TCSS 462/562: (SOFTWARE ENGINEERING FOR) CLOUD COMPUTING

Cloud Computing – *How did we get here?*

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

1

OBJECTIVES – 10/11

- **Questions from 10/6**
- Cloud Computing – How did we get here? *(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)*
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

2

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (47 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.89** *(↑ - previous 6.16)*

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.62** *(↑- previous 5.35)*

- **Response rates:**
- TCSS 462: 25/32 – 78.1%
- TCSS 562: 22/26 – 84.6%

3

FEEDBACK FROM 10/6

- ***I'm not quite clear on how Bit-Level and Instruction-Level Parallelism, being implicit, happens "automatically".***

- *With bit-level parallelism, arithmetic operations that require multiple instructions to perform on CPUs having lower word size can be accomplished with a single instruction on today's 64-bit CPUs*

- *Word "size" refers to the amount of data a CPU's internal data registers can hold and process at one time. Modern desktop computers have 64-bit words. Computers embedded in appliances and consumer products have word sizes of 8, 16 or 32 bits*

4

FEEDBACK - 2

- ***I understand multithreading to some degree, where multiple instructions can happen on a single or multiple cores, but multithreading requires specific code to 'break down' the program. How does parallel computing 'break down' tasks implicitly?***

- With instruction-level parallelism, CPU features like ***pipelining***, ***speculative execution***, and ***out-of-order execution*** help CPUs accomplish more than one operation per clock cycle, to appear to magically do things in parallel when developers write only sequential code to effectively gain a speed-up

- **Out-of-order execution (OoOE)** allows instructions for high-performance CPUs to begin execution as soon as their operands are ready. Although instructions are issued in-order, they can proceed out-of-order with respect to each other.

5

FEEDBACK - 3

- ***Speculative execution*** is an optimization technique where a CPU performs some task that may not be needed. Work is done before it is known if it is actually needed, to prevent a delay that would have been incurred by doing the work after it is known that it is needed.
  - This is one way for CPUs to be productive during otherwise "idle" times
- Modern pipelined microprocessors use speculative execution to reduce cost of conditional branch instructions by predicting a program's execution path based on history of branch executions. To improve performance and CPU utilization, instructions can be scheduled at a time when it is not yet known whether the instructions will need to be executed, ***ahead of a branch…***

6

FEEDBACK - 4

- ***Am seeking some clarification for what MAP-REDUCE is besides a framework that uses lots of data processed in parallel. Are cloud computing services built using this infrastructure and then it decides how the work is broken up for servers with different system hardware (heterogeneous, homogeneous, etc.)?***
- MapReduce is a programming model for writing applications to process vast amounts of data (multi-terabyte data-sets) in parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner
- We also consider for data parallelism, data processing tasks that can be sped up using a divide-and-conquer approach
- MapReduce provides a programming model and architecture for repeatedly applying the divide-and-conquer pattern

7

MAP-REDUCE

- **MapReduce** consists of two sequential tasks: Map and Reduce. **MAP** filters and sorts data while converting it into key-value pairs. **REDUCE** takes this input and reduces its size by performing some kind of summary operation over the dataset
- MapReduce drastically speeds up big data tasks by breaking down large datasets and processing them in parallel
- MapReduce paradigm was first proposed in 2004 by Google and later incorporated into the open-source **Apache Hadoop** framework for distributed processing over large datasets using files
- **Apache Spark** supports MapReduce over large datasets in RAM
- **Amazon Elastic Map Reduce (EMR)** provides cloud provider managed services for Apache Hadoop and Spark services

8

MAP-REDUCE - ADDITIONAL RESOURCES

- **Original Google paper on MapReduce:**
- https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf
- **Apache Spark:**
- https://spark.apache.org/
- **Apache Hadoop:**
- https://hadoop.apache.org/
- **Amazon Elastic Map Reduce:**
- https://aws.amazon.com/emr/

9

FEEDBACK - 3

- ***When you speak through the mic in class there's a bit of a delay and it can be somewhat distracting at times. Would you be able to change anything about that to minimize the delay?***
- Is this happening on Zoom? Or in the classroom?
- In the classroom I'm able to use the Zoom audio as output and am able to speak with less microphone feedback because of the delay (as long as the volume is not too high)
- I can not use the Zoom audio, but it may be hard to hear questions asked verbally over Zoom
- *This is a work in progress…*

10

AWS CLOUD CREDITS SURVEY

- If you did not provide your AWS account number on the AWS CLOUD CREDITS SURVEY to request AWS cloud credits and you would like credits this quarter, please contact the professor
- 56 of 58 survey completions logged as of early Oct 11th

