TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

AWS OVERVIEW, Cloud Enabling Technology GraphQL

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

1

OFFICE HOURS – FALL 2025

- **Thursdays:**
  - 6:00 to 7:00 pm - CP 229 & Zoom
- **Friday – *** THIS WEEK *****
  - 11:00 am to 12:00 pm – ONLINE via Zoom
- Or email for appointment

- *Office Hours set based on Student Demographics survey feedback*
- ** - Friday office hours may be adjusted or canceled due meeting conflicts or other obligations. Adjustments will be announced via Canvas.*

2

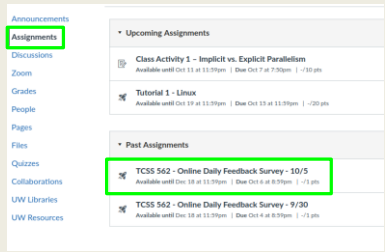
OBJECTIVES – 11/18

- **Questions from 11/13**
- Tutorials Questions
- Class Presentations: Cloud Technology or Research Paper Review
- AWS OVERVIEW - wrap up
- Ch. 5: Cloud Enabling Technology
- GraphQL
- Tutorial 6 Demo

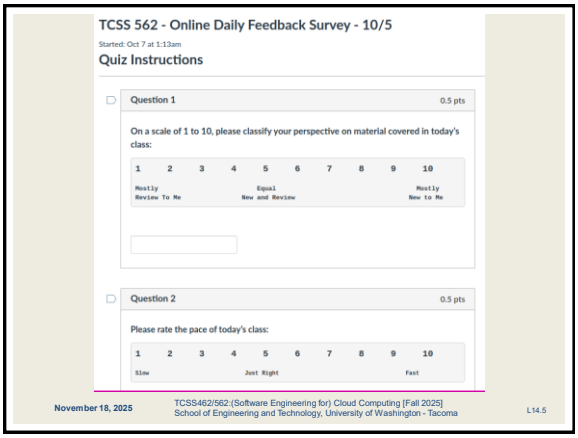
3

ONLINE DAILY FEEDBACK SURVEY

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



4



5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (41 respondents, 27 in-person, 14 online):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 5.46 (↓ - *previous* 5.64)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.07 (↑ - previous 4.96)**

6

FEEDBACK FROM 11/13

- ***Nitro does need a kernel? Is it like a real PC running on real hardware even though it's a VM ?***
- Nitro VM's still have an operating system kernel
- They just don't use an **amazon-kernel-image** (**aki**) file
  - Aki images are legacy
- With AWS nitro (and also XEN in hvm mode), the Linux kernel is on the root filesystem under the /boot directory

7

HEY, CHAT-GPT: ""ON AMAZON EC2 WHAT IS AN AKI?"

- An AKI is a bootable Linux kernel image used by older-generation EC2 instance types based on the paravirtual (PV) virtualization model. When launching a PV instance, you had to specify:
  - AMI – Amazon Machine Image (the root filesystem)
  - AKI – Amazon Kernel Image (the kernel the instance boots)
  - ARI – Amazon Ramdisk Image (optional initial ramdisk)
- **Why AKIs existed?**
  - Earlier EC2 instances (pre-XEN HVM virtualization) could not boot their own kernels directly. Instead, AWS provided a set of Amazon-managed kernels, each published as an AKI ID (e.g., aki-88aa75e1). The instance would boot using the AKI rather than kernel stored inside the AMI.
- **Are AKIs still relevant?**
  - Mostly no:
  - Modern EC2 instances use HVM (hardware virtualization).
  - HVM instances boot their own embedded kernel from the AMI, so AKIs and ARIs are unnecessary.
  - AWS has deprecated PV instances; AKI/ARI are rarely used today except for legacy workloads.

