GEN4MAST: A Tool for the Evaluation of Real-Time Techniques Using a Supercomputer

Juan M. Rivas, J. Javier Gutiérrez and Michael González Harbour
Software Engineering and Real-Time Group
University of Cantabria
Santander, Spain
{rivasjm, gutierjj, mgh}@unican.es

***Abstract*—The constant development of new approaches in real-time systems makes it necessary to create tools or methods to perform their evaluations in an efficient way. It is not uncommon for these evaluations to be constrained by the processing power of current personal computers. Thus, it is still a challenging issue to know whether a specific technique could perform better than another one, or the improvement remains invariable in all circumstances. In this paper we present the GEN4MAST tool, which can take advantage of the performance of a supercomputer to execute longer evaluations that wouldn't be possible in a common computer. GEN4MAST is built around the widely used MAST tool, automating the whole process of distributed systems generation, execution of the requested analysis or optimization techniques, and the processing of the results. GEN4MAST integrates several generation methods to create realistic workloads. We show that the different methods can have a great impact on the results of distributed systems.**

Keywords—Techniques validation; Real-time systems; Distributed systems; Supercomputer exploitation.

I. INTRODUCTION

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 could be from new scheduling policies that try to improve new metrics like power consumption, to less pessimistic schedulability analysis techniques, or more advanced scheduling parameters optimization techniques that improve the utilization of the resources.

With any new approach there is the need to make a performance study where the new technique is compared head-to-head with previous approaches. These studies are usually carried out by testing these techniques over a synthetic pool of examples, which provides a series of advantages over testing in a real system implementation:

- It can cover a wider spectrum of systems or circumstances cheaply.
- 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 no waiting time for real implementations is needed.
- In a real implementation it is not straightforward to induce the worst-case situation while this might be easier in a test environment.

Several tools for the creation of synthetic studies of real-time systems have been proposed. FORTAS [1] can massively perform the generation and analysis of multiprocessor systems with independent tasks. It also handles the processing of the results generating graphics. STORM [2] is a similar tool that enables the user to define new scheduling policies, but is not able to generate systems massively. RTMultiSim [3] is a tool that can generate massively distributed systems, but supports a reduced number of generation methods. Another tool is YARTISS [4], which is a simulator with a modular architecture for the creation of multiprocessors systems.

Supporting some of the best features of the previous tools, and integrating new ones, in this paper we present GEN4MAST (GENerator for MAST), an open source tool built in Python that automates the whole process of synthetic distributed systems generation, the execution of the selected techniques, and the processing of the results. It is built around MAST [5][6], which is an extensive and tested tool that integrates several techniques to analyze and optimize distributed systems.

The main feature that sets GEN4MAST apart from the previous tools is that it can execute the analyses in a supercomputer. In previous tests it was observed that it could take up to several hours to analyze certain systems in a normal PC. Considering that an evaluation of techniques can typically be composed of several thousands of executions of the analysis tool, being able to distribute these executions in a supercomputer opens the door for the execution of extensive studies in reasonable amounts of time. Although a supercomputer is a very expensive computing resource, during the last years its availability has been increased in many research institutions due to the added value they provide.

Based in MAST, GEN4MAST also inherits all of its benefits, the most relevant to GEN4MAST being:

This work has been funded in part by the Spanish Government and FEDER funds under grant number TIN2011-28567-C03-02 (HI-PARTES).

- MAST provides a rich system model to describe distributed systems, that is based in the OMG MARTE [7] standard.
- MAST is open source, and provides a methodology to integrate new analysis and optimization techniques [8].

The paper is organized as follows. In Section II we provide an overview of the MAST system model which is used by GEN4MAST. In Section III a description of the GEN4MAST architecture is given. Sections IV, V and VI address the three different phases in which GEN4MAST is divided: the generation, execution, and results processing phases, respectively. Section VII presents an evaluation of the generation methods included in GEN4MAST. And finally, in Section VIII we present the conclusions and future work.

