A New Method to Measure Real-Time Performance Parameters of Embedded Operating Systems Based on Benchmark Program

Wang Jianyu
School of Transportation
Huanggang Normal University
Hubei Huanggang, China
whutliuxiaojun@126.com

Liu Xiaojun*
School of Electronic & Information
Huanggang Normal University
Hubei Huanggang, China
whutliuxiaojun@126.com

Abstract—*This paper present s an improved benchmarking-based real-time performance parameters measurement method for embedded operating systems. A set of kernel codes of some common used embedded application programs is selected as workload. Some special codes are inserted in the selected programs to control the generation of the expected test behavior. μC/OS-Ⅱ embedded operating system are measured by the proposed method. The obtained results are compared with that of LMbench. The comparison result shows that the proposed method is more precise. By the method , the real-time performance of μC/OS-Ⅱ running on ARM architectures is also measured. Therefore , the correctness and effectiveness of the proposed method are further verified.*

Keywords-embedded operating system; performance measurement ; real-time performance ; benchmark program

I. INTRODUCTION

In 1989,Kar presented six key Real-Time Performance Parameters of operating system, context switch time, preemption time, interrupt latency, semaphore shuffling time, deadlock breaking time, message transfer latency. In the measurement method ,benchmark program has been widely used in the performance assessment of computer systems. The typical benchmark program method of measuring system's real-time performance parameters is Rhealstone[1-2] and LMbench[3].The main drawback of Rhealstonei is that the measurement condition is simple and the measurement result is the value when the system is at it's best ,which cannot truly reflect real-time performance of operating system because of the operating system running environment is far from the truth . LMbench just directly measure the context switch time of task and the result is an average single value. The method transfers control tokes among different tasks by pipes ,realizing switch and measurement of tasks .It can increase the reality of result by setting the number and the space of process. The program[4] LMbench is always used in many studies of home and abroad when measuring real-time performance of operating systems .The other benchmark program methods of measuring system's real-time performance parameters are method [5] of Hart stone and method [6-8] presented by L Abeni and so on. The main measurement object of Hart stone is the entire system with hardware and software included, not the operating system itself. Consistent with Rhealstone ,the measurement parameters of L Abeni reflect the scheduling and preempting performance of operating system.

This paper improves the nowadays benchmarking-based real-time performance parameters measurement method for embedded operating systems. The basic idea is that a set of kernel codes of some common used embedded application programs is selected as workload and some special codes are inserted in the selected programs to control the generation of the test behavior. By comparison, with the object of μC/OS-Ⅱembedded operating system ,the realistic measurement suggests that the measurement result gained by using this method is more real and accurate.

II. BENCHMARKING-BASED REAL-TIME PERFORMANCE PARAMETERS MEASUREMENT

The chapter mainly takes context switch time, deadlock breaking time and interrupt latency for example to illustrate the theory and enforcement process of new testing method because of the method for testing preemption time and message transfer latency is similar to context switch time ,while the method for testing semaphore shuffling time is similar to deadlock breaking time ..

A. *The choose of workload*

Embedded performance benchmark program Mibench was designed by researchers of American's Michigan university[9-10]. It contains six groups of thirty five embedded application program and every special group represents relevant embedded application The six groups respectively are industry control and automation ,personal consummation ,working automation ,network ,information security and communication .With good portability ,all programs are open source code of C programming language .Various kinds of application program of Mibench is selected as workload according to testing needs.

* Corresponding Author

B. 2.2 The measurement of context switch time

Switch events are needed to be artificially made among tasks for testing context switch time .In the condition with no preemption ,one scheme is letting the task running at present activate another task with specific ID, then suspends itself.. Therefore ,the activated task is in a steady state ,resulting in scheduling[11] .The switch process contains a series of operations ,for example ,the storage of present task context ,the recover of new task context and so on. The cost of time in the process is context switch time (TCS).Provided the ID of task running at present is TaskID[i] and the ID of new task running next is TaskID [i + 1].The t1 (TaskID[i + 1]) represents the moment when the new task is activated and ,t- (TaskID[i + 1]) represent the moment when the new task begins to run. So, t2-(TaskID [i +1]) -t1(TaskID [i + 1]) can approximately represent the switch time between two tasks in the users' layer. Fig.1 gives sketch map of two task's switch process.

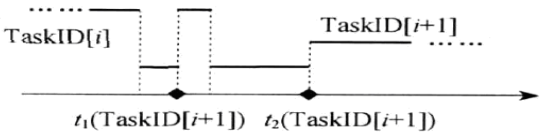

Figure 1. Process of task swiching

For decreasing error , choose n tasks and calculate the average algorithm value of context switch time according to the equation (1) below after testing the starting running moment and the activating moment of n tasks:

$$T = \frac{1}{n}\sum_{i=0}^{n-1}(t_2(taskID(i+1)/n) - t_1(taskID(i+1)/n)) \quad (1)$$

Repeat the process K times and the context switch time can be gotten.

