TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

AWS Demo

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

1

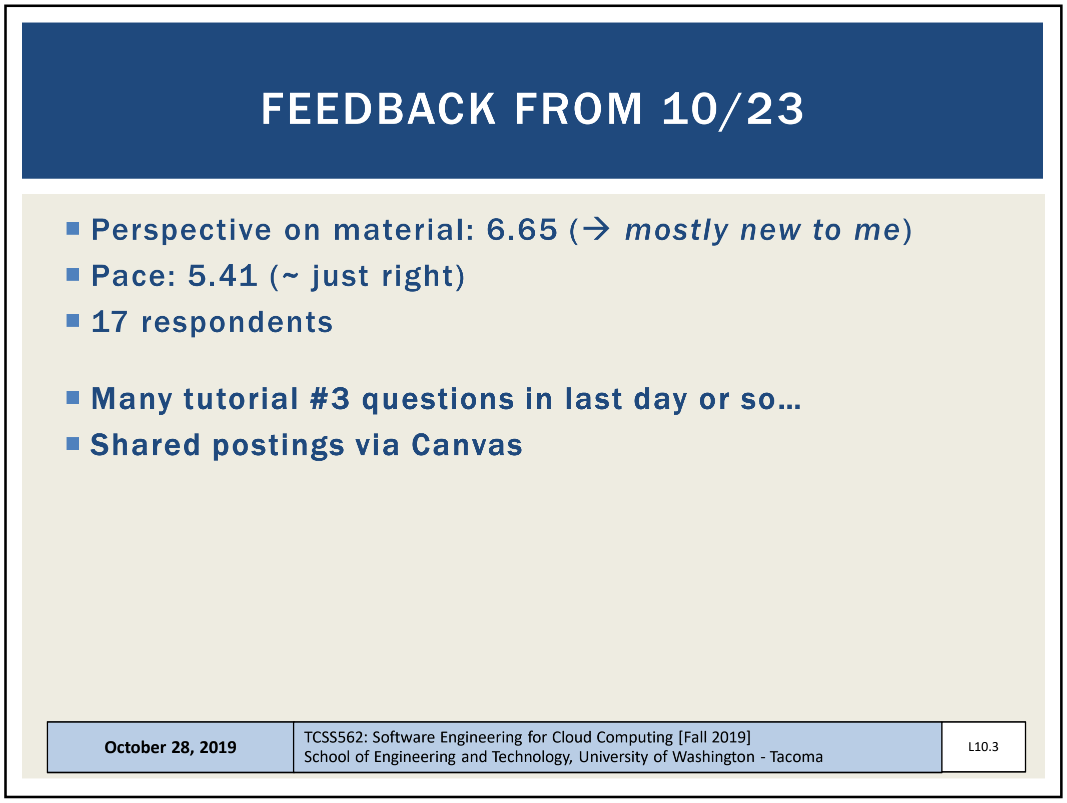
MIDTERM EXAM SCHEDULING

- **Practice midterm:**
- In class, Wednesday November 6th (for 1 hour)
- 1-week to study

- **Holiday:**
- Veteran's Day - Monday November 11th

- **Midterm:**
- Wednesday November 13th

2



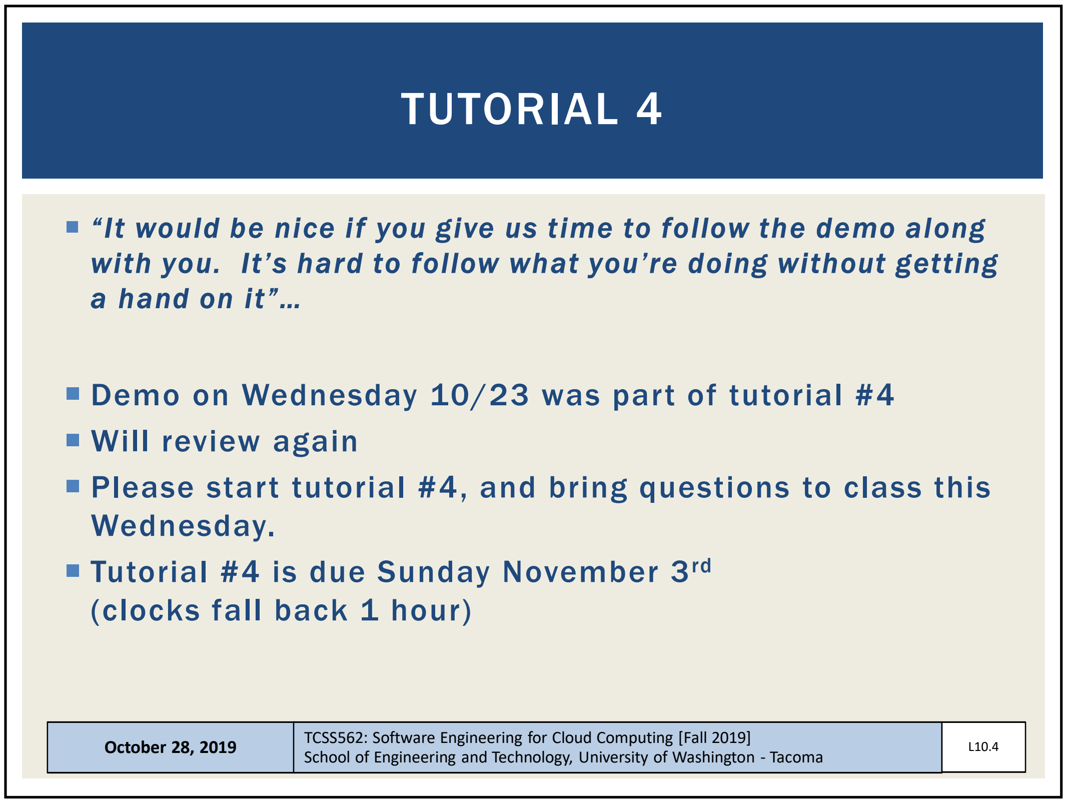
FEEDBACK FROM 10/23

- Perspective on material: 6.65 (→ *mostly new to me*)
- Pace: 5.41 (~ just right)
- 17 respondents

- Many tutorial #3 questions in last day or so…
- Shared postings via Canvas

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L10.3

3



TUTORIAL 4

- *"It would be nice if you give us time to follow the demo along with you. It's hard to follow what you're doing without getting a hand on it"…*

- Demo on Wednesday 10/23 was part of tutorial #4
- Will review again
- Please start tutorial #4, and bring questions to class this Wednesday.
- Tutorial #4 is due Sunday November 3rd
(clocks fall back 1 hour)

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L10.4

4

AWS DEMO

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fa 2019] School of Engineering and Technology, University of Wash gton - Tacoma | L10.5

5

CLOUD 101 WORKSHOP

- From the eScience Institute @ UW Seattle:
- https://escience.washington.edu/
- Offers 1-day cloud workshops

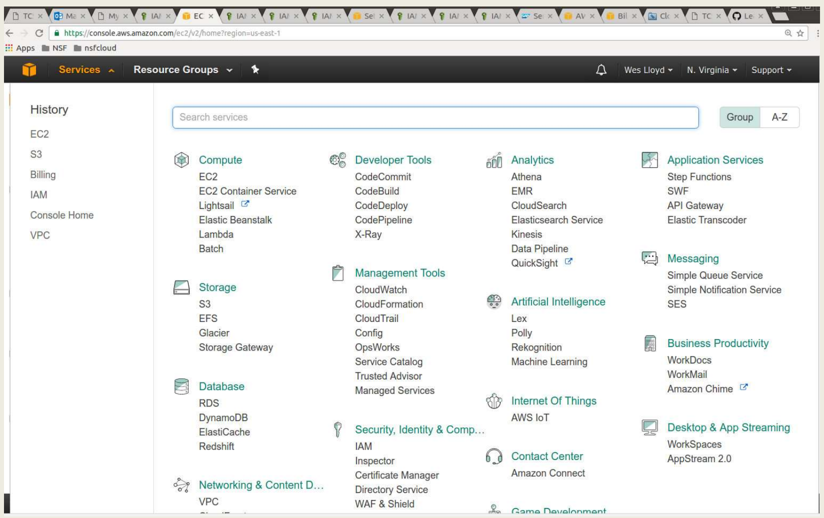
- Introduction to AWS, Azure, and Google Cloud
- Task: Deploying a Python DJANGO web application
- Workshop materials available online:

- https://cloudmaven.github.io/documentation/rc_cloud101_immersion.html

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.6

6

AWS MANAGEMENT CONSOLE



October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.7

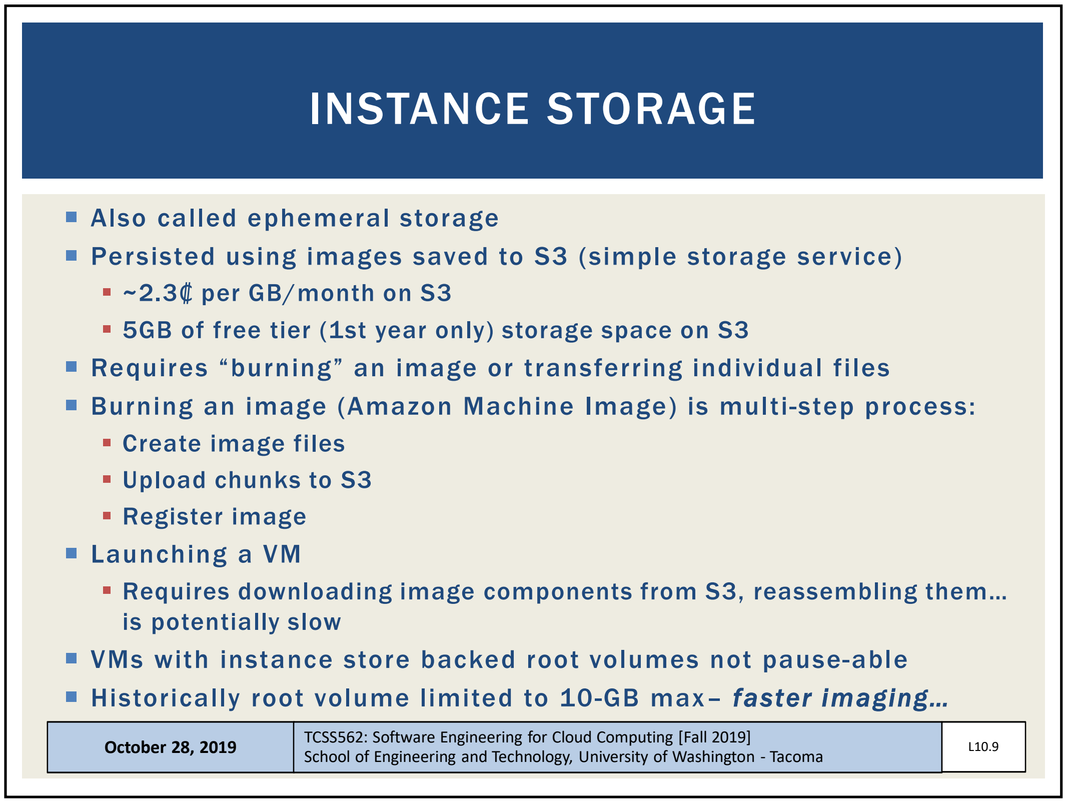
7

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

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.8

8

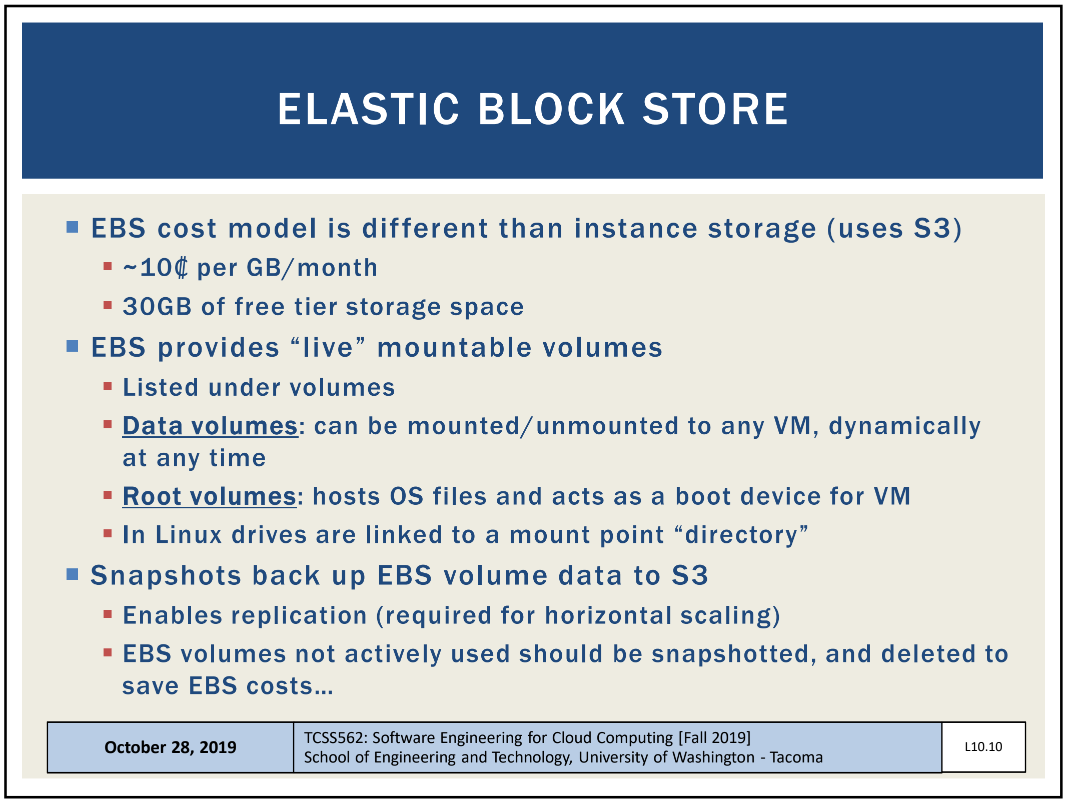


INSTANCE STORAGE

- Also called ephemeral storage
- Persisted using images saved to S3 (simple storage service)
  - ~2.3₵ per GB/month on S3
  - 5GB of free tier (1st year only) storage space on S3
- Requires "burning" an image or transferring individual files
- Burning an image (Amazon Machine Image) is multi-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…***

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.9

9

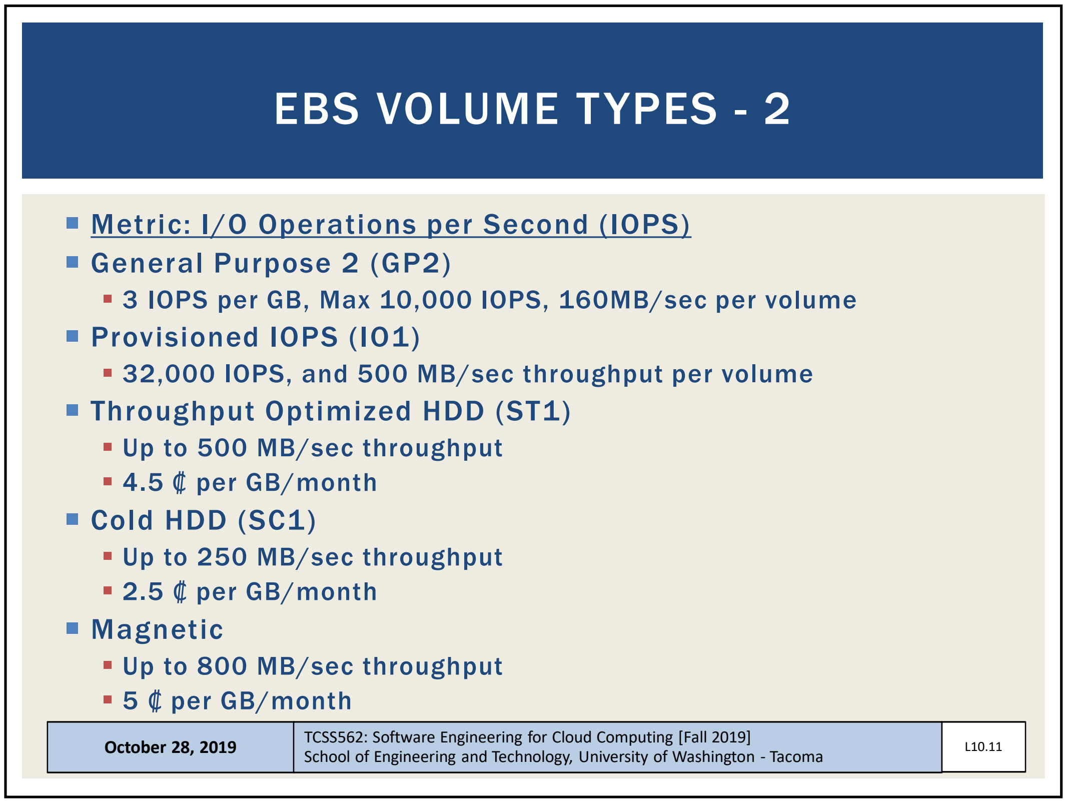


ELASTIC BLOCK STORE

- 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…

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.10

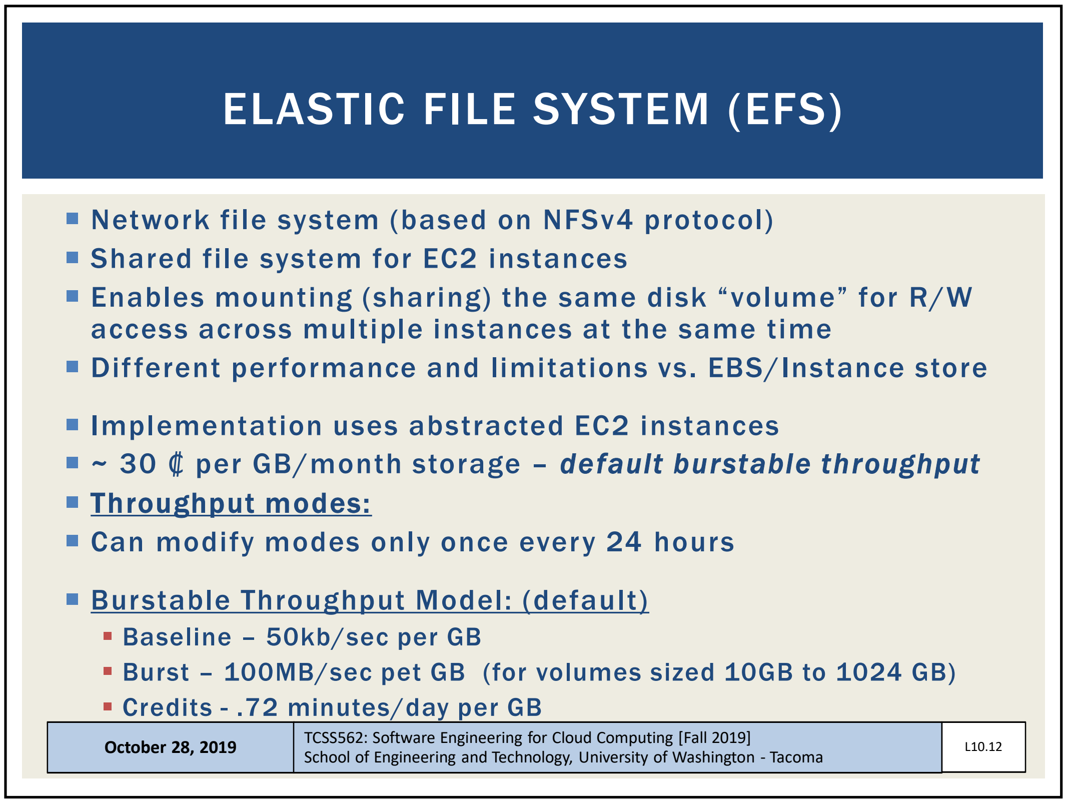
10



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

11



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: (default)
  - Baseline – 50kb/sec per GB
  - Burst – 100MB/sec pet GB (for volumes sized 10GB to 1024 GB)
  - Credits - .72 minutes/day per GB

12

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%

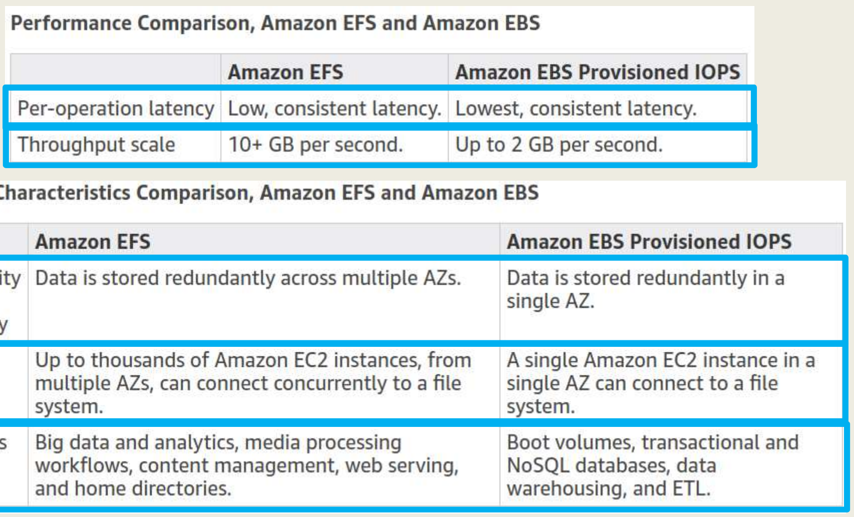
13

ELASTIC FILE SYSTEM (EFS) - 3

- Throughput Models
- Provisioned Throughput Model – (*pay for bandwidth*)
- For applications with:
  high performance requirements, but low storage requirements
- Get high levels of performance w/o overprovisioning capacity
- $6 for each 1 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

14

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

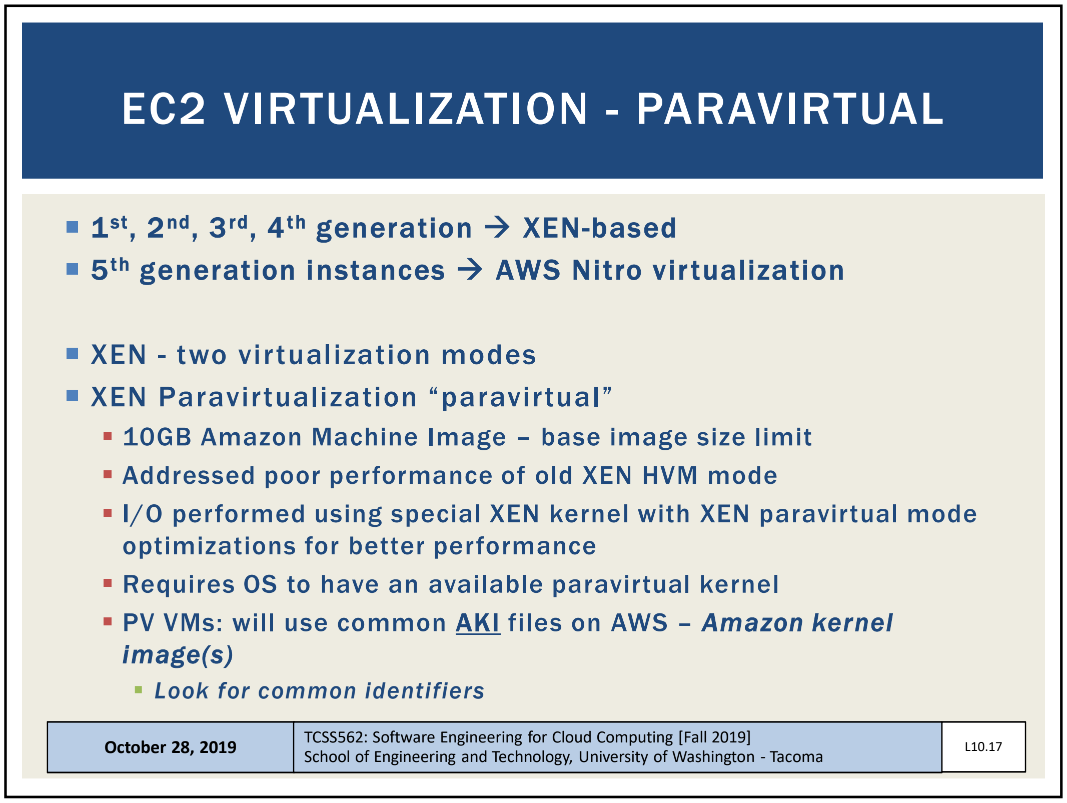
	Amazon EFS	Amazon EBS Provisioned IOPS
Availability and durability	Data is stored redundantly across multiple AZs.	Data is stored redundantly in a single AZ.
Access	Up to thousands of Amazon EC2 instances, from multiple AZs, can connect concurrently to a file system.	A single Amazon EC2 instance in a 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.

