The Journal of Systems and Software 124 (2017) 120–136

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

# A supercomputing framework for the evaluation of real-time analysis and optimization techniques

Juan M. Rivas, J. Javier Gutiérrez*, Michael González Harbour

*Software Engineering and Real-Time Group, University of Cantabria, Santander, Spain*

ARTICLE INFO

*Article history:*
Received 26 May 2016
Revised 4 November 2016
Accepted 5 November 2016
Available online 10 November 2016

*Keywords:*
Technique validation
Real-time systems
Distributed systems
Supercomputer exploitation

ABSTRACT

The evaluation of new approaches in the analysis and optimization of real-time systems usually relies on synthetic test systems. Therefore, the development of tools to create these test systems in an efficient way is highly desirable. It is usual for these evaluations to be constrained by the processing power of current personal computers. For example, in order to assess whether a specific technique generally performs better than others or whether the improvement observed is constrained to a limited set of circumstances, a vast set of examples must be tested, making the execution infeasible in a common PC. In this paper, we present a framework that defines the building blocks of a tool to enable the validation of real-time techniques, through the efficient execution of massive evaluations of event-driven synthetic distributed systems. Its main characteristic is that it can leverage the computing power of a supercomputer to perform large studies that otherwise could not be performed with a PC. The framework also defines different generation methods so that the generated systems can cover a wide range of characteristics that can be present in different application domains. As a case study, we also implement this framework based on a previously developed prototype.



## 1. Introduction

### 1.1. Motivation and contributions

The real-time systems field is an active area of research where new techniques or approaches are constantly being proposed. These contributions vary widely in their purpose. They can range from new scheduling policies that try to improve new metrics like power consumption, to less pessimistic schedulability analysis techniques or more advanced scheduling parameter optimization techniques that improve the utilization of the resources.

Since the seminal paper by Liu and Layland in 1973 (Liu and Layland, 1973), researchers in the real-time systems field have been constantly producing new techniques and approaches that have extended the scope of what a real-time system could achieve (Sha et al., 2004). For example, a particularly thriving area of research is the formulation of new schedulability analysis techniques that can determine whether a system will always meet its deadlines. These techniques can be applied for example to uniprocessor systems (Audsley et al., 1993; Bini and Buttazzo, 2004; Tindell, 1994) or distributed systems (Tindell and Clark, 1994; Palencia and González Harbour, 1998; Palencia and González Harbour, 1999; Mäki-Turja and Nolin, 2008). Another area of great interest is the optimization of the scheduling parameters for tasks in the processors and messages in the networks (Tindell et al., 1992; Gutiérrez and González Harbour, 1995; Kao and Garcia-Molina, 1997; Rivas et al., 2011; Azketa et al., 2011), i.e., the assignment of priorities for a fixed priority scheduling policy or scheduling deadlines for an EDF (Earliest Deadline First) policy.

With any new approach there is a need to study its performance compared head-to-head with previous similar techniques. These studies are usually carried out by testing these techniques over a synthetic pool of examples. A synthetic system is a theoretical representation of a real system, which is described using a particular syntax or language. Synthetic systems are usually generated in batches to cover different characteristics. Using pools of synthetic systems provides a series of advantages over testing on a real system implementation, namely:

- Testing on a real system only validates the new technique on a fixed architecture and/or software platform. For a more general verification, an extended pool of systems is needed. With synthetic systems, a wider spectrum of systems or circumstances can be covered at less cost.

* Corresponding author.
*E-mail addresses:* rivasjm@unican.es (J.M. Rivas), gutierjj@unican.es (J.J. Gutiérrez), mgh@unican.es (M. González Harbour).

http://dx.doi.org/10.1016/j.jss.2016.11.010
0164-1212/

- Hardware and software overheads can be included in the model of the system, for example in the worst and best case execution times specified in the synthetic system.
- Sometimes, the real implementation is not yet available or it is too expensive. Using a synthetic pool of examples enables a faster development of techniques, new modifications can be quickly tested and there is no need to wait for real implementations.
- In a real implementation it is not straightforward to induce the worst-case situation while this might be easier in a test environment.
- Many real implementations are covered by confidentiality clauses that prevent their open availability.

In opposition to benchmark systems, which are generally individual systems (real or synthetic) representing fixed architectures, synthetic systems are usually generated *ad-hoc* in each study. Since the studies using synthetic systems usually involve extensive pools of examples, the total computations times can require non-trivial amounts of time. For this type of studies, we think supercomputers can take an important role. The feasibility and benefits of using a supercomputer to evaluate real-time techniques have been tested in a previous work through a prototype tool called GEN4MAST (Rivas et al., 2014). This tool was built around the widely used MAST (Modeling and Analysis Suite for Real-Time Applications) tool (MAST 2016), which gathers together a set of analysis and optimization tools based on a model (González et al., 2001; González et al., 2013). This prototype tool written in Python has already been used with successful results in the performance study of new algorithms for the offset-based analysis of EDF distributed systems (Díaz-de-Cerio et al., 2014) and for the assignment of scheduling deadlines in these systems (Rivas et al., 2015). Those studies required large amounts of computation time that could be achieved reasonably quickly thanks to the distribution of the workload in a supercomputer.

