Journal of Parallel and Distributed Computing **60**, 113–133 (2000)
Article ID jpdc.1999.1594, available online at http://www.idealibrary.com on IDEAL®

A New Strategy for Improving the Effectiveness of Resource Reclaiming Algorithms in Multiprocessor Real-Time Systems

Indranil Gupta[1], G. Manimaran[2], and C. Siva Ram Murthy[3],

Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600 036, India

Received August 24, 1998; revised May 3, 1999; accepted August 25, 1999

The scheduling of tasks in multiprocessor real-time systems has attracted the attention of many researchers in the recent past. Tasks in such systems have deadlines to be met, and most real-time scheduling algorithms use worst case computation times to schedule these tasks. Many resources will be left unused if the tasks are dispatched purely based on the schedule produced by these scheduling algorithms, since most of the tasks will take less time to execute than their respective worst case computation times. Resource reclaiming refers to the problem of reclaiming the resources left unused by a real-time task when it takes less time to execute than its worst case computation time. Several resource reclaiming algorithms such as Basic, Early Start, and RV algorithms have been proposed in the recent past. But these pay very little attention to the strategy by which the scheduler can better utilize the benefits of reclaimed resources. In this paper, we propose an estimation strategy which can be used along with a particular class of resource reclaiming algorithms (such as Early Start and RV algorithms) by which the scheduler can estimate the minimum time by which any scheduled but unexecuted task will start or finish early, based solely on the start and finish times of tasks that have started or finished execution. We then propose an approach by which dynamic scheduling strategies, which append or reschedule new tasks into the schedules, can use this estimation strategy to achieve better schedulability. Extensive simulation studies are carried out to investigate the effectiveness of this estimation strategy versus its cost.

Key Words: multiprocessor real-time systems; dynamic scheduling; resource reclaiming; precedence constraints; resource constraints.

[1] Present address: Department of Computer Science, Cornell University, Ithaca, New York 14850. E-mail: gupta@cs.cornell.edu.

[2] Present address: Department of Electrical and Computer Engineering, Iowa State University, Ames, Iowa 50011. E-mail: gmani@iastate.edu.

[3] E-mail: murthy@iitm.ernet.in.

0743-7315/00 $35.00

1. INTRODUCTION

Because of their high performance and reliability, multiprocessors are emerging as powerful computing tools for safety-critical, real-time applications such as nuclear plant control and avionic control [10]. The problem of multiprocessor scheduling, which is to determine when and where a given task executes, has attracted considerable attention in the past [10–13]. Two classes of scheduling algorithms—static and dynamic—have emerged. In static algorithms, the assignment of tasks to processors and the time at which the tasks start execution are determined a priori. Static algorithms [3, 6] are often used to schedule periodic tasks with hard deadlines which are known a priori. The advantage is that if a solution is found, one can be sure that all deadlines will be guaranteed. However, this approach is not applicable to aperiodic tasks whose arrival times and deadlines are not known a priori. Scheduling such tasks in a multiprocessor real-time system requires dynamic scheduling algorithms. In dynamic scheduling [5, 7, 8, 13], when new tasks arrive, the scheduler dynamically determines the feasibility of scheduling these new tasks without jeopardizing the guarantees that have been provided for the previously scheduled tasks.

In general, for predictable execution, which is essential in a real-time system, schedulability analysis must be done before tasks start execution. For schedulability analysis, tasks' worst case computation times must be taken into account. A feasible schedule is generated if the timing, precedence, and resource constraints of all the tasks can be satisfied, i.e., if the schedulability analysis is successful. Tasks are dispatched according to this feasible schedule.

Dynamic scheduling algorithms can be either distributed or centralized. In a distributed dynamic scheduling scheme, tasks arrive independently at each processor. The local scheduler at the processor determines whether or not it can satisfy the constraints of the incoming task. If so, the task is accepted; otherwise, the local scheduler tries to find another processor to accept the task. In a centralized scheme, all the tasks arrive at a central processor called the *scheduler*, from which they are distributed to other processors in the system for execution. In this paper, we will assume a centralized scheduling scheme. The communication between the scheduler and the processors is through *dispatch queues* (DQs). Each processor has its own dispatch queue. This organization, shown in Fig. 1, ensures that the processors always find some tasks in the dispatch queues when they finish the execution of their current tasks. The scheduler runs in parallel with the processors, scheduling the newly arriving tasks, and periodically updating the dispatch queues. The scheduler has to ensure that the dispatch queues are always filled to their minimum capacity (if there are tasks left with it) for this parallel operation. This minimum capacity depends on the worst case time required by the scheduler to reschedule its tasks upon the arrival of a new task [4, 9]. The schedule constructed by the scheduler is assumed to be stored in a set of schedule queues (or SQs, one queue per processor), presumably in the scheduler's memory itself. The dispatch queues are updated by the scheduler from these schedule queues just before the invocation of the scheduling algorithm or when they become empty so that the processors can execute tasks in parallel with the scheduler's running.