


# TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

## Containerization



Wes J. Lloyd  
 School of Engineering and Technology  
 University of Washington - Tacoma

Credit: some content based on Salman A. Baset, IBM, WOC 2018 @ IC2E - Container Security

## FEEDBACK - 11/16

- Kernel Based Virtual Machines
  - Integrate the virtualization hypervisor directly into the Linux kernel
  - Allows host machine to boot a real OS directly
  - Host OS is no longer a virtualized guest (dom0 XEN)
- Interrupts and Timers
  - Last aspect to have HW level support to virtualization


Paper: "Achieving High Resolution Timer Events in Virtualized Environment" Plos One Journal article (2014)  
<https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0130887&type=printable>

- Describes problems with achieving reliable timer emulation with virtual machines

November 19, 2018 TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.2

## ACHIEVING HIGH RESOLUTION TIMER EVENTS IN VIRTUALIZED ENVIRONMENT

- OS: Linux Gentoo 3.17.8 host & guest OS
- HW: Intel E5645 2.4 GHz (old 2010 CPU, circa 1st/2nd gen EC2)
- Tested hypervisors:
  - VMWare Player 6.0
  - Virtual Box 4.3.18 (2018 release is at 5.2)
  - KVM with Linux 3.17.8 kernel
  - Xen 4.5



November 19, 2018 TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.3

## TIMER INTERRUPTS VIRTUALIZATION O/H

- Experiments to establish stability of timer events
 

timer setting = 100µs idle system	VMWare	Mean	Std. Dev
	VMWare	175.908	9.867
	VirtualBox	159.232	9.252
	Qemu	140.259	3.832
	KVM	114.087	0.753
	native Xen	100.221	0.339
	new timers	100.074	0.118
	Linux kernel	100.103	0.070

Emulation is slow
- With I/O load (shows isolation)
 

timer setting = 100µs system with heavy I/O	VMWare	Mean	Std. Dev
	VMWare	485.224	4.266K
	VirtualBox	156.717	13.723
	Qemu	177.905	80.585
	KVM	119.712	9.869
	native Xen	105.312	17.797
	new timers	100.151	0.168
	Linux kernel	101.454	6.940

Isolation is poor
- Suspect XEN is using paravirtual timer interrupts (2014)

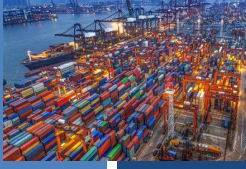
November 19, 2018 TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.4

## OBJECTIVES

- Term project questions
- Tutorial #6 - Serverless Database
- Tutorial #7 - Intro to Docker Containerization
- Containerization

November 19, 2018 TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.5

## CONTAINERIZATION



November 19, 2018 TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.6

## MOTIVATION FOR CONTAINERIZATION

- Containers provide “light-weight” alternative to full OS virtualization provided by a hypervisor
- Containers do not provide a full “machine”
- Instead use operating system constructs to provide “sand boxes” for execution
  - Linux cgroups, namespaces, etc.
- Containers can run on bare metal, or atop of VMs

**Containers**

**Hypervisor/VM**

November 19, 2018    TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma    L15.7

## CONTAINER PERFORMANCE - LU FACTORIZATION PERFORMANCE

■ Solve linear equations - matrix algebra

Performance data from IC2E 2015: Hypervisors vs. Lightweight Virtualization: A Performance Comparison

Platform	Mflops (higher is better)
KVM	~525
DOCKER	~528
LXC	~526
NATIVE	~524
OSV	~518

Fig. 4. The value of Linpack results on each platform over 15 runs. This is the particular case of N=1000.

November 19, 2018    TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma    L15.8

## CONTAINER PERFORMANCE - Y-CRUNCHER: PI CALCULATOR

Performance data from IC2E 2015: Hypervisors vs. Lightweight Virtualization: A Performance Comparison

Platform	Computation Time	Total Time
KVM	~1550	~1700
DOCKER	~1520	~1650
LXC	~1520	~1630
NATIVE	~1520	~1600

November 19, 2018    TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma    L15.9

## CONTAINER PERFORMANCE - BONNIE++

Performance data from IC2E 2015: Hypervisors vs. Lightweight Virtualization: A Performance Comparison

Platform	Block Output	Block Input
KVM	~25000	~100000
DOCKER	~55000	~210000
LXC	~60000	~205000
NATIVE	~75000	~220000

Fig. 6. Disk Throughput achieved by running Bonnie++ (test file of 25 GiB). Results for sequential writes and sequential read are shown.

