# TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

## Cloud Computing Concepts and Models

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

TR 5:50-7:50 PM

1

## OFFICE HOURS – FALL 2022

- **Tuesdays:**
  - 4:20 to 5:20 pm - CP 229
- **Fridays**
  - 12:00 to 1:00 pm – ONLINE via Zoom
- Or email for appointment

*> Office Hours set based on Student Demographics survey feedback*



2

## OBJECTIVES – 10/20

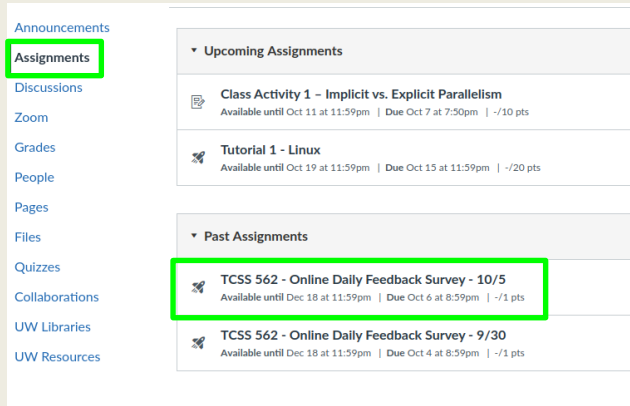
- **Questions from 10/18**
- Tutorials Questions
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms



3

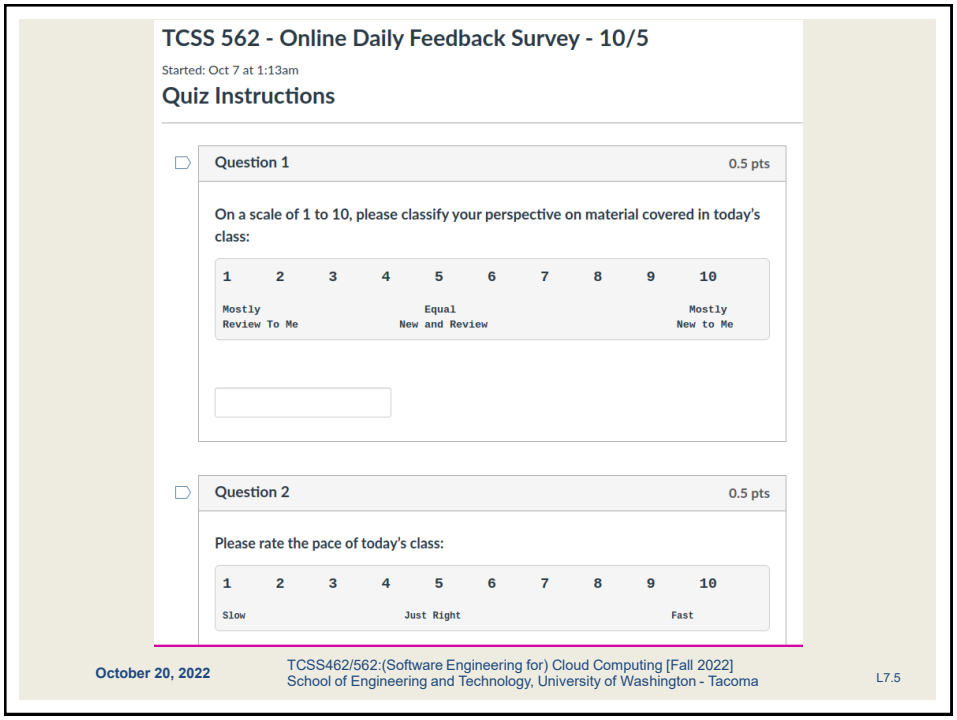
## ONLINE DAILY FEEDBACK SURVEY

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

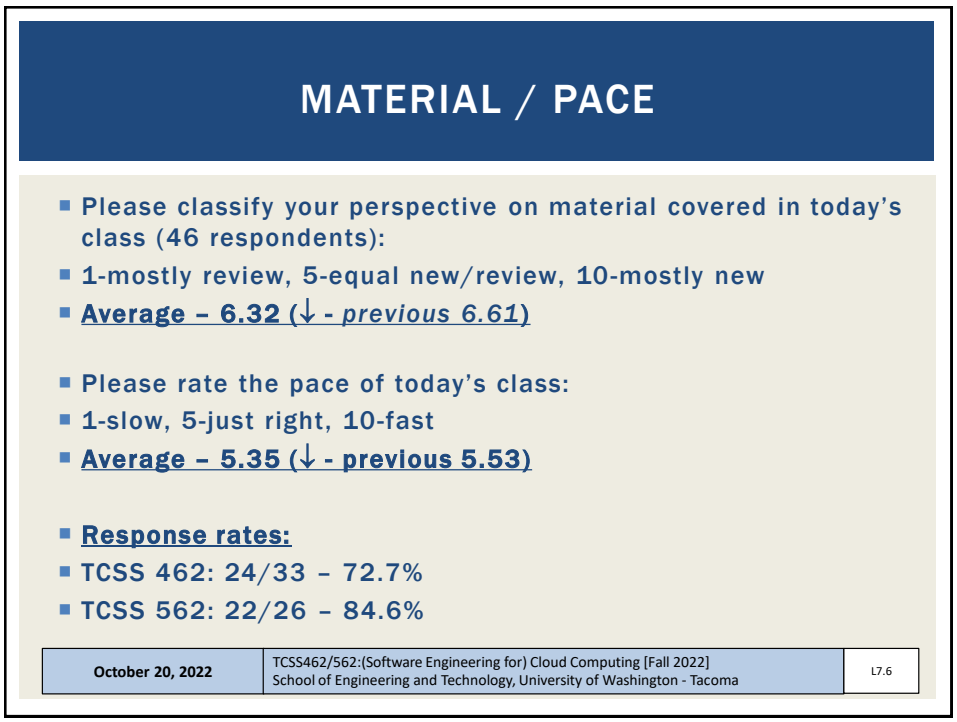





4


TCSS 562 - Online Daily Feedback Survey - 10/5
Started: Oct 7 at 1:13am
Quiz Instructions
Question 1
0.5 pts
On a scale of 1 to 10, please classify your perspective on material covered in today's class:
1 2 3 4 5 6 7 8 9 10
Mostly Review To Me
Equal New and Review
Mostly New to Me
Question 2
0.5 pts
Please rate the pace of today's class:
1 2 3 4 5 6 7 8 9 10
Slow
Just Right
Fast
October 20, 2022
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma
L7.5

5


MATERIAL / PACE
Please classify your perspective on material covered in today's class (46 respondents):
1-mostly review, 5-equal new/review, 10-mostly new
Average – 6.32 (↓ - previous 6.61)
Please rate the pace of today's class:
1-slow, 5-just right, 10-fast
Average – 5.35 (↓ - previous 5.53)
Response rates:
TCSS 462: 24/33 – 72.7%
TCSS 562: 22/26 – 84.6%
October 20, 2022
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma
L7.6

6

## FEEDBACK FROM 10/18

- ***When would we use Amdahl's law vs. scaled speedup (Gustafson's)? Why wouldn't we always use scaled speedup?***
  - Amdahl's law is helpful to estimate the speedup when the size of the computer is unknown or when wanting to estimate the speed-up outside the context of a specific machine (server)
  - Scaled speedup will further refine the expected speed-up (x factor) for a specific computer/server



7

## FEEDBACK - 2

- ***I have submitted the weather.sh last week and I just tested it that it was able to show 14 days forecast. But I heard that the 14 days forecast is only for new users who created an account in the last 30 days. Should I resubmit a 7 days forecast version?***
  - Any script producing a forecast of 7 days or more is fine
- ***But if I resubmit it, the file name will be changed to weather - 1.sh by Canvas. Is it ok?***
  - There is no problem if the file is renamed by resubmitting



8

## AWS CLOUD CREDITS

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



9

## OBJECTIVES – 10/20

- **Questions from 10/18**
- Tutorials Questions
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms



10

## TUTORIAL 2

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



11

## TUTORIAL 0

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



12

## TUTORIAL 3

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



## OBJECTIVES – 10/20

- **Questions from 10/18**
- Tutorials Questions
- Tutorial 4 – Intro to FaaS – AWS Lambda
- From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- 2nd hour:
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

## TUTORIAL 4

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



15

## CLOUD COMPUTING: CONCEPTS AND MODELS



16

## OBJECTIVES – 10/20

- **Questions from 10/18**
- Tutorials Questions
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms



17

## 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)



18

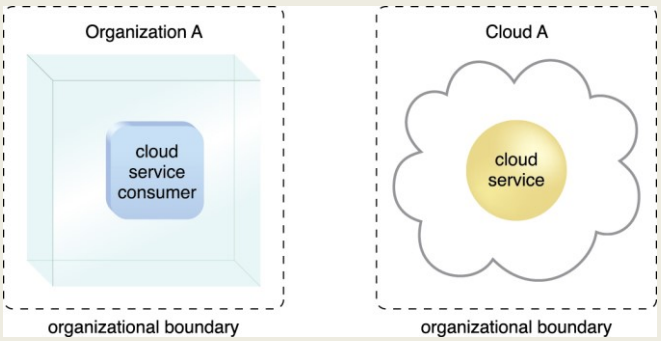
## ROLES - 2

- **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



19

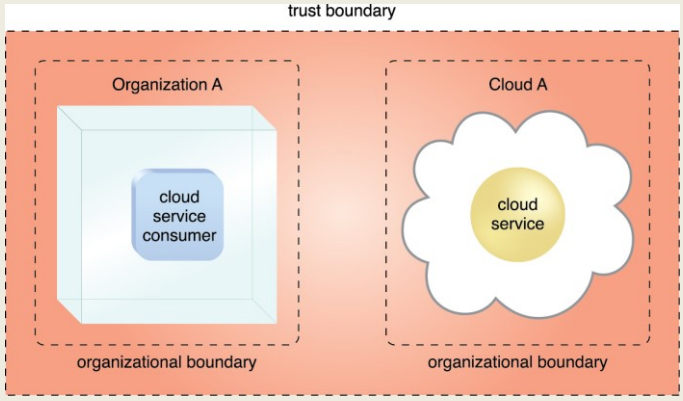
## ORGANIZATION BOUNDARY






20

## TRUST BOUNDARY






21

## OBJECTIVES – 10/20

- **Questions from 10/18**
- Tutorials Questions
- Tutorial 4 – Intro to FaaS – AWS Lambda
- From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- 2nd hour:
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms



22

## CLOUD CHARACTERISTICS

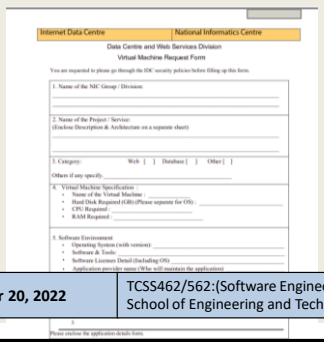
- On-demand usage
- Ubiquitous access
- Multitenancy (resource pooling)
- Elasticity
- Measured usage
- Resiliency

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



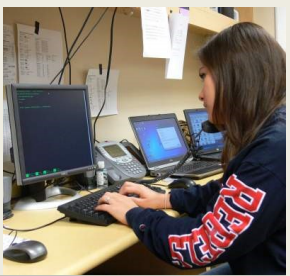
## ON-DEMAND USAGE

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


Internet Data Centre | National Informatics Centre

Data Centre and Web Services Division

Virtual Machine Request Form



## UBIQUITOUS ACCESS

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



25

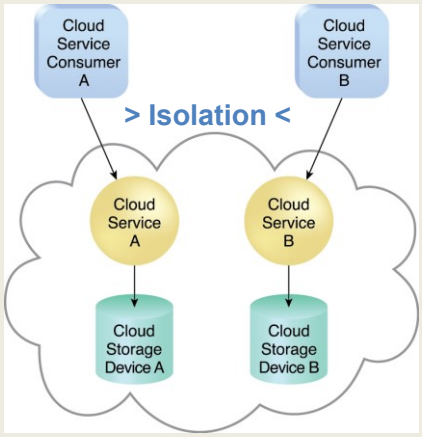
## MULTITENANCY

- 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



26

## SINGLE TENANT MODEL

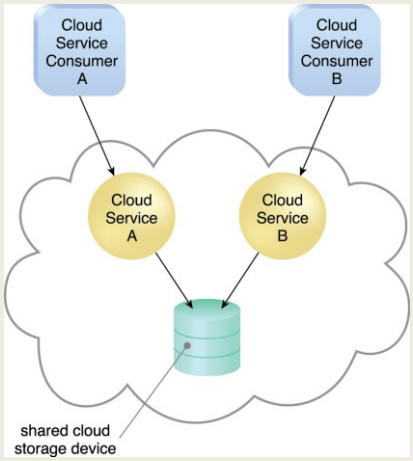





27

## MULTITENANT MODEL

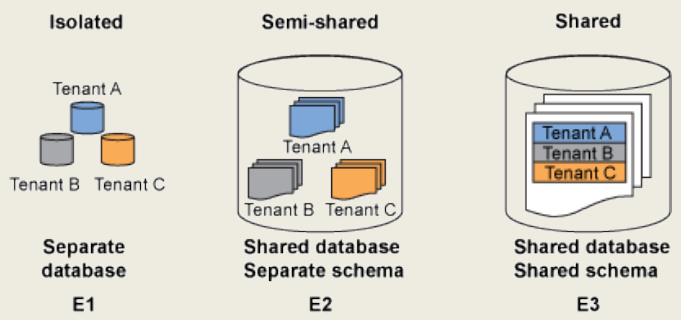
- Resource is "multiplexed" and share amongst multiple users
- Goal is to increase utilization
- Often server resources are underutilized
- There are many "sunk costs" whether usage is 0% or 100%
- Cloud computing tries to maximize "sunk cost" investments through **multi-tenancy**






28

# MULTITENANT DATABASE



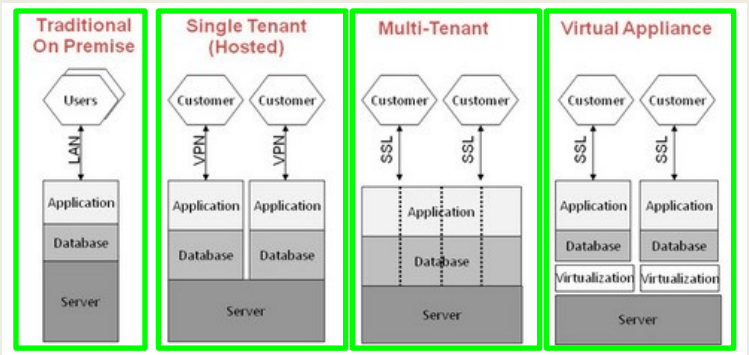

- Many users on a single database instance
- ***What issues may occur when sharing a single database instance?***



29

# MULTITENANCY OF RESOURCES

- Where is the multitenancy?
  - >> What is shared? What is isolated?



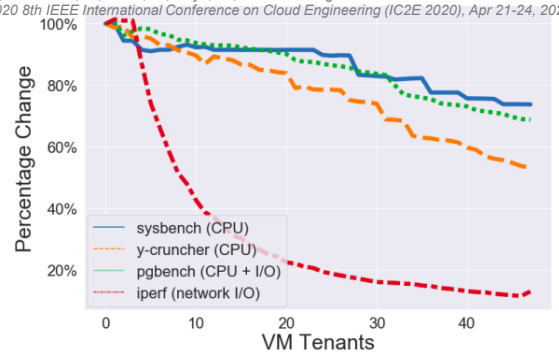



30

# RESOURCE CONTENTION FROM MUTLI-TENANCY

- Despite best efforts at isolation, co-resident VMs on a single cloud server running identical benchmarks simultaneously do not perform equally.

*From Han, X., Schooley, R., Mackenzie, D., David, O., Lloyd, W., Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction, 2020 8th IEEE International Conference on Cloud Engineering (IC2E 2020), Apr 21-24, 2020.*




***Up to 48 VMs sharing same server !!***



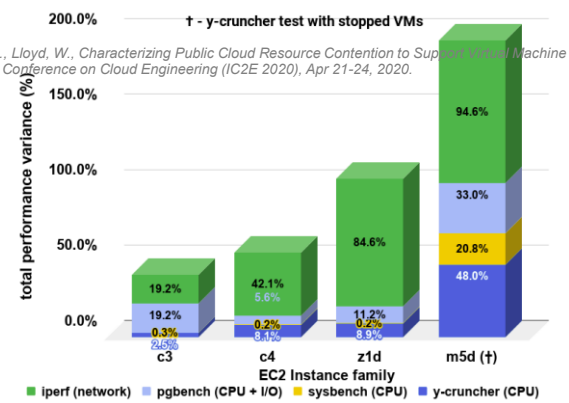
# RESOURCE CONTENTION FROM MUTLI-TENANCY - 2

- Performance variation from multi-tenancy is increasing as cloud servers add more CPU cores

*From Han, X., Schooley, R., Mackenzie, D., David, O., Lloyd, W., Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction, 2020 8th IEEE International Conference on Cloud Engineering (IC2E 2020), Apr 21-24, 2020.*

- Running many idle operating system instances can impose significant overhead for some workloads

***Maximum potential resource contention (i.e. worst-case scenario)*** →



## ELASTICITY

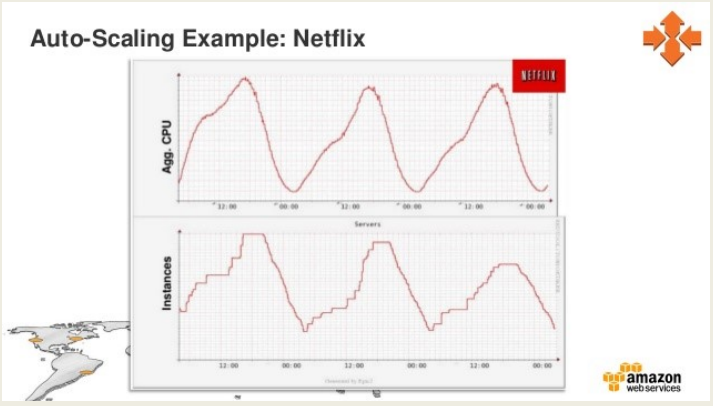
- Automated ability of cloud to transparently scale resources
- Scaling based on runtime conditions or pre-determined by cloud consumer or cloud provider
- Threshold based scaling
  - `CPU-utilization > threshold_A, Response_time > 100ms`
  - Application agnostic vs. application specific thresholds
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



33

## PREDICTABLE DEMAND

- AWS EC2 Scaling Example:




From: Kejariwal, A., 2013, March. Techniques for optimizing cloud footprint. In 2013 IEEE Int. Conf. on Cloud Engineering (IC2E), pp. 258-268.



34

## MEASURED USAGE

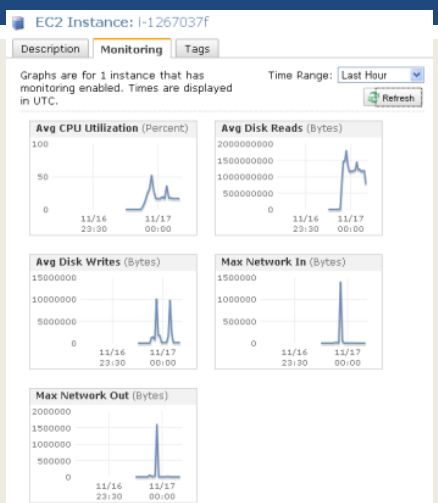
- Cloud platform tracks usage of IT resources
- For billing purposes
- Enables charging only for IT resources actually used
- Can be time-based (millisec, second, minute, hour, day)
  - Granularity is increasing…
- Can be throughput-based (data transfer: MB/sec, GB/sec)
- Can be resource/reservation based (vCPU/hr, GB/hr)

- Not all measurements are for billing
- Some measurements can support auto-scaling
- For example CPU utilization



