IPP HURRAY!

www.hurray.isep.ipp.pt

Technical Report

A conjecture about provably good task assignment on heterogeneous multiprocessor platforms but with a stronger adversary

Gurulingesh Raravi

Björn Andersson

Konstantinos Bletsas

HURRAY-TR-110502

Version:

Date: 05-30-2011

A conjecture about provably good task assignment on heterogeneous multiprocessor platforms but with a stronger adversary

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 scheduling a set of implicit-deadline sporadic tasks to meet all deadlines on a heterogeneous multiprocessor platform. We use an algorithm proposed in [1] (we refer to it as LP-EE) from state-of-the-art for assigning tasks to heterogeneous multiprocessor platform and (re-)prove its performance guarantee but for a stronger adversary.We conjecture that if a task set can be scheduled to meet deadlines on a heterogeneous multiprocessor platform by an optimal task assignment scheme that allows task migrations then LP-EE meets deadlines as well with no migrations if given processors twice as fast. We illustrate this with an example.

A conjecture about provably good task assignment on heterogeneous multiprocessor platforms but with a stronger adversary

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

***Abstract*—Consider the problem of scheduling a set of implicit-deadline sporadic tasks to meet all deadlines on a heterogeneous multiprocessor platform. We use an algorithm proposed in [1] (we refer to it as LP-EE) from state-of-the-art for assigning tasks to heterogeneous multiprocessor platform and (re-)prove its performance guarantee but for a stronger adversary. We conjecture that if a task set can be scheduled to meet deadlines on a heterogeneous multiprocessor platform by an optimal task assignment scheme that allows task migrations then LP-EE meets deadlines as well with no migrations if given processors twice as fast. We illustrate this with an example.**

***Keywords*-heterogeneous multiprocessor, task migrations, real-time scheduling.**

I. Introduction

A heterogeneous multiprocessor platform is a computer system where (i) not all processors are of the same type and (ii) the execution time of a task depends on the processor on which it executes. Many chip makers offer or plan to offer products for computers with different types of processors. The Cell processor is a single chip comprising one main processor (Power4) and eight so-called synergistic processors (optimized for executing SIMD instructions) [2]. NVIDIA (and also AMD) offers general purpose graphics processor units which together with a normal processor are found in most personal computers today [3]. The Intel Sandy Bridge processor [4] is a single chip comprising an x86 multicore processor and a graphics processor. The AMD Fusion processor is a planned single chip comprising an x86 multicore processor and a set of accelerator processors for both embedded platforms [5] and desktops [6]. In a joint effort, ARM and NVIDIA are planning to offer chips comprising one general-purpose and one graphics processor [7]. It is clear that the above mentioned chips are key components in heterogeneous multiprocessor systems and such systems are increasingly used in practice.

An algorithm for deciding whether or not an implicit-deadline task set can be scheduled on a heterogeneous platform exists [8] but it assumes that tasks can migrate. This assumption is often unrealistic in practice, since processors with different functionalities typically have different instruction sets. Thus, the problem of assigning tasks to processors and then scheduling them with a uniprocessor scheduling algorithm (i.e., without migration) is of much greater practical significance. It requires solving two subproblems: (i) assigning tasks to processors and (ii) once tasks are assigned to processors, performing a uniprocessor scheduling on each processor. The latter problem is well-understood (e.g., one may use Earliest Deadline First scheduling [9]) – the difficult part is the task assignment.

The task assignment on a heterogeneous multiprocessor platform is modeled as Zero-One Integer Linear Programming (ILP) in [1][10]. Such a formulation can be solved directly but has high computational complexity. In particular, the decision problem ILP is NP-complete and even with knowledge of the structure of the constraints in the modeling of heterogeneous multiprocessor scheduling, no polynomial-time algorithm is known ([11], p. 245). Via relaxation of ILP formulation to Linear Program (LP) and certain tricks [12], better time-complexity can be attained [1][10]. (Polynomial time-complexity for the algorithm in [10] and for the special case of fixed number of processors, the algorithm in [10] has polynomial time-complexity as well). Both approaches [1][10] offer a performance guarantee that if a task set can be scheduled to meet deadlines on a heterogeneous platform by an optimal task assignment scheme that does not allow task migrations then these approaches meet deadlines as well without allowing task migrations if given processors twice as fast.

