EXPERIENCES WITH AGAPE-TR: A SIMULATION TOOL FOR LOCAL REAL-TIME SCHEDULING

Gabriel A. Wainer

Computer Sciences Department.
Facultad de Ciencias Exactas y Naturales.
Universidad de Buenos Aires.
Ciudad Universitaria. Pabellón I. Buenos Aires.Argentina.
gabrielw@dc.uba.ar
http://www.angelfire.com/ga/gw

Abstract: Local real-time scheduling is a complex problem. The diversities of existing solutions hinder to decide which algorithm to use upon designing each system. In this report a tool to simulate real-time scheduling (AgaPé-TR) is presented. The tool helps to classify and compare different solutions. To do so, it reflects the general characteristics of the problem, and provides facilities to include new algorithms. Using a synthetic benchmarking model, it permits to analyze different scheduling algorithms to get useful metrics. It also allows to make off-line schedulability analysis. The experiments carried out allowed to show its usefulness.

Keywords: Real-Time, scheduling algorithms, simulation, performance evaluation.

1. INTRODUCTION

In real-time systems the computer control events occurring in the real world, hence the correct system behavior depends on the moments the computations are carried out. To meet the system's timing constraints is usually a hard work, as there are many restrictions to consider, usually conflicting.

Not only task's deadlines must be considered; each task also has to meet timing constraints to use systems resources. As complex real time systems are usually divided in multiple tasks, precedence constraints also must be considered. In this framework, each task should meet concurrence constraints. In distributed systems the situation is even more complex, as we should meet communications restrictions (including tasks' interconnection and timing requirements), and load balance constraints. Finally, the criticality level is another different constraint, telling that the execution of one task is more critical than other.

The programming environment should provide facilities to easily develop real-time applications. One way to do so is to rely on real-time scheduling policies that must insure timely execution of every task in the system. To do so, the scheduler must set a predictable execution order (the schedule).

At present, the number of real-time scheduling solutions is so diverse that it is difficult to decide which algorithm to use. In this work the results obtained using a tool called AgaPé-TR are presented. It is designed to ease the study of local real-time schedulers, allowing to classify them through simulation and collection of performance metrics. It also can be used by system designers to do off-line predictability analysis. The inclusion of new schedulers in the tool is easy, allowing the thorough study of new proposals.

2. REAL-TIME SCHEDULING

As explained in the previous section, the variety of constraints in real-time scheduling makes it an NP-hard problem. To reduce this complexity, most algorithms pose simplifications on the task models. They usually do not consider several constraints, and provide solutions useful for simplified task sets that can be scheduled in a timely fashion.

Due to the diversity of scheduling solutions (see, for instance, Cheng, *et al*., (1988), Mercer (1992), Stankovic, *et al*., (1994), or Wainer, (1995)), the designers face a complex decision to choose the right scheduling approach. AgaPé-TR was developed to allow the comparison of different approaches, helping the designer to decide. As it is easy to include new approaches or to change existing strategies, it can help the designer to compare with known strategies. This approximation is useful facing the implementation of new solutions, mainly where formal results of the advantages of a new scheduler are very complex or impossible to obtain. Finally, facilities to make off-line guarantee analysis are provided, allowing the designers to make schedulability studies of particular task sets.

To measure the merit of each scheduling solution, several metrics can be used. To decide which ones to include, the following goals of a real-time scheduler were considered:

- Predictable response of system tasks. A measure of **predictability** is the **guarantee ratio**, that is, the number of guaranteed tasks against the number of tasks arriving to the system.
- High degree of **schedulability**, that is, resource use meeting the tasks' timing constraints. As many real-time tasks as possible should be executed in a timely fashion. A measure related with schedulability is the processor **time loading**, that measures the useful processing time.
- The scheduler should provide **stability** under transient overloads. When the system is overloaded and it is impossible to meet the deadlines of every task, the behavior of the most critical ones should be guaranteed. A measure of system stability is the **guarantee ratio weighed** by the tasks' criticality.

When the scheduler meets these goals, the development complexity is reduced and the maintainability increased. Also, the safety of the developed system is improved (because it is less sensitive to timing errors). Hence, the design of AgaPé-TR, is oriented to collect data to check these features in any chosen scheduler.

