

Report from Dagstuhl Seminar 17131

Mixed Criticality on Multicore / Manycore Platforms

Edited by

Liliana Cucu-Grosjean¹, Robert Davis², Sanjoy K. Baruah³, and Zoë Stephenson⁴

1 INRIA – Paris, FR, liliana.cucu@inria.fr

2 University of York, GB, rob.davis@york.ac.uk

3 University of North Carolina at Chapel Hill, US, baruah@cs.unc.edu

4 Rapita Systems Ltd. – York, GB, zstephenson@rapitasystems.com

Abstract

This report provides an overview of the discussions, the program and the outcomes of the second Dagstuhl Seminar on Mixed Criticality on Multicore/Manycore Platforms. The seminar brought together researchers working on mixed criticality real-time applications, industrialists from the aerospace, railway, and automotive industries, and experts in certification.

Seminar March 26–31, 2017 – <http://www.dagstuhl.de/17131>

Keywords and phrases mixed-criticality multicore manycore real-time-systems

Digital Object Identifier 10.4230/DagRep.7.3.70

Edited in cooperation with Adriana Gogonel

1 Executive Summary

Liliana Cucu-Grosjean

Robert I. Davis

Sanjoy K. Baruah

Zoë Stephenson

License © Creative Commons BY 3.0 Unported license

© Liliana Cucu-Grosjean, Robert I. Davis, Sanjoy K. Baruah, Zoë Stephenson

Real-time applications are characterized by the need for both functional correctness and temporal correctness (appropriate timing behaviour). Real-time systems are present in many diverse areas such as avionics, automotive, space, robotics, and medical applications to cite only a few. Mixed Criticality Systems (MCS) have become an important topic for the real-time systems community. The first cluster of the European collaborative projects on MCS has been completed in September 2016, indicating a maturing of the related concepts within both industry and academia. Nevertheless many of the challenges brought about by the integration of mixed criticality applications onto multicore and manycore architectures remain to be solved. In reality mixed criticality problems have inherited the difficulty of real-time systems: being at the frontier of several domains including real-time scheduling, real-time operating systems / runtime environments, and timing analysis, as well as hardware architectures. This seminar promoted lively interaction, cross fertilization of ideas, synergies, and closer collaboration across different sub-communities of academics and industrialists from aerospace, automotive, and railway industries with specific interests in MCS, as well as with experts in certification.



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Mixed Criticality on Multicore / Manycore Platforms, *Dagstuhl Reports*, Vol. 7, Issue 3, pp. 70–98

Editors: Liliana Cucu-Grosjean, Robert Davis, Sanjoy K. Baruah and Zoë Stephenson



DAGSTUHL
REPORTS

Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In common with the first Dagstuhl Seminar on Mixed Criticality Systems, this seminar also focused on the two key conflicting requirements of MCS: separation between criticality levels for assurance and sharing for resource efficiency, along with the related requirement of time composability. An important aspect of this seminar was the presentation of different industry perspectives on the key problems. These perspectives formed the starting point of our seminar, with the first day mainly dedicated to industry statements on current practice and their perception of current work on MCS. The academic participants benefited from substantial and detailed arguments from the industry speakers. There were lively interactive discussions during the talks which led to much improved understanding of current industry practice, as well as helping to build a common vocabulary between academic and industry participants. The first day concluded with presentations by academic speakers presenting their thoughts on more practical mixed criticality models.

The next three days each included sessions devoted to an invited tutorial from a academic speaker. These covered the one-out-of-m multicore problem, Networks-on-Chip and mixed criticality, resource management, and statistical approaches to worst-case execution time estimation. The remaining sessions covered a range of fascinating open problems. In addition, a number of ad-hoc small working groups formed to collaborate on specific topics. We are pleased to report that a significant number of these initial collaborations have gained traction resulting in further work after the seminar, and in some cases the development and submission of papers.

Organization of the seminar report. Section 3 is an overview of the industry talks and Section 4 provides an overview of the academic talks. Section 5 presents working group discussions. Section 6 summarizes open problems discussed during the seminar. Finally outcomes from the seminar are listed in Section 7.

As organizers, we would like to thank Prof. Reinhard Wilhelm for joining us, Dagstuhl's Scientific Directorate for allowing us to run a second seminar on mixed criticality systems, and to the staff at Schloss Dagstuhl for their superb support during the seminar itself.

Finally, we would like to thank all of the participants for the very lively and open discussions. As organizers, we appreciated the feedback and enthusiasm which made running the seminar a great pleasure.

2 Table of Contents

Executive Summary

Liliana Cucu-Grosjean, Robert I. Davis, Sanjoy K. Baruah, Zoë Stephenson 70

Overview of industrial talks

Mixed Criticality Systems – view from the industry side

Cristian Maxim 74

Real-Time Systems in Railway

Stefan Resch 74

An independent assessors perspective

Philippa Ryan 76

Mixed Criticalities in Avionic Systems

Sascha Uhrig 76

Mixed Criticality and Real-Time in Automotive

Dirk Ziegenbein 76

Overview of academic talks

Deriving precise execution-time distributions of tasks

Sebastian Altmeyer 77

Realistic task model for multicore processors

Sebastian Altmeyer 77

Towards a (more) realistic task model for multicore processors

Sebastian Altmeyer 78

The One-Out-of-m Multicore Problem

James H. Anderson 78

Schedulability Analysis as Evidence?

Björn B. Brandenburg 79

How to Gracefully Degrade

Alan Burns 80

Resilient Mixed-Criticality Systems

Alan Burns 80

Reliability Optimization in MC2 Systems

Thidapat Chantem 80

Worst Case Execution Time measurement-based approach

Liliana Cucu-Grosjean, Adriana Gogonel, and Cristian Maxim 81

Practical Mixed-Criticality Model: Challenges

Arvind Easwaran 82

Runtime Verification, Runtime Enforcement, and Mixed Criticality System Design

Sébastien Faucou 82

Energy efficiency in factories: Benefit of renewable energy, IoT and Automatic Demand Response

Laurent George 83

Real-Time Mixed-Criticality Wormhole Networks <i>Leandro Soares Indrusiak</i>	83
Fixed-Priority Scheduling without Any Adaptation in Mixed-Criticality Systems <i>Jian-Jia Chen</i>	84
Improve the scalability of mixed-criticality parallel real-time systems <i>Jing Li</i>	84
Resource management in DREAMS <i>Claire Pagetti</i>	85
Probabilistic Analysis for Mixed Criticality Scheduling with SMC and AMC <i>Dorin Maxim, Liliana Cucu-Grosjean, Robert Davis, and Arvind Easwaran</i>	86
Timing Compositionality – Challenges and Opportunities <i>Jan Reineke</i>	86
Working Group Discussions	
The Meaning and Use of probabilistic Worst-Case Execution Time (pWCET) Distributions <i>Robert I. Davis and Alan Burns</i>	87
Open problems	
Mixed criticality scheduling under resource uncertainty <i>Kunal Agrawal</i>	91
Deriving Optimal Scheduling Policies for MC Task Systems <i>Sathish Gopalakrishnan</i>	91
Guaranteeing some service upon mode switch in mixed-criticality systems <i>Zhishan Guo</i>	91
Regarding the Optimality of Speedup Bounds of Mixed-Criticality Schedulability Tests <i>Zhishan Guo</i>	93
MC-ADAPT: Adaptive Mixed Criticality Scheduling through Selective Task Dropping <i>Jaewoo Lee</i>	94
Schedulability, Probabilities and Formal Methods <i>Luca Santinelli</i>	95
Safety Calling <i>Zoë Stephenson</i>	95
Outcomes of the seminar	97
Participants	98

3 Overview of industrial talks

3.1 Mixed Criticality Systems – view from the industry side

Cristian Maxim (Airbus S.A.S. – Toulouse, FR)

License  Creative Commons BY 3.0 Unported license
© Cristian Maxim

In avionics industry criticality is a designation of the level of assurance against failure needed for a system component and the notion of mixed-criticality is treated differently than in research domain. In my presentation I spotted the differences between the Vestal model and the approach in industry, explaining notions like safety integrity level and design assurance level (DAL). In industry the DAL is determined from the safety assessment process and hazard analysis and each software is included in one of the five distinct levels. The presentation focused on the way these levels are obtained and gave examples of softwares and the corresponding DALs. The main discrepancies between the levels of criticality in avionics and Vestal's model are:

1. In industry the criticality is given to a function while in research the criticality applies to a task.
2. For certification, the airplane manufacturers are supposed to give one WCET value while in research the concept of multiple WCET values for higher criticality tasks is observed.
3. The difficulty of implementation of Vestal's model makes it hard to benefit from the better CPU usage given by the existence of WCET for low criticality of certain tasks.
4. Task dropping is not conceivable in industry and the spatial isolation doesn't allow failure in a function to affect other functions.
5. The mode change in case of time violations would imply a new certification procedure and that is too costly to be practical. In the second part of the presentation, the IMA (integrated modular avionics) concept was presented as a midway between research and industry, focusing on the isolation procedures.