11

TUTORIAL 1

- **Introduction to Linux & the Command Line**
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_1.pdf
- **Tutorial Sections:**
  1. The Command Line
  2. Basic Navigation
  3. More About Files
  4. Manual Pages
  5. File Manipulation
  6. VI – Text Editor
  7. Wildcards
  8. Permissions
  9. Filters
  10. Grep and regular expressions
  11. Piping and Redirection
  12. Process Management

12

TUTORIAL 2

- **Introduction to Bash Scripting**
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_2.pdf
- Review tutorial sections:
- Create a BASH webservice client
  1. What is a BASH script?
  2. Variables
  3. Input
  4. Arithmetic
  5. If Statements
  6. Loops
  7. Functions
  8. User Interface
- Call service to obtain IP address & lat/long of computer
- Call service to obtain weather forecast for lat/long

13

OBJECTIVES – 10/11

- **Questions from 10/6**
- Cloud Computing – How did we get here? *(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)*
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

14

ACTIVITY 1

- Form groups of ~3 - in class or with Zoom breakout rooms
- Each group will complete a MSWORD DOCX worksheet
- Be sure to add names at top of document as they appear in Canvas
- Activity can be completed in class or after class
- The activity can also be completed individually
- When completed, **one person** should submit a PDF of the Google Doc to Canvas
- Instructor will score all group members based on the uploaded PDF file
- To get started:
  - Log into your UW Google Account (https://drive.google.com) using you UW NET ID
  - Follow the link: https://tinyurl.com/tcss462-562-a1

15

ACTIVITY 1

- Solutions to be discussed..

16

IMPLICIT PARALLELISM

- Applies to:
- Advantages:
- Disadvantages:

17

EXPLICIT PARALLELISM

- Applies to:
- Advantages:
- Disadvantages:

18

PARALLELISM QUESTIONS

- 7. For bit-level parallelism, should a developer be concerned with the available number of virtual CPU processing cores when choosing a cloud-based virtual machine if wanting to obtain the best possible speed-up? (Yes / No)
- 8. For instruction-level parallelism, should a developer be concerned with the physical CPU's architecture used to host a cloud-based virtual machine if wanting to obtain the best possible speed-up? (Yes / No)

19

PARALLELISM QUESTIONS - 2

- 9. For thread level parallelism (TLP) where a programmer has spent considerable effort to parallelize their code and algorithms, what consequences result when this code is deployed on a virtual machine with too few virtual CPU processing cores?
- What happens when this code is deployed on a virtual machine with too many virtual CPU processing cores?

20

OBJECTIVES – 10/11

- **Questions from 10/6**
- Cloud Computing – How did we get here? *(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)*
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

21

MICHAEL FLYNN'S COMPUTER ARCHITECTURE TAXONOMY

- Michael Flynn's proposed taxonomy of computer architectures based on concurrent instructions and number of data streams (1966)
- **SISD (Single Instruction Single Data)**
- **SIMD (Single Instruction, Multiple Data)**
- **MIMD (Multiple Instructions, Multiple Data)**
- *LESS COMMON*: MISD (Multiple Instructions, Single Data)
- Pipeline architectures: functional units perform different operations on the same data
- For fault tolerance, may want to execute same instructions redundantly to detect and mask errors – for task replication

22

FLYNN'S TAXONOMY

- **SISD (Single Instruction Single Data)** Scalar architecture with one processor/core.
  - Individual cores of modern multicore processors are "SISD"
- **SIMD (Single Instruction, Multiple Data)** Supports vector processing
  - When SIMD instructions are issued, operations on individual vector components are carried out concurrently
  - Two 64-element vectors can be added in parallel
  - Vector processing instructions added to modern CPUs
  - Example: Intel MMX (multimedia) instructions

23

(SIMD): VECTOR PROCESSING ADVANTAGES

- Exploit data-parallelism: vector operations enable speedups
- Vectors architecture provide vector registers that can store entire matrices into a CPU register
- SIMD CPU extension (e.g. MMX) add support for vector operations on traditional CPUs
- Vector operations reduce total number of instructions for large vector operations
- Provides higher potential speedup vs. MIMD architecture
- Developers can think sequentially; not worry about parallelism

24

FLYNN'S TAXONOMY - 2

- **MIMD (Multiple Instructions, Multiple Data)** - system with several processors and/or cores that function asynchronously and independently
- At any time, different processors/cores may execute different instructions on different data
- Multi-core CPUs are MIMD
- Processors share memory via interconnection networks
  - Hypercube, 2D torus, 3D torus, omega network, other topologies
- MIMD systems have different methods of sharing memory
  - Uniform Memory Access (UMA)
  - Cache Only Memory Access (COMA)
  - Non-Uniform Memory Access (NUMA)

25

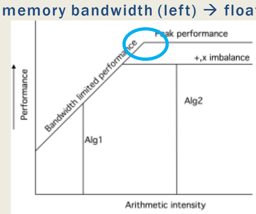
ARITHMETIC INTENSITY

- **Arithmetic intensity**: Ratio of work (W) to memory traffic r/w (Q) $I = \frac{W}{Q}$
  Example: # of floating point ops per byte of data read
- Characterizes application scalability with SIMD support
  - *SIMD can perform many fast matrix operations in parallel*
- ***High arithmetic Intensity:***
  Programs with dense matrix operations scale up nicely
  (many calcs vs memory RW, supports lots of parallelism)
- ***Low arithmetic intensity:***
  Programs with sparse matrix operations do not scale well with problem size
  (memory RW becomes bottleneck, not enough ops!)

