquid Crystal Display) in mobile device displays due to the improved power efficiency and stable display quality. Different from LCD, OLED has color-dependent power consumption and self-illuminating feature, thus no backlight is needed. Due to the inherent differences, previously proposed LCD DVS (Dynamic Voltage Scaling) approaches are not effective for OLED. Researches of OLED DVS strategies are consequently explored. For power-efficient display, static images can be divided into partitions, and local voltage levels are adjusted accordingly for each partition [2][4]. An offline DVS solution for video sequences is proposed recently [3]. However, it is computation intensive and not applicable for online video streaming. For real-time processing, the procedures of downloading, decoding, processing and displaying cooperate with each other and share the time intervals between adjacent frames during the video streaming. In this work, we propose an online OLED DVS scheme for video streaming applications on mobile devices, to minimize the display power consumption.

2. METHODOLOGY

For MPEG video sequences, the proposed approach identifies frame types and applies corresponding processing strategies accordingly. For an I frame, which is the first frame in one GOP (Group of Pictures) and the reference of the following B or P frames, DVS decisions are made after it is completely decoded because of its intra compression encoding nature. For the following P and B-frames, the voltage levels of the reference frames are either directly applied to P and B frames in this GOP, or modified to fit to the difference. To cope with timing constraints and leverage OLED color-dependent power consumption character, in this paper, the DVS is done only for selected representative regions. Representative regions are defined as those with large potential to achieve high power saving and small quality loss. Display quality is evaluated by SSIM (structural similarity index). The processing strategy is determined by the frame type (I, or non-I), then the number of representative regions is decided by the local deadline. Among the regions previously divided from one frame, corresponding representative regions are selected based on color-dependent power consumption and the SSIM quality. Only the selected representative regions are processed for voltage scaling so as to meet the timing constraints. Consequently, flexible online DVS solutions can be generated by the proposed technique.

3. SIMULATION RESULTS

We tentatively evaluate the effectiveness of the proposed OLED DVS technique with four different types of video streams, all of which have the resolution of 640×360 and refresh rate of 30fps. The offline optimal technique presented in [3] is used in the experiments as comparison reference.

Table 1: Display power saving.

testbench	offline[3](%)	online(%)	online/offline(%)
News	36.7	15.7	43.1
Cartoon	35.2	17.9	50.7
Game	46.9	37.6	80.1
Movie	45.6	36.8	80.7
Average	41.1	**26.9**	65.7

Table 2: Display quality: SSIM distribution.

	offline[3]	proposed online	
Quality (SSIM)	High (0.98-1)	High (0.98-1)	Medium (0.96-0.98)
News	100%	100%	-
Cartoon	100%	100%	-
Game	100%	97.3%	2.7%
Movie	100%	100%	-
Average	100%	**99.3**%	0.7%

The conclusions are:

- Compared with displays without DVS, the proposed online DVS attains 26.9% power saving, which is 65.7% of the saving from the offline solution (Table 1).
- The offline method keeps all frames in high quality (SSIM≥0.98), while the proposed online DVS delivers 99.3% high quality frames during displays (Table 2).
- Based on the instruction throughput and power model of mainstream mobile GPUs [1], the computation overhead is analyzed to be 1.37% of GPU's computation capacity. The proposed online DVS technique consumes only 0.05% of the screen power we have saved.

4. REFERENCES

[1] GPU Benchmarking in Mobile World, Alan Tsai, ARM, Nov, 2011.

[2] X. Chen, J. Zeng, Y. Chen, W. Zhang, and H. Li. Fine-grained dynamic voltage scaling on oled display. In *ASP-DAC*, pages 807 –812, Feb. 2012.

[3] X. Chen, M. Zhao, J. Zeng, J. Xue, and Y. Chen. Quality - retaining OLED dynamic voltage scaling for video streaming applications on mobile devices. In *DAC*, pages 1000–1005, 2012.

[4] D. Shin, Y. Kim, N. Chang, and M. Pedram. Dynamic voltage scaling of OLED displays. In *DAC*, pages 53–58, 2011.

An Asymptotically Optimal Real-Time Locking Protocol for Clustered Scheduling under Suspension-Aware Analysis

Björn B. Brandenburg
Max Planck Institute for Software Systems (MPI-SWS)
bbb@mpi-sws.org

1. OPTIMAL LOCKING PROTOCOLS

The purpose of real-time locking protocols is to limit *priority inversions* [5], which, intuitively, occur when a high-priority task is delayed by a lower-priority task. Such locking-related delay, also called *priority inversion blocking* (*pi-blocking*), is problematic in real-time systems because it can result in deadline misses. However, some pi-blocking is unavoidable when using locks and thus must be bounded and accounted for during schedulability analysis.

Clearly, an "optimal" locking protocol should minimize pi-blocking to the extent possible. Formally, a locking protocol is asymptotically optimal if it ensures that, for *any* task set, maximum pi-blocking is bounded within a constant factor of the minimal pi-blocking unavoidable in *some* task set [3]. Interestingly, there exist two classes of schedulability analysis that yield *different* lower bounds: under *suspension-oblivious* (*s-oblivious*) analysis, $\Omega(m)$ pi-blocking is fundamental, whereas under *suspension-aware* (*s-aware*) analysis, $\Omega(n)$ pi-blocking is unavoidable in the general case [2, 3], where m and n denote the number of processors and tasks, respectively. As the names imply, the key difference is that suspensions are accounted for explicitly under s-aware analysis, whereas they are (pessimistically) modeled as execution in the s-oblivious case.

For the simpler s-oblivious case, asymptotically optimal locking protocols have been designed for partitioned, global, and clustered *job-level fixed-priority*[1] (JLFP) scheduling [4]. The s-aware case, however, is much less understood: only two asymptotically optimal protocols for partitioned JLFP scheduling are known so far [2, 3].

In contrast, the problem of optimal s-aware locking under global and clustered JLFP scheduling has remained open to date. While it was initially assumed [3] that Block *et al.*'s *Flexible Multiprocessor Locking Protocol* (FMLP) [1]—which is based on $O(n)$ FIFO queues—is asymptotically optimal under global scheduling, it was later observed [2] that this holds only under some, but not all global JLFP schedulers. In fact, it was shown that both *priority inheritance* [5] and (unconditional) *priority boosting* [5], one of which is used in each previously proposed s-aware protocol to expedite the completion of critical sections by temporarily raising the effective priority of lock-holding jobs, can give rise to non-optimal $\Omega(\Phi)$ pi-blocking [2], where Φ is the ratio of the longest and the shortest period (and unbounded in general). Finally, to the best of our knowledge, no asymptotically optimal s-aware locking protocol for the general case of clustered JLFP scheduling has been proposed in prior work.

[1] The class of job-level fixed-priority schedulers includes both classic fixed-priority and EDF scheduling. Clustered scheduling is a generalization of both partitioned and global scheduling under which disjoint clusters of processors are scheduled globally.

2. THE GENERALIZED FMLP$^+$

We have solved the problem of asymptotically optimal s-aware locking under clustered JLFP scheduling by devising a new progress mechanism that circumvents the $\Omega(\Phi)$ bound mentioned above.

Priority boosting/inheritance is susceptible to $\Omega(\Phi)$ pi-blocking because a high-priority job J_h can be repeatedly preempted by critical sections that were started *after* J_h was already scheduled [2]. This is avoided by the following *restricted boosting* mechanism. Let $t_r(J_i)$ denote the latest point in time that a job J_i either **(i)** was *released*, **(ii)** *resumed* from a locking-unrelated self-suspension, or **(iii)** *requested* (*i.e.*, tried to lock) a resource. A priority-boosted, lower-priority job J_l may preempt a higher-priority, un-boosted job J_h only if $t_r(J_l) < t_r(J_h)$. This implies that J_h is preempted only by critical sections that were in progress when J_h became available for scheduling, of which there are at most $n - 1 = O(n)$ (*i.e.*, one per task, assuming tasks are sequential). Further, it can be shown that lock-holder progress is guaranteed in the sense that at least one lock-holder is always scheduled (if any exist). By scheduling priority-boosted jobs in order of increasing t_r timestamps (*i.e.*, FIFO w.r.t. lock request time), $O(n)$ pi-blocking per request is achieved.

Restricted boosting generalizes the idea underlying the partitioned *FIFO Multiprocessor Locking Protocol* (FMLP$^+$) [2], namely to order lock-holding jobs by request time. Combined with $O(n)$ FIFO queues, we obtain a locking protocol that is asymptotically optimal under clustered (and hence also under global) JLFP scheduling.

3. OUTLOOK

We believe the proposed protocol offers improved schedulability, in particular if Φ is large, and are in the process of deriving fine-grained (*i.e.*, non-asymptotic) pi-blocking bounds. We plan to implement and evaluate the protocol in LITMUS$^{\mathrm{RT}}$ and expect overheads to be relatively low due to the simplicity of FIFO queuing and timestamp-based preemption checks. Further, we will explore the potential of an analogously designed "restricted inheritance" mechanism.

References

[1] A. Block, H. Leontyev, B. Brandenburg, and J. Anderson. A flexible real-time locking protocol for multiprocessors. In *Proc. RTCSA*, 2007.
[2] B. Brandenburg. *Scheduling and Locking in Multiprocessor Real-Time Operating Systems*. PhD thesis, The University of North Carolina at Chapel Hill, 2011.
[3] B. Brandenburg and J. Anderson. Optimality results for multiprocessor real-time locking. In *Proc. RTSS*, 2010.
[4] B. Brandenburg and J. Anderson. The OMLP family of optimal multiprocessor real-time locking protocols. *Design Automation for Embedded Systems*, to appear, 2012.
[5] R. Rajkumar. *Synchronization In Real-Time Systems—A Priority Inheritance Approach*. Kluwer Academic Publishers, 1991.

The Fork-Join Real-Time Task Model

Martin Stigge, Pontus Ekberg and Wang Yi
Uppsala University, Sweden
Email: {martin.stigge | pontus.ekberg | yi}@it.uu.se

***Abstract*—Hard real-time task models have evolved from periodic models to more sophisticated graph-based ones like the Digraph Real-Time Task Model (DRT) [1]. These models have in common that tasks are *sequential* in nature and do not allow for *forking* structures, modeling job releases that occur in parallel within the same task. To capture these, we present a task model that extends the DRT model with the possibility of forking and joining release paths. We are developing an exact schedulability test for EDF on uniprocessor systems with a pseudo-polynomial bound of its runtime.**

I. Task Model and Problem Statement

A fork-join real-time (FJRT) task system $\tau = \{T_1, \ldots, T_N\}$ consists of N independent tasks. A task T is represented by a directed hypergraph[1] $G(T)$ with both vertex and edge labels. The vertices v_i represent the types of all the jobs that T can release and are labeled with ordered pairs $\langle e(v_i), d(v_i)\rangle$ of non-negative integers, denoting worst-case execution-time demand $e(v_i)$ and relative deadline $d(v_i)$ of the corresponding job. The hyperedges of $G(T)$ represent the order in which jobs generated by T are released. A hyperedge (U, V) is either a *sequence edge* with U and V being singleton sets of vertices, or a *fork edge* with U being a singleton set, or a *join edge* with V being a singleton set. In all cases, the edges are labeled with a non-negative integer $p(U, V)$ denoting the minimum job inter-release separation time. Note that this contains the DRT model as a special case if all hyperedges are sequence edges.

