Using Linked List in Exact Schedulability Tests for Fixed Priority Scheduling

Jiaming Lv*, Xingliang Zou§, Albert M. K. Cheng§, and Yu Jiang+*
*School of Computer Science and Technology, Heilongjiang University, Harbin, Heilongjiang 150080, China
Email: 962831141@qq.com, jiangyu@hlju.edu.cn
§Department of Computer Science, University of Houston, Houston, TX 77004, USA
Email: xzou@uh.edu, cheng@cs.uh.edu

***Abstract*—Efficient exact schedulability tests are one of important considerations of both research motivation and practice stage. In this paper, we investigate the exact response-time schedulability tests for fixed priority preemptive systems. The linked list is introduced to represent the simulated schedule of a given task set. Each node in the linked list represents a busy period. In addition, the memory space needed for the linked list is managed in the user space. Experiments show that the linked list-based exact test outperforms the current best exact response-time test and the hyperplanes exact tests (HET) in the case of task periods spanning no more than three orders of magnitude.**

I. Introduction

Real-time systems are playing a crucial role in our daily lives and in industry production, and fixed priority preemptive scheduling is widely supported by most commercial real-time operating systems [6].

In the context of fixed priority preemptive real-time systems, it is known that for periodic/sporadic tasks that comply with a restrictive system model and that have implicit deadlines the *Rate-Monotonic* (RM) scheduling is optimal, i.e., if a feasible scheduling exists for some task set then the RM scheduling is feasible for that task set [11], [14]. RM means that the priority of each task is assigned inversely proportional to its period (i.e., minimum inter-arrival time between jobs of the task). It is also known that when these tasks are released simultaneously (i.e., sharing a common release time) the time required by the first job of each task defines its response time [11], [14]. Therefore, it needs only to make response time analysis or conduct exact schedulability test within a time length no more than the maximum task period, and these tests are thus known to be pseudo-polynomial in time complexity [8], [9], [1].

Although the response time computation for RM schedules of implicit-deadline task-systems has been proved to be an NP-hard problem [7], the scale of many commercial systems is such that pseudo-polynomial exact tests can be used, and to achieve more efficient exact tests for use such as online response time analysis (RTA) is one of important considerations of both research motivation and practice stage. A significant research effort has been dedicated to improve the performance of exact response-time tests [8], [1], [12], [2], [6], [4], [13], such as finding good initial values, and to the best of our knowledge the authors of [6] presented the current best response-time test with better initial values.

In this paper, we investigate exact response-time schedulability tests of the RM scheduling in an n-task real-time system. For concision we use the *Burns Standard Notation* [5], such as the number of tasks n, for $1 \leq i \leq n$, the priority P_i, the worst-case execution time C_i, the relative deadline D_i, the period T_i, the worst-case response time R_i, and the utilization U_i, for a task τ_i.

The innovative aspect of our solution is that we use a linked list for representing the schedule in the exact response-time test, referred to as the *LList-based exact test*, for calculating the worst-case response time. The time complexity of the LList-based exact test is polynomial-time $O(N)$ where N is the total number of jobs within the time length T_n, while the total number of nodes used in the linked list is no more than $N - n + 1$ in the worst case. Our experiments show that the LList-based exact test outperforms the current best exact RTA test [6] and the hyperplanes exact tests (HET) [2] in the case of task periods spanning no more than three orders of magnitude, and the needed memory space is also affordable.

II. Our Method

We calculate the response time of each task set by simulating its schedule within a time length T_n. In our method, the schedule of a task set is represented by a linked list, and a *busy period* [10] in the schedule is represented by a linked list node. Each list node has three fields: the starting time of the busy period, the end time of the busy period, and the pointer to the next node. The simulation is performed task per task in the priority order, from 1 to n, and, when the starting time or the end time of a *priority level-i busy period* is the same as that of a *priority level-j busy period* where $j < i$, then the two nodes are merged into one node to represent a longer busy period.

Another key factor for improving the efficiency of the LList-based exact test is that before the simulation a memory array is allocated as a whole and then the following operations of memory allocation and recycle for each node are performed in the user space instead of in the operating system space.

This work is sponsored in part by the State Scholarship Fund of China under award No. 201308230034, the US National Science Foundation under award Nos. 0720856 and 1219082, the Funds of Heilongjiang Education Office of China under award Nos. 12541627 and GJZ201301027, and the Heilongjiang Student's Platform for Innovation and Entrepreneurship Training Program of China under award No. 2015102121008.

+Corresponding author.

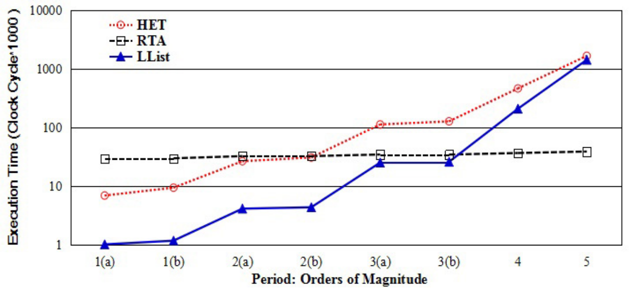

Fig. 1. Average execution time required by the HET, RTA, and LList-based exact schedulability tests versus number of orders of magnitude range of task periods. The range of periods starts from 10 and 10^4 for (a) and (b), respectively. Note both x and y-axes are logarithmic scales.

