TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Computing – *How did we get here? - II*

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

1

OBJECTIVES – 10/7

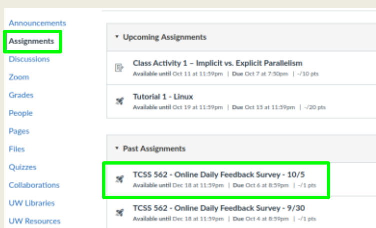
- Questions from 10/5
- Tutorial 2 – Introduction to Bash Scripting
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing –based on book #1: Cloud Computing Concepts, Technology & Architecture

October 7, 2021	TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.2

2

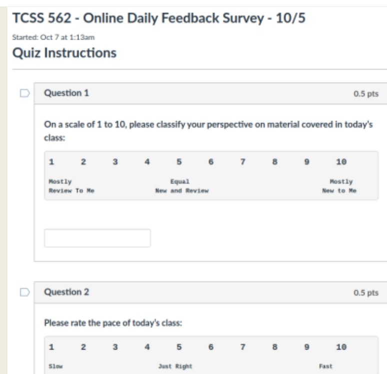
ONLINE DAILY FEEDBACK SURVEY

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



October 7, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.3

3



October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.4

4

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (15 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.58 (↑ - previous 6.15)**

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.65 (↑ - previous 5.19)**

October 7, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.5

5

FEEDBACK FROM 10/5

- ***Can we have a real-world example of how to split a problem into small chunks using parallelism?***
- The following provides a "word count" example using Hadoop and Java to introduce Map-Reduce: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
- Input data is mapped to key-value pairs
- Text file is read using StringTokenizer
- Each word is converted to a key value pair: `<The, 1>` (the unique word, and the number of occurrences)
- A local combiner, combines words and adds up counts for locally processed data to produce an output map
- All of the maps are then globally reduced by a reducer to obtain full word counts for the text file.
- Using Hadoop, tasks can run locally, or distributed across a cluster

October 7, 2021	TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.6

6

FEEDBACK - 2

- Parallelism is equivalent to multitasking, the notion of performing several things simultaneously
- In lecture #2 different types of parallelism were described: thread level, data level, etc.
- ***Does every computer now enact all types of parallelism and determine for which tasks automatically? Or is it configurable what type of parallelism is present on the computer?***

7

AVAILABLE ON X86 CPUS

	Available ?	Automatic ?
Thread-Level Parallelism (TLP)		
Data-Level Parallelism (DLP)		
Bit-Level Parallelism		
Instruction-Level Parallelism		

[1]- see: https://en.wikipedia.org/wiki/Streaming_SIMD_Extensions

8

AVAILABLE ON X86 CPUS

	Available ?	Automatic ?
Thread-Level Parallelism (TLP)	YES	NO Programmer implements threads
Data-Level Parallelism (DLP)		
Bit-Level Parallelism		
Instruction-Level Parallelism		

[1]- see: https://en.wikipedia.org/wiki/Streaming_SIMD_Extensions

9

AVAILABLE ON X86 CPUS

	Available ?	Automatic ?
Thread-Level Parallelism (TLP)	YES	NO Programmer implements threads
Data-Level Parallelism (DLP)	YES [1] But only available when using special extensions (e.g. SIMD instructions)	NO Programmer implements code to use DLP features
Bit-Level Parallelism		
Instruction-Level Parallelism		

[1]- see: https://en.wikipedia.org/wiki/Streaming_SIMD_Extensions

10

AVAILABLE ON X86 CPUS

	Available ?	Automatic ?
Thread-Level Parallelism (TLP)	YES	NO Programmer implements threads
Data-Level Parallelism (DLP)	YES [1] But only available when using special extensions (e.g. SIMD instructions)	NO Programmer implements code to use DLP features
Bit-Level Parallelism	YES	YES
Instruction-Level Parallelism		

[1]- see: https://en.wikipedia.org/wiki/Streaming_SIMD_Extensions

11

AVAILABLE ON X86 CPUS

	Available ?	Automatic ?
Thread-Level Parallelism (TLP)	YES	NO Programmer implements threads
Data-Level Parallelism (DLP)	YES [1] But only available when using special extensions (e.g. SIMD instructions)	NO Programmer implements code to use DLP features
Bit-Level Parallelism	YES	YES
Instruction-Level Parallelism	YES	YES