As an extension of the DRT model, an FJRT task system releases independent jobs, allowing to define concepts like *utilization* $U(\tau)$ and *demand bound function* just as before (in e.g. [1]).

Semantics: A task executes by following a path through the hypergraph, triggering releases of associated jobs each time a vertex is visited. Whenever a fork edge $(\{u\}, \{v_1, \ldots, v_m\})$ is taken, m independent paths starting in v_1 to v_m, respectively, will be followed in parallel until joined by a corresponding join edge. In order for a join edge $(\{u_1, \ldots, u_n\}, \{v\})$ to be taken, all jobs associated with vertices $u_1, \ldots, u_n$ must have been released and enough time must have passed to satisfy the join edge label. Forking can be nested, i.e., these m paths can lead to further fork edges before being joined. Note that meaningful models have to satisfy structural restrictions which we skip for space reasons, e.g., each fork needs to be joined by a matching join, and control is not allowed to "jump" between parallel sections. Consider the example in Figure 1.

The main objective of this work is to develop an efficient method for analyzing EDF schedulability for the above model. Thus, we seek to provide a proof to the following conjecture.

Conjecture I.1. *For an FJRT task system τ with $U(\tau) \leqslant c$ for some constant $c < 1$, feasibility can be decided in pseudo-polynomial time.*

Similar results have been shown for simpler task models like the DRT model [1].

[1] A hypergraph generalizes the notion of a graph by extending the concept of an edge (u, v) between two vertices u and v to hyperedges (U, V) between two sets U and V of vertices.

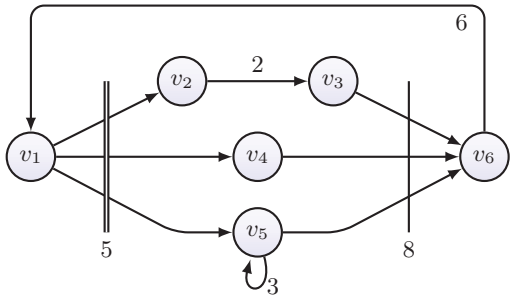


Fig. 1. Example FJRT task. The fork edge is depicted with an intersecting double line, the join edge with an intersecting single line. All edges are annotated with minimum inter-release delays $p(U, V)$. The vertex labels are omitted in this example. Assuming that vertex v_i releases a job of type J_i, a possible job sequence containing jobs with their types and absolute release times is $\sigma = [(J_1, 0), (J_2, 5), (J_5, 5), (J_4, 6), (J_3, 7), (J_5, 8), (J_6, 16), (J_1, 22)]$.

II. Solution Approach

Demand Bound Function: A demand bound function $dbf_T(t)$ for an FJRT task T can be defined as usual as the maximum cumulative execution time requirement of jobs with both release times and deadlines within any interval of length t. Using this, a precise schedulability test for EDF can be based on the following proposition (and EDF's optimality).

Proposition II.1. *An FJRT task system τ is preemptive uniprocessor feasible if and only if*

$$\forall t \geqslant 0 : \sum_{T \in \tau} dbf_T(t) \leqslant t. \tag{1}$$

Demand Tuples: For an FJRT task T without fork and join edges, $dbf_T(t)$ can be evaluated by traversing its graph $G(T)$ using a path abstraction called demand tuples [1]. We can extend this method to the new hyperedges by a recursive approach. Starting with "innermost" fork/join parts of the hypergraph, the tuples are merged at the hyperedges and then used as path abstractions like before. It can be shown that this method is efficient.

Utilization: Just computing $dbf_T(t)$ does not suffice for evaluating Condition (1) since it is also necessary to know which interval sizes t need to be checked. As for the DRT model [1], a bound can be derived from the task set's utilization $U(\tau)$. It turns out that an efficient way of computing $U(\tau)$ is surprisingly difficult to find and is still work in progress. The difficulty comes from parallel sections in the task with loops of different periods which, when joined, exhibit the worst case behavior in very long time intervals of not necessarily polynomial length.

References

[1] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The Digraph Real-Time Task Model," in *Proc. of RTAS 2011*, pp. 71–80.

Fixed Priorities or EDF for Distributed Real-Time Systems?

Juan M. Rivas
Computers and Real-Time Group
Universidad de Cantabria
39005-Santander, SPAIN
rivasjm@unican.es

J. Javier Gutiérrez
Computers and Real-Time Group
Universidad de Cantabria
39005-Santander, SPAIN
gutierjj@unican.es

Michael González Harbour
Computers and Real-Time Group
Universidad de Cantabria
39005-Santander, SPAIN
mgh@unican.es

EXTENDED ABSTRACT[1]

Although fixed priority (FP) scheduling is the most popular on-line scheduling policy in industrial environments, the earliest deadline first (EDF) policy is starting to get more attention, given its benefits in terms of increased resource usage. A comparison between FP and EDF under several aspects is presented in [1] for single-processor systems, showing that FP does not always have the commonly attributed advantages over EDF.

The starting point of this work is MAST[2], a suite of tools developed in our group that offers different schedulability analysis and scheduling parameter assignment techniques for event-driven systems. MAST is based on the end-to-end flow model proposed in the MARTE standard [2] and integrates techniques for FP, EDF, or heterogeneous FP/EDF [3] (e.g., EDF processors and a CAN bus), which can be applied to distributed systems. For EDF it supports both local schedulers, where task scheduling deadlines are referenced to their release times in their own processor, and global schedulers, where scheduling deadlines are referenced to the arrival of the event that releases the end-to-end flow, possibly in a different processor, and thus requiring clock synchronization. MAST is under active development [4], and new techniques can be easily integrated.

Our goal is to compare how FP and EDF can influence the schedulability of a distributed real-time system under a variety of conditions: different system sizes, deadline/period ratios, different lengths of end-to-end flows, etc. To this end, we propose an exhaustive study of current schedulability analysis and scheduling parameter assignment techniques over a wide range of examples, with the purpose of evaluating each algorithm and quantify the differences among them, as well as comparing the ability of the different scheduling algorithms to meet the deadlines.

We need the capability to execute the analysis and assignment techniques over an extensive set of examples. Thus, we are developing a benchmark generator tool for MAST, GEN4MAST, which can (1) generate a pool of MAST examples according to some specifications and evolution rules, (2) run a set of tests with different schedulability analysis or scheduling parameter assignment techniques, and finally (3) store and process the results. The first version of the tool deals only with tasks and processors, but it will be extended to messages and networks, and also to the network switches being included in future versions of MAST.

For the generation of examples, the user can specify the basic characteristics of the systems that will form the pool of examples: the number of end-to-end flows, tasks and processors; the deadline and period ranges and ratios; the type of schedulers (FP and local or global EDF); and the system loads. This allows adapting to the characteristics of a specific application domain. Each example is generated with a certain component of randomness (task allocation or actual deadline or period values), and it is also possible to specify how many examples with a similar architecture can be generated, so statistically relevant results can be obtained. In addition, it is possible to specify other parameters that fine-tune the behavior of the schedulability analysis or the scheduling parameter assignment algorithms (e.g., configuration parameters for the HOSPA algorithm).

For each execution of a test (example-algorithm combination), a set of results is recorded, including the worst-case response times obtained, the execution times required for the calculation and, optionally, the sensitivities given as percentages of system slacks.

Preliminary results using holistic analysis techniques show that global EDF reaches higher average utilizations than FP, which in turn gets higher utilizations than local EDF. The table below shows the maximum average schedulable utilization achieved by each technique for a medium to small size experiment (17 tasks and 3 processors). This experiment took 6 hours of computation to analyze 4200 examples.

	FP	*Local EDF*	*Global EDF*
Utilization %	82,6	67,0	95,7

REFERENCES

[1] G. Buttazzo, "Rate Monotonic vs. EDF: Judgment Day," *Real-Time Systems*, 29(1), pp. 5-26, 2005.

[2] Object Management Group, "UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems," 2009 OMG Document, v1.0 formal/2009-11-02.

[3] Juan M. Rivas, J.Javier Gutiérrez, J. Carlos Palencia, M. Gonzalez Harbour, "Schedulability Analysis and Optimization of Heterogeneous EDF and FP Distributed Real-Time Systems," Proceedings of the 23th Euromicro Conference on Real-Time Systems, Porto (Portugal), 2011.

[4] M. González Harbour, J. Javier Gutiérrez, José M. Drake, P. López Martínez and J. C. Palencia, "Modeling distributed real-time systems with MAST 2," In press, Journal of Systems Architecture, Elsevier, 2012.

[1] This work has been funded in part by the Spanish Government under grant number TIN2011-28567-C03-02 (HIPARTES).

[2] MAST is available at http://mast.unican.es/

Application of Mixed-Criticality Scheduling Model to Intelligent Transportation Systems Architectures

[Extended Abstract]

Vincent Sciandra
VERI
10 rue Jacques Daguerre
Rueil-Malmaison, France
vincent.sciandra@veolia.com

Pierre Courbin
ECE Paris
37 quai de Grenelle
Paris, France
courbin@ece.fr

Laurent George
LISSI
120 rue Paul Armangot
Vitry sur Seine, France
lgeorge@ieee.org

Keywords

Mixed-Criticality, ITS, V2I, Non-preemptive

1. INTRODUCTION

Intelligent Transportation Systems (ITS) usage has transformed public transports' vision on operation management. However, the lack of common communication interfaces has brought redundancy within on-board applications. In this context, the European Bus System of the Future (EBSF) project has specified a common architecture based on IP standards [1]. It provides a Service Oriented Architecture (SOA) that permits the ITS applications to publish services and subscribe to others. The EBSF architecture enables Vehicle To Infrastructure communications (V2I) through a unique communication gateway, with managing needs of the different data flows depending on their priority in the network and the available communication resources (bandwidth).

In this public transport context, we are searching to model the priority management in this Mixed-Criticality environment. Either for internal data exchanges in the SOA architecture or for shared resources such as the gateway, security recording services, etc. The increasing number of embedded applications in public transport has considerably brought new challenges when it comes to mixed-criticality management. In our actual research, we consider applying Mixed-Criticality Scheduling model presented in [2] for distributed embedded applications in public transport, based on the EBSF architecture.

2. THE "ITS" MODEL

The constraints of ITS applications in public transport are very close to the studied scope in real-time scheduling problems. Indeed, if we focus on the shared communication gateway, we see that we have multiple applications with different criticality levels, which need to send data to off-board back-offices. The data flows contain messages that are sent sporadically, with a constraint deadline (defined as D) such as, there never are two messages of the same type that are ready to be sent at the same instant t. Moreover, the studied model is considered as a non-preemptive scheduling problem, in the sense that when a message is sent, it cannot be preempted even in the case if another message with a higher priority needs to be executed.

In that context, we are currently extending the work done in [3] in order to adapt the solution to non-preemptive problems such as network applications. The network is considered as a uniprocessor with a dedicated speed of execution, this speed is, in our case, considered as the throughput. We consider $C_i(\ell)$ the transmission time of a message i at a criticality level-ℓ. This level-ℓ is defined by the throughput size (or processor speed). Using this model, an extension of the worst case response time ($WCRT$) and the *Latest Completion Time (LCT)* is being adapted for non-preemptive systems.

