Analysis of thread schedulability in Huawei LiteOS

Xiaochun Wang[1,*]

[1]School of AI, Shenzhen Polytechnic, 518055, Shenzhen, Guangdong, China

Abstract. Huawei LiteOS is a real-time operating system. Thread schedulability is an important thing to be considered first when we use the RTOS in an application. There are a lot of methods to value thread schedulability in practical application. Rate monotonic scheduling algorithm is a widely used static priority scheduling algorithm. We discussed the thread schedulability in Huawei LiteOS.

1. Foreword

Huawei LiteOS is lightweight IoT operating system, it is a real-time operating system. According to the document of the LiteOS[1], the minimum size of the kernel will be under 10KB, so the product use LiteOS will consume very little power and have a faster response. How about the performance of the LiteOS when it is in a real-time environment?

In Huawei LiteOS, it guaranteed the highest-priority thread scheduling, which will meet the time critical system demands.

Huawei LiteOS used as a real-time operating system in many application area, such as smart homes, wearables, Internet of Vehicles (IoV), and intelligent manufacturing. A real time application product must be designed to meet time demands, the system is not only need to provide the correct response, but also need to respond within a specified period.

It is must important to make sure that threads will be scheduled before its deadlines in Huawei LiteOS. Rate Monotonic scheduling algorithm (RM[2]) is an static-priority algorithm. Just like any other static-priority algorithm, RM can also produce a feasible schedule. We will discuss the thread schedulability in Huawei LiteOS.

2. Thread schedulability in Huawei LiteOS

2.1 RM scheduling algorithm

There are a set of periodic threads in a system, $S= \{t1,t2,\ldots,tn\}$. Threads are described as $ti=(Ti,Ci,Di,Pi,Ui)(i=1,\ldots,n)$。

The notation is：*Ti:* period of the thread *ti*, *Ci:* execution time of the thread *ti*, *Di:* deadline of the thread *ti*, *Pi:* priority of the thread *ti*, *Ui:* processor utilization of the thread, that is $Ui=Ci/Ti$.

* Corresponding author: wxc6502@szpt.edu.cn

For threads set *S*={*t*1,*t*2,…,*tn*}, there are another notations[2]:

$$Wi(t) = \sum_{j=1}^{i} (Ci)[\frac{t}{Tj}] \tag{1}$$

Wi(t) is the amount of work executed by threads *t1*,*t2*,… *tn*, initiated in the time between [0,*t*],.

Here is the RM Scheduling Algorithm:

If thread *ti* is RM-schedulable, when and only when[2]:

$$Li(t) = Wi(t)/t \tag{2}$$

$$Li = \min_{\{t \in Ti\}} Li(t) \le 1 \tag{3}$$

If the entire set of periodic threads S is RM-schedulable, when and only when[2]:

$$L = \max_{\{1 \le i \le n\}} Li \le 1 \tag{4}$$

RM algorithm cab be used to determine thread schedulability. If the total utilization of the threads is no greater than $n(2^{1/n}-1)$, where *n* is the number of scheduled threads, the RM algorithm can meet time demand of all threads in a system. That means all threads can be scheduled before their deadlines.

When all threads in a system of Huawei LiteOS are periodic threads, RM algorithm can be used easily according to the notation discussed above. But when we use RM Algorithm in a system that only part of threads are periodic, there are a lot conditions must be met first.

2.2 Thread Priority in RM scheduling system

In a RM scheduling system, the priority of a thread must be inversely related to its period. If thread *ti* has a smaller period than thread *tj*, *ti* must has a higher priority than *tj*.

In Huawei LiteOS, there are 32 priority level, the highest is level 0 and the lowest is level 31. The priority level of thread is defined by programmer. So the thread priority may not be inversely related to its period, it may not meet the demand of the RM algorithm application.

Liu and Layland proved that in any threads set, which can be scheduled in other static priority scheduling algorithm, can also be scheduled by RM algorithm. That is: They assume that there is a thread set S, thread *ti* and thread *tj* are two threads that their priorities are adjacent, *Ti*>*Tj* and *Pi*<*Pj,* they exchange the priority of thread *ti* and thread *tj* , they prove that this threads set can also be scheduled. According this, they prove that any other static priority system used scheduling algorithm can be changed to RM algorithm.

Here to the conclusion, RM algorithm can be used in any case of static priority scheduling algorithm.[4] Based on this, RM algorithm can also be used as a thread scheduling method in Huawei LiteOS system.