II. The MAST System Model

The MAST system model assumes a real-time distributed system with multiple processing resources (CPUs) and one or more communication networks. This system is composed of distributed end-to-end flows (following the terminology of the OMG's MARTE standard) with periodic or sporadic activations. Sporadic activations are treated as periodic, using a period equal to the minimum interarrival time.

Each end-to-end flow Γ_i is released by a periodic sequence of external events with period T_i, and contains a set of N_i *steps*. A step represents the execution of a task in a processor, or a message transmitted in a network. Each periodic release of an end-to-end flow causes the execution of one instance of the set of steps, each step being released when the preceding one in its end-to-end flow finishes its execution. The MAST analysis tools consider messages as tasks executing in a normal processor, adding blocking times due to the non-preemptability of the transmitted packets. For simplicity, and without losing generality, the current version of GEN4MAST only considers processors. The full support for communication networks will be added in future versions, including networks such as AFDX [9], which require special purpose analyses.

We assume that all event sequences that arrive at the system and their worst-case rates are known in advance, and we also assume that steps are statically assigned to processors (migration is not allowed). The relative phasing of the activations of different end-to-end flows is assumed to be arbitrary.

Fig. 1 shows an example of one end-to-end flow with three steps, each executing in a different processing resource PR_x. The arrival of the external event that releases the i-th end-to-end flow is represented by a thick horizontal arrow labeled e_i, and has a period of T_i. The thin horizontal arrows represent the release of the following steps in the end-to-end flow; a step cannot be executed before the preceding step has been completed. We assume that events represented in the figure are instantaneous and any activity in the system is modeled as a step. The j-th step of end-to-end flow Γ_i is identified as τ_{ij}; it has a worst-case execution time of C_{ij}, and a best-case execution time of Cb_{ij}. The utilization of the system is defined as the average of the utilizations of each processing resource in the system.

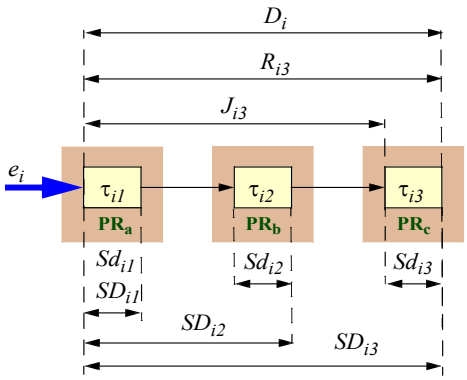

Fig 1. The model of end-to-end flow Γ_i

The timing requirements that we consider are end-to-end deadlines, D_i, that start at the end-to-end flow instance's period, and must be met by the final step in the flow. We allow deadlines to be larger than the periods. As a result of the schedulability analysis, each step τ_{ij} also has a worst-case response time (or an upper bound of it) called R_{ij}. The worst case response time estimation of the last step can be compared with the end-to-end deadline in order to determine the schedulability of the system.

Two different types of scheduling policies are supported, FP (Fixed Priorities) and EDF (Earliest Deadline First). Each step is scheduled by its scheduling parameter, which can be a priority for FP scheduling, or a scheduling deadline for EDF scheduling. A scheduling deadline is a value that is used in EDF to make scheduling decisions. It does not necessarily need to coincide with the timing requirement.

Depending on the clock synchronization available in the distributed EDF system, MAST distinguishes two different types of scheduling deadlines, global of local scheduling deadlines [10]:

- *Global Scheduling Deadline* SD_{ij}, which is referenced to the arrival of the event that releases the end-to-end flow, possibly in a different processing resource. This requires clock synchronization among all the processing resources, unless a special protocol such as the Distributed Deadline Synchronization Protocol (DDSP) [11] is used. We call this type of scheduling Global-Clock EDF, or GC-EDF.
- *Local Scheduling Deadline* Sd_{ij}, which is referenced to the release time of its associated step in its own processing resource, thus allowing the use of the local clock with no synchronization required. We call this kind of scheduling Local-Clock EDF, or LC-EDF.

