Research on Real - time Scheduling Method of RTAI - Linux Based on EDF Algorithm

Ma Yue[2], Zhao Yue-Qi[1,2], Yin Zhen-Yu[2]
1(University of Chinese Academy of Sciences, Beijing 100049, China)
2(Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)
*corresponding author's email: congmy@163.com

***Abstract*—This paper uses Linux4.2.0 and RTAI3.8.13 to form a dual-core solution to study the real-time performance of Linux systems. This paper studies the basic structure of RTAI-Linux kernel. On this basis, the ADBORROW algorithm is used to improve the time slice rotation strategy of EDF dynamic scheduling algorithm. Assign tasks to RTAI-Linux systems based on dual-core x86 architectures and perform real-time performance testing and analysis of new task scheduling algorithms. The scheduling model can make the real-time performance of RTAI-Linux system be further improved.**

Keywords- Fingerprinting; RTAI-Linux system; Real - time task scheduling; ADBORROW algorithm Introduction

I. INTRODUCTION

As a time-sharing system, Linux can not be directly applied to industrial production in real-time requirements of the higher environment. In this paper, RTAI and Linux dual kernel architecture, the task of high real-time requirements of Linux into the RTAI ultra-kernel space to run, the priority set to the highest, the normal task into the normal operation of user space. However, on this basis, the system real-time there is still room for further improvement. Based on the existing EDF dynamic scheduling algorithm, the time slice borrowing strategy is introduced. When the system judges that the current task has the highest priority and the next round of rotation has exceeded the deadline, the current CPU is not discarded and some of the time slices are borrowed from the next cycle until the current task continues. The use of dual-kernel mechanism and improved EDF algorithm, can make the Linux system real-time on the basis of the original, access to a substantial increase to meet the requirements of most industrial systems.

II. RTAI-LINUX ARCHITECTURE AND EDF DYNAMIC SCHEDULING ALGORITHM

RTAI is a hard real-time extension of the Linux kernel, using the ultra-kernel design method, through the ADEOS ideas to achieve. ADEOS in the bottom of the operating system to insert a micro-kernel, each operating system running on the micro-kernel, the various operating systems can operate control hardware. Adeos refers to the various operating systems under its jurisdiction as domain, the RTAI system itself as a domain in Adeos, in the Adeos management to achieve the RTAI system initialization, interrupt the application, interrupt service routine registration [2,4]. According to the Adeos scheduling scheme, the RTAI priority is set to the highest level, and the Linux task is run only if no RTAI task occupies the CPU.

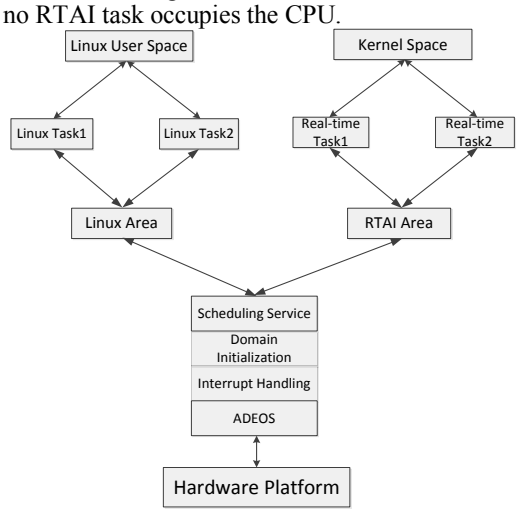

Figure 1. Rtai-linux system structure

In the given task set n = {1, 2, ..., n}, if and only if the CPU utilization of the task satisfies the following condition, as in equation (1) [5]:

$$\sum_{i=1}^{n} \frac{e_i}{p_i} \le 1 \tag{1}$$

T_i - the first i real-time task;

e_i - the execution time of the task T_i;

p_i - the cycle of the task T_i;

n - the number of tasks;

The task set is feasible with EDF algorithm scheduling. $\sum_{i=1}^{n} e_i / p_i$ refers to the workload of the task set, EDF as a dynamic scheduling algorithm, the need to respond in a changing environment, predictability is poor and running a larger cost, there is a certain process to miss the deadline.

