Quark Container

- Secure Container Runtime

--Next Generation Container Infrastructure

Quark - KVM based Secure Container from scratch

Secure + High Performance + Low Cost + Kubernetes Compatible

Secure Container

- Hypervisor based security isolation
- Dedicate OS Kernel per Container
- Design for Cloud native workload
- Low memory footprint
- Fast start up
- High performance
- Support OCI interface

Hypervisor virtual network

- Design for Cloud native workload
- Network virtualization in hypervisor
- Http stack in hypervisor
- High performance
- Interoperate with VM network and container network
- Support CNI interface

Container Orchestration

- Extension of Kubernetes
- Multi-tenant support
- Schedule both VM and Container
- Compatible operation interface as Kubernetes

Existing Secure Container

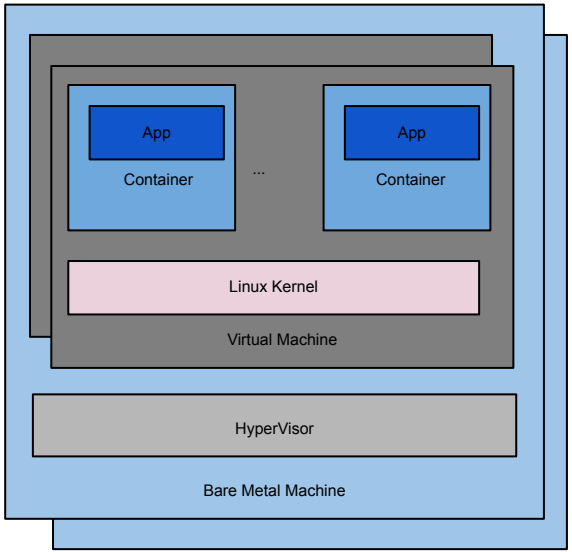
1. Open Source
2. Acquired by Ant Finance
3. Hypervisor: Optimized Qumu
4. OS Kernel: Optimized Linux Kernel

1. From Google
2. User for GCP Cloud function
3. Hypervisor: reimplementation
4. OS Kernel: reimplementation
5. Use golang and legacy design from ptrace

1. Firecracker: From AWS
2. Used for AWS Lambda
3. Hypervisor: Firecracker
4. OS Kernel: Normal Linux Kernel

	Quark Container	Kata	gVisor	Firecracker
Memory Overhead	12 MB	180MB	20 ~ 30 MB	5MB Hypervisor + 100+MB Linux Kernel
Optimization potential	Very High	Low	High	Medium
Kubernetes compatible	Yes	Yes	Yes	No
Performance	High	Low	Medium to High	Medium

Pain Point: Containerized Application on VM

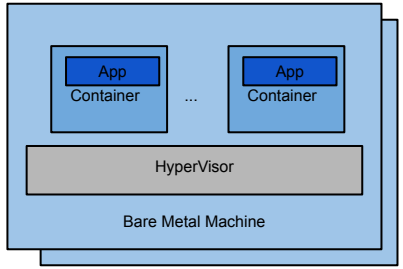


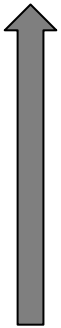
1. Docker container is not secure: VM isolation is must in multi-tenant environment
2. Virtual machine overhead
    a. Memory: 100MB ~ 300MB memory overhead per VM
    b. File IO: 10~20% throughput decrease
    c. Network IO: VM network + container network leads to 5~10% throughput decrease
3. Management: VM management (e.g. OpenStack) + Container orchestration (E.g. K8s)
    a. Extra management layer overhead
    b. Decrease resource utilization
    c. Increase IT operation cost
4. Can't meet cloud native resource management model
    a. Autoscale
    b. Service migration
    c. Cloud function

Container Infrastructure

Container Infrastructure V2: Security Container Central

1. High computation density: 2MB overload per container
2. Low latency auto scale: 50 ~ 100ms boot up overhead
3. Low overhead container runtime: 5% ~ 10%
    a. Network
    b. IO
    c. Computation
4. High security





Container Infrastructure V1: Virtual Machine Central

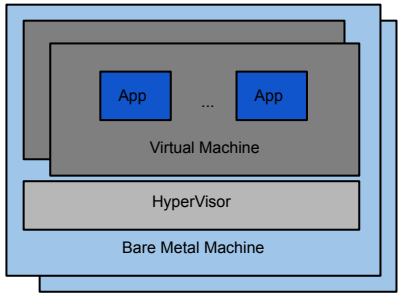
1. Security/Resource isolation: VM based isolation
2. Network: VM based VPC
3. Container: Running over VM

