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

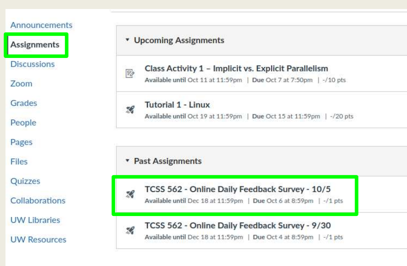
OBJECTIVES – 10/26

- Questions from 10/21
- Quiz 1 – posted on Canvas
- Class Activity #2 (will discuss on Wednesday)
- From: Cloud Computing Concepts, Technology & Architecture: Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS overview and demonstration
- 2nd hour:
- Tutorial #4 – Introduction
- AWS Lambda / SAAF Demonstration (Tutorial 4)
- Term project questions
- Team planning

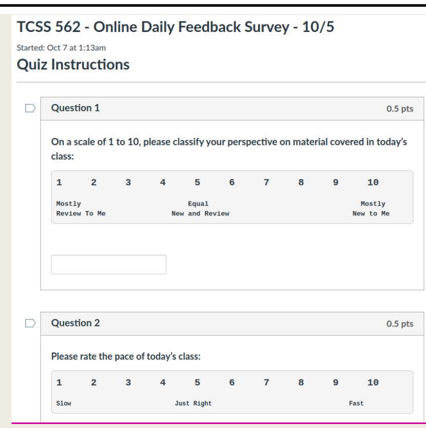
October 26, 2020 | TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.2

ONLINE DAILY FEEDBACK SURVEY

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



October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.3



October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.4

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (25 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.36 (↓ - *previous 6.50*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.52 (↓ - previous 5.58)**

October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.5

FEEDBACK FROM 10/21

- ***If computation requires more than 3 GB of memory then how it will be managed (on a Function-as-a-Service platform)?***
- Presently these workloads are not supported on Function-as-a-Service platforms
- On FAAS platforms, the idea is to decompose applications into smaller components that require less memory
- Alternatively, can host computation with another cloud platform:
  - AWS Fargate (Container-as-a-Service):
  - Host Docker containers w/ up to 30GB RAM & 4 vCPUs

October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.6

FEEDBACK - 2

- ***Can you elaborate on what containers are and their purpose?***
- (Application) Containers such as Docker provide a light-weight alternative to full virtual machines
- Covered in future 562 lecture
- They provide a possible vehicle for cloud computing service delivery to address the question →

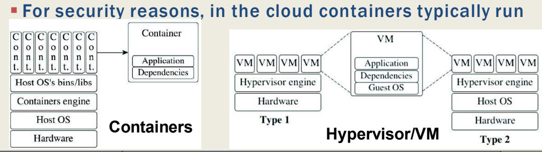
What is the right level of abstraction?

- What hardware aspects:
  - Should be hidden (no access) from the cloud consumer?
  - Should be exposed (read-only) to the cloud consumer?
  - Should be controllable by the cloud consumer?

October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.7

MOTIVATION FOR CONTAINERIZATION

- Containers provide "light-weight" alternative to full OS virtualization provided by a hypervisor
- Containers do not provide a full "machine"
- Containers use OS features to offer execution "sand boxes"
  - Linux cgroups, namespaces, etc.
- Containers can run on bare metal, or atop of VMs
  - For security reasons, in the cloud containers typically run



November 20, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L15.8

FEEDBACK - 3

- Do containers in Container-as-a-Service platforms provide more or less abstraction that functions in Function-as-a-Service platforms?
- Poll-ev

October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.9

Are containers more or less abstract than functions in cloud-based serverless computing platforms? Abstraction implies that cloud providers restrict HW/OS configurability and observability from end users.

Start the presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app

OBJECTIVES – 10/26

- Questions from 10/21
- Quiz 1 –posted on Canvas
- Class Activity #2 (will discuss on Wednesday)
- From: Cloud Computing Concepts, Technology & Architecture: Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS overview and demonstration
- 2nd hour:
- Tutorial #4 – Introduction
- AWS Lambda / SAAF Demonstration (Tutorial 4)
- Term project questions
- Team planning

October 26, 2020 | TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.11

OBJECTIVES – 10/26

- Questions from 10/21
- Quiz 1 –posted on Canvas
- Class Activity #2 (will discuss on Wednesday)
- From: Cloud Computing Concepts, Technology & Architecture: Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS overview and demonstration
- 2nd hour:
- Tutorial #4 – Introduction
- AWS Lambda / SAAF Demonstration (Tutorial 4)
- Term project questions
- Team planning

