Sustainable Computing: Informatics and Systems 1 (2011) 314–328

Contents lists available at SciVerse ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Energy efficient task partitioning and real-time scheduling on heterogeneous multiprocessor platforms with QoS requirements

Bader N. Alahmad*, Sathish Gopalakrishnan

University of British Columbia, Vancouver, BC, Canada V6T 1Z4

ARTICLE INFO

Article history:
Received 11 July 2011
Accepted 9 September 2011

Keywords:
Approximation algorithm
Energy
Scheduling
Multiprocessor
Real-time
Quality of service

ABSTRACT

We address the problem of partitioning a set of independent, periodic, real-time tasks over a fixed set of heterogeneous processors while minimizing the energy consumption of the computing platform subject to a guaranteed quality of service requirement. This problem is NP-hard and we present a fully polynomial time approximation scheme for this problem. The main contribution of our work is in tackling the problem in a completely discrete, and possibly arbitrarily structured, setting. In other words, each processor has a discrete set of speed choices. Each task has a computation time that is dependent on the processor that is chosen to execute the task and on the speed at which that processor is operated. Further, the energy consumption of the system is dependent on the decisions regarding task allocation and speed settings.

1. Introduction

We consider a resource allocation problem where we are given a set of heterogeneous processors with discrete speed settings and a set of periodic, independent, real-time tasks. How do we partition the tasks across the available set of processors and choose appropriate speed settings for those processors such that we achieve minimum energy consumption while satisfying some specified quality of service requirement? In this setting, the energy consumed by the complete system depends on the task allocation and on the speeds of the processors. This problem is motivated by two important issues to consider in the design of current and future embedded systems: energy consumption and processor heterogeneity. Furthermore, the emphasis on a system model with completely discrete set of choices is based on architectural considerations as well as empirical evidence that suggests that such a model is indeed the appropriate model to apply.

Energy in embedded systems and especially battery-powered devices is a valuable resource whose expenditure needs to be kept at minimum in order to increase the lifetime of such systems. At the same time, tasks running on those systems should be appropriately serviced according to their computational and timeliness requirements. There are two major contributors to the overall energy consumption in a processor: (i) the *dynamic* power consumption due to switching activities and (ii) the *leakage* power consumption due to leakage current draw in CMOS circuits (Jejurikar et al. [1]) The former depends on the speed at which the processor is operating, while the latter is present whenever the processor is *on*, and is a constant. In addition to the energy consumed by the processor, the total energy consumed by the computing systems depends on the energy consumed by other devices and peripherals in the system (e.g., memory, I/O). This energy consumption is not dependent on the processor speed. This aspect of energy modeling is important because it allows us to capture energy consumption beyond what is specific to the processing units.

Most embedded systems now constitute multiple processors in order to increase the processing throughput. In addition, each processor can be configured to run at a speed (frequency) from a limited set of allowable speeds. Moreover, the processor's operating speed can be varied while the system is running without interrupting task execution. Heterogeneous computing systems consist of multiple processing components having different architectures and computing capabilities, interconnected using different connectivity paradigms. For example, the Nomadik™ platform (Wolf [2]) includes an ARM processor, a video accelerator and an audio accelerator, each of which is itself a heterogeneous multiprocessor. Such systems better meet the demand of applications with diverse requirements, because the computational requirements of a task might differ significantly on different processing elements, and heterogeneous systems allow tasks to be matched to computing elements that better serve their requirements.

* Corresponding author.
E-mail addresses: bader@ece.ubc.ca, baderalahmad84@gmail.com (B.N. Alahmad), sathish@ece.ubc.ca (S. Gopalakrishnan).
URLs: http://blogs.ubc.ca/bader (B.N. Alahmad), http://radical.ece.ubc.ca/sathish (S. Gopalakrishnan).

2210-5379/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.suscom.2011.09.001

In addition to energy, we view the system from a quality-of-service perspective. In general, quality of service is a measure of the maximum error a task might tolerate, or the minimum perceived precision in a dual sense. Quality-of-service is almost always associated with certain cost/precision trade-offs. For example, consider a search engine where the back-end, having received a user query, searches its index database for matching documents, ranks those documents (using some ranking algorithm) and returns the top N documents in ranked order (Baek and Chilimbi [3]). The service provider might consider, for example, executing less cycles running the ranking algorithm for the sake of faster response, in addition to saving energy on the search engine servers. The consequences of such *approximation* of the output of computation become relevant when the QoS metric is clearly defined. We can define the QoS loss metric as the percentage of user requests, compared to the full fledged-service case, that either return the same top N documents but in a different rank order or return different top N documents. Towards the goal of practically enabling such approximations by systematic means, Baek and Chilimbi [3] developed a programming framework called "Green", that allows programmers to approximate loops and expensive functions, and provides statistical QoS guarantees.

