TCSS 462/562: (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 – 10/5

- Questions from 10/3
- Tutorial 0, Tutorial 1, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- 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

2

MATERIAL / PACE

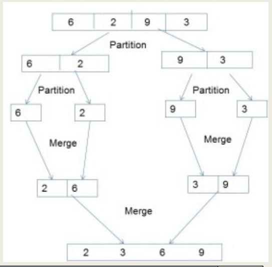
- Please classify your perspective on material covered in today's class (58 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.83** (↑ - *previous 6.79*)

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 6.26** (↓ - **previous 5.66**)

3

FEEDBACK FROM 10/3

- ***Parallelism and parallel algorithms***
- Example:
  Merge sort divides an unsorted list into the smallest possible sub-lists, compares them with the adjacent lists, and merges in a sorted order
- As the data is divided, operations can be made in parallel because they are independent
- The execution starts sequential, becomes increasingly parallel, and finishes as sequential
- Finding parallel algorithms often requires a "trick" or symmetry to enable parallelism



4

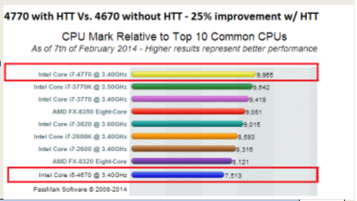
FEEDBACK - 2

- ***Difference between hyperthread and vCPU***
- vCPU stands for virtual CPU
- This refers to the CPUs provided by a virtual machine
- Since a virtual machine is a virtual server, the CPUs in virtual servers are called virtual CPUs
- Instructions executed on a virtual CPU get mapped to a physical CPU for execution
  - The virtual to physical mapping varies based on which physical CPUs are free and available
  - 4-core server, 2-core VM:
    - vCPU 0 → (CPU 0, CPU 1, CPU 2, CPU 3) mapped based on availability
    - vCPU 1 → (CPU 0, CPU 1, CPU 2, CPU 3) mapped based on availability

5

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%



6

HYPER-THREADING - 2

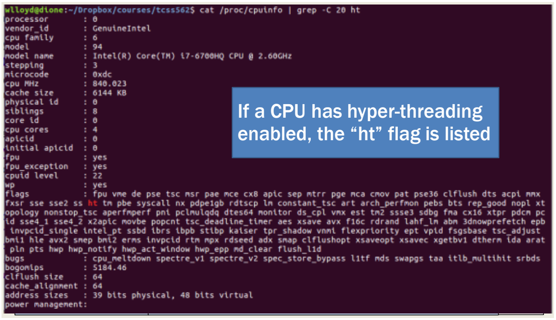
- ***How do I use hyper-threading?***

- Hyper-threading is automatic
- Modern CPUs expose each physical CPU core as two CPU cores
- `cat /proc/cpuinfo` command lists individual cores

- Operating system schedules processes & threads to run on a hyper-thread
- On CPUs with hyper-threading, each CPU core has two hyper-threads
- To the operating system they are seen as full-featured independent CPU cores

7

CAT /PROC/CPUINFO || LSCPU



```
wlloyd@dione:~/Dropbox/courses/tcss562$ cat /proc/cpuinfo | grep -C 20 ht
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 94
model name      : Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
stepping        : 3
microcode       : 0xdc
cpu MHz         : 840.023
cache size      : 6144 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx
fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xt
opology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pc
id sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb
 invpcid_single intel_pt ssbd ibrs ibpb stibp kaiser tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust
bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdseed adx smap clflushopt xsaveopt xsavec xgetbv1 dtherm ida arat
 pln pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa itlb_multihit srbds
bogomips        : 5184.46
clflush size    : 64
cache_alignment : 64
address sizes   : 39 bits physical, 48 bits virtual
power management:
```

If a CPU has hyper-threading enabled, the "ht" flag is listed

8

Hyper-Threading (HT) Technology

- Provides more satisfactory solution
- Single physical processor is shared as two logical processors
- Each logical processor has its own architecture state
- Single set of execution units are shared between logical processors
- N-logical PUs are supported
- Have the same gain % with only 5% die-size penalty.
- **HT allows single processor to fetch and execute two separate code streams simultaneously.**

Figure 2: Processors without Hyper-Threading Tech

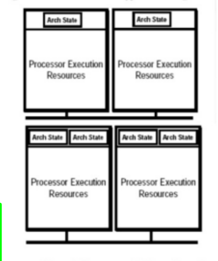

Arch State
Arch State
Processor Execution Resources
Processor Execution Resources
Arch State
Arch State
Arch State
Arch State
Processor Execution Resources
Processor Execution Resources

Figure 3: Processors with Hyper-Threading Technology

9

HYPER-THREADING - 3

- ***When should we use hyper-threading, and when should not?***
  - For personal computing, hyper-threading helps improve system performance when many programs use only short bursts of CPU time
  - Databases, HPC (science) applications, and others may benefit from disabling hyper-threading. Testing will help quantify performance.
  - Disabling hyper-threading (HW setting), cuts the number of CPU cores available to operating system in half
    - Can be disabled in the System BIOS or UEFI (uniform extensible firmware interface) software
    - BIOS / UEFI is a small resident program that can be accessed by pressing a function-key when rebooting the computer
    - BIOS / UEFI is used to configure hardware options
    - Making changes requires rebooting the computer

10

FEEDBACK - 3

- The topic of SMD architectures and vector processing was new and a little unclear
- SISD, SIMD, MIMD

