PROJECT DESCRIPTION

1 Introduction

Heterogeneous architecture is rapidly adopted by many hardware designers for better efficiency. However, programming and managing different hardware resources gradually becomes one of the headaches for programmers and researchers. On mobile SoCs(System-On-Chips), a wimpy but power-efficient, microcontroller-like cores are added to timely collect data from sensors. Compared with main CPU, the microcontroller has a much simpler microarchitecture and implements a trimmed-down version ISA, which provides optimization opportunities for architects to design chips more efficiently. For the server/data center, the hardware environment becomes more complicated. Various application specific accelerators are used, e.g., GPU, NPU, and FPGAs. They provides different properties on programmability, data access method, execution efficiency for specific workloads. Moreover, they are connected to the CPU via different interface, e.g., PCIE, Ethernet, and Infinite Band, which imposes disparate programming models to programmers.

To bridge the heterogeneity, we propose to establish a global view of the entire system. Each different types of hardware are labeled with its own capabilities(e.g., General purpose computing, AI accelerating, etc.) and programming models(e.g., asynchronous command, shared memory, etc.) Programmers only needs to provide application’s property, and write applications with generic system APIs provided by us. The compiler will compile the source code to binaries which can be executable across different hardware. Underlying the application, the runtime system will dynamically schedule user tasks to appropriate hardware. For legacy code which cannot be recompiled to different platforms, a dynamic binary translation engine will dynamically translate and execute the code for another hardware.
2 Intellectual Merits

Heterogeneous architecture has been discussed for years, and advocate a lot of fancy usages. Exiting work exploited offloading specific operating system phase to accelerators. For example, GPUs Silberstein et al. (2013) implements file systems on GPU and provide a set of POSIX like API to use. Solros Min et al. (2018) use Xeon Phi to accelerate high concurrent interrupt handling in data centers. However, there is no work that systematically study the tradeoff between different accelerators/processors.

This project will ease the effort to programming heterogeneous hardware. More specifically, programmers only needs to know little about subtleties in hardware. The heterogeneity is abstracted by our generic API. In addition, it will allow us to rethink the boundary between software and hardware, and propose guidelines for future software and hardware environment.

3 Research Plan

- Systematically study all processors/accelerators, especially how to program them, how to move the data, and different capability
- Come up with the API designs which manages the underlying heterogeneous hardware
- Design and implement dynamic binary translator to empower legacy software running on different hardware
- Evaluate the performance of the entire system, and compare with existing solution
- Based on the experiment, propose advice for designers on how to design software and hardware more heterogeneous platform friendly.

References