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

RESEARCH SEMINAR TALKS

- In TCSS 598
- **Bioinformatics - Kayee Yeung**
- Wednesday November 17 – 3:00pm
- https://washington.zoom.us/j/93994539232

- **Cloud Computing – Wes Lloyd**
- Wednesday November 17 – 4:10 pm
- https://washington.zoom.us/j/93994539232

- Please join to learn more about potential MSCSS Capstone / Thesis projects

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.2

2

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

<u>Spot instances:</u>

c5d.large instance @ ~2 cents / hour

$0.48 / day
$3.36 / week
$14.60 / month
$175.20 / year

AWS CREDITS → → → → → → → →

3

OBJECTIVES – 11/16

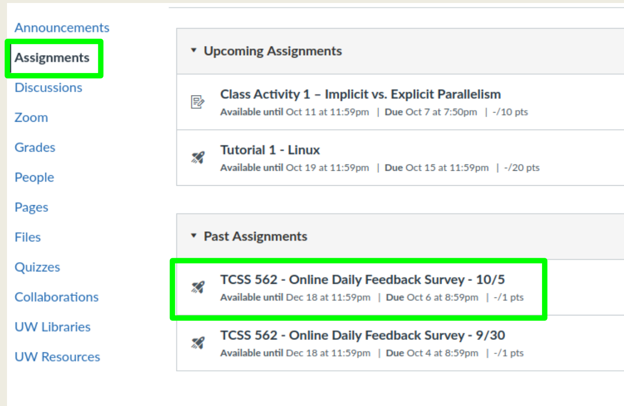
- Questions from 11/9
- Term Project Proposals – update by 11/19
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Quiz 1
- <u>Group Presentation Overview:</u>
  Cloud Technology or Research Paper for 11/30 – 12/9
- Term Project Check-in – due Wed 12/1 @ 11:59p
- Ch. 5: Cloud Enabling Technology
- Team planning

November 16, 2021	TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.4

4

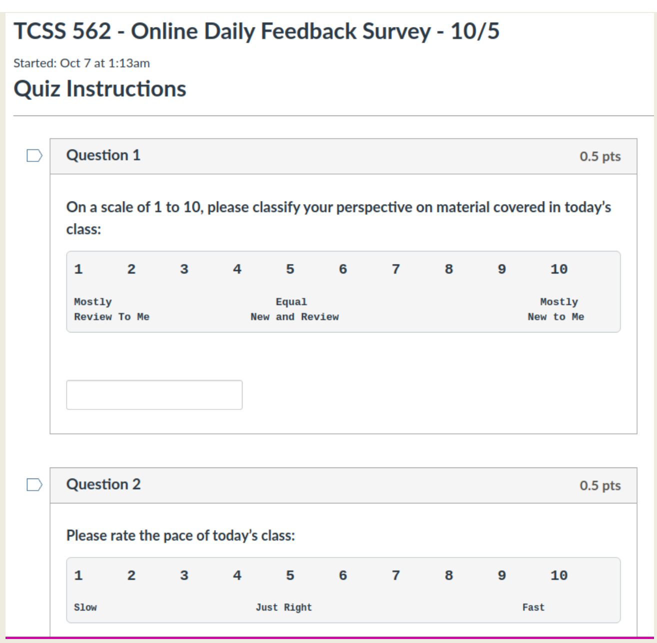
ONLINE DAILY FEEDBACK SURVEY

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



November 16, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L13.5

5



November 16, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L13.6

6

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (25 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.32 (↑ - *previous 6.18*)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.60 (↑ - previous 5.54)**

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.7

7

FEEDBACK FROM 11/9

- ***I noticed that the content of the class (lecture) doesn't correspond to HW tutorials. This makes it hard to complete tutorials.***
- We did complete most of Tutorial 4 in class.
- In addition, the AWS review discussed various aspects relating to EC2 and EBS (Tutorial 3)
- ***Also, sometimes tutorial doesn't explain steps well, leading to numerous questions and doubts.***
- Please do ask questions by: email, canvas message, Zoom chat, Slack channel, or verbally during lecture/office hours
- Tutorials are living documents - there is always potential for improvement with your feedback !

