TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Enabling Technology

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

TR 5:00-7:00 PM

1

OFFICE HOURS – FALL 2021

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

> Office Hours set based on Student Demographics survey feedback

2

OBJECTIVES – 11/9

- Questions from 11/4
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Tutorial 4 – Intro to FaaS – AWS Lambda
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Term Project Proposals – please update if requested
- Quiz 1 on Canvas available: Nov 10th 8am - Nov 14th 11:59pm

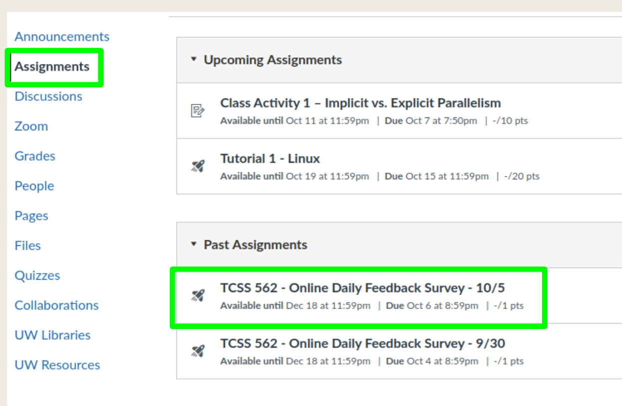
- Ch. 5: Cloud Enabling Technology
- Team planning

November 9, 2021 | TCSS562:Software Engineering for Cloud Computing [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma | L12.3

3

ONLINE DAILY FEEDBACK SURVEY

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



November 9, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma | L12.4

4

TCSS 562 - Online Daily Feedback Survey - 10/5

Started: Oct 7 at 1:13am

Quiz Instructions

Question 1 0.5 pts

On a scale of 1 to 10, please classify your perspective on material covered in today's class:

1	2	3	4	5	6	7	8	9	10
Mostly Review To Me				Equal New and Review					Mostly New to Me

Question 2 0.5 pts

Please rate the pace of today's class:

1	2	3	4	5	6	7	8	9	10
Slow				Just Right				Fast	

5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (28 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.18 (↑ - *previous 6.04*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.54 (↑ - previous 5.20)**

6

FEEDBACK FROM 11/4

7

OBJECTIVES – 11/9

- Questions from 11/4
- **Tutorial 6 - Intro to FaaS III - Serverless Databases**
- Tutorial 4 – Intro to FaaS – AWS Lambda
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Term Project Proposals – please update if requested
- Quiz 1 on Canvas available: Nov 10th 8am - Nov 14th 11:59pm

- Ch. 5: Cloud Enabling Technology
- Team planning

8

OBJECTIVES – 11/9

- Questions from 11/4
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- **Tutorial 4 – Intro to FaaS – AWS Lambda**
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Term Project Proposals – please update if requested
- Quiz 1 on Canvas available: Nov 10^{th} 8am - Nov 14^{th} 11:59pm

- Ch. 5: Cloud Enabling Technology
- Team planning

November 9, 2021	TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.9

9

OBJECTIVES – 11/9

- Questions from 11/4
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Tutorial 4 – Intro to FaaS – AWS Lambda
- **Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch**
- Term Project Proposals – please update if requested
- Quiz 1 on Canvas available: Nov 10^{th} 8am - Nov 14^{th} 11:59pm

- Ch. 5: Cloud Enabling Technology
- Team planning

November 9, 2021	TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.10

10

OBJECTIVES – 11/9

- Questions from 11/4
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Tutorial 4 – Intro to FaaS – AWS Lambda
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- **Term Project Proposals – please update if requested**
- Quiz 1 on Canvas available: Nov 10th 8am - Nov 14th 11:59pm
- Ch. 5: Cloud Enabling Technology
- Team planning

