

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

Introduction

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



1

OBJECTIVES - 10/3

- **Daily Feedback Surveys**
- Questions from Course Introduction
- Demographics Survey
- AWS Cloud Credits Survey
- Tutorial 0 - Getting Started with AWS
- Tutorial 1 - Intro to Linux
- Cloud Computing - How did we get here? (10/4)
Chapter 4 Marinescu 2nd edition:
Introduction to parallel and distributed systems

October 3, 2023	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.2
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------	------

2

ONLINE DAILY FEEDBACK SURVEY

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

- Announcements
- Assignments**
- Discussions
- Zoom
- Grades
- People
- Pages
- Files
- Quizzes
- Collaborations
- UW Libraries
- UW Resources

Upcoming Assignments

- Class Activity 1 – Implicit vs. Explicit Parallelism**
Available until Oct 11 at 11:59pm | Due Oct 7 at 7:50pm | -/10 pts
- Tutorial 1 - Linux**
Available until Oct 19 at 11:59pm | Due Oct 15 at 11:59pm | -/20 pts

Past Assignments

- TCSS 562 - Online Daily Feedback Survey - 10/5**
Available until Dec 18 at 11:59pm | Due Oct 6 at 8:59pm | -/1 pts
- TCSS 562 - Online Daily Feedback Survey - 9/30**
Available until Dec 18 at 11:59pm | Due Oct 4 at 8:59pm | -/1 pts

October 3, 2023TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - TacomaL5.3

3

TCSS 562 - Online Daily Feedback Survey - 10/5

Started: Oct 7 at 1:13am

Quiz Instructions

Question 1 0.5 pts

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

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

Question 2 0.5 pts

Please rate the pace of today's class:

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

October 3, 2023TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - TacomaL5.4

4

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (58 respondents):
 - 1-mostly review, 5-equal new/review, 10-mostly new
 - **Average - 6.79** (↓ - *previous 7.43 f2022*)

- Please rate the pace of today's class:
 - 1-slow, 5-just right, 10-fast
 - **Average - 5.66** (↓ - *previous 5.83 f2022*)

- **Response rates:**
 - TCSS 462: 40/45 - 88.9%
 - TCSS 562: 18/24 - 75.0%

October 3, 2023	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L5.5
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------	------

5

FEEDBACK FROM 9/28

- **I was not clear on whether the term project group needs to consist of at most 4 people or exactly 4 people**
 - *Ideally groups will be 4 people*
 - *Even with 4 people per team there will still be 18 groups (large number)*
 - *Smaller groups (< 4 people) have fewer resources*
 - *Larger groups may be more difficult to coordinate*
- **Are graduate students required to form groups of their own, or can groups include undergraduate and graduate students**
 - *Groups can consist of both undergrad and graduate students*
 - *The grading criteria is the same*
 - *The class presentation is separate from the term project*

October 3, 2023	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L5.6
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------	------

6

FEEDBACK - 2

- **I am confused about how to create IAM user.**
- IAM users are users created inside a primary AWS account
- The primary account holder manages security enabling IAM users access to specific cloud services
- Fine grained security can be used where detailed permissions are configured to allow just enough access when sharing an account
- Typically you'll create a ROOT account when opening up a new AWS account with a create card
- If you want to share access to your account with team members, you could set up individual IAM users and enable cloud service specific permissions
- If you don't have a credit card, the instructor can create an IAM user to enable working on the assignments
 - The ROOT account owner can view and manage activity of IAM users

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.7

7

TCSS562 – SOFTWARE ENGINEERING FOR CLOUD COMPUTING

- Course webpage is embedded into Canvas
 - In CANVAS to access links:
RIGHT-CLICK – Open in new window
- Daily Feedback Surveys online at:
<http://faculty.washington.edu/wlloyd/courses/tcss562/>
- Grading
- Schedule
- Assignments

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.8

8

OBJECTIVES - 10/3

- Daily Feedback Surveys
- **Questions from Course Introduction**
- Demographics Survey
- AWS Cloud Credits Survey

