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 5:00-7:00 PM

1

OFFICE HOURS – FALL 2021

- **Tuesdays:**
  - 4:00 to 4:30 pm - CP 229
  - 7:15 to 7:45+ pm – ONLINE via Zoom
- **Thursdays**
  - 4:15 to 4:45 pm – ONLINE via Zoom
  - 7:15 to 7:45+ pm – ONLINE via Zoom
- Or email for appointment
- Zoom Link sent as Canvas Announcement

> Office Hours set based on Student Demographics survey feedback

2

OBJECTIVES – 10/21

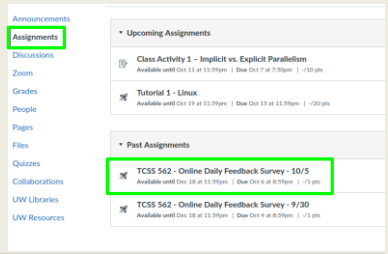
- **Questions from 10/19**
- Tutorial 3 - Best Practices with EC2
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4:** Cloud Computing Concepts and Models:
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models

- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

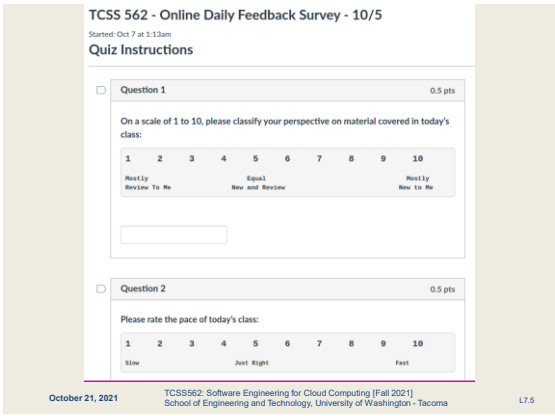
3

ONLINE DAILY FEEDBACK SURVEY

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



4



5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (24 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.00 (↓ - *previous 6.30*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.54 (↑ - previous 5.33)**

6

FEEDBACK FROM 10/19

- ***Unclear point: Challenges with shared database but non-shared schema, in the context of multi-tenancy***
  - When a physical database server is shared with many tenants, a key challenge is with resource contention (e.g. CPU, memory, disk, network)
  - Cloud provider resource constraints (limits) can restrict the resources a given user is able to acquire over time
    - For example: AWS Lambda tightly restricts CPU, disk, and network I/O resources for functions depending on their assigned memory
  - Even with resource constraints operations that for example generate excessive page faults and memory cache stress can have a detrimental impact on database performance for "isolated" users

7

OBJECTIVES – 10/21

- **Questions from 10/19**
- Tutorial 3 - Best Practices with EC2
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4:** Cloud Computing Concepts and Models:
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

8

OBJECTIVES – 10/21

- **Questions from 10/19**
- Tutorial 3 - Best Practices with EC2
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4:** Cloud Computing Concepts and Models:
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

9

CLOUD COMPUTING: CONCEPTS AND MODELS

10

OBJECTIVES – 10/21

- **Questions from 10/19**
- Tutorial 3 - Best Practices with EC2
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4:** Cloud Computing Concepts and Models:
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

11

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

12

W Elasticity is often provided using threshold based scaling. When can threshold based scaling (i.e. CPU utilization > 80%) under or over provision resources?

A. When the application is primarily I/O bound, a CPU threshold may never be met, or be met too late to scale up.
B. When the current resource utilization does not reflect future system demand.
C. 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.
D. When an application will soon complete a parallel phase, before executing a largely sequential phase
E. All of the above

13

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

14

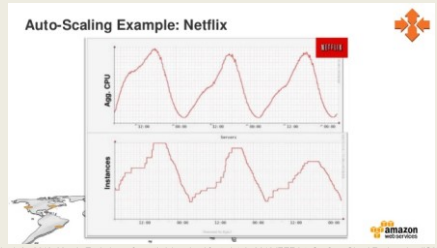
ELASTICITY

- 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

15

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.