15

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 to delete the snapshot leads to costly "orphaned" data
  - No way to instantiate a VM from deregistered AMIs
  - Data still in S3 (snapshot) resulting in charges

16



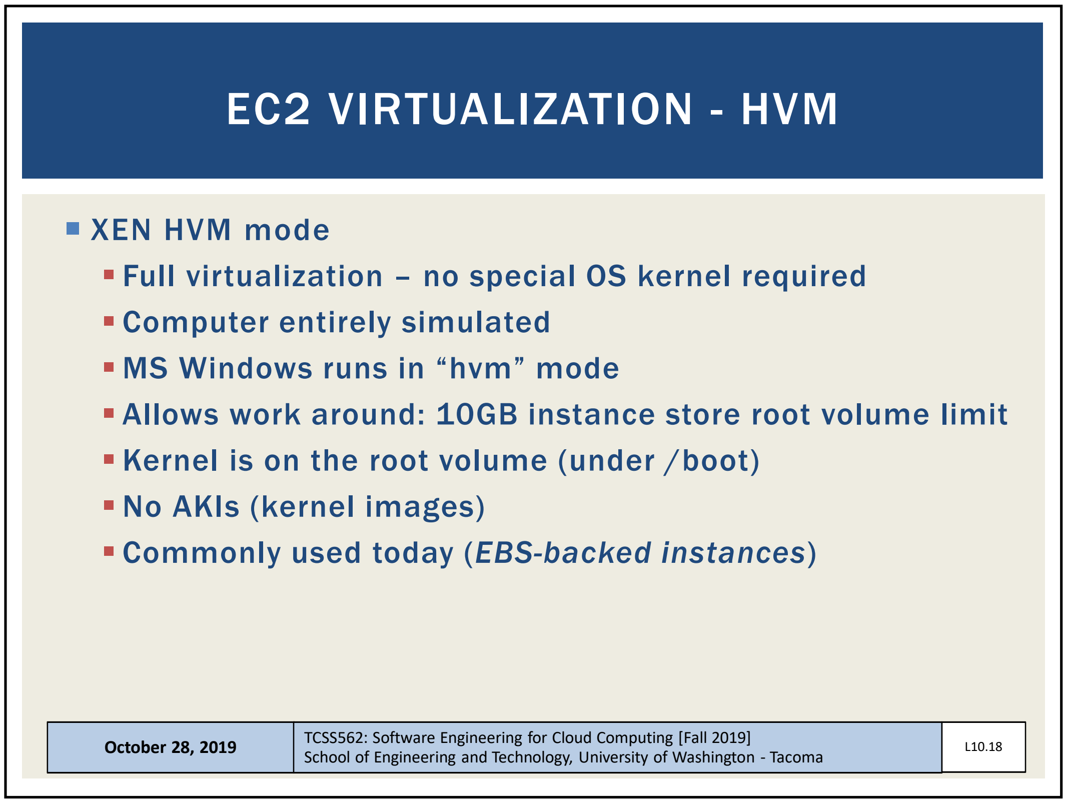
EC2 VIRTUALIZATION - PARAVIRTUAL

- 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*

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.17

17



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*)

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.18

18

EC2 VIRTUALIZATION - NITRO

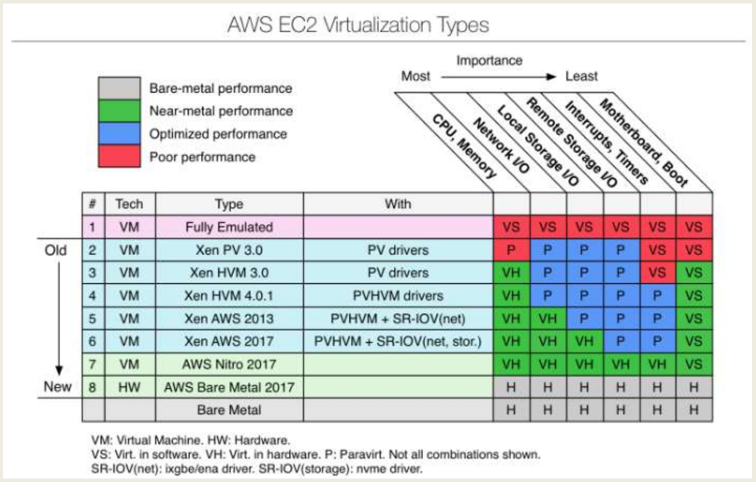
- Nitro based on Kernel-based-virtual-machines
  - Stripped down version of Linux KVM hypervisor
  - Uses KVM core kernel module
  - I/O access has a direct path to the device
- Goal: provide indistinguishable performance from bare metal

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.19

19

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
🟩	Near-metal performance
🟦	Optimized performance
🟥	Poor performance

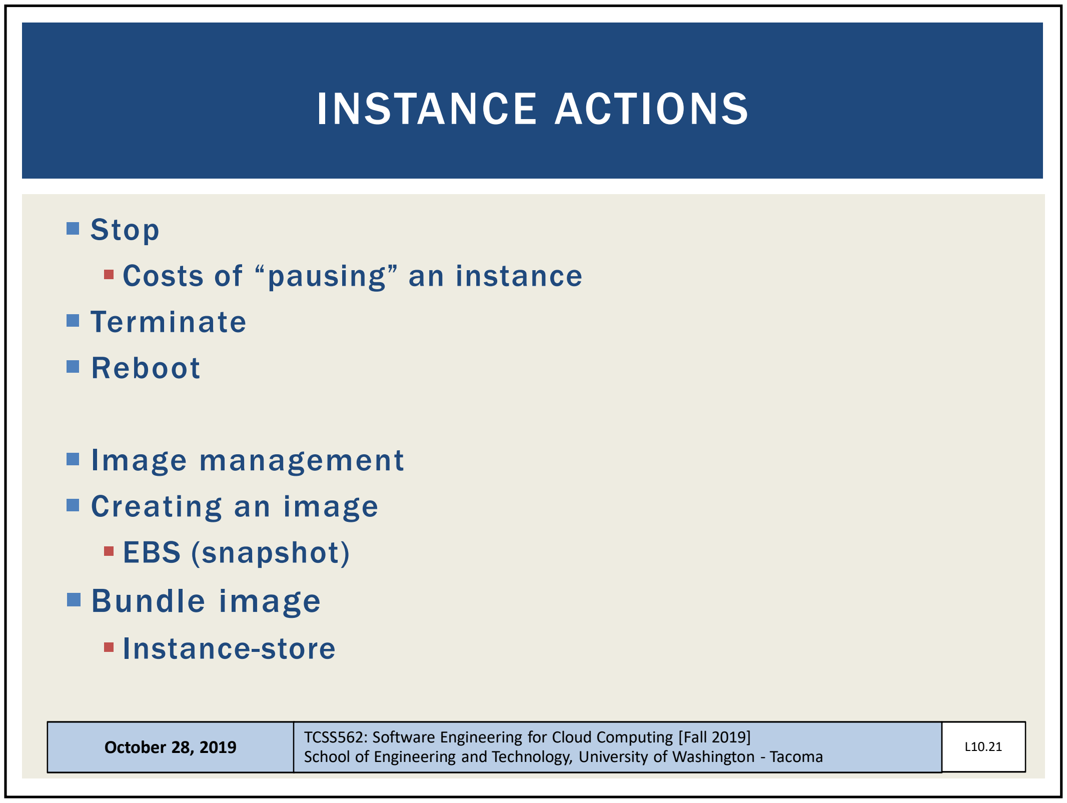
Importance: Most → Least

	#	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.

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.20

20



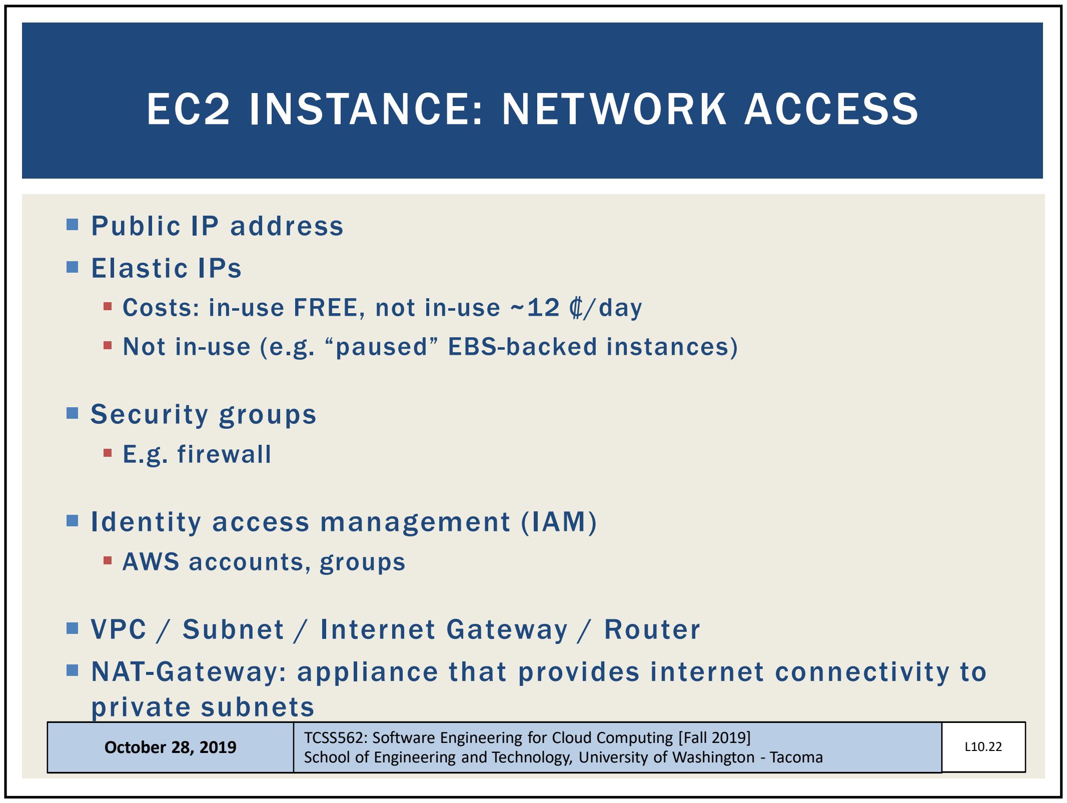
INSTANCE ACTIONS

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

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

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.21

21



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

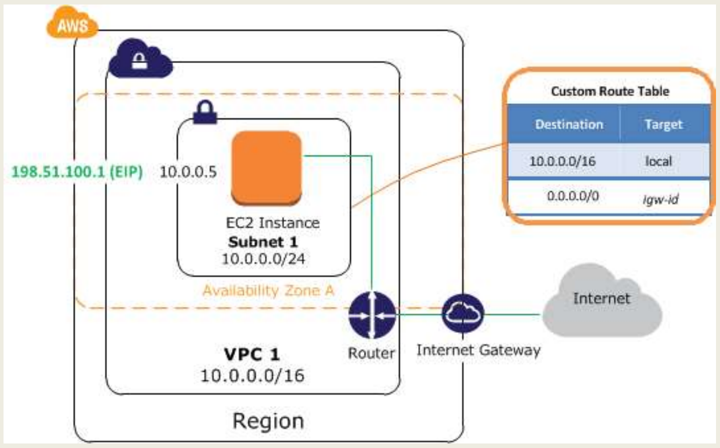
- VPC / Subnet / Internet Gateway / Router
- NAT-Gateway: appliance that provides internet connectivity to private subnets

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.22

22

SIMPLE VPC

- Recommended when using Amazon EC2

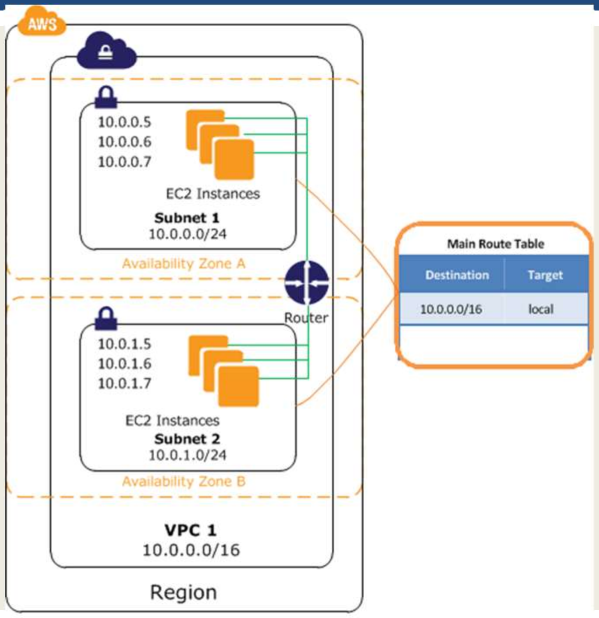


Custom Route Table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	*igw-id*

23

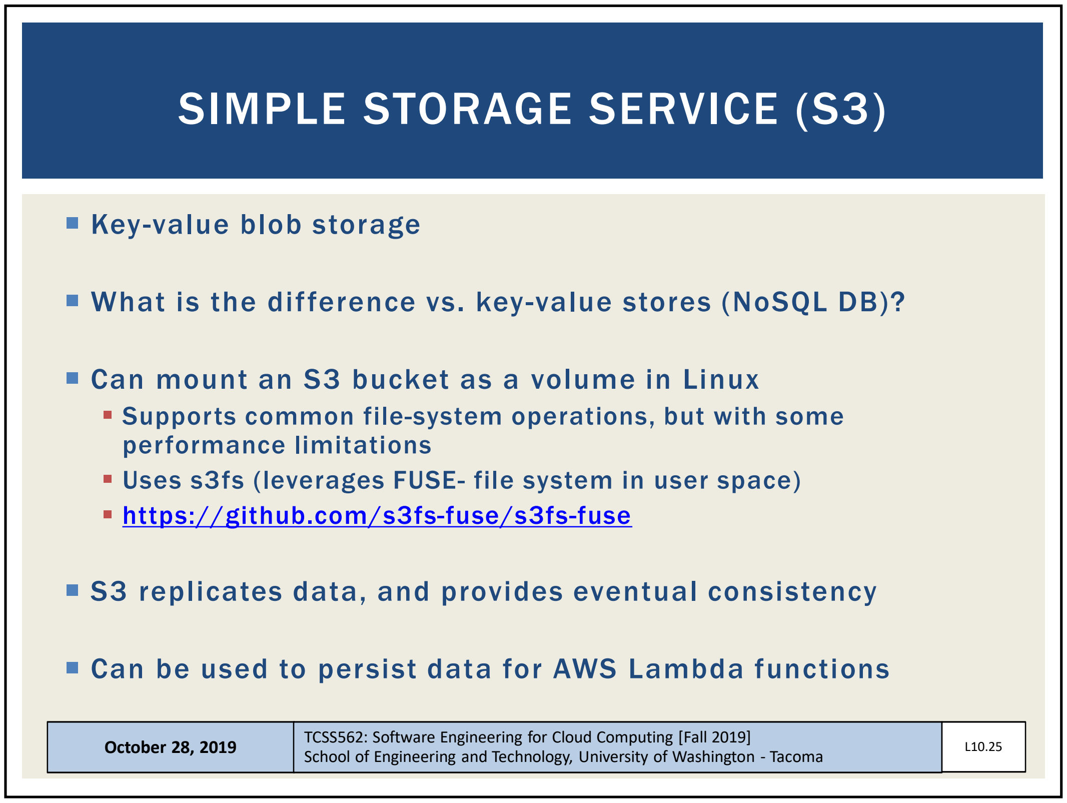
VPC SPANNING AVAILABILITY ZONES



Main Route Table

Destination	Target
10.0.0.0/16	local

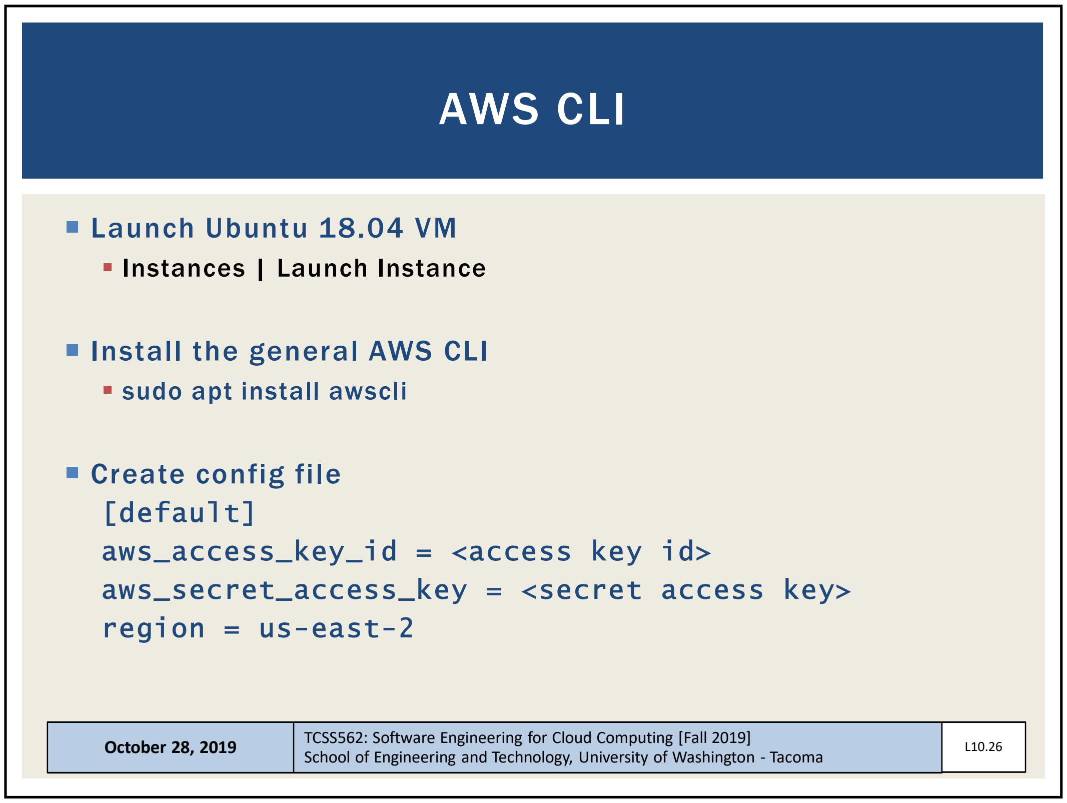
24



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, but with some performance limitations
  - Uses s3fs (leverages FUSE- file system in user space)
  - https://github.com/s3fs-fuse/s3fs-fuse
- S3 replicates data, and provides eventual consistency
- Can be used to persist data for AWS Lambda functions