3.2 Real-Time Systems in Railway

Stefan Resch (Thales – Wien, AT)

License  Creative Commons BY 3.0 Unported license
© Stefan Resch

In the railway domain systems with various timing requirements are to be found. By nature these systems are distributed over long distances. There are elements on track side such as axel counters, point control and signals, as well as interlocking systems in data centers and operation management centers controlling large parts of a country's railway network. On the trains there are on-board systems supervising and assisting the driver and communicating with the interlocking systems and operation control through balises (electronic beacons) or via GSM-R and radio block centers. The timing requirements of all these systems are highly dependent on their provided functions.

To avoid re-developing services for fault-tolerance, communication, real-time, etc., Thales provides a generic platform for its safety-critical railway applications – the TAS Control Platform¹. It provides hardware boards, software runtimes, development tools and a system safety case according to CENELEC EN 50129. TAS Control Platform is certified to the highest safety integrity level SIL4 as generic safety product. This is achieved by implementing a safety middleware on top of a COTS operating system and hardware. This layer provides replication and supervises all capabilities that are used by the safety-critical applications and provided by the underlying COTS layer, also with respect to the real-time. With a stable API to the applications hardware obsolescence is mitigated in the middleware and the COTS OS.

In the railway domain high integrity is required for safety, but availability is not a direct safety property since railway systems are fail-safe. This can be illustrated with the example of setting a route for a train in a railway network. First all elements of the route, such as points, tracks and signals are reserved. Then all points are commanded to be set into the correct position for the route. The points then report their updated positions. As soon as all elements of the route are in a correct state, the entry signal is set to permissive. In case one of these steps cannot be completed, the setting of the route is aborted. This example illustrates that rather than relying on a concrete action to be completed within a certain amount of time the railway approach is to wait until the system has reached the correct state. An example with tighter timing requirements is that of sending an emergency stop signal via the radio block center to a train. Here, as well as in the previous example, the overall reaction time must be guaranteed by all involved systems.

As in other safety-critical domains, in railway the criticality of a function is derived from the potential damage and likelihood of it being faulty. The application designer then has to ensure that the system design satisfies the according requirements. With respect to scheduling this means that the tasks of the application will have sufficient resources on the computing platform available. In case of mixed criticality, where several applications are integrated on top of the same hardware platform, the required resources must be guaranteed for all the integrated applications. In exceptional overload situation the platform might provide only limited resources to tasks that are marked by the application designer as low priority, independent of their application's level of criticality. This allows graceful degradation of the system, but is a result of the application design and has no direct relation to the criticality level of an application.

Applications on top of TAS Control Platform are supervised through timeouts and the periodic synchronization between different computing boards in redundant architectures. Based on their communication and computation demand and the TAS Control Platform synchronization granularity they can define and supervise whether their reaction time meets the requirements. The actual reaction time is then determined through measurements and verified during the system integration tests.

In the future TAS Control Platform wants to provide the applications the possibility to execute a generic integration test with respect to application requirements and then be deployed in different environments.

¹ This project has received funding from the ECSEL Joint Undertaking under grant agreement No 692455. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

3.3 An independent assessors perspective

Philippa Ryan (Adelard – London, GB)

License  Creative Commons BY 3.0 Unported license
© Philippa Ryan

Independent audits and assessments provide essential unbiased review of computer systems. In domains such as nuclear, defence, avionics and railway they are required by regulators. Few standards offer up to date guidance to deal with the complexity of mixed criticality and multi-core. Current focus is on safety, but security informed safety is increasingly important. With a limited amount of time and budget to perform an audit, how can the assessor be persuaded the system is acceptably safe and secure?

3.4 Mixed Criticalities in Avionic Systems

Sascha Uhrig (Airbus – München, DE)

License  Creative Commons BY 3.0 Unported license
© Sascha Uhrig

Mixed criticality systems on multicores will become very important in the avionic domain in the future. This is because more and more functionality needs to be integrated on light-weight computers demanding for less space and energy. In addition to that the new functionalities need to be even more reliable and available than current high criticality systems, for example because of the demand on autonomous flying vehicles. Current approaches exploiting high multicore performance by switching between two modes according to the actual execution state, i.e. execution times, are good starting points. Nevertheless, these approaches can be difficult to implement (and certify) in avionic systems because of their nature to change the timing (schedule) and consequently the behaviour of the complete system dynamically. Such mode switches will most probably not occur in standard situations tested on ground but in unforeseen situations in which a different behaviour can have unpredictable results. Accordingly, such systems must be designed even more careful than current highly critical systems and future mixed criticality systems are still challenging.

3.5 Mixed Criticality and Real-Time in Automotive

Dirk Ziegenbein (Robert Bosch GmbH – Stuttgart, DE)

License  Creative Commons BY 3.0 Unported license
© Dirk Ziegenbein

The talk gives a short overview of safety criticality, current approaches to real-time assurance as well as future challenges. In automotive systems, the criticality is given as ASIL (Automotive Safety Integrity Level) of a certain function. Since typically several SW and HW units work together to implement the function as well as to fulfil ASIL requirements, the paradigm to drop lower criticality tasks is not applicable in general. This is explained using an example. Timing assurance today is typically based on measured execution times and scheduling analysis or simulation. With the advent of multi-cores this well-known WCET abstraction does no longer hold due to cross-core influences. The trend towards large-scale software integration on heterogeneous HW platforms increases the need to find a new way to characterize the sequential SW units for system-level performance analysis.

4 Overview of academic talks

4.1 Deriving precise execution-time distributions of tasks

Sebastian Altmeyer (University of Amsterdam, NL)

License  Creative Commons BY 3.0 Unported license
© Sebastian Altmeyer

Research on the timing behaviour of embedded real-time systems has been primarily focused on determining the worst-case execution time (WCET). This focus is clearly motivated by the need for timing verification, i.e, the need to guarantee at design time that all deadlines will be met. While a WCET estimate can be used to verify that a system is able to meet deadlines, it does not contain any further information about how the system behaves most of the time. An execution time distribution does contain this information and can provide useful insights regarding the timing behaviour of a system. Furthermore, a correct execution time distribution can be used to evaluate the precision and correctness of (worst-case) execution time analyses. We have recently developed a measurement-based framework that derives execution time distributions by exhaustive evaluation of program inputs. We overcome the scalability and state-space explosion problem by i) using static analysis to reduce the input space and ii) using an anytime algorithm which allows deriving a precise approximation on the execution time distribution. We would like to extend this research to overcome some restrictions on the hardware and execution environment. But foremost, we would like to use the framework to evaluate the precision of recently developed timing analysis approaches.

4.2 Realistic task model for multicore processors

Sebastian Altmeyer (University of Amsterdam, NL)

License  Creative Commons BY 3.0 Unported license
© Sebastian Altmeyer

While processor architectures have changed fundamentally within the last decades, the task model that we still use today remained remarkably simple. Besides periods and deadlines, we mostly argue about execution time bounds. Each variation of this task model creates a plethora of new research questions, be it by defining different execution time bounds per criticality level, or per processor type. What rarely changes, however, is the assumption that the execution time bounds are complete and safe and encompass the entire processor system, irrespective of any interference on shared resources. Such a coarse abstraction mismatches the complexity of modern processors, especially for multi-core architectures with complex bus architectures and memory hierarchies: The processor itself is often not the only scarce resource anymore that needs to be scheduled. Arguing about the computation time is not very useful, when instead the memory bandwidth is the performance bottleneck.

The research problem that I would like to work on within this context are, amongst others:

- How realistic is the current assumption of a single execution time bound, valid for all scenarios, and how much performance do we lose?
- How can we define a more realistic task model that accurately represents not only the computation time, but also other shared resources?

4.3 Towards a (more) realistic task model for multicore processors

Sebastian Altmeyer (University of Amsterdam, NL)

License  Creative Commons BY 3.0 Unported license
© Sebastian Altmeyer

While processor architectures have changed fundamentally within the last decades, the task model that we still use today remained remarkably simple. Besides periods and deadlines, we mostly argue about execution time bounds. Each variation of this task model creates a plethora of new research questions, be it by defining different execution time bounds per criticality level, or per processor type. What rarely changes, however, is the assumption that the execution time bounds are complete and safe and encompass the entire processor system, irrespective of any interference on shared resources. Such a coarse abstraction mismatches the complexity of modern processors, especially for multi-core architectures with complex bus architectures and memory hierarchies: The processor itself is often not the only scarce resource anymore that needs to be scheduled. Arguing about the computation time is not very useful, when instead the memory bandwidth is the performance bottleneck.

The research problems that I would like to work on within this context are, among-st others:

- How realistic is the current assumption of a single execution time bound, valid for all scenarios, and how much performance do we lose?
- How can we define a more realistic task model that accurately represents not only the computation time, but also other shared resources?