In terms of challenges, this model needs to take into consideration a "relaxed" mixed-criticality model in order to ensure optimal transmissions to the back-offices.

3. EXPERIMENTAL WORK

In order to test the model, we use real life data of bus fleets operated under the authority of Vasttrafik in Sweden, where the throughput measurement have been logged along the bus journey during several weeks. As the vehicle is moving, the throughput evolves, however the deadline constraint of the messages stay identical. In that context it's of an interest to measure the $WCRT$ of the different messages depending on the level-ℓ of criticality. Depending on ℓ, the lowest the throughput will be, the longest $C_i(\ell)$ will take. In this way we see the number of dropped messages and the limits of the architecture. Applying the Mixed-Criticality real-time scheduling approach to public transport architectures brings clearly a new dynamic to both research domains. It enhances also the building of new certification models for such architecture.

4. REFERENCES

[1] EBSF partners WP2.3. D231 overall description of bus on-board and back-office ip network architecture. Technical report, EBSF, December 2009.

[2] Steve Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proceedings of the 28th IEEE RTSS*, pages 239–243. IEEE Computer Society, 2007.

[3] F. Santy, L. George, P. Thierry, and J. Goossens. Relaxing mixed-criticality scheduling strictness for task sets scheduled with fp. In *ECRTS, 2012 24th Euromicro Conference on*, pages 155 –165, july 2012.

High-Confidence Cyber-Physical Co-Design

David Broman

University of California, Berkeley, USA and Linköping University, Sweden
broman@eecs.berkeley.edu

1. INTRODUCTION

Cyber-physical systems (CPS) [4] are characterized by combining computations, networks, and physical processes. Engineering cyber-physical systems is not new; high-end automobiles have for decades included complex embedded systems that interact with the physical environment. As an intellectual discipline, however, CPS design poses both new opportunities and challenges. The rapid development of a CPS with high-confidence of its functional correctness is a *co-design problem*—the design of the cyber part (embedded control systems and networks) and the physical part influence each other. For instance, when designing an industrial robot, the thickness of the robot arms changes the physical behavior of the system; thinner arms have less inertia and can move faster, but introduce more flexibility and spring behavior, making the control algorithm harder to design. As a consequence, to meet increasingly challenging system-level objectives, the cyber and physical parts need to be concurrently designed.

2. CO-DESIGN CHALLENGES

Today, model-based design is a well established approach for the concurrent design of different parts of cyber-physical systems. Complex systems can be virtually modeled using languages and tools such as Modelica [6], Ptolemy II [3], and Matlab/Simulink. To achieve *high-confidence* co-design, however, there are two major challenges.

Firstly, the modeling languages and tools must be *expressive* enough to capture the dynamic semantics of the modeled system. Expressiveness is easy to achieve if no consideration is taken to the possibility of *analyzing* the model; an ANSI C program can model the dynamics of a very complex system, but is formally hard to analyze. Simulation is one form of analysis, where properties are checked by testing, but full coverage is often not possible. Formal verification, such as model checking [2], can prove properties of the model, but requires a less expressive model of computation (MoC). The challenge is to provide both expressiveness and analyzability.

Secondly, a necessary condition for high-confidence of CPSs is high *model fidelity*, meaning that the model accurately imitates the real system. The problem is to automatically synthesize the model's cyber parts, such that the simulated model and the behavior of the real running system coincide. Synthesizing the functional behavior can be done today; the main challenge is to guarantee the preservation of the *timing behavior*.

3. OUR APPROACH

We propose an integrated language and compiler based approach to address these challenges. Our work is based on an extensible host language called *Modelyze* [1] (MODEl and anaLYZE), where various MoCs may be embedded as domain-specific languages (DSLs). The key aspect of our approach is that both the definition of the DSL and the CPS models are defined within the same language—Modelyze. We are currently evaluating this embedded DSL approach for encoding different MoCs as well as how formal verification can be encoded in the same framework. As a consequence, we address the combined expressiveness-analyzability problem by enabling *extensibility* within the language; if certain semantics cannot be described within the framework, a user may embed a new DSL with the desired properties.

The second part of our approach concerns high-confidence model synthesis. In another project within our research group, we are developing a Precision Timed (PRET) infrastructure, including an intermediate language and an ARM-based PRET machine [5] with thread interleaved pipeline and scratchpad memories for predictability. Our objective is, within Modelyze libraries, to define both functional and timing semantics as a translation from a Modelyze DSL to a PRET intermediate language. We intend to evaluate our approach in the mechatronics domain, where both the model of the physical plant and the control system are defined in Modelyze.

Project funding: Swedish Research Council #623-2011-955.

4. REFERENCES

[1] D. Broman and J. G. Siek. Modelyze: a gradually typed host language for embedding equation-based modeling languages. Technical Report UCB/EECS-2012-173, EECS Department, University of California, Berkeley, June 2012.

[2] E. M. Clarke. Model checking. In *Foundations of software technology and theoretical computer science*, pages 54–56. Springer, 1997.

[3] J. Eker, J. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Sachs, and Y. Xiong. Taming heterogeneity - the Ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, 2003.

[4] E. A. Lee. Cyber Physical Systems: Design Challenges. In *Proc. of the 11th Symposium on Object Oriented Real-Time Distributed Computing*, pages 363–369. IEEE, 2008.

[5] I. Liu, J. Reineke, D. Broman, M. Zimmer, and E. A. Lee. A PRET Microarchitecture Implementation with Repeatable Timing and Competitive Performance. In *Proc. of the 30th IEEE International Conference on Computer Design*. IEEE, 2012.

[6] Modelica Association. *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling - Language Specification Version 3.3*, 2012.

Performance analysis of TDMA-based Wireless Network for Safety-critical Avionics

D-K. DANG [1], A. MIFDAOUI[1], T. GAYRAUD[2]
[1]University of Toulouse-ISAE, France
[2]University of Toulouse-UPS-LAAS, France

The opportunities and challenges for using wireless interconnects for safety-critical avionics have been discussed in our previous work[1]. A Wireless Avionics Network (WAN) has been proposed based on hybrid architecture UWB and Switched Ethernet with adequate reliability and security mechanisms to increase scalability and reduce electromagnetic susceptibility (figure 1). Furthermore, a TDMA-based protocol was considered to guarantee a contention free access and enhance communication predictability. However, the use of wireless technologies may increase the communication latencies due to transmission errors, and real time constraints have to be verified. In order to deal with the worst case performance analysis of such network, an appropriate schedulability analysis based on Network Calculus formalism is presented in this paper and obtained results for a realistic case study are discussed herein.

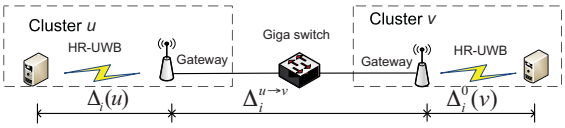


Figure 1: Proposed Wireless Avionics Network

In order to integrate the different characteristics of the traffic generated by avionics applications, three parameters (T_i, D_i, L_i) are defined for each traffic class i to denote respectively the period, the deadline and the message length that integrates the different protocol overheads. Let's consider $n_u^{(i,k)}$ the number of messages in traffic class i generated by the node k in cluster u and $p_u^{(i,k)}$ the associated packet error rate. Hence, the expected number of transmitted messages to deliver correctly $n_u^{(i,k)}$ messages is $\overline{n_{Cl_u}^{(i,k)}} = \lceil \frac{n_u^{(i,k)}}{1-p_u^{(i,k)}} \rceil$. Then, the transmitted traffic class i by node k is modeled with the affine arrival curve:

$$\alpha_u^{(i,k)}(t) = \overline{n_u^{(i,k)}}.(L_i + \frac{L_i}{T_i}) \quad (1)$$

Each considered node k in cluster u schedules its generated messages according to Weighted Fair Queuing (WFQ) policy where each traffic class i admits an associated weight $w_u^{i,k}$ where $\sum_i w_u^{i,k} = 1$. Furthermore, each node k in cluster u can transmit its messages only during its associated time slot s_u^k where $\sum_k s_u^k = c_u$ and c_u is the TDMA cycle.

[1]D. K. Dang, A. Mifdaoui, and T. Gayraud. "Fly-By-Wireless for next generation aircraft: Challenges and potential solutions". In IEEE Wireless Days conference, 2012

Hence, the service curve offered by the cluster u to each traffic class i transmitted by a node k is modeled as follows:

$$\beta_u^{(i,k)}(t) = B \max(\lfloor \frac{t}{c_u} \rfloor \phi_u^{i,k}, t - \lceil \frac{t}{c_u} \rceil (c_u - \phi_u^{i,k})) \quad (2)$$

where B is the transmission capacity of the network and $\phi_u^{i,k} = s_u^k * w_u^{i,k}$ is the residual time slot of node k to transmit traffic class i.

The considered Ethernet switch admits a Store and Forward mode and FCFS scheduling strategy. This can be modeled with the service curve $\beta_{sw}(t) = \max(0, C[t - \frac{L_v}{C})$ where L_v is the maximum size of Ethernet frame encapsulating the UWB messages sent by the different gateways.

In order to verify the schedulability of our proposal, the upper bounds on end to end delays are calculated and compared to respective deadlines. This delay consists of three parts as shown in figure 1: $\Delta_i^k(u)$ and $\Delta_i^0(v)$ that correspond to the worst case intra-cluster communication delays within clusters u and v, associated to traffic class i sent from node k. These delays correspond to the horizontal distance between the defined arrival and offered service curves; $\Delta_i^{u->v}$ that corresponds to the worst case inter-cluster communication delay from cluster u to cluster v associated to traffic class i due to the switch. This delay represents the horizontal distance between the output arrival curves coming from the gateways and the switch service curve.

Table 1: Maximal End to End delays bounds

Traffic classes	$PER = 0\%$	$PER = 1\%$	$PER = 2\%$
TC1 (P=2ms)	9,215ms	9,227ms	9,228ms
TC2 (P=32ms)	41,650ms	44,292ms	44,333ms

The maximal end to end delays with different values of packet error rates obtained for a representative A380's avionics communication network are described in table 1. This WAN consists of 56 endsystems separated in three clusters with almost 200 inter-cluster flows and 400 intra-cluster flows having periods of 2ms or 32ms. We consider as a first step a fair slots allocation for the different nodes based on their generated traffic rates to implement the TDMA protocol. As it can be noticed, the deadlines of the two traffic classes are not respected with this considered configuration. These first results show that the choice of the TDMA-cycle duration and the slots allocation scheme are of utmost importance to fulfill the temporal constraints. This optimization problem will be considered in our future work to find an adequate slots allocation scheme for each cluster to minimize the end to end delays and respect the deadline constraints.

