Application Oriented Performance Evaluation of Real-Time Systems Based on Modern Microprocessor Architectures

Frank Golatowski, Dirk Timmermann
{gol,dtim}@e-technik.uni-rostock.de
University of Rostock
Department of Electrical Engineering
Institute of Applied Microelectronics and Computer Science
Richard-Wagner-Str. 31
18119 Rostock- Warnemünde
Germany

Abstract. This paper aims at the performance evaluation of real-time computer systems consisting of a modern microprocessor architecture and a state of the art real-time operating system. Our approach is based on the Hartstone Uniprocessor Benchmark.
For the first time we have implemented all test series defined using C for real-time UNIX operating systems running on different microprocessors and architectures.
We use this implementation for a performance evaluation and a rapid analysis of real-time applications. We distinguish three methods to compare performance. The first is based on finding breakdown utilization points. At this point the system does not fulfill the required timing constraints. The second method inspects the special overload behavior beyond the breakdown utilization point. This observation shows very interesting behavior of the system under overload conditions.
Thirdly we carry out a performance evaluation on basis of a simulation of real-world real-time application.
We demonstrate the impact of different architectures on the real-time performance of a real-time UNIX operating system. Modifications have to be done to execute the Hartstone Uniprocessor Benchmark if the performance of various platforms will be considered.
We give results obtained running the Lynx Real-Time Operating system on a 80486 and a Pentium based PC.

1 Introduction

One attempt to evaluate the *overall* performance of a real-time system is the Hartstone Uniprocessor Benchmark. [1][2]. Another one covers a set of specific parameterized benchmarks and provides typical workloads for embedded control application [5].
The Hartstone benchmark is application oriented and defines some sets of requirements typical for real-time applications. Hartstone as understood by the inventors „is a system requirement rather than an implemented program“. Only a small subset of test series has been implemented in ADA [2].
The appropriate model of the benchmark 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

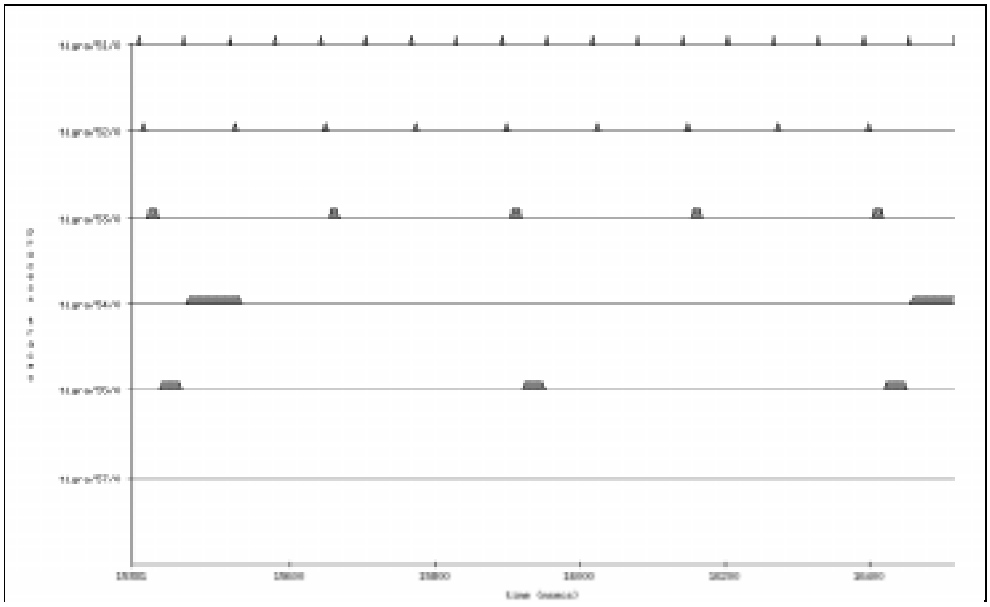
hard or soft deadlines. Aperiodic processes with soft deadlines run as background processes.
Experiments start with a baseline process set characterized by a number of processes, their priority, their synthetic load, process period, starting time and interarrival time.