4.4 The One-Out-of-m Multicore Problem

James H. Anderson (University of North Carolina at Chapel Hill, US)

License  Creative Commons BY 3.0 Unported license
© James H. Anderson

The multicore revolution is having limited impact in safety-critical application domains. A key reason is the “one-out-of-m” problem: when validating real-time constraints on an m-core platform, excessive analysis pessimism can effectively negate the processing capacity of the additional $m - 1$ cores so that only “one core’s worth” of capacity is utilized even though m cores are available. Two approaches have been investigated previously to address this problem: mixed-criticality allocation techniques, which provision less-critical software components less pessimistically, and hardware-management techniques, which make the underlying platform itself more predictable. A better way forward may be to combine both approaches, but to show this, fundamentally new criticality-cognizant hardware-management trade offs must be investigated. To enable such an investigation, my research group has developed a mixed-criticality scheduling framework called MC^2 that supports configurable criticality-based hardware management. This framework allows specific DRAM memory banks and areas of the last-level cache (LLC) to be allocated to certain groups of tasks. A linear-programming-based optimization framework is available for sizing such LLC areas. In this talk, I will discuss the design of MC^2 and the analysis that underlies it and present the results of an experimental study conducted to evaluate its efficacy. This study shows that mixed-criticality allocation and hardware-management techniques can be much more effective when applied together instead of alone.

4.5 Schedulability Analysis as Evidence?

Björn B. Brandenburg (MPI-SWS – Kaiserslautern, DE)

License © Creative Commons BY 3.0 Unported license
 © Björn B. Brandenburg
 URL <http://prosa.mpi-sws.org>

As part of the certification of safety-critical systems, it is required to make a *safety case*. Such a safety case rests on a series of *arguments* that establish that all unacceptable risks are being mitigated. These arguments in turn must be supported by *evidence* that has been produced using an effective, and widely accepted, *methodology*.

In the context of real-time systems – and in particular in the context of mixed-criticality real-time systems, which distinguish themselves by consisting of complex mixes of workloads with diverse timing requirements and non-obvious correctness criteria, which makes their analysis exceedingly difficult – the generally accepted methodology of ruling out the risk of timing errors is *schedulability analysis*, i.e., static analysis that determines whether all timing constraints will be met at runtime in all possible execution scenarios.

The general consensus of real-time researchers is that schedulability analysis should be employed as part of safety certification of critical real-time systems (e.g., as found in avionics or the automotive industry), as the alternative – purely testing-based methods – cannot yield strong guarantees, and thus inherently constitute a considerable source of uncertainty (or residual risk). In contrast, published and peer-reviewed schedulability analyses are considered to yield *sound* results that leave no room for doubts.

Unfortunately, this trust in published and peer-reviewed schedulability analyses is, historically speaking, not justified: over the years, significant flaws and gaps in proofs have been found in a surprisingly large number of well-known results, including in the foundational Liu & Layland analysis of rate-monotonic scheduling, in the response-time analysis of tasks with arbitrary deadlines, in the response-time analysis of non-preemptive tasks (or network messages as in CAN), in the literature on self-suspensions, in the analysis of multiprocessor real-time scheduling with affinity constraints, and in the worst-case blocking analysis of several classic multiprocessor real-time locking protocols (to name just a few examples; there exist many more).

For the design and certification of mixed-criticality systems, which by definition include critical components, this represents a major open problem: *how can we make complex schedulability analysis truly trustworthy?*

Given the community’s collective past record, just following the same procedure as before – primarily, manual “pen and paper” proofs and vetting to a varying degree of rigor by peer-reviewers – is arguably not going to work. Rather, a fundamentally more rigorous approach is needed.

Motivated by these observations, I argue that *schedulability analyses intended for use in safety-critical systems* should be *formally proven* with the help of a proof assistant such that all proofs are *machine-checked* to rule out human error. Following such an approach, the trust relies solely in the specification, and no longer in the much longer and much more intricate proofs (which no longer have to be trusted to be correct, as they can be automatically verified at the push of a button).

Given both that, historically, specification errors are much rarer than flaws in proofs, and that it is much easier to manually check a specification than it is to follow a proof in full detail, the adoption of machine-check-able proofs would represent a major advancement in the analysis and certification of real-time systems, and would enable unprecedented assurance in the temporal correctness of critical systems.

Towards this goal, I highlight one particular project – *PROSA*, a framework based on the Coq proof assistant – that is spearheading the drive towards a comprehensive foundation for formally verified schedulability analysis, and discuss and explain its major advantages as well as the remaining risks (such as specifications with contradicting hypotheses), and what is being done to mitigate them.

4.6 How to Gracefully Degrade

Alan Burns (University of York, GB)

License  Creative Commons BY 3.0 Unported license
© Alan Burns

Many approaches have been proposed for managing criticality-induced mode changes. A quick review is given, issues of integration are addressed as is the role that HI-crit tasks should take when there is a system overrun.

4.7 Resilient Mixed-Criticality Systems

Alan Burns (University of York, GB)

License  Creative Commons BY 3.0 Unported license
© Alan Burns

Certification authorities require correctness and resilience. In the temporal domain this requires a convincing argument that all deadlines will be met under error free conditions, and that when certain defined errors occur the behaviour of the system is still predictable and safe. This means that occasional execution-time overruns should be tolerated and where more severe errors occur levels of graceful degradation should be supported. With mixed-criticality systems, fault tolerance must be criticality aware, i.e. some tasks should degrade less than others. In this talk resilience is defined, and ways in which all levels of criticality can contribute to resilience are outlined. Discussions following this talk lead to a paper being produced that was offered for publication to the 2017 IEEE Real-Time Systems Symposium.

4.8 Reliability Optimization in MC2 Systems

Thidapat Chantem (Virginia Polytechnic Institute – Arlington, US)

License  Creative Commons BY 3.0 Unported license
© Thidapat Chantem

Reliability is an important consideration for many safety- and mission-critical systems. Broadly, reliability is influenced in part by soft (transient) errors and in part by permanent device or component failures. In addition, system reliability cannot typically be improved by independently minimizing the occurrence of soft and hard errors. This is because preventing the occurrence of a soft error by increasing the voltage, for instance, may inadvertently reduce component lifetimes due to the potentially high temperature. Either a soft or

hard error could cause deadline misses, or worse. With the increasing prevalence in mixed-criticality systems due to the size, weight, and power constraints, providing predictable and reliable performance in hard real-time systems becomes more important than ever. Since task assignment and scheduling is the main influencer of voltage and frequency assignment, a system-level, reliability-aware task assignment and scheduling framework is needed. I am interested in designing an adaptive fault tolerance framework that is able to (probabilistically) guarantee the schedulability of high-criticality tasks in face of soft errors and component failures.

4.9 Worst Case Execution Time measurement-based approach

Liliana Cucu-Grosjean (INRIA – Paris, FR), Adriana Gogonel (INRIA – Paris, FR), and Cristian Maxim (Airbus S.A.S. – Toulouse, FR)

License © Creative Commons BY 3.0 Unported license
© Liliana Cucu-Grosjean, Adriana Gogonel, and Cristian Maxim

The time behaviour of Cyber-Physical Systems relies on the execution time of programs representing the cyber parts of such systems. Our time estimation method is based on a measurement-based one providing results in absence of sufficiently large intervals of simulation while using the Extreme Value Theory (EVT). According to EVT if the maximum of execution times of a program converges, then this maximum of the execution times $C_i, \forall i \geq$ will converge to one of the three possible curves Frechet, Weibull, and Gumbel corresponding to a shape parameter $\xi < 0, \xi > 0$ and $\xi = 0$, respectively.

- Block size estimation. We compare all GEV curves obtained by varying the block size from 4 to $\frac{n}{4}$ where n is the cardinal of the set of execution times. We keep the block size corresponding to the shape parameter closest to 0, which corresponds to a Gumbel. We calculate the generalized EV curve corresponding to this parameter.
- Threshold level estimation. We compare all GPD curves obtained by varying the threshold levels u from 0% to 100%. We keep the threshold level u_0 such that the curve defined by $E(X - u_0) \approx u - u_0$ experiences linearity. The linearity of E indicates that the GPD curve goes close to a Gumbel. We calculate the generalized EV curve corresponding to u_0 .
- Comparing GEV and GPD pWCET estimates. The comparison of the GEV and GPD curves is done using the distance between the two distributions defined as $CRPS(GEV, GPD) = \sum_{z=x_{min}}^{z=x_{max}} [f_{GEV}(z) - f_{GPD}(z)]^2$. We consider in our experiments GEV and GPD as sufficiently close when $CRPS(GEV, GPD) \leq e$ with $e \approx 10^{-12}$. Other possible values of e , based for instance on the criticality level the pWCET estimation, may be decided. In order to decrease the error introduced by such estimation, we recommend calculating the pWCET estimate as a combination of GEV and GPD results. A joint pWCET estimate is obtained by choosing for each probability the largest value between GEV and GPD.

