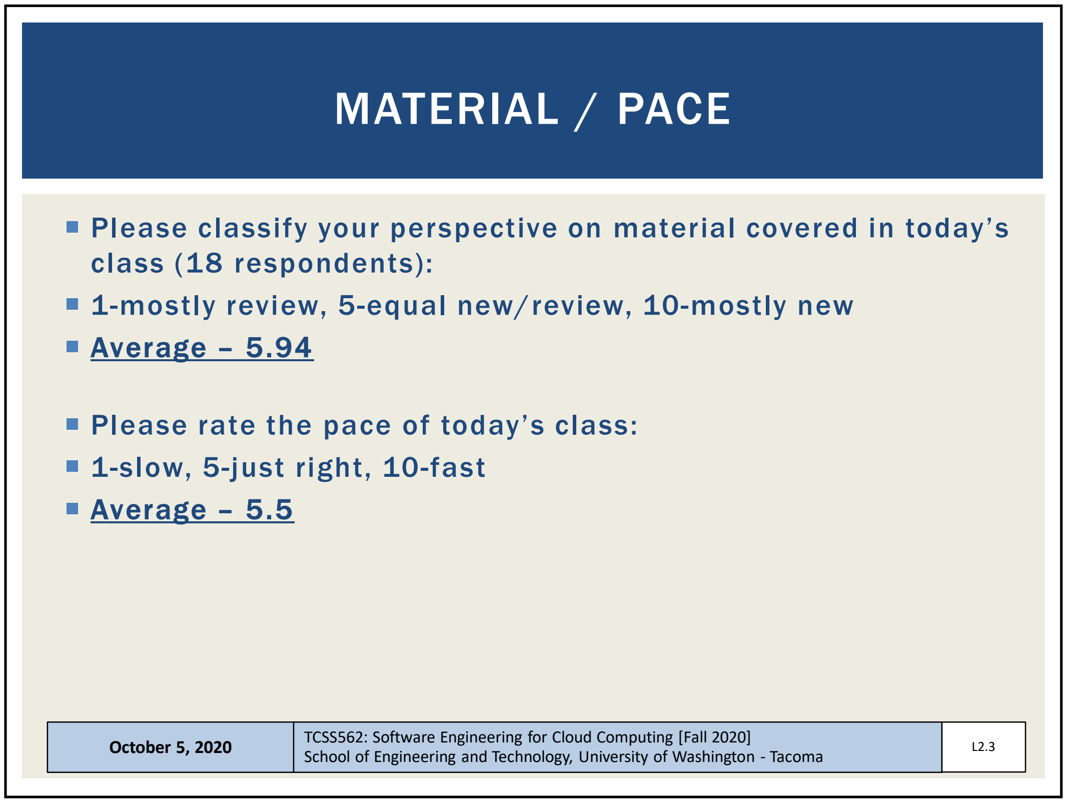
TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Cloud Computing – *How did we get here?*

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

OBJECTIVES – 10/5

- Questions from 9/30
- Cloud Computing – How did we get here? *(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)*
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

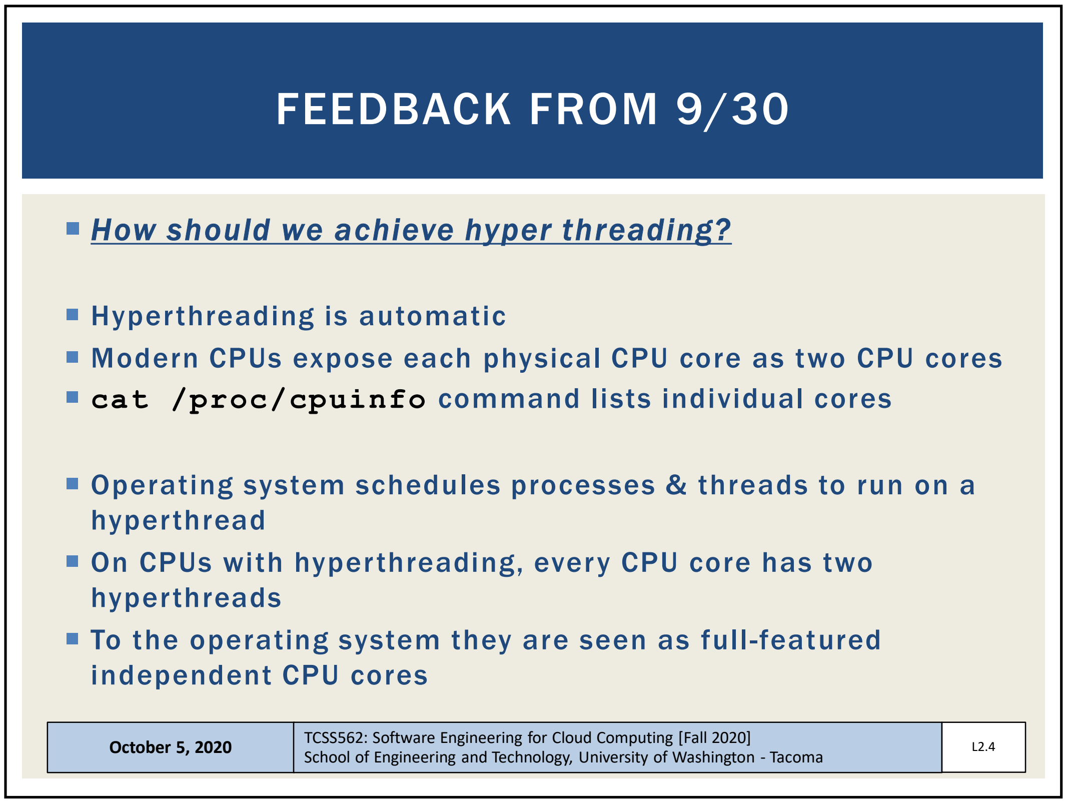


MATERIAL / PACE

- Please classify your perspective on material covered in today's class (18 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- <u>**Average – 5.94**</u>

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- <u>**Average – 5.5**</u>

October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.3



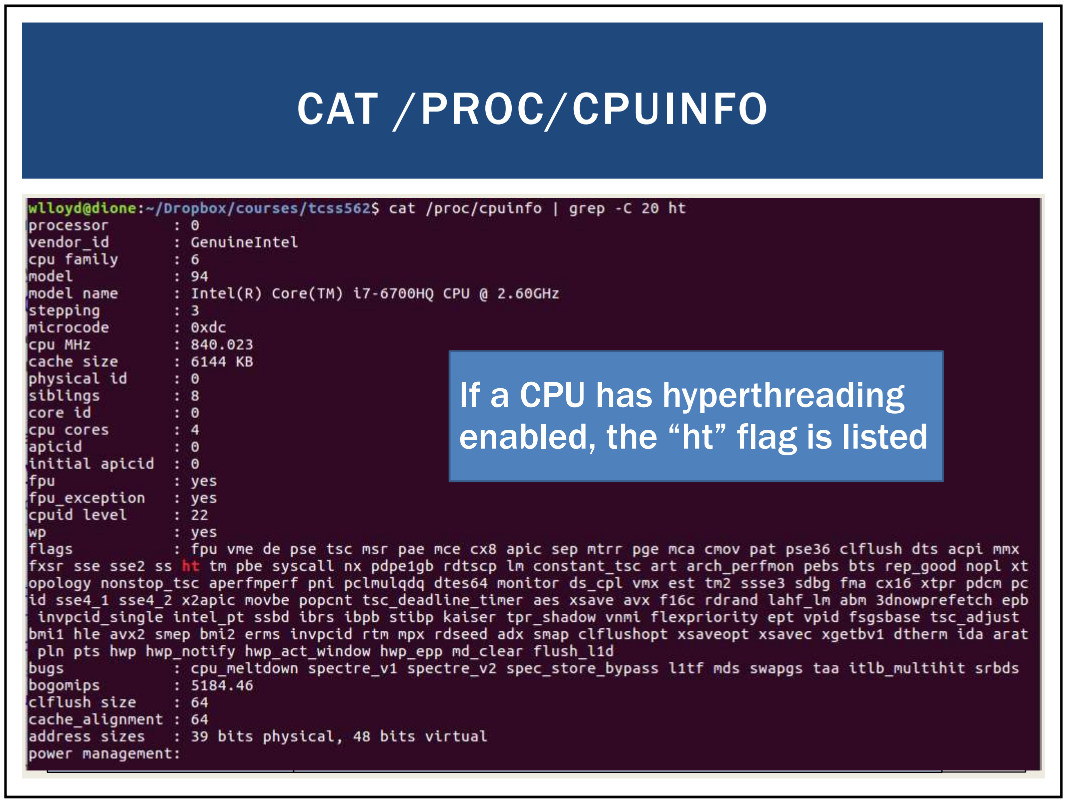
FEEDBACK FROM 9/30

- <u>***How should we achieve hyper threading?***</u>

- Hyperthreading 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 hyperthread
- On CPUs with hyperthreading, every CPU core has two hyperthreads
- To the operating system they are seen as full-featured independent CPU cores

October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.4



CAT /PROC/CPUINFO

```
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 hyperthreading enabled, the "ht" flag is listed

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

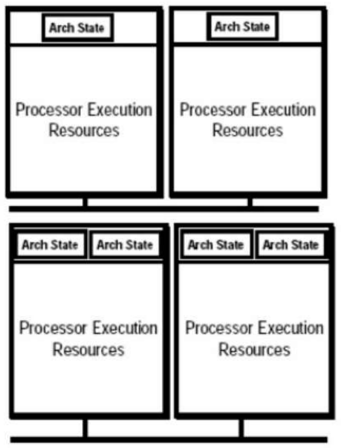
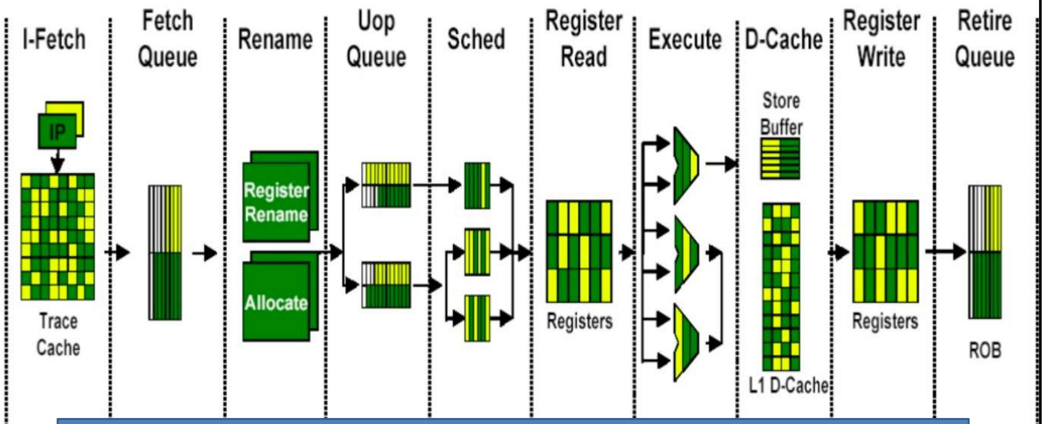

Arch State
Arch State
Processor Execution Resources
Processor Execution Resources
Arch State
Arch State
Arch State
Arch State
Processor Execution Resources
Processor Execution Resources

Figure 3: Processors with Hyper-Threading Technology

Execution Pipeline



Each processor core consists of multiple stages

Hyperthreading is the idea to share the physical stages of a CPU core to execute *two* instructions at once

L2.7

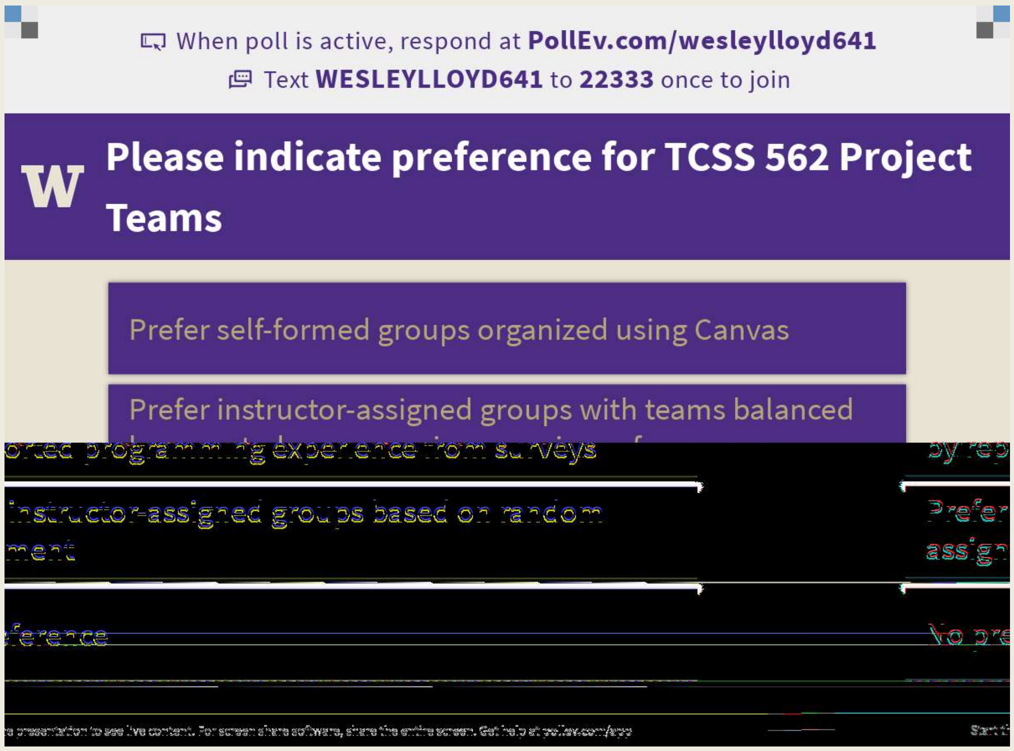
FEEDBACK - 2

- ***When should we use hyper threading if we can, and when should not?***
  - Disabling hyperthreading will reduce the number of CPU cores exposed to the operating system in half
  - Hyperthreading 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

FEEDBACK - 3

- ***What is infrastructure?***
  - Generic term referring to computer hardware used to host a cloud service
  - Infrastructure can consist of virtual machines, micro virtual machines, containers, functions
  - Infrastructure in the cloud suffers from sharing problems
  - How do we share resources among users?
    - Without suffering performance losses from resource contention
    - Without risking exposure of personal information to others

October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.9

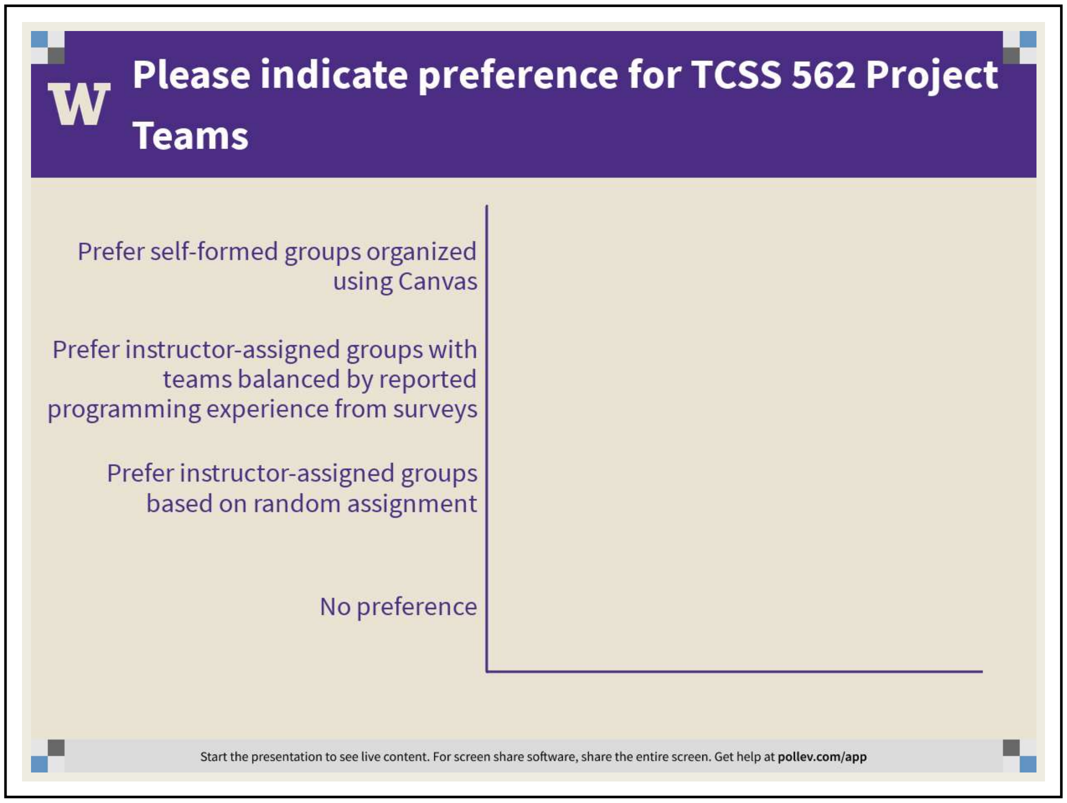


When poll is active, respond at **PollEv.com/wesleylloyd641**
Text **WESLEYLLOYD641** to **22333** once to join

Please indicate preference for TCSS 562 Project Teams

- Prefer self-formed groups organized using Canvas
- Prefer instructor-assigned groups with teams balanced
  orted programming experience from surveys
- instructor-assigned groups based on random
  ment
- ference

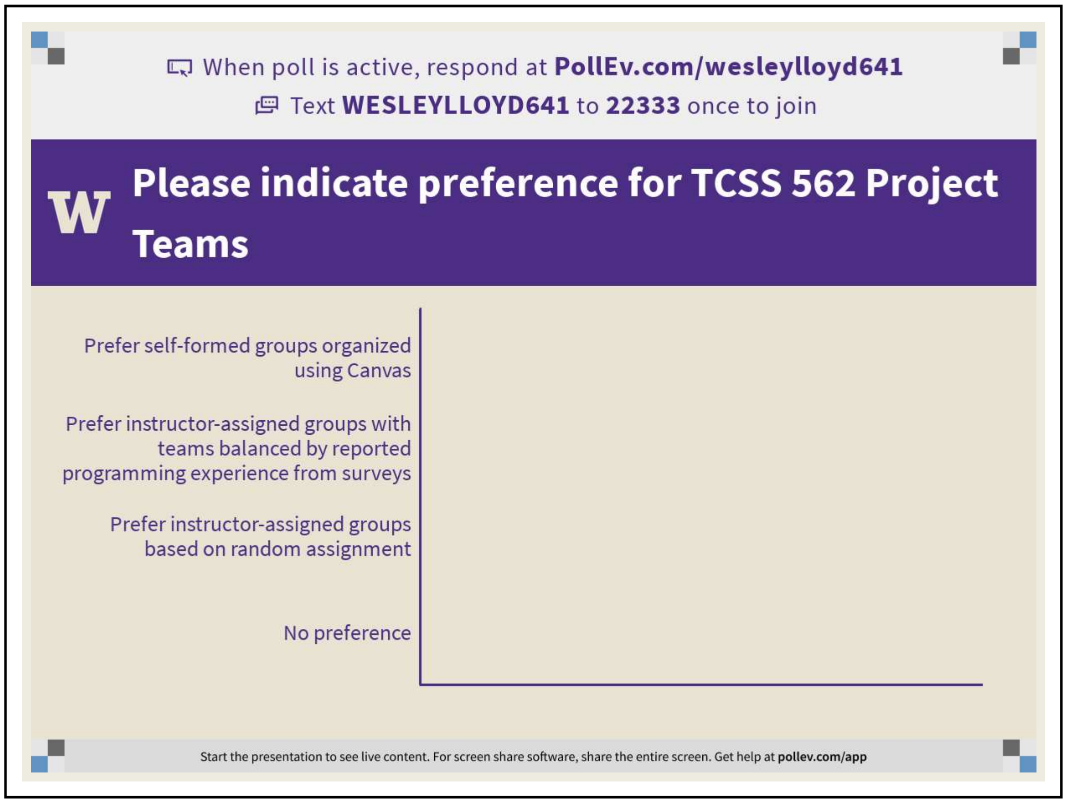
e presentation to see live content. For screen share software, share the entire screen. Get help at pollev.com/app



Please indicate preference for TCSS 562 Project Teams

- Prefer self-formed groups organized using Canvas
- Prefer instructor-assigned groups with teams balanced by reported programming experience from surveys
