Improving Real Time Performance of Linux System Using RT-Linux

IOP Conf. Series: Journal of Physics: Conf. Series **1237** (2019) 052017 doi:10.1088/1742-6596/1237/5/052017

Improving Real Time Performance of Linux System Using RT-Linux

Cailing Wang[1], Fan Yang[1], Hongwei Wang[2], Pu Guo[1], Jiale Hou[1]

[1]Xi'an Shiyou University, Xi'an 710065, China

[2] Engineering University of CAPF, Xi'an 710086，China

azering@163.com

Abstract： The Linux operating system is a general-purpose operating system, and its serious lack of real-time performance limits its development in the embedded field. This paper analyses the current development status of Linux system, and uses RT-Linux patch to improve its real time activity. In order to test the real-time differences between RT-Linux and Linux, the RT-test test tool is employed. The experimental results show that the Linux kernel optimized by RT-Linux patch consumes less delay than the Linux kernel in multi-thread scheduling, which fully verify the real-time optimization of RT-Linux patches.

1. Introduction

As a public embedded system, the Linux operating system has many advantages such as open source, free, and support for multiple architectures [1]. However, Linux's process scheduling algorithm mainly considers fairness. That is to say, the scheduler allocates the available resources as evenly as possible to all processes that need the processor, and ensures that each process can run. However, this design goal is contrary to the needs of real-time processes. Its non-preemptable kernel, coarse clock granularity, frequent shutdown interrupts, virtual memory and other buffering mechanisms make Linux not provide strong real-time performance [2]. With the development of mobile terminal technology, embedded devices require operating system Linux to have higher real-time characteristics.

At present, there are two main methods for real-time improvement of Linux system. One is to directly modify the Linux kernel. This method can make the Linux system meet the requirements of general soft real-time [3, 4], and the other is the dual-core idea. That is to add a special real-time kernel based on the original Linux kernel, responsible for the management and scheduling of real-time tasks. The original Linux is the lowest priority task of this real-time kernel. Linux modified by this method can achieve hard real-time requirements [5, 6]. Currently, there are off-the-shelf patches based on the above two methods, such as the kernel-modified Preempt RT kernel patch, which implements the dual-core RT-Linux kernel patch. The transformation of the Linux kernel can make full use of the various system calls of the original Linux kernel, and is more suitable for Linux real-time testing and development on the existing platform [7]. Therefore, in this paper, we select the RT-Linux patch to enhance the real-time nature of the Linux kernel.

The structure of this paper is organized as follows: In section 2, we introduces the RT-Linux kernel patch. In section 3, we complete the configuration of the patch and designs the test experiment to verify the real-time performance.

2. RT-Linux

RT-Linux (Real-Time Linux, also known as Real-Time Linux) is a real-time and embedded patch developed by fsmlabs (finite state machine labs) in New Mexico, USA. It is an extension of Linux in real-time, using a patented dual-core technology: a tiny RT-Linux kernel runs the original Linux kernel as a thread when it is idle. This opens up new ways to run different programs at two different kernel levels (real-time RT-Linux kernels and commonly used non-real-time Linux kernels), and the original Linux kernel accesses the hardware through the RT-Linux kernel [8]. RT-Linux transforms the Linux kernel and makes some changes to the Linux kernel working environment, as shown in Figure 1.

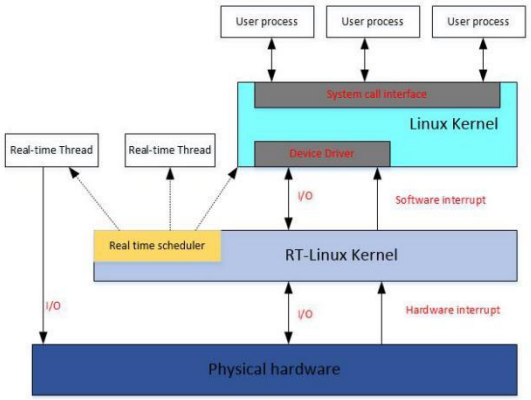


Fig.1 RT-Linux kernel principle

RT-Linux runs the standard Linux kernel (or sub-kernel) as the lowest priority thread in a simple real-time operating system, bypassing the Linux kernel performance. This means there are two separate sets of APIs, one for the Linux environment and one for the real-time environment. The RT-Linux program runs in both user and kernel mode space. RT-Linux provides an application interface. These API functions are used to write the real-time processing part into a kernel module, which is loaded into the RT-Linux kernel and runs in the kernel state of RT-Linux. Applications that are not real-time are executed in user space under Linux [9]. RT-Linux takes over the outage entirely through an efficient, preemptive real-time scheduling core and runs Linux as one of the lowest priority processes in this real-time core. RT-Linux runs real-time tasks when there are real-time tasks to process; RT-Linux runs non-real-time processes of Linux when there are no real-time tasks.

