TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Computing Concepts and Models

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

TR 5:50-7:50 PM

1

OFFICE HOURS – FALL 2022

- **THIS WEEK ONLY**
- **Tuesday:**
  - 4:20 to 5:20 pm - CP 229 and Zoom
- **Thursday***
  - 4:20 to 5:20 pm - CP 229 and Zoom
- Or email for appointment

** - Moved from Friday due to faculty meeting*

> Office Hours set based on Student Demographics survey feedback

2

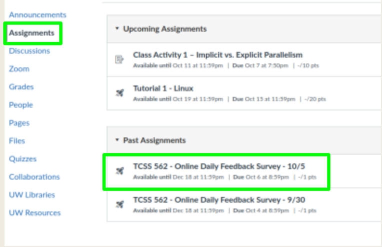
OBJECTIVES – 10/25

- **Questions from 10/20**
- Tutorials Questions
- Tutorial 5 - to be posted...
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Cloud delivery models
  - Cloud deployment models
- AWS Overview and demo
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

3

ONLINE DAILY FEEDBACK SURVEY

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



▾ Upcoming Assignments

Class Activity 1 – Implicit vs. Explicit Parallelism
Available until Oct 11 at 11:59pm | Due Oct 7 at 7:50pm | -/10 pts

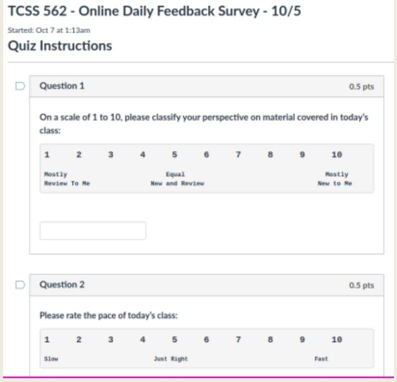
Tutorial 1 - Linux
Available until Oct 19 at 11:59pm | Due Oct 15 at 11:59pm | -/20 pts

▾ 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

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

5

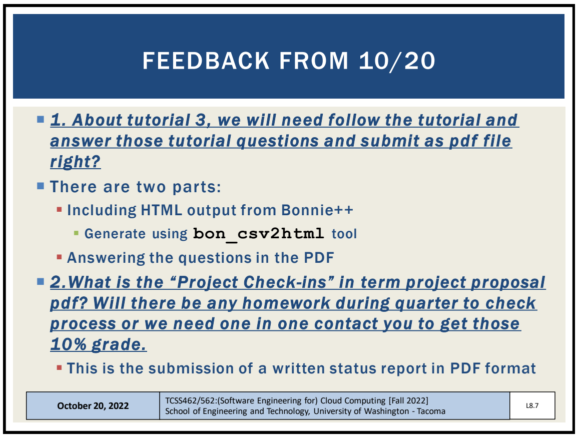
MATERIAL / PACE

- Please classify your perspective on material covered in today's class (**51** respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.54** (↑ - *previous* 6.32)

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.58 (↑ - previous 5.35)**

- **Response rates:**
- TCSS 462: 27/33 – 81.8%
- TCSS 562: 24/26 – 92.3%

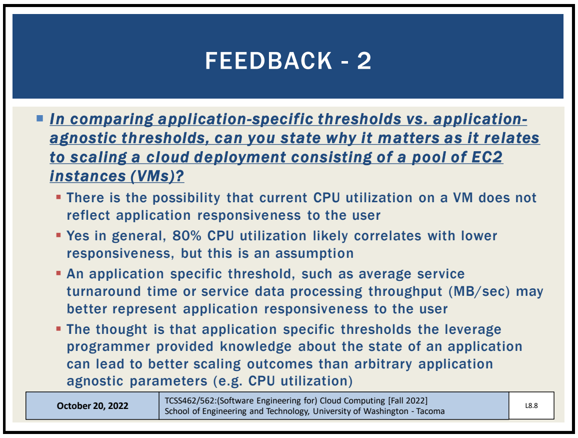
6



FEEDBACK FROM 10/20

- ***1. About tutorial 3, we will need follow the tutorial and answer those tutorial questions and submit as pdf file right?***
- There are two parts:
  - Including HTML output from Bonnie++
    - Generate using `bon_csv2html` tool
  - Answering the questions in the PDF
- ***2.What is the "Project Check-ins" in term project proposal pdf? Will there be any homework during quarter to check process or we need one in one contact you to get those 10% grade.***
  - This is the submission of a written status report in PDF format

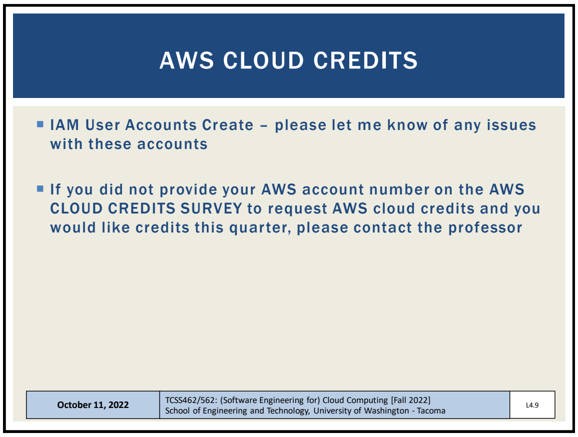
7



FEEDBACK - 2

- ***In comparing application-specific thresholds vs. application-agnostic thresholds, can you state why it matters as it relates to scaling a cloud deployment consisting of a pool of EC2 instances (VMs)?***
  - There is the possibility that current CPU utilization on a VM does not reflect application responsiveness to the user
  - Yes in general, 80% CPU utilization likely correlates with lower responsiveness, but this is an assumption
  - An application specific threshold, such as average service turnaround time or service data processing throughput (MB/sec) may better represent application responsiveness to the user
  - The thought is that application specific thresholds the leverage programmer provided knowledge about the state of an application can lead to better scaling outcomes than arbitrary application agnostic parameters (e.g. CPU utilization)

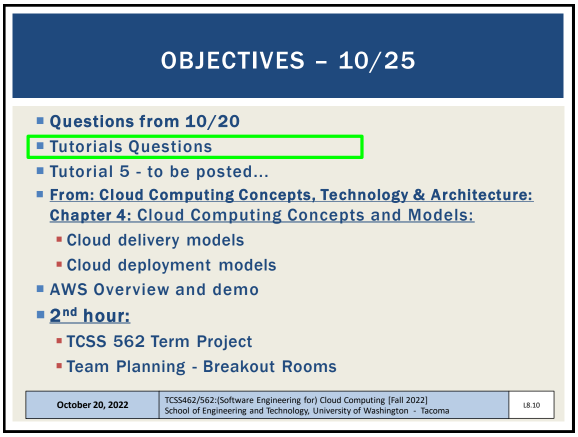
8



AWS CLOUD CREDITS

- IAM User Accounts Create – please let me know of any issues with these accounts
- If you did not provide your AWS account number on the AWS CLOUD CREDITS SURVEY to request AWS cloud credits and you would like credits this quarter, please contact the professor

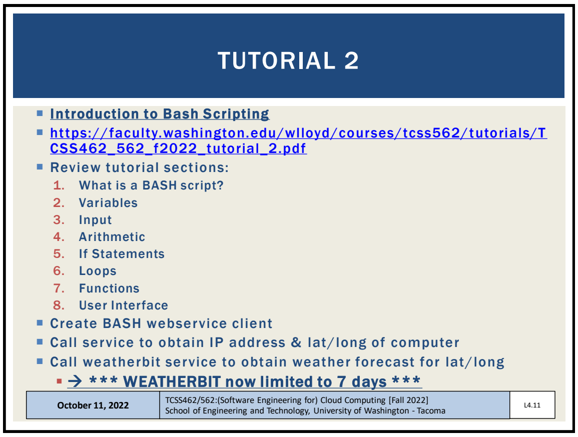
9



OBJECTIVES – 10/25

- **Questions from 10/20**
- Tutorials Questions
- Tutorial 5 - to be posted...
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Cloud delivery models
  - Cloud deployment models
- AWS Overview and demo
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

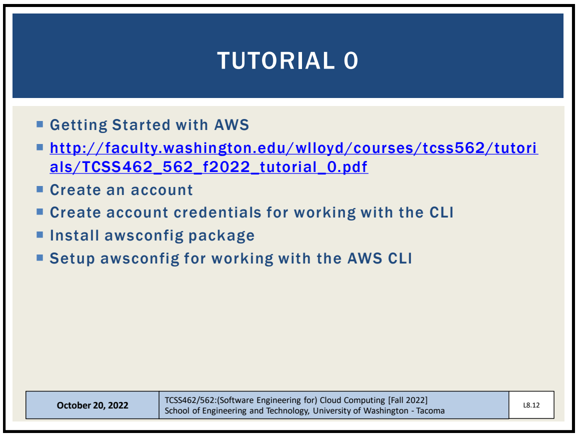
10



TUTORIAL 2

- **Introduction to Bash Scripting**
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_2.pdf
- Review tutorial sections:
  1. What is a BASH script?
  2. Variables
  3. Input
  4. Arithmetic
  5. If Statements
  6. Loops
  7. Functions
  8. User Interface
- Create BASH webservice client
- Call service to obtain IP address & lat/long of computer
- Call weatherbit service to obtain weather forecast for lat/long
  - **→ *** WEATHERBIT now limited to 7 days *****

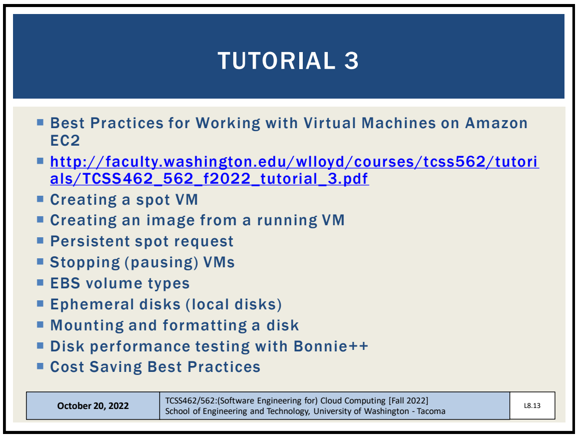
11



TUTORIAL 0

- Getting Started with AWS
- http://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_0.pdf
- Create an account
- Create account credentials for working with the CLI
- Install awsconfig package
- Setup awsconfig for working with the AWS CLI

12

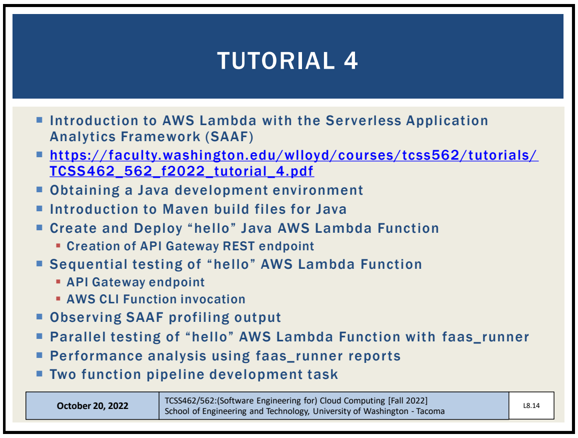


TUTORIAL 3

- Best Practices for Working with Virtual Machines on Amazon EC2
- http://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_3.pdf
- Creating a spot VM
- Creating an image from a running VM
- Persistent spot request
- Stopping (pausing) VMs
- EBS volume types
- Ephemeral disks (local disks)
- Mounting and formatting a disk
- Disk performance testing with Bonnie++
- Cost Saving Best Practices

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma | L8.13

13

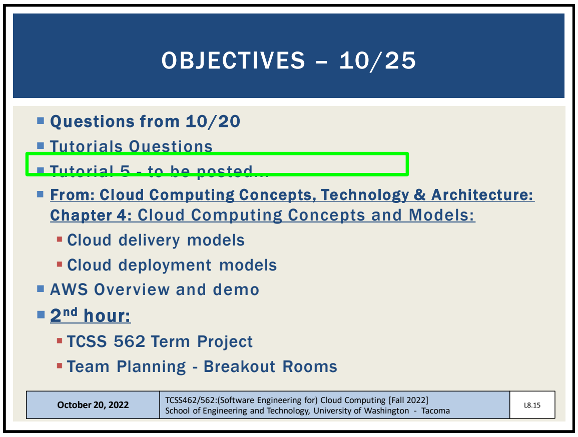


TUTORIAL 4

- Introduction to AWS Lambda with the Serverless Application Analytics Framework (SAAF)
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_4.pdf
- Obtaining a Java development environment
- Introduction to Maven build files for Java
- Create and Deploy “hello” Java AWS Lambda Function
  - Creation of API Gateway REST endpoint
- Sequential testing of “hello” AWS Lambda Function
  - API Gateway endpoint
  - AWS CLI Function invocation
- Observing SAAF profiling output
- Parallel testing of “hello” AWS Lambda Function with faas_runner
- Performance analysis using faas_runner reports
- Two function pipeline development task

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma | L8.14

14

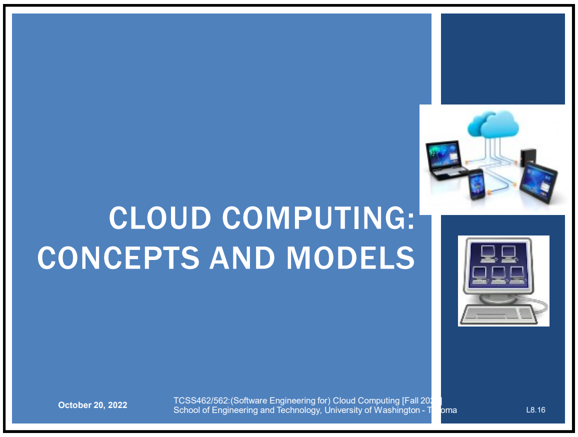


OBJECTIVES – 10/25

- **Questions from 10/20**
- Tutorials Questions
- Tutorial 5 - to be posted...
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Cloud delivery models
  - Cloud deployment models
- AWS Overview and demo
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma | L8.15

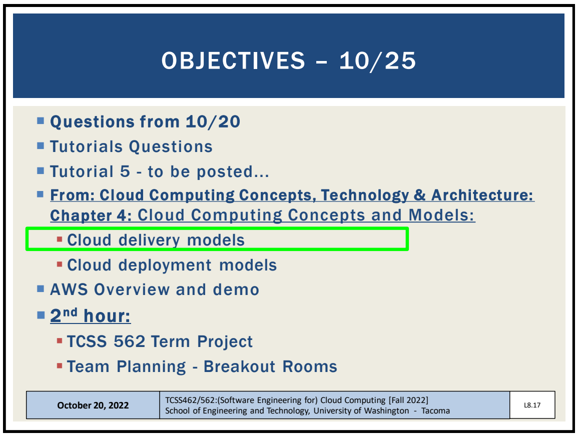
15



CLOUD COMPUTING: CONCEPTS AND MODELS

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma | L8.16

16

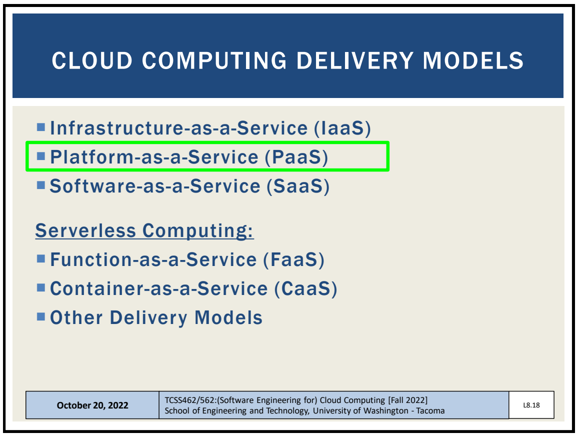


OBJECTIVES – 10/25

- **Questions from 10/20**
- Tutorials Questions
- Tutorial 5 - to be posted...
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Cloud delivery models
  - Cloud deployment models
- AWS Overview and demo
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma | L8.17

