ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Survey of Benchmarking Techniques for Real-Time Operating System Performance Analysis

P. Vetrivel[*1], S.Sivakumar[2], K. Shanmuga Sundara Babu[3]

[*1,2] Department of Electrical and Electronics, Vel-Tech Dr.RR & Dr.SR Technical University, Chennai-62, India

[3] CDAC/ACTS, Pune, India

vetrivel.est2014@gmail.com

Abstract

The recent advanced implementation Techniques of Real-time Operating System offer high performance and featured Real-time embedded systems. There are many kind of Implementation on Real-time operating Systems (RTOS) can be found at the present time. The correctness of an RTOS depends not only on the logical result of the computation, but also on the correctness of temporal parameters of the task set used in the system. So, it becomes necessary to measure the response time of real-time mechanisms to predict the performance of RTOS. In recent years, several approaches have been implemented for benchmarking the real-time parameters. In this paper we review the various benchmarking techniques for measuring the real-time parameters of RTOS. We discuss the need of measuring real-time parameters and provide a classification of the techniques on several analyses to highlight their similarities and differences. This paper is intended to help the researchers and application-developers in gaining insights into the working of Benchmarking techniques and designing even more efficient high-performance real-time operating systems of tomorrow.

Keywords: Real-time Systems, RTOS, Performance, Review..

Introduction

The general purpose operating system (GPOS) uses fair scheduling method to manage all resource allocation in the system and the CPU time allocation for every task. The goal is to achieve highest throughput by allocating as fair as possible [1]. Because of this, there is a possibility for a situation that the lower priority task will be executing when higher priority task is still waiting in queue. In RTOS, this situation should not exist. The main difference between general purpose operating system and real-time operating system is their implementation of scheduling algorithm. RTOS imposes strict task priority implementation, where a higher priority task can preempt a lower priority process which is currently executing.

There are different approaches in building a RTOS. First approach is by adding an additional function to the existing kernel in order to apply task priority strictly by means of two techniques namely build micro-kernel and patch kernel to build this additional functionality. The second approach is to build from scratch a new operating system which has real time capability. Most of this kind of operating system is specific to be used in a particular device, so the operating system will know exactly the behavior of the device. This operating system is called as embedded operating system.

There is a necessary for Research to measure and benchmark real time parameters of RTOS which is developed by using different approaches as different approach serves different performance to the application. In this paper we review different benchmarking techniques used to measure performance of RTOS.

In this paper, we highlight the need of benchmarking the performance parameters of RTOS and survey several research works which are aimed at benchmarking the real time parameters of RTOS. We believe that this paper will help the RTOS researches and RTOS designers in understanding the implementation details of RTOS and their behavior to the Real time tasks for providing the deterministic performance to the applications. Thus helps improving the methods for benchmarking the performance of RTOS in future approaches.

The remainder of this paper is organized as follows. Section II provides a background on Real-time embedded systems and also highlights the need

of benchmarking the real-time parameters of real-time operating systems. Section III provides an overview of benchmarking techniques used in Real-time operating systems. Section IV provides concluding and remarks and also discusses the future challenges in this field.

Background

An embedded system for computing task is designed for specific control functions and is embedded as part of the device which may include more hardware interfaces and mechanical parts. An embedded system performs a few pre-defined tasks, with its specific requirements. Operating system used in embedded system acts as an interface that connects and controls the activities of hardware to its user. Operating system is responsible for allocating all the available resources to its tasks and providing simple interface between user and hardware [2]. Based on their ability in handling real time task, Operating system can be divided into two kinds: Real-time Operating System and non-real time operating system (GPOS). Based on their deterministic performance RTOS is further classified into three types, Hard Real-time operating system, Firm Real-time operating system, Soft Real-time operating system.

Commercial RTOS

For data handling and communication using the VxWorks [3] platform a software framework [4] has been developed for providing common functionalities. VxWorks is widely used in physics research for several reasons, among which:

• It provides an integrated development platform, thus simplifying the development process. Programs can be developed and simulated in the host system before downloading them to the target system.