The current version of GEN4MAST supports the generation of a subset of what the complete MAST model supports. This subset is rich enough to obtain valid results. It is planned that future versions of GEN4MAST will widen this subset, adding support for the generation of systems with features such as shared resources, timing requirements in the intermediate steps of an end-to-end flow, and forking events inside the end-to-end flows and networks.

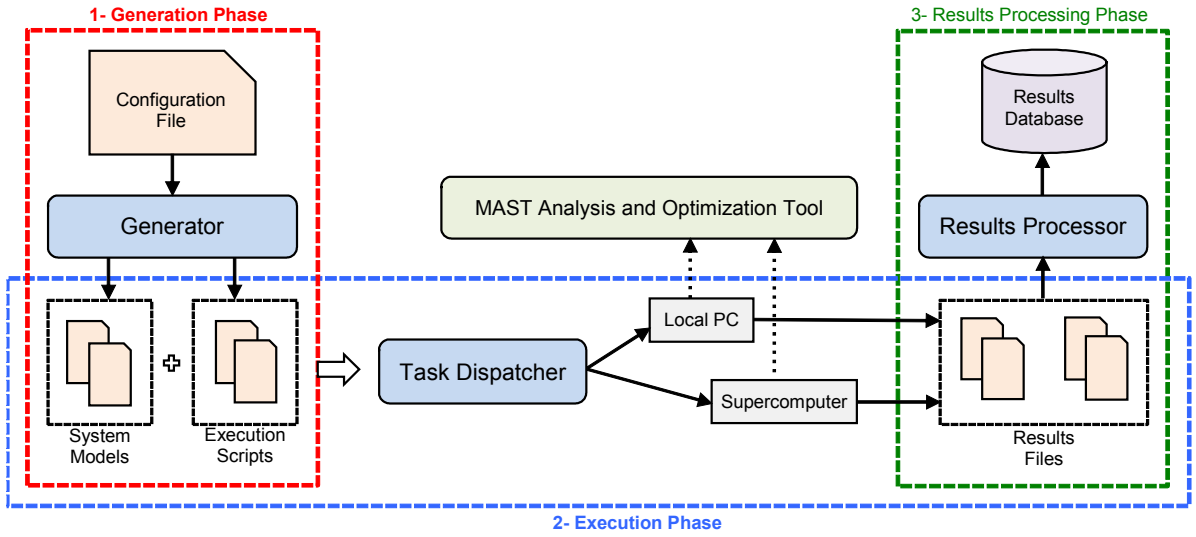

Fig. 2 GEN4MAST phases diagram

III. GEN4MAST Architecture

GEN4MAST is built as a wrapper around the MAST tool. MAST is designed to work on a "one system at a time" basis: it accepts an input file that contains a description of a real-time situation using the MAST system model, then the user selects which tools to apply over the system and which results should be calculated, and MAST returns a file with the requested results. GEN4MAST automates this whole process, coordinating the execution of many (potentially millions) of executions of MAST. Its operation is divided into three phases, which are also seen in Fig. 2:

- Generation phase: In this phase all the files required to execute MAST are created, including the system description and execution script files. The characteristics of these files are specified in the GEN4MAST configuration file.
- Execution phase: During this phase all the MAST executions are performed, either in a local PC, or in a distributed supercomputing cluster.
- Results processing phase. This phase collects all the results obtained, and stores them in an indexed database of easy access.

The three phases are described in more detail in the next sections.

IV. The Generation Phase

During the generation phase all the files necessary to perform the evaluation are created, that is, the system description files, and the MAST execution scripts. The generator module of GEN4MAST performs this process. The parameters that define the characteristics of the evaluation are provided to GEN4MAST via an external configuration file. These parameters set the characteristics of the pool of examples to be generated, and the techniques to apply over them. A configuration language has been defined naming the parameters with capital letters. If several values are given to each parameter, GEN4MAST will generate systems and execution scripts with every combination of the given values.

A. System Generation

The pool of systems is generated by following a set of generation parameters established in the input configuration file. These parameters set the basic characteristics that define a system:

- Basic architecture: number of end-to-end flows, number of steps in each flow (length), and step localization.
- Timing characteristics: periods, end-to-end deadlines and system workload.