Optimizing QoS in Energy-Aware Real-Time Systems

Ríad Nassiffe
Department of Automation and Systems Engineering
Federal Univ. of Santa Catarina
riad@das.ufsc.br

Eduardo Camponogara
Department of Automation and Systems Engineering
Federal Univ. of Santa Catarina
camponog@das.ufsc.br

George Lima
Computer Science Department
Federal Univ. of Bahia
gmlima@ufba.br

We consider the problem of optimizing application quality of service subject to energy and real-time constraints. The system is composed of n sporadic and independent real-time tasks to be scheduled on a single CPU capable of operating at different frequency/voltage levels, which can be selected at run-time. Each task i may run at a selected frequency f_i and mode of operation k_i with each mode inducing a quality of service level. Due to schedulability and energy bounds, CPU frequencies and task modes should be selected by the system aiming at maximizing the overall system quality of service, an optimization problem formulated as:

$$P: \quad \max \quad \sum_{i=1}^{n} \mathrm{QoS}(f_i, k_i) \tag{1a}$$

$$\text{s.t.}: \sum_{i=1}^{n} u(f_i, k_i) \leq \mathrm{ub} \tag{1b}$$

$$\sum_{i=1}^{n} e(f_i, k_i) \leq \mathrm{eb} \tag{1c}$$

$$f_i \in \mathcal{F}, k_i \in \mathcal{K}_i, \ i = 1, 2, \ldots, n \tag{1d}$$

where QoS represents the considered objective function, and constraints (1b) and (1c) are related to bounds on CPU (ub) and energy (eb), respectively. The set $\mathcal{F}$ may contain either the CPU frequency values allowed or represents a continuous frequency interval, a usual frequency model assumed in literature [1]. The set $\mathcal{K}_i$ represents a discrete set of modes that task i may operate. Different classes of optimization problems, with distinct degrees of difficulty, can be obtained depending on how $\mathcal{F}$ and $\mathcal{K}_i$ are modeled. In any case, the objective function QoS must capture some aspects: (I) *mode degradation* with priority given to the high quality operational modes; (II) *task value* which is associated with every task being strongly related to the application semantics; (III) *task weight* specifying the processing resources required by each task, thus a task that requires 10% of CPU should be treated differently from one that requires only 1%, possibly by receiving a higher task weight. Function (2) takes these aspects into account:

$$\mathrm{QoS}(f_i, k_i) = \frac{1}{|\mathcal{K}_i|} \left(k_i - 1 + w_i u_i(f_i, k_i)\right) \tag{2}$$

where $\mathcal{K}_i = \{1, 2, \ldots\}$ has the indices of the task modes and $w_i \in [0, 1]$ is the task value. Mode k_i has quality higher than mode k_i' whenever $k_i > k_i'$. Notice that QoS will be a value between 0 and 1 since the CPU utilization $u_i(f_i, k_i)$ cannot exceed the unit.

The reduction of CPU frequency results in an increase of task execution time which can further delay the completion of the tasks, specially those that need more resources than predicted, a common scenario for soft real-time systems. For this reason the task values should be adjusted depending on the CPU frequency, with higher values for tasks running at high frequency. Assuming that $w_i = f_i^2$, $\mathcal{F} = \{100\%, 50\%, 25\%\}$ and a task with 3 modes, where the utilization at $f_i = 100\%$ are respectively $0.042, 0.34, 0.60$, the QoS function (2) ensures that a configuration with high mode and frequency will always have greater value/system benefit, thereby avoiding unnecessary degradation of the system as shown in Table 1.

Table 1: Task configuration benefits

	$QoS(f_i, k_i)$		
	$k_i = 1$	$k_i = 2$	$k_i = 3$
$QoS(100\%, k_i)$	0.014	0.448	0.866
$QoS(50\%, k_i)$	0.007	0.391	0.766
$QoS(25\%, k_i)$	0.003	0.362	0.716

Scheduling problems can be modeled as continuous or discrete as shown in [3]. Continuous and convex models can be efficiently solved by interior-point methods [2]. Discrete models can be solved with relaxation-based heuristics [3, 4] or with specialized algorithms. Applying such techniques to problem (1a)-(1d) has led to promising preliminary results.

1. REFERENCES

[1] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Power-Aware Scheduling for Periodic Real-Time Tasks. *IEEE Transactions on Computers*, 53(5):584–600, 2004.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] E. Camponogara, A. B. de Oliveira, and G. Lima. Optimization-Based Dynamic Reconfiguration of Real-Time Schedulers with Support for Stochastic Processor Consumption. *IEEE Transactions on Industrial Informatics*, 6(4):594–609, Nov 2010.

[4] R. Nassiffe, E. Camponogara, and G. Lima. Optimizing Quality of Service in Real-Time Systems Under Energy Constraints. *ACM OSR*, 46(1):82–92, 2012.

Reliability-Aware Energy Minimization for Real-Time Embedded Systems with Window-Constraints

Linwei Niu and Luis Medina
Department of Computer and Electrical Engineering and Computer Science
California State University Bakersfield
Bakersfield, CA
{lniu, lmedina}@csub.edu

Yiran Chen
Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA
yic52@pitt.edu

ABSTRACT

In this work, we propose a reliability-aware energy minimization scheme that can satisfy the *window-constraints*, *i.e.*, no more than x_i deadlines are missed in each nonoverlapped sequence of y_i jobs in real-time task τ_i. The simulation reveals that our proposed techniques can outperform previous ones in energy reduction for real-time systems with reliability awareness under *window-constraints*.

1. INTRODUCTION

With system reliability in mind, plenty of works have been published in reducing the energy consumption for *hard* real-time systems. However, few real-time applications are truly hard. For example, in network packet streams, some deadline misses are allowed provided that user's perceived quality of service (QoS) constraints are satisfied. The *window-constrained* model [2] is a very suitable QoS model for such kind of applications [2, 1]. In [1] a technique was proposed that could ensure the *window-constraints* based on *evenly distributed pattern* such as in Figure 1(a). On the other hand, to preserve the system reliability, at the end of each mandatory job, a portion of available slack could be reserved for scheduling a recovery job for it [3]. If the mandatory job failed, the recovery job will be invoked and executed [3].

Note that in the above approach, in order to reserve space for the recovery jobs, the system feasibility could be affected. For example, for the task set in Figure 1(a), after we reserved a recovery job for each mandatory job before its deadline (for example, R_{10} for J_{10}), the system became overloaded and deadline missing would become inevitable. Fortunately, under *window-constrained* model, the recover jobs could be scheduled more flexibly. For example, for the task set in Figure 1(a), if we utilize the space belonging to the optional jobs to help schedule the recovery jobs, as shown in Figure 1(b), the task set could be well schedulable. Moreover, considering the large run-time variations in real-time systems, it would be extremely profitable to adjust the patterns as well as scale the job speeds dynamically. The idea is, if some optional jobs with short actual execution times could meet their deadlines with relatively lower speeds, the corresponding future mandatory (and recovery) jobs could be demoted to optional or even be dropped without execution, as shown in Figure 1(c) and (d). It is not hard to see that in Figure 1(d), since some optional jobs (for example, J_{20} and J_{30}) could meet deadlines with much lower speeds (s_{20} and s_{30}), significant energy could be saved compared to the schedule in Figure 1(b).

To formalize the above procedure, in our work, we first derived heuristics to determine the static job/recovery patterns. Then based on it we developed dynamic pattern variation algorithms to accommodate the dynamic nature of real-time systems.

2. EVALUATION AND CONCLUSIONS

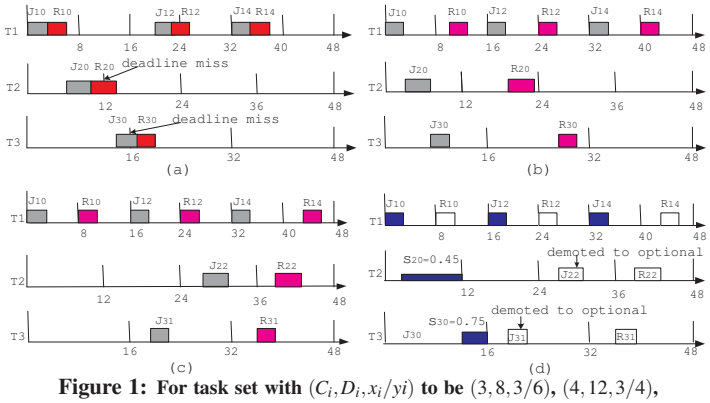


Figure 1: For task set with $(C_i, D_i, x_i/yi)$ to be $(3,8,3/6)$, $(4,12,3/4)$, and $(3,16,2/3)$, respectively: (a) under evenly distributed patterns; (b) using optional jobs to fulfill recovery jobs; (c) varied static job/recovery patterns; (d) dynamic job/recovery patterns with speed scaling.

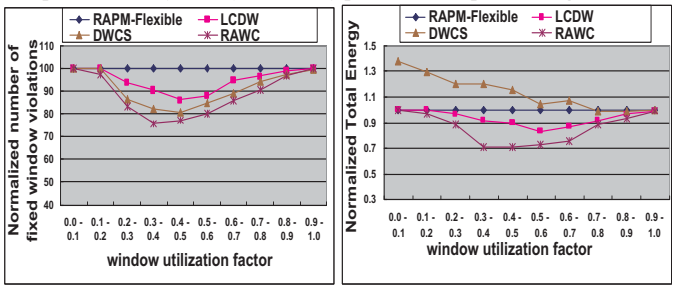


Figure 2: Comparison on: (a) window violations; (b) energy.

Our experiments adopted the same task and energy model as in [1] and fault/recovery models as in [3]. As shown in Figure 2(a), in most utilization intervals, Algorithm **DWCS** [2] has less window violations than **RAPM-Flexible** [3], and **LCDW** [1]. However, its energy consumption is also the highest (Figure 2(b)) because it executed a lot of redundant jobs. In all cases, our algorithm, *i.e.*, **RAWC** has the least number of window violations. Moreover, from Figure 2(b), **RAWC** can achieve significant energy reduction compared with all the other approaches.

And we proved that our dynamic algorithm can satisfy the window-constraints if task sets are schedulable under static recovery pattern.

3. REFERENCES

[1] N. Linwei. Energy efficient scheduling for real-time systems with qos guarantee. *Journal of Real-Time Systems*, 47(2):75–108, 2011.

[2] R. West, Y. Zhang, K. Schwan, and C. Poellabauer. Dynamic window-constrained scheduling of real-time streams in media servers. *IEEE Trans. on Computers*, 53(6):744–759, June 2004.

[3] D. Zhu, X. Qi, and H. Aydin. Energy management for periodic real-time tasks with variable assurance requirements. In *RTCSA*, 2008.

Thermal-Aware Energy Minimization for Real-Time Scheduling on Multi-core Systems

Ming Fan Vivek Chaturvedi Shi Sha Gang Quan
Electrical and Computer Engineering Department
Florida International University
Miami, FL, 33174
{mfan001, vchat001, ssha001, gaquan}@fiu.edu

1. INTRODUCTION

With exponentially increased transistor density on multi-core platforms, the power explosion and consequently soaring chip temperature have become critical challenges for system designers. Moreover, the increasing chip temperature results in higher leakage power, hence further aggravates the increment of the overall power consumption [1]. Thus, the dramatically growing power/energy consumption and chip temperature severely affect the cost, reliability and performance of the systems [4].

In this paper, we study the problem on how to minimize the overall energy consumption for a real-time schedule on multi-core platform with consideration of the interdependence between leakage and temperature. We first develop an analytical solution for energy calculation on multi-core systems, which has no computation of integration on power. Then, based on our energy calculation method, we propose an energy minimization algorithm to schedule real-time tasks on multi-core systems. Validation of our proposed scheme through experiments is a part of our future work.