- Prefer instructor-assigned groups based on random assignment
- No preference

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

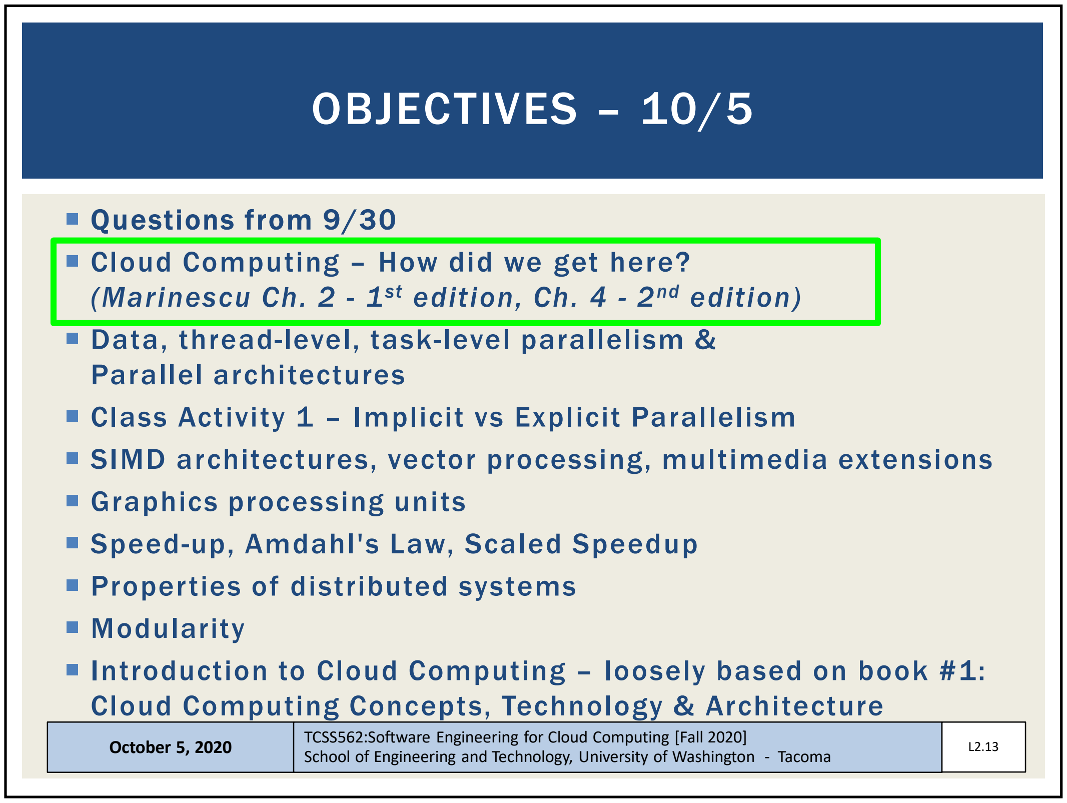


When poll is active, respond at **PollEv.com/wesleylloyd641**
Text **WESLEYLLOYD641** to **22333** once to join

Please indicate preference for TCSS 562 Project Teams

- Prefer self-formed groups organized using Canvas
- Prefer instructor-assigned groups with teams balanced by reported programming experience from surveys
- Prefer instructor-assigned groups based on random assignment
- No preference

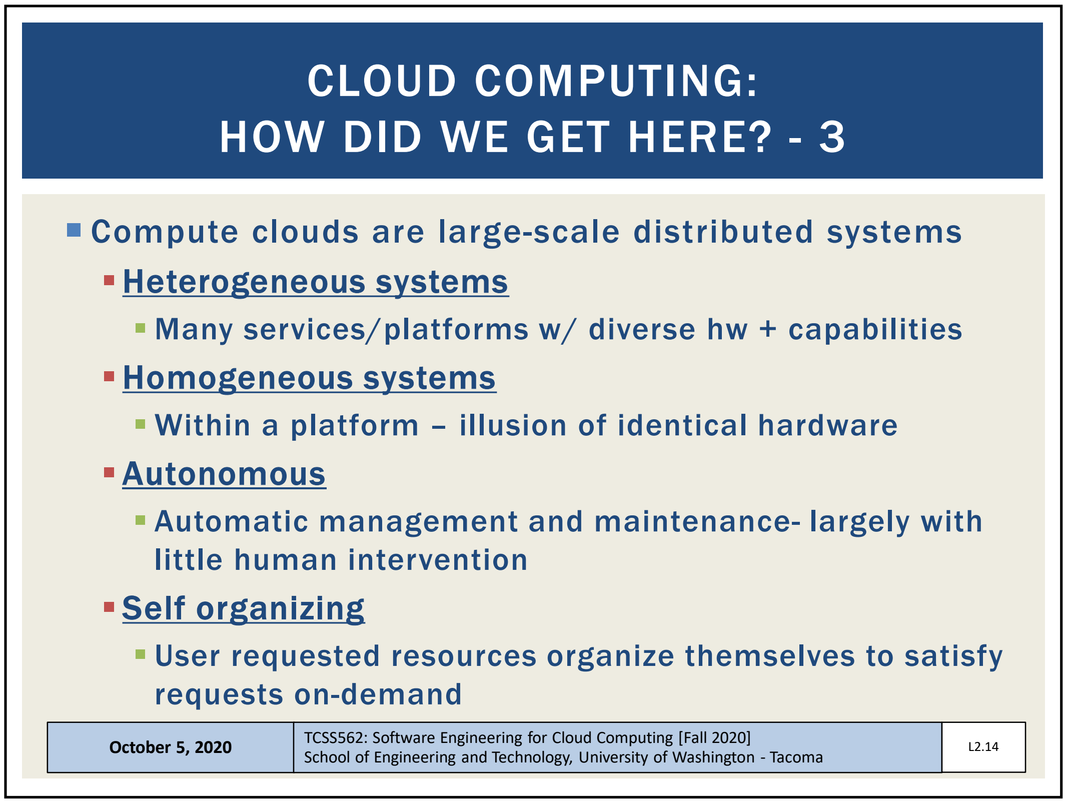
Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**



OBJECTIVES – 10/5

- Questions from 9/30
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism &
  Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1:
  Cloud Computing Concepts, Technology & Architecture

October 5, 2020 | TCSS562:Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma | L2.13



CLOUD COMPUTING:
HOW DID WE GET HERE? - 3

- 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

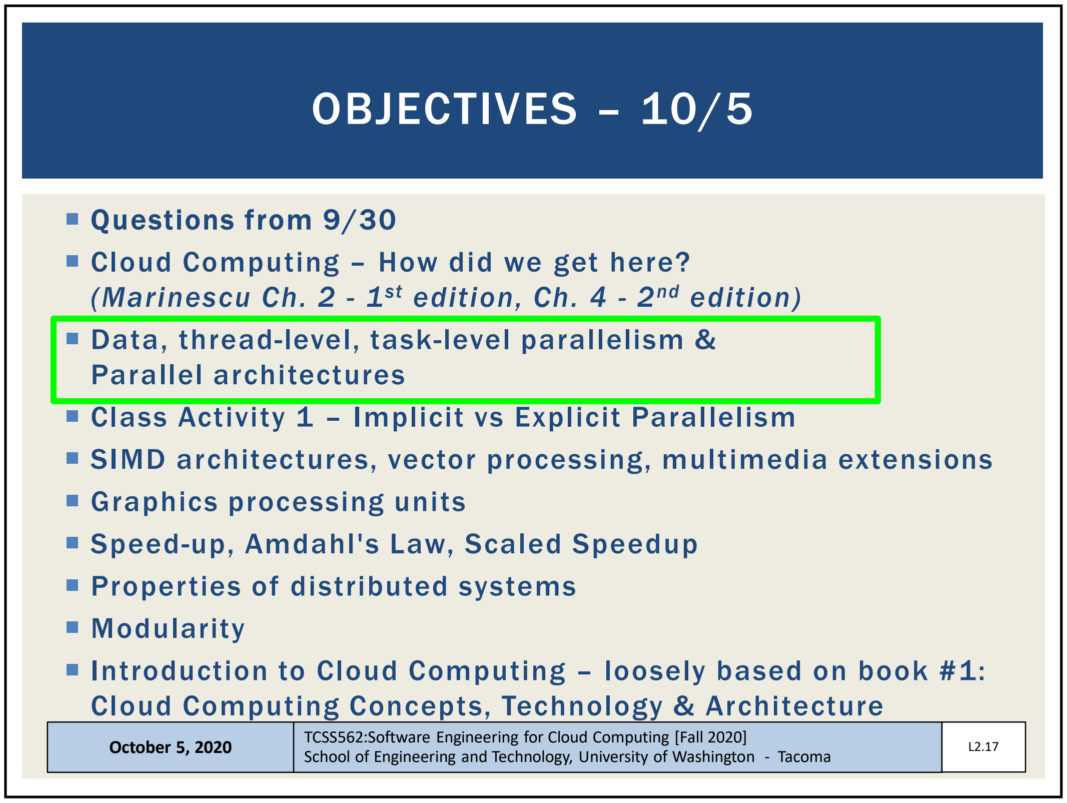
October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020]
School of Engineering and Technology, University of Washington - Tacoma | L2.14

CLOUD COMPUTING: HOW DID WE GET HERE?

- Compute clouds are large-scale distributed systems
- Infrastructure-as-a-Service (IaaS)
  - 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?

HARDWARE HETEROGENEITY

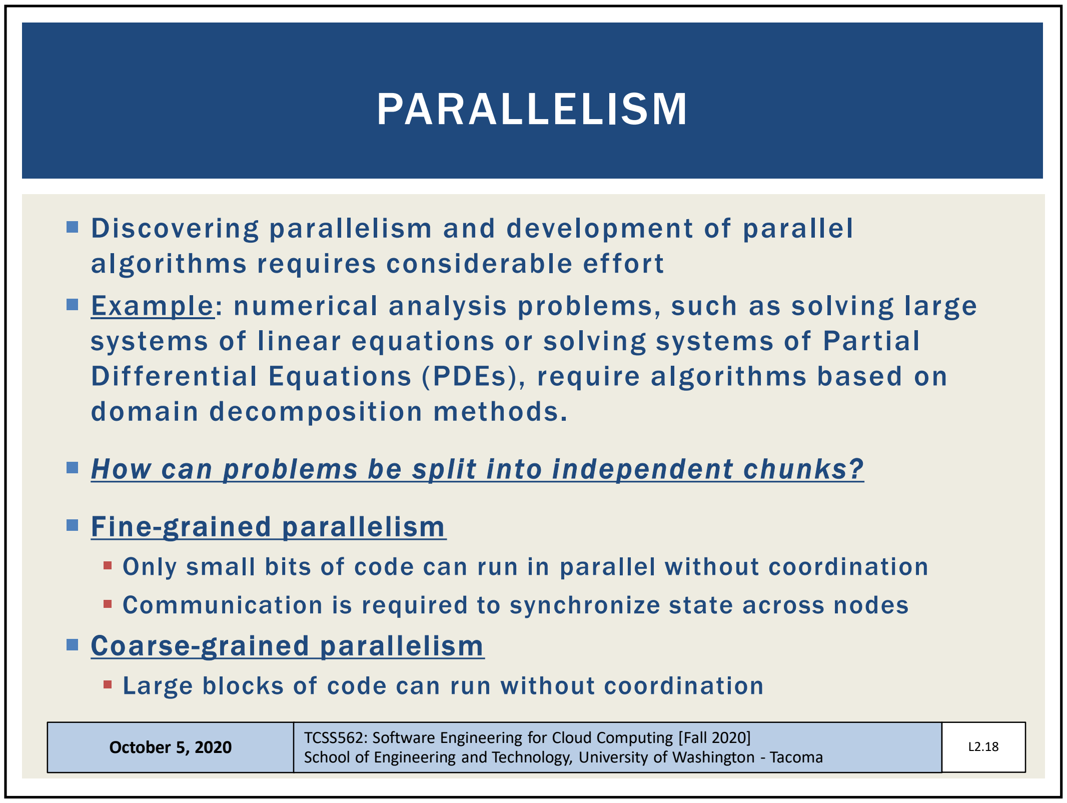
- If providing IaaS, what are advantages/ disadvantages of using homogeneous hardware?
  - Easier to provide same quality of service to end users
    - Less performance variance
    - Components with variable performance: CPUs, memory (speed differences), disks (SSDs, HDDs), network interfaces (caches?)
  - 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?



OBJECTIVES – 10/5

- Questions from 9/30
- Cloud Computing – How did we get here? *(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)*
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

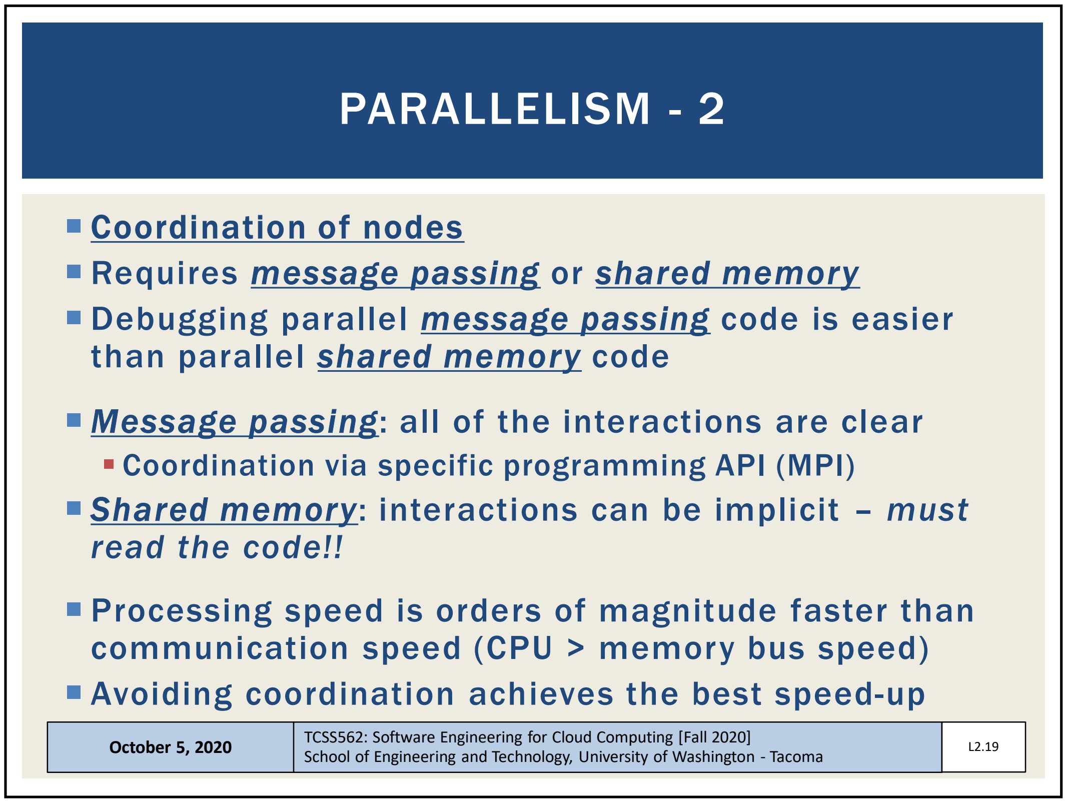
October 5, 2020 | TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.17



PARALLELISM

- Discovering parallelism and development of parallel algorithms requires considerable effort
- <u>Example</u>: 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.
- ***<u>How can problems be split into independent chunks?</u>***
- <u>Fine-grained parallelism</u>
  - Only small bits of code can run in parallel without coordination
  - Communication is required to synchronize state across nodes
- <u>Coarse-grained parallelism</u>
  - Large blocks of code can run without coordination

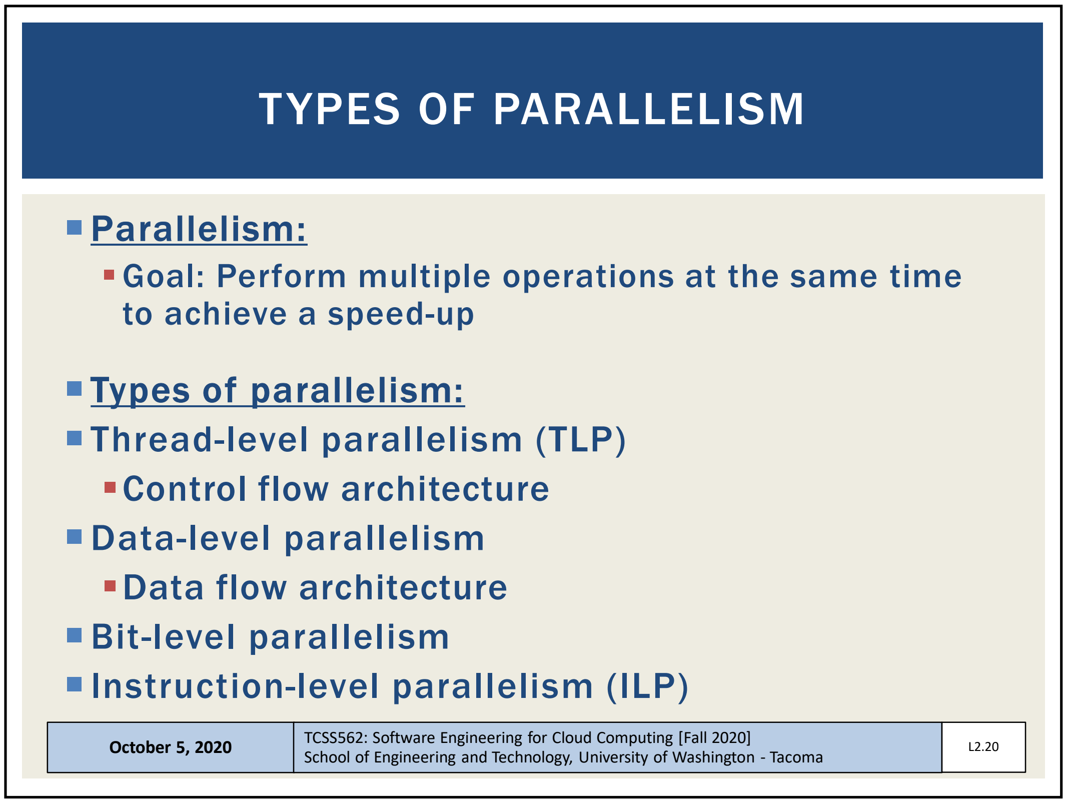
October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.18



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 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.19



TYPES OF PARALLELISM

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

October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.20

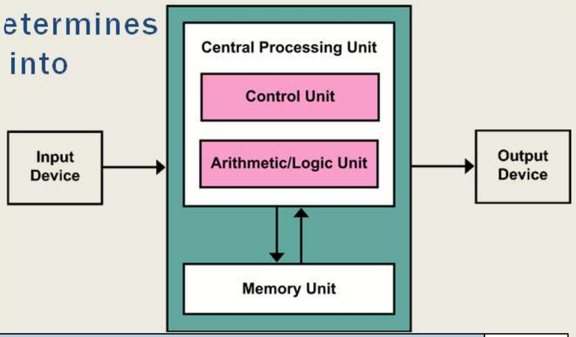
THREAD LEVEL PARALLELISM (TLP)

- 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…***

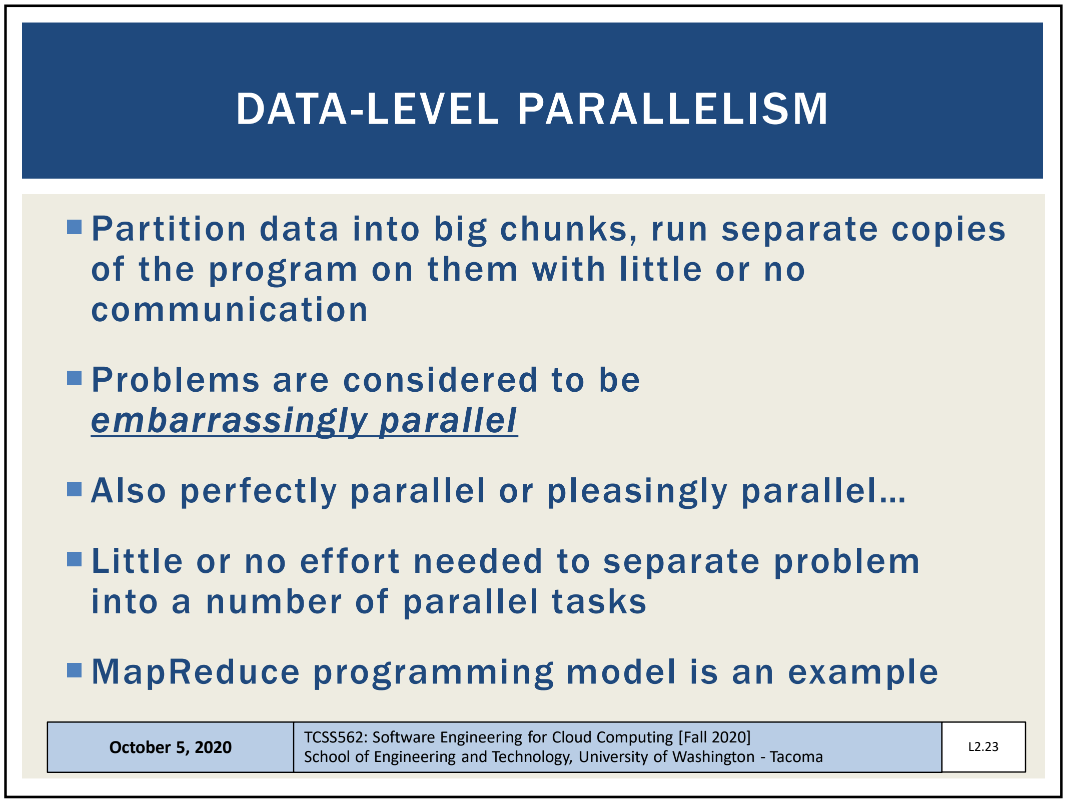
October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.21

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

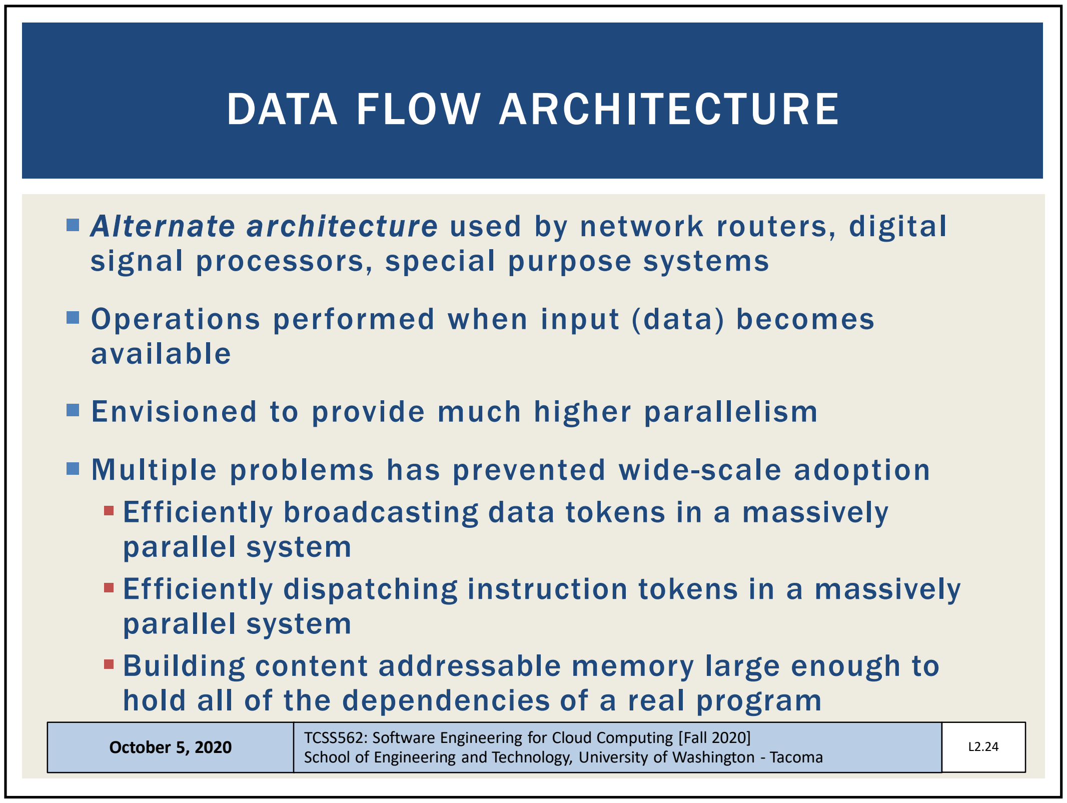


October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.22



