

# TCSS 462/562: (SOFTWARE ENGINEERING FOR) CLOUD COMPUTING

## Containerization

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



1

## OFFICE HOURS – FALL 2024

- **THIS WEEK**
- **Tuesdays:**
  - 2:30 to 3:30 pm - CP 229
- **Thursday\*:**
  - 6:00 pm to 7:00 pm – CP 229 and via Zoom\*
- Or email for appointment

> Office Hours set based on Student Demographics survey feedback  
\* - Four Friday faculty meetings have moved office hours to a 5pm starting time

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.2
-------------------	---	-------

2

## OBJECTIVES - 11/14

- **OBJECTIVES - 11/12**
- Tutorials Questions
- Class Presentations Schedule -  
Cloud Technology or Research Paper Review
- Quiz 1
- Tutorial 7
- Containerization
- Container Profiler

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.3
-------------------	---	-------

3

## ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas - Take After Each Class
- Extra Credit for completing

- Announcements
- Assignments**
- Discussions
- Zoom
- Grades
- People
- Pages
- Files
- Quizzes
- Collaborations
- UW Libraries
- UW Resources

▼ Upcoming Assignments

- 📄 **Class Activity 1 - Implicit vs. Explicit Parallelism**  
Available until Oct 11 at 11:59pm | Due Oct 7 at 7:50pm | -/10 pts
- 📄 **Tutorial 1 - Linux**  
Available until Oct 19 at 11:59pm | Due Oct 15 at 11:59pm | -/20 pts

▼ Past Assignments

- 📄 **TCSS 562 - Online Daily Feedback Survey - 10/5**  
Available until Dec 18 at 11:59pm | Due Oct 6 at 8:59pm | -/1 pts
- 📄 **TCSS 562 - Online Daily Feedback Survey - 9/30**  
Available until Dec 18 at 11:59pm | Due Oct 4 at 8:59pm | -/1 pts

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.4
-------------------	---	-------

4

TCSS 562 - Online Daily Feedback Survey - 10/5  
Started: Oct 7 at 1:13am  
Quiz Instructions

Question 1 0.5 pts

On a scale of 1 to 10, please classify your perspective on material covered in today's class:

1 2 3 4 5 6 7 8 9 10  
Mostly Review To Me Equal New and Review Mostly New to Me

Question 2 0.5 pts

Please rate the pace of today's class:

1 2 3 4 5 6 7 8 9 10  
Slow Just Right Fast

November 14, 2024 TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma L15.5

5

## MATERIAL / PACE

- Please classify your perspective on material covered in today's class (**42** respondents):
  - 1-mostly review, 5-equal new/review, 10-mostly new
  - **Average - 5.60** (↓ - *previous 5.84*)
- Please rate the pace of today's class:
  - 1-slow, 5-just right, 10-fast
  - **Average - 5.42** (↓ - *previous 5.13*)
- **Response rates:**
  - TCSS 462: 30/42 - 71.43%
  - TCSS 562: 12/20 - 60.00%

November 14, 2024 TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma L15.6

6

## FEEDBACK FROM 11/12

- ...

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.7
-------------------	---	-------

7

## AWS CLOUD CREDITS UPDATE

- **AWS CLOUD CREDITS ARE NOW AVAILABLE FOR TCSS 462/562**
- Credits provided on request with expiry of Sept 30, 2024
- Credit codes must be securely exchanged
- Request codes by sending an email with the subject “**AWS CREDIT REQUEST**” to [wllloyd@uw.edu](mailto:wllloyd@uw.edu)
- Codes can also be obtained in person (or zoom), in the class, during the breaks, after class, during office hours, by appt
  - 56 credit requests fulfilled as of Nov 13 @ 11:59p
- Codes not provided using discord

November 14, 2024	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.8
-------------------	--	-------

8

## OBJECTIVES - 11/14

- OBJECTIVES - 11/12
- **Tutorials Questions**
- Class Presentations Schedule -  
Cloud Technology or Research Paper Review
- Quiz 1
- Tutorial 7
- Containerization
- Container Profiler


November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.9
-------------------	---	-------

9

## Don't Forget to Terminate (Shutdown) all EC2 instances for Tutorials 3

Spot instances:  
c5d.large instance @ ~3.2 cents / hour

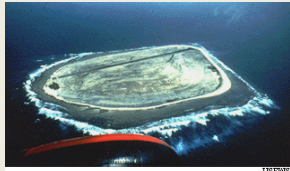
**\$0.78 / day**  
**\$5.48 / week**  
**\$23.78 / month**  
**\$285.42 / year**

