


TCSS 562:
SOFTWARE ENGINEERING
FOR CLOUD COMPUTING

Cloud Computing:
Concepts and Models

Wes J. Lloyd
School of Engineering and Technology
University of Washington – Tacoma
MW 5:50-7:50 PM



OBJECTIVES – 10/19

■ Questions from 10/14

■ From: Cloud Computing Concepts, Technology & Architecture:
Cloud Computing Concepts and Models:

- Roles and boundaries
- Cloud characteristics
- Cloud delivery models
- Cloud deployment models

■ 2nd hour:

■ Introduce Tutorial #3 – Best Practices for Working with Virtual Machines on Amazon EC2

■ Term project case studies

■ Team planning

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington – Tacoma

L6.2

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 Oct 18 at 11:59pm | Due Oct 6 at 8:59pm | ~1 pts

TCSS 562 - Online Daily Feedback Survey - 9/30
Available until Oct 18 at 11:59pm | Due Oct 4 at 8:59pm | ~1 pts

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington – Tacoma

L6.3

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 Now and Review

Mostly Now 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

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington – Tacoma

L6.4

MATERIAL / PACE

■ Please classify your perspective on material covered in today's class (16 respondents):

■ 1-mostly review, 5-equal new/review, 10-mostly new

■ Average – 6.43 (↑ - previous 6.31)

■ Please rate the pace of today's class:

■ 1-slow, 5-just right, 10-fast

■ Average – 5.48 (↑ - previous 5.38)

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington – Tacoma

L6.5

FEEDBACK FROM 10/14

■ No survey questions

■ But there was some good discussion during office hours

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington – Tacoma

L6.6

PROJECT PROPOSAL

- **If unsure of the case study topic:**
- Groups can propose a primary and backup case study topic
- Groups can propose a topic, and change once the project proposal is approved by notifying the instructor
- **Reasons for change:**
- Discover that original topic may not work, or may require too much effort...
- Once learning more and doing initial investigations, groups may acquire a sudden passion for a particular topic
- **How to change topics:**
- Provide instructor with revised proposal as soon as possible
- Instructor will review proposal to approve/deny within ~1 day

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.7

AMDAHL'S LAW

- Portion of computation which cannot be parallelized determines the overall speedup
- For an embarrassingly parallel job of fixed size
- Assurance In the last class, we were having difficulty moving and a perfect between Amdahl's Law and Gustafson's because as it turns out, this formula was OVERSIMPLIFIED

α: fraction of the text parallel (e.g. must run sequentially)

LESSON LEARNED !!!
DO NOT TRY TO MOVE BETWEEN THE FORMULAS WHEN USING THE SIMPLIFIED FORM OF AMDAHL'S LAW

Max (N):

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.8

AMDAHL'S LAW

- Portion of computation which cannot be parallelized determines the overall speedup
- For an embarrassingly parallel job of fixed size
- Assuming no overhead for distributing the work, and a perfectly even work distribution

α: fraction of program run time which can't be parallelized (e.g. must run sequentially)

- Maximum speedup with a large number of processors (N):

$$S = 1 / \alpha$$

Where $\alpha = \sigma / (\pi + \sigma)$
Where σ = sequential time, π = parallel time
Where $T(1) = \sigma + \pi$
And $T(N) = \sigma + \pi / N$, where N = parallel computations performed

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.9

AMDAHL'S LAW

- Alternate form (may see this form more often):

$$S = \frac{1}{(1 - f) + \frac{f}{N}}$$

- f = fraction that is parallel
- N = number of processors

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.10

GUSTAFSON'S LAW

- Calculates the **scaled speed-up** using "N" processors

$$S(N) = N + (1 - N) \alpha$$

N: Number of processors
α: fraction of program run time which can't be parallelized (e.g. must run sequentially)

Here Gustafson's was also simplified, we need to substitute for α...

- Can be used to estimate runtime of parallel portion of program

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.11

GUSTAFSON'S LAW

- Calculates the **scaled speed-up** using "N" processors

$$S(N) = N + (1 - N) \alpha$$

N: Number of processors
α: fraction of program run time which can't be parallelized (e.g. must run sequentially)


- Can be used to estimate runtime of parallel portion of program
- Where $\alpha = \sigma / (\pi + \sigma)$
- Where σ = sequential time, π = parallel time
- → NEXT TIME will work to provide examples...