17



CLOUD COMPUTING DELIVERY MODELS

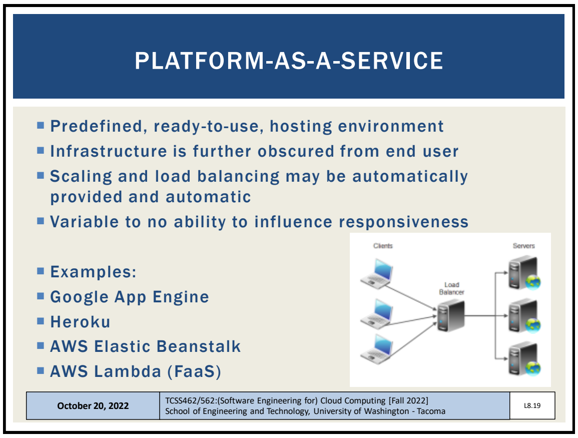
- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Serverless Computing:

- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other Delivery Models

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma | L8.18

18



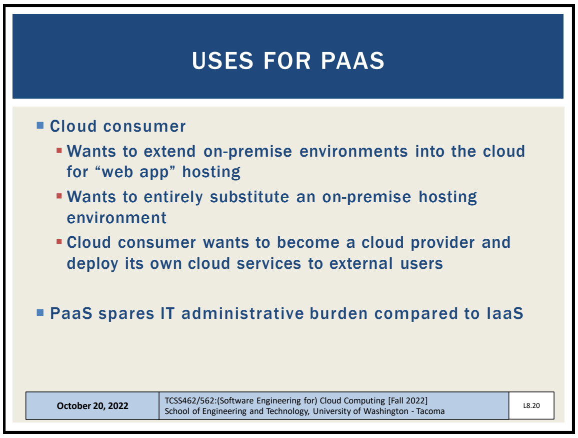
PLATFORM-AS-A-SERVICE

- Predefined, ready-to-use, hosting environment
- Infrastructure is further obscured from end user
- Scaling and load balancing may be automatically provided and automatic
- Variable to no ability to influence responsiveness

- Examples:
- Google App Engine
- Heroku
- AWS Elastic Beanstalk
- AWS Lambda (FaaS)

Clients
Load Balancer
Servers

19

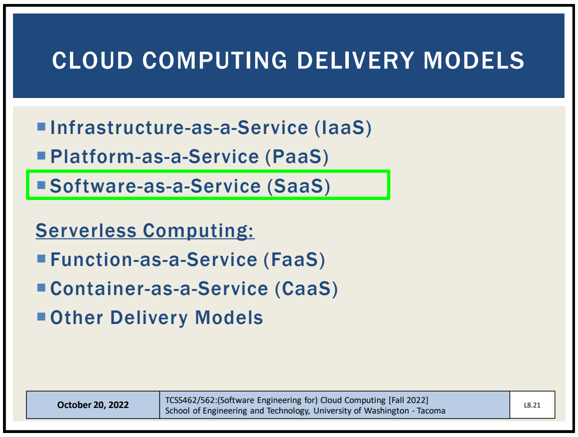


USES FOR PAAS

- Cloud consumer
  - Wants to extend on-premise environments into the cloud for “web app” hosting
  - Wants to entirely substitute an on-premise hosting environment
  - Cloud consumer wants to become a cloud provider and deploy its own cloud services to external users

- PaaS spares IT administrative burden compared to IaaS

20

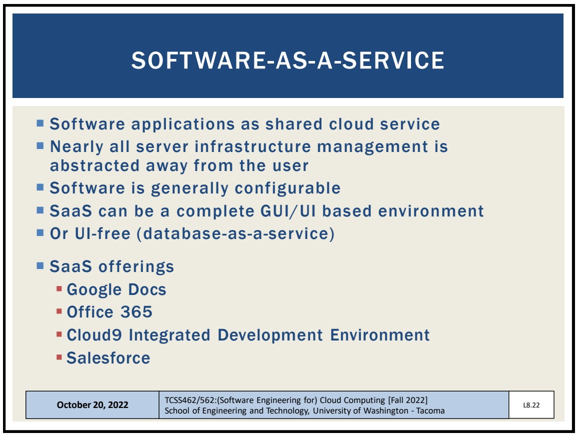


CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Serverless Computing:
- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other Delivery Models

21

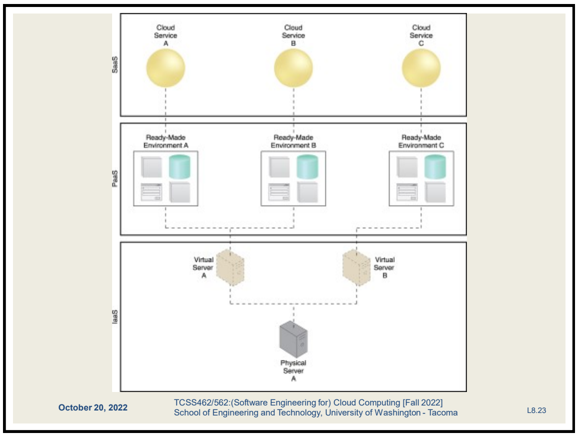


SOFTWARE-AS-A-SERVICE

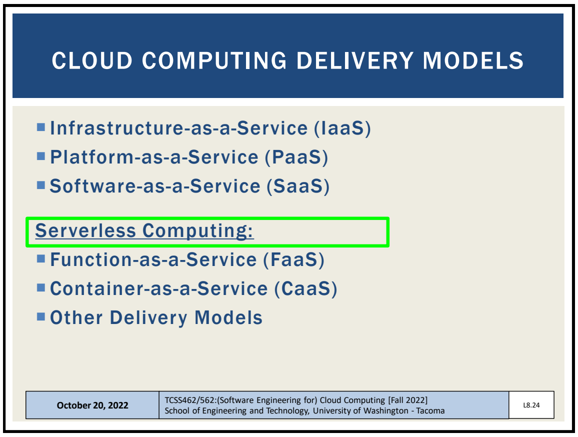
- Software applications as shared cloud service
- Nearly all server infrastructure management is abstracted away from the user
- Software is generally configurable
- SaaS can be a complete GUI/UI based environment
- Or UI-free (database-as-a-service)

- SaaS offerings
  - Google Docs
  - Office 365
  - Cloud9 Integrated Development Environment
  - Salesforce

22



23



CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Serverless Computing:
- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other Delivery Models

24

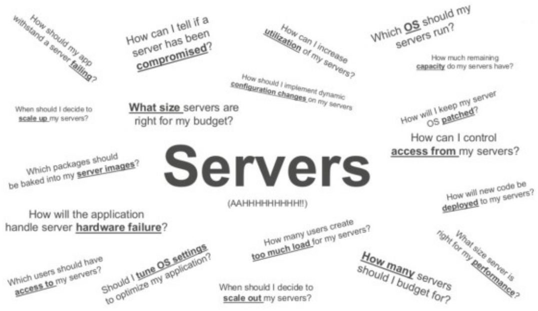
SERVERLESS COMPUTING

Introducing Cloud 2.0

Image from: https://mobisoftinfotech.com/resources/blog/serverless-computing-deploy-applications-without-fiddling-with-servers/

25

SERVERLESS COMPUTING



How should my app withstand a server **failing**?

How can I tell if a server has been **compromised**?

How can I increase **utilization** of my servers?

Which **OS** should my servers run?

How much remaining **capacity** do my servers have?

How should I implement dynamic **configuration changes** on my servers

When should I decide to **scale up** my servers?

What size servers are right for my budget?

How will I keep my server OS **patched**?

How can I control **access from** my servers?

Which packages should be baked into my **server images**?

Servers

(AAHHHHHHHHH!!)

How will new code be **deployed** to my servers?

How will the application handle server **hardware failure**?

How many users create **too much load** for my servers?

What size server is right for my **performance**?

Which users should have **access to** my servers?

Should I **tune OS settings** to optimize my application?

When should I decide to **scale out** my servers?

How many servers should I budget for?

26

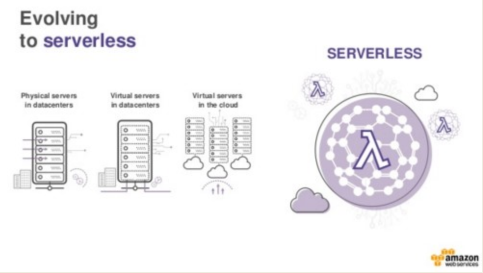
SERVERLESS COMPUTING

What is serverless?

Build and run applications without thinking about servers

27

SERVERLESS COMPUTING - 2

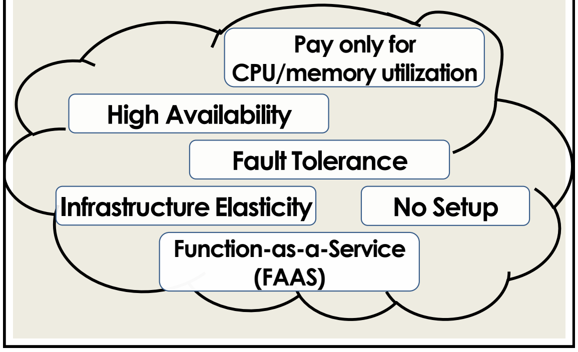


Evolving to serverless

Physical servers in datacenters
Virtual servers in datacenters
Virtual servers in the cloud
SERVERLESS

28

SERVERLESS COMPUTING



Pay only for CPU/memory utilization

High Availability

Fault Tolerance

Infrastructure Elasticity

No Setup

Function-as-a-Service (FAAS)

29

SERVERLESS COMPUTING

Why Serverless Computing?

Many features of distributed systems, that are challenging to deliver, are provided automatically

…they are built into the platform

30

CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Serverless Computing:

- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other Delivery Models

31

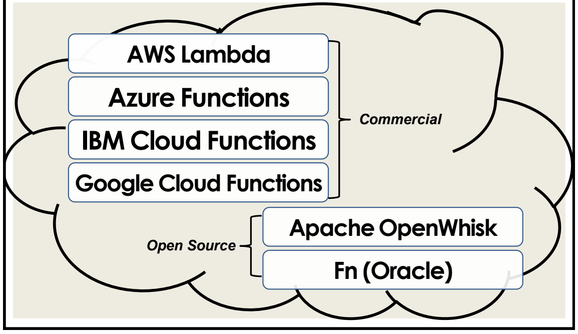
SERVERLESS VS. FAAS

- **Serverless Computing**
- Refers to the avoidance of managing servers
- Can pertain to a number of "as-a-service" cloud offerings
- Function-as-a-Service (FaaS)
  - Developers write small code snippets (microservices) which are deployed separately
- Database-as-a-Service (DBaaS)
- Container-as-a-Service (CaaS)
- Others…

- Serverless is a buzzword
- This space is evolving…

32

FAAS PLATFORMS

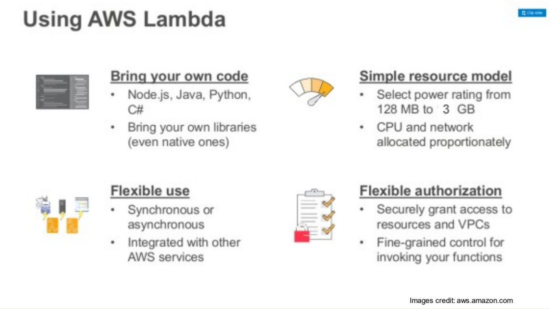


AWS Lambda
Azure Functions
IBM Cloud Functions
Google Cloud Functions
Commercial

Open Source
Apache OpenWhisk
Fn (Oracle)

33

AWS LAMBDA



Using AWS Lambda

Bring your own code
- Node.js, Java, Python, C#
- Bring your own libraries (even native ones)

Simple resource model
- Select power rating from 128 MB to 3 GB
- CPU and network allocated proportionately

Flexible use
- Synchronous or asynchronous
- Integrated with other AWS services

Flexible authorization
- Securely grant access to resources and VPCs
- Fine-grained control for invoking your functions

Images credit: aws.amazon.com

34

FAAS PLATFORMS - 2

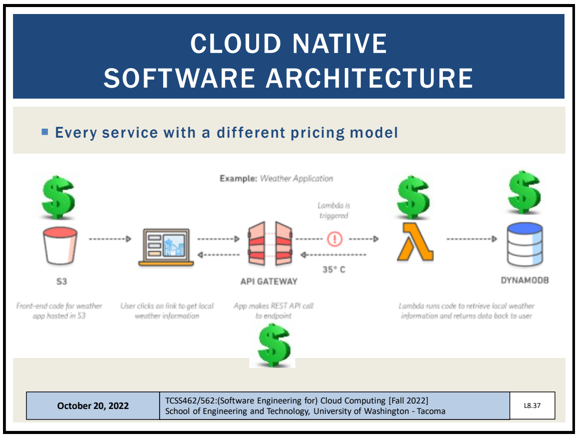
- New cloud platform for hosting application code
- Every cloud vendor provides their own:
  - AWS Lambda, Azure Functions, Google Cloud Functions, IBM OpenWhisk
- Similar to platform-as-a-service
- Replace opensource web container (e.g. Apache Tomcat) with abstracted vendor-provided **black-box** environment

35

FAAS PLATFORMS - 3

- Many challenging features of distributed systems are provided automatically
- ***Built into the platform:***
- Highly availability (24/7)
- Scalability
- Fault tolerance

36

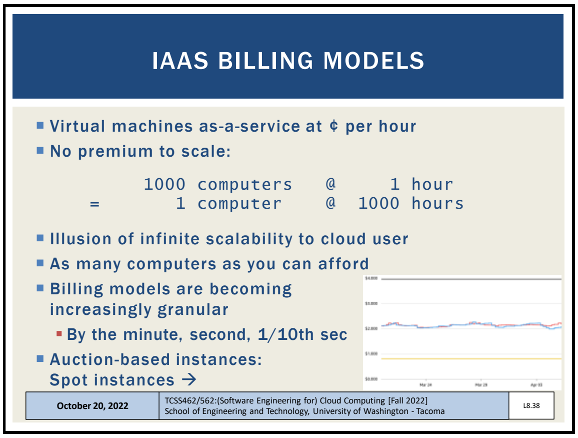


CLOUD NATIVE SOFTWARE ARCHITECTURE

- Every service with a different pricing model

Example: Weather Application
S3
Front-end code for weather app hosted in S3
User clicks on link to get local weather information
API GATEWAY
App makes REST API call to endpoint
Lambda is triggered
35° C
Lambda runs code to retrieve local weather information and returns data back to user
DYNAMODB