AWS CREDITS → → → → → → → → 

10

## TUTORIAL SUBMISSION TIME

- Tutorials can now be submitted on the due date until the very last minute of the day **Anywhere-on-Earth (AOE)**
  - Equivalent to **4:59 AM Pacific Standard Time (PST)**
- Anywhere-on-Earth timezone: **Baker Island, Pacific Ocean**
- <https://www.timeanddate.com/time/zones/aoe>
- Uninhabited island in Pacific Ocean
- Coordinates            0° 11' 45" N 176° 28' 45" W
- Area                    2.1 km<sup>2</sup> (0.81 sq mi)
- Length                1.81 km (1.125 mi)
- Width                 1.13 km (0.702 mi)
- Coastline            4.8 km (2.98 mi)
- Highest elevation    8 m (26 ft)
- Population            0 (2000)



November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.11
-------------------	---	--------

11

## TUTORIAL 5 – DUE NOV 14

- Introduction to Lambda II: Working with Files in S3 and CloudWatch Events
- [https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462\\_562\\_f2024\\_tutorial\\_5.pdf](https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_tutorial_5.pdf)
- Customize the Request object (add getters/setters)
  - Why do this instead of HashMap ?
- Import dependencies (jar files) into project for AWS S3
- Create an S3 Bucket
- Give your Lambda function(s) permission to work with S3
- Write to the CloudWatch logs
- Use of CloudTrail to generate S3 events
- Creating CloudWatch rule to capture events from CloudTrail
- Have the CloudWatch rule trigger a target Lambda function with a static JSON input object (hard-coded filename)
- **Optional:** for the S3 PutObject event, dynamically extract the name of the file put to the S3 bucket for processing

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.12
-------------------	---	--------

12

## TUTORIAL 6 – NOV 23

- Introduction to Lambda III: Serverless Databases
- [https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462\\_562\\_f2024\\_tutorial\\_6.pdf](https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_tutorial_6.pdf)
- Create and use Sqlite databases using sqlite3 tool
- Deploy Lambda function with Sqlite3 database under /tmp
- Compare in-memory vs. file-based Sqlite DBs on Lambda
- Create an Amazon Aurora “Serverless” v2 MySQL database
- Using an ec2 instance in the same VPC (Region + availability zone) connect and interact with the database using the mysql CLI app
- Deploy an AWS Lambda function that uses the MySQL “serverless” database

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.13
-------------------	---	--------

13

## OBJECTIVES – 11/14

- OBJECTIVES – 11/12
- Tutorials Questions
- **Class Presentations Schedule -  
Cloud Technology or Research Paper Review**
- Quiz 1
- Tutorial 7
- Containerization
- Container Profiler

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.14
-------------------	---	--------

14

## GROUP PRESENTATION

- **TWO OPTIONS:**
- **Cloud technology presentation**
- **Cloud research paper presentation**
  - Recent & suggested papers will be posted at:  
<http://faculty.washington.edu/wlloyd/courses/tcss562/papers/>
- **Submit presentation type and topics (paper or technology) with desired dates of presentation via Canvas by: Sunday November 17<sup>th</sup> @ 11:59pm**
- **Presentation dates:**
  - Tuesday November 26
  - Tuesday December 3\*, Thursday December 5
    - \* - day of quiz 2. only 1 presentation slot

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.15
-------------------	---	--------

15

## OBJECTIVES - 11/14

- **OBJECTIVES - 11/12**
- **Tutorials Questions**
- **Class Presentations Schedule - Cloud Technology or Research Paper Review**
- **Quiz 1**
- **Tutorial 7**
- **Containerization**
- **Container Profiler**

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.16
-------------------	---	--------

16



## OBJECTIVES - 11/14


- OBJECTIVES - 11/12
- Tutorials Questions
- Class Presentations Schedule -  
Cloud Technology or Research Paper Review
- Quiz 1
- **Tutorial 7**
- Containerization
- Container Profiler

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.19
-------------------	---	--------

19

# TUTORIAL #7

## DOCKER, CGROUPS, RESOURCE ISOLATION



November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.20
-------------------	---	--------

20

## TUTORIAL 7 – DEC 1

- Introduction to Docker
- [https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462\\_562\\_f2024\\_tutorial\\_7.pdf](https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2024_tutorial_7.pdf)
- Complete tutorial using Ubuntu 24.04 (for cgroups v2)
- Complete using **c6i.large ec2 instance** (for consistency)
- Use DOCX file for copying and pasting Docker install commands
- Topics:
  - Installing Docker
  - Creating a container using a Dockerfile
  - Using cgroups virtual filesystem to monitor CPU utilization of a container
  - Persisting container images to Docker Hub image repository
  - Container vertical scaling of CPU/memory resources
  - Testing container CPU and memory isolation

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.21
-------------------	---	--------

21

## TUTORIAL COVERAGE

- Docker CLI → Docker Engine (dockerd) → containerd → runc
- Working with the docker CLI:
  - `docker run`            create a container
  - `docker ps -a`        list containers, find CONTAINER ID
  - `docker exec --it`    run a process in an existing container
  - `docker stop`        stop a container
  - `docker kill`        kill a container
  - `docker help`        list available commands
  - `man docker`        Docker Linux manual pages