November 19, 2018    TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma    L15.10

## WHAT IS A CONTAINER?

According to NIST (National Institute of Standards Technology)

- **Virtualization**: the simulation of the software and/or hardware upon which other software runs. (800-125)
- **System Virtual Machine**: A System Virtual Machine (VM) is a software implementation of a complete system platform that supports the execution of a complete operating system and corresponding applications in a cloud. (800-180 draft)
- **Operating System Virtualization** (aka OS Container): Provide multiple virtualized OSES above a single shared kernel (800-190). E.g., Solaris Zone, FreeBSD Jails, LXC
- **Application Virtualization** (aka Application Containers): Same shared kernel is exposed to multiple discrete instances (800-180 draft). E.g., Docker (containerd), rkt

November 19, 2018    TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma    L15.11

## OPERATING SYSTEM CONTAINERS

- Virtual environments: share the host kernel
- Provide user space isolation
- Replacement for VMs: run multiple processes, services
- Mix different Linux distros on same host
- Examples: LXC, OpenVZ, Linux Vserver, BSD Jails, Solaris zones

Identical OS containers

Different flavoured OS containers

\* Credit: <https://blog.risingstack.com/operating-system-containers-vs-application-containers/>

November 19, 2018    TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma    L15.12

## APPLICATION CONTAINERS

- Designed to package and run a single service
- All containers share host kernel
- Subtle differences from operating system containers
- Examples: Docker, Rocket
- Docker: runs a single process on creation
- OS containers: run many OS services, for an entire OS
- Create application containers for each component of an app
- Supports a micro-services architecture
- DevOPS: developers can package their own components in application containers
- Supports horizontal and vertical scaling

November 19, 2018 TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.13

## APPLICATION CONTAINERS - 2

- Container images are "layered"
- Base image: common for all components
- Add layers that are specific for components, services as needed
- Layering promotes reuse
- Reduces duplication of data across images

November 19, 2018 TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.14

## AUFS

- Union file system
- Copy-on-write
- <https://medium.com/@nagarwal/docker-containers-filesystem-demystified-b6ed8112a04a>
- <https://www.slideshare.net/jpetazzo/scale11x-lxc-talk-1/>

November 19, 2018 TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.15

## THREE-TIER ARCHITECTURE

OS containers

- Node.js
- Postgres
- Ngix

App containers

- Node.js
- Postgres
- Ngix

- Meant to be used as an OS - run multiple services
- No layered filesystems by default
- Built on cgroups, namespaces, native process resource isolation
- Examples - LXC, OpenVZ, Linux VServer, BSD Jail, Solaris Zones

- Meant to run for a single service
- Layered filesystems
- Built on top of OS container technologies
- Examples - Docker, Rocket

November 19, 2018 TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.16

## CONTAINER ISOLATION

- Is the host isolated from application containers?
- Are application containers isolated from each other?

November 19, 2018 TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.17

## LXC (LINUX CONTAINERS)

- Operating system level virtualization
- Run multiple isolated Linux systems on a host using a single Linux kernel
- Control groups(cgroups)
  - Including in Linux kernels => 2.6.24
  - Limit and prioritize sharing of CPU, memory, block/network I/O
- Linux namespaces
- Docker initially based on LXC

November 19, 2018 TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L15.18

## LINUX KERNEL NAMESPACES

- Partitions kernel resources
- Processes see only their set of resources
- Provides isolation
- Namespaces are hierarchical
- Parent processes can see down the hierarchy
- 7 namespaces in Linux (cgroups not shown)
- Each process can only see resources associated with the namespace, and descendent namespaces

