

# TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

## Cloud Computing: Concepts and Models - II

Wes J. Lloyd

School of Engineering and Technology  
University of Washington – Tacoma

MW 5:50-7:50 PM



## OBJECTIVES – 10/21

- **Questions from 10/19**
- Quiz 1 – to be posted on Canvas
- From: Cloud Computing Concepts, Technology & Architecture:  
Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- 2<sup>nd</sup> hour:
  - Class Activity #2
  - Term project questions
  - Team planning

October 21, 2020

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

L7.2

# 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 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

October 21, 2020

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

L7.3

## 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:

12345678910

Mostly Review To MeEqual New and ReviewMostly New to Me

Question 2

0.5 pts

Please rate the pace of today's class:

12345678910

SlowJust RightFast

October 21, 2020

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

L7.4

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (18 respondents):
  - 1-mostly review, 5-equal new/review, 10-mostly new
  - **Average – 6.50 (↑ - previous 6.43)**
- Please rate the pace of today's class:
  - 1-slow, 5-just right, 10-fast
  - **Average – 5.58 (↑ - previous 5.48)**

October 21, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.5
------------------	------------------------------------------------------------------------------------------------------------------------------------------	------

FEEDBACK FROM 10/19



- No survey questions

October 21, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.6
------------------	------------------------------------------------------------------------------------------------------------------------------------------	------

# CLOUD COMPUTING: CONCEPTS AND MODELS

October 21, 2020

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



## OBJECTIVES – 10/21

- Questions from 10/19
- Quiz 1 – to be posted on Canvas
- From: Cloud Computing Concepts, Technology & Architecture:  
Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- 2<sup>nd</sup> hour:
  - Class Activity #2
  - Term project questions
  - Team planning

October 21, 2020

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

L7.8

## 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 21, 2020

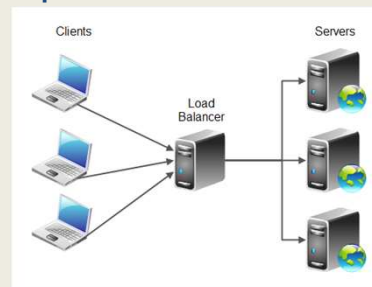
TCSS562: Software Engineering for Cloud Computing [Fall 2020]  
School of Engineering and Technology, University of Washington - Tacoma