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.22
-------------------	---	--------

22

```
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
```

## Docker CLI

23

# TUTORIAL 7

- Tutorial introduces use of two common Linux performance benchmark applications
  - stress-ng
  - 100s of CPU, memory, disk, network stress tests
- Sysbench
  - Used in tutorial for memory stress test

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.24
-------------------	---	--------


24

**WE WILL RETURN AT  
~4:50 PM**



25

**CONTAINERIZATION**



November 14, 2024

TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L15.26

26

## OBJECTIVES - 11/14

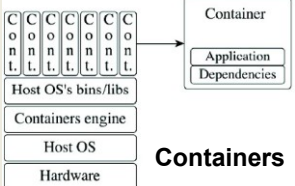
- OBJECTIVES - 11/12
- Tutorials Questions
- Class Presentations Schedule -  
Cloud Technology or Research Paper Review
- Quiz 1
- Tutorial 7
- **Containerization**
- Container Profiler

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.27
-------------------	---	--------

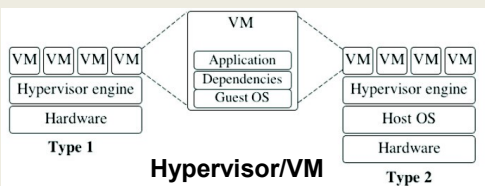
27

## MOTIVATION FOR CONTAINERIZATION

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



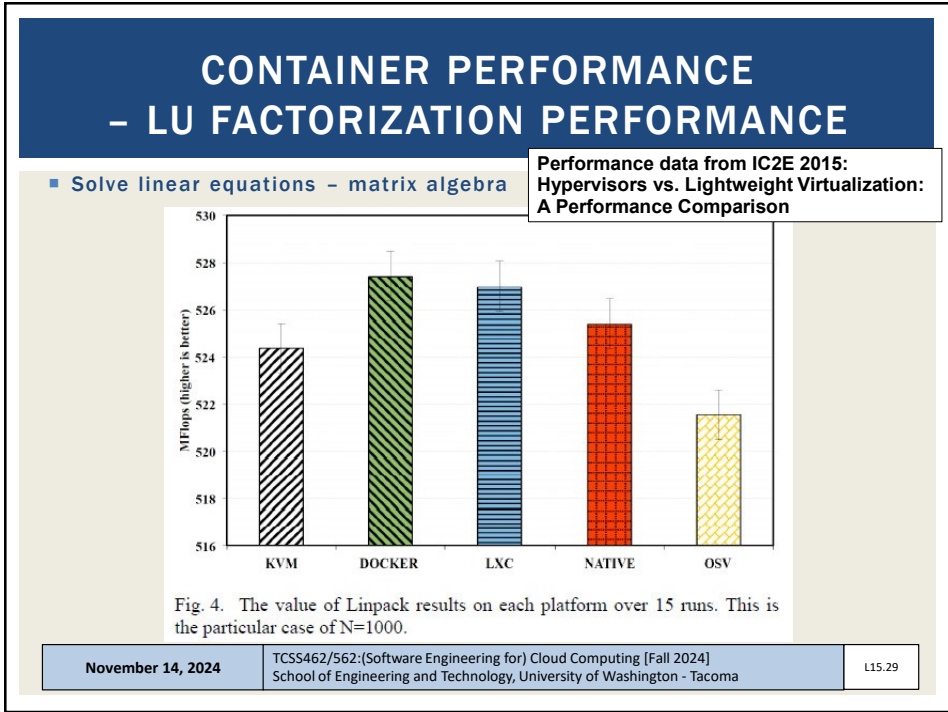
The diagram shows a stack of layers: Hardware at the bottom, then Host OS, then Containers engine, then Host OS's bins/libs, and finally four individual Containers. Each container contains an Application and Dependencies. An arrow points from the Containers layer to a larger box labeled 'Container'.