8

FEEDBACK - 2

- ***Quiz 2 only covers lecture 12-17 ?***
- ***Can you provide several Qs related to critical sections that may appear in quiz 2?***
- After each lecture, will repost slides:
  - For L13-17: will remove slides we did not cover that day
  - Will mark key slides with important content for quiz 2 with a green star:

9

OBJECTIVES – 11/18

- Questions from 11/13
- **Tutorials Questions**
- Class Presentations:
  Cloud Technology or Research Paper Review
- AWS OVERVIEW - wrap up
- Ch. 5: Cloud Enabling Technology
- GraphQL
- Tutorial 6 Demo

10

Don't Forget to Terminate (Shutdown) all EC2 instances for Tutorials 3

Spot instances:
c5d.large instance @ ~3.2 cents / hour

$0.78 / day
$5.48 / week
$23.78 / month
$285.42 / year

AWS CREDITS → → → → → → → →

11

TUTORIAL 5 – DUE NOV 16

- Introduction to Lambda II: Working with Files in S3, Cloud Trail, and Amazon Event Bridge Rules
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2025_tutorial_5.pdf
- Customize the Request object (add getters/setters)
  - Why do this instead of HashMap ?
- Import dependencies (jar files) into project for AWS S3
- Create an S3 Bucket
- Give your Lambda function(s) permission to work with S3
- Write to the CloudWatch logs
- Use of CloudTrail to generate S3 events
- Creating Event Bridge rule to capture events from CloudTrail
- Have the Event Bridge rule trigger a Lambda function with a static JSON input object (hard-coded filename)
- **Optional**: for the S3 PutObject event, dynamically extract the name of the file put to the S3 bucket for processing

12

TUTORIAL 6 – NOV 23

- Introduction to Lambda III: Serverless Databases
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2025_tutorial_6.pdf
- Create and use Sqlite databases using sqlite3
- Deploy Lambda function with Sqlite3 database under /tmp
- Compare in-memory vs. file-based Sqlite DBs on Lambda
- Create an Amazon Aurora "Serverless" v2 MySQL database
- Using the AWS CloudShell in the same VPC (Region + availability zone) connect and interact your Aurora serverless database using the mysql CLI app
- Deploy an AWS Lambda function that uses the MySQL "serverless" database

13

TUTORIAL 7 – DEC 4

- Introduction to Docker
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2025_tutorial_7.pdf
- Must complete using c7i-flex.large ec2 instance & Ubuntu 24.04 (for cgroups v2)
- Use DOCX file for copying and pasting Docker install commands
- Topics:
  - Installing Docker
  - Creating a container using a Dockerfile
  - Using cgroups virtual filesystem to monitor CPU utilization of a container
  - Persisting container images to Docker Hub image repository
  - Container vertical scaling of CPU/memory resources
  - Testing container CPU and memory isolation

14

TUTORIAL COVERAGE

- **Docker CLI → Docker Engine (dockerd) → containerd → runc**
- Working with the docker CLI:
- docker run create a container
- docker ps -a list containers, find CONTAINER ID
- docker exec --it run a process in an existing container
- docker stop stop a container
- docker kill kill a container
- docker help list available commands
- man docker Docker Linux manual pages

15

```
Commands:
attach      Attach local standard input, output, and error streams to a running container
build       Build an image from a Dockerfile
commit      Create a new image from a container's changes
cp          Copy files/folders between a container and the local filesystem
create      Create a new container
deploy      Deploy a new stack or update an existing stack
diff        Inspect changes to files or directories on a container's filesystem
events      Get real time events from the server
exec        Run a command in a running container
export      Export a container's filesystem as a tar archive
history     Show the history of an image
images      List images
import      Import the contents from a tarball to create a filesystem image
info        Display system-wide information
inspect     Return low-level information on Docker objects
kill        Kill one or more running containers
load        Load an image from a tar archive or STDIN
login       Log in to a Docker registry
logout      Log out from a Docker registry
logs        Fetch the logs of a container
pause       Pause all processes within one or more containers
port        List port mappings or a specific mapping for the container
ps          List containers
pull        Pull an image or a repository from a registry
push        Push an image or a repository to a registry
rename      Rename a container
restart     Restart one or more containers
rm          Remove one or more containers
rmi         Remove one or more images
run         Run a command in a new container
save        Save one or more images to a tar archive (streamed to STDOUT by default)
search      Search the Docker Hub for images
start       Start one or more stopped containers
stats       Display a live stream of container(s) resource usage statistics
stop        Stop one or more running containers
tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top         Display the running processes of a container
unpause     Unpause all processes within one or more containers
update      Update configuration of one or more containers
version     Show the Docker version information
wait        Block until one or more containers stop, then print their exit codes
```

