Nested Virtualization meets Micro-Hypervisors: Towards a Virtualization Architecture for User-Centric Multi-Clouds

Alex Palesandro, Marc Lacoste
Orange Labs, France
firstname.lastname@orange.com

Nadia Bennani
University of Lyon, CNRS,
INSA-Lyon, LIRIS, UMR5205,
F-69621, France
nadia.bennani@insa-lyon.fr

Chirine Ghedira-Guegan
Magellan, IAE, University of Lyon 3,
France
chirine.ghedira-guegan@univ-lyon3.fr

1. INTRODUCTION

After a cloud computing decade, the user-centric, fully interoperable, multi-provider cloud remains a mirage. In currently deployed architectures, "horizontal" multi-cloud interoperability limitations come on top of "vertical" multi-layer security concerns. Such issues may be addressed through the virtualization architecture.

For security, two main design requirements are:

1. **Minimal Trusted Computing Base (TCB)** Malicious colocated-tenants may try to exploit flaws in the huge TCB of modern virtualization stacks to escape execution environment isolation.
2. **User-centric Resource Control** Malicious administrators may leverage their extended privileges to snoop the private execution state of applications, without any possible user control. User-centric security allows the user to customize the protection of the virtualization layer to prevent such threats.

For interoperability, two further design requirements are:

3. **Interoperability** An important unsatisfied property of multi-clouds concerns the possibility to migrate execution environments across different providers as transparently as for single clouds. This lack of flexibility is mainly due to provider lock-ins and incompatible technological choices.
4. **Legacy Support** To encourage its adoption, a multi-cloud architecture should require minimal porting effort in application and control logic adaptation.

Existing virtualization architectures only partially such issues, and do not provide an exhaustive solution simultaneously to vertical and horizontal concerns (see Table 1)[1].

Minimal TCB: Micro-Hypervisors (MHs) are the best design alternative. General-Purpose Hypervisors (GPHs) present a TCB bigger than 100 KLoCs, and are traditionally prone to failures due to the size of critical code. On the contrary, Micro-Hypervisors (MHs) [7] reduce the TCB through modularization, expelling in user-space device drivers and other system components.

User-centric resource control: unlike the GPH monolithic approach, Component-based Hypervisors (CBH) [2] offer an interesting architecture. They modularize the traditional control-plane of GPHs (e.g. Dom0 in Xen), opening to users a fine-grained control of their deployed resources. However, this approach requires the user to adapt his control framework to this new logic, breaking compatibility with legacy management toolkits.

Interoperability: several GPH-based Nested Virtualization (NV) architectures may provide a user-centric virtualization layer that could be executed over different providers to implement a multi-cloud virtualization layer. However, such solutions cannot implement a minimal TCB architecture or address interoperability issues [1] without requiring more than two layers of virtualization.

Legacy support: NV inherits the GPH transparency, allowing to control resources with the same interfaces, while CBH and MH introduce a new control logic and/or hypervisor interface.

[1]A number of other related virtualization technologies such as micro-hypervisors and microvisor-based architectures [4] could be considered with intesting trade-offs.

A container-based design might represent a very good trade-off between legacy support, interoperability, near-optimal performance, also providing good scalability. However, security and control requirements are not completely satisfied.

In this paper, we argue that an architecture with a hybrid design could be a viable solution. Indeed, we present a new virtualization architecture combining MH, NV and CBH. Leveraging NV interoperability and legacy support, the architecture provides to users a transparent federation of multiple-provider resources. We also adopt a MH including CBH-like modules as NV lower-layer hypervisor to achieve both a minimal TCB and to enable users to directly control hypervisor resource management components.

