Hard Real Time Task Execution on Virtual Hardware Using EDF Scheduling Policy

Dr. Ramesh S¹, Shanmukh D. M¹
Associate Professor¹, M. Tech Scholar²
Department of ECE
Dr. Ambedkar Institute of Technology, Bengaluru, India

Abstract:
The electronic waste is increasing as years go by, currently e-waste generated by India every year is 92 thousand tons. There is a prediction that the e-waste will reach 250 thousand tons by 2020[1] unless steps are taken to reduce it. There are multiple ways to reduce it. One of the methods is to virtualize the hardware. All the hard-real time tasks in today’s technology need hardware for execution. This leads to increase in e-waste. Hard real time task can be implemented on a virtual hardware if the overheads and latencies of the tasks executed in virtual machine is close enough to overheads and latencies values of tasks executed on hardware value. In this paper, EDF scheduling policy is used to schedule hard real time tasks, overhead and latencies are presented and method of computation of real time overhead and latencies are also discussed.

Keywords: EDF, Overhead, Latency

I. INTRODUCTION

Today, there is vast improvement in the field of electronics. One of the major contribution of electronics is computers, today's computers have multicore processors. Classic models policy was to run one operating system in one core. Multicore processors/systems have made was for multitasking, execution of real time task which will have a dedicated core, while rest of the cores will manage general purpose tasks of system. Due improvement in the components of system today's systems can run multiple operating system on one hardware, this is achieved with the help of virtualization technique. Virtualization uses a secure virtual machine, each operating system is running inside it.

Virtualization technique exploits multicore system to full potential. Virtual machine is virtual hardware which is required to run an operating system. However, operating system running in virtual machine can be compared to operating system running on hardware has impacts on timing of kernel.

In (Mehdi A) [2] overhead and latency of tasks executed on operating system (OS) running on hardware as well as on operating system running which is hosted virtual machine were compared. The authors executed tasks under 3 cases, best case, Average case and worst case. The results were, best case measurements were almost similar in both cases. Average cases values had a slight variation. Worst case measurements were found to be almost double. They conclude that if values of overhead and latency in worst case is made within the tolerance value then real time tasks can be executed in virtual machine and exploit full potential of system cores.

This looks for the possible ways to reduce the values. Initially started with the operating system, Ubuntu was chosen as the guest operating system on virtual machine, windows operating system as host which runs virtual machine on it. Next thing that could reduce values was scheduling policy. Earliest Deadline Policy was chosen.

In the reminder of this paper, in section 2 previous proposals are discussed, in section 3 proposed work are discussed, in section 4 detailed work is discussed. Finally, conclusion, advantages, disadvantages and future scope for implementation of this concept is discussed.

II. PREVIOUS WORKS

In (Fabrice B) [5] Portable dynamic translator was used for faster emulation. It was found that several different CPUs could be emulated using this emulator. Complete system could be emulated using this emulator.

In (Nils F) [6] investigations of previous attempts that resulted in failure, analysis of those attempts which were in early stages of development were conducted. Best solution among them ever taken for practical implementation. Experimental issues lead to failure of work.

Combination of virtual environment and Linux was given [5]. Virtualization starts to play a major role in real time embedded systems at days go by. The author has used KVM (Kernel based Virtual Machine) and QEMU (Q-Emulator) as virtual environment to run Linux.

The authors have scheduled hard real time task set on partial and full reconfigurable platform [7]. Based on the results obtained by them, they conclude that resource utilization is high and task rejections is very less.

The authors have scheduled soft real tasks in EDF scheduling policy under multiprocessor environment [8].

The authors have selected multicore and in that one core is used to run RTOS and rest of it is used to RUN GPOS. The experiment was concluded with results that it can be used in automobile if hard real time constraints are meet [9].

The author has spoken about execution of hard real time task in virtual environment. The challenging task to know if task is executed or not. It will be easy to know the where about of tasks if it is running in one physical server having multiple cores [10].

III. PROPOSED WORK

Future enchantments suggested in (Mehdi A) [2] paper which was presented in SIES2013, was considered base for this work. The suggestions that were made:

- Consider different VM (virtual machines) for code execution.
• Use different scheduling policies to execute the code.
• Try to close down the overhead and latencies value got when code was executed on VM to hardware.

A. Concept of virtualization
Before exploring the first point, let us first try to understand what virtualization is.

The concept of virtualization started in 1962 by general electicals. Initially virtualization concept was just limited to virtual memory. IBM was the first company to roll out a virtual memory concept based system[3] [4]. Virtual machine concept came into picture later in 1980's. IBM is the major contributor for the development of virtual machine. Virtual machine is a software which is loaded on top of OS which is running on hardware. VM is used to host an OS without any additional hardware (H/W hosted VM is sufficient).