37



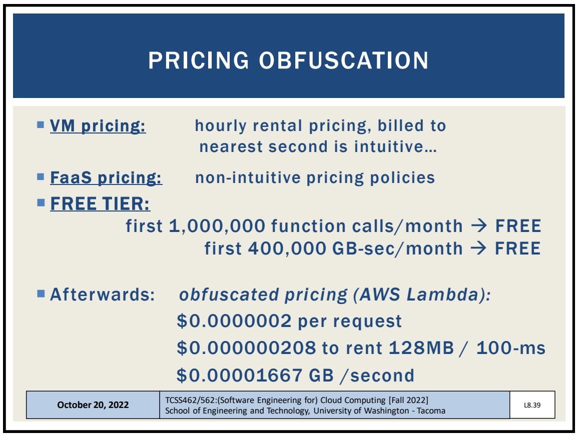
IAAS BILLING MODELS

- Virtual machines as-a-service at ¢ per hour
- No premium to scale:

```
       1000 computers   @     1 hour
  =       1 computer    @  1000 hours
```

- Illusion of infinite scalability to cloud user
- As many computers as you can afford
- Billing models are becoming increasingly granular
  - By the minute, second, 1/10th sec
- Auction-based instances: Spot instances →

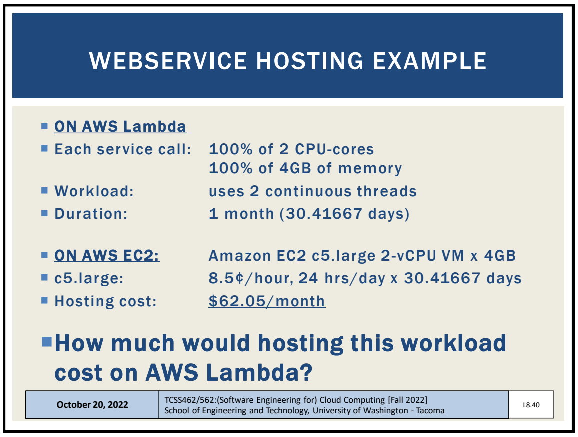
38



PRICING OBFUSCATION

- **VM pricing:** hourly rental pricing, billed to nearest second is intuitive…
- **FaaS pricing:** non-intuitive pricing policies
- **FREE TIER:**
  first 1,000,000 function calls/month → FREE
  first 400,000 GB-sec/month → FREE
- Afterwards: *obfuscated pricing (AWS Lambda):*
  $0.0000002 per request
  $0.000000208 to rent 128MB / 100-ms
  $0.00001667 GB /second

39



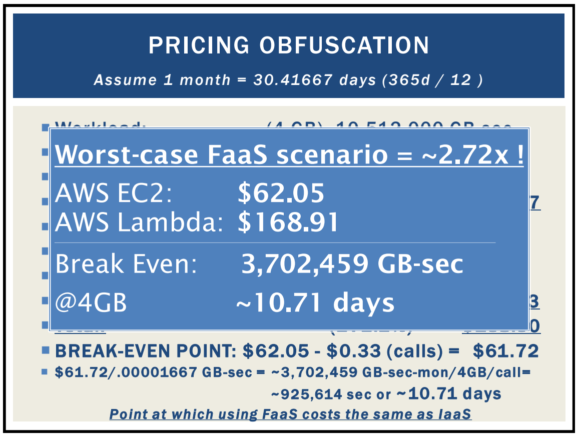
WEBSERVICE HOSTING EXAMPLE

- **ON AWS Lambda**
- Each service call: 100% of 2 CPU-cores
  100% of 4GB of memory
- Workload: uses 2 continuous threads
- Duration: 1 month (30.41667 days)

- **ON AWS EC2:** Amazon EC2 c5.large 2-vCPU VM x 4GB
- c5.large: 8.5¢/hour, 24 hrs/day x 30.41667 days
- Hosting cost: $62.05/month

- **How much would hosting this workload cost on AWS Lambda?**

40



PRICING OBFUSCATION

Assume 1 month = 30.41667 days (365d / 12)

- Workload: (4 GB) 10,512,000 GB-sec

Worst-case FaaS scenario = ~2.72x !

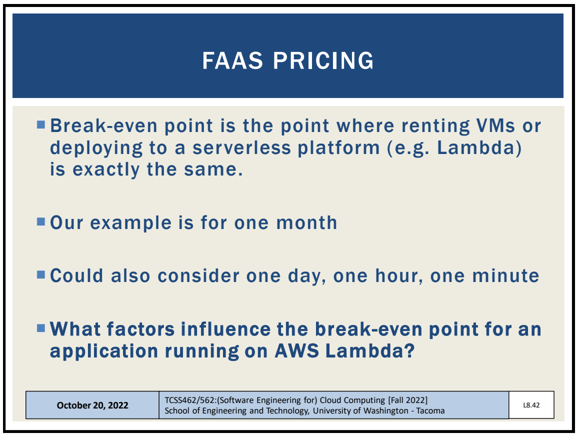
AWS EC2: $62.05
AWS Lambda: $168.91

Break Even: 3,702,459 GB-sec
@4GB ~10.71 days

- **BREAK-EVEN POINT: $62.05 - $0.33 (calls) = $61.72**
- **$61.72/.00001667 GB-sec = ~3,702,459 GB-sec-mon/4GB/call= ~925,614 sec or ~10.71 days**

Point at which using FaaS costs the same as IaaS

41



FAAS PRICING

- Break-even point is the point where renting VMs or deploying to a serverless platform (e.g. Lambda) is exactly the same.
- Our example is for one month
- Could also consider one day, one hour, one minute
- What factors influence the break-even point for an application running on AWS Lambda?

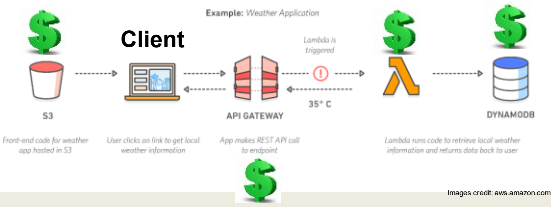
42

FAAS CHALLENGES

- Vendor architectural lock-in – how to migrate?
- Pricing obfuscation – is it cost effective?
- Memory reservation – how much to reserve?
- Service composition – how to compose software?
- Infrastructure freeze/thaw cycle – how to avoid?
- Performance – what will it be?

VENDOR ARCHITECTURAL LOCK-IN

- Cloud native (FaaS) software architecture requires external services/components


Example: Weather Application
$
Client
Lambda is triggered
35° C
S3
API GATEWAY
DYNAMODB
Front-end code for weather app hosted in S3
User clicks on link to get local weather information
App makes REST API call to endpoint
Lambda runs code to retrieve local weather information and returns data back to user

Images credit: aws.amazon.com

- Increased dependencies → increased hosting costs

PRICING OBFUSCATION

- **VM pricing:** hourly rental pricing, billed to nearest second is intuitive…
- **FaaS pricing:**

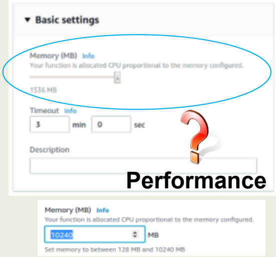
AWS Lambda Pricing

FREE TIER:	first 1,000,000 function calls/month → FREE
	first 400,000 GB-sec/month → FREE

- Afterwards: $0.0000002 per request
  $0.000000208 to rent 128MB / 100-ms

MEMORY RESERVATION QUESTION…

- Lambda memory reserved for functions
- UI provides text box formerly "slider bar" to set function's memory
- Resource capacity (CPU, disk, network) coupled to slider bar: *"every **doubling** of memory, **doubles** CPU…"*



▾ Basic settings

Memory (MB) Info
Your function is allocated CPU proportional to the memory configured.
1536 MB

Timeout Info
3 min 0 sec

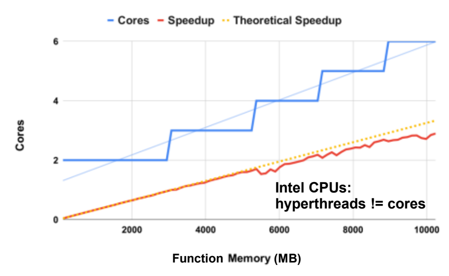
Description

Performance

Memory (MB) Info
Your function is allocated CPU proportional to the memory configured.
10240 MB
Set memory to between 128 MB and 10240 MB

- **But how much memory do FaaS functions require?**

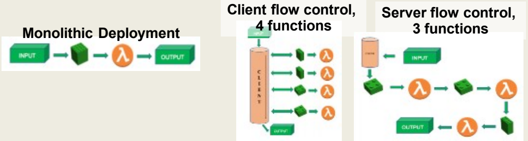
AWS LAMBDA COUPLES FUNCTION MEMORY TO CPU CORES & TIME SHARE



Intel CPUs: hyperthreads != cores

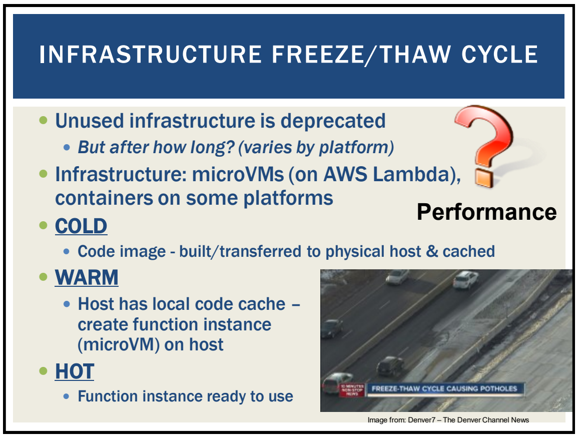
SERVICE COMPOSITION

- **How should application code be composed for deployment to serverless computing platforms?**


Monolithic Deployment
Client flow control, 4 functions
Server flow control, 3 functions
INPUT
OUTPUT
λ
CLIENT

- Recommended practice: Decompose into many microservices
- Platform limits: code + libraries ~250MB
- **How does composition impact the number of function invocations, and memory utilization?**

Performance



INFRASTRUCTURE FREEZE/THAW CYCLE

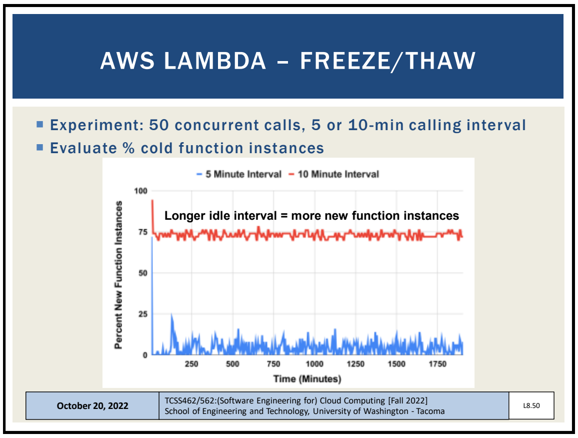
- Unused infrastructure is deprecated
  - *But after how long? (varies by platform)*
- Infrastructure: microVMs (on AWS Lambda), containers on some platforms
- COLD
  - Code image - built/transferred to physical host & cached
- WARM
  - Host has local code cache – create function instance (microVM) on host
- HOT
  - Function instance ready to use

Performance

FREEZE-THAW CYCLE CAUSING POTHOLES

Image from: Denver7 – The Denver Channel News

49

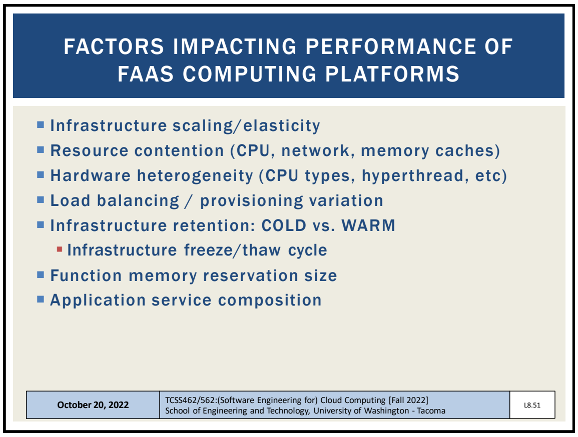


AWS LAMBDA – FREEZE/THAW

- Experiment: 50 concurrent calls, 5 or 10-min calling interval
- Evaluate % cold function instances

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.50

50

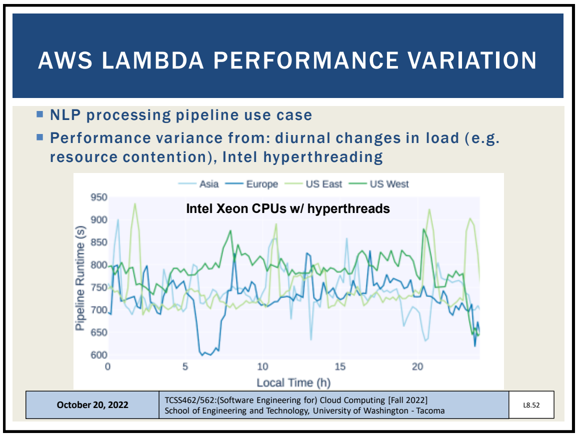


FACTORS IMPACTING PERFORMANCE OF FAAS COMPUTING PLATFORMS

- Infrastructure scaling/elasticity
- Resource contention (CPU, network, memory caches)
- Hardware heterogeneity (CPU types, hyperthread, etc)
- Load balancing / provisioning variation
- Infrastructure retention: COLD vs. WARM
  - Infrastructure freeze/thaw cycle
- Function memory reservation size
- Application service composition

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.51

51

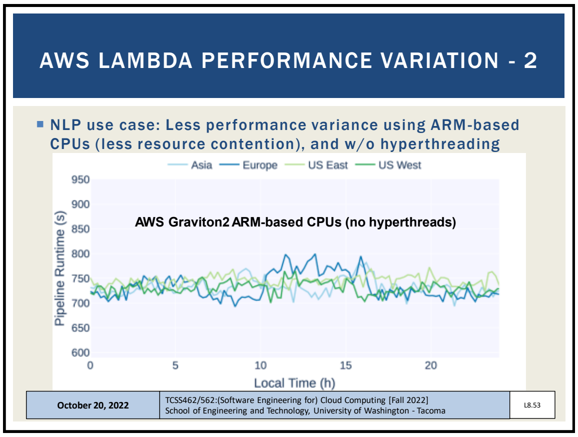


AWS LAMBDA PERFORMANCE VARIATION

- NLP processing pipeline use case
- Performance variance from: diurnal changes in load (e.g. resource contention), Intel hyperthreading