pid	mnt
	ipc
user	net
UTS	

November 19, 2018
TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L15.19

## NAMESPACES - 2

```

root@358f3dfc3e:/# top -m 30 -n 10
top - 08:34:28 up 0:14, 0 users, load average: 0.00, 0.00, 0.00
Tasks: 4 total, 1 running, 3 sleeping, 0 stopped, 0 zombie
Mem(s): 0.0 total, 0.0 free, 0.0 buffers, 0.0 cached, 0.0 swap
Mem: 3053108 total, 2798844 free, 537548 used, 98668 buff/cache
Mem Swap: 0 total, 0 free, 0 used, 3560784 avail Mem

PID USER   PR  NI  VIRT  RES  SHR  S#    %CPU  %MEM     time+   COMMAND
1 root    20  0  1316  744  764  S   0.0  0.0  0:00.00  smdmonitd
5 root    20  0  4532  764  764  S   0.0  0.0  0:00.00  sleep
6 root    20  0  18596 3416 3665  S  0.0  0.1  0:00.01  bash
14 root   20  0  36596 3228 2796  R   0.0  0.1  0:00.04  top
                    
```

- Provides isolation of OS entities for containers
- mnt**: separate filesystems
- pid**: independent PIDs; first process in container is PID 1
- ipc**: prevents processes in different IPC namespaces from being able to establish shared memory. Enables processes in different containers to reuse the same identifiers without conflict. ... provides expected VM like isolation...
- user**: user identification and privilege isolation among separate containers
- net**: network stack virtualization. Multiple loopbacks (lo)
- UTS (UNIX time sharing)**: provides separate host and domain

November 19, 2018
TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L15.20

## CONTROL GROUPS (CGROUPS)

- Collection of Linux processes
- Group-level resource allocation: CPU, memory, disk I/O, network I/O
- Resource limiting**
  - Memory, disk cache
- Prioritization**
  - CPU share
  - Disk I/O throughput
- Accounting**
  - Track resource utilization
  - For resource management and/or billing purposes
- Control**
  - Pause/resume processes
  - Checkpointing → Checkpoint/Restore in Userspace (CRIU)
  - <https://criu.org>

November 19, 2018
TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L15.21

## CGROUPS - 2

- Control groups are hierarchical
- Groups inherit limits from parent groups
- Linux has multiple cgroup controllers (subsystems)
- ls /proc/cgroups
- "memory" controller limits memory use
- "cpuacct" controller accounts for CPU usage
- cgroup filesystem**:
  - /sys/fs/cgroup
  - Can browse resource utilization of containers...

Subsys name	hierarchy	num_cgroups	enabled
cpuacct	3	2	1
cpu	5	97	1
cpuacct	5	97	1
blkio	8	97	1
memory	9	218	1
devices	6	97	1
freezer	4	2	1
net_cls	2	2	1
perf_event	10	2	1
net_prio	2	2	1
hugetlb	7	2	1
pids	11	98	1

November 19, 2018
TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L15.22

## 2016 DOCKER SURVEY

- Docker application containers
  - Leading containerization vehicle

**80%**

say Docker is part of cloud strategy

**60%**

plan to use Docker to migrate workloads to cloud

**41%**

want application portability across environments

**35+%**

want to avoid cloud vendor lock-in

November 19, 2018
TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L15.23

## DOCKER EXECUTION ENVIRONMENTS

- Original default Docker execution environment: LXC
- Docker v0.9: libcontainer introduced (~2014)
- Now runc (2015)

- Provides Docker access to Linux container APIs
- Execution drivers concept:
  - libvirt
  - lxc
  - systemd-nspawn
- Enable docker to leverage many OS containers as the exec environment
- OpenVZ, system-nspawn, libvirt-lxc, libvirt-sandbox, qemu/kvm, BSD Jails, Solaris Zones, and chroot

November 19, 2018
TCSS562: Software Engineering for Cloud Computing [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L15.24

## DOCKER

- Docker daemon “dockerd”
  - Provides docker services to Linux
- Docker 1.11+
- Open Container Initiative
- June 2015: Industry standard for container runtimes and formats
- Ensure containers are portable among different execution environments (engines)

Docker Client-Server Architecture

Credit: <https://hackernoon.com/docker-containerd-standalone-runtimes-heres-what-you-should-know-9834e155426>

November 19, 2018	<small>TCCS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma</small>	L15.25
-------------------	--	--------

## DOCKER - 2

Containerd Integration Architecture

- Docker CLI: interfaces with dockerd daemon
- Docker engine: dockerd daemon, interfaces with Containerd
- Containerd: simple daemon, interfaces with runc to manage containers; CRUD interface for containers, images, volumes, networks, builds; HTTP API → Google RPC (gRPC) interface;
- runc: lightweight command-line tool for running containers; Interfaces with Linux cgroups, namespaces; Runs an OCI container