35

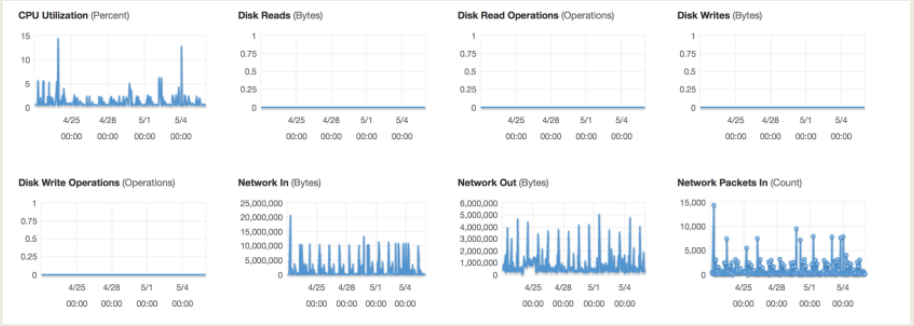
## EC2 CLOUDWATCH METRICS






36

## EC2 CLOUDWATCH METRICS

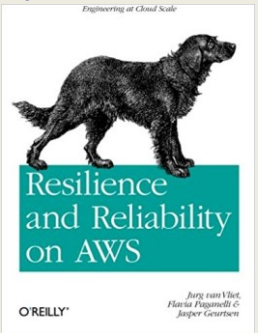





37

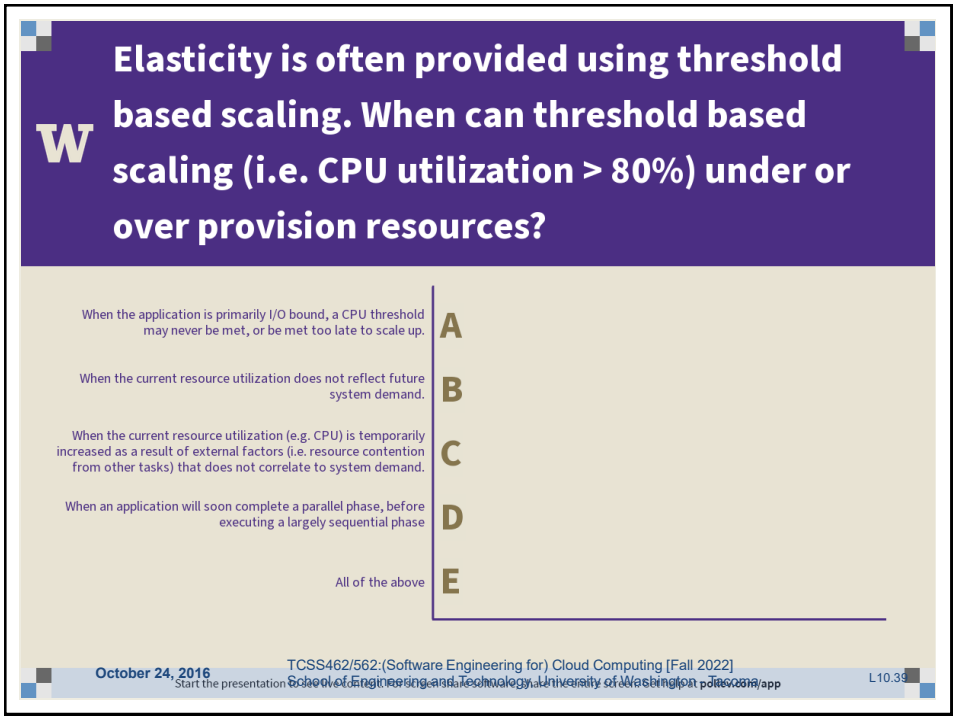
## RESILIENCY

- Distributed redundancy across physical locations (regions on AWS)
- 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

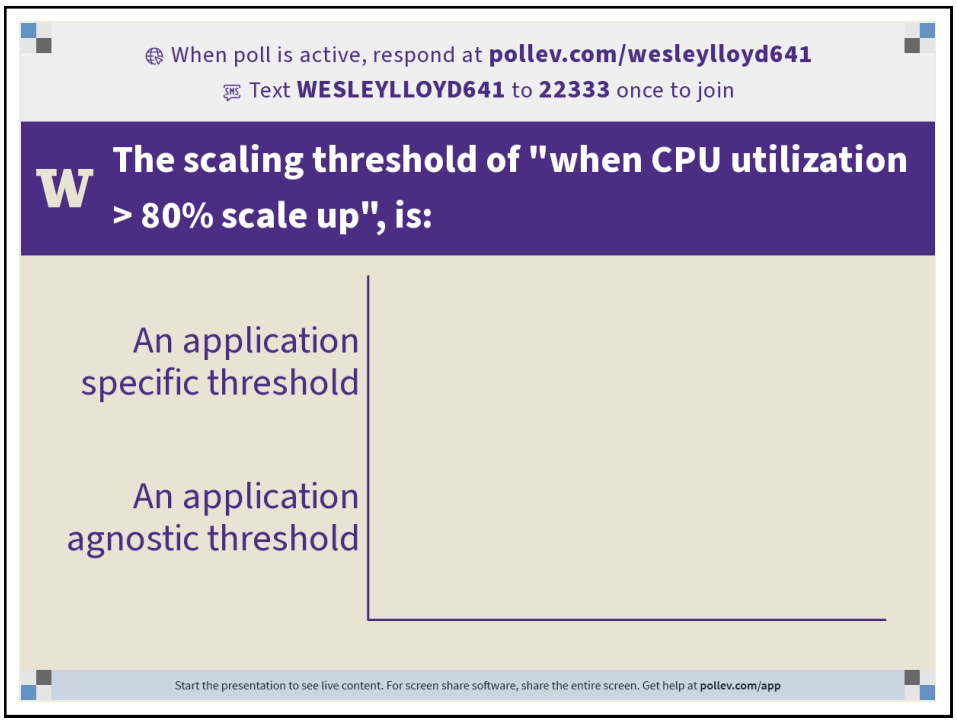





38


W
Elasticity is often provided using threshold based scaling. When can threshold based scaling (i.e. CPU utilization > 80%) under or over provision resources?
When the application is primarily I/O bound, a CPU threshold may never be met, or be met too late to scale up.
A
When the current resource utilization does not reflect future system demand.
B
When the current resource utilization (e.g. CPU) is temporarily increased as a result of external factors (i.e. resource contention from other tasks) that does not correlate to system demand.
C
When an application will soon complete a parallel phase, before executing a largely sequential phase
D
All of the above
E
October 24, 2016
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma
L10.39

39


When poll is active, respond at pollev.com/wesleylloyd641
Text WESLEYLLOYD641 to 22333 once to join
W
The scaling threshold of "when CPU utilization > 80% scale up", is:
An application specific threshold
An application agnostic threshold
Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

40

## OBJECTIVES – 10/20

- **Questions from 10/18**
- Tutorials Questions
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms



41

## 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



42

## 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



43

## CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS) delivery model
- V
- U
t

**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*



44

## CLOUD COMPUTING DELIVERY MODELS

- Infrastructure-as-a-Service (IaaS) delivery model
- V… …d
- U… t…

**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*



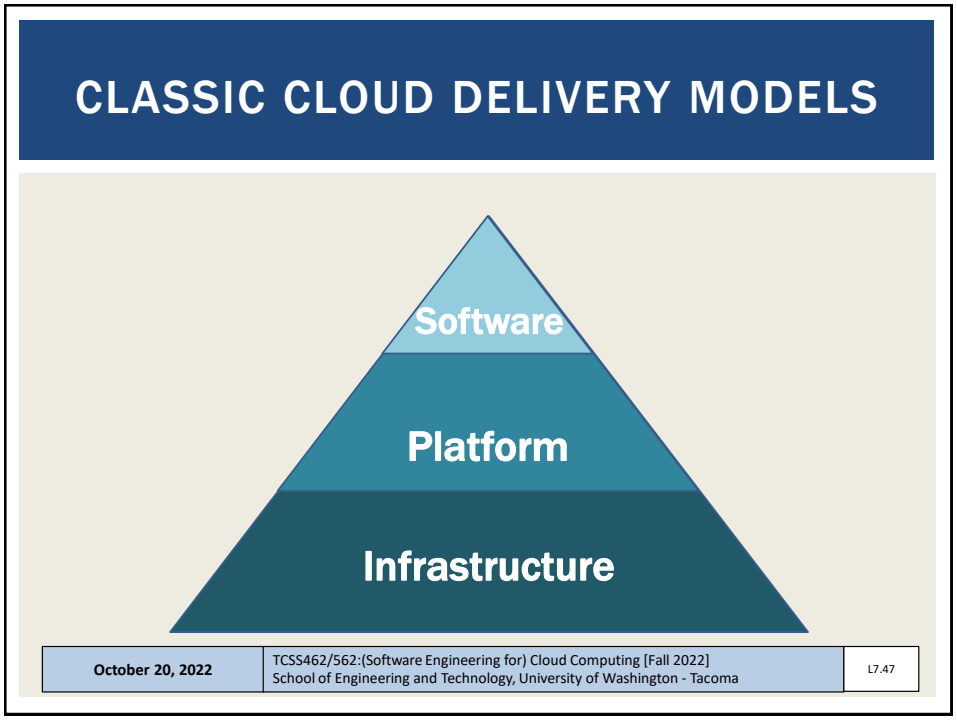
45

## CLOUD DELIVERY MODELS

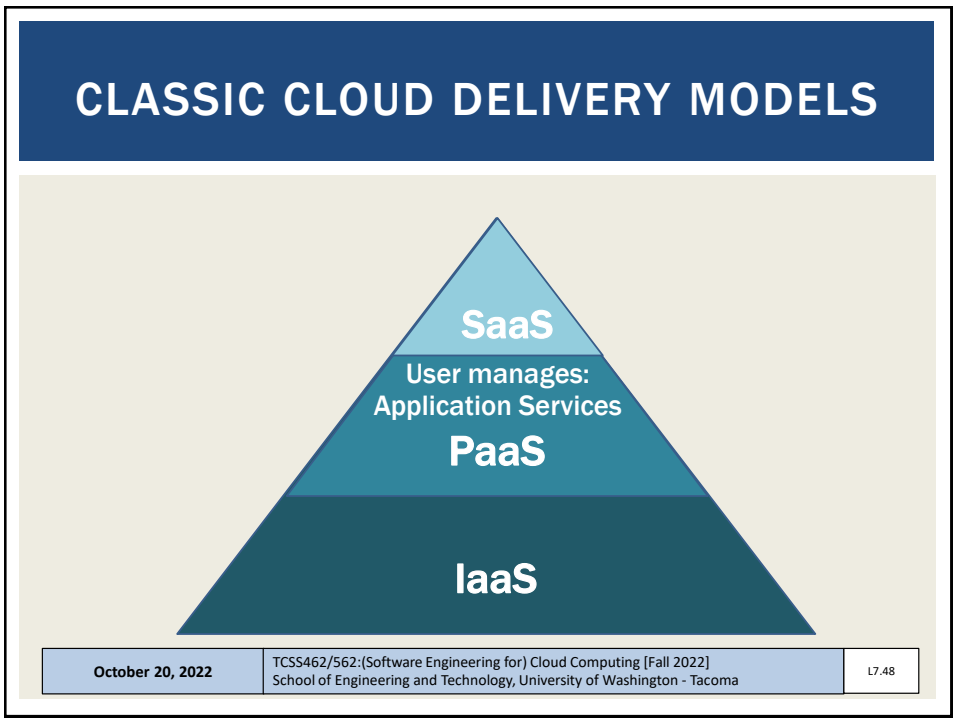
- What is the appropriate level of **abstraction**?
- How should applications be deployed?
  - IaaS, PaaS, SaaS, DbaaS, FaaS
- How do we ensure Quality-of-Service?
  - Performance, Availability, Responsiveness, Fault Tolerance
- How is ***scalability*** provided?
- As users, how do we minimize hosting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



