Microprocessors and Microsystems 48 (2017) 21–35

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Executing secured virtual machines within a manycore architecture

Clément Dévigne*, Jean-Baptiste Bréjon, Quentin L. Meunier, Franck Wajsbürt

Sorbonne Universités UPMC Univ Paris 06, CNRS, LIP6 UMR 7606 4 place Jussieu 75005 Paris, France

ARTICLE INFO

Article history:
Received 22 February 2016
Revised 31 May 2016
Accepted 16 September 2016
Available online 22 September 2016

Keywords:
Manycore architecture
Virtualization
Hypervisor
Physical isolation
Security

ABSTRACT

Manycore processors are a way to face the always growing demand in digital data processing. However, by putting closer distinct and possibly private data, they open up new security breaches. Splitting the architecture into several partitions managed by a hypervisor is a way to enforce isolation between the running virtual machines. Thanks to their high number of cores, these architectures can mitigate the impact of dedicating cores both to the virtual machines and the hypervisor, while allowing an efficient execution of the virtualized operating systems. We present such an architecture allowing the execution of fully virtualized multicore operating systems benefiting of hardware cache coherence. The physical isolation is made by the means of address space via the introduction of a light hardware module similar to a memory-management unit at the network-on-chip entrance, but without the drawback of relying on a page table. We designed a cycle-accurate virtual prototype of the architecture, controlled by a light blind hypervisor with minimum rights, only able to start and stop virtual machines. Experiments made on our virtual prototype shows that our solution has a low time overhead – typically 3% on average.

1. Introduction

The computer world is facing an explosion in the amount of digital data. This data can come from social networks as well as new uses of mobile computing as communicating objects. The information contained in these data is valuable either for commercial purpose, or for economic, environmental or health-related purposes as well. Clearly, the issue of security for accessing such information is critical, as is the protection of personal data.

By their nature, manycore processors are able to run multiple applications in parallel and thus allow to process a large data stream. However, they must be able to guarantee the security properties for such applications, namely integrity and confidentiality, in particular if the data processed are from different clients.

We propose a mixed hardware/software solution which can be used as a cloud platform, allowing to execute numerous independent applications, while providing an isolated execution environment as a response to the confidentiality and integrity problematics. The choice of a manycore architecture seems particularly suited to this goal, since the high number of cores allows to respond to all kinds of computational demands. However, existing manycore architectures do not provide security extensions, so they cannot propose an efficient solution. The baseline manycore architecture used in this work is the TSAR [1] architecture, which is a manycore architecture with hardware cache coherence and virtual memory support, but no particular mechanism for addressing security issues. The security-enhanced version of this architecture will be called the Tsunamy architecture.

The proposed architecture can typically be used by cloud platform servers, to which several clients can connect and execute their program for processing data. In such a context, two clients' applications need to be isolated with more than just processes, because a bug exploit in the operating system could lead to data leakage and corruption between the two applications. In our proposed solution, we make thus the assumption that each client runs an entire operating system, using the well-known technique of operating system virtualization.

An ideal framework for cloud platforms would meet the following goals and constraints: little or no hardware extension, no performance penalty compared to an operating system running alone on the platform, support for general purpose (e.g. Unix-like) multicore operating systems, hardware cache coherence support, unmodified (bare-metal) execution of guest operating systems and of course security concerns: virtual machine isolation and small Trusted Computing Base (TCB). We will discuss how our solution answers these constraints along the article.

We believe that this paper makes three contributions:

- We provide the design of a secure manycore architecture allowing the execution of physically isolated virtual machines of variable size, and supporting cache coherency.

* Corresponding author.
E-mail addresses: clement.devigne@lip6.fr (C. Dévigne), jean-baptiste.brejon@lip6.fr (J.-B. Bréjon), quentin.meunier@lip6.fr (Q.L. Meunier), franck.wajsburt@lip6.fr (F. Wajsbürt).

http://dx.doi.org/10.1016/j.micpro.2016.09.008
0141-9331/

- We discuss the design of a blind hypervisor adapted to this architecture, and requiring little hardware extensions
- We demonstrate the feasibility of our approach by the implementation and evaluation of a cycle-accurate virtual prototype, and we show that the virtualization overhead remains low.

The rest of the document is organized as follows: Section 2 gives more details about the background of hypervisors and manycore architectures, and discusses related works; Section 3 presents our design choices based on the security properties we target; Section 4 contains a description of the existing components upon which this work is based, and the proposed modifications; Section 5 presents our hypervisor and its basic functionalities, comprising the virtual machine boot and shutdown; Section 6 presents our experimental procedures and the obtained simulations results; finally, Section 7 concludes and summarizes the remaining work.

