CMPSCI 677 Distributed and Operating Systems **Spring 2022**

Lecture 10: Feb 28

Lecturer: Prashant Shenoy *Scribe:* ***Shiksha Rawat, Bharath Narasimhan***

10.1 Virtual Machine Migrations

Process and code migration in heterogeneous systems, i.e., systems in which the destination architecture is different from source architecture, pose a challenge. Migration via interpreted code is one possibility but it is often clumsy in practice and is almost never used. Moreover, interpreted code migration only supports weak mobility. In such cases, it is beneficial to look at techniques for VM migration.

VMs can be migrated from one machine to another irrespective of architectural differences. A VM consists of its OS and some applications running on this OS. So, in VM migration the OS and these applications are migrated with negligible down time. As the processes inside a VM will also move, VM migration also involves process migration. VM migration is usually done live, that is, it keeps executing during migration. Applications continue to run, nothing has gone down, and then after a while the VM disappears from one machine and shows up on another machine with the applications still continuing to run.

There are two methods for VM migration:

Pre-copy Migration: The process of pre-copy migration involves the following steps:

1. Enable dirty page tracking. This is required to keep a track of pages which have been written to.
2. Copy all memory pages to destination.
3. Copy memory pages which were changed during the previous copy.
4. Repeat step 2 until the number of memory pages is small.
5. Stop VM, copy rest of memory pages at destination and start VM at the destination.
6. Send ARP packet to switch

Post-copy Migration: This is also called lazy copy. The process of post-copy migration involves the following steps:

1. Stop VM and move non-memory VM states to destination
2. Start executing on new machine
3. In case of page faults in the new VM, copy the page from the source machine. Copying of other pages is also started in the background. Background copying also ensures that every page is copied even if it was required during a page fault before the VM is deleted from source.

One advantage of post-copy over pre-copy is that there is no iterative copying and each page has to be fetched only once. Pre-copying is preferred in some cases when we do not want applications to keep waiting in case of page faults so there is less impact on application performance.

Question: How much copying is enough to restart VM at the destination?
Answer: Moving at least all the registers, a few OS pages that program counter is pointing to, should be good enough to restart the VM. More details depend on the hardware architecture.

Question: Programs often have spatial and temporal locality of references, can we use it to intelligently figure out what to pre-fetch?
Answer: Post-copy migration can make use of such optimizations where for example, one can get the working set of the programs first and then fetch the rest.

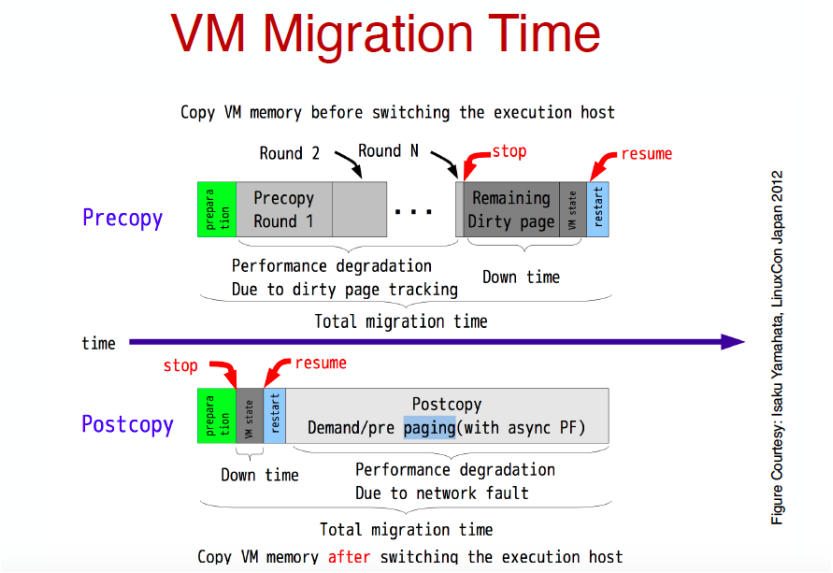

Figure 10.1: Visualization and comparison of pre-copy and post-copy VM Migration.

10.2 Types of Migration

There are three types of migration:

Cold Migration

1. VM is not running. There is just an image of VM/container or virtual image of VM/container on the disk.
2. Copy this image and data files to new machine.
3. Start on new machine.
4. No state preserved.

Warm Migration

Incurs downtime to migrate state from previous instance, but preserves the state.

1. Suspend running VM/container to disk.
2. Copy image, data, suspended memory state.
3. Resume execution of suspended VM.

Hot/Live Migration

Migrate state but with no downtime so copy state while VM executes. Most complex to execute.

10.3 Container Migration

Containers are light weight VMs. When you are migrating a container you are migrating only the processes and some resources that it accesses. The underlying OS is not getting migrated.

10.4 Snapshots

A snapshot is a copy of some object (file, disk, VM, container) at a certain point in time (point-in-time copy). We can preserve the contents of the VM(the memory and disk content) as they existed at that point, if one takes a snapshot of a VM at a certain instant. Snapshots are also known as checkpoints. Snapshots help in rolling back to a point in time and creating backups.

There are two ways to create a snapshot.