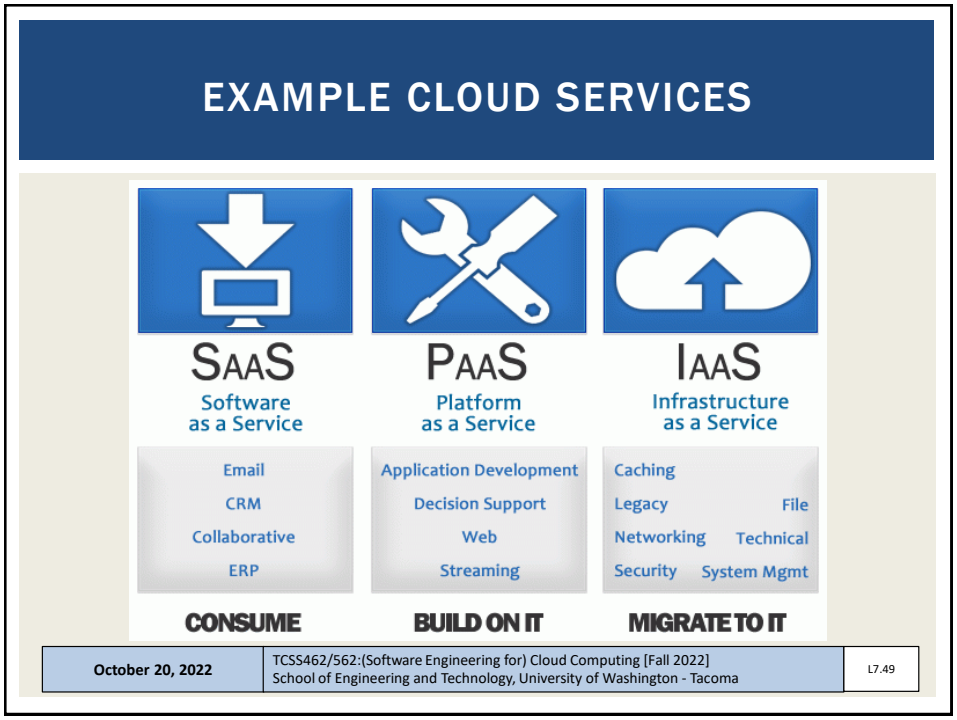
46


CLASSIC CLOUD DELIVERY MODELS
Software
Platform
Infrastructure
October 20, 2022
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma
L7.47

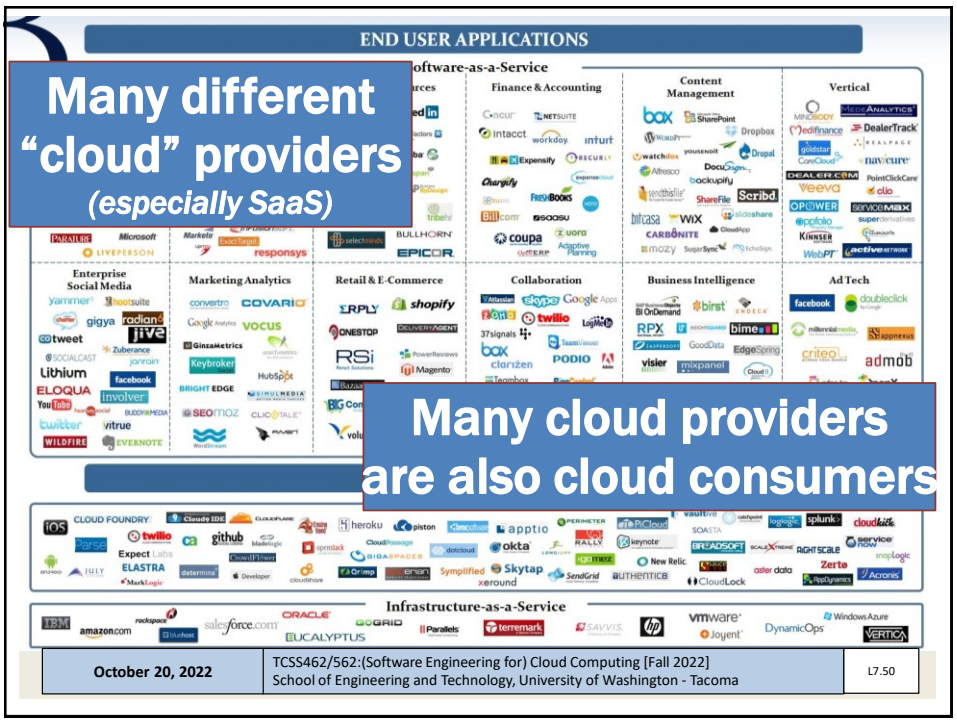
47


CLASSIC CLOUD DELIVERY MODELS
SaaS
User manages:
Application Services
PaaS
IaaS
October 20, 2022
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma
L7.48

48


EXAMPLE CLOUD SERVICES
SaaS
Software as a Service
Email
CRM
Collaborative
ERP
CONSUME
PaaS
Platform as a Service
Application Development
Decision Support
Web
Streaming
BUILD ON IT
IaaS
Infrastructure as a Service
Caching
Legacy
File
Networking
Technical
Security
System Mgmt
MIGRATE TO IT
October 20, 2022
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma
L7.49

49


Many different "cloud" providers
(especially SaaS)
END USER APPLICATIONS
Software-as-a-Service
Finance & Accounting
Content Management
Vertical
Enterprise Social Media
Marketing Analytics
Retail & E-Commerce
Collaboration
Business Intelligence
Ad Tech
Many cloud providers are also cloud consumers
Infrastructure-as-a-Service
October 20, 2022
TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma
L7.50

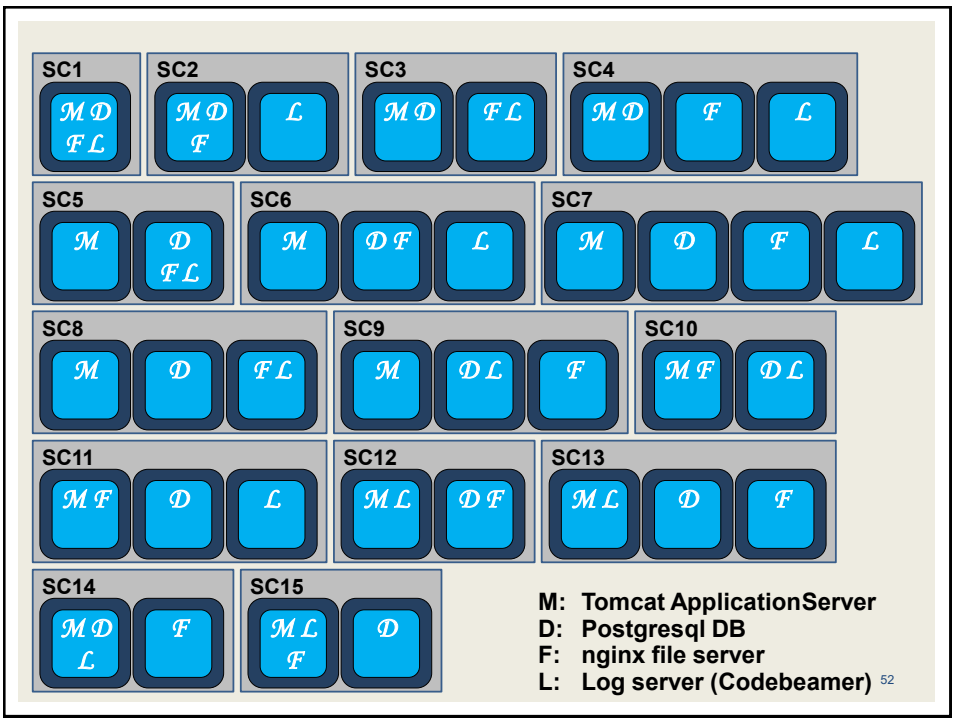
50

## INFRASTRUCTURE-AS-A-SERVICE

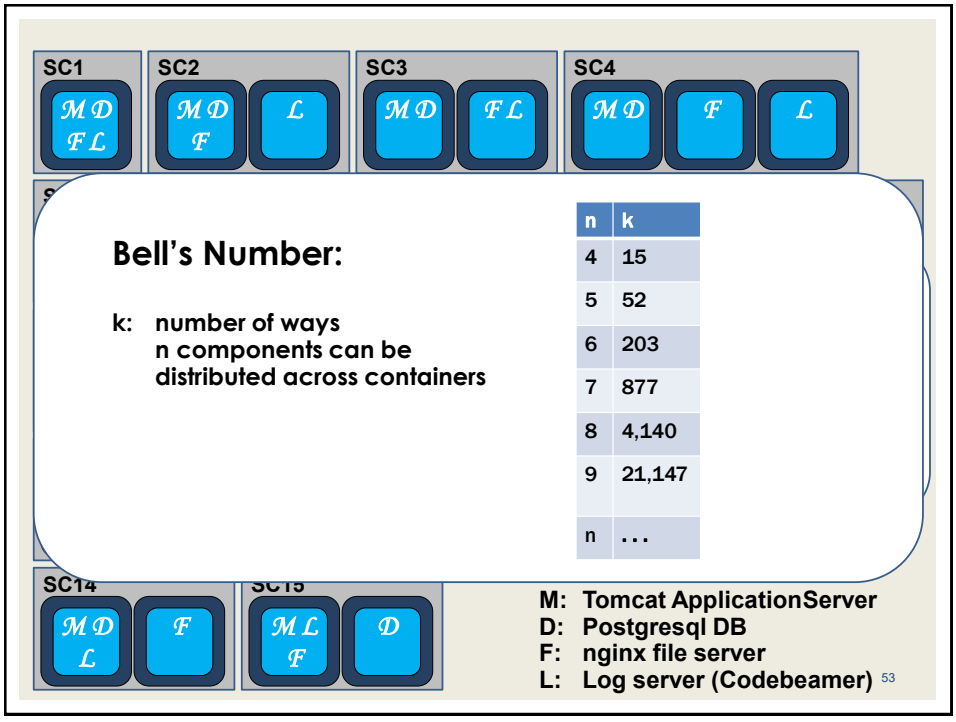
- Compute resources, on demand, as-a-service
  - Generally raw "IT" resources
  - Hardware, network, containers, operating systems
- Typically provided through virtualization
- Generally, not-preconfigured
- Administrative burden is owned by cloud consumer
- Best when high-level control over environment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