October 26, 2020 | TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.12

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

CLOUD COMPUTING: CONCEPTS AND MODELS


OBJECTIVES – 10/26

- **Questions from 10/21**
- **Quiz 1 –posted on Canvas**
- **Class Activity #2 (will discuss on Wednesday)**
- **From: Cloud Computing Concepts, Technology & Architecture:** Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS overview and demonstration

- 2nd hour:
- Tutorial #4 – Introduction
- AWS Lambda / SAAF Demonstration (Tutorial 4)
- Term project questions
- Team planning

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

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

Assume a month equals 30.416667 days
365 days / 12 months

- **ON AWS Lambda**
- Each service call: 100% of 2 CPU-cores
  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: c5.large: $62.05/month
  8.5¢/hour, 24 hrs/day x 30.41667 days

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

PRICING OBFUSCATION

- Workload: (3GB) 7,884,000 GB-sec
- FF... ec
- Ch... ec
- M... 43
- 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?**

FACTORS IMPACTING PERFORMANCE OF FAAS COMPUTING PLATFORMS

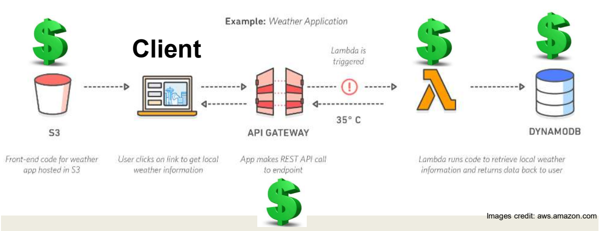
- Infrastructure elasticity
- Load balancing
- Provisioning variation
- Infrastructure retention: COLD vs. WARM
  - Infrastructure freeze/thaw cycle
- Memory reservation
- Service composition

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?

VENDOR ARCHITECTURAL LOCK-IN

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



Images credit: aws.amazon.com

- Increased dependencies → increased hosting costs

PRICING OBFUSCATION

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

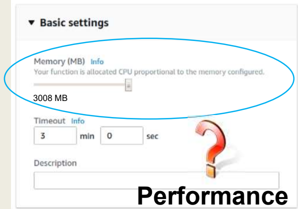
AWS Lambda Pricing

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

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

MEMORY RESERVATION QUESTION…

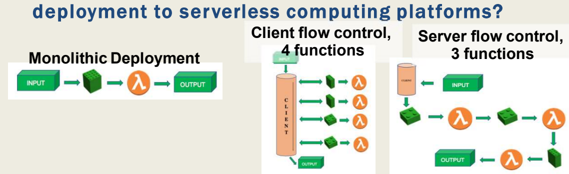
- Lambda memory reserved for functions
- UI provides "slider bar" to set function's memory allocation
- Resource capacity (CPU, disk, network) coupled to slider bar: "*every **doubling** of memory, **doubles** CPU…*"
- **But how much memory do model services require?**



Performance

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

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

28

THE SERVERLESS APPLICATION ANALYTICS FRAMEWORK

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

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 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.31

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 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.32

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 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L10.33

OBJECTIVES – 10/26

- **Questions from 10/21**
- **Quiz 1 –posted on Canvas**
- **Class Activity #2 (will discuss on Wednesday)**
- **From: Cloud Computing Concepts, Technology & Architecture:** Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS overview and demonstration
- **2nd hour:**
- Tutorial #4 – Introduction
- AWS Lambda / SAAF Demonstration (Tutorial 4)
- Term project questions
- Team planning

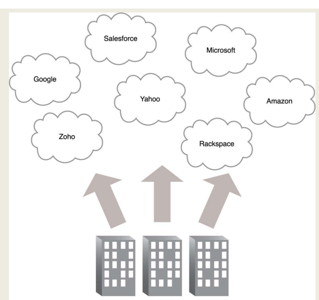
October 26, 2020 | TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.34