1. Full (Real) Snapshots - Actually make a real copy. Very inefficient.
2. Virtual Snapshots - Here a virtual copy is created. Instead of making a real second copy of a file we just copied the metadata and all of that is pointing to the previous copy. The previous copy can continue to change because the VM can write on the disk. So, in case of virtual snapshots, copy-on-write is used. Whenever a VM is about to write to a previous copy, a new copy is created first and pointers are shifted to this new copy. Virtual snapshots are very efficient as compared to full snapshots.

Question: Snapshots are useful for migration. Is it only valid for real snapshot real copies as opposed to a virtual copy?
Answer: Even in a virtual copy you have access to all of the data because it's just pointing to the same blocks. So, it will not matter whether you took a virtual snapshot or a real copy for migration purposes.

10.5 CheckPoint and Restore

This is a warm container migration technique. Many containers actually support checkpoint and restore as the first option because live migration is a bit more complicated. Migration in containers is a little more complicated than VMs as in containers we only migrate processes. Steps of CheckPoint and Restore are :

1. Pause container execution.
2. Checkpoint (save) memory contents of container to disk.
3. Copy checkpoint to new machine (memory + disk image).
4. Resume execution on new machine.

Question: Is a checkpoint the same as a snapshot?
Answer: Checkpoints are like real snapshots.

10.6 Linux CRIU

1. Linux CRIU (Checkpoint Restore In User Space) : It is used for warm or live migrations, snapshots and debugging. CRIU is not a container-specific technique, it is a process-specific technique. As a container is a collection of processes, it suspends all of the processes in the container and writes it out on the disk one by one.
2. CRIU uses `/proc` file system to gather all info about each process in the container. In the proc file linux keeps its OS data structures. It's like a file system—you can go and look at the files. There's information about every active process in the system.
3. CRIU copies saved state to another machine.
4. CRIU restorer
   (a) Use fork to recreate processes to be restored
   (b) Restorer also restores the resources being used by processes; for container, restores namespace
   (c) If any network connections were being used, we have to do extra work if we have migrated the container to a new machine. We can migrate active sockets only if the IP address moves along with the container to the new machine. To do so, we can use virtual network devices in containers and move them.

Question: Why can't the process bring the IP address of the previous machine?
Answer: We can not do this because the socket connection is always tied to the IP address on which socket was established.

Question: If we assign a new IP address to the container, how to make sure it is unique?
Answer: We can have either a public IP or a private IP. If it's a public IP by definition it has to be unique. Many containers will not have a second public IP. They'll give themselves a 192 address which is a private IP address. So when you move a container over a new machine, to make this private IP work, the new machine should not have the same IP address running on it already.

10.7 Kubernetes (k8s)

Container orchestration is a form of cluster scheduling but rather than scheduling jobs or http requests you're scheduling containers. There is a pool of machines and applications are coming as containers. The goal of the container manager is to now assign this container onto some physical host. When the application is done, the container is terminated. The scheduler does not care about what is inside the container, it just schedules the container on the basis of its resources requirements.

Kubernetes is one of the most popular container orchestration systems. It is based on Google's Borg/Omega cluster managers. In Kubernetes, it is assumed that all applications are containerized. K8s will deploy them onto machines of the cluster. Kubernetes provides the following features:

1. Replication of apps on multiple machines if requested (fault tolerance)
2. Load balance across replicas
3. Can scale up or down dynamically (vary replica pool size, a concept similar to dynamic thread/process pools)
4. Provide automated restart upon detecting failure (self-healing)

10.8 K8s Pods

Kubernetes has the concept of pods. A *pod* is an abstraction where we can have more than one container in it. Containers inside pods share volumes and namespace. Kubernetes doesn't directly deal with containers, it deals with pods. So, pods are the smallest granularity of allocation in k8s.

In a distributed application, because your application has multiple components in the kubernetes world each component has to be containerized first. So, each pod consists of one or more components/containers.

Pod can contain all containers of an application but if a component needs to be scaled, put that component in a separate pod. As a good design principle, each independently scalable component should be put in a different pod. Constructing applications in pod is the job of an application developer. Deploying and scaling it is the responsibility of kubernetes. Pods of an application can span multiple cluster machines.

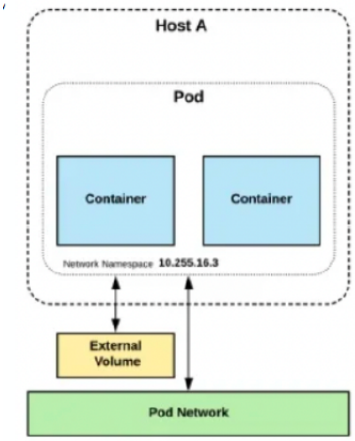

Figure 10.2: Visualization of the relationship between containers, pods, and hosts.

10.9 K8s Services

Pods are used to construct kubernetes services. A service is a method to access a pod's exposed interfaces. Features of services include:

1. Static cluster IP address
2. Static DNS name
3. Services are not ephemeral
4. Collection of pods

Pods are ephemeral. Each has its own IP. They can be migrated to another machine. Pods can communicate with one another using IP.