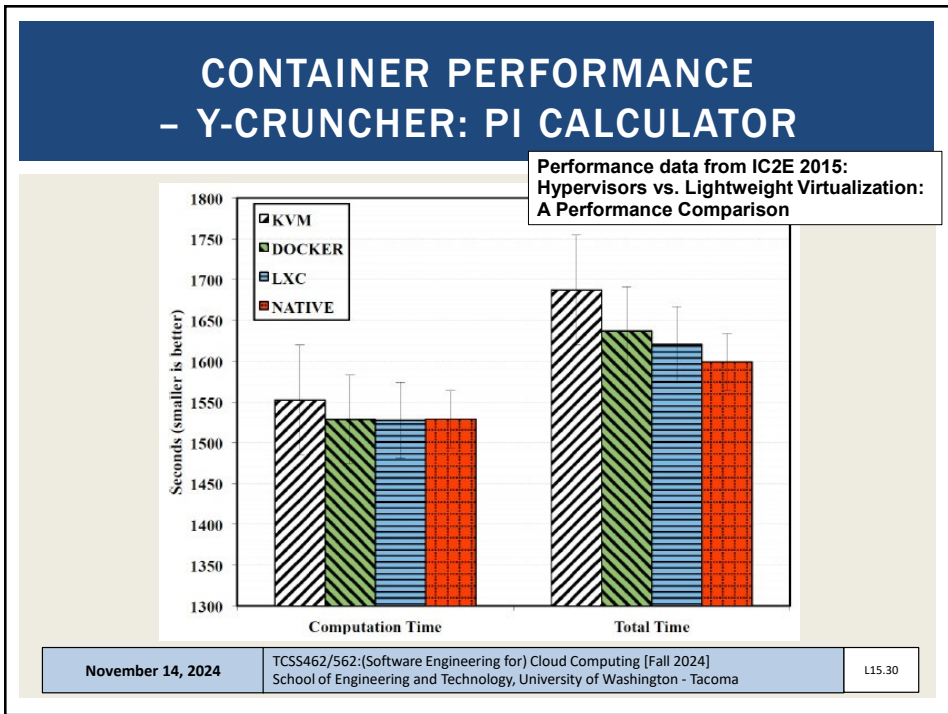
The diagram shows two types of virtualization. Type 1 (Bare Metal) has Hardware at the bottom, then Hypervisor engine, then four VMs. Each VM contains an Application, Dependencies, and Guest OS. Type 2 (Hosted) has Hardware at the bottom, then Host OS, then Hypervisor engine, then four VMs. Each VM contains an Application, Dependencies, and Guest OS. Dashed lines connect the VMs in Type 1 to those in Type 2.

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.28
-------------------	---	--------

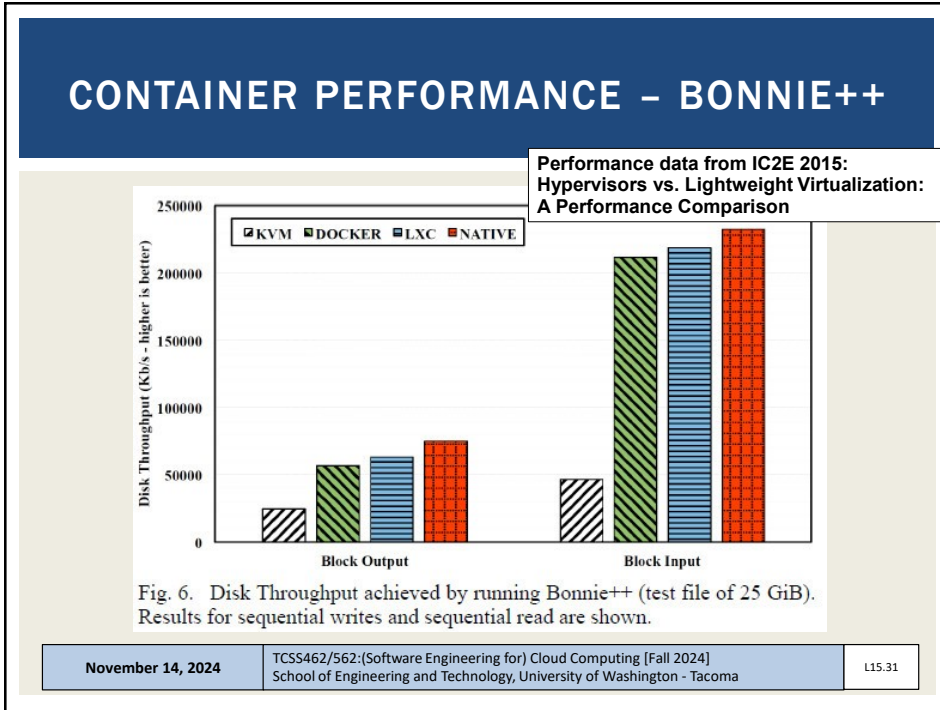
28



29



30



31

## 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 14, 2024    TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma    L15.32

32

## 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**

The diagram illustrates two scenarios for operating system containers. On the left, labeled 'Identical OS containers', a 'Host OS' (blue box) contains three 'Ubuntu 14.04 Container' (green boxes) and one 'Ubuntu 14.04 image' (red box) at the bottom. On the right, labeled 'Different flavoured OS containers', a 'Host OS' (blue box) contains three containers: 'Ubuntu 14.04 Container' (green), 'RHEL 7 Container' (green), and 'CentOS 6.6 Container' (green). Below the host OS are three separate image layers: 'CentOS 6.6 image' (red), 'RHEL 7 image' (red), and 'Ubuntu 14.04 image' (red).

