Evaluating Real Time Features of Unix Systems

Susanna Delfino and Giancarlo Succi

DIST - Università di Genova
via Opera Pia 11a
I-16145 Genova
Italia

Abstract

The aim of this paper is to give a paradigm for comparing real time UNIX operating systems and the traditional ones, mostly under the point of view of the user. Standard tools as well as specific real time ones are not suitable [2690] for these systems: the former because too generic and the latter because too specific. Therefore a new set of tests is introduced and it is applied to a wide range of architectures. The results that have been obtained are interesting and, under certain point of view, quite surprising.

Introduction

In these last years many efforts are made to adapt UNIX to the real-time world, resulting in the realization of new systems with new functionality and performance tuning. There are indications that in the future this kind of systems will be the majority, but many users tend to think of real-time UNIX as a dedicated operating system for high-level applications; they do not imagine that the real-time capabilities added to UNIX can provide better performance and reliability for all the applications. So can be interesting trace a system profile comparing this implementation and traditional UNIX with a common test [CDS91]. In this way who approaches a new machine can know, mostly under the user point of view, the supplied capabilities to use it in the best way. Actually benchmarks exist, but they need to distinguish between the traditional UNIX systems and the real-time ones. This work attempts to design a set of new tests able to stimulate some particular common situations to emphasize the different behaviours. The design is developed as a set of applications able to run in every UNIX system and that extract significant data that are analyzed and elaborated obtaining the required system profile.

In Section 2 are summarized the real-time features, while Section 3 consists in the definition of expressive parameters selected for the characterization and in a briefly tests description. Section 4 introduces the results obtained running this tests on different machines. Finally Section 5 is devoted to some evaluations and conclusions and presents the planned future research.

2 Requirements for real-time

A time-sharing operating system must share system resources equally among a number of users, while a real-time one presents a deterministic behaviour in all the situations, assuring always response-time under a threshold quite little.

The UNIX operating system can be augmented to meet this requirements adding some features [FGG*91] that are introduced in the follows:

- a priority-based preemptive scheduling so to assure that the most important process can execute first;
- a guaranteed interrupt response that is the ability to respond on both hardware and software interrupts;
- interprocess communications and synchronization facilities to exchange information among processes;
- high speed data acquisition because it must be able to optimize data storage on disk, motion of the disk heads and the data transfer from the buffers;
- 1/O support for asynchronous 1/O;
- user control of system resources such as the priority-based scheduling technique to set the priority of a user process and the memory locking allowing a user to lock a program or part of it in primary memory to avoid unwanted page-faults.

Pubblicato in AFUE 98 Conference Paris
3 Our approach in benchmarking real-time UNIX

The work plan is started with the identification of those parameters that, if well-known, can be especially expressive in order to evaluate the performance of real-time and time-sharing UNIX systems.

The first selected in this direction was the context switch latency, that is the time needed for a context switch. It seems to be indicative of the machine capability to schedule the most important process as soon as possible. This time depends greatly on the kernel constitution; in a machine with a fully preemptible kernel in fact, the worst case context switch time will be long as the longest running system call. For real-time systems this interval is usually very little \(^1\) and it is hardly measurable [ZN89], so the approach taken in this case was to consider only the worst cases. They represent the results that would not exist (failures) and are measurable with means inside the machine such as the interval timer. An application called CSW makes a context switch while many processes competing for the CPU must execute time-consuming system calls like a `open()` on a very long path [LS90] and then it measures the latency.

The results obtained in this way are analyzed and elaborated to extract comparable data on the form of percentage of value that are over the interval timer granularity. It is also possible to provide information about the assumed values.

The second test regards the functionality of the system handling asynchronous events. Also in this circumstance it must present a deterministic behaviour [Tan87], especially in a real-time framework. It is interesting to see how the signal interface of a certain implementation responds when it have to manage a signals sequence. A critical situation is for example, if more signals are sent to a process while the system is already serving another. In this case some signals are not received and so they are lost. The application dedicated to realized this is called SIGKILL and it is organized as follows:

- a C executable send two consecutive signals\(^2\) to a process and counts the number of calls to the manipulator of this signals;
- The program is repeated for many times, and the results are elaborated to extract the number of failures\(^3\);
- to collect a valid set of results the last operation is repeated for many times and is built a file containing the failures number for each cycle.

Moreover the test is also executed giving a higher priority\(^4\) to the process receiving the signals [QSP85], so the system is forced to schedule it as soon as possible.