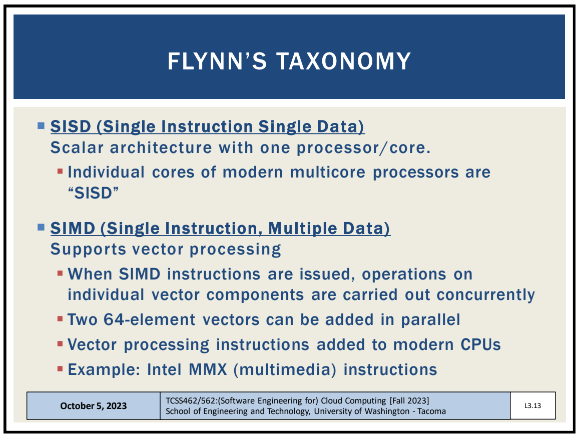
11

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

12

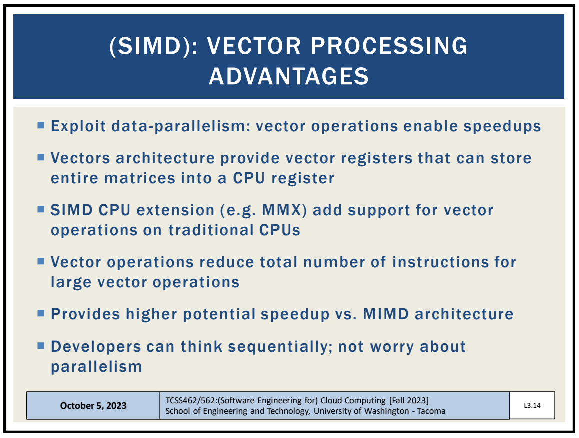


FLYNN'S TAXONOMY

- **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 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.13

13

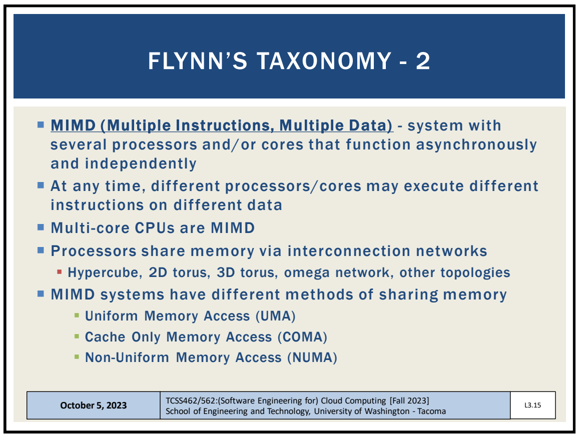


(SIMD): VECTOR PROCESSING ADVANTAGES

- 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 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.14

14

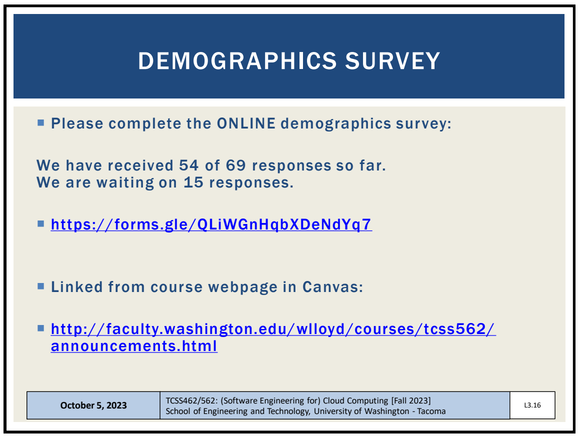


FLYNN'S TAXONOMY - 2

- **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 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.15

15



DEMOGRAPHICS SURVEY

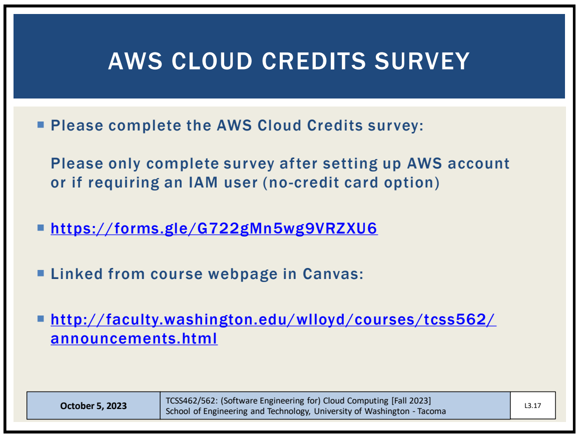
- Please complete the ONLINE demographics survey:

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

October 5, 2023 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.16

16



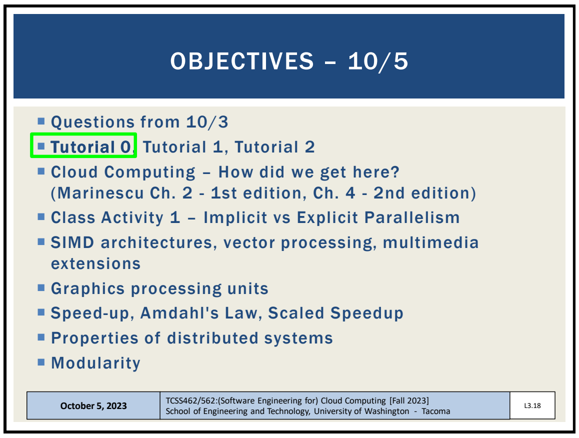
AWS CLOUD CREDITS SURVEY

- 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

October 5, 2023 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.17

17

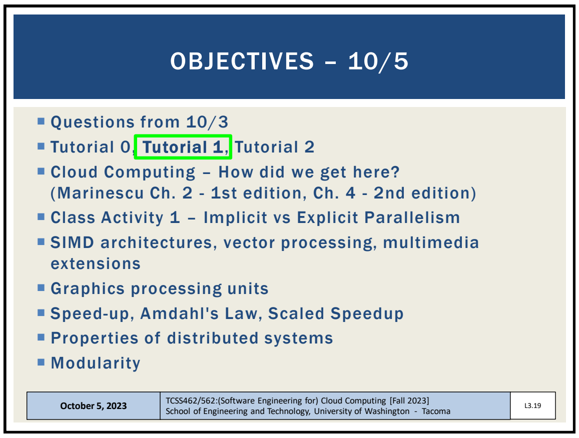


OBJECTIVES – 10/5

- Questions from 10/3
- **Tutorial 0**, Tutorial 1, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- 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

