Live Container Migration: Opportunities and Challenges

Niroj Pokhrel
Aalto University

Abstract—The usage of container for hosting applications is getting popular due to its inherent capability to isolate resources among different containers within the same kernel. With growing popularity of container technology, demand for containers to be more available and scalable is increasing. Solution to this problem can be swift migration of a container from one machine to another without obstructing operations of the applications. This paper reviews and explores opportunities and challenges of such live container migration.

I. INTRODUCTION

With the development of powerful processing capability of the servers and cloud architecture, the interest in the virtualization has increased. By the virtue of virtualization, server can be partitioned into multiple execution environments called Virtual Machine (VM) which are isolated from one another and run their own operating system. [1] However, there are several issues with the virtual machines such as portability and redundancy. [2] Portability cannot be realized properly as images of the VM are bounded to the platform. Similarly, there will be limited reuse of components and redundant number of guest Operating Systems (OS).

To overcome major shortcomings of VM, containers were developed which are used as their substitute. Containers are also called OS level virtualization as they share underlying host OS. However, early challenge in the development of container technology was to find effective way to isolate and provide security to different containers in the same machines [3]. The isolation was achieved by incremental development and usage of chroot, jail, control groups (cgroups) and namespaces. Namespaces helps in creating an isolated container preventing accessibility to objects outside the container. The concept of container itself was based on namespace feature of kernel which was designed for handling high performance clusters. Clone() system call access this feature to create separate instance of previous global namespaces. Some of the namespaces used by Linux systems are network, PID, filesystem, IPC, hostname and user. A filesystem namespace has own root directory and mount table making it more powerful than chroot(). [1] The concept of Linux container itself was based on namespaces.

Cgroups or Linux Control Groups is used to group processes and control their resource consumption. The cgroups can be used to limit CPU and memory consumption of containers and, thus facilitate easy resizing of the container just by changing the limit of the corresponding cgroup. It can also be used as way of terminating processes inside container. [1] In other words, cgroups helps in performance isolation by making the resources of a container independent of the requirement of other containers.

Another important feature of Docker container is layered filesystem images powered by Advanced multi-layered Unification FileSystem (AUFS) but it is not present in most of other container tools. In this feature, new image layer is created and added to image for every instruction in Dockerfile. The image created will become a parent image for any successor and added to image for every instruction in Dockerfile. The image created will become a parent image for any successor images of the VM.

II. BACKGROUND

Understanding basic functioning of containers is helpful to better understand the process of migration. This section discusses various features of container such as chroot, jail, control groups (cgroups) and namespaces. Namespaces helps in creating an isolated container preventing accessibility to objects outside the container. The concept of container itself was based on namespace feature of kernel which was designed for handling high performance clusters. Clone() system call access this feature to create separate instance of previous global namespaces. Some of the namespaces used by Linux systems are network, PID, filesystem, IPC, hostname and user. A filesystem namespace has own root directory and mount table making it more powerful than chroot(). [1] The concept of Linux container itself was based on namespaces.

Cgroups or Linux Control Groups is used to group processes and control their resource consumption. The cgroups can be used to limit CPU and memory consumption of containers and, thus facilitate easy resizing of the container just by changing the limit of the corresponding cgroup. It can also be used as way of terminating processes inside container. [1] In other words, cgroups helps in performance isolation by making the resources of a container independent of the requirement of other containers.

Another important feature of Docker container is layered filesystem images powered by Advanced multi-layered Unification FileSystem (AUFS) but it is not present in most of other container tools. In this feature, new image layer is created and added to image for every instruction in Dockerfile. The image created will become a parent image for any successor instructions. The docker caches image layer and will be used for subsequent builds of same Dockerfile which reduces total build time. It also leads to faster deployment as only a part of the image needs to be copied over the cloud. [4]
III. Migration

Migration is a process of moving a container from one server to another server. Migration can facilitate in providing fault tolerance as container or VM can be migrated to another host if the system experience failures. It can also serve for balancing load, tackling hardware failures, scaling and reallocating resources. The process of migration either of VM or container consists of mainly three classes: memory migration, process migration and disk migration. The expectation of migration mechanism is to have a zero downtime. For all the applications running in the container and the container itself, it should appear that container is in the same location even during migration except for a slightly reduced performance during migration and improvement afterwards. This section discusses different migration and replication mechanism. Ideally during migration complete state of original container should be transferred including disk, memory and network connections. [5]

A. Memory migration

Memory migration can be divided into two types: precopy and postcopy. In postcopy, a container transfers memory after the processor state is sent to the target location whereas in precopy migration mechanism memory is transferred repetitively first and processor state afterwards to the target. Steps of postcopy migration is given below

- Stop container at the source
- Sent processor state, registers state and devices states to destination
- Resume destination container with no memory
- In scenario where container tries to access pages not transferred yet, container is stopped and fault page is demanded over the network

The process of migration is summarized in Figure 1. [6]

![Postcopy memory migration](Fig. 1. Postcopy memory migration)

Similarly, precopy follows similar steps with the difference in time of transferring memory. Steps of precopy migration is summarized below

- Container at source continues to run while memory pages are getting copied to the destination
- Copying is repetitive but the subsequent steps only copy pages modified during last transfer
- Container at source is stopped then cpu state is copied
- Destination container is started

The steps of precopy can be summarized in Figure 2. [6]

![Precopy memory migration](Fig. 2. Precopy memory migration)

The comparison between two different migration method is tabulated below.