TABLE I
HET, RTA, AND LLIST-BASED ALGORITHMS, EXECUTION TIME IN COUNTER CLOCK CYCLES $\times$ 1,000

	Orders of magnitude spanning tasks periods						
Algorithm	1(a)	1(b)	2(a)	2(b)	3(a)	3(b)	4
HET	6.9	9.4	26.6	30.8	112.8	127.4	464.9
RTA	29.56	29.82	32.47	32.48	34.63	34.49	36.67
LList	1.01	1.17	4.12	4.35	25.19	25.94	208.9
RTA/LList	29.3	25.5	7.9	7.5	1.4	1.3	0.2

For an n-task set, the total number of jobs, N, in the time interval [0, T_n) is $N = \sum_{i=1}^{n}(T_n/T_i)$ and only these jobs are simulated in the LList-based exact test. This number is sensitive to the span and the distribution of task periods. Since one busy period in the schedule is represented with merely one node and the simulation is performed task per task in the task priority order, the total running time is mainly determined by the time of operating linked list nodes, and thus the time complexity of the LList-based exact test is related to the number of nodes used in the simulation. Particularly, as long as task periods span only one order of magnitude, only $O(n)$ time are needed for simulating an n-task set, regardless of the length of the maximum period of the task set as well as the total utilization. In the above mentioned case, the total number of list nodes used in simulation is very small, and this is the root cause why the LList-based test is performing better.

III. Experiment and Preliminary Results

For comparison, we use the same parameters as those in [6], the current best exact RTA test. Specifically, for each 24-task set $24/M$ tasks were assigned to each of the M order of magnitude ranges (e.g., 100-1,000, 1,000-10,000, 10,000-100,000, etc.). Task periods were then uniformly and randomly generated from the assigned range. The overall utilization was fixed at 0.85 and the UUniFast algorithm [3] was used to determine task utilizations U_i, and, hence, task execution times, $C_i = U_i * T_i$. There are 10,000 task sets in each order of magnitude ranging from 1 to 5.

Fig. 1 and Table I show how the average number of clock cycles required by the HET algorithm with initial values $R_{i-1} + C_i$, by the RTA test with initial values $max_{k=1}^{i}(R_i^{LB}(k))$ [6], and by the LList-based test varied with the number of orders of magnitude spanning task periods. From the figure we can see that the execution time of the RTA test goes nearly steady while the HET and the LList-based tests increase exponentially with increasing number of orders of magnitude spanning task periods. Within three orders of magnitude spanning task periods, however, the performance of the LList-based test outperforms both the HET test and the RTA test with the current best initial values.

TABLE II
THE MAXIMUM NUMBER OF NODES AND CORRESPONDING MEMORY SPACE

	Orders of magnitude spanning tasks periods						
	1(a)	1(b)	2(a)	2(b)	3(a)	3(b)	4
max # nodes	28	58	236	441	2302	3246	20996
KB(int32)	0.3	0.7	2.8	5.2	27.0	38.0	246.0

Table II shows the maximum number of list nodes used in the LList-based test versus number of orders of magnitude range of task periods. From the table we can see that, within three orders of magnitude spanning task periods, the memory space needed by the linked list is completely affordable.

IV. Conclusion

The work presented in this paper is part of our ongoing research on the response time analysis and exact scheduability test for fixed priority preemptive systems. Our preliminary results have shown that the LList-based exact test is a better candidate in exact response-time tests when task periods span no more than three orders of magnitude. More comprehensive experiments will be conducted to investigate the suitable scope of the LList-based exact test by varying the number of tasks, the range of task periods, and the total utilizations.

References

[1] N. C. Audsley, A. Burns, M. Richardson, K. W. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.
[2] E. Bini and G. C. Buttazzo. Schedulability analysis of periodic fixed priority systems. *IEEE Trans. on Computers*, 53(11):1462–1473, 2004.
[3] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Syst.*, 30(1-2):129–154, 2005.
[4] R. I. Davis. A review of fixed priority and EDF scheduling for hard real-time uniprocessor systems. *ACM SIGBED Review*, 11(1):8–19, 2014.
[5] R. I. Davis and A. Burns. Burns standard notation for real-time scheduling. In *Real-Time Systems: The past, the present, and the future. N. Audsley, S.K. Baruah Editors*, pages 38–41, Mar. 2013.
[6] R. I. Davis, A. Zabos, and A. Burns. Efficient exact schedulability tests for fixed priority real-time systems. *IEEE Trans. on Computers*, 57(9):1261–1276, 2008.
[7] F. Eisenbrand and T. Rothvoss. Static-priority real-time scheduling: Response time computation is NP-hard. In *IEEE RTSS 2008*, pages 397–406, 2008.
[8] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer J.*, 29(5):390–395, 1986.
[9] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *IEEE RTSS 1989*, pages 166–171, 1989.
[10] J. P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *IEEE RTSS 1990*, pages 201–209, 1990.
[11] C. Liu and L. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of ACM*, 20(1):46–61, 1973.
[12] W. C. Lu, K. J. Lin, H. W. Wei, and W. K. Shih. Period-dependent initial values for exact schedulability test of rate monotonic systems. In *Proc. IPDPS 2007*, pages 1–8.
[13] M. Park and H. Park. An efficient test method for rate monotonic schedulability. *IEEE Trans. on Computers*, 63(5):1309–1315, 2014.
[14] O. Serlin. Scheduling of time critical processes. In *Proc. AFIPS Spring Computing Conf.*, pages 925–932, 1972.