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.18

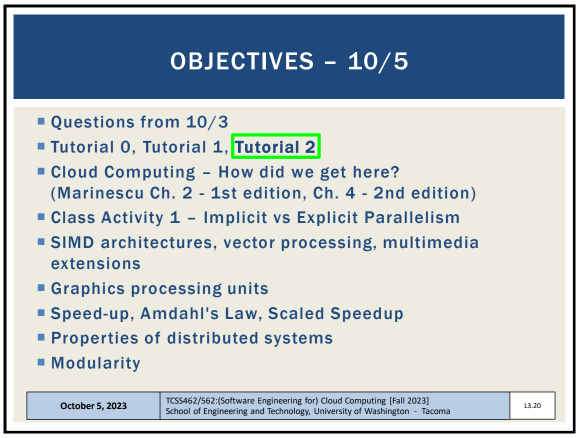
18



OBJECTIVES – 10/5

- Questions from 10/3
- Tutorial 0, **Tutorial 1**, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- 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

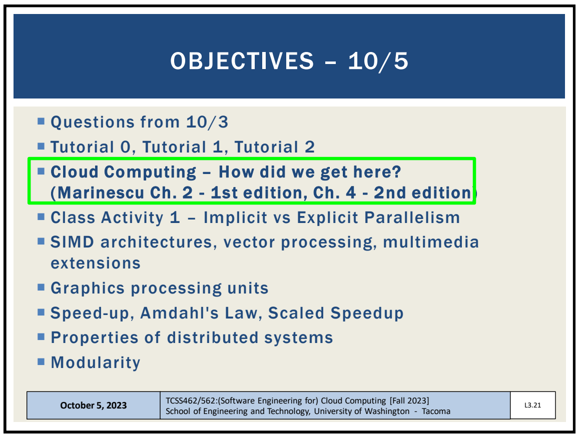
19



OBJECTIVES – 10/5

- Questions from 10/3
- Tutorial 0, Tutorial 1, **Tutorial 2**
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- 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

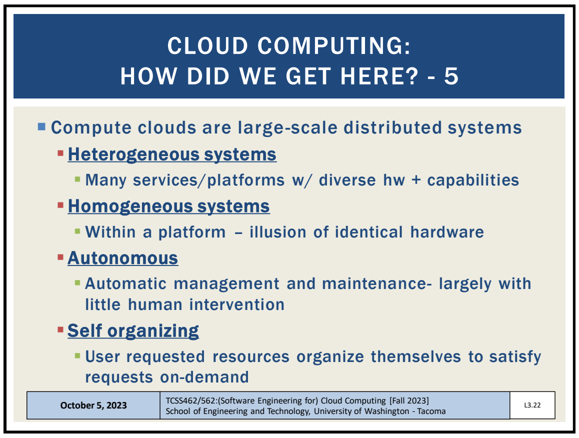
20



OBJECTIVES – 10/5

- Questions from 10/3
- Tutorial 0, Tutorial 1, Tutorial 2
- **Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)**
- 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

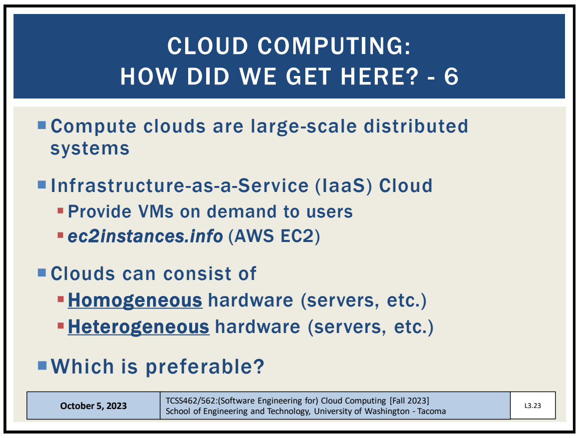
21



CLOUD COMPUTING: HOW DID WE GET HERE? - 5

- Compute clouds are large-scale distributed systems
  - **Heterogeneous systems**
    - Many services/platforms w/ diverse hw + capabilities
  - **Homogeneous systems**
    - Within a platform – illusion of identical hardware
  - **Autonomous**
    - Automatic management and maintenance- largely with little human intervention
  - **Self organizing**
    - User requested resources organize themselves to satisfy requests on-demand

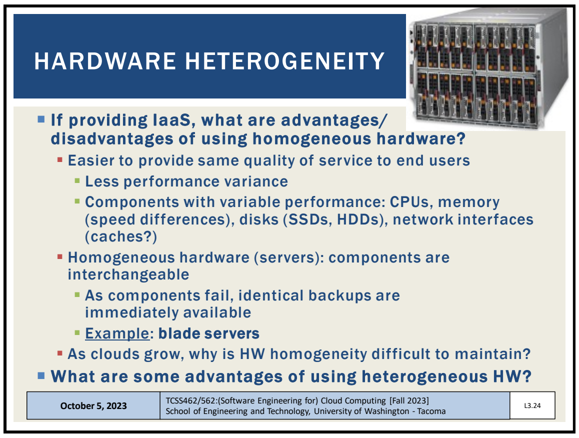
22



CLOUD COMPUTING: HOW DID WE GET HERE? - 6

- Compute clouds are large-scale distributed systems
- Infrastructure-as-a-Service (IaaS) Cloud
  - Provide VMs on demand to users
  - ***ec2instances.info*** (AWS EC2)
- Clouds can consist of
  - **Homogeneous** hardware (servers, etc.)
  - **Heterogeneous** hardware (servers, etc.)
- Which is preferable?