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

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.33
-------------------	---	--------

33

## 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 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.34
-------------------	---	--------

34



## 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

The diagram illustrates the layered structure of container images. It shows a stack of four blue rectangular blocks. From bottom to top, they are labeled: 'Base Image', 'add emacs', 'add Apache', and 'writable Container'. The 'Base Image' block is also labeled 'Debian' on its left side. Below this stack is a larger, light blue block labeled 'Kernel'. An arrow points from the 'Base Image' block to the text 'references parent image'.

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.35
-------------------	---	--------

35

## 2016 DOCKER SURVEY

- Docker application containers
  - Leading containerization vehicle

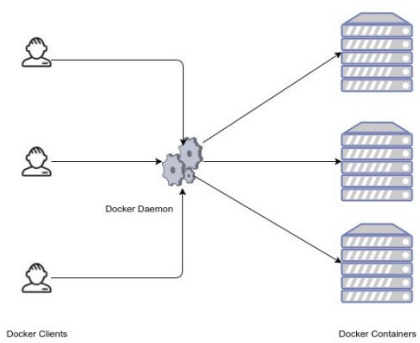
The infographic features a blue whale carrying a stack of blue containers on its back. To the right, three clouds also carry stacks of containers, with arrows indicating movement between them. The statistics are: 80% say Docker is part of cloud strategy; 60% plan to use Docker to migrate workloads to cloud; 41% want application portability across environments; 35+% want to avoid cloud vendor lock-in. The Docker logo is in the bottom right corner.

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.36
-------------------	---	--------

36

# DOCKER

- Docker daemon “dockerd”
  - Implements docker engine that interprets CLI requests and creates/manages containers using backend layered Docker architecture
- Starting in 2017 version numbering switches from 1.x to YR.x
- 2017 releases: 17.03 – 17.12
- 2018 releases: 18.01 – 18.09
- 2019 releases: 19.03.0 – 19.03.13



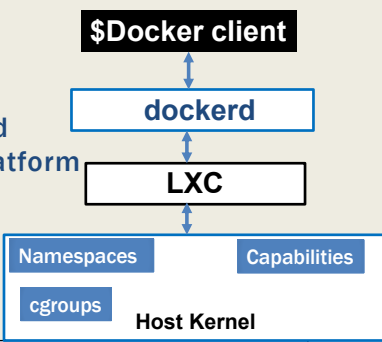
■ Credit: <https://hackernoon.com/docker-containerd-standalone-runtimes>

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.37
-------------------	---	--------

37

# ORIGINAL DOCKER ENGINE IMPLEMENTATION

- (1) Original Docker engine relied on LXC
  - LXC itself is a containerization tool predating Docker
  - Original Docker API just called it
  - LXC originally provided access to Linux kernel features: namespaces and cgroups
  - LXC was Linux specific – caused issues if wanting to be multi-platform
  - Docker implemented their own replacement for LXC



November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.38
-------------------	---	--------

38

## INTRODUCTION OF LIBCONTAINER

- Docker v0.9: **libcontainer** introduced (~2014) to replace LXC as the default Docker daemon

```
graph TD; Client["$Docker client"] <--> dockerd; dockerd <--> libcontainer; libcontainer <--> HostKernel; subgraph HostKernel; Namespaces; Capabilities; cgroups; end;
```

The diagram illustrates the Docker architecture. At the top is the '\$Docker client' (in a black box), which communicates with 'dockerd' (in a white box). 'dockerd' then communicates with 'libcontainer' (in a white box). 'libcontainer' interacts with the 'Host Kernel', which contains 'Namespaces', 'Capabilities', and 'cgroups' (all in blue boxes).

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.39
-------------------	---	--------

39

## OPEN CONTAINER INITIATIVE (OCI)

- OCI created container standards for:
  - Image specification
  - Container runtime specification
- Docker 1.1 (2016): Docker refactored the docker engine to be compliant with OCI standards
  - Essentially this introduced abstraction layers (i.e. generic interfaces that map to the implementation) so that Docker's design conformed to the OCI standard
- **Runc** was added to implement the OCI container runtime spec
  - Provides small, lightweight wrapper for libcontainer
  - Can build and run OCI compliant containers directly using runc provided in Docker, but it is "bare bones" and low-level.
    - The Docker API is much more user friendly
- Support for OCI compliant images was added to **Containerd**

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.40
-------------------	---	--------

40

## CREATING A CONTAINER

**\$ docker run -it --rm tcss558client sh** \$ Docker client

- Docker CLI posts request to **Docker daemon**
- Daemon calls **containerd**
- **Containerd** passes of request to **runc**
  - **Containerd** converts docker image into OCI compliant bundle
  - This step would allow any OCI compliant container to be plugged into the back-end
- **Runc** interfaces with the Linux kernel (namespaces, cgroups, etc.) to create container
- **Shim**: once a container is created, runc exits
  - Shim remains as a daemonless stub to implement the container
  - Allows Docker to be upgraded w/o stopping the container !!!

```

            graph TD
                Client["$ Docker client"] <--> dockerd
                dockerd <--> containerd
                containerd <--> shim
                shim <--> runc
                runc <--> HostKernel
                subgraph HostKernel
                    Namespaces
                    Capabilities
                    cgroups
                end
            
```