Measurement control code and time counter need to be inserted in the proper place of every task's source code for realizing the switch of tasks. The using ratio of CPU, memory and so on is different in different stages of every task, therefore , every task's codes need analyzing and codes are inserted into many different stages of the program to get more and more true data. The place where codes are selected can choose initial segment ,middle segment and ending segment of program running .

C. The measurement of deadlock breaking time

Fig.2 shows the testing process of deadlock breaking time .Task TaskID[0] firstly runs and enters into the critical section then it activates task TaskID[1]. TaskID[1] will preempt the running of TaskID[0] ,for it has the highest priority .When TaskID[0] enters into critical section ,due to the critical resources are occupied by TaskID[1] ,it will suspend itself until the resources are released. TaskID[0] continues to be scheduled to run and activates TaskID[2 .Meanwile , ,TaskID[2] will preempt the running of TaskID[0],for it has higher priority than TaskID[0] .

The situation where TaskID[2] of middle priority is running all the time whereas TaskID[1] of high priority is suspended occurs at the same time. In the situation ,operating system can adopt proper method (priority inheriting, priority ceiling and so on)to break , deadlock .Provided ta represents the moment when TaskID[1] tries to enter into the critical section ,tb represents the moment when TaskID[1] successfully enters into the critical section. Therefore ,tb-ta is the tested deadlock breaking time .

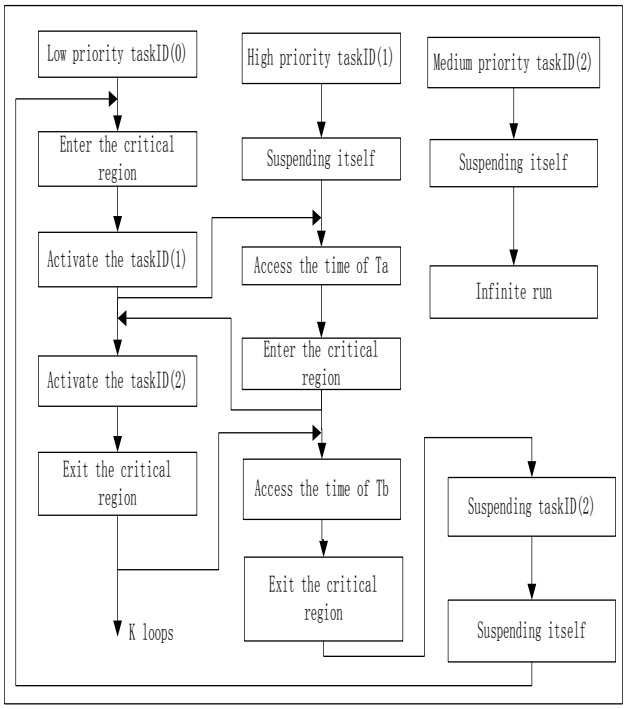

Figure 2. the flow chart for deadlock breaking time measurement

D. The measurement of interrupt latency

Interrupt latency is the time from CPU receive interrupt requests to the first instruction of interrupt service program begins to be executed. Write a simple interrupt service program and fix it to relevant interrupt number . In order to test the interrupt latency in worst situation ,the picked workload should contain the manipulation of disk reading and writing ,data processing ,system scheduling and so on that might affect interrupt latency and testing control codes are inserted into different places of these programs. Control codes of interrupt latency measurement are relatively simple.

```
//Interrupted service program
{
   ......
   Get time Ta;
   Get time Tb;
   T=Tb-Ta;
   Save(T);

}
```

```
//taskID
{   ......
    source code
   ......
    Get time(Ta);
    call int;
   ......
    source code
   ......
}
```

Figure 3. An example of workload for interrupt latency measurment

Fig.3 is a workload example for measuring interrupt latency ,the call-int in it represents the function generating interrupt ,it also can be replaced by software

interrupt .Provided ta is the moment of generating interrupt request ,tb is the moment of executing interrupt service software ,therefore ,tb-ta can be regarded as interrupt latency .

III. THE VERIFICATION OF MEASURING METHOD

The context switch time is taken for example to illustrate the effectiveness of method .The measuring object is μC/OS-Ⅱembedded operating system and the comparison measurement adopts

The main difference of the two methods lies in this method tries to make the running environment close to reality, all the running tasks are kernel codes of some of frequently used application programs. By the technology of code inserting, which controls generation switch of these tasks to measure and gets data in different stages of task running .But the task of LMbench is empty process ,these processes themselves haven't any work except switching each other and the measuring environment is simple.

The platform measuring hardware is embedded developing board with CPU of STPC2DX2,frequency of which is 133MHz ,and memory of 64MB. Timer 0 of 8254 ,with a frequency of 12 MHz is used for increasing the measuring accuracy .

The code of timer needs modifying for measuring several groups of data in a time because of there is an average value when LMbench measures context switch time each time. Respectively choose one representative program from six classifications of Mibench to form workload .They are: jpeg ,fft ,pat ricia, sha,ispell and bit count. The measuring result is showed as table 1.