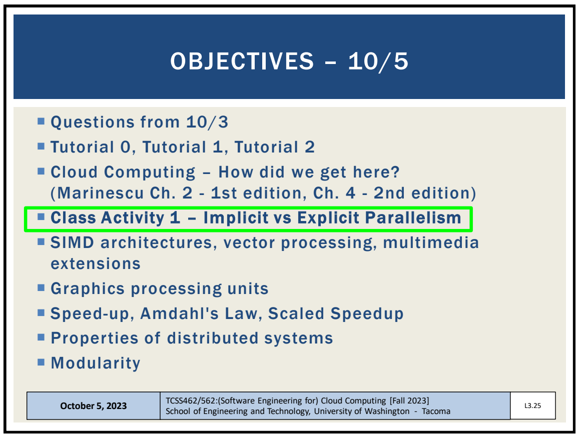
23



HARDWARE HETEROGENEITY

- **If providing IaaS, what are advantages/ disadvantages of using homogeneous hardware?**
  - Easier to provide same quality of service to end users
    - Less performance variance
    - Components with variable performance: CPUs, memory (speed differences), disks (SSDs, HDDs), network interfaces (caches?)
  - Homogeneous hardware (servers): components are interchangeable
    - As components fail, identical backups are immediately available
    - Example: **blade servers**
  - As clouds grow, why is HW homogeneity difficult to maintain?
- **What are some advantages of using heterogeneous HW?**

24

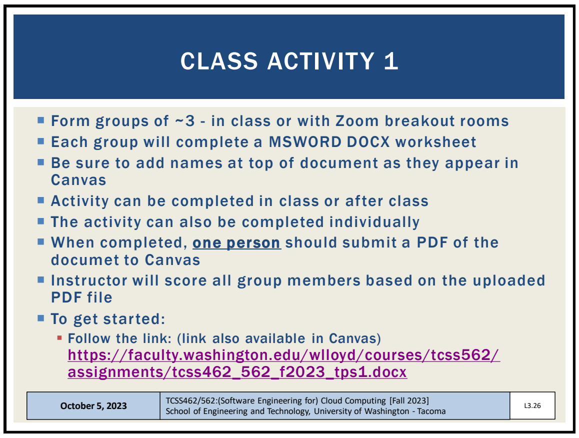


OBJECTIVES – 10/5

- Questions from 10/3
- Tutorial 0, Tutorial 1, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- **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

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.25

25

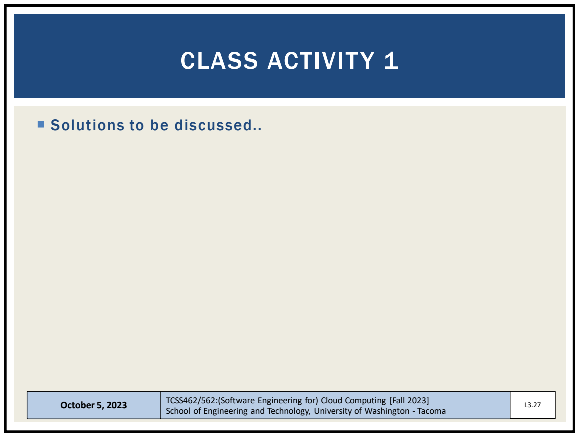


CLASS ACTIVITY 1

- Form groups of ~3 - in class or with Zoom breakout rooms
- Each group will complete a MSWORD DOCX worksheet
- Be sure to add names at top of document as they appear in Canvas
- Activity can be completed in class or after class
- The activity can also be completed individually
- When completed, one person should submit a PDF of the documet to Canvas
- Instructor will score all group members based on the uploaded PDF file
- To get started:
  - Follow the link: (link also available in Canvas) https://faculty.washington.edu/wlloyd/courses/tcss562/assignments/tcss462_562_f2023_tps1.docx

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.26

26

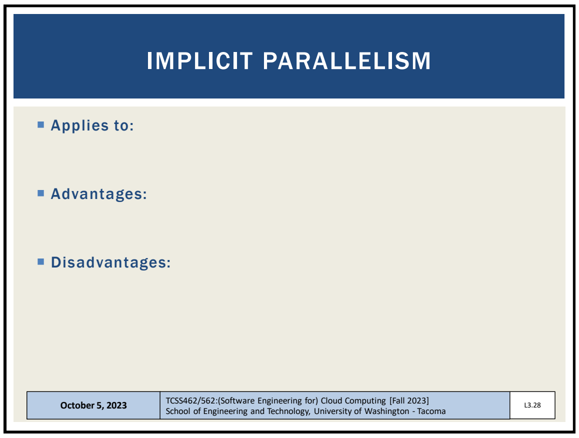


CLASS ACTIVITY 1

- Solutions to be discussed..

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.27

27

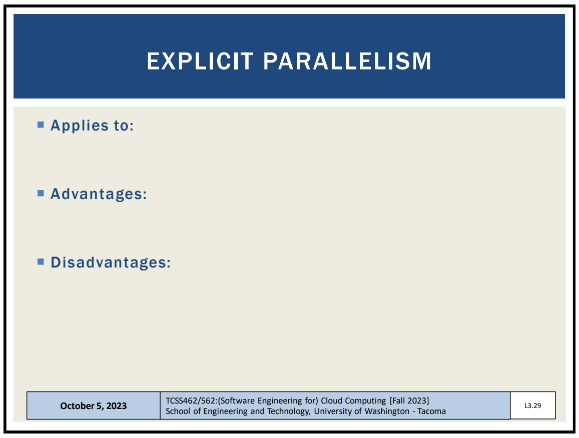


IMPLICIT PARALLELISM

- Applies to:
- Advantages:
- Disadvantages:

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.28

28

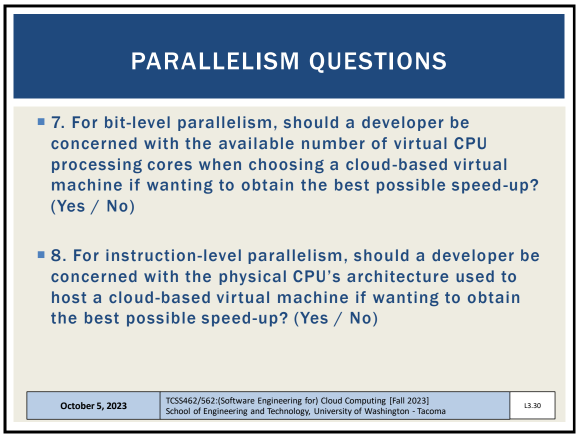


EXPLICIT PARALLELISM

- Applies to:
- Advantages:
- Disadvantages:

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.29

29



PARALLELISM QUESTIONS

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

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.30

30

PARALLELISM QUESTIONS - 2

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

- What happens when this code is deployed on a virtual machine with too many virtual CPU processing cores?