CLOUD DEPLOYMENT MODELS

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

October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.35

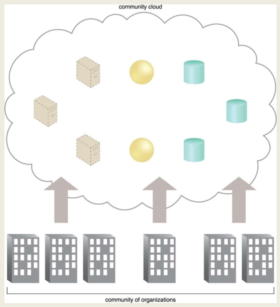
PUBLIC CLOUDS



October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.36

COMMUNITY CLOUD

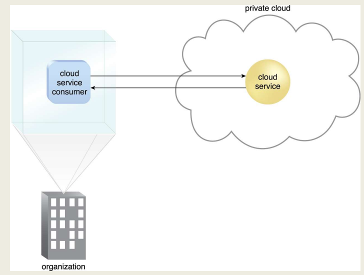
- Specialized cloud built and shared by a particular community
- Leverage economies of scale within a community
- Research oriented clouds
- Examples:
  - Bionimbus - bioinformatics
  - Chameleon
  - CloudLab



October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.37

PRIVATE CLOUD

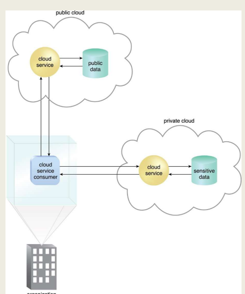
- Compute clusters configured as IaaS cloud
- Open source software
- Eucalyptus
- Openstack
- Apache Cloudstack
- Nimbus
- Virtualization: XEN, KVM, …



October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.38

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 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.39

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 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.40

OBJECTIVES – 10/26

- **Questions from 10/21**
- **Quiz 1 –posted on Canvas**
- **Class Activity #2 (will discuss on Wednesday)**
- **From: Cloud Computing Concepts, Technology & Architecture:** Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS overview and demonstration
- **2nd hour:**
- Tutorial #4 – Introduction
- AWS Lambda / SAAF Demonstration (Tutorial 4)
- Term project questions
- Team planning

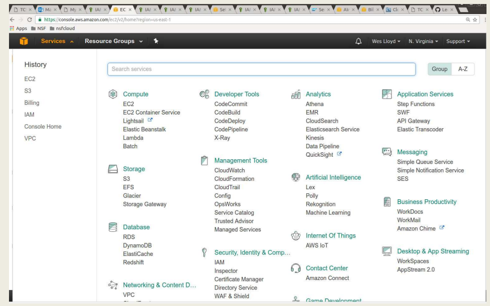
October 26, 2020 | TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.41

AWS DEMO

CLOUD 101 WORKSHOP

- From the eScience Institute @ UW Seattle:
- https://escience.washington.edu/
- Offers 1-day cloud workshops
- Introduction to AWS, Azure, and Google Cloud
- Task: Deploying a Python DJANGO web application
- Self-guided workshop materials available online:
- https://cloudmaven.github.io/documentation/rc_cloud101_immersion.html
- AWS Educate provides access to many online tutorials / learning resources

AWS MANAGEMENT CONSOLE



AWS EC2

- Elastic Compute Cloud
- Instance types: https://ec2instances.info
  - On demand instance – full price
  - Reserved instance – contract based
  - Spot instance – auction based, terminates with 2 minute warning
  - Dedicated/reserved host – reserved HW
  - Reserved host
  - Instance families:
    General, compute-optimized, memory-optimized, GPU, etc.
- Storage types
  - Instance storage - ephemeral storage
  - EBS - Elastic block store
  - EFS - Elastic file system

INSTANCE STORAGE

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

ELASTIC BLOCK STORE

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

EBS VOLUME TYPES - 2

- Metric: I/O Operations per Second (IOPS)
- General Purpose 2 (GP2)
  - 3 IOPS per GB, Max 10,000 IOPS, 160MB/sec per volume
- Provisioned IOPS (IO1)
  - 32,000 IOPS, and 500 MB/sec throughput per volume
- Throughput Optimized HDD (ST1)
  - Up to 500 MB/sec throughput
  - 4.5 ¢ per GB/month
- Cold HDD (SC1)
  - Up to 250 MB/sec throughput
  - 2.5 ¢ per GB/month
- Magnetic
  - Up to 800 MB/sec throughput
  - 5 ¢ per GB/month

ELASTIC FILE SYSTEM (EFS)

