OS Virtualization

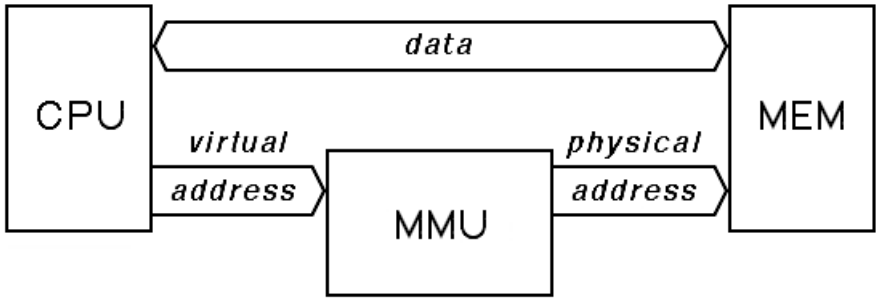
Bogdan Purcăreață
Software Engineer, Virtualization Team

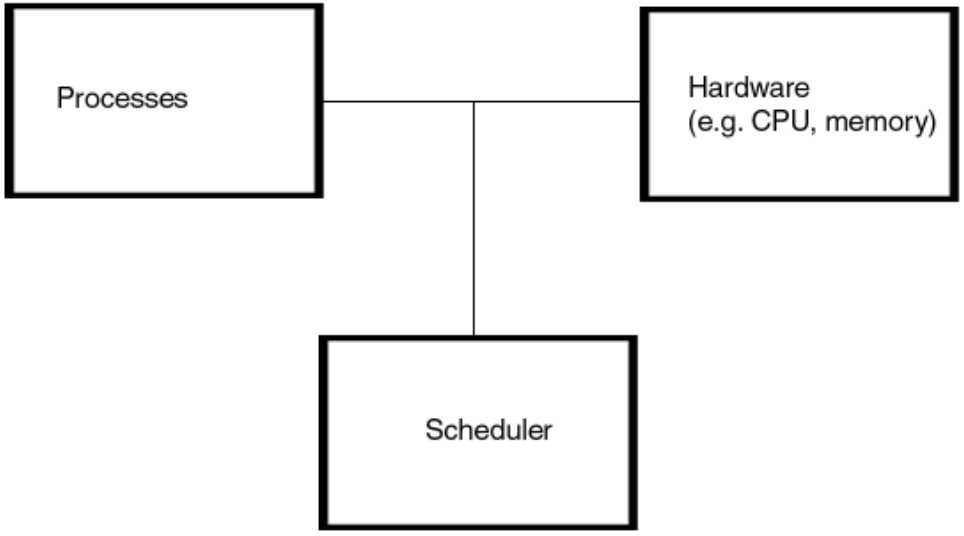
June 2013

Table of Contents

OS Recap

- Resources
  - CPU
  - Memory
  - Peripherals
- Structures
  - The scheduler
  - The MMU subsystem
  - Filesystems
- The kernel
  - Handles hardware
  - Exposes capabilities
  - Manages resources