[1]- see: https://en.wikipedia.org/wiki/Streaming_SIMD_Extensions

12

GRANULARITY IN PARALLEL COMPUTING

- ***Can we measure granularity in parallel computing?***

$$G = \frac{T_{comp}}{T_{comm}}$$

Granularity (grain size) is measured as the ratio between:
Tcomp (computation time) and Tcomm (communication time)

- Fine grained parallelism indicates small grains
  - Lots of communication overhead is required for the parallel computation to proceed
- Coarse-grained parallelism indicates large grains (rocks)
  - Little/no communication is required for parallel work

13

FEEDBACK - 3

- Recommended paper on the future of computing in light of the decline of Moore's law →

PDF:
https://tinyurl.com/y4p8yeyj

https://dl.acm.org/doi/abs/10.1145/3282307


turing lecture

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture

14

OBJECTIVES – 10/7

- **Questions from 10/5**
- Tutorial 2 – Introduction to Bash Scripting
- 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 –based on book #1: Cloud Computing Concepts, Technology & Architecture

15

OBJECTIVES – 10/7

- **Questions from 10/5**
- Tutorial 2 – Introduction to Bash Scripting
- 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 –based on book #1: Cloud Computing Concepts, Technology & Architecture

16

ACTIVITY 1

- We will form groups of ~3 using Zoom breakout rooms
- Each group will complete a Google Doc worksheet
- Add names to Google Doc as they appear in Canvas
- The activity can be completed in class or after class
- The activity can also be completed indivually
- 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/kp2jm9pj

17

ACTIVITY 1

- Solutions to be discussed..

18

OBJECTIVES – 10/7

- **Questions from 10/5**
- Tutorial 2 – Introduction to Bash Scripting
- 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 –based on book #1: Cloud Computing Concepts, Technology & Architecture

19

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

20

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

21

(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

22

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)

23

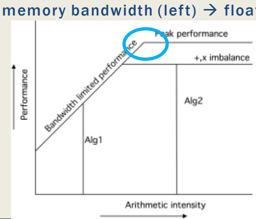
ARITHMETIC INTENSITY

- **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!)

24

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??*

OBJECTIVES – 10/7

- **Questions from 10/5**
- Tutorial 2 – Introduction to Bash Scripting
- 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 –based on book #1: Cloud Computing Concepts, Technology & Architecture

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

OBJECTIVES – 10/7

- **Questions from 10/5**
- Tutorial 2 – Introduction to Bash Scripting
- 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 –based on book #1: Cloud Computing Concepts, Technology & Architecture

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

SPEED-UP EXAMPLE

- Consider embarrassingly parallel image processing
- Eight images (multiple data)
- Apply image transformation (greyscale) in parallel
- 8-core CPU
- 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

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

31

AMDAHL'S LAW

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

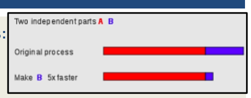
- S = theoretical speedup of the whole task
- f= fraction of work that is parallel (ex. 25% or 0.25)
- N= proposed speed up of the parallel part (ex. 5 times speedup)

- % improvement of task execution = 100 * (1 – (1 / S))

- <u>Using Amdahl's law, what is the maximum possible speed-up?</u>

32

AMDAHL'S LAW EXAMPLE



from Wikipedia

- 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
- 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%**

33

GUSTAFSON'S LAW

- Calculates the ***<u>scaled speed-up</u>*** 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*

34

GUSTAFSON'S LAW

- Calculates the ***<u>scaled speed-up</u>*** 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

35

GUSTAFSON'S LAW

- Calculates the ***<u>scaled speed-up</u>*** 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)

- <u>Example:</u>
  Consider a program that is embarrassingly parallel, but 75% cannot be parallelized. α=.75
  <u>QUESTION:</u> ***If deploying the job on a 2-core CPU, what scaled speedup is possible assuming the availability of two processor cores to run code in parallel?***

36

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) = ?

