Survey on Linux Containers and Its Applications

Hyungro Lee
School of Informatics and Computing
Indiana University
Bloomington, USA
lee212@indiana.edu

Gregor von Laszewski
School of Informatics and Computing
Indiana University
Bloomington, USA
gvonlasz@indiana.edu

Geoffrey Fox
School of Informatics and Computing
Indiana University
Bloomington, USA
gcf@indiana.edu

ABSTRACT

Last two and half years, virtualization using containers such as Docker, LXC has emerged as a new trend for running applications on the isolated environment. Container-based virtualization is appealing to developers and service operators since it provides portable and lightweight runtime environments for your cloud applications. However, the containers offer native-like performance and promising features over the Infrasture-as-a-Service, the container virtualization is evolving with many issues and missing features. In this paper, we present a survey of container virtualization and its tools including differences from hypervisor-based virtualization, and the Linux kernel supports. We expect to provide a guideline and current state of the development from the containers community and address current challenges from the research aspects.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Linux Containers, Virtualization

1. INTRODUCTION

1.1 Virtualization

Virtualization is a technology to enable cloud computing which provides isolated computing resources for high level applications without dependency of physical hardware. Infrastructure as a Service (IaaS) offers hardware virtualization with additional abstract layer called hypervisor in a virtual machine (VM). Building clusters of virtual servers with different operating systems is one of the capabilities of utilizing IaaS compute resources. On the other hand, Platform as a Service (PaaS) offers application centered services to users who wish to focus on application development rather than building and configuring servers themselves. Different abstract layers between IaaS and PaaS offer flexibility to run user applications with virtualization technology.

Operating System (OS) level virtualization provides a separated user space above the shared kernel of a hosted operating system. The isolation guarantees individual resource limits per each user space so that heavy and noisy neighbors (processes) do not affect other processes regarding the use of resources such as CPU, memory, disk I/O, or network on a same physical hardware. In the Linux kernel, containers provide isolated virtual environment for each process under the OS-level virtualization, which is also called container virtualization or container-based virtualization. The control groups (cgroups) in the Linux kernel provides a resource limitation towards a group of processes and multiple workloads run with resource partitioning on a same host. The six namespaces in the Linux kernel provides isloation of process trees, (PIDs), network interface controllers, user lists, filesystems, a hostname and IPC from the global resource in the same kernel and each process has individual space from the shared resources. Linux Containers (LXC), Docker libcontainer (previously Google's lmctfy; Let Me Contain That For You), and Open Container Project's (OCP) runc use the cgroups and namespace as container execution drivers.

1.2 Container Virtualization

Container virtualization is evolving technology and there are several applications to support large number of containers for production scale operations. This study focuses on Linux container virtualization although there are other implementations including FreeBSD Jails, Solaris Containers (also known as Zones), OpenVZ, and AIX WPARs.

1.3 Kernel Support

The Linux kernel creates a container by isolating user spaces and managing resources via kernel supports i.e. cgroups and namespaces which are mostly available in the recent kernel. The unified control group hierarchy, for instance, is available in the kernel 3.16 and the Ubuntu 14.04 or higher release only has the latest kernel which is necessary to avoid any missing features for container virtualization. For example, the state of the secure computing is added to the process status file (/proc/[pid]/status) by the kernel 3.8. The secure computing facility (seccomp) in the kernel is necessary to reduce kernel attack surface because the seccomp controls systems calls with filters. There are list of system calls to be blocked from containers because they canbe used to obtain acccess of other containers through a kernel vulnerability. Linux Containers (LXC) 1.0 requires the kernel 3.12 or above to enable unprivileged containers and 3.10 Linux kernel is a minimum requirement for Docker.

1.4 Execution Engines

To run multiple containers on a single kernel, execution engines like Docker work with the Linux kernel features such as namespaces, cgroups, seccomp, chroot and apparmor. Each container runs applications in an isolated user space with own filesystems, process trees, user ids and network interfaces but share a same hardware on a single operating system. More than a thousand developers contributed Docker, open source application container engine, last two and half years, "to pack, ship and run any application as a lightweight container" in their own words from the Docker's github repostiory. Docker is an implementation of container virtualzation with the execution drivers such as LXC, libcontainer or runc and application images to build containers. Execution drivers provide abstraction between the kernel and containers and container images like Docker Image are used to contain all required resources to run the applications. For example, Docker Image is built with a Dockerfile which is information of applications to run and the image files are stored in a registry to retrieve and download. Docker Hub is a public registry to store and share docker images but other software or services are available to offer a private registry such as Quay.io or Dogestry. The portable container images can be launched on any linux hosts with Docker client tools by pulling images from the registry without dependency issues.