Docker CLI

16

TUTORIAL 7

- Tutorial introduces use of two common Linux performance benchmark applications
- stress-ng
- 100s of CPU, memory, disk, network stress tests
- Sysbench
- Used in tutorial for memory stress test

17

OBJECTIVES – 11/18

- Questions from 11/13
- Tutorials Questions
- **Class Presentations: Cloud Technology or Research Paper Review**
- AWS OVERVIEW - wrap up
- Ch. 5: Cloud Enabling Technology
- GraphQL
- Tutorial 6 Demo

18

GROUP PRESENTATION

- TWO OPTIONS:
- ***Cloud technology presentation***
- ***Cloud research paper presentation***
  - Recent & suggested papers will be posted at: http://faculty.washington.edu/wlloyd/courses/tcss562/papers/
- Submit presentation type and topics (paper or technology) with desired dates of presentation via Canvas by: ***Tuesday November 18th @ 11:59pm***
- Presentation dates:
  - Tuesday November 25
  - Tuesday December 2*, Thursday December 4
    - ** - day of quiz 2. only 1 presentation slot*

19

OBJECTIVES – 11/18

- Questions from 11/13
- Tutorials Questions
- Class Presentations: Cloud Technology or Research Paper Review
- **AWS OVERVIEW - wrap up**
- Ch. 5: Cloud Enabling Technology
- GraphQL
- Tutorial 6 Demo

20

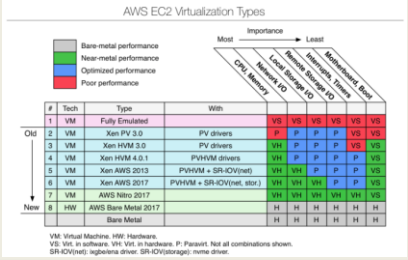
EC2 VIRTUALIZATION - NITRO

- 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
- There are 5 Nitro versions (v2 to v6)
- Article describes features and provides links to instance families (m, c, r, etc.) where it is possible to check the Nitro version for specific instance types to understand feature evolution
- https://docs.aws.amazon.com/ec2/latest/instancetypes/ec2-nitro-instances.html

21

EVOLUTION OF AWS VIRTUALIZATION

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



#	Tech	Type	With					
1	VM	Fully Emulated		VS	VS	VS	VS	VS
2	VM	Xen PV 3.0	PV drivers	P	P	P	VS	VS
3	VM	Xen HVM 3.0	PV drivers	VH	P	P	VS	VS
4	VM	Xen HVM 4.0.1	PVHVM drivers	VH	P	P	P	VS
5	VM	Xen AWS 2013	PVHVM + SR-IOV(net)	VH	VH	P	P	P
6	VM	Xen AWS 2017	PVHVM + SR-IOV(net, stor.)	VH	VH	VH	P	P
7	VM	AWS Nitro 2017		VH	VH	VH	VH	VS
8	HW	AWS Bare Metal 2017		H	H	H	H	H
		Bare Metal		H	H	H	H	H

