Technical Report

Conjecture about global fixed-priority preemptive multiprocessor scheduling of implicit-deadline sporadic tasks: The utilization bound of SM-US(sqrt(2)-1) is sqrt(2)-1

Björn Andersson

HURRAY-TR-100512
Version:
Date: 05-29-2010

Conjecture about global fixed-priority preemptive multiprocessor scheduling of implicit-deadline sporadic tasks: The utilization bound of SM-US(sqrt(2)-1) is sqrt(2)-1

Björn Andersson

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

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

Abstract

Consider global fixed-priority preemptive multiprocessorscheduling of implicit-deadline sporadic tasks. I conjecturethat the utilization bound of SM-US(sqrt(2)-1) is sqrt(2)-1.

Conjecture about global fixed-priority preemptive multiprocessor scheduling of implicit-deadline sporadic tasks: The utilization bound of SM-US($\sqrt{2}-1$) is $\sqrt{2}-1$

Björn Andersson
CISTER/IPP-Hurray Research Unit at the Polytechnic Institute of Porto
Email: bandersson@dei.isep.ipp.pt

***Abstract*—Consider global fixed-priority preemptive multiprocessor scheduling of implicit-deadline sporadic tasks. I conjecture that the utilization bound of SM-US($\sqrt{2}-1$) is $\sqrt{2}-1$.**

I. Preliminaries

Consider the problem of preemptively scheduling n sporadically arriving tasks on $m \geq 2$ identical processors. A task τ_i is uniquely indexed in the range $1..n$ and a processor likewise in the range $1..m$. A task τ_i generates a (potentially infinite) sequence of jobs. The arrival times of these jobs cannot be controlled by the scheduling algorithm and are a priori unknown. We assume that the arrival time between two successive jobs by the same task τ_i is at least T_i. Every job by τ_i requires at most C_i time units of execution over the next T_i time units after its arrival. We assume that T_i and C_i are real numbers and $0 \leq C_i \leq T_i$. A processor executes at most one job at a time and a job is not permitted to execute on multiple processors simultaneously. The utilization is defined as $U_s = (1/m) \cdot \sum_{i=1}^{n} \frac{C_i}{T_i}$. The utilization bound UB_A of an algorithm A is the maximum number such that all tasks meet their deadlines when scheduled by A, if $U_s \leq UB_A$.

Global fixed-priority preemptive scheduling is a specific class of algorithms where each task is assigned a priority, a number which remains unchanged during the operation of the system. At every moment, the m highest-priority tasks are selected for execution among tasks that are ready to execute and has remaining execution. The scheduling decisions are therefore determined by the assignment of priorities to tasks. The priority-assignment scheme in the current state-of-art which offers the highest utilization bound is SM-US($2/(3+\sqrt{5})$; its utilization bound is $2/(3+\sqrt{5})$ [3]. It categorize a task as heavy or light. A task is said to be heavy if $\frac{C_i}{T_i}$ exceeds $2/(3+\sqrt{5})$ and a task is said to be light otherwise. Heavy tasks are assigned the highest priority and the light tasks are assigned a lower priority; the relative priority order among light tasks is given by SM; slack monotonic, meaning that if $T_i - C_i < T_j - C_j$ then τ_i is given higher priority than τ_j.

One can show (last page of [1]) that each priority assignment scheme which is scale-invariant and independent has a utilization bound at most $\sqrt{2}-1$. (A priority-assignment scheme is scale-invariant if the relative priority order of a priority assignment given does not change when we multiply T_i and C_i of all tasks by the same positive constant. A priority-assignment scheme is independent if $\text{priority}_\text{i} = f(T_i, C_i)$, that is the priority of a task τ_i depends only on its own parameters.)

II. The conjecture

Let SM-US($\sqrt{2}-1$) denote a priority-assignment scheme which categorized a task as heavy if $\frac{C_i}{T_i}$ exceeds $\sqrt{2}-1$ and a task is categorized as light otherwise. Heavy tasks are assigned the highest priority and the light tasks are assigned a lower priority; the relative priority order among light tasks is given by SM; slack monotonic, meaning that if $T_i - C_i < T_j - C_j$ then τ_i is given higher priority than τ_j.

