Application Oriented Performance Evaluation of Real-Time Systems

Frank Golatowski, Dirk Timmermann
University of Rostock
Department of Electrical Engineering
Institute of Applied Microelectronics and Computer Science
Richard-Wagner-Str. 31, 18119 Rostock- Warnemünde, Germany
Phone (+49) 381 4983535 Fax (+49) 381 4983601 E-mail: {gol,dtim}@baltic.e-technik.uni-rostock.de

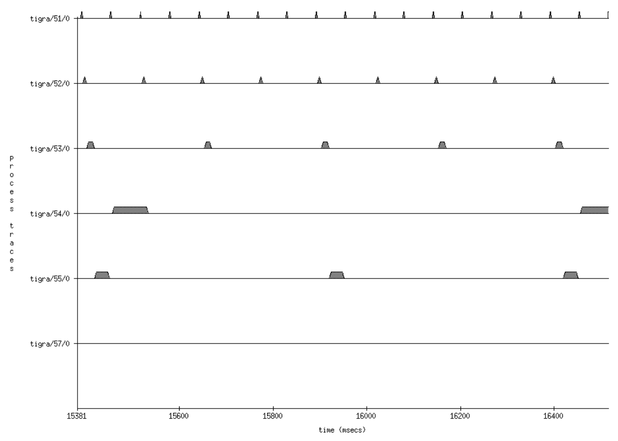
Abstract - This paper aims at the performance evaluation of real-time operating systems. Our approach is based on the Hartstone Uniprocessor Benchmark.
We have implemented this benchmark on different real-time UNIX operating systems that are running on different platforms. Based on this we distinguish three different methods to compare performance.
The first one finds breakdown utilization (BU) points. At this point a real-time system miss hard deadlines.
The second method inspects the special overload behavior beyond the BU point. This observation shows very interesting behavior of the system under overload conditions.
Thirdly our implementation considers performance evaluation but also on simulation of real-time applications. But this will not be considered here.
In this paper we will present our implementation of Hartstone benchmark for real-time UNIX operating systems. Our implementation realizes all parts of the Hartstone but modifies them where necessary to suit current hardware and software environments.
Selected results will be shown for real-time UNIX operating systems SORIX and Lynx-OS.

I. Introduction

Fine-grained performance values are used by different vendors to evaluate the performance of a real-time operating system. Performance criterions are context switching time, interrupt latency time, etc. The best-known fine-grained real-time benchmark is the Rhealstone Benchmark.
Alternative approaches to evaluate real-time system performances are application oriented or application based methods and simulation.
One attempt to evaluate the *overall* performance of a real-time system is the Hartstone Uniprocessor Benchmark. Hartstone as understood by the inventors „is a system requirement rather than an implemented program“[1]. However only a small subset of test series has been implemented in ADA [2].
The model focuses on applications typical for real-time systems. The benchmark has five different test series consisting of several experiments. Each series consists of different processes: periodic, aperiodic and synchronization processes. Periodics have hard deadlines determined by their period; aperiodics may have hard or soft deadlines. Aperiodic processes with soft deadlines run as background processes.
Experiments start with a baseline process set characterized by a number of processes, their priority, their synthetic load, process period, starting time and interarrival time.
The program uses small portions of well-known Whetstone-Benchmark as synthetic load. Each process executes this synthetic load in a loop according to a definition within a test description file.
Using synthetic load the execution of these process sets is possible. A process set is feasible for the underlying system if all deadlines of the process set are met. The breakdown utilization point is reached if at least one deadline is missed.
In this paper we concentrate on the comparison of different real-time operating systems (Lynx-OS, SORIX) running on the same machine.