1) Basic architecture

The number of flows is set with a positive number (N_FLOWS). There are three different parameters that define their lengths. N_STEPS is a positive number that sets the maximum number of steps any flow can have. FIXED_LENGTH is a boolean that, if True, establishes that every task has N_STEPS steps. If False, the lengths are selected randomly in the range [2,N_STEPS] for every flow. MONO_FLOWS establishes the percentage of flows that have one step only, modelling independent tasks.

The localization of the steps is the process that statically assigns a step to a processing resource. In GEN4MAST this process is controlled with two parameters. The first is N_PROCESSORS, which is a positive number that defines the number of processing resources that are available for assigning steps to. The second parameter is REPETITION. If it is set to "YES", the steps are located in a random processor. If set to "NO", the processing resource cannot be repeated for the steps of an end-to-end flow. This latter behavior can only be achieved if the length of the flow is not larger than the number of processing resources available in the system. If this

requirement is not met, GEN4MAST reverts to a random allocation of steps to processors.

Finally, GEN4MAST supports two different types of schedulers, FP and EDF, set with parameter SCHEDULING_POLICY. Currently only homogeneous systems (in which every processing resource uses the same scheduling policy) are supported. Support for heterogeneous systems [8] where different policies could coexist will be added in the future.

2) Timing characteristics

The end-to-end flows generated with GEN4MAST are periodic, with periods for each flow randomly selected within a specified range. The range is set as [PERIOD_BASE, PERIOD_BASE*PERIOD_RATIO]. The probability distribution used to select the period within the range is set with the PERIOD_DISTRIBUTION parameter. The distribution can be uniform ("UNIFORM"), or logarithmic-uniform ("LOG-UNIFORM") [12]. A logarithmic-uniform represents more closely the behavior of real systems.

The end-to-end deadline of each flow is always calculated relative to its period. Two methods are provided to specify these deadlines, set with parameter DEADLINES. The first method establishes a segment of values between T and N*T, where T is the period of the flow, and N its number of steps. Several points are defined in this segment, whose names are: "T" and "NT" (first and last point of the segment); "T1" and "T2" (first and second third of the segment); "Q1", "Q2" and "Q3" (first, second and third quarter of the segment); and "RANDOM" (random value in the segment). The second method sets a constant positive number "K". The end-to-end deadlines are calculated as K*T.

With all the previous parameters already set in a given system, only the execution times of the steps are needed to complete the description of a system. These execution times of the steps represent the workload of the system. In GEN4MAST, a system utilization is configured, and from this utilization and the periods, the worst case execution times of the steps are calculated. Two methods are provided to generate the workload, set with the parameter WORKLOAD. The first one is SCALE-WCET, which generates the worst-case execution times so that every step in each processing resource contributes with the same step utilization (C_{ij}/T_i). The second method is UUNIFAST [13], which distributes the step utilization uniformly, generating unbiased workloads. Although it was designed for use in single processors, since we apply UUnifast individually in each of the processors of our generated systems, it is compatible with our distributed system model.

Additionally, the best-case execution time of the steps can be set as a percentage of the worst-case values, using the BEST_CASE parameter.

In real-time systems it is typical to study the behavior of a technique over a range of utilizations, observing for example the maximum utilization for which the system meets all of its deadlines. This maximum utilization is usually called the breakdown utilization [14], or just the maximum schedulable utilization. For this reason, in GEN4MAST a series of utilizations is generated for each system configuration. Each series is defined by its initial utilization, the last utilization, and the step, so that the series is composed of utilizations in the range [initial, initial+step,…,last]. The parameters that set these characteristics are UTILIZATION_START for the initial utilization, UTILIZATION_STEP for the step, and UTILIZATION_STOP for the last utilization.

The system utilization is the average utilization among the processors in the system. If the parameter UNIFORM_UTILIZATION is set to True, every processor has the same utilization in every step of the range. If set to False, the processors can have different utilizations, averaging the same system utilization.

