Microprocessors and Microsystems 43 (2016) 14–25

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Accurate energy modeling for many-core static schedules with streaming applications

Simon Holmbacka[a,*], Jörg Keller[b], Patrick Eitschberger[b], Johan Lilius[a]

[a] Faculty of Science and Engineering, Åbo Akademi University, Turku, Finland
[b] Faculty of Mathematics and Computer Science, FernUniversität in Hagen, Hagen, Germany

ARTICLE INFO

Article history:
Received 29 June 2015
Revised 8 November 2015
Accepted 13 January 2016
Available online 6 February 2016

Keywords:
Power management
Many-core systems
Power model
Static schedule

ABSTRACT

Many-core systems provide a great performance potential with the massively parallel hardware structure. Yet, these systems are facing increasing challenges such as high operating temperatures, high electrical bills, unpleasant noise levels due to active cooling and high battery drainage in mobile devices; factors caused directly by poor energy efficiency. Furthermore by pushing the power beyond the limits of the power envelope, parts of the chip cannot be used simultaneously – a phenomenon referred to as "dark silicon". Power management is therefore needed to distribute the resources to the applications on demand. Traditional power management systems have usually been agnostic to the underlying hardware, and voltage and frequency control is mostly driven by the workload. Static schedules, on the other hand, can be a preferable alternative for applications with timing requirements and predictable behavior since the processing resources can be more precisely allocated for the given workload. In order to efficiently implement power management in such systems, an accurate model is important in order to make the appropriate power management decisions at the right time. For making correct decisions, practical issues such as latency for controlling the power saving techniques should be considered when deriving the system model, especially for fine timing granularity. In this paper we present an accurate energy model for many-core systems which includes switching latency of modern power saving techniques. The model is used when calculating an optimal static schedule for many-core task execution on systems with dynamic frequency levels and sleep state mechanisms. We derive the model parameters for an embedded processor with the help of benchmarks, and we validate the model on real hardware with synthetic applications that model streaming applications. We demonstrate that the model accurately forecasts the behavior on an ARM multicore platform, and we also demonstrate that the model is not significantly influenced by variances in common type workloads.

1. Introduction

Computer systems often face a trade-off decision between performance and power dissipation. High power dissipation and fast execution usually lead to high energy consumption in modern many-core systems [12,13]. Execution speed is usually optimized by the programmer and compiler while minimizing energy is often left to the operating system which employs Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) using sleep states. However, an operating system with a dynamic scheduler has no knowledge about the application, its behavior and its timeline. In practice, the power management for dynamic schedules is performed with respect only to the workload level, which does not describe performance requirements. This means that the application is normally executed faster than what is actually required, and energy is being wasted because of the unnecessarily high power dissipation.

For applications consisting of a set of tasks with a predictable behavior and a known execution deadline, a schedule with the information when to execute which task at which speed can be devised at compile time (i.e. a static schedule). With hints from the application, the power management techniques can more precisely scale the hardware according to the software performance demands, and energy is minimized by eliminating unnecessary resource allocation. However, Power management is a practical interplay between software algorithms and physical hardware actions. This means that accessing power management techniques in general purpose operating systems introduces practical shortcomings such as access latency. Two separate mechanisms – DVFS and DPM

* Corresponding author. Tel.: +358 505310467.
E-mail addresses: sholmbac@abo.fi, simon.holmbacka@abo.fi (S. Holmbacka), jorg.keller@fernuni-hagen.de (J. Keller), patrick.eitschberger@fernuni-hagen.de (P. Eitschberger), johan.lilius@abo.fi (J. Lilius).

http://dx.doi.org/10.1016/j.micpro.2016.01.008
0141-9331/

– are currently used for minimizing the CPU power dissipation. As DVFS regulates voltage and frequency to minimize the dynamic power, DPM is used to switch off parts of the CPU to minimize the rapidly growing static power [16]. The techniques are therefore complementing each other and a minimal energy consumption is achieved by proper coordination of both techniques [1,24]. While both mechanisms have been evaluated in the literature [9,17], no work has been done to determine the practical latency of both DVFS and DPM on a single platform, and its impact on power management.

In this work we present an accurate energy model for static schedules in many-core systems using DVFS and DPM. The model is based on measurements of a single synthetic application on real hardware to conform with complete platform details and a realistic view of the static and dynamic power balance. In practical real-world systems, there is always a certain latency for utilizing DVFS and DPM both due to hardware and software implementations (in this paper Linux). Instead of focusing on eliminating or minimizing this latency, we chose to acknowledge this short coming in current computing systems, and learn how to integrate this detail in the system model. We account for the latency of using DVFS and DPM on a statically scheduled many-core system by including the timings in the decision making process of power management techniques. We validate the results by implementing a framework for synthetic workloads on real hardware (ARM multicore) running an unmodified Linux OS. The evaluation demonstrates that the model is able to accurately forecast the energy consumption of a selected static schedule under different workload configurations and different deadlines. We also demonstrate by experiments that the forecasts of the model remain stable if the benchmarks used to generate the model and the applications scheduled by the model differ in their characteristics.

