Software compute virtualization - priority, performance, and programmability

AMOGH AKSHINTALA

Abstract

The proposed dissertation will explore developer effort and compatibility in software virtualization of CPU ISAs, as well as, questions of performance and programmability in software virtualization of specialized compute devices (e.g., GPUs, TPUs, etc.) that are programmed through an API. Although binary translation is a well-established software ISA virtualization technique, given the size and complexity of today's dominant ISAs, developers are routinely forced to adopt ad-hoc techniques to prioritize development effort. The proposed dissertation will present a principled approach to determine priority among different parts of the ISA. Specialized compute accelerators, such as GPUs and TPUs, are usually controlled through a user-space API. The proposed dissertation will explore the implications of ISA virtualization for these devices, the performance and programmability implications of a novel hypervisor-mediated API-remoting virtualization technique, and present a novel nomenclature for cleanly understanding the design space for virtualizing compute accelerators.

1 Introduction

Virtualization, defined broadly, is a means to an end: fair, isolated, and efficient sharing of resources among mutually distrustful entities. Virtualization is vital to achieving high utilization of available physical resources in large computing installations such as clusters and data centers. Virtualization is the main technical force behind cloud computing.

Virtualizing a compute resource, such as a CPU, typically involves mediating access to said resource either by exposing an interface that is identical to that of the encapsulated resource (full-virtualization), or by exposing an alternative abstract interface, accesses to which are in-turn synthesized to the native interface (para-virtualization). The exposed interface is *virtual*, in that it is not directly exposed by the physical underlying hardware, and instead is entirely under the control of supervisory virtualization software, the *hypervisor* (also known as the *Virtual Machine Monitor*). The hypervisor is responsible for ensuring

Despite nearly four decades of attention from both the academic community and industry, efficient virtualization of compute resources remains poorly understood. As new compute devices emerge (e.g., GPGPUs, TPUs, IPUs, IO accelerators), virtualization developers find themselves once again balancing the essential characteristics of a good virtualization scheme—compatibility, interposition, sharing, isolation—with the need to preserve the raw performance these emerging compute resources provide.

Concretely, the proposed dissertation will evaluate the following hypotheses:

- Priority among instructions in an ISA, in the context of binary translation, can be automatically inferred from user preferences.
- ISA virtualization is untenable for performant virtualization of compute accelerators.
- Hypervisor-mediated API-remoting is a low-overhead virtualization scheme for API-controlled compute accelerators.
- The characteristics of a virtualization technique can be succinctly described by a scheme that explicitly captures the *Interface* interposed, the *Endpoints* interposed on, the *Mechanism* of interposition, the *Transport* used to connect the interposed endpoints, and the mechanism used to *Synthesize* the interposed operation on the host.

2 Description of Chapters

The Instruction Set Architecture is typically the interface of choice when virtualizing CPUs. ISA virtualization has a long and storied history (IBM 360, Popek and Goldberg, Xen, VMware, etc.). While the reigning ISAs of the day (x86-64, ARM, etc.) all have special virtualization extensions for performance optimization (Intel VT, AMD-V), binary translation based virtualization schemes are still a necessity (a recent example being the HVX nested hypervisor [4] from 2014). However, when setting out to build a new binary translation based virtualization system, developers need a way to determine implementation and optimization priority, especially for a gargantuan ISA like x86-64. We will present a methodology (and the resulting data) that leverages user preference to systematically answer such questions in Chapter 1. This is completed work [1].

Virtualizing a Graphics Processing Unit (GPU) for the purposes of graphics rendering is a well studied problem, with existing commercial solutions (VMware's SVGA [3]). Over the last decade, GPUs have been re-purposed for parallel general purpose compute (commonly known as GPGPU). Chapter 2 will present our findings from attempting to extend the SVGA model of GPU virtualization to cover GPGPU virtualization as well. We find that the tight coupling between ISA virtualization and device virtualization in SVGA leads to poor performance for GPGPU compute. We propose a new virtualization scheme, Trillium, that doesn't rely on ISA virtualization and show that Trillium outperforms all other traditional virtualization schemes while retaining hypervisor interposition. Material presented in Chapter 2 will be drawn from a published paper [2].

The slowing down of Moore's law and the increasing importance of compute-heavy workloads such as Deep Neural Networks and Graph analytics have led to the proliferation of specialized compute units. These specialized processors are typically exposed to developers via a user-space API, which is typically implemented by a combination of proprietary and opaque software and hardware interfaces. Chapters 3, 4 and 5 will explore the performance implications of virtualizing the user-space API for specialized compute accelerators. Chapter 3 will present an overview of AvA, a framework that enables automated virtualization of accelerator APIs. Chapter 4 will focus on the performance implications of API-remoting based virtualization of a single specialized accelerator. Chapter 5 will explore performance issues that arise when an application uses multiple

API-remoted virtual accelerators in a pipelined fashion. Chapters 3 and 4 will draw on material that appeared in a HotOS workshop paper [5] and a full paper that is currently under submission. Chapter 5 is proposed work.

Virtualization schemes are traditionally taxonomized according to the core techniques employed (e.g. emulation, full- or para-virtualization, API remoting, etc.), and evaluated in a property trade-off space comprising performance, compatibility, interposition, and isolation. We argue that both the de facto taxonomy and the property trade-off space are illustrative but not informative for GPGPU virtualization: there is a large body of research that has had little influence on practice. We suggest an alternative framework called `IEMTS` that teases apart design axes that are implicitly and unnecessarily intertwined in much of the literature. By focusing on the **I**nterface interposed, the interposition **E**ndpoints, the **M**echanism of interposition, the **T**ransport used to move the interposed operations between the guest and the host, and the mechanism used to **S**ynthesize the interposed interface, `IEMTS` enables a clearer understanding of trade-offs in prior designs and provides a model for comparison of alternative designs. `IEMTS` will be presented in Chapter 6, along with analysis of traditional virtualization techniques in the context of GPGPUs.

References

[1] Amogh Akshintala, Bhushan Jain, Chia-Che Tsai, Michael Ferdman, and Donald E Porter. x86-64 instruction usage among c/c++ applications. In *Proceedings of the 12th ACM International Conference on Systems and Storage*, pages 68–79. ACM, 2019.

[2] Amogh Akshintala, Hangchen Yu, Arthur Peters, and Christopher J Rossbach. Trillium: The code is the ir. In *The Second Special Session on Virtualization in High Performance Computing and Simulation (VIRT 2019), Dublin, Ireland*, 2019.

[3] Micah Dowty and Jeremy Sugerman. Gpu virtualization on vmware's hosted i/o architecture. *ACM SIGOPS Operating Systems Review*, 43(3):73–82, 2009.

[4] Alex Fishman, Mike Rapoport, Evgeny Budilovsky, and Izik Eidus. HVX: Virtualizing the cloud. In *Presented as part of the 5th USENIX Workshop on Hot Topics in Cloud Computing*, San Jose, CA, 2013. USENIX.

[5] Hangchen Yu, Arthur M Peters, Amogh Akshintala, and Christopher J Rossbach. Automatic virtualization of accelerators. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 58–65. ACM, 2019.