B. What is need of Virtual Machine in this work?
Virtual machine is one of the methods that will be a common thing in future electronic systems. The growth of the electronic systems as years go by is very huge, systems are not used to full potential as cores incorporated in the system is increasing. Not only cores, but all the other hardware components in the system, one of the methods in which we can exploit the optimum potential of the hardware is by incorporating multiple systems on one hardware, by doing so there are many advantages hardware usage reduces, hardware can we to its full potential. Reduction in usage of hardware indirectly contributes to reduction of electronic waste. India alone contributes 92 thousand tons currently; it is estimated that if steps are not take to reduce it by 2020 it will grow to 250 thousand tons. One of the best solution is virtual machine.

C. Methods to Implement Virtual Machine
The vintage approach of implementation virtual machine was virtual machine monitor was placed on top of hardware, and virtual machine is to fit it on top. VMM was having highest privilege and guest operating systems had lower privileges. VMM had control and could intercept and run guest operating systems which needed to interact with hardware.

There are many different implementation methods for virtual machine. One of the best and most common way of implementing it was, to run it on a host operating system. These methods of implementation are called as hosted VM. In the figure 1 below different implementation methods are shown.

![Fig. 1. Different methods of implementation of VM](http://ijesc.org/)

D. Operating system hosted by virtual machine and hardware
In this work we have used RT-Linux. RT-Linux is a flavor of Linux. Linux is open source operating system which was Linux Torvalds is the person behind Linux. The purpose of this initially was as General Purpose Operating System (GPOS). As the Operating system was open source anyone in the world could improve it and sell it. Linux is the most vastly explored operating system. People improved the Linux operating system so that it could be used as Real Time Operating System (RTOS).

The reason behind using Linux as OS in this work is, there are many versions of the Linux. The versions vary based on application and usage. RT-Linux is open source flavor of Linux. It follows non-preemptive based scheduling.

E. RT-Linux
As we mentioned earlier, RT-Linux is an open source operating system which is flavor of Linux. The purpose of RT-Linux operating system is to serve real time applications.

F. RT-Linux additional features
• RT-Linux can be run on 15 different architecture
• Linux operating system (OS) can access files in windows OS
• 6 different run level modes (1-3,5,6)
• It is a multiuser OS,7 users can log in parallel at once (6 CLI and 1 GUI)
• It uses GRUB (grand unified boot loader) to load OS
• Linux OS has mono kernel
• Mono kernel: - single large process running entirely in single address space.

G. Windows
The host OS in this work is Windows. Windows is a very famous and popular OS. Windows is a GUI (graphical user interface) OS. Windows is not an open source OS. Windows OS is having kernel which is combination of mono and micro kernel which is called hybrid kernel. Windows 10 is the latest version of windows operating system. The features of it are
• Touch based user interface
• Multitasking with the help of virtual window
• 21st century command prompts
• Continuum (automatic conversion from keyboard to touch when keyboard is removed)

H. Electronic Control Unit (ECU) Automoblies
Today’s automobile is being included with embedded of the user-friendly interaction with the automobiles. With just one press of switch the automobile can be changed into sport purpose, for performance, city usage automobile. The problem in the car can easily be found, this is all possible due to inclusion of embedded systems into the automobile.

The main idea behind this work is to introduce virtualization into automobile ECU (Electronic Control unit), it is just like a CPU in system. Existing ECU’s are implemented using just hardware.

If overhead and latency of hard real time task in virtual machine is made close to hardware. Virtualization can be implemented in automobiles.
I. Advantages of EDF Scheduling policy

If EDF scheduling policy is used then, there are advantages:

- Once task starts executing there is a guarantee that it will be completed within the deadline.
- EDF is Non-preemptive based scheduling.
- Overhead of the tasks are very low compared to other scheduling policies.
- If task can’t be completed within the deadline task is not taken up for execution.
- Other tasks are not affected due to task failing to meet the deadline.

J. Advantages of Linux and Windows

Linux is taken up as guest OS in our work as it is open source OS, and can be modified to meet the needs of application. Windows is taken up as host OS in this work as its one of the most sophisticated OS, it is getting updated continuously with the improvements in the computer world. As it’s not an open source OS, completely details are not available, due this this reason we can say it is a secure system.

IV. H/W S/W AND SCHEDULING POLICY USED FOR COMPUTATION OF LATENCIES

- Scheduling overhead is the time taken to perform a process selection.
- Event Latency is the delay from the raising of the interrupt signal by the hardware device until the start execution of the associated interrupt service routine (ISR).

A. Hardware and Software Platform

Windows OS was loaded on top of 2.2GHz Intel hardware platform. Then VM workstation was loaded on top of the Windows OS. VM workstation hosted Linux OS. PREEMPT-RT patch was added to Linux kernel to convert Linux to RT-Linux.

B. EDF Scheduler

Functioning of the algorithm used in this work, as the name itself implies, task with the earlier deadline time available is selected for execution and it is executed. It is required that the deadline of each task is known beforehand. EDF will finish tasks according to deadline time and it is optimal if tasks arrive sequentially. If we have tasks A (2 execution time, 5 deadline), B (3, 10), C (1, 3), D (5, 6), E (1, 1) and they are all available from the start the execution order will be: E -> C -> A -> B, task D will not be executed. If they become available one by one at different times, a newly arrived task will not interrupt a running task if it is having earlier deadline than the running one.