3. DESCRIPTION OF THE TOOL

Several scheduling algorithms rely on a task model with two kinds of tasks: the periodic and sporadic. Periodic tasks have a continuous series of regular invocations (the task period), and a worst case execution time (WCET). Their deadlines can be defined at the beginning, before or after the next period, depending of the task model. They are very common, because most real-time systems have some type of regular sampling. Sporadic tasks only have one activation instance with a start time and deadline. We also must consider their WCET. They are useful to respond to events with random arrival time (for instance, emergency tasks).

The simulator models the execution of periodic and sporadic tasks using this model. Its inputs are a set of parameters of each task: class (periodic or sporadic), period (or deadline for aperiodics), WCET and start time. Task's criticality and a semaphore set related with the task are also included. Finally, the WCET of an alternative routine associated with the task is also included (new parameters can be easily added). Using this infomation, guarantee tests are executed to study predictability of the task set. The initial time loading is also computed. The tests are repeated each time a new task start or when a sporadic instance finish.

Then, the execution of the task set is simulated using the periodic tasks model. It is considered that when an instance finishes the task is delayed up the beginning of the next period in a queue driven by the timer interrupt.

At present some well-known scheduling algorithms have been included :

- Rate Monotonic (RM) (Liu and Layland, 1973; Lehoczky, *et al*., 1986): it is used to schedule periodic independent tasks. The tasks are selected accordingly with their activation rate (preemptive fixed priorities).
- Earliest Deadline First (EDF) (Liu and Layland, 1973): it selects the system tasks using variable preemptive priorities, giving higher priority to tasks with the earliest deadlines.
- Earliest Deadline (ED) (Dertouzos, 1974): the highest priority is assigned to the task with the earliest deadline (fixed priorities without preemption).
- Least Laxity First (LLF) (Mok and Dertouzos, 1978): it is an Earliest Deadline algorithm, but considering the task laxity, that is, the remaining execution time up to the deadline. Tasks with smaller laxity are chosen to execute first.

- Least Slack First (LSF) (Mok and Dertouzos, 1978): the highest priority is assigned to the task with the least laxity, determined statically at task arrival and not adjusted afterwards.
- Deadline Monotonic (DM) (Audsley, *etr al.*, 1993): it requires static priorities in ascending order according with the periods, but does not require deadlines equal to the period.

A graphical interface is used to show task exe cution, as well as preemptions and deadline missing. In this way the exe cution trace of an individual task set can be studied, and off-line predictability analysis can be done.

As the main goal is to compare different algorithms, a second option generates task sets, and analyzes the results of their execution, computing different metrics. In each case, metrics are stored for the whole task set, discriminating the results for periodic and sporadic ones. To analyze system predictability, information is collected to study the relative guarantee ratio (relationship between the number of successful instances and the number of instances). The schedulability is studied analyzing the system time loading (the percentage of useful processing done by the system). The number of preemptions and context switches entitle the study of overheads imposed by the environment. The idle processor time is also examined. This metric is related with system schedulability, available time to run alternate routines and fault-tolerance. As several schedulers deliver rejected tasks to other processors, the number of rejected tasks against total tasks arrived is also measured.

The stability is studied computing the relative guarantee ratio, weighted by the tasks' criticality. The idea is to associate a criticality value to each task (a real number among 0 and 1, increas ing with the task criticality). Let us call **xi** to the criticality of task ti, and **Gp** to the relative weighted guarantee ratio. Then, let us define two sets, **P** = {instances loosing its deadline}, and **M** = {Number of instances}. Hence,

$$\text{Gp} = \frac{\sum_{i \in P} xi}{\sum_{i \in M} xi}. \qquad \text{As } \forall\, i \in [1,n]$$

$xi \in R$, $xi \in [0,1]$, and $P \subseteq M$, then $Gp \in R$, and $Gp \in [0,1]$. This value expresses a relative weighted ratio to compare the execution of the most crucial tasks. As **Gp** gets closer to 1, the system stability in - creases. If **Gp**=1, the scheduler meets every deadline.