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.12

CLOUD COMPUTING: CONCEPTS AND MODELS



October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.13

OBJECTIVES – 10/19

- Questions from 10/14
- From: **Cloud Computing Concepts, Technology & Architecture:** Cloud Computing Concepts and Models:
 - Roles and boundaries
 - Cloud characteristics
 - Cloud delivery models
 - Cloud deployment models
- 2nd hour:
 - Introduce Tutorial #3 – Best Practices for Working with Virtual Machines on Amazon EC2
 - Term project case studies
 - Team planning

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.14

ROLES

- **Cloud provider**
 - Organization that provides cloud-based resources
 - Responsible for fulfilling SLAs for cloud services
 - Some cloud providers “resell” IT resources from other cloud providers
 - Example: Heroku sells PaaS services running atop of Amazon EC2
- **Cloud consumers**
 - Cloud users that consume cloud services
- **Cloud service owner**
 - Both cloud providers and cloud consumers can own cloud services
 - A cloud service owner may use a cloud provider to provide a cloud service (e.g. Heroku)

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.15

ROLES - 2

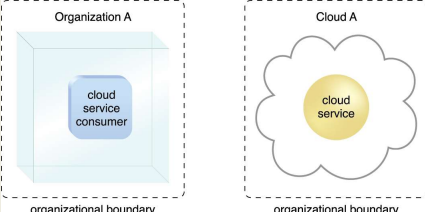
- **Cloud resource administrator**
 - Administrators provide and maintain cloud services
 - Both cloud providers and cloud consumers have administrators
- **Cloud auditor**
 - Third-party which conducts independent assessments of cloud environments to ensure security, privacy, and performance.
 - Provides unbiased assessments
- **Cloud brokers**
 - An intermediary between cloud consumers and cloud providers
 - Provides service aggregation
- **Cloud carriers**
 - Network and telecommunication providers which provide network connectivity between cloud consumers and providers

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.16

ORGANIZATION BOUNDARY

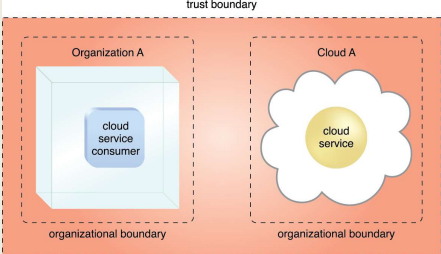


October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.17

TRUST BOUNDARY



October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.18

OBJECTIVES – 10/19

- Questions from 10/14
- From: **Cloud Computing Concepts, Technology & Architecture:**
 - Cloud Computing Concepts and Models:
 - Roles and boundaries
 - Cloud characteristics
 - Cloud delivery models
 - Cloud deployment models
- 2nd hour:
- Introduce Tutorial #3 – Best Practices for Working with Virtual Machines on Amazon EC2
- Term project case studies
- Team planning

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.19

CLOUD CHARACTERISTICS

- On-demand usage
- Ubiquitous access
- Multitenancy (resource pooling)
- Elasticity
- Measured usage
- Resiliency

Assessing these features helps measure the value offered by a given cloud service or platform


October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.20

ON-DEMAND USAGE

- The freedom to self-provision IT resources
- Generally with automated support
- Automated support requires no human involvement
- Automation through software services interface



October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.21

UBIQUITOUS ACCESS

- Cloud services are widely accessible
- Public cloud: internet accessible
- Private cloud: throughout segments of a company's intranet
- 24/7 availability

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.22

MULTITENANCY

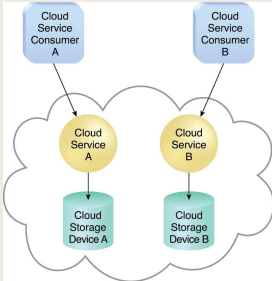
- Cloud providers pool resources together to share them with many users
- Serve multiple cloud service consumers
- IT resources can be dynamically assigned, reassigned based on demand
- Multitenancy can lead to performance variation

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.23

SINGLE TENANT MODEL



October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.24

MULTITENANT MODEL

- Resource is “multiplexed” and share amongst multiple users
- Goal is to increase utilization
- Often server resources are underutilized
- There are many “sunk costs” whether usage is 0% or 100%
- Cloud computing tries to maximize “sunk cost” investments

The diagram illustrates a cloud environment where multiple consumers (A and B) access cloud services (A and B). These services are connected to a shared cloud storage device, demonstrating resource multiplexing.

