


TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Enabling Technology, Containerization

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



1

FEEDBACK FROM 11/4

- Perspective on material: 6.46 (→ *mostly new to me*)
- Pace: 5.23 (~ just right)
- 14 respondents
- Should a team present both technical and research paper presentation?
 - No, only one presentation per team.
 - Each team member presents for ~7-8 mins
 - Presentations scaled based on team size
 - Groups will submit ranked list of 3 topic alternatives
 - Either 3 papers, or 3 technologies

November 6, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L15.2
------------------	--	-------

2

FEEDBACK - 2

- Can we turn in late tutorials, if so what is the late penalty?
 - Late assignment policy is described online on the website
- Is it possible that you give us extra activity instead of midterm
 - Yes...
 - We can label the “midterm” as an activity if this lowers stress
 - However it is still the same format, and weighted at 20%
 - Not a take-home, closed laptop, but open note/book
 - Midterm grades are curved
 - It will be okay...
- Can push tutorial due dates back one week
- Tutorials 1-7 are required
- Remainder are extra credit (scored out of 0) - better than a drop

November 6, 2019

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

L15.3

3

TERM PROJECT DELIVERABLES

- Nine project teams
- Term project lightning presentations
 - Monday December 9th (5:50-7:50pm)
 - Takes place of final exam
 - Presentation length: 5 minutes + questions, total 8 minutes
 - Format and rubric coming soon
- Term project final paper and source code repository
 - Friday December 13 @ 11:59pm
 - Paper template to be provided

November 6, 2019

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

L15.4

4

IEEE CBDCOM 2020

- IEEE 6th Int. Conference on Cloud and Big Data Computing
- June 22-26, 2020 – Calgary, Alberta, Canada
- <http://cyber-science.org/2020/cbdcom/>
- Full papers: 6-8 pgs (Feb 15), Short papers: 4-6 pgs (Mar 10), Posters: 2-4 pgs (Mar 10)
- Track 4 – Cloud Management & Virtualization
 - (*) Cloudlet and serverless computing
- Opportunity to develop course projects into submissions
- Co-located events:
 - IEEE Int. Conf on Dependable, Autonomic & Secure Computing
 - IEEE Int. Conf on Pervasive Intelligence and Computing
 - IEEE Cyber Science and Technology Congress

November 6, 2019

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

L15.5

5

GROUP PRESENTATION

- Cloud technology presentation
- Cloud research paper presentation
- Submit topics and desired dates of presentation via Canvas by Monday November 18th @ 11:59pm
- Presentation dates:
 - Monday November 25 (3 groups)
 - Monday December 2 (3 groups)
 - Wednesday December 4 (3 groups)

November 6, 2019

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


L15.6

6

CLOUD ENABLING TECHNOLOGY

November 6, 2019

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




7

CLOUD ENABLING TECHNOLOGY

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

From Chapter 5



November 6, 2019

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

L15.8

8

VIRTUAL INFRASTRUCTURE MANAGEMENT (VIM)

- Middleware to manage virtual machines and infrastructure of IaaS “clouds”
- Examples
 - OpenNebula
 - Nimbus
 - Eucalyptus
 - OpenStack

November 6, 2019

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

L15.9

9

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

November 6, 2019

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

L15.10

10

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

November 6, 2019

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

L15.11

11

VIM FEATURES - 3

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

November 6, 2019

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

L15.12

12

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

November 6, 2019

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

L15.13

13

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)
 - Google Cloud Run (Summer 2019)
 - Serverless...

November 6, 2019

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

L15.14

14

CLOUD ENABLING TECHNOLOGY


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

November 6, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L15.15
------------------	--	--------

15

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



November 6, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L15.16
------------------	--	--------

16

MULTITENANT APPS - 2

■ Forms the basis for SaaS (applications)

The diagram illustrates a multi-tenant architecture. At the top, two separate boxes represent 'Organization A' and 'Organization B'. Inside each organization box is a smaller box labeled 'cloud service consumer'. Arrows from these consumers point down to a central cloud shape. Inside the cloud, there is a 'multitenant application' represented by a grid of squares and a 'hosting virtual server' represented by a stack of blocks. A dashed line connects the multitenant application and the hosting virtual server, indicating their interaction within the cloud environment.