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

ELASTIC FILE SYSTEM (EFS) - 2

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

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

ELASTIC FILE SYSTEM (EFS) - 3

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

ELASTIC FILE SYSTEM (EFS) - 4

Performance Comparison, Amazon EFS and Amazon EBS

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

Storage Characteristics Comparison, Amazon EFS and Amazon EBS

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

AMAZON MACHINE IMAGES

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

EC2 VIRTUALIZATION - PARAVIRTUAL

- **1st, 2nd, 3rd, 4th generation → XEN-based**
- **5th generation instances → AWS Nitro virtualization**
- XEN - two virtualization modes
- XEN Paravirtualization "paravirtual"
  - 10GB Amazon Machine Image – base image size limit
  - Addressed poor performance of old XEN HVM mode
  - I/O performed using special XEN kernel with XEN paravirtual mode optimizations for better performance
  - Requires OS to have an available paravirtual kernel
  - PV VMs: will use common **AKI** files on AWS – ***Amazon kernel image(s)***
    - *Look for common identifiers*

EC2 VIRTUALIZATION - HVM

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

EC2 VIRTUALIZATION - NITRO

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

EVOLUTION OF AWS VIRTUALIZATION

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

AWS EC2 Virtualization Types

Bare-metal performance (grey); Near-metal performance (green); Optimized performance (blue); Poor performance (red)

Importance: Most → Least

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

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

INSTANCE ACTIONS

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

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

EC2 INSTANCE: NETWORK ACCESS

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

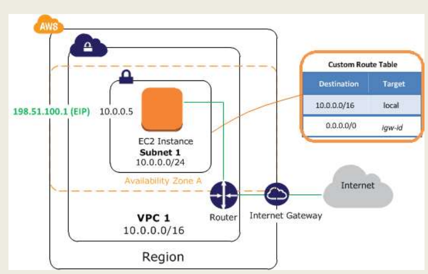
- Security groups
  - E.g. firewall

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

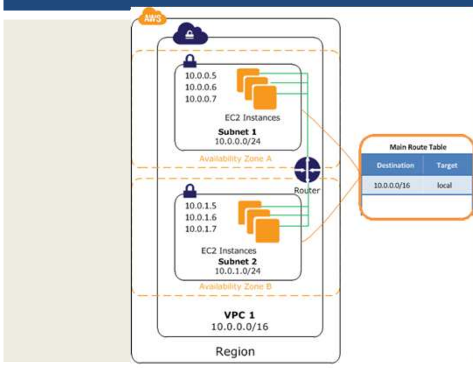
- VPC / Subnet / Internet Gateway / Router
- NAT-Gateway

SIMPLE VPC

- Recommended when using Amazon EC2



VPC SPANNING AVAILABILITY ZONES



SIMPLE STORAGE SERVICE (S3)

- Key-value blob storage
- What is the difference vs. key-value stores (NoSQL DB)?
- Can mount an S3 bucket as a volume in Linux
  - Supports common file-system operations
- Provides eventual consistency
- Can store Lambda function state for life of container.

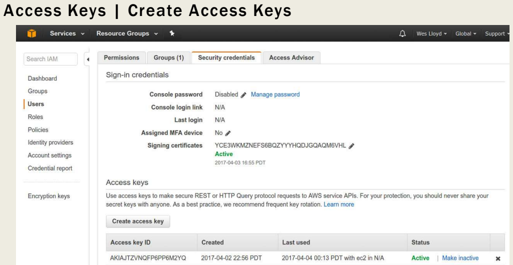
AWS CLI

- Launch Ubuntu 16.04 VM
  - Instances | Launch Instance
- Install the general AWS CLI
  - sudo apt install awscli
- Create config file

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


AWS CLI - 2

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



AWS CLI - 3

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

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

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

ALTERNATIVE CLI

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

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

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

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

INSPECTING INSTANCE INFORMATION

- Find your instance ID:

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

- ec2-get-info command (??)

PRIVATE KEY AND CERTIFICATE FILE

- Install openssl package on VM

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

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

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

PRIVATE KEY, CERTIFICATE FILE

- These files, combined with your AWS_ACCESS_KEY and AWS_SECRET_KEY and AWS_ACCOUNT_ID enable you to publish new images from the CLI
- Objective:
  1. Configure VM with software stack
  2. Burn new image for VM replication (**horizontal scaling**)
- Some folks may just install Docker. . .
- Create image script . . .