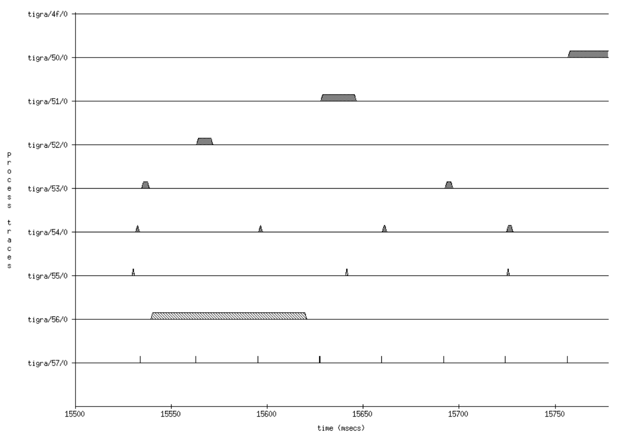
II. Benchmark model

The used workload derived from the Whetstone benchmark is called Kilo-Whetstone Instructions (KWI), because this synthetic load consists of thousand instructions of the Whetstone benchmark. The load executed by a process during an activation is measured in Kilo-Whetstone instructions per period (KWIPP) and the executed load during one second is measured in Kilo-Whetstone instructions per second (KWIPS). The benefit of this load is that granularity of the Small-Whetstone load is much fine than the Whetstone instructions in their whole.
Each series has different experiments. Experiments proceed stepwise, one parameter of the process set being changed at each step and the others kept constant.
The various series characterized by increasing complexity result from variation of the different parameters within the process set. An overview of the series is shown in Table 1.
In the model one deadline is said to be missed if a process can not finish the execution within its period (periodic processes) or up to the next activation (sporadic processes). During execution of the experiments the load is increased dynamicly up to the point where deadlines are missed. In the next steps of an experiment it is possible to observe the system under overload conditions when the synthetic load raised up.
The first series of experiments (PH-serie) starts with a baseline process set consisting of five independent periodic processes with harmonic frequencies. Periodic processes are harmonic if the relation between their frequencies is a multiple of the smallest frequency. Figure 1 shows the five periodic and harmonic processes.
The most complex series is the SA-series starting with five periodic processes, one synchronization process and one aperiodic process working as a background process. All the periodics have to synchronize during their period once with the synchronization process (see Figure 2)



PID-No.	51	52	53	55	54
process types	periodic5	periodic 4	periodic3	peridic 2	periodic1

Figure 1: A baseline process set of the PH-serie



PID-No.	50 - 54	55	56	57
process types	periodics	sporadic	aperiodic	server

Figure 2: A baseline process set of the SA-serie

Description of the PH-experiments

In the first experiment of the PH-serie (PH-1) the frequency of the fifth process will be increased by the amount equal to the frequency of the third process until a deadline is missed. The experiment will be stopped after a fixed number of missed deadlines. This experiment tests the ability to switch between processes and reveals the influence of kernel delays.

In experiment PH-2 all frequencies increase; the workload of all processes is unchanged. The experiment demonstrates the ability of the underlying system to handle increasing workload and increasing scheduling activities.

In experiment PH-3 the workload of each process is increased by an additional workload (measured in KWIPP) in such a way that the relation between process frequencies still harmonic. The utilization should be greater than in PH-2 because keeping the relation between process frequencies' harmonic result in less scheduling activity than in PH-2.

In the last experiment of the series (PH-4) the load of the system will be increased by additional processes. In this experiment the influence of additional tasks on scheduling is shown.

Table 1: Series of the Hartstone Uniprocessor Benchmark

PH-serie 1-4	5 independent periodic processes with harmonic frequencies and hard deadlines
PN-serie 1-4	5 independent periodic processes with non-harmonic frequencies and hard deadlines
AH-serie 1-6	5 independent periodic processes with harmonic frequencies 1 independent aperiodic process with soft deadline
SH-serie 1-5	5 independent periodic processes with harmonic frequencies 1 synchronization process, the periodic processes have to synchronize one time in each period
SA-serie 1-4	5 independent periodic processes with harmonic frequencies 1 independent aperiodic process with soft deadline 1 synchronization process of the SH-serie