C. VM workstation (Virtual Machine) used

VMWare is one of the VM which is made of windows OS. VMware workstation was preferred in this work as it having user friendly interface, it one of the best software in the field of VM. VMware Workstation is a hosted hypervisor that runs on x64 versions of Windows and Linux operating systems (an x86 version of earlier releases was available); it enables users to set up virtual machines (VMs) on a single physical machine, and use them simultaneously along with the actual machine. Each virtual machine can execute its own operating system, including versions of Microsoft Windows, Linux, BSD, and MS-DOS. VMware Workstation is developed and sold by VMware.

D. Realtime scheduling

In this work, real time scheduling used, the code should simulate real-time scenario. In the real-world tasks, can arrive at any point of time and all tasks are hard real time tasks. The scheduling of real-time systems requires some attention compared to that of normal systems. Here the highest concern is to make sure that each task and process gets enough execution time and resources to complete its goals and, most importantly, that they can complete before their deadline. In order to accomplish this each task’s requirements must be known before they can be scheduled. There are two ways to do this: Dynamic and static scheduling. Dynamic scheduling decides how to schedule tasks at runtime, making decisions based on current requirements and the state of tasks. Static scheduling works the opposite way: Scheduling decisions are made before execution, during compilation of the scheduler. The prerequisite is that data about the tasks that will run on the system must be collected before the scheduler can be built. This is optimal for systems whose behavior remains the same, since no changes to the scheduling can be made once the system is operational.

V. RESULT

In this section, virtualized RTOS evaluation is presented on two aspects overhead and latency. Here tasks are generated and latencies of tasks which are executed as per the EDF scheduling policy within deadline are calculated and added together.

In the code, which was used for schedule tasks, 5 tasks were created, deadline and execution of each task was determined during its creation. The latency of tasks was calculated using the formula:

\[ \text{LATENCY} = \text{TTE} - \text{SETET} \]

TTE: - Total Time for Execution
SETET: - Sum of Execution Time of Each Task

Deadline were generated using the following functions:

\[ \text{I} = \text{rand (PID)} \]

\[ \text{Snand used PID (process ID) each time to create random deadline which is similar to real-time scenario. Each deadline value was allocated using rand function.} \]

In this work tasks were considered for execution under 3 different cases, case 1 tasks were executed only in guest OS (Linux), case 2 tasks executed in both host (Windows) and guest (Linux) OS’s, and in case 3 also tasks are executed on both host and guest OS’s, but tasks executing in host OS are more compared to case 2.

Case 1 tasks are only executed in guest OS, four tasks arrive requesting for execution in guest OS (Linux), based on EDF (earliest deadline first), executable tasks are executed.

After execution based on EDF policy in this work. The following

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Execution Time (ms)</th>
<th>Deadline (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>
tasks were executed, first task to enter scheduler was tasks B, followed by task C and D, according to EDF policy task A can’t be executed within the deadline, so it is not taken up for execution.

Case2 tasks are only executed in guest OS, four tasks arrive requesting for execution in guest OS (Linux), based on EDF (earliest deadline first), executable tasks are executed. 

After execution based on EDF policy in this works. The following tasks were executed, first task to enter scheduler was tasks B, followed by task C and D, according to EDF policy task A can’t be executed within the deadline, so it is not taken up for execution.

Case3 tasks are executed in both in guest and host OS, four tasks arrive requesting for execution in guest OS (Linux), based on EDF (earliest deadline first), executable tasks are executed. 

After execution based on EDF policy in this works. The following tasks were executed, task A was only task to enter the scheduler, according to EDF policy tasks B, C and D can’t be executed within the deadline, so it is not taken up for execution.

Case1: - Task B execution was completed at 2 ms, followed by task C which completed execution at 4ms, and finally task D completed execution at 9ms. 
Total latency was found to be 0.32ms

Case2: - Task A execution was completed at 2 ms. 
Total latency was found to be 3.1ms

Case3: - Task C execution was completed at 3ms, followed by task D which completed execution at 3ms, and finally task B completed execution at 2ms. 
Total latency was found to be 5.73ms

VI. CONCLUSIONS
E-waste generated can be reduced by half using half the electronic circuits used to implement the same functionality in cars electronic control unit. Tasks in this work was executed in 3 cases and latencies during the execution of tasks were calculated successfully. The benefits of the proposed method are that if task is taken up of execution, there is a guarantee that it will be completed within the deadline and overhead generated during the execution of tasks are negligible. The trade-off being that as its priority, independent, higher priority tasks may be delayed or will not be executed. In the below figure number of tasks rejected during execution in different scheduling policies is shown.

Furthermore the Latency values obtained can be reduced and the ISR (Interrupt Service Routine) can be used to calculate latencies during execution.

REFERENCES