J. Parallel Distrib. Comput. ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Keep it cool and in time: With runtime monitoring to thermal-aware execution speeds for deadline constrained systems

Kai Lampka *, Björn Forsberg, Vasileios Spiliopoulos

Department of Information Technology, Uppsala University, Sweden

HIGHLIGHTS

- Dynamic power management for reducing heat emission in multicores.
- Run-time monitoring for tracking real-time workloads with respect to a bound.
- Introduces the concept of a worst-case ready queue for computing speed assignments.
- Speed assignment computation scheme that guarantee timing correctness of workloads.

ARTICLE INFO

Article history:
Received 29 September 2015
Received in revised form
18 January 2016
Accepted 2 March 2016
Available online xxxx

Keywords:
Real-time computing
Multicore architectures
Dynamic Voltage Frequency Scaling
Dynamic power and temperature management
Run-time monitoring
Online real-time scheduling

ABSTRACT

The Dynamic Power and Thermal Management (DPTM) system of Dynamic Voltage Frequency Scaling (DVFS) enabled processors compensates peak temperatures by slowing or even powering parts of the system down. While ensuring the integrity of computations, this comes with the drawback of losing performance.

In the context of hard real-time systems, such unpredictable losses in performance are unacceptable, as they may lead to deadline misses which may yet compromise the integrity of the system. To safely execute hard real-time workloads on such systems, this article presents an online scheme for assigning speeds in such a way that (a) the system executes at low clock speed as often as possible, while (b) deadline violations are strictly ruled out.

The proposed scheme is compared with an offline scheme which has complete knowledge about arrival times and execution demands of the workload. The benchmarking shows that for a workload which is always very close to the modelled maximum, our approach performs on-par with the offline scheme. In case of a workload which diverges from the modelled maximum more often, the speed assignments produced by our scheme become more pessimistic, as to ensure that all deadlines are met.

1. Introduction

1.1. Motivation

As Moore's law continues to apply, the number of transistors that can be put on a single chip continues to increase. This has and will give rise to thermal problems and power shortages within processors, as it may not be possible to sufficiently cool or power such a high density of transistors at once. These effects are further amplified by increases in the clock frequencies applied to systems.

To counter the hazards introduced by these phenomena, modern processors are equipped with Dynamic Power and Thermal Management (DPTM) systems. The DPTM system monitors the temperature or power budget of processor components, e.g., at core-level, and takes different measures to ensure that set bounds hold.

Examples of such measures are to adapt the capacity of the cooling systems accordingly, or to simply power down unneeded parts of the processor infrastructure. The latter policy is sometimes referred to as *dark silicon* [5]. For multi-core processors that support Dynamic Voltage Frequency Scaling, the darkening of parts of the processor is unnecessary. Instead, the processor-proprietary clocks can be throttled, at the expense of a loss of computing performance. An example to this is illustrated in Fig. 1.

With hard real-time systems, workloads need to adhere to deadline constraints. Examples of such workloads range from SW-based control task up to distributed multi-media applications where audio and video-frames need to be played synchronously and in time. With unknown and possibly bursty arrival times of

* Corresponding author.
E-mail address: kai.lampka@it.uu.se (K. Lampka).

http://dx.doi.org/10.1016/j.jpdc.2016.03.002
0743-7315/

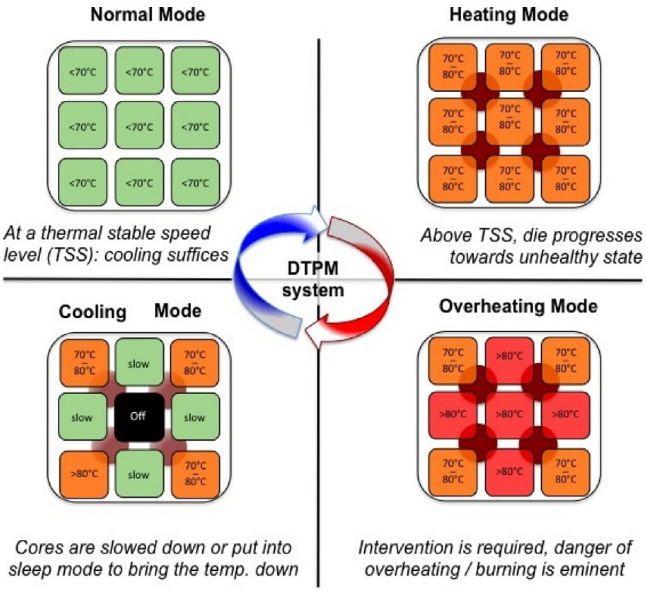

Fig. 1. A 9-core processor and the functioning of a DPTM system.