III. REQUIREMENTS OF THE IMPLEMENTATION

We have found that the following requirements must be fulfilled for implementation:

- free selection of process parameters (by using a test description file)
- feasibility of all tests defined in the Hartstone model
- free definitions of process sets
- portability and applicability of the Benchmark for commercial real-time UNIX systems
- portability to other real-time UNIX systems
- termination of tests after an adjustable amount of missed or skipped deadlines
- termination of tests after a definite amount of time
- output of all intermediate steps on screen and parallel recording in protocol files

One objective is to find a vendor independent method to evaluate a real-time system and compare different real-time operating systems. We decided to implement the requirements following the Hartstone model for commercial real-time operating systems in C
[5].

IV. EXECUTION OF THE EXPERIMENTS

At the beginning of each experiment the raw performance executed within one process is measured.

Thereafter the test description file will be interpreted. It starts with the specification of the baseline process set. The following data in the file describe the following tests.

According to the specifications processes are created and started at the same time. A experiment terminates after a definite amount of time or definite amount of missed deadlines.

Table 2 gives the determined baseline process set for the selected reference hardware (Table 3). Note that on various machines this process set may differ because the raw performance characteristics are different. This process set is characterised by low frequencies. It produces approximatly 50 % of load. In the appropriate experiments this basic load will increased to find out the breakdown utilization point and the behavior of the system under overload conditions.

Table 2: Definition of baseline process set for PH-series

	start time (sec.)	duration (sec.)	priority SORIX/ LYNX	frequency (Hertz)	workload per period (KWIPP)	workload per second (KWIPS)
1	5	10	0/20	1.00	512	512
2	5	10	1/21	2.00	256	512
3	5	10	2/22	4.00	128	512
4	5	10	3/23	8.00	64	512
5	5	10	4/24	16.00	32	512
Σ	5	10		31.00		2560

Table 3: Reference system

Processor:	i80486
CPU-frequency	33 MHz
Cache	2nd Level/ 256 Kbytes
Wait States	1 Wait State
Memory Size	16 Mbytes

V. *Interpreting the Results*

For *n* independent periodic tasks where priorities assigned to tasks in rate-monotonic order *(according to real-time scheduling theory)* a process set is schedulable if the following condition holds:

$$n(2^{\frac{1}{n}} - 1) \geq \sum_{i=1}^{n} \frac{C_i}{T_i} \qquad (1)$$

n...number of processes T...period of the i-th process
C...computation time of i-th process

For large values of n the upper bound for utilization is is 69,31 %; for n=5 processes the limit is 74,34 %. That is, if the utilization of five processes is less than 74,34 % all deadlines will be met.

Note this is a sufficient condition and the bound is conservative, that means a process set with a utilization greater than 74,34 % could be possibly scheduled on one processor. For a detailled description of appropriate scheduling theory see [3] [4].

In our examples the achieved utilization exceeds 74,34 %. With higher frequencies in a process set there may be the case where utilization is below this threshold because for higher frequencies the system overhead increases.

System behavior

The diagrams (fig. 3 - fig. 5) show the measured utilization and the number of missed deadlines on y-axis and the nominal workload the real-time system has to execute on x-axis. The measured utilization is based upon the raw performance measured when only one process executes the synthetic load.

We present only a selection of results; further results may be obtained by the authors.

The represented utilization is equal to the nominal workload up to the breakdown utilization point. Behind that point the measured utilization is smaller than the nominal workload because missed deadlines results in not executed activities.

The demonstrated behavior matches our theoretical assumptions: achieved utilization of harmonic processes is greater than that of non-harmonics.

Processes miss their deadlines in the sequence of their priority; first the lowest prioritised process misses all its deadlines. The next prioritised process misses its deadlines after that.

The BU points of the PH-1 experiments shown in Figure 3 and Figure 5 are 97 % for Lynx and 80 % for SORIX.