1.5 Execution Drivers with Specification

libcontainer introduced with Docker 0.9 release to provide built-in execution driver on Docker and become a standard driver. The development and specification of libcontainer has been merged to RunC recently to provide a open standard with App Container (appc). RunC unites efforts from Docker's libcontainer and CoreOS's appc by Open Container Initiative (previously called the Open Container Project).

1.6 Performance

The studies [3, 4] indicate that linux containers outperformed hypervisor-based virtualization in CPU, Memory, Network, and storage evaluation.

2. CONTAINERS AND ITS APPLICATIONS

The current implementations of containers are addressed in this section. The core components for containers are described and the additional software are introduced in terms of container clustering in a large scale.

2.1 Container technologies

2.1.1 Docker

According to the recent reports [1, 2], the majority of respondents chosen Docker as their containers management tools (92% of 285 responses by ClusterHQ and DevOps.com in May 2015, 70% of 745 responses from StackEngine in January 2015) . Docker, the set of software creating, initializing and running containers on Linux-based systems, gains a lot of popularity due to its portability with the kernel supports and simplicity of running applications without worrying about dependencies. Their in-house execution driver, libcontainer, is now integrated to the universal specification, RunC by Open Container Initiative (OCI) to increase the support on various container tools, services, etc. One of the key components of Docker is the union file systems (UnionFS) which provides different branches for read-only and read-write with copy-on-write (CoW) with several implementations such as aufs, overlayfs, btrfs, vfs and DeviceMapper. Other software managed by Docker, Inc and the community developers ease building, deploying and managing Docker containers. Docker Swarm provides a container mamangenet in a cluster level, Docker Compose (previously known as Fig by Orchard) connects container images to run a service by reading their instruction file "docker-compose.yml". More than 140,000 repositories are available on Docker Hub to share or download Docker container images. Docker Machine helps you start using Docker on a desktop or on the cloud. Docker was an internal project in dotCloud, a PaaS company, and became the most popular container management tool since its first release in March 2013.

2.1.2 Rocket

CoreOS group launched Rocket (rkt) project, a containerization system like Docker but based on App Container (appc) specification and a different approach on the architecture. Appc is a container specification designed by CoreOS group for image format, runtime environment and discovery protocol but migrated to the Open Container specification to develop and offer a standard container format together with many other groups in industry. The difference between Docker and Rocket is the existence of a daemon. Rocket runs container images directly with init systems and its commaind line tool, while Docker requires a daemon to run containers. Docker image can be used in the Rocket command line tool or the docker2aci converter. Along with the image format, the Application Container Image (ACI) defined by appc, the Pod id implemented in Rocket. Pod is a group of application images for the execution environment and Google's Kubernetes has a pod with a same idea. According to their roadmap, rkt 1.0 will be released in late 2015 with a non-root mode and the support for other linux distributions such as Fedora, Gentoo, Debian, CentOS, and OpenSUSE.

2.1.3 LXD

Canonical, the company behind the Ubuntu distribution, announced LXD, the LXC container management tool and shipped LXD 0.7 on the latest Ubuntu 15.04. Its client-server mode allows to create LXC containers remotely via the OpenStack Nova Compute plugin for LXD, nova-compute-lxd (nclxd) is a example using the REST API. LXD seems the first userspace tool that supports live migration ans snapshot using CRIU, an utility to checkpoint/restore a process tree for Linux. Canonical uses 'the container lighervisor' to describe LXD since their user interface behaves like IaaS with a hypervisor.

2.2 drivers

Container drivers are important to create and manage container virtualization environments by communicating with the kernel from a userspace. Hypervisor creates a virtual environment with a new operating system but the container drivers share the kernel of the host operating system with multiple containers without starting a new operating system. With this different approache between a container and

a hypervisor, we find the pros and cons in running applications on virtual environments. For example, hypervisor allows to run Windows, Linux, or other operating systems in a virtual machine (VM) but overhead is expected due to the additional layer between hardware and a VM. Containers provide native-like performance but can not use other operating systems because containers run under a same kernel on a single operating system.

2.2.1 LXC

The linux container (LXC) is to create and run containers on Linux systems by using the kernel features like chroot, cgroups, and namespaces. Multiple virtual environments can be created on a single kernel by LXC but separated namespaces for process identifiers (PIDs), user identifiers (UIDs), etc. are guaranteed to enable isolation.