31

OBJECTIVES – 10/5

- Questions from 10/3
- Tutorial 0, Tutorial 1, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- 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

32

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*

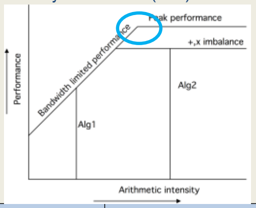
- ***High arithmetic Intensity:***
  Programs with dense matrix operations scale up nicely (many calcs vs memory RW, supports lots of parallelism)

- ***Low arithmetic intensity:***
  Programs with sparse matrix operations do not scale well with problem size
  (memory RW becomes bottleneck, not enough ops!)

33

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??

34

OBJECTIVES – 10/5

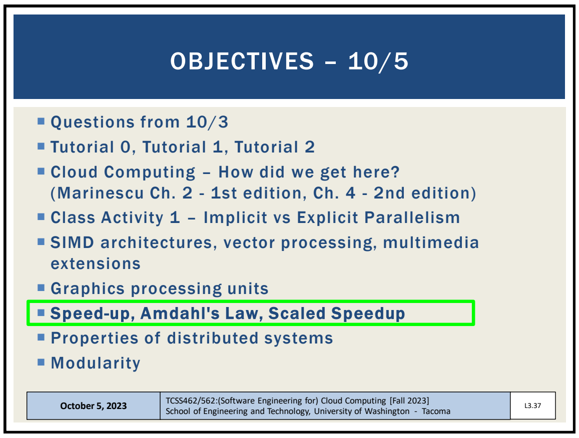
- Questions from 10/3
- Tutorial 0, Tutorial 1, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- **Graphics processing units**
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity

35

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

36

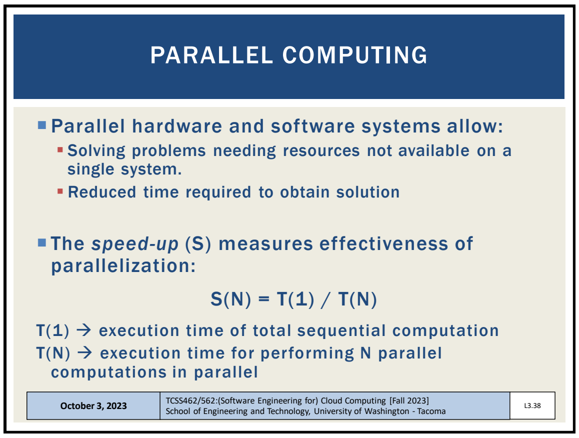


OBJECTIVES – 10/5

- Questions from 10/3
- Tutorial 0, Tutorial 1, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- 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

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.37

37



PARALLEL COMPUTING

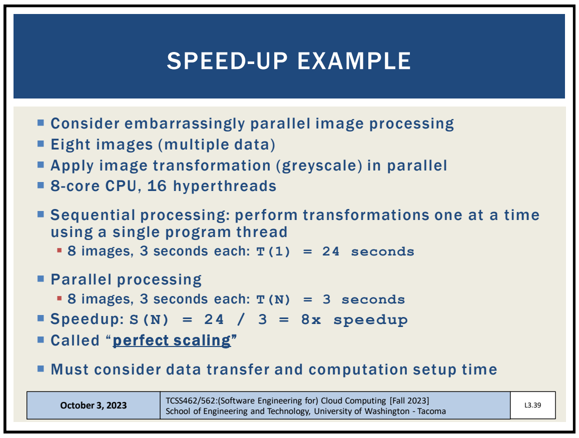
- Parallel hardware and software systems allow:
  - Solving problems needing resources not available on a single system.
  - Reduced time required to obtain solution
- The *speed-up* (S) measures effectiveness of parallelization:

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 | L3.38

38

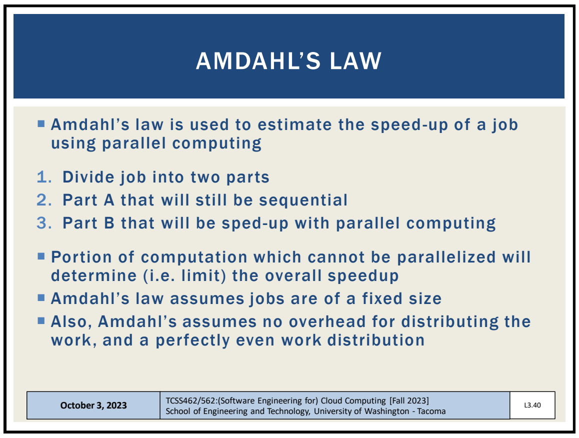


SPEED-UP EXAMPLE

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

October 3, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.39

39



AMDAHL'S LAW

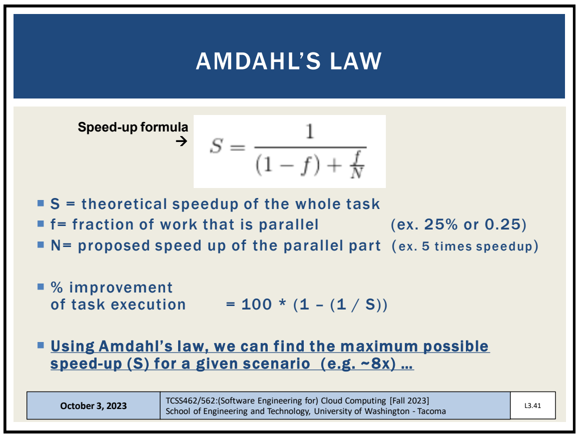
- Amdahl's law is used to estimate the speed-up of a job using parallel computing