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.52

52

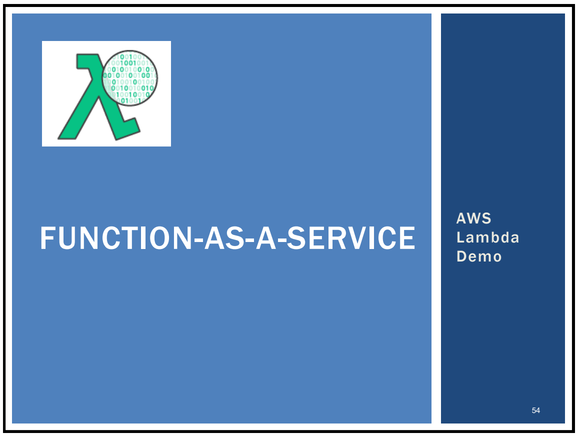


AWS LAMBDA PERFORMANCE VARIATION - 2

- NLP use case: Less performance variance using ARM-based CPUs (less resource contention), and w/o hyperthreading

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.53

53

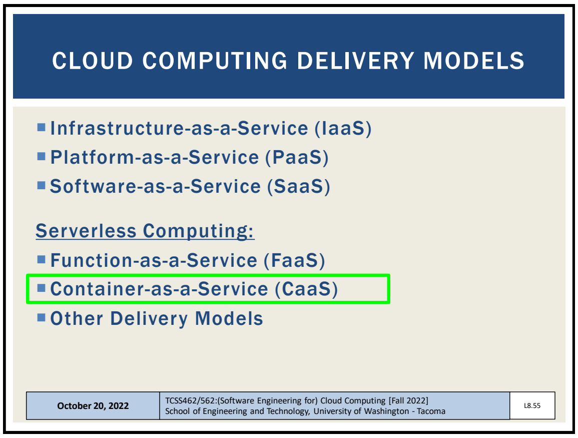


FUNCTION-AS-A-SERVICE

AWS Lambda Demo

54

54



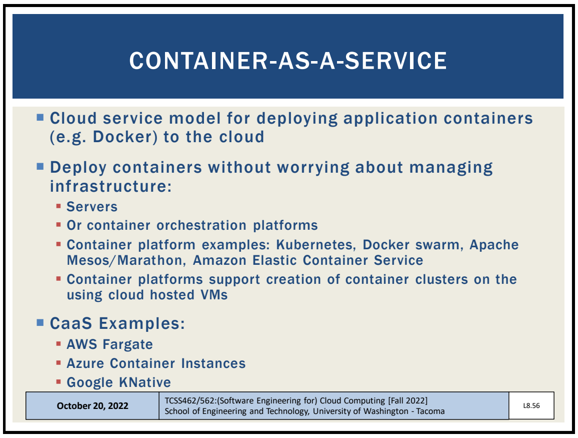
CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Serverless Computing:

- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other Delivery Models

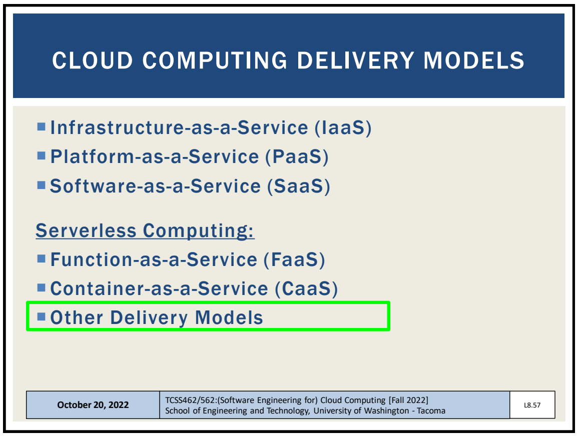
55



CONTAINER-AS-A-SERVICE

- Cloud service model for deploying application containers (e.g. Docker) to the cloud
- Deploy containers without worrying about managing infrastructure:
  - Servers
  - Or container orchestration platforms
  - Container platform examples: Kubernetes, Docker swarm, Apache Mesos/Marathon, Amazon Elastic Container Service
  - Container platforms support creation of container clusters on the using cloud hosted VMs
- CaaS Examples:
  - AWS Fargate
  - Azure Container Instances
  - Google KNative

56



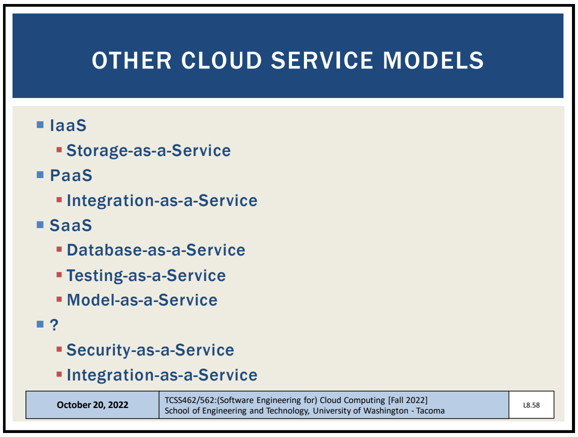
CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Serverless Computing:

- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other Delivery Models

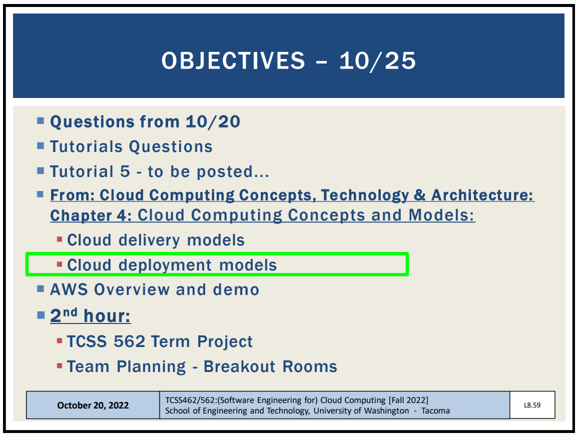
57



OTHER CLOUD SERVICE MODELS

- IaaS
  - Storage-as-a-Service
- PaaS
  - Integration-as-a-Service
- SaaS
  - Database-as-a-Service
  - Testing-as-a-Service
  - Model-as-a-Service
- ?
  - Security-as-a-Service
  - Integration-as-a-Service

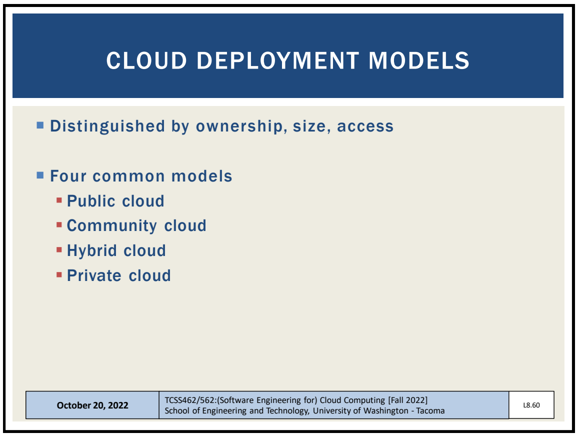
58



OBJECTIVES – 10/25

- **Questions from 10/20**
- Tutorials Questions
- Tutorial 5 - to be posted...
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4:** Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS Overview and demo
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

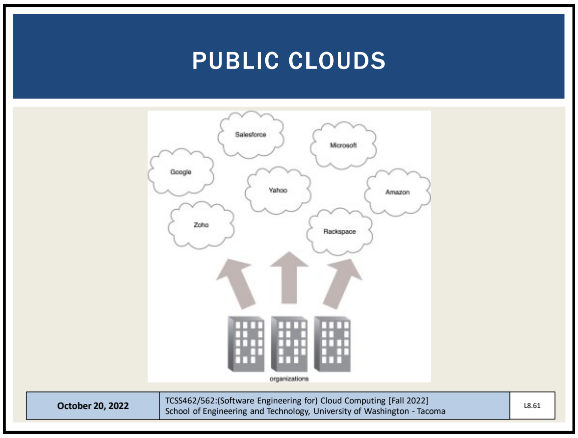
59



CLOUD DEPLOYMENT MODELS

- Distinguished by ownership, size, access
- Four common models
  - Public cloud
  - Community cloud
  - Hybrid cloud
  - Private cloud

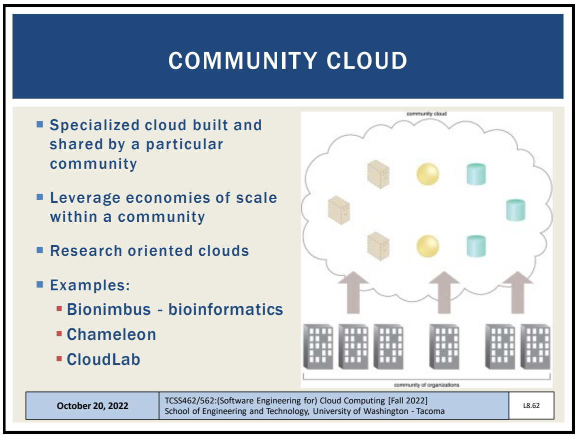
60



PUBLIC CLOUDS

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.61

61

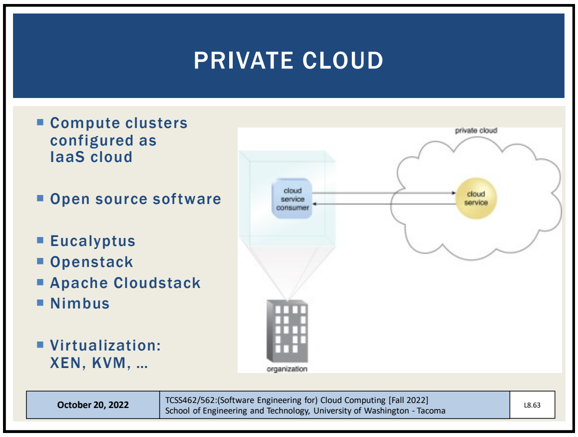


COMMUNITY CLOUD

- Specialized cloud built and shared by a particular community
- Leverage economies of scale within a community
- Research oriented clouds
- Examples:
  - Bionimbus - bioinformatics
  - Chameleon
  - CloudLab

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.62

62

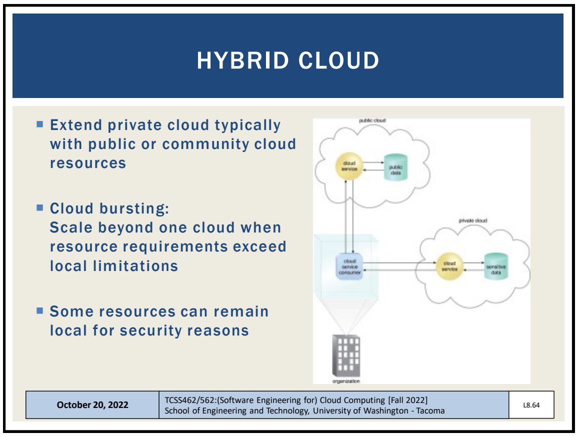


PRIVATE CLOUD

- Compute clusters configured as IaaS cloud
- Open source software
- Eucalyptus
- Openstack
- Apache Cloudstack
- Nimbus
- Virtualization: XEN, KVM, …

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.63

63

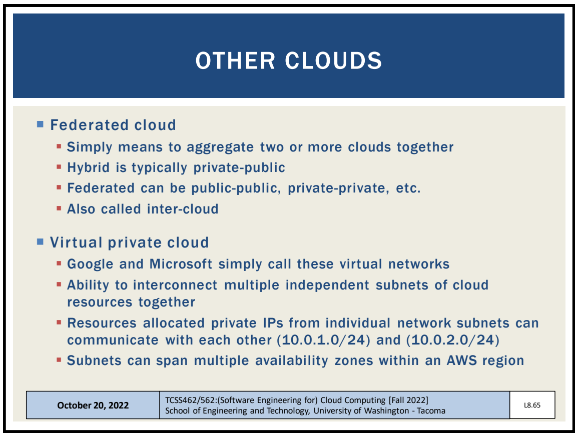


HYBRID CLOUD

- Extend private cloud typically with public or community cloud resources
- Cloud bursting: Scale beyond one cloud when resource requirements exceed local limitations
- Some resources can remain local for security reasons

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.64

64



OTHER CLOUDS

- Federated cloud
  - Simply means to aggregate two or more clouds together
  - Hybrid is typically private-public
  - Federated can be public-public, private-private, etc.
  - Also called inter-cloud
- Virtual private cloud
  - Google and Microsoft simply call these virtual networks
  - Ability to interconnect multiple independent subnets of cloud resources together
  - Resources allocated private IPs from individual network subnets can communicate with each other (10.0.1.0/24) and (10.0.2.0/24)
  - Subnets can span multiple availability zones within an AWS region

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.65

65

WE WILL RETURN AT 7:00 PM

66

OBJECTIVES – 10/25

- **Questions from 10/20**
- Tutorials Questions
- Tutorial 5 - to be posted…
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4:** Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS Overview and demo
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

AWS OVERVIEW AND DEMO

ONLINE CLOUD TUTORIALS

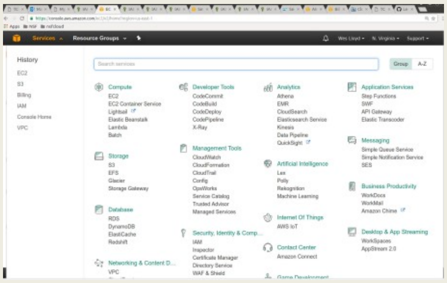
- From the eScience Institute @ UW Seattle:
- https://escience.washington.edu/
- Online cloud workshops
- Introduction to AWS, Azure, and Google Cloud
- Task: Deploying a Python DJANGO web application
- Self-guided workshop materials available online:
- https://cloudmaven.github.io/documentation/

- AWS Educate provides access to many online tutorials / learning resources:
- https://aws.amazon.com/education/awseducate/

LIST OF TOPICS

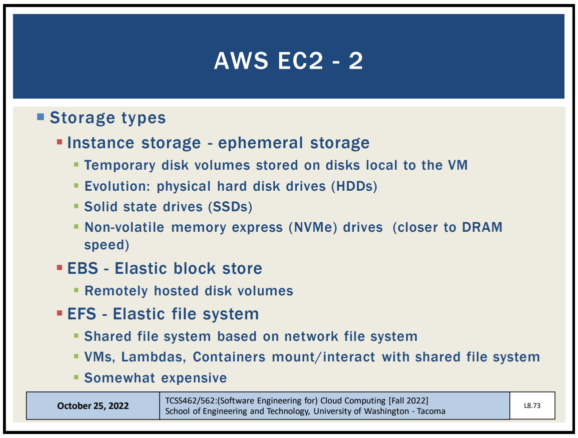
- AWS Management Console
- Elastic Compute Cloud (EC2)
- Instance Storage: Virtual Disks on VMs
- Elastic Block Store: Virtual Disks on VMs
- Elastic File System (EFS)
- Amazon Machine Images (AMIs)
- EC2 Paravirtualization
- EC2 Full Virtualization (hvm)
- EC2 Virtualization Evolution
- (VM) Instance Actions
- EC2 Networking
- EC2 Instance Metadata Service
- Simple Storage Service (S3)
- AWS Command Line Interface (CLI)
- Legacy / Service Specific CLIs
- AMI Tools
- Signing Certificates
- Backing up live disks
- Cost Savings Measures

AWS MANAGEMENT CONSOLE



AWS EC2

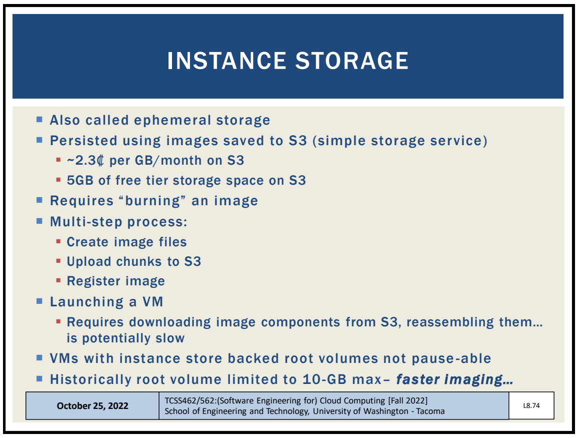
- Elastic Compute Cloud
- Instance types: https://ec2instances.info
  - **On demand instance** – full price
  - **Reserved instance** – contract based where customer guarantees VM rental for a fixed period of time (e.g. 1 year, 3 years, etc.) Deeper discounts with longer term commitments
  - **Spot instance** – portion of cloud capacity reserved for low cost instances, when demand exceeds supply instances are randomly terminated with 2 minute warning
    - Users can make diverse VM requests using different types, zones, regions, etc. to minimize instance terminations
    - Developers can design for failure because often only 1 or 2 VMs in a cluster fail at any given time. They then need to be replaced.
  - **Dedicated host** – reserved private HW (server)
  - Instance families - General, compute-optimized, memory-optimized, GPU, etc.