The tool simulates different task loads, and collects the mentioned metrics to compare the scheduler be - havior for different loads. To do so, an experimental frame was used to generate task loads based on the Hartstone benchmark (Weiderman, 1989). Certain changes to this benchmarking model were proposed, and new tests were introduced, allowing to run task sets with higher degree of offered load. The offered load is not the real system load (that can never be more than 100%), but the one obtained using the parameters declared by the designer, and used for the guarantee tests. The following guides suggested by Hartstone benchmarks were considered:

- Make growing complexity tests, taking basic requirements and establishing strategies to change them.
- The load must represent real-time systems.
- Relative merit figures should be provided.

To achieve these goals, the following kind of loads have been included :

- PH: Periodic Harmonic task sets are executed. To begin, an underloaded system is generated, and it is gradually loaded. 1) Tasks' WCETs are augmented (increasing the total system load, al-lowing to test the influence of system overloading without new overhead). 2) Tasks' periods are reduced (allowing to test system ability to handle an in cremental workload). 3) One task is selected its frequency increased, augmenting context switches, and testing the ability to manage fine time granularity. 4) The overload is increased adding new tasks, checking the ability to handle a large number of tasks.
- PN: The previous tests are repeated, but in this case the Periodic tasks have Non harmonic peri-ods. This second kind of test was included, as non harmonic tasks produce worst processor use than harmonic ones.
- AH: a set of Harmonic periodic tasks is combined with Aperiodic ones. 1) The system is overloaded, reducing periodic tasks' periods. This test allows to test fine granularity of peri-odic tasks when aperiodics have relaxed dead - lines. 2) Overloads are generated by adding new periodic tasks to the task set. In this case system behavior when scheduling many periodic tasks and how they influence sporadic ones can be studied. 3) Sporadic tasks are added to study systems with a large set of them (to study, for in - stance, emergency conditions where many spo - radic tasks are started). 4) The influence of spo - radic tasks with short deadlines is studied re - ducing their deadlines. 5) The the WCET is augmented for the sporadic tasks to test the in - fluence of overloading them.
- AN: the AH tests are repeated where the periodic tasks have Non harmonic periods.

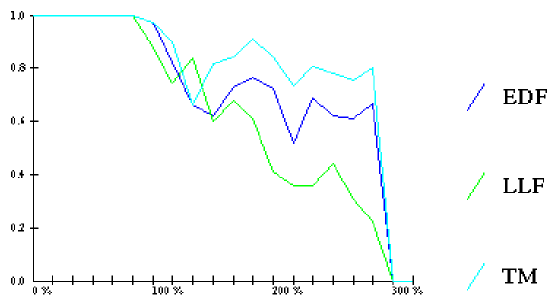
If a significant sample is collected, the system behavior of each algorithm can be studied for different loads using each metric. To do so, we compute statistical measures to analyze the behavior of the different policies, classifying the tests and the different task classes.

The simulated executions are classified according their processor system load, starting with under-loaded systems. For each metric, minimum and maximum values, average and standard deviation are studied (at present, we are adding confidence intervals). In this way a wide variety of information is available to compare the simulated algorithms. Due to the diversity of information provided, it is difficult to study the results. Hence, a simple graphical interface was built, allowing to select the information to study and easing its visualization (it can be seen in the Appendix).

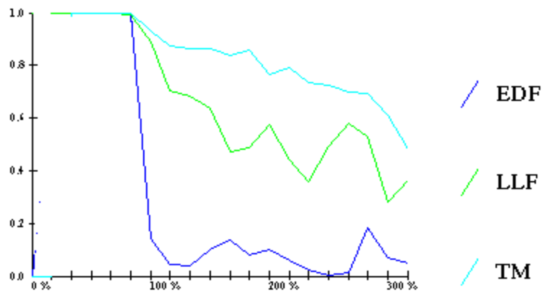
One problem is related with the compute time to obtain significant samples. To improve the performance two different simulation approaches have been used. One of them is time-driven, and the other is event-driven. Each of them provides different results depending on the algorithm to simulate. For instance, RM can obtain significant speedup with the event-driven approach, as future events can be easily determined. On the other hand, a time-driven approach provides better results for algorithms such as LLF because it has potential context switches in every tick. In this case, the overhead wasted in the manipulation of an event list reduces performance. As the execution of each load is is independent from the others, speedup can be obtained through system replication on multiple processors.