2.3 Thread schedulability in sporadic threads system

In a system of Huawei LiteOS, we think that all threads are periodic, so RM algorithm can be used to determine every thread schedulability. In a real application, there may be threads released irregularly, these threads are sporadic threads, Can RM algorithm be used in such system?

We suggest in the worst situation, when a period released, at the same time, there is a sporadic thread released too. In periodic thread set S, there are *n* threads need to be scheduled. Meanwhile in sporadic threads set St, there are also *n* threads need to be scheduled. When thread *ti* is released, thread *sti* is released at the same time with the thread *ti*. In generally, the priority of thread *ti* is higher than *sti*, so thread *ti* runs before thread sti.

Here an example, there are three periodic threads, *t*1, *t*2, *t*3, the periods time are *T*1, *T*2, *T*3, the execution time are *C*1, *C*2, *C*3, the priority are *P*1, *P*2, *P*3, and *P*1<*P*2<*P*3. We assume that every periodic thread is released at time zero. There are three sporadic threads, *st*1, *st*2, *st*3 , released at the same time, the execution time are S*C*1, *SC*2, *SC*3, and their priority are lower than the periodic thread released at the same time.

Thread *t*1: As sporadic thread *st*1 released at the same time, to ensure thread *t*1 can be scheduled, the time must be satisfied is $C1 + SC1 \le T1$.

Thread *t*2: In the interval [0,*t*], thread *t*1 has already released [*t*/*T*1] times, and *st*1 has released [*t*/*T*1] times too. To ensure that thread *t*2 can be scheduled correctly, periodic thread *t*1 and sporadic threads *st*1 must be iterated [*t*/*T*1] times, and there must be some time left to finish periodic thread *t*2 and sporadic thread *st*2. That is:

$$t = (C1 + SC1)\cdot t/T1 + (C2 + SC2) \tag{5}$$

There must be a time *t* to meet the demand:

$$t \ge (C1 + SC1)\cdot t/T1 + (C2 + SC2) \text{ and } t \le T2. \tag{6}$$

Thread t3: In the interval [0,t], There must be a time t to meet the inequality[3]

$$t \ge (C1 + SC1)\cdot t/T1 + (C2 + SC2)\cdot t/T2 + (C3 + SC3) \text{ and } t \le T3. \tag{7}$$

If this *t* can be found, the thread *t*3 can be scheduled.

Based on the conclusion described above, for a periodic threads set S, the conclusion is:

For periodic threads set S={*t*1,*t*2,…,*t*n}, in each thread period, there will be a sporadic thread released, sporadic thread queue is St={*st*1,*st*2,*st*3, …,*st*n}. Sporadic thread *sti* will be released at the same time with periodic thread *ti*, there must be a time *t* to meet:

$$t \ge \sum_{j=1}^{i} (Cj + SCj)[\frac{t}{Tj}] \text{ and } t \le Ti \tag{8}$$

Then the thread *ti* can be scheduled.

So, in the worst situation, when each periodic thread is executed, there is a sporadic thread released too.

The system must meet the conditions described as:

The thread *ti* ($1 \le i \le n$) is RM-scheduling, when and only when[3]:

$$\min_{\{t \in Ti\}}\{\sum_{j=1}^{i}(Cj + SCj)[\frac{t}{Tj}] \Big/ t\} \le 1 \tag{9}$$

$$\max_{\{1 \le i \le n\}}\{\sum_{j=1}^{i}(Cj + SCj)[\frac{t}{Tj}] \Big/ t\} \le 1 \tag{10}$$

In real application, the execution time of each sporadic thread must be calculated first, and then the total amount of threads execution time is analyzed to ensure every thread in

Huawei LiteOS system can be scheduled.

3 Conclusion

Huawei LiteOS was released in June, 2017. It is widely used in an IoT-oriented application integrating an IoT operating system and middleware. Generally, the application is a real time environment, so it is much important to determine each thread can be scheduled before their deadline. In practical development, different conditions may affect the real time performance of the system, we discuss sporadic threads here to handle the system schedulability.

Reference

1. Huawei LiteOS Documents. (2020)
2. C.M .Krishna, Real-time Systems [M], Beijing: Tsinghua University Press, (2001)
3. Wangxiaochun. International Conference on Applied Informatics and Communication, ICAIC (2011)
4. 4 LIU Jun-Xiang, WANG Yong-Ji, [J].Journal of Software, 15, 799-814 (2004)