4.10 Practical Mixed-Criticality Model: Challenges

Arvind Easwaran (Nanyang TU – Singapore, SG)

License © Creative Commons BY 3.0 Unported license
© Arvind Easwaran

Joint work of Arvind Easwaran, Vijayakumar Sundar, Bibin Nair

Main reference A. Easwaran, V. Sundar, B. Nair, “Mixed Criticality Scheduling Research in Automotive: Making Research More Practical”, Workshop on Collaboration of Academia and Industry for Real World Embedded Systems (CAIRES) 2016.

URL <https://caires2016.inria.fr/>

The current trend in the automotive industry is focused towards Electronic Control Unit (ECU) consolidation. Increasing the number of ECUs to satisfy increased demand in the safety and comfort features of the vehicle is not a sustainable solution. Mixed criticality scheduling can be one of the key factors to drive ECU consolidation. Academic research has been focusing on different scheduling techniques and mode change protocols for mixed criticality systems. The research is further motivated with the introduction of ISO26262 which is a functional safety standard for safety critical applications in automotive. Functional safety of the automotive applications is represented in terms of Automotive Safety Integrity Levels (ASIL). Although the terms ‘ASIL’ of ISO26262 and ‘criticality’ of the existing research work looks similar, there is no clarity in the exact relationship between them. This can be attributed to the factors involved in determining the ASILs. There is also a need to validate the assumptions made in the existing research work with the safety critical behaviour of the applications. Certain assumptions may not even reflect the actual behaviour of automotive applications. This talk focuses on such issues for building mixed criticality systems. Solutions can be arrived at by considering various factors that might be missing in the current mixed criticality models considered by the research community.

4.11 Runtime Verification, Runtime Enforcement, and Mixed Criticality System Design

Sébastien Faucou (University of Nantes, FR)

License © Creative Commons BY 3.0 Unported license
© Sébastien Faucou

Mixed criticality is a way to think about uncertainty of parameters during the design of a critical system. A mixed criticality system must be capable of graceful degradation in order to preserve its critical functions when something goes wrong, i.e. when a design assumption is violated at runtime. To do so, it must (i) detect the violation; and (ii) react to this violation to ensure the preservation of its critical activities.

Similar problems have been studied in the formal methods community. Problem (i) is akin to runtime verification. Problem (ii) is akin to runtime enforcement. The objective of the talk is to establish a parallel between these problems and mixed criticality system design, and draw attentions to some works developed in the formal method community that could provide rigorous techniques to build proved components for mixed criticality systems.

Runtime verification has already been used in the context of mixed criticality system design, with promising results. Still, further investigations are required to identify its benefits and limits. For runtime enforcement, it is still not clear if it is powerful and/or scalable enough to provide satisfying answer to our problem.

4.12 Energy efficiency in factories: Benefit of renewable energy, IoT and Automatic Demand Response

Laurent George (ESIEE – Champs sur Marne, FR)

License © Creative Commons BY 3.0 Unported license
© Laurent George

Demand Response is a new approach to smooth pics of energy consumption at the scale of a country. Demand response can be used by a legacy energy provider to prevent from buying energy at the highest price on spot markets. When receiving a demand response request, a factory is requested to decide whether it could stop (or reduce) its energy consumption and for how long. Re-scheduling the activity of equipments in a production line by using standby or shutdown modes can help reducing energy consumption in factories. This requires taking scheduling decisions concerning the production line on relatively short reaction times (few minutes) upon demand response request. The benefit of Demand Response for a factory is that it gets paid for not consuming by legacy energy provider.

4.13 Real-Time Mixed-Criticality Wormhole Networks

Leandro Soares Indrusiak (University of York, GB)

License © Creative Commons BY 3.0 Unported license
© Leandro Soares Indrusiak

Joint work of Leandro Soares Indrusiak, Alan Burns, James Harbin

Main reference L. S. Indrusiak, J. Harbin, A. Burns, “Average and Worst-Case Latency Improvements in Mixed-Criticality Wormhole Networks-on-Chip”, in Proc. of the 27th Euromicro Conference on Real-Time Systems (ECRTS 2015), pp. 47–56, IEEE, 2015.

URL <https://doi.org/10.1109/ECRTS.2015.12>

Main reference A. Burns, J. Harbin, L. S. Indrusiak, “A Wormhole NoC Protocol for Mixed Criticality Systems”, IEEE Real-Time Systems Symposium (RTSS 2014), pp. 184–195, IEEE, 2014.

URL <https://doi.org/10.1109/RTSS.2014.13>

Wormhole switching is a widely used network protocol due to small buffering requirements on each network router, which in turn results in low area and energy overheads. This is of key importance in multi-core and many-core processors based on Networks-on-Chip, as the area and energy share of the on-chip interconnect itself can reach up to 30% of the area and energy used by the whole processor. However, the nature of wormhole switching allows a single packet to simultaneously acquire multiple links as it traverses the network, which can make worst-case packet latencies hard to predict. This becomes particularly severe in large and highly congested networks, where complex interference patterns become the norm.

This talk focuses on the use of priority-preemptive wormhole networks, and the latest research on analytical methods aimed at predicting worst-case packet latency over such networks. Then, I’ll show how to extend the network and the respective analysis to provide different levels of guarantees to network packets of different criticality sharing the same network. By doing that, highly-critical packets will always be given sufficient service, even in situations of overload or degraded network capacity.

4.14 Fixed-Priority Scheduling without Any Adaptation in Mixed-Criticality Systems

Jian-Jia Chen

License © Creative Commons BY 3.0 Unported license
© Jian-Jia Chen

Main reference G. von der Bruggen, K.-H. Chen, W.-H. Huang, J.-J. Chen, “Systems with Dynamic Real-Time Guarantees in Uncertain and Faulty Execution Environments”, IEEE Real-Time Systems Symposium (RTSS 2016), pp. 303–314, IEEE, 2016.

URL <http://dx.doi.org/10.1109/RTSS.2016.037>

In many practical real-time systems, the physical environment and the system platform can impose uncertain execution behaviour to the system. For example, if transient faults are detected, the execution time of a task instance can be increased due to recovery operations. Such fault recovery routines make the system very vulnerable with respect to meeting hard real-time deadlines. In theory and in practical systems, this problem is often handled by aborting not so important tasks to guarantee the response time of the more important tasks. However, for most systems such faults occur rarely and the results of not so important tasks might still be useful, even if they are a bit late. This implicates to not abort these not so important tasks but keep them running even if faults occur, provided that the more important tasks still meet their hard real time properties. In this paper, we present Systems with Dynamic Real-Time Guarantees to model this behaviour and determine in [1] if the system can provide full timing guarantees or limited timing guarantees without any online adaptation after a fault occurred. We present a schedulability test, provide an algorithm for optimal priority assignment, determine the maximum interval length until the system will again provide full timing guarantees and explain how we can monitor the system state online. The approaches presented in [1] can be applied to mixed criticality systems with dual criticality levels.

References

- 1 Georg von der Bruggen, Kuan-Hsun Chen, Wen-Hung Huang, Jian-Jia Chen: Systems with Dynamic Real-Time Guarantees in Uncertain and Faulty Execution Environments. RTSS 2016: 303-314

4.15 Improve the scalability of mixed-criticality parallel real-time systems

Jing Li (Washington University – St. Louis, US)

License © Creative Commons BY 3.0 Unported license
© Jing Li

Recent years have witnessed the convergence of two important trends in real-time systems: growing computational demand of applications and the adoption of processors with more cores. As real-time applications now need to exploit internal parallelism to meet their real-time requirements, they face a new challenge of scaling up computations on a large number of cores.

Randomized work stealing has been adopted as a highly scalable scheduling approach for general-purpose computing for parallel programs. In randomized work stealing, each core steals work from a randomly chosen core in a randomized and decentralized manner. Randomized work stealing has been proved to have a high-probability bound on the execution

time of a parallel job on multiple cores. In other words, it can complete the execution of a parallel job by the job's deadline with a high probability, when given sufficient number of cores. However, in the worst case (with very low probability), it could still run a parallel job sequentially. Therefore, if a parallel task is executed by randomized work stealing, the variation in the parallel execution times of the task depends not only upon the fluctuation in its computational demand, but also upon the randomness of its internal scheduling by work stealing.

The mixed-criticality real-time model typically captures the uncertainty of task executions to improve the average resource efficiency while providing hard guarantees in the worst case. Therefore, to improve the scalability of parallel real-time tasks, we could consider exploiting the work stealing strategy and model its variation in task execution times under the mixed-criticality framework. To do so, we need to consider the following problems.

- First, consider the simple scenario where a parallel task always releases jobs with the same parallel structure and computation. How can we modify the work stealing strategy to provide different (parallel) execution time estimates, so that it fits the Vestal model for mixed-criticality systems? In particular, the modified work stealing strategy must provide a worst-case progress guarantee that is better than executing a parallel task sequentially.
- Based on the modified work stealing strategy, can we design a mixed-criticality scheduling algorithm for parallel tasks executed by work stealing? Can we design a global mixed-criticality scheduling algorithm, so that low-criticality tasks can use the under-utilized cores by the high-criticality tasks while providing progress guarantee to high-criticality tasks?
- Finally, we need to consider the more general case where a parallel task releases jobs with different parallel structures and different computational demands and these parallel jobs are executed by work stealing. How can we simultaneously model the fluctuation in task's computational demand and the randomness of its internal scheduling by work stealing to improve the resource efficiency of the system?