2 Benchmark model

The main idea consists of increasing a definite workload up to the breakdown utilization point. For example see Figure 1: the represented figure shows the execution of five independent periodic and harmonic processes. Harmonic process means that frequencies of the processes are a multiple of the smallest frequency. On a real-time system timing constraints has to be held. In the used model the timing constraint for periodic processes is the period; each process must be executed within its period. If the workload is to high and a process is interrupted by higher prioritized processes a process can miss this timing constraint; it misses its deadline. On a real system exist a system overhead caused for example by context switches, scheduling overhead, kernel delay and times of system calls.
The Hartstone Uniprocessor Benchmark defines some experiments with different series. It considers not only harmonic and non-harmonic periodic processes but also aperiodic, sporadic and synchronization processes.
Each experiment starts with a baseline process set. A baseline process set consists of five processes at least. Each process executes a definite workload.
What kind of workload is to be executed? The used synthetic workload consists of small portions of well-known Whetstone-Benchmark. Each process executes this synthetic load in a loop according to a definition within a test description file. The benefit of this load is that granularity of the Small-Whetstone load is much fine than the Whetstone instructions in their whole.
The used workload derived from the Whetstone benchmark is measured in 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 different series of an experiment are resulting from the kind of increasing workload. Increase of workload is executed by increasing the frequency of one process, of all processes, the number of the executed synthetic workload and additional processes that has to be executed. Table 1 shows the full set of the Hartstone experiments. For derivation of the Hartstone experiments see [1].
The first series of experiments (PH-series) starts with a baseline process set consisting of five independent periodic processes with harmonic frequencies. Figure 1 shows this five periodic and harmonic processes. According to Table 1 the load of the system is increased in the different experiments and the real-time system is observed.
In experiments the breakdown utilization point is determined. This is the utilization rate where deadlines are missed for the first time. With our implementation we observe the system behind the breakdown utilization; we observe the system behavior under overload conditions.
As an another example of a series we consider SA-series of the benchmark here. This is the most complex series. Each experiments starts with a baseline process set. This series consists of periodic, sporadic, aperiodic and a server process. All the periodic processes have to synchronize during their period once with the server process. The server executes workload within and outside its critical section.

In the several experiments of the SA-series the workload is increased by

- decreasing the interarrival IAT_{sp} time of the sporadic process (SA-1)
- increasing the workload of the server within the critical section (SA-2)
- increase the workload of all periodic processes (SA-3)
- add a new process with characteristic of the third periodic (SA-4)



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-series. All deadlines are met.

Table 1: Hartstone Uniprocessor Benchmarks experiments [1]

experiments / series	expr1	expr2	expr3	expr4	expr5	expr6
PH	$T_5 \downarrow$	$T_1 - T_5 \downarrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$		
PN	$T_5 \downarrow$	$T_1 - T_5 \downarrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$		
AH	$IAT_{sp} \downarrow$	$C_{sp} \uparrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$	$IAT_{ap} \downarrow$	$C_{ap} \uparrow$
SH	$C_{s\text{-}within} \uparrow$	$C_{s\text{-}outside} \uparrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$	$n_s \uparrow$	
SA	$IAT_{sp} \downarrow$	$C_{s\text{-}within} \uparrow$	$C_1 - C_5 \uparrow$	$n_{Task3} \uparrow$		

T_i ... period of i-th process
C_i ... computation time of i-th process
C_{sp} ... computation time of the sporadic process
C_{ap} ... computation time of the aperiodic process
$C_{s\text{-}within}$ ... computation time of the server within its critical section
$C_{s\text{-}outside}$ ... computation time of the server outside its critical section
IAT_{sp} ... interarrival time of sporadic process
n_{Task3} ... number of tasks with characteristics of the third task
n_s ... number of server tasks