26

ROOFLINE MODEL

- When program reaches a given arithmetic intensity performance of code running on CPU hits a "roof"
- CPU performance bottleneck changes from: memory bandwidth (left) → floating point performance (right)



Key take-aways:
When a program's has **low** Arithmetic Intensity, memory bandwidth limits performance..

With **high** Arithmetic intensity, the system has peak parallel performance…
→ *performance is limited by??*

27

OBJECTIVES – 10/11

- **Questions from 10/6**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

28

GRAPHICAL PROCESSING UNITS (GPUs)

- GPU provides multiple SIMD processors
- Typically 7 to 15 SIMD processors each
- 32,768 total registers, divided into 16 lanes (2048 registers each)
- GPU programming model: single instruction, multiple thread
- Programmed using CUDA- C like programming language by NVIDIA for GPUs
- CUDA threads – single thread associated with each data element (e.g. vector or matrix)
- Thousands of threads run concurrently

29

OBJECTIVES – 10/11

- **Questions from 10/6**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

30

PARALLEL COMPUTING

- Parallel hardware and software systems allow:
  - Solve problems demanding resources not available on single system.
  - Reduce time required to obtain solution
- The *speed-up* (S) measures effectiveness of parallelization:

$$S(N) = T(1) / T(N)$$

T(1) → execution time of total sequential computation
T(N) → execution time for performing N parallel computations in parallel

31

SPEED-UP EXAMPLE

- Consider embarrassingly parallel image processing
- Eight images (multiple data)
- Apply image transformation (greyscale) in parallel
- 8-core CPU, 16 hyperthreads
- Sequential processing: perform transformations one at a time using a single program thread
  - 8 images, 3 seconds each: `T(1) = 24 seconds`
- Parallel processing
  - 8 images, 3 seconds each: `T(N) = 3 seconds`
- Speedup: `S(N) = 24 / 3 = 8x speedup`
- Called "**perfect scaling**"
- Must consider data transfer and computation setup time

32

AMDAHL'S LAW

- Amdahl's law is used to estimate the speed-up of a job using parallel computing

1. Divide job into two parts
2. Part A that will still be sequential
3. Part B that will be sped-up with parallel computing

- Portion of computation which cannot be parallelized will determine (i.e. limit) the overall speedup
- Amdahl's law assumes jobs are of a fixed size
- Also, Amdahl's assumes no overhead for distributing the work, and a perfectly even work distribution

33

AMDAHL'S LAW

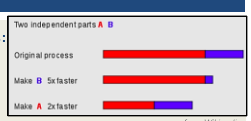
$$S = \frac{1}{(1-f) + \frac{f}{N}}$$

- S = theoretical speedup of the whole task
- f= fraction of work that is parallel (ex. 25% or 0.25)
- N= proposed speed up of the parallel part (ex. 5 times speedup)
- % improvement of task execution = 100 * (1 – (1 / S))
- **Using Amdahl's law, what is the maximum possible speed-up?**

34

AMDAHL'S LAW EXAMPLE

- Program with two independent parts:
  - Part A is 75% of the execution time
  - Part B is 25% of the execution time
- Part B is made 5 times faster with parallel computing



from Wikipedia

- Estimate the percent improvement of task execution
- Original Part A is 3 seconds, Part B is 1 second
- N=5 (speedup of part B)
- f=.25 (only 25% of the whole job (A+B) will be sped-up)
- S=1 / ((1-f) + f/S)
- S=1 / ((.75) + .25/5)
- S=1.25
- % improvement = 100 * (1 – 1/1.25) = **20%**

35

GUSTAFSON'S LAW

- Calculates the ***scaled speed-up*** using "N" processors

$$S(N) = N + (1 - N)\ \alpha$$

N: Number of processors
α: fraction of program run time which can't be parallelized (e.g. must run sequentially)

- *Can be used to estimate runtime of parallel portion of program*

36

GUSTAFSON'S LAW

- Calculates the ***scaled speed-up*** using "N" processors
  S(N) = N + (1 - N) α

N: Number of processors

α: fraction of program run time which can't be parallelized (e.g. must run sequentially)

- *Can be used to estimate runtime of parallel portion of program*
- Where α = σ / (π + σ)
- Where σ= sequential time, π =parallel time
- Our Amdahl's example: σ= 3s, π =1s, α =.75