The system models used in this work are briefly introduced below. We consider a multi-core platform consisting of m identical cores. Each core has n different *running modes*, each of which is characterized by a pair of supply voltage and working frequency. Let $\mathbf{S}$ represent a speed schedule, which indicates how to vary the frequency and voltage on each core at different time instants. We call an interval $[t_{q-1}, t_q]$ as a *state interval* if each core runs only at one running mode within that interval. Let s denote the total number of state intervals in $\mathbf{S}$, and let κ_q denote the running modes of all cores within the q^{th} state interval. The system power consumption at time t, denoted by $\mathbf{P}(t)$, can be formulated as [2]

$$\mathbf{P}(t) = \mathbf{\Psi} + \mathbf{\Phi} \cdot \mathbf{T}(t) \tag{1}$$

where $\mathbf{T}(t)$ represents the system temperature vector at time t, and $\mathbf{\Psi}$ and $\mathbf{\Phi}$ are constant matrices depending on the system running mode. Then the system thermal model can be formulated as [3, 5]

$$\mathbf{C}\frac{d\mathbf{T}(t)}{dt} + \mathbf{G}\big(\mathbf{T}(t) - T_{amb}\big) = \mathbf{\Psi} \tag{2}$$

where $\mathbf{C}$ is the system capacitance matrix, $\mathbf{G}$ is a comprehensive matrix with consideration of leakage power and system conductances, and T_{amb} is the ambient temperature. For a state interval $[t_{q-1}, t_q]$, once $\mathbf{T}(t_{q-1})$ is determined, based on equation (2), $\mathbf{T}(t_{q-1})$ can be solved by

$$\mathbf{T}(t_q) = e^{\mathbf{A}_{\kappa_q}\Delta t_q}\big(\mathbf{T}(t_{q-1}) - T_{amb}\big) + \mathbf{A}_{\kappa_q}^{-1}(e^{\mathbf{A}_{\kappa_q}\Delta t_q} - \mathbf{I})\mathbf{B}_{\kappa_q} + T_{amb} \tag{3}$$

where $\mathbf{A}_{\kappa_q} = -\mathbf{C}^{-1}\mathbf{G}_{\kappa_q}$, $\mathbf{B}_{\kappa_q} = \mathbf{C}^{-1}\mathbf{\Psi}_{\kappa_q}$, and $\Delta t_q = t_q - t_{q-1}$. Note that, bold text and normal text are respectively used for a vector/matrix and a value. Thus, given a schedule $\mathbf{S}$ and its initial temperature $\mathbf{T}(0)$, we can sequentially calculate the temperature at the end of each state interval.

2. OUR PROPOSED WORK

THEOREM 1. *Given a state interval $[t_{q-1}, t_q]$, let $\boldsymbol{T}(t_{q-1})$ and $\boldsymbol{T}(t_q)$ represent the temperature at time t_{q-1} and t_q, respectively. Then the system energy consumption within $[t_{q-1}, t_q]$ can be formulated by*

$$\boldsymbol{E}(t_{q-1}, t_q) = \Delta t_q \mathbf{\Psi}_{\kappa_q} + \mathbf{\Phi}_{\kappa_q}\boldsymbol{G}_{\kappa_q}^{-1}\Big(\Delta t_q \mathbf{\Psi}_{\kappa_q} - \boldsymbol{C}\big(\boldsymbol{T}(t_q) - \boldsymbol{T}(t_{q-1})\big)\Big) \tag{4}$$

From Theorem 1, we see that once the temperature trace of a schedule is determined, its energy consumption for each state interval can be rapidly calculated. Accordingly, the total system energy consumption for a schedule $\mathbf{S}$, denoted by $E_{total}(\mathbf{S})$, can be represented as

$$E_{total}(\mathbf{S}) = \sum_{q=1}^{s}\sum_{i=1}^{m} E_i(t_{q-1}, t_q) \tag{5}$$

Given a real-time task set and a multi-core system, let $\mathcal{S}$ represent the set of feasible schedules, i.e. $\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_L\}$. Then, based on the energy calculation method, our energy minimization algorithm can be briefly described as: Find a schedule $\mathbf{S}_{k^*}$, $\mathbf{S}_{k^*} \in \mathcal{S}$, such that

$$E_{total}(\mathbf{S}_{k*}) = min\{E_{total}(\mathbf{S}_k)\big|\ k = 1, 2, ..., L\} \tag{6}$$

In the future, we plan to conduct extensive experiments to evaluate the accuracy of our proposed energy formulation technique, and estimate the performance of the proposed energy minimization method.

3. REFERENCES

[1] V. Chaturvedi and G. Quan. Leakage conscious dvs scheduling for peak temperature minimization. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 135 –140, Jan. 2011.

[2] G. Quan and Y. Zhang. Leakage aware feasibility analysis for temperature-constrained hard real-time periodic tasks. In *Real-Time Systems, 2009. ECRTS '09. 21st Euromicro Conference on*, pages 207 –216, Jul. 2009.

[3] S. Sharifi, R. Ayoub, and T. Rosing. Tempomp: Integrated prediction and management of temperature in heterogeneous mpsocs. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 593–598, Mar. 2012.

[4] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, pages 2 – 13, june 2003.

[5] I. Ukhov, M. Bao, P. Eles, and Z. Peng. Steady-state dynamic temperature analysis and reliability optimization for embedded multiprocessor systems. In *Design Automation Conference (DAC)*, Jun. 2012.

RT-WiFi: Real-Time High Speed Communication Protocol for Wireless Control Systems

Yi-Hung Wei, Quan Leng, Song Han, Aloysius K. Mok
University of Texas at Austin
{yhwei, qleng, shan, mok}@cs.utexas.edu

Wenlong Zhang, Masayoshi Tomizuka
University of California, Berkeley
{wlzhang, tomizuka}@berkeley.edu

Tianji Li, David Malone, Douglas Leith
National University of Ireland Maynooth
{tianji.li, David.Malone, doug.leith}@nuim.ie

Due to their enhanced mobility and reduced configuration and maintenance cost, wireless control systems (WCSs) are widely used in process and vibration control systems, on medical devices, unmanned vehicles and robotics. However, most literatures in WCSs focus on monitoring and low speed control, and less effort has been made on high speed WCSs. It is because most existing wireless communication protocols cannot provide real-time and reliable communication links with preferable high speed by taking energy saving into consideration.

In our current joint project [1], between University of Texas at Austin and University of California, Berkeley, we are building a network-based human rehabilitation system to provide wire-free rehabilitation at local site and bio-feedback and remote mobility for tele-rehabilitation. In this system, sensors, controllers, and actuators are distributed in different locations and connected over high speed wireless networks. In order to achieve real-time motion control, high sampling rate and reliable communication links are critical, which brings great challenges to the wireless protocol design. Fig. 1 demonstrates the result of how sampling rate could affect the performance of a human rehabilitation system. In this simulation, the nominal model of the rehabilitation device in [1] was employed as the controlled plant and a PD (Proportional plus Derivative) controller was implemented. The reference was set as a unit square signal with the frequency of $2Hz$. As shown in Fig. 1, higher sampling rate leads to smaller overshoot and shorter settling time. The result also indicates that a motion control system usually prefers sampling rate higher than $1kHz$ to guarantee good tracking performance. However, current commercially available wireless technologies cannot be directly applied to the high speed WCSs. For example, wireless protocols designed for low-power personal area networks including Bluetooh, ZigBee and WirelessHART do not provide sufficient data rate to support sampling rate as high as $1kHz$. On the other hand, although Wi-Fi offers enough data rate, it does not have any timing guarantee on packet delivery and it is not designed to be energy efficient.

To address this problem, in this work we propose Real-Time WiFi (RT-WiFi) which is a real-time high speed wireless communication protocol. At the very bottom, RT-WiFi adopts physical layer of Wi-Fi in order to support high data rate. On top of that, we are hacking MAC (medium access control) layer of Wi-Fi to adopt TDMA (Time Division Multiple Access) for providing real-time data delivery and to

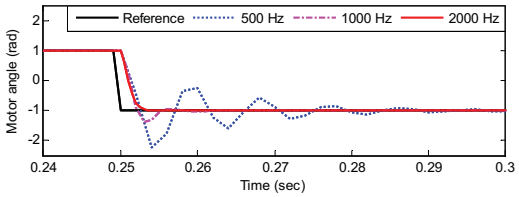


Figure 1: Tracking performance of PD control systems with selected sampling rates (partial amplification)

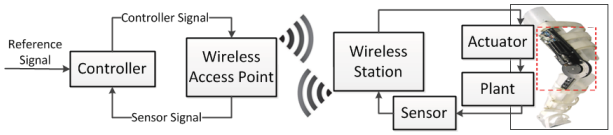


Figure 2: Conceptual block diagram of the human rehabilitation system

explore the channel diversity. Fig. 2 shows the architecture of a wire-free human rehabilitation system at local site. The wireless station and the access point here are Linux boxes, each equipped with Atheros AR9285 Wi-Fi card. The duration of each time slot is set as $500\mu s$, so that we can achieve a data rate of $2kHz$ which is sufficient for a wide range of wireless control applications. In order to provide reliable communication, RT-WiFi utilizes channel hopping and channel blacklists mechanisms to avoid interference, and it supports acknowledgment and retransmission to further improve reliability. Moreover, since the sensors and actuators in WCSs are usually attached at battery-powered mobile devices, RT-WiFi takes an energy-efficient design by turning on its wireless radio only in time slots when transmitting or receiving is scheduled, and it aggressively puts devices in power saving mode to minimize energy consumption.

With the design goal to support reliable, real-time, and high speed communication, we envision RT-WiFi can serve as an ideal platform to high speed WCSs. By adjusting the data rate of RT-WiFi, our wireless platform can support a wide range of WCSs and achieve good balance among sampling rate, reliability, and energy efficiency.

1. REFERENCES

[1] W. Zhang, X. Zhu, S. Han, N. Byl, A. K. Mok, and M. Tomizuka. Design of a network-based mobile gait rehabilitation system. In *IEEE Robotics and Biomimetics (ROBIO)*, 2012.

An Evaluation of the RUN Algorithm in LITMUS^{RT}

[Extended Abstract]

Hiroyuki Chishiro[†‡], James H. Anderson[†], and Nobuyuki Yamasaki[‡]
[†]Department of Computer Science, The University of North Carolina at Chapel Hill, NC, USA
[‡]Department of Information and Computer Science, Keio University, Yokohama, Japan

1. INTRODUCTION

Existing multiprocessor real-time scheduling algorithms follow partitioning/global scheduling approaches or some hybrid approaches of the two. Under partitioning, all tasks are assigned to specific processors. Under global scheduling, tasks may migrate among processors. Global scheduling has the advantage of better schedulability compared to partitioning. However, optimal algorithms based on global scheduling such as PD^2 [5] and LLREF [3] incur significant overhead.

2. RUN ALGORITHM

The Reduction to UNiprocessor (RUN) algorithm [4] is optimal multiprocessor real-time scheduling with a hybrid partitioning/global scheduling approach, called semi-partitioning. Under semi-partitioning, most tasks are assigned to processors and the remaining tasks may migrate among processors. RUN achieves low overhead in simulation studies but the practical viability of it remains unclear.