RT-Linux adopts a priority scheduling policy by default, that is, the system scheduler determines the order of execution according to the priority of each real-time task. High priority first execution, low priority post execution, thus ensuring rapid scheduling of real-time processes. At the same time, RT-Linux also supports other scheduling strategies, such as the shortest time first scheduling (EDP) and the determined periodic scheduling (RM) (the real-time tasks of the periodic segment have high priority). RT-Linux design the task scheduler itself as a loadable kernel module, users can write their own scheduling algorithm according to their actual needs [10].

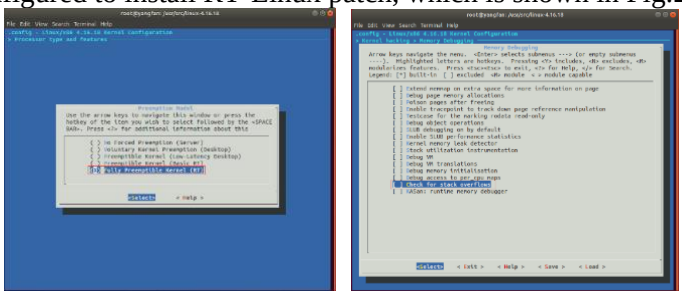
For an operating system, the precise timing mechanism can improve the efficiency of the task schedule, but it increases the time overhead for the CPU to handle timed interrupts. RT-Linux compromises the time precision and time overhead of clock interrupt processing. Set the timer chip to the terminal timing interrupt mode. The timing interval of the timer is constantly adjusted according to the time requirements of the most recent process. This not only achieves high timing accuracy, but also minimizes the overhead of interrupt processing.

3. Real time performance test by using RT-Linux patch

3.1 Experimental environment configuration

The experiment used Ubuntu18.04.1 system as the test system. The system Linux kernel is version 4.16.18, and the RT-Linux version is selected to correspond to the Linux kernel version.

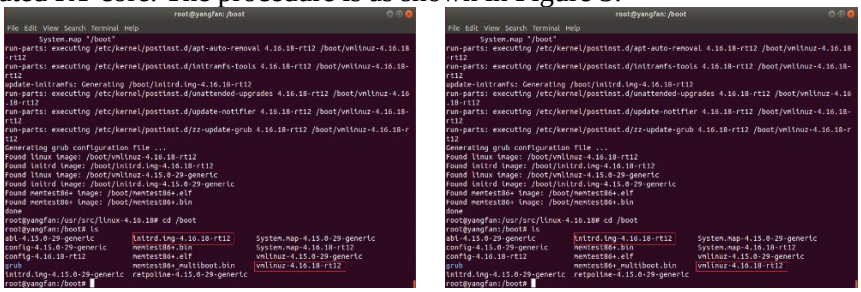
Linux kernel is configured to install RT-Linux patch, which is shown in Fig.2.



(a) (b)

Fig.2. (a) Choose RT kernel, (b) close check stack overflow

Use the console command to install RT-Linux. After the installation is done, we update grub to view the generated RT core. The procedure is as shown in Figure 3.



(a) (b)

Fig.3 (a) check whether the RT core is generated (b) reboot to check whether the RT core is generated

3.2Test tool installation

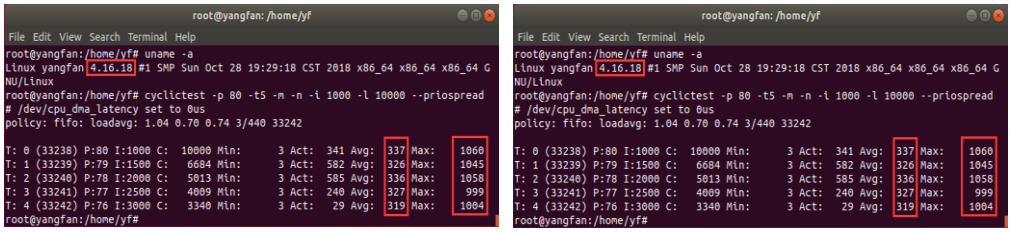
In this experiment, the test tool RT-tests was selected. When we apply for a process from the CPU, the shortest time we need the CPU to give feedback is the real time. When the request is made, the CPU will receive the request, allocate a certain resource space to the required program application, and then interrupt the execution program. This time is the real-time running time from the time the program is applied to the time before the program runs. The principle of the tool is to detect the running time and reflect the response time by number. On Ubuntu systems, we use the command: sudo apt-get install rt-tests to install the test harness.

3.3Experimental result

This experiment compares the thread response delay of the system with the delay of the thread response of the operating system after the RT-Linux patch is not installed. The patch improves the real-time performance of the system. In order to make the comparison sufficient, this experiment created five priority-decreasing threads and 15 priority-decreasing threads for performance testing.