November 9, 2021	TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.11

11

OBJECTIVES – 11/9

- Questions from 11/4
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Tutorial 4 – Intro to FaaS – AWS Lambda
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Term Project Proposals – please update if requested
- **Quiz 1 on Canvas available: Nov 10th 8am - Nov 14th 11:59pm**
- Ch. 5: Cloud Enabling Technology
- Team planning

November 9, 2021	TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.12

12

OBJECTIVES – 11/9

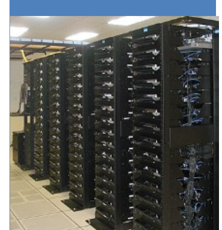
- Questions from 11/4
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Tutorial 4 – Intro to FaaS – AWS Lambda
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Term Project Proposals – please update if requested
- Quiz 1 on Canvas available: Nov 10th 8am - Nov 14th 11:59pm

- Ch. 5: Cloud Enabling Technology
- Team planning

November 9, 2021 | TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L12.13

13

CLOUD ENABLING TECHNOLOGY



November 9, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L12.14

14

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

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.15

15

1. BROADBAND NETWORKS AND INTERNET ARCHITECTURE

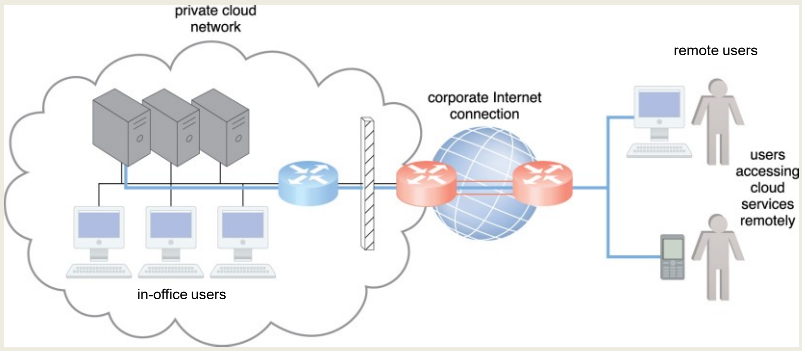
- Clouds must be connected to a network
- Inter-networking: Users' network must connect to cloud's network
- Public cloud computing relies heavily on the **internet**

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.16

16

PRIVATE CLOUD NETWORKING

- For institutions with in-house private clouds

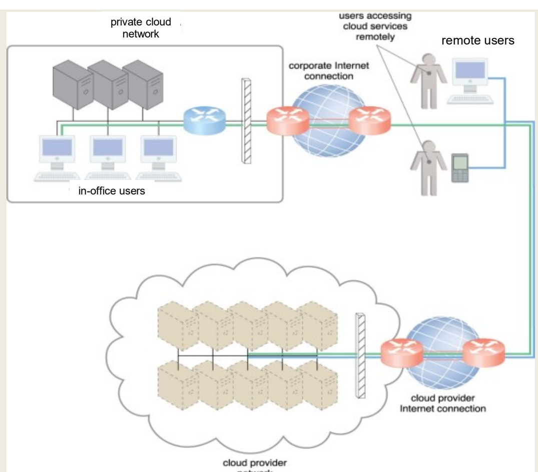


November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.17

17

PUBLIC CLOUD NETWORKING

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



November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.18

18

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

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

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***

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.21

21

CLUSTER MANAGEMENT TOOLS

Example: Hyak Cluster UW-Seattle

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.22

22

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

23

CLOUD ENABLING TECHNOLOGY

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

24

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

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

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**

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.27

27

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

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.28

28

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

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

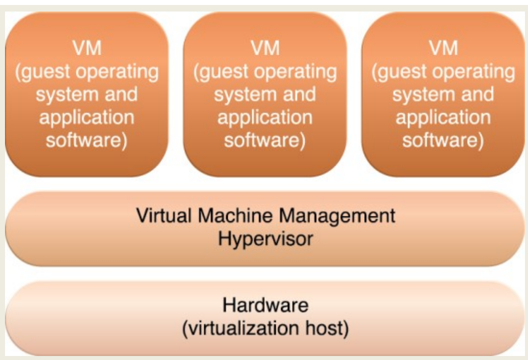
VIRTUAL MACHINES

- **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

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.31

31

TYPE 1 HYPERVISOR



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

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.32

32

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)