25



AWS CLI

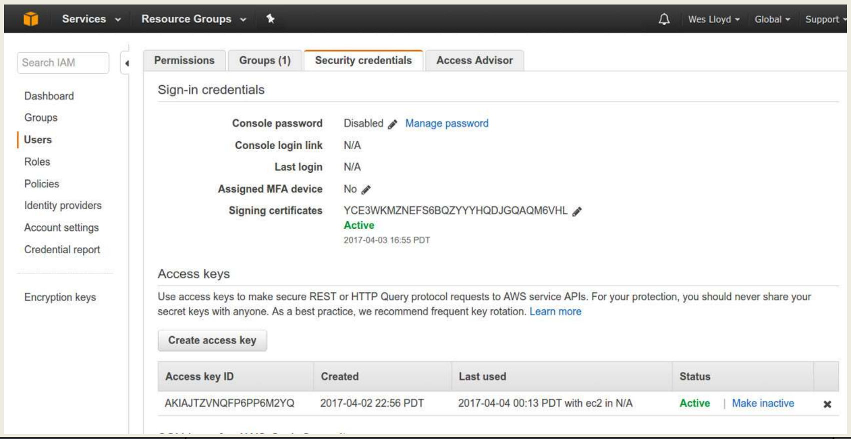
- Launch Ubuntu 18.04 VM
  - **Instances | Launch Instance**
- Install the general AWS CLI
  - sudo apt install awscli
- Create config file

```
[default]
aws_access_key_id = <access key id>
aws_secret_access_key = <secret access key>
region = us-east-2
```

26

AWS CLI - 2

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



Sign-in credentials

Console password	Disabled Manage password
Console login link	N/A
Last login	N/A
Assigned MFA device	No
Signing certificates	YCE3WKMZNEFS6BQZYYYHQDJGQAQM6VHL Active 2017-04-03 16:55 PDT

Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. Learn more

Create access key

Access key ID	Created	Last used	Status	
AKIAJTZVNQFP6PP6M2YQ	2017-04-02 22:56 PDT	2017-04-04 00:13 PDT with ec2 in N/A	Active | Make inactive	✖

27

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

28

ALTERNATIVE CLI

- `sudo apt install ec2-api-tools`
- Predates "awscli" package
- API specifically for ec2 (not all amazon web services)
- Provides more concise output (generally no JSON)
- 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`

29

INSPECTING INSTANCE INFORMATION

- Find your instance ID (from any EC2 VM):

```
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 (if available on VM??)

30

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 . . .

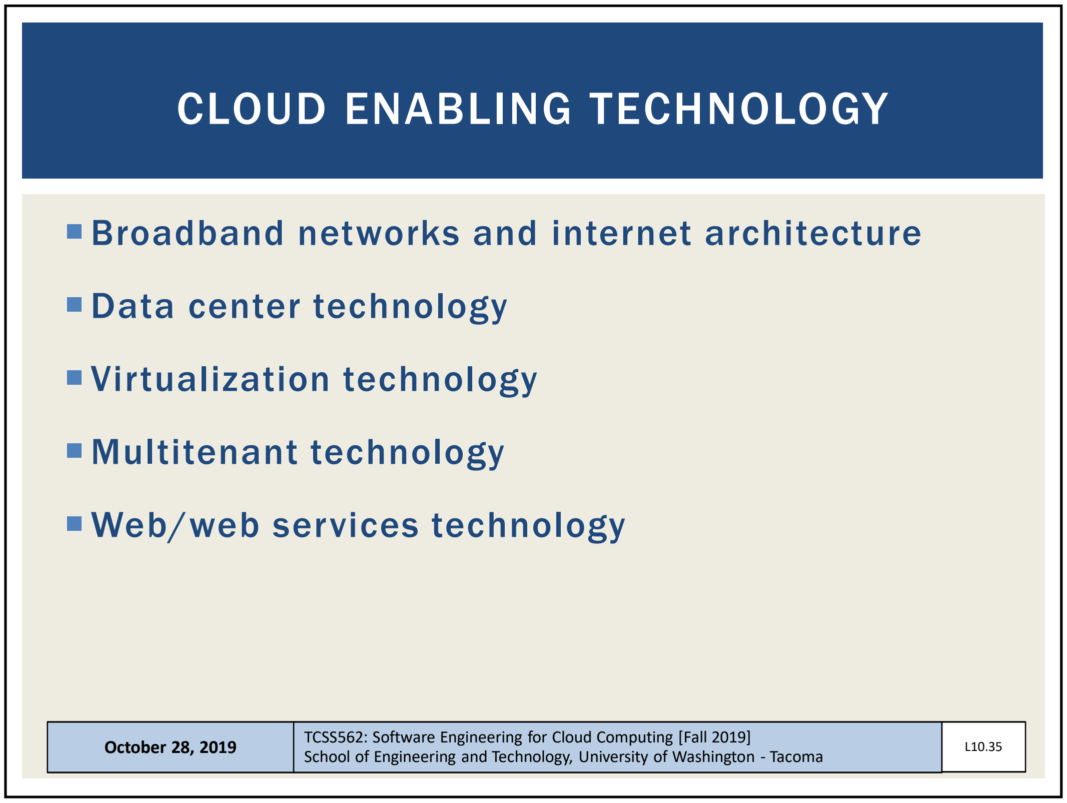
CREATE, UPLOAD, AND REGISTER NEW INSTANCE STORE AMI SCRIPT:

```
# for legacy instance store images - requires ec2-ami-tools package
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
```

33

CLOUD ENABLING TECHNOLOGY

34

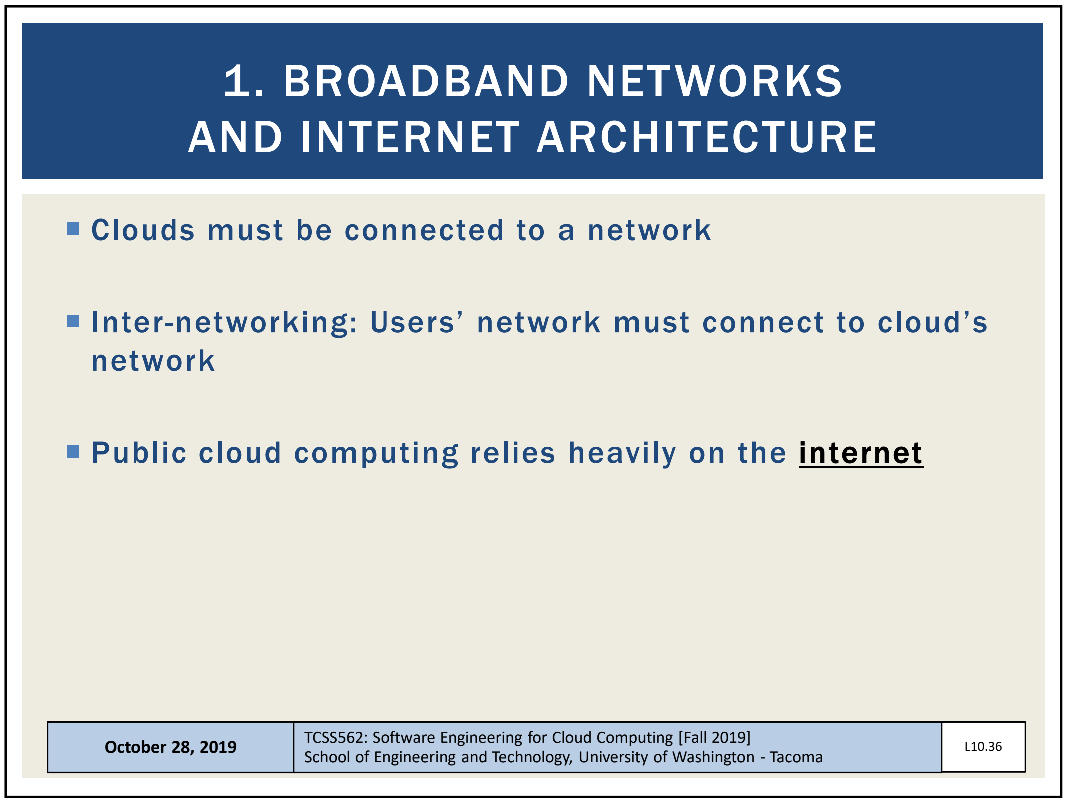


CLOUD ENABLING TECHNOLOGY

- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.35

35



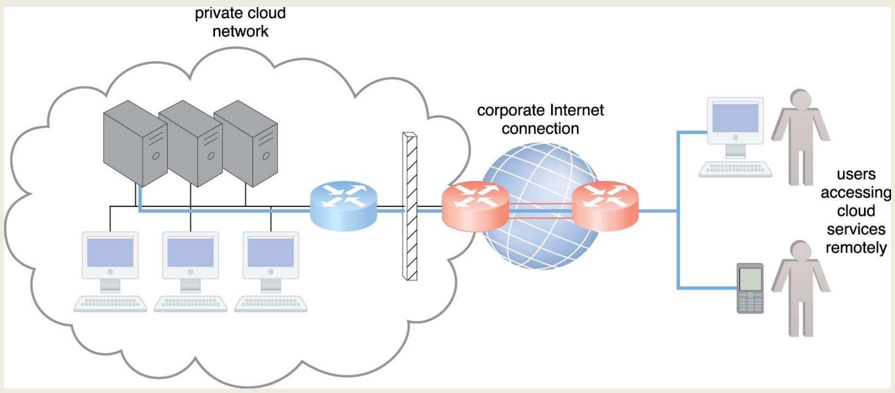
1. BROADBAND NETWORKS AND INTERNET ARCHITECTURE

- Clouds must be connected to a network
- Inter-networking: Users' network must connect to cloud's network
- Public cloud computing relies heavily on the **internet**

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.36

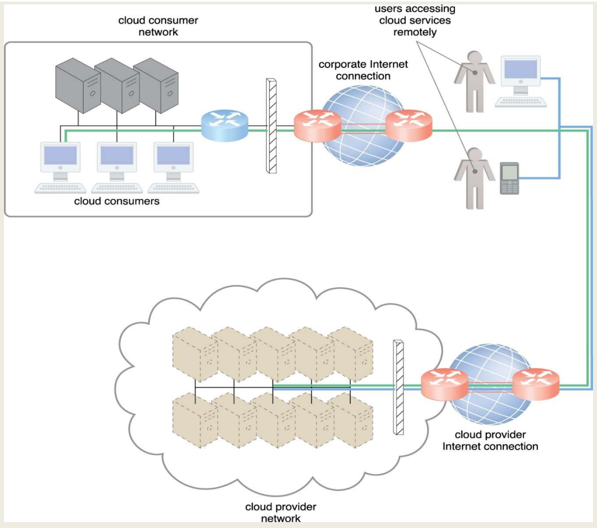
36

PRIVATE CLOUD NETWORKING


private cloud network
corporate Internet connection
users accessing cloud services remotely

37

PUBLIC CLOUD NETWORKING


cloud consumer network
users accessing cloud services remotely
corporate Internet connection
cloud consumers
cloud provider Internet connection
cloud provider network

38

INTERNETWORKING KEY POINTS

- Cloud consumers and providers typically communicate via the internet
- Decentralized provisioning and management model is not controlled by the cloud consumers or providers
- Inter-networking (internet) relies on connectionless packet switching and route-based interconnectivity
- Routers and switches support communication
- Network bandwidth and latency influence QoS, which is heavily impacted by network congestion

39

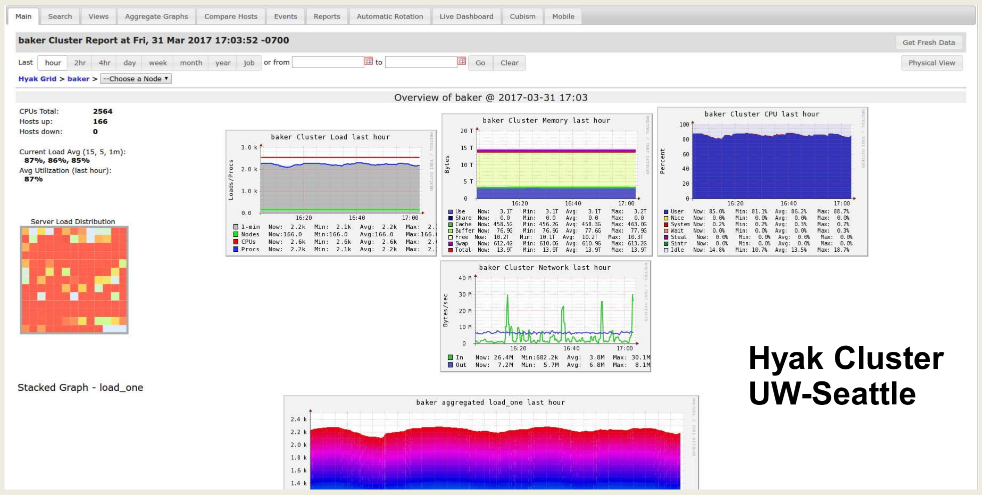
2. DATA CENTER TECHNOLOGY

- Grouping servers together (clusters):
- Enables power sharing
- Higher efficiency in shared IT resource usage (less duplication of effort)
- Improved accessibility and organization

- Key components:
  - Virtualized and physical server resources
  - Standardized, modular hardware
  - Automation support: ease server provisioning, configuration, patching, monitoring without supervision… ***tools are desirable***

40

CLUSTER MANAGEMENT TOOLS



baker Cluster Report at Fri, 31 Mar 2017 17:03:52 -0700

Hyak Grid > baker > --Choose a Node

Overview of baker @ 2017-03-31 17:03

CPUs Total: 2564
Hosts up: 166
Hosts down: 0

Current Load Avg (15, 5, 1m):
87%, 86%, 85%
Avg Utilization (last hour):
87%

```
baker Cluster Load last hour
1-min   Now:  2.2k   Min:  2.1k   Avg:  2.2k   Max:  2.
Nodes   Now:166.0    Min:166.0    Avg:166.0    Max:166.
CPUs    Now:  2.6k   Min:  2.6k   Avg:  2.6k   Max:  2.
Procs   Now:  2.2k   Min:  2.1k   Avg:  2.2k   Max:  2.
```

```
baker Cluster Memory last hour
Use    Now:   3.1T   Min:   3.1T   Avg:   3.1T   Max:   3.2T
Share  Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
Cache  Now: 458.5G   Min: 456.2G   Avg: 458.3G   Max: 463.0G
Buffer Now:  76.9G   Min:  76.9G   Avg:  77.6G   Max:  77.9G
Free   Now:  10.2T   Min:  10.1T   Avg:  10.2T   Max:  10.3T
Swap   Now: 612.4G   Min: 610.0G   Avg: 610.9G   Max: 613.2G
Total  Now:  13.9T   Min:  13.9T   Avg:  13.9T   Max:  13.9T
```

```
baker Cluster CPU last hour
User   Now: 85.0%   Min: 81.1%   Avg: 86.2%   Max: 88.7%
Nice   Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
System Now:  0.2%   Min:  0.2%   Avg:  0.3%   Max:  0.7%
Wait   Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.3%
Steal  Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
Sintr  Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
Idle   Now: 14.8%   Min: 10.7%   Avg: 13.5%   Max: 18.7%
```