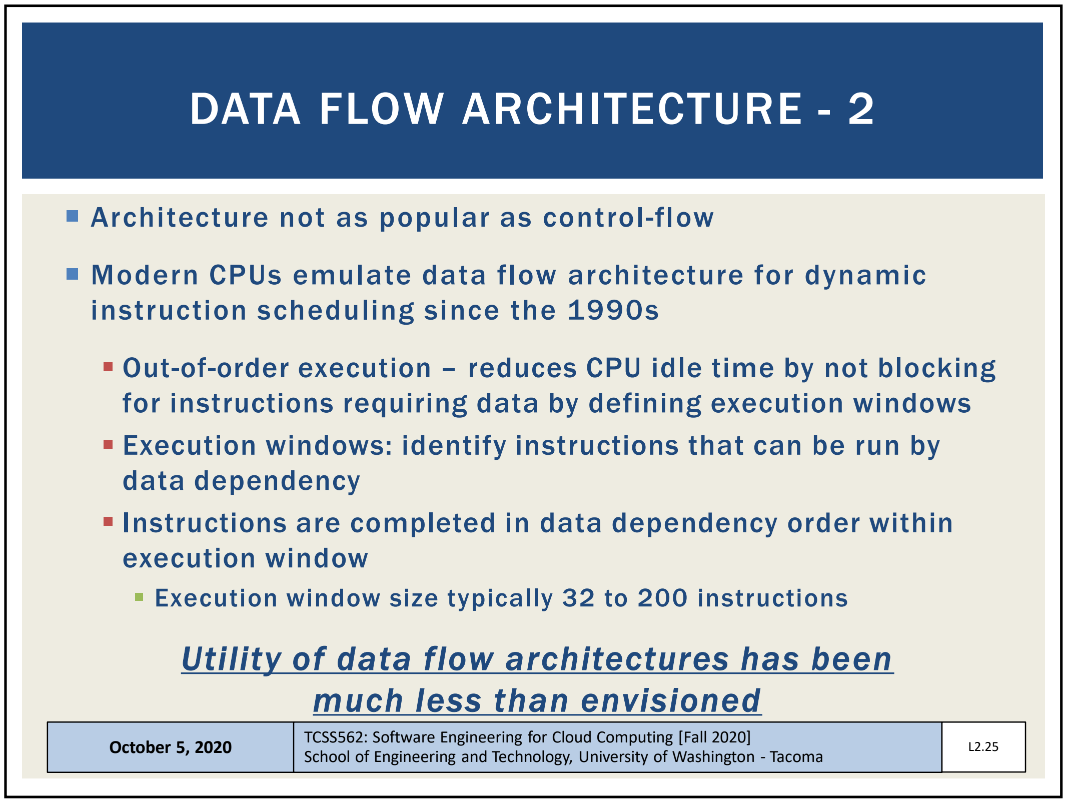
DATA-LEVEL PARALLELISM

- 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



DATA FLOW ARCHITECTURE

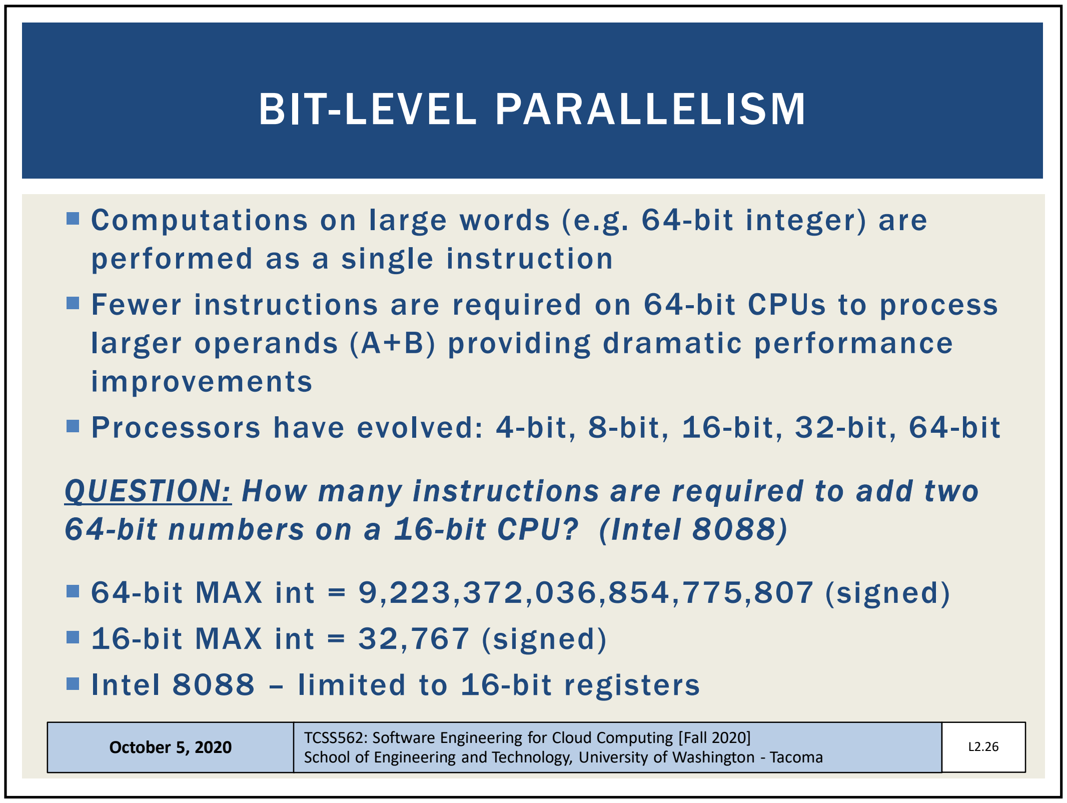
- *Alternate architecture* used by network routers, digital signal processors, special purpose systems
- Operations performed when input (data) becomes available
- Envisioned to provide much higher parallelism
- Multiple problems has prevented wide-scale adoption
  - Efficiently broadcasting data tokens 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



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



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

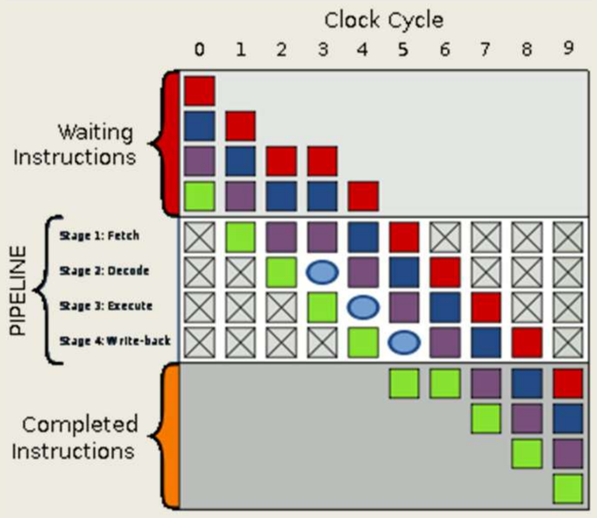
QUESTION: ***How many instructions are required to add two 64-bit numbers on a 16-bit CPU? (Intel 8088)***

- 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

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*

CPU PIPELINING



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!

OBJECTIVES – 10/5

- Questions from 9/30
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

CLASS ACTIVITY 1

- We will form groups of ~3 and go into breakout rooms
- Each group will complete a Google Doc worksheet
- Add names to Google Doc as they appear in Canvas
- Once completed, one person should submit a PDF of the Google Doc to Canvas
- Instructor will score all group members based on the uploaded PDF file
- To get started:
  - Log into your UW Google Account
  - Link to shared Google Drive
  - Follow link:
    https://tinyurl.com/yy9jlz3y

IMPLICIT PARALLELISM

- Applies to:
- Advantages:
- Disadvantages:

EXPLICIT PARALLELISM

- Applies to:
- Advantages:
- Disadvantages:

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)

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?

OBJECTIVES – 10/5

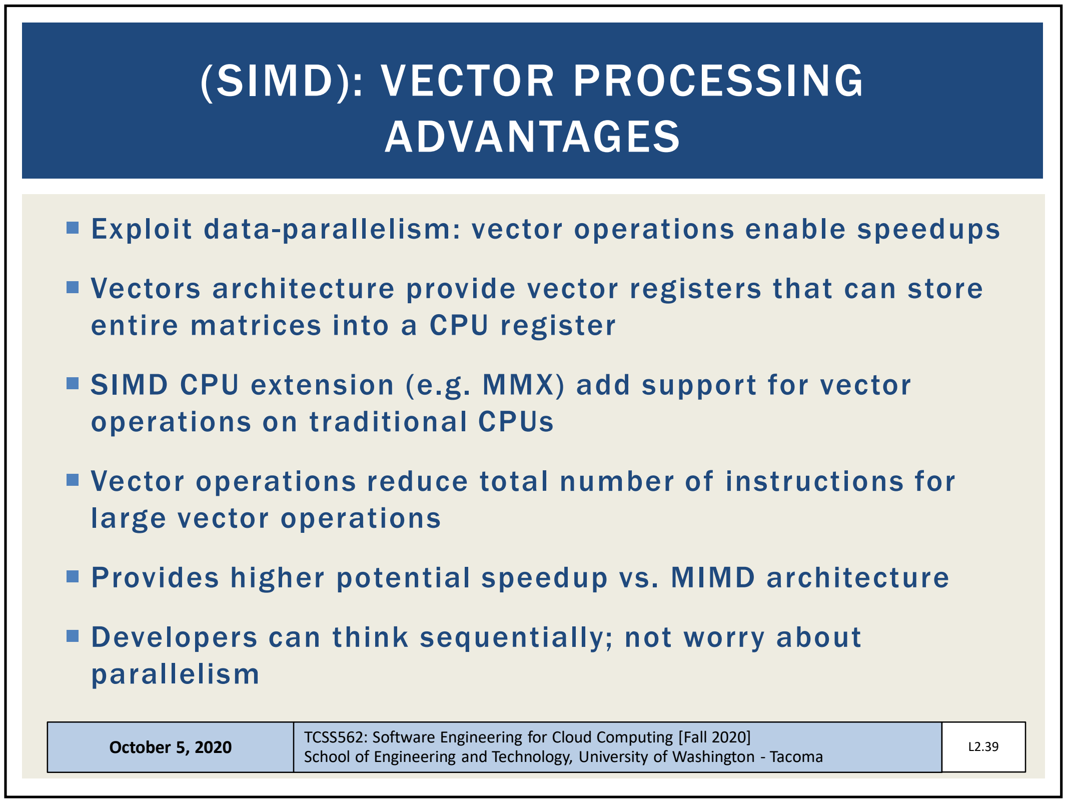
- Questions from 9/30
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

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

FLYNN'S TAXONOMY

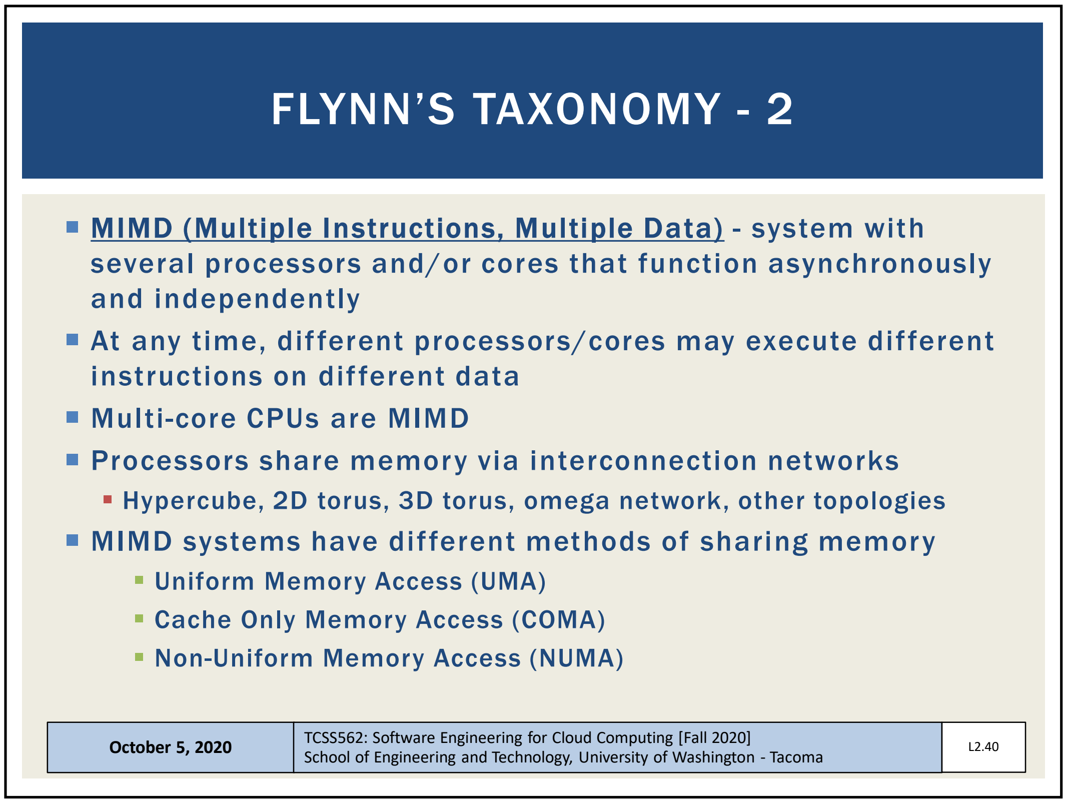
- SISD (Single Instruction Single Data)
  Scalar architecture with one processor/core.
  - Individual cores of modern multicore processors are "SISD"
- SIMD (Single Instruction, Multiple Data)
  Supports vector processing
  - When SIMD instructions are issued, operations on individual vector components 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



(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, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.39



FLYNN'S TAXONOMY - 2

- <u>MIMD (Multiple Instructions, Multiple Data)</u> - 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)

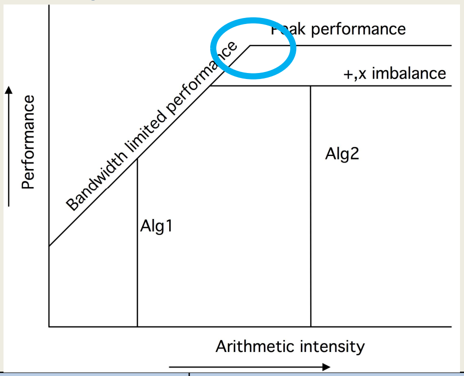
October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.40

ARITHMETIC INTENSITY

- <u>Arithmetic intensity</u>: 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*
- ***<u>High arithmetic Intensity:</u>***
  Programs with dense matrix operations scale up nicely (many calcs vs memory RW, supports lots of parallelism)
- ***<u>Low arithmetic intensity:</u>***
  Programs with sparse matrix operations do not scale well with problem size
  (memory RW becomes bottleneck, not enough ops!)

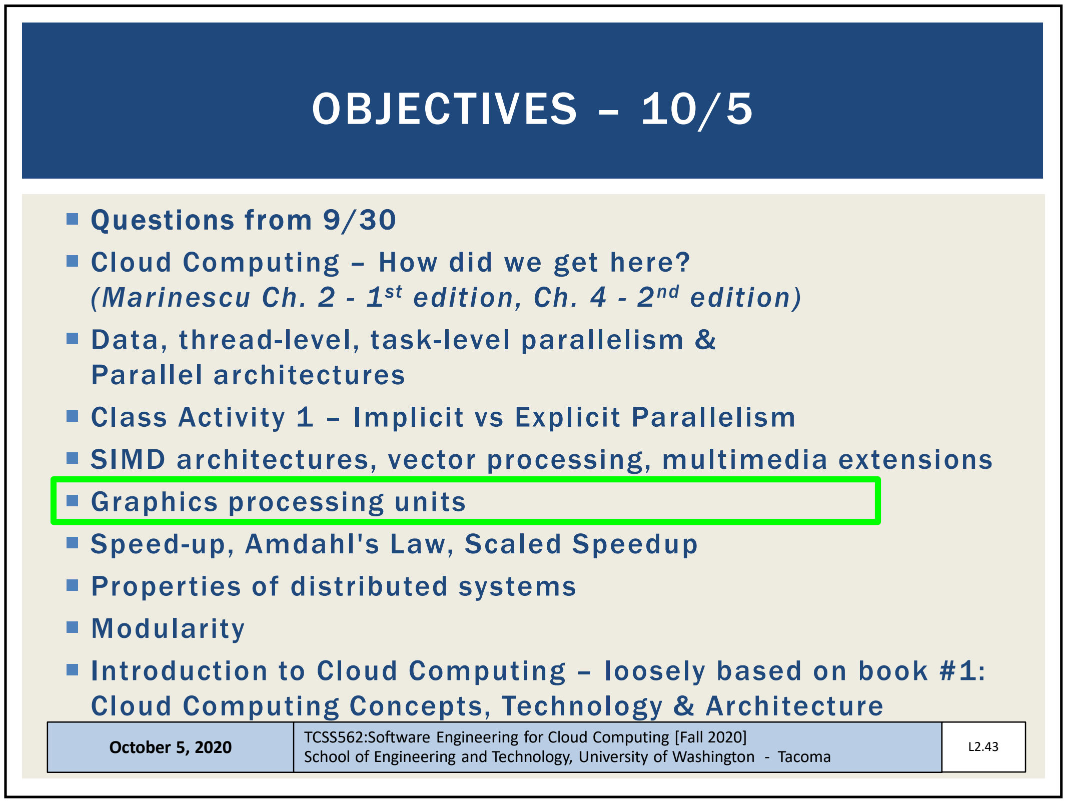
ROOFLINE MODEL

- 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)



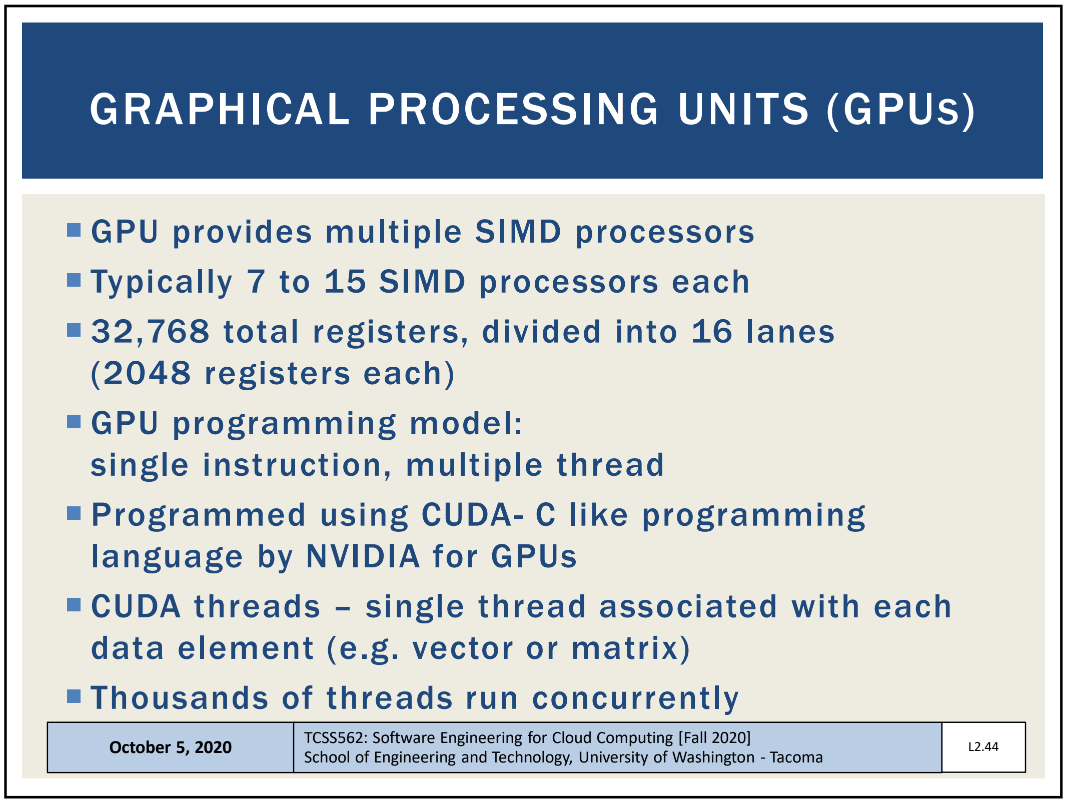
<u>Key take-aways:</u>
When a program’s has **low** Arithmetic Intensity, memory bandwidth limits performance..