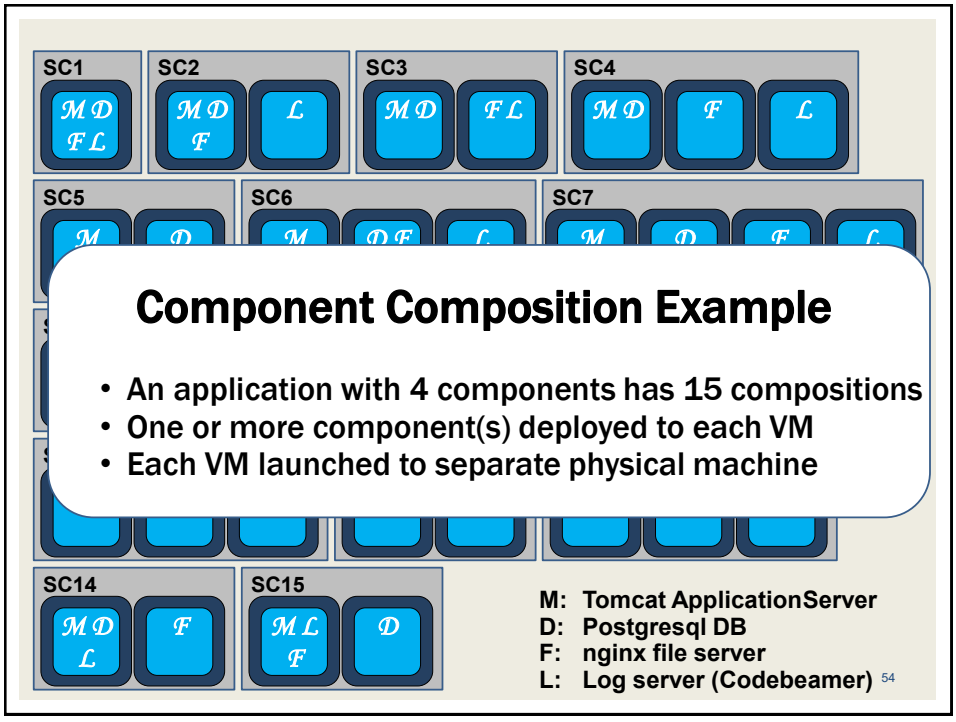
51



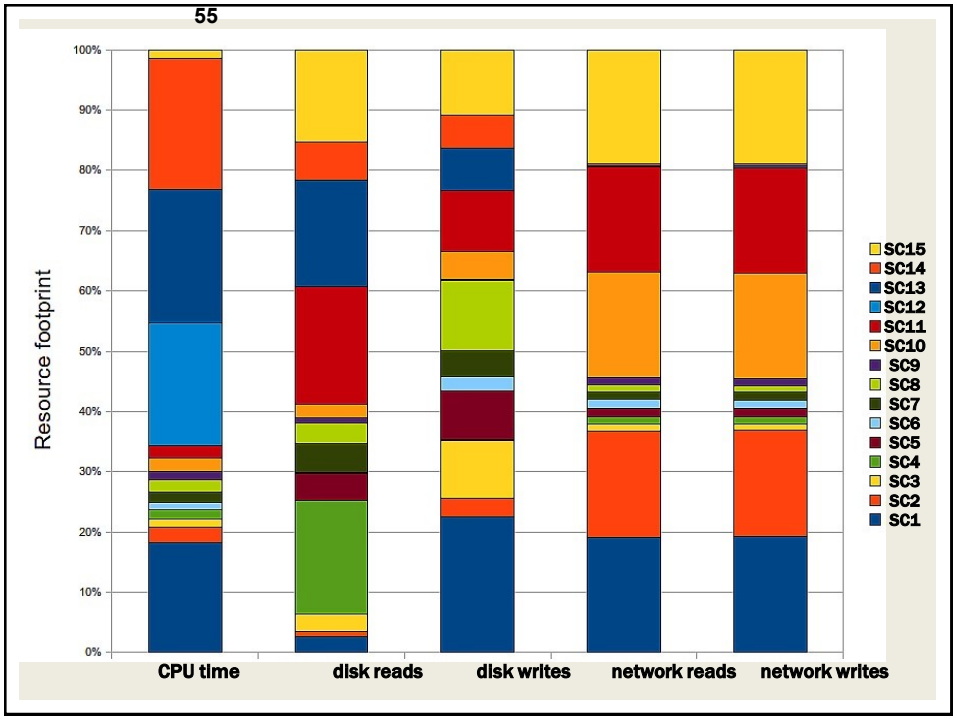

52


SC1
SC2
SC3
SC4
SC14
SC15
Bell's Number:
k: number of ways
n components can be
distributed across containers
n k
4 15
5 52
6 203
7 877
8 4,140
9 21,147
n ...
M: Tomcat ApplicationServer
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)
53

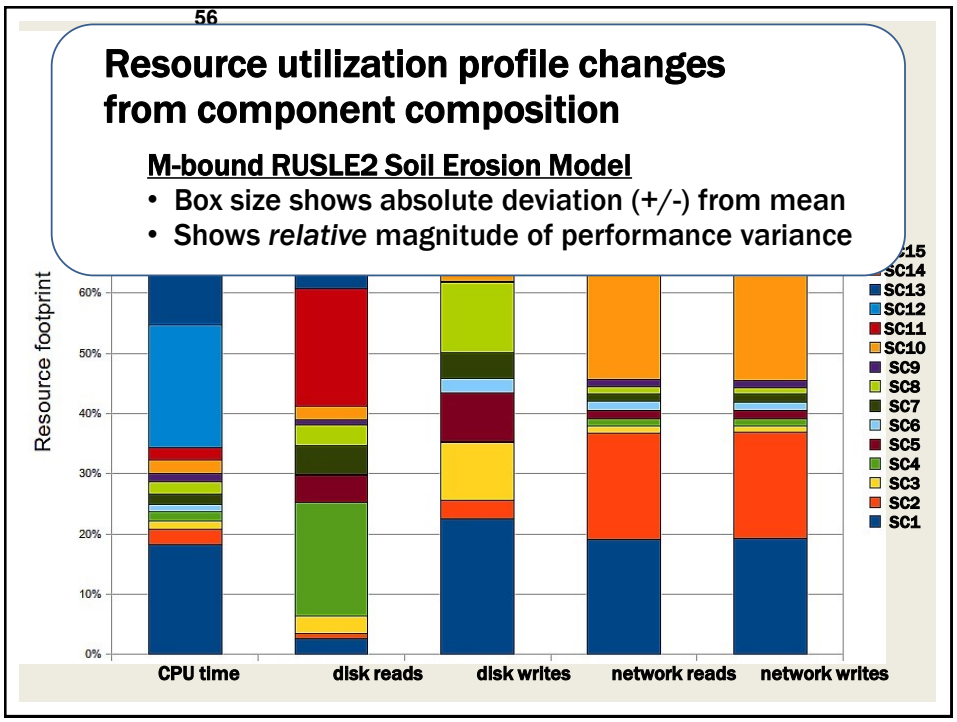
53


SC1
SC2
SC3
SC4
SC5
SC6
SC7
SC14
SC15
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
M: Tomcat ApplicationServer
D: Postgresql DB
F: nginx file server
L: Log server (Codebeamer)
54

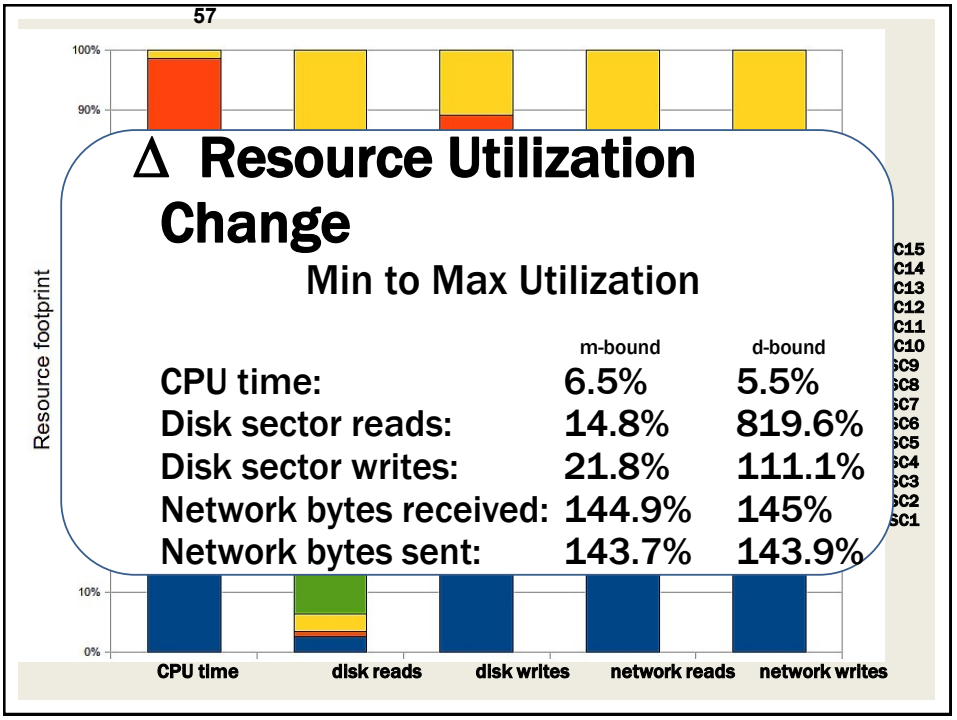
54


55
Resource footprint
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
CPU time
disk reads
disk writes
network reads
network writes
SC15
SC14
SC13
SC12
SC11
SC10
SC9
SC8
SC7
SC6
SC5
SC4
SC3
SC2
SC1