3 Execution of the experiments

According to the described testseries every experiment of a series starts with a baseline process set. This baseline process set is equal to all series but in the various tests of an experiment one parameter of a series is changed, and holding the other constant. The process set is specified in a test description file. At the beginning of each experiment the raw performance executed within one process is measured.
Thereafter the test description file is interpreted. It starts with the specification of the baseline process set. The following data in the test description file describe the following tests. According to the specifications processes are created and started. An 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 with 80486 microprocessor (See Table 4). Note that on various machines this process set may differ because the raw performance characteristics are different. This process set is characterized by low frequencies. It produces approximately 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 LYNX	frequency (Hertz)	workload per period (KWIPP)	workload per second (KWIPS)
1	5	10	20	1.00	512	512
2	5	10	21	2.00	256	512
3	5	10	22	4.00	128	512
4	5	10	23	8.00	64	512
5	5	10	24	16.00	32	512
Σ	5	10		31.00		2560

Table 3: Definition of baseline process set for PH-series on the reference Pentium system

	start time (sec.)	duration (sec.)	priority LYNX	frequency (Hertz)	workload per period (KWIPP)	workload per second (KWIPS)
1	5	10	20	1.00	3072	3072
2	5	10	21	2.00	1536	3072
3	5	10	22	4.00	768	3072
4	5	10	23	8.00	384	3072
5	5	10	24	16.00	192	3072
Σ	5	10		31.00		15360

Table 4: Reference systems

Processor:	i80486	Pentium
CPU-frequency	33 MHz	66 MHz
Cache	2nd Level/ 256 Kbytes	2nd Level/ 256 Kbytes
Wait States	1 Wait State	0 Wait State
Memory Size	16 Mbytes	32 Mbytes

On the reference system a raw performance of 4910 KWIPS is achieved. Thus, the baseline process set corresponds to a nominal workload of 52 %.
If this process set is executed on the Pentium system this process set is resulting in a nominal workload of 7.6 %. (The raw performance of Pentium system is 33470 KWIPS.) Using this process set on the better system would result in a different charcteristic. See for example PH-1 experiment. The influence of the highest prioritized process running on the Pentium system would be greater than in the same experiment on the 80486 system. That's why we use another baseline process set that is adapted to the performance of the better system (Table 4).

4 Interpreting the results

For *n* independent periodic tasks characterized by their period T_i and their computation time C_i, where priorities assigned to tasks in rate-monotonic order 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_i...period of the i-th process
C_i...computation time of i-th process

This theorem means if the cumulative utilization of all processes running on a single processor is less than a upper bound $n(2^{1/n} - 1)$ the processes can be scheduled by rate monotonic scheduling with guaranteed meeting of all their deadlines. Rate monotonic scheduling assigns priorities to tasks according to their frequencies: the shorter the process period, the higher its priority.
For large values of n the upper bound for utilization 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 are 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. With the selected examples of independent processes the achieved utilization exceeds 74,34 %.
Rate monotonic scheduling theory gives necessary conditions for schedulability of independent tasks.

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

This condition says that the cumulative utilization of all processes required less than 1. This condition is necessary but it is not sufficient. Thus if a process set satisfies this condition scheduling could be infeasible. In our examples the measured utilization is beyond this limit because the system overhead is excluded in the condition.
For a detailed description of appropriate scheduling theory see[3][4].

System behavior

The diagrams show the measured utilization (right y-axis) and the number of missed deadlines (left 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 only when one process executes the synthetic load. The nominal workload is the ratio between raw performance and theoretical workload the system has to executed. The theoretical workload follows from the testdescription. The measured utilization is the ratio between measured workload and raw performance. Both values are given in percent.