16

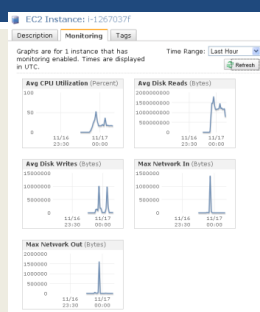
MEASURED USAGE

- 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

17

EC2 CLOUDWATCH METRICS



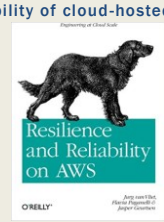
18

EC2 CLOUDWATCH METRICS

19

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



20

OBJECTIVES – 10/21

- **Questions from 10/19**
- Tutorial 3 - Best Practices with EC2
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4:** Cloud Computing Concepts and Models:
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

21

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

22

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

23

CLOUD COMPUTING DELIVERY MODELS

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

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

24

CLOUD COMPUTING DELIVERY MODELS

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

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

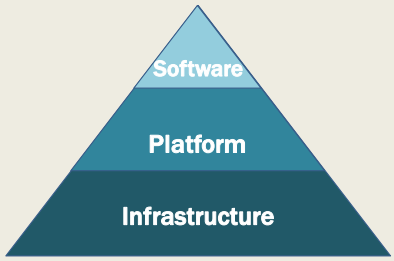
25

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?