2. Background and related works

2.1. Manycore architectures

Manycore architectures are architectures containing from a few tens to thousands of cores integrated on the same chip. Such architectures use simple cores in order to maximize the performance per Watt ratio [2]. They are typically clustered, the clusters being connected together *via* a Network-on-Chip [3,4]. Each cluster (Fig. 1) usually contains one or several cores and a few peripherals, connected over a fast local interconnect. Apart from the performance per Watt, the biggest advantage of manycore architecture is their inherent redundancy, which allows both power dissipation reduction by dynamically turning off idle cores, and fault-tolerance through deactivation of faulty cores while using the remaining functional ones.

Manycore architectures vary in the way the cores can communicate, either inside a cluster or between two different clusters. Some architectures use specialized interfaces (e.g [5]) or dedicated hardware buffers to make two cores communicate, while some others support shared memory. Among the shared memory architectures, some support hardware coherence [6,7] while others do not [8].

We believe that a manycore architecture should provide shared memory with hardware cache coherence in order to support general purpose operating systems. In a cloud platform context, it is true that we need not allocate all of the resources to a single user, but providing a minimum number of cores is essential to have a sufficient computational power since cores are simple. Besides, running a general-purpose multicore operating system almost requires to have hardware coherence.

The TSAR architecture [1] described in Section 4 and used as a baseline for this work thus provides shared memory with hardware cache coherence.

2.2. Logical partitions and dedicated hardware

A logical partition is an independent operating environment, consisting of a subset of the architecture processors, memory and I/O devices, and running a guest operating system. The guest operating system is a virtualized operating system, running above some kind of hypervisor. As such, a logical partition is one type of virtual machine [9]. Logical partitioning is used in some virtualized environments requiring high insurance, such as separation kernels [10,11]. Commercial services using architecture partitioning for virtual machines include the Infrastructure as a Service (IaaS) provided by IBM [12,13], or Hitashi embedded virtualization technology [14].

Logical partitions can either have dedicated processors or share them. Dedicating hardware to specific guest operating systems has the drawback of rigidity and non optimal use of the resources. However, it comes with a big advantage: by dedicating these resources to the guest operating system, the hypervisor does not necessarily need to interact with the latter, therefore minimizing risks of being compromised. This technique is known as hypervisor disengagement [15]. Besides, this reduced interaction in turn results in a low performance overhead for the virtual machine compared to a non virtualized execution of the operating system.

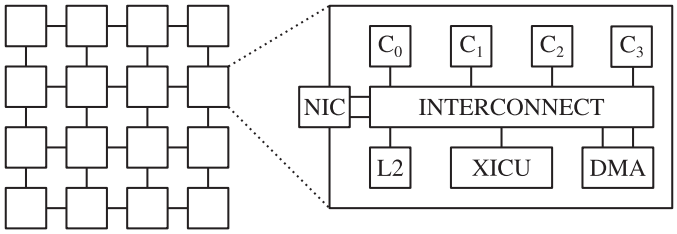

Fig. 1. Manycore architecture.

2.3. Hypervisors and security concerns

Operating system virtualization [16] is a technique which allows to execute an unmodified operating system on a part of an architecture. A hypervisor is generally used to manage the different virtualized operating systems [17,18]. It is a software agent located between the hardware and the virtualized operating systems, and its role is to allocate hardware resources to guest operating systems. As such, it is a security critical point, since every breach in the hypervisor can lead to:

- unauthorized reads of data of a virtual machine (confidentiality violation);
- unauthorized modification of pieces of data of a virtual machine (integrity violation);
- information leakage – data left in memory or hardware components which can be exploited by another malicious virtual machine.

Thus, the hypervisor must be part of the Trusted Computing Base (TCB), i.e. the trusted elements in the system. This is why the hypervisor should remain as small as possible, so as to minimize the risks of it being compromised [9] defines two properties for measuring the hypervisor sensitivity to attacks: *small footprint* and *reduced interaction*. The footprint is traditionally measured in lines of code (LoC), fewer lines meaning fewer bugs in average, and thus fewer possibilities for an attacker to exploit a flaw. Using hardware virtualization extensions, hypervisor can be as small as 4K LoC [19], whereas hypervisor implementing all the virtualization mechanism can reach 100K LoC [20].

Interactions between the hypervisor and a guest operating system happen at launch and shutdown, and every time a virtual machine requires a service from the hypervisor, for example during an I/O access. Hypervisor disengagement allows to limit interactions at their minimum, i.e. launch and shutdown, thus reducing the possibilities for an attacker to exploit a bug in a hypervisor function.

Hypervisors can be classified into several categories. In traditional T1 hypervisors (Fig. 2), a single hypervisor instance manages all the resources, allocates them, and interacts frequently with the guest operating system. For example, every I/O interrupt triggers a context switch to the hypervisor. Other interactions may be required, in particular for memory management if there is no specific hardware extensions, which include an additional privilege mode to the CPU to the user and kernel modes, combined with a MMU extension to translate machine addresses to another