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



OBJECTIVES – 10/5

- Questions from 9/30
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism &
  Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1:
  Cloud Computing Concepts, Technology & Architecture



GRAPHICAL PROCESSING UNITS (GPUs)

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

OBJECTIVES – 10/5

- Questions from 9/30
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism &
  Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1:
  Cloud Computing Concepts, Technology & Architecture

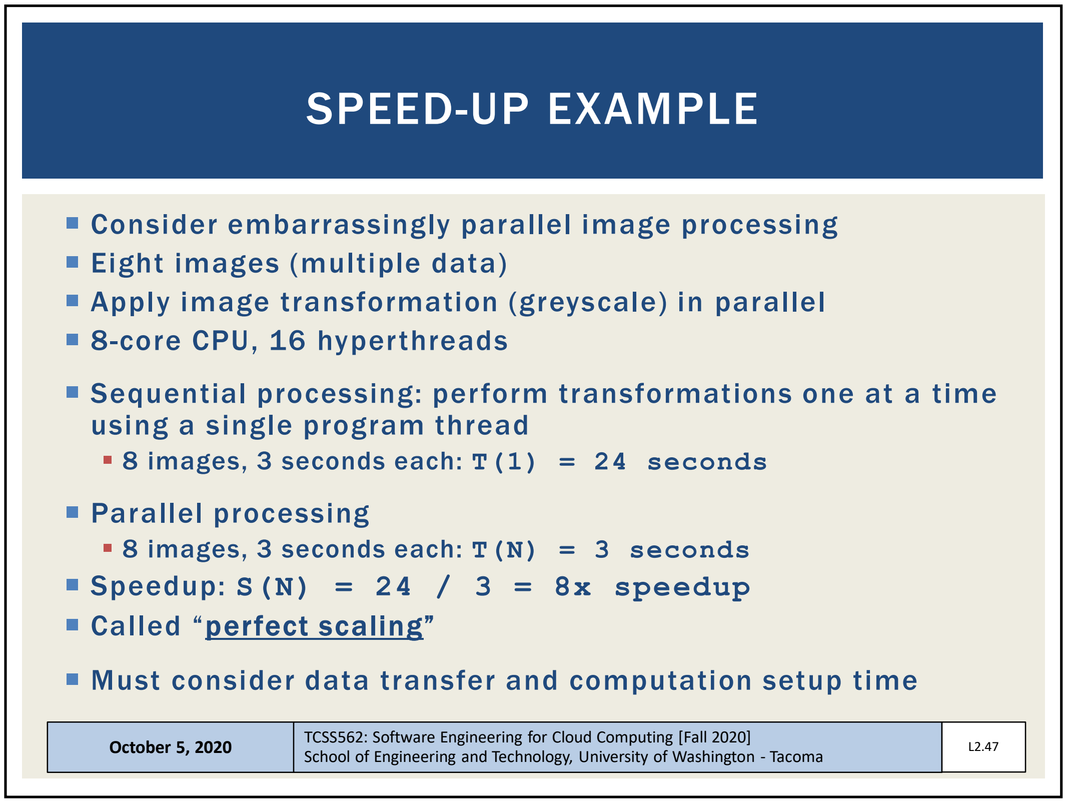
PARALLEL COMPUTING

- 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:

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

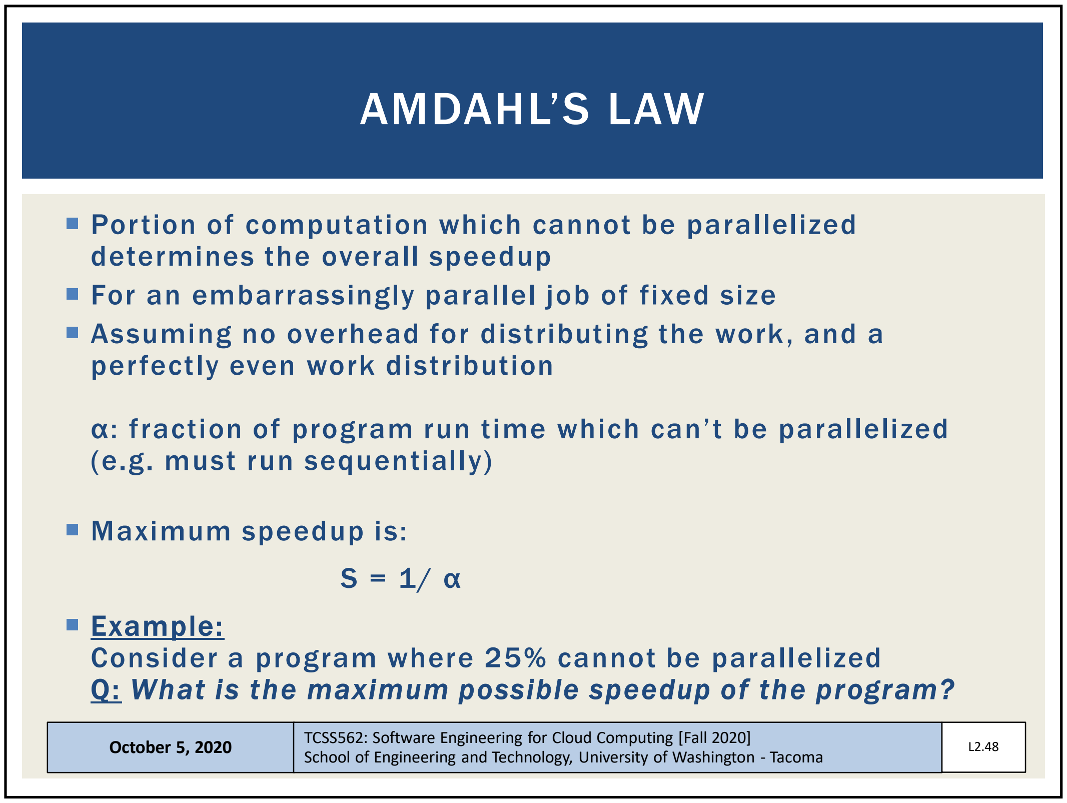
$T(1) \rightarrow$ execution time of total sequential computation
$T(N) \rightarrow$ execution time for performing N parallel computations in parallel



SPEED-UP EXAMPLE

- Consider embarrassingly parallel image processing
- Eight images (multiple data)
- Apply image transformation (greyscale) in parallel
- 8-core CPU, 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 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.47



AMDAHL'S LAW

- Portion of computation which cannot be parallelized determines the overall speedup
- For an embarrassingly parallel job of fixed size
- Assuming no overhead for distributing the work, and a perfectly even work distribution

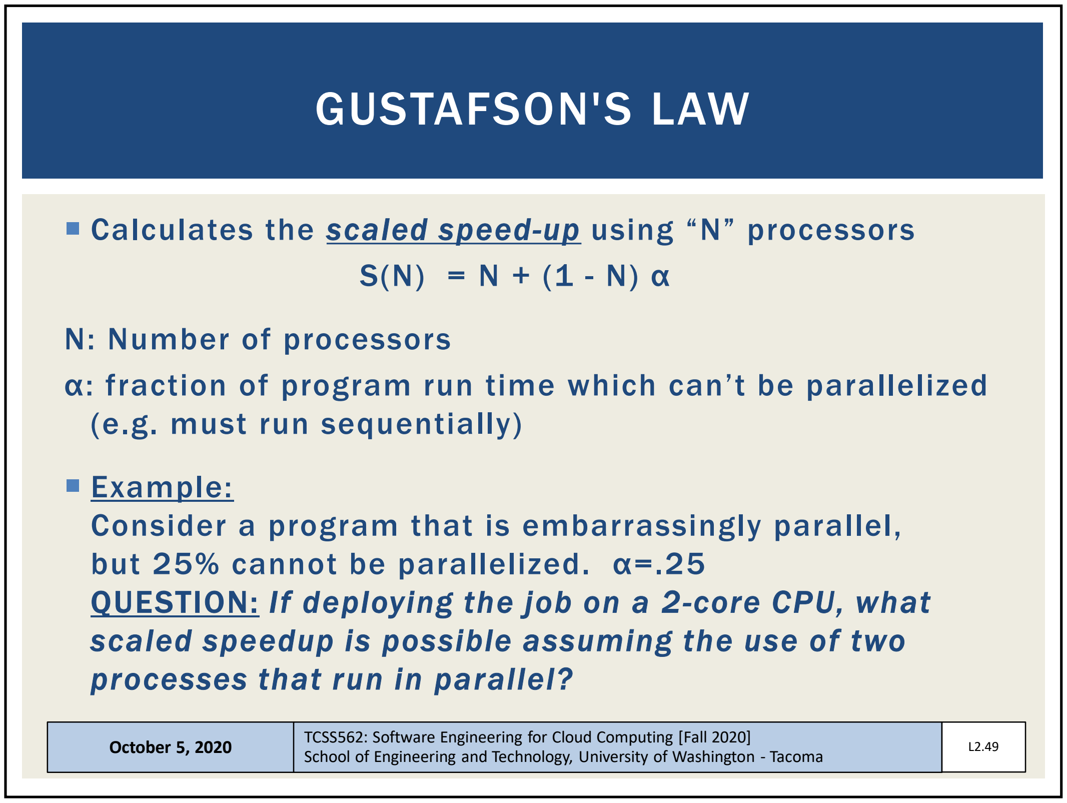
  α: fraction of program run time which can't be parallelized (e.g. must run sequentially)

- Maximum speedup is:

$$S = 1/\alpha$$

- **Example:**
  Consider a program where 25% cannot be parallelized
  Q: What is the maximum possible speedup of the program?

October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.48



GUSTAFSON'S LAW

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

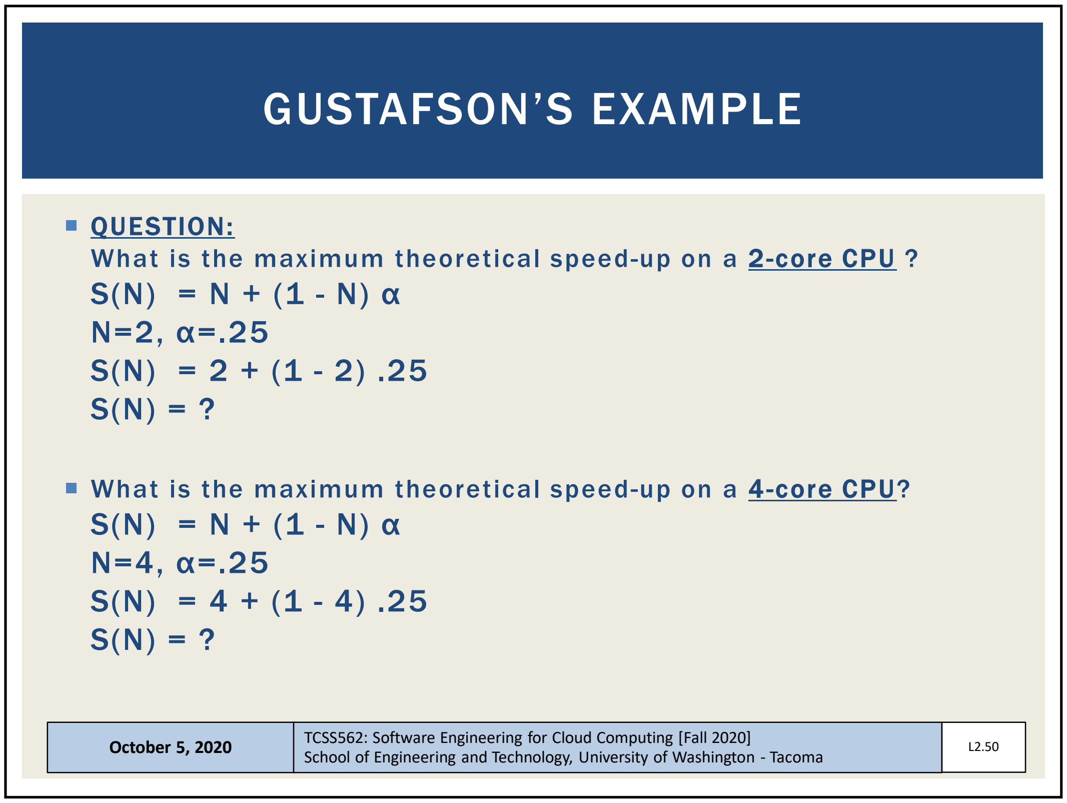
$$S(N) = N + (1 - N)\,\alpha$$

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 25% cannot be parallelized. α=.25
  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 5, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L2.49

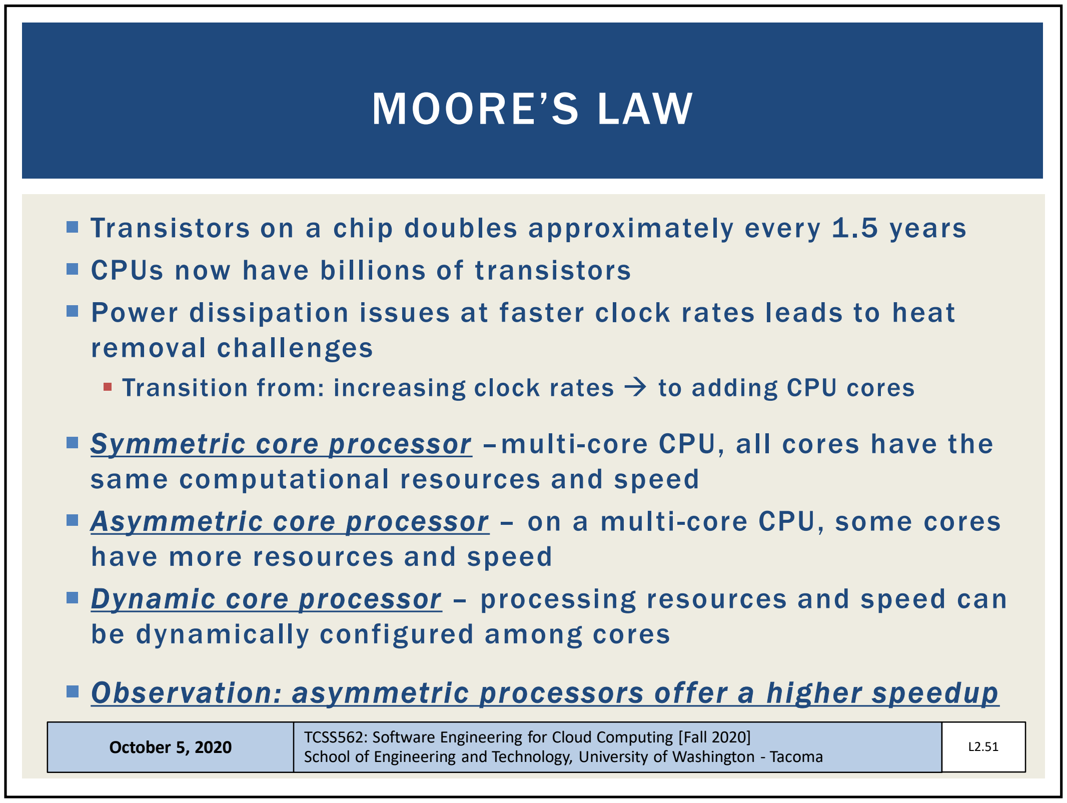


GUSTAFSON'S EXAMPLE

- **QUESTION:**