OS-level Virtualization

- One host
- Multiple running OS instances
- Rootfs, system libs, binaries

OS instance = a process hierarchy

OS level virtualization = **partitioning the process tree**

Advantage: **close to 0% performance overhead**

Flaw: **shared kernel**

OS Virtualization

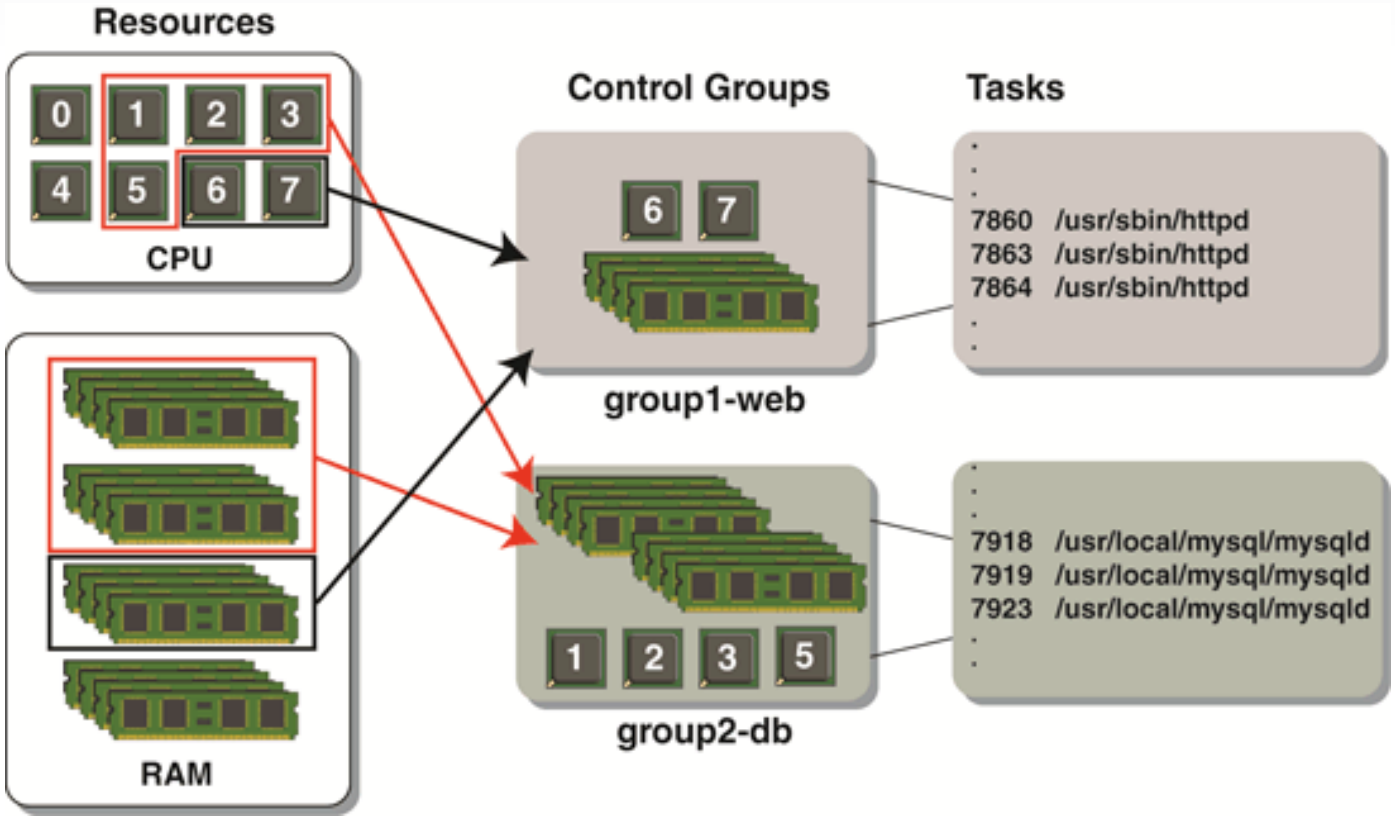
Kernel Features

June 2013

Control Groups

- Resource management among processes
- Hierarchical support
- Interaction with resource responsible structures:
  - Scheduler
  - MMU
- Memory, CPU, devices, etc

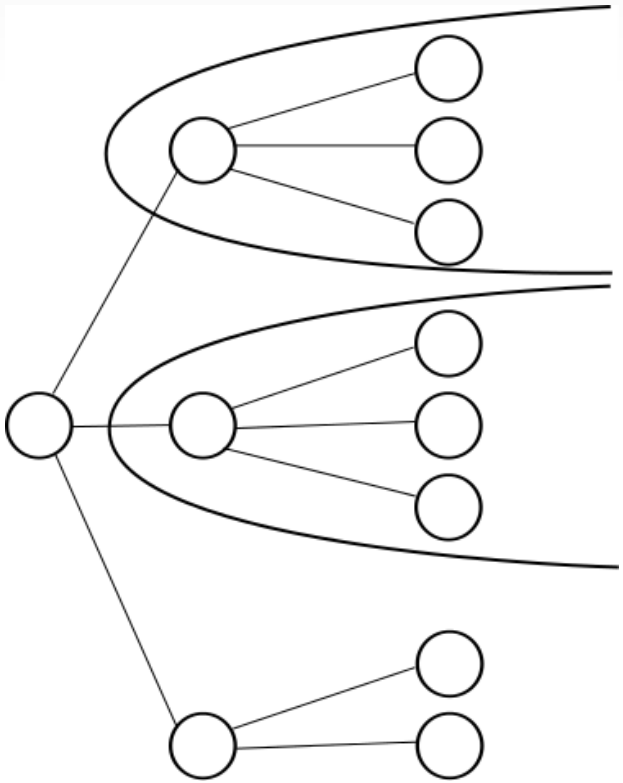
Interaction



picture from http://www.oracle.com/ocom/groups/public/@otn/documents/digitalasset/1506615.gif