4.16 Resource management in DREAMS

Claire Pagetti (ONERA – Toulouse, FR)

License © Creative Commons BY 3.0 Unported license
© Claire Pagetti

Joint work of Guy Durrieu, Gerhard Fohler, Gautam Gala, Sylvain Girbal, Daniel Gracia Pérez, Eric Noulard, Claire Pagetti

Main reference G. Durrieu, G. Fohler, G. Gala, S. Girbal, D. Gracia Pérez, E. Noulard, C. Pagetti, S. Pérez, , “DREAMS about reconfiguration and adaptation in avionics”, in Proc. of the 8th Conf. on Embedded Real Time Software and Systems (ERTS'16).

URL <https://hal.archives-ouvertes.fr/hal-01258701>

URL <https://www.youtube.com/watch?v=Wf6gx1S8c0w&feature=youtu.be>

The DREAMS (Distributed REal-Time Architecture for Mixed Criticality Systems) FP7 project addresses the design of a cross-domain architecture for executing applications of different criticality levels in networked multi-core embedded systems. A DREAMS architecture is composed of several multi-code chips (such as the ST Micro Spidergon NOC or the Freescale T4240) connected through a TTEthernet network.

This presentation focuses on the adaptation strategies and their implementation in the avionic demonstrator. Adaptations only take place upon failures on a core, with the purpose to bring the system back to a functioning state. We consider two types of failures:

1. A permanent core failure. Intensive integration of small devices on chip increases the permanent failures occurrence due to various phenomena such as aging, wear-out or infant mortality.
2. A temporal overload situation, resulting in deadline miss without corrective action.

We will describe the resource management proposed in the DREAMS middle-ware. We will then detail the adaptation strategies defined for mitigating the above-defined failures. Finally we will give the main ideas of the implementation for the avionic demonstrator.

4.17 Probabilistic Analysis for Mixed Criticality Scheduling with SMC and AMC

Dorin Maxim (LORIA & INRIA – Nancy, FR), Liliana Cucu-Grosjean (INRIA – Paris, FR), Robert Davis (University of York, GB), and Arvind Easwaran (Nanyang TU – Singapore, SG)

License © Creative Commons BY 3.0 Unported license

© Dorin Maxim, Liliana Cucu-Grosjean, Robert Davis, and Arvind Easwaran

Main reference D. Maxim, R. Davis, L. Cucu-Grosjean, A. Easwaran, “Probabilistic Analysis for Mixed Criticality Scheduling with SMC and AMC”, in Proc. of the Workshop on Mixed Criticality Systems (WMC 2016), collocated with RTSS 2016.

URL <https://www-users.cs.york.ac.uk/~robdavis/papers/WMC2016pAMC.pdf>

This work introduces probabilistic analysis for fixed priority preemptive scheduling of mixed criticality systems on a uniprocessor using the Adaptive Mixed Criticality (AMC) and Static Mixed Criticality (SMC) schemes. We compare this analysis to the equivalent deterministic methods, highlighting the performance gains that can be obtained by utilising more detailed information about worst-case execution time estimates described in terms of probability distributions.

4.18 Timing Compositionality – Challenges and Opportunities

Jan Reineke (Universität des Saarlandes, DE)

License © Creative Commons BY 3.0 Unported license

© Jan Reineke

Joint work of Sebastian Hahn and Michael Jacobs

Main reference S. Hahn, M. Jacobs, J. Reineke, “Enabling Compositionality for Multicore Timing Analysis”, in Proc. of the 24th Int’l Conf. on Real-Time Networks and Systems (RTNS’16), pp. 299–308, ACM, 2016.

URL <http://doi.acm.org/10.1145/2997465.2997471>

How does the execution time of a task respond to interference on shared resources like processor cores, caches, or buses? A common assumption is that a task’s response time increases by the amount of interference it experiences. We call this the “compositionality assumption”. It underlies most of the approaches to response-time analysis for multi-cores systems known today.

In recent work, we have shown that this assumption is both unsound and imprecise even for simple microarchitectures. I would like to discuss, how to overcome this problem to enable sound and more precise multi-core timing analysis.

5 Working Group Discussions

5.1 The Meaning and Use of probabilistic Worst-Case Execution Time (pWCET) Distributions

Robert I. Davis and Alan Burns

License  Creative Commons BY 3.0 Unported license
© Robert I. Davis and Alan Burns

Research into probabilistic Worst-Case Execution Time (pWCET) analysis can be classified into two main categories:

- *Analytical methods*: referred to as *Static Probabilistic Timing Analysis (SPTA)* [4, 7, 2, 1, 12]. SPTA is applicable when some part of the system or its environment contributes random or probabilistic timing behaviour. SPTA methods analyse the software and use a model of the hardware behaviour to derive an estimate of worst-case timing behaviour represented by a pWCET distribution, that is valid for any possible inputs and paths through the code. SPTA does not execute the code on the actual hardware.
- *Statistical methods*: referred to as *Measurement-Based Probabilistic Timing Analysis (MBPTA)* [3, 10, 11, 6, 15, 13]. MBPTA makes use of measurements (*observations*) of the overall execution time of a software component, obtained by running it on the actual hardware, using test vectors i.e. inputs that exercise a relevant subset of the possible paths through the code. These methods use a statistical analysis of the observations based on *Extreme Value Theory (EVT)* to estimate the pWCET distribution.

It is important to understand the precise meaning of a pWCET distribution since this impacts how such information can be used. In fact there are two subtly different meanings originating from SPTA and MBPTA.

The timing behaviour of a system may be characterised as deterministic or it may depend on some element that can be characterised by a random variable, for example a random replacement cache. In general, uncertainty about the timing behaviour of a system can be classified into two categories:

- *Aleatoric variability* depends on chance or random behaviour within the system itself or its environment.
- *Epistemic uncertainty* is due to things that could in principle be known about the system or its environment, but in practice are not, because the information is hidden or cannot be measured or modelled.

While complex software running on advanced time-predictable hardware may in theory exhibit deterministic timing behaviour and therefore have a single absolute WCET, in practice this actual WCET often cannot be determined and must therefore be estimated. Such an estimate is subject to epistemic uncertainty. In contrast, software running on simple time-randomised hardware exhibits aleatoric variability in its execution time. SPTA can be used to model aleatoric variability, but must deal with any epistemic uncertainty by upper bounding its effects in the model used. MBPTA can be used with systems that are characterised by either or both aleatoric variability and epistemic uncertainty.

As an example, it is instructive to consider a thought experiment involving two hypothetical systems. Both systems have 10 inputs which can take values in the range 1-6.

- **System A**: has two paths through the code. The first path is taken if the sum of the input values is odd, and takes 40 cycles to execute. The second path is taken if the sum of the input values is even, it has 10 instructions, each of which takes a random amount

of time from 1-6 cycles to execute (independent of any other instruction or input value). Thus the overall execution time of this path resembles the total from rolling 10 fair dice.

- **System B:** has a single path, it uses a huge internal 10-dimensional array (with 6^{10} entries) that maps from the values of the 10 inputs to a delay. The values for the delays are the totals for each possible permutation of 10 dice rolls; however, they are randomly arranged in the array, and we do not necessarily know what that arrangement is. Further, half of the values have been set to 40 cycles; again, we do not necessarily know which ones. This system looks up its execution time from the table, using the input values, and executes in total for that amount of time.

Intrinsically, System A has only aleatoric uncertainty, while System B has only epistemic uncertainty.

Consider applying SPTA to System A. With an accurate model of the instruction timing behaviour, SPTA could be used to compute a pWCET distribution that upper bounds the timing behaviour of this system *irrespective* of its inputs.

In the context of SPTA, the meaning of a pWCET distribution can be defined as follows, building on the definition in [7]:

- **Definition 1.** The pWCET distribution from SPTA is a tight upper bound² on all of the probabilistic execution time (pET) distributions that could be obtained for each individual combination of inputs, software states, and hardware states, excluding the random variables which give rise to variation in the timing behaviour. (Note, each individual pET distribution depends on the random variables, but not on the inputs or states, which are fixed in a particular combination).

In the absence of any random variables contributing to probabilistic timing behaviour, then the above definition of a pWCET distribution reduces to the familiar one for a single valued WCET obtained via conventional static WCET analysis. It is a tight upper bound on all the execution times that may be obtained for different combinations of inputs, software states, and hardware states.

If the random variables contributing to a probabilistic execution time behaviour are independent, then it follows that the pWCET distribution obtained by SPTA is independent with respect to any particular execution of that component. (This is the case, since the pWCET distribution from SPTA upper bounds every possible pET distribution). This has implications for the use of pWCET distributions, since they are independent they may be composed using basic convolution to derive probabilistic Worst-Case Response Time (pWCRT) distributions [8, 14], which can then be compared to the appropriate deadline to determine the probability of a deadline miss.