L7.9

## 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)



October 21, 2020

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

L7.10

## 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 21, 2020

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

L7.11

## 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 21, 2020

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

L7.12

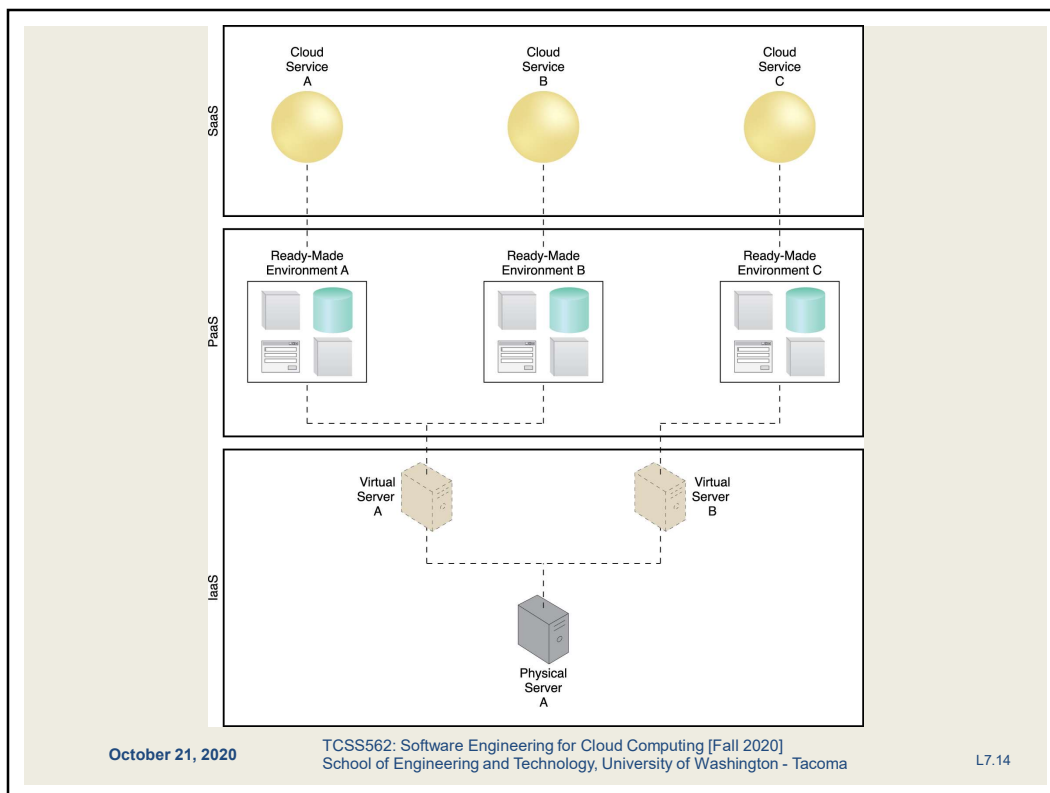
## SOFTWARE-AS-A-SERVICE

- 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 21, 2020

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

L7.13



October 21, 2020

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

L7.14

## 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 21, 2020

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

L7.15

## SERVERLESS COMPUTING

Introducing Cloud 2.0

### Serverless Computing

Deploy Applications Without  
Fiddling With Servers



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



SERVERLESS COMPUTING

How should my app withstand a server falling?

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?

How will the application handle server hardware failure?

How will new code be deployed to my servers?

What size server is right for my performance?

How many users create too much load for my servers?

How many servers should I budget for?

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?


Servers


(AAHHHHHHHHH!!!)

SERVERLESS COMPUTING

What is serverless?

