OS Virtualization

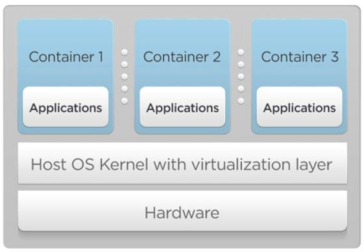
- Part 1: OS Virtualization
- Part 2: Fair share allocation
- Part 3: Docker and linux containers

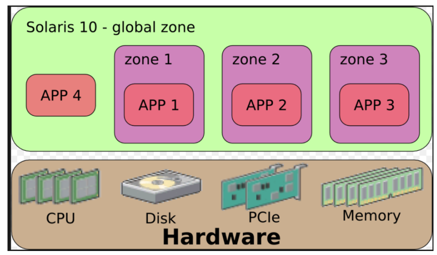
Part 1: OS Virtualization

- Recall virtualization: use native interface to emulate another one
- Broader view of OS virtualization:
  - OS interface (e.g., sys call interface) can emulate another OS interface
    - E.g., Solaris zone can emulate older kernel version
- Modern view of OS virtualization
  - **OS paradigm where kernel allows multiple isolated user space instances**
  - Each instance looks like real machine running OS
  - Outside processes can see all resources; processes inside isolated instances see a restricted set

OS Virtualization

- Emulate OS-level interface with native interface
- “Lightweight” virtual machines
  - No hypervisor, OS provides necessary support

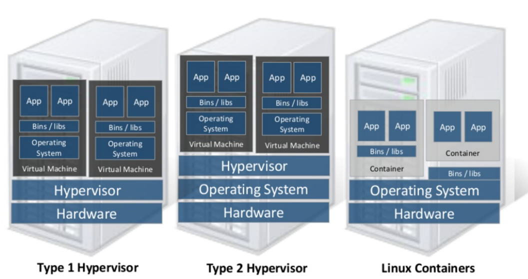




- Referred to as *containers* (“isolated set of processes”)
  - Solaris containers, BSD jails, Linux containers

Linux Containers (LXC)

- Containers share OS kernel of the host
  - OS provides resource isolation
- Benefits
  - Fast provisioning, bare-metal like performance, lightweight



Material courtesy of “Realizing Linux Containers” by Boden Russell, IBM

OS Mechanisms for LXC

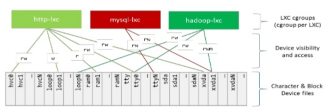
- OS mechanisms for resource isolation and management
- namespaces: process-based resource isolation
- Cgroups: limits, prioritization, accounting, control
- chroot: apparent root directory
- Linux security module, access control
- Tools (e.g., docker) for easy management

Linux Namespaces

- Namespace: restrict what can a container see?
  - Provide process level isolation of global resources
- Processes have illusion they are the only processes in the system
- MNT: mount points, file systems (what files, dir are visible)?
- PID: what other processes are visible?
- NET: NICs, routing
- Users: what uid, gid are visible?
- chroot: change root directory

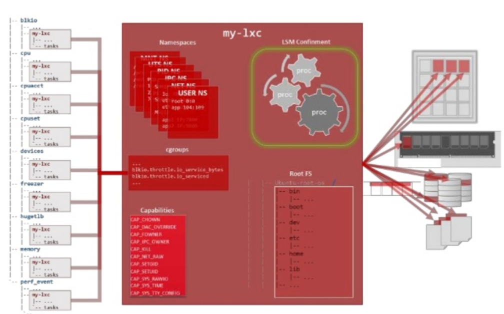
Linux cgroups

- Resource isolation
  - what and how much can a container use?
    - Set upper bounds (limits) on resources that can be used
    - Fair sharing of certain resources
- Examples:
  - cpu: weighted proportional share of CPU for a group
  - cpuset: cores that a group can access
  - block io: weighted proportional block IO access
  - memory: max memory limit for a group



Putting it all together

- Images: files/data for a container
  - can run different distributions/apps on a host
- Linux security modules and access control
- Linux capabilities: per process privileges



Part 2: Proportional Share Scheduling

- Proportional-share scheduling: allocate a fraction ("slice/share") of the resource
  - allocate CPU capacity to containers, VM, or a process
  - allocate network bandwidth to an application, container
- *Share-based* scheduling:
  - Assign each process a weight w_i (a "share")
  - Allocation is in proportional to share
  - fairness: reused unused cycles to others in proportion to weight
  - Examples: fair queuing, start time fair queuing
- *Hard limits*: assign upper bounds (e.g., 30%), no reallocation