Next, consider System B. Applying SPTA using a precise model of the software and hardware would result in a single WCET, since there are no random variables involved, and we assume no information about the frequency of any combination of input values. By contrast, if we apply MBPTA, then we can estimate the WCET; however, this estimate has *epistemic* uncertainty. There are things we do not know about the system when we consider it as a “black box”, and we have only taken a sample of execution time observations, hence we cannot be 100% confident that our estimate is correct.

In the context of MBPTA, the meaning of a pWCET distribution can be defined as follows:

² In the sense of the greater than or equal to operator defined on the 1 - CDF of the distributions [9].

► **Definition 2.** The pWCET distribution from MBPTA is a statistical estimate giving an upper bound p on the probability that the execution time of a component will be greater than some arbitrary value x , valid for any possible distribution of input values that could occur during deployment.

Thus the pWCET distribution characterises the probability $(1 - p)$ that the WCET of a component will be no greater than some arbitrary value x [5], or as noted by Edgar and Burns [10] the pWCET distribution reflects the *confidence* we have that the statement, “*the WCET does not exceed x for some threshold x* ” is true.

We note that the definitions of a pWCET distribution originating from MBPTA and by SPTA are different. The definition from SPTA reflects aleatoric variability, while that from MBPTA reflects epistemic uncertainty.

Since the pWCET definition from MBPTA reflects epistemic uncertainty, i.e. what isn’t known about the system, then if it turns out that a WCET estimate x is exceeded, it is possible that it could be exceeded for *every* one of a number of runs of the component in a sequence, depending on the input values used. This is the case since the pWCET distribution effectively gives the probability that *at least one* run of the component has an execution time which exceeds x , but given that event, it provides no additional information about the execution times of individual runs.

For example, for System B, let us assume that MBPTA [6] estimates that there is a probability of 10^{-y} that the WCET exceeds x . However, if that WCET estimate is exceeded, then it could be that it is exceeded *every* time the component runs, depending on the particular input values used. This has implications for how the pWCET distribution may be used in probabilistic schedulability analysis. Assuming a pWCET distribution derived via MBPTA where a WCET of x has an exceedance probability of 10^{-y} . We may only infer that N runs of the component have a probability of no more than 10^{-y} of exceeding a total execution time of Nx . Contrast this with a similar pWCET distribution derived via SPTA. In this case, assuming the aleatoric variability was due to independent random variables, then it would be valid to apply basic convolution to upper bound the overall execution time of N runs. This conclusion would not in general be sound with a pWCET distribution derived via MBPTA, due to its different meaning.

In the case of System A, the pWCET distribution from SPTA tells us that the probability that the execution time on any single run will exceed x is 10^{-y} . If we observe a value larger than x at some point in a large number of runs, then that is not in itself incompatible with the information that we have, which characterises aleatoric variability. By contrast, in the case of system B, the pWCET distribution from MBPTA gives us a *measure of confidence* that the WCET is no more than x . If we observe a value larger than x then that confidence falls to zero.

Acknowledgments. The ideas in this short paper were presented and discussed in an ad-hoc working group comprising Liliana Cucu-Grosjean, Adriana Gogonel, Cristian Maxim, Iain Bate, Philipa Conway, Zoe Stephenson, Alan Burns and Robert Davis.

References

- 1 S. Altmeyer, L. Cucu-Grosjean, and R. I. Davis. Static probabilistic timing analysis for real-time systems using random replacement caches. *Springer Real-Time Systems*, 51(1):77–123, 2015.
- 2 S. Altmeyer and R. I. Davis. On the correctness, optimality and precision of static probabilistic timing analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, pages 26:1–26:6, 2014.

- 3 A. Burns and S. Edgar. Predicting computation time for advanced processor architectures. In *Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS)*, pages 89–96, 2000.
- 4 F. J. Cazorla, E. Quiñones, T. Vardanega, L. Cucu, B. Triquet, G. Bernat, E. Berger, J. Abella, F. Wartel, M. Houston, L. Santinelli, L. Kosmidis, C. Lo, and D. Maxim. Proartis: Probabilistically analyzable real-time systems. *ACM Transactions on Embedded Computing Systems*, 12(2s):94:1–94:26, May 2013.
- 5 L. Cucu-Grosjean. Independence a misunderstood property of and for probabilistic real-time systems. In *Real-Time Systems: the past, the present and the future*, pages 29–37, 2013.
- 6 L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quiñones, and F. J. Cazorla. Measurement-based probabilistic timing analysis for multi-path programs. In *Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS)*, pages 91–101, July 2012.
- 7 R. I. Davis, L. Santinelli, S. Altmeyer, C. Maiza, and L. Cucu-Grosjean. Analysis of probabilistic cache related pre-emption delays. In *Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS)*, pages 168–179, July 2013.
- 8 J. L. Diaz, D. F. Garcia, K. Kim, C-G. Lee, L. Lo Bello, J. M. Lopez, S. L. Min, and O. Mirabella. Stochastic analysis of periodic real-time systems. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, pages 289–300, 2002.
- 9 J. L. Diaz, J. M. Lopez, M. Garcia, A. M. Campos, Kanghee Kim, and L. L. Bello. Pessimism in the stochastic analysis of real-time systems: concept and applications. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, pages 197–207, Dec 2004.
- 10 S. Edgar and A. Burns. Statistical analysis of wcet for scheduling. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, pages 215–224, Dec 2001.
- 11 J. Hansen, S. A. Hissam, and G. A. Moreno. Statistical-based WCET estimation and validation . In *Proceedings of the Workshop on Worst-Case Execution Time Analysis (WCET)*, volume 252, 2009.
- 12 B. Lesage, D. Griffin, S. Altmeyer, and R. I. Davis. Static probabilistic timing analysis for multi-path programs. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, pages 361–372, Dec 2015.
- 13 G. Lima, D. Dias, and E. Barros. Extreme value theory for estimating task execution time bounds: A careful look. In *Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS)*, July 2016.
- 14 D. Maxim and L. Cucu-Grosjean. Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, pages 224–235, Dec 2013.
- 15 L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart. On the Sustainability of the Extreme Value Theory for WCET Estimation. In *Proceedings of the Workshop on Worst-Case Execution Time Analysis (WCET)*, pages 21–30, 2014.

6 Open problems

6.1 Mixed criticality scheduling under resource uncertainty

Kunal Agrawal (Washington University - St. Louis, US)

License  Creative Commons BY 3.0 Unported license
© Kunal Agrawal

Most real time scheduling theory for mixed criticality systems deals with uncertainty about task parameters and assumes that resources remain the same as the system executes. For cloud and shared environments, however, one can imagine that resource availability changes as the system executes. I would like to explore if we can use the ideas developed in mixed criticality scheduling to provide tiered guarantees of the following form: If “adequate” resources are available, the scheduler must schedule all tasks. If fewer resources are available, the scheduler is allowed to drop “low-criticality” tasks, but must still schedule the important tasks.

6.2 Deriving Optimal Scheduling Policies for MC Task Systems

Sathish Gopalakrishnan (University of British Columbia - Vancouver, CA)

License  Creative Commons BY 3.0 Unported license
© Sathish Gopalakrishnan

Assuming execution time distributions are available for tasks in a mixed-criticality setting, how do we use this distributional information to schedule tasks? One approach, where tasks – depending on their criticality levels – have an acceptable failure rate is to model the entire task system using chance-constrained Markov decision processes. This model can then be used to derive a feasible scheduling policy (when one exists). I will briefly describe some progress made and challenges that remain.

6.3 Guaranteeing some service upon mode switch in mixed-criticality systems

Zhishan Guo (University of Missouri - Rolla, US)

License  Creative Commons BY 3.0 Unported license
© Zhishan Guo

6.3.1 Introduction

Epistemic uncertainty widely exists in real-time systems that the precise nature of the external environment, as well as the run-time behavior of the implemented platform, cannot be predicted with complete certainty prior to deployment. However, systems nevertheless must be designed and analyzed prior to deployment in the presence such uncertainty – the widely-studied [3] Vestal model [12] for mixed-criticality workloads addresses uncertainties in estimating the worst-case execution time (WCET) of real-time code. Different estimations, at different levels of assurance, are made about these WCET values; it is required that all functionalities execute correctly if the less conservative assumptions hold, while only the more critical functionalities are required to execute correctly in the (presumably less likely) event that the less conservative assumptions fail to hold but the more conservative assumptions do.

Here we briefly introduce some generalizations of the Vestal model, where degraded (but non-zero) level of services can be guaranteed for the less critical functionalities even in the event of only the more conservative assumptions holding. If such service degradation is represented by a shorter allowed execution for each job, or a longer period, recent work has suggested some MC scheduling algorithms; while for other degradation definition, we seek for further discussions perhaps with the industry.

