Abstract

The correctness of hard real-time systems depends not only on its logical correctness but also, on its ability to meet all its deadline. Existing real-time systems use either a pure real-time scheduler or a real-time scheduler embedded as a real-time scheduling class in the scheduler of an operating system. Existing schedulers in multicore systems that support both real-time and non-real-time tasks, permit the execution of non-real-time tasks in all the cores with priorities lower than those of real-time tasks, but interrupts and softirqs associated with these non-real-time tasks can execute in any core with priorities higher than those of real-time tasks. In such systems, there is a need to develop a scheduler which minimizes the execution overhead of real-time tasks and ensures that the tasks runtime is not affected. To this end, we develop an integrated scheduler architecture on Linux kernel, called SchedISA, which executes hard real-time tasks with minimal interference from the Linux tasks while ensuring a fair share of CPU resources for the Linux tasks. We compared the execution overhead of real-time tasks in SchedISA implementing partitioned earliest deadline first (P-EDF) scheduling algorithm with SCHED_DEADLINEs P-EDF implementation. The experimental results show that the execution overhead of real-time tasks in SchedISA is considerably less than that in SCHED_DEADLINE.

Having developed a multicore scheduling architecture for scheduling hard real-time tasks, we explore existing multicore scheduling techniques and propose a new scheduling technique that is better in terms of efficiency and suitability than the existing multicore scheduling techniques. Existing real-time multicore schedulers use either global or partitioned scheduling technique to schedule real-time tasks. Partitioned scheduling is a static approach in which, a task is mapped to a per-processor ready queue prior to scheduling it and it cannot migrate. Partitioned scheduling makes ineffective use of the available processing power and incurs high overhead when real-time tasks are dynamic in nature. Global scheduling is a dynamic scheduling approach, where the processors share a single ready queue to execute the highest priority tasks. Global scheduling allows task migration which results in high scheduling overhead. In our work, we present a dynamic partitioning-based scheduling of real-time tasks, called DP scheduling. In DP scheduling, jobs of tasks are assigned to cores when they are released and remain in the same core till they finish execution. The partitioning in DP scheduling is done based on the slack time and priority of the existing jobs. If a job cannot be allocated to any core, then it is split, and executed on more than one core. DP scheduling technique attempts to retain good features of both global and partitioned scheduling without compromising on resource utilization, and at the same time, also tries to minimize the scheduling overhead. We have tested DP scheduling technique with EDF scheduling policy at each core, called DP-EDF scheduling algorithm, and implemented it using the concept of SchedISA. We compared the performance of DP-EDF with P-EDF and global EDF scheduling algorithms. Both simulation and experimental results show that DP-EDF scheduling algorithm has better performance in terms of resource utilization, and comparable or better performance in terms of scheduling overhead in comparison to these scheduling algorithms.