37

GUSTAFSON'S LAW

- Calculates the ***scaled speed-up*** using "N" processors
  S(N) = N + (1 - N) α

N: Number of processors

α: fraction of program run time which can't be parallelized (e.g. must run sequentially)

- Example:
  Consider a program that is embarrassingly parallel, but 75% cannot be parallelized. α=.75
  QUESTION: ***If deploying the job on a 2-core CPU, what scaled speedup is possible assuming the use of two processes that run in parallel?***

38

GUSTAFSON'S EXAMPLE

- **QUESTION:**
  What is the maximum theoretical speed-up on a **2-core CPU** ?
  S(N) = N + (1 - N) α
  N=2, α=.75
  S(N) = 2 + (1 - 2) .75
  S(N) = ?

- What is the maximum theoretical speed-up on a 16-**core CPU**?
  S(N) = N + (1 - N) α
  N=16, α=.75
  S(N) = 16 + (1 - 16) .75
  S(N) = ?

39

GUSTAFSON'S EXAMPLE

- **QUESTION:**
  What is the maximum theoretical speed-up on a **2-core CPU** ?
  S(N) = N + (1 - N) α
  N=2, α=
  S(N) = 2
  S(N) = ?

For 2 CPUs, speed up is 1.25x

For 16 CPUs, speed up is 4.75x

- What is the maximum theoretical speed-up on a 16-core CPU?
  S(N) = N + (1 - N) α
  N=16, α=.75
  S(N) = 16 + (1 - 16) .75
  S(N) = ?

40

MOORE'S LAW

- Transistors on a chip doubles approximately every 1.5 years
- CPUs now have billions of transistors
- Power dissipation issues at faster clock rates leads to heat removal challenges
  - Transition from: increasing clock rates → to adding CPU cores
- ***Symmetric core processor*** –multi-core CPU, all cores have the same computational resources and speed
- ***Asymmetric core processor*** – on a multi-core CPU, some cores have more resources and speed
- ***Dynamic core processor*** – processing resources and speed can be dynamically configured among cores
- ***Observation: asymmetric processors offer a higher speedup***

41

OBJECTIVES – 10/11

- **Questions from 10/6**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

42

DISTRIBUTED SYSTEMS

- Collection of autonomous computers, connected through a network with distribution software called "middleware" that enables coordination of activities and sharing of resources
- **Key characteristics:**
- Users perceive system as a single, integrated computing facility.
- Compute nodes are autonomous
- Scheduling, resource management, and security implemented by every node
- Multiple points of control and failure
- Nodes may not be accessible at all times
- System can be scaled by adding additional nodes
- Availability at low levels of HW/software/network reliability

43

DISTRIBUTED SYSTEMS - 2

- Key non-functional attributes
  - Known as "ilities" in software engineering
- Availability – 24/7 access?
- Reliability - Fault tolerance
- Accessibility – reachable?
- Usability – user friendly
- Understandability – can under
- Scalability – responds to variable demand
- Extensibility – can be easily modified, extended
- Maintainability – can be easily fixed
- Consistency – data is replicated correctly in timely manner

44

TRANSPARENCY PROPERTIES OF DISTRIBUTED SYSTEMS

- **Access transparency**: local and remote objects accessed using identical operations
- **Location transparency**: objects accessed w/o knowledge of their location.
- **Concurrency transparency**: several processes run concurrently using shared objects w/o interference among them
- **Replication transparency**: multiple instances of objects are used to increase reliability
  - users are unaware if and how the system is replicated
- **Failure transparency**: concealment of faults
- **Migration transparency**: objects are moved w/o affecting operations performed on them
- **Performance transparency**: system can be reconfigured based on load and quality of service requirements
- **Scaling transparency**: system and applications can scale w/o change in system structure and w/o affecting applications

45

OBJECTIVES – 10/11

- **Questions from 10/6**
- Cloud Computing – How did we get here? *(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)*
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

46

TYPES OF MODULARITY

- ***Soft modularity:*** TRADITIONAL
- Divide a program into modules (classes) that call each other and communicate with shared-memory
- A procedure calling convention is used (or method invocation)
- ***Enforced modularity:*** CLOUD COMPUTING
- Program is divided into modules that communicate only through message passing
- The ubiquitous client-server paradigm
- Clients and servers are independent decoupled modules
- System is more robust if servers are stateless
- May be scaled and deployed separately
- May also FAIL separately!