Namespaces

- Abstract resources
- Processes see the resource as their own
- Isolation between namespaces
- PID, network, user, etc.



OS Virtualization

LXC

June 2013

LinuX Containers Overview

- a.k.a. LXC:
  - Mature technology implementation
  - Mainline kernel support
  - Application vs. System
  - Active development
- Components:
  - *Kernel features*
  - *Userspace tools*
  - *Configuration files*
  - *Template files*

Sample Process Hierarchy

```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)
        |-lxc-start(2964)---init(2966)----+-init(2972)
        |                                 |-sh(2971)
        |                                 `-syslogd(2969)
        |
        |
        |-lxc-start(2974)---init(2976)----+-init(2982)
        |                                 |-sh(2981)
        |                                 `-syslogd(2979)
        |
        |
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

Process IDs

```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)
        |-lxc-start(2964)---init(2966)(1)-+-init(2972)(7)
        |                                 |-sh(2971)(6)
        |                                 `-syslogd(2969)(4)
        |
        |
        |-lxc-start(2974)---init(2976)(1)-+-init(2982)(7)
        |                                 |-sh(2981)(6)
        |                                 `-syslogd(2979)(4)
        |
        |
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

Namespace Segregation

```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)
        |-lxc-start(2964)---init(2966)(1)-+-init(2972)(7)
        |                                 |-sh(2971)(6)
        |                                 `-syslogd(2969)(4)
        |                    PID Namespace 1
        |
        |-lxc-start(2974)---init(2976)(1)-+-init(2982)(7)
        |                                 |-sh(2981)(6)
        |                                 `-syslogd(2979)(4)
        |                    PID Namespace 2
        |
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

Filesystem Segregation ("chroot on steroids")

```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)
        |-lxc-start(2964)---init(2966)(1)-+-init(2972)(7)
        |                                 |-sh(2971)(6)
        |                                 `-syslogd(2969)(4)
        |
        |-lxc-start(2974)---init(2976)(1)-+-init(2982)(7)
        |                                 |-sh(2981)(6)
        |                                 `-syslogd(2979)(4)
        |
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

root:/var/lib/lxc/foo1/rootfs

PID Namespace 1

root:/var/lib/lxc/foo2/rootfs

PID Namespace 2

CPU Partitioning

```
init(1)-+-dnsmasq(2162)
        |-klogd(2175)
    ,---|-lxc-start(2964)---init(2966)(1)-+-init(2972)(7)
   |    |                                  |-sh(2971)(6)
   |    |                                  `-syslogd(2969)(4)
   |    |
   |    |
 1 core |-lxc-start(2974)---init(2976)(1)-+-init(2982)(7)
   |    |                                  |-sh(2981)(6)
   |    |                                  `-syslogd(2979)(4)
   |    |
    `---|-----------------------------------------------------
        |-netserver(2167)
        |-sh(2179)
        |-syslogd(2173)
        `-udevd(962)-+-udevd(1189)
                     `-udevd(1190)
```