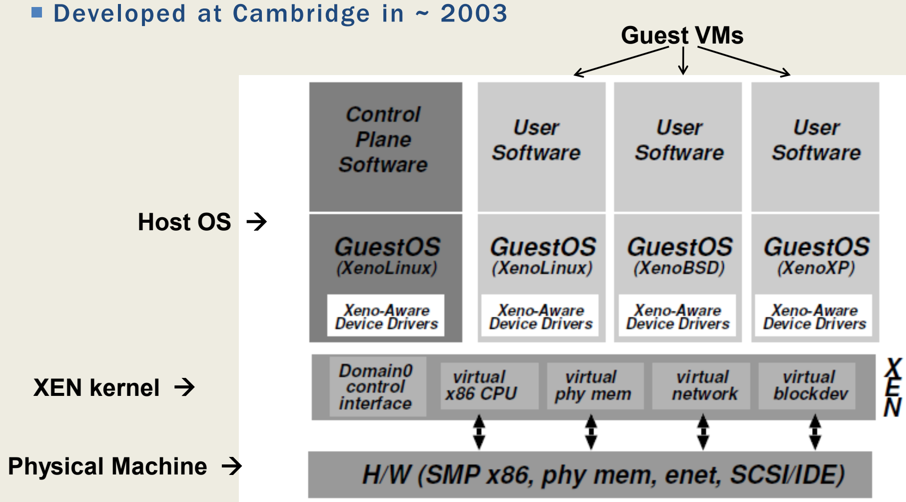
COMMON VMMS: PARAVIRTUALIZATION

- **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

XEN

- Developed at Cambridge in ~ 2003



35

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
```


36

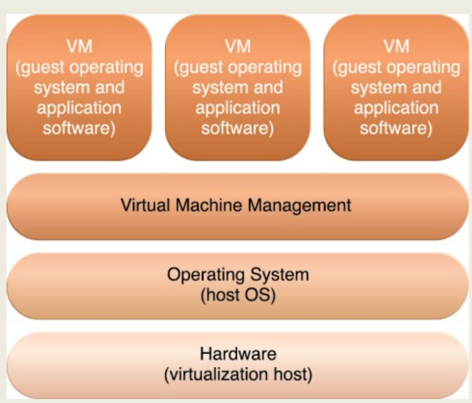
XEN - 3

- 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

37

TYPE 2 HYPERVISOR

- Adds additional layer



38

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

CHECK FOR VIRTUALIZATION SUPPORT

- See: https://cyberciti.biz/faq/linux-xen-vmware-kvm-intel-vt-amd-v-support
- # check for Intl 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`

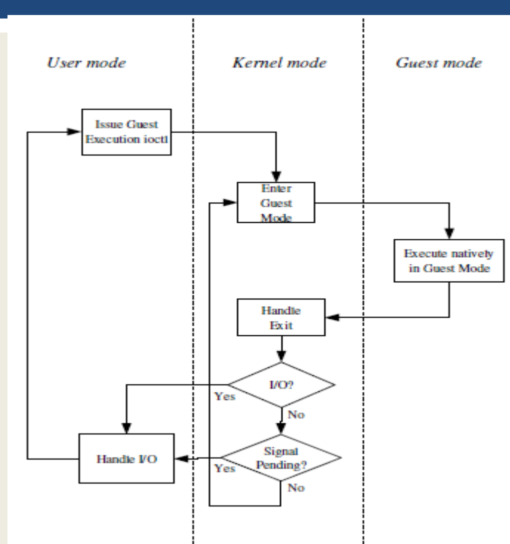
KERNEL BASED VIRTUAL MACHINES (KVM)

- 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

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.41

41

KVM – 2



42

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"

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

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

KVM ENHANCEMENTS

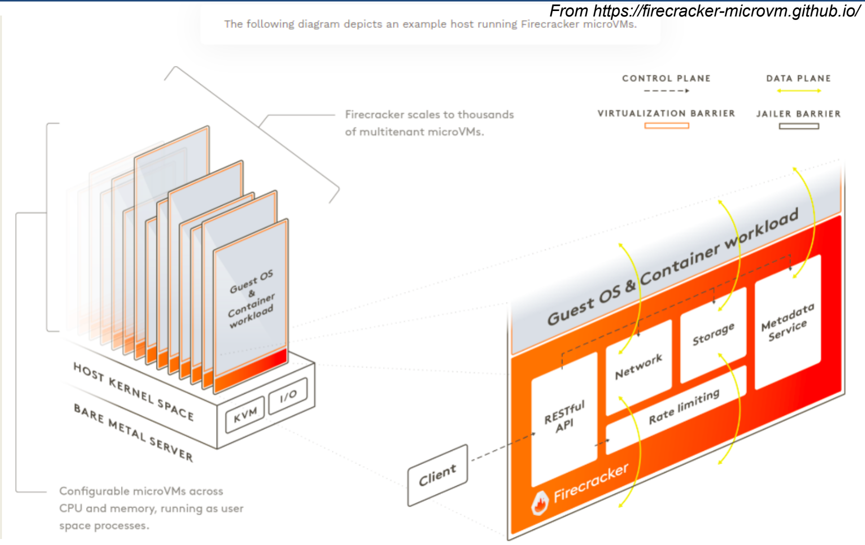
- Paravirtualized device drivers
  - Virtio
- Guest Symmetric Multiprocessor (SMP) support
  - Leverages multiple on-board CPUs
  - Supported as of Linux 2.6.23
- VM Live Migration
- Linux scheduler integration
  - Optimize scheduler with knowledge that KVM processes are virtual machines

WE WILL RETURN AT ~6:10 PM

47

FIRECRACKER MICRO VM

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



48

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

49

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

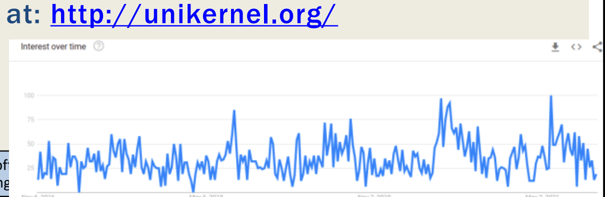
50

FIRECRACKER - 2

- **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

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 →



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

53

VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

- Middleware to manage virtual machines and infrastructure of IaaS "clouds"

- Examples
  - OpenNebula
  - Nimbus
  - Eucalyptus
  - OpenStack

54

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

55

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

56

VIM FEATURES - 3

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

57

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

58

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)

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

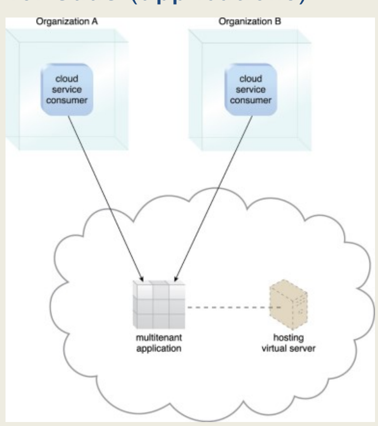
4. MULTITENANT APPLICATIONS

- 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

MULTITENANT APPS - 2

- Forms the basis for SaaS (applications)



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

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.63

63

5. 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

November 9, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L12.64

64

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, …

```
// 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>
```


67

```
// 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>
```


68

```
// 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>
```


69

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
  }
  ]
}
```


70

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

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

QUESTIONS-