2. ARCHITECTURE OVERVIEW

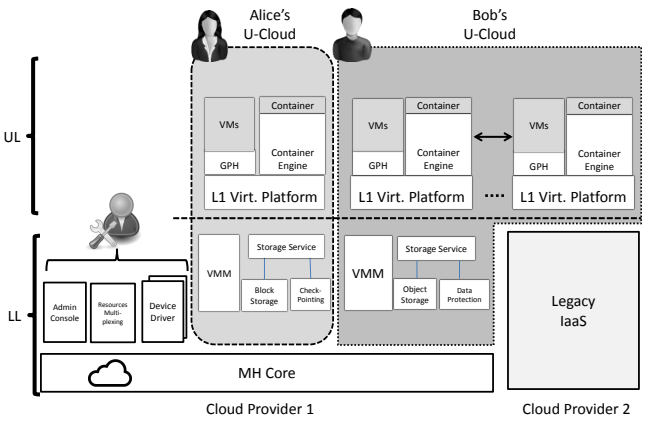

Figure 1: Multi-Cloud Architecture Proposal.

There are two main alternatives to service provider VM protection: (1) direct security control by the customer; and (2) protection by the cloud provider. In (1), security is managed through customer-controlled appliances or intra-VM mechanisms, but remain with very limited privileges. In (2), protection is achieved through provider-controlled appliances or hypervisor-level mechanisms. The objective of the architecture is to overcome flexibility limitations that tie infrastructure services to the provider, causing vendor lock-in for the customer.

High-Level Design In terms of system architecture, we consider a NV-based design to support user-centric clouds *(U-Clouds)* (see Figure 1) [1]. The architecture is composed of two independent virtualization layers: the *Lower Layer (LL)* (realizing the L0 virtualization layer) and the *Upper Layer (UL)* (realizing the L1 virtualization layer).

The LL is composed of a MH under provider control and of a set of user-land infrastructure services for increased user control over U-Clouds. Leveraging the strict modular architecture of MH, the TCB is now reduced and some of most failure-prone components (e.g. Device Drivers) are now outside the architecture core.

Table 1: Existing Virtualization Architectures

Design Requirements	General-Purpose Hypervisor (GPH)	Nested Virtualization (NV)	Component-Based Hypervisor (CBH)	Micro-Hypervisor (MH)	Containers	Best Design Approach
Minimal TCB	X	(✓)	X	✓	X	MH
User-Centric Resource Control	X	(✓)	✓	X	X	CBH
Interoperability	X	✓	X	X	✓	NV
Legacy Support	✓	✓	X	✓	X	NV

The UL federates cloud resources in a provider-independent manner. It implements the interconnection between different cloud providers. The UL also enables to realize the U-Clouds, ranging from lightweight and less isolated execution environments (e.g. Linux Containers) to typical IaaS execution environments (e.g. hardware-assisted VMs). It provides the means for full U-Cloud SLA customization, notably through configuration of the required LL-level user-infrastructure services[2].

LL Design The general LL design is hybrid between minimizing the virtualization layer (micro-hypervisor à la NOVA [7]) and control disaggregation with several user domains (CBH à la SSC [2]). We assume the provider-layer to be MH-based. This assumption may hold for an open cloud provider architecture (Cloud Provider 1 on the Figure). This design choice provides a solid foundation to meet all vertical features, notably smaller TCB size and enhanced global security such as guaranteeing VM security even if intermediate layers are compromised. This assumption does not seem unreasonable for upcoming years as: (1) despite most current IaaS platforms still being GPH-based (Cloud Provider 2 on the Figure), there is a strong trend towards making the hypervisor more minimal and flexible, as witnessed by disaggregation of hypervisors of the mainstream IaaS platforms; (2) MHs themselves are becoming component-based, with possibility of interoperability to avoid any further risk of IaaS lock-in as shown by first multi-MH OSes.

The *MH core layer* provides a tiny set of key kernel features, e.g., scheduling, MMU management, IPCs, handling VMX-related events and interrupts. Following CBH principles, each user controls a *VMM* and a set of *user-space services*. The VMM is a lightweight implementation of the L0 hypervisor logic, with the same reliability and security benefits as in NOVA. The VMM is also crafted to enable NV through support for running nested VMX instructions. User-space services enable users to customize infrastructure resource management for their U-Clouds. The cloud system administrator directly controls resource management policies, system-wide resource multiplexing, and device drivers, but without control over the MH core.