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.25

MULTITENANT DATABASE

The diagram compares three database architectures:
E1: Isolated - Each tenant (A, B, C) has its own separate database.
E2: Semi-shared - Tenants A, B, and C share a single database but maintain separate schemas.
E3: Shared - Tenants A, B, and C share both the database and the schema.

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.26

MULTITENANCY OF RESOURCES

- Where is the multitenancy?

The diagram shows four architectural models:
1. Traditional On Premise: Users connect via LAN to a single application, database, and server.
2. Single Tenant (Hosted): Customers connect via VPN to their own dedicated application, database, and server.
3. Multi-Tenant: Customers connect via TSS to a shared application, database, and server.
4. Virtual Appliance: Customers connect via TSS to virtualized applications, databases, and servers running on a single physical server.

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.27

ELASTICITY

- Automated ability of cloud to transparently scale resources
- Scaling based on runtime conditions or pre-determined by cloud consumer or cloud provider
- Threshold based scaling
 - CPU-utilization > threshold_A, Response_time > 100ms
 - Application agnostic vs. application specific thresholds
 - Why might an application agnostic threshold be non-ideal?
- Load prediction
 - Historical models
 - Real-time trends

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.28

PREDICTABLE DEMAND

- Example:

The figure shows two line graphs for Netflix's auto-scaling. The top graph, 'App CPU', shows periodic peaks in CPU usage. The bottom graph, 'Instance', shows the number of EC2 instances being scaled up during these peaks and scaled down during troughs. Logos for Netflix and Amazon are present.

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.29

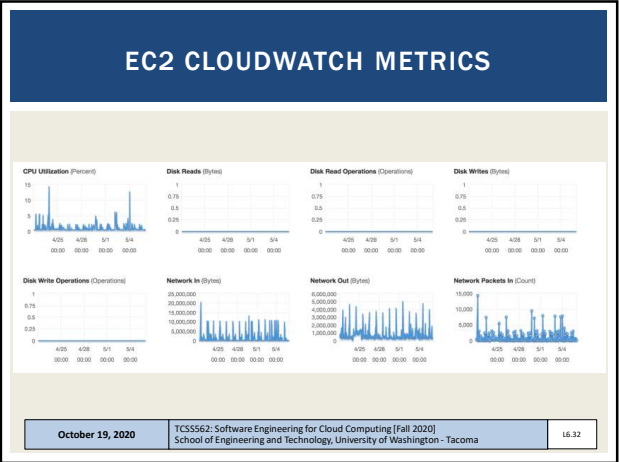
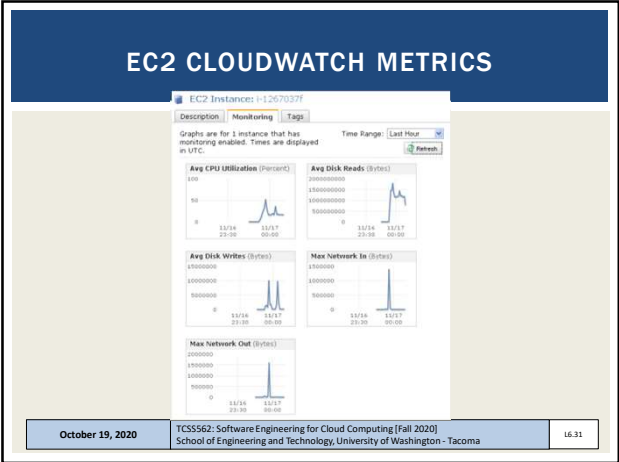
MEASURED USAGE

- Cloud platform tracks usage of IT resources
- For billing purposes
- Enables charging only for IT resources actually used
- Can be time-based (minute, hour, day)
- Can be throughput-based (MB, GB)
- Not all measurements are for billing
- Some measurements can support auto-scaling
- For example CPU utilization

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.30



RESILIENCY

- Distributed redundancy across physical locations
- Used to improve reliability and availability of cloud-hosted applications
- Very much an engineering problem
- No “resiliency-as-a-service” for user deployed apps
- Unique characteristics of user applications make a one-size fits all service solution challenging