AWS EC2 - 2

- Storage types
  - Instance storage - ephemeral storage
    - Temporary disk volumes stored on disks local to the VM
    - Evolution: physical hard disk drives (HDDs)
    - Solid state drives (SSDs)
    - Non-volatile memory express (NVMe) drives (closer to DRAM speed)
  - EBS - Elastic block store
    - Remotely hosted disk volumes
  - EFS - Elastic file system
    - Shared file system based on network file system
    - VMs, Lambdas, Containers mount/interact with shared file system
    - Somewhat expensive

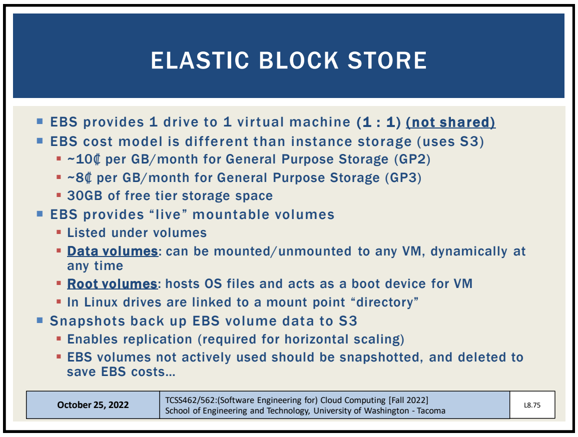
73



INSTANCE STORAGE

- Also called ephemeral storage
- Persisted using images saved to S3 (simple storage service)
  - ~2.3¢ per GB/month on S3
  - 5GB of free tier storage space on S3
- Requires “burning” an image
- Multi-step process:
  - Create image files
  - Upload chunks to S3
  - Register image
- Launching a VM
  - Requires downloading image components from S3, reassembling them… is potentially slow
- VMs with instance store backed root volumes not pause-able
- Historically root volume limited to 10-GB max– ***faster imaging…***

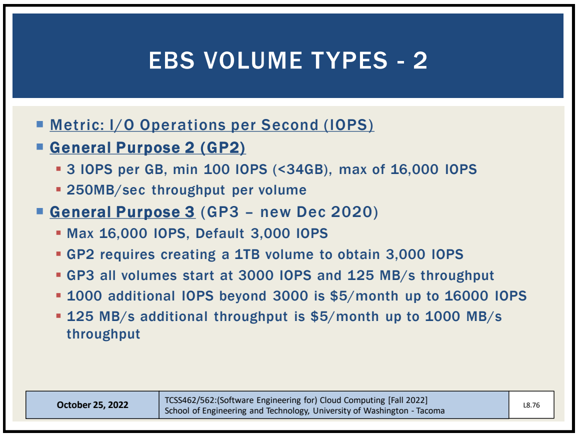
74



ELASTIC BLOCK STORE

- EBS provides 1 drive to 1 virtual machine **(1 : 1)** **<u>(not shared)</u>**
- EBS cost model is different than instance storage (uses S3)
  - ~10¢ per GB/month for General Purpose Storage (GP2)
  - ~8¢ per GB/month for General Purpose Storage (GP3)
  - 30GB of free tier storage space
- EBS provides “live” mountable volumes
  - Listed under volumes
  - **<u>Data volumes</u>**: can be mounted/unmounted to any VM, dynamically at any time
  - **<u>Root volumes</u>**: hosts OS files and acts as a boot device for VM
  - In Linux drives are linked to a mount point “directory”
- Snapshots back up EBS volume data to S3
  - Enables replication (required for horizontal scaling)
  - EBS volumes not actively used should be snapshotted, and deleted to save EBS costs…

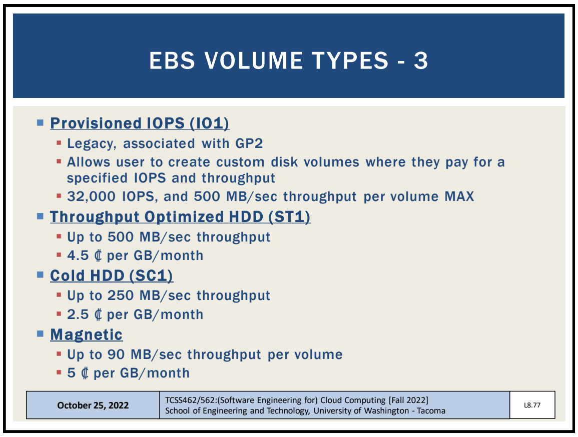
75



EBS VOLUME TYPES - 2

- <u>Metric: I/O Operations per Second (IOPS)</u>
- **<u>General Purpose 2 (GP2)</u>**
  - 3 IOPS per GB, min 100 IOPS (<34GB), max of 16,000 IOPS
  - 250MB/sec throughput per volume
- **<u>General Purpose 3</u>** (GP3 – new Dec 2020)
  - Max 16,000 IOPS, Default 3,000 IOPS
  - GP2 requires creating a 1TB volume to obtain 3,000 IOPS
  - GP3 all volumes start at 3000 IOPS and 125 MB/s throughput
  - 1000 additional IOPS beyond 3000 is $5/month up to 16000 IOPS
  - 125 MB/s additional throughput is $5/month up to 1000 MB/s throughput

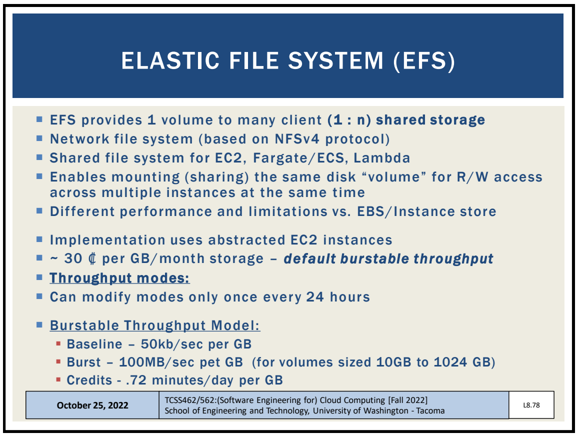
76



EBS VOLUME TYPES - 3

- **<u>Provisioned IOPS (IO1)</u>**
  - Legacy, associated with GP2
  - Allows user to create custom disk volumes where they pay for a specified IOPS and throughput
  - 32,000 IOPS, and 500 MB/sec throughput per volume MAX
- **<u>Throughput Optimized HDD (ST1)</u>**
  - Up to 500 MB/sec throughput
  - 4.5 ¢ per GB/month
- **<u>Cold HDD (SC1)</u>**
  - Up to 250 MB/sec throughput
  - 2.5 ¢ per GB/month
- **<u>Magnetic</u>**
  - Up to 90 MB/sec throughput per volume
  - 5 ¢ per GB/month

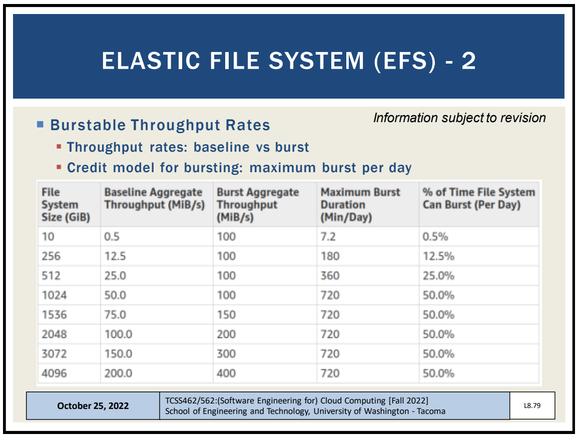
77



ELASTIC FILE SYSTEM (EFS)

- EFS provides 1 volume to many client **(1 : n) shared storage**
- Network file system (based on NFSv4 protocol)
- Shared file system for EC2, Fargate/ECS, Lambda
- Enables mounting (sharing) the same disk “volume” for R/W access across multiple instances at the same time
- Different performance and limitations vs. EBS/Instance store
- Implementation uses abstracted EC2 instances
- ~ 30 ¢ per GB/month storage – ***default burstable throughput***
- **<u>Throughput modes:</u>**
- Can modify modes only once every 24 hours
- <u>Burstable Throughput Model:</u>
  - Baseline – 50kb/sec per GB
  - Burst – 100MB/sec pet GB (for volumes sized 10GB to 1024 GB)
  - Credits - .72 minutes/day per GB

78



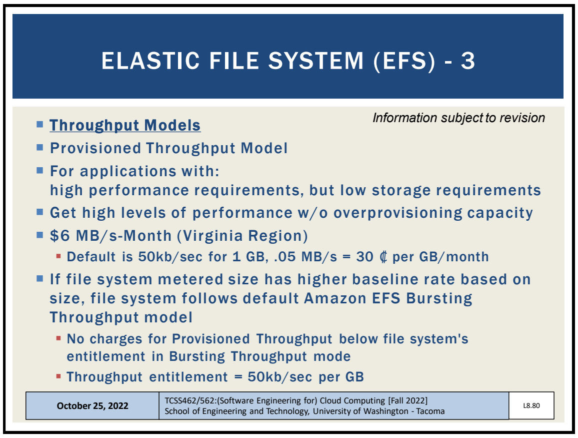
ELASTIC FILE SYSTEM (EFS) - 2

Information subject to revision

- Burstable Throughput Rates
  - Throughput rates: baseline vs burst
  - Credit model for bursting: maximum burst per day

File System Size (GiB)	Baseline Aggregate Throughput (MiB/s)	Burst Aggregate Throughput (MiB/s)	Maximum Burst Duration (Min/Day)	% of Time File System Can Burst (Per Day)
10	0.5	100	7.2	0.5%
256	12.5	100	180	12.5%
512	25.0	100	360	25.0%
1024	50.0	100	720	50.0%
1536	75.0	150	720	50.0%
2048	100.0	200	720	50.0%
3072	150.0	300	720	50.0%
4096	200.0	400	720	50.0%

79

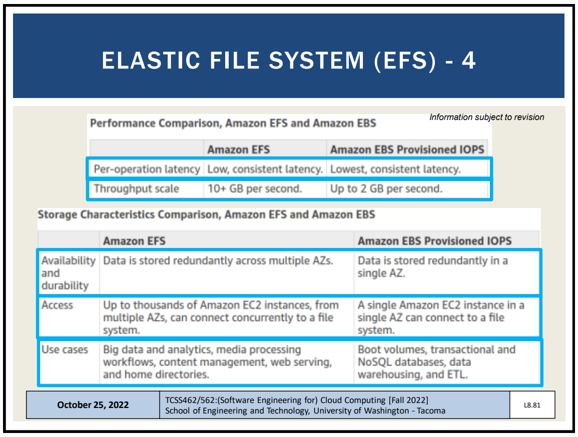


ELASTIC FILE SYSTEM (EFS) - 3

Information subject to revision

- **Throughput Models**
- Provisioned Throughput Model
- For applications with:
  high performance requirements, but low storage requirements
- Get high levels of performance w/o overprovisioning capacity
- $6 MB/s-Month (Virginia Region)
  - Default is 50kb/sec for 1 GB, .05 MB/s = 30 ₵ per GB/month
- If file system metered size has higher baseline rate based on size, file system follows default Amazon EFS Bursting Throughput model
  - No charges for Provisioned Throughput below file system's entitlement in Bursting Throughput mode
  - Throughput entitlement = 50kb/sec per GB

80



ELASTIC FILE SYSTEM (EFS) - 4

Information subject to revision

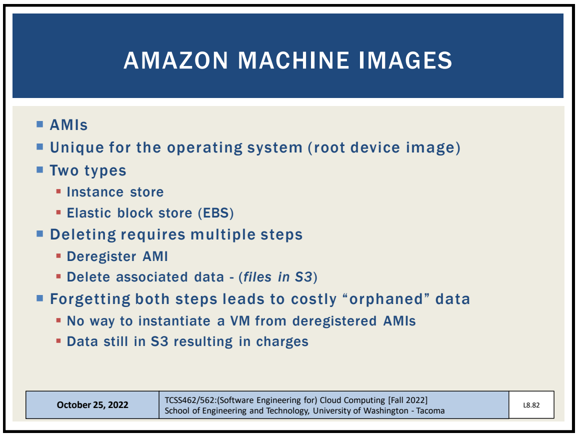
Performance Comparison, Amazon EFS and Amazon EBS

	Amazon EFS	Amazon EBS Provisioned IOPS
Per-operation latency	Low, consistent latency.	Lowest, consistent latency.
Throughput scale	10+ GB per second.	Up to 2 GB per second.

Storage Characteristics Comparison, Amazon EFS and Amazon EBS

	Amazon EFS	Amazon EBS Provisioned IOPS
Availability and durability	Data is stored redundantly across multiple AZs.	Data is stored redundantly in a single AZ.
Access	Up to thousands of Amazon EC2 instances, from multiple AZs, can connect concurrently to a file system.	A single Amazon EC2 instance in a single AZ can connect to a file system.
Use cases	Big data and analytics, media processing workflows, content management, web serving, and home directories.	Boot volumes, transactional and NoSQL databases, data warehousing, and ETL.