- Tutorial 0 - Getting Started with AWS
- Tutorial 1 - Intro to Linux

- Cloud Computing - How did we get here?
Chapter 4 Marinescu 2nd edition:
Introduction to parallel and distributed systems

October 3, 2023	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.9
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------	------

9

OBJECTIVES - 10/3

- Daily Feedback Surveys
- Questions from Course Introduction
- **Demographics Survey**
- AWS Cloud Credits Survey

- Tutorial 0 - Getting Started with AWS
- Tutorial 1 - Intro to Linux

- Cloud Computing - How did we get here?
Chapter 4 Marinescu 2nd edition:
Introduction to parallel and distributed systems

October 3, 2023	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.10
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------	-------

10

DEMOGRAPHICS SURVEY

- Please complete the ONLINE demographics survey:

We have received 54 of 69 responses so far.
We are waiting on 15 responses.

- <https://forms.gle/QLiWgnHqbXDeNdYq7>
- Linked from course webpage in Canvas:
- <http://faculty.washington.edu/wlloyd/courses/tcss562/announcements.html>

September 28, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L1.11
--------------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

11

OBJECTIVES – 10/3

- Daily Feedback Surveys
- Questions from Course Introduction
- Demographics Survey
- **AWS Cloud Credits Survey**
- Tutorial 0 - Getting Started with AWS
- Tutorial 1 – Intro to Linux
- Cloud Computing – How did we get here?
Chapter 4 Marinescu 2nd edition:
Introduction to parallel and distributed systems

October 3, 2023	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.12
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------	-------

12

AWS CLOUD CREDITS SURVEY

- Please complete the AWS Cloud Credits survey:

Please only complete survey after setting up AWS account or if requiring an IAM user (no-credit card option)
- <https://forms.gle/G722gMn5wg9VRZXU6>
- Linked from course webpage in Canvas:
- <http://faculty.washington.edu/wlloyd/courses/tcss562/announcements.html>

September 28, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L1.13
--------------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

13

OBJECTIVES – 10/3

- Questions from Course Introduction
- Daily Feedback Surveys
- Demographics Survey
- AWS Cloud Credits Survey
- **Tutorial 0 - Getting Started with AWS**
- Tutorial 1 – Intro to Linux
- Cloud Computing – How did we get here?
Chapter 4 Marinescu 2nd edition:
Introduction to parallel and distributed systems

October 3, 2023	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.14
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------	-------

14

OBJECTIVES - 10/3

- Questions from Course Introduction
- Daily Feedback Surveys

- Demographics Survey
- AWS Cloud Credits Survey

- Tutorial 0 - Getting Started with AWS
- **Tutorial 1 - Intro to Linux**

- Cloud Computing - How did we get here?
Chapter 4 Marinescu 2nd edition:
Introduction to parallel and distributed systems

October 3, 2023	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.15
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------	-------

15

OBJECTIVES - 10/3

- Daily Feedback Surveys
- Questions from Course Introduction

- Demographics Survey
- AWS Cloud Credits Survey

- Tutorial 0 - Getting Started with AWS
- Tutorial 1 - Intro to Linux

- **Cloud Computing - How did we get here?
Chapter 4 Marinescu 2nd edition:
Introduction to parallel and distributed systems**

October 3, 2023	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.16
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------	-------

16

OBJECTIVES

■ Cloud Computing: How did we get here?

- *Parallel and distributed systems*
(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism
- Parallel architectures
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.17

17

CLOUD COMPUTING: HOW DID WE GET HERE?