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.33

OBJECTIVES - 10/19

- Questions from 10/14
- From: **Cloud Computing Concepts, Technology & Architecture:** Cloud Computing Concepts and Models:
 - Roles and boundaries
 - Cloud characteristics
 - Cloud delivery models
 - Cloud deployment models
- 2nd hour:
- Introduce Tutorial #3 – Best Practices for Working with Virtual Machines on Amazon EC2
- Term project case studies
- Team planning

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.34

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 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.35

CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS) delivery model
- Virtualization is a key-enabling technology of IaaS cloud
- Uses virtual machines to deliver cloud resources to end users

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.36

CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS) delivery model

Virtualization is key to sharing powerful servers among users by running *many* isolated private virtual computers known as virtual machines (VMs)
...VMs are the basis of cloud v1.0

October 19, 2020


TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.37

CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS) delivery model

Virtual Machines are the building blocks for “Cloud Service Delivery Models”
They are the “vehicles” used to deliver compute resources to end users...
cloud 1.0





October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.38

CLOUD DELIVERY MODELS

- What is the appropriate level of **abstraction**?
- How should applications be deployed?
 - IaaS, PaaS, SaaS, DbaaS, FaaS
- How do we ensure Quality-of-Service?
 - Performance, Availability, Responsiveness, Fault Tolerance
- How is **scalability** provided?
- As users, how do we minimize hosting costs?
 - How do we estimate hosting costs?

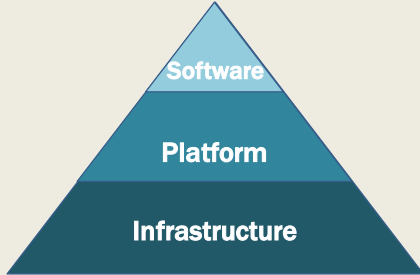


October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.39

CLASSIC CLOUD DELIVERY MODELS

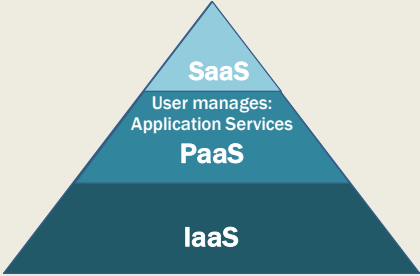


October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.40

CLASSIC CLOUD DELIVERY MODELS






October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.41

EXAMPLE CLOUD SERVICES

 SAAS Software as a Service	 PAAS Platform as a Service	 IAAS Infrastructure as a Service
<div>Email CRM Collaborative ERP</div>	<div>Application Development Decision Support Web Streaming</div>	<div>Caching Legacy Networking Security File Technical System Mgmt</div>
CONSUME	BUILD ON IT	MIGRATE TO IT

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.42

END USER APPLICATIONS

Many different “cloud” providers (especially SaaS)

Many cloud providers are also cloud consumers

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.43

INFRASTRUCTURE-AS-A-SERVICE

- Compute resources, on demand, as-a-service
 - Generally raw “IT” resources
 - Hardware, network, containers, operating systems
- Typically provided through virtualization
- Generally not-preconfigured
- Administrative burden is owned by cloud consumer
- Best when high-level control over environment is needed
- Scaling is generally **not** automatic...
- Resources can be managed in bundles
- AWS CloudFormation: Allows specification in JSON/YAML of cloud infrastructures

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.44

SC1 SC2 SC3 SC4

SC5 SC6 SC7

SC8 SC9 SC10

SC11 SC12 SC13

SC14 SC15

M: Tomcat ApplicationServer
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)

45

SC1 SC2 SC3 SC4

SC5 SC6 SC7

SC8 SC9 SC10

SC11 SC12 SC13

SC14 SC15

M: Tomcat ApplicationServer
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)

46

Bell's Number:

n	k
4	15
5	52
6	203
7	877
8	4,140
9	21,147
n	...

