Hardware-enhanced run-time Management for Many-Core Processors

Daniel Gregorek, Alberto Garcia-Ortiz
Integrated Digital Systems Group
ITEM, University of Bremen, Germany
{gregorek,agarcia}@item.uni-bremen.de

Abstract— Many-core systems provide abundant computing power for parallel applications. Unfortunately, processor design reaches the so-called utilization-wall (dark silicon). If designers want to achieve a higher performance, their systems need to become more energy-efficient and specialized. This thesis focuses on the hardware optimization of the run-time manager to improve the overall system efficiency. We present a transaction-level simulation framework for the design-space exploration of hardware-enhanced run-time management. Based on the framework a concrete architecture for hardware-enhanced run-time management has been implemented. The evaluation identifies the crucial output parameters which are affected by the hardware-enhancement and quantifies the benefits compared to state-of-the-art software approaches.

I. INTRODUCTION

Due to rising power-density and a limited power budget the whole chip can no more be fully utilized and parts of the chip have to be turned off [1]. After hitting the utilization-wall, chip designers have to look for specialized implementations that achieve a higher energy efficiency. The optimization of the run-time management system is of particular interest for this work. Dynamic hardware faults and/or changes of the user settings make run-time management a recurring and necessary feature. The many-core domain exacerbates the management complexity and demands for scalable and high-responsive implementations. Low-latency and a small energy footprint can be realized by using a dedicated hardware implementation. The integration of dedicated hardware for run-time management is therefore predicted to become mainstream for many-core systems [10].

II. RELATED WORK

One of the first works which explicitly addresses energy efficiency by using a hardware scheduler has been published by Gupta et al. [6]. They use a centralized hardware scheduler for priority-based scheduling and achieve 2-3 orders of magnitude higher energy efficiency compared to a software implementation. In contrast, Isonet [7] tries to overcome the limitations of centralized architectures and applies a fully distributed infrastructure of load balancing nodes. Due to a high communication overhead their approach does only address for locally optimal load balancing. The STHORM architecture uses clusters of processing elements and is scalable up to 64 PEs. The presented run-time management for STHORM [8] is restricted to intra-cluster hardware synchronization.

III. CONTRIBUTION

The thesis follows the hypothesis that having a dedicated infrastructure for run-time management improves the efficiency of the computation. Fig. 1 shows the conceptual comparison of a homogeneous many-core versus a dedicated infrastructure. The infrastructure allocates dedicated computing nodes for run-time management and provides a dedicated communication network. The goal of the thesis is to identify the profiting output parameters, to quantify the benefits and as well the limits of the approach. Multimedia application are of special interest for this work since they usually contain a high degree of parallelism and have considerable timing requirements. We present a tailored simulation framework for design-space exploration (AGAMID) and a concrete architecture (DRACON) for hardware-enhanced run-time management. Their particular contributions are outlined in the following subsections.

AGAMID Framework

AGAMID is a transaction-level design space exploration framework [2] [5]. The framework allows the comparison of many-core systems with an emphasis on the run-time management architecture. Applications are abstracted as transaction-level traces while the run-time management is simulated at a higher level of detail. The framework uses SystemC / TLM 2.0 for system modeling and a generic OS template. The generic OS is composed of three major parts: The OS-Master performs decision making (e.g. task mapping). The OS-Slave assists (e.g. context-switching) and provides the interface to the user-space. The Resource Management Blocks contain the management data (e.g. PE states, task ready queues) and can be protected by locks to guarantee data consistency. Particular configurations for the generic OS are shown in Fig. 2.

![Homogeneous vs Dedicated Infrastructure](image)

Fig. 1: Conceptual comparison of (a) a homogeneous system versus (b) a dedicated run-time management infrastructure. The number of computing nodes is identical for both cases.

![Generic OS Design Points](image)

Fig. 2: Design point examples for the generic OS. The OS-Masters can either run at a common PE or at a dedicated node. Legend: Resource Man. Block (circle), OS-Master (diamond), PE (square).
The elementary part of the OS is defined by a common OS-Master class which is encapsulated by a timing model. Each architecture-dependent OS-Master instantiates the timed model and additionally must implement an abstract class interface. The design pattern (see Fig. 3) allows to compare different communication models and different timings while using an identical set of elementary functions.

**B. DRACON Architecture**

DRACON [3] is a many-core system having a hardware-enhanced (dedicated) infrastructure for run-time task management. The architecture outlined in Fig. 4 provides dedicated computing nodes and dedicated interconnects. The run-time management performs global-view task synchronization by means of a distributed data structure. The nodes use message passing for communication. Each global node (AGM) constitutes an OS-Master, has a private memory and cooperates with the other AGMs. The local nodes (LMC) constitute the OS-Slaves and connect the PEs to the dedicated infrastructure.

**IV. Evaluation**

The evaluation focuses on the problem of non-preemptive run-time scheduling for homogeneous many-core processors. We consider the DRACON architecture and for reference a symmetric and an asymmetric software OS [10]. The task graph is exposed to the run-time manager and used for priority-based scheduling. Our results indicate significant advantages for the DRACON architecture considering application speedup and deviation of application response time (jitter). Fig. 5 shows representative results when simultaneously running a set of workloads consisting of independent tasks and real-world applications [9]. In general, a dedicated infrastructure cannot provide a significantly higher throughput since sequential task execution time is still the same. But in contrast, it can provide a higher responsiveness and potentially better scheduling quality. We achieve a 40% higher speedup and 16% less jitter compared to a competitive asymmetric software approach. The efficiency has been improved while using almost the same amount of hardware. We argue that timing requirements can be met while requiring a smaller energy footprint. The dedicated hardware has even more optimization potential due to the

definite position of the management nodes. Designers can trade-off the $O(\log n)$ time complexity for priority sorting into additional hardware but gaining constant time $O(1)$ scheduling overhead. Related experiments using a single best-effort hardware scheduler revealed sustainable scalability for more than 64 PEs [4].

**V. Conclusion**

In the dark-silicon regime chip area becomes abundant but still must be carefully spent. This thesis focusses on the specialization of the run-time management for many-cores by using a dedicated infrastructure. The hardware-enhanced run-time management uses optimized hardware for computation and communication. Our evaluation substantiates a higher system efficiency compared to conventional run-time management. We predict a wide applicability in the domain of multimedia applications where a high degree of parallelism and narrow timing requirements exist.

**References**