A collaborative approach to leverage overlapping-ISA heterogeneous multicore architectures

Jiaxun Yang

March 12, 2024

A heterogeneous processor incorporates cores with varying levels of performance and capabilities, enabling the optimization of specific cores for particular tasks. This configuration facilitates the execution of a diverse array of applications with greater efficiency compared to homogeneous processors. Recent years have seen a rise in the popularity of single-Instruction Set Architecture (ISA) heterogeneous architectures due to their transparent operation with applications, while still preserving many advantages of a heterogeneous multicore architecture. Nonetheless, this approach restricts core capabilities' heterogeneity by assuming that every instruction should be executable on all cores within a system. Such a limitation leads to the unnecessary replication of seldom-utilized function units in cores, thereby hindering the potential for high-level specialization of cores. An alternative design, the overlapping-ISA heterogeneous multicore architecture, permits all system cores to share a common baseline ISA subset alongside distinct ISA extensions. Prior studies on software support for this architectural design, based on fault-and-migrate or emulation strategies, are all exposing a uniform ISA to application and incur penalty to migrate or emulate when the application is running on suboptimal core, thus inefficient for real world applications. In our view, these penalties stem from the operating system's lack of control over the ISA features an application can utilize at runtime. We propose a novel software architecture that encompasses applications, the toolchain, and the operating system to address this issue.

In this proposed architecture, applications would identify functions that could benefit from specific ISA extensions, providing two versions for each such function: one optimized with the ISA extension and another as a fallback variant using the baseline ISA. Additionally, applications would furnish details regarding the runtime performance and speed-up of these two implementations.

The toolchain is expected to compile both function variants, generating two Global Offset Tables to organize each variant and insert NOP trampolines at the entry and exit points of functions.

The operating system would then determine the optimal placement of an application based on toolchain metadata and runtime profiling information. It could switch the register holding the global offset table to manage the application's ISA usage and insert trampoline functions to assist in profiling or identifying precise migration points. Moreover, the operating system should offer lightweight methods to support Just-In-Time (JIT) applications in selecting the most suitable ISA feature set.

We believe our approach can help software leverage overlapping-ISA heterogeneous multicore architectures at its best. This concept is in its nascent stages, and we welcome feedback from other system researchers to refine our proposal.