The disadvantage of the EDF algorithm is that if a real-time task has completed the work in this cycle ahead of the allocated time slice, the rest of the free time and the rest of the real-time tasks can not use the CPU will cause the processor to be idle ; If a real-time task is not finished when the time slice is running out of work, it will keep the current state to continue, which will lead to the subsequent follow-up tasks have been postponed to form a chain effect,

978-1-5386-1230-9/17 $31.00 © 2017 IEEE
DOI 10.1109/ICICTA.2017.20

resulting in multiple tasks beyond the deadline, System performance, can not meet the real-time requirements [6].

III. IMPROVEMENT BASED ON EDF PRIORITY SCHEDULING ALGORITHM

Based on the above shortcomings, based on the EDF scheduling algorithm to improve, we introduce ADBORROW (ADVANCE BORROW) algorithm [9]. ADBORROW idea is to assign real-time tasks to several levels of service level, each server in accordance with a certain proportion of CPU, if a server on the real-time task overrun, the next cycle of time using the borrow strategy, A time slice of the rotation to the current use, the next round of the use of time automatically push back In this way, you can reduce the impact on other real-time tasks to a certain extent, to reduce the deadline miss rate DMR (deadline miss ratio) have a certain effect.

The ADBORROW scheduling algorithm is based on the following assumptions:

(1) high priority tasks can preempt resources for low priority tasks;

(2) the cost of preemptive process can be ignored;

(3) only CPU resources are competitive, memory, I / O and other resources are sufficient;

(4) there is no binding between tasks;

(5) all tasks in the task set are periodic;

(6) the relative time limit of the task is equal to its cycle.

As shown in Figure 2, the three ready tasks T1, T2 and T3, where the task T1 M = 2, N = 5, P = 6; task T2 M = 2, N = 6, P = 7; M = 2, N = 7, P = 8. Task T1 exceeds the budget of 0.5 time slices, the remaining part of T1 is delayed until time 6 to start, has timed out. At this time using the cycle borrow strategy, as shown in Figure 3, the task T1 from the next cycle to borrow 0.5 time slice, continue to complete the current task, and then run T2, T3. So that three tasks are not overdue [6]. (M: Time Slice Length; N: Deadline; P: Task Length)

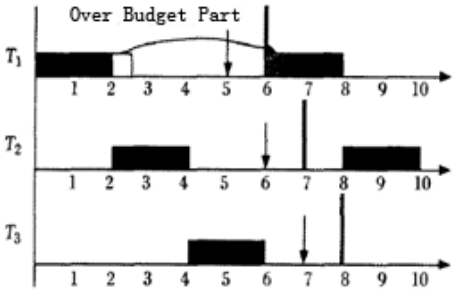

Figure 2. No free time slice

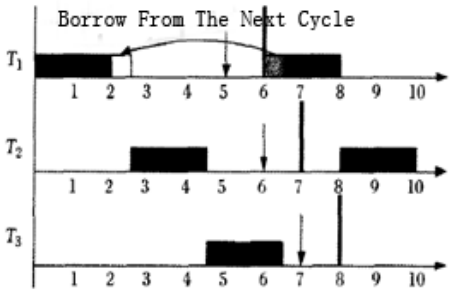

Figure 3. Borrowing strategy

IV. IMPROVED IMPLEMENTATION OF ADBORROW ALGORITHM

For ADBORROW test design, we use one-shot timer to simulate the operation of real-time tasks, the use of periodic timer to simulate the hardware clock.

The periodic timer updates the priority of the task and triggers the scheduler at the time of arrival of the execution cycle of the task and the arrival time of the deadline. The design of the C language to build a structural variable, the application is ready and suspend the structure of the two queues will be sent to the scheduler, trigger scheduling [6].