22

INSTANCE ACTIONS ★

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

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

23

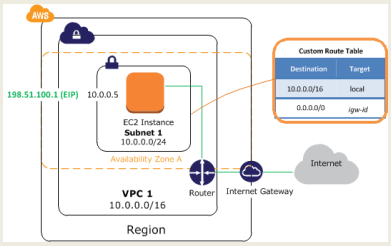
EC2 INSTANCE: NETWORK ACCESS ★

- 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

24

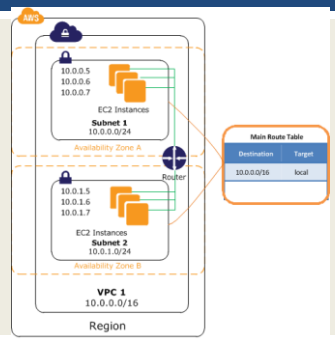
SIMPLE VPC

- Recommended when using Amazon EC2



25

VPC SPANNING AVAILABILITY ZONES ★



26

INSPECTING INSTANCE INFORMATION

- EC2 VMs run a local metadata service
- Can query instance metadata to self discover cloud config attributes
- **Version 2 (default) of the metadata service requires a token**
- Get Token:

```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api /token" -H
"X-aws-ec2-metadata-token-ttl-seconds: 21600"`
```

- Find your instance ID:

```
curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/

curl -H "X-aws-ec2-metadata-token: $TOKEN"
http://169.254.169.254/latest/

curl -H "X-aws-ec2-metadata-token: $TOKEN"
http://169.254.169.254/latest/meta-data/

curl -H "X-aws-ec2-metadata-token: $TOKEN"
http://169.254.169.254/latest/meta-data/instance-id ; echo
```

See: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-instance-metadata-service.html#instance-metadata-retrieval-examples

27

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, but now strong consistency
- Can store Lambda function state for life of container.

28

AWS CLI

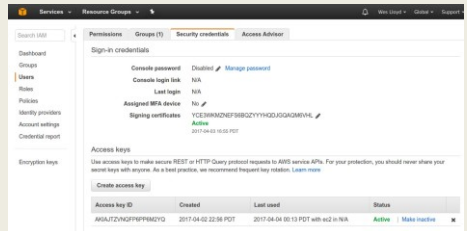
- Launch Ubuntu VM
  - Instances | Launch Instance
- Install the general AWS CLI
  - sudo apt install awscli
- Create config file

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


29

AWS CLI - 2

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



30

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

31

LEGACY / SERVICE SPECIFIC CLI(S)

- 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
- Some AWS services have separate CLI installable by package

32

AMI TOOLS

- Amazon Machine Images tools
- For working with disk volumes
- Can create live copies of any disk volume
  - Your local laptop, ec2 root volume (EBS), ec2 ephemeral disk
- Installation:
  https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ami-tools-commands.html
- AMI tools reference:
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ami-tools-commands.html
- Some functions may require private key & certificate files

33

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

34

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)**
- An alternative to bundling volumes and storing in S3 is to use a containerization tool such as Docker. . .
- Create image script . . .

35

SCRIPT: CREATE A NEW INSTANCE STORE IMAGE FROM LIVE DISK VOLUME

```
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
```


36

MAKE A DISK FROM AN IMAGE FILE

```
# ****************  ON THE LOCAL COMPUTER ********************
# create 1200 MB virtual disk = 1,258,291,200 bytes
sudo dd if=/dev/zero of=vhd.img bs=1M count=1200
# format the disk using the ext4 filesystem
sudo mkfs.ext4 vhd.img
# mount the disk at "/mnt"
sudo mount -t auto -o loop vhd.img /mnt
# check that the disk is mounted
df -h
# create a hello file (or copy data) to the new virtual disk
cd /mnt
sudo echo "hello world !" > hello.txt
ls -l
cd
# unmount the virtual disk
sudo umount /mnt
```


37

COMPRESS IMAGE, PUSH TO S3

```
# compress the disk
bzip2 vhd.img

# push the disk image to S3
aws s3 cp vhd.img.bz2 s3://tcss562-f21-images
```


38

RESTORE ON THE CLOUD

```
# ******************  ON THE AWS EC2 VM ********************
# with the awscli installed and configured

# download the image from S3
aws s3 cp s3://tcss562-f21-images/vhd.img.bz2 vhd.img.bz2

# uncompress the image
bzip2 -d vhd.img.bz2

# we need to calculate the number of sectors for the
partition
# disk sectors are 512 bytes each
# divide the disk size by 512 to determine sectors
# sectors = 1258291200 / 512 = 2459648

# create a disk partition for this disk that is
# 2459648 sectors in size using the ephemeral drive or
# a newly mounted EBS volume that is unformatted

sudo fdisk /dev/nvme1n1
```