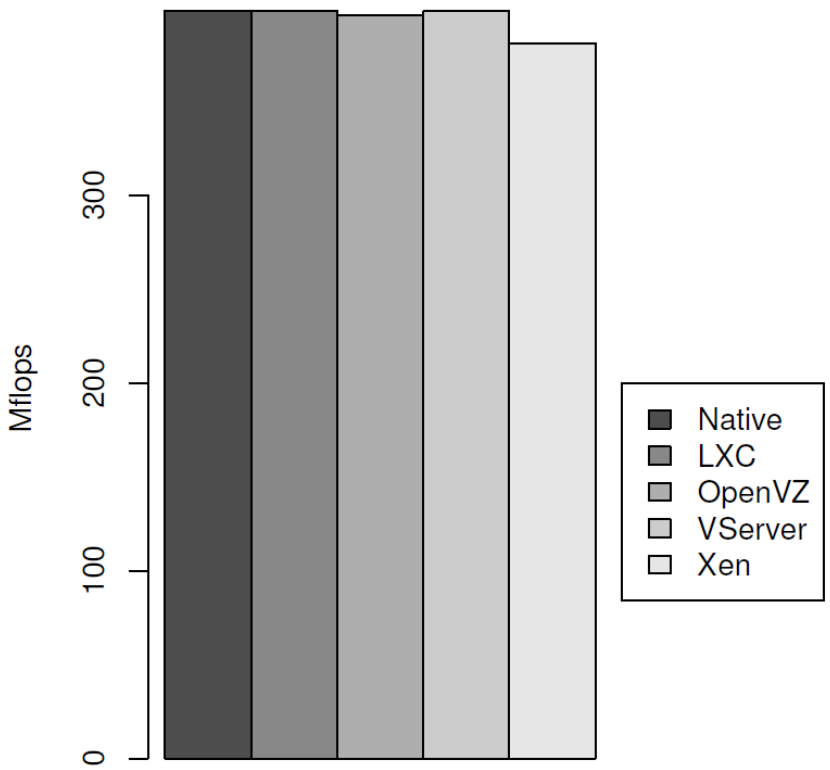
root:/var/lib/lxc/foo1/rootfs

PID Namespace 1

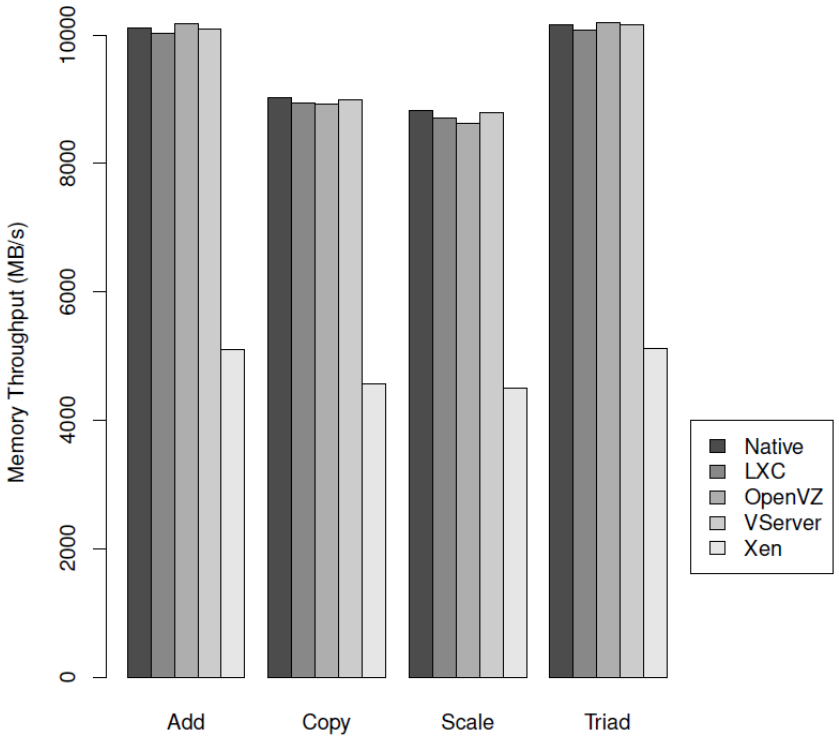
root:/var/lib/lxc/foo2/rootfs

PID Namespace 2

System Performance

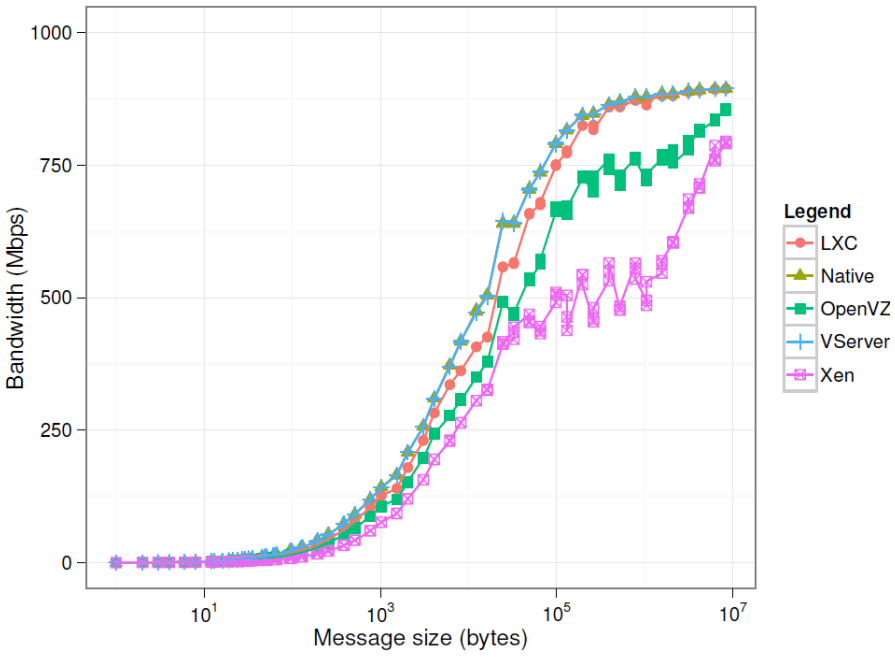


CPU Performance
Linpack

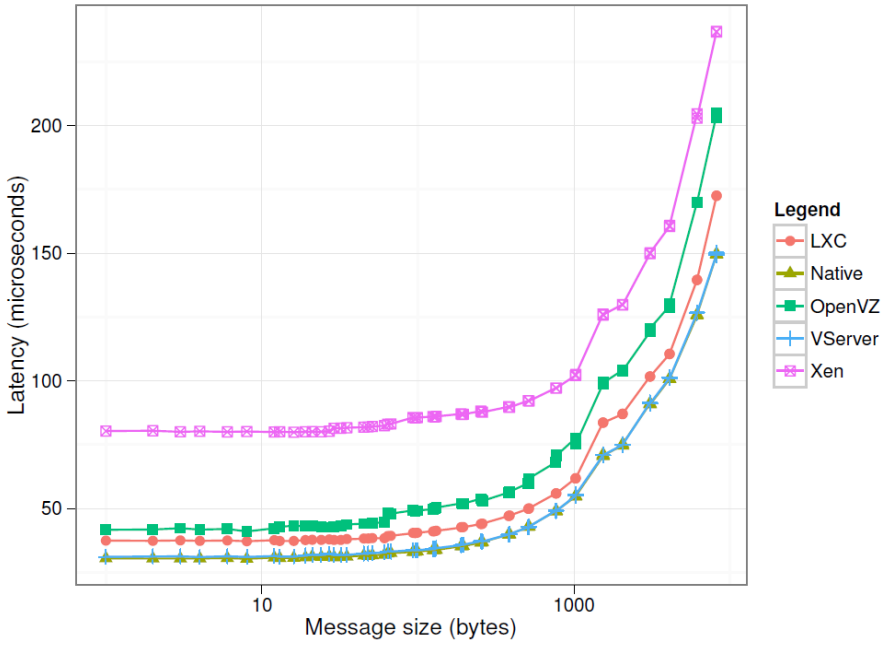


Memory Throughput
Stream

Networking Performance



Bandwidth
NetPIPE



Latency
NetPIPE

Isolation

PERFORMANCE ISOLATION FOR LU APPLICATION. THE RESULTS REPRESENT HOW MUCH THE APPLICATION PERFORMANCE IS IMPACTED BY DIFFERENT STRESS TESTS IN ANOTHER VM/CONTAINER. DNR MEANS THAT APPLICATION WAS NOT ABLE TO RUN.