47

CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS

- Multi-core CPU technology and hyper-threading
- What is a
  - Heterogeneous system?
  - Homogeneous system?
  - Autonomous or self-organizing system?
- **Fine grained vs. coarse grained parallelism**
- Parallel message passing code is easier to debug than shared memory (e.g. p-threads)
- Know your application's max/avg **Thread Level Parallelism** (***TLP***)
- **Data-level parallelism**: Map-Reduce, (SIMD) Single Instruction Multiple Data, Vector processing & GPUs

48

CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS - 2

- **Bit-level parallelism**
- **Instruction-level parallelism** (CPU pipelining)
- **Flynn's taxonomy**: computer system architecture classification
  - **SISD** – Single Instruction, Single Data (modern core of a CPU)
  - **SIMD** – Single Instruction, Multiple Data (Data parallelism)
  - **MIMD** – Multiple Instruction, Multiple Data
  - MISD is RARE; application for fault tolerance…
- **Arithmetic intensity**: ratio of calculations vs memory RW
- **Roofline model:**
  Memory bottleneck with low arithmetic intensity
- **GPUs**: ideal for programs with high arithmetic intensity
  - SIMD and Vector processing supported by many large registers

October 11, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L4.49

49

CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS - 3

- **Speed-up (S)**
  S(N) = T(1) / T(N)
- **Amdahl's law:**
  S = 1/ α
  α = percent of program that must be sequential
- **Scaled speedup with N processes:**
  S(N) = N – α(N-1)
- Moore's Law
- Symmetric core, Asymmetric core, Dynamic core CPU
- Distributed Systems Non-function quality attributes
- Distributed Systems – Types of Transparency
- Types of modularity- Soft, Enforced

October 11, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L4.50

50

INTRODUCTION TO CLOUD COMPUTING

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.51

51

OBJECTIVES – 10/11

- **Questions from 10/6**
- Cloud Computing – How did we get here? *(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)*
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 11, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L4.52

52

OBJECTIVES – 10/11

- **Introduction to Cloud Computing**
  - Why study cloud computing?
  - History of cloud computing
  - Business drivers
  - Cloud enabling technologies
  - Terminology
  - Benefits of cloud adoption
  - Risks of cloud adoption

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.53

53

WHY STUDY CLOUD COMPUTING?

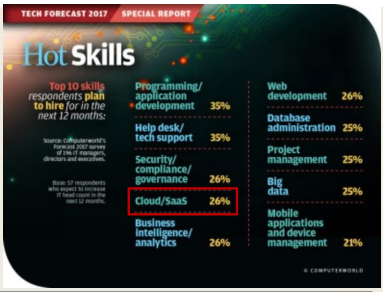
- LINKEDIN - TOP IT Skills from job app data
  - #1 Cloud and Distributed Computing
  - https://learning.linkedin.com/week-of-learning/top-skills
  - #2 Statistical Analysis and Data Mining
- FORBES Survey – 6 Tech Skills That'll Help You Earn More
  - #1 Data Science
  - #2 Cloud and Distributed Computing
  - http://www.forbes.com/sites/laurencebradford/2016/12/19/6-tech-skills-thatll-help-you-earn-more-in-2017/

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.54

54

WHY STUDY CLOUD COMPUTING? - 2

- Computerworld Magazine



October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.55

55

OBJECTIVES – 10/11

- **Introduction to Cloud Computing**
  - Why study cloud computing?
  - History of cloud computing
  - Business drivers
  - Cloud enabling technologies
  - Terminology
  - Benefits of cloud adoption
  - Risks of cloud adoption

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.56

56

A BRIEF HISTORY OF CLOUD COMPUTING

- John McCarthy, 1961
  - Turing award winner for contributions to AI
- "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility… The computer utility could become the basis of a new and important industry…"

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.57

57

CLOUD HISTORY - 2

- Internet based computer utilities
- Since the mid-1990s
- Search engines: Yahoo!, Google, Bing
- Email: Hotmail, Gmail

- 2000s
- Social networking platforms: MySpace, Facebook, LinkedIn
- Social media: Twitter, YouTube

- Popularized core concepts
- Formed basis of cloud computing

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.58

58

CLOUD HISTORY: SERVICES - 1

- Late 1990s – Early Software-as-a-Service (SaaS)
  - Salesforce: Remotely provisioned services for the enterprise
- 2002 -
  - Amazon Web Services (AWS) platform: Enterprise oriented services for remotely provisioned storage, computing resources, and business functionality
- 2006 – **Infrastructure-as-a-Service (IaaS)**
  - Amazon launches Elastic Compute Cloud (EC2) service
  - Organization can "lease" computing capacity and processing power to host enterprise applications
  - Infrastructure

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.59

59

CLOUD HISTORY: SERVICES - 2

- 2006 – **Software-as-a-Service (SaaS)**
  - Google: Offers Google DOCS, "MS Office" like fully-web based application for online documentation creation and collaboration
- 2009 – **Platform-as-a-Service (PaaS)**
  - Google: Offers Google App Engine, publicly hosted platform for hosting scalable web applications on google-hosted datacenters

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.60

60

CLOUD COMPUTING NIST GENERAL DEFINITION

"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications and services) that can be rapidly provisioned and reused with minimal management effort or service provider interaction"...

