Energy Efficient Allocation and Scheduling for DVFS-enabled Multicore Environments using a Multiobjective Evolutionary Algorithm

Zorana Banković
IMDEA Software Institute
Campus Montegancedo s/n
28223 Pozuelo de Alarcon, Madrid, Spain
zorana.bankovic@imdea.org

Pedro López-García
IMDEA Software Institute and CSIC
Campus Montegancedo s/n
28223 Pozuelo de Alarcon, Madrid, Spain
pedro.lopez@imdea.org

ABSTRACT

We present an approach for the automatic solving of the scheduling and allocation problem in multicore environments, as well as assigning optimal voltage and frequency levels to each core, using a multiobjective evolutionary algorithm (EA) where both energy consumption and the makespan are optimised and all deadlines are met. The main advantage of our approach is that we deal with all the aspects of the problem at once, which allows searching the whole solution space. In addition, the algorithm introduces the possibility of task migration, which is a novelty in EA-based approaches. Our results show that proper scheduling and allocation can provide significant energy savings in different scenarios: for our test case, and comparing to the well known YDS algorithm, up to 76% on average in the case of loose deadlines, and up 70% on average in the case of tight deadlines can be saved.

CCS Concepts

•**Computing methodologies** → **Planning for deterministic actions;** •**Information systems** → *Information systems applications;*

Keywords

Evolutionary algorithms, multiobjective optimization, scheduling, allocation, multicore

1. INTRODUCTION

The objective of this work is to efficiently solve the general problem of the scheduling and allocation of different tasks in multicore environments, and assign voltage and clock frequency to each core in a way the total energy consumption is minimised, while meeting all task deadlines. The tasks are characterised with their release time, execution time and deadline. In general, this problem is NP-hard, and it has been typically solved with heuristic algorithms since they can provide good solutions within an acceptable amount of time. For this reason, we use Evolutionary Algorithms (EAs). Furthermore, different levels of voltage and frequency are achieved by applying the Dynamic Voltage and Frequency Scaling (DVFS) technique. Since this technique reduces energy, but increases execution time, these two magnitudes are in conflict. For this reason, we use multiobjective optimisation.

All the existing solutions based on EA or Genetic Algorithms (GA) that treat a similar problem address it by dividing it into subproblems by first performing the scheduling, and then assigning the proper values of voltage and frequency. In this way the search space is reduced, which is not the case with our solution. Finally as far as we know, none of the existing solutions introduces the possibility of task migration.

We test our approach on multicore voltage and frequency scalable architectures designed by XMOS [2]. For this reason, our EA is based on an existing instruction-level energy model, which is described in [1]. However, the approach can easily be adapted to any multicore environment.

2. OUR PROPOSED APPROACH

We use the multiobjective NSGA-II algorithm to approximate the Pareto front, where the objectives are the energy consumption, calculated in the way presented in [1], and the execution time, calculated as the time spent until the last task finishes its execution. Both magnitudes should be minimised. However, all task deadlines have also to be met: the execution time objective is penalised by multiplying it by 10 for each task that does not met its deadline.

2.1 Representation of Individuals

Our proposed representation for individuals is shown in Fig.1. Each gene representing a unique task ID is followed by a gene representing the number of cycles of the task that is being executed in the current run. The order of task IDs represents the order of their temporal execution, i.e, scheduling. We also use negative two digit numbers to code the spatial allocation of the tasks in order to distinguish it from the tasks. The first digit represents the core where the tasks are being executed and the second one an encoding of the (V, f) state of that core. The tasks following the allocation code are executed on that coded location.

The population is randomly initialised: tasks are assigned to random cores and random number of cycles are assigned

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

GECCO '15 July 11-15, 2015, Madrid, Spain

© 2015 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-3488-4/15/07.
DOI: http://dx.doi.org/10.1145/2739482.2764645

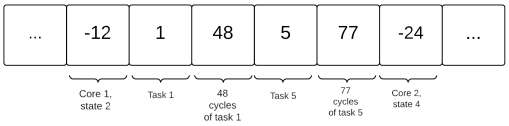

Figure 1: An example of (part of) an individual.

to tasks. However, the probability of selecting a core decreases as more tasks are assigned to it.

