Real-Time Syst (2017) 53:669–672
DOI 10.1007/s11241-017-9288-1

# Guest editorial: special issue on mixed-criticality, multi-core, and micro-kernels

**Robert I. Davis**[1]

Published online: 16 August 2017

Today, hard real-time systems are found in many diverse application areas including; autonomous vehicles, avionics, space systems, and robotics. In these areas, technological progress is resulting in rapid increases in both software complexity and processing demands. To address these requirements, silicon vendors have shifted their focus from single-core to multicore hardware platforms.

Multicore hardware integrates multiple processing cores onto a single chip. To reduce costs and to improve performance in the average case, these cores typically share a number of hardware resources; including parts of the memory hierarchy such as the last level cache (LLC), main memory (DRAM), and the interconnect. By contending for these shared hardware resources, tasks executing on one core can potentially interfere with tasks executing on another core, increasing their worst-case execution times. Contention for shared resources thus poses a significant challenge in the development of real-time systems. In some cases the contention is so great that the analysable, guaranteed real-time performance may be no better than that for a single core.

With the adoption of multicore technology comes the opportunity to combine different applications on the same hardware platform, reducing size, weight and power consumption, as well as assembly and production costs for the overall system. Here, different applications may have different criticality levels, designating the level of assurance needed against failure. For example, in an avionics context, flight control and surveillance applications in unmanned aerial vehicles (UAVs) are of high and low criticality respectively. The problem of combining high and low criticality applications on the same hardware platform raises issues of how to reconcile the conflicting

✉ Robert I. Davis
rob.davis@york.ac.uk

1 Department of Computer Science, University of York, Deramore Lane, York YO10 5GH, UK

requirements of separation for assurance, and sharing for efficient use of resources. Further, with the advent of applications such as autonomous driving, the complexity and processing load of software tasks has increased to the point where individual tasks may need to be executed in parallel across multiple cores, if they are to meet their deadlines; adding a further layer of complexity.

The development of systems running high criticality applications requires the support of a high-assurance real-time operating system (RTOS) or micro-kernel. Not only must the functional behaviour of the RTOS be proven correct, but also its timing behaviour.

This special issue contains four papers at the forefront of real-time systems research into mixed-criticality scheduling, multicore systems, and high-assurance micro-kernels. Each of these papers appeared in preliminary form at the 22nd IEEE real-time and embedded technology and applications symposium (RTAS) 2016. These preliminary papers received *Outstanding Paper Awards* identifying them as research of the highest quality. Overall, 92 papers were submitted to RTAS 2016. After a detailed review process, involving 58 Program Committee members and four reviews per paper, 25 papers were selected to appear at the conference; an acceptance rate of 27%. The papers that appear in this special issue include significant and comprehensive additional contributions to what is already regarded as outstanding research.

The first paper is "*Addressing isolation challenges of non-blocking caches for multicore real-time systems*" by Prathap Kumar Valsan, Heechul Yun, and Farzad Farshchi. In its preliminary form, this paper received the *Best Paper Award* at RTAS 2016.

In multicore systems where the last level cache and memory (DRAM) are shared between cores, cache partitioning is often seen as an effective way of isolating tasks executing on one core from contention due to tasks executing on another core. This paper shows that for a number of COTS processors with out-of-order cores, and non-blocking caches, cache partitioning alone is not sufficient to achieve such isolation. Rather, there are a set of hardware registers called (cache) miss status holding registers (MSHR) which the tasks running on different cores can effectively contend for. In some cases, this leads to a slowdown of over 20 times when contenders are introduced, for a task running on a dedicated core and using a dedicated cache partition.

This paper provides a thorough investigation of the problem including measurement of the effects on a number of different multicore hardware platforms, and further investigation using a cycle-accurate simulator. A low cost hardware extension is proposed in conjunction with an operating system solution. This approach alleviates the problem of contention for MSHRs by providing operating system control of the memory level parallelism for each core.

The second paper is "*Attacking the one-out-of-m multicore problem by combining hardware management with mixed-criticality provisioning*" by Namhoon Kim, Bryan C. Ward, Micaiah Chisholm, Cheng-Yang Fu, James H. Anderson and F. Donelson Smith. In its preliminary form, this paper received the *Best Student Paper Award* at RTAS 2016.

The potential for significant contention over shared hardware resources by tasks executing on different cores of a multicore system can lead to substantial pessimism in the verification of timing constraints. In the worst-case, tasks executing on one core of an $m$-core system may be subject to so much contention that their execution is slowed by a factor of $m$ or more, compared to the baseline of running in isolation without contention. In terms of guaranteed real-time performance, the advantages of having $m$-cores can be entirely negated. This is referred to as the “one out of $m$” problem, since it would be more effective for the system to utilize only one core. This paper addresses the “one out of $m$” problem by combining allocation techniques that consider the criticality levels of application tasks with hardware mechanisms that can make execution on a multicore platform more predictable, for those criticality levels that need it. The mixed criticality framework employed enables DRAM banks and areas of last level cache to be allocated to groups of tasks according to their criticality level. Further, a linear programming method is proposed which is able to size the LLC areas in a way that benefits schedulability. The paper closes with an evaluation of the effectiveness of the framework based on a large-scale overhead-aware schedulability study.

The third paper is “*Mixed-criticality federated scheduling for parallel real-time tasks*” by Jing Li, David Ferry, Shaurya Ahuja, Kunal Agrawal, Christopher Gill, and Chenyang Lu.

High performance real-time applications, such as those that facilitate autonomous driving, have both high processing demands and tight deadlines. Such tasks may require the use of internal parallelism to meet their deadlines (i.e. a single task may need to execute in parallel on a number of cores). This paper addresses the problem of mixed criticality scheduling of parallel real-time tasks that are modelled using a directed acyclic graph (DAG). A mixed criticality federated scheduling (MCFS) algorithm is proposed, with capacity augmentation bounds proven for dual- and multi-criticality systems. An improved algorithm is also proposed, which retains the same capacity augmentation bounds, but is able to schedule many more task sets in practice. An MCFS runtime is implemented using Linux with the RT_PREEMPT patch as the underlying operating system and OpenMP to manage threads and assign workload. The MCFS runtime provides both graceful degradation and the ability to recover back to normal operation. An empirical evaluation shows the effectiveness of the approach in practice.

The fourth paper is “*High-assurance timing analysis for a high-assurance real-time operating system*” by Thomas Sewell, Felix Kam, and Gernot Heiser.

It is essential that the system software e.g., real-time operating system used to support high criticality applications is proven correct in terms of both its functional behaviour and its timing behaviour (i.e. worst-case execution time (WCET) bounds). This paper addresses the latter aspect for the high-assurance seL4 micro-kernel. WCET analysis is performed on the binary code; however, at that level, information about loop bounds and infeasible paths is not normally available. This information can be provided manually; however, that approach can be error-prone. Instead, this work proposes the use of a translation-validation (TV) framework, and an optimizing complier. The TV tool relates control flow at the source level to that at the binary level enabling the WCET analysis to make use of information present in the source that is missing in the binary. This provides a high-assurance means of automatically determining loop bounds and infeasible paths in the kernel. The technique is also shown to work with standard WCET benchmarks.

Together, these papers form an excellent cross-section of state-of-the-art real-time systems research into mixed-criticality scheduling, multicore systems, and micro-kernels. They will undoubtedly provide a catalyst for further exciting research in this field.

Rob Davis
Guest Editor