Build and run applications without thinking about servers





October 21, 2020

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

L7.18

# SERVERLESS COMPUTING - 2

## Evolving to serverless

Physical servers in datacenters

Virtual servers in datacenters

Virtual servers in the cloud

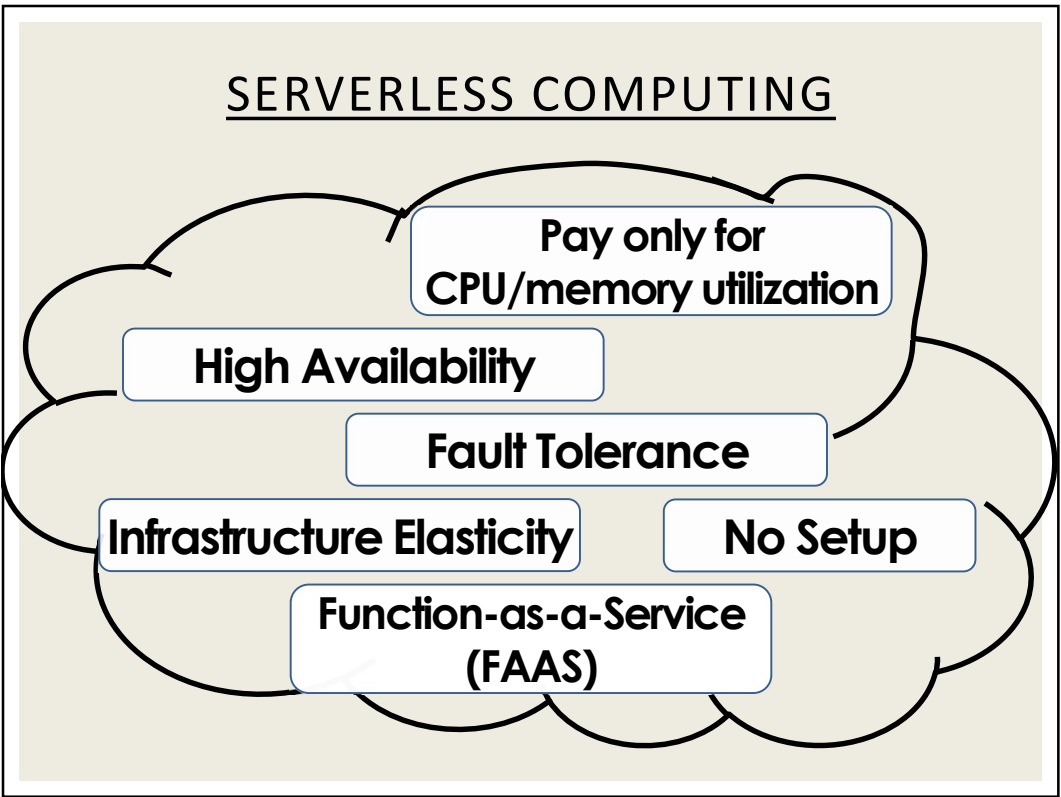
SERVERLESS

amazon web services

October 21, 2020

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

L7.19



## 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

## SERVERLESS VS. FAAS

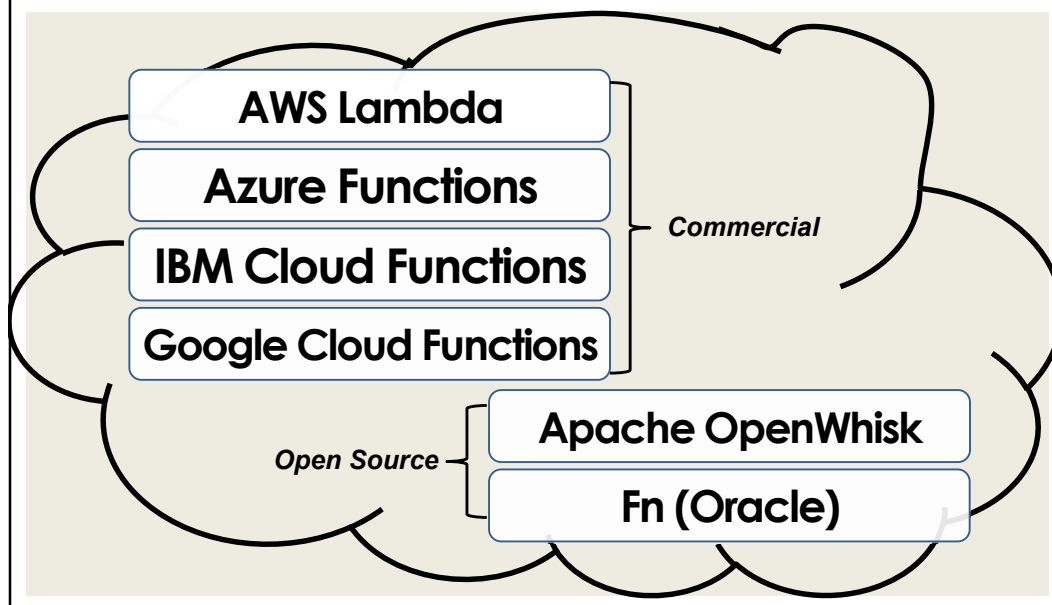
- **Serverless Computing**
- Refers to the avoidance of managing servers
- Can pertain to a number of “as-a-service” cloud offerings
- **Function-as-a-Service (FaaS)**
  - Developers write small code snippets (microservices) which are deployed separately
- **Database-as-a-Service (DBaaS)**
- **Container-as-a-Service (CaaS)**
- Others...
- **Serverless is a buzzword**
- **This space is evolving...**

October 21, 2020

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


L7.23


## FAAS PLATFORMS



# 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

# 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 21, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.26
------------------	------------------------------------------------------------------------------------------------------------------------------------------	-------