k: number of ways
n components can be
distributed across containers

SC1 SC2 SC3 SC4

SC5 SC6 SC7

SC8 SC9 SC10

SC11 SC12 SC13

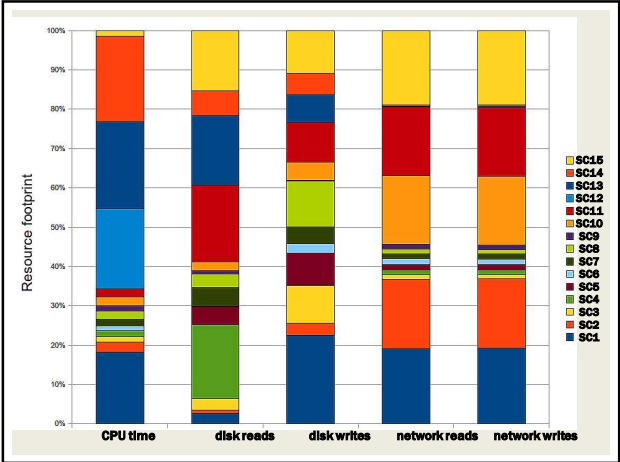
SC14 SC15

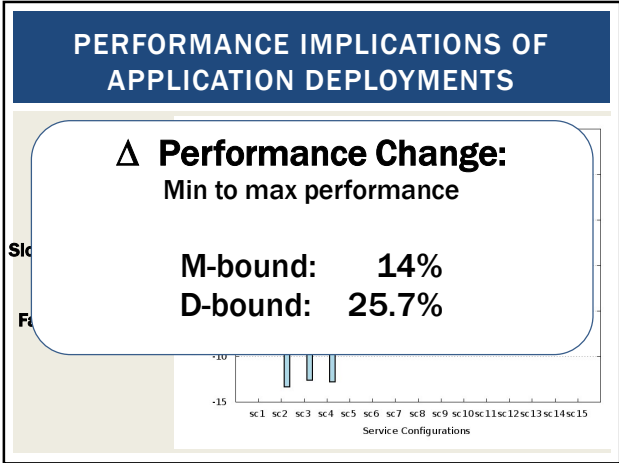
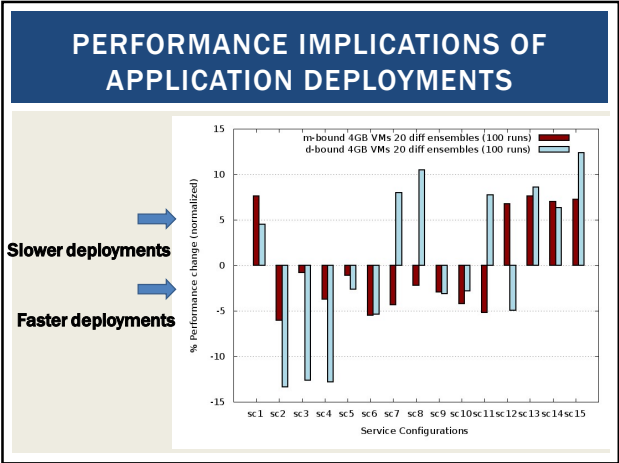
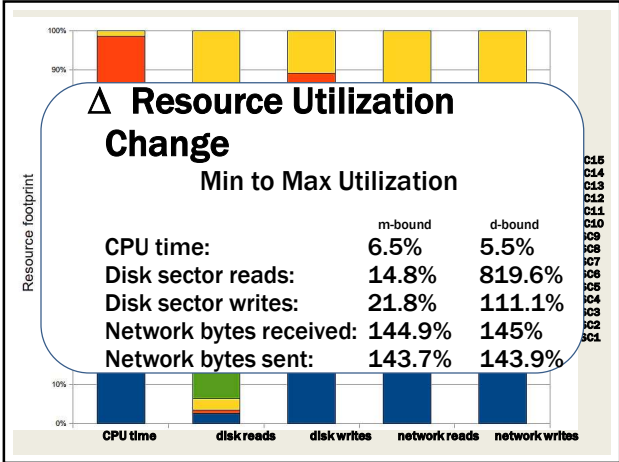
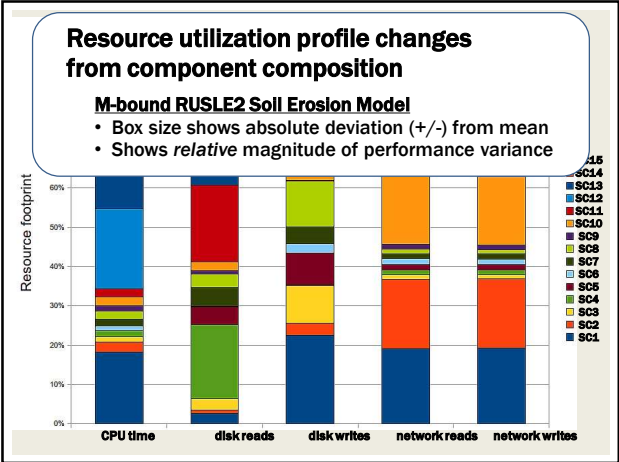
M: Tomcat ApplicationServer
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)