The previous parameters set the rules with which the actual systems characteristics are calculated. Additionally, GEN4MAST supports the direct specification of some of these system characteristics, such as the actual values of the flow lengths, period and deadlines, and the exact localization of every step.

Many of the attributes of the systems generated with GEN4MAST are calculated using a probability distribution. To be able to obtain a statistically relevant pool of examples, the POPULATION parameter sets the number of series that are going to be generated with the same set of characteristics.

B. Execution scripts generation

Once the pool of description files has been generated, the next step is to create the MAST execution scripts. Each of these scripts defines one execution of the MAST Analysis tool, that is, the application of some schedulability technique over a system and, optionally, the calculation of scheduling parameters or slacks [6]. The scope of the execution scripts is also configured in the GEN4MAST input configuration file with a series of parameters:

- MAST_PATH: Sets the MAST analysis executable path. Several paths can be given to compare the results of different versions of MAST, for example to test a custom version of MAST.
- ANALYSIS_TOOL: Sets the schedulability analysis techniques to perform over the pool of examples. The accepted values for this parameter are HOLISTIC, OFFSET, SLANTED, OFFSET_OPT and BRUTE_FORCE. This values are mapped into the techniques available in MAST [6], which are, respectively: holistic analysis, offset based, offset based slanted, offset based with precedence optimizations, and offset based exact. If a system uses a scheduling policy not supported by the analysis technique, GEN4MAST reverts to holistic, which is available for FP, GC-EDF and LC-EDF
- STOP_FACTOR: Positive number that sets the schedulability analysis stop factor. The analysis stops if any response time is larger than STOP_FACTOR*D, where D is the deadline of the end-to-end flow. This factor is used to avoid large executions or convergence problems of the analysis tool in non-schedulable systems.

- DEADLINES_TYPE: For distributed EDF systems only. It specifies whether the scheduling deadlines apply to GC-EDF or to LC-EDF. The valid values for this parameter are "GLOBAL" and "LOCAL" respectively.
- ASSIGNMENT_TOOL: Specifies the scheduling parameter assignment techniques to apply over the pool of generated examples. The accepted values are PD, NPD and HOSPA. These values are mapped into the assignment techniques with the same names available in MAST [6].
- SLACK: Boolean. If set to "True", the overall system slack if calculated.
- NO_JITTER: Boolean. If set to "True", the analysis tool will emulate the usage of a jitter avoidance technique, such as sporadic servers, or constant offsets.

One of the scheduling parameters assignment techniques available in MAST is HOSPA [8], which is a heuristic algorithm that assigns and optimizes fixed priorities and scheduling deadlines in distributed heterogeneous systems. Its behavior can be modified with a set of user configurable parameters. GEN4MAST can configure HOSPA with these parameters:

- K: Establishes the pairs of the k parameters to use with HOSPA.
- ITERATIONS_PER_PAIR: Sets the number of iterations that HOSPA will perform for each pair of k values.
- OPTIMIZATION_ITERATIONS: Sets the number of iterations over an already schedulable assignment that HOSPA will perform, to further optimize the solution.
- INITIALIZATION_TYPE: Sets the technique used for the first assignment in HOSPA.

V. The Execution Phase

The generation phase creates both the pool of examples under analysis, and the MAST execution scripts that define which techniques to apply over them. During the execution phase, the GEN4MAST Task Dispatcher module is responsible for the execution of these scripts, so the relevant results can be obtained. The GEN4MAST Task Dispatcher supports two different methods of execution: local, and distributed execution.

Local execution is a straightforward execution mode in GEN4MAST, in which the scripts are executed sequentially in the host PC. This mode is only viable for small studies with short total execution times.

With the distributed method of execution, GEN4MAST can take advantage of a cluster of supercomputing to minimize the total execution times of the study, and thus gaining the ability to perform larger studies in reasonable amounts of time. In a distributed environment, the Task Dispatcher module submits jobs (MAST executions scripts) to the cluster job manager, which are then executed in parallel in the available computing nodes. Every MAST execution is completely independent from the others, so no communication between the jobs is needed, simplifying the implementation and execution of GEN4MAST.