	LXC	OpenVZ	VServer	Xen
CPU Stress	0	0	0	0
Memory	88.2%	89.3%	20.6%	0.9%
Disk Stress	9%	39%	48.8%	0
Fork Bomb	DNR	0	0	0
Network Receiver	2.2%	4.5%	13.6%	0.9%
Network Sender	10.3%	35.4%	8.2%	0.3%

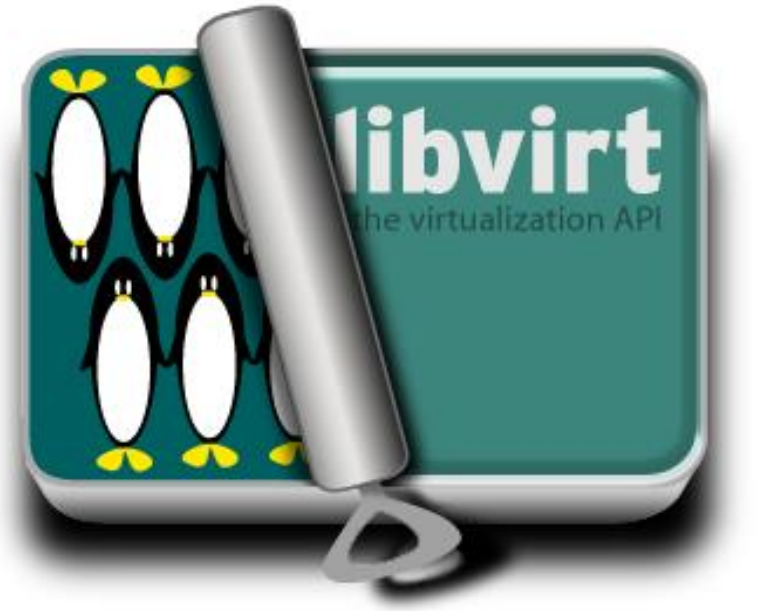
OS Virtualization

Libvirt

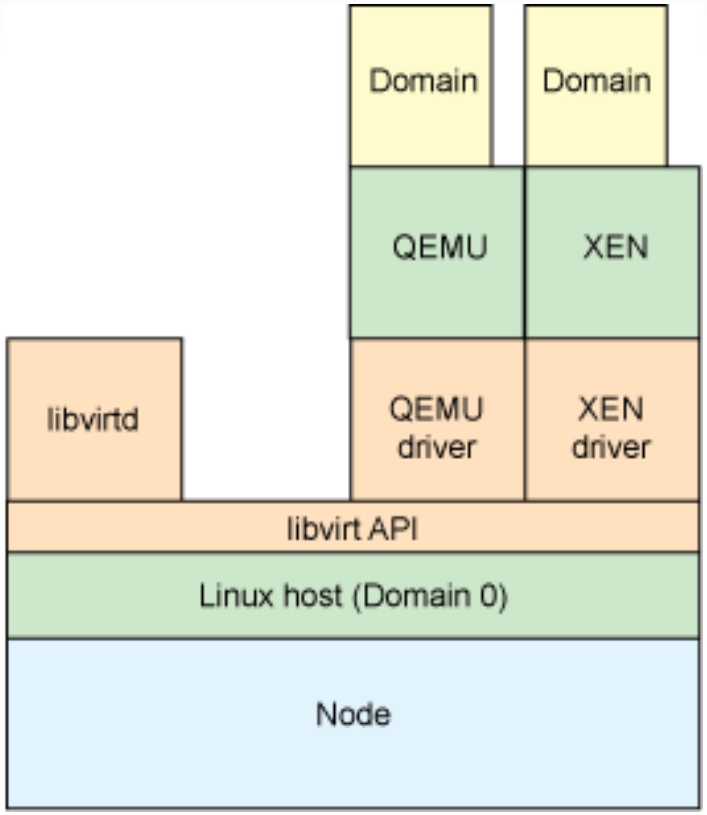
June 2013

About

- The “Virtualization API”
- Multiple supported technologies:
  - KVM / QEMU
  - Xen
  - LXC
  - VMWare
  - VirtualBox
- Stable C API
- Remote management

