Guaranteeing some service upon mode switch in mixed-criticality systems

Zhishan Guo
Missouri University of Science and Technology
guozh@mst.edu

Keywords: mixed-criticality, speedup bounds, optimality, clairvoyance

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 platform upon which it is implemented, 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 (see [3] for a thorough review) Vestal model [11] 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.

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 [9] 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 [9] has suggested following DP-Fair framework [6], while we believe the number of preemptions can be hugely reduced if we follow Boundary Fair [12] with well defined per-mode boundary setting at task release — see our recent submission [7] for more details. Some recent work has studied the uni-processor scheduling case and proposed EDF based methods [10]

3 Is Degraded Utilization Good Enough?

The aforementioned schedulers may deal with a degraded utilization requirement for LO-criticality tasks upon a mode switch. However, 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 time period to finish its execution, while the timeliness remains the same (i.e., result is useful only when finishing within the same deadline conditions). A degraded service can be defined as the allowance of certain amount 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.

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. Available at http://web.mst.edu/ guozh/pubs/RTSS2017a.pdf

[8] M. Hamdaoui and P. Ramanathan. A service policy for real-time customers with (m,k) firm deadlines. FTCS 1994.

[9] 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.

[10] 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.

[11] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. IEEE RTSS 2007.

[12] D. Zhu, Xuan Qia, Daniel Mossél, and Rami Melhem. An optimal boundary fair scheduling algorithm for multiprocessor real-time systems. Journal of Parallel and Distributed Computing, vol. 71, no. 10, pp. 14111425, 2011.