4. RESULTS

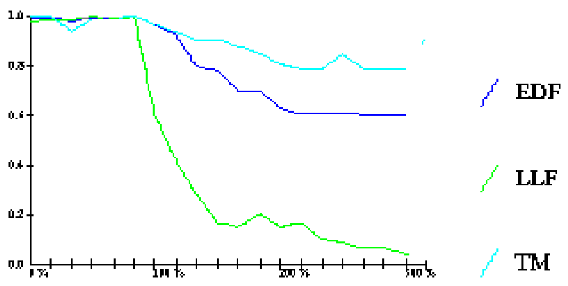
AgaPé-TR has been used in our Department, where the students could implement and experiment with different scheduling algorithms. Some well-known results found in the real-time scheduling literature could be checked in an experimental fashion. Simple comparisons were made between RM (TM in the figures), EDF and LLF, that can be seen in the following figures.



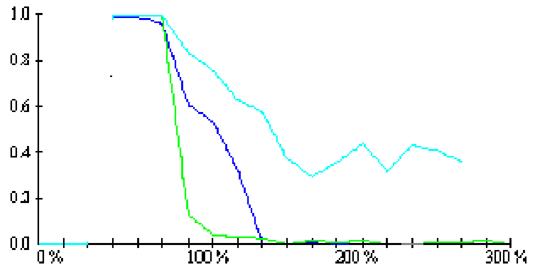
(a)



(b)

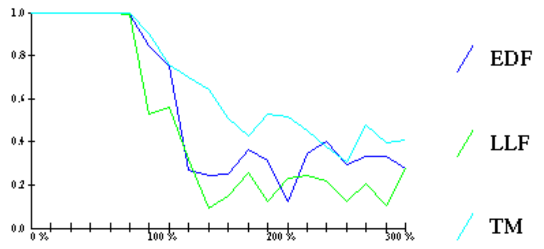


(c)

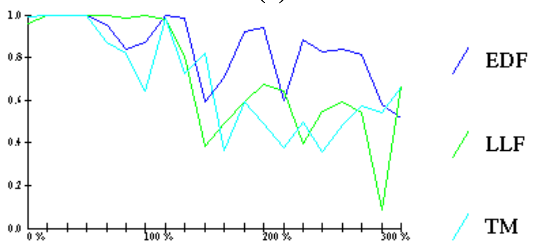


(d)

Figure 1. Minimum Guarantee Ratio; Periodic Tests: (a) PA1; (b) PN1; (c) PA3 ; (d) PN3.

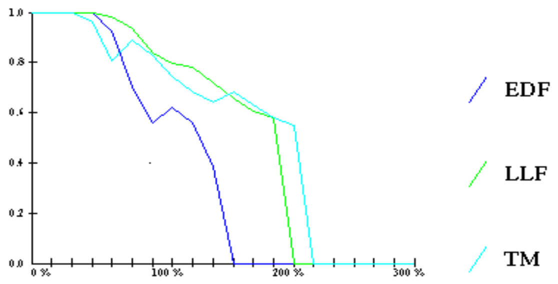


(a)

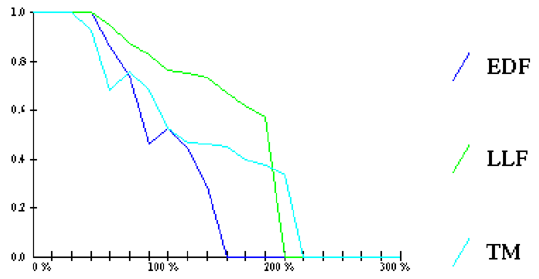


(b)

Figure 2. Aperiodic tasks - test AA1. (a) Minimum G.R. (b) Min. wheighed G.R. (only aperiodics).

