

TCSS 562:
SOFTWARE ENGINEERING
FOR CLOUD COMPUTING

Containerization

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



1

OVERVIEW

- Midterm review
- Tutorial 7 to be posted this week (Tuesday)
- Tutorial 8 & 9 to be posted soon after
- Only 7 tutorials are required
 - Additional tutorials beyond 7 provide extra credit
- Group presentation topics are due tonight: 11/18
 - Each groups provides quick report answering questions submitting a PDF file on Canvas
- Term project checkin - due Sunday 11/24
- Grading: tutorial 4 this week, tutorial 5 next

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.2

2

TERM PROJECT DELIVERABLES

- Nine project teams
- Term project lightning presentations
 - Monday December 9th (5:50-7:50pm)
 - Takes place of final exam
 - Presentation length: 5 minutes + questions, total 8 minutes
 - Format and rubric coming soon
- Term project final paper and source code repository
 - Friday December 13 @ 11:59pm
 - Paper template to be provided

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.3

3

GROUP PRESENTATION

- Cloud technology presentation
- Cloud research paper presentation
- Submit topics and desired dates of presentation via Canvas by Monday November 18th @ 11:59pm
- Presentation dates:
 - Monday November 25 (3 groups)
 - Monday December 2 (3 groups)
 - Wednesday December 4 (3 groups)


November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.4

4

CONTAINERIZATION



November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.5

5

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

Container

Application

Dependencies

Host OS's bins/libs

Containers engine

Host OS

Hardware

Containers

Type 1

VM

VM

VM

VM

Application

Dependencies

Guest OS

Hypervisor engine

Hardware

Hypervisor/VM

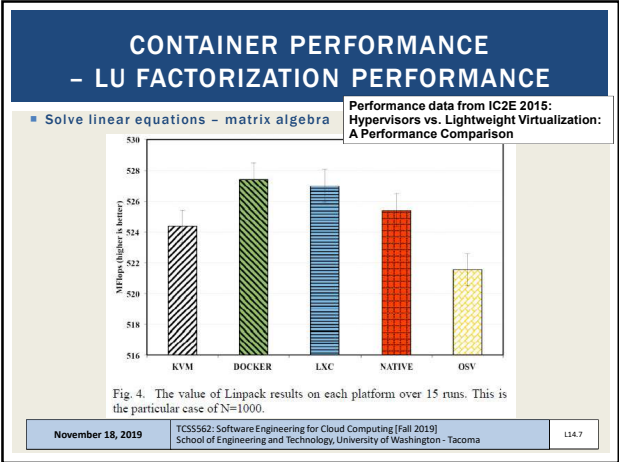
Type 2

November 18, 2019

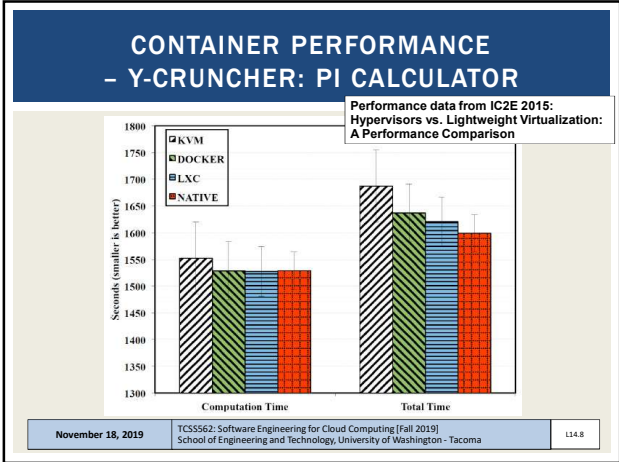
TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.6

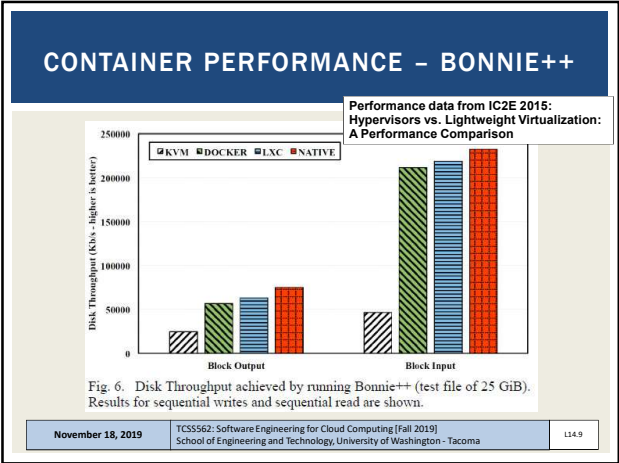
6



7



8



9

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 18, 2019TCCS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - TacomaL14.10