6.3.2 Low Critical \neq Non Critical

The original Vestal model was very successful in dealing with the resource inefficiency with the verification of mixed-criticality systems. However, this model has met with some criticism from systems engineers; e.g., in the event of some (Hi criticality) jobs executing beyond their less pessimistic WCET estimates, LO-criticality jobs are treated same as non-critical jobs that no guarantees can be made to their service.

This desideratum was addressed in [1] by introducing an additional less pessimistic WCET parameter for LO-criticality jobs – a guaranteed service level regardless of the behaviors/executions of HI-criticality jobs. Following the MC-Fluid framework [10] that was shown to have the best possible speedup factor (4/3) [5] versus clairvoyant optimal scheduler, we have identified in [4] a nice scheduler that handles such LO-criticality service separately. MC-Fluid framework assumes fluid scheduling which may involve too many preemptions.

The authors in [10] have suggested to follow the DP-Fair framework [6], while we believe the number of preemptions can be hugely reduced if we follow Boundary Fair [13] with well defined per-mode boundary setting at task release – see our recent submission [7] for more details. EDF based methods maybe another option – some recent work has studied the uniprocessor scheduling case [11].

The aforementioned schedulers may deal with a degraded utilization requirement for LO-criticality tasks upon a mode switch. However, a shorter execution or a longer period may not be enough (or proper) to guarantee certain level of service – a piece of code may need the original estimated execution length to finish any single execution, while the timeliness remains the same (i.e., the result is useful only when a job is finished within the same deadline conditions). A degraded service may be defined as the allowance of certain portion of jobs to be dropped, while others remain the same execution time and deadline. This leads to the (m,k)-firm deadline scheduling problem, on which there is no existing solution for mixed-criticality system schedulability analysis, and may worth investigating – see our recent submission [8] for more details.

References

- 1 A. Burns and S. Baruah. Towards a more practical model for mixed criticality systems. In WMC2014.
- 2 S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie. The preemptive uniprocessor scheduling of MC implicit-deadline sporadic task systems. ECRTS 2012.
- 3 A. Burns and R. Davis. Mixed-criticality systems: A review. <http://www-users.cs.york.ac.uk/burns/review.pdf>.
- 4 S. Baruah, A. Burns, and Z. Guo. Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors. ECRTS 2016.
- 5 S. Baruah, A. Easwaran, and Z. Guo. MC-Fluid: simplified and optimally quantified. RTSS 2015.
- 6 S. Funk, G. Levin, C. Sadowski, I. Pye, and S. Brandt. DP-Fair: a unifying theory for optimal hard real-time multiprocessor scheduling. Real-Time Systems, 2011.
- 7 Z. Guo, S. Sruti, and N. Guan. From fluid into non-fluid: multi-processor mixed-criticality scheduling with limited preemption. In submission.
- 8 Z. Guo, K. Yang, S. Arefin, S. Vaidhun, and H. Xiong. Uniprocessor Mixed-Criticality Scheduling with Graceful Degradation. In submission.
- 9 M. Hamdaoui and P. Ramanathan. A service policy for real-time customers with (m,k) firm deadlines. FTCS 1994.
- 10 J. Lee, K.-M. Phan, X. Gu, J. Lee, A. Easwaran, I. Shin, and I. Lee. MC-Fluid: Fluid model-based mixed-criticality scheduling on multiprocessors. RTSS 2014.

- 11 D. Liu, J. Spasic, N. Guan, G. Chen, S. Liu, T. Stefanov, and W. Yi. EDF-VD Scheduling of Mixed-Criticality Systems with Degraded Quality Guarantees. RTSS 2016.
- 12 S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. IEEE RTSS 2007.
- 13 D. Zhu, X. Qia, D. Mossél, and R. Melhem. An optimal boundary fair scheduling algorithm for multiprocessor real-time systems. Journal of Parallel and Distributed Computing, vol. 71, no. 10, pp. 1411–1425, 2011.

6.4 Regarding the Optimality of Speedup Bounds of Mixed-Criticality Schedulability Tests

Zhishan Guo (University of Missouri - Rolla, US)

License © Creative Commons BY 3.0 Unported license
© Zhishan Guo

Much existing research on Mixed-Criticality (MC) scheduling (see [3] for a review) has focused on dealing with the Vestal model [12], where different WCET estimations of a single piece of code are provided. This is typically a consequence of different tools for determining worst-case execution time (WCET) bounds being more or less conservative than each other. It is known [1] that mixed criticality (MC) scheduling under such model is highly intractable, such that polynomial-time optimal solution is impossible unless $P = NP$. As a result, *speedup bound* is widely used in MC scheduling for measuring how close to optimal is a given schedulability analysis.

- A schedulability test \mathcal{A} has *speedup factor* of $s (s \geq 1)$, if any task set that is schedulable by any algorithm on a given platform with processing speed of 1, it will be deemed schedulable by Test \mathcal{A} upon a platform that is s times as fast.

Of course, when deriving MC schedulers and associated schedulability tests, one of the goals is to identify/prove a relative small speedup bound (that is closer to 1). A minimum possible speedup is often presented as the “*optimal speedup bound*” of a given MC scheduling problem. However, we would like to point out that:

- Optimality of scheduler should not be derived against optimal speedup bounds.

6.4.1 Non-Optimal Schedulers with Optimal Speedup Bounds

For scheduling (dual-criticality) Vestal job set on a uniprocessor platform, it has been shown [2] that OCBP algorithm (following the idea of Audsley’s priority assignment mechanism) has an optimal speedup bound of $(\sqrt{5} - 1)/2$. However, several algorithms has been identified to *strictly dominate* OCBP; e.g., Lazy Priority Adjustment [8], LE-EDF [10] [9] – with the same speedup bound and at *all* time, better schedulability.

Similar results can be observed when we consider the scheduling of Vestal task set as well. It has been shown that $4/3$ is the best speedup that any non-clairvoyant scheduler can achieve. Upon proposing a speedup-optimal uniprocessor scheduler named EDF-VD [2], improvements on the schedulability can still be made, e.g., [7] [6]. As for the multiprocessor case, it is proved [3] that both MC-Fluid [10] and MCF [3] achieve the optimal speedup of $4/3$. However, MCF is a simplified version of (and is dominated by) MC-Fluid. Moreover, improvements on schedulability can be further made to MC-Fluid [11].

6.4.2 Speedup over Non-Clairvoyance?

When deriving speedup bounds, in most of the existing works of the community, the proposed algorithm is compared with a *clairvoyant optimal scheduler*, and adapts the necessary conditions for MC schedulability. This may not be a very fair way of comparison since the penalty for unawareness

of the future is applied into the speedup bounds. Following the varying-speed MC model [5] [4], we have identified an on-line optimal³ scheduler in [9] that has a speedup factor significantly greater than 1 when comparing to an optimal clairvoyant algorithm. However, such a speedup factor only reflects the price one must pay for not knowing the future (or the difficulty of the scheduling problem itself) – it has nothing to do with the MC scheduler design anymore. In other words, most existing speedup bounds may only be capturing the gap between clairvoyance and non-clairvoyance.

Since MC schedulability analysis is for off-line verification of correctness of real-time systems, all possible scenarios should be taken into consideration (which is non-clairvoyance). We believe speedup results comparing to optimal non-clairvoyance schedule may be worth investigating for MC systems.

References

- 1 S. Baruah. Mixed criticality scheduling is highly intractable. <http://www.cs.unc.edu/~baruah/Submitted/02cxy.pdf>.
- 2 S. Baruah, H. Li, and L. Stougie. Towards the design of certifiable MC systems. IEEE RTAS 2010.
- 3 S. Baruah, A. Easwaran, and Z. Guo. MC-Fluid: simplified and optimally quantified. RTSS 2015.
- 4 S. Baruah and Z. Guo. Scheduling mixed-criticality implicit-deadline sporadic task systems upon a varying-speed processor. RTSS 2014. IEEE Computer Society Press.
- 5 S. Baruah and Z. Guo. Mixed-criticality scheduling upon varying-speed processors. IEEE RTSS 2013.
- 6 A. Easwaran. Demand-based MC scheduling of sporadic tasks on one processor. IEEE RTSS 2013.
- 7 P. Ekberg, W. Yi. Bounding and shaping the demand of generalized mixed-criticality sporadic task systems. Real-Time Systems, 50(1): 48-86, 2014.
- 8 C. Gu, N. Guan, Q. Deng, and W. Yi. Improving OCBP-based scheduling for MC sporadic task systems. RTCSA 2013.
- 9 Z. Guo and S. Baruah. The concurrent consideration of uncertainty in WCETs and processor speeds in mixed criticality systems. IEEE RTNS 2015.
- 10 S. Baruah and Z. Guo. Mixed-criticality scheduling upon varying-speed multiprocessors. Leibniz Transactions on Embedded Systems, 1(2): 3:1–3:19, 2014.
- 11 S. Ramanathan and A. Easwaran. MC-fluid: rate assignment strategies. WMC 2015.