LXC 1.0 is announced in early 2014 with five year long-term supports. The key feature of LXC 1.0 is unprivileged containers which runs containers as non-root users on the host operating system. This prevents taking a root privilege on the host from any incident because a root user (uid 0) inside a container actually has no privileges on the host with uid 100000 or higher, for example.

2.2.2 libcontainer

Docker used to use LXC as its container execution driver before Docker 0.9. Libcontainer is a built-in execution driver, written by Go programming language to access kernel features for creating containers. libcontainer is migrated to the container command line tool (runc) under OCI as collaborative effort with others including CoreOS's AppC.

2.2.3 systemd-nspawn

systemd-nspawn is a command line tool for testing, debugging and building a command or a operating system in a container like chroot but with an init binary if available. It may be called "nspawn containers powered by systemd". systemd-nspawn was in between chroot and LXC in terms of features and supports for containers. systemd-nspawn is powerful than chroot because its configuration by systemd but systemd-nspawn is originally developed for testing experimenting and debugging, which is not meant for a production service. For example, networking was only available by sharing the host networking, otherwise loopback interafece was only available in a separate network namespace. In the latest version of systemd-nspawn, many features are added including network supports. Virtual network interface can be established working with systemd-netword which is a system daemon for network configuration. As systemd-nspawn matures, production use is considered. Rocket uses systemd-nspawn as its backend container manager and Docker supports system-nspawn as one of the execution drivers.

2.3 Service Discovery

ZooKeeper has been a coordination service to manage a cluster of machines regarding service discovery, recovery, updates of node information and leader election. In the container virtualization, similar features but different approaches have been made with init systems like systemd. This helps to check the health of containers and state with the applications running on.

2.3.1 doozerd

Doozerd provides key-value store in memory using paxos algorithm. The format of storing key-value data is like using a filesystem with a directory tree. For example, database information can be stored in the following format: /services/db/hostname:port/id = "albert" and /services/db/hostname:port/pass = "admin". Stackato, PaaS service by HP (previously ActiveState), used doozered in their system.

2.3.2 Consul

Consul by HashiCorp uses the raft protocol to store key-value data. Service discovery, configuration, membership status, leader election and health checking are supported with UDP for less sensitive data and TCP connections.

2.3.3 etcd

Etcd is another key-value store service developed by CoreOS. Etcd watcher provides realtime push notification using HTTP long-polling which is similar to Doozerd Watchers. Raft protocol is also used in etcd.

2.3.4 zookeeper

ZooKeeper has been adopted in many software packages as a coordination service. Ubuntu Juju, Katta, Mesos, Neo4j, Apache Hadoop, Apache Kafka and Apache Solr uses ZooKeeper. Zab protocol is used which is like paxos.

The other tools that we haven't addresse are SmartStack, confd, Noah, Corosync, Jgroups, Accord and OpenReplica.

3. CONTAINERS IN VIRTUAL CLUSTERS

If you need to run container applications for your service with heavy traffics or large datasets, cluster managment is necessary to control containers on multiple server groups. Job management, service discovery, configuration management, image management are required along with resource scheduling. A group of software packages are used to build Virtual Clusters (VC) using the container virtualization. Minimal lightweight operating systems are preferred with the latest kernel supports to utilize process virtualization which are application containers and Table 1 describes the list of software packages. The job management including execution and scheduling describes managing workload with priority, availability and dependency. The image management describes downloading, powering up, and allocating images i.e. VM images or container images for applications in terms of planning, preparing and booting images. The configuration service describes storing cluster membership, service discovery, leader election and quorum for high availability. The language indicates a development programming language used in the tool and the 'what can be specified with extent of ontologies' describes supported applications to use. The Figure 1 shows container software with categories based on each role in the container virtualization. The container runtime environment provides the isolation of a virtual environment for a running container process by cgroups and namespaces from the linux kernel without a hypervisor. The tools in this category are mainly used to start, build and run container applications. The supports from operating systems help start a virtual environment quickly with less security issues from unnecessary packages but with new features

such as checkpoint/restart or unprivileged containers. The operating systems for the containers intend to provide the latest kernel with minimal package extensions. The cluster management has a crucial role in controlling large number of containers for diverse application. Service discovery, job execution, resource allocation and key-value store for configuration data should be addressed to ensure scalability and high availability. VM-focused experiences from IaaS continue on the containers. Running containers on current IaaS platforms provides practices of running applications without dependencies during the transition between the containers and hypervisor-based virtualization. The coordination service is required that each container application communicates with others to exchange data, configuration information, host information, and ip addresses via network. In a cluster environment, access information and membership information need to be gathered and managed in a quorum. Zookeeper, etcd, consul, and doozer fall into this category. DevOps tools with their configuration scripts e.g. Chef Cookbooks or Ansible playbooks are useful to maintain and construct clusters with necessary software packages on target cluster nodes, for example, installing and configuring Mesos with container runtime environments.