November 14, 2024
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma
L15.41

41

## CREATING A CONTAINER - 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 14, 2024
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma
L15.42

42

## SUPPORT FOR ALTERNATE CONTAINER RUNTIMES

- Modularity of Docker implementation supports “execution drivers concept”:
- Enables docker to support many alternate container backends
- OpenVZ, system-nspawn, libvirt-lxc, libvirt-sandbox, qemu/kvm, BSD Jails, Solaris Zones, and chroot

```
graph TD; Docker --> libcontainer; Docker --> libvirt; Docker --> lxc; Docker --> systemd_nspawn; libcontainer --> Linux; libvirt --> Linux; lxc --> Linux; systemd_nspawn --> Linux; subgraph Linux; cgroups; namespaces; netlink; selinux; netfilter; capabilities; apparmor; end;
```

November 14, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] | L15.43  
School of Engineering and Technology, University of Washington - Tacoma

43

## LINUX KERNEL NAMESPACES

- 7 different namespaces in Linux (cgroups not shown)
  - pid, mnt, ipc, user, net, UTS
- Partitions kernel resources

pid	mnt	
	ipc	
	user	net
	UTS	

November 14, 2024 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] | L15.44  
School of Engineering and Technology, University of Washington - Tacoma

44

## NAMESPACES - 2

```

root@93bfc3df0c3e: /
top - 08:34:29 up 6:24, 0 users, load average: 0.00, 0.00, 0.00
Tasks: 4 total, 1 running, 3 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 3853100 total, 2798844 free, 157568 used, 896688 buff/cache
KiB Swap: 0 total, 0 free, 0 used, 3500784 avail Mem

  PID USER   PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
    1 root    20   0  18376 3032 2780  S   0.0   0.1   0:00.02  entrypoint_te+
    5 root    20   0   4532   764   784  S   0.0   0.0   0:00.00  sleep
    6 root    20   0  18508 3476 3064  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 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.45
-------------------	---	--------

45

## LINUX KERNEL NAMESPACES - 3

pid	mnt	
	ipc	
	user	net
	UTS	

- Processes see only their set of resources
- Provides isolation
- Namespaces are hierarchical
- Parent processes can see down the hierarchy
- Each process can only see resources associated with the namespace, and descendent namespaces

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.46
-------------------	---	--------

46

# 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 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.47
-------------------	---	--------

47

# 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
cpuset	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 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.48
-------------------	---	--------

48

# OVERLAY FILE SYSTEMS

- Docker leverages overlay filesystems
- 1<sup>st</sup>: 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 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.49
-------------------	---	--------

49

# LAYERED FS: BUILDING A CONTAINER

- **Dockerfile:**


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

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.50
-------------------	---	--------

50

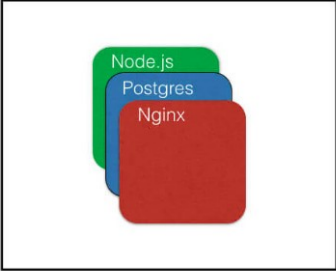


## THREE-TIER ARCHITECTURE



**OS containers**

- 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



**App containers**

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

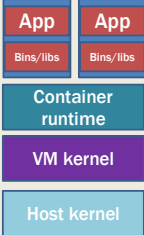
November 14, 2024
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma
L15.51

51

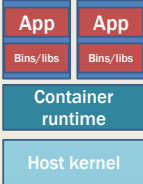
## CONTAINER ISOLATION

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

**Application containers**



**Application containers**



November 14, 2024
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma
L15.52

52

## 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 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.53
-------------------	---	--------

53

## 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