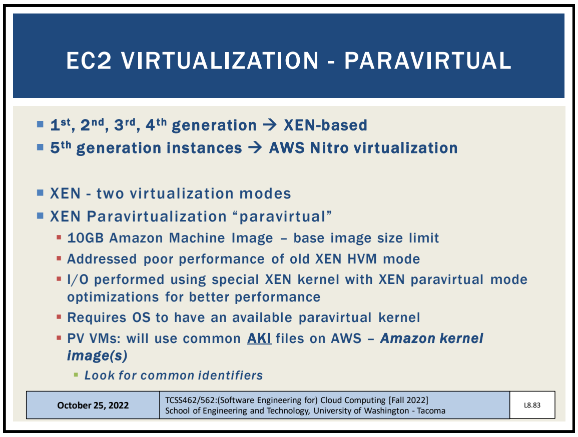
81



AMAZON MACHINE IMAGES

- AMIs
- Unique for the operating system (root device image)
- Two types
  - Instance store
  - Elastic block store (EBS)
- Deleting requires multiple steps
  - Deregister AMI
  - Delete associated data - (*files in S3*)
- Forgetting both steps leads to costly "orphaned" data
  - No way to instantiate a VM from deregistered AMIs
  - Data still in S3 resulting in charges

82

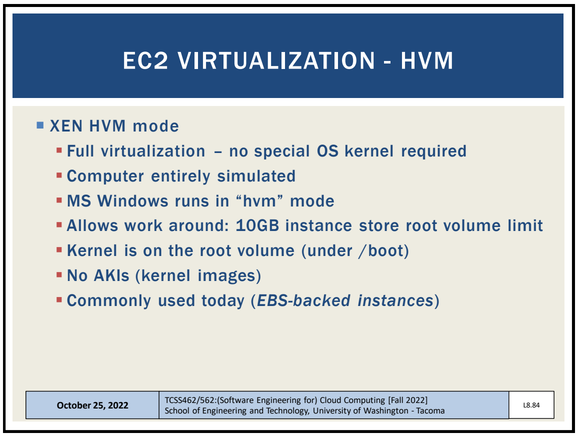


EC2 VIRTUALIZATION - PARAVIRTUAL

- **1st, 2nd, 3rd, 4th generation → XEN-based**
- **5th generation instances → AWS Nitro virtualization**

- XEN - two virtualization modes
- XEN Paravirtualization "paravirtual"
  - 10GB Amazon Machine Image – base image size limit
  - Addressed poor performance of old XEN HVM mode
  - I/O performed using special XEN kernel with XEN paravirtual mode optimizations for better performance
  - Requires OS to have an available paravirtual kernel
  - PV VMs: will use common **AKI** files on AWS – ***Amazon kernel image(s)***
    - *Look for common identifiers*

83



EC2 VIRTUALIZATION - HVM

- XEN HVM mode
  - Full virtualization – no special OS kernel required
  - Computer entirely simulated
  - MS Windows runs in "hvm" mode
  - Allows work around: 10GB instance store root volume limit
  - Kernel is on the root volume (under /boot)
  - No AKIs (kernel images)
  - Commonly used today (*EBS-backed instances*)

84

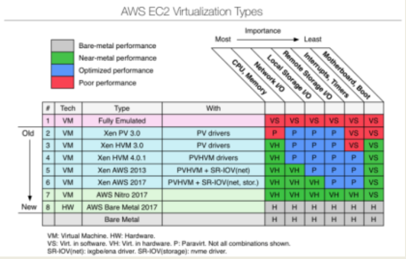
EC2 VIRTUALIZATION - NITRO

- Nitro based on Kernel-based-virtual-machines
  - Stripped down version of Linux KVM hypervisor
  - Uses KVM core kernel module
  - I/O access has a direct path to the device
- Goal: provide indistinguishable performance from bare metal

85

EVOLUTION OF AWS VIRTUALIZATION

- From: http://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html



AWS EC2 Virtualization Types

Importance: Most → Least

🟫 Bare-metal performance
🟩 Near-metal performance
🟦 Optimized performance
🟥 Poor performance

	#	Tech	Type	With	CPU, Memory	Network I/O	Local Storage I/O	Remote Storage I/O	Interrupts, Timers	Motherboard, Boot
	1	VM	Fully Emulated		VS	VS	VS	VS	VS	VS
Old	2	VM	Xen PV 3.0	PV drivers	P	P	P	P	VS	VS
	3	VM	Xen HVM 3.0	PV drivers	VH	P	P	P	VS	VS
	4	VM	Xen HVM 4.0.1	PVHVM drivers	VH	P	P	P	P	VS
	5	VM	Xen AWS 2013	PVHVM + SR-IOV(net)	VH	VH	P	P	P	VS
	6	VM	Xen AWS 2017	PVHVM + SR-IOV(net, stor.)	VH	VH	VH	P	P	VS
	7	VM	AWS Nitro 2017		VH	VH	VH	VH	VH	VS
New	8	HW	AWS Bare Metal 2017		H	H	H	H	H	H
			Bare Metal		H	H	H	H	H	H

VM: Virtual Machine. HW: Hardware.
VS: Virt. in software. VH: Virt. in hardware. P: Paravirt. Not all combinations shown.
SR-IOV(net): ixgbe/ena driver. SR-IOV(storage): nvme driver.

86

INSTANCE ACTIONS

- Stop
  - Costs of "pausing" an instance
- Terminate
- Reboot

- Image management
- Creating an image
  - EBS (snapshot)
- Bundle image
  - Instance-store

87

EC2 INSTANCE: NETWORK ACCESS

- Public IP address
- Elastic IPs
  - Costs: in-use FREE, not in-use ~12 ₵/day
  - Not in-use (e.g. "paused" EBS-backed instances)

- Security groups
  - E.g. firewall

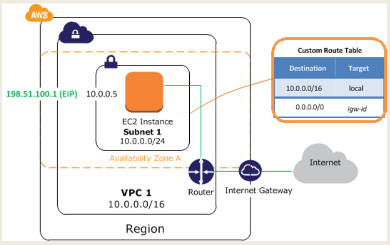
- Identity access management (IAM)
  - AWS accounts, groups

- VPC / Subnet / Internet Gateway / Router
- NAT-Gateway

88

SIMPLE VPC

- Recommended when using Amazon EC2

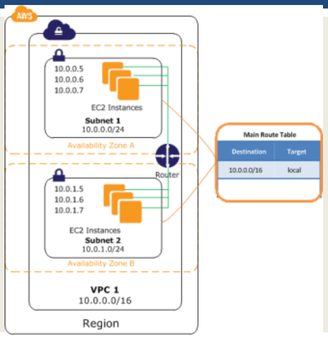


Custom Route Table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-id

89

VPC SPANNING AVAILABILITY ZONES



Main Route Table

Destination	Target
10.0.0.0/16	local

90

INSPECTING INSTANCE INFORMATION

- EC2 VMs run a local metadata service
- Can query instance metadata to self discover cloud configuration attributes

- Find your instance ID:

```
curl http://169.254.169.254/
curl http://169.254.169.254/latest/
curl http://169.254.169.254/latest/meta-data/
curl http://169.254.169.254/latest/meta-data/instance-id
; echo
```

- `ec2-get-info` command
- Python API that provides easy/formatted access to metadata

SIMPLE STORAGE SERVICE (S3)

- Key-value blob storage

- What is the difference vs. key-value stores (NoSQL DB)?

- Can mount an S3 bucket as a volume in Linux
  - Supports common file-system operations

- Provides eventual consistency

- Can store Lambda function state for life of container.

AWS CLI

- Launch Ubuntu 16.04 VM
  - Instances | Launch Instance

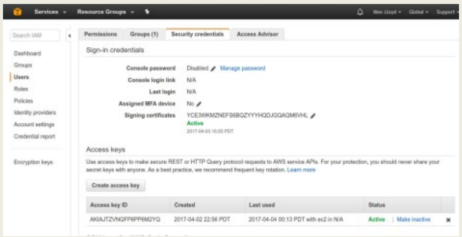
- Install the general AWS CLI
  - sudo apt install awscli

- Create config file

```
[default]
aws_access_key_id = <access key id>
aws_secret_access_key = <secret access key>
region = us-east-1
```

AWS CLI - 2

- **Creating access keys:** IAM | Users | Security Credentials | Access Keys | Create Access Keys



Sign-in credentials

Console password: Disabled Manage password
Console login link: N/A
Last login: N/A
Assigned MFA device: No
Signing certificates: YCE3WKMZNEFS6BQZYYYHQDJGQAQM6VHL
Active
2017-04-03 16:55 PDT

Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. Learn more

Access key ID	Created	Last used	Status
AKIAJTZVNQFP6PP6M2YQ	2017-04-02 22:56 PDT	2017-04-04 00:13 PDT with ec2 in N/A	Active

AWS CLI - 3

- Export the config file
  - Add to /home/ubuntu/.bashrc

```
export AWS_CONFIG_FILE=$HOME/.aws/config
```

- Try some commands:
  - `aws help`
  - `aws command help`
  - `aws ec2 help`
  - `aws ec2 describes-instances --output text`
  - `aws ec2 describe-instances --output json`
  - `aws s3 ls`
  - `aws s3 ls vmscaleruw`

LEGACY / SERVICE SPECIFIC CLI(S)

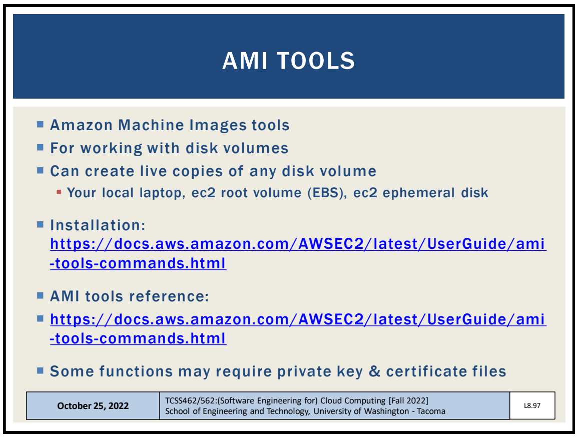
- `sudo apt install ec2-api-tools`
- Provides more concise output
- Additional functionality

- Define variables in .bashrc or another sourced script:
- `export AWS_ACCESS_KEY={your access key}`
- `export AWS_SECRET_KEY={your secret key}`

- `ec2-describe-instances`
- `ec2-run-instances`
- `ec2-request-spot-instances`

- EC2 management from Java:
- http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html

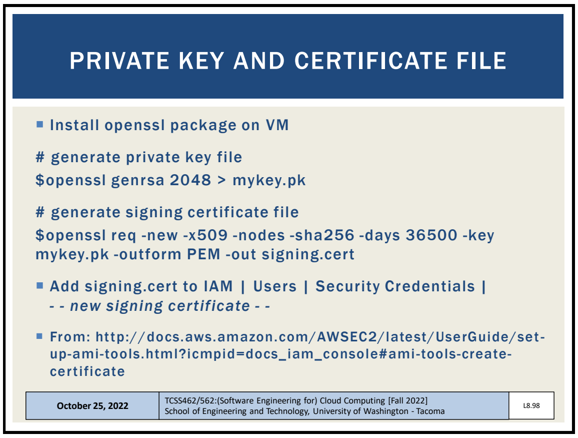
- Some AWS services have separate CLI installable by package



AMI TOOLS

- Amazon Machine Images tools
- For working with disk volumes
- Can create live copies of any disk volume
  - Your local laptop, ec2 root volume (EBS), ec2 ephemeral disk
- Installation: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ami-tools-commands.html
- AMI tools reference:
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ami-tools-commands.html
- Some functions may require private key & certificate files

97



PRIVATE KEY AND CERTIFICATE FILE

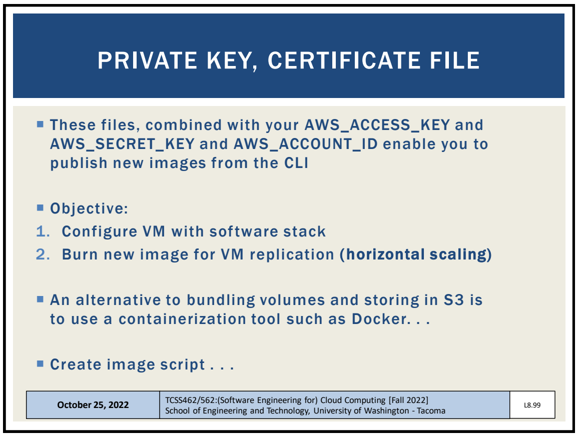
- Install openssl package on VM

# generate private key file
$openssl genrsa 2048 > mykey.pk

# generate signing certificate file
$openssl req -new -x509 -nodes -sha256 -days 36500 -key mykey.pk -outform PEM -out signing.cert

- Add signing.cert to IAM | Users | Security Credentials | - - *new signing certificate* - -
- From: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/set-up-ami-tools.html?icmpid=docs_iam_console#ami-tools-create-certificate

98



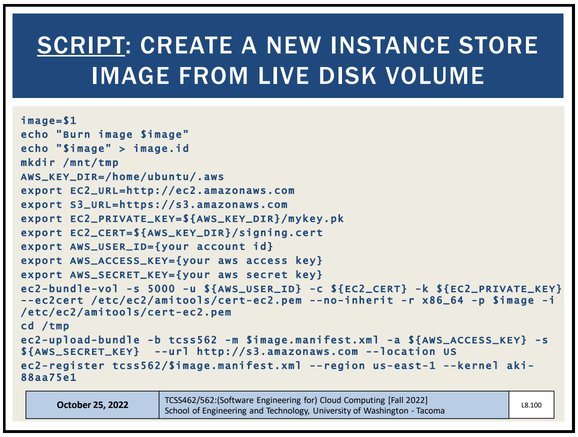
PRIVATE KEY, CERTIFICATE FILE

- These files, combined with your AWS_ACCESS_KEY and AWS_SECRET_KEY and AWS_ACCOUNT_ID enable you to publish new images from the CLI
- Objective:

1. Configure VM with software stack
2. Burn new image for VM replication (**horizontal scaling**)

- An alternative to bundling volumes and storing in S3 is to use a containerization tool such as Docker. . .
- Create image script . . .

