A Real-Time Process Scheduling Policy in Windows

Lifang Wang
School of Computer Science and Technology
Northwestern Polytechnical University
Xian, China
wanglf@nwpu.edu.cn

XingShe Zhou
School of Computer Science and Technology
Northwestern Polytechnical University
Xian, China
zhouxs@nwpu.edu.cn

ZeJun Jiang
School of Computer Science and Technology
Northwestern Polytechnical University
Xian, China
claud@nwpu.edu.cn

Aihua Zhang
School of Software and Microelectronics
Northwestern Polytechnical University
Xian, China
aihua.cherry@hotmail.com

***Abstract*—Microsoft Windows offers a world-class interface and excellent development tools for embedded systems. but, it cannot deliver the necessary real-time feature to meet mission-critical task. In this paper, A real-time process scheduling policy for Microsoft windows running on multiple-core CPU is presented. The policy make use of Microsoft windows' process affinity and the clock interrupt technology. Affinity can limit the process or thread to work on a subset of core on the available CPU, and high-frequency clock interrupt service can drive the real-time processes or threads to switch at an appropriate time. Interrupt Service Routine, that is executed once every clock interrupt, is core of preemptive process scheduling algorithm. Through the experiment, we can see that the policy provides a real-time task environment in Microsoft Windows that the task running period is less than 8ms. And this policy can limit the switching time between processes within a period of 200us .**

Keywords- Real-time; windows; preemptive process schedule; affinity

I. INTRODUCTION

Microsoft Windows is general-purpose operating system, suitable for use both as an interactive system on the desktop and as a server system on a network. Microsoft windows is a time-sharing multitasking operating system, it offers a world-class interface and excellent development tools for embedded systems. However, for mission-critical threads that require hard real-time, deterministic responses - e.g. real-time simulation, robots, vision systems, simulators, test and measurement systems - the Windows scheduler cannot deliver the necessary prioritization and control. The shortcomings of Microsoft windows is which lacks real-time performance, makes it a challenge to change the Windows into a real-time operating system[1, 2].

Real-time systems[3, 4] are widely used in aerospace[5], industrial control and many other fields. In the real-time calculation, the correctness of system not only depends on the logical result of the calculation, but also depends on the generated time of result. The real-time control requires the control system delay is predictable, In other words, the response time of control process is equivalent in general.

There is great difficulty in real-time scheduling for the study of real-time systems. Real–time scheduling decides the time when a series of tasks get resources which they need. The goal of Real-time process scheduling is to meet the time constraints of real-time process. The process real-time scheduling includes the preemptive process scheduling mechanism and the real-time process scheduling algorithm.

RTX[6] is a expensive software solution architected as a high-performance real-time extension to control Microsoft Windows.

In this paper, we study a real-time process scheduling mechanism based on windows operating system and focus on a preemptive process scheduling mechanism based on multiple-core CPU under the windows platform. Using the Window device drive interface and Interrupt Service Routine, we implemented real-time kernel to support real-time capabilities on Microsoft Windows.

II. THE DESCRIPTION OF SCHEDULING SYSTEM

Nowadays, CPU usually has two or more cores, and each of them has a strong capability to handle tasks. Windows are perfectly support on the multiple-core CPU. In general, each core of CPU can work well with balanced load.

Windows operating systems provide the affinity operating of process and thread. Affinity can limit the process or thread to work on a subset of core on the available CPU. In some case, the affinity can designate the thread to work on the last running core to reuse the data in catch of the processor, which will improve operational efficiency.

To some extent, the affinity exposes the process and thread scheduling mechanism working on the multiple-processor system to system programmers, and they can develop their own scheduling policies that are much more efficient in some case.

Clock Interrupt periodically generated hardware interrupt that is used by the OS for CPU-usage bookkeeping, context

978-0-7695-4719-0/12 $26.00 © 2012 IEEE
DOI 10.1109/CSSS.2012.14

switching, time-of-day updating, processing of callouts, processing of alarms, etc. It is usually the highest priority interrupt (or the second highest, after the power-failure interrupt) and it is never disabled. The period of the clock interrupt in most current operating systems is 10 msec. Increasing the Clock Interrupt Frequency can mainly improve real-time capability of non-real-time operating systems.