• It provides a powerful multitasking environment. Tasks have a fixed priority and can communicate via a rich set of inter-process communication (IPC) mechanisms.

• The software model of VxWorks is quite similar to that of UNIX, in particular for I/O and networking, thus simplifying software writing for developers who have experience with UNIX.

Opensource RTOS

RTAI [5], [6] and Xenomai [7]. RTAI and Xenomai share most concepts and both represent, rather than a replacement of Linux, an additional component that works in conjunction with Linux, handling the scheduling of real-time tasks and letting Linux provide all the remaining functionality. In order to co-operate with Linux it is however necessary that the underlying hardware be shared by Linux and the additional component. This is achieved in both RTAI and Xenomai by using the ADEOS nanokernel [8], [9], which acts as a broker of the hardware functionality. In particular, hardware interrupts are normally handled by ADEOS, which propagates notifications in sequence to the other components. In this case Linux and RTAI (or Xenomai) represent ADEOS domains, and are logically organized by the nanokernel as a pipeline. The component that is declared to be at the head of the pipe will receive interrupt notifications first and may then decide whether letting ADEOS propagate them along the domain pipe. In this case RTAI (or Xenomai) is at the head of the pipe and has therefore precedence over Linux, thus allowing deterministic response times regardless of the actual Linux implementation. This organization is fully achieved in Xenomai. RTAI has a somewhat different organization Instead of letting ADEOS handle all the interrupt sources, RTAI intercepts them, using ADEOS to propagate those interrupt notifications to Linux in which RTAI is not interested in (i.e., the interrupt does not affect real-time scheduling). The reason for this mixed approach is performance, because in this case, if the interrupt is going to awake a real-time task, the overhead due to the management of the interrupt by ADEOS is avoided.

Benchmarking Techniques

Benchmarking the RTOS's performance is important; because there are some cases that task execution time needs to be calculated for deterministic performance of application. To measure the quality of a RTOS, we need to understand their implementation to manage its task. Another important thing in RTOS is responsiveness, by means of which we will be able to know how fast a RTOS manage their task. A lot of researches have been done in the field of RTOS. Four performance metrics which are interrupt latency, task switching time, preemption time, and deadlock break time were analyzed in [10]. This research used one test bed application in benchmarking to assess the performance of RTOS.

K. Ghosh, B. Mukherjee, K. Schwan have compared many kind of RTOS [11]. They analyses some RTOS and its features are studied, but did not quantitatively analyze the performance of the RTOS. They discusses about the features of commercial RTOS and the comparison of commercial RTOS. J. Carbone proposes other research that discuss ways to

measure performance RTOS on the topic of testing RTOS [12]. This measurement has been applied to measure Express Logic's ThreadX RTOS.

Most research in the RTOS field is about kernel based tasks, as kernel tasks has less latency and high privileges to access kernel data structures results in good performance results. One thesis research at [13] is about the comparison of three RTOS kernel based tasks, namely RTAI Linux, Xenomai, Real-Time patch. In this research paper they compared about the real-time parameters like jitter, interrupt latency, IPC, maximum frequency; overload behavior, and priority functionality. Other study related to this kernel-based RTOS is conducted by [14]. The research is done by comparing Xenomai and RT Patch Linux to measure four metrics, task switching time, interrupt latency, preemption time, and deadlock break time.

D. Lohmann, et al. has done research on quantitative aspects of eCos [15]. In this research, analysis and test of eCos kernel are focused on runtime and memory cost aspects. Performance aspect on eCos has not been considered in this research. The research in this paper [16] tests the performance of eCos based on certain performance metrics. The test results in eCos will be benchmarked with test results on Xenomai and RT-Patch that has been done by [14]. By means of this analysis, we can compare the performance of RTOS kernel based with embedded OS. The test using data packet processing application is also conducted in addition to the four metric scenarios. By doing this the real-time application which runs on RTOS is consider for the overall performance measurement.