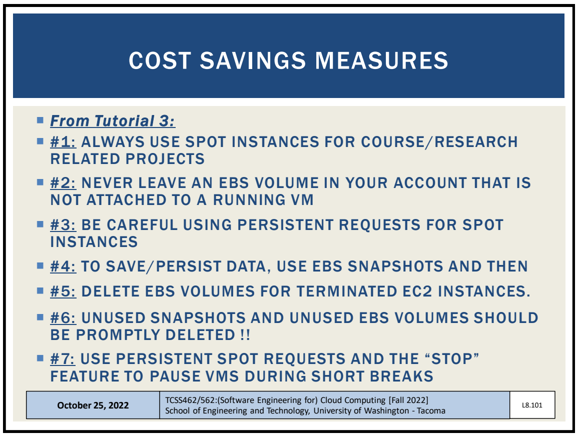
99



SCRIPT: CREATE A NEW INSTANCE STORE IMAGE FROM LIVE DISK VOLUME

```
image=$1
echo "Burn image $image"
echo "$image" > image.id
mkdir /mnt/tmp
AWS_KEY_DIR=/home/ubuntu/.aws
export EC2_URL=http://ec2.amazonaws.com
export S3_URL=https://s3.amazonaws.com
export EC2_PRIVATE_KEY=${AWS_KEY_DIR}/mykey.pk
export EC2_CERT=${AWS_KEY_DIR}/signing.cert
export AWS_USER_ID={your account id}
export AWS_ACCESS_KEY={your aws access key}
export AWS_SECRET_KEY={your aws secret key}
ec2-bundle-vol -s 5000 -u ${AWS_USER_ID} -c ${EC2_CERT} -k ${EC2_PRIVATE_KEY}
--ec2cert /etc/ec2/amitools/cert-ec2.pem --no-inherit -r x86_64 -p $image -i
/etc/ec2/amitools/cert-ec2.pem
cd /tmp
ec2-upload-bundle -b tcss562 -m $image.manifest.xml -a ${AWS_ACCESS_KEY} -s
${AWS_SECRET_KEY}  --url http://s3.amazonaws.com --location US
ec2-register tcss562/$image.manifest.xml --region us-east-1 --kernel aki-
88aa75e1
```

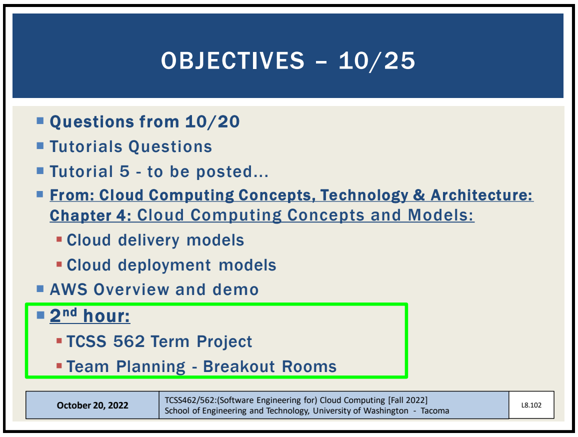
100



COST SAVINGS MEASURES

- ***From Tutorial 3:***
- #1: ALWAYS USE SPOT INSTANCES FOR COURSE/RESEARCH RELATED PROJECTS
- #2: NEVER LEAVE AN EBS VOLUME IN YOUR ACCOUNT THAT IS NOT ATTACHED TO A RUNNING VM
- #3: BE CAREFUL USING PERSISTENT REQUESTS FOR SPOT INSTANCES
- #4: TO SAVE/PERSIST DATA, USE EBS SNAPSHOTS AND THEN
- #5: DELETE EBS VOLUMES FOR TERMINATED EC2 INSTANCES.
- #6: UNUSED SNAPSHOTS AND UNUSED EBS VOLUMES SHOULD BE PROMPTLY DELETED !!
- #7: USE PERSISTENT SPOT REQUESTS AND THE “STOP” FEATURE TO PAUSE VMS DURING SHORT BREAKS

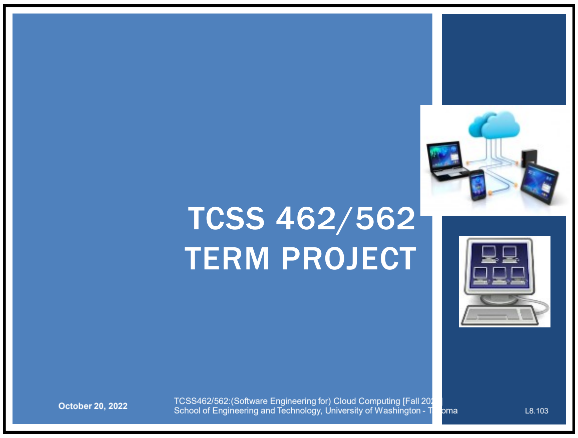
101



OBJECTIVES – 10/25

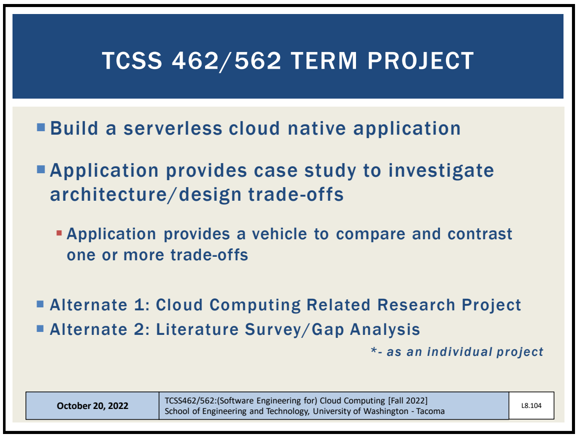
- Questions from 10/20
- Tutorials Questions
- Tutorial 5 - to be posted...
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Cloud delivery models
  - Cloud deployment models
- AWS Overview and demo
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

102



TCSS 462/562 TERM PROJECT

103

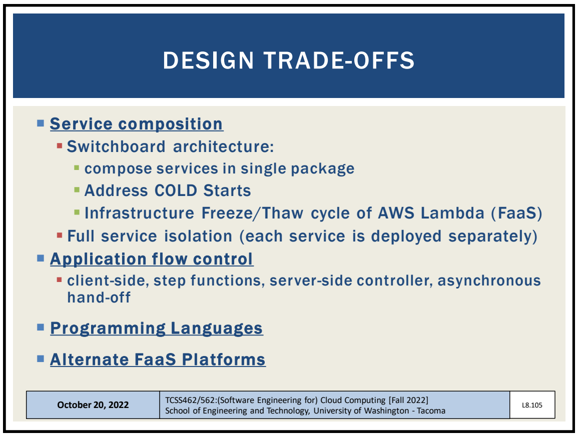


TCSS 462/562 TERM PROJECT

- Build a serverless cloud native application
- Application provides case study to investigate architecture/design trade-offs
  - Application provides a vehicle to compare and contrast one or more trade-offs
- Alternate 1: Cloud Computing Related Research Project
- Alternate 2: Literature Survey/Gap Analysis

**- as an individual project*

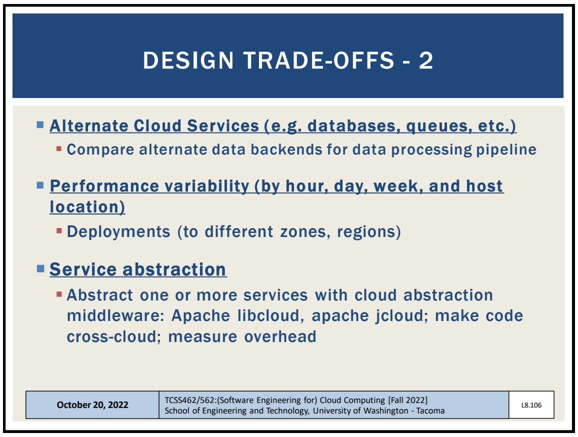
104



DESIGN TRADE-OFFS

- Service composition
  - Switchboard architecture:
    - compose services in single package
    - Address COLD Starts
    - Infrastructure Freeze/Thaw cycle of AWS Lambda (FaaS)
  - Full service isolation (each service is deployed separately)
- Application flow control
  - client-side, step functions, server-side controller, asynchronous hand-off
- Programming Languages
- Alternate FaaS Platforms

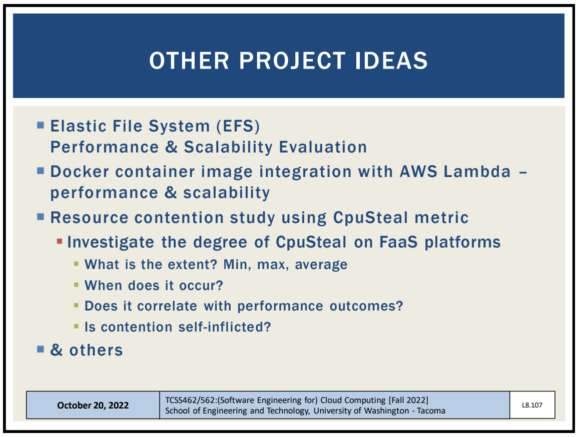
105



DESIGN TRADE-OFFS - 2

- Alternate Cloud Services (e.g. databases, queues, etc.)
  - Compare alternate data backends for data processing pipeline
- Performance variability (by hour, day, week, and host location)
  - Deployments (to different zones, regions)
- Service abstraction
  - Abstract one or more services with cloud abstraction middleware: Apache libcloud, apache jcloud; make code cross-cloud; measure overhead

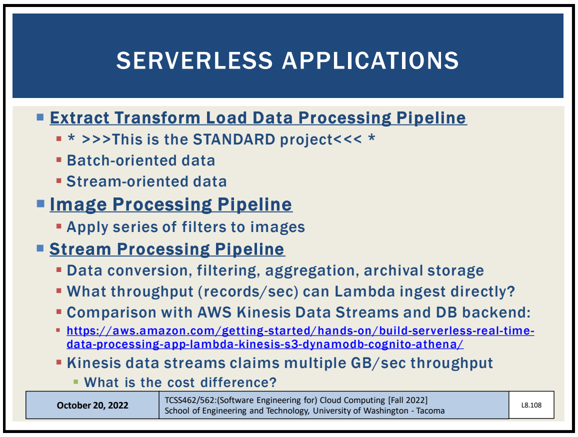
106



OTHER PROJECT IDEAS

- Elastic File System (EFS) Performance & Scalability Evaluation
- Docker container image integration with AWS Lambda – performance & scalability
- Resource contention study using CpuSteal metric
  - Investigate the degree of CpuSteal on FaaS platforms
    - What is the extent? Min, max, average
    - When does it occur?
    - Does it correlate with performance outcomes?
    - Is contention self-inflicted?
- & others

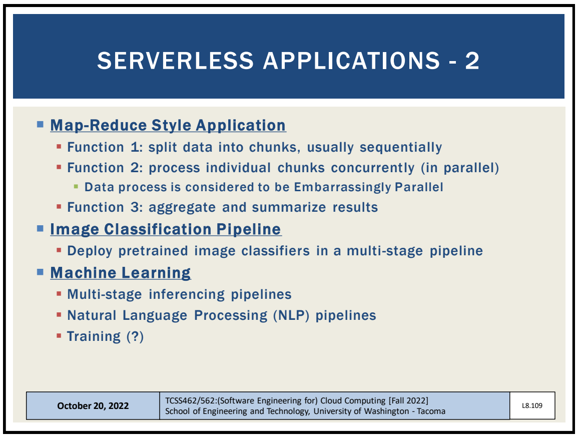
107



SERVERLESS APPLICATIONS

- Extract Transform Load Data Processing Pipeline
  - * >>>This is the STANDARD project<<< *
  - Batch-oriented data
  - Stream-oriented data
- Image Processing Pipeline
  - Apply series of filters to images
- Stream Processing Pipeline
  - Data conversion, filtering, aggregation, archival storage
  - What throughput (records/sec) can Lambda ingest directly?
  - Comparison with AWS Kinesis Data Streams and DB backend:
  - https://aws.amazon.com/getting-started/hands-on/build-serverless-real-time-data-processing-app-lambda-kinesis-s3-dynamodb-cognito-athena/
  - Kinesis data streams claims multiple GB/sec throughput
    - What is the cost difference?

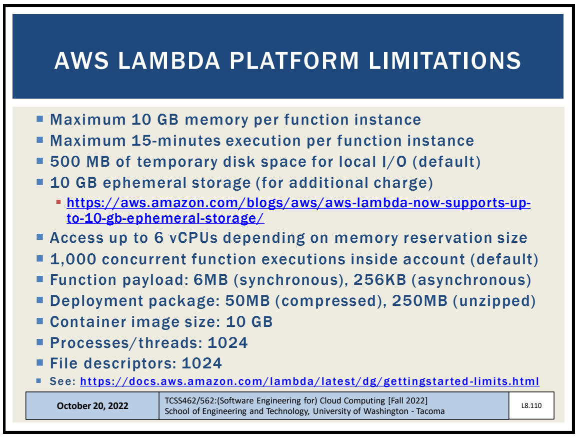
108



SERVERLESS APPLICATIONS - 2

- **Map-Reduce Style Application**
  - Function 1: split data into chunks, usually sequentially
  - Function 2: process individual chunks concurrently (in parallel)
    - Data process is considered to be Embarrassingly Parallel
  - Function 3: aggregate and summarize results
- **Image Classification Pipeline**
  - Deploy pretrained image classifiers in a multi-stage pipeline
- **Machine Learning**
  - Multi-stage inferencing pipelines
  - Natural Language Processing (NLP) pipelines
  - Training (?)

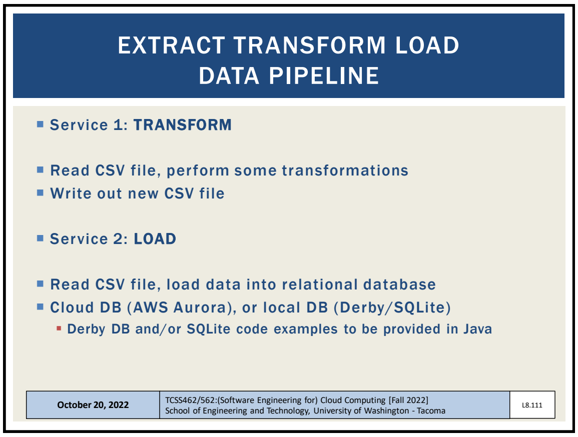
109



AWS LAMBDA PLATFORM LIMITATIONS

- Maximum 10 GB memory per function instance
- Maximum 15-minutes execution per function instance
- 500 MB of temporary disk space for local I/O (default)
- 10 GB ephemeral storage (for additional charge)
  - https://aws.amazon.com/blogs/aws/aws-lambda-now-supports-up-to-10-gb-ephemeral-storage/
- Access up to 6 vCPUs depending on memory reservation size
- 1,000 concurrent function executions inside account (default)
- Function payload: 6MB (synchronous), 256KB (asynchronous)
- Deployment package: 50MB (compressed), 250MB (unzipped)
- Container image size: 10 GB
- Processes/threads: 1024
- File descriptors: 1024
- See: https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html

110



EXTRACT TRANSFORM LOAD DATA PIPELINE

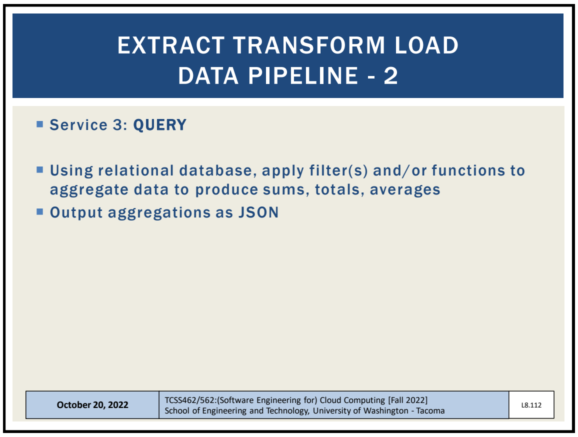
- Service 1: **TRANSFORM**

- Read CSV file, perform some transformations
- Write out new CSV file

- Service 2: **LOAD**

- Read CSV file, load data into relational database
- Cloud DB (AWS Aurora), or local DB (Derby/SQLite)
  - Derby DB and/or SQLite code examples to be provided in Java

111

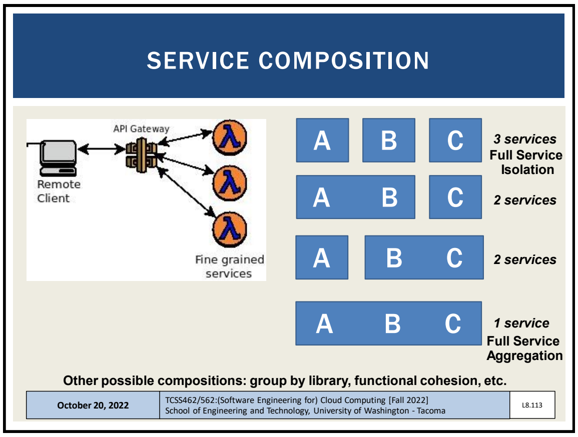


EXTRACT TRANSFORM LOAD DATA PIPELINE - 2

- Service 3: **QUERY**

- Using relational database, apply filter(s) and/or functions to aggregate data to produce sums, totals, averages
- Output aggregations as JSON

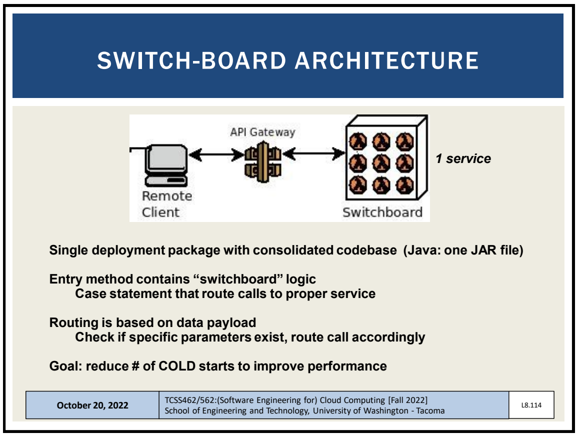
112



SERVICE COMPOSITION

Other possible compositions: group by library, functional cohesion, etc.