From the user standpoint, the LL architecture increases control over the infrastructure, adding incrementally new infrastructure management services under the hood without disrupting legacy applications. From the provider standpoint, security is strongly enhanced due to two-level isolation by the MH core and L0 hypervisor virtualization. Integrity of user-controlled L0 system services can be guaranteed through hardware security mechanisms[3].

UL Design The general UL design does not export hardware resources with a single interface, but with a spectrum of interfaces of varying abstraction levels. Indeed, the UL should allow building U-Clouds where VM, network, and storage SLAs may be personalized independently from the underlying provider. Such U-clouds may be defined at the PaaS or IaaS levels. We consider a L1 virtualization platform enabling to support both VMs and containers, and thus having a flexible virtualization interface, ranging from hypervisor to OS-level virtualization on which such EEs may run [6].

[2]Extra arbitration components are needed in the architecture for the customer to choose between infrastructure services under his control and those under provider control.

[3]For instance, Intel TXT/SGX technologies.

Extending the library OS philosophy that adapts the OS to the application [5], the L1 platform should adapt the virtualization technology to applications according to different trade-offs. The UL should thus realize the widest possible spectrum for supported L1 virtualization techniques. Eventually, the UL will support dynamic on-demand provisioning of virtual execution environments with their GPH/container-engine. The L1-virtualization platform is GPH-based to guarantee interoperability and other horizontal features through an inter-cloud blanket layer [1].

3. NEXT STEPS

Ongoing work focuses in two main directions. First, we are implementing a proof-of-concept prototype of this architecture using Linux as L1 GPH and NOVA as L0 MH. NOVA appears promising for L0 due to its clean design and code availability. With this platform, we are focusing on implementing vertical properties in a single cloud setting[4], with extension to horizontal properties in a distributed cloud setting.

Second, we are investigating how this architecture could enable to build a unified security management plane for multi-clouds with the following features: (1) unified autonomic security management of the infrastructure both across layers and providers; (2) building U-Clouds with fully à la carte security SLAs. Indeed, such management plane will face 2D threats, vertical spanning layers, and horizontal, spanning domains – multi-cloud being both multi-layer and multi-domain. To implement 2D security management proper, we intend to rely on self-protection frameworks which have been defined for both cross-layer [8] and multi-IaaS integrated security monitoring. Furthermore, we are investigating metrics and identifying concrete evaluation criteria for the overall architecture.

Acknowledgment

This work was partly funded by the French Ministry of Education and Research (CIFRE grant) and by the EU H2020 SUPERCLOUD project (`www.supercloud-project.eu`) (grant no. 643964).

References

[1] D. Williams et al. "The Xen-Blanket: virtualize once, run everywhere". In: *EuroSys '12*.
[2] S. Butt et Al. "Self-service Cloud Computing". In: *CCS '12*.
[3] *I/O Virtualization - SRIOV*. URL: `https://www.pcisig.com/specifications/iov/`.
[4] A. Iqbal et al. *An Overview of Microkernel, Hypervisor and Microvisor Virtualization Approaches for Embedded Systems*. Tech. rep. Lund University, 2009.
[5] A. Kivity et al. "OSv - Optimizing the Operating System for Virtual Machines". In: *USENIX ATC 14*.
[6] D. C. van Moolenbroek et al. "Towards a Flexible, Lightweight Virtualization Alternative". In: SYSTOR 2014.
[7] U. Steinberg et al. "NOVA: a microhypervisor-based secure virtualization architecture". In: EuroSys '10.
[8] A. Wailly et al. "VESPA: Multi-layered Self-protection for Cloud Resources". In: ICAC '12.

[4]Device drivers are a major MH issue, due to costs of development or of adaptation to a new architecture. Such challenges may be addressed using device assignment: the L1 GPH hypervisor manages the physical device with the proper driver now running in a lower privilege level. To be workable, this approach would require exclusive allocation to be simply implementable, e.g., using SR-IOV [3].