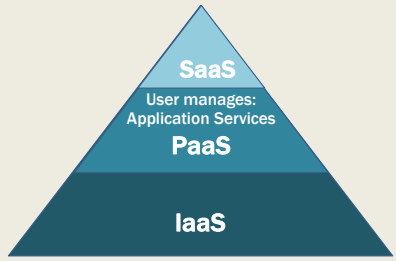
26

CLASSIC CLOUD DELIVERY MODELS



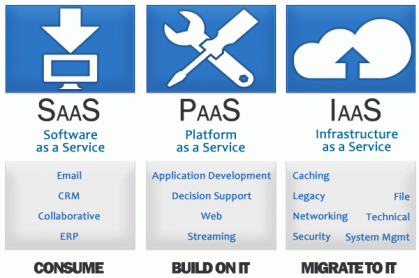
27

CLASSIC CLOUD DELIVERY MODELS

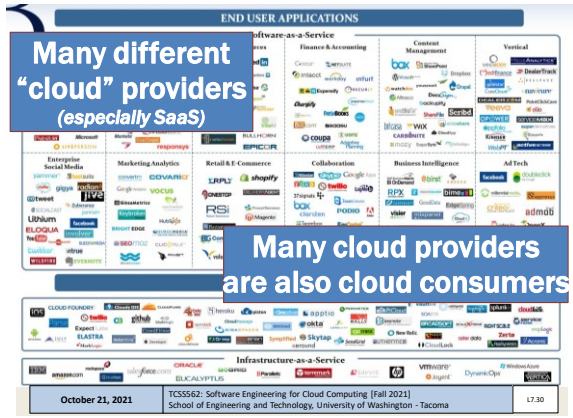


28

EXAMPLE CLOUD SERVICES



29

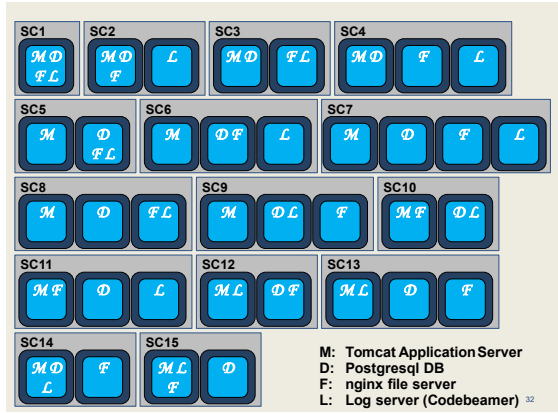


30

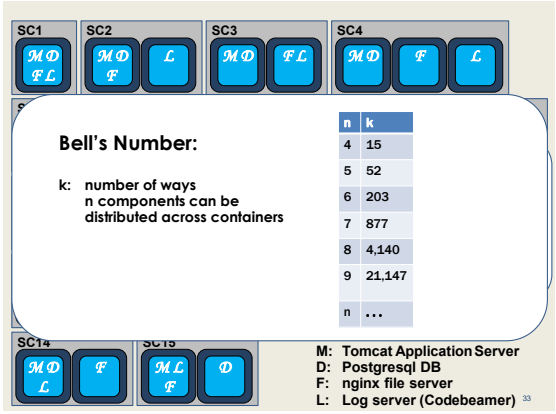
INFRASTRUCTURE-AS-A-SERVICE

- 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

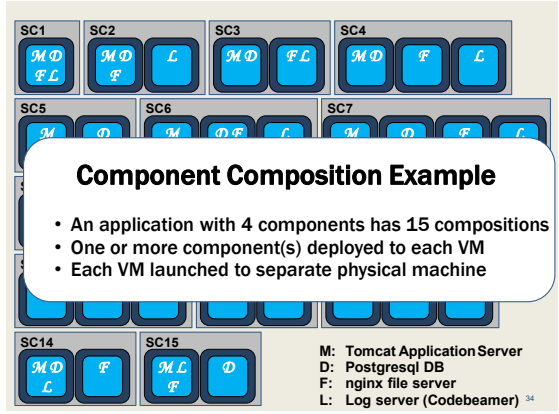
31



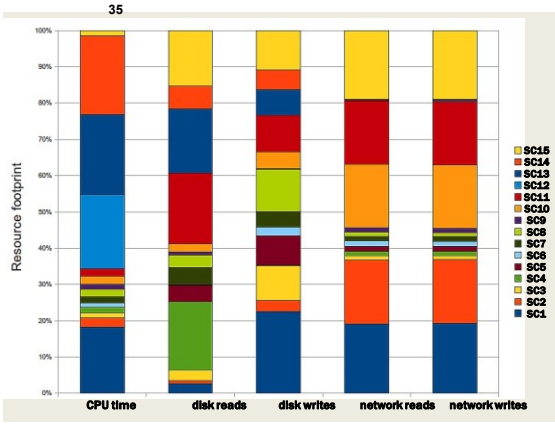
32



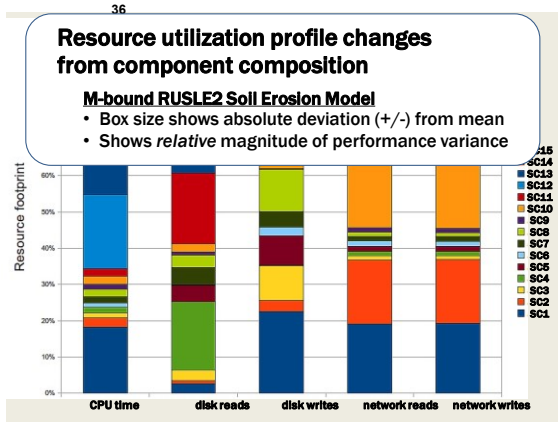
33



34

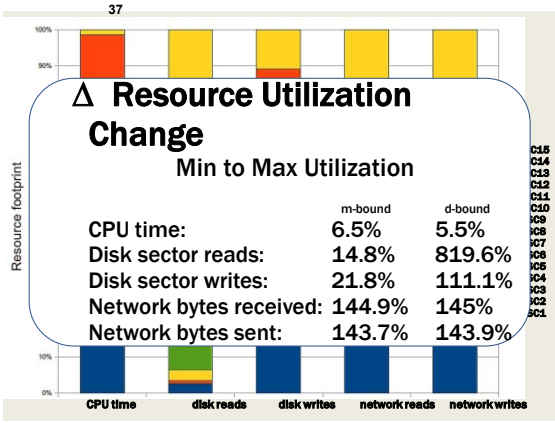


35



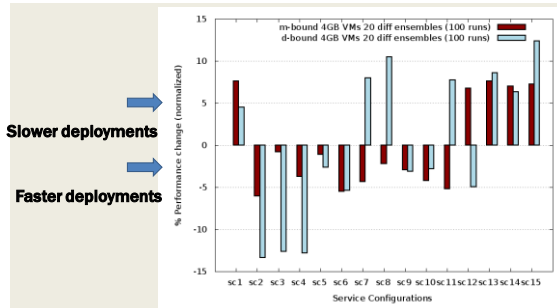
36

37



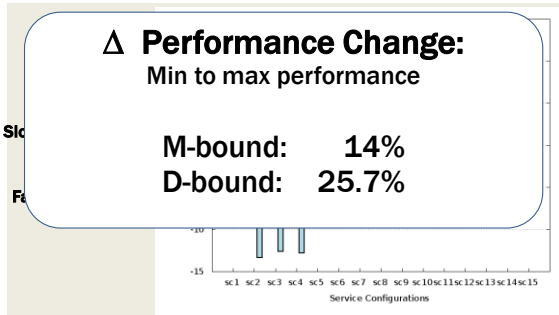
37

PERFORMANCE IMPLICATIONS OF APPLICATION DEPLOYMENTS



38

PERFORMANCE IMPLICATIONS OF APPLICATION DEPLOYMENTS



39

CLOUD COMPUTING DELIVERY MODELS

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