By using the process affinity and modifying the clock interrupt to achieve the following capabilities:

1. We can make use of the affinity to control the threads, executed on different cores guided by the operating system, to achieve the goal that some specific kind of threads can only be executed on a designated core.

2. We can add a new high-frequency clock interrupt and write the corresponding interrupt service. In general, the clock interrupt period of windows is 10ms, and now we increase 100us clock interrupt, and make system execute specific interrupt service process according to this period.

We design a real-time process scheduling policy based on windows according to the above techniques. There are two prerequisites if we implement this policy.

1. CPU contains two or more computing cores.

2. The version of windows kernel is 5.1 or 5.2 (Windows XP).

The substance of scheduling is allocation of resources. The reasonable real-time scheduling can take full advantage of the limited hardware resources. As a result, it can ensure to meet the time constraints of real-time task. The principle of fairness of the real-time scheduling is different from time-sharing system's principle of fairness. Real-time scheduling allocates these resources according to the real-time requirements of each different task.

We achieve the real-time scheduling as follows (Fig. 1):

1. By the process affinity to make all non-real-time processes of windows operating system run on the n-1(n>1, is the number of computing cores of CPU) cores.

2. The remaining one computing core is used for the real-time core. It is designed to run teal-time processes specifically. All the real-time processes are fixed to run on the real-time core based on the process affinity.

3. We add in a 100us clock interrupt for the real-time core.

4. Writing a new Interrupt Service Routine (ISR) for the new high-frequency clock interrupt, the ISR accomplish the preemptive process scheduling mechanism. The ISR change the process priority in order to achieve process scheduling, And operating system schedule these processes.

Interrupt Service Routine, that is executed once every clock interrupt, is core of preemptive process scheduling algorithm.

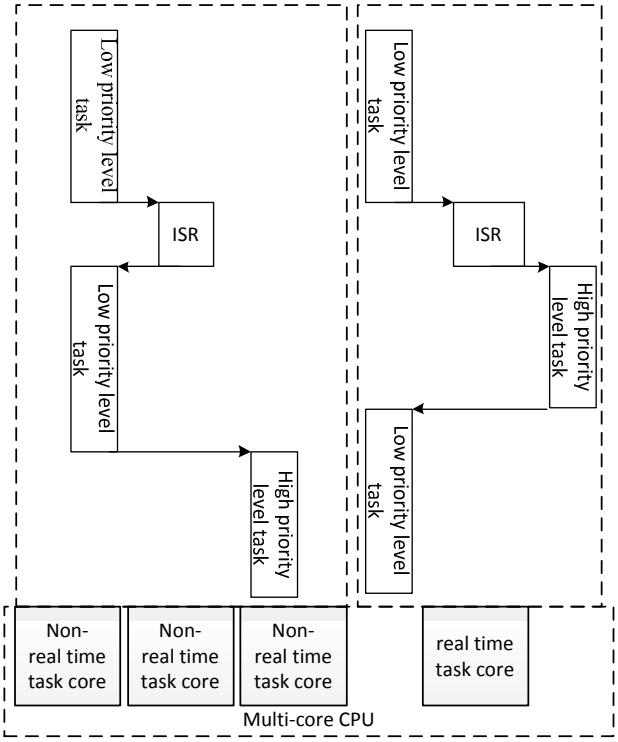

Figure 1. Real-Time Process Scheduling Policy

III. EXPERIMENT

There are two experiments:

1. We make two tasks run on the general windows XP operating system, and then monitor the scheduling situation.

2. we make two real-time tasks run on the real-time scheduling platform, and then measure the scheduling situation.

