Resource Sharing under Server-based Multiprocessor Scheduling

Sara Afshar, Moris Behnam
Mälardalen University, Västerås, Sweden
{sara.afshar, moris.behnam}@mdh.se

ABSTRACT

In this paper, we investigate a mechanism for handling resource sharing among tasks under a server-based scheduling technique in multiprocessor platforms, which combines partitioned and global scheduling to benefit a better scheduling method compared to conventional techniques.

1. INTRODUCTION

Semi-partitioned scheduling for multiprocessors benefits from both conventional global and partitioned approaches such that most tasks are assigned statically to processors similar to partitioned scheduling, while a low number of tasks are split and migrate among processors similar to global scheduling, [1, 2, 3]. Another recent multiprocessor scheduling approach is based on hierarchical scheduling, which utilizes servers and is called Synchronized Deferrable Servers (*SDS*) [4]. Under SDS, similar to the semi-partitioned approach, some tasks are bound to processors (*non-migrating tasks*), while others migrate among processors (*migrating tasks*). The key difference with semi-partitioned scheduling is that in SDS the migrated tasks are processed within servers allocated to processors. The major distinction between the semi-partitioned approach and SDS is that under SDS tasks which migrate between processors may run in any available server on any processor while in the semi-partitioned approach each part of a split task always executes on a specific processor which is determined during the partitioning phase. Therefore, SDS provides more flexibility to execute migrating tasks on processors that can improve the schedulability performance. However, in [4] it is assumed that tasks are independent, i,e., they do not share any resource. In this paper, we propose a resource sharing protocol for the case when tasks under the SDS multi-core hierarchical scheduling share resources with each other. The main challenge is to adjust the response time analysis presented in [4] to include the effect of resource sharing.

2. General Description

Our considered system consists of a set of *n* tasks that run on a set of *m* identical processors. One deferrable server is assigned to each processor that could provide a capacity during partitioning phase. We define a common replenishment period for all servers in the system. The scheduling policy includes two levels: (i) a fixed priority uniprocessor scheduling that schedules non-migrating tasks along with the server on each core and (ii) migrating tasks scheduling decision which determines in which server the non-migrating task execute. Next we develop our protocol rules based on the SDS structure [4] and inspiration from the synchronization protocol for semi-partitioned system [5].

1) Local resources are handled by uniprocessor protocols.
2) One global priority-ordered queue (*Q)* enqueues the ready or preempted migrating tasks. However, in each processor a local ready queue enqueues the non-migrating tasks.
3) After a migrating task is released, it is added to *Q*. The task at the head of *Q* executes on a ready server with the available capacity. If more than one ready server is available, the server with highest assigned capacity is chosen. If the lowest priority running task in any server has a priority lower than that of the task at the head of *Q*, it will be preempted.
4) A global queue is dedicated to each global resource to enqueue tasks from different processors which get blocked on the resource; however one local queue is assigned to each processor to enqueue local non-migrating tasks that are granted access to different global resources. The tasks in the global resource queues can be migrating or non-migrating tasks. However, the migrating tasks which are granted access to their requested resource will be inserted and wait in *Q*.
5) All resources that are requested in the migrating tasks are assumed global since they can be requested in any processor.
6) The priority of a task accessing a global resource is boosted to maximum priority to decrease the blocking times of tasks.
7) In order to prevent a migrating task holding a resource to migrate to another processor, an overrun approach is performed if the capacity of the server is finished and the task is in a global critical section.

In our ongoing work we are performing system analysis, and the challenge is to find an upper bound of the response time of migrating tasks which share resources with other parts of the system and can execute in any server on any processor.

3. REFERENCES

[1] J. Anderson, V. Bud, and U. Devi, "An EDF-based scheduling algorithm for multiprocessor soft real-time systems," *(ECRTS'05)*.

[2] S.Kato and N. Yamasaki, "Semi-partitioned fixed-priority scheduling on multiprocessors," *(RTAS'09)*.

[3] N. Guan, M. Stigge, W. Yi, and G. Yu, "Fixed-priority multiprocessor scheduling with Liu and Layland's utilization bound," *(RTAS'10)*.

[4] H. Zhu, S. Goddard, and M. Dwyer, "Response time analysis of hierarchical scheduling: The synchronized deferrable servers approach *(RTSS'11)*.

[5] S. Afshar, F. Nemati, and T. Nolte, "Resource sharing under multiprocessor semi-partitioned scheduling," *(RTCSA'12)*.