  What is the maximum theoretical speed-up on a **2-core CPU** ?
  $S(N) = N + (1 - N)\,\alpha$
  $N=2, \alpha=.25$
  $S(N) = 2 + (1 - 2)\,.25$
  $S(N) = ?$

- What is the maximum theoretical speed-up on a **4-core CPU**?
  $S(N) = N + (1 - N)\,\alpha$
  $N=4, \alpha=.25$
  $S(N) = 4 + (1 - 4)\,.25$
  $S(N) = ?$

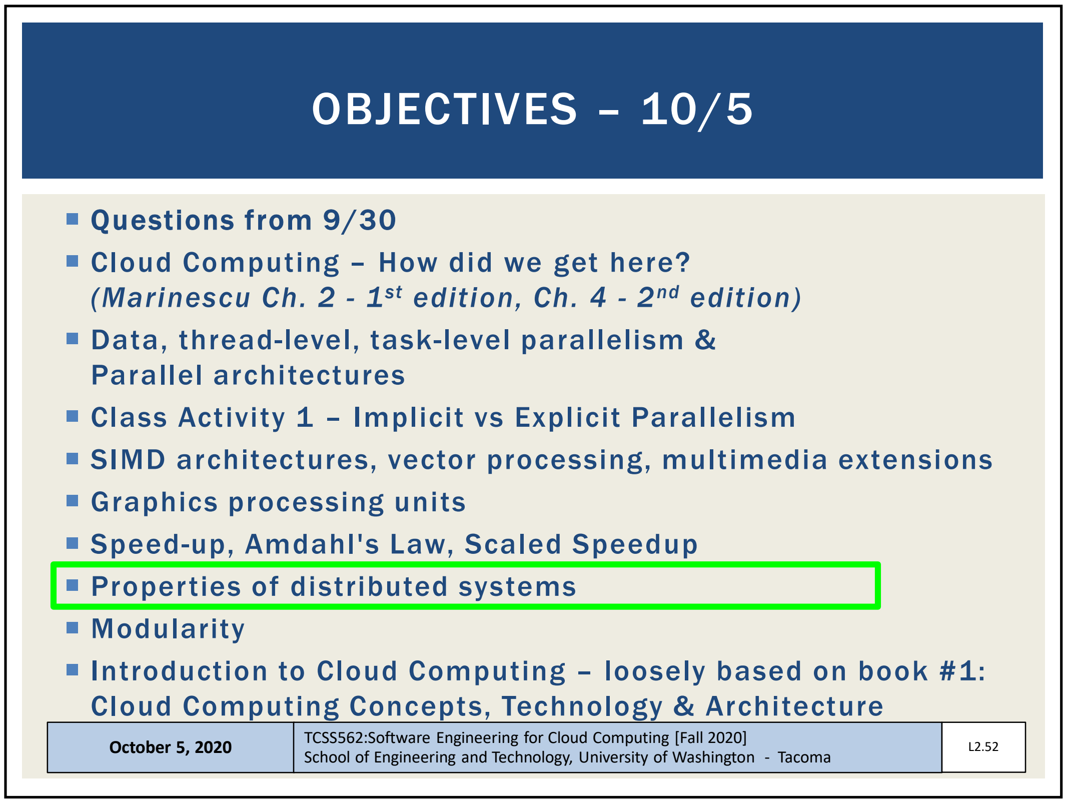
October 5, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L2.50



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***

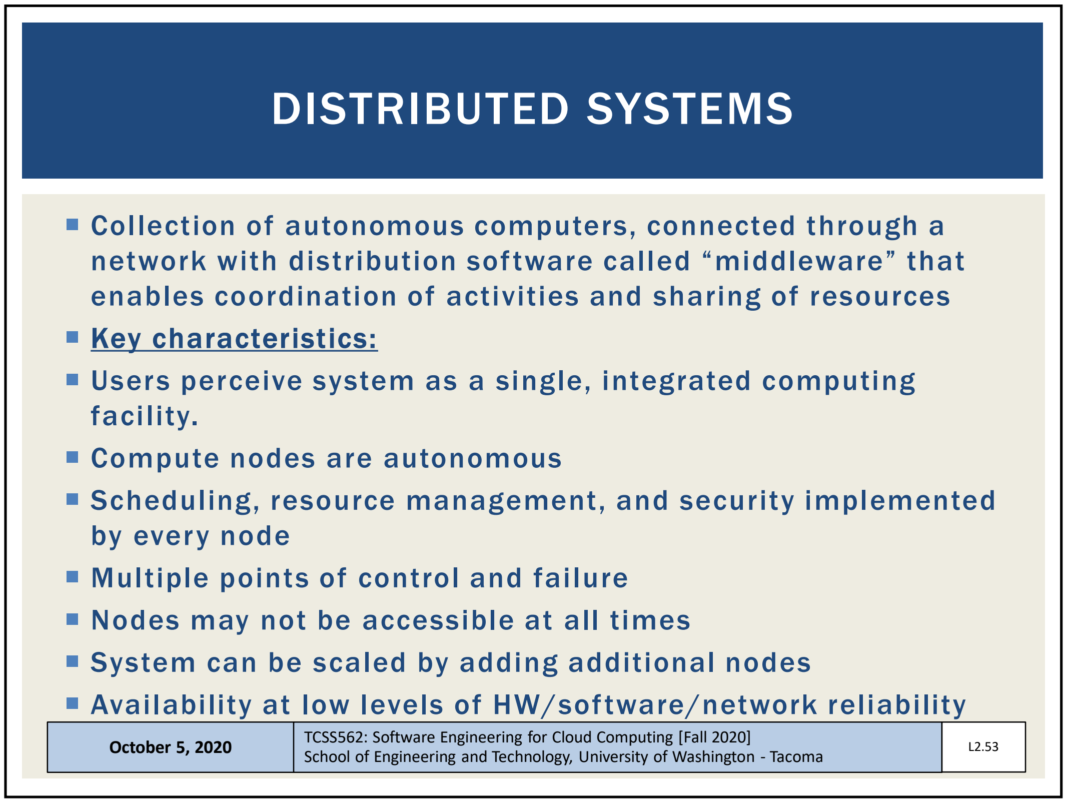
October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.51



OBJECTIVES – 10/5

- Questions from 9/30
- Cloud Computing – How did we get here? *(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)*
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

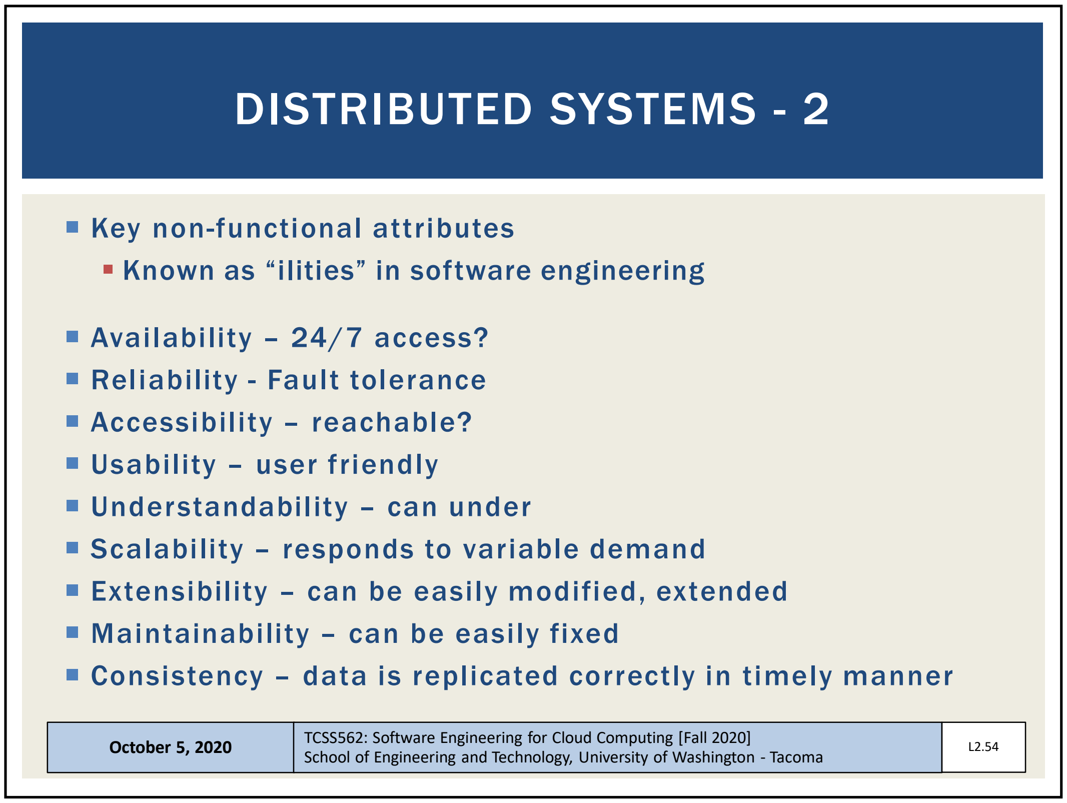
October 5, 2020 | TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.52



DISTRIBUTED SYSTEMS

- Collection of autonomous computers, connected through a network with distribution software called “middleware” that enables coordination of activities and sharing of resources
- <u>Key characteristics:</u>
- 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

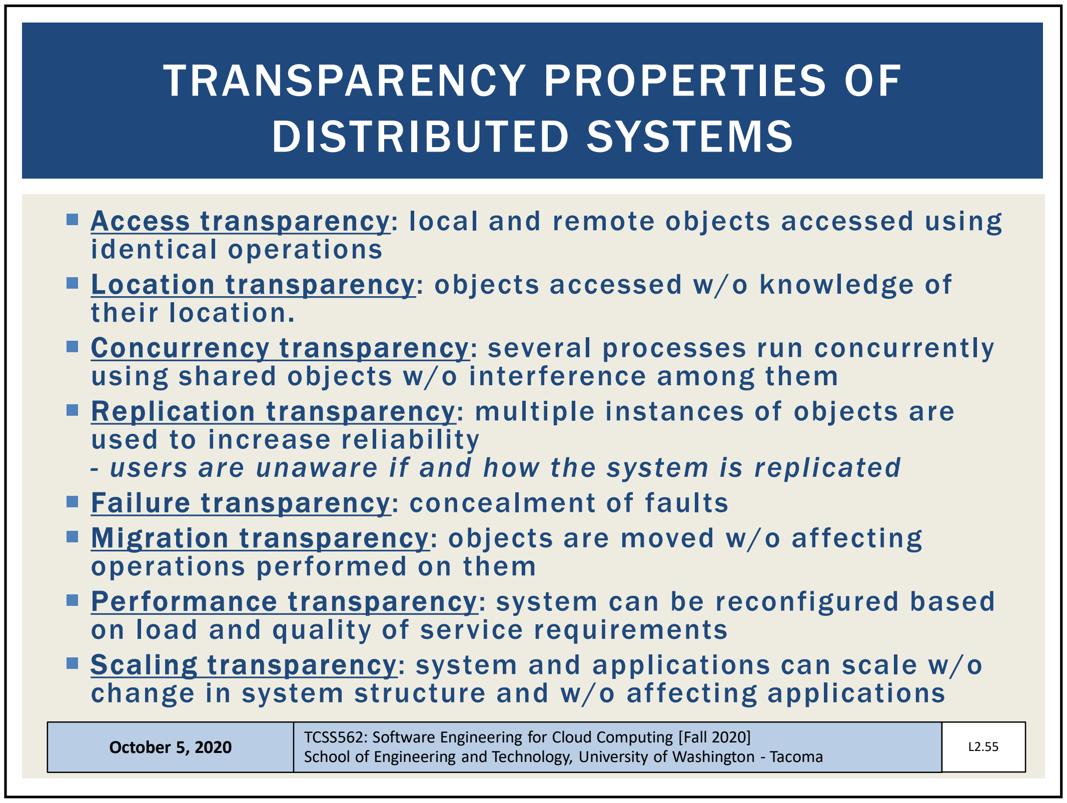
October 5, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L2.53



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

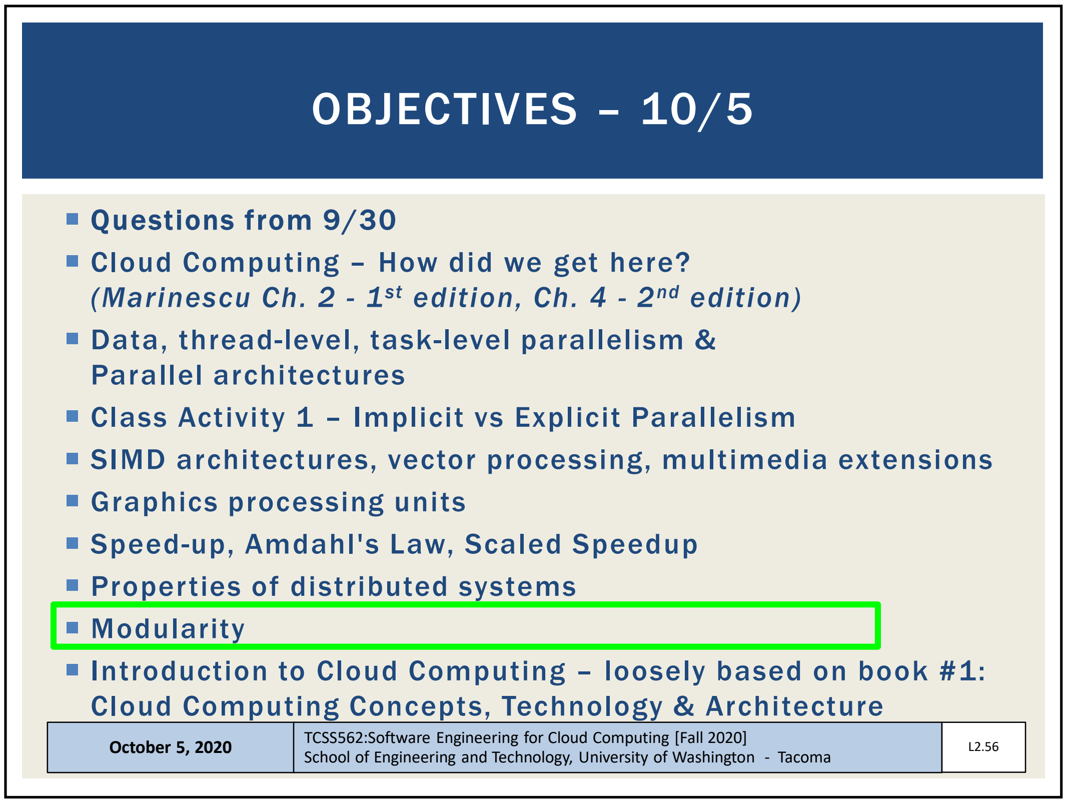
October 5, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L2.54



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

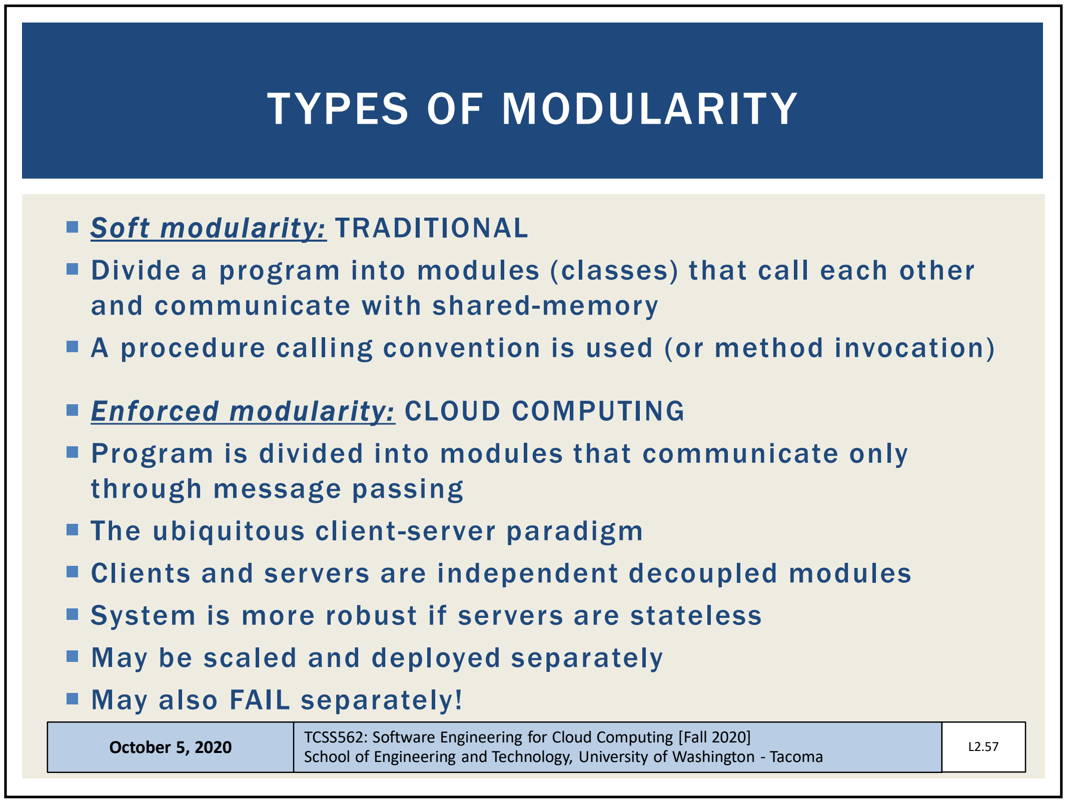
October 5, 2020	TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L2.55



OBJECTIVES – 10/5

- Questions from 9/30
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 5, 2020	TCSS562:Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma	L2.56

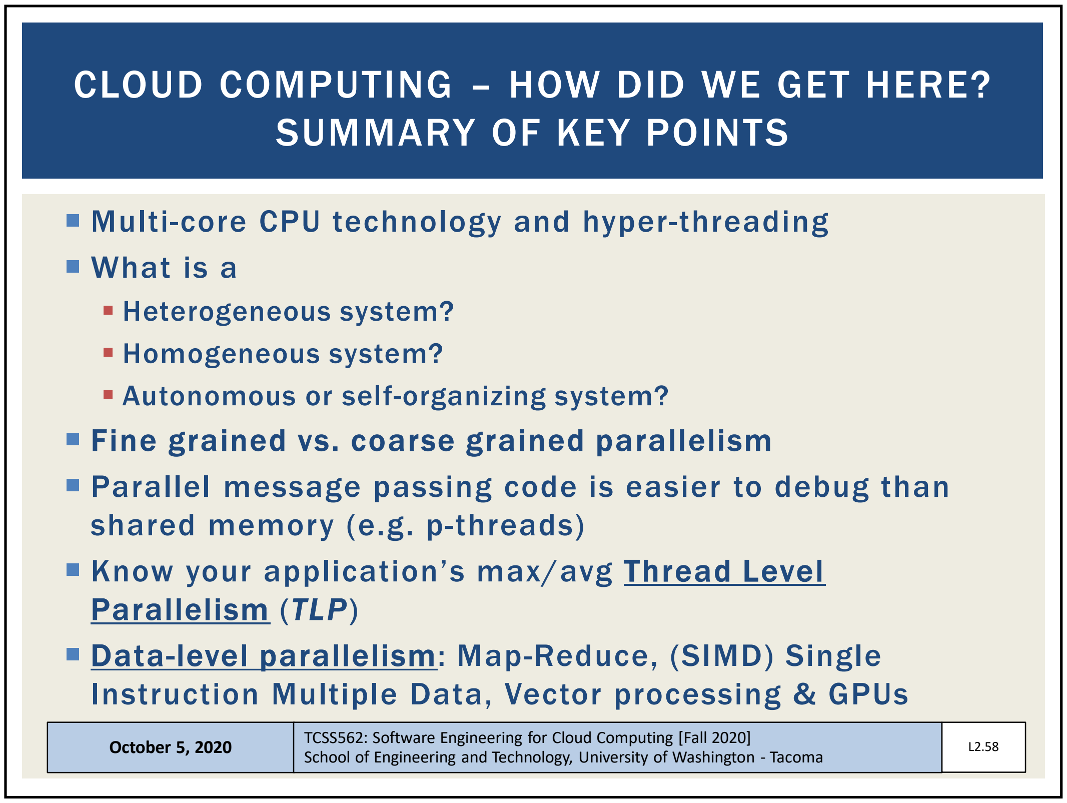


TYPES OF MODULARITY

- ***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!

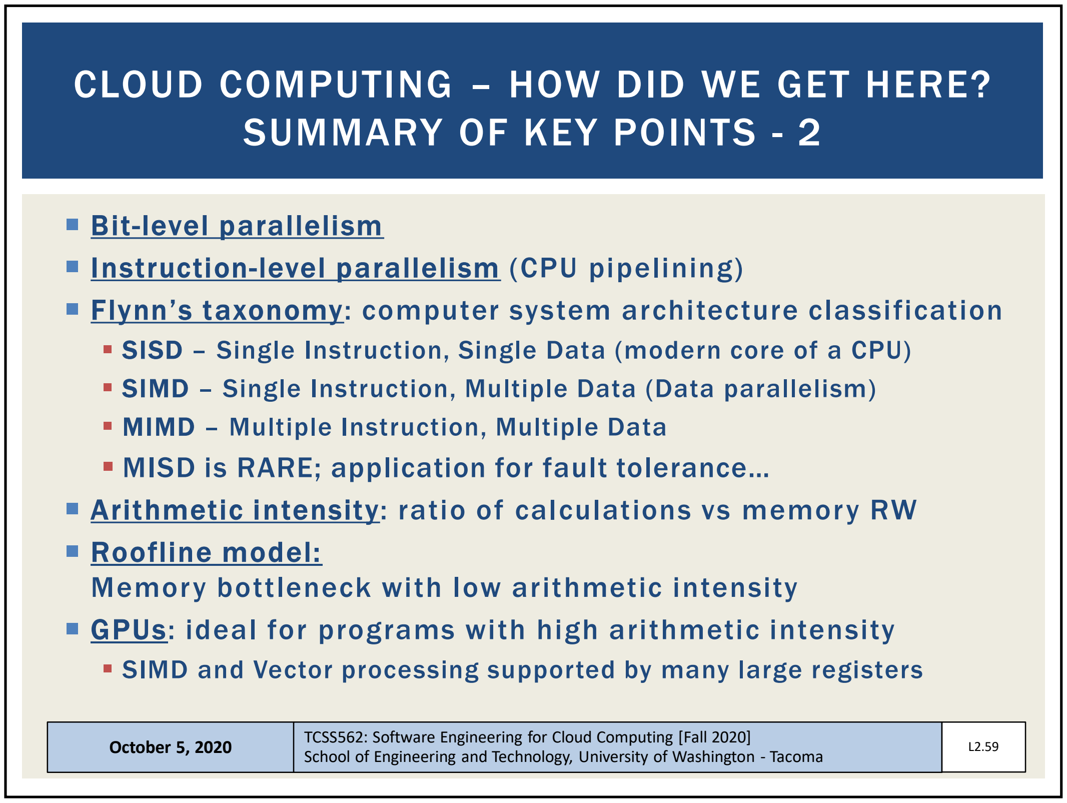
October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.57



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

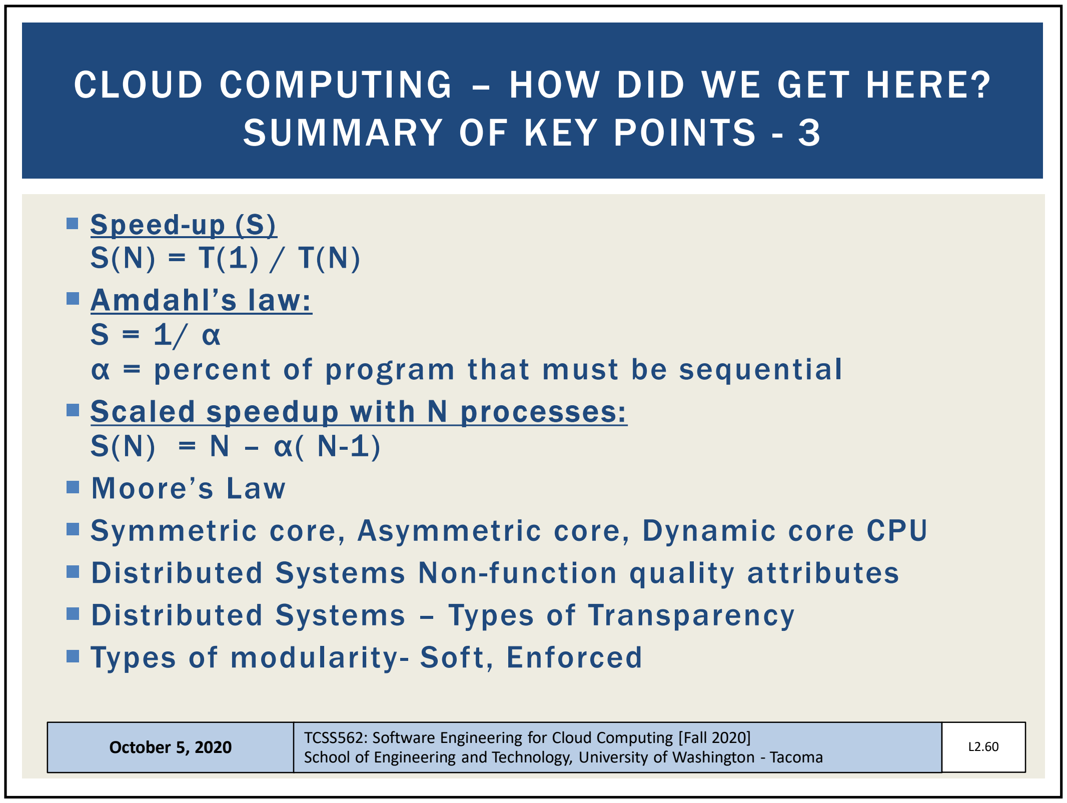
- Multi-core CPU technology and hyper-threading
- What is a
  - Heterogeneous system?
  - Homogeneous system?
  - Autonomous or self-organizing system?