```
baker Cluster Network last hour
In   Now: 26.4M   Min:682.2k   Avg:  3.8M   Max: 30.1M
Out  Now:  7.2M   Min:  5.7M   Avg:  6.8M   Max:  8.1M
```

**Hyak Cluster
UW-Seattle**

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.41

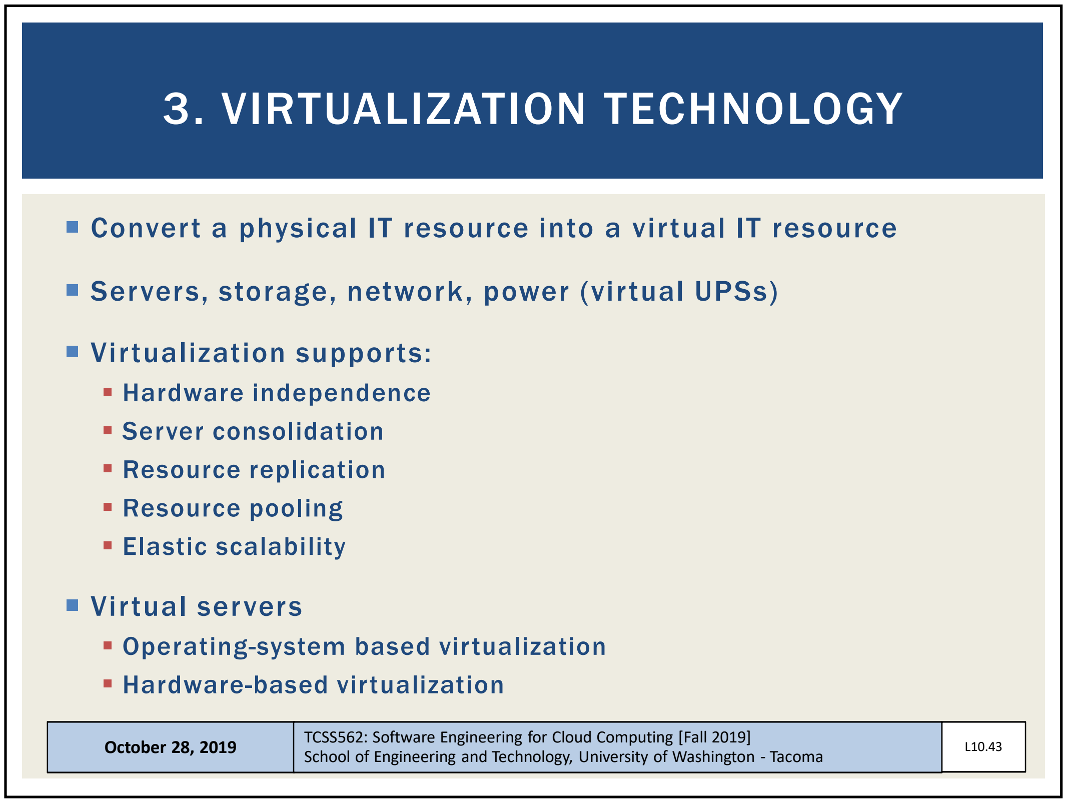
41

DATA CENTER TECHNOLOGY – KEY COMPONENTS

- Remote operation / management
- **High availability support**: **redundant everything** Includes: power supplies, cabling, environmental control systems, communication links, duplicate warm replica hardware
- **Secure design**: physical and logical access control
- **Servers**: rackmount, etc.
- **Storage**: hard disk arrays (RAID), storage area network (SAN): disk array with dedicated network, network attached storage (NAS): disk array on network for NFS, etc.
- **Network hardware**: backbone routers (WAN to LAN connectivity), firewalls, VPN gateways, managed switches/routers

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.42

42

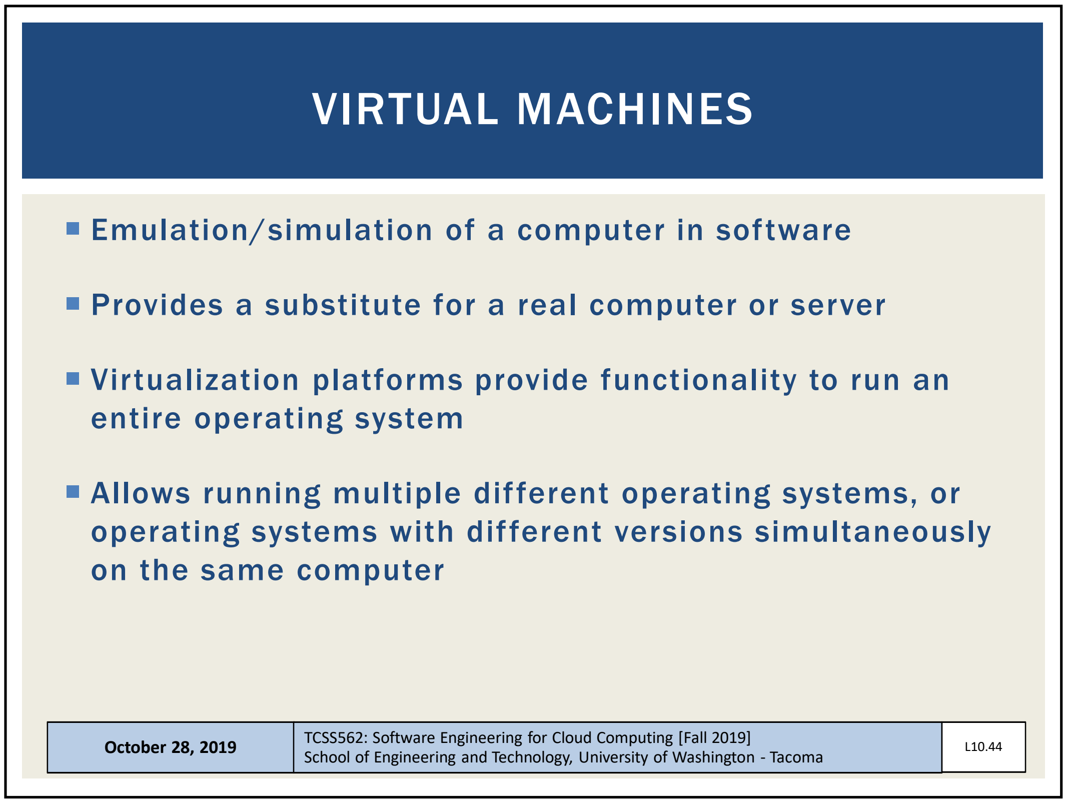


3. VIRTUALIZATION TECHNOLOGY

- Convert a physical IT resource into a virtual IT resource
- Servers, storage, network, power (virtual UPSs)
- Virtualization supports:
  - Hardware independence
  - Server consolidation
  - Resource replication
  - Resource pooling
  - Elastic scalability
- Virtual servers
  - Operating-system based virtualization
  - Hardware-based virtualization

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.43

43



VIRTUAL MACHINES

- Emulation/simulation of a computer in software
- Provides a substitute for a real computer or server
- Virtualization platforms provide functionality to run an entire operating system
- Allows running multiple different operating systems, or operating systems with different versions simultaneously on the same computer

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.44

44

KEY VIRTUALIZATION TRADEOFF

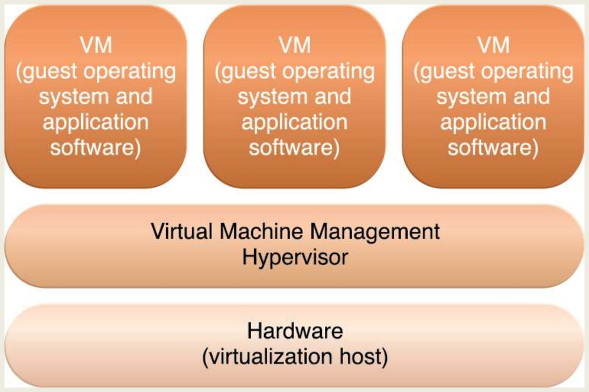
- Tradeoff space:

Degree of Hardware Abstraction

Concerns:
Overhead
Performance
Isolation
Security

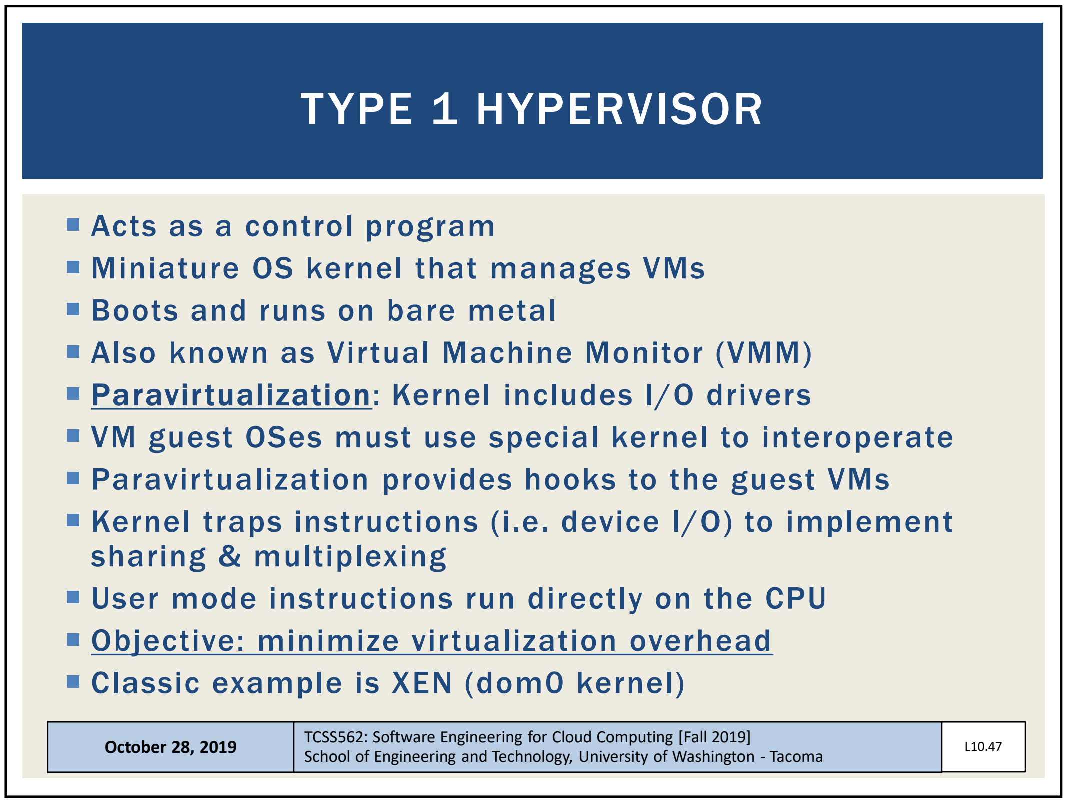
45

TYPE 1 HYPERVISOR


VM (guest operating system and application software)
VM (guest operating system and application software)
VM (guest operating system and application software)
Virtual Machine Management Hypervisor
Hardware (virtualization host)

- Host OS and VMs run atop the hypervisor
- The boot OS is the hypervisor kernel
- Xen dom0

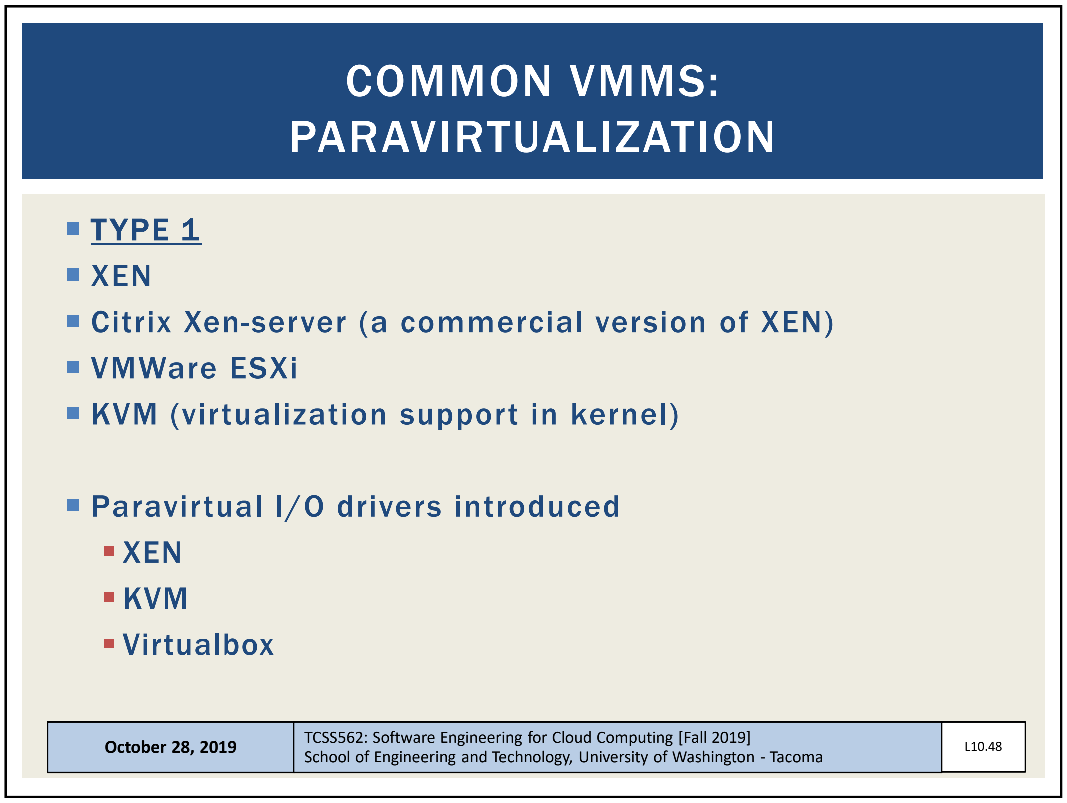
46



TYPE 1 HYPERVISOR

- Acts as a control program
- Miniature OS kernel that manages VMs
- Boots and runs on bare metal
- Also known as Virtual Machine Monitor (VMM)
- Paravirtualization: Kernel includes I/O drivers
- VM guest OSes must use special kernel to interoperate
- Paravirtualization provides hooks to the guest VMs
- Kernel traps instructions (i.e. device I/O) to implement sharing & multiplexing
- User mode instructions run directly on the CPU
- Objective: minimize virtualization overhead
- Classic example is XEN (dom0 kernel)

47



COMMON VMMS: PARAVIRTUALIZATION

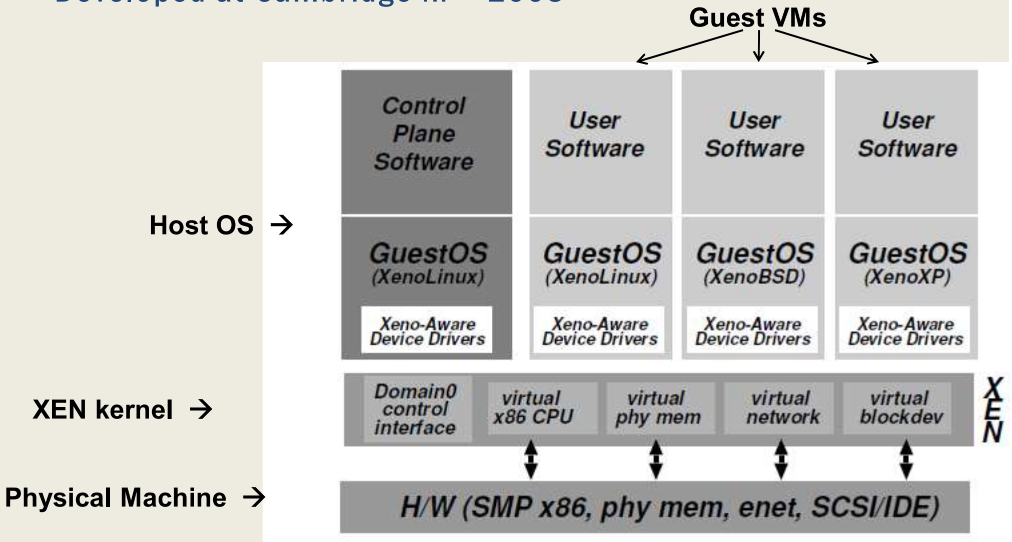
- TYPE 1
- XEN
- Citrix Xen-server (a commercial version of XEN)
- VMWare ESXi
- KVM (virtualization support in kernel)

- Paravirtual I/O drivers introduced
  - XEN
  - KVM
  - Virtualbox

48

XEN

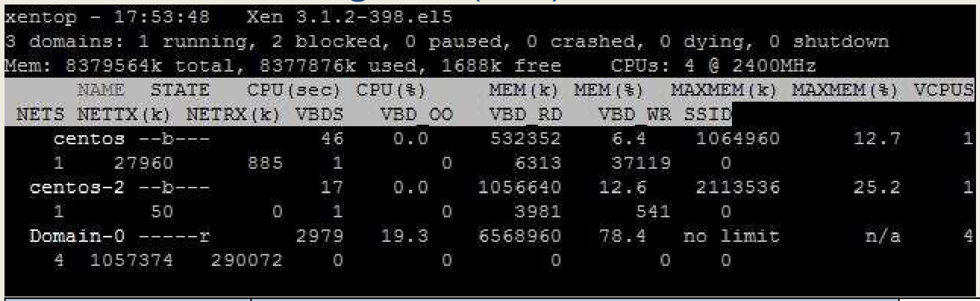
- Developed at Cambridge in ~ 2003



49

XEN - 2

- VMs managed as "domains"
- Domain 0 is the hypervisor domain
  - Host OS is installed to run on bare-metal, but doesn't directly facilitate virtualization (*unlike KVM*)
- Domains 1..n are guests (VMs) – not bare-metal