- General interest in parallel computing
 - Moore's Law - # of transistors doubles every 18 months
 - Post 2004: heat dissipation challenges: can no longer easily increase cloud speed
 - Overclocking to 7GHz takes more than just liquid nitrogen:
 - <https://tinyurl.com/y93s2yz2>
- Solutions:
 - Vary CPU clock speed
 - Add CPU cores
 - **Multi-core technology**

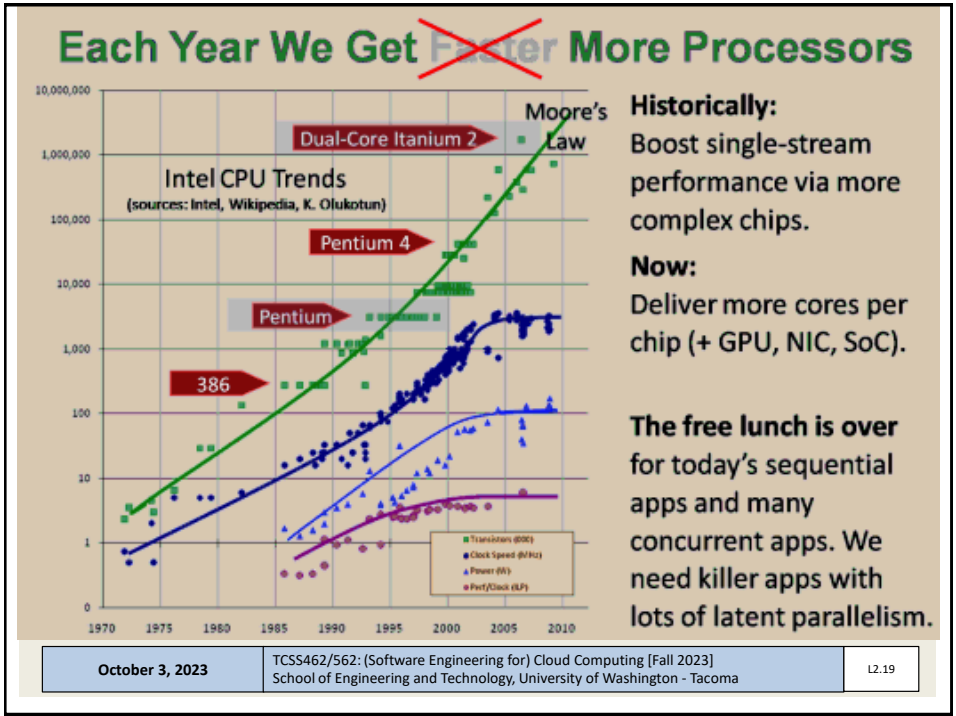


October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.18

18



19

AMD'S 64-CORE 7NM CPUS

- Epyc Rome CPUs
- Announced August 2019
- EPYC 7H12 requires liquid cooling

AMD EPYC 7002 Processors (2P)						
	Cores Threads	Frequency (GHz)		L3*	TDP	Price
		Base	Max			
EPYC 7H12	64 / 128	2.60	3.30	256 MB	280 W	?
EPYC 7742	64 / 128	2.25	3.40	256 MB	225 W	\$6950
EPYC 7702	64 / 128	2.00	3.35	256 MB	200 W	\$6450
EPYC 7642	48 / 96	2.30	3.20	256 MB	225 W	\$4775
EPYC 7552	48 / 96	2.20	3.30	192 MB	200 W	\$4025

October 3, 2023 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] | School of Engineering and Technology, University of Washington - Tacoma | L2.20

20

HOST SERVER VCPUS – AMAZON EC2 INFRASTRUCTURE-AS-A-SERVICE CLOUD

- Cloud server virtual CPUs/host
- Growth since 2006 - Amazon Compute Cloud (EC2)

▪ 1 st generation Intel: m1 – 8 vCPUs / host	(Aug 2006)
▪ 2 nd generation Intel: m2 – 16 vCPUs / host	(Oct 2009)
▪ 3 rd generation Intel: m3 – 32 vCPUs / host	(Oct 2012)
▪ 4 th generation Intel: m4 – 48 vCPUs / host	(June 2015)
▪ 5 th generation Intel: m5 – 96 vCPUs / host	(Nov 2017)
▪ 6 th generation Intel: m6i – 128 vCPUs / host	(Aug 2021)
▪ 6 th generation AMD: m6a – 192 vCPUs / host	(Nov 2021)