39

PARTITION THE DISK

```
Welcome to fdisk (util-linux 2.34).

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)

Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-97656249, default 2048): 2048
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-97656249, default
97656249): 2459648

Created a new partition 1 of type 'Linux' and of size 1.2 GiB.

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 83
Changed type of partition 'Linux' to 'Linux'.

Command (m for help): w  (to write and exit)
```


40

COPY DATA TO NEW DISK PARTITION

```
# now check if the partition has been created.
# it should be listed as /dev/nvme1n1p1:
ls /dev/nvme1n1*

# now copy the data to the partition
sudo dd if=vhd.img of=/dev/nvme1n1p1

# mount the disk
sudo mount /dev/nvme1n1p1 /mnt

# and check if the hello file is there
cat /mnt/hello.txt

# we were able to copy the disk image to the cloud
# and we never had to format the cloud disk
# this examples copies a filesystem from a local disk
# to the cloud disk
```


41

FOR MORE INFORMATION

- Example script:
- https://faculty.washington.edu/wlloyd/courses/tcss562/examples/copy-disk-to-cloud.sh

- URLs:
- https://help.ubuntu.com/community/DriveImaging
- https://www.tecmint.com/create-virtual-harddisk-volume-in-linux/

42

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

43

OBJECTIVES – 11/18

- Questions from 11/13
- Tutorials Questions
- Class Presentations: Cloud Technology or Research Paper Review
- AWS OVERVIEW - wrap up
- **Ch. 5: Cloud Enabling Technology**
- Tutorial 5 Demo
- Tutorial 6 Demo

44

CLOUD ENABLING TECHNOLOGY

45

CLOUD ENABLING TECHNOLOGY

- *Adapted from Ch. 5 from Cloud Computing Concepts, Technology & Architecture*
- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

46

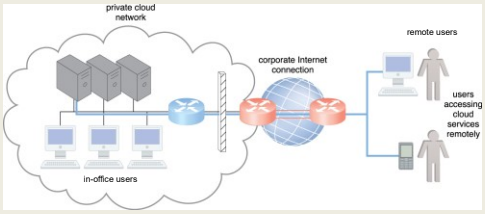
1. BROADBAND NETWORKS AND INTERNET ARCHITECTURE

- 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**

47

PRIVATE CLOUD NETWORKING

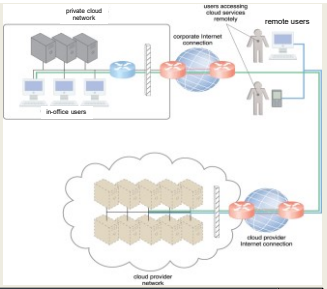
- For institutions with in-house private clouds



48

PUBLIC CLOUD NETWORKING

- Resources can be extended by adding public cloud
- Places further dependency on the internet to provide connectivity



49

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

50

CLOUD ENABLING TECHNOLOGY

- *Adapted from Ch. 5 from Cloud Computing Concepts, Technology & Architecture*
- Broadband networks and internet architecture
- Data center technology
- Virtualization technology
- Multitenant technology
- Web/web services technology

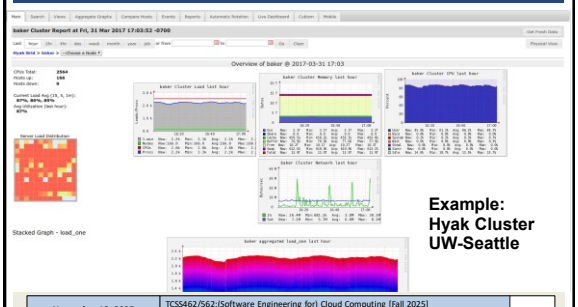
51

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: enable server provisioning, configuration, patching, monitoring without supervision… ***tool/API support is desirable***

52

CLUSTER MANAGEMENT TOOLS



53

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 HW
- **Secure design**: physical and logical access control
- **Servers**: rackmount, etc.
- **Storage**: hard disk arrays (RAID)
- storage area network (SAN): disk array w/ multiple servers (individual nodes w/ disks) and a dedicated network
- network attached storage (NAS): inexpensive single node with collection of disks, provides shared filesystems, for NFS, etc.
- **Network hardware**: backbone routers (WAN to LAN connectivity), firewalls, VPN gateways, managed switches/routers

54

CLOUD ENABLING TECHNOLOGY

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

55

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

56

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

57

KEY VIRTUALIZATION TRADEOFF

- Tradeoff space:

What is the "right" level of abstraction in the cloud for sharing resources with users?