In a heterogeneous platform the assignment of tasks to processors may impact the end-user quality of service because certain processor types may be better suited to certain tasks. For example, executing a graphics task on a specialized graphics processor yields better results than performing the same operation on a general-purpose processor. In a system with heterogeneous compute units and multiple speed settings for each processor, a system architect may explore the trade-off between quality of service and energy efficiency. This is the stage of the design process that we target in this article.

To simplify the discussion, we can think of a platform with a fixed number of processors. We need to decide the processor type for each processor from a set of available processor types (permitting heterogeneity in the platform), then selecting a particular speed for a processor and finally assigning tasks to that processor. A task would have a certain worst-case execution time at the selected speed and each instance of a periodic task will consume a certain amount of energy (*active-time energy*) that depends on the type of processor selected and the associated speed of the processor. Additionally when a processor is idle it will consume some energy (*idle-time energy*).

Our energy expenditure model is central to our contribution. We relax many of the impractical assumptions underlying state-of-the-art solutions, including the work of Yang et al. [4], which is the closest to our efforts (see Section 7 for a discussion of related work). In a sense, prior models assume (1) continuity of speed levels on machines, (2) linearity of the worst case execution time (WCET) requirements of tasks with respect to processor speed, (3) constant worst case execution cycles (WCEC) of tasks with respect to speed levels on processors, (4) linear interpolation of the energy expenditure when calculations result in speed levels that are not available on the processor, in case of a discrete processor speed model, and (5) that energy expenditure on a processor depends solely on the total utilization of the processor. These assumptions are agnostic to the fact that different tasks exercise differential, and somewhat arbitrary quality of execution as speed varies on a processor. A consequence of such assumptions is that previous models cannot capture the energy as consumed by devices other than the processor, which is due to nonlinearities of task execution requirements and the overall energy consumption when tasks spend considerable fraction of their time interacting with I/O devices, such as memory and disk (I/O bound tasks). Therefore, those models are very difficult to scale to account for energy expenditure of the compute system as a whole (Section 2.6 provides a more elaborate discussion).

Table 1
Basic math benchmark (single iteration).

Frequency (GHz)	Average power (W)	Execution time (s)
0.8	86.5	32.34
1.6	89.9	31.7
2.1	94	31.61
2.8	100.2	31.75

Table 2
Fast Fourier transform (FFT) benchmark (1000 iterations).

Frequency (GHz)	Average power (W)	Execution time (s)
0.8	84.3	1080.4
1.6	88.96	538.2
2.1	94.78	410.23
2.8	104.11	307.74

To illustrate the abovementioned drawbacks in prior work and to motivate our work with discrete settings, we performed empirical studies using two applications from the MiBench embedded applications benchmark suite by Guthaus et al. [5]: *Basic Math* and *Fast Fourier Transform (FFT)* (with 64 random sinusoids and 65, 536 samples). We executed the former once and the latter 1000 times on an AMD® Phenom™ II X4 925 Quad Core Processor 2.8 GHz, which has discretely variable speeds in the set {800 MHz, 1.6 GHz, 2.1 GHz, 2.8 GHz} per core, and then measured the execution time as well as the energy consumption of each application at each speed step. The Basic Math benchmark includes some I/O operations and we found that speed scaling does not result in corresponding changes in execution time (Table 1). For this benchmark, we also found that energy consumption increases with increases in speed. For the FFT benchmark, which does not include the same amount of I/O as the Basic Math benchmark, execution time decreased as we would expect with an increase in processor speed and it is energy-efficient to run this application at a higher speed (Table 2). Our observations highlight the fact that variations in execution time and energy consumption are different for different applications. These variations are not easily captured by closed-form functions and require discrete modeling.

We present a system and task model (Section 2) that captures system implementations better than earlier models. We focus on determining static speed settings for processors and identifying a mapping between tasks and processors such that the energy expenditure of the system is minimized and the timeliness requirements of tasks respected. In addition, the task allocation scheme should guarantee that the overall QoS satisfies a specified requirement (Section 3).

Due to the practical needs of discrete and arbitrarily structured settings, our solution is combinatorial, and our algorithm employs a blend of matching and enumeration techniques, with dynamic programming being the general algorithmic tool. In summary, our contributions are

1. New model for energy expenditure that alleviates previous impractical assumptions.
2. An FPTAS for the (NP-hard in the ordinary sense) problem of allocating tasks on unrelated parallel machines (heterogeneous platform), where the number of machines is *fixed*, and the goal is to find an assignment of tasks to service classes and simultaneously build a platform from an assortment of given machine types, and then assign tasks (as they are equipped with service classes) to the machines comprising the platform such that the platform expends as minimum energy as possible while the