```
graph TD;
    DE[Docker Engine] --- C[containerd];
    C --- CS1[containerd-shim];
    C --- CS2[containerd-shim];
    C --- CS3[...];
    CS1 --- R1[runC];
    CS2 --- R2[runC];
    CS3 --- R3[...];
```

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.54
-------------------	---	--------

54

## 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 14, 2024

TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L15.55

55

## 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 14, 2024

TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L15.56

56

## 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 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.57
-------------------	---	--------

57

## OBJECTIVES - 11/14


- OBJECTIVES - 11/12
- Tutorials Questions
- Class Presentations Schedule -  
Cloud Technology or Research Paper Review
- Quiz 1
- Tutorial 7
- Containerization
- **Container Profiler**

November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.58
-------------------	---	--------

58

<https://github.com/wlloyduw/ContainerProfiler>

# CONTAINER PROFILER



**OXFORD** (GIGA)<sup>n</sup> SCIENCE

GigaScience, 2023, 12, 1–11  
DOI: 10.1093/gigascience/giad069  
Tech Note

## Container Profiler: Profiling resource utilization of containerized big data pipelines

Varik Hoang<sup>1</sup>, Ling-Hong Hung<sup>1</sup>, David Perez, Huazeng Deng, Raymond Schooley<sup>1</sup>, Niharika Arumilli, Ka Yee Yeung<sup>1</sup> and Wes Lloyd<sup>1\*</sup>

School of Engineering and Technology, University of Washington, Tacoma, WA 98402, USA  
\*Correspondence address. Wes Lloyd, 1900 Commerce St #358426, Tacoma, WA 98402, USA. E-mail: [wlloyd@uw.edu](mailto:wlloyd@uw.edu)  
<sup>1</sup> Contributed equally.

**Abstract**  
**Background:** This article presents the Container Profiler, a software tool that measures and records the resource usage of any container.

Downloaded from https://ac

59

# CONTAINER PROFILER

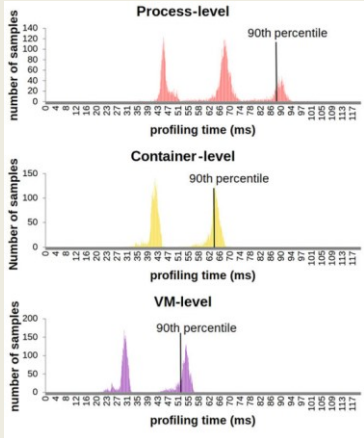
- Captures resource utilization metrics for containers
- Profiles CPU, memory, disk, and network utilization collecting over 60 metrics available from the Linux OS
- Supports two types of profiling
  - **Δ “Delta” Resource Utilization:** Records and calculates total resource utilization from when an initial selection is provided before implementation is verified.
  - **Time series sampling:** supports a configurable sampling interval for continuous monitoring of resources consumed by containers
- Similar profiling techniques compared to SAAF
- Uses Linux proc filesystem “man procs”
- Implemented with a combination of custom code and the Python-based **psutil** library to obtain resource utilization data rapidly