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 | L3.40

40



AMDAHL'S LAW

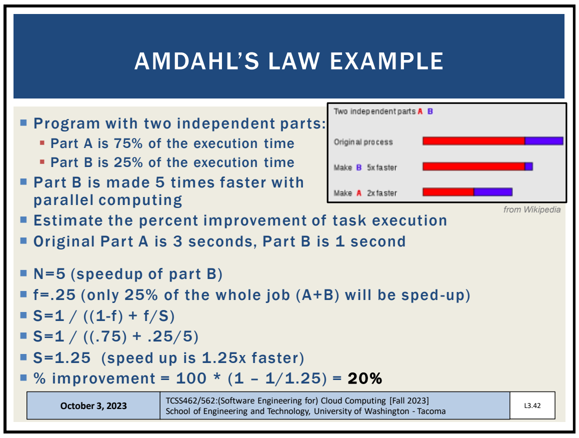
Speed-up formula →

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

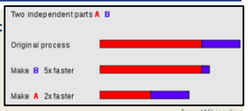
- S = theoretical speedup of the whole task
- f= fraction of work that is parallel (ex. 25% or 0.25)
- N= proposed speed up of the parallel part (ex. 5 times speedup)
- % improvement of task execution = 100 * (1 – (1 / S))
- **Using Amdahl's law, we can find the maximum possible speed-up (S) for a given scenario (e.g. ~8x) …**

October 3, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.41

41



AMDAHL'S LAW EXAMPLE

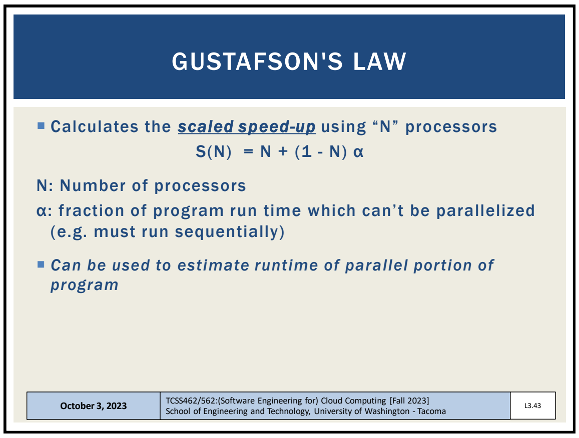


from Wikipedia

- 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 (speed up is 1.25x faster)
- % improvement = 100 * (1 – 1/1.25) = **20%**

October 3, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.42

42



GUSTAFSON'S LAW

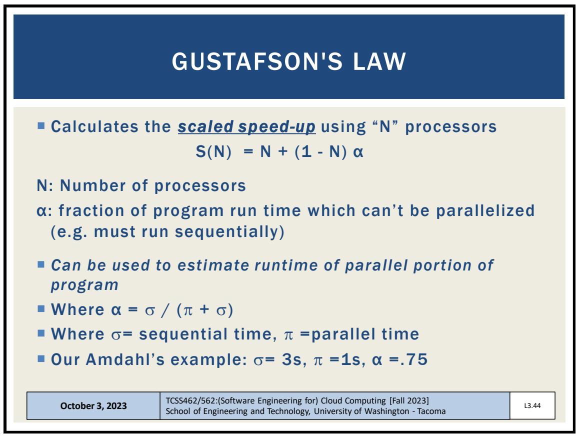
- Calculates the ***scaled speed-up*** using "N" processors
  S(N) = N + (1 - N) α

N: Number of processors

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

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

43



GUSTAFSON'S LAW

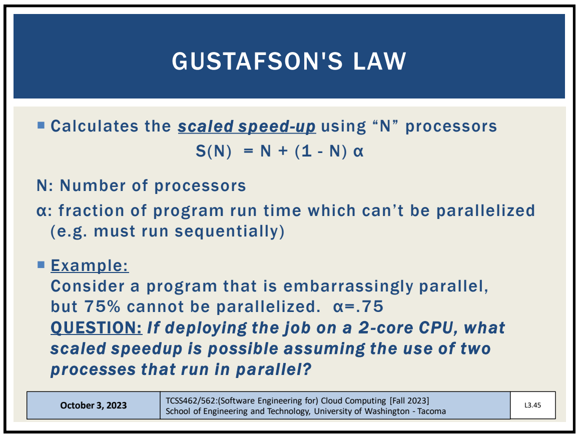
- Calculates the ***scaled speed-up*** using "N" processors
  S(N) = N + (1 - N) α

N: Number of processors

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

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

44



GUSTAFSON'S LAW

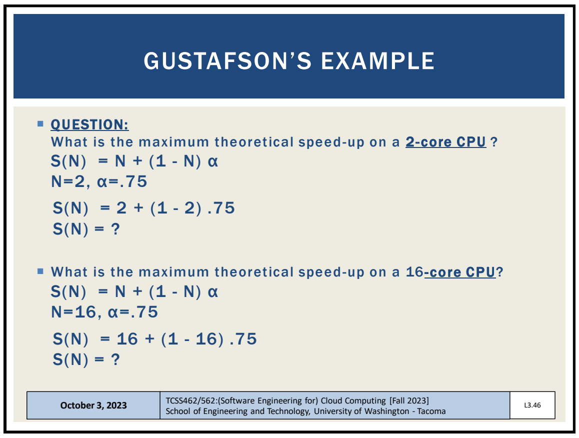
- Calculates the ***scaled speed-up*** using "N" processors
  S(N) = N + (1 - N) α