<table>
<thead>
<tr>
<th>Destination Node Failure</th>
<th>precopy</th>
<th>postcopy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtime</td>
<td>More</td>
<td>Less</td>
</tr>
<tr>
<td>Up state after migration</td>
<td>Fast</td>
<td>Slow</td>
</tr>
<tr>
<td>Write Intensive application</td>
<td>Worse</td>
<td>Nodifference</td>
</tr>
<tr>
<td>Read Intensive application</td>
<td>Good</td>
<td>Nodifference</td>
</tr>
</tbody>
</table>

B. Network Migration

Network connectivity should be maintained after migration by preserving open connections. When migrating within the same LAN, original IP address should be retained even after migration. Nonetheless, an unsolicited ARP reply is generated to advertise location of destination. However, if the migration is in WAN, technologies such as Virtual Private Networks(VPNs), tunneling and DNS servers can be used.

C. Suspend/Resume Migration

Suspend/Resume migration technology is based on providing user mobility in a secure way. In this process, container or VM is migrated to a destination host where it is inactive during transfer. Main points to undertake for suspend resume migration is listed below.

- Network connections dropped and reestablished at the destination host
- Processor state, registers state and devices states sent to destination
- Images, local persistent state, ongoing network connections are transferred and Support for disconnected operation is provided
- Apply delta disk operations for optimizing disk transfer

Disk migration can be optimized by using the concept of deltas. In this process, write operations in the sources are intercepted and deltas are generated. Deltas are the communication units containing written data, size of the data and location on the disk. First step of the process is examining the stored data and locating blocks which have changed since last write. The changed data is then sent to the destination through WAN or LAN.

Another important feature of Suspend/Resume operation is disconnected operations. In the disconnected operation, a client can access critical data during temporary failures of data repository through the use of contents of the cache. The modifications in the cache can be transferred when disconnection ends.
D. Record/Replay Migration

Record/Replay migration is usually used for recovering state. Steps of Record/Replay migration is listed below.

- Find checkpoint of last state
- Repeat the events from log to get to the desired state

Events can be classified into deterministic or non-deterministic. For containers and VM, replaying needs logging of non-deterministic events which affects the computation. Deterministic events are the regular events such as arithmetic, memory and branch instructions and the outcome of such events can be deterministic. Non-deterministic events such as interrupts, input from devices such as keyboard, mouse, network and clock outcome cannot be determined when the process is repeated. Non-deterministic events can be classified into two categories as external input and time. Time events is the exact point during execution when the event occurs. External input is data from other devices or human beings. For replaying a container or VM, non-deterministic events that affect the computation are required to be logged. Deterministic events are not logged and can be computed during replay. Replaying the non-deterministic events from the log and computing deterministic events can get container to the desired state.

However, Record/Replay method should try to minimize challenges such as maximizing trace completeness, reducing log file size and providing low performance overhead.

IV. CASE STUDY: CHECKPOINTING AND RESTORING IN OPENVZ

OpenVZ uses checkpointing and restoring as a mechanism for live migration. During checkpointing, state of running container is saved or checkpointed and is restored later on same or different system. Whole process of checkpointing and restoring is transparent for the applications and network connections. Container has capability to reboot independently, have Ip addresses, users, root accesses, memory, processes and filesystem. [5] A container can be regarded as isolated entity, all inter process communications and parent child relationships are within container boundary, thus, allowing a complete state to be saved in a disk file. This process of saving complete state of container is known as checkpointing. The saved disk file can be used to restart a container. Entire process has been summarized in Figure 3.

First step of checkpointing and last step of restarting is process freeze. It is done to keep the process consistent and also reconstructing a freezeed process is easier. Freezing is done by sending TIF_FREEZE signal on all processes threads.

All the dependencies should be saved such as identifiers, process hierarchy and shared resources such as opened files and shared objects. These state should be restored while restarting in the destination host. Network should be disabled but preserved. Network can be disabled by dropping all incoming packets. Resources is restored from process context and to facilitate that a special functions called hook is added on top of each process stack during restarting. The process will then first run hook on restart and restore all of its resources. In the same time, for init process of container, the hook restores networking state such as interfaces, iptables and route tables, IPC objects and mount points and initiates process tree construction. [5]

Fig. 3. Steps of Checkpointing and Restoring

V. IMPLEMENTATION IN LINUX

Checkpointing and Restoring is done using open source project CRIU and P.Haul in OpenVZ. [7] [8] CRIU is low level abstraction that takes care of saving and restoring state. Similarly, it performs memory precopy and postcopy. On the other hand, P.Haul is an implementation on top of CRIU which orchestrate all checkpointing and restoring steps and deals with filesystem.

VI. CONCLUSION

Live Migration is essential for high availability and load balancing. There are several migration methods but each one of them has their own advantages and disadvantages. Particular migration technique can better perform for certain type of application. Thus, if a system is aware of type of application it is running, the best possible migration mechanism can be used. The extensively researched migration class is memory migration which has two popular methods precopy and postcopy. These techniques are already implemented in tools already available the market and can be easily implemented in linux with CRIU and P.Haul. OpenVZ and docker are examples of containers which have implemented live migration with CRIU and P.Haul.

REFERENCES