

# TCCS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

## Containerization

Wes J. Lloyd  
 Institute of Technology  
 University of Washington - Tacoma



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

## OBJECTIVES

- Term project questions
- AWS Educate
- Tutorial #2
- Midterm Wednesday 5/9
- Containerization
- Tutorial #3 - Containers, cgroups, isolation

April 30, 2018 TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma L9.2

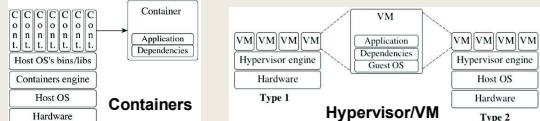
## FEEDBACK

- ...

April 30, 2018 TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma L9.3

## 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



April 30, 2018 TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma L9.4

## CONTAINER PERFORMANCE - LU FACTORIZATION PERFORMANCE

Solve linear equations - matrix algebra

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

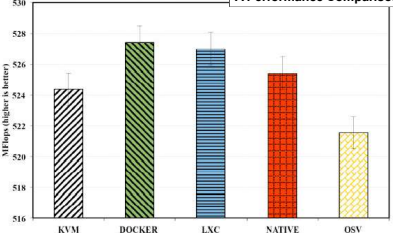
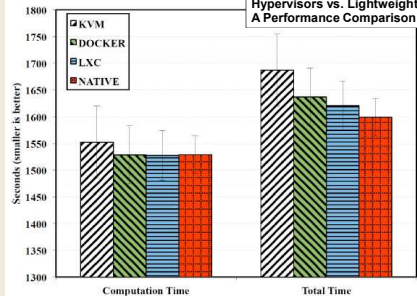


Fig. 4. The value of Linpack results on each platform over 15 runs. This is the particular case of N=1000.

April 30, 2018 TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma L9.5

## CONTAINER PERFORMANCE - Y-CRUNCHER: PI CALCULATOR

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



April 30, 2018 TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma L9.6

### CONTAINER PERFORMANCE – BONNIE++

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

Category	KVM	DOCKER	LXC	NATIVE
Block Output	~25,000	~55,000	~65,000	~75,000
Block Input	~45,000	~210,000	~215,000	~220,000

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

April 30, 2018 | TCCS562: Software Engineering for Cloud Computing [Spring 2018] | Institute of Technology, University of Washington - Tacoma | L9.7

### 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

April 30, 2018 | TCCS562: Software Engineering for Cloud Computing [Spring 2018] | Institute of Technology, University of Washington - Tacoma | L9.8

### 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/>

April 30, 2018 | TCCS562: Software Engineering for Cloud Computing [Spring 2018] | Institute of Technology, University of Washington - Tacoma | L9.9

### 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

April 30, 2018 | TCCS562: Software Engineering for Cloud Computing [Spring 2018] | Institute of Technology, University of Washington - Tacoma | L9.10

### 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

Kernel | Base Image | add emacs | add Apache | writable Container Image | references parent image

April 30, 2018 | TCCS562: Software Engineering for Cloud Computing [Spring 2018] | Institute of Technology, University of Washington - Tacoma | L9.11

### THREE-TIER ARCHITECTURE

OS containers

- Node.js
- Postgres
- Nginx

App containers

- Node.js
- Postgres
- Nginx

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

April 30, 2018 | TCCS562: Software Engineering for Cloud Computing [Spring 2018] | Institute of Technology, University of Washington - Tacoma | L9.12

### CONTAINER ISOLATION

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

The diagram illustrates the isolation layers. On the left, two application containers (App) sit on a Binaries/Bin layer, which sits on a Container runtime layer, which sits on a VM kernel layer, which sits on the Host kernel. On the right, two application containers (App) sit on a Binaries/Bin layer, which sits on a Container runtime layer, which sits on the Host kernel. This shows that containers are isolated from each other and from the host kernel through the VM kernel and container runtime layers.

April 30, 2018
TCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.13

### 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

April 30, 2018
TCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.14

### 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

The diagram shows a tree-like structure of namespaces. 'pid' and 'mnt' are at the top level. 'ipc' is nested under 'pid'. 'user' and 'net' are nested under 'ipc'. 'UTS' is nested under 'user'.

April 30, 2018
TCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.15

### NAMESPACES - 2

- 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

The screenshot shows a terminal window with system statistics. The output of the 'top' command is visible, showing system load averages, memory usage, and a list of processes with their PIDs, users, and commands.

April 30, 2018
TCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.16

### 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>

April 30, 2018
TCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.17

### 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	cgrouops	enabled
1	cpuacct	3	97	1	1
2	cpu	5	97	1	1
3	cpuacct	5	97	1	1
4	blkio	8	97	1	1
5	memory	9	218	1	1
6	devices	6	97	1	1
7	freezer	4	2	1	1
8	net_cls	2	2	1	1
9	perf_event	10	2	1	1
10	net_prio	2	2	1	1
11	hugetlb	7	2	1	1
12	pids	11	98	1	1

April 30, 2018
TCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.18

### 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

April 30, 2018
TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.19

### DOCKER EXECUTION ENVIRONMENTS

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

■ Provides Docker access to Linux container APIs

■ Execution drivers concept:

- 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

April 30, 2018
TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.20

### 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

April 30, 2018
TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.21

### DOCKER - 2

Container 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

April 30, 2018
TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.22

### 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

April 30, 2018
TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.23

### 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

April 30, 2018
TCCS562: Software Engineering for Cloud Computing [Spring 2018]  
Institute of Technology, University of Washington - Tacoma
L9.24

## 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

April 30, 2018
TCSS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma
L9.25


## 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 aurora

April 30, 2018
TCSS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma
L9.26

## TUTORIAL #3 DOCKER, CGROUPS, RESOURCE ISOLATION

April 30, 2018      TCSS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma



L9.27

## DOCKER CLI

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

April 30, 2018
TCSS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma
L9.28

```

commands:
  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
  unpause     Unpause all processes within one or more containers
  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
    
```

## TUTORIAL 3

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

April 30, 2018
TCSS562: Software Engineering for Cloud Computing [Spring 2018]  
 Institute of Technology, University of Washington - Tacoma
L9.30