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.61

61

MORE CONCISE DEFINITION

"Cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning scalable and measured resources."

From Cloud Computing Concepts, Technology, and Architecture
Z. Mahmood, R. Puttini, Prentice Hall, 5th printing, 2015

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.62

62

OBJECTIVES – 10/11

- **Introduction to Cloud Computing**
  - Why study cloud computing?
  - History of cloud computing
  - Business drivers
  - Cloud enabling technologies
  - Terminology
  - Benefits of cloud adoption
  - Risks of cloud adoption

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.63

63

BUSINESS DRIVERS FOR CLOUD COMPUTING

- Capacity planning
- Cost reduction
- Operational overhead
- Organizational agility

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.64

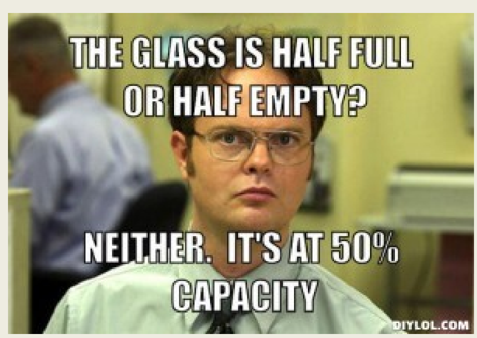
64

BUSINESS DRIVERS FOR CLOUD COMPUTING

- Capacity planning
  - Process of determining and fulfilling future demand for IT resources
  - Capacity vs. demand
  - Discrepancy between capacity of IT resources and actual demand
  - Over-provisioning: resource capacity exceeds demand
  - Under-provisioning: demand exceeds resource capacity
  - Capacity planning aims to minimize the discrepancy of available resources vs. demand

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.65

65



Dwight, The Office TV sitcom

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.66

66

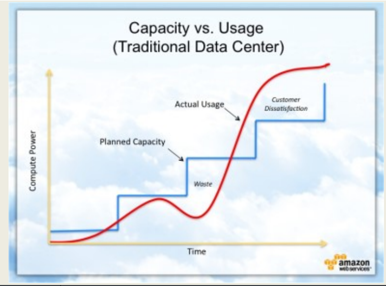
BUSINESS DRIVERS FOR CLOUD - 2

- Capacity planning
  - Over-provisioning: is costly due to too much infrastructure
  - Under-provisioning: is costly due to potential for business loss from poor quality of service
- Capacity planning strategies
  - Lead strategy: add capacity in anticipation of demand (pre-provisioning)
  - Lag strategy: add capacity when capacity is fully leveraged
  - Match strategy: add capacity in small increments as demand increases
- Load prediction
  - Capacity planning helps anticipate demand flucations

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.67

67

CAPACITY PLANNING

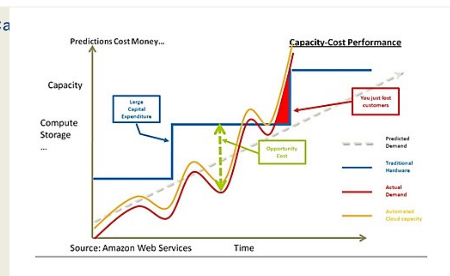


October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.68

68

CAPACITY PLANNING - 2

- Ca



October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.69

69

BUSINESS DRIVERS FOR CLOUD - 3

- Cost reduction
  - IT Infrastructure acquisition
  - IT Infrastructure maintenance
- Operational overhead
  - Technical personnel to maintain physical IT infrastructure
  - System upgrades, patches that add testing to deployment cycles
  - Utility bills, capital investments for power and cooling
  - Security and access control measures for server rooms
  - Admin and accounting staff to track licenses, support agreements, purchases

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.70

70

BUSINESS DRIVERS FOR CLOUD - 4

- Organizational agility
  - Ability to adapt and evolve infrastructure to face change from internal and external business factors
  - Funding constraints can lead to insufficient on premise IT
  - Cloud computing enables IT resources to scale with a lower financial commitment

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.71

71

OBJECTIVES – 10/11

- **Introduction to Cloud Computing**
  - Why study cloud computing?
  - History of cloud computing
  - Business drivers
  - Cloud enabling technologies
  - Terminology
  - Benefits of cloud adoption
  - Risks of cloud adoption

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.72

72

TECHNOLOGY INNOVATIONS LEADING TO CLOUD

- Cluster computing
- Grid computing
- Virtualization
- Others

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.73

73

CLUSTER COMPUTING

- Cluster computing (clustering)
  - Cluster is a group of independent IT resources interconnected as a single system
  - Servers configured with homogeneous hardware and software
    - Identical or similar RAM, CPU, HDDs
  - Design emphasizes redundancy as server components are easily interchanged to keep overall system running
    - Example: if a RAID card fails on a key server, the card can be swapped from another redundant server
  - Enables warm replica servers
    - Duplication of key infrastructure servers to provide HW failover to ensure high availability (HA)