In this paper, we use the approach proposed in [1] (for convenience, we refer to it as *Linear Programming with Exhaustive Enumeration*, abbreviated as LP-EE, described in Section III-A), and (re-)prove its performance guarantee but for a stronger *adversary* (i.e., the set of algorithms against which we evaluate the performance of our algorithm) that allows task migrations. We conjecture that, if a task set can be scheduled to meet deadlines on a heterogeneous platform by an optimal task assignment scheme that allows task migrations then LP-EE meets deadlines as well without allowing task migrations if given processors twice as fast. We would like to reiterate that, the claim in this paper is stronger than the previous state-of-the-art approaches [1][10], as the adversary is more powerful since it allows task migrations.

II. System Model and Assumptions

A. System Model

We consider the problem of scheduling implicit-deadline sporadic tasks on a heterogeneous multiprocessor platform. The system is specified as follows:

- **Computing Platform (denoted as Π):** The computing platform consists of m processors. A processor is denoted as $\pi_j \in \Pi$, where $j \in \{1, \cdots, m\}$.

Minimize $\mathbb{U}$ subject to the following constraints :

C1.	$\sum_{j=1}^{m} x_i^j = 1$	$(i = 1, 2, \cdots, n)$
C2.	$\sum_{i=1}^{n} \left(x_i^j \cdot u_i^j\right) \leq \mathbb{U}$	$(j = 1, 2, \cdots, m)$
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)

Minimize $\mathbb{U}$ subject to the following constraints :

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

Figure 2. LP formulation – LP-Feas(τ, Π)

- **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.

B. Assumptions

We make the following assumptions:

- **Independent tasks:** The executions of jobs are independent, i.e., they do not share any resources and do not have any data dependency.
- **Migrations:** In our approach, we constrain the system by assuming that the tasks are not allowed to migrate between processors. However, in our adversary, we relax this constraint on the system by allowing jobs to migrate between processors thereby making the adversary more powerful.
- **No job parallelism:** A job can be executing on at most one processor at any time instant.

III. The Methodology: Linear Programming with Exhaustive Enumeration (LP-EE)

A. Background and Previous Result

We briefly describe the approach proposed in [1] before proceeding to discuss how we intend to use it and (re-)prove its performance guarantee for a stronger adversary.

In [1], the problem of assigning tasks to processors has been formulated as Zero-One ILP as shown in Figure 1. Here $\mathbb{U}$ denotes the maximum capacity of any processor that is used and is set as the objective function (to be minimized). $\mathbb{U} \leq 1$ implies that the sum of utilization of tasks assigned to any processor is less than or equal to the available capacity on that processor. 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 (such tasks are referred to as *integrally assigned tasks*), $x_i^j = 0$ 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 $\mathbb{U}$. The third constraint ($C3$) indicates that the indicator variables must be non-negative integers.

Since, ILP is NP-complete, the formulation is relaxed to LP by allowing the indicator variables to be non-negative **real numbers** (instead of just 0 or 1). The relaxed LP formulation is shown in Figure 2. As we can see, the only change in LP-Feas(τ, Π) formulation compared to ILP-Feas(τ, Π) formulation is that, the $C3$ constraint now allows x_i^j variables to take real numbers instead of just 0 or 1. The semantics of the x_i^j variable remain the same, in addition, $0 < x_i^j < 1$ indicates that *fraction* x_i^j *of* τ_i is assigned to processor π_j (such tasks are referred to as *fractionally assigned tasks*).

Then a two-step algorithm (referred to as LP-EE) is proposed to assign tasks on a heterogeneous platform. The algorithm is as follows:

1) The LP formulation is solved using an LP solver. If $x_i^j = 1$ then task τ_i is (integrally) assigned to processor π_j. Using certain tricks [12], it is shown that there exists a solution to LP-Feas(τ, Π) in which all but at most $(m - 1)$ tasks are integrally assigned to processors.
2) The remaining at most $(m - 1)$ tasks are integrally assigned on the remaining capacity of the processors using exhaustive enumeration.

Finally, the performance guarantee of this algorithm is proven which is stated as Lemma 1 below.