3. LITMUS^{RT}

LITMUS^{RT} [2] is a real-time extension of the Linux kernel that allows schedulers to be developed as plugin components. Current plugins include PD^2 and partitioned/global fixed-priority/earliest deadline first schedulers.

4. THE RUN PLUGIN

This work introduces an additional plugin to implement RUN in LITMUS^{RT}. RUN assigns tasks to processors offline. The RUN plugin simply utilizes an offline constructed dispatching table to make scheduling decisions at runtime.

The implementation of RUN differs substantially from that of PD^2, the only optimal plugin currently available in LITMUS^{RT}. PD^2 is actually only optimal if the quantum size can be made sufficiently small. In practice, each quantum boundary creates overheads due to the need to handle interrupts, make scheduling decisions, and perform context switches. This limits the practical quantum-size. Quantum-driven scheduling also creates additional release delay since the scheduler does not process scheduling events immediately. On the other hand, RUN is a time-driven scheduler that uses a dispatching table so that such additional release delay does not occur. The other advantage of the RUN plugin reduces cache-related preemption and migration delay. RUN has been shown to reduce the number of preemptions/migrations compared to LLREF with lower overhead than PD^2 in simulation studies [4].

5. PLANNED EVALUATION

We introduce a planned evaluation to verify whether RUN is practical. In this evaluation, we will measure relevant overheads with a variable number of tasks/cores, several ready queue/interrupt handling methods and the schedulability of RUN on both hard/soft real-time systems with respect to [1].

6. CONCLUSIONS AND FUTURE WORK

This work introduces the RUN plugin in LITMUS^{RT}. We discussed the difference between the implementations of RUN and PD^2. We conclude that RUN is more practical than PD^2 with respect to overheads. In future work, we will perform the planned evaluation of RUN.

7. ACKNOWLEDGEMENT

This research was supported in part by CREST, JST. This research was also supported in part by Grant in Aid for the Global Center of Excellence Program for "Center for Education and Research of Symbiotic, Safe and Secure System Design" from the Ministry of Education, Culture, Sport, and Technology in Japan. In addition, this research was also supported in part by Grant in Aid for the JSPS Fellows.

8. REFERENCES

[1] A. Bastoni, B. B. Brandenburg, and J. H. Anderson. An Empirical Comparison of Global, Partitioned, and Clustered Multiprocessor Real-Time Schedulers. In *Proceedings of the 31th IEEE Real-Time Systems Symposium*, pages 14–24, Dec. 2010.

[2] J. M. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. H. Anderson. LITMUS^{RT}: A Testbed for Empirically Comparing Real-Time Multiprocessor Schedulers. In *Proceedings of the 27th IEEE Real-Time Systems Symposium*, pages 111–123, Dec. 2006.

[3] H. Cho, B. Ravindran, and E. D. Jensen. An Optimal Real-Time Scheduling Algorithm for Multiprocessors. In *Proceedings of the 27th IEEE Real-Time Systems Symposium*, pages 101–110, Dec. 2006.

[4] P. Regnier, G. Lima, E. Massa, G. Levinand, and S. Brandt. RUN: Optimal Multiprocessor Real-Time Scheduling via Reduction to Uniprocessor. In *Proceedings of the 32th IEEE Real-Time Systems Symposium*, pages 104–115, Nov. 2011.

[5] A. Srinivasan and J. H. Anderson. Optimal rate-based scheduling on multiprocessors. *Journal of Computer and System Sciences*, 72(6):1094–1117, 2006.

Automated Model Translations for Vehicular Real-Time Embedded Systems with Preserved Semantics

Saad Mubeen*, Mikael Sjödin*, Jukka Mäki-Turja*†, Kurt-Lennart Lundbäck† and Peter Wallin‡

*Mälardalen University, Sweden † Arcticus Systems ‡ Volvo Construction Equipment, Sweden

saad.mubeen@mdh.se

Extended Abstract

Model-based development of software architecture for real-time embedded systems in modern vehicles has had a surge in the last few years. While the introduction of models into the development of real-time embedded systems has increased efficiency in some parts of the engineering process, the models are also cause of novel concerns. In particular, mismatch between structural and semantic assumptions in modeling languages used in different parts of the design process of such systems cause large problems when design artifacts are transformed between modeling languages.

In the industry, productivity is hampered by incompatible tools and file-formats, in conjunction with the need for non-trivial, manual and tedious translations between different model-formats. Moreover, these translations are done in ad hoc fashion making the result of the translation unpredictable and potentially with altered semantics. There is a strong need to investigate how to effectively and efficiently work with exiting modeling languages for real-time embedded systems. A solution must entail possibilities to make tools inter-operable to allow automated (and semi-automated) translations between modeling languages and tools with preserved semantics.

In this paper, we present the work in progress on bridging the semantic gap that exits between such models that are used for the development of real-time embedded systems in the segment of construction-equipment vehicles. Benefits that are sought include; use of precise and unambiguous notations to describe complex features, preservation of timing properties after translations, faster turn-around times in early design-phases, possibilities to automatically derive test-cases and possibilities to automatically generate code.

A. Main Goals and Contributions

We are investigating how research oriented and/or standardized component models intended for the automotive real-time embedded systems can be used together with component models actually used in the industry today to provide both a functional description of the system as well as providing an analyzable and a resource-efficient model.

1) Model inter-operability: Since different modeling languages support different types of expressions, it is often impossible to define a one-to-one mapping between different constructs in different languages. However, if designers choose to use a certain subset of the full expressiveness of a language, or choose to use a certain style of expression in the language, it can be possible to define unambiguous mappings from those subsets or styles to other languages. Such subsets or styles can be expressed by a set of patterns. Our goal is to identify such patterns to allow expression of common solutions in a transformable style.

2) Automated model translation: Given above identified patterns that can be translated with preserved semantics, we need to find out which of these can be automatically translated and derive the corresponding translations.

3) Identification of non-translatable models: We also need to identify when a construct cannot be translated with preserved semantics. This functionality is useful both to find errors in the model, and to prevent erroneous translations with (potentially subtle) modifications to model semantics.

B. Research Plan and Current Work

In particular, we attack the gap between functional models (expressed in standard languages as EAST-ADL [3] and/or proprietary languages such as Simulink or Statemate) and execution models (expressed in standard languages like AUTOSAR [2], TADL [4] and TIMMO [5] and/or proprietary languages like Rubus Component Model, RCM [1, 7, 6]) used for the development of software for real-time embedded systems. We plan to extract a large enough set of patterns from existing solutions at the partner companies and existing literature. Development of automated translations and also automatic detection of design-patterns that do not allow unambiguous transformation will be the second step. Based on these automated translations, it will be possible to assemble a seamless tool chain for the development of software for vehicular real-time embedded systems.

References

[1] Arcticus Systems. http://www.arcticus-systems.com.

[2] AUTOSAR Overview, Rel.4.0, Ver.1.2.1., Rev.3, 2012.

[3] EAST-ADL Domain Model Specification, D4.1.1.

[4] Timing Augmented Description Language, Ver.2, D6.

[5] TIMMO Methodology , Ver.2, D7. October 2009.

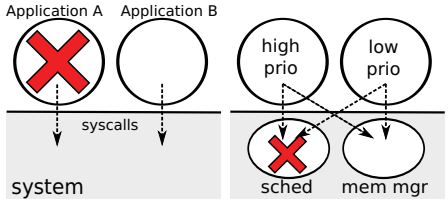
[6] S. Mubeen, J. Mäki-Turja, and M. Sjödin. Support for Holistic Response-time Analysis in an Industrial Tool Suite: Implementation Issues, Experiences and a Case Study. In *19th IEEE Conference on Engineering of Computer Based Systems*, pages 210 –221, April 2012.

[7] S. Mubeen, J. Mäki-Turja, M. Sjödin, and J. Carlson. Analyzable modeling of legacy communication in component-based distributed embedded systems. In *37th Euromicro Conference on Software Engineering and Advanced Applications*, pages 229–238, Sep. 2011.

Predictable, System-Level Fault Tolerance in Composite

Jiguo Song, Gabriel Parmer
The George Washington University
Washington, DC
{jiguos,gparmer}@gwu.edu

Intermittent faults are an increasingly challenging difficulty in embedded and real-time systems. As process technologies shrink circuitry, it becomes increasingly susceptible to transient faults from radiation sources such as cosmic rays. Additionally, as software complexity increases, intermittent faults such as race conditions challenge software reliability. Given these motivations, research has approached the paired problems of recovering from a fault, and doing so predictably. However, most past research has been limited in focus to the predictable recovery of faults at the *application-level*. Examples include systems infrastructures [2] enabling application fault recovery, and scheduling theory [3] that considers periodic faults, and the impact on schedulability for recovery and re-execution of failed applications.



(a) Traditional Systems (b) C^3

Figure 1: (a) Traditional application-level, predictable fault tolerance using a technique such as check-pointing or recovery blocks for recovery. (b) C^3: for the recovery of failed system services, within bounded time, and at the priority of any application that requires service.

In this paper, we discuss a system we're researching called C^3, the Computational Crash Cart. Table 1(a) depicts a traditional system focused on application-level recovery, and (b) C^3 for predictable recovery of system-level services such as the scheduler and memory manager. We implementing C^3 in the COMPOSITE component-based operating system [1].

Challenges and techniques for predictable, system-level fault recovery:

- **Fault propagation.** Service implementation in the kernel in common monolithic systems makes fault propagation more likely. A fault in one logical service cannot be prevented from corrupting memory in other, unrelated system services. C^3 uses COMPOSITE's pervasive use of hardware protection domains to constrain propagation.
- **State recovery.** When a system service fails, it is not straightforward to recover its state. Such services are highly concurrent, and contain data-structures describing many task's resources. To illustrate the difficulties in state recovery, we observe a traditional technique: checkpointing. A checkpoint of the service has periods of inconsistency with the state of a task – a scheduler that dispatches a task after a checkpoint will lose the accounting for that time if it rolled back. Instead, C^3 takes advantage of communication protocols between tasks and services, and records the state of all resources manipulated via that communication. This communication is *replayed* when a service fails to reestablish a consistent service state.

Operation	Memory Manager	Scheduler
Reboot: End to End	35.24(0.61)	46.34(1.65)
Reboot: Memory Ops	31.72(0.33)	36.86(1.72)
COMPOSITE: Invocations	2.10(0.23)	1.29(0.13)
C^3: Invocations	2.31(0.08)	1.33(0.02)

Table 1: Recovery Costs (avg(stddev) in μsecs)

- **Prioritized recovery.** System services might be utilized by both hard real-time tasks of different priorities, and best-effort tasks. The recovery of a system service should execute at the proper system priority. C^3 does this via 1) explicit priority inheritance of the recovery process, and 2) per-task recovery of their own state in the service via replay. These techniques bound interference in the recovery process, and minimize inter-task interference.
- **Schedulable service failure.** Traditional techniques [3] for scheduling recovery do not apply to system service recovery as the timing of possibly all tasks in the system are effected by recovery. We are currently defining schedulability analysis for system recovery in C^3.

