An Evaluation of the RUN Algorithm in LITMUS$^{\mathrm{RT}}$

[Extended Abstract]

Hiroyuki Chishiro[†‡], James H. Anderson[†], and Nobuyuki Yamasaki[‡]
[†]Department of Computer Science, The University of North Carolina at Chapel Hill, NC, USA
[‡]Department of Information and Computer Science, Keio University, Yokohama, Japan

1. INTRODUCTION

Existing multiprocessor real-time scheduling algorithms follow partitioning/global scheduling approaches or some hybrid approaches of the two. Under partitioning, all tasks are assigned to specific processors. Under global scheduling, tasks may migrate among processors. Global scheduling has the advantage of better schedulability compared to partitioning. However, optimal algorithms based on global scheduling such as PD^2 [5] and LLREF [3] incur significant overhead.

2. RUN ALGORITHM

The Reduction to UNiprocessor (RUN) algorithm [4] is optimal multiprocessor real-time scheduling with a hybrid partitioning/global scheduling approach, called semi-partitioning. Under semi-partitioning, most tasks are assigned to processors and the remaining tasks may migrate among processors. RUN achieves low overhead in simulation studies but the practical viability of it remains unclear.

3. LITMUS$^{\mathrm{RT}}$

LITMUS$^{\mathrm{RT}}$ [2] is a real-time extension of the Linux kernel that allows schedulers to be developed as plugin components. Current plugins include PD^2 and partitioned/global fixed-priority/earliest deadline first schedulers.

4. THE RUN PLUGIN

This work introduces an additional plugin to implement RUN in LITMUS$^{\mathrm{RT}}$. RUN assigns tasks to processors offline. The RUN plugin simply utilizes an offline constructed dispatching table to make scheduling decisions at runtime.

The implementation of RUN differs substantially from that of PD^2, the only optimal plugin currently available in LITMUS$^{\mathrm{RT}}$. PD^2 is actually only optimal if the quantum size can be made sufficiently small. In practice, each quantum boundary creates overheads due to the need to handle interrupts, make scheduling decisions, and perform context switches. This limits the practical quantum-size. Quantum-driven scheduling also creates additional release delay since the scheduler does not process scheduling events immediately. On the other hand, RUN is a time-driven scheduler that uses a dispatching table so that such additional release delay does not occur. The other advantage of the RUN plugin reduces cache-related preemption and migration delay. RUN has been shown to reduce the number of preemptions/migrations compared to LLREF with lower overhead than PD^2 in simulation studies [4].

5. PLANNED EVALUATION

We introduce a planned evaluation to verify whether RUN is practical. In this evaluation, we will measure relevant overheads with a variable number of tasks/cores, several ready queue/interrupt handling methods and the schedulability of RUN on both hard/soft real-time systems with respect to [1].

6. CONCLUSIONS AND FUTURE WORK

This work introduces the RUN plugin in LITMUS$^{\mathrm{RT}}$. We discussed the difference between the implementations of RUN and PD^2. We conclude that RUN is more practical than PD^2 with respect to overheads. In future work, we will perform the planned evaluation of RUN.

7. ACKNOWLEDGEMENT

This research was supported in part by CREST, JST. This research was also supported in part by Grant in Aid for the Global Center of Excellence Program for "Center for Education and Research of Symbiotic, Safe and Secure System Design" from the Ministry of Education, Culture, Sport, and Technology in Japan. In addition, this research was also supported in part by Grant in Aid for the JSPS Fellows.

8. REFERENCES

[1] A. Bastoni, B. B. Brandenburg, and J. H. Anderson. An Empirical Comparison of Global, Partitioned, and Clustered Multiprocessor Real-Time Schedulers. In *Proceedings of the 31th IEEE Real-Time Systems Symposium*, pages 14–24, Dec. 2010.

[2] J. M. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. H. Anderson. LITMUS$^{\mathrm{RT}}$: A Testbed for Empirically Comparing Real-Time Multiprocessor Schedulers. In *Proceedings of the 27th IEEE Real-Time Systems Symposium*, pages 111–123, Dec. 2006.

[3] H. Cho, B. Ravindran, and E. D. Jensen. An Optimal Real-Time Scheduling Algorithm for Multiprocessors. In *Proceedings of the 27th IEEE Real-Time Systems Symposium*, pages 101–110, Dec. 2006.

[4] P. Regnier, G. Lima, E. Massa, G. Levinand, and S. Brandt. RUN: Optimal Multiprocessor Real-Time Scheduling via Reduction to Uniprocessor. In *Proceedings of the 32th IEEE Real-Time Systems Symposium*, pages 104–115, Nov. 2011.

[5] A. Srinivasan and J. H. Anderson. Optimal rate-based scheduling on multiprocessors. *Journal of Computer and System Sciences*, 72(6):1094–1117, 2006.