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.8

8

OBJECTIVES – 11/16

- Questions from 11/9
- **Term Project Proposals – update by 11/19**
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Quiz 1
- **Group Presentation Overview:**
  Cloud Technology or Research Paper for 11/30 – 12/9
- Term Project Check-in – due Wed 12/1 @ 11:59p

- Ch. 5: Cloud Enabling Technology
- Team planning

9

OBJECTIVES – 11/16

- Questions from 11/9
- Term Project Proposals – update by 11/19
- **Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch**
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Quiz 1
- **Group Presentation Overview:**
  Cloud Technology or Research Paper for 11/30 – 12/9
- Term Project Check-in – due Wed 12/1 @ 11:59p

- Ch. 5: Cloud Enabling Technology
- Team planning

10

OBJECTIVES – 11/16

- Questions from 11/9
- Term Project Proposals – update by 11/19
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- **Tutorial 6 - Intro to FaaS III - Serverless Databases**
- Quiz 1
- **Group Presentation Overview:**
  Cloud Technology or Research Paper for 11/30 – 12/9
- Term Project Check-in – due Wed 12/1 @ 11:59p

- Ch. 5: Cloud Enabling Technology
- Team planning

11

OBJECTIVES – 11/16

- Questions from 11/9
- Term Project Proposals – update by 11/19
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- **Quiz 1**
- **Group Presentation Overview:**
  Cloud Technology or Research Paper for 11/30 – 12/9
- Term Project Check-in – due Wed 12/1 @ 11:59p

- Ch. 5: Cloud Enabling Technology
- Team planning

12

OBJECTIVES – 11/16

- Questions from 11/9
- Term Project Proposals – update by 11/19
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Quiz 1
- **Group Presentation Overview:**
  Cloud Technology or Research Paper for 11/30 – 12/9
- Term Project Check-in – due Wed 12/1 @ 11:59p

- Ch. 5: Cloud Enabling Technology
- Team planning

November 16, 2021 | TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L13.13

13

OBJECTIVES – 11/16

- Questions from 11/9
- Term Project Proposals – update by 11/19
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Quiz 1
- Group Presentation Overview:
  Cloud Technology or Research Paper for 11/30 – 12/9
- **Term Project Check-in – due Wed 12/1 @ 11:59p**

- Ch. 5: Cloud Enabling Technology
- Team planning

November 16, 2021 | TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L13.14

14

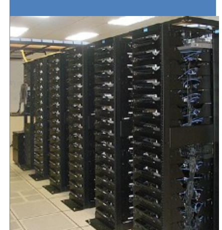
OBJECTIVES – 11/16

- Questions from 11/9
- Term Project Proposals – update by 11/19
- Tutorial 5 – Intro to FaaS II – Files in S3, CloudWatch
- Tutorial 6 - Intro to FaaS III - Serverless Databases
- Quiz 1
- **Group Presentation Overview:**
  Cloud Technology or Research Paper for 11/30 – 12/9
- Term Project Check-in – due Wed 12/1 @ 11:59p

- **Ch. 5: Cloud Enabling Technology**
- Team planning

15

CLOUD ENABLING TECHNOLOGY



16

CLOUD ENABLING TECHNOLOGY

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

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS/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

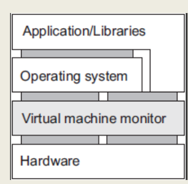
TYPE 1 VS TYPE 2 HYPERVISOR

- Hypervisor also called Virtual Machine Monitor (VMM)
- **Type 1 hypervisor**
- Typically involves 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

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.19

19

TYPE 1 HYPERVISOR



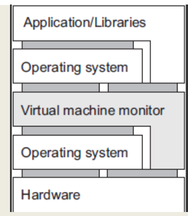
- **Native virtual machine monitor (VMM)**
  - Type 1 Hypervisor
  - Example: XEN - small OS with its own kernel
  - Provides an interface for multiple guest OSes
  - Facilitates sharing/scheduling of CPU, device I/O among many guests
  - Has its own resource scheduler(s)
  - Guest OSes require special kernel to interface with VMM
  - Supports both:
    - Paravirtualization
    - Full Virtualization

October 19, 2017	TCSS558: Applied Distributed Computing [Fall 2017] Institute of Technology, University of Washington - Tacoma	L7.20