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.74

74

GRID COMPUTING

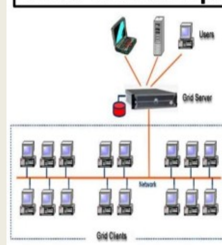
- On going research area since early 1990s
- Distributed heterogeneous computing resources organized into logical pools of loosely coupled resources
- For example: heterogeneous servers connected by the internet
- Resources are heterogeneous and geographically dispersed
- Grids use middleware software layer to support workload distribution and coordination functions
- Aspects: load balancing, failover control, autonomic configuration management
- Grids have influenced clouds contributing common features: networked access to machines, resource pooling, scalability, and resiliency

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.75

75

GRID COMPUTING - 2

How Grid computing works ?

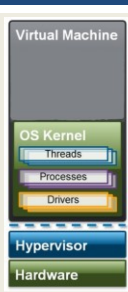


In general, a grid computing system requires:
- At least one computer, usually a server, which handles all the administrative duties for the System
- A network of computers running special grid computing network software.
- A collection of computer software called middleware

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.76

76

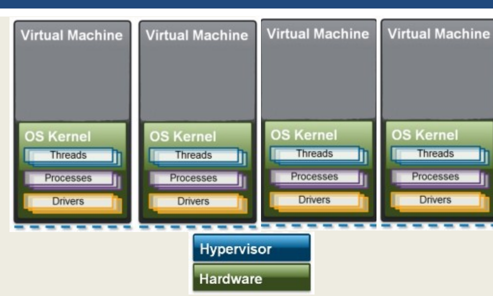
VIRTUALIZATION



October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.77

77

VIRTUALIZATION



October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.78

78

VIRTUALIZATION

- Simulate physical hardware resources via software
  - The virtual machine (virtual computer)
  - Virtual local area network (VLAN)
  - Virtual hard disk
  - Virtual network attached storage array (NAS)
- Early incarnations featured significant performance, reliability, and scalability challenges
- CPU and other HW enhancements have minimized performance GAPs

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.79

79

OBJECTIVES – 10/11

- **Introduction to Cloud Computing**
  - Why study cloud computing?
  - History of cloud computing
  - Business drivers
  - Cloud enabling technologies
  - Terminology
  - Benefits of cloud adoption
  - Risks of cloud adoption

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.80

80

KEY TERMINOLOGY

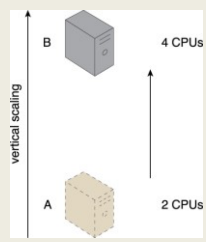
- **On-Premise Infrastructure**
  - Local server infrastructure not configured as a cloud
- **Cloud Provider**
  - Corporation or private organization responsible for maintaining cloud
- **Cloud Consumer**
  - User of cloud services
- **Scaling**
  - **Vertical scaling**
    - Scale up: increase resources of a single virtual server
    - Scale down: decrease resources of a single virtual server
  - **Horizontal scaling**
    - Scale out: increase number of virtual servers
    - Scale in: decrease number of virtual servers

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.81

81

VERTICAL SCALING

- Reconfigure virtual machine to have different resources:
  - CPU cores
  - RAM
  - HDD/SDD capacity
- May require VM migration if physical host machine resources are exceeded

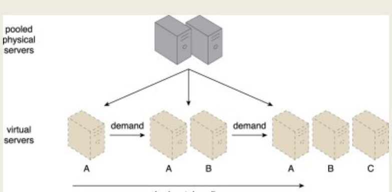


October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.82

82

HORIZONTAL SCALING

- Increase (scale-out) or decrease (scale-in) number of virtual servers based on demand



October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.83

83

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.84

84

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available

85

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available
Resource replication and automated scaling	Additional setup is normally needed

86

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available
Resource replication and automated scaling	Additional setup is normally needed
Additional servers required	No additional servers required

87

HORIZONTAL VS VERTICAL SCALING

Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available
Resource replication and automated scaling	Additional setup is normally needed
Additional servers required	No additional servers required
Not limited by individual server capacity	Limited by individual server capacity

88

KEY TERMINOLOGY - 2

- Cloud services
  - Broad array of resources accessible "as-a-service"
  - Categorized as Infrastructure (IaaS), Platform (PaaS), Software (SaaS)
- Service-level-agreements (SLAs):
  - Establish expectations for: uptime, security, availability, reliability, and performance

89

OBJECTIVES – 10/11

- **Introduction to Cloud Computing**
  - Why study cloud computing?
  - History of cloud computing
  - Business drivers
  - Cloud enabling technologies
  - Terminology
  - Benefits of cloud adoption
  - Risks of cloud adoption