Lemma 1. ***(from Theorem 3 in [1])***
If there is a feasible mapping of a task set τ on a heterogeneous platform Π in which at most half the capacity of every processor is used, then it is guaranteed that LP-EE generates a feasible mapping (as well) of τ on Π.

B. New Result

Now, with the knowledge of the algorithm proposed in [1] and its performance guarantee, let us proceed to discuss the approach in which the adversary is migrative.

It is tempting to believe that the problem of scheduling a task set τ of n tasks on a heterogeneous multiprocessor platform Π of m processors allowing task migrations can be formulated as a Linear Programming problem using the LP-Feas(τ, Π) formulation shown in Figure 2. Though LP-Feas(τ, Π) formulation may seem correct at first sight, it is not, as shown by Example 1.

Example 1. *Consider a heterogeneous multiprocessor platform with four processors and a task set with three tasks whose utilizations are shown in Table I.*

We can see that the task set is infeasible since the utilization of τ_3 exceeds one on all the processors – it is impossible to schedule τ_3 to meet its deadline (unless it executes on multiple processors simultaneously, which our system model forbids). Yet, if we formulate this problem using LP-Feas(τ, Π) shown in Figure 2 and input it to an LP solver (such as IBM ILOG CPLEX [13]), the LP solver yields a solution with the values of variables shown in Table II and with $\mathbb{U} = 1$. Hence, the task set is erroneously deemed feasible on the given platform. The entries in

$(\tau_i \downarrow)(u_i^j \rightarrow)$	$\mathbf{u_i^1}$	$\mathbf{u_i^2}$	$\mathbf{u_i^3}$	$\mathbf{u_i^4}$
τ_1	1	1	$1+\epsilon$	$1+\epsilon$
τ_2	$1+\epsilon$	$1+\epsilon$	1	1
τ_3	2	2	2	2

Table I
A TASK SET TO ILLUSTRATE THE DRAWBACK OF LP-FEAS(τ, Π) FORMULATION.

$(\tau_i \downarrow)(\pi_i \rightarrow)$	π_1	π_2	π_3	π_4
τ_1	1	0	0	0
τ_2	0	0	1	0
τ_3	0	0.5	0	0.5

Table II
A SOLUTION (I.E., THE VALUES OF x_i^j VARIABLES) BY LP SOLVER TO THE TASK SET SHOWN IN TABLE I.

Table II are the values of indicator variables (i.e., x_i^j variables) and indicate the task assignment to processors – e.g., task τ_1 is assigned to processor π_1.

As illustrated in Example 1, the problem with the LP-Feas(τ, Π) formulation shown in Figure 2 is that it outputs a solution in which a task (whose utilization is more than one, τ_3 in this example) is assigned such that it is required to execute on more than one processor simultaneously.

Hence, to address this issue, we introduce one more constraint that prohibits job parallelism, i.e., a constraint that forces the assignment of tasks such that they are not allowed to execute on more than one processor simultaneously. The revised formulation namely, LP-Feas-Rev(τ, Π), is shown in Figure 3. The three constraints, i.e., $C1$, $C2$ and $C3$ are same as the ones discussed earlier (for LP-Feas(τ, Π) formulation) and the fourth constraint ($C4$) indicates that no task should be executed simultaneously on more than one processor.

Upon formulating our example using the revised LP-Feas-Rev() formulation and inputting it to an LP solver, we get an output indicating that the task set is infeasible on the given platform.

Conjecture 1. *A task set τ is feasible on a heterogeneous platform Π with task migrations permitted if and only if LP-Feas-Rev(τ, Π) gives a solution with $\mathbb{U} \leq 1$.*

Throughout this paper, we illustrate our claims with a (randomly generated) running example. Consider a task set τ with seven tasks to be scheduled on a heterogeneous computing platform Π with three processors. The utilization of tasks on each processor is shown in Table III.

Formulating this system as a linear program using LP-Feas-Rev(τ, Π) shown in Figure 3 and inputting it to an LP solver, we obtain the solution shown in Table IV and $\mathbb{U} = 0.999999$. It indicates that τ is feasible on Π. The values in Table IV indicate the task assignment to processors. For example, τ_1 is integrally assigned to π_2, τ_2 is fractionally assigned to π_2 and π_3, and so on.