10

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: Ubuntu 14.04 image, Ubuntu 14.04 Container, Ubuntu 14.04 Container, Ubuntu 14.04 Container

Different flavoured OS containers: Ubuntu 14.04 image, RHEL 7 image, CentOS 6.6 image, RHEL 7 Container, CentOS 6.6 Container

November 18, 2019TCCS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - TacomaL14.11

11

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 18, 2019TCCS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - TacomaL14.12

12

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 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.13

13

OVERLAY FILE SYSTEMS

- Docker leverages overlay filesystems
- 1st: AUFS - Advanced multi-layered Unification filesystem
- Now: overlay2
- Union mount file system: combine multiple directories into one that appears to contain combined contents
- Idea: Docker uses layered file systems
- Only the top layer is writeable
- Other layers are read-only
- Layers are merged to present the notion of a real file system
- Copy-on-write- implicit sharing
 - Implement duplicate copy

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

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.14

14

LAYERED FS: BUILDING A CONTAINER

- Dockerfile:

```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.15

15

THREE-TIER ARCHITECTURE

OS containers

- Meant to 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

App containers

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

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.16

16

CONTAINER ISOLATION

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

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.17

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 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.18

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 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.19

19

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

```
root@355f3dfc1e:/# top -m 1024000 up 0:14, 0 users, load average: 0.00, 0.00, 0.00
Tasks: 4 total, 1 running, 3 sleeping, 0 stopped, 0 zombie
NIch(s): 0:00 us, 0:00 sy, 0:00 ni, 0:00 id, 0:00 wa, 0:00 hi, 0:00 si, 0:00 st
MiB Mem : 3053108 total, 2798864 free, 137548 used, 89868 buff/cache
MiB Swap: 0 total, 0 free, 0 used, 3580784 avail Mem

PID USER   PR  NI  TIDT  CPU S  MEM%  VSZ  RSS  UID  COMMAND
1 root    20  0  1370  997  0.0 S  0.0  0  0  0  0:00.00 bash
5 root    20  0  4532  764  0.0 S  0.0  0  0  0  0:00.00 sleep
6 root    20  0  18596 3456 0.0 S  0.0  0  0  0  0:00.01 bash
14 root   20  0  36396 3228 2796 R 0.0 0.1 0:00.04 top
```

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.20

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 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.21

21

CGROUPS - 2

- Control groups are hierarchical
- Groups inherit limits from parent groups
- Linux has multiple cgroup controllers (subsystems)
- Is /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 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.22

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 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.23

23

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:
 - 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

Docker (libcontainer) connects to libvirt, lxc, and systemd-nspawn, which all connect to Linux. Linux includes cgroups, namespaces, netlink, capabilities, selinux, netfilter, and apparmor.

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.24

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)

November 18, 2019

TCCS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.25

25

DOCKER - 2

- 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 18, 2019

TCCS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.26

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 18, 2019

TCCS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.27

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 18, 2019

TCCS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.28

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 18, 2019

TCCS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.29

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 aurora
- Container-as-a-Service
 - Serverles containers without managing clusters
 - Azure Container Instances, AWS Fargate...


November 18, 2019

TCCS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.30

30

TUTORIAL #7
DOCKER, CGROUPS,
RESOURCE ISOLATION



November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.31

31

DOCKER CLI

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

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.32

32

Commands:

attach

Build an image from a Dockerfile

build

Create a new image from a container's changes

commit

Copy files/folders between a container and the local filesystem

cp

Create a new container

create

Deploy a new stack or update an existing stack

deploy

Inspect changes to files or directories on a container's filesystem

diff

Get real time events from the server

events

Run a command in a running container

exec

Export a container's filesystem as a tar archive

export

Show the history of an image

history

List images

images

Import the contents from a tarball to create a filesystem image

import

Display system-wide information

info

Return low-level information on Docker objects

inspect

Kill one or more running containers

kill

Load an image from a tar archive or STDIN

load

Log in to a Docker registry

login

Log out from a Docker registry

logout

Fetch the logs of a container

logs

Pause all processes within one or more containers

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

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 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.34

34

QUESTIONS



November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.35

35

EXTRA SLIDES

November 18, 2019

TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma

L14.36

36