In the PN-experiments it is possible that both the lowest and the next following process misses their deadlines (Figure 4 and Figure 6)

In the case of SORIX there is a point where the highest prioritised processes miss some deadlines although the processes periodic 3 and periodic 4 meet their deadlines. This behavior is dangerous in hard real-time systems. (See Figure 5).

It can be seen in most experiments that real-time performance of Lynx-OS is better than the performance of SORIX. Note both operating systems are running on the same machine.

In the SA-1 experiment the periodic processes have to synchronize with a server process. The server process has the highest priority and the number of its activations corresponds to the number of activations of all periodics. First the server misses its deadlines at utilization of 40 %. If a periodic process misses a deadline the server misses the deadlines too because server and periodic process are synchronized.

VI. *Conclusion*

In this paper, we show selected results of the evaluation of two commercial real-time UNIX operating systems using our implementation of the Hartstone Uniprocessor Benchmark. By the execution of the benchmark it is possible to find the breakdown utilization point of a system and to observe the behavior of a real-time system under overload conditions.

Applying the described method it is possible to find process sets representing the critical value of workload where the given system is still able to operate correctly in a hard real-time sense. Behind that point any increasing of workload results in missed deadlines.

The program allows the description of real world applications in a process set and its execution. Due to this fact our implementation is suitable for the simulation of a wide range of application classes. The objective is to find out whether the real-time system will be able to meet the given requirements.

Also we use a simulated application on different environments to compare the performance of different systems. Besides using these results for performance evaluation we can predict in a very early design step the feasibility of implementing a real-time application on the specific platform.

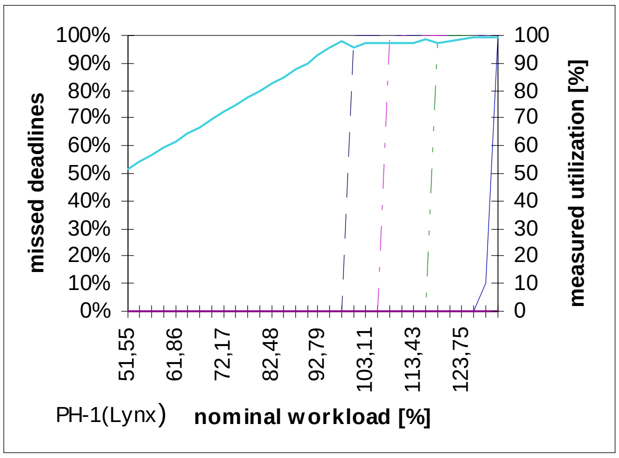


Figure 3: utilization of PH-1 tests (Lynx-OS)

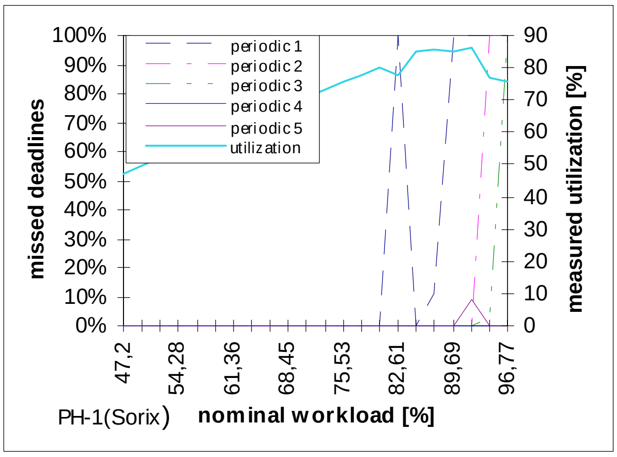


Figure 4: utilization of PH-1 tests (SORIX)

