One-pager 5

Martin VASSOR

1 Introduction

The question of comparing *processes* and *virtual machines* can be addressed in two –equivalent– ways: from the viewpoint of a process (resp. VMM), what is expected from the OS (resp. VMM)? Or from the viewpoint of the OS (resp. VMM), what should be provided? The approach of this proposal is the latter.

Models As VMMs and OSes can be implemented in many ways. To provide a general answer, I consider minimal definitions for both systems: micro-kernel (as in [1]) and VMM as in [2]. LIEDTKE claims two requirements for micro-kernels: *independence* and *integrity*. This proposal shows that VMMs ensure the same properties. This equivalence results in equivalent exposed interfaces, hence similar *process* and *virtual machine* foundations.

2 Independence

Each arbitrary *subsystem* have to be implementable such that it is not corrupted nor disturbed by another subsystem. In both systems, this is guaranteed by virtual memory and pre-emptive scheduling.

Virtual memory In a micro-kernel, virtual memory is aided by hardware (TLB and page-walk) and in the VMM abstraction, this is guaranteed by the usage of *relocation-bounds register*.

Pre-emptive scheduling Both systems have the possibility to preempt and abort subsystem. An example to achieve that is to share the computing time in timeslices managed by hardware counters.

3 Integrity

To cooperate, processes should be able to exchange messages safely. Micro-kernels have to implement a trusted IPC. Integrity is defined as the following: for any two processes P_1 and P_2, if P_1 knows P_2, P_1 should be able to create a channel to P_2 which is neither corrupted nor eavesdropped.

In VMMs, VMs are isolated and the only knowledge they have of other VMs is via the network card (if any). All communication is done via the network, and the VM does not distinguish an other VM from a real distance machine. Integrity is hence ensured depending on the network properties.

4 Limits & conclusion

Limits: While Sections 2 and 3 provide strong similitudes between OSes and VMMs, there are still some differences:

Exposed interface OSes provide processes an ideal machine abstraction while VMMs provide a real machine abstraction.

Other kernels Regular OSes provide much more features than micro-kernels such as filesystems, etc. Which have not been studied here. However, VMMs have to provide at least basic equivalents (for instance virtual hard drive), etc.

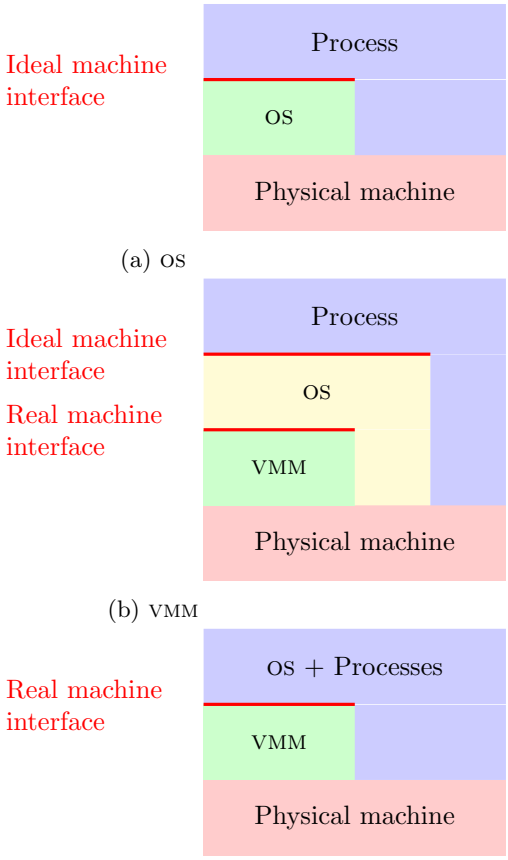
Conclusion: OSes and VMMs fundamentally serve the same purpose: providing a machine abstraction to upper layers (see Figure 1a and 1c). The difference lies in the usage of the abstraction: OSes provide an ideal abstract machine (for programming purposes) and VMMs provide an exact hardware machine abstraction.

(word count: 499 including section titles, caption, excluding references)

References

[1] J. Liedtke. On Micro-kernel Construction. *SIGOPS Oper. Syst. Rev.*, 29(5):237–250, December 1995.

[2] Gerald J. Popek and Robert P. Goldberg. Formal Requirements for Virtualizable Third Generation Architectures. *Commun. ACM*, 17(7):412–421, July 1974.



(a) OS

(b) VMM

(c) VMM as an OS

Figure 1: From the point of view of the VMM, the *process* and OS layers are indistinguishable (1b is view as 1c), resulting in the same pattern than a regular OS.