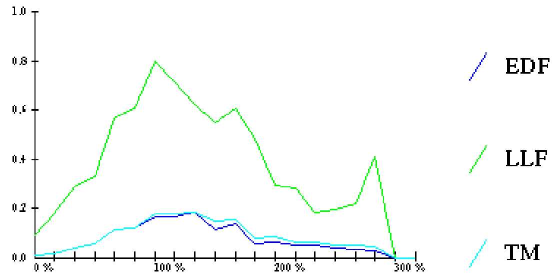


(a)

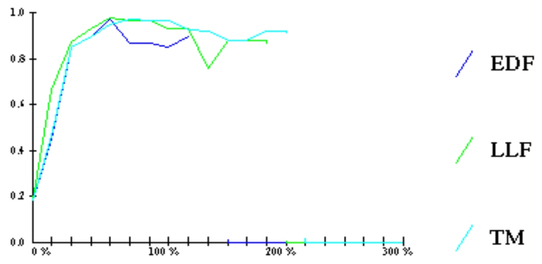


(b)

Figure 3. Aperiodic tasks - test AN4. (a) Minimum G.R. (b) Min. Wheighed G.R. (only aperiodics).



(a)



(b)

Figure 4. (a) Preemptions (PA1); (b) Context Switches (AN4).

We could check that:

- Harmonic periodic tasks produce 100% of guarantee ratio when the schedulability tests are met. Instead, guarantee ratio is under 100% for non-harmonic task sets.
- RM algorithm has stability under transient overloading for certain periodic tasks, since executes the tasks with shorter periods predictably.
- EDF and LLF algorithms handle better the sporadic tasks, if the system is not overloaded.
- EDF and LLF algorithms run erratically under transient overloading. EDF is better than LLF when the system is not loaded for harmonic periodic tasks. The results are opposite when we run non-harmonic ones.
- LLF is the algorithm with higher number of context switches and preemptions. Instead, RM and EDF have about the same number of context switches.

At present, the tool has been used to test easily new scheduling algorithms (Wainer, 1996). New studies have been started, including ED, LSF and DM algorithms. These have been programmed by undergraduates in about 10 man-hours, including 8 hours devoted to study the tool. This fact allowed to check that the system design makes easy the inclusion of new approaches.

5. PRESENT WORK AND CONCLUSION

In this work the features of a tool used to study real-time scheduling algorithms have been presented. An environment has been built considering the main features of the problem and the metrics to use.

It could be shown the capability of AgaPé-TR to make experimental analysis. Its flexibility allows to test new solutions easily. In particular, a designer interested to improve some aspects of an algorithm can include them and study the aspects to improve. Without extra effort, collateral effects can be detected (for instance, increases in overhead or reduction of stability).

AgaPé-TR also serves to make off-line schedulability studies of particular loads through system trace. In this way a designer can study the timeliness of a given task set. It has been used with success to analyze the behavior of some traditional algorithms, and now it is being used to compare them with new solutions.

At present new tests are being added to study tasks with precedence constraints, and facing new changes to study fault-tolerant scheduling solutions. The Hartstone model was found incomplete to test several existing algorithms and some solutions for these problems are being studied at present. The design of the tool makes that new tests can be added easily. The tool is courrently being prepared to run in Unix environments.

REFERENCES

Audsley, N., Burns, A. and Wellings, J. (1993). "Deadline monotonic scheduling theory and application". *Control Engineering Practice*, **1, (1),** 71-78.

Cheng, S., Stankovic, J. and Ramamritham, K. (1993). "Scheduling Algorithms for Real-time systems: a brief survey". In: *Real-Time Systems* (J. Stankovic, K. Ramamritham (Ed.)). IEEE Press, pp. 150-173.

Dertouzos, M. (1974). "Control robotics: the procedural control of physical process". In: *Proceedings of the IFIP Congress*.

Lehoczky, J.P., Sha, L. and Ding. Y. (1986).."The Rate Monotonic Scheduling algorithm - exact characterization and average case behavior". In: *Proceedings IEEE Real-Time Systems Symposium.* CS Press, Los Alamitos, California. 166-171.

Liu, C. and Layland, J. (1973). "Scheduling algorithms for multiprogramming in a Hard Real Time System Environment". *Journal of the ACM*, **Vol. 20, No. 1.** 46-61.