October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.37

37

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) = ?
- 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) = ?

For 2 CPUs, speed up is 1.25x

For 16 CPUs, speed up is 4.75x

October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.38

38

MOORE'S LAW

- Transistors on a chip doubles approximately every 1.5 years
- CPUs now have billions of transistors
- Heat dissipation at faster clock rates leads to cooling 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***

October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.39

39

OBJECTIVES – 10/7

- **Questions from 10/5**
- Tutorial 2 – Introduction to Bash Scripting
- 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 –based on book #1: Cloud Computing Concepts, Technology & Architecture

October 7, 2021 | TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.40

40

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

October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.41

41

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

October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.42

42

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

October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.43

43

OBJECTIVES – 10/7

- **Questions from 10/5**
- Tutorial 2 – Introduction to Bash Scripting
- 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 –based on book #1: Cloud Computing Concepts, Technology & Architecture

October 7, 2021 | TCSS562:Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.44

44

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!

October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.45

45

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

October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.46

46

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 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.47

47

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

- **Speed-up (S)** $S(N) = T(1) / T(N)$
- **Amdahl's law:** $S = 1/ \alpha$
  α = percent of program that must be sequential
- **Scaled speedup with N processes:** $S(N) = N - \alpha(N-1)$
- 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 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma | L3.48

48

INTRODUCTION TO CLOUD COMPUTING

49

OBJECTIVES – 10/7

- Questions from 10/5
- Tutorial 2 – Introduction to Bash Scripting
- 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 –based on book #1: Cloud Computing Concepts, Technology & Architecture

50

OBJECTIVES – 10/5

- 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

51

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/

52

WHY STUDY CLOUD COMPUTING? - 2

- Computerworld Magazine

53

OBJECTIVES – 10/5

- 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

54

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…"

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.55

55

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.56

56

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.57

57

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.58

58

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"…

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.59

59

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.60

60

OBJECTIVES – 10/5

- **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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.61

61

BUSINESS DRIVERS FOR CLOUD COMPUTING

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.62

62

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.63

63

Dwight, The Office TV sitcom

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.64

64

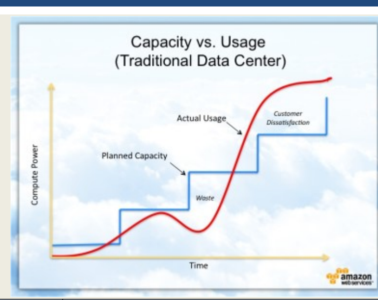
BUSINESS DRIVERS FOR CLOUD - 2

- 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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.65

65

CAPACITY PLANNING

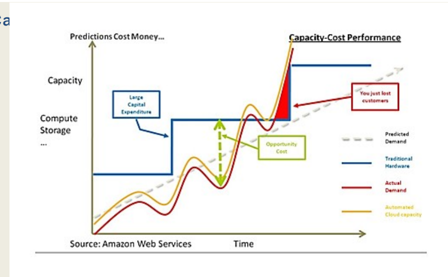


September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.66

66

CAPACITY PLANNING - 2

- Ca



67

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

68

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

69

OBJECTIVES – 10/5

- **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

70

TECHNOLOGY INNOVATIONS LEADING TO CLOUD

- Cluster computing
- Grid computing
- Virtualization
- Others

71

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)