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. (See the utilization curve in the diagrams.)
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 prioritized process misses all its deadlines. The next prioritized process misses its deadlines after that.
The BU points of the PH-1 experiments shown in [Figure 2] are 97% for Lynx on 80486.
In the PN-experiments it is possible that both the lowest and the next following process misses their deadlines. For the PN-2 experiment the utilization is 85,5%. This value is greater than the upper bound (see (1)) because the process set is not designed to be the worst case.
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 activation corresponds to the number of activation 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. (Figure 4).
The given results are based on the measured raw performance. The evaluated real-time operating system is characterized by the same system behavior under both evaluted machines assuming the same process set characteristics.

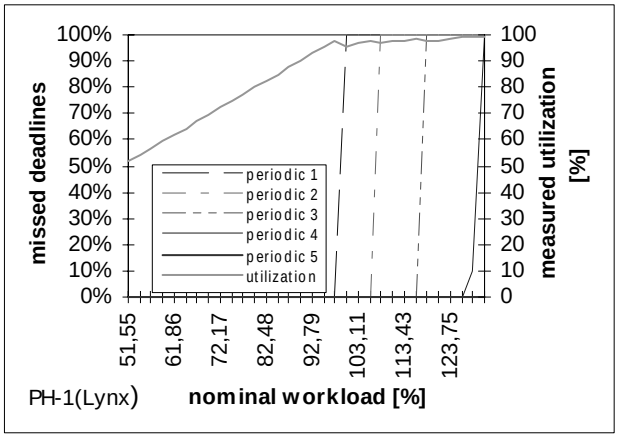


Figure 2: Utilization of PH-1 tests (Lynx-OS)

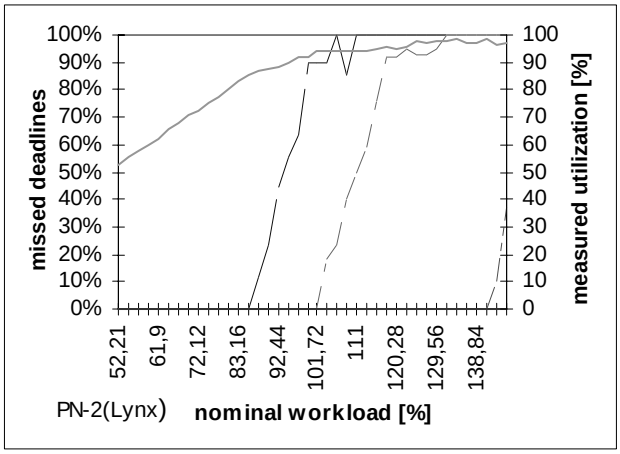


Figure 3: Utilization of PN-2 tests (LynxOS)

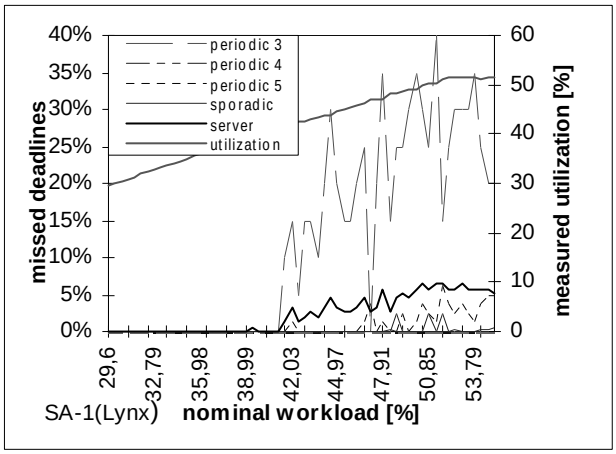


Figure 4: Utilization of SA-1 tests (Lynx)

5 CONCLUSION

In this paper, we show selected results of the evaluation of LynxOS- a real-time UNIX operating system- 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.

[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, Mai 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] Lindmeier, H., Rauh, D., Viguier M.: Benchmarking for Embedded Control and Real-Time Applications, Report, ESPRIT III - OMI, Project No. 6271, Dec. 1992