The GEN4MAST supercomputer support is built using the TORQUE [15] resource manager, which is a widely used job management system. With this system, new jobs are sent to a ready job queue. The resource manager then picks which jobs in the queue are executed taking into consideration factors such as processor availability, job and user priority, or system congestion.

A supercomputing system is typically used by several users at the same time. The Task Dispatcher module has to take measures to avoid overloading the cluster job queue, and thus degrading the performance of the system for the rest of the users. These measures are:

- The jobs that are submitted to the cluster execution queue are formed by one or more of the MAST execution scripts. In this way the total number of jobs submitted is reduced. The job size can be modified with the PACK_SIZE parameter, which sets the number of scripts packed in a cluster job.
- A maximum number of jobs executing or waiting at any time in the cluster can be set. If this limit is reached, a pre-established amount of time is awaited before trying to send a new job, so previous jobs can finish. The total number of jobs, and the waiting time, can be configured with parameters BATCH_SIZE and BATCH_SLEEP, respectively.
- A timeout value can be set with parameter TIMEOUT to stop the execution of jobs with convergence problems, thus freeing a computing node in that case.
- GEN4MAST cluster jobs are assigned by default the minimum priority. A different priority can be set with parameter GROUP. The accepted values of this parameter are cluster dependent.

VI. The Results Processing Phase

Independently of the method used (Local or Distributed), the result of executing every script of the study is a set of files with the results, one file for every MAST execution. The results stored are the worst-case response times of every end-to-end flow, the end-to-end deadlines used to check the schedulability, the execution times, and if calculated, the system slack. These results are written in plain text, and can be easily read with a text editor.

To facilitate the batch processing and access to the results, especially in large evaluations, GEN4MAST provides a module that compiles all the results into an easily accessible database. This database is in PyTables format [16], which is a Python package for managing large datasets efficiently. PyTables is well documented, and its API can be used to access the results stored in the database. GEN4MAST provides a Python library with functions to easily filter the information contained in the database, and present the results with tables and charts.

TABLE I. SYSTEM CHARACTERISTICS

Number of end-to-end flows	Flows length	Number of processors	Period selection ratio	End-to-end deadlines	Scheduling Policies
10	10	5	100	$10*T_i$ ($2*T_i$ for GC-EDF)	FP LC-EDF GC-EDF

VII. EVALUATION OF GEN4MAST'S GENERATION METHODS

One of the main objectives of GEN4MAST is to generate synthetic systems that represent real situations as close as possible, and with which unbiased results can be obtained. To achieve these goals we have shown that GEN4MAST integrates several methods for the generation of the systems workloads. In this section we will evaluate the influence on the results of these techniques. This evaluation will also serve as a showcase of GEN4MAST.

We will use GEN4MAST to generate a pool of systems with the characteristics listed in Table I. To generate the periods, both uniform and log-uniform methods are used. For the worst-case execution times, SCALE-WCET and UUnifast are used. For each combination of characteristics and methods, 300 series of utilizations in the range [10-99%] are generated. In total, 324000 systems were generated.

We analyze the schedulability of each system using the holistic technique, which is the only one available in MAST for the three scheduling policies under study [6]. We will observe the evolution of the maximum schedulable utilization in two different directions, the number of series averaged (from the 300 created), and the different combinations of generation methods. The execution of this evaluation was performed using a supercomputer available at the University of Cantabria composed of 824 cores (Intel Xeon and AMD Opteron), and almost 3TB of RAM. For this evaluation we had 300 cores available. The execution of the study required a total of 30 minutes. If it had been executed in a conventional computer it would have needed almost one week.

Before showing the results, it is important to note that in the case of systems scheduled by GC-EDF, the end-to-end deadlines were set to $D_i=2*T_i$ instead of $D_i=10*T_i$. GC-EDF has a much larger scheduling capability than FP and LC-EDF [17] , and using $D_i=10*T_i$ would yield maximum schedulable utilizations that would saturate around 100%. Setting lower end-to-end deadlines lets us appreciate the differences among the techniques under study.

