Real-Time Scheduling in Distributed Systems

Nguyen Duc Thai
Institute of Informatics, Slovak Academy of Sciences
Dubravska cesta 9, 84237 Bratislava, Slovakia
upsythai@savba.sk

Abstract

Real-time computer systems are expected to greatly benefit from multiprocessor technology and employing multiprocessor systems for real-time applications has shown to be difficult. A major obstacle is that scheduling algorithms for real-time multiprocessor systems are significantly more complex than for uniprocessor systems. In multiprocessor systems, the scheduling algorithm must not only specify the order of tasks, but also determine the specific processor to be used. Any practical scheduling algorithm for assigning real-time tasks to a multiprocessor system presents trade-off between its computational complexity and its performance. Finding a minimal schedule for a given set of real-time tasks in a multiprocessor system has shown to be ***NP****-hard. In this paper, we show several scheduling algorithms used in such multiprocessor systems and their performance with experimental results.*

1. Introduction

Real-time systems are those systems in which the correctness of the systems depends not only on the logical results but also on the time at which the results are produced [5]. Real-time tasks, which have more deterministic properties when compared to tasks in general computer systems [1,2,3], are characterized by deadline, release time and execution time. More and more real-time systems are relying on multiprocessors. Unfortunately, less is known about how to schedule multiprocessor-based real-time systems than for uni-processors. Optimal scheduling of real-time tasks on multiprocessor systems is known to be computationally intractable for large task sets. Any practical scheduling algorithm for assigning real-time tasks to a multiprocessor system presents trade-off between its computational complexity and its performance. The performance of a scheduling algorithm is measured in terms of the additional number of processors required to arrive at a schedule without deadline violations as compared to an optimal algorithm. Leung and Whitehead [4] proved that finding a minimal schedule for a given set of real-time tasks in a multiprocessor system is **NP**-*hard*. Therefore, research efforts have focused on the development of suitable heuristic algorithms, which can be efficiently implemented, required only a limited number of additional processors as compared to an optimal algorithm.

This paper is concerned with providing schedulability analysis of real-time tasks in multiprocessor systems. Section 2 describes model of distributed real-time systems and related works. Section 3 provides several algorithms for scheduling a set of tasks in multiprocessor systems. Section 4 shows experimental results and section 5 concludes the paper.

2. System Model

We consider here a distributed real-time system that consists of a set of m processors $P = \{P_1, P_2, \ldots, P_m\}$, a set of n tasks $T = \{T_1, T_2, \ldots, T_n\}$ that execute on m processors. Each processor has its own scheduler and schedulers on different processors coordinate with each other according to a synchronization protocol. Each task T_i has one or more subtasks $T_{i,1}, T_{i,2}, \ldots, T_{i,m}$ and subtasks can execute on different processors. These subtasks execute in order: first $T_{i,1}$ on processor P_1, then $T_{i,2}$ on processor P_2, and so on. Each subtasks $T_{i,j}$ has execution time $\tau_{i,j}$ and all instances of the subtasks have the same priority $\phi_{i,j}$.

3. Scheduling Algorithms

There are two strategies for scheduling real-time tasks on a multiprocessor system. In a global scheme each occurrence of a real-time task may be executed on a different processor. In contrast, a partitioning scheme enforces that all occurrences of a particular task are executed on the same processor. A partitioning scheme has several advantages over the global scheme. First, partitioning schemes are less complex since the overhead of multiprocessor scheduling merely consists in assigning tasks to processors. Note that the assignment is performed only once for each task, i. e., before the task is executed for the first time. Secondly, if the assignment of tasks to processors is completed, well-known uni-processor scheduling algorithms can be used for each processor.

The performance of a partitioning scheme is determined by two factors, the task assignment algorithm, which distributes tasks to the processors, and the scheduling algorithm, which determines the order of task executions