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.21

21

HYPER THREADING

- Modern CPUs provide multiple instruction pipelines, supporting multiple execution threads, usually 2 to feed instructions to a single CPU core...
- Two hyper-threads are not equivalent to (2) CPU cores
- i7-4770 and i5-4760 same CPU, with and without HTT
- Example: → hyperthreads add +32.9%

4770 with HTT Vs. 4670 without HTT - 25% improvement w/ HTT

CPU Mark Relative to Top 10 Common CPUs
 As of 7th of February 2014 - Higher results represent better performance

Processor	CPU Mark
Intel Core i7-4770 @ 3.40GHz	9,985
Intel Core i7-3770K @ 3.50GHz	9,842
Intel Core i7-3770 @ 3.40GHz	9,419
AMD FX-8350 Eight-Core	9,051
Intel Core i7-3820 @ 3.60GHz	9,015
Intel Core i7-2600K @ 3.40GHz	8,593
Intel Core i7-2600 @ 3.40GHz	8,316
AMD FX-8320 Eight-Core	8,121
Intel Core i5-4760 @ 3.40GHz	7,513

PassMark Software © 2008-2014

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.22

22

CLOUD COMPUTING: HOW DID WE GET HERE? - 2

- To make computing faster, we must go “parallel”
- Difficult to expose parallelism in scientific applications
- Not every problem solution has a parallel algorithm
 - Chicken and egg problem...
- Many commercial efforts promoting pure parallel programming efforts have failed
- Enterprise computing world has been *skeptical* and less involved in parallel programming

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.23
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

23

CLOUD COMPUTING: HOW DID WE GET HERE? - 3

- **Cloud computing** provides access to “infinite” scalable compute infrastructure on demand
- Infrastructure availability is key to exploiting parallelism
- **Cloud applications**
 - Based on **client-server** paradigm
 - **Thin clients** leverage compute hosted on the cloud
 - Applications run many web service instances
 - Employ load balancing

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.24
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

24

CLOUD COMPUTING: HOW DID WE GET HERE? - 4

- **Big Data** requires massive amounts of compute resources
- **MAP – REDUCE**
 - Single instruction, multiple data (SIMD)
 - Exploit data level parallelism
- **Bioinformatics example**

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.25
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

25

SMITH WATERMAN USE CASE

- Applies dynamic programming to find best local alignment of two protein sequences
 - Embarrassingly parallel, each task can run in isolation
 - Use case for GPU acceleration
- **AWS Lambda Serverless Computing Use Case:**
Goal: Pair-wise comparison of all unique human protein sequences (20,336)
 - Python client as scheduler
 - C Striped Smith-Waterman (SSW) execution engine

From: Zhao M, Lee WP, Garrison EP, Marth GT: SSW library: an SIMD Smith-Waterman C/C++ library for use in genomic applications. PLoS One 2013, 8:e82138

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.26
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

26

SMITH WATERMAN RUNTIME

- Laptop server and client (2-core, 4-HT): 8.7 hours
- AWS Lambda FaaS, laptop as client: 2.2 minutes
 - Partitions 20,336 sequences into 41 sets
 - Execution cost: ~ 82¢ (~237x speed-up)
- AWS Lambda server, EC2 instance as client: 1.28 minutes
 - Execution cost: ~ 87¢ (~408x speed-up)
- Hardware
 - Laptop client: Intel i5-7200U 2.5 GHz :4 HT, 2 CPU
 - Cloud client: EC2 Virtual Machine - m5.24xlarge: 96 vCPUs
 - Cloud server: Lambda ~1000 Intel E5-2666v3 2.9GHz CPUs

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.27
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

27

CLOUD COMPUTING: HOW DID WE GET HERE? - 3

- Compute clouds are large-scale distributed systems
 - Heterogeneous systems
 - Homogeneous systems
 - Autonomous
 - Self organizing