Mercer, C. (1992)."An introduction to real-time operating systems: scheduling theory". *Technical Report. Carnegie Mellon University.*

Mok, A.; Dertouzos, M. (1978)."Multiprocessor scheduling in a hard real-time environment". *Proceedings of the Seventh Texas Conference on Computing System.*

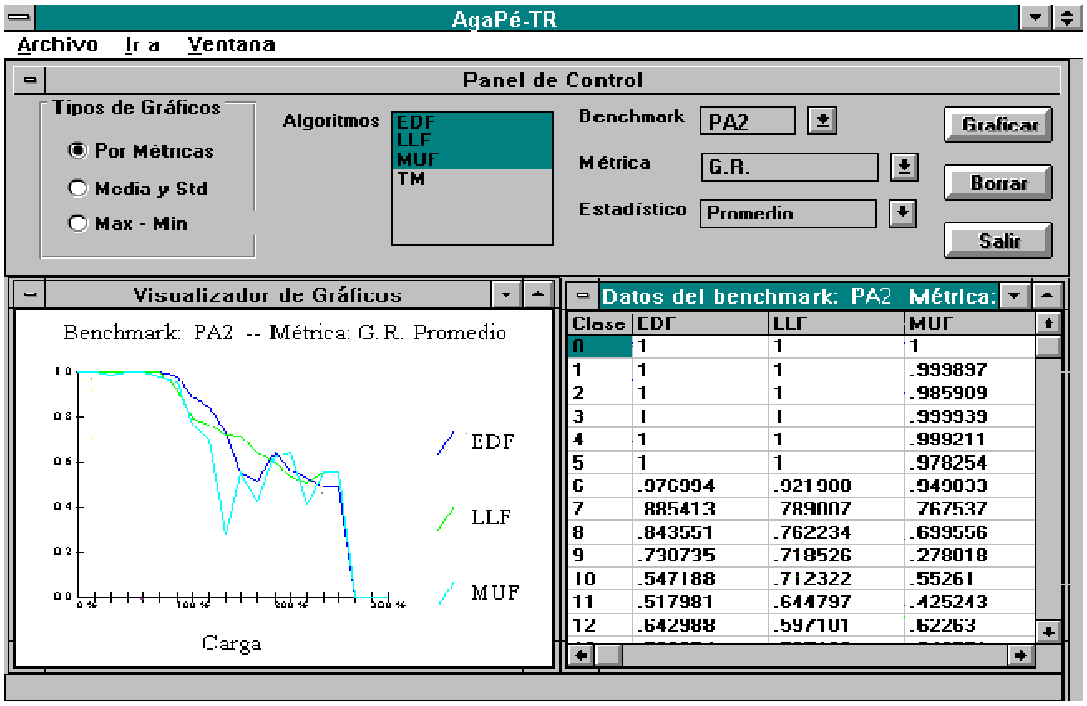
Stankovic, J., Spuri, M., Di Natale, M., Butazzo, G. (1994). "Implications of classical scheduling results for Real-Time systems". *CMPSCI Technical Report 93-23. University of Massachussets at Amherst*.

Wainer, G. (1995). "Some results of local real-time schedulin". (in Spanish). In: *Proceedings of the 24th. Jornadas Argentinas de Informática e Investigación Operativa.* Buenos Aires, Argentina. 433-451.

Wainer, G. (1996)."Improving the performance of real-time scheduling algorithms". *Technical Report No. 96-007.* Facultad de Ciencias Exactas y Nagturales. Universidad de Buenos Aires. To be published in the Proceedings of AARTC'97.

Weiderman, N. (1989). "Hartstone: synthetic benchmark requirements for Hard Real-Time applications". *Technical Report CMU/SEI-89-TR-23*. Carnegie Mellon University.

APPENDIX - Graphical interface (selection - in Spanish)



Tipos de Gráficos: [✓ Por Métricas] Media y Std Max - Min

Algoritmos: EDF, LLF, MUF (selected); TM

Benchmark: PA2

Métrica: G.R.

Estadístico: Promedio

Datos del benchmark: PA2 Métrica:

Clase	EDF	LLF	MUF
0	1	1	1
1	1	1	.999897
2	1	1	.985909
3	1	1	.999939
4	1	1	.999211
5	1	1	.978254
6	.976994	.921900	.949033
7	.885413	.789007	.767537
8	.843551	.762234	.699556
9	.730735	.718526	.278018
10	.547188	.712322	.55261
11	.517981	.644797	.425243
12	.642988	.597101	.62263