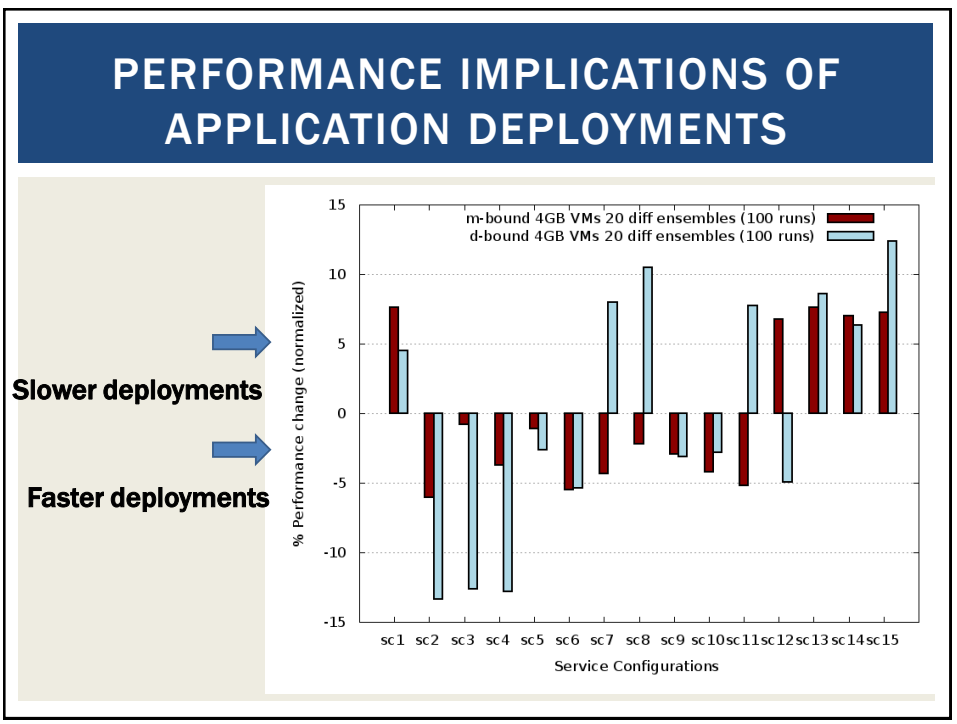
55


56
Resource utilization profile changes from component composition
M-bound RUSLE2 Soil Erosion Model
• Box size shows absolute deviation (+/-) from mean
• Shows *relative* magnitude of performance variance
Resource footprint
60%
50%
40%
30%
20%
10%
0%
CPU time
disk reads
disk writes
network reads
network writes
SC15
SC14
SC13
SC12
SC11
SC10
SC9
SC8
SC7
SC6
SC5
SC4
SC3
SC2
SC1

56


57
Δ Resource Utilization Change
Min to Max Utilization
m-bound
d-bound
CPU time: 6.5% 5.5%
Disk sector reads: 14.8% 819.6%
Disk sector writes: 21.8% 111.1%
Network bytes received: 144.9% 145%
Network bytes sent: 143.7% 143.9%
Resource footprint
100%
90%
10%
0%
C15
C14
C13
C12
C11
C10
C9
C8
C7
C6
C5
C4
C3
C2
C1
CPU time
disk reads
disk writes
network reads
network writes

57

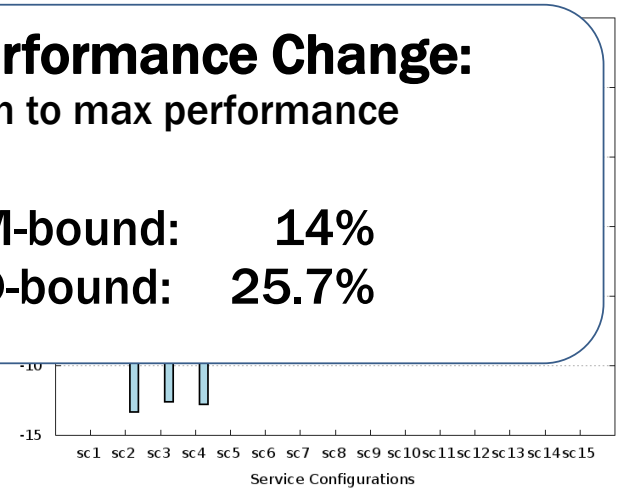

PERFORMANCE IMPLICATIONS OF APPLICATION DEPLOYMENTS
Slower deployments
Faster deployments
m-bound 4GB VMs 20 diff ensembles (100 runs)
d-bound 4GB VMs 20 diff ensembles (100 runs)
% Performance change (normalized)
15
10
5
0
-5
-10
-15
sc1 sc2 sc3 sc4 sc5 sc6 sc7 sc8 sc9 sc10 sc11 sc12 sc13 sc14 sc15
Service Configurations

58

## PERFORMANCE IMPLICATIONS OF APPLICATION DEPLOYMENTS

**Δ Performance Change:**
**Min to max performance**

**M-bound: 14%**
**D-bound: 25.7%**




59

## 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

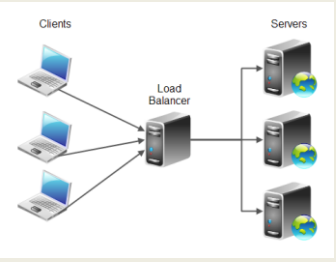


60

## 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)






61

## 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



62

## 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



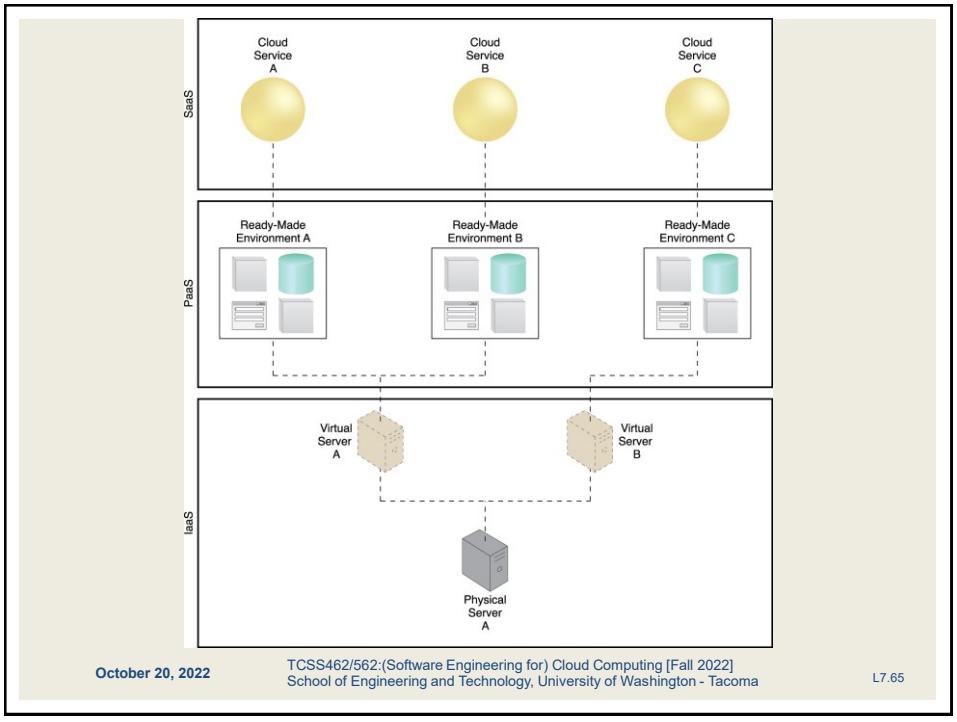
63

## 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



64




65

## CLOUD COMPUTING DELIVERY MODELS

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

**Serverless Computing:**

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

| October 20, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]<br>School of Engineering and Technology, University of Washington - Tacoma | L7.66 |
|---|---|---|