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.28
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

28

OBJECTIVES

- **Cloud Computing: How did we get here?**
 - *Parallel and distributed systems*
(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
 - Data, thread-level, task-level parallelism
 - Parallel architectures
 - SIMD architectures, vector processing, multimedia extensions
 - Graphics processing units
 - Speed-up, Amdahl's Law, Scaled Speedup
 - Properties of distributed systems
 - Modularity

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.29
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

29

PARALLELISM

- Discovering parallelism and development of parallel algorithms requires considerable effort
- **Example:** numerical analysis problems, such as solving large systems of linear equations or solving systems of Partial Differential Equations (PDEs), require algorithms based on domain decomposition methods.
- **How can problems be split into independent chunks?**
- **Fine-grained parallelism**
 - Only small bits of code can run in parallel without coordination
 - Communication is required to synchronize state across nodes
- **Coarse-grained parallelism**
 - Large blocks of code can run without coordination

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.30
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

30

PARALLELISM - 2

- **Coordination of nodes**
- Requires **message passing** or **shared memory**
- Debugging parallel **message passing** code is easier than parallel **shared memory** code
- **Message passing**: all of the interactions are clear
 - Coordination via specific programming API (MPI)
- **Shared memory**: interactions can be implicit – *must read the code!!*
- Processing speed is orders of magnitude faster than communication speed (CPU > memory bus speed)
- Avoiding coordination achieves the best speed-up

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.31
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

31

TYPES OF PARALLELISM

- **Parallelism**:
 - Goal: Perform multiple operations at the same time to achieve a speed-up
- **Thread-level parallelism (TLP)**
 - Control flow architecture
- **Data-level parallelism**
 - Data flow architecture
- **Bit-level parallelism**
- **Instruction-level parallelism (ILP)**

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.32
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

32

THREAD LEVEL PARALLELISM (TLP)

- Number of threads an application runs at any one time
- Varies throughout program execution
- As a metric:
- **Minimum: 1 thread**
- Can measure **average, maximum (peak)**
- **QUESTION: What are the consequences of average (TLP) for scheduling an application to run on a computer with a fixed number of CPU cores and hyperthreads?**
- Let's say there are 4 cores, or 8 hyper-threads...
- **Key to avoiding waste of computing resources is knowing your application's TLP...**

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.33

33

TLP – PRIMES EXAMPLE

- Multi-threaded prime number generation
- Compute-bound workload
- Can use variable # of threads
- Generates n prime numbers
- Runtimes: 100,000 primes
- 1 thread: 59.15 s
- 2 threads: 30.957 s
- 4 threads: 15.539 s
- 8 threads: 12.112 s
- Observe TLP with top

```
time ./primes8 30000 >/dev/null
```

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.34

34

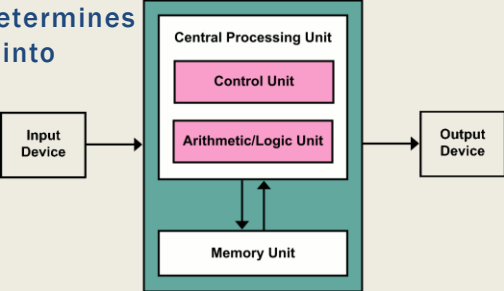
WE WILL RETURN AT
4:50PM



35

CONTROL-FLOW ARCHITECTURE

- Typical architecture used today - w/ multiple threads
- By John von Neumann (1945)
- Also called the Von Neumann architecture
- Dominant computer system architecture
- Program counter (PC) determines next instruction to load into *instruction register*
- Program execution is sequential



The diagram illustrates the Von Neumann architecture. It features a central box labeled 'Central Processing Unit' which contains a 'Control Unit' and an 'Arithmetic/Logic Unit'. Below this box is a 'Memory Unit'. An 'Input Device' is connected to the left side of the Central Processing Unit, and an 'Output Device' is connected to the right side. A double-headed arrow connects the Central Processing Unit and the Memory Unit, indicating bidirectional communication.