I conjecture that the priority-assignment scheme SM-US($\sqrt{2}-1$) has the utilization bound $\sqrt{2}-1$.

III. Significance of the conjecture

If the conjecture would be true then we would have at our disposal a priority-assignment scheme that attains the best performance possible in the class of scale-invariant and independent priority-assignment schemes.

IV. The rationale for stating the conjecture

We can understand this conjecture by considering two task set examples. As a first example, consider tasks $\tau_1, \tau_2, \ldots, \tau_m$ with $T_i = 1$,$C_i = \sqrt{2}$-1 and $T_{m+1} = \sqrt{2}$ and $C_{m+1} = 2-\sqrt{2}$ to be scheduled on m processors. For these tasks, it holds that the utilization of each task is $\sqrt{2}-1$. Increasing the execution time by an arbitrarily small amount will cause a deadline miss for the case that all tasks arrive simultaneously. We conclude from this example that we cannot prove a higher utilization bound than $\sqrt{2}-1$ for the algorithm SM-US($\sqrt{2}-1$).

References

[1] B. Andersson and J. Jonsson, "The utilization bounds of partitioned and pfair static-priority scheduling on multiprocessors are 50%", Proceedings of the 15th Euromicro Conference on Real-Time Systems (ECRTS'03), pp. 33 - 40, 2003.

[2] L. Lundberg, "Analyzing Fixed-Priority Global Multiprocessor Scheduling", Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02), pp. 145-153, 2002.

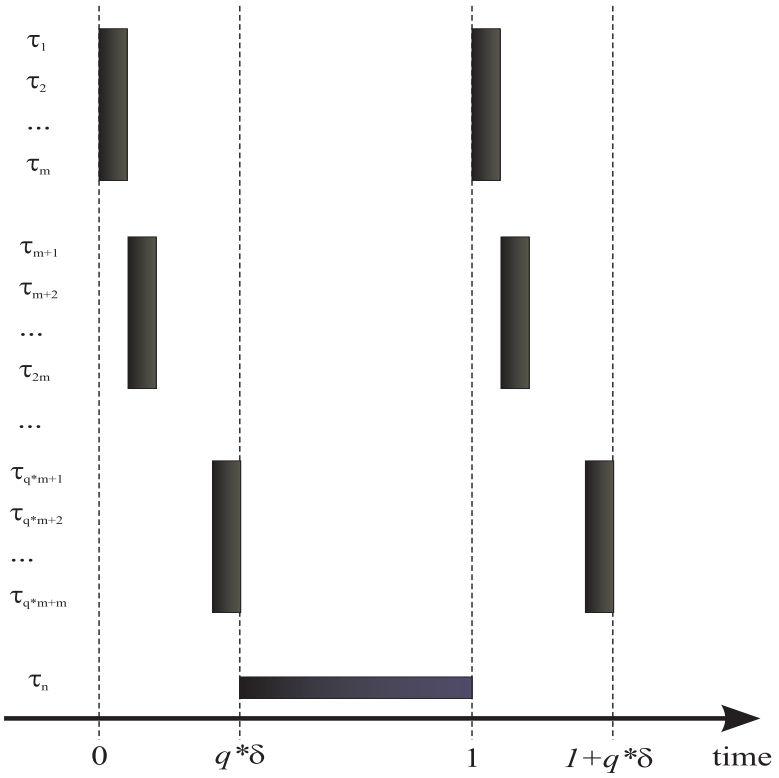

Fig. 1. [Adapted from [2]] An example of a task set where RM-US(0.375) performs poorly. All tasks arrive at time 0. Tasks $\tau_1, \tau_2, \ldots, \tau_m$ are assigned the highest priority and execute on the m processors during [0,δ). Then the tasks $\tau_{m+1}, \tau_{m+2}, \ldots, \tau_{2m}$ execute on the m processors during [δ,2δ). The other groups of tasks execute in analogous manner. Task τ_n executes then until time 1. Then the groups of tasks arrive again. The task set meets its deadlines but an arbitrarily small increase in execution times causes a deadline miss.

[3] B. Andersson, "Global Static-Priority Preemptive Multiprocessor Scheduling with Utilization Bound 38%", Proceedings of the 12th International Conference On Principles Of DIstributed Systems (OPODIS'08), pp. 73-88, 2008.