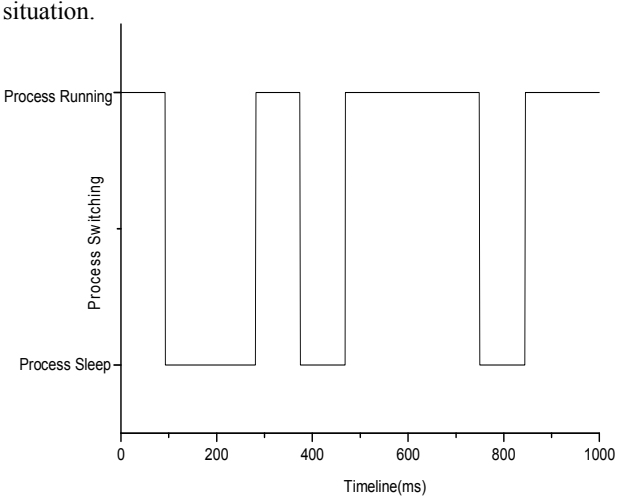

Figure 2. windows XP process scheduling situation

It is the scheduling situation of two tasks which run on the general windows XP operating system in the figure 2.

From this figure, we can see that the process scheduling of general windows XP operating system works on a very large time size, and the task is scheduled with 100ms period.

Figure 3 is the scheduling situation of two tasks which run on the real-time scheduling platform proposed in this paper. From this figure, we can see that real-time task is scheduled with a very small time size. The size is 6-8ms period. And the time of scheduling a thread is less than 1ms (within the 200us clock period).

From the results list above, we can get a conclusion that the real-time scheduling platform proposed in this paper is able to provide a 8ms scheduling cycle and can satisfy the requirement that real-time tasks run with this time magnitude.

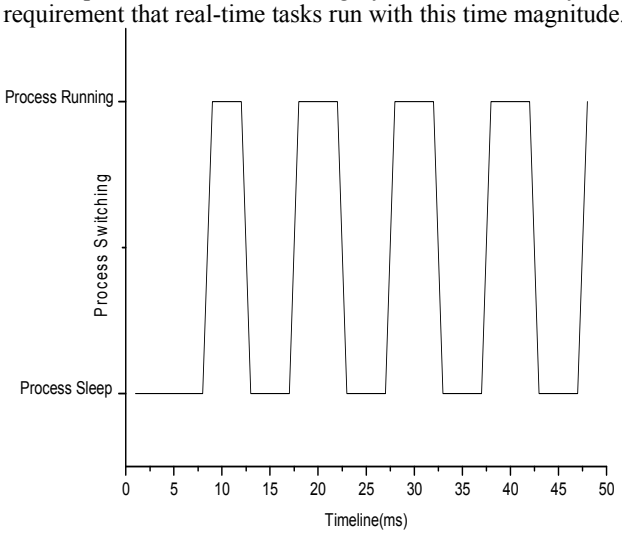

Figure 3. Real-Time Process Scheduling Policy

IV. CONCLUSION

In this paper, we make use of the process affinity and the clock interrupt technology to design a preemptive process scheduling policy based on Windows platform running on multiple-core CPU. Through the experiment, we can see that the policy provides a real-time task environment that the task running period is less than 8ms. And this policy can limit the switching time between processes within a period of 200us .

REFERENCES

[1] A. Gotsman and H. Yang, "Modular verification of preemptive OS kernels," ACM SIGPLAN Notices, vol. 46, pp. 404-417, 2011.

[2] J. Min-Gyu, L. Seung-Hoon, and L. Cheol-Hoon, "RTiK: Real-time implant kernel on microsoft windows," in IEEE Region 10 Conference (TENCON), Bali, 2011, pp. 1418-1421.

[3] P. M. Menghal and A. J. Laxmi, "Real time simulation: Recent progress & challenges," in International Conference on Power, Signals, Controls and Computation, Thrissur, Kerala, India, 2012, pp. 1-6.

[4] S. Stuijk, T. Basten, B. Akesson, M. Geilen, O. Moreira, and J. Reineke, "Designing next-generation real-time streaming systems," presented at the the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, Taipei, Taiwan, 2011.

[5] R. Bradford, S. Fliginger, R. Collins, C. R. IA, M.-Y. Nam, S. Mohan, R. Pellizzoni, C. Kim, M. Caccamo, and L. Sha, "Exploring the design space of IMA system architectures," in 2010 IEEE/AIAA 29th Digital Avionics Systems Conference (DASC), Valencia, 2010, pp. 5-15.

[6] I. Kawakami, Y. Nimura, and K. Hamada, "Real-time extension for Windows NT/CE used for control systems," presented at the the 39th SICE Annual Conference, Iizuka, 2000.