Degree of Hardware Abstraction

Abstraction Concerns:
- **Overhead**
- **Performance**
- **Isolation**
- **Security**

58

ABSTRACTION CONCERNS

- **Overhead with too many instances w/ heavy abstractions**
  - Too many instances using a heavy abstraction can lead to hidden resource utilization and waste
  - Example: Dedicated server with 48 VMs each with separate instance of Ubuntu Linux
  - Idle VMs can reduce performance of co-resident jobs/tasks
- **"Virtualization" Overhead**
  - Cost of virtualization an OS instance
  - Overhead has dropped from ~100% to ~1% over last decade
- **Performance**
  - Impacted by weight of abstraction and virtualization overhead

59

ABSTRACTION CONCERNS - 2

- **Isolation**
  - From others:
    What user A does should not impact user B in any noticeable way
- **Security**
  - User A and user B's data should be always separate
  - User A's actions are not perceivable by User B

60

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS and Application Containers** – seen with CaaS
  - **OS Container** – replacement for VM, mimics full OS instance, heavier
  - OS containers run 100s of processes just like a VM
  - **App Container** – Docker: packages dependencies to easily transport and run an application anywhere
  - Application containers run only a few processes
- **Micro VMs** – FaaS / CaaS
  - Lighter weight alternative to full VM (KVM, XEN, VirtualBox)
  - Firecracker
- **Unikernel Operating Systems** – research mostly
  - Single process, multi-thread operating system
  - Designed for cloud, objective to reduce overhead of running too many OS instances

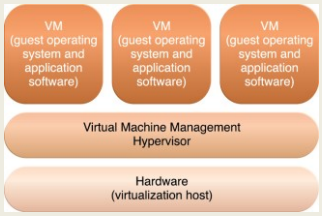
61

VIRTUAL MACHINES

- **Type 1 hypervisor**
  - Typically involves a special virtualization kernel that runs directly on the system to share the underlying machine with many guest VMs
  - Paravirtualization introduced to directly share system resources with guests bypassing full emulation
  - VM becomes equal participant in sharing the network card for example
- **Type 2 hypervisor**
  - Typically involves the **Full Virtualization** of the guest, where everything is simulated/emulated
- Hardware level support (i.e. features introduced on CPUs) have made virtualization faster in all respects shrinking virtualization overhead

62

TYPE 1 HYPERVISOR



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

63

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)

64

COMMON VMMS: PARAVIRTUALIZATION

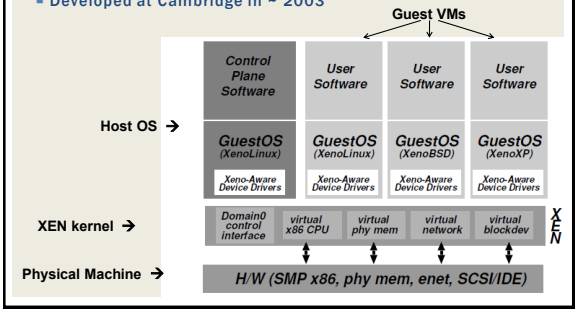
- **TYPE 1 Hypervisor**
- XEN
- Citrix Xen-server (a commercial version of XEN)
- VMWare ESXi
- KVM (virtualization support in kernel)