## FAAS PLATFORMS - 3

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

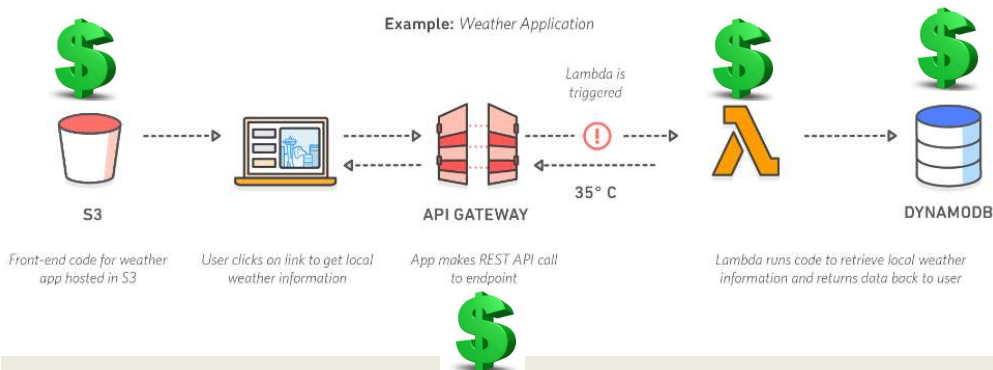
October 21, 2020

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

L7.27

## CLOUD NATIVE SOFTWARE ARCHITECTURE

- Every service with a different pricing model



October 21, 2020

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

L7.28

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/UNIX (Amazon VPC)

Instance Type: c1.xlarge

October 21, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.29
------------------	------------------------------------------------------------------------------------------------------------------------------------------	-------

FAAS COMPUTING BILLING MODELS

- AWS Lambda Pricing
- 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

(more common) \$0.00001667 to rent GB/sec

October 21, 2020

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

L7.30

Assume a month equals  
30.416667 days  
365 days / 12 months

- ON AWS Lambda
  - Each service call: 100% of 1 CPU-core  
100% of 3GB of memory (*max function instance*)
  - Workload: 2 continuous client threads
  - Duration: 1 month (30.41667 days)
- ON AWS EC2:
  - Amazon EC2 c5.large 2-vCPU VM x 4GB
  - Hosting cost: \$62.05/month  
c4.large: 8.5¢/hour, 24 hrs/day x 30.41667 days
- How much would hosting this workload cost on AWS Lambda?

October 21, 2020

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

L7.31

PRICING OBFUSCATION

- Workload: (3GB) 7.884.000 GB-sec
- FF
- Ch
- M
- In
- FREE: - 1,000,000 calls
- Charge: 1,628,000 calls
- Calls: \$.33
- Total: \$131.76
- BREAK-EVEN POINT = ~3,924,455 GB-sec-month  
~15.14 days

Worst-case scenario= ~2.12x !