66

# SERVERLESS COMPUTING

## Introducing Cloud 2.0


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

67

# SERVERLESS COMPUTING

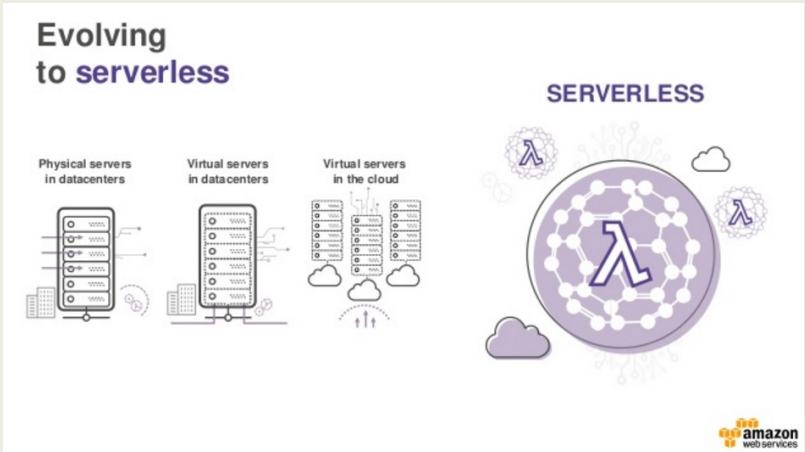

68

## SERVERLESS COMPUTING



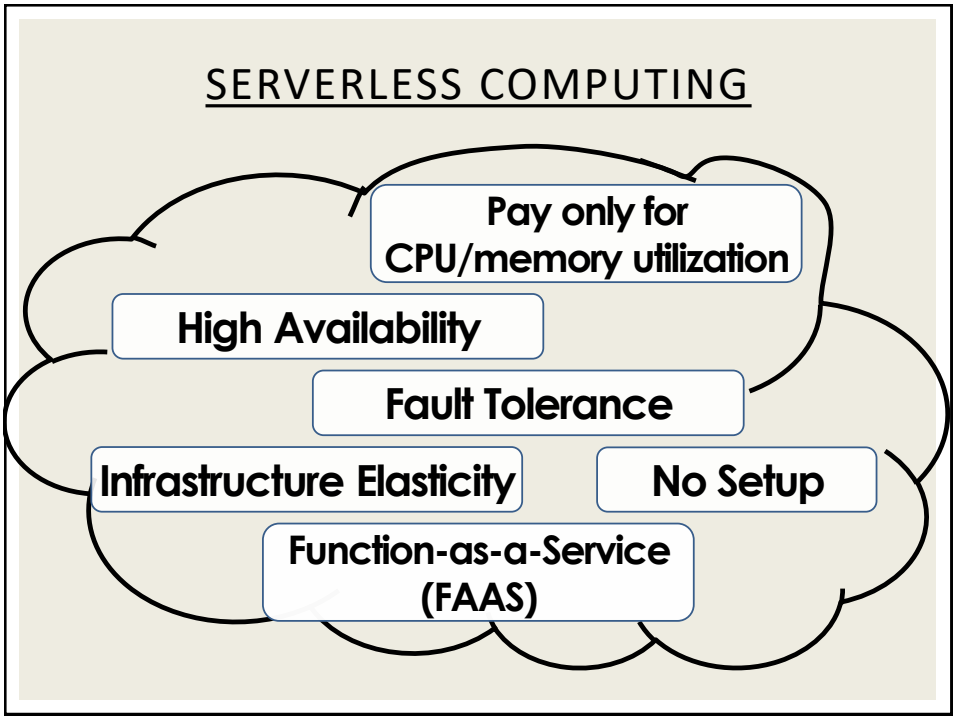

69

## SERVERLESS COMPUTING - 2

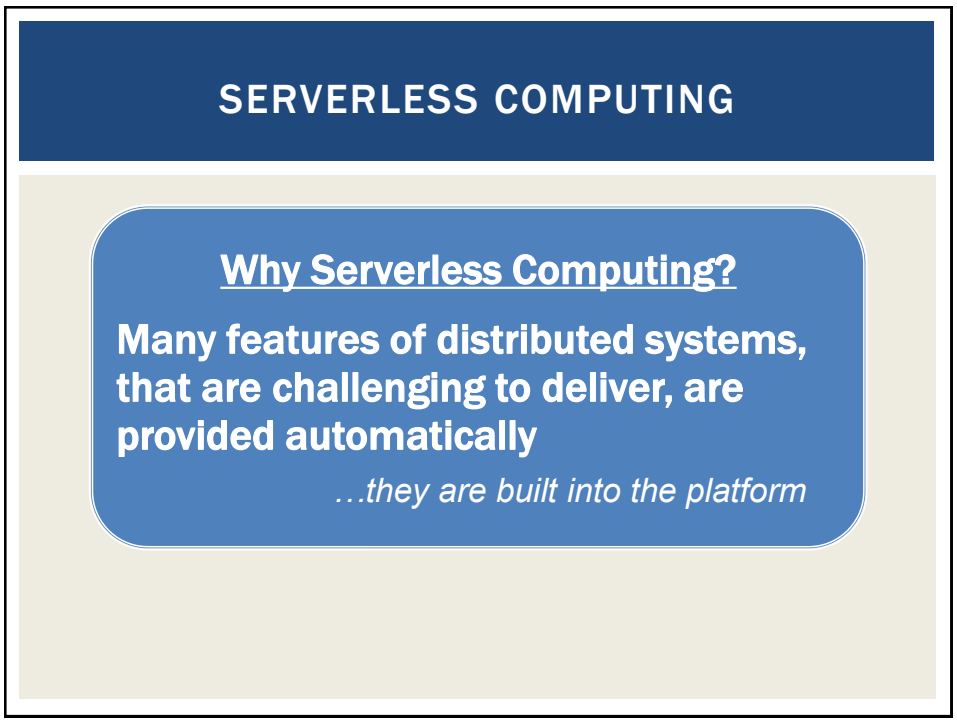





70


SERVERLESS COMPUTING
Pay only for CPU/memory utilization
High Availability
Fault Tolerance
Infrastructure Elasticity
No Setup
Function-as-a-Service (FAAS)

71


SERVERLESS COMPUTING
Why Serverless Computing?
Many features of distributed systems, that are challenging to deliver, are provided automatically
…they are built into the platform

72

## CLOUD COMPUTING DELIVERY MODELS

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

Serverless Computing:

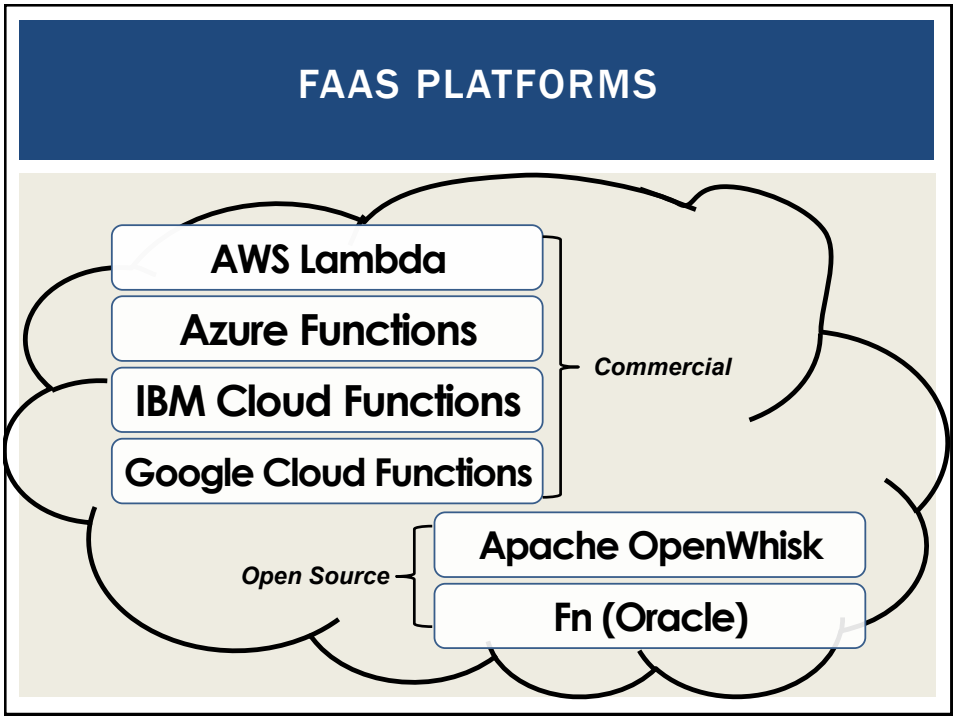
- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other Delivery Models



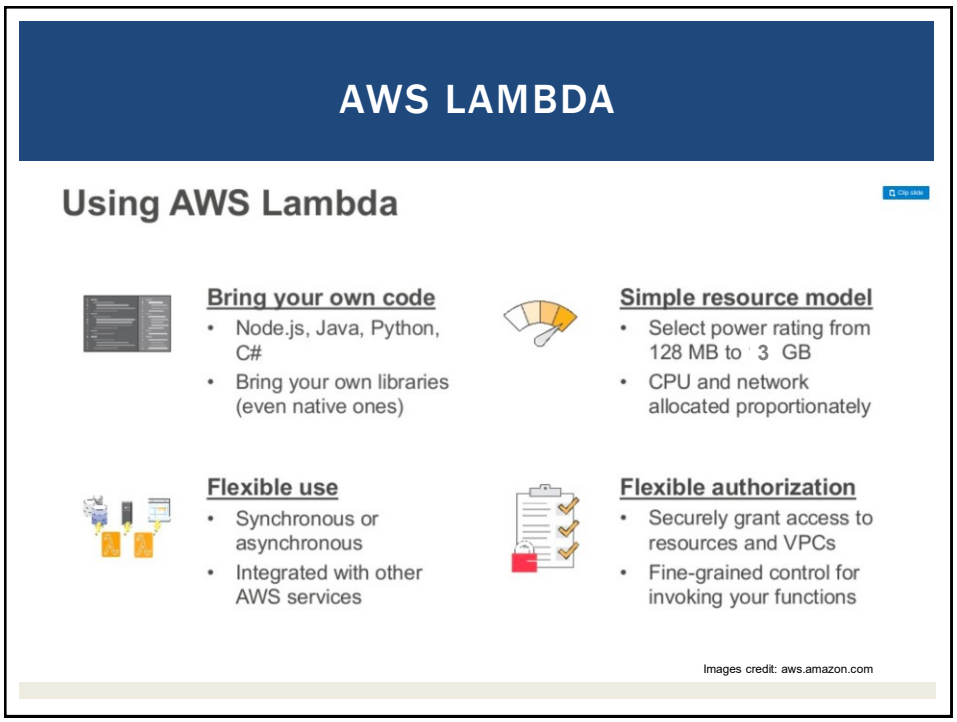
## SERVERLESS VS. FAAS

- **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…


FAAS PLATFORMS
AWS Lambda
Azure Functions
IBM Cloud Functions
Google Cloud Functions
Commercial
Apache OpenWhisk
Fn (Oracle)
Open Source

75


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

76

## 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



77

## FAAS PLATFORMS - 3

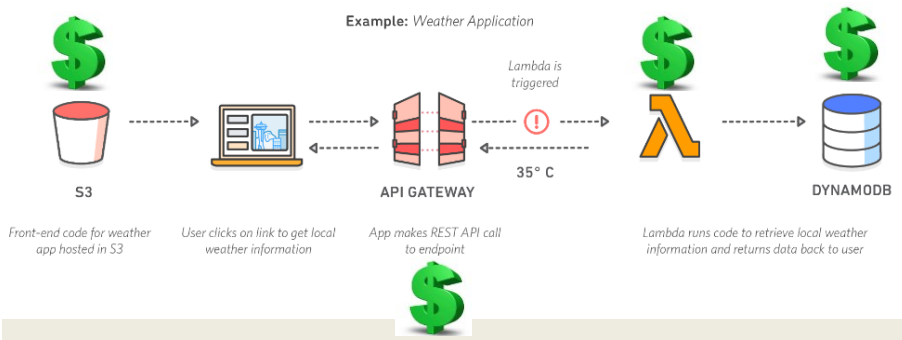
- Many challenging features of distributed systems are provided automatically
- ***Built into the platform:***
- Highly availability (24/7)
- Scalability
- Fault tolerance



78

## CLOUD NATIVE SOFTWARE ARCHITECTURE

- Every service with a different pricing model






79

## 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 →




80

## PRICING OBFUSCATION

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



81

## 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?**



82

## PRICING OBFUSCATION

- **Worst-case scenario = ~2.32x !**
  - AWS EC2: $72.00
  - AWS Lambda: $167.01
  - Break Even: 4,319,136 GB-sec
  - Two threads @2GB-ea: ~12.5 days
- **BREAK-EVEN POINT: ~4,319,136 GB-sec-month ~12.5 days 2 concurrent clients @ 2GB**

83

## 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?**



84

## 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



85

## FAAS CHALLENGES

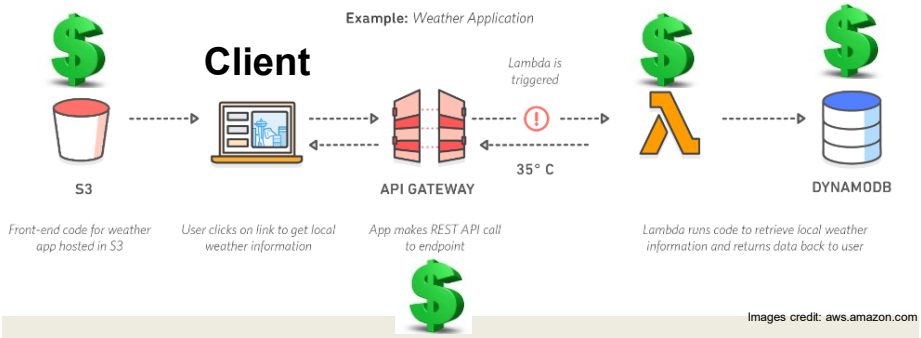
- Vendor architectural lock-in – how to migrate?
- Pricing obfuscation – is it cost effective?
- Memory reservation – how much to reserve?
- Service composition – how to compose software?
- Infrastructure freeze/thaw cycle – how to avoid?