CREATE A NEW INSTANCE STORE IMAGE SCRIPT

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


COST SAVINGS MEASURES

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

WE WILL RETURN AT ~7:10PM

OBJECTIVES – 10/26

- **Questions from 10/21**
- **Quiz 1 – posted on Canvas**
- **Class Activity #2 (will discuss on Wednesday)**
- **From: Cloud Computing Concepts, Technology & Architecture:** Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS overview and demonstration
- **2nd hour:**
- Tutorial #4 – Introduction
- AWS Lambda / SAAF Demonstration (Tutorial 4)
- Term project questions
- Team planning

OBJECTIVES – 10/26

- **Questions from 10/21**
- **Quiz 1 – posted on Canvas**
- **Class Activity #2 (will discuss on Wednesday)**
- **From: Cloud Computing Concepts, Technology & Architecture:** Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS overview and demonstration
- **2nd hour:**
- Tutorial #4 – Introduction
- AWS Lambda / SAAF Demonstration (Tutorial 4)
- Term project questions
- Team planning

OBJECTIVES – 10/26

- **Questions from 10/21**
- **Quiz 1 – posted on Canvas**
- **Class Activity #2 (will discuss on Wednesday)**
- **From: Cloud Computing Concepts, Technology & Architecture:** Cloud Computing Concepts and Models:
  - Cloud delivery models
  - Cloud deployment models
- AWS overview and demonstration
- **2nd hour:**
- Tutorial #4 – Introduction
- AWS Lambda / SAAF Demonstration (Tutorial 4)
- Term project questions
- Team planning

TCSS 562 TERM PROJECT

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

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**

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

OTHER PROJECT IDEAS

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

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?

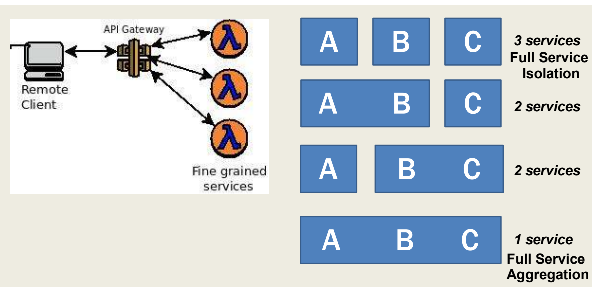
EXTRACT TRANSFORM LOAD DATA PIPELINE

- 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

EXTRACT TRANSFORM LOAD DATA PIPELINE - 2

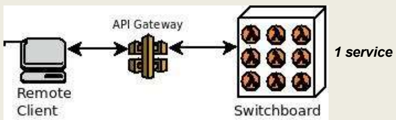
- Service 3: **QUERY**
- Using relational database, apply filter(s) and/or functions to aggregate data to produce sums, totals, averages
- Output aggregations as JSON

SERVICE COMPOSITION



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

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

APPLICATION FLOW CONTROL

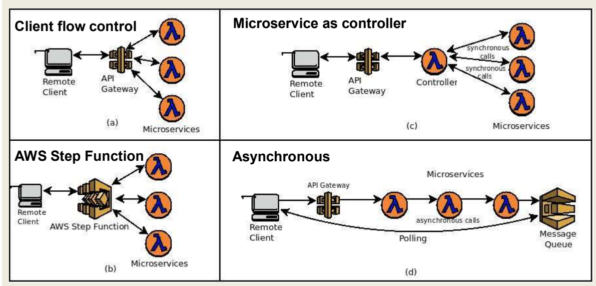
- Serverless Computing:
- AWS Lambda (FAAS: Function-as-a-Service)
- Provides HTTP/REST like web services
- Client/Server paradigm

- 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

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)

APPLICATION FLOW CONTROL - 3



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

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

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

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…

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

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

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?

QUESTIONS

QUESTIONS

TCSS 562 OFFICE HOURS

PLEASE SAY HELLO

OBJECTIVES – 10/19

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

AMDAHL'S LAW

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

  α: fra... parall... (e.g. must run sequentially)
- Max... (N):

In the last class, we were having difficulty moving between Amdahl's Law and Gustafson's because as it turns out, this formula was OVERSIMPLIFIED from the text

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

AMDAHL'S LAW

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

  α: fraction of program run time which can't be parallelized (e.g. must run sequentially)
- Maximum speedup with a large number of processors (N):

  S = 1/ α

Where α = σ / (π + σ)
Where σ= sequential time, π =parallel time
Where T(1) = σ + π
And T(N) = σ + π / N, where N = parallel computations performed

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

GUSTAFSON'S LAW

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

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

N: Number o

α: fraction o ... arallelized (e.g. must

- *Can be use ... tion of program*

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

October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.103

GUSTAFSON'S LAW

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

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

N: Number of processors

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

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

October 26, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L8.104