```
xentop - 17:53:48   Xen 3.1.2-398.el5
3 domains: 1 running, 2 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 8379564k total, 8377876k used, 1688k free    CPUs: 4 @ 2400MHz
      NAME  STATE   CPU(sec) CPU(%)     MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%) VCPUS
 NETS NETTX(k) NETRX(k) VBDS   VBD_OO   VBD_RD   VBD_WR SSID
    centos --b---         46    0.0     532352    6.4    1064960      12.7     1
   1    27960      885    1        0     6313    37119    0
  centos-2 --b---         17    0.0    1056640   12.6    2113536      25.2     1
   1       50        0    1        0     3981      541    0
  Domain-0 -----r       2979   19.3    6568960   78.4   no limit       n/a     4
   4  1057374   290072    0        0        0        0    0
```

50

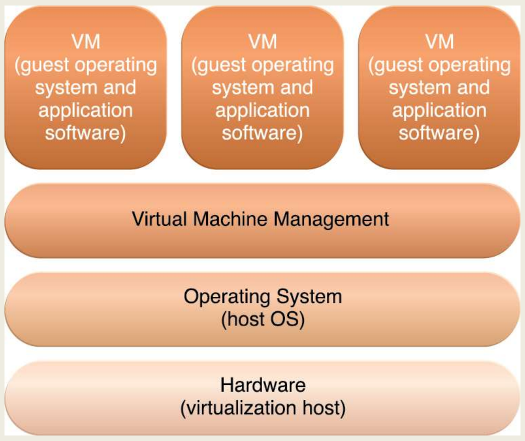
XEN - 3

- Physical machine boots special XEN kernel
- Kernel provides paravirtual API to manage CPU & device multiplexing
- Guests require modified XEN-aware kernels
- Xen supports full-virtualization for unmodified OS guests in hvm mode
- Amazon EC2 largely based on modified version of XEN hypervisor (EC2 gens 1-4)
- XEN provides its own CPU schedulers, I/O scheduling

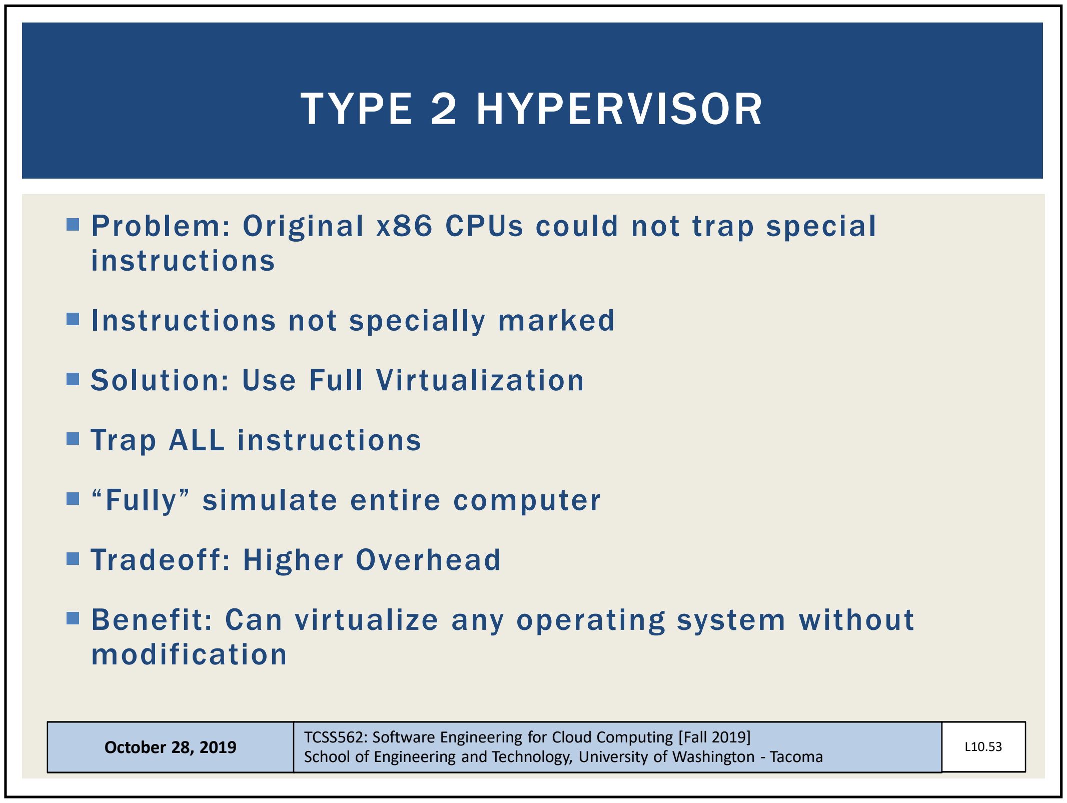
51

TYPE 2 HYPERVISOR

- Adds additional layer


VM
(guest operating system and application software)
VM
(guest operating system and application software)
VM
(guest operating system and application software)
Virtual Machine Management
Operating System
(host OS)
Hardware
(virtualization host)

52

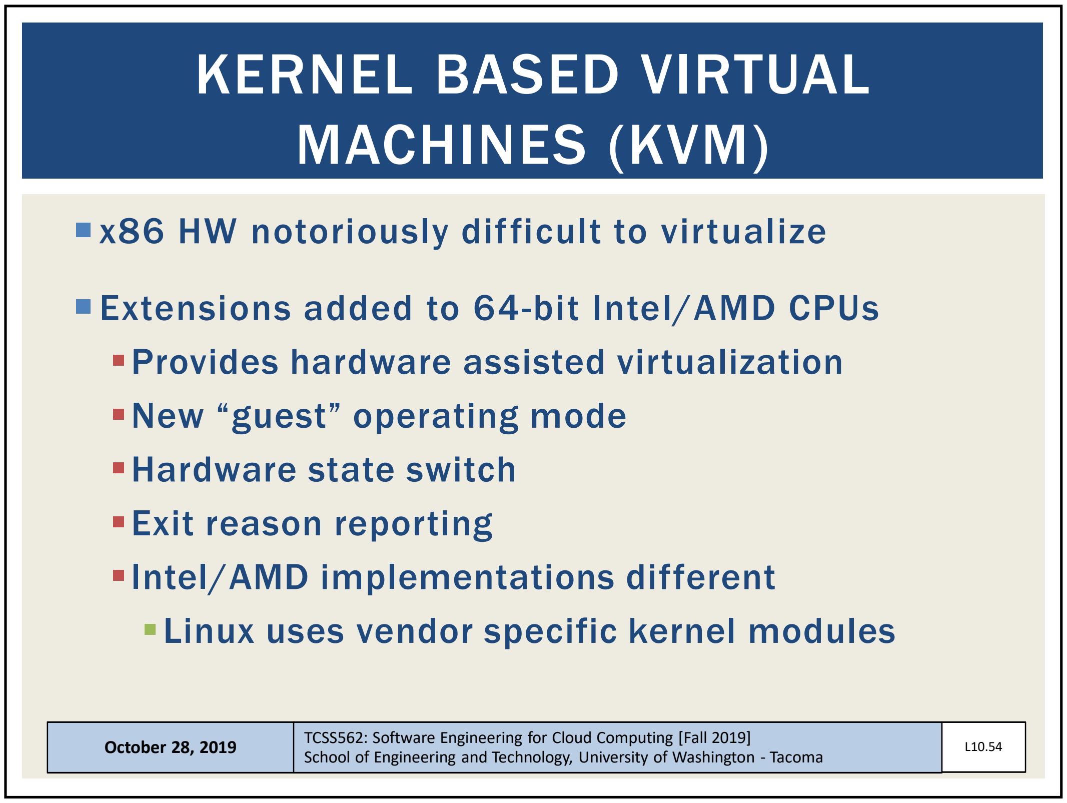


TYPE 2 HYPERVISOR

- Problem: Original x86 CPUs could not trap special instructions
- Instructions not specially marked
- Solution: Use Full Virtualization
- Trap ALL instructions
- "Fully" simulate entire computer
- Tradeoff: Higher Overhead
- Benefit: Can virtualize any operating system without modification

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.53

53



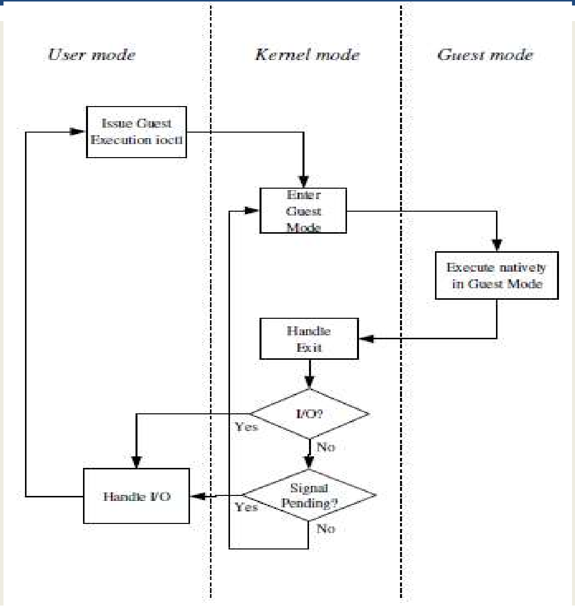
KERNEL BASED VIRTUAL MACHINES (KVM)

- x86 HW notoriously difficult to virtualize
- Extensions added to 64-bit Intel/AMD CPUs
  - Provides hardware assisted virtualization
  - New "guest" operating mode
  - Hardware state switch
  - Exit reason reporting
  - Intel/AMD implementations different
    - Linux uses vendor specific kernel modules

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.54

54

KVM – 2

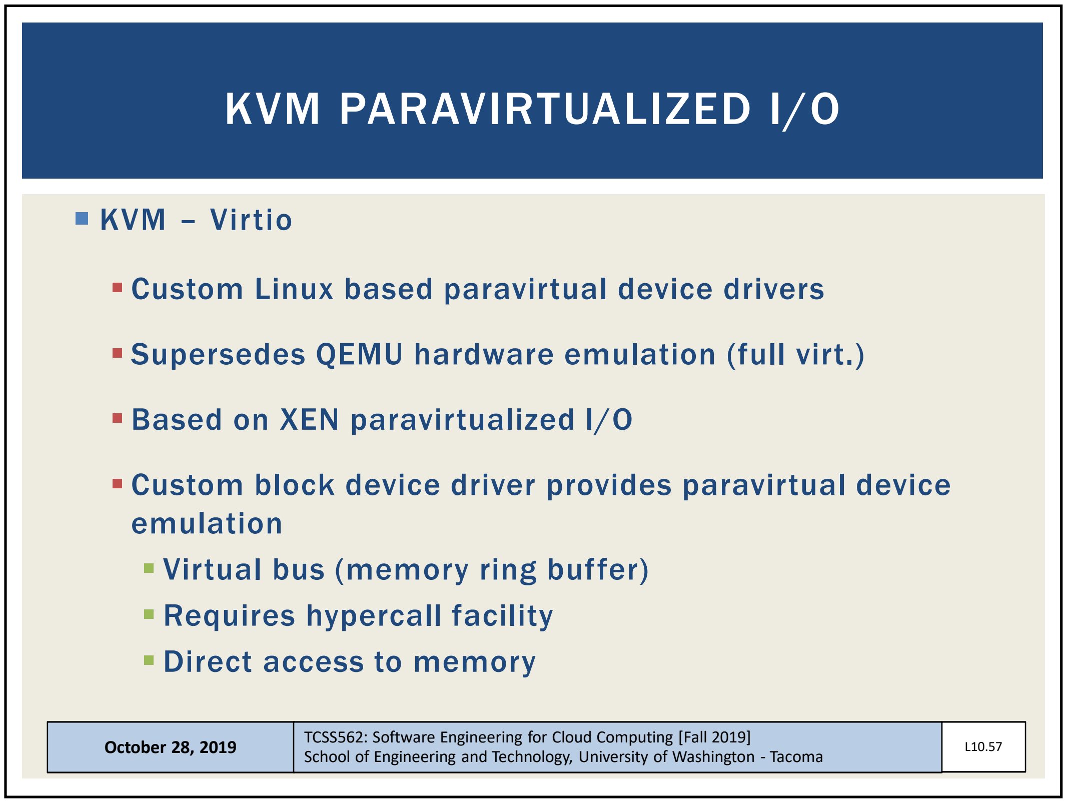


55

KVM – 3

- KVM has /dev/kvm device file node
  - Linux character device, with operations:
    - Create new VM
    - Allocate memory to VM
    - Read/write virtual CPU registers
    - Inject interrupts into vCPUs
    - Running vCPUs
- VMs run as Linux processes
  - Scheduled by host Linux OS
  - Can be pinned to specific cores with "taskset"

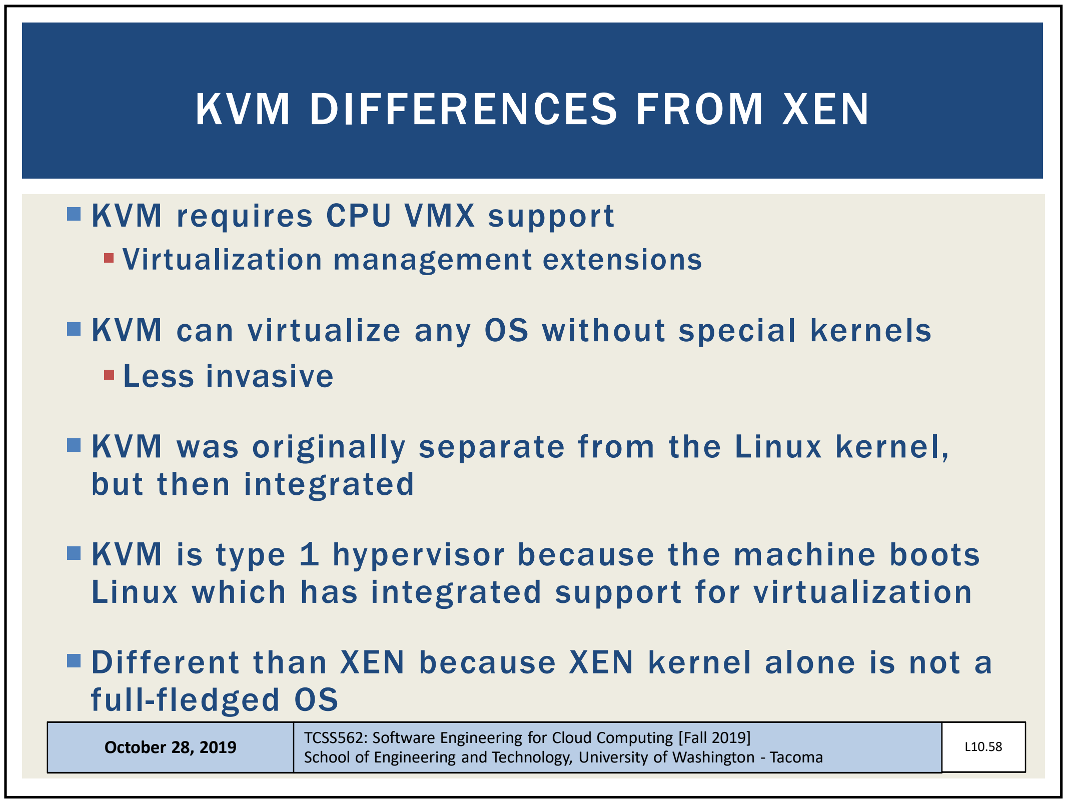
56



KVM PARAVIRTUALIZED I/O

- KVM – Virtio
  - Custom Linux based paravirtual device drivers
  - Supersedes QEMU hardware emulation (full virt.)
  - Based on XEN paravirtualized I/O
  - Custom block device driver provides paravirtual device emulation
    - Virtual bus (memory ring buffer)
    - Requires hypercall facility
    - Direct access to memory

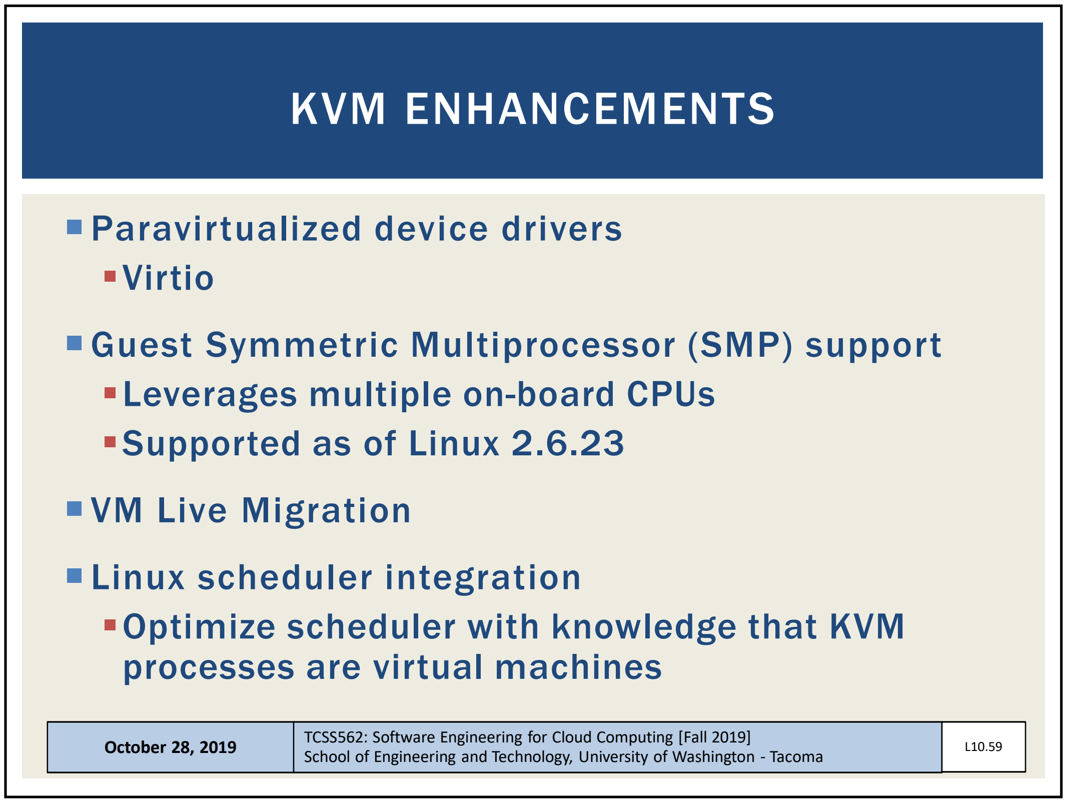
57



KVM DIFFERENCES FROM XEN

- KVM requires CPU VMX support
  - Virtualization management extensions
- KVM can virtualize any OS without special kernels
  - Less invasive
- KVM was originally separate from the Linux kernel, but then integrated
- KVM is type 1 hypervisor because the machine boots Linux which has integrated support for virtualization
- Different than XEN because XEN kernel alone is not a full-fledged OS

58

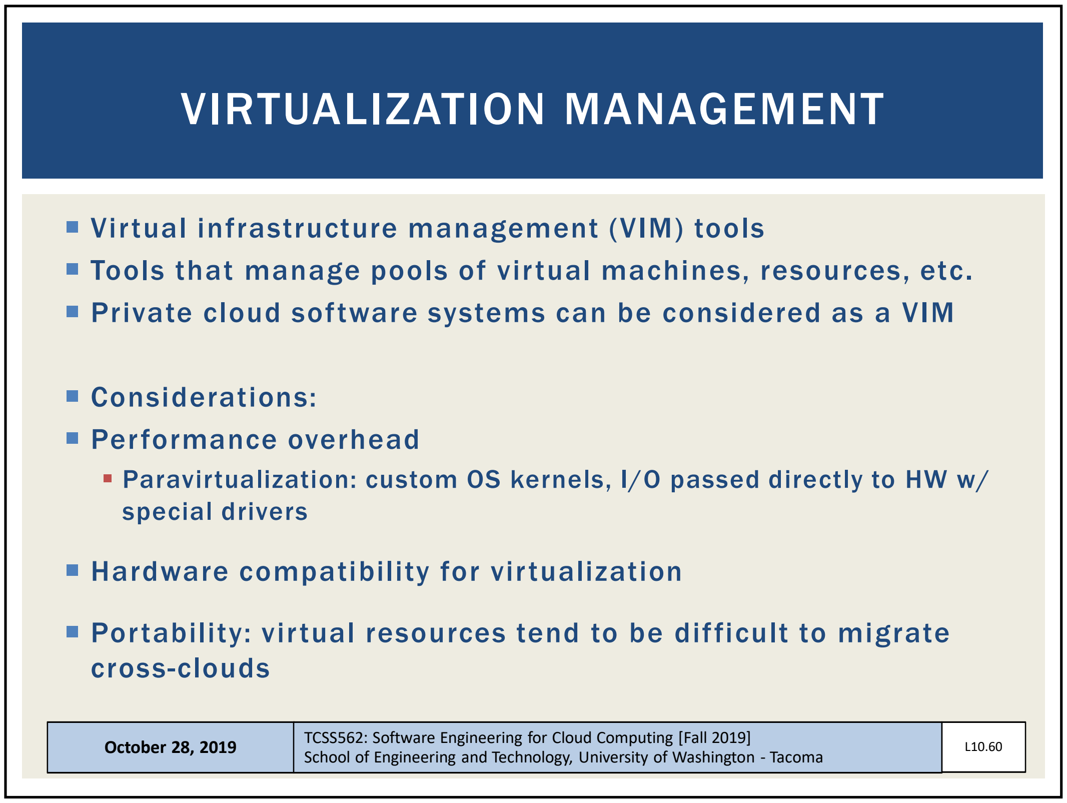


KVM ENHANCEMENTS

- Paravirtualized device drivers
  - Virtio
- Guest Symmetric Multiprocessor (SMP) support
  - Leverages multiple on-board CPUs
  - Supported as of Linux 2.6.23
- VM Live Migration
- Linux scheduler integration
  - Optimize scheduler with knowledge that KVM processes are virtual machines

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.59

59



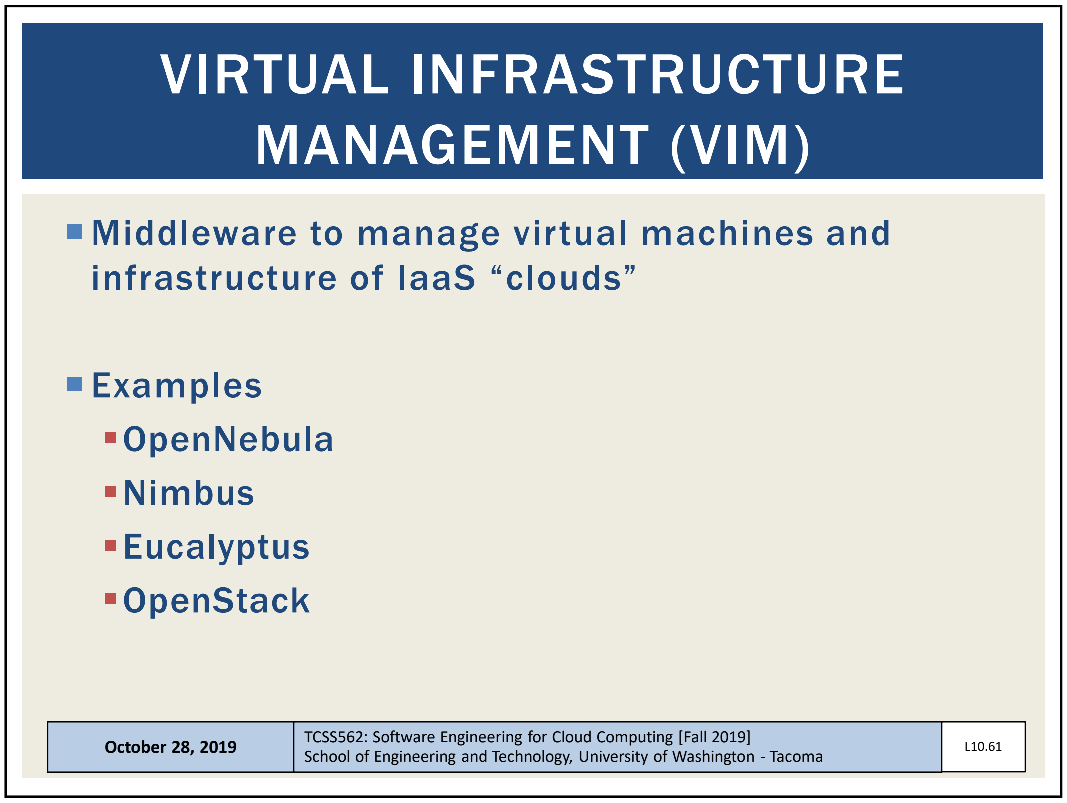
VIRTUALIZATION MANAGEMENT

- Virtual infrastructure management (VIM) tools
- Tools that manage pools of virtual machines, resources, etc.
- Private cloud software systems can be considered as a VIM

- Considerations:
- Performance overhead