90

GOALS AND BENEFITS

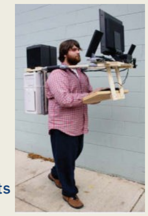
- Cloud providers
  - Leverage economies of scale through mass-acquisition and management of large-scale IT resources
  - Locate datacenters to optimize costs where electricity is low
- Cloud consumers
  - Key business/accounting difference:
  - **Cloud computing enables anticipated capital expenditures to be replaced with operational expenditures**
  - Operational expenditures always scale with the business
  - Eliminates need to invest in server infrastructure based on anticipated business needs
  - Businesses become more agile and lower their financial risks by eliminating large capital investments in physical infrastructure

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.91

91

CLOUD BENEFITS - 2

- On demand access to pay-as-you-go resources on a short-term basis (less commitment)
- Ability to acquire "unlimited" computing resources on demand when required for business needs
- Ability to add/remove IT resources at a fine-grained level
- Abstraction of server infrastructure so applications deployments are not dependent on specific locations, hardware, etc.
  - The cloud has made our software deployments more agile...



October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.92

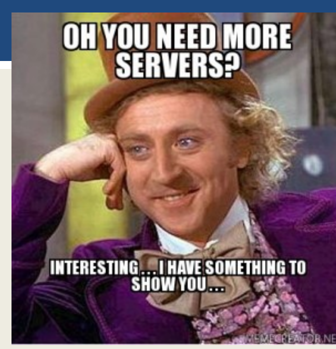
92

CLOUD BENEFITS - 3

- Example: Using 100 servers for 1 hour costs the same as using 1 server for 100 hours
- Rosetta Protein Folding: Working with a UW-Tacoma graduate student, we recently deployed this science model across 5,900 compute cores on Amazon for 2-days...
- ***What is the cost to purchase 5,900 compute cores?***
- Recent Dell Server purchase example: 20 cores on 2 servers for $4,478...
- Using this ratio 5,900 cores costs $1.3 million (purchase only)

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.93

93

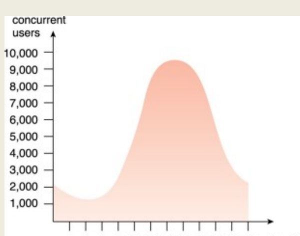


Gene Wilder, Charlie and the Chocolate Factory

94

CLOUD BENEFITS

- Increased scalability
  - Example demand over a 24-hour day →
- Increased availability
- Increased reliability



October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.95

95

OBJECTIVES – 10/11

- **Introduction to Cloud Computing**
  - Why study cloud computing?
  - History of cloud computing
  - Business drivers
  - Cloud enabling technologies
  - Terminology
  - Benefits of cloud adoption
  - Risks of cloud adoption

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.96

96

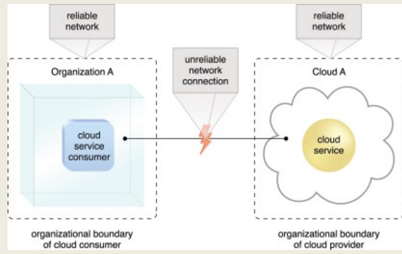
CLOUD ADOPTION RISKS

- **Increased security vulnerabilities**
  - Expansion of trust boundaries now include the external cloud
  - Security responsibility shared with cloud provider
- **Reduced operational governance / control**
  - Users have less control of physical hardware
  - Cloud user does not directly control resources to ensure quality-of-service
  - Infrastructure management is abstracted
  - Quality and stability of resources can vary
  - Network latency costs and variability

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.97

97

NETWORK LATENCY COSTS



October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.98

98

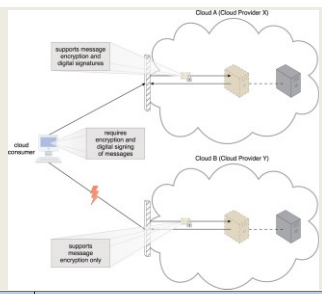
CLOUD RISKS - 2

- **Performance monitoring of cloud applications**
  - Cloud metrics (AWS cloudwatch) support monitoring cloud infrastructure (network load, CPU utilization, I/O)
  - Performance of cloud applications depends on the health of aggregated cloud resources working together
  - User must monitor this aggregate performance
- **Limited portability among clouds**
  - Early cloud systems have significant "vendor" lock-in
  - Common APIs and deployment models are slow to evolve
  - Operating system containers help make applications more portable, but containers still must be deployed
- **Geographical issues**
  - Abstraction of cloud location leads to legal challenges with respect to laws for data privacy and storage

October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.99

99

CLOUD: VENDOR LOCK-IN



October 11, 2022 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L4.100

100

QUESTIONS

October 11, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 202 School of Engineering and Technology, University of Washington - T oma | L4.101

101