47

Component Composition Example

- An application with 4 components has 15 compositions
- One or more component(s) deployed to each VM
- Each VM launched to separate physical machine





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 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.53

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

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.54

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.55

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 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.56

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.57

The diagram illustrates the cloud computing stack. At the top is the SaaS layer with three cloud service icons (A, B, C). Below is the PaaS layer with three 'Ready-Made Environment' boxes (A, B, C). At the bottom is the IaaS layer, which contains two 'Virtual Server' boxes (A and B) connected to a single 'Physical Server A' box. Dashed lines indicate the flow and dependencies between these layers.

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.58

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 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.59

SERVERLESS COMPUTING

Introducing Cloud 2.0

Serverless Computing

Deploy Applications Without Fiddling With Servers

Image from: <https://medium.com/tech-tellus/serverless-computing-deploy-applications-without-fiddling-with-servers/>

SERVERLESS COMPUTING

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

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?

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

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

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

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

How many servers should I budget for?

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

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

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

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

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

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

Servers

(AAAAAAAAHHHHH!)


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

October 19, 2020TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - TacomaL6.62

SERVERLESS COMPUTING

What is **serverless**?

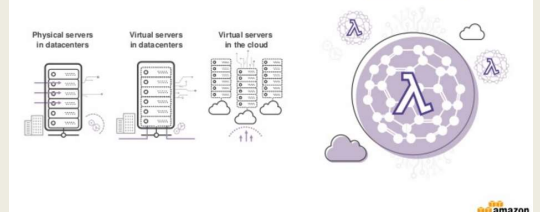
Build and run applications without thinking about servers



October 19, 2020TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - TacomaL6.62

SERVERLESS COMPUTING - 2

Evolving to **serverless**



Physical servers in datacenters

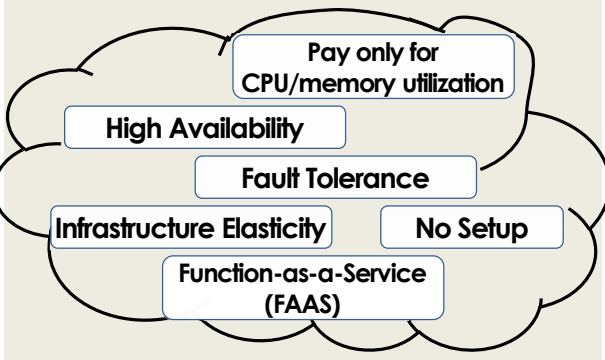
Virtual servers in datacenters

Virtual servers in the cloud

SERVERLESS

October 19, 2020TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - TacomaL6.63

SERVERLESS COMPUTING



Pay only for CPU/memory utilization

High Availability

Fault Tolerance

Infrastructure Elasticity

No Setup

Function-as-a-Service (FAAS)

SERVERLESS COMPUTING

Why Serverless Computing?

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

...they are built into the platform

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 19, 2020TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - TacomaL6.66

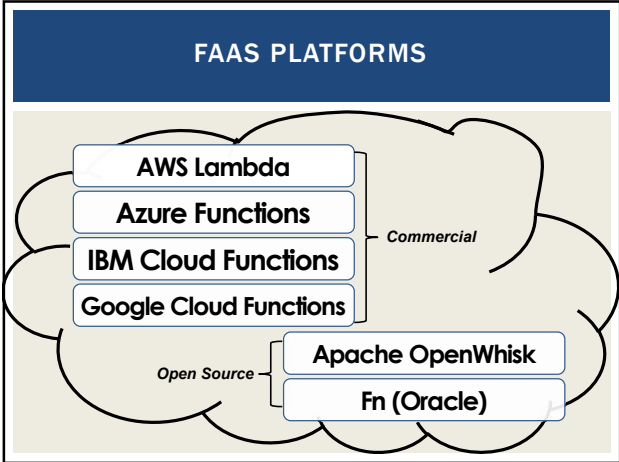
SERVERLESS VS. FAAS

- **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...**

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.67



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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.67

