Technical Report

Two-type Heterogeneous Multiprocessor Scheduling: Is there a Phase Transition? (Extended Abstract)

Gurulingesh Raravi
Björn Andersson
Konstantinos Bletsas

HURRAY-TR-110503
Version:
Date: 05-30-2011

Two-type Heterogeneous Multiprocessor Scheduling: Is there a Phase Transition? (Extended Abstract)

Gurulingesh Raravi, Björn Andersson, Konstantinos Bletsas

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 the problem of non-migratively scheduling a set of implicit-deadline sporadic tasks to meet all deadlines on a two-type heterogeneous multiprocessor platform. We ask the following question: Does there exist a phase transition behavior for the two-type heterogeneous multiprocessor scheduling problem? We also provide some initial observations via simulations performed on randomly generated task sets.

Two-type Heterogeneous Multiprocessor Scheduling: Is there a Phase Transition?

Gurulingesh Raravi*, Björn Andersson†* and Konstantinos Bletsas*
*CISTER-ISEP Research Center, Polytechnic Institute of Porto, Portugal
†Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA
Email: *{ghri, baa, ksbs}@isep.ipp.pt, †baandersson@sei.cmu.edu

I. Introduction

Consider the problem of non-migratively scheduling a set of implicit-deadline sporadic tasks to meet all deadlines on a two-type heterogeneous multiprocessor platform.

A. System Model and Assumptions

The system is as follows:

- **Computing Platform (denoted as Π):** The computing platform consists of m processors; Of those, $m_1 \geq 1$ are of type-1, and $m_2 \geq 1$ are of type-2, i.e., $m_1 + m_2 = m$. A processor is denoted as $\pi_j \in \Pi$, where $j \in \{1, \cdots, m\}$.
- **Task Set (denoted as τ):** The task set comprises n implicit-deadline sporadic tasks (i.e., for each task, its deadline is equal to its minimum inter-arrival time). A task is denoted as $\tau_i \in \tau$, where $i \in \{1, \cdots, n\}$.
- **Utilization (denoted as U):** The utilization of a task τ_i on a processor π_j is given by u_i^j, a non-negative real number.

The following assumptions are made:

- **No job parallelism:** A job can be executing on at most one processor at any given time instant
- **Independent tasks:** The execution of jobs are independent, i.e., they neither share any resources nor have data dependency and
- **No migration:** All the jobs released by a task must execute on the same processor to which the task is assigned.

B. Phase Transition

A behavior in which a given system transitions from one state to another is known as *phase transition* behavior. During the phase transition, certain properties of the system change "drastically". In context of real-time scheduling, one way to relate this concept is to reason about the difficulty of scheduling problems. We can say that when a scheduling problem satisfies certain property (i.e., when the system is in a certain phase), it is almost certain to schedule the task set and upon changing the property (the system enters a new phase), it is hardly possible to schedule the task set. For example, such a behavior has been observed for a uniprocessor non-preemptive scheduling problem. It has been shown that there exists a utilization threshold U^* such that, for large task sets, task sets with utilization $U < U^*$ can almost surely be scheduled and task sets with utilization $U > U^*$ almost surely cannot be scheduled [3]. It is also believed that such a behavior exists for identical multiprocessor scheduling problem [4]. We are interested in finding whether such a behavior exists for two-type heterogeneous multiprocessor platforms.

II. Open Problem

Does there exist a phase transition behavior for the two-type heterogeneous multiprocessor scheduling problem?

III. Some Insights

In a quest to find an answer to the question, we performed some simulations and did not observe the phase transition behavior in our simulations. We understand that these simulations/observations are not enough to answer the question and hence more work needs to be done in this regard. We brief our simulation setup and observations in this section.

We randomly generated the problem instances comprising the task set (with an upper bound of 15 tasks) and the computing platform (with an upper bound of 2 processors of each type). We then formulated the task assignment problem as Zero-One Integer Linear Program (ILP) as discussed in [1]. This formulation is shown in Figure 1. Here Z denotes the maximum capacity of any processor

Minimize Z subject to the following constraints :

C1.	$\sum_{j=1}^{m} x_i^j = 1$	$(i = 1, 2, \cdots, n)$
C2.	$\sum_{i=1}^{n} \left(x_i^j \cdot u_i^j \right) \leq Z$	$(j = 1, 2, \cdots, m)$
C3.	x_i^j is a non-negative **integer**	$(i = 1, 2, \cdots, n)$; $(j = 1, 2, \cdots, m)$

