CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Poster

Towards the Combination of Work-Stealing and Semi-Partitioned Scheduling for Parallel Tasks

Cláudio Maia
Luís Nogueira
Patrick Meumeu Yomsi
Luis Miguel Pinho

CISTER-TR-151103

Towards the Combination of Work-Stealing and Semi-Partitioned Scheduling for Parallel Tasks

Cláudio Maia, Luís Nogueira, Patrick Meumeu Yomsi, Luis Miguel Pinho

CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail:

http://www.cister.isep.ipp.pt

Abstract

We consider the scheduling of sequential and fork-join hard real-time tasks by following semi-partitioned scheduling. In this paper we briefly introduce a modification to the limited migrative model in order to support parallel tasks.

Towards the Combination of Work-Stealing and Semi-Partitioned Scheduling for Parallel Tasks

Cláudio Maia, Luís Nogueira, Patrick Meumeu Yomsi, Luís Miguel Pinho

CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Portugal
{crrm, lmn, pamyo, lmp}@isep.ipp.pt

1 Motivation and Problem

Executing parallel tasks on multicore platforms brings several challenges into the real-time systems domain. Nevertheless, several scheduling approaches have already been proposed in the literature specifically considering this type of tasks. Among the proposed approaches, some use decomposition-based techniques where parallel tasks are treated as a set of sequential tasks so that schedulability analyses developed for sequential tasks can be applied. Others, perform the schedulability by using response-time analysis techniques, or by analysing the resource augmentation bounds of the underlying partitioned or global scheduler. An alternative class of schedulers which has proven to be viable and that has not yet been explored in the context of parallel tasks is that of semi-partitioned schedulers. In this class, a subset of tasks is fully partitioned to the cores at design time (i.e., no migration is allowed for these tasks at runtime) and the remaining tasks are scheduled using a global scheduling approach.

2 This research

We consider the scheduling of sequential and fork-join hard real-time tasks by following semi-partitioned scheduling. We propose a modification to the limited migrative model in order to support parallel tasks. In this modification, some tasks are fully partitioned to the cores at design time and each remaining task (which cannot be assigned to any core without violating a timing requirement) may execute on a subset of cores by following an execution pattern defined at design time and guaranteed during runtime. Fully partitioned tasks are referred to as *non-migrating tasks* and remaining tasks are referred to as *migrating tasks*. Migrating tasks are treated as multi-frame tasks [1]. A multi-frame task is modeled by using a finite list of worst-case execution time values corresponding to successive jobs (a.k.a. frames), repeating in a cyclic manner for all subsequent jobs. To avoid migration overheads, a copy of each migrating task is kept in each core where it has a job to execute. In addition, *work-stealing* is allowed among the cores sharing a copy of a task. Whenever a job of a migrating task is *running* on a core (say *core A*) and another core (say *core B*) with a copy of this task is idle, then core B can help in the execution of the parallel task by stealing workload from core A. Prior to applying stealing, a demand bound function (DBF) analysis is performed on core B so as to guarantee that it can accommodate the "steal workload" without jeopardizing the schedulability of the other tasks already executing on this core.

3 Proposed Approach

Our approach envisions three phases.

1. Assignment phase. In this phase, tasks are categorized into *light* (task utilization ≤ 0.5) and *heavy* (task utilization > 0.5) tasks. Then, a task-to-core assignment heuristic is applied to determine the non-migrating tasks. In this process, sequential tasks are evaluated first. The intuition behind this choice is to fill the capacity of the cores as much as possible with sequential tasks and favor the work stealing mechanism for parallel tasks in order to decrease their response times. At the moment, a pessimistic approach which evaluates parallel tasks as sequential tasks in the schedulability analysis is adopted. This point will be refined later on in our approach to answer the question: how to provide a tight and efficient evaluation of the schedulability of a parallel task on a multicore platform?

2. Offline scheduling phase. In this phase, the execution pattern of each migrating task (i.e., its execution sequence) is determined so as to meet all the timing requirements of the system. This process consists of mapping the jobs (frames) of each migrating task to the cores in an execution sequence. Because of this mapping, the schedulability on each core can be verified by using uniprocessor schedulability techniques. For example, assuming that each core is executing an Earliest Deadline First (EDF) scheduler, then a demand-bound analysis can be performed. If no pattern exists to schedule a migrating task, then the task set is deemed not schedulable.

3. Online scheduling phase. In this phase, the work-stealing mechanism from one core to another is implemented. During runtime an idle core (say *core B*) with a copy of a migrating task can contribute to the execution of this task by stealing workload from another core (say *core A*) and executing the (stolen) workload. Before stealing any workload, an admission test is performed on core B in order not to jeopardize the schedulability of the tasks already assigned to this core in Phase 2.

Concluding Remarks. A number of important properties are very encouraging in the proposed approach: (1) Work-stealing operations at runtime favor load-balancing among cores, and consequently a decrease the average response-times of parallel tasks; (2) The number of migrations among cores is limited as a LMM model is adopted and only cores sharing a task can steal workload from one another; (3) The initiative of stealing workload always comes from an idle core.

References

1. F. Dorin, P. M. Yomsi, J. Goossens, and P. Richard. Semi-partitioned hard real-time scheduling with restricted migrations upon identical multiprocessor platforms. In *RTNS*, 2010.

Acknowledgements: This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and when applicable, co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within project UID/CEC/04234/2013 (CISTER Research Centre); also by FCT/MEC and ERDF through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within project(s) FCOMP-01-0124-FEDER-020447 (REGAIN); also by FCT/MEC and the ESF (European Social Fund) through POPH (Portuguese Human Potential Operational Program), under PhD grant SFRH / BD / 88834 / 2012.