Container-based operating system virtualization: A Scalable high - performance alternative to hypervisors

Stephen Soltesz, et al.— Mario Vazquez Corte (127252)

I. Resume

Virtualization requires a share of resources while having two, probably opposing, objectives. On the one hand, users are looking for and isolated and secure environment. On the other hand, users need efficiency for running multiple tasks. For example, personal computers focus on efficiency rather than isolation, while the hypervisors that support the virtual machines (VMs) favor security and isolation. VMs are better in terms pf security and isolation, while container virtualization focuses on efficiency and scalability. The paper compares both technologies using the Xen VM technology versus Linux-VServer that is container based, as representatives for each type of technology.

Hypervisors allow to run multiple independent kernels (even from the host OS). From an abstraction point of view VMs try to emulate the physical hardware, meanwhile, containers share the same kernel as the host and are treated as any other process (but using their own namespaces and cgroups). Containers (COS) are directly embedded with the host OS and thus allow only a single kernel.

For fault isolation, both technologies, have their own directory space. Both have mechanism for avoiding other instances to consume all resorces. Eaxh hypervisor controls a particular and immutable set of hardware/resources, while COS resources are mostly orchestrated by the host OS. For distributing the CPU, each technology receives a set of tokens that is spent every time an instance uses the CPU, and must leave the resource if it runs out of them. VMs are ordered in a queue to avoid *cross talk* (interaction between VMs). If a machine is consuming more than the available resources the daemon watchdog intervenes to solve the problem. Security is implemented via contexts and filters at system call's layer. The former stops resource sharing and the latter gives access controls. COS share common libraries and binaries which improves efficiency. If and instance needs to modify them extra copy is created for such instance.

The authors proceed to run some experiments to compare Xen and VServer. For the micro-benchmarks the performance of VServer is always within 1% of the native/Linux performance while Xen struggles a little more in this department. In terms of bandwidth they measure the raw throughput and the CPU utilization of the receiver under TCP conditions. For both measures Xen behaves worst. The same occurs for the macro-benchmarks where VSerer has basically the same performance as the LInux/native while Xen has overhead due to buffering, copying and synchronizing with the host.

In the performance at scale chapter, the authors run both technologies with an instance of PostgreSQL to serve the OSDB test. Xen degrades quickly when more instances appear, therefore the performance of VServer is significantly larger. VServer clearly has a better scheduler for this conditions than Xen.

To study fair shares and reservations, authors use a combination of CPU intensive tasks that involve no I/O. Two experiments are analyzed. In the first one, the two technologies receive a a fair share of CPU time. In the second one, both are given a fourth of the overall CPU time. In general VServer has better results than Xen. For performance isolation, as expected Xen is better than VServer. The paper compares both technologies under attacks. Xen explicitly partitions of the physical memory makes it less vulnerable to attacks.

The paper concludes that both technologies are good for certain requirements, and therefore there is a necessity to improve them since there is a natural trade off between isolation and performance.

II. Critique

The paper makes a great case for comparing both technologies or paradigms. Although, it seems a little older since recent literature makes a distinction for both technologies as light-weight virtualization (containers) vs hypervisors (VMs). They seem to generalize the results rather quickly, Xen pr VServer may not be representative of both technologies. Recent literature usually uses KVM and Xen as representatives of VMs while Docker, OpenVZ and LXC as the standard container tech.

The abstraction model they use for containers and hypervisors is not clearly defined in the paper, it was a little shaded. This may be caused by the year the paper was created. New literature make a better case separating bare-metal hypervisors, hosted hypervisors and containers, although the division for the former is not specially clear.

They generalize the scaling results using too little instances. The most extreme case they present uses 8 instances. This is too few instances for a Data Center or cloud services. This may be due to the age of the paper.

In general the paper has good and robust experiments, both fails in terms of generalization since such experiments are not exhaustive or representative.