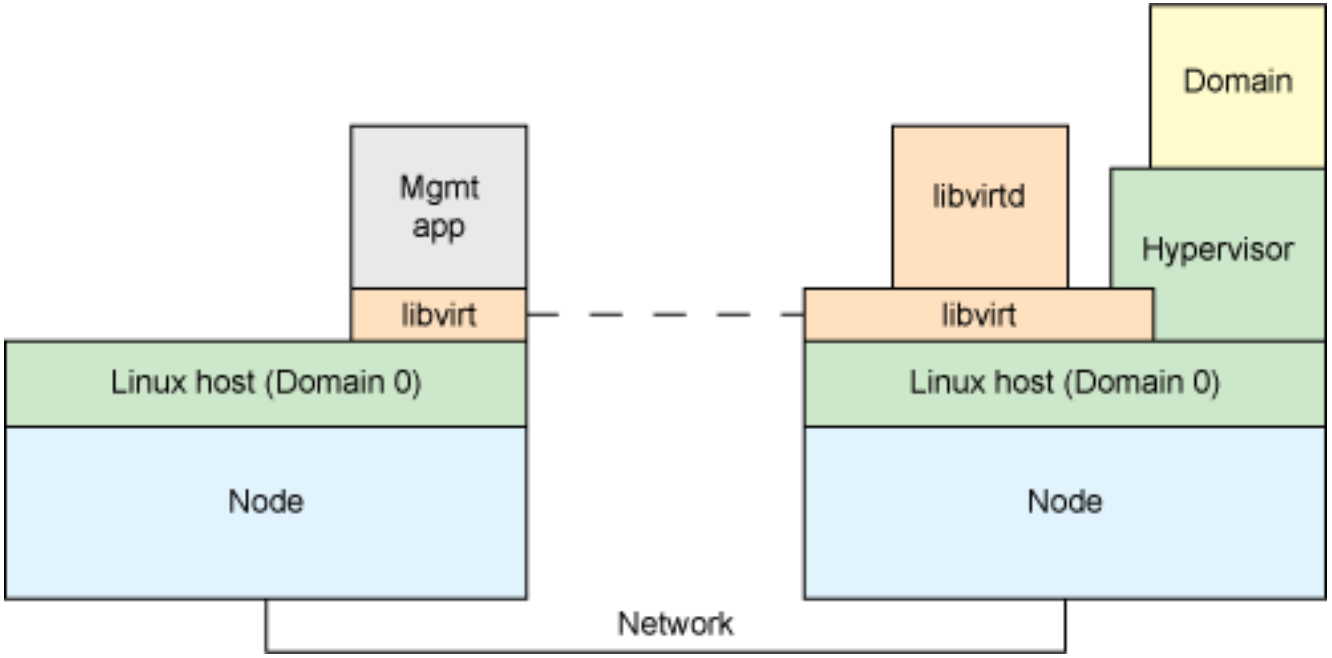


Driver Based Architecture



picture from http://www.ibm.com/developerworks/library/l-libvirt/figure3.gif

Hypervisor Control



picture from http://www.ibm.com/developerworks/ssa/linux/library/l-libvirt/figure2.gif

OS Virtualization

Relevance

June 2013

Popularity

- Running on:
  - Major distros: Fedora, Debian, Ubuntu, ...
  - Android
  - Virtually any system with Linux >= 2.6.26
- Connected projects:
  - **docker** - The Linux Container Runtime
  - **CRIU** - Checkpoint-Restart in Userspace
  - **lmctfy –** Let Me Contain That For You
- Maintained by both kernel and userspace developers

Use Cases

- General:
  - Server replication
  - Application sandboxing
  - Legacy software support
  - Live migration
- Embedded (networking, smartphones):
  - Separate traffic from different departments
  - Separate QoS policies
  - Run RTOS and HLOS at the same time

Freescale USDPAA in Containers

- DPAA - DataPath Acceleration Architecture
  - HW architecture providing advanced networking capabilities
  - Present in dedicated networking equipment
  - Traffic shaping, package accelerators, cryptography engine
- USDPAA - User Space DPAA
  - Userspace drivers based on the kernel UIO framework
  - Increased flexibility in application development
  - Reduced risk of bugging the kernel
  - Better error handling and system protection
  - Performance overhead
- Multiple USDPAA instances in containers
  - Improved isolation
  - Additional protection layer
  - Finer resource tuning

References

- https://linuxcontainers.org/
- https://www.stgraber.org/2013/12/20/lxc-1-0-blog-post-series/
- https://www.youtube.com/channel/UCIxsmRWj3-795FMIrsikd3A/videos
- http://libvirt.org/drvlxc.html
- http://www.docker.com/
- https://github.com/google/lmctfy
- http://criu.org/Main_Page


freescale™