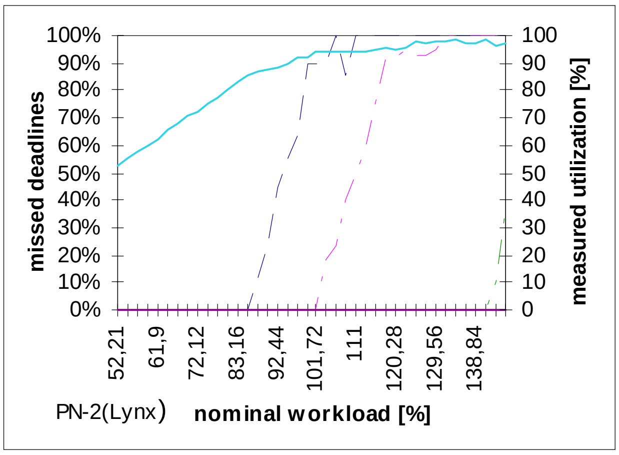


Figure 5: utilization of PN-2 tests (LynxOS)

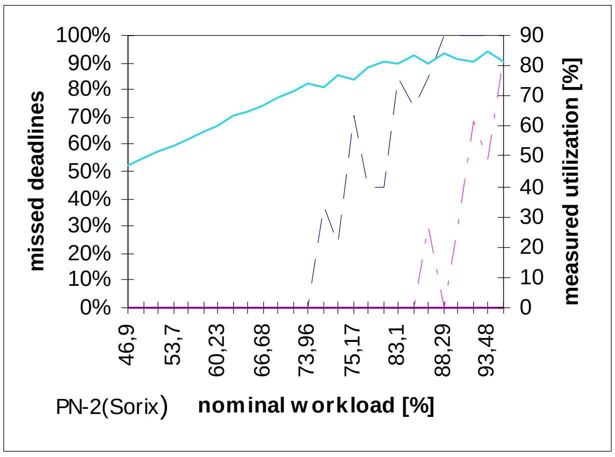


Figure 6: utilization of PN-2 tests (SORIX)

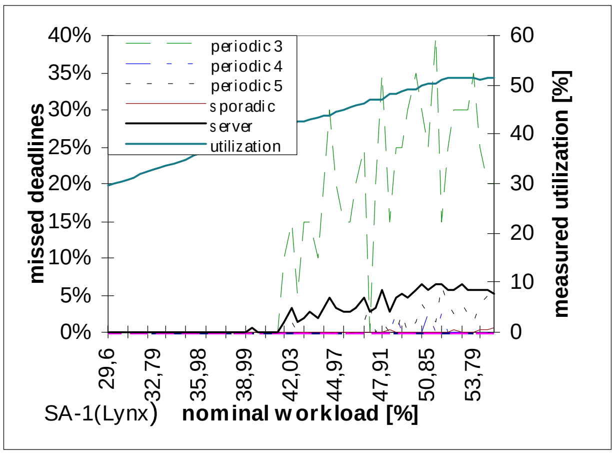


Figure 7: utilization of SA-1 tests (Lynx)

[1] Weidermann, N.H., Kamenoff, N.I., "Hartstone Uniprocessor Benchmark: Definitions and experiments for real-time systems, " in Real-Time Systems Journal, vol. 4, no. 4, pp. 353-383, Kluwer Academic Publishers, 1992

[2] Donohoe,P., Shapiro, R., Weidermann, N., "Hartstone Benchmark user's guide," Software Engineering Institute, Carnegie Mellon University, Technical Report CMU-SEI-90-TR-1, May 1990

[3] Liu, C.L. Layland, J.W, "Scheduling Algorithms for Hard Real-Time Environments". Journal of the ACM, vol. 20, no. 1, pp. 46-61, 1973

[4] Joseph, M. (ed.), "Real-time Systems-Specification, Verification and Analysis, " Prentice Hall, 1996

[5] Golatowski, F., Bösel, H., Paukert, A., "Implementierung eines applikationsorientierten Benchmarks für Echtzeit-Unix-Betriebssysteme, " in Echtzeit 95 Conference Proceedings, Rzehak, H., Ed. Karlsruhe, 1995, pp. 63-70