November 23, 2016	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L19.60
-------------------	---	--------

60

## CONTAINER PROFILER: PROFILING OVERHEAD

- Profiling overhead (9,000 samples):
- Use case: RNA-sequencing data processing pipeline (containerized)
- Hardware: IBM Cloud dual bx2d metal 96 vCPUs processors, 384 GB RAM
  - **Process-level:** 3 peaks indicate different behavior presumably based on the number of processes running inside the container cpulde time.
  - Process level collects and reports all available metrics
- Other Supported Profiling Modes:
  - Container-level profiling
    - Does not collect process-level metrics
    - Faster
  - VM-level profiling:
    - Even faster
    - Only collects host-level metrics

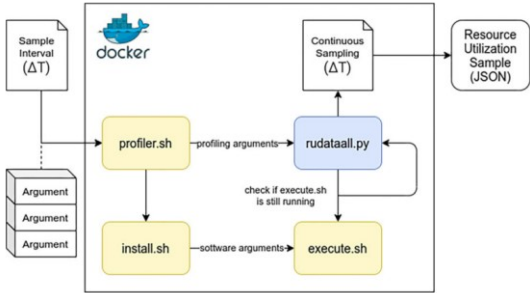


November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.61
-------------------	---	--------

61

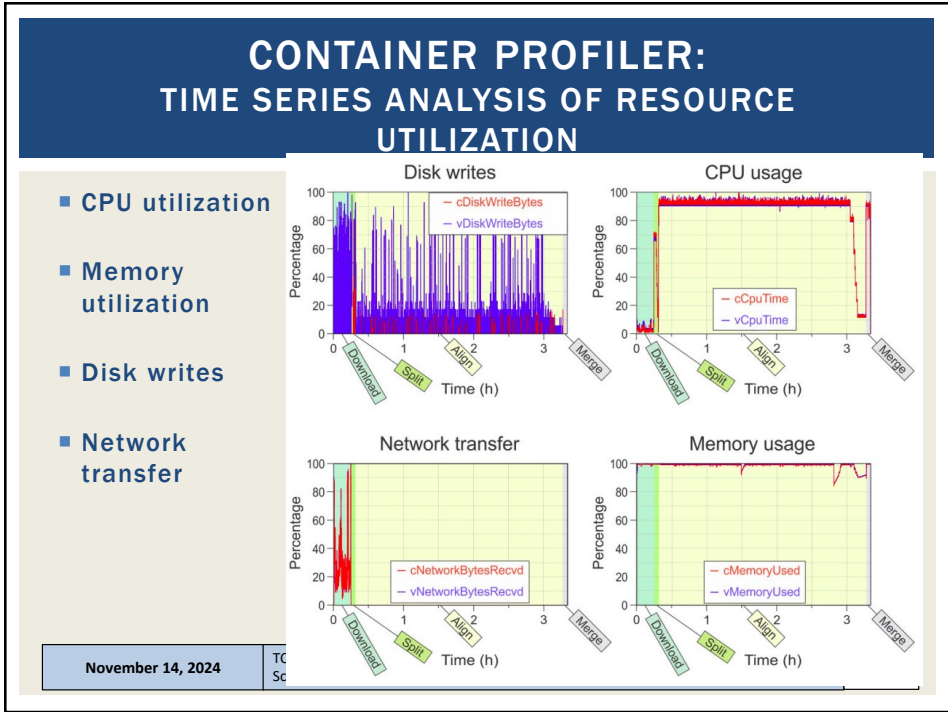
## CONTAINER PROFILER: PROFILING OVERHEAD EXAMPLE

- 99.95% of process level samples were collected under 100ms
- All container-level samples collected under 74ms
- All host (VM-level) samples collected under 60ms
- UMI RNA-sequencing pipeline use case required 2.5 hours to execute with 1-second sampling at full verbosity (all CPU, network I/O, disk I/O, and memory metrics collected)

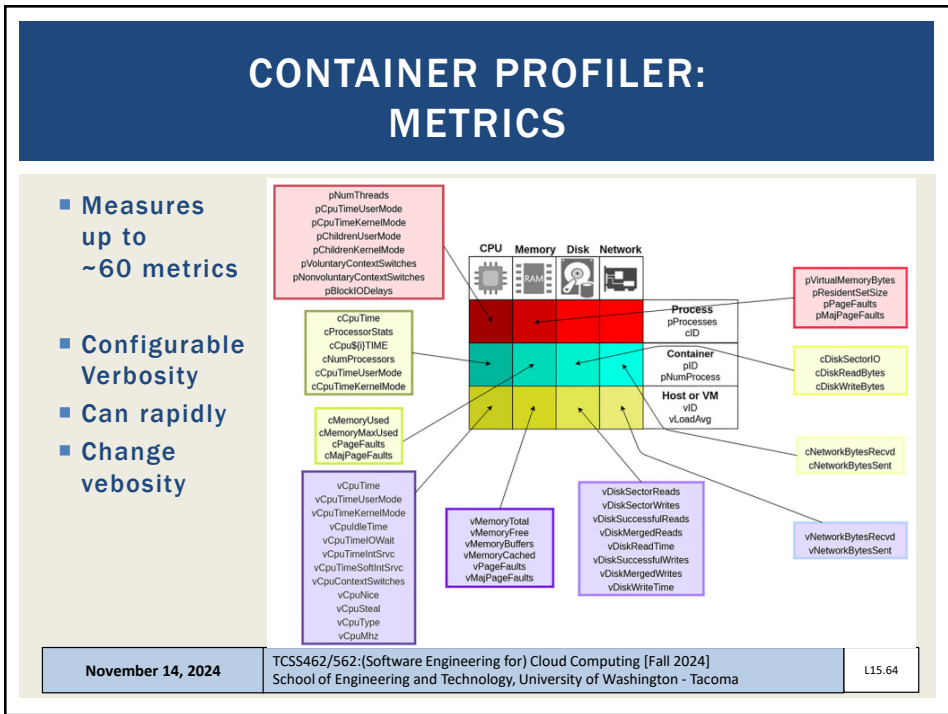


November 14, 2024	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma	L15.62
-------------------	---	--------

62



63



64

## CONTAINER PROFILER: USE CASE

- Profiling overhead for jobs profiled by the ContainerProfiler:
 

Configuration	Overhead (%)
No Profiling	100.00%
VM	100.07%
VM + Container	100.09%
Full Verbosity	100.71%
- Use case: Uniform Molecular Identifier (UMI) RNA-seq pipeline
- Four state: download, split, align, merge

November 14, 2024
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma
L15.65

65

## CONTAINER PROFILER: DELTA PROFILING

- Delta profiling supports viewing total resource utilization (CPU, disk, network) for a containerized task or multi-stage pipeline
- Task = 1 container
- Pipeline = many containers
- Delta captures the perceived system resources used by the task

Task	cpuSftIntSrcv %	cpuloWait %	cpuldle %	cpuKrn %	cpuUsr %
Download	0.46%	0.12%	97.97%	1.06%	0.10%
Split	0.55%	2.18%	68.62%	4.49%	24.10%
Align	0.08%	0.05%	30.30%	0.96%	68.54%
Merge	0.31%	3.55%	29.51%	3.78%	62.07%

- Here is the delta profiling graph for resource utilization:
- Four tasks: Download, Split, Align, Merge
- Graph shows % time in different CPU modes (cpuUsr, cpuKrn, cpuldle, etc.)

November 14, 2024
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma
L15.66

66



**QUESTIONS**

November 14, 2024

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma

L15.67

67