Rearchitecting System Software for the Cloud

Muli Ben-Yehuda
Technion
muli@cs.technion.ac.il

Dan Tsafrir
Technion
dan@cs.technion.ac.il

1 The Cloud: Challenges and Opportunities

The computing landscape is changing and the system software stack must change with it. Infrastructure-as-a-Service cloud computing, as exemplified by Amazon's EC2, is a new kind of run-time platform. In the cloud, the resources available to a virtual machine (e.g., CPU cores, memory, I/O devices) may be priced differently based on supply and demand [1, 2]; they may be available when the price is low and may cease to be available when the price is high. In such an environment, both the users—who pay for resources—and the providers—who pay for infrastructure and operations—have strong economic incentives for running ever more efficient computing systems. When you pay for every cycle and every byte, run-time performance matters.

Traditional system software stacks are ill-equipped for such a run-time platform. Traditional operating systems abstract away from their applications the underlying hardware resources, incurring overhead; they control all resources and decide how to allocate them to their applications; they provide many one-size-fits-all services, regardless of their applications' specific needs; and they assume that the underlying platform and its resources remains constant.

The rise of cloud computing has also brought with it a host of new hardware support for efficient machine virtualization. New CPU, MMU, chipset, and I/O device support can greatly increase machine virtualization run-time performance, to the extent that it is now possible to run an I/O-intensive virtual machine with runtime performance that is within 97%–100% of bare-metal performance [6]. This new hardware support also enables multiple virtual machines to share an I/O device without any hypervisor involvement on their I/O paths [3, 7].

The rise of cloud computing provides an opportunity to rethink the entire software stack. The goal of this work is to explore a new system software architecture for applications running in the cloud, under two fundamental assumptions: that such systems will seek utmost efficiency (measured in cost per unit of useful work) and that such systems must be able to adapt to changing resource conditions. The primary vehicle for this exploration is a new operating system—called `nom`—which we are currently developing.

2 The `nom` Cloud Operating System

In the 1990's, Engler et al. [4, 5] proposed the exokernel operating system. In the exokernel model, the OS provides minimal services to its applications, imposing no interfaces on them except those of the underlying hardware. Yet the actual underlying hardware could not have been *really* shared between multiple uncooperative applications, so the exokernel did not expose it directly. Instead, it attempted to provide the same interfaces as the underlying hardware. Thus, the I/O path of exokernel applications still involved many user/kernel crossings and the applications depended on the kernel for their I/O needs.

Conversely, `nom` takes the exokernel vision to its ultimate conclusion, by removing the kernel from the I/O path completely. Using natively-sharable SR-IOV devices [3] and IOMMUs with I/O page faults, `nom` exposes I/O devices to applications safely and directly. In addition to the exokernel advantages of application specialization and driver reliability, exposing I/O devices directly to applications improves efficiency, since there is less non-useful work to do, and it provides applications with the opportunity to adapt to the underlying resource availability.

The `nom` operating system has no kernel-resident drivers. Instead, applications link at run-time with libraries that provide I/O services without any kernel involvement. The kernel's role is limited to resource management—handing out frames of physical memory, coarse-grained CPU core scheduling, allocating I/O devices to applications—and it does not provide any I/O services. The `nom` kernel provides default versions of I/O-libraries that applications link with and use, but applications are free to use their own specialized I/O-libraries.

Through `nom`, this research will attempt to answer questions such as: What does "useful application work" mean and how do we measure it? How can applications adapt to changing resource availability, making optimal use of the available resources? What are the roles of operating systems and hypervisors in such an environment, and how should they be constructed? For example, do we need a hypervisor/supervisor/user split? Who should own memory translation, setting up and tearing down page tables? How do we provide applications with safe, direct hardware access, without involving the kernel on the I/O path? How do applications communicate with other applications without kernel involvement? In such a system, is there a fundamental difference between applications and virtual machines? Or are all applications, in effect, virtual machines?

References

[1] Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., and Tsafrir, D. Deconstructing Amazon EC2 spot instance pricing. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)* (2011).

[2] Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., and Tsafrir, D. The Resource-as-a-Service (RaaS) cloud. In *USENIX Conference on Hot Topics in Cloud Computing (HotCloud)* (2012).

[3] Dong, Y., Yang, X., Li, X., Li, J., Tian, K., and Guan, H. High performance network virtualization with SR-IOV. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2010).

[4] Engler, D. R., Kaashoek, M. F., and O'Toole Jr., J. Exokernel: an operating system architecture for application-level resource management. In *ACM Symposium on Operating Systems Principles (SOSP)* (1995).

[5] Ganger, G. R., Engler, D. R., Kaashoek, M. F., Briceno, H. M., Hunt, R., and Pinckney, T. Fast and flexible application-level networking on exokernel systems. *ACM Transactions on Computer Systems (TOCS) 20*, 1 (February 2002), 49–83.

[6] Gordon, A., Amit, N., Har'El, N., Ben-Yehuda, M., Landau, A., Tsafrir, D., and Schuster, A. Eli: Bare-metal performance for i/o virtualization. In *ACM Architectural Support for Programming Languages & Operating Systems (ASPLOS)* (2012).

[7] LeVasseur, J., Uhlig, V., Stoess, J., and Götz, S. Unmodified device driver reuse and improved system dependability via virtual machines. In *Symposium on Operating Systems Design & Implementation (OSDI)* (2004).