113



SWITCH-BOARD ARCHITECTURE

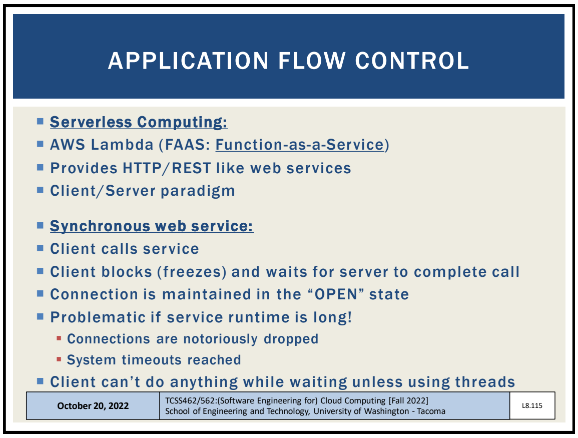
Single deployment package with consolidated codebase (Java: one JAR file)

Entry method contains "switchboard" logic
Case statement that route calls to proper service

Routing is based on data payload
Check if specific parameters exist, route call accordingly

Goal: reduce # of COLD starts to improve performance

114

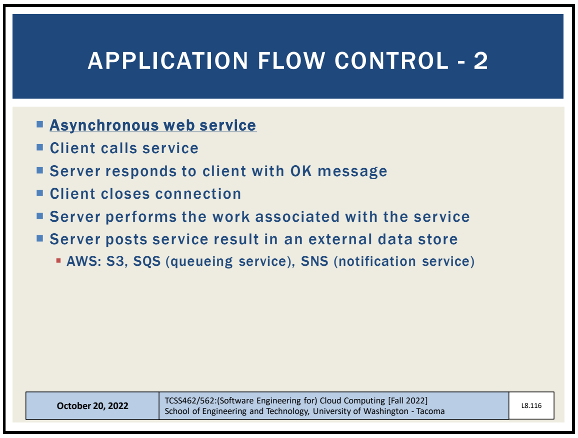


APPLICATION FLOW CONTROL

- **Serverless Computing:**
- AWS Lambda (FAAS: Function-as-a-Service)
- Provides HTTP/REST like web services
- Client/Server paradigm

- **Synchronous web service:**
- Client calls service
- Client blocks (freezes) and waits for server to complete call
- Connection is maintained in the "OPEN" state
- Problematic if service runtime is long!
  - Connections are notoriously dropped
  - System timeouts reached
- Client can't do anything while waiting unless using threads

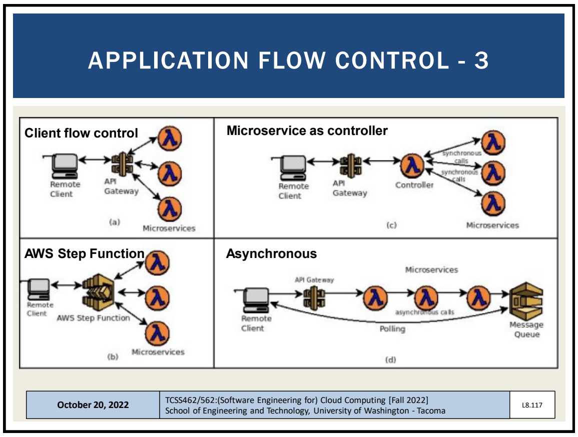
115



APPLICATION FLOW CONTROL - 2

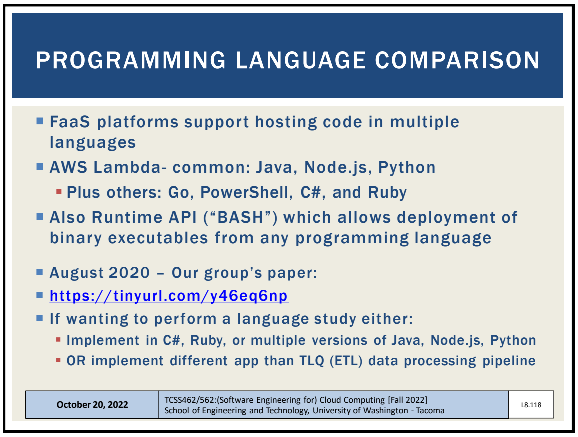
- **Asynchronous web service**
- Client calls service
- Server responds to client with OK message
- Client closes connection
- Server performs the work associated with the service
- Server posts service result in an external data store
  - AWS: S3, SQS (queueing service), SNS (notification service)

116



APPLICATION FLOW CONTROL - 3

117

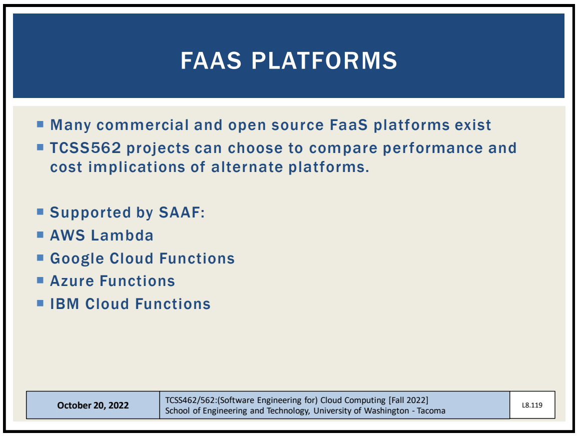


PROGRAMMING LANGUAGE COMPARISON

- FaaS platforms support hosting code in multiple languages
- AWS Lambda- common: Java, Node.js, Python
  - Plus others: Go, PowerShell, C#, and Ruby
- Also Runtime API ("BASH") which allows deployment of binary executables from any programming language

- August 2020 – Our group's paper:
- https://tinyurl.com/y46eq6np
- If wanting to perform a language study either:
  - Implement in C#, Ruby, or multiple versions of Java, Node.js, Python
  - OR implement different app than TLQ (ETL) data processing pipeline

118

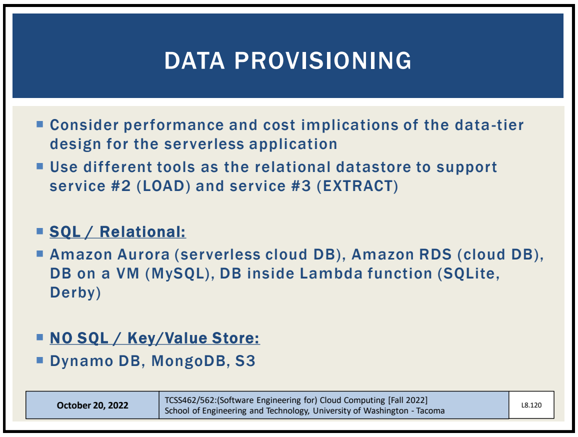


FAAS PLATFORMS

- Many commercial and open source FaaS platforms exist
- TCSS562 projects can choose to compare performance and cost implications of alternate platforms.

- Supported by SAAF:
- AWS Lambda
- Google Cloud Functions
- Azure Functions
- IBM Cloud Functions

119



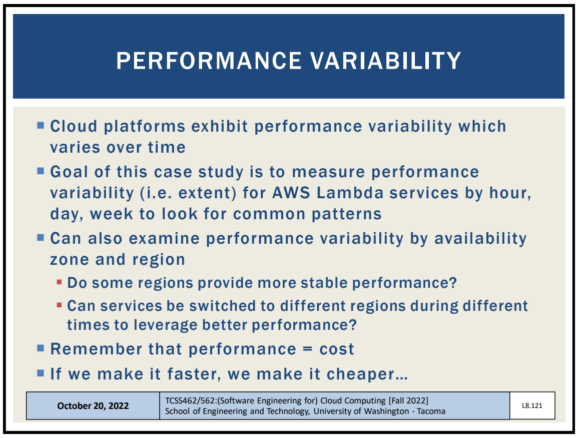
DATA PROVISIONING

- Consider performance and cost implications of the data-tier design for the serverless application
- Use different tools as the relational datastore to support service #2 (LOAD) and service #3 (EXTRACT)

- **SQL / Relational:**
- Amazon Aurora (serverless cloud DB), Amazon RDS (cloud DB), DB on a VM (MySQL), DB inside Lambda function (SQLite, Derby)

- **NO SQL / Key/Value Store:**
- Dynamo DB, MongoDB, S3

120

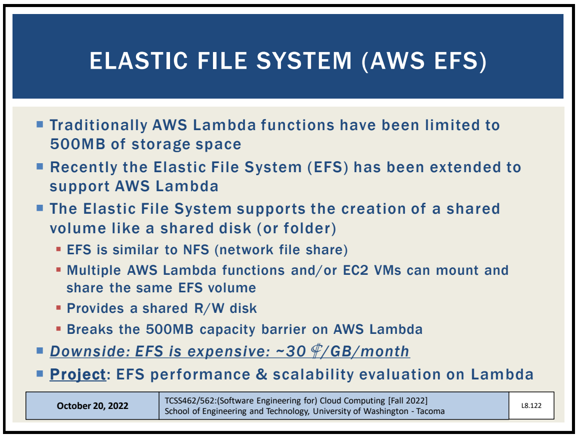


PERFORMANCE VARIABILITY

- Cloud platforms exhibit performance variability which varies over time
- Goal of this case study is to measure performance variability (i.e. extent) for AWS Lambda services by hour, day, week to look for common patterns
- Can also examine performance variability by availability zone and region
  - Do some regions provide more stable performance?
  - Can services be switched to different regions during different times to leverage better performance?
- Remember that performance = cost
- If we make it faster, we make it cheaper…

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.121

121

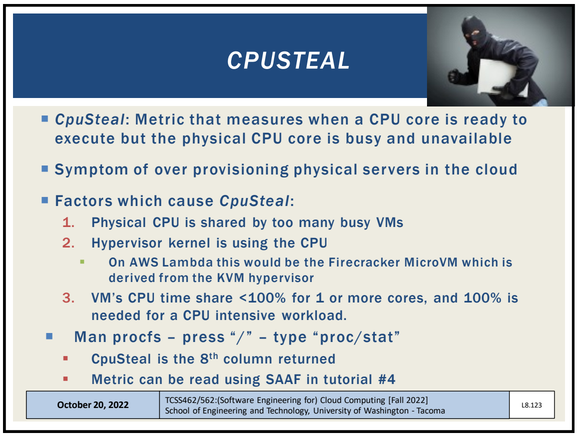


ELASTIC FILE SYSTEM (AWS EFS)

- Traditionally AWS Lambda functions have been limited to 500MB of storage space
- Recently the Elastic File System (EFS) has been extended to support AWS Lambda
- The Elastic File System supports the creation of a shared volume like a shared disk (or folder)
  - EFS is similar to NFS (network file share)
  - Multiple AWS Lambda functions and/or EC2 VMs can mount and share the same EFS volume
  - Provides a shared R/W disk
  - Breaks the 500MB capacity barrier on AWS Lambda
- ***Downside: EFS is expensive: ~30 ¢/GB/month***
- **Project**: EFS performance & scalability evaluation on Lambda

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.122

122

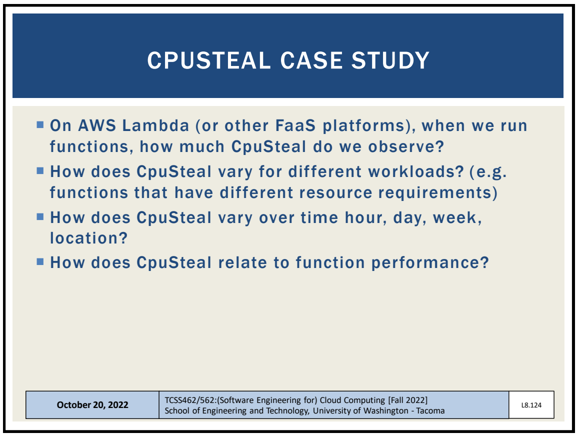


CPUSTEAL

- *CpuSteal*: Metric that measures when a CPU core is ready to execute but the physical CPU core is busy and unavailable
- Symptom of over provisioning physical servers in the cloud
- Factors which cause *CpuSteal*:
  1. Physical CPU is shared by too many busy VMs
  2. Hypervisor kernel is using the CPU
     - On AWS Lambda this would be the Firecracker MicroVM which is derived from the KVM hypervisor
  3. VM's CPU time share <100% for 1 or more cores, and 100% is needed for a CPU intensive workload.
- Man procfs – press "/" – type "proc/stat"
  - CpuSteal is the 8th column returned
  - Metric can be read using SAAF in tutorial #4

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.123

123

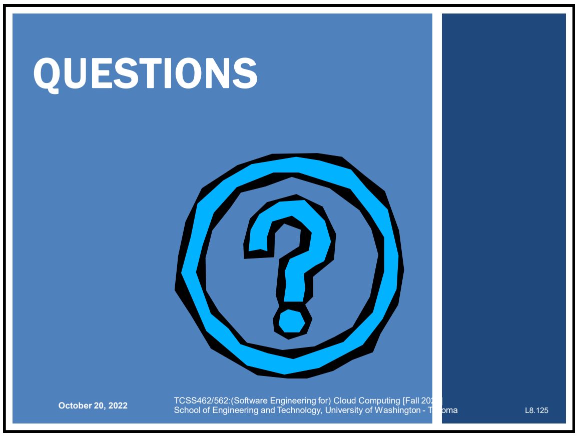


CPUSTEAL CASE STUDY

- On AWS Lambda (or other FaaS platforms), when we run functions, how much CpuSteal do we observe?
- How does CpuSteal vary for different workloads? (e.g. functions that have different resource requirements)
- How does CpuSteal vary over time hour, day, week, location?
- How does CpuSteal relate to function performance?

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.124

124



QUESTIONS

October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L8.125

125