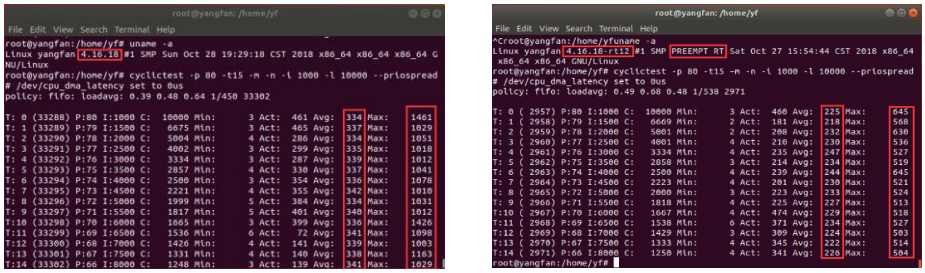
The command used in the test is cyclictest -p 80 -t5 -m -n -i 1000 -l 10000 --priospread, where -p is the priority of the thread, -t is the number of threads, and -m is the current and future lock Memory allocation, -n is to use nanosleep instead of simple sleep, -i is the basic thread interval, the default is 1000 (unit is us), -l is the number of loops, --priospread is to make the thread's priority decrement.

Figure 4 and Figure 5 show the results of the comparison experiment to create 5 threads and 15 threads respectively. Where: T: indicates the thread number; P: indicates the thread priority; I: indicates the time interval; C: indicates the counter, each time the thread's time interval arrives, the counter is incremented by 1; Min: the minimum delay (us); Act: the nearest One time delay (us); Avg: average delay of multiple scheduling (us); Max: maximum delay (us).



(a) (b)

Fig.4 (a) 5 different priority threads running in Linux (b) 5 different priority threads running in RT-Linux



(a) (b)

Fig.5 (a) 15 different priority threads running in Linux (b) 15 different priority threads running in RT-Linux

From Figure 4, it can be concluded that for an unpatched Linux system, the average delay of 5 threads is distributed at 319-337 (us), the maximum delay is distributed at 999-1060 (us); 5 threads with the same priority running in the generated RT-Linux kernel system, the average delay of 5 threads is distributed at 228-258 (us), the maximum delay is distributed at 645-771 (us), and the average delay and maximum delay are obvious reduced.

From Figure 5, we can see that for an unpatched Linux system, the average delay of 15 threads is distributed at 334-341 (us), the maximum delay is distributed at 1003-1461 (us); 15 threads with the same priority Running in the generated RT-Linux kernel system, the average delay of 15 threads is distributed at 218-247 (us), and the maximum delay is distributed at 503-645 (us); the maximum delay is significantly reduced.

4. Conclusions

In this paper, we improve the real time performance of Linux system by installing RT-Linux patch on Linux system. We test the real-time performance by RT-test tool between the system installed RT-Linux patch and system without RT-Linux patch. By comparing the five threads before and after the RT patch, it is obvious that RT-Linux has increased the average Linux by 136% on average latency; on the maximum latency, RT-Linux has increased Linux by an average of 150%. By comparing the 15 threads before and after the RT patch, it is obvious that on average latency, RT-Linux has an average increase of 146% for Linux; on the maximum latency, RT-Linux has increased Linux by an average of 199%. The experimental results show that the delay of the Linux kernel optimized by RT-Linux patch in multi-thread scheduling is obviously improved.

Acknowledgments.

This work has been supported by the National Science foundations of China (Grant Nos.: 41301382, 61401439, 41604113, 41711530128) and graduate student innovation and practice program, Xi'an Shiyou University.

References

[1] Fayyad-Kazan H, Perneel L, Timmerman M. Linux PREEMPT-RT v2.6.33 versus v3.6.6 [J]. ACM SIGBED Review, 2014, 11(1): 26-31.

[2] Lei Qi. ARM-based embedded system kernel migration and real-time research [D]. 2016(in Chinese).
[3] Yushua Liui,Yu Su,Jinbo Wang, et al. Research on real-time performance optimization of aerospace embedded Linux[J]. Space Control, 2018(3)(in Chinese).
[4] Ling Gan ,Yihong Liu.Research and real-time testing of RTAI/Linux based on ADEOS[J]. Journal of Computer Applications and Software, 2010, 27(2):163-165(in Chinese).
[5] Sheng Liu , Lifang Wang,Zejun Jiang. Real-time Transformation of Linux System Based on Multi-core PC[J]. Microelectronics & Computer, 2013(8): 120-123(in Chinese).
[6] Wei Zhang. Analysis and Research of Linux Kernel Real-Time[J]. Silicon Valley, 2012(22): 97-98(in Chinese).
[7] Yao Zhang,Jianjing,Mao,Danqi Niu.Real-time optimization of embedded Linux terminal system based on ARM[J]. Science Technology and Engineering, 2018(21)(in Chinese).
[8] Longmei Ji , Youjun Xu ,Guoqiang Shao. Research on real-time testing of embedded Linux kernel[J]. Journal of Intelligent Computer and application, 2016, 6(3): 105-107(in Chinese).
[9] Peiyu Hu . Research and design of real-time embedded system development platform [D]. Guangdong University of Technology, 2016(in Chinese).
[10] Xiaolong Zhang.Semi-physical simulation of embedded system based on Linux/RTAI [D]. Huazhong University of Science and Technology, 2014(in Chinese).