20

TYPE 2 HYPERVISOR

- **Hosted virtual machine monitor (VMM)**
  - Type 2 hypervisor
  - Example: Virtualbox
  - Runs atop of host operating system
  - Uses host OS facilities for CPU scheduling, I/O
  - Focus on full virtualization
  - Paravirtual device drivers may be supported



21

IMPORTANCE OF HARDWARE SUPPORT

- Hardware level support (i.e. features introduced on CPUs, network cards, SSD/HDD controllers) have made virtualization faster in all respects reducing virtualization overhead

- *MAIN IDEA: full vs. paravirtualization*
- **GOAL**: run all user mode instructions directly on the CPU (*this will be fastest !*)

- x86 instruction set has ~17 problematic user mode instructions
- These must be trapped and not run by the VM
- **Full virtualization**: scan the program EXE, insert code around privileged instructions to divert control to the VMM
- **Paravirtualization**: special OS kernel eliminates side effects of privileged instructions

22

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`

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.23

23

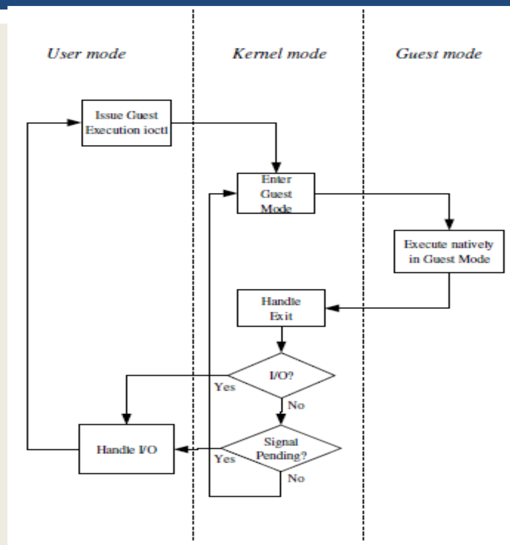
VIRTUALIZATION HARDWARE SUPPORT

- Extensions added to 64-bit Intel/AMD CPUs
- Provides hardware assisted virtualization
- Adds new "guest" operating mode to the CPU
  - Acts like hardware state switch
- Intel/AMD implementations different

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.24

24

INTEL VT-X: GUEST OPERATING MODE



25

KERNEL BASED VIRTUAL MACHINES (KVM)

- KVM integrates hypervisor functionality directly into the Linux kernel (as of Linux kernel version 2.6.20 - 2007)
- Machine “boots” the hypervisor kernel, because the Linux kernel itself ***is the hypervisor kernel***
  - KVM could be considered a “hybrid” (blend)
- KVM converts Linux into a type-1 (bare-metal) hypervisor
- KVM users Linux memory manager, process scheduler, input/output (I/O) stack, device drivers, security manager, network stack, and more
- VMs are implemented as regular Linux processes, scheduled by the Linux scheduler
- KVM requires HW-level support to run

26

KVM – 2

- 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 - 3

- KVM is successor to QEMU hypervisor (type 2 - full virtualization)
- QEMU can interface with KVM to create type 1 VMs by using KVM in the backend to implement the VM
- KVM consists of a set of Linux kernel modules:

```
$ ls -l /lib/modules/$(uname -r)/kernel/arch/x86/kvm
total 2136
-rw-r--r-- 1 root root  209665 Sep 28 08:37 kvm-amd.ko
-rw-r--r-- 1 root root  669793 Sep 28 08:37 kvm-intel.ko
-rw-r--r-- 1 root root 1298585 Sep 28 08:37 kvm.ko
```

KVM DIFFERENCES FROM XEN

- KVM requires CPU support (Intel VT-X, AMD-V)
  - 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

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS/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

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.31

31

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS/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

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.32

32

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS/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