N: Number of processors

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

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

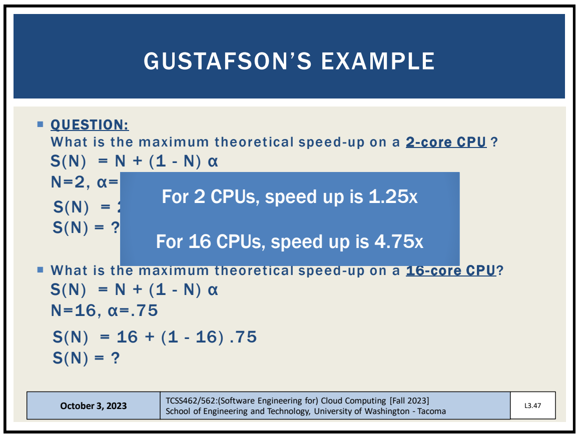
45



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) = ?

46



GUSTAFSON'S EXAMPLE

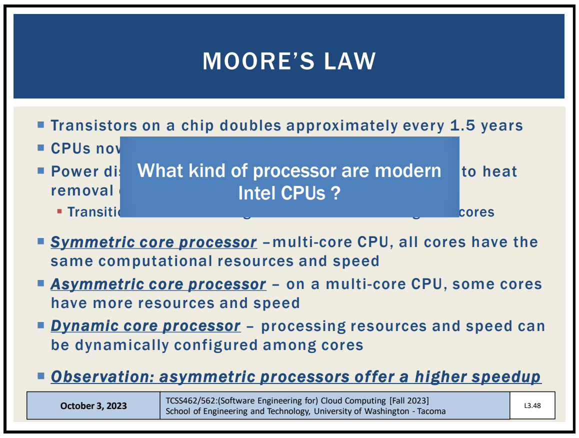
- **QUESTION:**
  What is the maximum theoretical speed-up on a **2-core CPU** ?
  S(N) = N + (1 - N) α
  N=2, α=
  S(N) = 2
  S(N) = ?

For 2 CPUs, speed up is 1.25x

For 16 CPUs, speed up is 4.75x

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

47



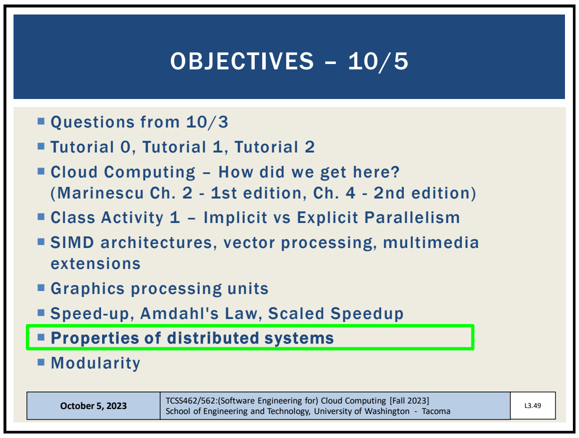
MOORE'S LAW

- Transistors on a chip doubles approximately every 1.5 years
- CPUs nov
- Power di to heat removal
  - Transiti cores

What kind of processor are modern Intel CPUs ?

- ***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***

48

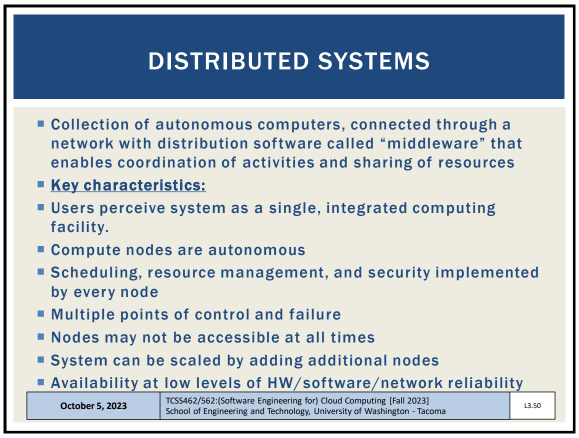


OBJECTIVES – 10/5

- Questions from 10/3
- Tutorial 0, Tutorial 1, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- 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

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.49

49

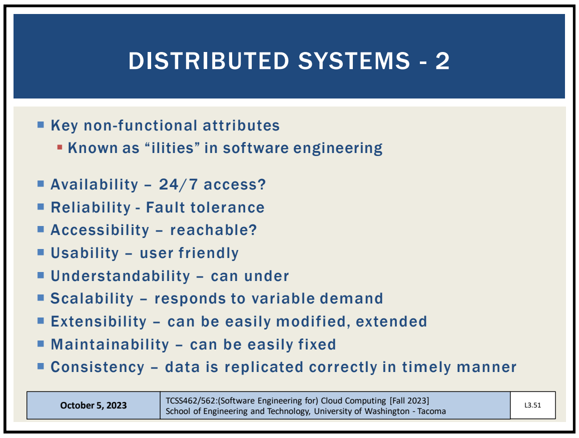


DISTRIBUTED SYSTEMS

- 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 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.50

50

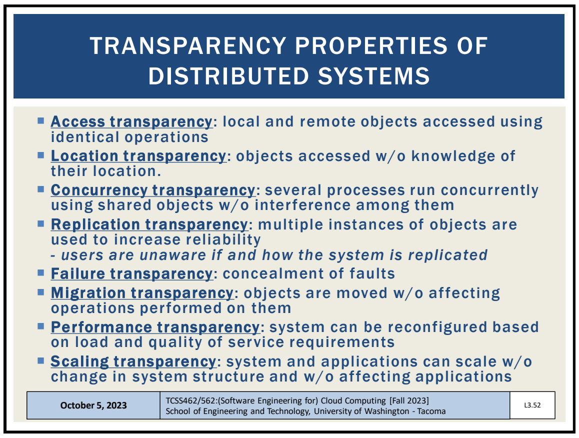


DISTRIBUTED SYSTEMS - 2

- 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 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.51