2.2 Solution Perturbations

2.2.1 The Crossover Operator

We have designed our own operator in the following way: each child takes the order of appearance of the tasks and their allocation from one of the parents. However, it can take the distribution of the number of cycles from any of the parents with equal probability.

2.2.2 The Mutation Operator

The mutation operator can perform different actions involving one or two tasks. Consider two tasks i and t. In each generation we perform one of the following operations with the same probability: *swapping:* i and t, together with their corresponding number of cycles, change their positions in the solution; *moving*: move i to a random position j; and *changing the number of cycles:* assigns a different number of cycles to task i in all of its appearances.

3. EXPERIMENTAL EVALUATION

The great majority of the existing work on applying DVFS concentrates only on dynamic power. However, it is not beneficial to scale down voltage and frequency indefinitely: as we keep decreasing the dynamic power, the static power is increased, which at some point becomes the predominant part, and as a result, the total power starts increasing again. This issue becomes important in the case when the tasks have loose deadlines. We have experimentally compared the energy savings obtatined by our EA-based algorithm with the YDS algorithm [3], which was designed having in mind only dynamic power. The results, obtained by repeating the same experiment 20 times, are presented in Figures 2 and 3 for the scenarios where task deadlines are loose and tight respectivelly. As we can observe from Figure 2, energy savings are significant when task deadlines are loose: on average, up to 76.57% after 200 generations.

Moreover, Figure 3 shows that our EA-based algorithm can find a feasible solution even if task deadlines are tight, obtaining average savings of up to 70.4% after 150 generations. We believe that the reasons for such improvements are the following, in this order: taking into account the static power, the energy-aware scheduling of the tasks, and the good characteristic of EA of not getting stuck in a local optimum, which can happen in the YDS algorithm.

4. CONCLUSIONS AND FUTURE WORK

We have presented an approach for energy-aware scheduling, allocation and optimal DVFS assignment of a set of tasks in a multicore environment based on EAs. Our experimental results show the great potential of our approach. However, the energy model we use introduces significant

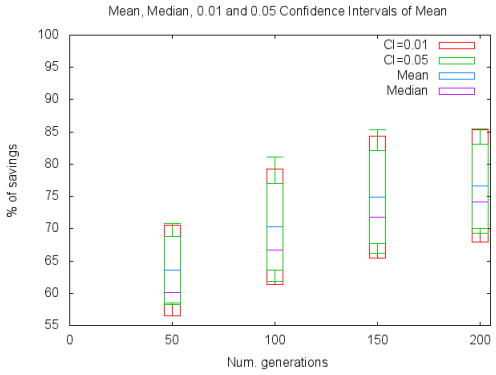

Figure 2: Energy savings of our EA algorithm vs. YDS when task deadlines are loose.

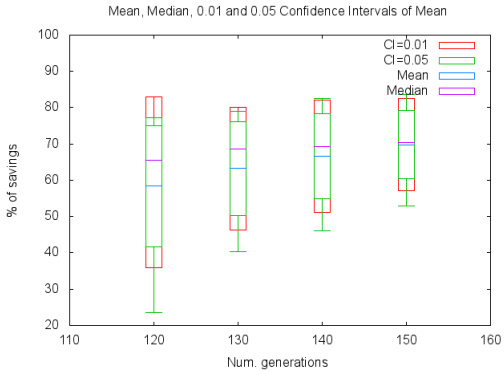

Figure 3: Energy savings of our EA algorithm vs. YDS when task deadlines are tight.

time overhead. In order to overcome this issue we plan to use a static analysis that estimates the energy consumed by concurrent tasks (without actually running them) which will significantly speed up our approach, since such estimations can be efficiently computed.

5. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union 7th Framework Programme under grant agreement 318337, ENTRA - Whole-Systems Energy Transparency, Spanish MINECO TIN'12-39391 *StrongSoft* and TIN'08-05624 *DOVES* projects, and Madrid TIC-1465 *PROMETIDOS-CM* project. We also thank XMOS Ltd. and Henk Muller in particular for providing benchmarks and useful information for our experimental evaluation.

6. REFERENCES

[1] S. Kerrison and K. Eder. Measuring and modelling the energy consumption of multithreaded, multi-core embedded software. *ICT Energy Letters*, pages 18–19, July 2014.

[2] XMOS. xcore : Architecture overview. Technical report, XMOS Ltd., 2013.

[3] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 0:374, 1995.