Heterothread: Hybrid Thread Level Parallelism on Heterogeneous Multicore Architectures

Chao Wang
School of Computer Science
Univ. of Sci. & Tech. of China
chao.wang@ieee.org

Xi Li
School of Computer Science
Univ. of Sci. & Tech. of China
llxx@ustc.edu.cn

Xuehai Zhou
School of Computer Science
Univ. of Sci. & Tech. of China
xhzhou@ustc.edu.cn

ABSTRACT

In this paper, we introduce middleware architecture to support hybrid thread level parallelism on heterogeneous multicore architectures, called Heterothread. Heterothread constructs a hierarchical level model for user programming and parallel task execution dataflow. Heterothread can provide unified programming interfaces which allow applications remain the same when the physical hardware (CPU, GPU, DSP and FPGA) is reconfigured or migrated. Consequently it can meet the real-time demands more efficiently than most state-of-the-art operating systems. A prototype has been built on FPGA to demonstrate Heterothread can achieve significant speedups against Linux kernels.

Categories and Subject Descriptors

D.4.1 [**Operating Systems**]: *Multiprocessing/multiprogramming/*

General Terms

Performance, Design

Keywords

Architectural support; middleware; dynamic reconfiguration;

1. INTRODUCTION

Heterogeneous architecture is dominating [1]. However, it still poses a significant challenge to build a moderate operating system or middleware to manipulate threads running on different architectures efficiently (e.g. CPU, GPU, DSP, and FPGA). Current ongoing projects like fos [2] and barrelfish [3] is still pursuing a fine approach. With respect to the real time demand on heterogeneous architectures, how to obtain sufficient timing predictability is still posing significant challenge. In order to tackle this problem, the major contribution of this paper is to propose a middleware hierarchical model to support hybrid MPSoC reconfigurations. Heterothread provides acceleration engines to diverse applications by substituting IP cores dynamically. After hardware is reconfigured, Heterothread will reorganize the task partitioning, mapping and scheduling strategies, which keep the APIs unchanged to users. The entire architectural model consists of both software and hardware thread level controller modules.

First, to provide an execution environment for tasks, Heterothread employs run-time libraries. These user libraries along with other kernel libraries provide well-structured application programming interfaces (APIs). APIs show a high-level abstract view of the internal implementations. When the APIs are defined, the user interfaces will be kept as persistent units, even the hardware (DSP, FPGA and GPU kernel) replacement is invisible to programmers.

Task partitioning and scheduling methods play a vital role in Heterothread. Before tasks are offloaded to acceleration engines, Heterothread must decide which task runs on which core, and also when the task is issued. For the sake of automatic thread level parallelism, we use out-of-order task execution engine to solve the inter-task data dependencies [4].

Finally, the communication interfaces layer is in charge of data transmission between Heterothread scheduling kernel and the diverse processing kernels. Generally there are three kinds of primitives: the unified software interface (USI), unified hardware interface (UHI), and unified reconfiguration interface (URI): USI primitive is employed when the information in transferred among microprocessors, including a series of function in libraries. UHI primitive is introduced to model the communication between microprocessor and hardware RTL implemented IP cores or DSP engines. Furthermore, an interrupt controller is employed to detect interrupt requests for synchronization. Finally, URI primitive is invoked only for IP reconfiguration to switch the pre-downloaded partial bitstream at run-time.

We have constructed our prototype OS using with Xilinx FPGA board. As a working-in-progress paper, we are still running large scale Benchmarks on both a Minicore system of Heterothread and Linux 2.4.18 operating system kernel. At the time of writing this paper, we have evaluated the Anubis, 10-step FIR, and 20-step FIR applications on the Minicore OS, which achieve the speedup from 4.5x to 170x, respectively. Preliminary results demonstrate Heterothread provides new insights to the real time operating system research paradigms on heterogeneous multicore architectures.

2. ACKNOWLEDGEMENT

This work was supported by the NSFC grants (No. 61272131, No. 61202053), Jiangsu Provincial NSF (No. SBK201240198).

2. REFERENCES

[1] S. Singh, Computing without processors [J]. *Communications of ACM,* 2011. 54(8): p. 46-54. DOI= http://doi.acm.org/10.1145/1978542.1978558.

[2] D. Wentzlaff,A. Agarwal, Factored operating systems (fos): the case for a scalable operating system for multicores[J]. *ACM SIGOPS Operating Systems Review,* 2009. 43(2): p. 76-85. DOI= http://doi.acm.org/10.1145/1531793.1531805

[3] J. C. Mogul, A. Baumann, T. Roscoe, L. Soares. Mind the gap: reconnecting architecture and OS research. in Proceedings of the *13th USENIX conference on Hot topics in operating systems*.2011.

[4] G. Gupta,G. S. Sohi. Dataflow execution of sequential imperative programs on multicore architectures. *in Micro 44.* 2011.59-70. DOI= http://doi.acm.org/10.1145/2155620.2155628