33

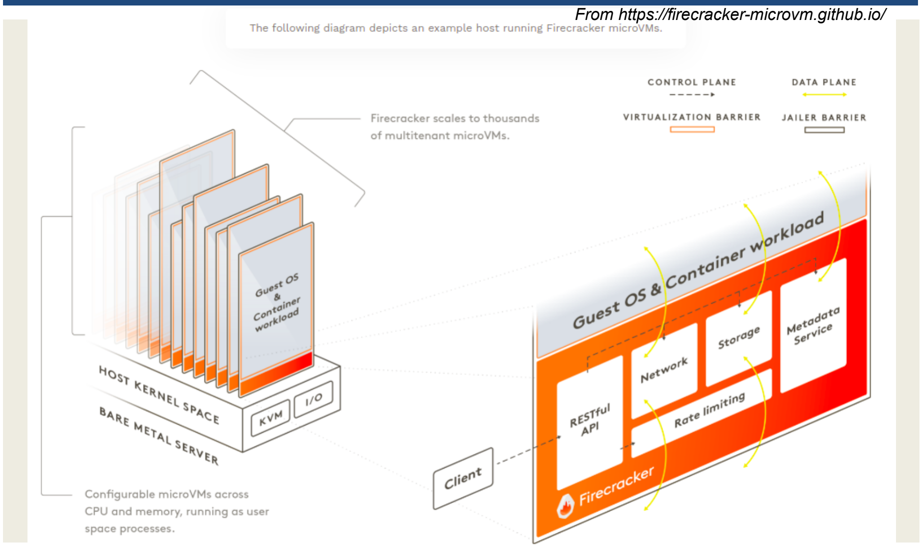
TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS/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

34

FIRECRACKER MICRO VM

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



35

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

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.36

36

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

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

WE WILL RETURN AT ~6:17 PM

39

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS/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

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.40

40

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS/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

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.41

41

TYPES OF ABSTRACTION IN THE CLOUD

- **Virtual Machines** – original IaaS cloud abstraction
- **OS/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

November 16, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L13.42

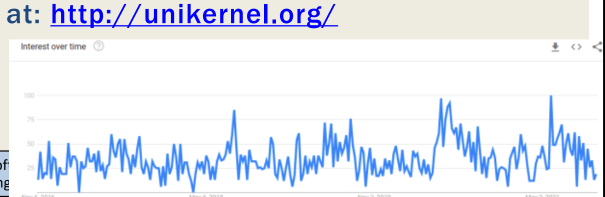
42

CASE FOR LIGHT-WEIGHT CLOUD ABSTRACTIONS

- 48 x m5d instances on single cloud server (ec2 dedicated host)
- Ran sysbench benchmark to generate prime numbers
- Ran from running 48 to 1 program instances across VMs on a shared host
- With 47 stopped VMs (not running benchmark) performance improved when running sysbench on one VM by 20.81% relative to leaving VMs up and idle
- **CONCLUSION**: idle VMs in the cloud can have a negative performance impact even when they do nothing at all
- Idle Linux servers may have hundreds of processes and perform context switches, creating possible memory cache stress
- *From: Han, X., Schooley, R., Mackenzie, D., David, O., Lloyd, W., Characterizing Public Cloud Resource Contention to Support Virtual Machine Co-residency Prediction, IC2E 2020, Apr 2020.*

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

VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

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

- Examples
  - OpenNebula
  - Nimbus
  - Eucalyptus
  - OpenStack

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

47

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

48

VIM FEATURES - 3

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

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

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

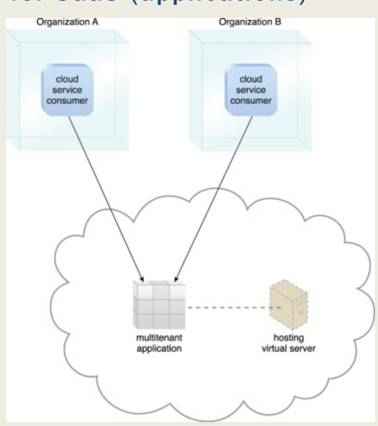
- Each tenant (like in an apartment) has their own view of the application
- Tenants are unaware of their neighbors
- Tenants can only access their data, no access to data and configuration that is not their own

- 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

53

MULTITENANT APPS - 2

- Forms the basis for SaaS (applications)



54

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 16, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L13.55

55

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 16, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L13.56

56

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>
```



```
// 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>
```

```
// 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>
```


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
  }
  ]
}
```

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

63

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

64

QUESTIONS-

65