The repeated execution of our synthetic applications (set of tasks with predictable workload and common deadline) models streaming applications with throughput requirements, which form a large class of applications for many-core processors. Considering several executions (rounds) together can allow further optimizations, which we include into our model. While our experiments only address a single platform, we note that the approach can be applied to other hardware platforms without re-engineering the core algorithm, as long as the platforms allow measurement for creating the model parameters.

The remainder of this article is structured as follows. In Section 2, we discuss related work. In Section 3, we investigate the latency and energy overhead of DVFS and DPM mechanisms as experienced by applications running on typical platform (ARM multicore) with a typical operating system (Linux). From these measurements we derive an energy model for task-based applications in Section 4. In Section 5, we evaluate the forecasting capabilities of this energy model with respect to execution on a real hardware. We extend the model towards streaming applications in Section 6, and present additional energy optimizations possible by scheduling several rounds of computation together. In Section 7, we give conclusions and an outlook onto future work.

2. Related work

DVFS and its efficiency for multi-cores has been studied in the past [9,17], but mostly the focus has been put directly on measuring the overhead of physically switching hardware states [17,27] including PLL locking, voltage level switching etc. Mazouz et al. present in [25] a frequency transition latency estimator called FTaLaT, which chooses a frequency depending on the current phase of a program. They argue that programs mostly have either CPU intensive phases in which the CPU is running on a high clock frequency or memory intensive phases in which the clock frequency can be decreased to save power. For very small memory intensive regions, it is favorable to ignore the frequency scaling because the switching delay would be higher than the length of the memory phase. They evaluate their estimator with a few micro-benchmarks (based on OpenMP) on different Intel machines, and they show that the transition latency varies between 20 and 70 microseconds depending on the machine. As the total switching latency is the sum of both hardware and software mechanisms, we study in this paper the practical aspects of switching latency in both DVFS and DPM for off-the-shelf operating systems running on real hardware. Influences of user space interaction and the kernel threads which control the power saving mechanisms are studied, and related to the effects on the energy consumption.

The paper of Schöne et al. [29] describes the implementation of the low-power states in current x86 processors. The wake-up latencies of various low-power states are measured and the results are compared with the vendor's specifications that are exposed to the operating system. The results show fluctuations e.g. depending on the location of the callee processor. Their work complements ours, but rather than using the x86 architecture we focus on mobile ARM processors with less support for hardware power management.

Algorithms for minimizing energy based on power and execution time have been presented in previous work such as [3,9,10]. Cho et al. define an analytical algorithm for expressing dynamic and static power in a multi-core system with multiple frequency levels. The minimum-energy-operation point is then calculated by determining the first order derivative of the system energy with respect to time. The mathematical expression defined in [3] exploits the task parallelism in the system to determine the amount of processing elements required, and hence influencing the static power dissipation. In our work, we define the system power model based on experiments on real hardware rather than analytical expressions in order to tailor the model closer to real-world devices.

The work in [10] uses similar reasoning to determine an *energy efficient frequency* based on the timing guarantees and available resources for real-time tasks. Their mechanism was able to map a certain number of replica tasks in a multi-core system in order to parallelize the work to an optimal number of cores running on an optimal frequency. Similar to [3], the authors used a bottom-up model to define the power as an analytical expression without taking temperature into account.

The work in [20] defines an algorithm for calculating the minimum energy consumption for a microprocessor when taking both dynamic power and static power into account in a system with DVFS and DPM. The authors define an analytical expression for the power dissipation and divides the workload for a given timeline into active time slots and sleep slots, and calculates the total energy over a given time with a given power dissipation for each slot.

In [21] an Energy-Aware Modeling and Optimization Methodology (E-AMOM) framework is presented. It is used to develop models of runtime and power consumption with the help of performance counters. These models are used to reduce the energy by optimizing the execution time and power consumption with focus on HPC systems and scientific applications. Our approach follows the same principle, but instead we use a top-down power model based on real experiments rather than analytical expressions. We also account for the latency of both DVFS and DPM which, as explained, becomes important when the time scale is shrinking.

Gerards and Kuper [6] describe various possibilities and techniques to reduce the energy consumption of a device (e.g. a processor core) under real-time constraints. They describe the problems and present optimal solutions for DPM-based and also combined DPM and DVFS approaches when both the energy and time for scaling the frequency and shutdown or wakeup a core are considered. The theoretical part is close to ours but they only focus