Figure 1. ILP formulation – ILP-Feas(τ, Π)

that is used and is set as the objective function (to be minimized). $Z \leq 1$ implies that the sum of utilization of tasks assigned to any processor is less than or equal to the available capacity on that processor – hence, $Z \leq 1$ indicates that the task set is feasible on the platform. The variable x_i^j (referred to as *indicator variable*) indicate the assignment of task τ_i to processor π_j, i.e., $x_i^j = 1$ implies that τ_i is (entirely) assigned to processor π_j, $x_i^j = 0$ implies that τ_i is not assigned to processor π_j. The first constraint $(C1)$ indicates that every task must be assigned to processors. The second constraint $(C2)$ indicates that no processor capacity should be used more than Z. The third constraint $(C3)$ indicates that the indicator variables must be non-negative integers.

We extracted only those problem instances that are feasible (i.e., a problem instance in which the task set

could be assigned on the platform without missing any deadlines when EDF [2] is used to schedule the tasks on each processor – ILP solver returns $Z \leq 1$ for such task sets). For each of the feasible problem instances, we computed the *success ratio* which is defined as follows:

$$\text{success ratio} = \frac{N_{succ}}{N_{valid}}$$

where, N_{succ} denotes the number of assignments that meet all the deadlines of the tasks and N_{valid} denotes the total number of possible *valid* assignments. A valid assignment is one in which (i) no task is left unassigned and (ii) the task assignment in one valid assignment is different from other valid assignments. All the possible valid assignments are generated using exhaustive enumeration.

We then plotted our observations for 10000 feasible task sets with Z on X-axis and 'average success ratio' (for each Z) on Y-axis as shown in Figure 2. As we can see from the graph, there is a gradual decrease in the value of 'average success ratio' and hence no sharp threshold on a particular value of Z where 'average success ratio' reduces significantly. The fluctuation in the 'average success ratio' in the initial half of the graph (where $0 < Z \leq 0.45$) can be attributed to the fact that our task set generator generated very few task sets for which ILP solver gave the output $0 < Z \leq 0.45$ — to be precise, among 10000 task sets, only 250 were in this category. Hence, we believe that with a more balanced task set generator, we will not observe those fluctuations.

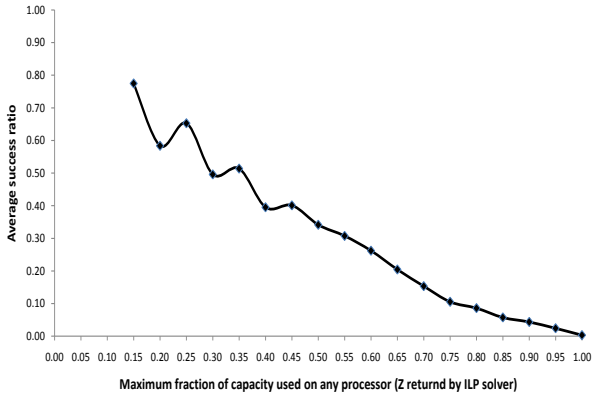

Figure 2. Average success ratio of randomly generated 10000 (feasible) task sets

From the observations made, we tend to believe that with the parameters that we have chosen, it may not be possible to observe the phase transition behavior if there is one. However, these are initial observations and we need to carry out more work to answer the following questions:

1) Does there exist a phase transition behavior? If there is one, then with what parameters we can observe such a behavior?
2) If there is no phase transition, then what is its implication considering the fact that such a behavior has been observed in the past for a uniprocessor scheduling problem [3][4]?

ACKNOWLEDGMENTS

This work was partially supported by the REHEAT project, ref. FCOMP-01-0124-FEDER-010045, funded by FEDER funds through COMPETE (POFC - Operational Programme Thematic Factors of Competitiveness), National Funds (PT) through FCT - Portuguese Foundation for Science and Technology and REJOIN project of FLAD (Luso-American Development Foundation).

REFERENCES

[1] S. Baruah, *Task partitioning upon heterogeneous multiprocessor platforms*, 10th IEEE International Real-Time and Embedded Technology and Applications Symposium (2004).

[2] C. L. Liu and J. W. Layland, *Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment*, Journal of the ACM (1973).

[3] S. Gopalakrishnan, M. Caccamo and L. Sha, *Sharp Thresholds for Scheduling Recurring Tasks with Distance Constraints*, IEEE Transactions on Computers (2008).

[4] S. Gopalakrishnan, *A sharp threshold for rate monotonic schedulability of real-time tasks*, 1st International Real-Time Scheduling Open Problems Seminar (2010).