Power-Aware Performance Increase via Core/Uncore Reinforcement Control

Da-Cheng Juan

Diana Marculescu

Microprocessor design trends have shifted to chip-multiprocessors (CMPs) from the classic monolithic, single core systems. To reduce the overhead of inter-core communications, the network-on-chip (NoC) paradigm has been proposed as a promising solution for massively-integrated CMPs. However, the enhanced performance and capabilities of such platforms are usually constrained by the on-chip power consumption. While dynamic power management has been extensively studied for multi-core systems while considering only core or uncore resources, the cooperative power management of core and uncore has remained a critical issue not sufficiently explored. This cooperative power management by using dynamic voltage and frequency scaling (DVFS) introduces additional challenges that require maintaining appropriate performance levels for parallel applications executing on the system. As a result, how to reliably control cores and uncores in synergy via DVFS while maintaining power constraints remains an important problem that needs to be addressed in the context of advanced multi-core systems.

In this work, we apply reinforcement learning (RL) to enable an adaptive, model-free control of advanced CMPs. Furthermore, compared to conventional DVFS schemes that address power reduction under performance constraints, we propose a "*reverse*" DVFS: maximize performance while ensuring that power stays within prescribed power constraints. Figure 1 illustrates the analysis of the program execution time among core-only (C-D), uncore-only (U-D) and our proposed RL. All results are normalized to the baseline, *i.e.*, the case without any DVFS. On average, the proposed control achieves 10.9% better performance. Figure 2 provides the comparisons of energy-delay products. It can be seen that the proposed control achieves the lowest energy-delay product − 6% reduction on average compared to the baseline.

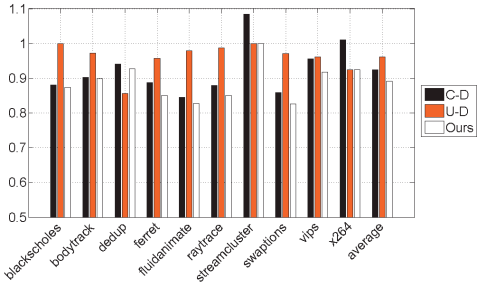

Fig. 1: Normalized execution time

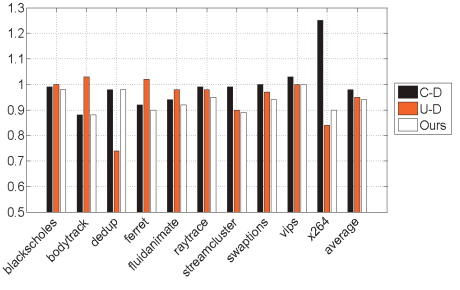

Fig. 2: Normalized energy-delay product