The results are shown in Fig. 3, 4 and 5 for systems scheduled by FP, LC-EDF and GC-EDF respectively. The figures make clear the great influence on the results of using the different generation methods, where the highly biased Uniform and SCALE-WCET methods induce lower schedulable utilizations than Log-Uniform and UUnifast. The differences vary from up to 16% in FP, to around 25% in both LC-EDF and GC-EDF.

Furthermore, independently of the generation method used, it can be observed that the average maximum schedulable utilization stabilizes with around 60 series. Therefore, generating 60 series is enough to obtain valid results in this system.

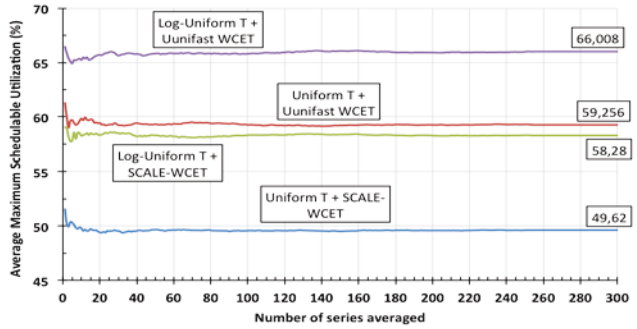

Fig. 3. Average maximum schedulable utilization evolution of the systems scheduled by FP.

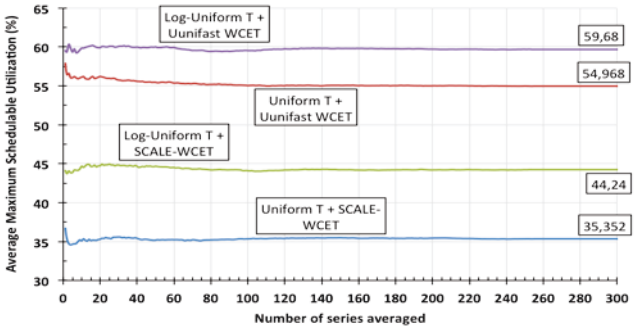

Fig. 4. Average maximum schedulable utilization evolution of the systems scheduled by LC-EDF.

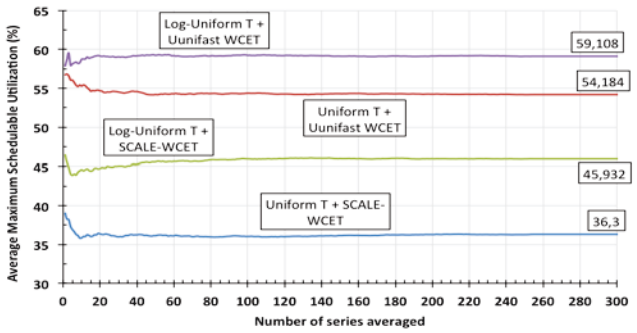

Fig. 5. Average maximum schedulable utilization evolution of the systems scheduled by GC-EDF.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we have presented GEN4MAST, as a wrapper of MAST that automates the generation of systems, the execution of techniques over them, and the processing of the results. Its main advantage over previous tools is that it can leverage the performance of a supercomputer to create large evaluations in reasonable amounts of time.

We have shown its basic architecture and configuration parameters. As an example of a typical use-case of GEN4MAST, we performed an evaluation of its generation

methods. We showed that the different generation methods have a decisive influence in the results of distributed systems. The generation method must be chosen to mimic the characteristics of the actual systems being modeled.

GEN4MAST is still under development, and it is planned that it will be evolved alongside MAST. The source code and manual of GEN4MAST will be added to the MAST home page [6]. GEN4MAST has already been successfully used in several research works including a comparison among scheduling algorithms [17], a test of new deadline assignment algorithms for EDF [10], or a test of a new scheduling analysis technique for LC-EDF [18].

Finally, one of the early observations when applying GEN4MAST to large experiments requiring for example more than one year of CPU usage (a little bit more than one day of execution time in the supercomputer), is that a large amount of data is generated (tens of Gigabytes in the situation described above). Currently, plain results are processed and represented, but the application of Big Data techniques will be explored for a more sophisticated processing.