computation demands, online DPTM with real-time workloads is not straightforward. Particularly, as for real-time applications, the loss in performance which results from interventions of the DPTM system has the potential to lead to timing faults. As such, and depending on the safety level of the application, the timing faults have the potential to provoke faulty behaviour or even the complete loss of a system. Consider, for example, a real-time system tasked with controlling the pressure in a pipe. Should the system not respond in a timely manner to changes in the environment, the pressure may increase to such a level that the pipe and any other systems connected to it may be damaged beyond repair. To avoid this and in support of building reliable and safe real-time systems on top of modern multicore processors, this article proposes an adaptive scheme for explicit clock speed management of cores executing hard real-time workloads. To minimize the interventions of the DPTM system on the remaining cores, the scheme executes the real-time cores at the lowest possible voltage/frequency level, as to not produce more heat than absolutely necessary.

1.2. Contribution

This article presents an online DVFS scheme for explicitly controlling the clock frequency of cores executing a hard real-time workload. As main challenge, the scheme ensures that real-time workloads do not experience timing faults, i.e., misses of deadlines, while keeping the interventions of the DPTM system to a minimum.

In contrast to existing work, the presented scheme is aware of the history of activations of processes, but does neither require the explicit storage of activation times, nor is it restricted to highly regular patterns of workload occurrences.

The small (and constant) memory and computation footprint, makes this online scheme suitable for being implemented at the lower levels of the HW/SW stack, i.e., either in specialized HW or as part of an Real-time operating system. An implementation of the workload tracking scheme with a FPGA-platform has been presented in [10].

The scheme only requires two design time parameters. The first is an upper bounding function on the arrivals of computation requests (workloads) from an application w.r.t. the interval-of-time domain (release bound function). In addition to this, the scheme requires an application-uniform bound on the processing time for any of these requests (worst-case execution time).

The article presents an evaluation of the presented scheme using the gem5 hardware simulator. For investigating the capabilities of the scheme, we make use of predefined task activation traces, in which the arrival times of computation requests are either regular or irregular, with their individual execution times randomly chosen.

The evaluation with this synthetic workload shows that for workloads adhering close to the workload bounding function, the scheme performs on-par with an offline implementation with perfect knowledge about arrival times of processing requests and their individual durations, i.e., execution times. With the traces of workload arrivals diverging further from the (static) upper bound on the number of arrivals per interval of time, the scheme starts over-estimating required clock speeds. However, this pessimism comes at the benefit that deadline misses are strictly ruled out, even when the workload produces peaks in the computational demand.

1.3. Organization

The following two sections present the models and mechanisms upon which the presented work builds. In particular, Section 2 presents the system and workload models, whereafter Section 3 presents the workload monitoring model. Section 4 introduces the novel scheme, which is evaluated in Section 5. Lastly, the related work is presented in Section 6 and we conclude in Section 7.

2. Background: system model

The scheme relies on an abstract processor model and a (formal) real-time task model. The latter abstractly describes the workload of the processing unit. Once these models have been presented, we provide an example which highlights the problem that the novel scheme intends to solve.

2.1. Processor model

The processor is assumed to be a K-core DVFS capable processor, executing a hard real-time workload. It is assumed that each task within the system is mapped to a fixed core, and as such there is no task migration between cores at runtime. This is done to simplify the presentation. An extension to the case of migrating tasks away from hot spots is out of scope of this work.

As the clock frequency f of the processor varies, so does the execution time required to finish a job. However, estimating the execution time of a job under different core frequencies is not straightforward. The reason for this is that not all relevant system components are affected by the frequency change. At higher clock frequencies the processor is capable of executing more instructions per time unit, but the time that the processor has to wait for other subsystems, such as off-core memory, remains constant. Because of this, the execution time often increases less than proportionally to the decrease of the clock frequency of the core.

As a (simplified) example, consider that the fetching of data from memory takes 100 cycles at clock frequency f. In this case, the same fetch operation will only require 50 cycles at clock frequency $0.5 \times f$, as the memory operates at a constant speed, and each clock cycle now takes twice as long. Assume that, in addition to the cycles spent waiting for memory operations, the executing program requires 100 cycles of calculations to finish. Under these conditions, the program execution would take $100 + 100 = 200$ cycles at speed f, but only $100 + 50 = 150$ cycles at speed $0.5 \times f$. In other words, scaling the frequency down by $\frac{1}{2}$ only gives a performance slowdown of $\frac{\frac{150}{0.5 \times f}}{\frac{200}{f}} = 1.5$.

Please cite this article in press as: K. Lampka, et al., Keep it cool and in time: With runtime monitoring to thermal-aware execution speeds for deadline constrained systems, J. Parallel Distrib. Comput. (2016), http://dx.doi.org/10.1016/j.jpdc.2016.03.002