Cloud Computing, Containers and VMs

Dan Williams (based on prior material from Godmar Back)

Virginia Tech

December 6, 2022

VIRGINIA TECH

Cloud Computing

- "X as a service"
- Infrastructure as a Service
  - user manages software - from kernel to applications
  - cloud manages "infrastructure" (power, cooling, etc.)
  - Ex: Amazon EC2
- Platform as a Service
  - user manages application and specifies software configuration (e.g., which runtime to use) on cloud
  - Ex: Heroku
- Software as a Service
  - application runs on servers in the cloud
  - Ex: Google Docs

VIRGINIA TECH

Deployment Models

- Cloud
- Hybrid Cloud
- On-prem cloud

VIRGINIA TECH

Defining characteristics

- Elasticity
  - scale up and down number of instances (often automatically)
  - only pay for what you use
- Multi-tenant
  - ensure resources are highly utilized (statistical multiplexing)
- Isolation
  - each user has "private view" of system
  - tenants should not interfere with each other
  - may be mutually distrusting

VIRGINIA TECH

How to run user workloads?

- Process?
  - private address space..
- How to package application dependencies?
  - executable built on different system
  - will target system have the right shared libraries?
- Will processes interfere with each other?
  - Yes - because so much of the system is shared:
  - shared filesystem - what if different needs for /etc/config?
  - shared port space - what if multiple on port 80?
  - shared pids
  - others...

Bottom Line

Processes are an ill-suited abstraction for a multi-tenant cloud scenario.

VIRGINIA TECH

How to run user workloads? Solutions:

- Bare Metal
  - each user workload gets its own physical machine
- Virtual Machines
  - each user workload gets its own virtual machine
  - everything application needs is in a virtual disk image
  - guest kernels: entire kernel is not shared
  - hypervisor/virtual machine monitor multiplexes physical machine
- Containers
  - private filesystem (chroot/layered)
  - kernel mechanisms provide "private view" of resources (namespaces, cgroups)

VIRGINIA TECH

Virtual Machines in the 70s

- Goldberg 1972, 1974
- Some reasons:
  - improving and testing OS software
  - running H/W diagnostic software
  - running different OSes or versions
  - running with a virtual hardware configuration different than physical machine
  - etc.
- Popek/Goldberg Requirements (1974)
  - equivalence/fidelity
    - program should exhibit same behavior
  - resource control
    - VMM must have full control of resources
  - efficiency
    - most instructions should execute natively

VIRGINIA TECH

How to run a virtual machine

- Direct Execution
- Basic idea: deprivilege (run OS as user instead of supervisor)
- "trap and emulate"
- question: will all instructions trap?
  - IBM/360 (70's) yes
  - x86 (prior to VT extensions) no...

VIRGINIA TECH

VM resurgence in early 2000s

- Dynamic binary translation
  - replace supervisor instructions in guest or force them to trap for trap-and-emulate
  - VMware workstation released in 1999
- Modify guest: paravirtualization
  - Xen (open source 2002)
  - Amazon EC2 was originally based on Xen
- Hardware extensions
  - 2005/2006 Intel VT-x AMD-V

VIRGINIA TECH

Memory management in a VM

- Extra layer of translation
  - guest virtual → guest physical → host physical (machine)
- Approaches:
  - Shadow page table
    - hypervisor makes copy of page table, installs copy in MMU
  - Paravirtualization
    - cooperation of guest
  - Extended/nested page tables
    - hardware performs additional translation

VIRGINIA TECH

Resource management for virtual machines

- OSes are used to fixed/dedicated RAM/CPU/devices
  - can we get better efficiency?
- Virtual CPUs
- Memory
  - Page sharing
  - Memory ballooning
- I/O devices
  - virtio
  - passthrough
  - "smart devices"

Resource management

Lots of classic mechanisms to rethink for VMs!

VIRGINIA TECH

Are VMs Overkill?

OS kernel is already good at resource management...

VIRGINIA TECH

Kernel mechanisms for Containers

- chroot
  - "chroot jail"
  - use some other directory as "/" for this process and all children
- Namespaces
  - private filesystem (chroot/layered)
  - UTS - hostname
  - mount - filesystem
  - PID - pids (ps only shows pids in namespace)
  - IPC - shmem, semaphores
  - user - users, groups, etc.
  - network - ports, devices, etc.
- Cgroups
  - limit resource consumption
  - CPU, mem

VIRGINIA TECH

Clone and unshare(2)

- Recall *fork* and *pthread_create* use *clone* under the covers
- Flags to *clone* control sharing (e.g., CLONE_VM), including namespaces!
- Can also *unshare* after the fact (e.g., CLONE_NEWPID)

VIRGINIA TECH

So what is a container?

- a set of processes sharing dedicated chroot, namespaces, cgroups

VIRGINIA TECH

How to build containers

- Dockerfile - defacto standard for building containers
  - filesystem image for container
  - build from other images: layering

VIRGINIA TECH

Building containers: Dockerfile

```
# Pull base image.
FROM dockerfile/ubuntu

# Install Nginx.
RUN \
  add-apt-repository -y ppa:nginx/stable && \
  apt-get update && \
  apt-get install -y nginx && \
  rm -rf /var/lib/apt/lists/* && \
  echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
  chown -R www-data:www-data /var/lib/nginx

# Define mountable directories.
VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/etc/nginx/conf.d", "/var/log/nginx", "/var/www/html"]

# Define working directory.
WORKDIR /etc/nginx

# Define default command.
CMD ["nginx"]

# Expose ports.
EXPOSE 80
EXPOSE 443
```

VIRGINIA TECH

The container ecosystem

- CNCF - cloud native computing foundation
- Kubernetes
- Service Mesh
- Pods and sharing
  - containers that share some namespaces
  - e.g., share filesystem, network

VIRGINIA TECH

Virtualization spectrum

- threads
- processes
- containers
- virtual machines
- physical machines

- higher density, lower overhead, ease of sharing
-
-
-
- better isolation, protection, more user control

VIRGINIA TECH

Container security concerns

- Attack surface through system call interface
- Same as processes!
- How to reduce it:
  - filter system calls (ex: seccomp)
  - use virtualization!?

VIRGINIA TECH

Virtual machine concerns

- Large, unwieldy images
- Slow to boot
  - problem for serverless

VIRGINIA TECH

Serverless: a new model

- function as a service
- user only writes/supplies function in supported language
- event-based: attach to event
  - thumbnail example
- "stateless"
- charged on ms granularity

VIRGINIA TECH

State of the art "containers": microVMs

- run container in a lightweight VM
  - AWS Firecracker
  - Kata containers
- Integrated with container orchestration systems (e.g., Kubernetes)

VIRGINIA TECH

References

VIRGINIA TECH