REFERENCES

[1] P. Courbin, and L. George, "FORTAS: Framework for real-time analysis and simulation," Proceedings of the 2nd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), Porto (Portugal), pp. 21-26, 2011.

[2] R. Urunuela, A. Déplanche, and Y. Trinquet, "STORM: a simulation tool for real-time multiprocessor scheduling evaluation," Proc. of the 15th IEEE Conference on Emerging Technologies and Factory Automation (ETFA), Bilbao (Spain), 2010.

[3] A. Hangan, and G. Sebestyen, "RTMultiSim: A Versatile Simulator for Multiprocessor Real-Time Systems," Proceedings of the 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), Pisa (Italy), 2012.

[4] Y. Chandarli, F. Fauberteau, D. Masson, S. Midonnet, and M. Qamhieh, "YARTISS: A tool to visualize, test, compare and evaluate real-time scheduling algorithms," Proceedings of the 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), Pisa (Italy), 2012.

[5] M. González Harbour, J.J. Gutiérrez, J.C. Palencia, and J.M. Drake, "MAST: Modeling and Analysis Suite for Real Time Applications," Proc. of the 13th Euromicro Conference on Real-Time Systems (ECRTS), Delft (The Netherlands), pp. 125-134, 2001.

[6] MAST home page, http://mast.unican.es/

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

[8] J.M. Rivas, J.J. Gutiérrez, J.C. Palencia, and M. González Harbour, "Schedulability Analysis and Optimization of Heterogeneous EDF and FP Distributed Real-Time Systems," Proc. of the 23rd Euromicro Conference on Real-Time Systems (ECRTS), Porto (Portugal), pp. 195-204, 2011.

[9] Airlines Electronic Engineering Committee, Aeronautical Radio INC., "ARINC Specification 664P7: Aircraft Data Network, Part 7 - Avionics Full Duplex Switched Ethernet (AFDX) Network," June 27, 2005.

[10] J.M. Rivas, J.J. Gutiérrez, J.C. Palencia, and M. González Harbour, "Deadline Assignment in EDF Schedulers for Real-Time Distributed Systems," In press, IEEE Transactions on Parallel and Distributed Systems, DOI: 10.1109/TPDS.2014.2359449.

[11] N. Serreli, G. Lipari, and E. Bini, "The Distributed Deadline synchronization Protocol for Real-Time Systems Scheduled by EDF," Proc. of the 15th IEEE Conference on Emerging Technologies and Factory Automation (ETFA), Bilbao (Spain), 2010.

[12] P. Emberson, R. Stafford, and R.I. Davis, "Techniques for the synthesis of multiprocessor tasksets," Proceedings of the 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), pp. 6-11, 2010.

[13] E. Bini, and G.C. Buttazzo, "Measuring the performance of schedulability tests," Real-Time Systems 30.1-2, pp. 129-154, 2005.

[14] J.P. Lehoczky, L. Sha, and Y. Ding, "The rate-monotonic scheduling algorithm: Exact characterization and average case behavior," Proceedings of the 10th IEEE Real-Time Systems Symposium (RTSS), Santa Monica (CA, USA), pp. 166-171, 1989.

[15] Torque Resource Manager home page: http://www.adaptivecomputing.com/products/open-source/torque/

[16] PyTables home page: http://www.pytables.org

[17] J.M. Rivas, J.J. Gutiérrez, and M. González Harbour, "Fixed Priorities or EDF for Distributed Real-Time Systems?," Proc. of the WiP Session of the 33rd IEEE Real-Time Systems Symposium, San Juan (Puerto Rico), 2012.

[18] U. Diaz De Cerio, M. González Harbour, J. C. Palencia, and J.P. Uribe, "Adding Precedence Relations to the Response-Time Analysis of EDF Distributed Real-Time Systems," Proc. of the 22nd International Conference on Real-Time Networks and Systems (RTNS), Versailles (France), pp. 129-138, 2014.