It can be seen from table 1 that the data standard deviation gained using LMbench is much smaller ,but the data standard deviation measured using this method is much bigger .The data measured using this method is more real, not only can it represent the total average value of measurement parameters ,it also can reflect the switch time when the operating system is in worse situation ;furthermore , interrupt latency, task switching time are close to the parameters provided by the tested operating system's developer. Certainly ,the measured results contain some cost of scheduling time counter ,combined the accuracy problem of time counter itself, there exists some degree of error.

TABLE I. COMPARISON OF MEASUREMENT RESULT OF CONTEXT SWITCHING TIMES BY TWO METHODS

Method	Average value(μs)	Minimum value(μs)	Maximum value(μs)	standard deviation
LMbench	39.9505	32.0153	57.4115	4.1405
This paper	42.6541	33.2658	87.5468	6.7603

TABLE II. THE MEASUREMENT RESULT OF μC/OS-Ⅱ

Method	**Average value(μs)**	**Minimum value(μs)**	**Maximum value(μs)**	**standard deviation**
Context switch	55.1294	38.5214	125.2563	18.2543
Preemption	85.2314	67.5412	189.2563	21.5634
Interrupt latency	83.2568	65.4231	178.2563	15.6745
Semaphore shuffling	68.1234	65.1254	89.5487	8.5426
Deadlock breaking	238.5623	214.2356	398.5647	12.3568
Message transfer latency	69.5423	68.2456	123.2354	4.2536

IV. THE MEASUREMENT RESULT OF μC/OS-Ⅱ EMBEDDED OPERATING SYSTEM

The measurement for μC/OS-Ⅱ embedded operating system is carried out on ARM platforms ,they respectively are Arm11 developing board, the main frequency of which is 600 MHz. ,the six performance parameters measuring results are showed as table 2.

It can be seen from the table 2 that the standard deviation of task preemption time of the platforms is much bigger ,the maximal value is 190μs or so. This is because the kernel version of μC/OS-Ⅱ is of no preemption or the preemption performance is bad.

V. CONCLUSION

Making the measurement conditions closer to the realistic working environment of operating system for gaining more real measurement results ,the work improved nowadays benchmarking-based real-time performance parameters measurement method for embedded operating systems. The realistic measurement result of a embedded operating system μC/OS-Ⅱ suggests that the accuracy of new method is increased compared to the LMbench .The measurement of μC/OS-Ⅱ carried out on ARM platforms further verify the effectiveness of new method and it also get the valuable real-time performance data of operating system. For the measurement of different parameters ,there are different visions of source codes for different tasks in workload ,which makes the entire program more huge, making it less convenient to measure the embedded system with limited resources due to benchmarking-based real-time performance parameters measurement for embedded operating systems needs to modify the codes of workload .In addition, the realizing of interrupt latency measurement method relies more on hardware and operating system itself ,the portability is relatively worse, so, the measurement method also needs improving continuously.

ACKNOWLEDGEMENTS

The project was supported by the Electrical and Electronic Experimental Teaching Demonstration Center Project of Huanggang Normal University (Grant No.zj201257).

REFERENCES

[1] Kar K P ,Porter K. Rhealstone ——— A real-time benchmarking proposal[J] . Dr. Dobb's Journal ,1989, (3):14.P12-19

[2] J IANG Jianhui. Performance assessment of embedded systems by benchmarking[J]. Machinery & Electronics ,2002 , (4) :43. P-19

[3] McVoy L, Graphics S. LMbench : Portable tools for performance analysis [C] // Proc USENIX 1996 technical conference. San

[4] Diego : USENIX Association ,1996 :279 - 294.

[5] L I Jiang ,DAI Shenghua. Rea1-time performance test and analysis of Linux [J] . Computer Applications ,2005 , (25) :1679.

[6] Weiderman N. Hartstone : synthetic benchmark requirements for hard real2time appications [C] // Proc Working Group on Ada Performance Issues. Maryland :ACM SIGAda ,1990 :126 2 136.

[7] Abeni L ,Goel A , Krasic C,etal. A measurement2based analysis of the real2time performance of Linux [C],Proc 8th IEEE real2 time and embedded technology and applications symposium. San Jose : IEEE Computer Society ,2002:133-142.

[8] Yaghmour K, Masters J, Gerum P, etal. Building embedded Linux systems[M] . Cambridge :O'Reilly ,2003.

[9] Guthaus M R ,Ringenberg J S ,Ernst D ,et al. MiBench : A free commercially representative embedded benchmark suit [C] // Proc 4th IEEE international workshop on workload characteriza2 tion. Washington : IEEE Computer Society ,2001:1-12.

[10] Sweetman D. See MIPS run[M] . 2nd ed. San Francisco :Morgan Kaufmann , 2006.

[11] Jiang Jianhui , Tang Zhijie, A Novel Method to Measure Real-Time Performance Parameters of Embedded Operating Systems [J], Journal of Tongji university (Natural Sciences), 2008,Vol.36,No.9, P:1260-1265