AWS EC2: \$62.05

AWS Lambda: \$131.76



## 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 21, 2020

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

L7.33

## 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 21, 2020

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

L7.34

## 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?

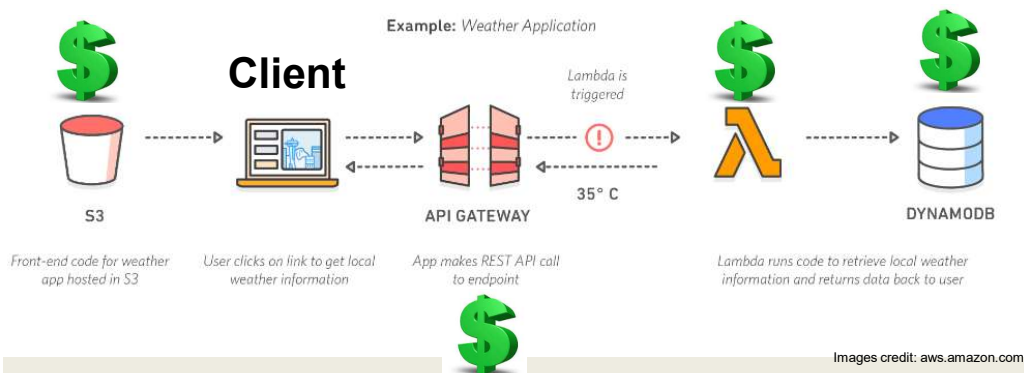
October 21, 2020

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

L7.35

## VENDOR ARCHITECTURAL LOCK-IN

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



- 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

October 21, 2020

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

L7.37

## 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...”*

▼ Basic settings

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

3008 MB

Timeout Info  
3 min 0 sec

Description

**Performance**

- **But how much memory do model services require?**

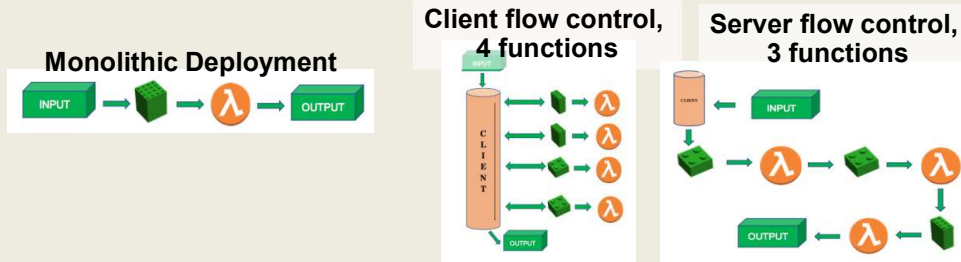
October 21, 2020

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

L7.38

## SERVICE COMPOSITION

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



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



## 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



**Performance**



Image from: Denver7 – The Denver Channel News




# FUNCTION-AS-A-SERVICE

AWS  
Lambda  
Demo

41

# THE SERVERLESS APPPLICATION ANALYTICS FRAMWORK

<https://tinyurl.com/y4ulvl9k>



**The Serverless Application Analytics Framework:  
Enabling Design Trade-off Evaluation for Serverless Software**

Robert Cordingly, Hanfei Yu, Varik Hoang, Zohreh Sadeghi, David Foster, David Perez, Rashad Hatchett, Wes Lloyd  
School of Engineering and Technology  
University of Washington  
Tacoma WA USA  
rcording, hanfeiyu, varikmp, zsadeghi, davidf94, daperez, rhatch26, wlloyd@uw.edu

**ABSTRACT**

To help better understand factors that impact performance on Function-as-a-Service (FaaS) platforms we have developed the Serverless Application Analytics Framework (SAAF). SAAF provides a reusable framework supporting multiple programming languages that developers can integrate into a function's package for deployment to

**1 Introduction**

In recent years Function-as-a-service (FaaS) platforms have arisen offering many desirable features for applications deployed to the cloud. FaaS platforms offer high availability, fault tolerance, automatic scaling, while billing developers only for the runtime of functions. As runtime is the primary factor driving hosting costs, it is

## 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 21, 2020

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

L7.43

## 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 21, 2020

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

L7.44

## 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 21, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.45
------------------	------------------------------------------------------------------------------------------------------------------------------------------	-------

## 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 21, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L10.46
------------------	------------------------------------------------------------------------------------------------------------------------------------------	--------

## OBJECTIVES – 10/21

- Questions from 10/19
- Quiz 1 – to be posted on Canvas
- From: Cloud Computing Concepts, Technology & Architecture:  
Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- 2<sup>nd</sup> hour:
- Class Activity #2
- Term project questions
- Team planning

October 21, 2020

TCS5562: Software Engineering for Cloud Computing [Fall 2020]  
School of Engineering and Technology, University of Washington - Tacoma

L7.47

## CLOUD DEPLOYMENT MODELS

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

October 21, 2020

TCS5562: Software Engineering for Cloud Computing [Fall 2020]  
School of Engineering and Technology, University of Washington - Tacoma

L7.48



# PUBLIC CLOUDS

The diagram illustrates the public cloud model. At the bottom, three server racks are labeled 'organizations'. Three large upward-pointing arrows connect these organizations to a central cloud. Inside the cloud, several major cloud service providers are listed: Google, Salesforce, Microsoft, Yahoo, Amazon, Zoho, and Rackspace.