6.5 MC-ADAPT: Adaptive Mixed Criticality Scheduling through Selective Task Dropping

Jaewoo Lee (University of Pennsylvania - Philadelphia, US)

License  Creative Commons BY 3.0 Unported license
© Jaewoo Lee

Mixed-criticality real-time scheduling aims to ensure deadline satisfaction of higher-criticality tasks, while achieving efficient resource utilization. To this end, many approaches have been proposed to execute more lower-criticality tasks without affecting the timeliness of higher-criticality tasks. Those previous approaches however have at least one of the two limitations; i) they penalize all

³ If an *on-line optimal* scheduling strategy fails to maintain correctness for a given MC instance I , no non-clairvoyant algorithm can ensure correctness for I (without making lucky guesses to the future).

lower-criticality tasks at once upon a certain situation, or ii) they make decision how to penalize lower-criticality tasks at design time. As a consequence, they under-utilize resources by imposing an excessive penalty on low-criticality tasks.

Unlike those existing studies, our approach aims to minimally penalize lower-criticality tasks by fully reflecting the dynamically changing system behavior into adaptive decision making. We propose a new scheduling algorithm which supports selective task dropping and develop its runtime schedulability analysis capturing the dynamic system state. Our proposed algorithm adaptively decides task dropping based on the runtime analysis.

To determine the quality of task dropping, we propose the speedup factor for task dropping. While the conventional speedup factor for the MC scheduling problem only evaluates MC scheduling algorithms in terms of the worst-case schedulability, we apply the speedup factor for the task dropping problem, which is an extended version of the MC scheduling problem. The task dropping problem is an optimization problem for task dropping under different MC scheduling scenarios.

6.6 Schedulability, Probabilities and Formal Methods

Luca Santinelli

License  Creative Commons BY 3.0 Unported license
© Luca Santinelli

The abstract is about developing probabilistic schedulability analysis with formal methods, in particular the Continuous Time Markov Chain models for jobs and tasks with continuous input distribution as probabilistic Worst-Case Execution Time (pWCET).

The open problem presented composes of building jobs and tasks CTMC models which are able to capture every [probabilistic] execution behavior; the models composed constitutes the real-time system with its jobs/tasks ordering. Then, such models can be formally verified with properties like deadline miss ration and systems schedulability. With the model proposed, alternative properties for already existing scheduling algorithms of newly proposed probabilistic schedulability algorithms would be verified.

6.7 Safety Calling

Zoë Stephenson (Rapita Systems Ltd. - York, GB)

License  Creative Commons BY 3.0 Unported license
© Zoë Stephenson

There is a need for vocabulary and models to support discussion between researchers in the domain of mixed-criticality scheduling and experts in the field of safety analysis. In this abstract, I explain why this is important and what form this could take.

In a traditional scheduling regime such as a preemptive, fixed-priority scheduler, we present an argument showing that the system is schedulable with respect to some assumptions. Those assumptions relate to jitter, timing anomalies (particularly regarding the cache), task switch latency and the validity of WCET figures. The scheduler typically provides overrun monitoring, and the system design would typically also include a watchdog, to detect cases where the schedulability prediction is incorrect. The safety engineer then designs responses to the detection of this condition – for example, rebooting a partition, switching to a diverse implementation with fewer features, switching to a reversionary schedule, or resetting the entire controller.

When we move to mixed-criticality scheduling, we get the following features:

- Scheduling algorithms that aim to provide better overall utilisation of resources, particularly multicore resources

- Flexibility in the predictions made at design time so that the system designer can bias confidence in execution time bounds towards higher-criticality functionality and take advantage of this biased confidence in the schedulability analysis
- Flexibility in the behaviour of the scheduler in responding to violations of these execution-time bounds

With this flexibility, there are new considerations in the system design and safety analysis, and clear communication will make it easy to work with these considerations. The aim is to help the safety engineer to devise appropriate mitigation strategies for overruns, dividing the work up between system requirements and scheduler requirements within a process such as the following:

1. System design identifies functions allocated to software. Each has an associated assurance level representing the severity of the consequence of a failure to provide the function.
2. Software design identifies components to implement functions; components also have assurance levels.
3. Software design creates a schedule for the components, and presents this to the safety analysis in the form of response time bounds, relative confidence (compared to the project's traditional approach) and possible responses for exceeding those time bounds. The aim is to describe the capabilities of the scheduler and show how confidence and responses can match up with the components' assurance levels.
4. Safety analysis feed the scheduler behaviour, as well as other functional and non-functional behaviours, into system safety assessment and determine whether additional measures are needed for assuring adequate provision of system functions. These become derived requirements that call for design changes such as resets, reversionary task-sets or watchdogs, and additional analyses.
5. Software design, system design and safety analysis iterate until appropriate assurance is reached.
6. Safety analysis completes the assurance argument that mitigation strategies for the software components are appropriate for the criticality levels of the functions they provide.

From these steps a picture emerges of a scheduler response model, structured as:

- Function and Component
 - WCRT
 - WCRT confidence
 - Overrun response
 - * Call an exception handler
 - * Cancel (this task / some other tasks)
 - * Reduce releases (this task / some other tasks)
 - * Increase clock frequency
 - * Reduce algorithm detail
 - * Switch schedule
 - * ...

The range of WCRT confidence descriptions and possible responses will depend on the exact mixed-criticality scheduler in use. By presenting the scheduling approach in terms of confidence and overrun responses, the model provides useful detail for the safety analyst and improves the ability of the system to take full advantage of the facilities provided by advanced scheduling algorithms.

7 Outcomes of the seminar

- *On the Meaning of probabilistic Worst-Case Execution Time (pWCET) Distributions and their use in Schedulability Analysis* by Robert I. Davis, Alan Burns, David Griffin. Under submission.
- *Resilient Mixed-Criticality Systems* by A. Burns, R.I. Davis, S. Baruah, I. Bate. Under submission.
- *On the Existence of a Cyclic Schedule for Non-Preemptive Periodic Tasks with Release Offset* by Mitra Nasri and Emmanuel Grolleau. Under submission.
- *Uniprocessor Mixed-Criticality Scheduling with Graceful Degradation* by Zhishan Guo, Kecheng Yang, Samsil Arefin, Sudharsan Vaidhun, and Haoyi Xiong. Under submission.
- *Sustainability in Mixed-Criticality Scheduling* by Zhishan Guo, Sai Sruti, Bryan Ward, and Sanjoy Baruah. Under submission.
- *Sustainability in Mixed-Criticality Scheduling* by Ying Zhang, Zhishan Guo, Lingxiang Wang, Haoyi Xiong, and Zhenkai Zhang. Under submission.

Participants

- Kunal Agrawal
Washington University –
St. Louis, US
- Sebastian Altmeyer
University of Amsterdam, NL
- James H. Anderson
University of North Carolina at
Chapel Hill, US
- Sanjoy K. Baruah
University of North Carolina at
Chapel Hill, US
- Iain Bate
University of York, GB
- Enrico Bini
University of Turin, IT
- Björn B. Brandenburg
MPI-SWS – Kaiserslautern, DE
- Alan Burns
University of York, GB
- Thidapat Chantem
Virginia Polytechnic Institute –
Arlington, US
- Jian-Jia Chen
TU Dortmund, DE
- Liliana Cucu-Grosjean
INRIA – Paris, FR
- Robert Davis
University of York, GB
- Arvind Easwaran
Nanyang TU – Singapore, SG
- Pontus Ekberg
Uppsala University, SE
- Sébastien Faucou
University of Nantes, FR
- Madeleine Faugère
Thales Research and Technology –
Palaiseau, FR
- Christian Ferdinand
AbsInt – Saarbrücken, DE
- Laurent George
ESIEE – Champs sur Marne, FR
- Adriana Gogonel
INRIA – Paris, FR
- Sathish Gopalakrishnan
University of British Columbia –
Vancouver, CA
- Emmanuel Grolleau
ENSMA – Chasseneuil, FR
- Zhishan Guo
University of Missouri –
Rolla, US
- Leandro Soares Indrusiak
University of York, GB
- Jaewoo Lee
University of Pennsylvania –
Philadelphia, US
- Jing Li
Washington University –
St. Louis, US
- Martina Maggio
Lund University, SE
- Alberto Marchetti-Spaccamela
Sapienza University of Rome, IT
- Cristian Maxim
Airbus S.A.S. – Toulouse, FR
- Dorin Maxim
LORIA & INRIA – Nancy, FR
- Mitra Nasri
MPI-SWS – Kaiserslautern, DE
- Claire Pagetti
ONERA – Toulouse, FR
- Kirk Pruhs
University of Pittsburgh, US
- Gurulingesh Ravari
ISEP Porto, PT
- Jan Reineke
Universität des Saarlandes, DE
- Stefan Resch
Thales – Wien, AT
- Philippa Ryan
Adelard – London, GB
- Luca Santinelli
ONERA – Toulouse, FR
- Zoë Stephenson
Rapita Systems Ltd. – York, GB
- Sascha Uhrig
Airbus – München, DE
- Wang Yi
Uppsala University, SE
- Dirk Ziegenbein
Robert Bosch GmbH –
Stuttgart, DE