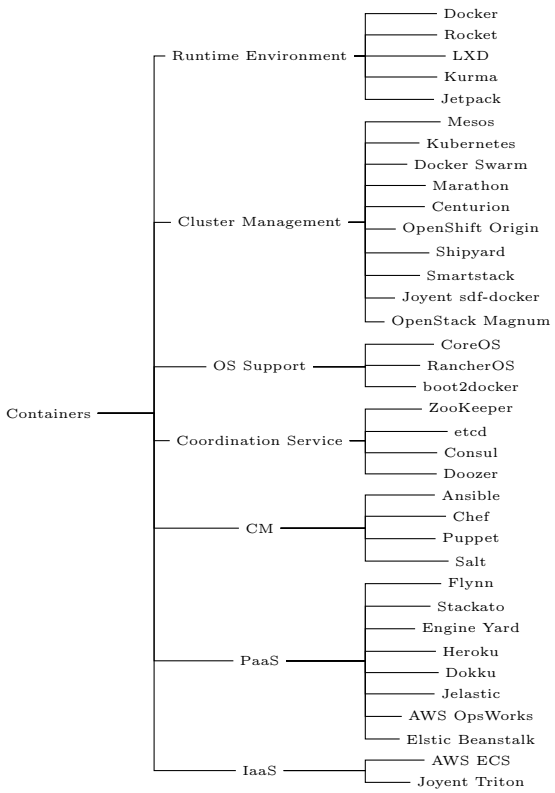

Figure 1: Mind Map of Containers

4. PAAS

One of the goals in Platform-as-a-Service (PaaS) was to ease deploying and managing a set of applications as a single platform since an application runs with other applications to deal with dependencies. With the recent improvement of the container virtualization, the benefits of microservices architecture such as increasing utilization and simplifying deployment can be enabled in PaaS compared to running on IaaS or bare-metal. There are many examples in this effort, for example, OpenShift Origin (also known as OpenShift 3) provides application development and deployment with the combination of Docker containers and Google's Kubernetes. HP's Stackato (previously acquired by ActiveState) aims to provide PaaS with Cloud Foundry and Docker. Cloud Foundry Diego is a PaaS service to "combine a scheduler, runner and health manager" withDocker. NewRelic Centurion, Deis, Flynn, and Dokku are other efforts in PaaS with containers.

5. REFERENCES

[1] The current state of container usage.

[2] State of container survey 2015 - docker adoption survey.

[3] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. An updated performance comparison of virtual machines and linux containers. *technology*, 28:32, 2014.

[4] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose. Performance evaluation of container-based virtualization for high performance computing environments. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 233–240. IEEE, 2013.

Table 1: Cluster Capabilities for Containers

Name	Job Management	Image Management	Configuration Service	Languague	What can be specified (extent of ontologies)
Apache Mesos	Chronos, two-level scheduling	SchedulerDriver with ContainerInfo	ZooKeeper	C++	Hadoop, Spark, Kafka, Elastic Search
Google Kubernetes	kube-scheduler: Scheduling units (pods) with location affinity	Image Policy with Docker	SkyDNS	Go	Web applications
Docker Swarm	Swarm Filters and Strategies	Node agent	Etcd, Consul ZooKeeper	Go	Dokku, Docker, Compose, Krane, Jenkins
Apache YARN	Resource Manager	Docker Container Executor	YARN Service Registry, ZooKeeper	Java	Hadoop, MapReduce2 Tez, Impala
Apache Myriad	Marathon, Chronos, Myriad Executor	SchedulerDriver with ContainerInfo, dcos	ZooKeeper	Java	Spark, Cassandra, Storm, Hive, Pig, Impala, Drill
Engine Yard Deis	CoreOS Fleet	Docker	etcd	Python, Go	Application from Heroku Buildpacks, Dockerfiles, Docker Images