36

DATA-LEVEL PARALLELISM

- Partition data into big chunks, run separate copies of the program on them with little or no communication
- Problems are considered to be ***embarrassingly parallel***
- Also perfectly parallel or pleasingly parallel...
- Little or no effort needed to separate problem into a number of parallel tasks
- MapReduce programming model is an example

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.37

37

DATA FLOW ARCHITECTURE

- ***Alternate architecture*** used by network routers, digital signal processors, special purpose systems
- Operations performed when input (data) becomes available
- Envisioned to provide much higher parallelism
- Multiple problems has prevented wide-scale adoption
 - Efficiently broadcasting data tokens in a massively parallel system
 - Efficiently dispatching instruction tokens in a massively parallel system
 - Building content addressable memory large enough to hold all of the dependencies of a real program

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.38

38

DATA FLOW ARCHITECTURE - 2

- Architecture not as popular as control-flow
- Modern CPUs emulate data flow architecture for dynamic instruction scheduling since the 1990s
 - Out-of-order execution – reduces CPU idle time by not blocking for instructions requiring data by defining execution windows
 - Execution windows: identify instructions that can be run by data dependency
 - Instructions are completed in data dependency order within execution window
 - Execution window size typically 32 to 200 instructions

Utility of data flow architectures has been much less than envisioned

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.39
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

39

BIT-LEVEL PARALLELISM

- Computations on large words (e.g. 64-bit integer) are performed as a single instruction
- Fewer instructions are required on 64-bit CPUs to process larger operands (A+B) providing dramatic performance improvements
- Processors have evolved: 4-bit, 8-bit, 16-bit, 32-bit, 64-bit

QUESTION: How many instructions are required to add two 64-bit numbers on a 16-bit CPU? (Intel 8088)

- 64-bit MAX int = 9,223,372,036,854,775,807 (signed)
- 16-bit MAX int = 32,767 (signed)
- Intel 8088 – limited to 16-bit registers

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.40
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

40

INSTRUCTION-LEVEL PARALLELISM (ILP)

- CPU pipelining architectures enable ILP
- CPUs have multi-stage processing pipelines
- Pipelining: split instructions into sequence of steps that can execute concurrently on different CPU circuitry
- Basic RISC CPU - Each instruction has 5 pipeline stages:
 - **IF** - *instruction fetch*
 - **ID** - *instruction decode*
 - **EX** - *instruction execution*
 - **MEM** - *memory access*
 - **WB** - *write back*

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.41

41

CPU PIPELINING

Clock Cycle

	0	1	2	3	4	5	6	7	8	9
PIPELINE	Waiting Instructions									
	Stage 1: Fetch									
	Stage 2: Decode									
	Stage 3: Execute									
	Stage 4: Write-back									
Completed Instructions										

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.42

42

INSTRUCTION LEVEL PARALLELISM - 2

- RISC CPU:
 - After 5 clock cycles, all 5 stages of an instruction are loaded
 - Starting with 6th clock cycle, one full instruction completes each cycle
 - The CPU performs 5 tasks per clock cycle!
Fetch, decode, execute, memory read, memory write back
- Pentium 4 (CISC CPU) – processing pipeline w/ 35 stages!

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.43

43

OBJECTIVES

- **Cloud Computing: How did we get here?**
 - *Parallel and distributed systems*
(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
 - Data, thread-level, task-level parallelism
 - Parallel architectures
 - SIMD architectures, vector processing, multimedia extensions
 - Graphics processing units
 - Speed-up, Amdahl's Law, Scaled Speedup
 - Properties of distributed systems
 - Modularity

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.44

44

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

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.45
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

45

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

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.46
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

46

(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

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.47

47

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)

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.48

48

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!)

