Study on Acceleration of Real-Time Task Scheduling

Doan Duy (1410221)

School of Information Science,
Japan Advanced Institute of Science and Technology

August, 2016

Keywords: real-time scheduling, virtual release advancing, EDF-based scheduling, TBS server.

It is a fact that real-time computing systems have become more and more popular and played a very important role in our life and science. Real-time applications are now employed in a large spectrum of areas from normal applications such as home appliances, telecommunication, automobile, to high precise ones such as industrial production lines, surgical operations, and aerospace controls. As a result, tasks that real-time systems have to handle are increasingly diverse in types and constraints. The importance of real-time task scheduling is hence growing.

In recent decades, there are a lot of researchers have made an attempt at proposing such an effective and practical real-time scheduling algorithm. However, because of the diversity of requirements for the real-time systems, algorithms that are (semi-)optimal in terms of schedulability or response time usually have high complexity, which prevents actual systems from adopting them. Thus, there are existentially aim and challenge to researchers.

Virtual release advancing is one of the successful algorithms in terms of substantially improving the response time of real-time task scheduling. This is a technique based-on the original TBS server and EDF algorithm. Although achieving a good response time, this technique generates not a small scheduling overhead, which makes it difficult for these algorithms to be applied to future real-time systems that operate at high precision (or fine periods).

In this research, an enhanced virtual release advancing algorithm is proposed to improve the time complexity of the original virtual release advancing algorithm for real time task scheduling. As a developed research from

the original work, the proposed algorithm still stays on the idea that a virtual release time is introduced in order to obtain an earlier deadline for the target task. This idea is aimed to shorten task's response time. For the goal of reducing runtime overhead, the proposed algorithm introduces different techniques to moderate the advancing repeats that cause the runtime overhead in the original algorithm. There are two main techniques presented in this research. The first one is technique of limiting the advancing by the checking boundary. The checking boundary is actually calculated based-on the last starting time of the periodic task having the maximum period. The second one is technique of advancing by instance. Different from the original algorithm where the virtual release time is moved backward to the past slot by slot, the proposed algorithm advances the virtual release time instance by instance. Since moving by length of instances is mostly more than by time slot, the number of advancing repeat may be effectively decreased.

To achieve the goals, the research first introduces an theoretical algorithm and then implements the algorithm on both software and hardware designs. The theoretical algorithm is to explain fully the procedure, definitions, and idea of the proposed techniques. Then, it is proved to be effective on simulations where the runtime overhead and response time are calculated approximately based-on the estimation of instruction cycle of the ARM Cortex-A9 processor. The simulation environtment is set up on software for both periodic and aperiodic tasks. Each task set consists of periodic tasks with the total processor utilization (U_p) of 60% to 95% at intervals of 5% and aperiodic tasks with the total utilization of about 2% in the observation period (100,000 ticks). All the aperiodic servers are supposed to have the utilization $U_s = 1 - U_p$. Both orignal and enhanced algorithms are under the simulations to executed totally 80 periodic and 10 aperiodic task sets. The simulation results show that the new algorithm reduces significantly the runtime overhead of scheduling. At $U_p = 95\%$, the proposed algorithm can reduce approximately 56% runtime overhead compared to the original one while guaranteeing the response time.

After that, the software and hardware of the proposed algorithm are implemented based-on the ITRON operating system and the Zynq7000 FPGA ZedBoard where the ARM Cortex-A9 processor core is integrated. In the context of software and hardware implementations, the runtime overhead and response time are calculated actually on the real operation of the processing core. Both original and proposed algorithm are implemented and evaluated in software while the proposed one is designed on hardware only.

The hardware design is to confirm the actually effective performance of hardware processing compared to sofware processing in real-time systems. The software and hardware algorithms are set up to be evaluated in five scenarios. In each scenario, there are six periodic task sets utilizing up to 73% of processor utilization and ten times of aperiodic task's entering. According to the results of maximum overhead per tick of evaluations, the proposed algorithm can decrease around 20% overhead of the original algorithm in software implementation. In the same trend, the hardware implementation has approximately 13.7% lower overhead compared to the software implementation.

The positive results on evaluations eventually indicate that the proposed algorithm has entirely lower runtime overhead than the original one while keeping the same response times. With the successful hardware mechanism, the proposed algorithm also displays its feasibility in the real-time systems. Such achievements of this research show that this algorithm is worthy for a continuous research to become closer to the real systems. Applying the algorithm on multiprocessor systems may be an alternative approach.