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.70

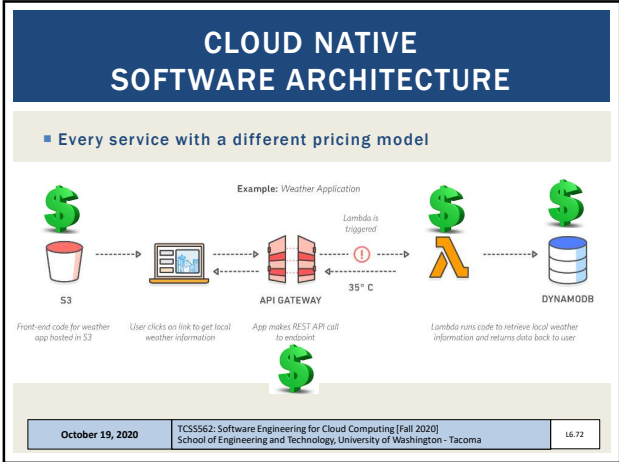
FAAS PLATFORMS - 3

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.71



October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.72


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 →

Spot Instance Pricing History



Product: Linux.R5 (Amazon VPC) Instance Type: C4.Large

October 19, 2020 TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma L6.73

FAAS COMPUTING BILLING MODELS

- AWS Lambda Pricing**
- FREE TIER:**

first 1,000,000 function calls/month → FREE

first 400 GB-sec/month → FREE
- Afterwards: *obfuscated pricing (AWS Lambda):*

\$0.0000002 per request

\$0.000000208 to rent 128MB / 100-ms

October 19, 2020 TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma L6.74

WEBSERVICE HOSTING EXAMPLE

- ON AWS Lambda**
- Each service call: 100% of 1 CPU-core
100% of 4GB of memory
- Workload: 2 continuous client threads
- Duration: 1 month (30 days)
- ON AWS EC2:**
- Amazon EC2 c4.large 2-vCPU VM
- Hosting cost: \$72/month
- c4.large: 10¢/hour, 24 hrs/day x 30 days

How much would hosting this workload cost on AWS Lambda?

October 19, 2020 TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma L6.75

PRICING OBFUSCATION

- Workload: 20,736,000 GB-sec
- FREE: - 400 GB-sec
- Ch AWS EC2: \$72.00
- M AWS Lambda: \$345.88
- In
- FF
- Ch AWS Lambda: \$345.88
- Charge.
- Calls: \$0.84
- Total: \$345.88
- BREAK-EVEN POINT = ~4,326,927 GB-sec-month

October 19, 2020 TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma L6.76

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?

October 19, 2020 TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma L6.77

FACTORS IMPACTING PERFORMANCE OF FAAS COMPUTING PLATFORMS

- Infrastructure elasticity
- Load balancing
- Provisioning variation
- Infrastructure retention: COLD vs. WARM
 - Infrastructure freeze/thaw cycle
- Memory reservation
- Service composition

October 19, 2020 TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma L6.78

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?

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.79

VENDOR ARCHITECTURAL LOCK-IN

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

Example: Weather Application

The diagram shows a Client interacting with an S3 bucket (containing front-end code), an API Gateway, and a Lambda function. The Lambda function is triggered by a 35° C event and interacts with a DynamoDB database. The flow is: Client → S3 → API Gateway → Lambda → DynamoDB. A red circle highlights the Lambda function, with a note 'Lambda is triggered'.

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 is triggered
Lambda runs code to retrieve local weather information and returns data back to user

Images credit: aws.amazon.com

- Increased dependencies → increased hosting costs

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.80

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 GB-sec/month → FREE

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

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.81

MEMORY RESERVATION QUESTION...

- Lambda memory reserved for functions
- UI provides “slider bar” to set function's memory allocation
- Resource capacity (CPU, disk, network) coupled to slider bar: “every doubling of memory, doubles CPU...”
- But how much memory do model services require?

Basic settings

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

1536 MB

Timeout info
5 min 0 sec

Description

Performance

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.82

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

Performance

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

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.83

INFRASTRUCTURE FREEZE/THAW CYCLE

- Unused infrastructure is deprecated
 - But after how long?
- Infrastructure: VMs, “containers”
- **Provider-COLD / VM-COLD**
 - “Container” images - built/transferred to VMs
- **Container-COLD**
 - Image cached on VM
- **Container-WARM**
 - “Container” running on VM

FREEZE-THAW CYCLE CAUSING POTHOLES

Image from: Denver7 - The Denver Channel News

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.84



FUNCTION-AS-A-SERVICE

AWS
Lambda
Demo

85

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 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.86

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

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.87

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 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.88