Serverless Computing:

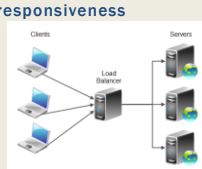
- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other Delivery Models

40

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)



41

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

42

CLOUD COMPUTING DELIVERY MODELS

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

Serverless Computing:

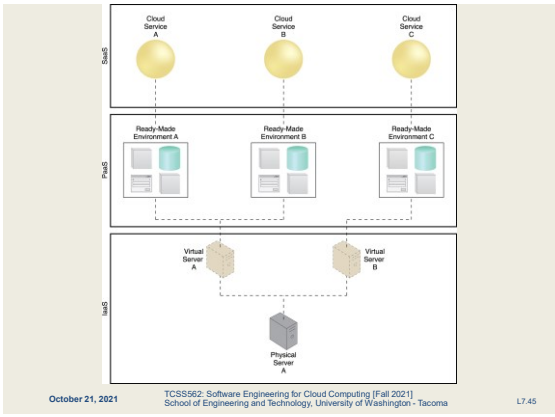
- Function-as-a-Service (FaaS)
- Container-as-a-Service (CaaS)
- Other Delivery Models

43

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

44



45

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

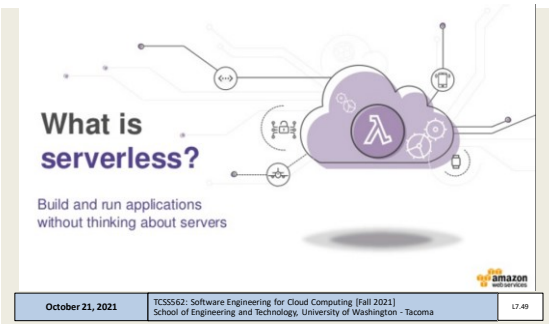
46

47

SERVERLESS COMPUTING

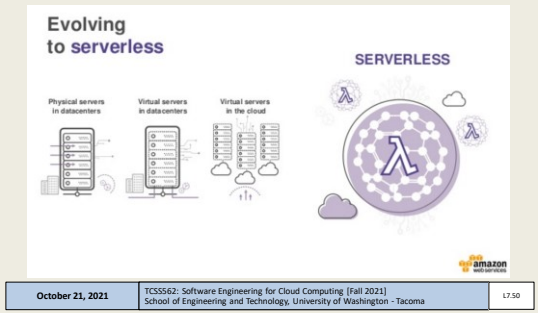
48

SERVERLESS COMPUTING



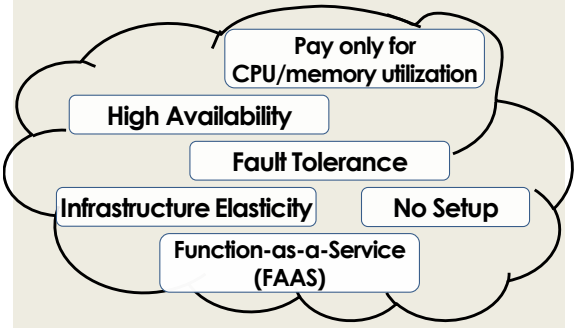
49

SERVERLESS COMPUTING - 2



50

SERVERLESS COMPUTING



51

SERVERLESS COMPUTING

Why Serverless Computing?