We are not focusing on related problems such as fault detection, and instead rely on complementary techniques [4].

Table 1 shows preliminary C^3 experiments for the recovery of the system scheduler and physical memory mapper, measured on an Intel i7 at 2.4 Ghz. Each service executes in a private memory protection domain (via page-tables), and invocations are via RPC. For the memory manager, a task maps a page, aliases it, and then removes both mappings (3 RPCs). For the scheduler, two threads switch back and forth by blocking and waking up (2 RPCs and a thread switch). End to end recovery costs measure the cost of these invocations when a fault occurs in the service. This necessitates rebooting the service, and replaying the communication from the tasks. We note that the costs of recovery are small (35 and 46 μsec) with small variation, and are dominated (over 80% of the cost) by `mem_cpy` and `mem_set` to reset the service's memory to an initial state. The overhead of tracking communication between components differs for the services: 12% overhead for C^3 invocations over native COMPOSITE for the memory manager, and 2% for the scheduler.

Continued and future work. We believe these results show the promise of C^3, and system-level fault recovery. Research remains to address the issues of schedulability, generality to other system services, ease of programming, and a validation of the timing properties.

References

[1] The COMPOSITE component-based system: http://composite.seas.gwu.edu.
[2] A. Egan, D. Kutz, D. Mikulin, R. Melhem, and D. Mosse. Fault-tolerant rt-mach and an application to real-time train control. *Software Practice and Experience*, 1999.
[3] P. Mejia-Alvarez and H. Aydin. Scheduling optional computations in fault-tolerant real-time systems. In *RTCSA*, 2000.
[4] K. Pattabiraman, V. Grover, and B. Zorn. Protecting critical data in unsafe languages. In *Eurosys*, 2008.

Real-time Fault Tolerant Deployment and Configuration Framework for Cyber Physical Systems

Subhav Pradhan, Aniruddha Gokhale, William R. Otte, Gabor Karsai
Dept of Electrical Engineering and Computer Science
Vanderbilt University
Nashville, TN 37235, USA
{subhav.m.pradhan, a.gokhale, w.otte, gabor.karsai}@vanderbilt.edu

ABSTRACT
This paper describes ongoing work on making the deployment and configuration functionality for cyber physical systems reliable and tolerant to failures, while also supporting predictable and incremental online redeployment and reconfiguration of application functionality. Our work is currently designed and evaluated in the context of a system of fractionated spacecrafts, which is a representative CPS system.

Categories and Subject Descriptors
C.4 [**Computer-Communication Networks**]: Distributed Systems—*Fault tolerant software deployment*

General Terms
Reliability, Software Deployment

Keywords
CPS, deployment and configuration, reliability

1. INTRODUCTION
In cyber physical systems (CPS), such as air traffic management, distributed smart grid, and fractionated spacecrafts, deployment and configuration (D&C) of application components to physical resources of the system must happen in a timely and predictable manner. Although timely and predictable D&C is an important requirement for mission-critical CPS, at least two additional key requirements are expected of any D&C capabilities for CPS. First, these CPS applications require that the D&C capabilities themselves be reliable and tolerant to failures where recovery from failures is timely and predictable, and minimizes adverse impact on CPS quality of service (QoS) requirements. Second, due to changes in the mode of operation or fluctuations in resource availability, mission-critical CPS often must undergo redeployment and reconfiguration including incremental D&C, which must be supported predictably within any D&C capability designed for the CPS.

Our prior work on a D&C engine called LE-DAnCE (Locality Enabled Deployment and Configuration Engine) [3, 1] has focused on making application deployment and configuration predictable and timely for large-scale mission-critical cyber applications. However, our prior work did not consider reliability and predictable fault tolerance as well as incremental runtime (re)deployment and (re)configuration. Moreover, our prior work also did not consider extending the functionality to support CPS applications.

2. ONGOING WORK AND CHALLENGES
Our ongoing work focuses on supporting these key requirements within the LE-DAnCE framework and is evaluated in the context of fractionated spacecrafts [2], which is a representative CPS that brings new challenges to an already hard set of problems. For example, in our prior work on LE-DAnCE we had assumed stable networks. For fractionated spacecrafts, the available network bandwidth changes with changing distance between spacecrafts and also the positioning of the spacecrafts and their orbits. Furthermore, space hardware is inherently constrained in resources, which means spacecrafts will comprise limited computing power and memory in addition to the need to conserve power. Thus, significant optimizations are needed to existing D&C schemes to address these limitations.

Currently we are focusing on implementing a pluggable verification mechanism within the LE-DAnCE framework. These plugins are decoupled from the common framework, and help to ascertain if correctness and QoS criteria for the specific application is satisfied.

3. REFERENCES
[1] G. Deng, J. Balasubramanian, W. Otte, D. Schmidt, and A. Gokhale. Dance: A qos-enabled component deployment and configuration engine. *Component Deployment*, pages 67–82, 2005.

[2] A. Dubey, W. Emfinger, A. Gokhale, G. Karsai, W. Otte, J. Parsons, C. Szabó, A. Coglio, E. Smith, and P. Bose. A software platform for fractionated spacecraft. In *Aerospace Conference, 2012 IEEE*, pages 1–20. IEEE, 2012.

[3] W. Otte, A. Gokhale, and D. Schmidt. Predictable deployment in component-based enterprise distributed real-time and embedded systems. In *Proceedings of the 14th international ACM Sigsoft symposium on Component based software engineering*, pages 21–30. ACM, 2011.

ProtoDrive: An Experimental Platform for Electric Vehicle Energy Scheduling and Control

William Price Harsh Jain Yash Pant Rahul Mangharam
Dept. Electrical & System Engineering
University of Pennsylvania
{wprice, harjain, yashpant, rahulm}@seas.upenn.edu

Categories and Subject Descriptors

B.8 [**Performance of Systems**]: B.8.1 Reliability, Testing, and Fault-Tolerance

Keywords

Real-time scheduling, control systems, electric vehicles

1. INTRODUCTION

Vehicles involved in urban commutes are subjected to highly variable loads as they traverse varying gradients and stop-and-go traffic. Electric Vehicles can achieve a high efficiency under these conditions due to their ability to recover energy during braking. However, the high current loads during both charging and discharging cause battery energy losses, making them less efficient and degrading their useful lifetime. Super capacitors work well under high power charge and discharge cycles, however, their high cost and low energy density prevent them from being a viable replacement for batteries. A hybrid system consisting of a battery and a super capacitor has the potential to offer the benefits of both devices, which may increase vehicle range and battery lifetime. Consequently, the goal of the project is to: (a) investigate the use of a hybrid battery/super capacitor system in response to real commuter drive cycles. (b) develop scheduling algorithms that optimize the flow of energy between the battery, super capacitor and motor.

In order to simulate a battery/super capacitor system in software, it is necessary to know a vehicle's physical parameters as well as the velocity profile, altitude and weather conditions of the route followed. This information can be obtained from conventional internal combustion engine cars outfitted with a GPS unit, and CMU's ChargeCar project has already collected this data for many cars, all over the US. Additionally, they have written software to simulate the vehicle's performance if it were electric, and also if it had a super capacitor. This software demonstrated that there can be a significant reduction in the duty of a battery by using a super capacitor, and that this reduction is highly dependent on the route travelled.

2. PROTODRIVE PLATFORM OVERVIEW

A software simulation of a drive cycle provides some good insight into the problem but the model may be somewhat idealized and omits the intricacies that are present in a real vehicle. However, it is both time and cost intensive to build a real battery and super capacitor vehicle and attempt to drive the multitude of real commuting routes for the sake of predicting how such a vehicle would perform on the given route. Additionally, using a human as the controller of the vehicle makes it difficult to produce repeatable results.

To this effect, we are developing a low cost, small-scale electric vehicle platform called Protodrive which is capable of simulating a drive cycle in hardware, while remaining small enough to fit on a lab desk. It consists of a physical model of an electric vehicle powertrain (motor, controller, battery, super capacitor) coupled to an active dynamometer, making it possible to run the powertrain through its full speed and torque range. Electronic control of the platform enables consistent testing conditions and fair comparison between battery and hybrid systems, and simulation in hardware will capture elements of the real system that may be missed in an idealized software model.

3. PROTODRIVE – SYSTEM EVALUATION

In our discussion, a velocity profile, altitude data, vehicle parameters and weather data will be used as inputs to generate a scaled torque and speed profile which will be run on the Protodrive hardware. Various power distribution schedules are being implemented over the drive cycle, enabling the comparison of a hybrid system to a battery-only system, or the comparison of various current distribution algorithms. The output will show the current load on the battery and the super capacitor, which can be used to determine the battery's State of Charge and the efficiency of the vehicle. Ultimately, we aim to determine if a battery/supercapacitor system offers significant benefits over a battery-only system, by simulating real commuting routes in hardware.

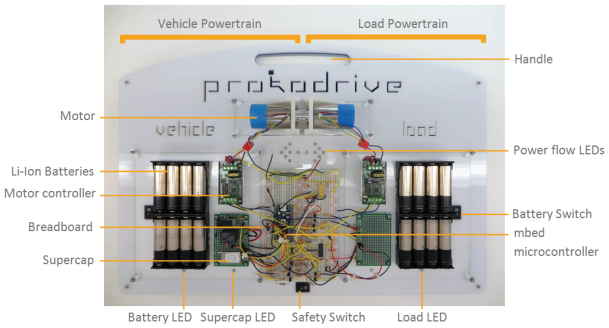


Figure 1: ProtoDrive Platform with coupled vehicle and load motors, battery and super capacitor.

MLE+: A Tool for Integrated Design and Deployment of Energy Efficient Building Controls

Willy Bernal, Madhur Behl, Truong Nghiem and Rahul Mangharam
Department of Electrical and Systems Engineering
University of Pennsylvania
{mbehl, willyg, nghiem, rahulm}@seas.upenn.edu

ABSTRACT

Simulation engines for buildings can be realistic and accurate, but only provide basic control interfaces. Control engineers have developed robust and complex controls for energy-efficient building operation though such methods are often based on simplistic physical models. To address this issue, we developed MLE+, a tool for energy-efficient building automation design, co-simulation and analysis. This tool leverages the high-fidelity plant simulation capabilities of EnergyPlus building modeling software and the scientific computation and design capabilities of Matlab/Simulink for controller design. MLE+ facilitates integrated building simulation and controller formulation with integrated support for system identification, control design, optimization, simulation analysis and communication between software applications and building equipment. MLE+ has been successfully used to generate schedules for building HVAC control systems for peak power minimization.

1. INTRODUCTION

MLE+ (Fig. 1) is intended as a tool for building energy research and development for researchers familiar with Matlab that want to couple it to realistic building energy simulation software like EnergyPlus. EnergyPlus is one of the most robust and complete buidling energy analysis and thermal load simulation tools available and it has become the de facto whole building simluation of the U.S. Department of Energy. The following are the main features of MLE+:

1. **Simulation configuration**: The MLE+ front-end (Fig. 2) streamlines the co-simulation configuration process by linking the building model and the controllers. This reduces setup time and configuration problems.
2. **Controller design**: MLE+ provides a control development workflow and graphical front-ends for designing advanced control strategies.
3. **Simulation-based optimization**: MLE+ can be used to find optimal parameters or control sequences for building system simulations in EnergyPlus.
4. **Data analysis**: After a co-simulation run, the output data from EnergyPlus can be aggregated, analyzed and visualized in Matlab.
5. **Building Management System Interface**: MLE+ provides a BACnet interface for real-time deployment of green schedulig methods for building equipment.
6. **Matlab environment**: MLE+ allows complete access to the Matlab environment and toolboxes such as Global Optimization Toolbox, System Identification Toolbox and Model Predictive Control Toolbox.

