Networking in Containers and Container Clusters

Victor Marmol
vmarmol@google.com
Containers

Isolate and package application

“Lightweight VMs”

Isolates a machine’s

• Resources (CPU, memory, IO)
• Namespaces (PIDs, users, network)
• Filesystem
• Capabilities
Containers

Today’s focus: **networking**

UTS namespace
- Isolate hostname

Network namespace
- Network interface
- Loopback device
- Routing table
- iptable rules

net cgroups: mostly **unused** today
Containers - Docker

Docker
- Popular implementation of Linux containers
- Open source, written in Go
- Hardware and platform agnostic
- Easy management of filesystem images

libcontainer
- Sub-project of Docker
- Written in Go, with some bits in C
- Implements the container abstraction
- Today: container == libcontainer container

https://github.com/docker/docker
https://github.com/docker/libcontainer
Networking in Docker - Configuration

Hostname

Networks
- How container is **exposed** to the network
- MAC and IP address
- Gateway, MTU, queue length, ...

Routes
- Route table entries inside the namespace

```go
type Config struct {
    // Hostname optionally sets the container's
    // hostname if provided
    Hostname string

    // Networks specifies the container's network
    // setup to be created
    Networks []Network

    // Routes can be specified to create entries
    // in the route table as the container is started
    Routes []Route
}
```
Networking in Docker - Strategies

loopback
- Loopback device included in all containers
- **No external** networking

veth
- **Default** strategy in Docker
- veth pair used inside/outside container
- Attached to Docker bridge
- iptables connect to outside world
- Measured **performance hit** is significant
Networking in Docker - veth Strategy

Connections **IN**
- By default: blocked
- Unless port is exposed
- Mapping: host <-> container ports
Connections **OUT**
- To other containers: blocked
- Unless through host-exposed port
- To internet: MASQUERADE as host
Networking in Docker - Strategies

netns
  • Allows *sharing* of network namespaces
  • Can use *host namespace* for native performance

MACVLAN/VLAN
  • In the works
  • Great performance
  • IPVLAN support coming too
Networking in Docker - Future Work

Support more network strategies
  • **Pluggable** network options

Better **performing** networking options
  • MACVLAN
  • IPVLAN

Native checkpoint restore
  • Using **CRIU**
Networking in Container Clusters
Networking in Container Clusters - Kubernetes

Greek for “Helmsman”; also the root of the word “Governor”

- Container orchestrator
- Schedules and runs Docker containers
- Supports **multiple** cloud and bare-metal environments
- Inspired and informed by Google’s experiences
- **Open source**, written in **Go**

Manage **applications**, not machines

https://github.com/GoogleCloudPlatform/kubernetes
Kubernetes - Pods

**Small group** of containers

**Tightly** coupled
- Run together on same machine
- Shared resources and fate

Scheduling atom

Assigned an IP

Shared network namespace
- **Share IP** address & localhost

**Example:** data puller & web server
Pod IPs are **routable**
- Docker default is private IP

Pods can reach each other without NAT
- Even across nodes

Pods can egress traffic
- If allowed by cloud environment

**No brokering** of port numbers
Kubernetes - Services

Pods are **ephemeral** they come and go
- Incorrect to talk to one of them directly

Services are groups of pods that **act as one**
- Like a group of pods in front of a load-balancer

 Gets a **stable** virtual IP and port
Kubernetes - Services

Client

10.0.0.1 : 9376

TCP / UDP

iptables DNAT

kube-proxy

apiserver

watch

10.240.1.1 : 8080
10.240.2.2 : 8080
10.240.3.3 : 8080

TCP / UDP
Environment variables
• Exposed inside pod containers
• Difficult to scale

NEW: Internal cluster DNS
• Service IPs are stable

KUBERNETES_RO_SERVICE=10.0.0.1
MONITORING_SERVICE=10.0.0.10
FOO_SERVICE=10.0.0.11
BAR_SERVICE=10.0.0.4

kubernetes_ro
monitoring
foo
bar
Kubernetes - Configurations

Andromeda
- Google’s SDN
- Program underlying network fabric

Flannel
- Overlay network
- UDP packet encapsulation

Others
- OVS-based
- More overlays
Kubernetes - Future Work

Resource management
• Cap network at node
• Cap cluster flows

Migratable IPs
• Enable container migration

Real load balancing
• Cluster-wide
• Use utilization and pod health