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

65

XEN

- Developed at Cambridge in ~ 2003



66

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
```


67

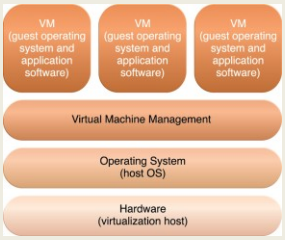
XEN - 3

- 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

68

TYPE 2 HYPERVISOR

- Adds additional layer



69

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

70

CHECK FOR VIRTUALIZATION SUPPORT

- See: https://cyberciti.biz/faq/linux-xen-vmware-kvm-intel-vt-amd-v-support
- # check for Intel VT CPU virtualization extensions on Linux
  `grep -color vmx /proc/cpuinfo`
- # check for AMD V CPU virtualization extensions on Linux
  `grep -color svm /proc/cpuinfo`
- Also see '`lscpu`' → "Virtualization:"
- Other Intel CPU features that help virtualization:
  `ept  vpid  tpr_shadow  flexpriority  vnmi`

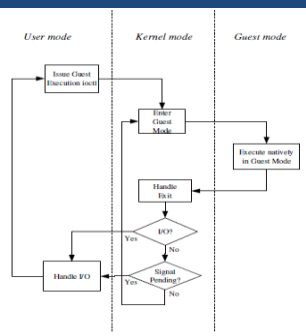
71

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

72

KVM – 2



73

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"

74

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

75

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

76

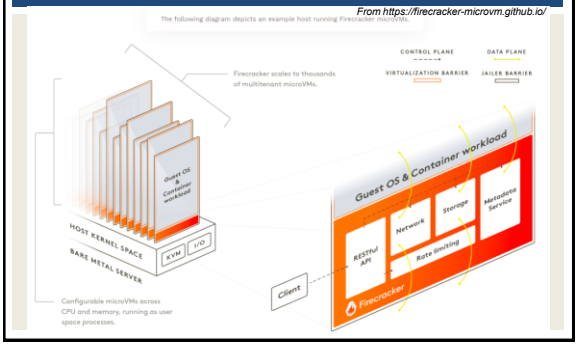
KVM ENHANCEMENTS

- 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

77

FIRECRACKER MICRO VM

From https://firecracker-microvm.github.io/



78

FIRECRACKER MICRO VM

- Provides a virtual machine monitor (VMM) (i.e. hypervisor) using KVM to create and manage microVMs
- Has a minimalist design with goals to improve security, decreases the startup time, and increases hardware utilization
- Excludes unnecessary devices and guest functionality to reduce memory footprint and attack surface area of each microVM
- Supports boot time of <125ms, <5 MiB memory footprint
- Can run 100s of microVMs on a host, launching up to 150/sec
- Is available on 64-bit Intel, AMD, and Arm CPUs
- Used to host AWS Lambda and AWS Fargate
- Has been open sourced under the Apache 2.0 license

79

FIRECRACKER - 2

- **Minimalistic**
- MicroVMs run as separate processes on the host
- Only 5 emulated devices are available: virtio-net, virtio-block, virtio-vsock, serial console, and a minimal keyboard controller used only to stop the microVM
- Rate limiters can be created and configured to provision resources to support bursts or specific bandwidth/operation limitations
- **Configuration**
- A RESTful API enables common actions such as configuring the number of vCPUs or launching microVMs
- A metadata service between the host and guest provides configuration information

80

FIRECRACKER - 3

- **Security**
- Runs in user space ***(not the root user)*** on top of the Linux Kernel-based Virtual Machine (KVM) hypervisor to create microVMs
- Lambda functions, Fargate containers, or container groups can be encapsulated using Firecracker through KVM, enabling workloads from different customers to run on the same machine, without sacrificing security or efficiency
- MicroVMs are further isolated with common Linux user-space security barriers using a companion program called "jailer" which provides a second line of defense if KVM is compromised

81

UNIKERNELS

- Lightweight alternative to containers and VMs
  - Custom Cloud Operating System
  - Single process, multiple threads, runs one program
  - Launch separately atop of hypervisor (XEN/KVM)
  - Reduce overhead, duplication of heavy weight OS
  - OSv is most well known unikernel
  - Several others exist has research projects
  - More information at: http://unikernel.org/
  - Google Trends OSv →

82

QUESTIONS

111