  - Paravirtualization: custom OS kernels, I/O passed directly to HW w/ special drivers
- Hardware compatibility for virtualization
- Portability: virtual resources tend to be difficult to migrate cross-clouds

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.60

60

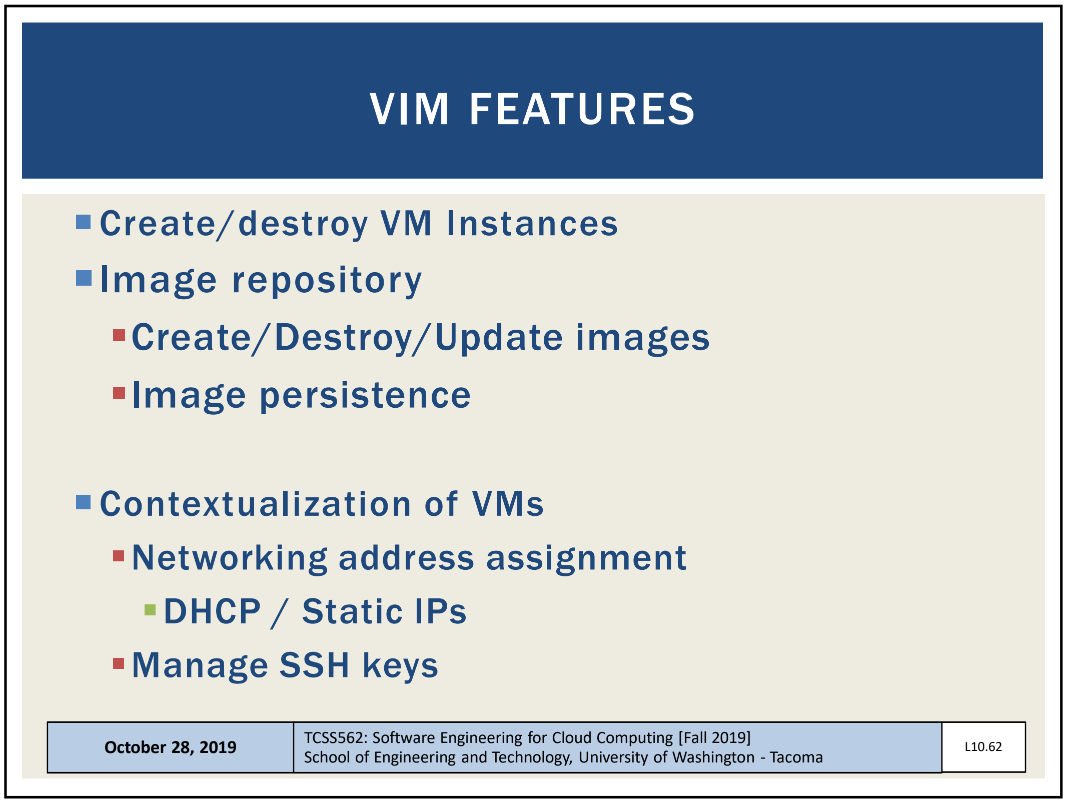


VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

- Middleware to manage virtual machines and infrastructure of IaaS “clouds”

- Examples
  - OpenNebula
  - Nimbus
  - Eucalyptus
  - OpenStack

61

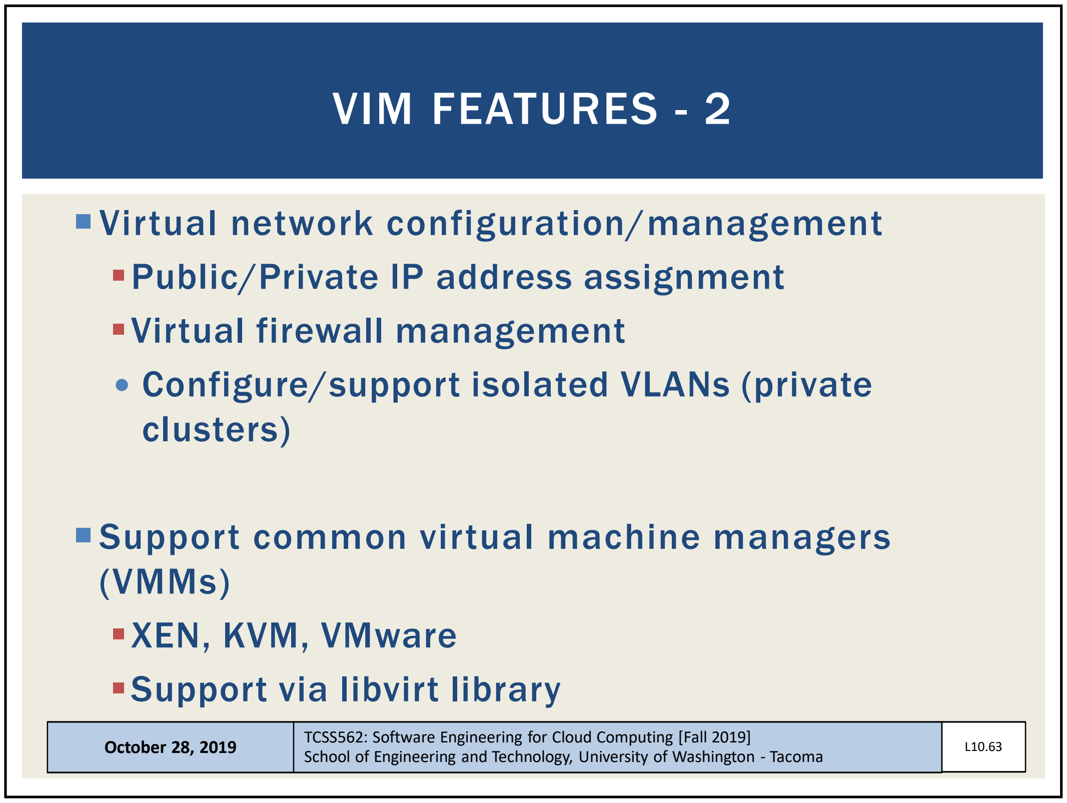


VIM FEATURES

- Create/destroy VM Instances
- Image repository
  - Create/Destroy/Update images
  - Image persistence

- Contextualization of VMs
  - Networking address assignment
    - DHCP / Static IPs
  - Manage SSH keys

62

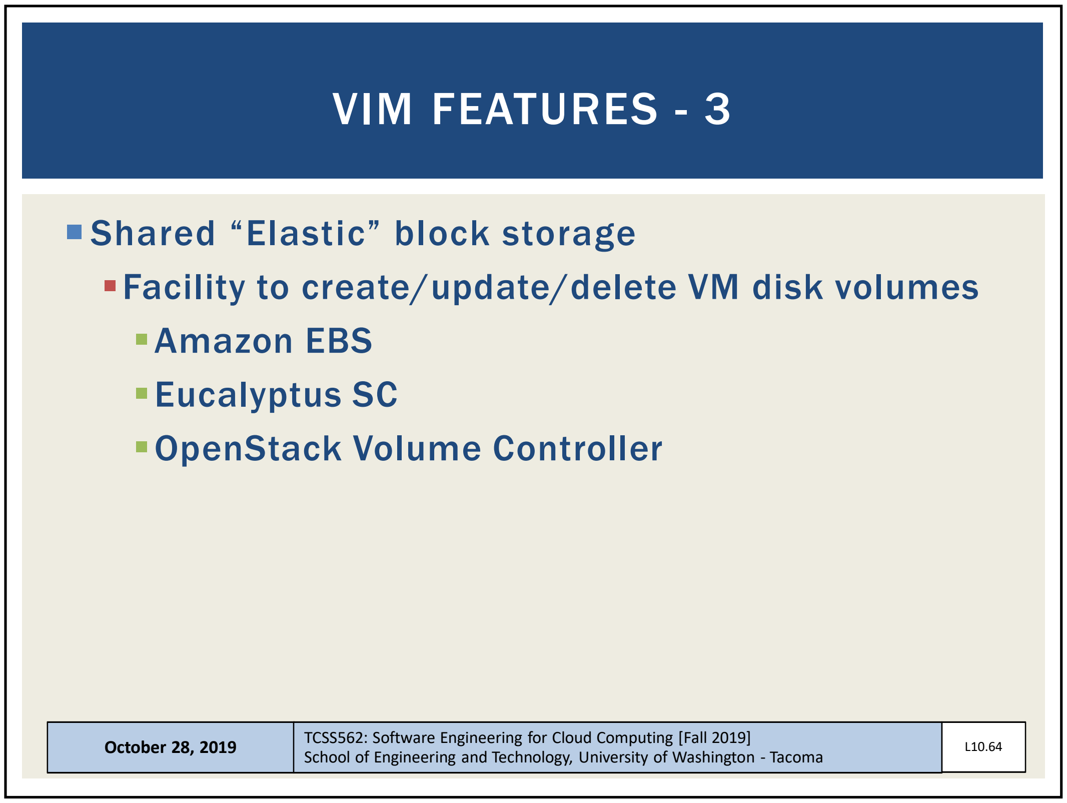


VIM FEATURES - 2

- Virtual network configuration/management
  - Public/Private IP address assignment
  - Virtual firewall management
  - Configure/support isolated VLANs (private clusters)
- Support common virtual machine managers (VMMs)
  - XEN, KVM, VMware
  - Support via libvirt library

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.63

63

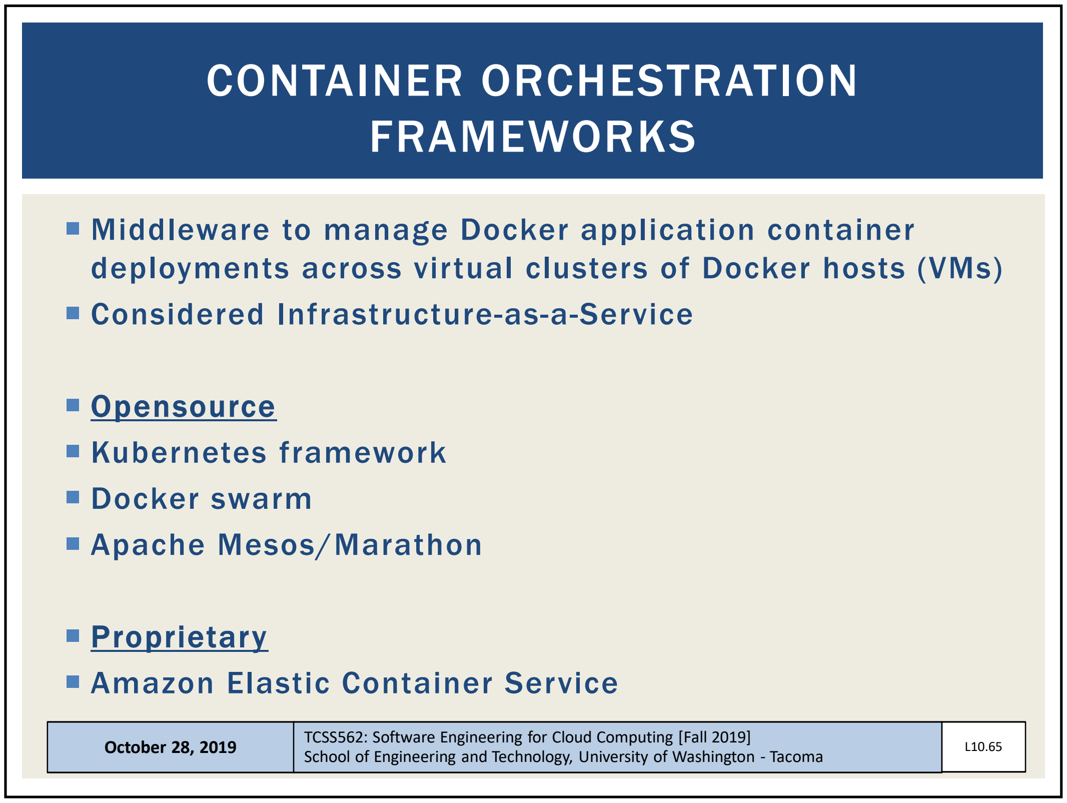


VIM FEATURES - 3

- Shared "Elastic" block storage
  - Facility to create/update/delete VM disk volumes
    - Amazon EBS
    - Eucalyptus SC
    - OpenStack Volume Controller

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.64

64



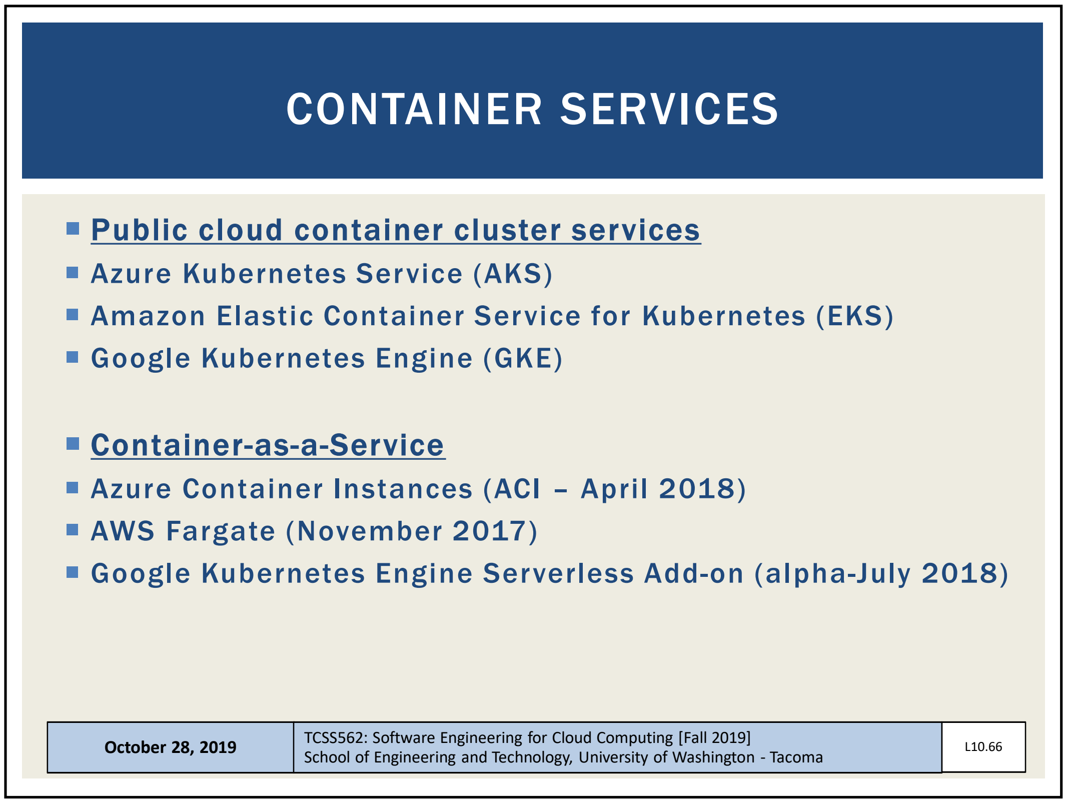
CONTAINER ORCHESTRATION FRAMEWORKS

- Middleware to manage Docker application container deployments across virtual clusters of Docker hosts (VMs)
- Considered Infrastructure-as-a-Service

- **Opensource**
- Kubernetes framework
- Docker swarm
- Apache Mesos/Marathon

- **Proprietary**
- Amazon Elastic Container Service

65



CONTAINER SERVICES

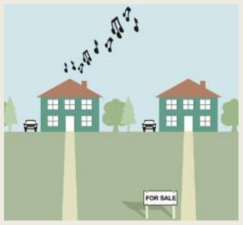
- **Public cloud container cluster services**
- Azure Kubernetes Service (AKS)
- Amazon Elastic Container Service for Kubernetes (EKS)
- Google Kubernetes Engine (GKE)

- **Container-as-a-Service**
- Azure Container Instances (ACI – April 2018)
- AWS Fargate (November 2017)
- Google Kubernetes Engine Serverless Add-on (alpha-July 2018)

66

4. MULTITENANT APPLICATIONS

- Each tenant (like in an apartment) has their own view of the application
- Tenants are unaware of their neighbors
- Tenants can only access their data, no access to data and configuration that is not their own

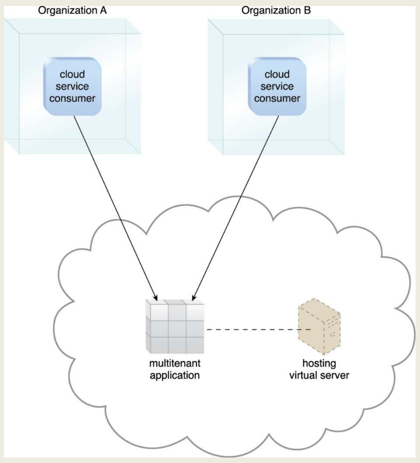


- Customizable features
  - UI, business process, data model, access control
- Application architecture
  - User isolation, data security, recovery/backup by tenant, scalability for a tenant, for tenants, metered usage, data tier isolation

67

MULTITENANT APPS - 2

- Forms the basis for SaaS (applications)



68

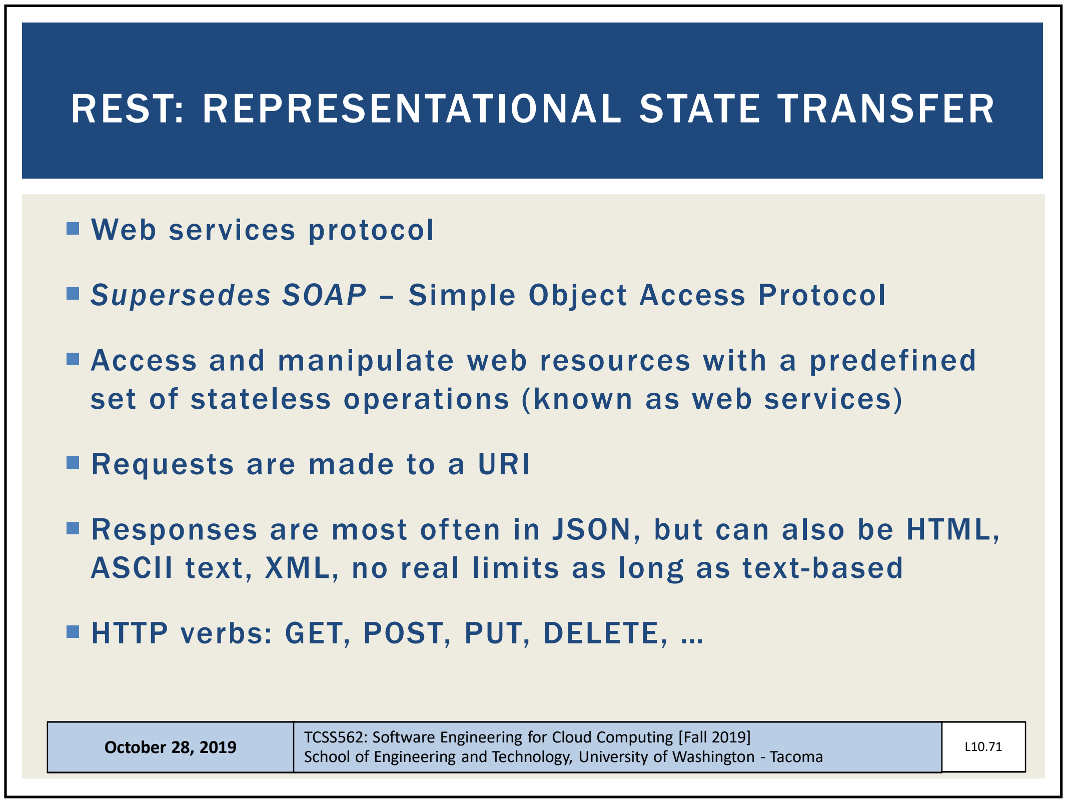
WEB SERVICES/WEB

- Web services technology is a key foundation of cloud computing's "as-a-service" cloud delivery model
- SOAP – "Simple" object access protocol
  - First generation web services
  - WSDL – web services description language
  - UDDI – universal description discovery and integration
  - SOAP services have their own unique interfaces
- REST – instead of defining a custom technical interface REST services are built on the use of HTTP protocol
- HTTP GET, PUT, POST, DELETE

HYPERTEXT TRANSPORT PROTOCOL (HTTP)

- An ASCII-based request/reply protocol for transferring information on the web
- HTTP request includes:
  - request method (GET, POST, etc.)
  - Uniform Resource Identifier (URI)
  - HTTP protocol version understood by the client
  - headers—extra info regarding transfer request
- HTTP response from server
  - Protocol version & status code →
  - Response headers
  - Response body

HTTP status codes:
2xx — *all is well*
3xx — *resource moved*
4xx — *access problem*
5xx — *server error*

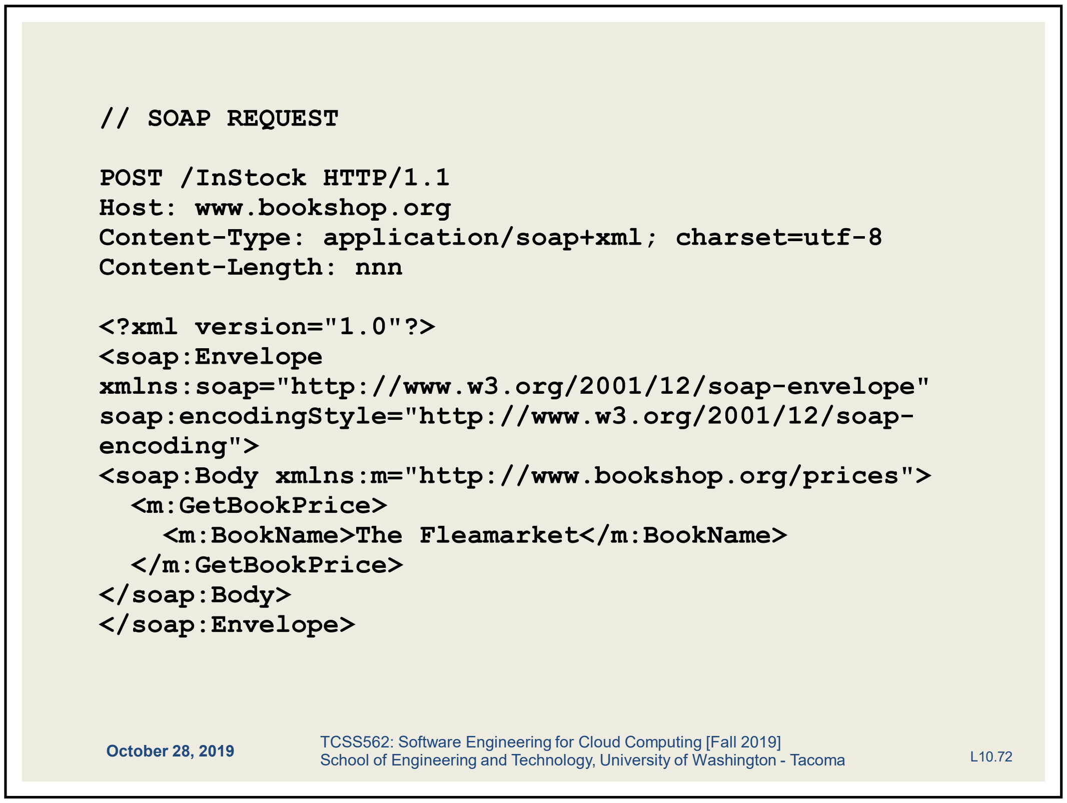


REST: REPRESENTATIONAL STATE TRANSFER

- Web services protocol
- *Supersedes SOAP* – Simple Object Access Protocol
- Access and manipulate web resources with a predefined set of stateless operations (known as web services)
- Requests are made to a URI
- Responses are most often in JSON, but can also be HTML, ASCII text, XML, no real limits as long as text-based
- HTTP verbs: GET, POST, PUT, DELETE, …

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.71

71