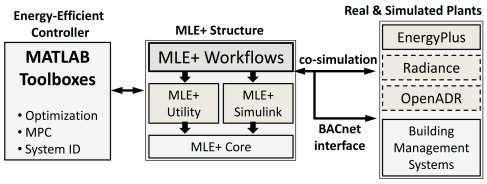


Figure 1: MLE+ interfaces control systems toolboxes with building models and systems

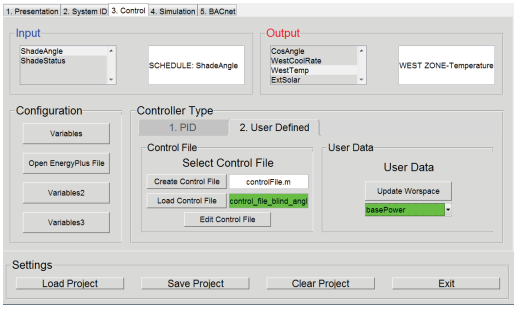


Figure 2: MLE+ tool interface

2. TEST CASES

We have implemented two test studies to showcase MLE+ capabilities. The objective of the first case is to generate energy-efficient HVAC schedules for a building simulated in EnergyPlus and to show the potential and ease of use of MLE+ in implementing and evaluating such controls. For the second case study, MLE+ is interfaced with a instrumented test-bed that simulates both the dynamics of a building and the behavior of BACnet devices. The test-bed consists of a scaled building with four zones with independent heating and cooling elements. Sensor nodes monitor the temperature and energy levels in different zones of the building. The test-cases are simple examples that pave the way for using MLE+ for modern energy-efficient control methods such as MPC.

3. CONCLUSION

MLE+ provides high fidelity simulation while allowing complete access to Matlab's built-in capabilities and computational power to design advanced building controls. Our current effort involves extentions to work with other building energy simulation tools like Radiance and OpenADR. Also, we aim at coupling it with other optimization and modeling tools.

Resource Sharing under Server-based Multiprocessor Scheduling

Sara Afshar, Moris Behnam
Mälardalen University, Västerås, Sweden
{sara.afshar, moris.behnam}@mdh.se

ABSTRACT

In this paper, we investigate a mechanism for handling resource sharing among tasks under a server-based scheduling technique in multiprocessor platforms, which combines partitioned and global scheduling to benefit a better scheduling method compared to conventional techniques.

1. INTRODUCTION

Semi-partitioned scheduling for multiprocessors benefits from both conventional global and partitioned approaches such that most tasks are assigned statically to processors similar to partitioned scheduling, while a low number of tasks are split and migrate among processors similar to global scheduling, [1, 2, 3]. Another recent multiprocessor scheduling approach is based on hierarchical scheduling, which utilizes servers and is called Synchronized Deferrable Servers (*SDS*) [4]. Under SDS, similar to the semi-partitioned approach, some tasks are bound to processors (*non-migrating tasks*), while others migrate among processors (*migrating tasks*). The key difference with semi-partitioned scheduling is that in SDS the migrated tasks are processed within servers allocated to processors. The major distinction between the semi-partitioned approach and SDS is that under SDS tasks which migrate between processors may run in any available server on any processor while in the semi-partitioned approach each part of a split task always executes on a specific processor which is determined during the partitioning phase. Therefore, SDS provides more flexibility to execute migrating tasks on processors that can improve the schedulability performance. However, in [4] it is assumed that tasks are independent, i,e., they do not share any resource. In this paper, we propose a resource sharing protocol for the case when tasks under the SDS multi-core hierarchical scheduling share resources with each other. The main challenge is to adjust the response time analysis presented in [4] to include the effect of resource sharing.

2. General Description

Our considered system consists of a set of *n* tasks that run on a set of *m* identical processors. One deferrable server is assigned to each processor that could provide a capacity during partitioning phase. We define a common replenishment period for all servers in the system. The scheduling policy includes two levels: (i) a fixed priority uniprocessor scheduling that schedules non-migrating tasks along with the server on each core and (ii) migrating tasks scheduling decision which determines in which server the non-migrating task execute. Next we develop our protocol rules based on the SDS structure [4] and inspiration from the synchronization protocol for semi-partitioned system [5].

1) Local resources are handled by uniprocessor protocols.
2) One global priority-ordered queue (*Q)* enqueues the ready or preempted migrating tasks. However, in each processor a local ready queue enqueues the non-migrating tasks.
3) After a migrating task is released, it is added to *Q*. The task at the head of *Q* executes on a ready server with the available capacity. If more than one ready server is available, the server with highest assigned capacity is chosen. If the lowest priority running task in any server has a priority lower than that of the task at the head of *Q,* it will be preempted.
4) A global queue is dedicated to each global resource to enqueue tasks from different processors which get blocked on the resource; however one local queue is assigned to each processor to enqueue local non-migrating tasks that are granted access to different global resources. The tasks in the global resource queues can be migrating or non-migrating tasks. However, the migrating tasks which are granted access to their requested resource will be inserted and wait in *Q*.
5) All resources that are requested in the migrating tasks are assumed global since they can be requested in any processor.
6) The priority of a task accessing a global resource is boosted to maximum priority to decrease the blocking times of tasks.
7) In order to prevent a migrating task holding a resource to migrate to another processor, an overrun approach is performed if the capacity of the server is finished and the task is in a global critical section.

In our ongoing work we are performing system analysis, and the challenge is to find an upper bound of the response time of migrating tasks which share resources with other parts of the system and can execute in any server on any processor.

3. REFERENCES

[1] J. Anderson, V. Bud, and U. Devi, "An EDF-based scheduling algorithm for multiprocessor soft real-time systems," *(ECRTS'05)*.

[2] S.Kato and N. Yamasaki, "Semi-partitioned fixed-priority scheduling on multiprocessors," *(RTAS'09)*.

[3] N. Guan, M. Stigge, W. Yi, and G. Yu, "Fixed-priority multiprocessor scheduling with Liu and Layland's utilization bound," *(RTAS'10)*.

[4] H. Zhu, S. Goddard, and M. Dwyer, "Response time analysis of hierarchical scheduling: The synchronized deferrable servers approach *(RTSS'11)*.

[5] S. Afshar, F. Nemati, and T. Nolte, "Resource sharing under multiprocessor semi-partitioned scheduling," *(RTCSA'12)*.

Using NPS-F for Mixed-Criticality Multicore Systems

Konstantinos Bletsas and Stefan M. Petters
CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Portugal
ksbs,smp @isep.ipp.pt

1. MOTIVATION

Hard real-time multiprocessors scheduling has recently seen the flourishing of semi-partitioned scheduling algorithms – a category of scheduling schemes that combine elements of partitioned and migrative scheduling to allow more efficient processor usage, while providing improved schedulability guarantees at the same time. Yet, so far, semi-partitioning has not made any inroads into mixed-criticality scheduling. We aim to address this by proposing a way of combining mixed-criticality scheduling with semi-partitioning. Specifically, we adapt the NPS-F scheduling algorithm [1] for this purpose. The timeslot-based dispatching of NPS-F allows for fairness and responsiveness for lower-criticality workloads with no detriment to the schedulability guarantees of high-criticality tasks.

NPS-F can be outlined as follows. Initially, tasks are partitioned to servers offline, using bin-packing. These servers are then mapped to one or more processors each (using non-overlapping time windows, for each server that employs multiple processors), creating a sort of cyclic executive which repeats every S time units – the timeslot. At run-time dispatching inside each server is dynamic, using EDF. The share of the timeslot S reserved for each server is static (computed offline) and depends on its workload. The principle guiding the sizing of each server is that its share of the timeslot should be sufficiently large (according to schedulability analysis) such that no deadlines may be missed by the tasks it serves.

2. NPS-F FOR MIXED CRITICALITIES

Our approach for adapting NPS-F to mixed-criticality workloads consists in partitioning the high-criticality tasks ("H-tasks") over m non-migrating servers, each server mapped to one corresponding processor. The remaining capacity on all processors (the shares of the timeslot that remain unassigned) is "recycled" for mapping servers for the non-critical workload ("L-tasks").

Two system operation modes exist: "normal" mode and "high-criticality" mode. Each H-task τ_i has two different estimates of its WCET: C_i^L, which is considered sufficient, but lacks proof and C_i^H, which is provably always safe and possibly much greater than C_i^L. For L-tasks, only C_i^L is determined. Under "normal" mode, every task τ_i (irrespective of its criticality) executes and both critical and non-critical tasks are guaranteed to meet their deadlines as long as they run for no more than their low-criticality estimate C_i^L. However, as soon as any H-task overruns its C_i^L, the system switches to "high-criticality" mode: non-critical workload is idled and the system switches to partitioned EDF scheduling of H-tasks only, assuming a WCET of C_i^H for each H-task. In high-criticality mode, and during the mode transition, the deadlines of all H-tasks must be met, which poses challenges.

A naive approach would (i) partition the H-tasks such that they are schedulable under uniprocessor EDF, considering the overly conservative estimates C_i^H (to meet deadlines in high-criticality mode) and, for normal mode execution, would (ii) assign budgets to the respective servers so that they meet deadlines with WCETs of C_i^L. However, this may lead to missed deadlines during mode transition.

A conservative approach would, instead size budgets for H-task servers according to their C_i^H. However, this would be too inefficient for normal mode operation, because it would limit the processor capacity available for scheduling L-task servers. Ideally, we would like to analytically identify the "sweet spot" between these two extremes, so as to (i) minimise the processor capacity used for H-task servers (i.e. maximise the capacity available for L-task servers), while (ii) ensuring that no H-tasks may ever miss deadlines (even when a mode transition is triggered).

3. APPROACH OUTLINE

The approach taken centers around (i) identifying (after an overrun has been detected in a server) the amount of execution which may need to be executed before the respective task deadlines and (ii) sizing the servers such that the point of detection is sufficiently early to execute the remaining workload in time. Issues to solve are (i) identifying the critical instant, which leads to the latest detection of an overrun of any task and (ii) sizing the servers such that the regular load can be executed beforehand. The particular challenge here is that these two issues are not independent.

Pen and paper exercises with a number of case studies indicate that most servers can be kept at substantially lower utilization than what is required if straight C_i^H is considered. The benefit of the proposed approach is that it maintains the isolation to L-tasks, in most cases with a larger share utilization than what would otherwise be possible, leading to a more even and fair distribution of resources.

Acknowledgements: Work partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within SMARTS project, ref. FCOMP-01-0124-FEDER-020536.

4. REFERENCES

[1] K. Bletsas and B. Andersson, "Preemption-light multiprocessor scheduling of sporadic tasks with high utilisation bound," *Journal of Real-Time Systems*, vol. 47, no. 4, pp. 319–355, 2011.