Software Engineering

- network virtualization & namespaces and cgroups in Linux -

Professor Han-gyoo Kim

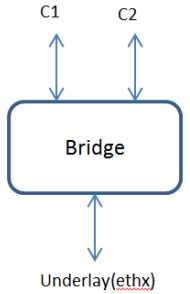
2022

Virtualization

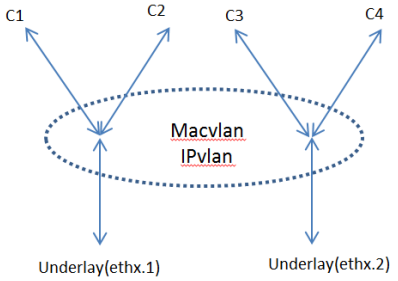
- OS = 하드웨어 virtualization S/W
  - Physical 하드웨어를 사용하는 데에서 생기는 제한을 인식하고 모든 하드웨어 자원을 virtualize 할 필요 => OS가 driver를 사용하여 하드웨어를 구동하는데, driver는 소프트웨어이므로 driver가 실제 하드웨어의 자원을 접근하는 대신에 일종의 두 단계의 driver 층을 두어 하층 driver는 기존 그대로 하드웨어를 직접 접근하되 OS에게 보여주고 서비스 하는 driver(?)는 하드웨어의 일부를 독립적으로 구분하고 떼어 내거나 또는 하나의 하드웨어가 아닌 다수의 하드웨어처럼 사용할 수 있도록 제공하면 그 아니 좋을시고!
  - VM 전성시대 -> VM은 machine 즉 하드웨어를 virtualize 한 것
  - 어떤 하드웨어 부품들을 어떻게 virtualize 하면 좋을까?
    - CPU, DRAM, Storage : 실제 하드웨어의 자원을 나누어 서로 영향 주지 않게 독립적으로 사용 -> virtualization이 간단
    - Network interfaces : BW는 나눌 수 밖에 없지만 하나의 하드웨어 인터페이스를 여러 개 있는 것처럼 제공하고 원래의 하드웨어가 제공하는 해당 network 계층의 기능 외에 상위 계층의 기능도 제공 -> 다수의 하드웨어 장치를 나누어 쓰는 것이 아니라 하나의 하드웨어를 다수인 것처럼 사용하고 경우에 따라 상위 계층 기능을 부여하는 것은 보다 복잡함
- => network virtualization도 필요 -> 양은 많지만 간편(?) -> 기존 IP 프로토콜 계층 그대로 사용하며, 호스트 기계 뿐만 아니라 하드웨어 스위치, 라우터도 virtualize 하여 하드웨어 링크/장치를 그대로 사용하되 하드웨어 네트워크 위에 virtual network를 소프트웨어로 구현 <- 아주 저렴한 스위치 외에는 모든 network 장치들은 예외 없이 Linux 기계이므로 네트워크 virtualization 용이
- 왜 virtualization?
  - 돈 때문에? 그보다도 별도의 하드웨어 추가 없이 서로 완전히 독립적으로 사용할 수 있는 환경이 필요해서!

VLAN(macvlan과 ipvlan) in Linux

- Underlay network drivers in Linux that provides base for network virtualization = ethernet interface와 같은 하드웨어에 직접 접근하는 네트워크 가상화 드라이버
  - bridge – virtualized L2 switch



  - macvlan / ipvlan
    - MAC 계층 / IP 계층 virtualized driver
    - 하나의 physical network interface에 다수의 MAC 주소 배정
    - 하나의 MAC 주소에 다수의 IP 주소 배정