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

…they are built into the platform

52

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

53

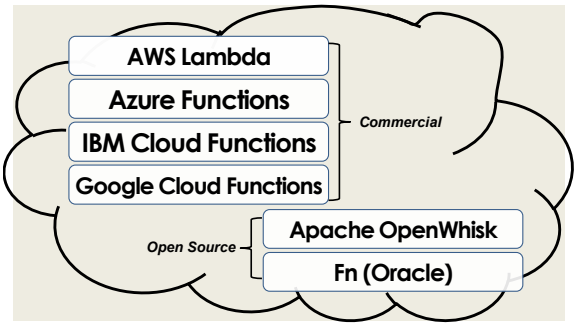
SERVERLESS VS. FAAS

- **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…

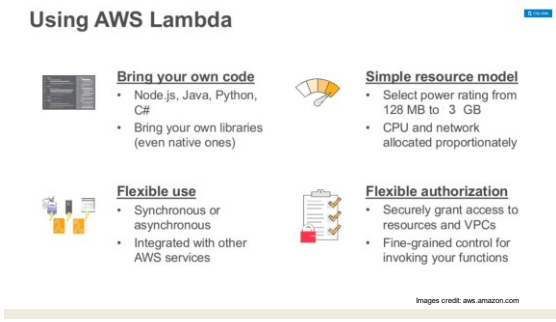
54

FAAS PLATFORMS



55

AWS LAMBDA



56

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

57

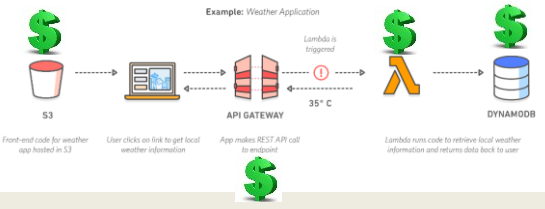
FAAS PLATFORMS - 3

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

58

CLOUD NATIVE SOFTWARE ARCHITECTURE

- Every service with a different pricing model



59

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 →

60

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

61

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?**

62

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

63

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?**

64

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

65

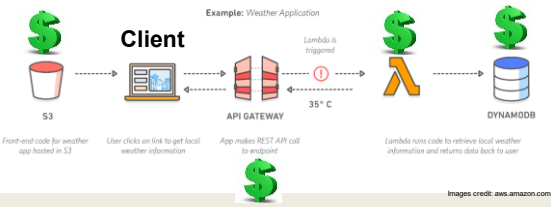
FAAS CHALLENGES

- 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?

66

VENDOR ARCHITECTURAL LOCK-IN

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



- Increased dependencies → increased hosting costs

67

PRICING OBFUSCATION

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

AWS Lambda Pricing

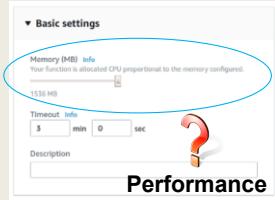
FREE TIER: first 1,000,000 function calls/month → FREE
first 400,000 GB-sec/month → FREE

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

68

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…*"

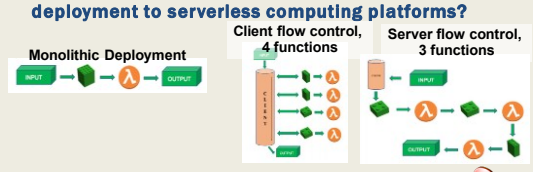


- **But how much memory do model services require?**

69

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
- **How does composition impact the number of function invocations, and memory utilization?**

Performance

70

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

71

FUNCTION-AS-A-SERVICE

AWS Lambda Demo

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

73

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

74

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

75

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

76

OBJECTIVES – 10/21

- **Questions from 10/19**
- Tutorial 3 - Best Practices with EC2
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4:** Cloud Computing Concepts and Models:
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

