Containing the Hype

Kavita Agarwal, Bhushan Jain, and Donald E. Porter
Stony Brook University
{*kaagarwal, bpjain, porter*}@*cs.stonybrook.edu*

1. Abstract

Operating System-level virtualization, also known as a *container*, is an increasingly popular approach to isolating applications that use the same underlying OS kernel [2, 6–8]. Containers have increased in popularity recently as the default back-end for Docker, an application packaging and distribution system used by companies including Google [5].

The purported reason to use containers over a hardware virtual machine, such as VMware or Xen, is reduced overheads. Containers are generally implemented by copying a subset of OS data structures, which one would expect to be lighter-weight than running another complete OS instance. Similarly, data structure initialization can be faster than booting a legacy OS kernel. However, most existing comparisons of these two technologies are ad hoc at best. There is a remarkable lack of scientific studies on this topic, and existing studies are either dated in a rapidly evolving field; narrowly focused on particular application areas; or overlook important optimizations, leading to exaggerated results.

In the interest of adding experimental data to the scientific discourse on this topic, this paper contributes a careful, updated comparison of the two technologies. Throughout the paper, we use Linux's KVM [4] and LXC [1] as representative examples of both technologies. We organize this comparison around two important metrics for cloud efficiency: consolidation *density*, or guests per physical machine, and the latency to start a new guest—a metric relevant to quickly meeting to spikes in demand. If start-up latency is too high, providers will generally provision for peak demand instead of average demand to minimize worst-case request latency.

Our comparison yields a more nuanced set of benefits and some drawbacks to each option. For instance, simple checkpointing optimizations yield start-up times for KVM that are an order of magnitude lower than reported by previous studies [3]. Moreover, the start-up time of either is unacceptably high to dynamically scale instances of latency-sensitive applications. Similarly, overall density of either technology is highly dependent on the most contended resource, and, for several resources, neither approach has a clear advantage.

Finally, this paper contributes the first detailed analysis of the worst case for VMs relative to containers: memory consumption. We identify several opportunities to improve both technologies and reduce this overhead for VMs. We believe creating VMs with overheads equivalent to containers is a useful and interesting "challenge problem" for future research. We expect this challenge may not be completely realizable, as there are likely fundamental memory overheads for VMs relative to containers, but we also identify considerable bloat that could be reduced.

This is a poster for an accepted paper.

References

[1] Linux container. `https://linuxcontainers.org/`.

[2] S. Bhattiprolu, E. W. Biederman, S. Hallyn, and D. Lezcano. Virtual servers and checkpoint/restart in mainstream Linux. *ACM SIGOPS Operating Systems Review*, 42:104–113, July 2008.

[3] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. An updated performance comparison of virtual machines and linux containers. Technical Report RC25482(AUS1407-001), IBM Research Division, 11501 Burnet Road, Austin, TX, 2014.

[4] Kernel-based virtual machine. `http://www.linux-kvm.org/`.

[5] C. Metz. Google embraces docker, the next big thing in cloud computing. *WIRED*, June 2014. `http://www.wired.com/2014/06/eric-brewer-google-docker/`.

[6] D. Price and A. Tucker. Solaris Zones: Operating system support for consolidating commercial workloads. In *Proceedings of the Large Installation System Administration Conference (LISA)*, pages 241–254, 2004.

[7] S. Soltesz, H. Potzl, M. E. Fiuczynski, A. Bavier, and L. Peterson. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. In *Proceedings of the ACM European Conference on Computer Systems (EuroSys)*, 2007.

[8] M. Stokely and C. Lee. The FreeBSD Handbook, 3rd Edition, Vol 1: Users's Guide, 2003.