Weighted Fair Queuing (WFQ)

- One of the original proportional share schedulers
- Each process /container assigned a weight w_i
  - each receives $w_i / \sum_j w_j$ fraction of resource
- OS keep a counter for each process s_i
  - Tracks how much CPU service the process has received
  - After each quantum, $s_i = s_i + \frac{q}{w_i}$ where q is quantum length
  - Scheduler schedules task with min s_i
  - what happens when process blocks: accumulates "credit" and can starve others
    - Track $s_{min} = min(s_1, s_2, ..)$ and $s_i = max(s_{min}, s_i + \frac{q}{w_i})$

Share-based Schedulers

```
From     paolo <>
Subject  [PATCH RFC RESEND 00/14] New version of the BFQ I/O Scheduler
Date     Tue, 27 May 2014 14:42:24 +0200

From: Paolo Valente <paolo.valente@unimore.it>

[Re-posting, previous attempt seems to have partially failed]

Hi,
this patchset introduces the last version of BFQ, a proportional-share
storage-I/O scheduler. BFQ also supports hierarchical scheduling with
a cgroups interface. The first version of BFQ was submitted a few
```

[PATCH RFC 00/22] Replace the CFQ I/O Scheduler with BFQ

From: Paolo Valente
Date: Mon Feb 01 2016 - 17:50:39 EST

- **Next message:** Paolo Valente: "[PATCH RFC 03/22] block, cfq: remove deep seek queues logic"

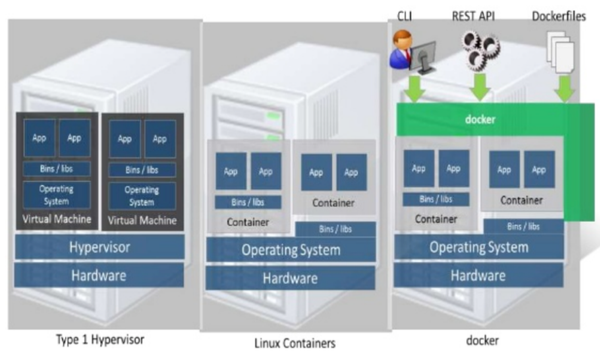
T2 instances' baseline performance and ability to burst are governed by CPU Credits. Each T2 instance receives CPU Credits continuously, the rate of which depends on the instance size. T2 instances accrue CPU Credits when they are idle, and use CPU credits when they are active. A CPU Credit provides the performance of a full CPU core for one minute.

Docker

- Linux containers are a set of kernel features
  - Need user space tools to manage containers
  - Virtuozo, OpenVZm, VServer,Lxc-tools, Docker
- What does Docker add to Linux containers?
  - Portable container deployment across machines
  - Application-centric: geared for app deployment
  - Automatic builds: create containers from build files
  - Component re-use
- Docker containers are self-contained: no dependencies

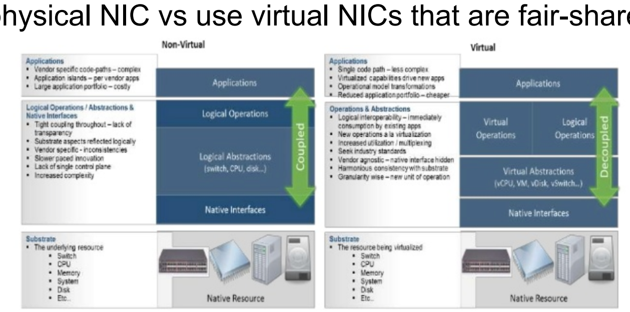
Docker

- Docker uses Linux containers



LXC Virtualization Using Docker

- Portable: docker images run anywhere docker runs
- Docker decouples LXC provider from operations
  - uses virtual resources (LXC virtualization)
    - fair share of physical NIC vs use virtual NICs that are fair-shared



Docker Images and Use

- Docker uses a union file system (AuFS)
  - allows containers to use host FS safely
- Essentially a copy-on-write file system
  - read-only files shared (e.g., share glibc)
  - make a copy upon write
- Allows for small efficient container images
- Docker Use Cases
  - “Run once, deploy anywhere”
  - Images can be pulled/pushed to repository
  - Containers can be a single process (useful for microservices) or a full OS

Case Study: PlanetLab

- Distributed cluster across universities
  - Used for experimental research by students and faculty in networking and distributed systems
- Uses a virtualized architecture
  - Linux Vservers
  - Node manager per machine
  - Obtain a “slice” for an experiment: slice creation service