Toward On-demand Nested Virtualization for Live-Refreshing Cloud Systems

Ryosuke Yasuoka
yasuoka@osss.cs.tsukuba.ac.jp
University of Tsukuba

Takaaki Fukai
fukai@os.ecc.u-tokyo.ac.jp
The University of Tokyo

Takahiro Shinagawa
shina@ecc.u-tokyo.ac.jp
The University of Tokyo

Extended Abstract

Virtual machine monitors (VMM) are a crucial component in cloud systems. To achieve high availability, VMMs should run continuously without stopping. However, VMMs must be rebooted frequently for self-refreshing, such as applying security patches, upgrading VMMs, and rejuvenating the entire systems. Since stopping cloud services while rebooting VMMs is unacceptable, live-refreshing cloud systems including VMMs is a practical and vital issue for cloud vendors [2]. VM live migration is a generic and effective approach to refreshing VMMs without stopping virtual machines (VM) running on them. However, traditional VM live migration requires many spare physical machines and vast network resources to transfer VM images up to several GiB each [2].

Recent works exploit nested virtualization to run two VMMs on a single machine [1]. This approach significantly reduces the memory copy cost for VM migration. Unfortunately, current nested virtualization still incurs high overhead due to complex multi-level virtualization. Since the nested virtualization overhead is incurred continuously even when only one VMM is running, reducing the overhead during long normal runtime is indispensable. However, eliminating nested virtualization overhead in the traditional VMM architecture is difficult because the upper (L1) VMM heavily depends on the execution environment provided by the lower (L0) VMM. In addition, since traditional VMMs are large and complex, aging of the L0 VMM becomes a problem.

In this poster, we propose a new concept of *on-demand nested virtualization*, which allows disabling nested virtualization during normal runtime and temporary enabling nested virtualization when refreshing VMMs. The main challenges of this approach are how to enable and disable nested virtualization dynamically and how to run the two VMMs after enabling nested virtualization.

We achieved dynamic enabling and disabling of nested virtualization by swapping the VMM states of the hardware-assisted virtualization of the L0 and L1 VMM. In disabling and enabling nested virtualization, we must avoid changing the hardware interface exposed to the L1 VMM so that the L1 VMM can continue to run. To achieve this, the L0 VMM partitions hardware resources (CPU, memory, and I/O devices) into two logical partitions and assigns a partition to each L1 VMM instead of fully-virtualizing hardware. Avoiding full hardware virtualization also contributes to simplifying the implementation and reducing the aging of the L0 VMM. To run the second L1 VMM in addition to the first L1 VMM, the L0 VMM partially virtualizes hardware resources such as a physical address space, interrupt controller, and firmware.

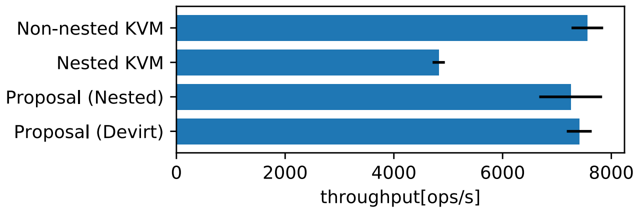

Figure 1. Redis YCSB

We implemented a prototype of our proposal on Intel CPUs. We omit the protection between the L0 and L1 VMMs in nested virtualization because we assume cloud administrators manage both L0 and L1 VMMs. It significantly simplifies the implementation and improves performance. We used KVM as the L1 VMM and appended a small piece of code to implement the starting point of the nested virtualization. For dynamic resource partitioning, we exploited hot plug/unplug features of Linux running as the host OS of KVM.

We believe that our work is the first one to propose the concept of on-demand nested virtualization as well as demonstrating its feasibility by presenting its design, implementation, and useful application.

As a preliminary evaluation, we measured the performance of Redis (version 4.0.2) using the YCSB benchmark client (version 0.12.0). The workload used was A — update heavy where read and write operations were both 50%. Figure 1 shows the result. In "Nested KVM," the performance was significantly degraded; the overhead was 36.2%. On the other hand, "Proposal (Nested)" incurred only 4.1% overhead, although the standard deviation was slightly larger. The overhead in "Proposal (Devirt)" was only 2.0%, and the standard deviation was similar to that of "Non-nested KVM."

References

[1] Spoorti Doddamani et al. 2019. Fast and Live Hypervisor Replacement. In *Proc. 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2019)*. 45–58. https://doi.org/10.1145/3313808.3313821

[2] Xiantao Zhang et al. 2019. Fast and Scalable VMM Live Upgrade in Large Cloud Infrastructure. In *Proc. 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. 93–105. https://doi.org/10.1145/3297858.3304034