October 21, 2020

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

L7.49

# 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

The diagram illustrates the community cloud model. At the bottom, six server racks are labeled 'community of organizations'. Three large upward-pointing arrows connect these organizations to a central cloud. Inside the cloud, various cloud resources are shown, including server racks, storage cylinders, and a yellow sphere. The cloud is labeled 'community cloud' at the top.

October 21, 2020

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

L7.50

# PRIVATE CLOUD

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

October 21, 2020      TCSS562: Software Engineering for Cloud Computing [Fall 2020]  
School of Engineering and Technology, University of Washington - Tacoma      L7.51

# 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 21, 2020      TCSS562: Software Engineering for Cloud Computing [Fall 2020]  
School of Engineering and Technology, University of Washington - Tacoma      L7.52

## OTHER CLOUDS

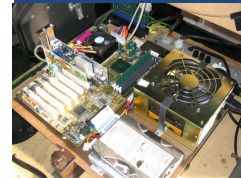
- **Federated cloud**
  - Simply means to aggregate two or more clouds together
  - Hybrid is typically private-public
  - Federated can be public-public, private-private, etc.
  - Also called inter-cloud
- **Virtual private cloud**
  - Google and Microsoft simply call these virtual networks
  - Ability 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 21, 2020

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

L7.53

WE WILL RETURN AT  
~7:05PM



## OBJECTIVES – 10/21

- Questions from 10/19
- Quiz 1 – to be posted on Canvas
- From: Cloud Computing Concepts, Technology & Architecture:  
Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- 2<sup>nd</sup> hour:
  - Class Activity #2
  - Term project questions
  - Team planning

October 21, 2020

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

L7.55

## CLASS ACTIVITY 2

- Horizontal Scaling in the Cloud
- Cost Comparison Activity
- For 2 to 3 person teams
- Using breakout rooms
- 1 person submits completed worksheet as PDF on Canvas
- Link:
  - <https://tinyurl.com/y3czofn9>

October 21, 2020

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

L7.56

OBJECTIVES – 10/21

- Questions from 10/19
- Quiz 1 – to be posted on Canvas
- From: Cloud Computing Concepts, Technology & Architecture:  
Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- 2<sup>nd</sup> hour:
- Class Activity #2
- Term project questions
- Team planning



October 21, 2020	TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.57
------------------	-----------------------------------------------------------------------------------------------------------------------------------------	-------

OBJECTIVES – 10/21

- Questions from 10/19
- Quiz 1 – to be posted on Canvas
- From: Cloud Computing Concepts, Technology & Architecture:  
Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- 2<sup>nd</sup> hour:
- Class Activity #2
- Term project questions
- Team planning

October 21, 2020	TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.58
------------------	-----------------------------------------------------------------------------------------------------------------------------------------	-------

# TCSS 562 TERM PROJECT



October 21, 2020

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

L7.59

## 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 21, 2020

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

L7.60

## 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 21, 2020

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

L7.61

## 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 21, 2020

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

L7.62

## OTHER PROJECT IDEAS

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

October 21, 2020

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

L7.63

## 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 21, 2020

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

L7.64



## 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 21, 2020

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

L7.65

## 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 21, 2020

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

L7.66

# SERVICE COMPOSITION

Remote Client

API Gateway

Fine grained services

A	B	C	3 services Full Service Isolation
A	B	C	2 services
A	B	C	2 services
A	B	C	1 service Full Service Aggregation