- Fine grained vs. coarse grained parallelism
- Parallel message passing code is easier to debug than shared memory (e.g. p-threads)
- Know your application's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

October 5, 2020 | TCSS562: Software Engineering for Cloud Computing [Fall 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.58



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



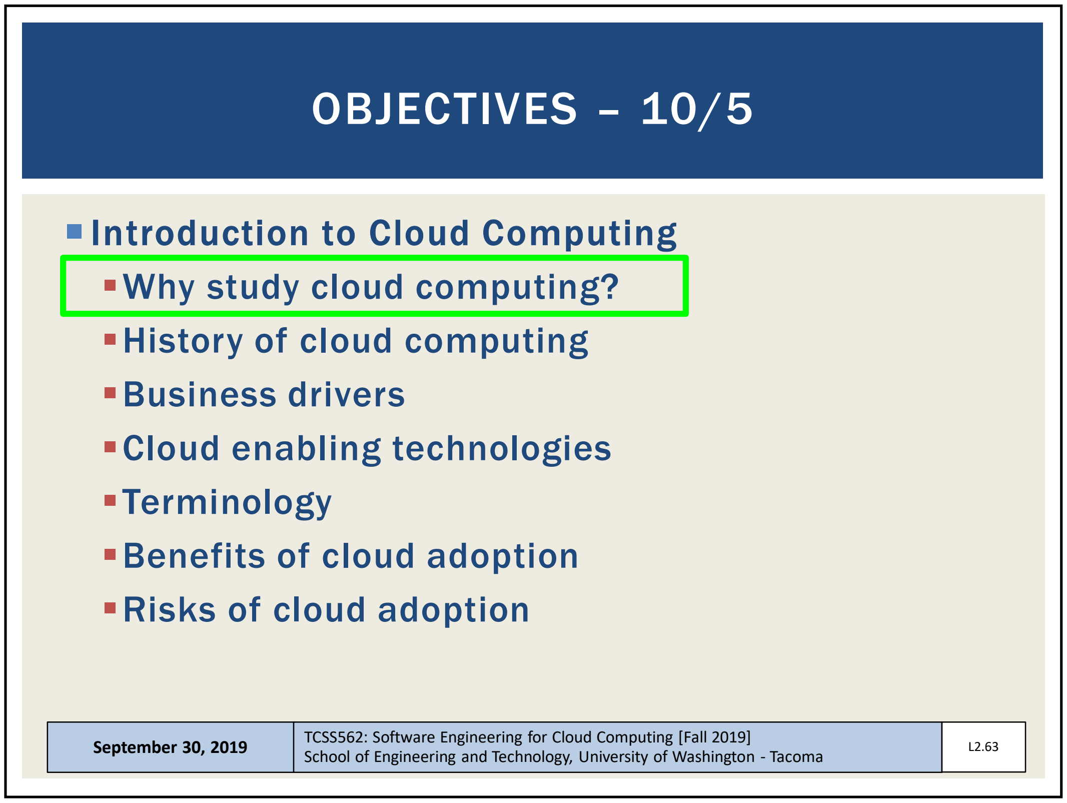
CLOUD COMPUTING – HOW DID WE GET HERE? SUMMARY OF KEY POINTS - 3

- **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

INTRODUCTION TO CLOUD COMPUTING

OBJECTIVES – 10/5

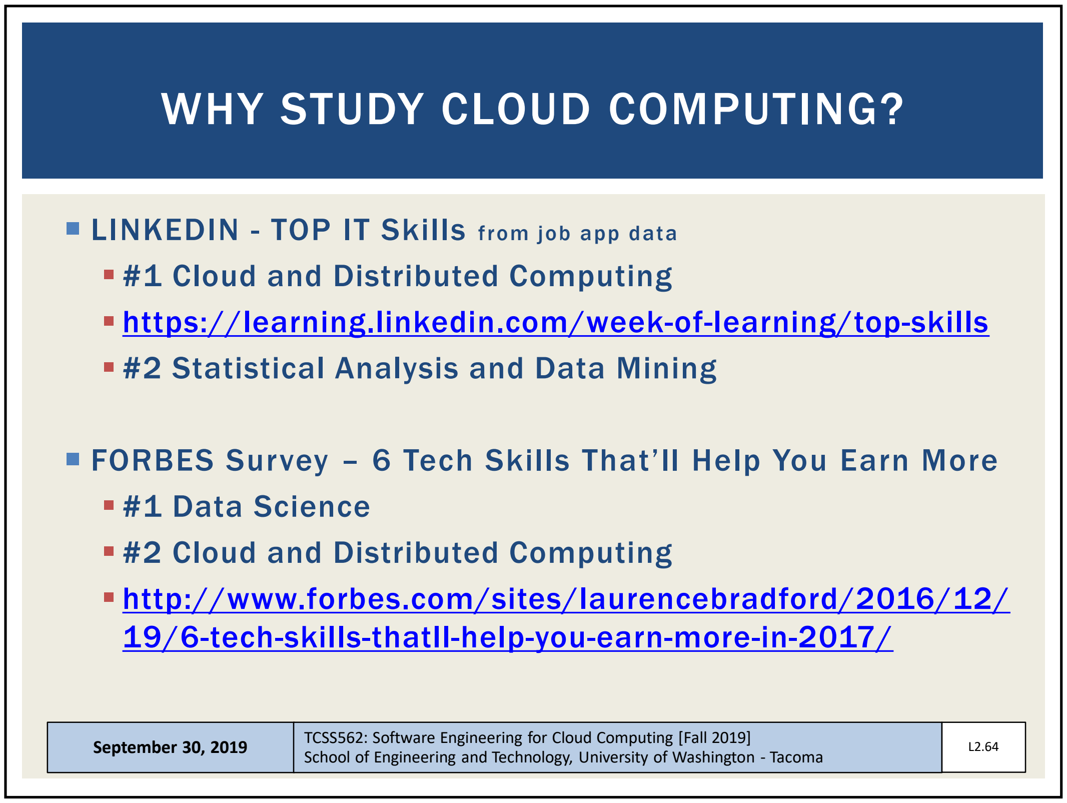
- Questions from 9/30
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism &
  Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1:
  Cloud Computing Concepts, Technology & Architecture



OBJECTIVES – 10/5

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

September 30, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L2.63



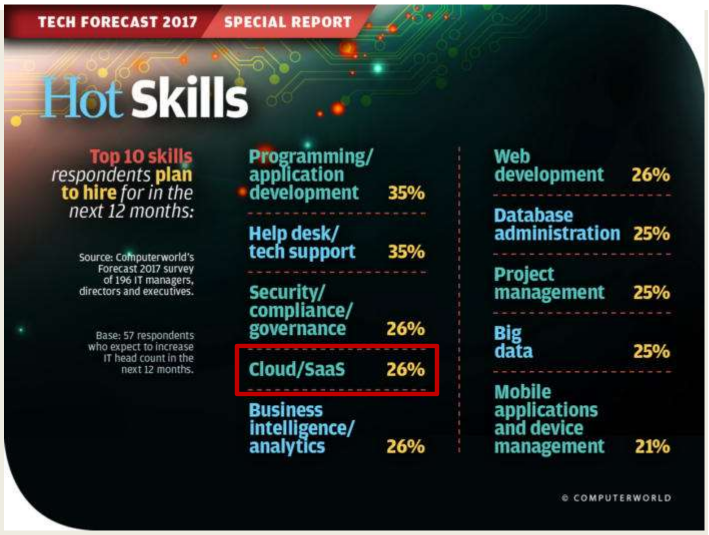
WHY STUDY CLOUD COMPUTING?

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

September 30, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L2.64

WHY STUDY CLOUD COMPUTING? - 2

- Computerworld Magazine



TECH FORECAST 2017 SPECIAL REPORT

Hot Skills

Top 10 skills *respondents* **plan to hire** *for in the next 12 months:*

Source: Computerworld's Forecast 2017 survey of 196 IT managers, directors and executives.

Base: 57 respondents who expect to increase IT head count in the next 12 months.

Programming/application development	35%
Help desk/tech support	35%
Security/compliance/governance	26%
Cloud/SaaS	26%
Business intelligence/analytics	26%
Web development	26%
Database administration	25%
Project management	25%
Big data	25%
Mobile applications and device management	21%

© COMPUTERWORLD

September 30, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L2.65

OBJECTIVES – 10/5

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

September 30, 2019	TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma	L2.66

A BRIEF HISTORY OF CLOUD COMPUTING

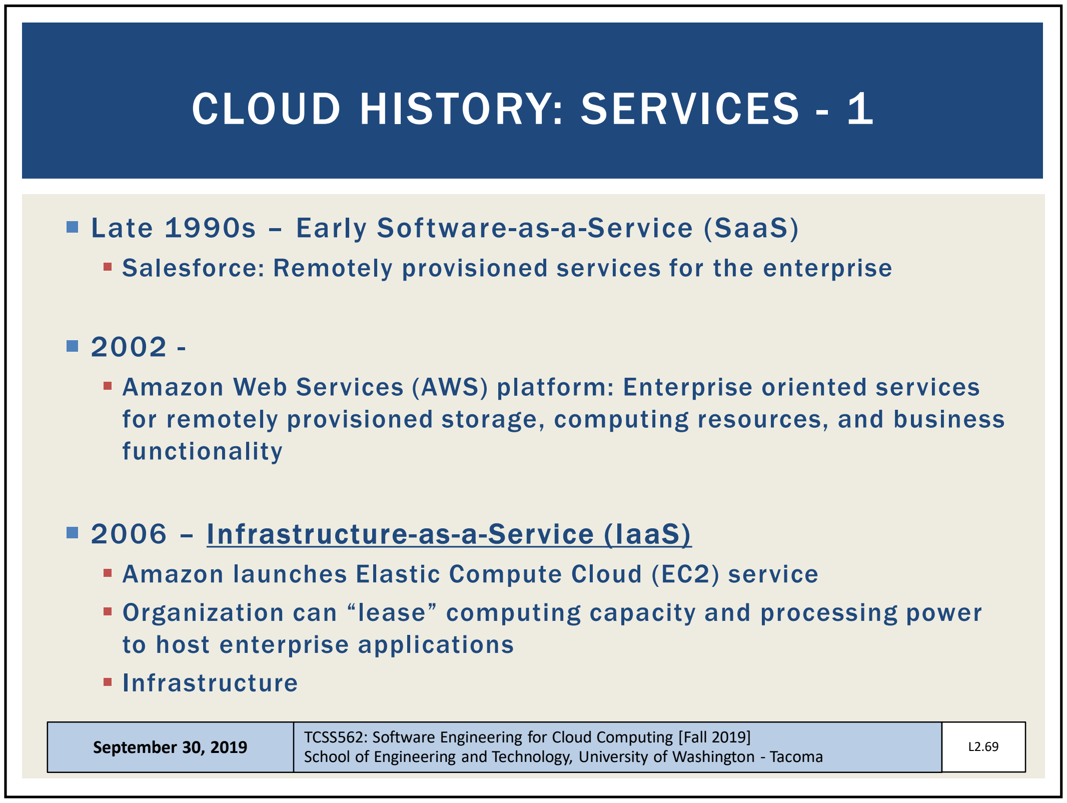
- John McCarthy, 1961
  - Turing award winner for contributions to AI
- "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility… The computer utility could become the basis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

CLOUD HISTORY - 2

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

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

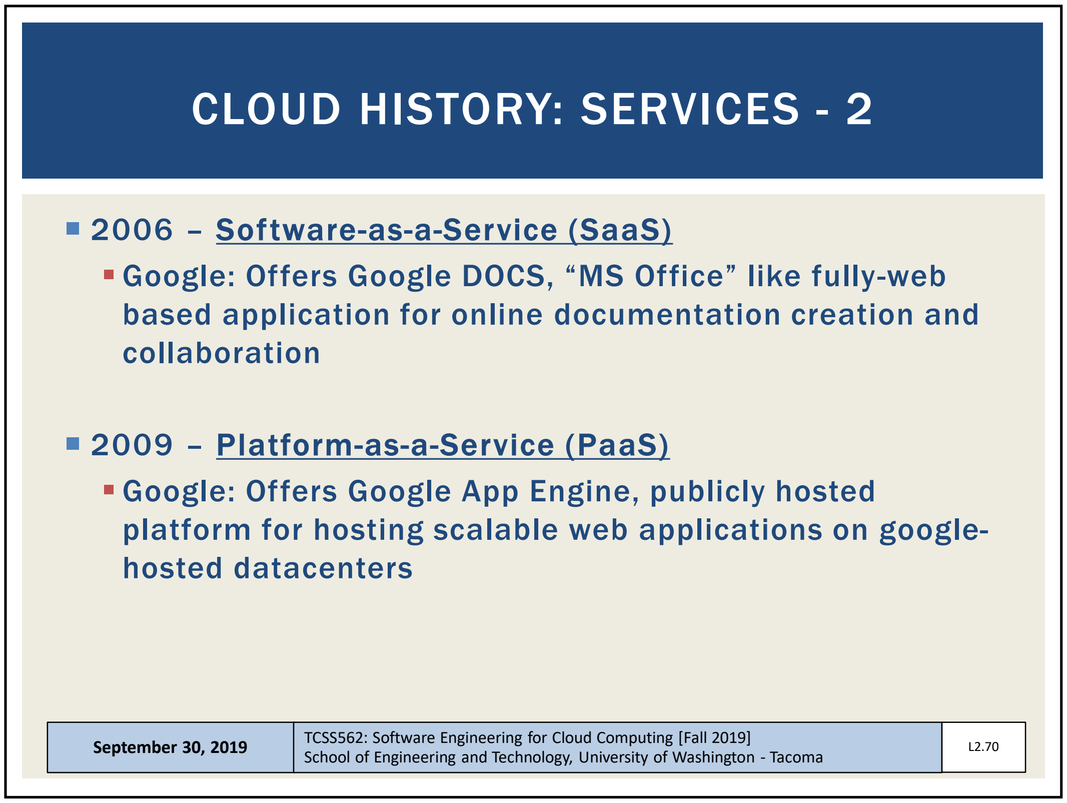
- Popularized core concepts
- Formed basis of cloud computing



CLOUD HISTORY: SERVICES - 1

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

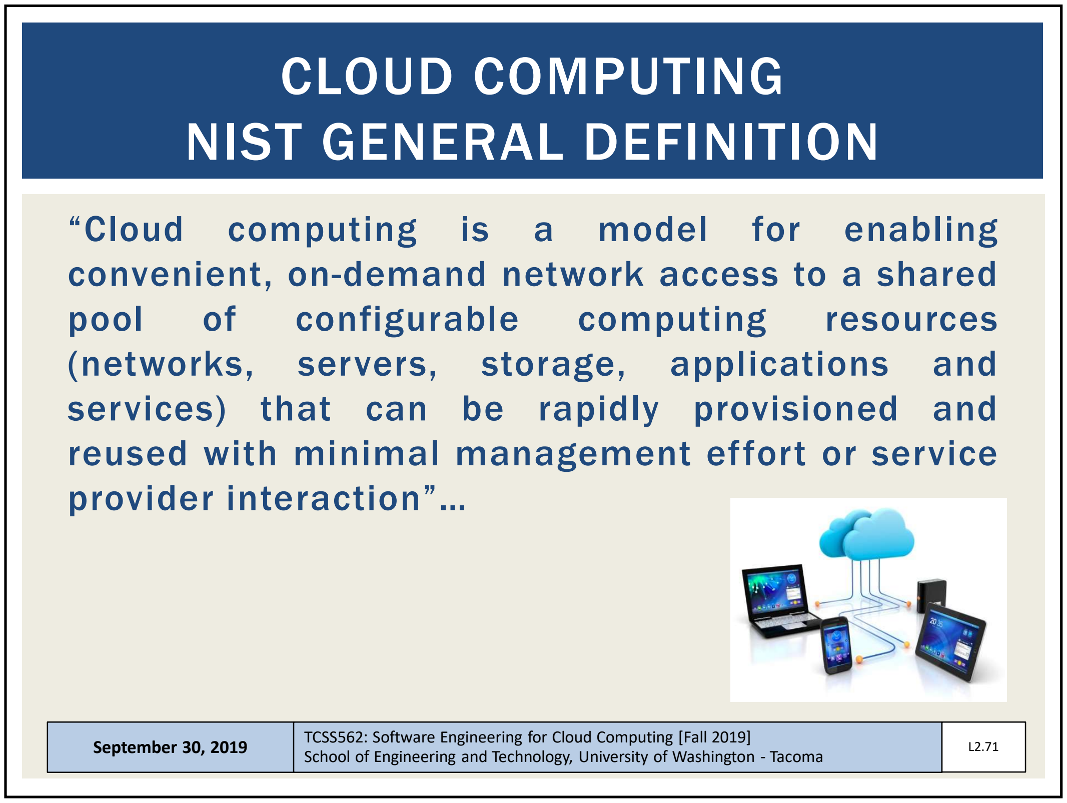
September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L2.69



CLOUD HISTORY: SERVICES - 2

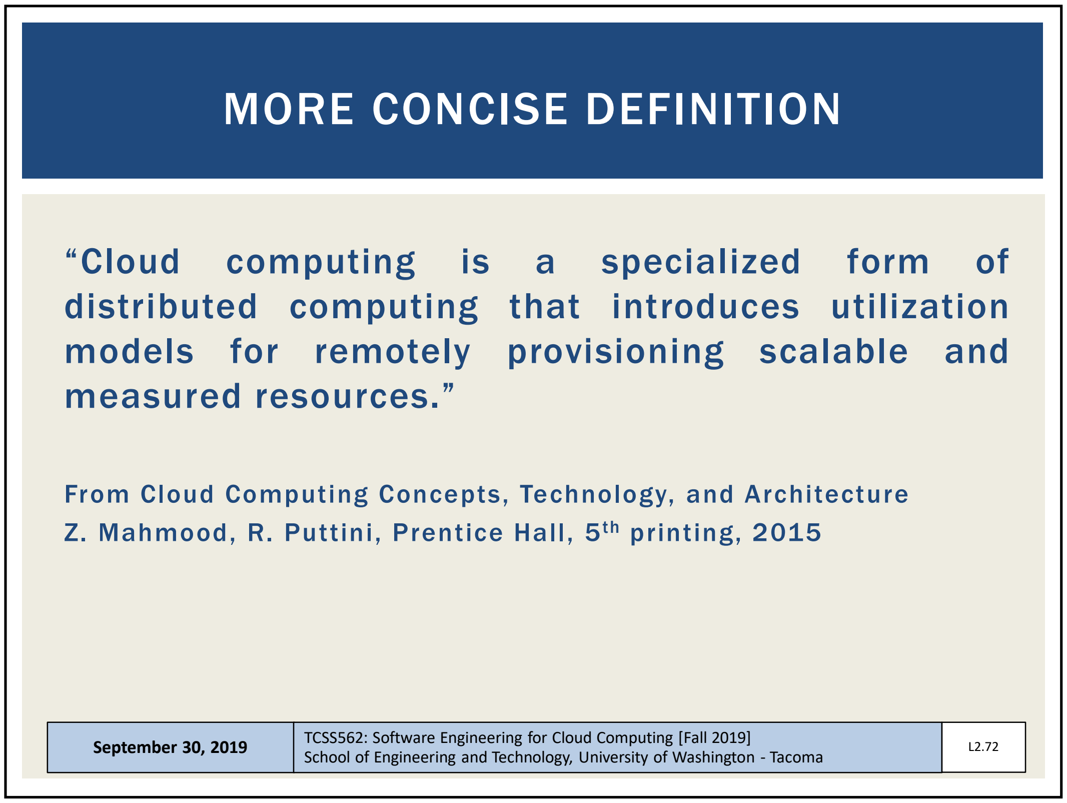
- 2006 – Software-as-a-Service (SaaS)
  - Google: Offers Google DOCS, "MS Office" like fully-web based application for online documentation creation and collaboration
- 2009 – Platform-as-a-Service (PaaS)
  - Google: Offers Google App Engine, publicly hosted platform for hosting scalable web applications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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L2.70



CLOUD COMPUTING NIST GENERAL DEFINITION

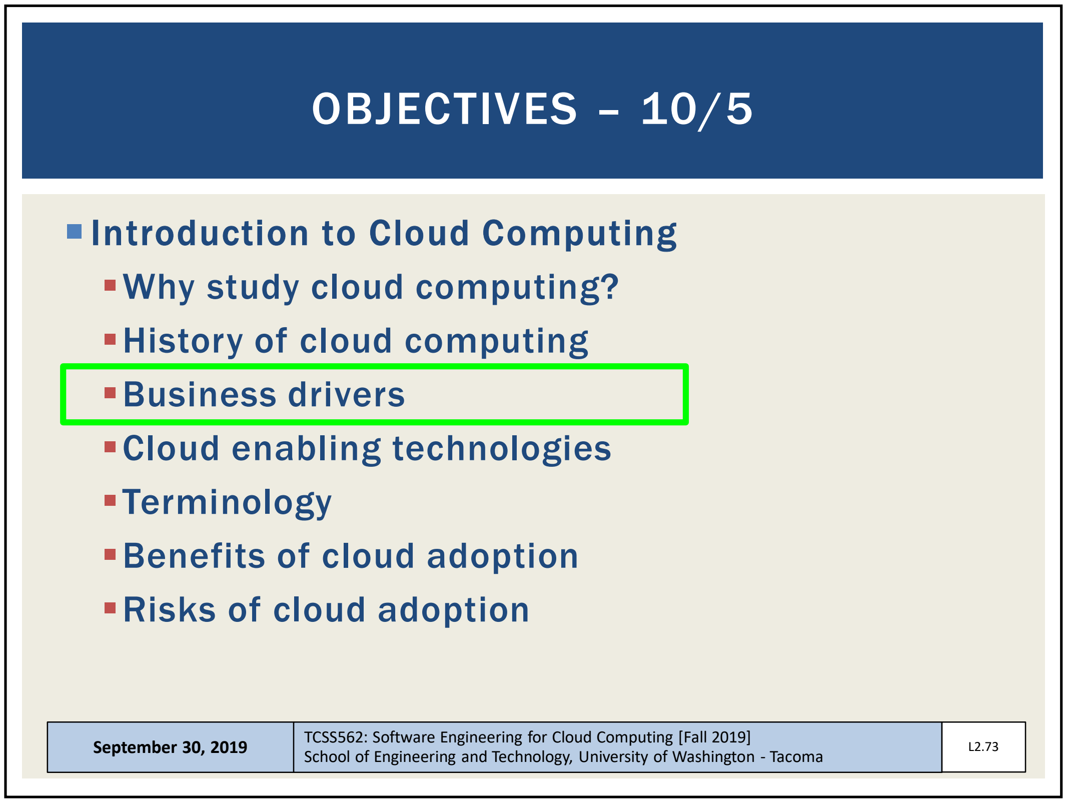
"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications and services) that can be rapidly provisioned and reused with minimal management effort or service provider interaction"…



MORE CONCISE DEFINITION

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

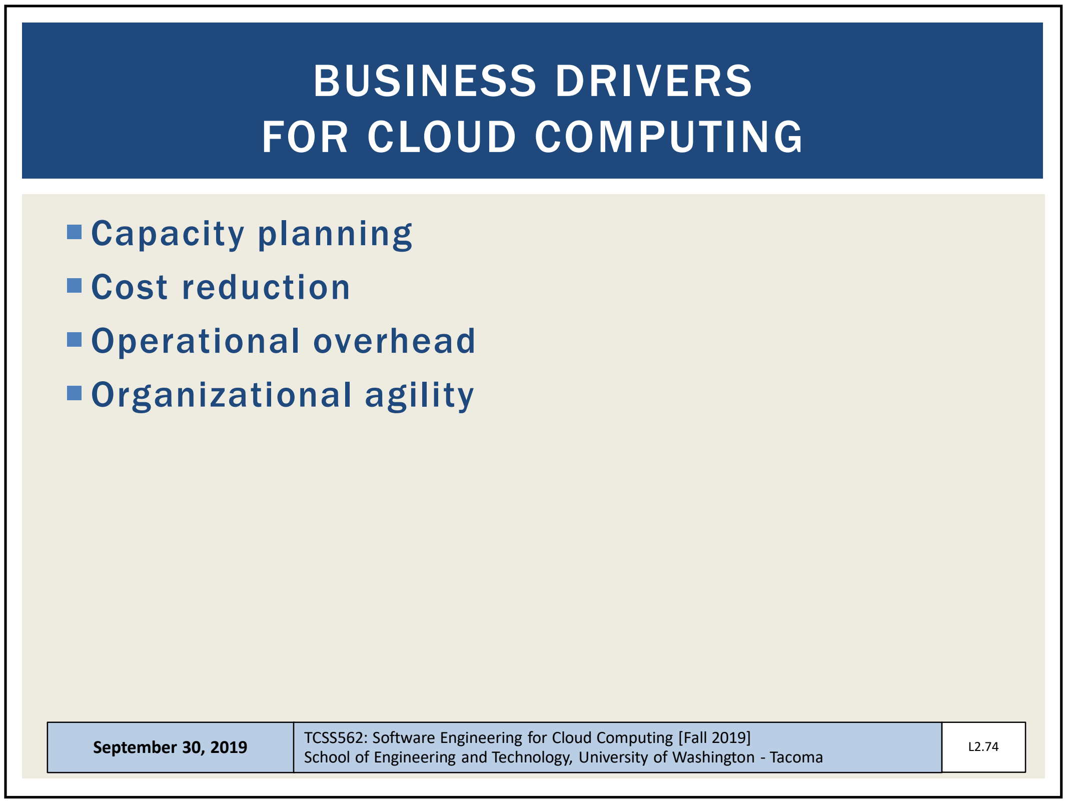
From Cloud Computing Concepts, Technology, and Architecture
Z. Mahmood, R. Puttini, Prentice Hall, 5th printing, 2015



OBJECTIVES – 10/5

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L2.73



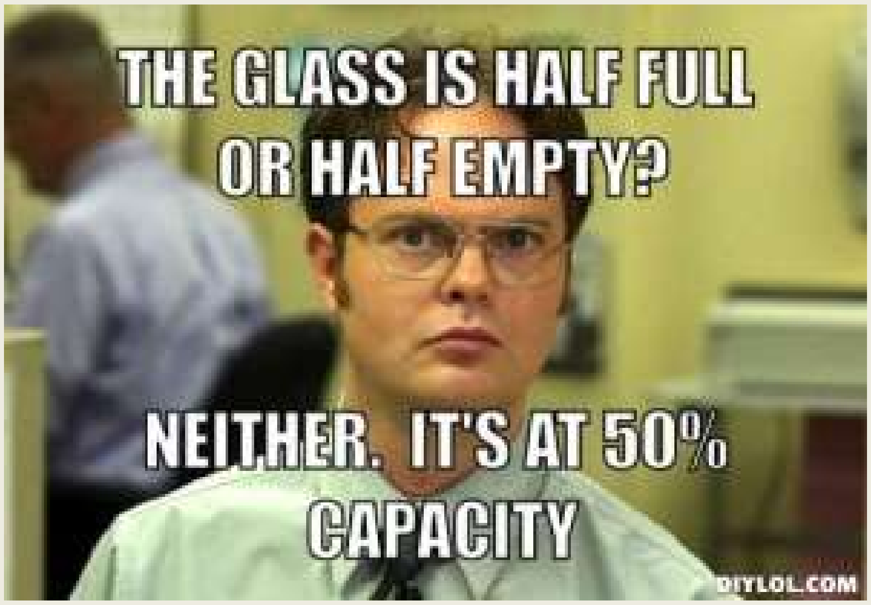
BUSINESS DRIVERS FOR CLOUD COMPUTING

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019]
School of Engineering and Technology, University of Washington - Tacoma | L2.74

BUSINESS DRIVERS FOR CLOUD COMPUTING

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

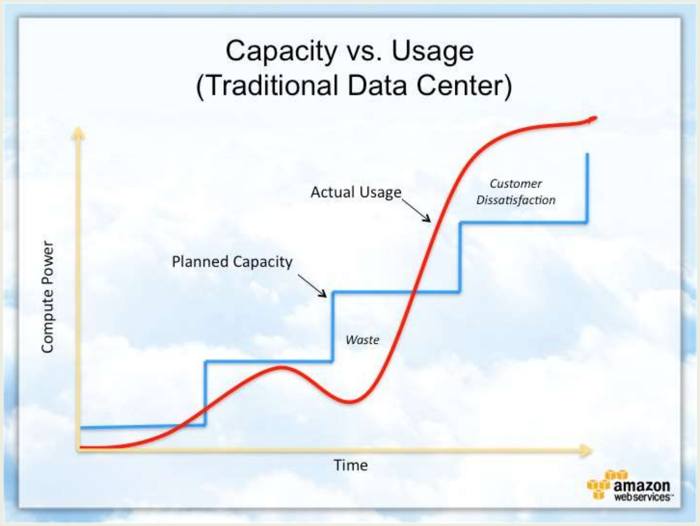

THE GLASS IS HALF FULL OR HALF EMPTY?