Since the I/O reliability and flexibility are very important requirements in real-time framework, there is the necessity to know the amount of data that can be transferred in a fixed interval of time. The test called TRANSFER measures the number of Kbyte that the system can move in a certain time obtaining the throughput value for that interval. The test is iterated for growing times until the Kbyte value equals the buffer cache dimension.

Furthermore are introduced some means for computing the CPU speed. This is to satisfy a many users common demand. The tests dedicated to measure the processor performances are two; the first one is called ESTIMATE and determines a regression line using the minumun squares method estimating the time needed to execute a single cycle of a test, a decrement and a jump in assembler language. The last one is called DECREASE and produces as output the time needed to execute a sequence of cycle constitutes by a empty while loop. The result is the number of cycle executed in a second.

4 Results on real-time systems

Now are transcribed the results obtained running these tests on different real-time systems together with a briefly introduction to their main project features.

4.1 MASSCOMP RTU

The RTU real-time system of Masscomp belongs to the category of the systems that are implemented adding extension to the core of UNIX operating systems [RTU88]. It presents the following main real-time features:

- kernel preemption;

\(^1\)In normal conditions of working it is surely smaller than the system clock.
\(^2\)Precisely two SIGUSR1 signals
\(^3\)One or both the signals are lost.
\(^4\)When allowed imposing a real-time priority.
• process memory locking;
• interprocess communication and synchronization;
• real-time priorities;
• asynchronous and synchronous I/O.

For this system the test CSW has revealed that the 17% of the context switch times was superior to the threshold (10 msec.), but the differences are very little (max 0.05 sec). This is because the kernel is not fully preemptible and always exist a preemption latency.

The results of the test SIGHUP showed a remarkable difference between what happens with standard priorities and with the real-time ones. In fact with the standard priorities all the results presents a high percentage of failures (range 90-100%), while with real time priority are not failures.

With regard to the TRANSFER test the throughput obtained was been of 260 Kbyte/sec, but this value is greatly dependent by the buffer-cache dimension that is quite little (about 0.15 Mbyte).

It instead behaved quite poorly in the DECREASE and ESTIMATE tests were was outperformed by any other system.

4.2 HP-UX Version 8

HP-UX Version 8 is a Hewlett-Packard real-time operating system. It also is created adding real-time capabilities to the traditional UNIX [hp89]. They can be summarized as follows:

• priority-based preemptive scheduling;
• memory locking;
• synchronous and asynchronous I/O;
• interprocess communication and synchronization.

The test CSW for this system has showed an excellent result, because both with the standard priority and the real-time ones it have presented an uniform behaviour, because the percentage of failures is null. This is very interesting because also this implementation have not a kernel fully preemptible.

Furthermore the SIGHUP results is quite surprising: all the latency times are smaller than the threshold value because the time needed to the open()/system call is very short.

In the TRANSFER test it obtain a throughput of 4500 Kbyte/sec with a buffer cache of about 7.5 Mbyte.

In the CPU speed tests the data are still very good: it is on the first position because its results are all superior than the other one.

4.3 AIX Version 3.1

This is a fully preemptible operating system developed by IBM as a proprietary operating system [BO90]. It satisfies the main real-time requirements and, moreover it provide the possibility of dinamically extended the kernel by adding device drivers, system calls, kernel services or private kernel routines. In the CSW test it presents a percentage of 2.8% of values over the threshold value. The differences range until 0.4 seconds.

Also for this system the execution of SIGHUP with the real time priority has not failures, and this result depends surely on the fully preemptible kernel. The throughput obtained with TRANSFER is of 3000 Kbyte/sec with a buffer cache of about 10 Mbyte and the processor performances are on the mean.

5 Evaluations

The tests are developped in according with the POSIX[po88] suite and they are able to execute on every UNIX implemention. They not needed of any particular software and they have been tried in a great number of different system and machines and they have not presented problems. In this paper we presents only a part of the results we have collected also in non real-time systems, because they needed more space.

The next step is now to take in consideration also the scheduling algorithms and try new evaluation parameters.

6 Acknowledgments

The authors thank Prof. Joy Marino for his support the HP and IBM for their availability.
References


[QSP85] J. S. Quarterman, A. Silberschatz, and J. L. Peterson. 4.3BSD and 4.3BSD as Examples of the UNIX System. Department of Computer Sciences, University of Texas, 1985.