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.49
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

49

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??*

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.50
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

50

OBJECTIVES

- **Cloud Computing: How did we get here?**
 - *Parallel and distributed systems*
(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
 - Data, thread-level, task-level parallelism
 - Parallel architectures
 - SIMD architectures, vector processing, multimedia extensions
 - **Graphics processing units**
 - Speed-up, Amdahl's Law, Scaled Speedup
 - Properties of distributed systems
 - Modularity

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.51
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

51

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

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.52
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

52

OBJECTIVES

- **Cloud Computing: How did we get here?**
 - *Parallel and distributed systems*
(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
 - Data, thread-level, task-level parallelism
 - Parallel architectures
 - SIMD architectures, vector processing, multimedia extensions
 - Graphics processing units
 - **Speed-up, Amdahl's Law, Scaled Speedup**
 - Properties of distributed systems
 - Modularity

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.53

53

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

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.54

54

SPEED-UP EXAMPLE

- Consider embarrassingly parallel image processing
- Eight images (multiple data)
- Apply image transformation (greyscale) in parallel
- 8-core CPU, 16 hyper threads
- 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

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.55
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

55

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

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.56
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

56

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?**

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.57
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

57

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
- 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 = $100 * (1 - 1/1.25) = 20\%$

Two independent parts A B

from Wikipedia

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.58
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

58

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*

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.59
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

59

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*
- Where $\alpha = \sigma / (\pi + \sigma)$
- Where σ = sequential time, π = parallel time
- Our Amdahl's example: $\sigma = 3s$, $\pi = 1s$, $\alpha = .75$

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.60
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

60

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)

 - Example:**
Consider a program that is embarrassingly parallel, but 75% cannot be parallelized. $\alpha=.75$
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?*

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.61
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

61

GUSTAFSON'S EXAMPLE

- QUESTION:**
What is the maximum theoretical speed-up on a **2-core CPU** ?
$$S(N) = N + (1 - N) \alpha$$
$$N=2, \alpha=.75$$
$$S(N) = 2 + (1 - 2) .75$$
$$S(N) = ?$$
- What is the maximum theoretical speed-up on a **16-core CPU**?
$$S(N) = N + (1 - N) \alpha$$
$$N=16, \alpha=.75$$
$$S(N) = 16 + (1 - 16) .75$$
$$S(N) = ?$$

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.62
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

62

GUSTAFSON'S EXAMPLE

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

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**?
 $S(N) = N + (1 - N) \alpha$
 $N=16, \alpha=.75$
 $S(N) = 16 + (1 - 16) .75$
 $S(N) = ?$

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.63
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

63

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**

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.64
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

64

OBJECTIVES

- **Cloud Computing: How did we get here?**
 - *Parallel and distributed systems*
(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
 - Data, thread-level, task-level parallelism
 - Parallel architectures
 - SIMD architectures, vector processing, multimedia extensions
 - Graphics processing units
 - Speed-up, Amdahl's Law, Scaled Speedup
 - **Properties of distributed systems**
 - Modularity

October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.65
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

65

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 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.66
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

66

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 understand
- 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 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.67

67

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 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.68

68

OBJECTIVES

- **Cloud Computing: How did we get here?**
 - *Parallel and distributed systems*
(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
 - Data, thread-level, task-level parallelism
 - Parallel architectures
 - SIMD architectures, vector processing, multimedia extensions
 - Graphics processing units
 - Speed-up, Amdahl's Law, Scaled Speedup
 - Properties of distributed systems
 - **Modularity**

October 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.69

69

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 3, 2023

TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023]
School of Engineering and Technology, University of Washington - Tacoma

L2.70

70

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 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.71
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

71

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 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.72
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

72


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

- **Speed-up (S)**
 $S(N) = T(1) / T(N)$
- **Amdahl's law:**
 $S = 1 / ((1-f) + f/N)$, s=latency, f=parallel fraction, N=speed-up
- 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 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.73
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

73

QUESTIONS



October 3, 2023	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma	L2.74
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------	-------

74