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

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L10.89

OBJECTIVES – 10/19

- **Questions from 10/14**
- **From: Cloud Computing Concepts, Technology & Architecture:**
Cloud Computing Concepts and Models:
 - Roles and boundaries
 - Cloud characteristics
 - Cloud delivery models
 - Cloud deployment models
- **2nd hour:**
- Introduce Tutorial #3 – Best Practices for Working with Virtual Machines on Amazon EC2
- Term project case studies
- Team planning

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.90

CLOUD DEPLOYMENT MODELS

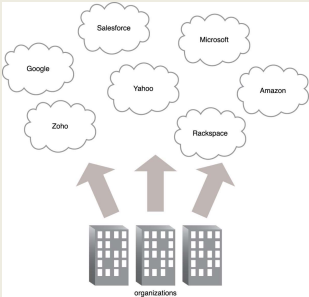
- Distinguished by ownership, size, access
- Four common models
 - Public cloud
 - Community cloud
 - Hybrid cloud
 - Private cloud

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.91

PUBLIC CLOUDS



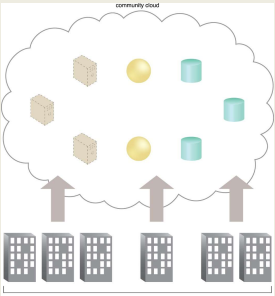
October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.92

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



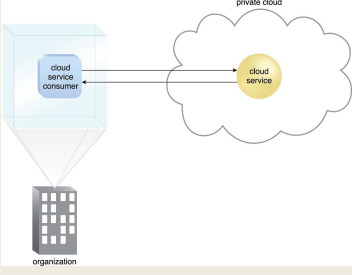
October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.93

PRIVATE CLOUD

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



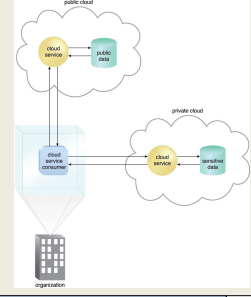
October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.94

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 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.95

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 19, 2020


TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.96

WE WILL RETURN AT
~7:12PM



TCSS 562
TERM PROJECT



October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.98

TCSS 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

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.99

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

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.100

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

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.101

OTHER PROJECT IDEAS

- Elastic File System (EFS)
Performance & Scalability Evaluation
- Resource contention study using CpuSteal metric
- & others...

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.102

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
 - Can use 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 claim multiple GB/sec throughput
 - What throughput can Lambda ingest directly?
 - What is the cost difference?

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.103

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.104

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.105

SERVICE COMPOSITION

Remote Client

API Gateway

Fine grained services

A	B	C	3 services
A	B	C	2 services
A	B	C	2 services
A	B	C	1 service

Full Service Isolation

Full Service Aggregation

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.106

SWITCH-BOARD ARCHITECTURE

Remote Client

API Gateway

Switchboard

1 service

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.107

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.108

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)

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.109

APPLICATION FLOW CONTROL - 3

(a) Client flow control: A Remote Client calls an API Gateway, which then calls multiple Microservices. The client waits for a response.

(b) AWS Step Function: A Remote Client calls an AWS Step Function, which then calls multiple Microservices. The client waits for a response.

(c) Microservice as controller: A Remote Client calls an API Gateway, which then calls a Controller. The Controller then calls multiple Microservices. The client waits for a response.

(d) Asynchronous: A Remote Client calls an API Gateway, which then calls a Microservice. The Microservice calls a Message Queue. The client polls the Message Queue for a response.

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.110

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.111

FAAS PLATFORMS

- Many commercial and open source FaaS platforms exist
- TCCS562 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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.112

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

October 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.113

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 19, 2020

TCCS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.114

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 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.115

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*:
 - Physical CPU is shared by too many busy VMs
 - Hypervisor kernel is using the CPU
 - On AWS Lambda this would be the Firecracker MicroVM which is derived from the KVM hypervisor
 - VM's CPU time share <100% for 1 or more cores, and 100% is needed for a CPU intensive workload.
- Man procs – press “/” – type “proc/stat”
 - CpuSteal is the 8th column returned
 - Metric can be read using SAAF in tutorial #4

October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.116

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 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.117

QUESTIONS




October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.118

QUESTIONS



October 19, 2020

TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma

L6.119

TCSS 562
OFFICE HOURS

PLEASE SAY HELLO



L6.120