ADBORROW algorithm priority calculation formula(2):

$$priority = \lfloor (d_{s,k} - t)/p \rfloor \quad (2)$$

$d_{s,k}$- the deadline for the s and k real-time tasks;

t - current time;

p - the cycle of the task;

Implementation of the flow chart shown in Figure 4:

(1) according to ADBORROW algorithm, the queue in the ready task in real-time update;

(2) When the task scheduling time comes, select the task queue in the highest priority task and running the task to compare the priority;

(3) If the priority of the ready queue task is higher, the next scheduling time is set to the preemption time; otherwise, the time slice borrowing strategy is used to borrow from the next time slice of the task until the task is completed.

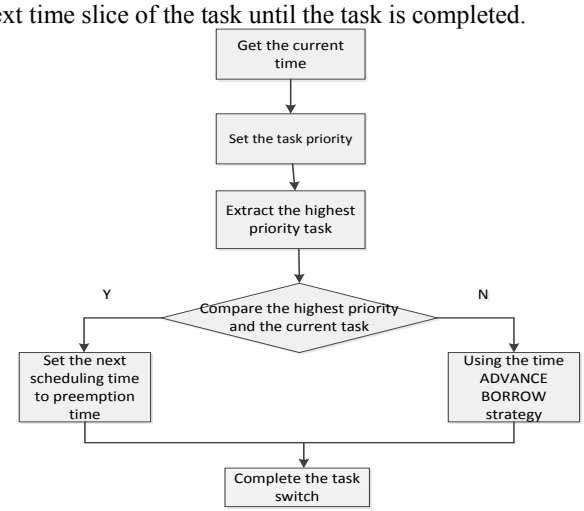

Figure 4. Algorithm flow

V. SIMULATION TEST AND REAL - TIME ANALYSIS

5.1 system test environment

As the test results are closely dependent on the test environment, some important hardware and software parameters [8], as follows:

```
CPU :Intel I5 2450M 2.5G Hz
Mainboard :ACER VA50_HC_CR
Cache :256KB
RAM :8GB
Opration System:Ubuntu14.04
X server :4. 3. 0 , VIA8606 Integrated Savage 2D
/3D
video A ccelerator , 4XAGP and 64-bit engine
GCC :4.2.0 (Linux 4. 2. 0-27), -O2 , libc-2. 3.
```

5.2 Simulation test

Based on dual-core x86 architecture RTAI-Linux system, do multiple simulation tests.

By using the RTAI comes with the test program, you can display the task switching delay and average, output the results every second [7]. The specific demonstration process is as follows:

```
##RTAIlatencycalibrationtool##
#period=100000(ns)
#avrgtime=1(s)
# check overallworst case
# do not use the FPU
#start the timer
#timer_mode is oneshot
```

The positive and negative values of the RTAI kernel patch are slower than expected, and negative values are opposite. In the test results, the average is negative, indicating that the task switch delay is shorter than expected, that is, RTAI real-time requirements.

5.3 Test results

According to the test results, the system delay chart of Fig. 5 and Fig. 6 is plotted as follows:

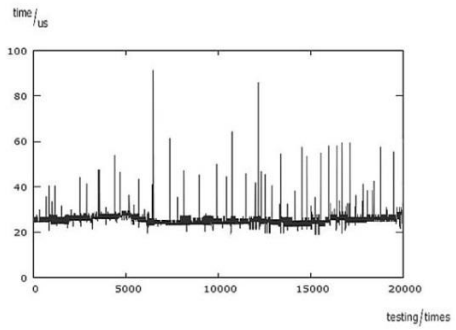

Figure 5. RTAI-Linux system latency

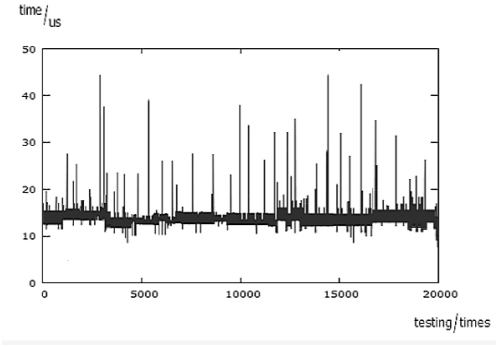

Figure 6. System delay for improved algorithm

5.4 Real-time analysis

There are two important indicators that affect the real-time performance of a real-time operating system:

(1) task switching time

The time the task is toggled depends on the number of registers that the CPU needs to wait for the stack. The longer the number of CPU registers, the longer the switching time. Task switch state, as shown in Figure 7:

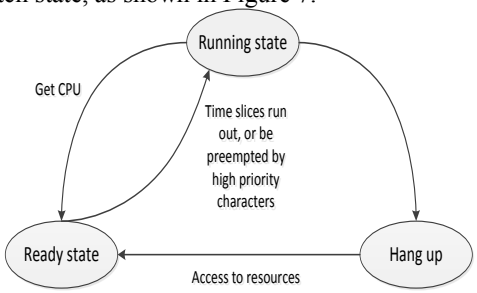

Figure 7. Task Scheduling Statechart

(2) Interrupt response time (maskable interrupt)

Interrupt Response Time = Longest Interruption Time + Time to Protect CPU Internal Register + Execution Time of Interrupt Service Function + Start Execute the first instruction time of the interrupt service routine (ISR).

Table 1. Typical values for important indicators of system real-time performance

	VxWorks	uC/OS-II	RT-Linux3.8	QNX6
Hardware Platform	MC68000	33MHz-486	60MHz-486	33MHz-486
Task Switch	3.6us	<9us	15us	12.8us
Interrupt Response	<3.4us	<8us	16us	7.4us

Through our current more commonly used four real-time operating system testing and access to relevant information [3], as shown in Table 1 shows the most important real-time impact of several key factors of time. Due to the different test methods and platforms, there may be a slight error. Among them, RTAI-Linux system delay time and test results roughly match. After testing, the delay of 30us to 15us or so.

VI. CONCLUSIONS

The ADBORROW algorithm improves the real-time response of EDF to a certain extent by using the comparison

of priority and the time slice borrowing strategy, which is disadvantageous to the low utilization rate of idle time slice of EDF algorithm. Can further meet the real-time requirements of higher embedded systems.

ACKNOWLEDGEMENTS

This work is supported by National Science and Technology Major Project—Core Electronic Devices, High-end Generic Chips and Basic Software (Grant Nos. 2017ZX01030—201).

REFERENCES

[1] Mo Yi-Jian. Hard real-time and soft real-time differences.[J]. Computer Programming and Design Patterns, 2011

[2] Wang Xiao-Ke. Real - time LINUX Analysis and Implementation Based on RTAI. Xi'an Railway Vocational and Technical College

[3] Wang Wei,Jiang Bin. Real - time Analysis of Four Real - time Operating Systems. [J]. The world of electronic products.

[4] A. Barbalace, A. Luchetta, G. Manduchi, M. Moro, A. Soppelsa, and C. Taliercio. Real Performance Comparison of VxWorks, Linux, RTAI and Xenomai Systems in Real Real - Time Application Environment

[5] Xiao Wei,Feng Yi-Bao,Ying Qi-Jia. Research and Implementation of Improved EDF Scheduling Algorithm. [J]. Computer Engineering. 2009

[6] Yu Zu-Feng,Cai Qi-Xian,Liu Ming. EDF scheduling algorithm real-time improvement. [N]. Journal of Guangxi Institute of Technology.2010

[7] Liu Yu,Fan Wei-Hua,Luo Zi-Ming. Real - time transformation and testing of Linux [J]. Theory and Research

[8] Chu Wen-Kui,Zhang Feng-Ming,Fan Xiao-Guang. Research on Real - time Performance Test of Embedded Linux System [J]. Systems Engineering and Electronics.2007

[9] Lin Caixue,Scott A.Brandt. Improving soft real-time performance through beter slack reclaiming[C].26th IEEE Real-time Systems Symposium