Journal of Systems Architecture 59 (2013) 144–154

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Formal virtualization requirements for the ARM architecture

Niels Penneman [a,b,*,1], Danielius Kudinskas [c], Alasdair Rawsthorne [c], Bjorn De Sutter [a], Koen De Bosschere [a]

[a] Computer Systems Lab, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium
[b] Vrije Universiteit Brussel (VUB), Dept. of Electronics and Informatics (ETRO), Pleinlaan 2, B-1050 Brussels, Belgium
[c] School of Computer Science, The University of Manchester, Oxford Road, Manchester M13 9PL, UK

ARTICLE INFO

Article history:
Received 29 March 2012
Received in revised form 22 November 2012
Accepted 11 February 2013
Available online 27 February 2013

Keywords:
Hypervisor
Instruction set architecture
Binary translation
Virtualization
Virtual machine monitor

ABSTRACT

We present an analysis of the virtualizability of the ARMv7-A architecture carried out in the context of the seminal paper published by Popek and Goldberg 38 years ago. Because their definitions are dated, we first extend their machine model to modern architectures with paged virtual memory, IO and interrupts. We then use our new model to show that ARMv7-A is not classically virtualizable.

Insights such as binary translation enable efficient virtualization beyond the original criteria. Companies are also making their architectures virtualizable through extensions. We analyse both approaches for ARM and conclude that both have their use in future systems.

1. Introduction

The term virtualization is used in many different contexts, ranging from storage or network technologies to execution environments and even to virtual realities. This paper refers to system virtualization—technology which allows multiple operating systems to be executed on the same physical machine simultaneously. It achieves this by decoupling the underlying hardware from the operating systems, which are executed deprivileged as shown in Fig. 1. These operating systems are now called guest operating systems. In general, a guest or virtual machine (VM) refers to all software that is executed deprivileged and in an isolated environment under the control of a virtual machine monitor (VMM).

In 1974, Popek and Goldberg [44] wrote a classic paper on ‘‘Formal requirements for virtualizable third generation architectures’’. It defines strict system virtualization criteria for computer architectures, and proves that if they are met, an ‘‘efficient’’ VMM can be constructed for that architecture. Since its publication, the paper has been used as a solid reference point for designing hardware platforms capable of supporting an efficient VMM and it has become the groundwork of virtualization technology.

The criteria defined by Popek and Goldberg are now known as the conditions for *classic virtualizability*. Architectures that meet these criteria are particularly suited for full virtualization. *Full virtualization* (also known as pure or faithful virtualization) is a technique in which the VMM presents a virtual platform that is an exact replica of the real hardware to each of its guests. Hence, applications and operating systems do not require any modification to support virtualization.

Popek and Goldberg based their model on computers available at the time, such as DEC PDP-10 and IBM 360/67. Due to advances in microprocessor architecture, their model no longer fits current architectures. Furthermore, new approaches in the construction of efficient VMMs that do not fit Popek and Goldberg's model and criteria [24], have enabled virtualization of many more contemporary architectures.

Virtualization in the server and desktop world has already matured, with both software and hardware solutions available for several years [1,14,16,45,48,50,53]. However, virtualization on embedded systems has only been explored for the past 7 years, and is an area of ongoing research [22,29]. Solutions for data centres and desktop computing cannot be readily applied to embedded systems, because of differences in requirements, use cases, and computer architecture. We provide a new perspective on the Popek and Goldberg virtualizability requirements. We develop a model that focuses on modern microprocessors and use it to analyse the ARM architecture. We illustrate that our model can also help in the construction of VMMs based on dynamic binary translation (DBT) by using our analysis to show the pitfalls for constructing a DBT VMM on ARM.

* Corresponding author at: Computer Systems Lab, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium. Tel.: +32 (0)9 264 9857; fax: +32 (0)9 264 3594.

E-mail addresses: Niels.Penneman@elis.UGent.be, niels@penneman.org (N. Penneman), kudinsd6@cs.man.ac.uk (D. Kudinskas), Alasdair.Rawsthorne@manchester. ac.uk (A. Rawsthorne), Bjorn.DeSutter@elis.UGent.be (B. De Sutter), Koen. DeBosschere@elis.UGent.be (K. De Bosschere).

[1] Supported by the agency for Innovation by Science and Technology (IWT).

1383-7621/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2013.02.003

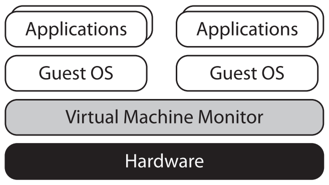

Fig. 1. Overview of a virtualized system.