51

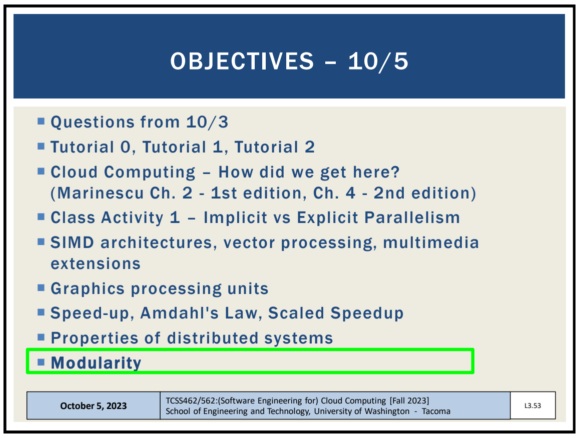


TRANSPARENCY PROPERTIES OF DISTRIBUTED SYSTEMS

- **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 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.52

52

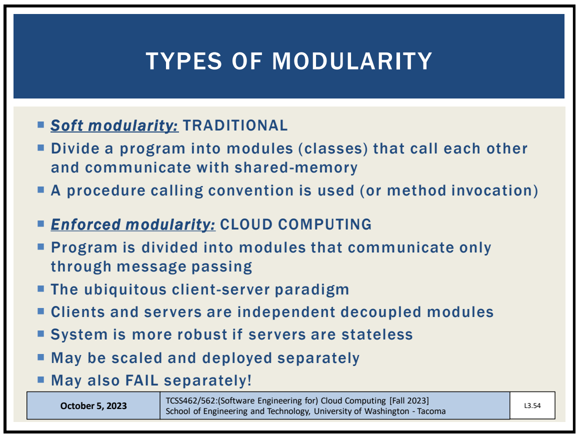


OBJECTIVES – 10/5

- Questions from 10/3
- Tutorial 0, Tutorial 1, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- 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**

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.53

53

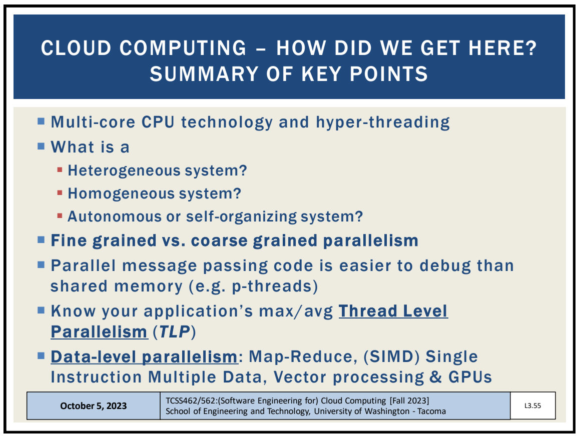


TYPES OF MODULARITY

- ***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 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.54

54

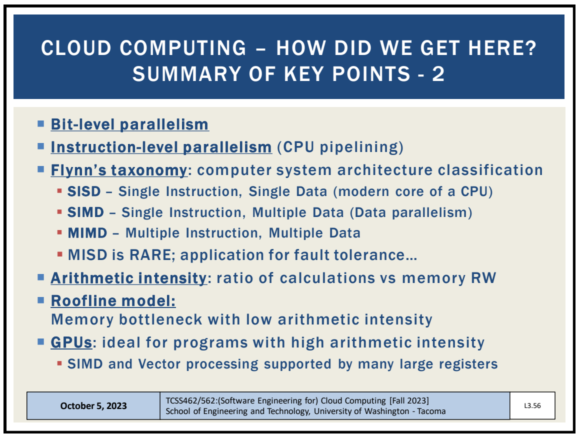


CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS

- 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 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.55

55

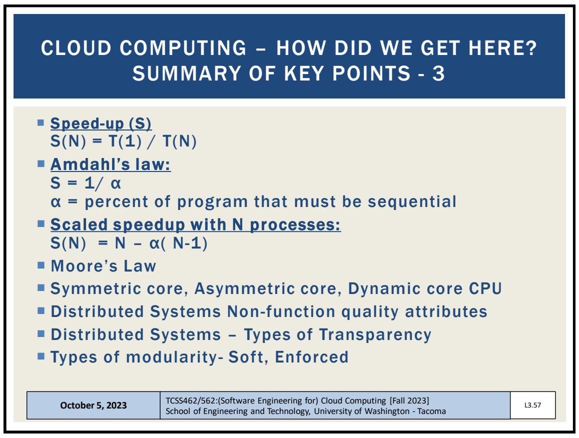


CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS - 2

- **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 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.56

56

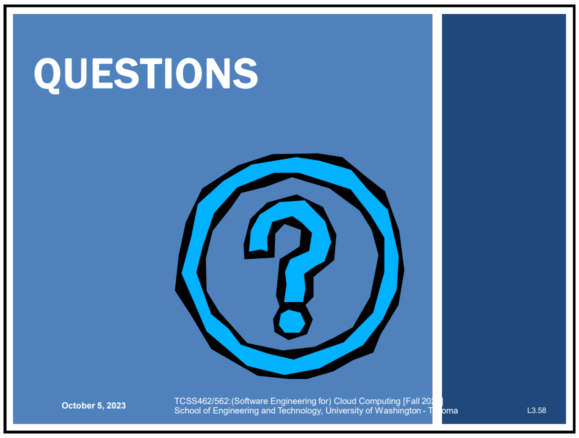


CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS - 3

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

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.57

57



QUESTIONS

October 5, 2023 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2023] School of Engineering and Technology, University of Washington - Tacoma | L3.58

58