Based on our experience with this prototype tool, in this paper we present a general framework to enable the testing of real-time techniques in an extensive pool of synthetic distributed systems using a supercomputer. Unlike the prototype tool, this framework is composed of a complete set of Java interfaces and abstract classes that, once implemented, defines the basic architecture of a tool to (1) massively generate synthetic distributed systems, (2) apply the real-time techniques (implemented in a real-time tool) on the generated systems and (3) store and process the results obtained. The usage of a supercomputer opens the door to exhaustive studies that better guarantee the validity of the results and that would not be feasible in a common PC. As far as we know, this framework, and the prototype tool GEN4MAST, are the first tools to use supercomputers for the evaluation of techniques for real-time systems.

The framework we present is designed to be as independent as possible of any real-time tool in particular, but at the same time, be able to generate systems that are representative of what is used in the field of real-time systems. For this purpose, the framework targets the generation of event-driven reactive software models like those defined in the MARTE (Modeling and Analysis of Real-time and Embedded systems) standard (Object Management Group 2009). To achieve our goal, the framework defines a synthetic distributed system model that represents a simplified view of the SAM (Schedulability Analysis Modeling) profile of MARTE. Any implementation of the framework is then responsible of transforming this synthetic system model into a particular meta-model used by a real-time tool, so the real-time techniques implemented in it can be applied. As a practical case study, in this paper we propose an implementation of the new framework to work inside the MAST environment, thus representing an evolution of the GEN4MAST prototype tool. Since MAST defines a model that is MARTE-like, the transformation from the synthetic system model of the framework to MAST is straightforward. Both the generic framework and the new version of GEN4MAST are available as open source (GEN4MAST 2016).

A transformation to other meta-models that also follow the MARTE standard (even loosely) can be achieved with reasonable ease too. For example, in (Rivas et al., 2016a) we show the transformation path from Thales' Tempo-Analysis model (Henia et al., 2013) to MAST and in (Rivas et al., 2016b) we show how to transform an Amalthea model (Amalthea 2016) used by Bosch to describe engine management systems, to MAST. These two works help illustrate the ideas that (1) MARTE-like system models are used in industry and (2) that the transformations between models that follow the MARTE standard are achievable with an almost one-to-one relationship. These two points help validate the decision of using MARTE/SAM as the basis of the synthetic system defined in the framework.

### 1.2. Paper organization

The remainder of this document is structured as follows. In Section 2 we provide an overview of related tools for the generation and analysis of systems. In Section 3 we describe the real-time system model that is generated by the framework, while Section 4 describes the implementation of the synthetic system model and the mechanism for applying a real-time tool on the generated systems. In Section 5 we discuss the framework and its main components. The following four sections deal with the components of the framework in more detail. Thus, Sections 6, 7 and 8 discuss the three main modules of the framework: the Dispatcher, Evaluation Engine and Results Manager modules respectively. In Section 9 we present a case study with the implementation of the framework for MAST. Section 10 presents an evaluation of the generation methods proposed in the framework, as well as the assessment of the benefits obtained when using the supercomputer on sample tests. Finally, Section 11 addresses the conclusions of the work and it also envisages possible extensions of the proposed framework.

## 2. Related work

This paper deals with the problem of generating systems and the application of real-time techniques on them, usually implemented in an analysis tool. In the literature we can find works that deal with each problem individually and also in conjunction.

### 2.1. Generation of systems

For the generation of synthetic systems, researchers have usually relied on *ad-hoc* tools, with functionality that tends to be limited to solving one particular problem each time. Some examples of these tools related to schedulability analysis techniques can be found in the literature. In Bini and Buttazzo (2004), simulations are performed to assess a schedulability test for uniprocessors by generating task sets with random periods and worst-case execution times (with uniform distribution). Extensive simulations with different task sets whose execution times and periods are generated randomly to evaluate an analysis technique for distributed systems with offsets or precedence relationships are proposed in Palencia and González Harbour (1998) and Palencia and González Harbour (1999) respectively. In order to evaluate a new analysis technique for distributed systems with offsets, a task generator is presented in Mäki-Turja and Nolin (2008), which takes into account parameters such as the system load (CPU utilization), the number of end-to-end flows or the number of tasks per end-to-end flow.