ARM is by far the leading architecture in the embedded and mobile market [47], and recently multiple software virtualization solutions have been developed for it, such as Codezero, OKL, Red Bend VLX, VIRTUS, VMware MVP and Xen-ARM [11,12,15,28,31,34,37]. Due to architectural problems none of the currently deployed solutions offers full virtualization [51].

ARM has already published a preliminary specification of extensions to its architecture, which will facilitate full virtualization [8]. The appearance of hardware platforms that implement these extensions is imminent. It is therefore important to understand the problems faced by virtualization technology on ARM today, and evaluate the possible solutions against the formal requirements derived by Popek and Goldberg [44] nearly 40 years ago. Popek and Goldberg also proposed techniques to virtualize architectures which did not meet their criteria. VMMs based on DBT, which do not require changes to the architecture, can be regarded as an evolution of their techniques. Their analysis therefore remains useful for the construction of such VMMs and for understanding the advantages and disadvantages of both the HW supported and DBT supported approaches. On the one hand, hardware extensions enable simple VMM implementations but are tied to specific architectures and platforms. On the other hand, DBT is notorious for complicating VMM implementations, but is known to be versatile and usable in situations where hardware extensions would be infeasible or impractical.

This paper analyses the application profile of the current ARM architecture, ARMv7-A, because it serves as the base for the upcoming virtualization extensions [8]. Our model excludes micro-controllers for deeply embedded systems. Those architectures often lack advanced memory management features and hence they are incapable of running full-blown operating systems.

Our contributions include:

- an update to the model of Popek and Goldberg with paged virtual memory, IO, and interrupts, and the application of such model to analyse the ARMv7-A architecture both without and with the virtualization extensions;
- an example of how an analysis according to our model helps in the construction of a DBT VMM;
- a discussion on the trade-offs between the use of architectural extensions for virtualization and DBT, in which we argue that both have their use in future systems.

The rest of this paper is organized as follows: Section 2 discusses our motivation and gives a quick overview of the model introduced by Popek and Goldberg. In Section 3 we extend this model to include paged virtual memory, IO, and interrupts. We use the updated model to show that the ARM architecture is not classically virtualizable in Section 4. In Section 5 we analyse ARM's upcoming hardware support and discuss how ARM can be fully virtualized using DBT. We show how our analysis from Section 4 helps in the construction of a DBT VMM. We then compare the use of hardware extensions with DBT-based approaches on a theoretical level.

2. Background and motivation

2.1. Existing virtualization solutions

Paravirtualization is the only virtualization technique deployed in today's ARM-based embedded systems. In paravirtualization, a VMM presents a custom interface to its VMs which is similar, but not identical to the underlying hardware [54].

Despite the popularity of paravirtualization, none of the efforts to standardize the required VMM interfaces has gained sufficient momentum to spread to more than a few operating systems or VMMs [3,38,46]. As a consequence, the majority of contemporary embedded operating systems do not support such interfaces out of the box. Instead, VMM vendors and third parties must provide source code patch sets to support specific VMM interfaces for specific operating system versions. This situation comes with four major drawbacks:

1. Developing, maintaining, and testing patch sets for each and every combination of a specific operating system version and a VMM interface is an expensive process. Although semantic patches may offer a solution to simplify patch management [10], the effort required for testing remains.
2. Patched operating systems may exhibit unexpected behaviour because their reliability is not guaranteed and patches may introduce new security issues.
3. Licenses may prevent or restrict modifications to operating systems source code, and often impose rules on the distribution of patch sets or patched code.
4. Previously certified software stacks will need to be re-certified after patching. The recertification process is expensive and always specific to a particular VMM interface, thereby stimulating vendor lock-in.

This analysis is shared by major players in industry including ARM, Nokia and STMicroelectronics [19].

2.2. Classic virtualizability

The drawbacks of paravirtualization can be avoided by using full virtualization. With full virtualization, guest operating systems run unmodified on top of a VMM. This also enables virtualization of a priori unknown software. Popek and Goldberg [44] formally derived sufficient (but not necessary) criteria that determine whether an architecture is suitable for full virtualization. In this section, we briefly explain their model and results.

The machine model used by Popek and Goldberg is deliberately simplified. It includes a processor and a linearly addressable memory, but does not consider interaction with IO devices and interrupts.

The processor operates in either supervisor mode or user mode. Supervisor mode is a privileged mode, meant for the operating system, while user mode is unprivileged. The processor also features a program counter and a relocation-bounds register, used for relative memory addressing. An instruction set architecture (ISA) for such a processor can move, look up or process data and alter program control flow. Based on this description, they defined the concept of machine state.

Definition 1. The machine state S is defined by the contents of the memory E, the processor mode M, the program counter P, and the relocation-bounds register R:

$$S \equiv \langle E, M, P, R \rangle.$$