```
// SOAP REQUEST

POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
<soap:Body xmlns:m="http://www.bookshop.org/prices">
  <m:GetBookPrice>
    <m:BookName>The Fleamarket</m:BookName>
  </m:GetBookPrice>
</soap:Body>
</soap:Envelope>
```

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.72

72

```
// SOAP RESPONSE
POST /InStock HTTP/1.1
Host: www.bookshop.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
<soap:Body xmlns:m="http://www.bookshop.org/prices">
  <m:GetBookPriceResponse>
    <m: Price>10.95</m: Price>
  </m:GetBookPriceResponse>
</soap:Body>
</soap:Envelope>
```

73

```
// WSDL Service Definition
<?xml version="1.0" encoding="UTF-8"?>
<definitions  name ="DayOfWeek"
  targetNamespace="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:tns="http://www.roguewave.com/soapworx/examples/DayOfWeek.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="DayOfWeekInput">
    <part name="date" type="xsd:date"/>
  </message>
  <message name="DayOfWeekResponse">
    <part name="dayOfWeek" type="xsd:string"/>
  </message>
  <portType name="DayOfWeekPortType">
    <operation name="GetDayOfWeek">
      <input message="tns:DayOfWeekInput"/>
      <output message="tns:DayOfWeekResponse"/>
    </operation>
  </portType>
  <binding name="DayOfWeekBinding" type="tns:DayOfWeekPortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetDayOfWeek">
      <soap:operation soapAction="getdayofweek"/>
      <input>
        <soap:body use="encoded"
          namespace="http://www.roguewave.com/soapworx/examples"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
        namespace="http://www.roguewave.com/soapworx/examples"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>
  <service name="DayOfWeekService" >
    <documentation>
      Returns the day-of-week name for a given date
    </documentation>
    <port name="DayOfWeekPort" binding="tns:DayOfWeekBinding">
      <soap:address location="http://localhost:8090/dayofweek/DayOfWeek"/>
    </port>
   </service>
</definitions>
```

74

REST CLIMATE SERVICES EXAMPLE

- USDA Lat/Long Climate Service Demo
- Just provide a Lat/Long

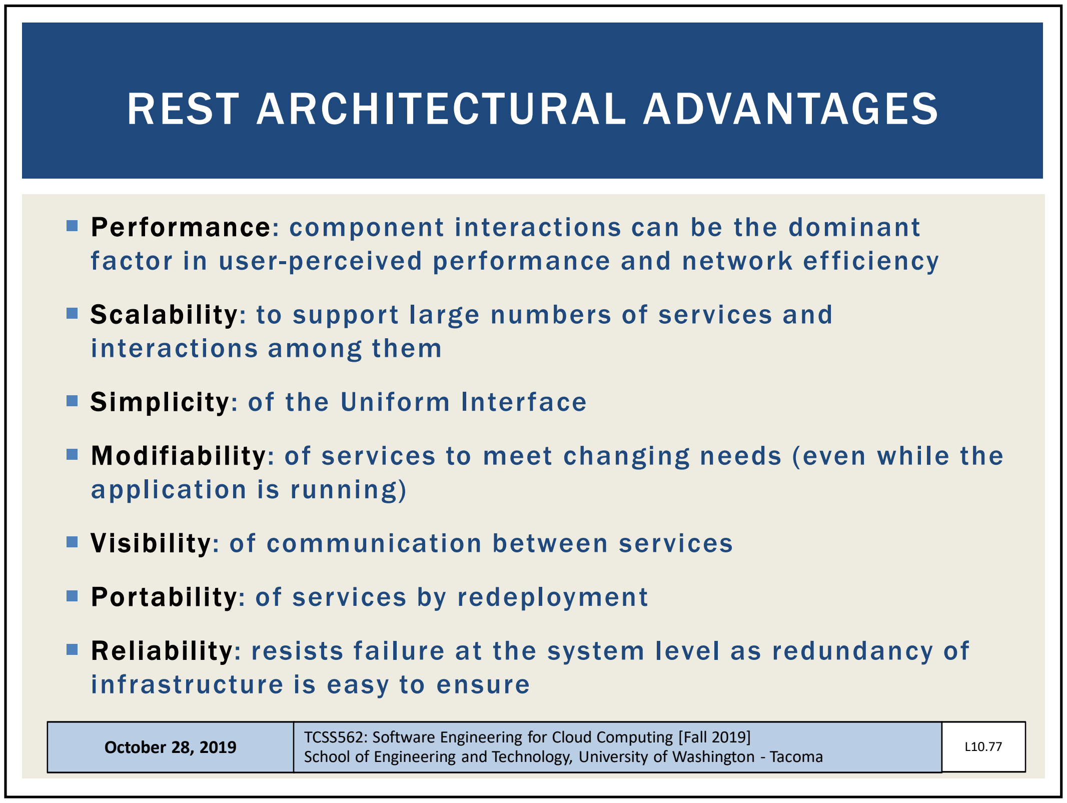
```
// REST/JSON
// Request climate data for Washington