November 19, 2018	<small>TCCS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma</small>	L15.26
-------------------	--	--------

## DOCKER - 3

- Docker architecture:
- Other Docker tools:
- **Docker Machine:** automatically provision and manage sets of docker hosts to form a cluster
- **Docker Swarm:** Clusters multiple docker hosts together to manage as a cluster.
- **Docker Compose:** Config file (YAML) for multi-container application; Describes how to deploy and configure multiple containers

November 19, 2018	<small>TCCS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma</small>	L15.27
-------------------	--	--------

## CONTAINER ORCHESTRATION FRAMEWORKS

- Framework(s) to deploy multiple containers
- Provide container clusters using cloud VMs
- Similar to “private clusters”
- Reduce VM idle CPU time in public clouds
- Better leverage “sunk cost” resources
- Compact multiple apps onto shared public cloud infrastructure
- Generate to cost savings
- Reduce vendor lock-in

November 19, 2018	<small>TCCS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma</small>	L15.28
-------------------	--	--------

## KEY ORCHESTRATION FEATURES

- Management of container hosts
- Launching set of containers
- Rescheduling failed containers
- Linking containers to support workflows
- Providing connectivity to clients outside the container cluster
- Firewall: control network/port accessibility
- Dynamic scaling of containers: horizontal scaling
  - Scale in/out, add/remove containers
- Load balancing over groups of containers
- Rolling upgrades of containers for application


November 19, 2018	<small>TCCS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma</small>	L15.29
-------------------	--	--------

## CONTAINER ORCHESTRATION FRAMEWORKS - 2

- Docker swarm
- Apache mesos/marathon
- Kubernetes
  - Many public cloud provides moving to offer Kubernetes-as-a-service
- Amazon elastic container service (ECS)
- Apache auroa
- Container-as-a-Service
  - Serverles containers without managing clusters
  - Azure Container Instances, AWS Fargate...

November 19, 2018	<small>TCCS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma</small>	L15.30
-------------------	--	--------

# TUTORIAL #7 DOCKER, CGROUPS, RESOURCE ISOLATION



November 19, 2018    TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L15.31

## DOCKER CLI

- Docker CLI → Docker Enginer (dockerd) → containerd → runc
- Docker installation
- Docker file
- Docker run
- Docker ps
- Docker exec -it
- Docker stop

November 19, 2018    TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L15.32

```

arguments:
attach      Attach local standard input, output, and error streams to a running container
build      Build an image from a Dockerfile
commit     Create a new image from a container's changes
cp         Copy files/folders between a container and the local filesystem
create     Create a new container
deploy     Deploy a new stack or update an existing stack
diff       Inspect changes to files or directories on a container's filesystem
events     Get real time events from the server
exec       Run a command in a running container
export     Export a container's filesystem as a tar archive
history    Show the history of an image
images    List images
import     Import the contents from a tarball to create a filesystem image
info       Display system-wide information
inspect    Return low-level information on Docker objects
kill       Kill one or more running containers
load      Load an image from a tar archive or STDIN
login     Log in to a Docker registry
logout    Log out from a Docker registry
logs      Fetch the logs of a container
pause     Pause all processes within one or more containers
port      List port mappings or a specific mapping for the container
ps        List containers
pull      Pull an image or a repository from a registry
push      Push an image or a repository to a registry
rename    Rename a container
restart   Restart one or more containers
rm        Remove one or more containers
rmi       Remove one or more images
run       Run a command in a new container
save     Save one or more images to a tar archive (streamed to STDOUT by default)
search    Search the Docker Hub for images
start     Start one or more stopped containers
stats    Display a live stream of container(s) resource usage statistics
stop     Stop one or more running containers
tag       Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top       Display the running processes of a container
update   Update configuration of one or more containers
version  Show the Docker version information
wait     Block until one or more containers stop, then print their exit codes
    
```


November 19, 2018    TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L15.33

## TUTORIAL 7

- Linux performance benchmarks
- stress-ng
- 100s of CPU, memory, disk, network stress tests
- Sysbench
- Used in tutorial for memory stress test

November 19, 2018    TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L15.34

# QUESTIONS



November 19, 2018    TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L15.35