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

October 21, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.67
------------------	------------------------------------------------------------------------------------------------------------------------------------------	-------

# 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 21, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.68
------------------	------------------------------------------------------------------------------------------------------------------------------------------	-------

## 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 21, 2020

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

L7.69

## APPLICATION FLOW CONTROL - 2

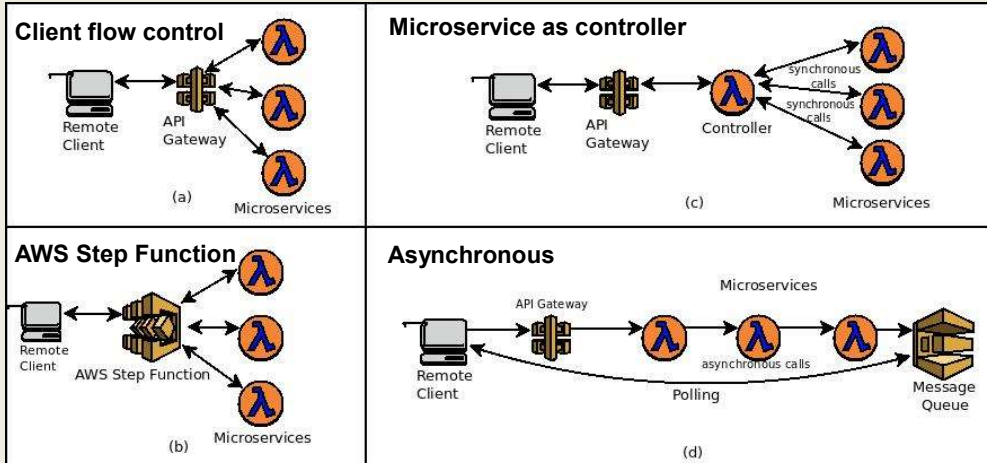
- **Asynchronous web service**
  - Client calls service
  - Server responds to client with OK message
  - Client closes connection
  - Server performs the work associated with the service
  - Server posts service result in an external data store
    - AWS: S3, SQS (queueing service), SNS (notification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

October 21, 2020

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

L7.70

## APPLICATION FLOW CONTROL - 3



October 21, 2020

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

L7.71

## 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 21, 2020

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

L7.72

## 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

October 21, 2020

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

L7.73

## 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 21, 2020

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

L7.74

## 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 21, 2020

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

L7.75

## 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 21, 2020

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

L7.76

## 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 8<sup>th</sup> column returned
  - Metric can be read using SAAF in tutorial #4

October 21, 2020

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

L7.77

## 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 21, 2020

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

L7.78

QUESTIONS




October 21, 2020

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

L7.79

QUESTIONS



October 21, 2020

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

L7.80



# TCSS 562 OFFICE HOURS

*PLEASE SAY HELLO*



L7.81

## 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
- **2<sup>nd</sup> hour:**
  - Introduce Tutorial #3 – Best Practices for Working with Virtual Machines on Amazon EC2
  - Term project case studies
  - Team planning

October 21, 2020	TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L7.82
------------------	-----------------------------------------------------------------------------------------------------------------------------------------	-------

AMDAHL'S LAW

- Portion of computation which cannot be parallelized determines the overall speedup
- For an embarrassingly parallel job of fixed size
- Assume 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

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

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

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

October 21, 2020

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

L7.83

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

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

October 21, 2020

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

L7.84

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 21, 2020

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

L7.85

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 that is parallelized (e.g. must be 0 ≤ α ≤ 1)

Can be used to determine the number of processors required to achieve a given speed-up

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

October 21, 2020

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

L7.86

## GUSTAFSON'S LAW

- Calculates the scaled speed-up using “N” processors

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

N: Number of processors

$\alpha$ : 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  $\sigma$  = sequential time,  $\pi$  = parallel time
- → NEXT TIME will work to provide examples...

October 21, 2020

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

L7.87