Levine [17] and his peers have done benchmark on context switch time and priority inversion Protocol latency of a real-time CORBA2 architecture and presented their results in their research paper. The method used to detecting and observing priority inversion is complicated in his approach. Obenland's [18] article explains another straightforward way to create priority inversion scenario. Sohal [19] has considered both the analytical and empirical approaches to measure different phases of interrupt latency of a real-time operating system. Interrupt latency is an important parameter of an RTOS to analyze the performance of different implementations.

Conclusion and Remarks

The advanced real-time systems will possess capabilities for high-speed data processing and communication which will require very high time bound processing than what is available in state-of-the-art systems. This necessitates the need of improving the real-time mechanisms in RTOS for our future need. To cope with these challenges, Very high performance is necessary at all levels of implementation application level and system level. In this paper we reviewed several Benchmarking techniques for measuring the performance of RTOS using real-time parameters. It is hoped that by providing insights into the Benchmarking techniques, this paper would help the researches in addressing the implementation challenges of RTOS for efficient real-time systems of tomorrow.

References

[1] *M. Barabanov, A Linux based Real Time Operating System, 1997.*
[2] *A. S. Tanenbaum, Modern Operating System, 2nd ed. Prentice Hall. 2002.*
[3] *Wind River home page, [Online]. Available: http: www.windriver.com.*
[4] *M. Cavinato, G. Manduchi, A. Luchetta, and C. Taliercio, "General-purpose framework for real time control in nuclear fusion experiments," Trans. Nucl. Sci., vol. 53, pp. 1002–1008, 2006.*
[5] *RTAI Home page, [Online]. Available: http:// www.rtai.org.*
[6] *P. Cloutier, P.Mantegazza, S. Papacharalambous, I. Soanes, S. Hughes, and K. Yaghmour, in DIAPM-RTAI position paper, Nov. 2000, RTSS 2000—Real Time Operating System Workshop, 2000.*
[7] *Xenomai home page, [Online]. Available: http://www.Xenomai.org.*
[8] *ADEOS home page, [Online]. Available: http://www.adeos.org.*
[9] *KarimYaghmour Opersys Inc., Adaptive Domain Environment for Operating Systems, 2001. [Online]. Available: http://www.opersys.com/ftp/pub/Adeos/adeos.ps.*
[10] *P. Feuerer, "Benchmark and Comparison of Real-Time Solutions Based On Embedded Linux". Hochschule Ulm, German : 2007.*
[11] *K. Ghosh, B. Mukherjee, K. Schwan, "A Survey of Real-Time Operating Systems". Technical Report Nr : 1994.*
[12] *J. Carbone, "Measuring Real-Time Performance of An RTOS". Express Logic, inc. San Diego, CA : 2003.*
[13] *P. Feuerer, "Benchmark and Comparison of Real-Time Solutions Based On Embedded Linux". Hochschule Ulm, German : 2007.*

ISSN: 2277-9655
Impact Factor: 1.852

[14] *I. S. Wijayanto, "Analisys and Benchmarking Real-Time Performance in Linux Based Operating System: Xenomai and Real-Time Patch". Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung. Bandung : 2008.*

[15] *D. Lohmann, et al. "A Quantitative Analysis of Aspects in The eCos Kernel".ACM Journal:2006.*

[16] *On Performance of Kernel Based and Embedded Real-Time Operating System: Benchmarking and Analysis.*

[17] *D. Levine, S. Flores-Gaitan, C. D. Dill, and D. C. Schmidt, "Measuring OS Support for Real-Time CORBA ORBs", in 4th IEEE International Workshop on Object-oriented Real-Time Dependable Systems 00', Santa Babara, California, Jan. 27-29.*

[18] *K. Obenland, "Real-Time Performance of Standards Based Commercial Operating Systems"*

[19] *V. Sohal, "How To Really Measure Real-Time", Embedded System Conference, Spring 2001*