{
 "parameter": [
  {
    "name": "latitude",
    "value":47.2529
  },
  {
    "name": "longitude",
    "value":-122.4443
  }
  ]
}
```

75

REST - 2

- App manipulates one or more types of resources.
- Everything the app does can be characterized as some kind of operation on one or more resources.
- Frequently services are CRUD operations (create/read/update/delete)
  - Create a new resource
  - Read resource(s) matching criterion
  - Update data associated with some resource
  - Destroy a particular a resource
- Resources are often implemented as objects in OO languages

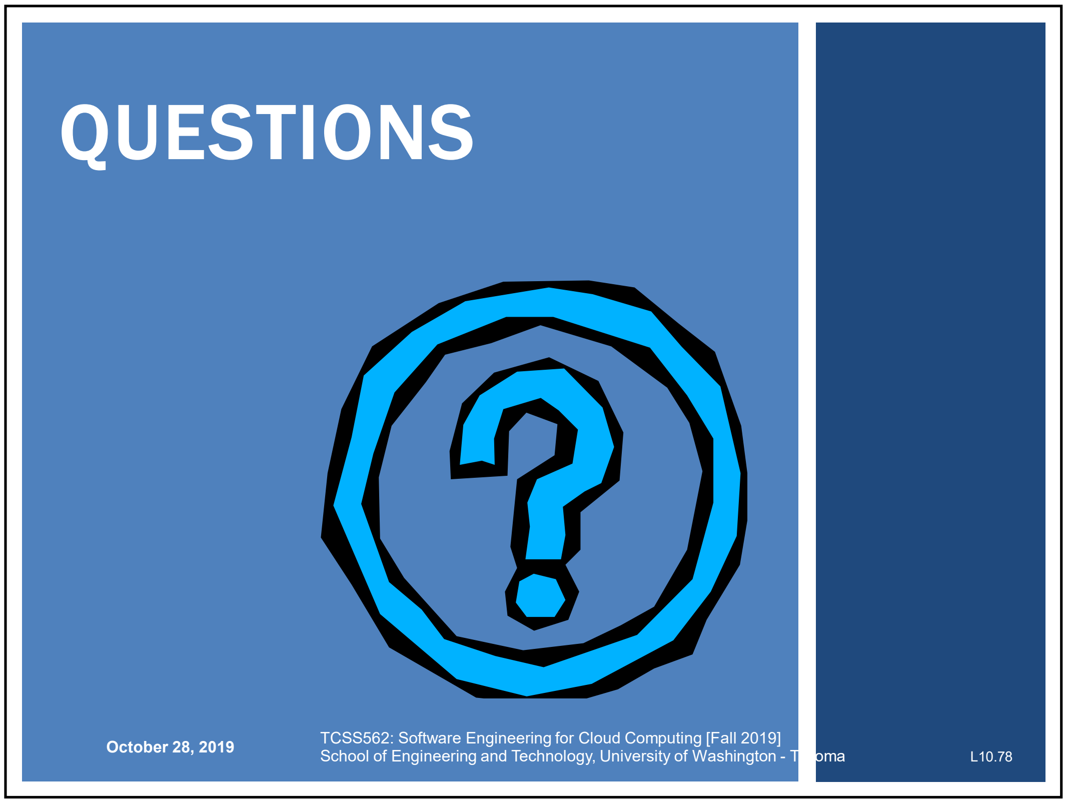
76



REST ARCHITECTURAL ADVANTAGES

- **Performance**: component interactions can be the dominant factor in user-perceived performance and network efficiency
- **Scalability**: to support large numbers of services and interactions among them
- **Simplicity**: of the Uniform Interface
- **Modifiability**: of services to meet changing needs (even while the application is running)
- **Visibility**: of communication between services
- **Portability**: of services by redeployment
- **Reliability**: resists failure at the system level as redundancy of infrastructure is easy to ensure

77



QUESTIONS

78

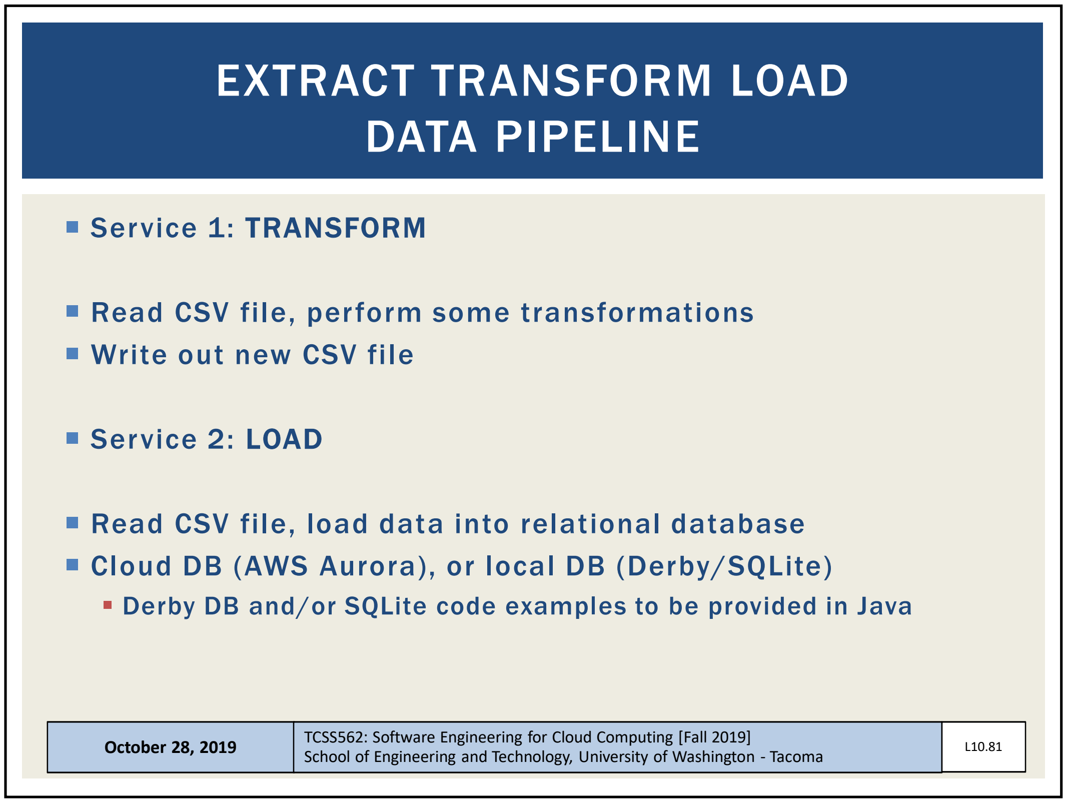
TCSS 562
TERM PROJECT

79

TCSS 562 TERM PROJECT

- Build a serverless cloud native application
- Application provides a case study to design trade-offs:
- Projects will compare and contrast one or more trade-offs:
- Service composition
  - Switchboard architecture
    - Address COLD Starts
    - Infrastructure Freeze/Thaw cycle of AWS Lambda (FaaS)
  - Full service isolation, full service aggregation
- Application flow control
- Programming Languages
- Alternate FaaS Platforms
- Data provisioning

80



EXTRACT TRANSFORM LOAD DATA PIPELINE

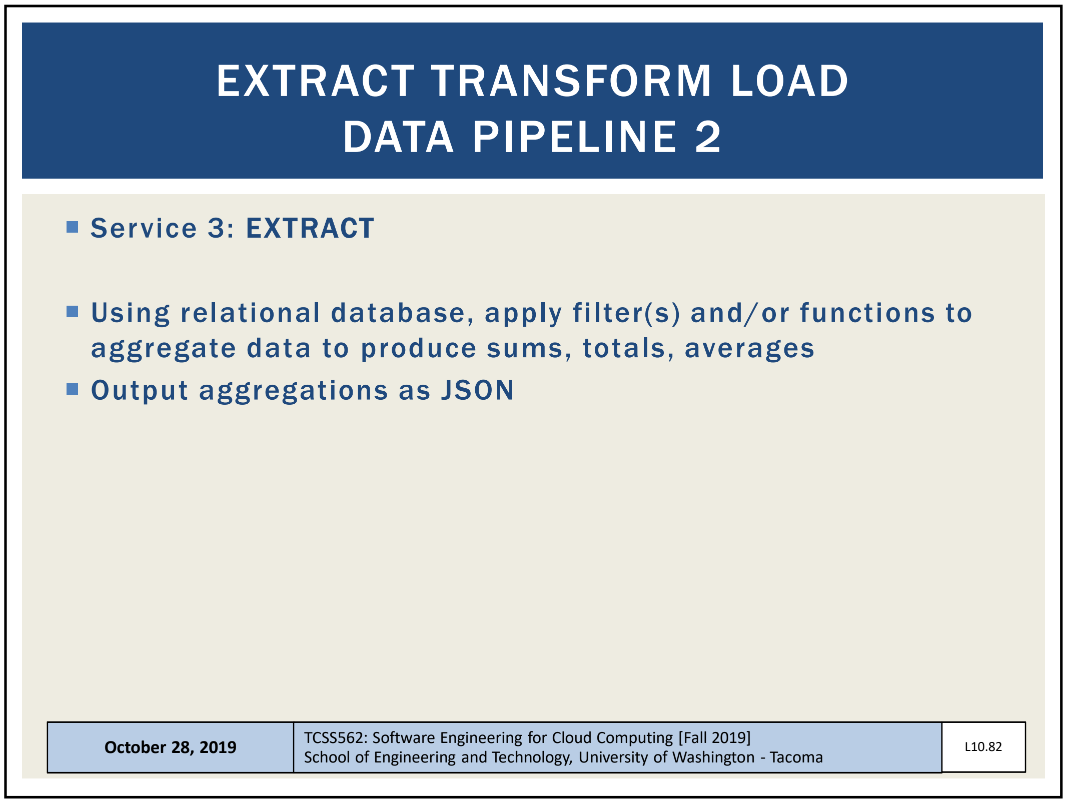
- Service 1: **TRANSFORM**

- 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

81



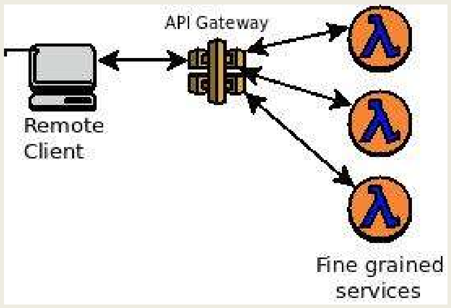
EXTRACT TRANSFORM LOAD DATA PIPELINE 2

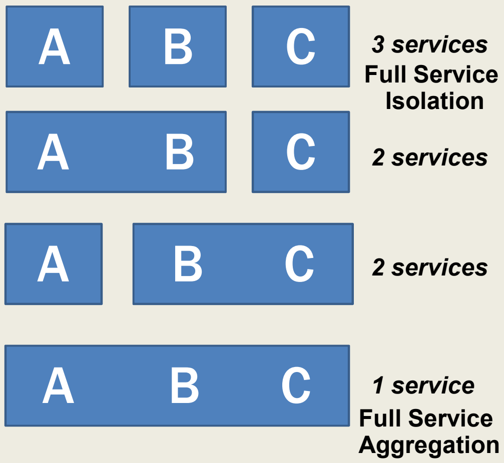
- Service 3: **EXTRACT**

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

82

SERVICE COMPOSITION

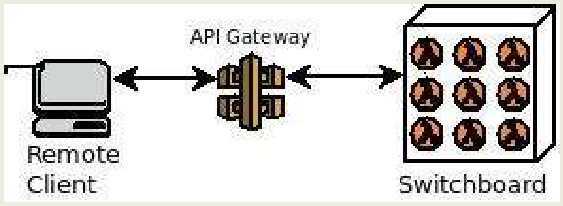




Other possible compositions: group by library, functional cohesion, etc.

83

SWITCH-BOARD ARCHITECTURE



1 service

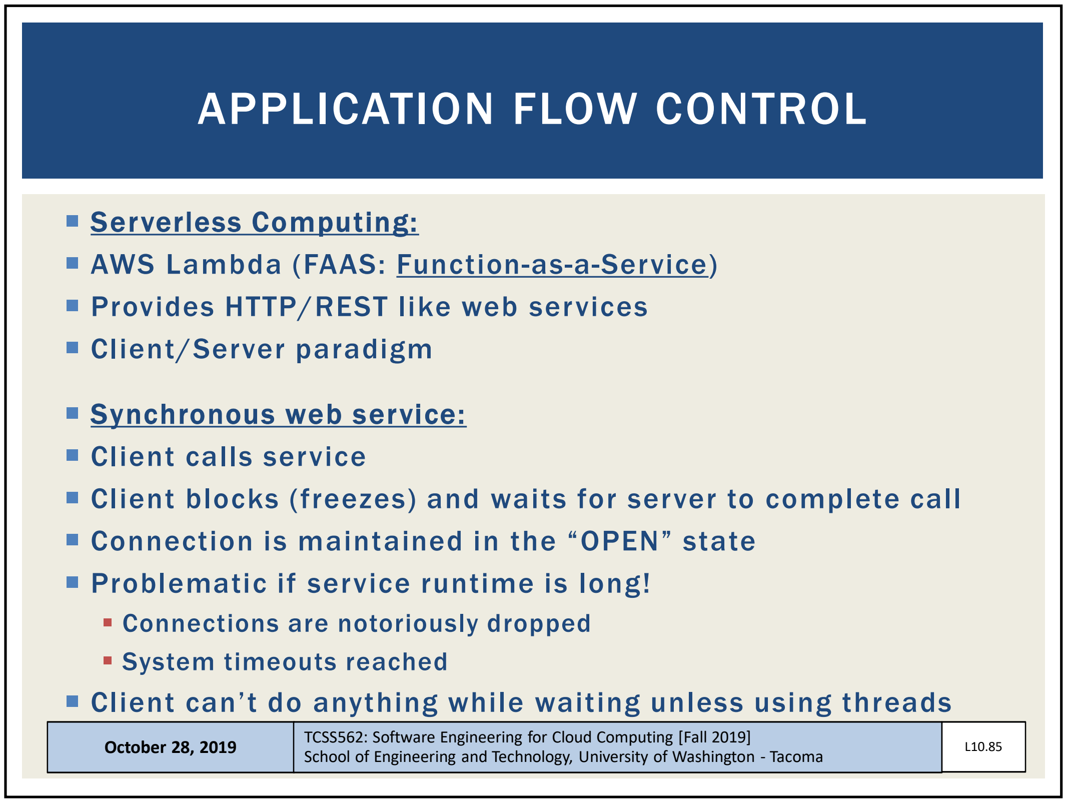
Single deployment package with consolidated codebase (Java: one JAR file)

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

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

Goal: reduce # of COLD starts to improve performance

84



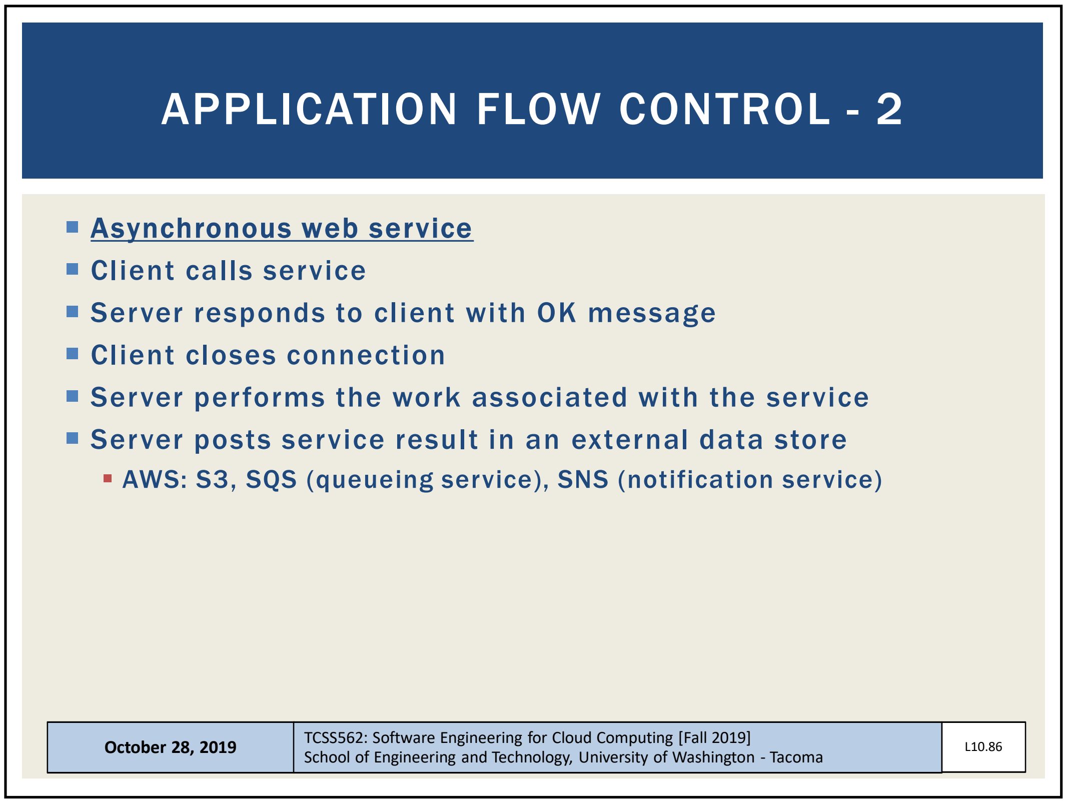
APPLICATION FLOW CONTROL

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

- **Synchronous web service:**
- Client calls service
- Client blocks (freezes) and waits for server to complete call
- Connection is maintained in the "OPEN" state
- Problematic if service runtime is long!
  - Connections are notoriously dropped
  - System timeouts reached
- Client can't do anything while waiting unless using threads

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.85

85



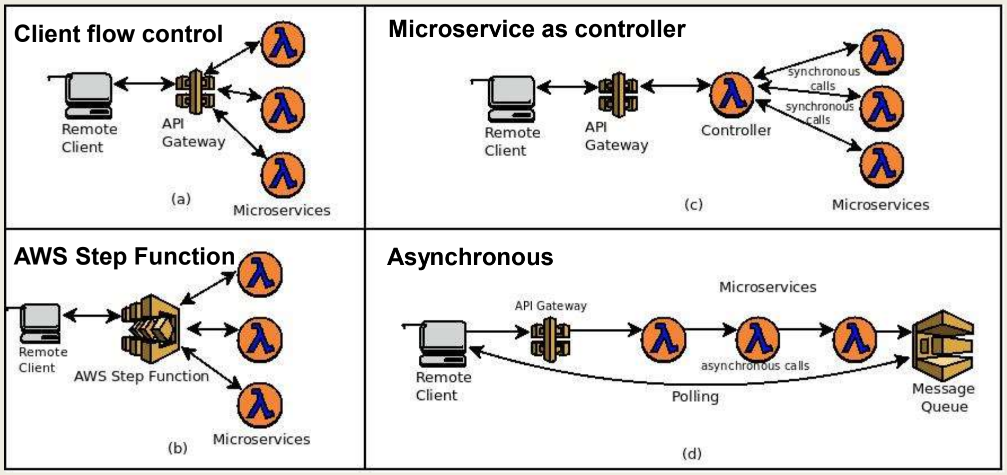
APPLICATION FLOW CONTROL - 2

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

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.86

86

APPLICATION FLOW CONTROL - 3

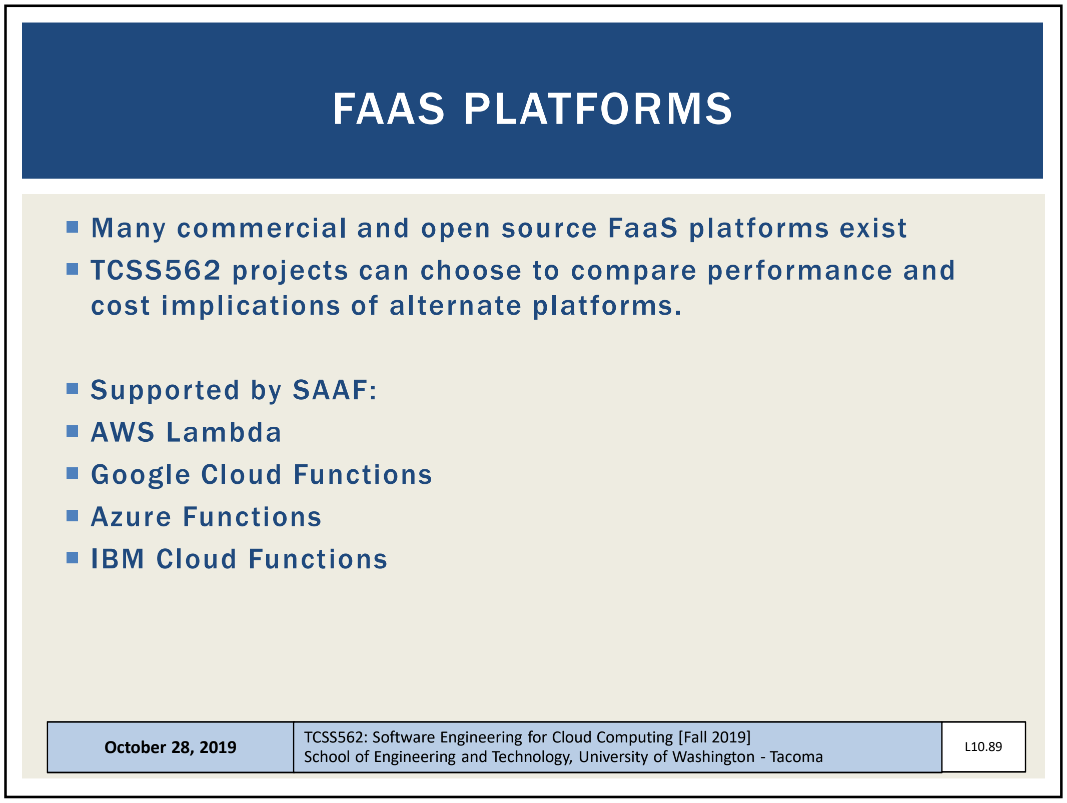


87

PROGRAMMING LANGUAGE

- Function-as-a-Service platforms support hosting services 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 any binary executable in any programming languages
- Jackson D, Clynch G. An Investigation of the Impact of Language Runtime on the Performance and Cost of Serverless Functions. In Proc. Of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion) 2018 Dec 17 (pp. 154-160).
- http://faculty.washington.edu/wlloyd/courses/tcss562/papers/AnInvestigationOfTheImpactOfLanguageRuntimeOnThePerformanceAndCostOfServerlessFunctions.pdf

88



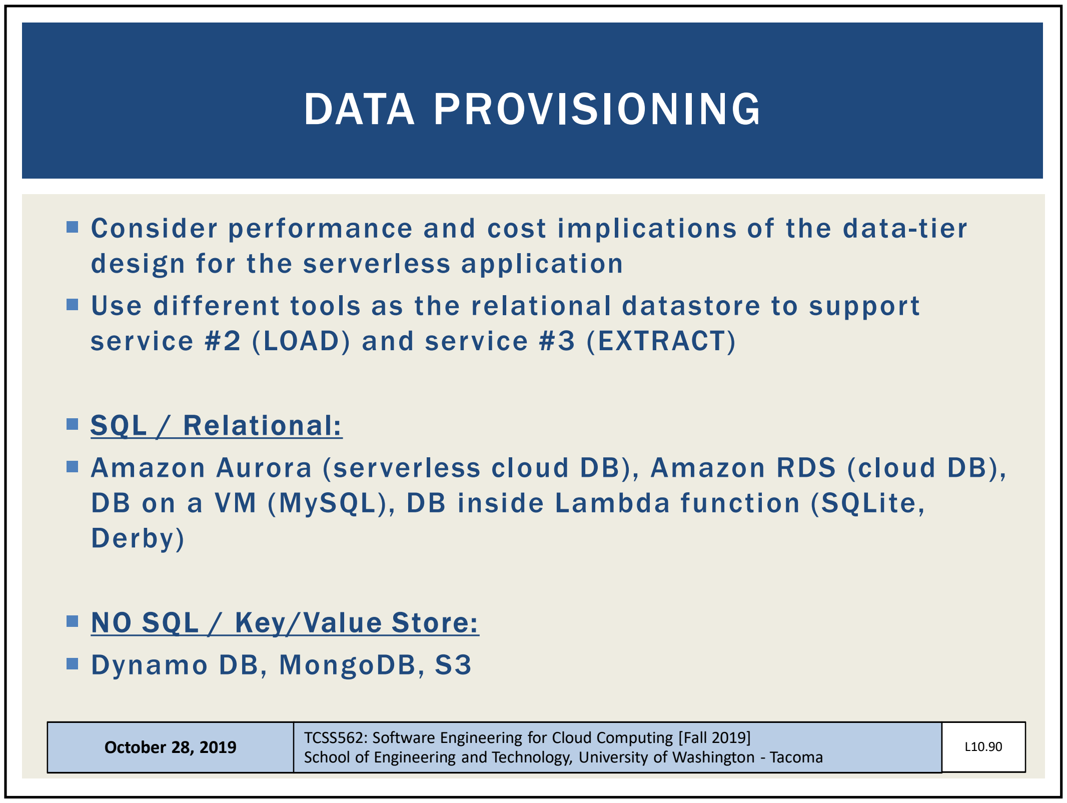
FAAS PLATFORMS

- Many commercial and open source FaaS platforms exist
- TCSS562 projects can choose to compare performance and cost implications of alternate platforms.

- Supported by SAAF:
- AWS Lambda
- Google Cloud Functions
- Azure Functions
- IBM Cloud Functions

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.89

89



DATA PROVISIONING

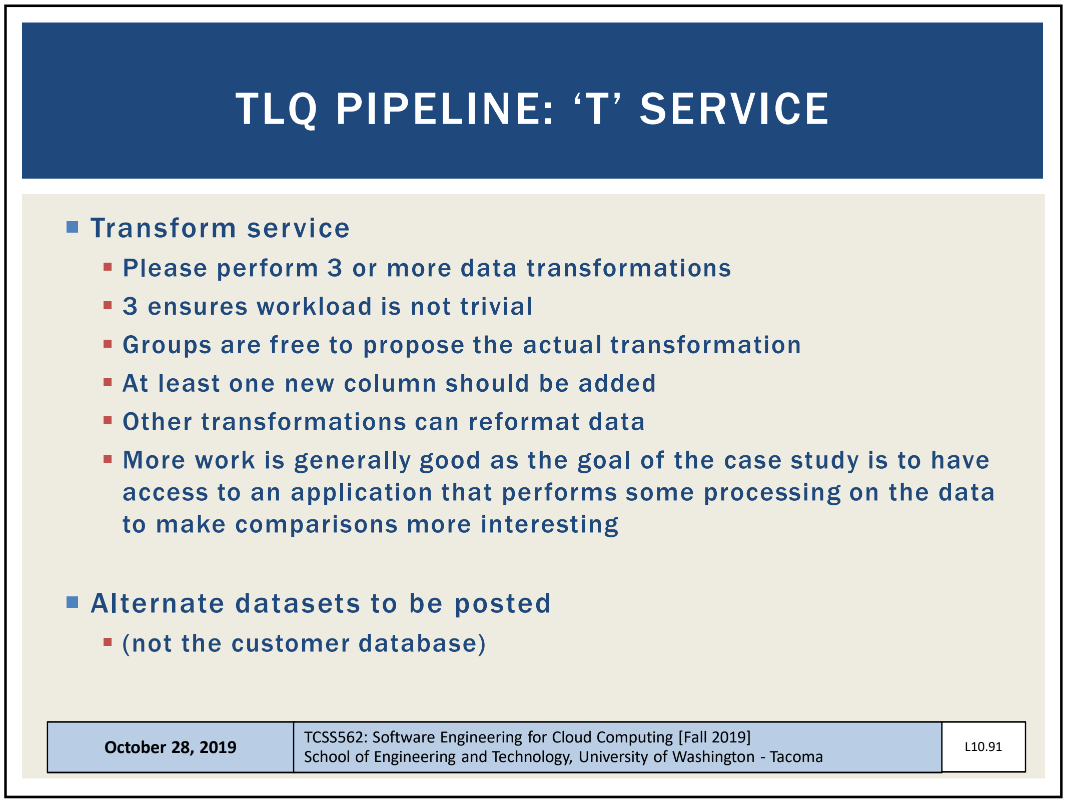
- Consider performance and cost implications of the data-tier design for the serverless application
- Use different tools as the relational datastore to support service #2 (LOAD) and service #3 (EXTRACT)

- <u>SQL / Relational:</u>
- Amazon Aurora (serverless cloud DB), Amazon RDS (cloud DB), DB on a VM (MySQL), DB inside Lambda function (SQLite, Derby)

- <u>NO SQL / Key/Value Store:</u>
- Dynamo DB, MongoDB, S3

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.90

90

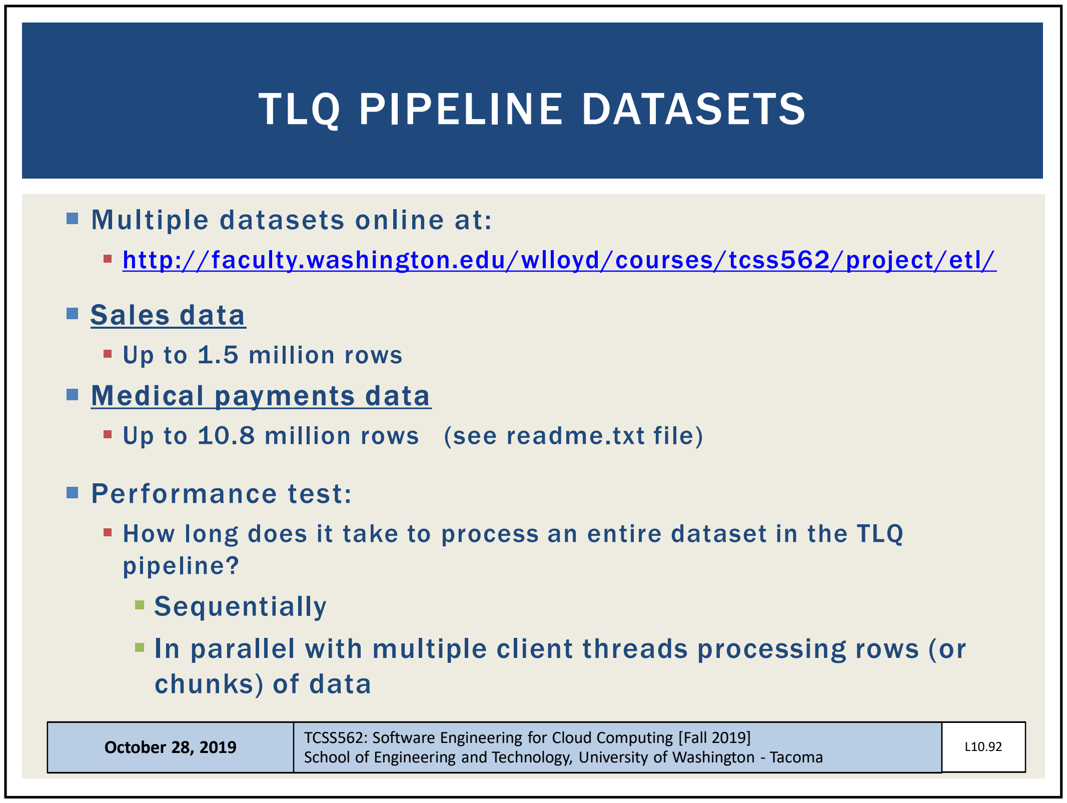


TLQ PIPELINE: 'T' SERVICE

- Transform service
  - Please perform 3 or more data transformations
  - 3 ensures workload is not trivial
  - Groups are free to propose the actual transformation
  - At least one new column should be added
  - Other transformations can reformat data
  - More work is generally good as the goal of the case study is to have access to an application that performs some processing on the data to make comparisons more interesting
- Alternate datasets to be posted
  - (not the customer database)

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.91

91

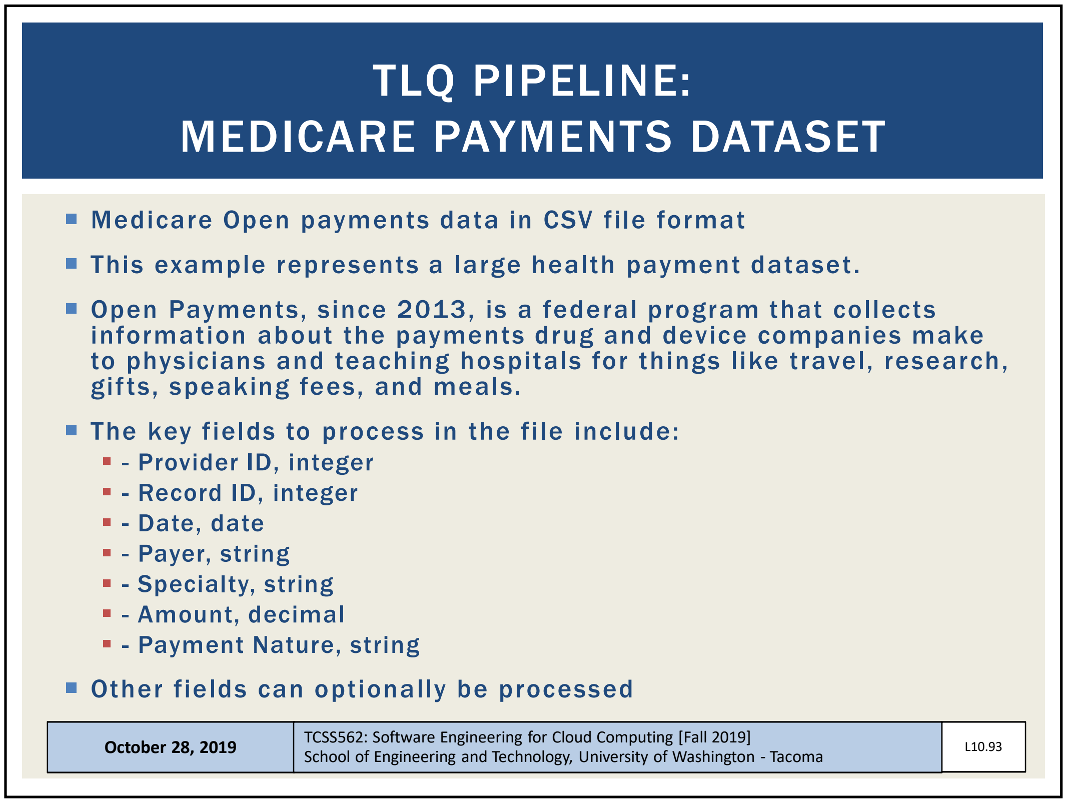


TLQ PIPELINE DATASETS

- Multiple datasets online at:
  - http://faculty.washington.edu/wlloyd/courses/tcss562/project/etl/
- Sales data
  - Up to 1.5 million rows
- Medical payments data
  - Up to 10.8 million rows (see readme.txt file)
- Performance test:
  - How long does it take to process an entire dataset in the TLQ pipeline?
    - Sequentially
    - In parallel with multiple client threads processing rows (or chunks) of data

October 28, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L10.92

92

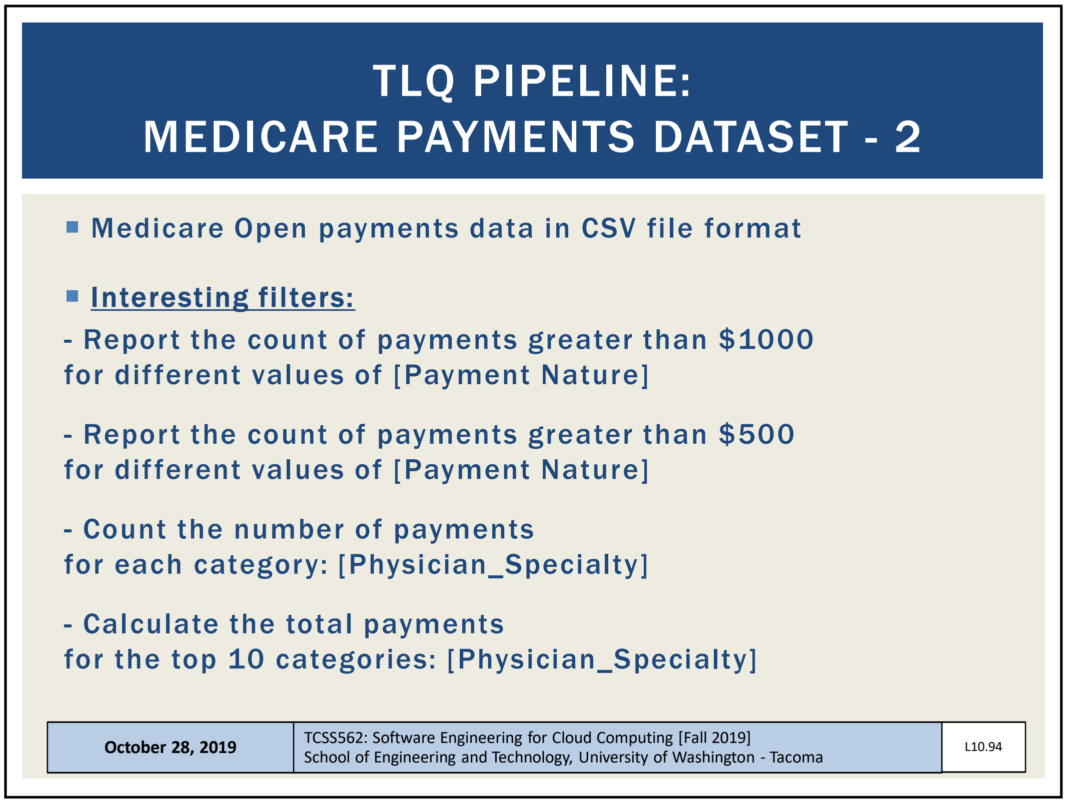


TLQ PIPELINE: MEDICARE PAYMENTS DATASET

- Medicare Open payments data in CSV file format
- This example represents a large health payment dataset.
- Open Payments, since 2013, is a federal program that collects information about the payments drug and device companies make to physicians and teaching hospitals for things like travel, research, gifts, speaking fees, and meals.
- The key fields to process in the file include:
  - - Provider ID, integer
  - - Record ID, integer
  - - Date, date
  - - Payer, string
  - - Specialty, string
  - - Amount, decimal
  - - Payment Nature, string
- Other fields can optionally be processed

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.93

93



TLQ PIPELINE: MEDICARE PAYMENTS DATASET - 2

- Medicare Open payments data in CSV file format
- <u>**Interesting filters:**</u>

- Report the count of payments greater than $1000
for different values of [Payment Nature]

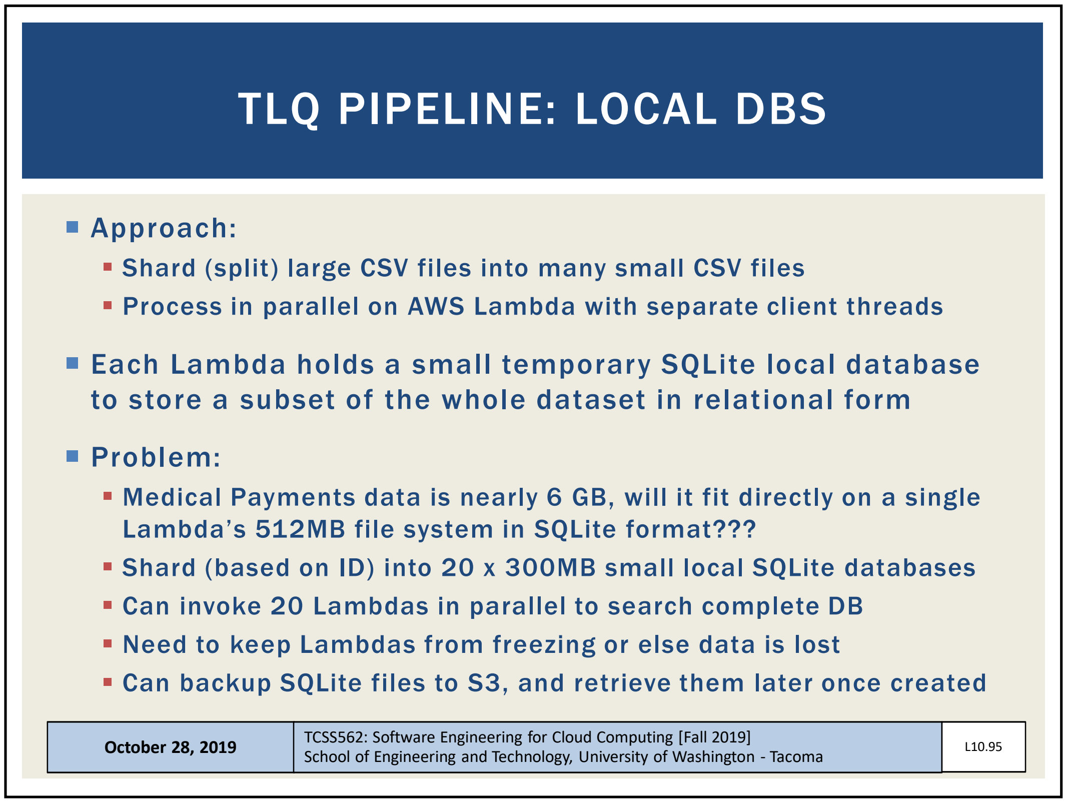
- Report the count of payments greater than $500
for different values of [Payment Nature]

- Count the number of payments
for each category: [Physician_Specialty]

- Calculate the total payments
for the top 10 categories: [Physician_Specialty]

October 28, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L10.94

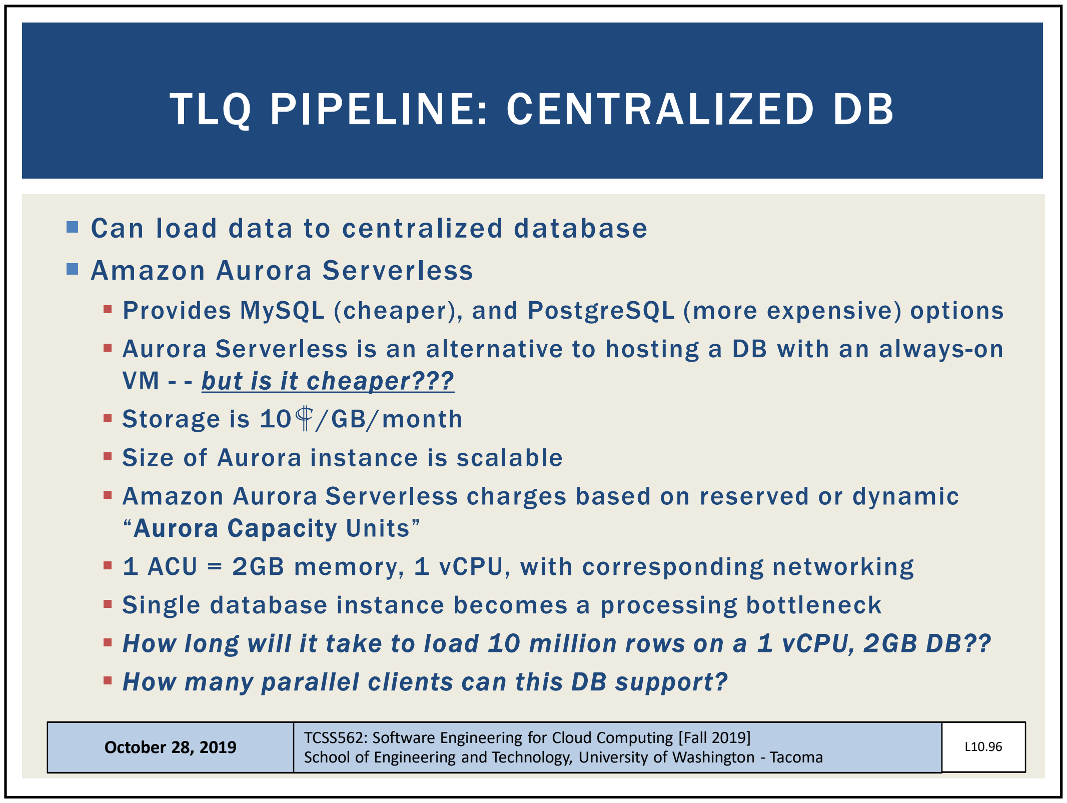
94



TLQ PIPELINE: LOCAL DBS

- Approach:
  - Shard (split) large CSV files into many small CSV files
  - Process in parallel on AWS Lambda with separate client threads
- Each Lambda holds a small temporary SQLite local database to store a subset of the whole dataset in relational form
- Problem:
  - Medical Payments data is nearly 6 GB, will it fit directly on a single Lambda's 512MB file system in SQLite format???
  - Shard (based on ID) into 20 x 300MB small local SQLite databases
  - Can invoke 20 Lambdas in parallel to search complete DB
  - Need to keep Lambdas from freezing or else data is lost
  - Can backup SQLite files to S3, and retrieve them later once created

95



TLQ PIPELINE: CENTRALIZED DB

- Can load data to centralized database
- Amazon Aurora Serverless
  - Provides MySQL (cheaper), and PostgreSQL (more expensive) options
  - Aurora Serverless is an alternative to hosting a DB with an always-on VM - - ***<u>but is it cheaper???</u>***
  - Storage is 10¢/GB/month
  - Size of Aurora instance is scalable
  - Amazon Aurora Serverless charges based on reserved or dynamic "Aurora Capacity Units"
  - 1 ACU = 2GB memory, 1 vCPU, with corresponding networking
  - Single database instance becomes a processing bottleneck
  - ***How long will it take to load 10 million rows on a 1 vCPU, 2GB DB??***
  - ***How many parallel clients can this DB support?***

96