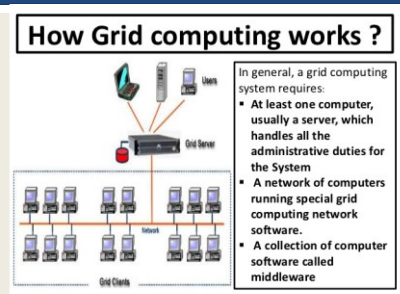
72

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

73

GRID COMPUTING - 2

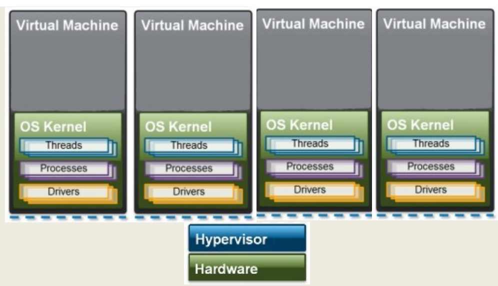


74

VIRTUALIZATION

75

VIRTUALIZATION



76

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

77

OBJECTIVES – 10/5

- **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

78

KEY TERMINOLOGY

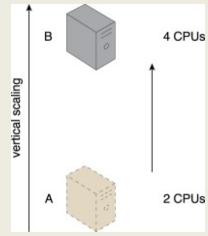
- On-Premise Infrastructure
  - Local server infrastructure not configured as a cloud
- Cloud Provider
  - Corporation or private organization responsible for maintaining cloud
- Cloud Consumer
  - User of cloud services
- Scaling
  - Vertical scaling
    - Scale up: increase resources of a single virtual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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.79

79

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

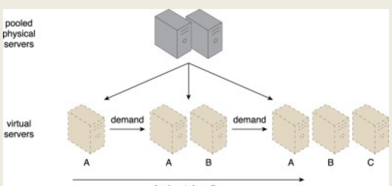


September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.80

80

HORIZONTAL SCALING

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



September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.81

81

HORIZONTAL VS VERTICAL SCALING

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.82

82

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.83

83

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.84

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
Resource replication and automated scaling	Additional setup is normally needed
Additional servers required	No additional servers required

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.85

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
Additional servers required	No additional servers required
Not limited by individual server capacity	Limited by individual server capacity

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.86

86

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.87

87

OBJECTIVES – 10/5

- 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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.88

88

GOALS AND BENEFITS

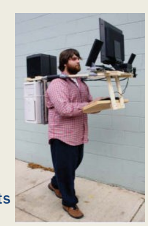
- **Cloud providers**
  - Leverage economies of scale through mass-acquisition and management of large-scale IT resources
  - Locate datacenters to optimize costs where electricity is low
- **Cloud consumers**
  - Key business/accounting difference:
  - **Cloud computing enables anticipated capital expenditures 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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.89

89

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...



September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.90

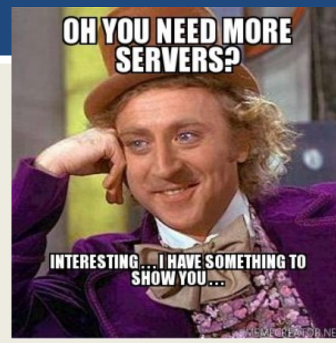
90

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)

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.91

91

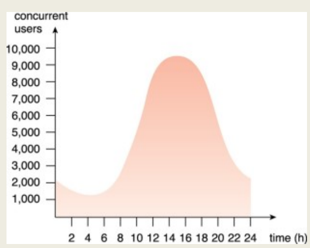


Gene Wilder, Charlie and the Chocolate Factory

92

CLOUD BENEFITS

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



September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.93

93

OBJECTIVES – 10/5

- **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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.94

94

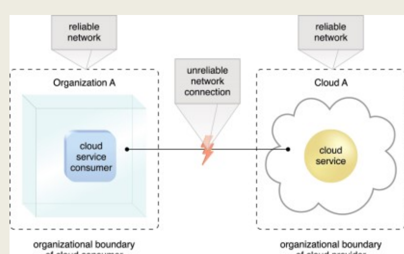
CLOUD ADOPTION RISKS

- **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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.95

95

NETWORK LATENCY COSTS



September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L3.96

96

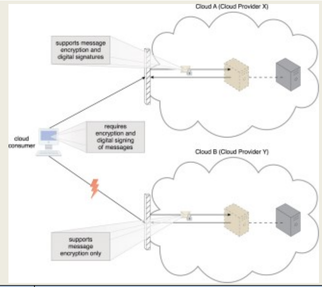
CLOUD RISKS - 2

- **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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L3.97

97

CLOUD: VENDOR LOCK-IN



September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L3.98

98

QUESTIONS

October 7, 2021 | TCSS562: Software Engineering for Cloud Computing [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma | L3.99

99