November 6, 2019

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

L15.17

17

CLOUD ENABLING TECHNOLOGY

■ Broadband networks and internet architecture

■ Data center technology

■ Virtualization technology

■ Multitenant technology

■ Web/web services technology

November 6, 2019

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

L15.18

18

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 6, 2019

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

L15.19

19

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

November 6, 2019

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

L15.20

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

20

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, ...

November 6, 2019

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

L15.21

21

```
// 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>
```

November 6, 2019

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

L15.22

22

```
// 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>
```

November 6, 2019

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

L15.23

23

```
// 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>
```

November 6, 2019

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

L15.24

24

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
 }
]
}
```

November 6, 2019

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

L15.25

25

## 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

November 6, 2019

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

L15.26


26

# 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

|                  |                                                                                                                                          |        |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------|
| November 6, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]<br>School of Engineering and Technology, University of Washington - Tacoma | L15.27 |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------|

27



# CONTAINERIZATION

|                  |                                                                                                                                          |        |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------|
| November 6, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]<br>School of Engineering and Technology, University of Washington - Tacoma | L15.28 |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------|

28

MOTIVATION FOR CONTAINERIZATION

- Containers provide “light-weight” alternative to full OS virtualization provided by a hypervisor
- Containers do not provide a full “machine”
- Instead use operating system constructs to provide “sand boxes” for execution
  - Linux cgroups, namespaces, etc.
- Containers can run on bare metal, or atop of VMs

The diagram shows a stack of layers: Hardware at the bottom, then Host OS, then Containers engine, then Host OS's bins/libs, and finally five individual Containers on top. Each container contains an Application and Dependencies. An arrow points from the Containers engine layer to the Containers layer.

**Containers**

The diagram shows two types of virtualization. Type 1 (Type 1 Hypervisor) has Hardware at the bottom, then Hypervisor engine, then four VMs. Each VM contains Application, Dependencies, and Guest OS. Type 2 (Type 2 Hypervisor) has Hardware at the bottom, then Host OS, then Hypervisor engine, then four VMs. Each VM contains Application, Dependencies, and Guest OS.

**Hypervisor/VM**

November 6, 2019

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

L15.29

29

CONTAINER PERFORMANCE  
– LU FACTORIZATION PERFORMANCE

- Solve linear equations – matrix algebra

Performance data from IC2E 2015:  
Hypervisors vs. Lightweight Virtualization:  
A Performance Comparison

The bar chart displays MFlops (higher is better) on the y-axis, ranging from 516 to 530. The x-axis lists five platforms: KVM, DOCKER, LXC, NATIVE, and OSV. The bars are patterned: KVM (diagonal lines), DOCKER (green diagonal lines), LXC (blue horizontal lines), NATIVE (orange cross-hatch), and OSV (yellow cross-hatch). Error bars are present on each bar.

| Platform | MFlops (approx.) |
|----------|------------------|
| KVM      | 524.5            |
| DOCKER   | 527.5            |
| LXC      | 527.0            |
| NATIVE   | 525.5            |
| OSV      | 521.5            |

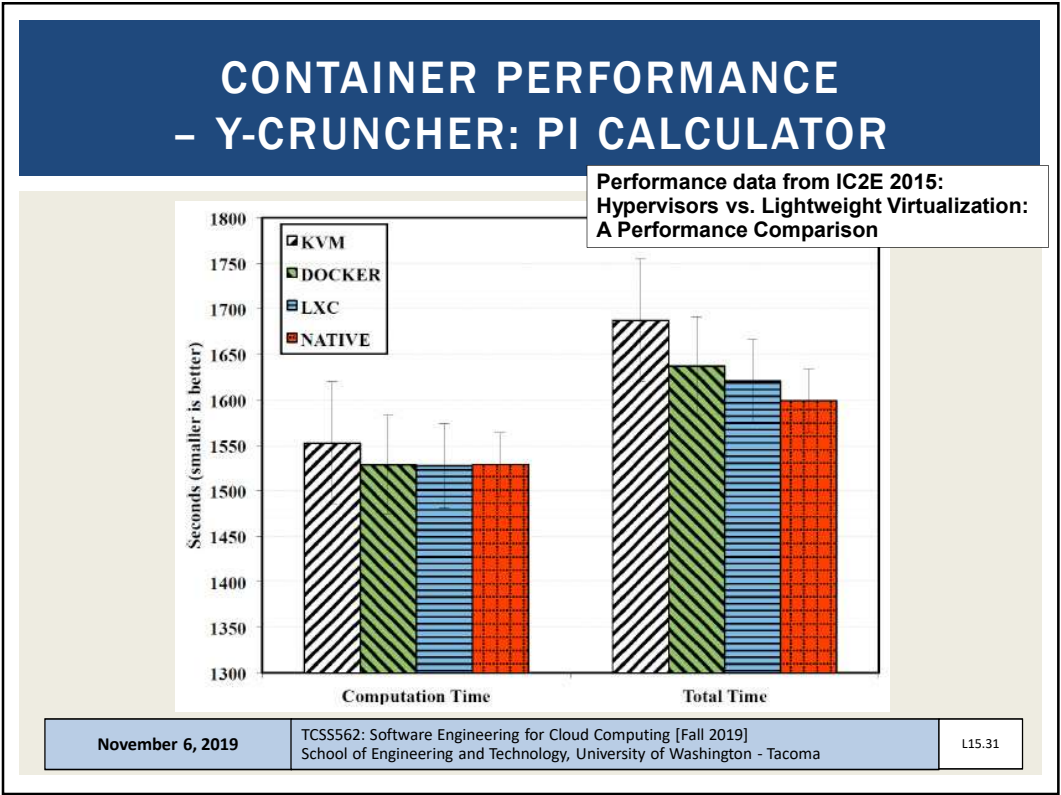
Fig. 4. The value of Linpack results on each platform over 15 runs. This is the particular case of N=1000.

November 6, 2019

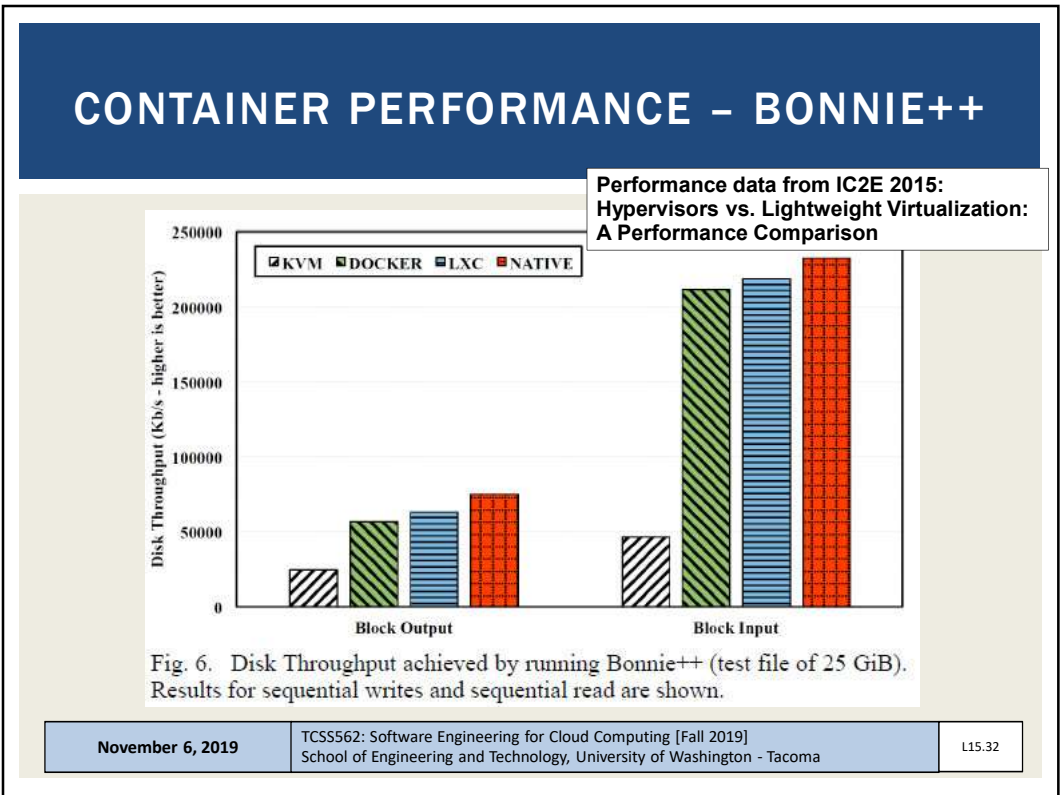
TCSS562: Software Engineering for Cloud Computing [Fall 2019]  
School of Engineering and Technology, University of Washington - Tacoma

L15.30

30




31



32



QUESTIONS



November 6, 2019

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

L15.33

33