Not suitable for Cloud native application

1. High overhead
    a. Memory
    b. IO
    c. Network
2. Difficult to support auto scale

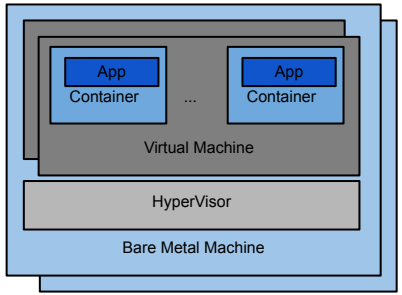
Virtual Machine

Easy to migrate from Legacy workload



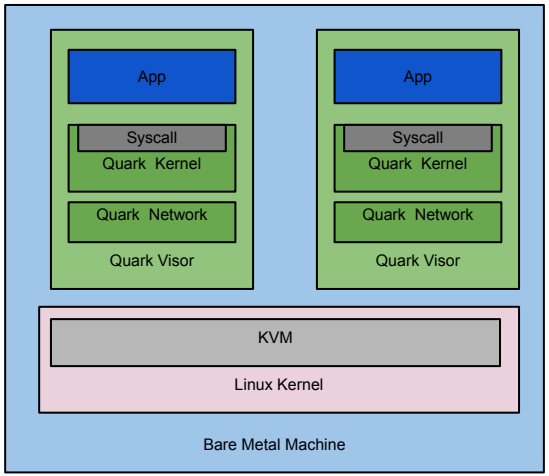
Virtual Machine + Container

Running containerized workload on existing infrastructure



Hypervisor OS

-- Reimplement OS hosted on Linux Hypervisor for containerized workload



Quark Kernel

- Reimplemented Linux compatible OS running in Hypervisor
- Running as Linux application
- No need to support physical device

Quark Network

- Underlay TCP network stack running as Linux application
- Virtual Private Network
- High performance and low latency

Quark Visor

- HyperVisor only support Quark Kernel
- Optimized for Linux Syscall interface
- High secure, Low cost and High performance
- Kubernetes Container interface (OCI)

Hypervisor OS

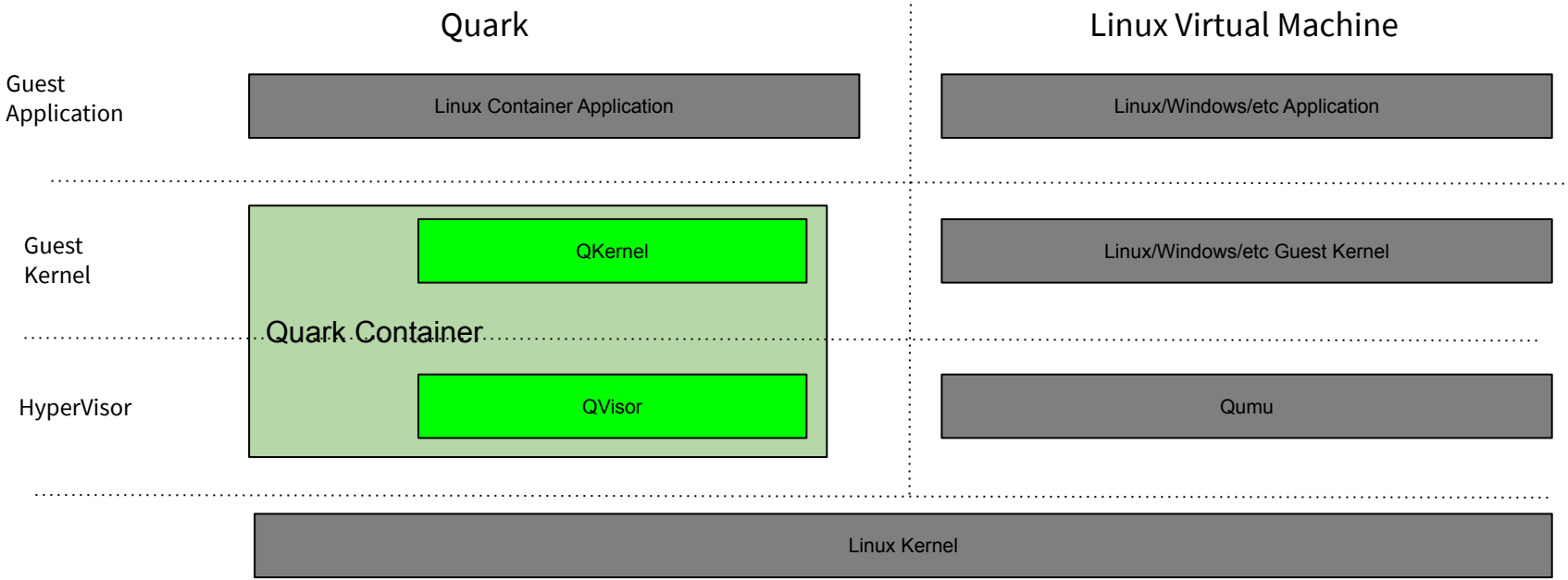
1. Designed for running as Linux application
2. Design for containerized workload: Limited System Call
3. High performance, Low cost
4. Embedded Container interface

VS

Linux VM

1. Design running all kind of hardware
2. Design for all kind of workload, complex full system call
3. High cost to run on hypervisor
4. Extra layer for Container support

Quark Architecture



Quark Architecture