Now, if we "divide the result in Conjecture 1 by 2", i.e., divide the utilization of every task on every processor by a factor of 2 (we refer this new task set as τ'), and divide the speed of every processor by 2 (we refer this new task set as Π'), we get the following result.

Minimize $\mathbb{U}$ subject to the following constraints :

C1.	$\sum_{j=1}^{m} x_i^j = 1$	$(i = 1, 2, \cdots, n)$
C2.	$\sum_{i=1}^{n} \left(x_i^j \cdot u_i^j\right) \leq \mathbb{U}$	$(j = 1, 2, \cdots, m)$
C3.	x_i^j is a non-negative real number	$(i = 1, 2, \cdots, n)$; $(j = 1, 2, \cdots, m)$
C4.	$\sum_{j=1}^{m} \left(x_i^j \cdot u_i^j\right) \leq \mathbb{U}$	$(i = 1, 2, \cdots, n)$

Figure 3. Revised LP formulation – LP-Feas-Rev(τ, Π)

$(\tau_i \downarrow)(u_i^j \rightarrow)$	$\mathbf{u_i^1}$	$\mathbf{u_i^2}$	$\mathbf{u_i^3}$
τ_1	0.087002	0.066455	1.952548
τ_2	1.294308	0.528062	0.906763
τ_3	0.802204	0.488072	1.240208
τ_4	0.448277	1.076216	1.825816
τ_5	0.573124	1.287740	0.982321
τ_6	0.148060	1.933626	0.654599
τ_7	0.331234	1.284164	0.814624

Table III
AN EXAMPLE TASK SET TO ILLUSTRATE CONCEPTS.

$(\tau_i \downarrow)(\pi_i \rightarrow)$	π_1	π_2	π_3
τ_1	0.000000	1.000000	0.000000
τ_2	0.000000	0.843599	0.156401
τ_3	0.000000	1.000000	0.000000
τ_4	1.000000	0.000000	0.000000
τ_5	0.126375	0.000000	0.873625
τ_6	1.0000	0.0000	0.0000
τ_7	1.0000	0.0000	0.0000

Table IV
A SOLUTION (I.E., THE VALUES OF x_i^j VARIABLES) BY LP SOLVER TO THE TASK SET SHOWN IN TABLE III.

Conjecture 2. *A task set τ' is feasible on a heterogeneous platform Π' with task migrations permitted if and only if LP-Feas-Rev(τ', Π) gives a solution with $\mathbb{U} \leq 0.5$.*

Our example task set (shown in Table III), after dividing by 2, is shown in Table V. Formulating this system as a linear program using LP-Feas-Rev(τ', Π) and inputting it to an LP solver, we obtain the solution that is same as the one shown in Table IV but with $\mathbb{U} = 0.499999$.

$(\tau_i \downarrow)(u_i^j \rightarrow)$	$\mathbf{u_i^1}$	$\mathbf{u_i^2}$	$\mathbf{u_i^3}$
τ_1	0.043501	0.033227	0.976274
τ_2	0.647153	0.264030	0.453381
τ_3	0.401102	0.244036	0.620103
τ_4	0.224138	0.538108	0.912908
τ_5	0.286561	0.643870	0.491160
τ_6	0.074030	0.966813	0.327299
τ_7	0.165616	0.642082	0.407311

Table V
TRANSFORMED TASK SET OBTAINED AFTER DIVIDING THE ORIGINAL TASK SET (SHOWN IN TABLE III) BY 2.

From here on, we will use the algorithm proposed in [1] (and eventually illustrate its performance guarantee for the migrative adversary). Since, LP-Feas-Rev(τ', Π) gives a solution with $\mathbb{U} \leq 0.5$ for a task set τ' that is feasible on a heterogeneous platform Π', it is easy to see that LP-Feas(τ', Π) also gives a solution with $\mathbb{U} \leq 0.5$. This is due to the fact that the LP-Feas() formulation is more relaxed than the LP-Feas-Rev() formulation (due to the absence of non-parallelism constraint $C4$ in it).