86

## VENDOR ARCHITECTURAL LOCK-IN

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




- Increased dependencies → increased hosting costs

87

## PRICING OBFUSCATION

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

***AWS Lambda Pricing***

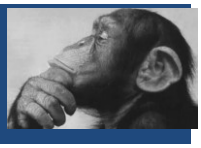
**FREE TIER:** first 1,000,000 function calls/month → FREE
first 400,000 GB-sec/month → FREE

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

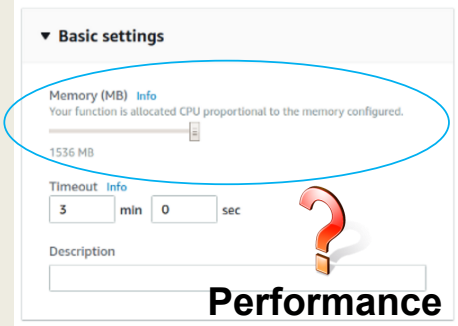


88

## 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?**



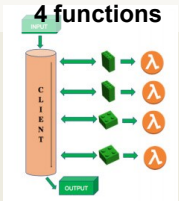
89

## SERVICE COMPOSITION

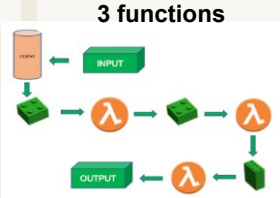
- **How should application code be composed for deployment to serverless computing platforms?**

Monolithic Deployment


Client flow control, 4 functions




Server flow control, 3 functions




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

Performance

90

## 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

91

## FUNCTION-AS-A-SERVICE

AWS Lambda Demo

92

92

## 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



## 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

## 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



95

## 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



96

## OBJECTIVES – 10/20

- **Questions from 10/18**
- Tutorials Questions
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms



97

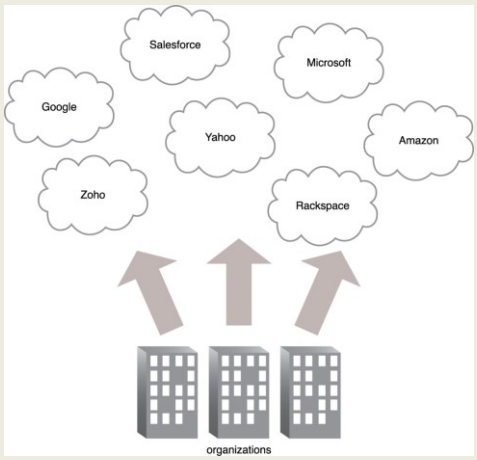
## CLOUD DEPLOYMENT MODELS

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



98

## PUBLIC CLOUDS

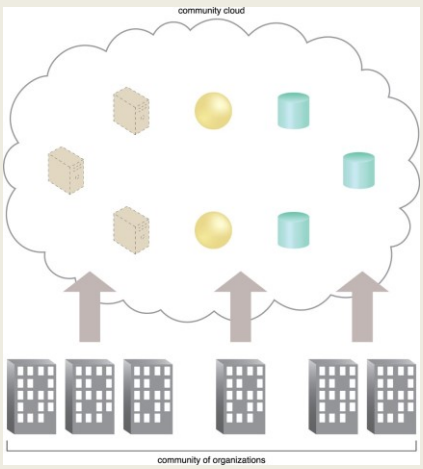





99

## 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

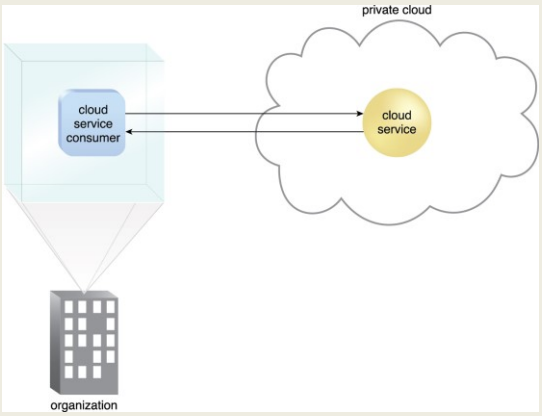





100

# PRIVATE CLOUD

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

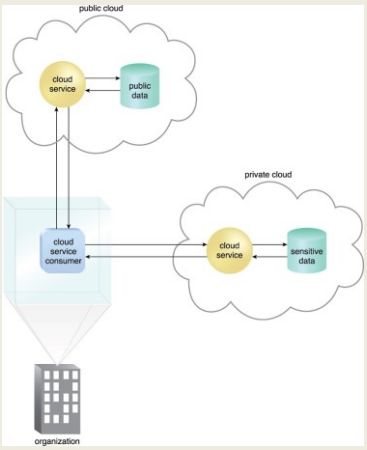





101

# 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






102

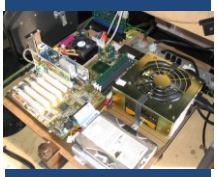
## OTHER CLOUDS

- 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



103

## WE WILL RETURN AT 7:00 PM



104

## OBJECTIVES – 10/20

- **Questions from 10/18**
- Tutorials Questions
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4: Cloud Computing Concepts and Models:**
  - Roles and boundaries
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms



105

## TCSS 462/562 TERM PROJECT



106

## TCSS 462/562 TERM PROJECT

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

**- as an individual project*



107

## 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**



108

## 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



109

## OTHER PROJECT IDEAS

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



110

## SERVERLESS APPLICATIONS

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



111

## SERVERLESS APPLICATIONS - 2

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



112

## AWS LAMBDA PLATFORM LIMITATIONS

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



113

## 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



114

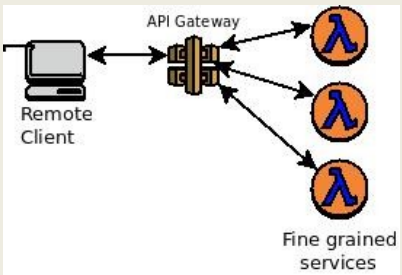
## EXTRACT TRANSFORM LOAD DATA PIPELINE - 2

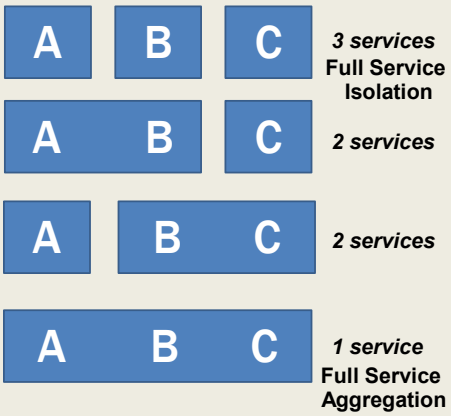
- Service 3: **QUERY**
- Using relational database, apply filter(s) and/or functions to aggregate data to produce sums, totals, averages
- Output aggregations as JSON



115

## SERVICE COMPOSITION



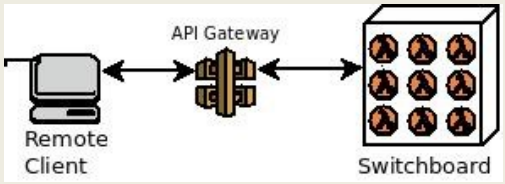




**Other possible compositions: group by library, functional cohesion, etc.**



116

## SWITCH-BOARD ARCHITECTURE




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

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

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

**Goal: reduce # of COLD starts to improve performance**



117

## 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



118

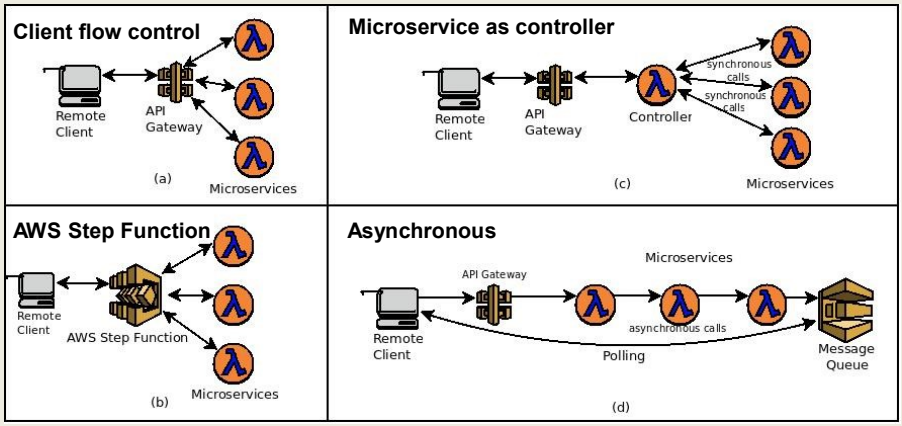
## APPLICATION FLOW CONTROL - 2

- 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)



119

## APPLICATION FLOW CONTROL - 3






120

## 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



121

## 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



122

## 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



123

## 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…



124

## 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



125

## CPUSTEAL

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



126

## 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?



127

## QUESTIONS



128