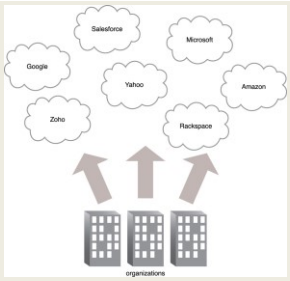
77

CLOUD DEPLOYMENT MODELS

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

78

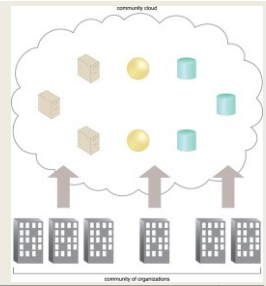
PUBLIC CLOUDS



79

COMMUNITY CLOUD

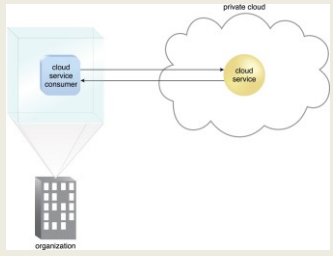
- Specialized cloud built and shared by a particular community
- Leverage economies of scale within a community
- Research oriented clouds
- Examples:
  - Bionimbus - bioinformatics
  - Chameleon
  - CloudLab



80

PRIVATE CLOUD

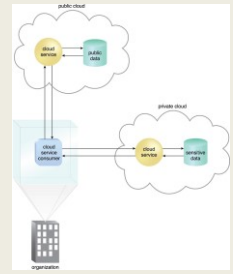
- Compute clusters configured as IaaS cloud
- Open source software
- Eucalyptus
- Openstack
- Apache Cloudstack
- Nimbus
- Virtualization: XEN, KVM, …



81

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



82

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

83

WE WILL RETURN AT 6:20 PM

84

OBJECTIVES – 10/21

- **Questions from 10/19**
- Tutorial 3 - Best Practices with EC2
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **From: Cloud Computing Concepts, Technology & Architecture: Chapter 4:** Cloud Computing Concepts and Models:
  - Cloud characteristics
  - Cloud delivery models
  - Cloud deployment models
- **2nd hour:**
  - TCSS 562 Term Project
  - Team Planning - Breakout Rooms

85

TCSS 562 TERM PROJECT

86

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
- Alternate 1: Cloud Computing Related Research Project
- Alternate 2: Literature Survey/Gap Analysis

**- as an individual project*

87

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**

88

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

89

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

90

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?

91

SERVERLESS APPLICATIONS - 2

- **Map-Reduce Style Application**
  - Function 1: split data into chunks, usually sequentially
  - Function 2: process individual chunks concurrently (in parallel)
    - Data processing 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 (?)

92

AWS LAMBDA PLATFORM LIMITATIONS

- Maximum 10 GB memory per function instance
- Maximum 15-minutes execution per function instance
- Access to 500 MB of temporary disk space for local I/O
- 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

93

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

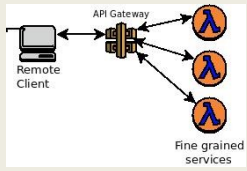
94

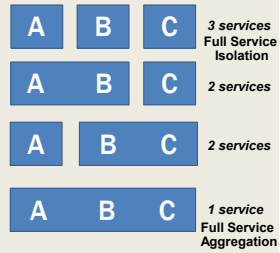
EXTRACT TRANSFORM LOAD DATA PIPELINE - 2

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

95

SERVICE COMPOSITION

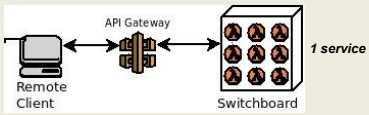




Other possible compositions: group by library, functional cohesion, etc.

96

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

97

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

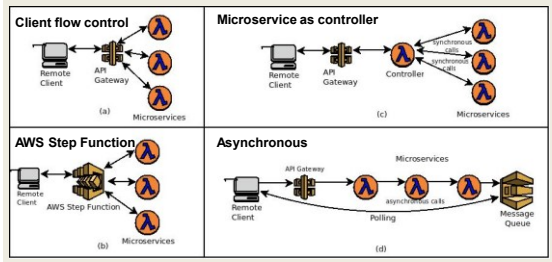
98

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)

99

APPLICATION FLOW CONTROL - 3



100

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

101

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

102

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

103

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…

104

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

105

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

106

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?

107

QUESTIONS

108