Conjecture 3. *If a task set τ' is feasible on a heterogeneous platform Π' with task migrations permitted then LP-Feas(τ', Π) gives a solution with $\mathbb{U} \leq 0.5$.*

Formulating the scheduling problem of the transformed task set τ' (shown in Table V) as a linear program using LP-Feas(τ', Π) (that allows job parallelism) shown in Figure 2 and inputting it to an LP solver, we obtain the solution which is the same as the one shown in Table IV and with $\mathbb{U} = 0.499999$.

We know the upper bound on the number of fractionally assigned tasks in the assignment corresponding to the solution given by an LP solver for the LP-Feas(τ', Π') formulation [1]:

Fact 1. *If there are tasks that were fractionally assigned, in accordance with the solution returned by LP solver after solving LP-Feas(τ', Π) formulation, then there can be at most $m - 1$ such tasks.*

We also know from [1] that, the remaining at most $m - 1$ tasks in τ' can be successfully assigned to processors using exhaustive enumeration without violating the schedulability test on any of the processors in Π – see Lemma 1 in Section III. This is possible because we know from Conjecture 3 that LP-Feas(τ', Π) has a solution with $\mathbb{U} \leq 0.5$ (which indicates that no processor has been used more than half its capacity). We can then use EDF [9] to schedule the tasks assigned on each processor. By combining all these observations, we get the next result.

Conjecture 4. *If a task set τ' is feasible on a heterogeneous platform Π' with task migrations permitted then LP-EE succeeds in assigning τ' on Π as well (with no task migrations), where each processor in Π is at most twice faster than the corresponding processor in Π'.*

Coming back to our example task set τ', the solution provided by the LP solver (see Table IV) has two fractionally assigned tasks, i.e., τ_2 and τ_5. We can see from the solution (ignoring the fractional assignment of τ_2 and τ_5) that π_1, π_2 and π_3 have remaining utilizations of 0.536216, 0.722737 and 1.000000 respectively on computing platform Π (where a processor speed is twice the corresponding processor speed in Π'). Hence, we can assign τ_2 to π_2 and τ_5 to π_1 without violating the EDF schedulability test on any processor.

IV. Summary

We used an existing approach [1] from state-of-the-art for assigning tasks on a heterogeneous multiprocessor platform and illustrated its performance guarantee for a stronger adversary. We conjectured that if a task set can be scheduled to meet deadlines on a heterogeneous platform by an optimal task assignment scheme that allows task migrations then LP-EE meets deadlines as well with no migrations if given processors twice as fast. We have left open the problem of mathematically proving the correctness of our claim.

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," in *Proceedings of the 10th IEEE International Real-Time and Embedded Technology and Applications Symposium*, 2004, pp. 536–543.

[2] IBM Corp., "The Cell Project at IBM Research," http://www.research.ibm.com/cell/.

[3] NVIDIA, "Dell and NVIDIA Workstation Solutions," http://www.nvidia.com/object/IO_16084.html.

[4] Intel Corporation, "The 2nd generation Intel Core processor family," http://www.intel.com/consumer/products/processors/core-family.htm.

[5] AMD Inc., "AMD Embedded G-Series Platform," http://www.amd.com/us/products/embedded/processors/Pages/g-series.aspx.

[6] ——, "The AMD Fusion Family of APUs," http://sites.amd.com/us/fusion/apu/Pages/fusion.aspx.

[7] IEEE Spectrum, "With Denver Project NVIDIA and ARM Join CPU-GPU Integration Race," http://spectrum.ieee.org/tech-talk/semiconductors/processors/with-denver-project-nvidia-and-arm-join-cpugpu-integration-race.

[8] S. Baruah, "Feasibility analysis of preemptive real-time systems upon heterogeneous multiprocessor platforms," in *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, 2004, pp. 37–46.

[9] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM*, vol. 20, pp. 46–61, 1973.

[10] S. Baruah, "Partitioning real-time tasks among heterogeneous multiprocessors," in *Proc. of the* 33^{rd} *International Conference on Parallel Processing*, 2004, pp. 467–474.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co, 1979.

[12] C. N. Potts, "Analysis of a linear programming heuristic for scheduling unrelated parallel machines," *Discrete Applied Mathematics*, vol. 10, pp. 155–164, 1985.

[13] IBM, "IBM ILOG CPLEX Optimizer," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.