- VXLAN, NVGRE
  - 위 underlay 네트워크와 container/VM 사이를 연결하는 overlay 가상화 드라이버

Docker networks

- docker-compose up -d
- docker network ls
- Docker networks = networks provided by docker for communication among containers
  = **network drivers implemented in Linux**
  1) bridge – default – virtual L2 switch - each container has its own private IP address
  2) host – host의 네트워크와 동일한 네트워크 사용
  3) overlay – docker swarm service 네트워크 – conventional overlay network (i.e., conventional virtual network)
  4) macvlan – MAC 주소에 대한 완전한 제어 제공 – each container has its own MAC address+IP address
  5) ipvlan – IP 주소 사용에 대한 완전한 제어를 제공 – each container has its own IP address sharing MAC with others
  6) none – no network interface provided for full customization
- User-defined bridge networks are best when you need multiple containers to communicate on the same Docker host.
- Host networks are best when the network stack should not be isolated from the Docker host, but you want other aspects of the container to be isolated.
- Overlay networks are best when you need containers running on different Docker hosts to communicate, or when multiple applications work together using swarm services.
- Macvlan / ipvlan networks are best when you want your containers to look like physical hosts on your network.
- Third-party network plugins allow you to integrate Docker with specialized network stacks.

Namespaces and Cgroups in Linux

- 기본적으로 OS는 투명
  - 일반 user가 시스템의 모든 자원, 다른 사용자, PID, 파일 시스템 마운트, 메모리 등등을 모두 다 볼 수 있음
  - 독립적으로 서로 구분된 시스템 view가 필요한 경우에는 부적절
- Linux kernel 은 시스템 환경을 가상화 하는 다양한 lightweight tool들을 구성하는 cgroups 와 namespaces를 제공
- Docker가 cgroups 과 namespaces 위에 구축한 대표적인 프레임워크
- Namespaces는 process trees, network interfaces, user IDs, filesystem mounts 등 시스템 요소들을 경계를 긋고 제한하는 mechanism

다양한 namespaces in Linux (평행 우주?)

- Mount namespaces
  - 프로세스가 보는 mount 구조를 다른 프로세스들과 분리
- UTS(unix time sharing) namespaces
  - 호스트와 도메인 이름을 다른 uts 와 분리
- IPC namespaces
  - 서로 다른 프로세스 그룹 사이에서 IPC를 서로 분리
- PID namespaces
  - 임의의 프로세스가 자신이 init 프로세스처럼 PID1이 되는 process tree를 구성할 수 있도록 하여 분리
- Network namespaces
  - Network protocol services 와 interfaces 를 가상화
  - 각 network namespace 는 자신만의 가상화된 네트워크 장치로 구성된 개별 네트워크를 가짐
- User namespaces
  - 서로 다른 namespace에 속한 user들은 다른 namespace의 user들의 존재를 모름 (동명 이인)
- Cgroup namespaces
  - /proc/self/cgroup file의 내용을 가상화
  - 한 cgroup namespace에 속한 프로세스들은 자기 namespace 내의 자원에만 접근

Control groups (cgroups)

- Cgroups는 각 프로세스 그룹에 OS 자원을 배정하고 측정하는 커널 장치
- Cgroups를 사용하여 CPU time, network, memory, block io 같은 자원을 배정
- 각 cgroup마다 다른 권한을 부여할 수 있음
- namespaces에 cgroups를 적용하여 일단의 프로세스들 container로 구성하여 해당 프로세스들만의 virtual 시스템 구현
- Container는 같은 호스트 내의 다른 container의 존재에 대해 알지 못함
- Linux에 구현되어 있는 namesapce system calls (commands)
  - clone(2)
  - setns(2)
  - unshare(2) -> 7가지 namespace를 만드는 명령

- Cgroups (control groups)

  cgroups is a Linux kernel feature that limits and isolates the resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes

- Namespaces

  A namespace is a set of unique names to identify software objects of different kinds in computing

  Name reuse is common practice in computing in various hierarchy or in various classes of computing

  In Linux, namespaces are a kernel feature that allow a set of processes see a set of names of kernel resources such as IP addresses, file names, process IDs, user IDs, etc. while other set of processes see other set of names where names for a set of processes can be “same” to those for another set of processes

결론

- Namespaces 와 cgroup을 사용하여 단일 호스트 (VM 포함) 에서 서로 독립적으로 분리 운영되는 프로세스 그룹을 만드는 것 (VM을 만드는 것이 아님)
- 즉 container 구현을 위한 기본 장치
- 당연히 모든 자원을 virtualize 하여 구성해야만 가능 (physical 자원은 서로 다른 그룹에서 공유하면서 독립적으로 사용할 수 없으니까)