NEITHER. IT'S AT 50% CAPACITY
DIYLOL.COM

Dwight, The Office TV sitcom

BUSINESS DRIVERS FOR CLOUD - 2

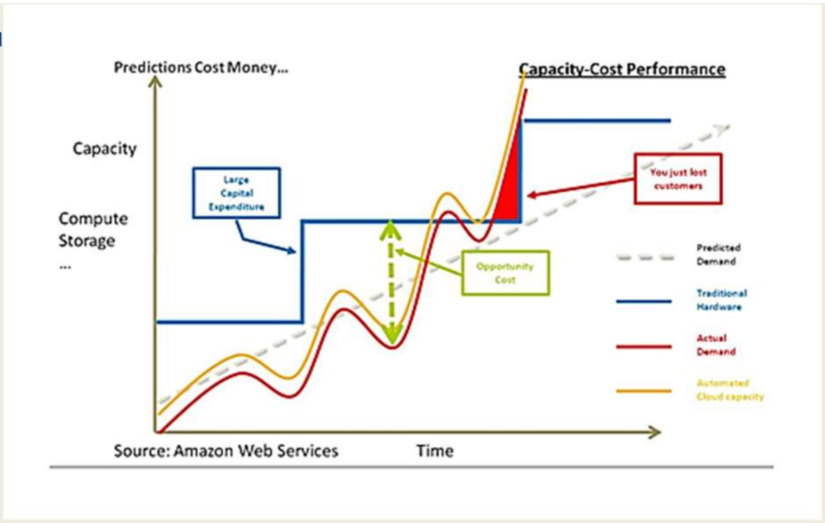
- Capacity planning
  - Over-provisioning: is costly due to too much infrastructure
  - Under-provisioning: is costly due to potential for business loss from poor quality of service
- Capacity planning strategies
  - Lead strategy: add capacity in anticipation of demand (pre-provisioning)
  - Lag strategy: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

CAPACITY PLANNING



CAPACITY PLANNING - 2

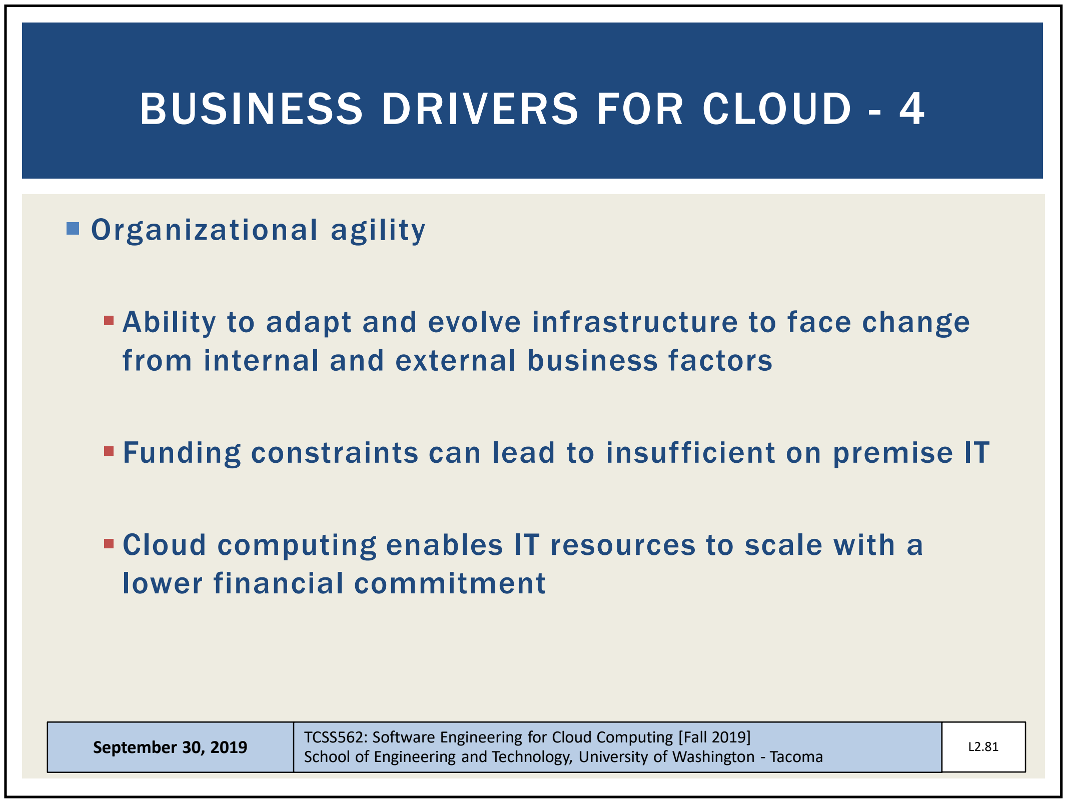
- Ca



BUSINESS DRIVERS FOR CLOUD - 3

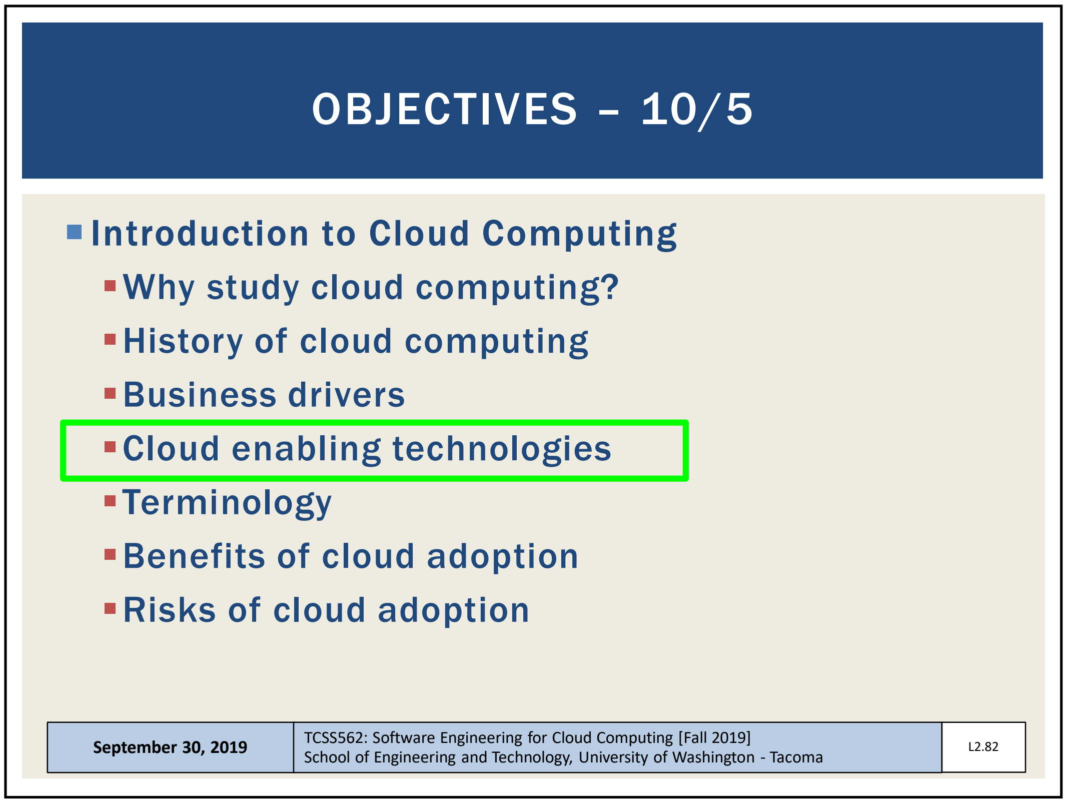
- Cost reduction
  - IT Infrastructure acquisition
  - IT Infrastructure maintenance

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



BUSINESS DRIVERS FOR CLOUD - 4

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



OBJECTIVES – 10/5

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

TECHNOLOGY INNOVATIONS LEADING TO CLOUD

- Cluster computing
- Grid computing
- Virtualization
- Others

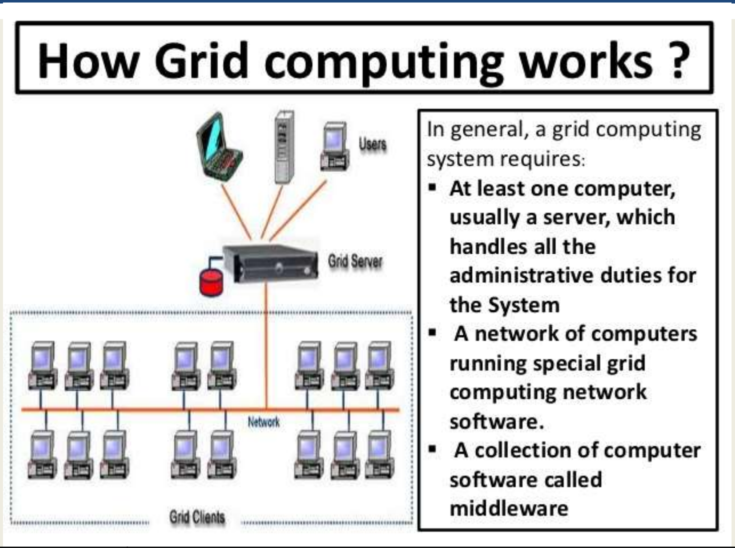
CLUSTER COMPUTING

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

GRID COMPUTING

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

GRID COMPUTING - 2

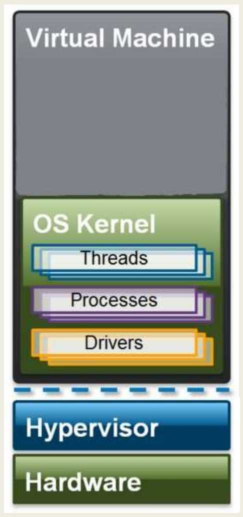


How Grid computing works ?

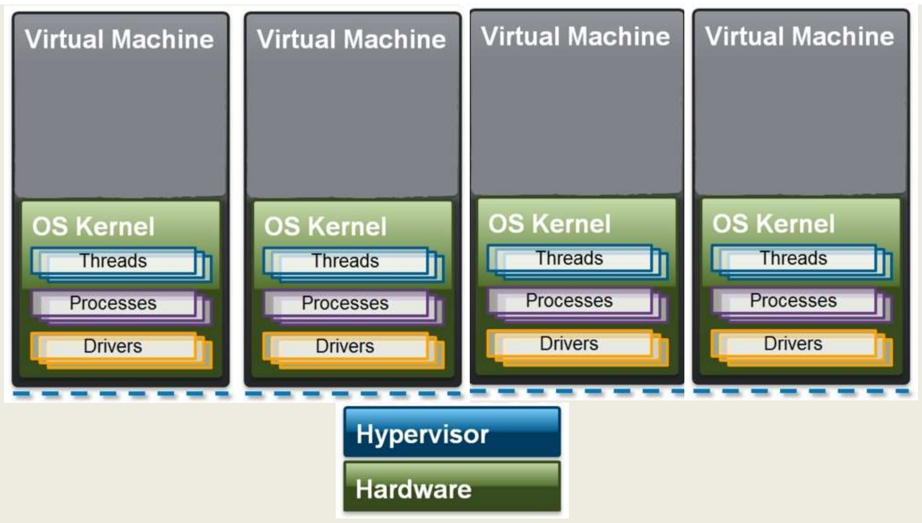
In general, a grid computing system requires:

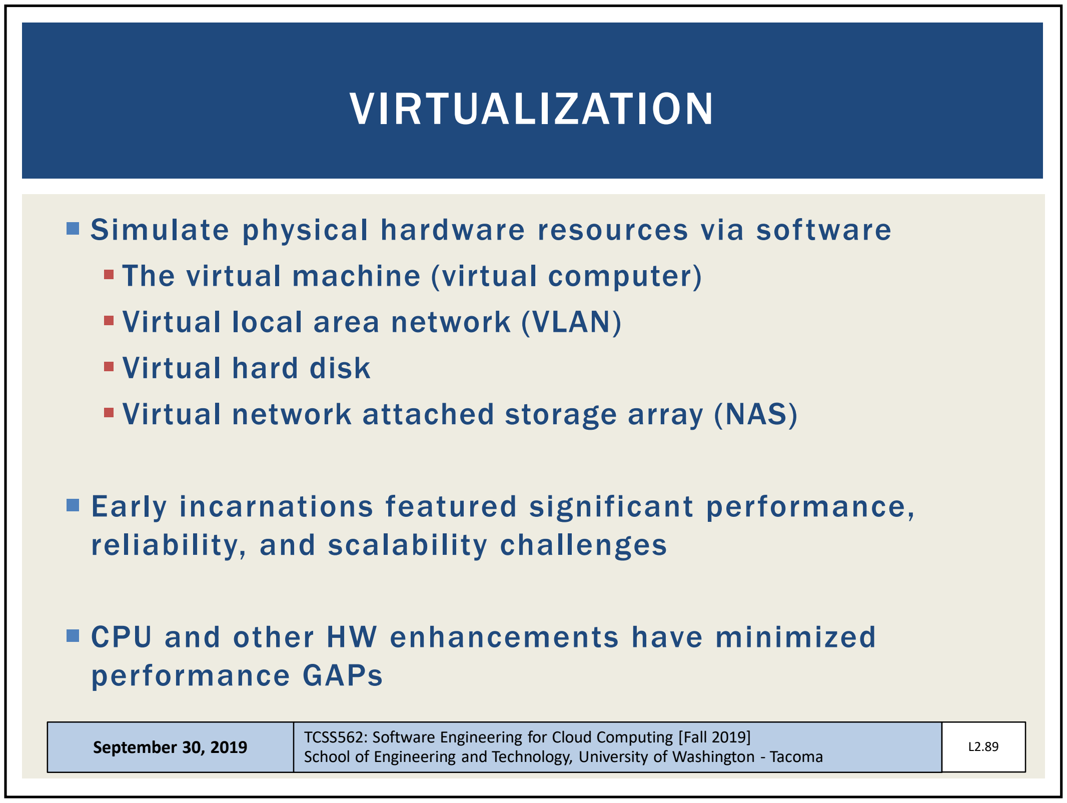
- **At least one computer, usually a server, which handles all the administrative duties for the System**
- **A network of computers running special grid computing network software.**
- **A collection of computer software called middleware**

VIRTUALIZATION



VIRTUALIZATION

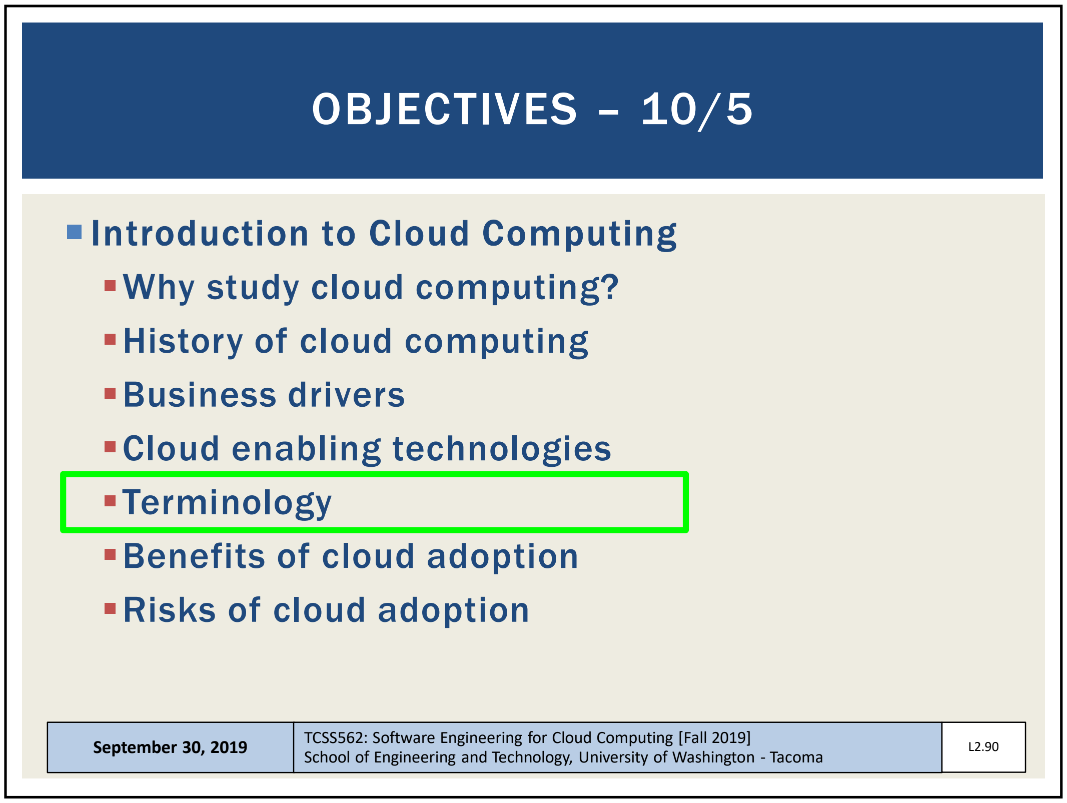




VIRTUALIZATION

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L2.89



OBJECTIVES – 10/5

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

September 30, 2019 | TCSS562: Software Engineering for Cloud Computing [Fall 2019] School of Engineering and Technology, University of Washington - Tacoma | L2.90

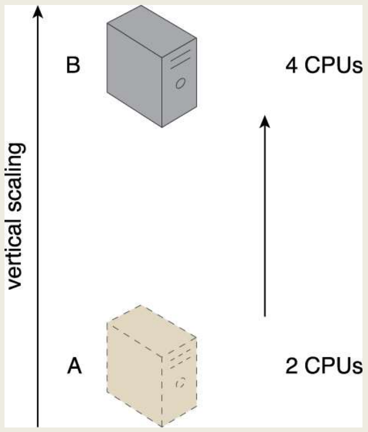
KEY TERMINOLOGY

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

VERTICAL SCALING

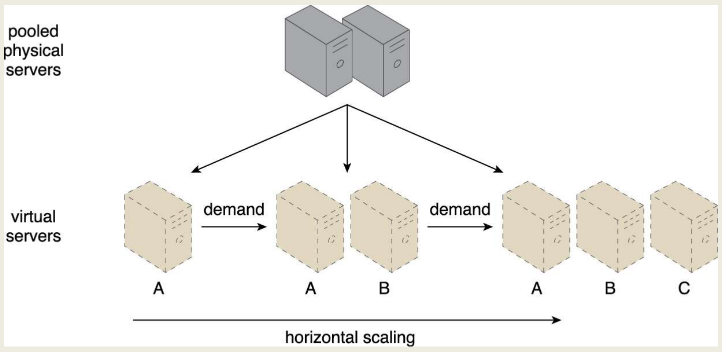
- Reconfigure virtual machine to have different resources:
  - CPU cores
  - RAM
  - HDD/SDD capacity

- May require VM migration if physical host machine resources are exceeded



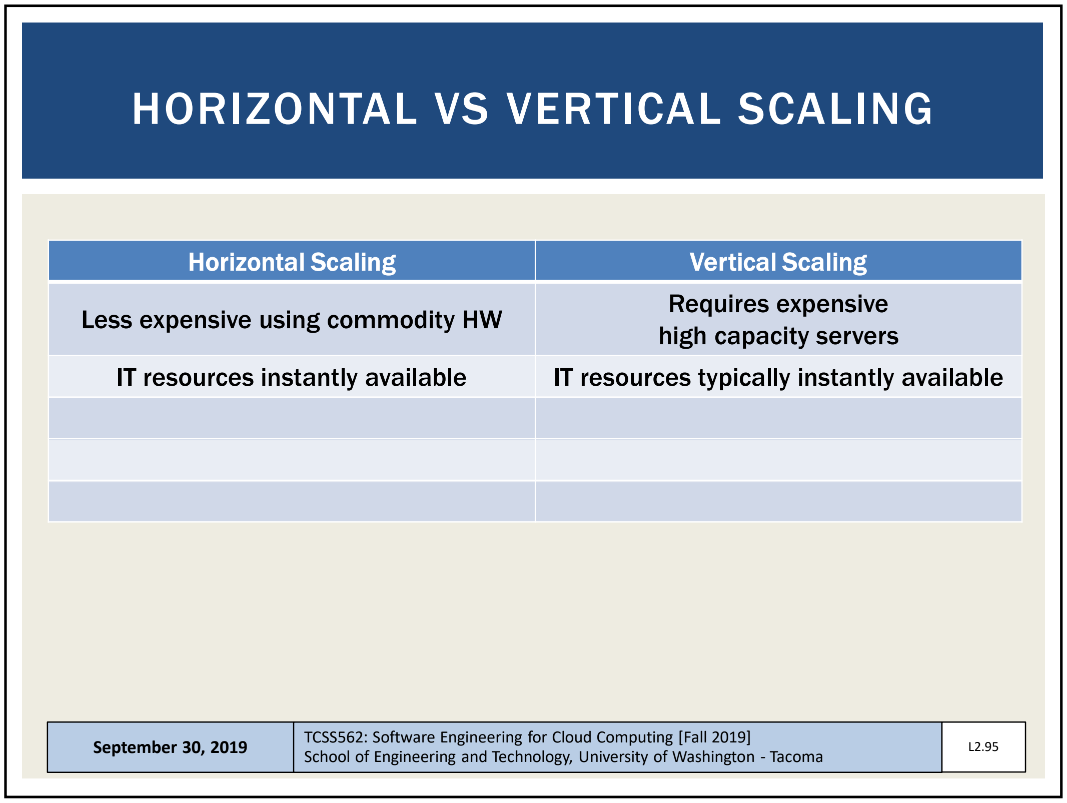
HORIZONTAL SCALING

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



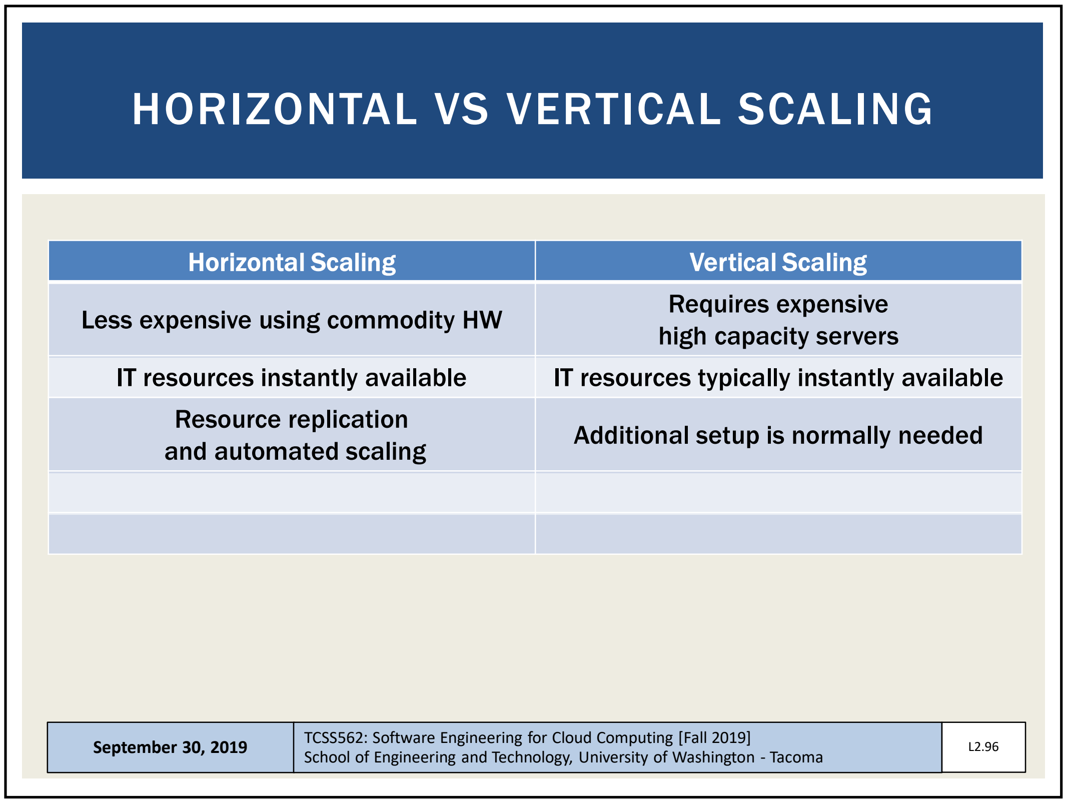
HORIZONTAL VS VERTICAL SCALING

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



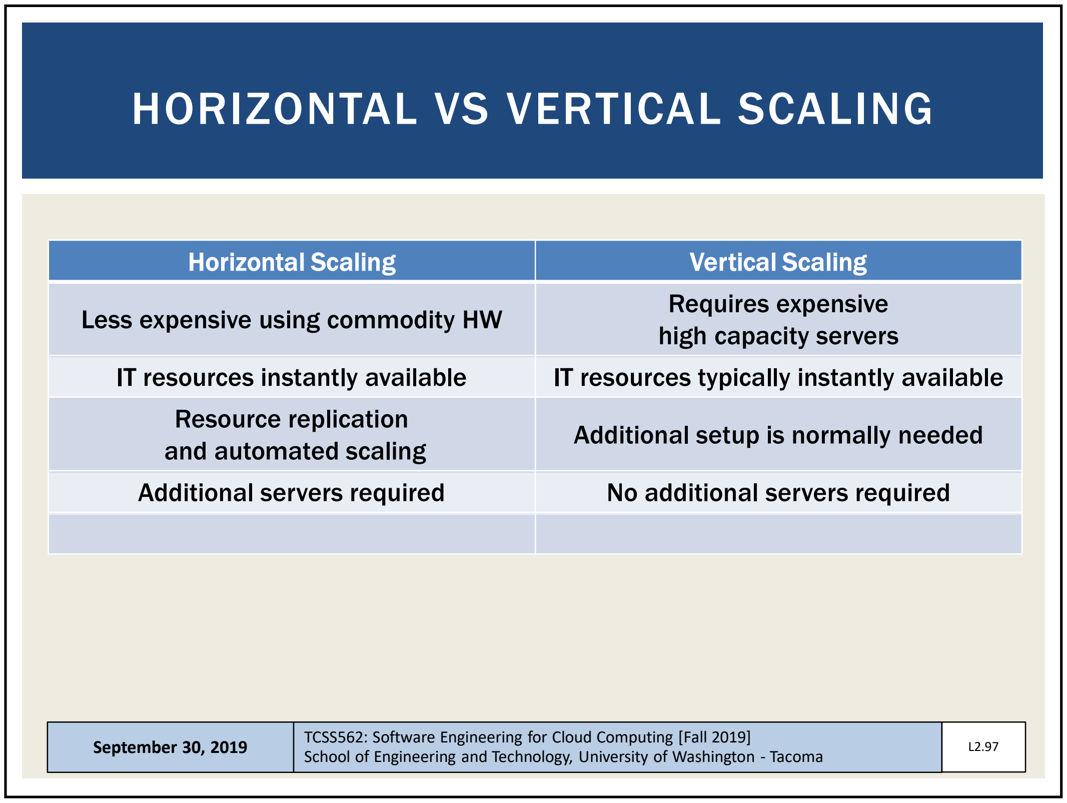
HORIZONTAL VS VERTICAL SCALING

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



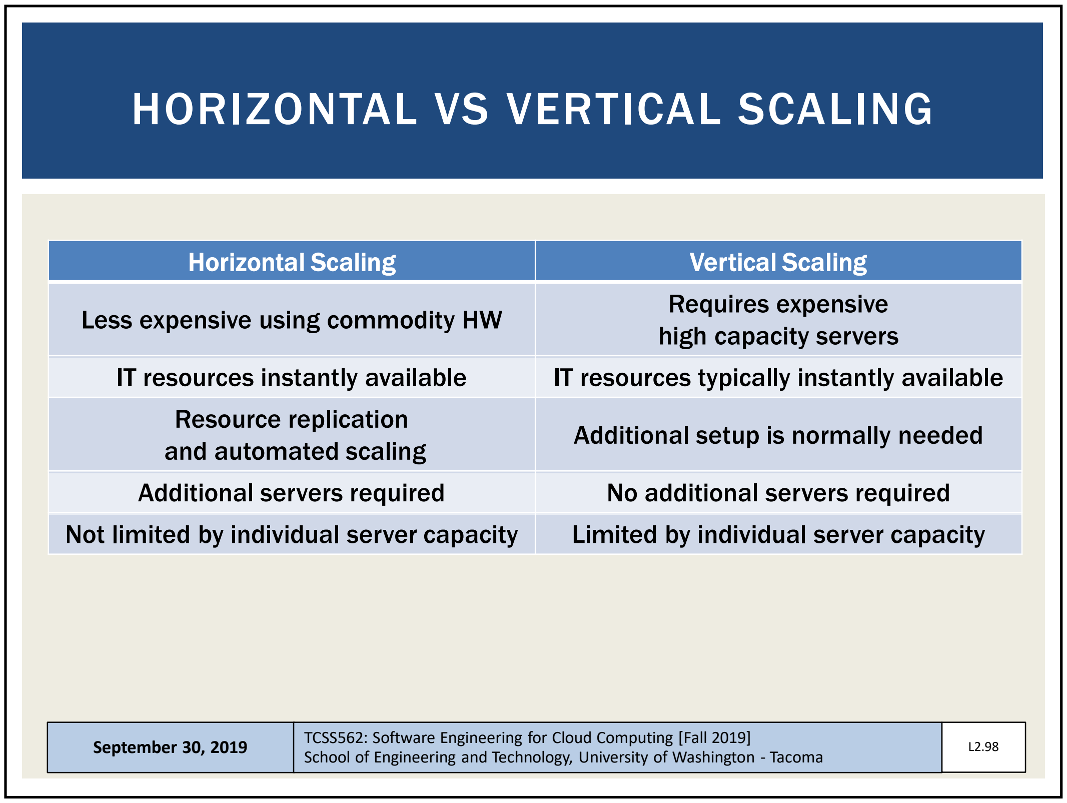
HORIZONTAL VS VERTICAL SCALING

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



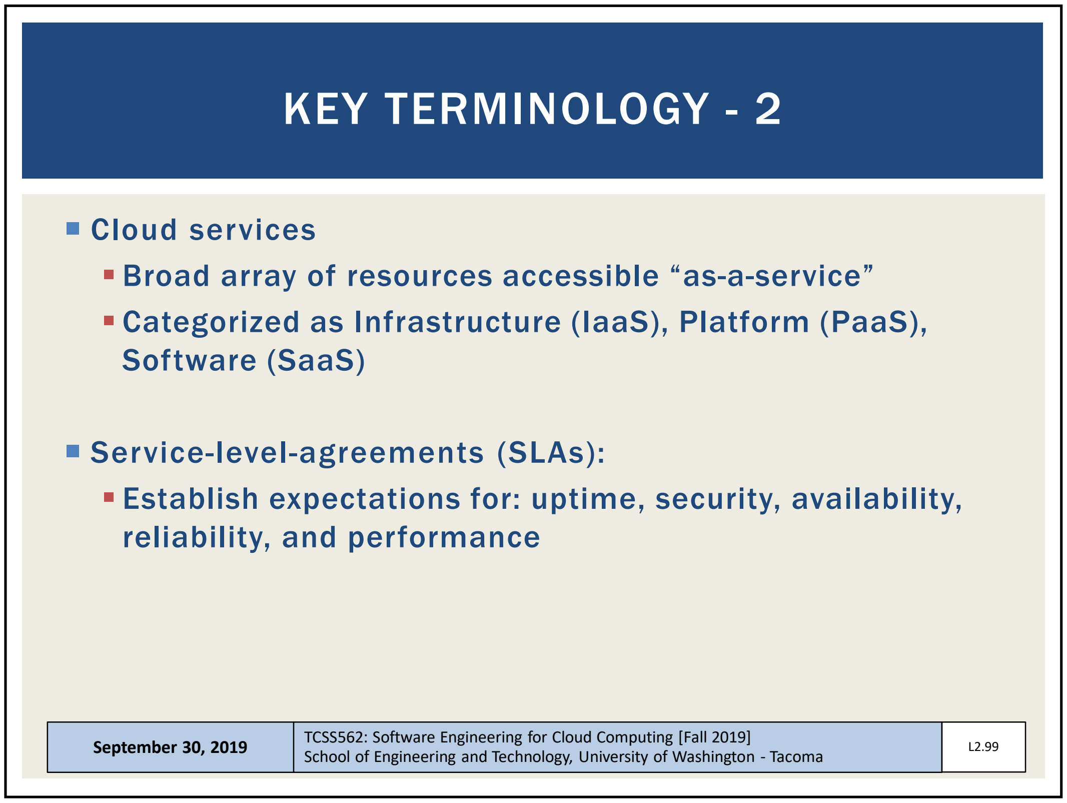
HORIZONTAL VS VERTICAL SCALING

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



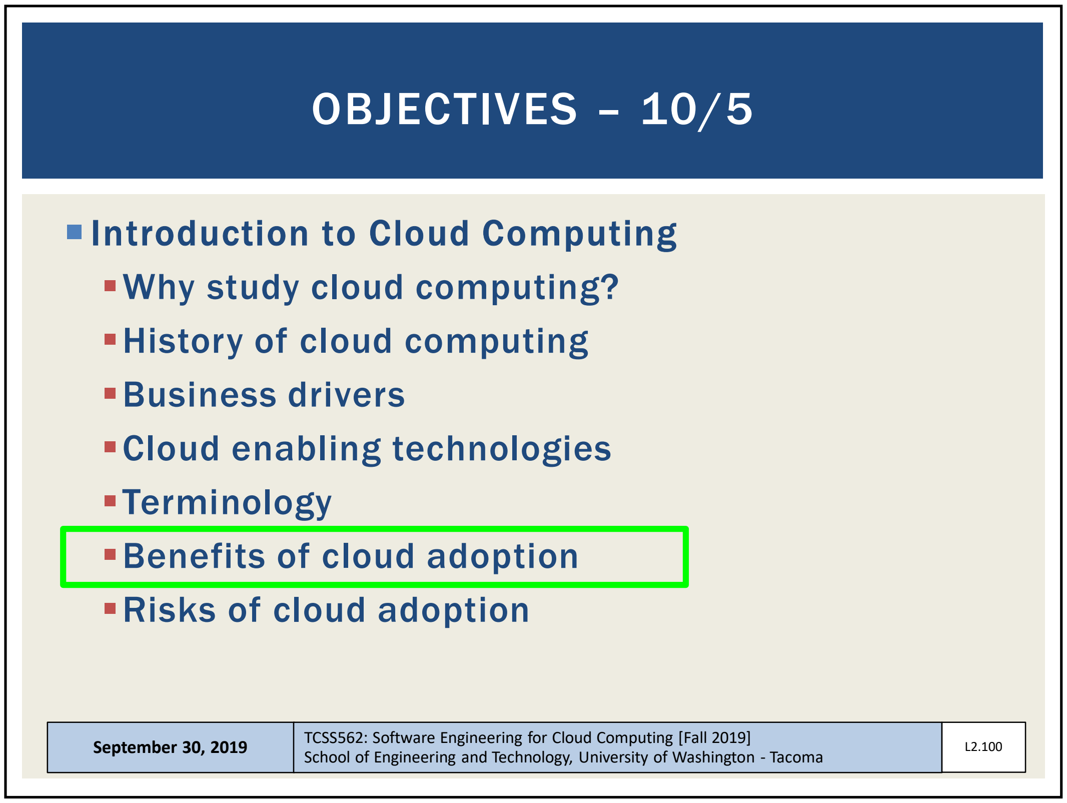
HORIZONTAL VS VERTICAL SCALING

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



KEY TERMINOLOGY - 2

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



OBJECTIVES – 10/5

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

GOALS AND BENEFITS

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

CLOUD BENEFITS - 2

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

CLOUD BENEFITS - 3

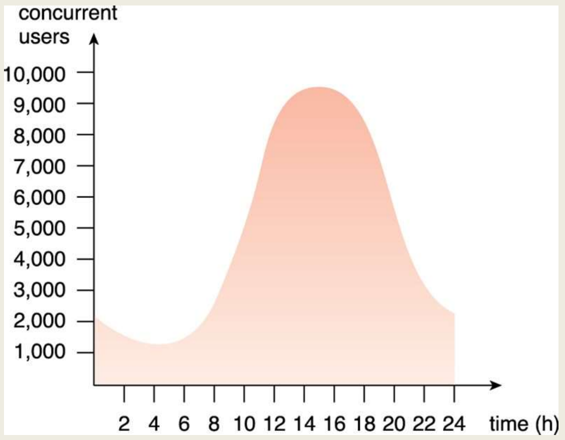
- Example: Using 100 servers for 1 hour costs the same as using 1 server for 100 hours
- Rosetta Protein Folding: Working with a UW-Tacoma graduate student, we recently deployed this science model across 5,900 compute cores on Amazon for 2-days…
- *What is the cost to purchase 5,900 compute cores?*
-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

OH YOU NEED MORE SERVERS?
INTERESTING... I HAVE SOMETHING TO SHOW YOU...
MEMECREATOR.NET

Gene Wilder, Charlie and the Chocolate Factory

CLOUD BENEFITS

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



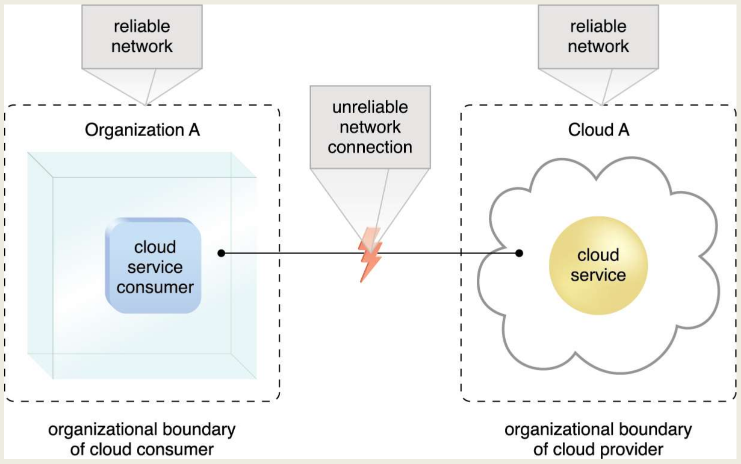
OBJECTIVES – 10/5

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

CLOUD ADOPTION RISKS

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

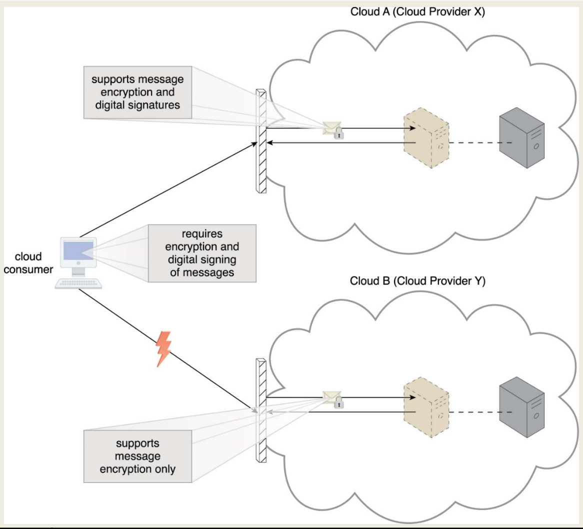
NETWORK LATENCY COSTS


reliable network
reliable network
unreliable network connection
Organization A
Cloud A
cloud service consumer
cloud service
organizational boundary of cloud consumer
organizational boundary of cloud provider

CLOUD RISKS - 2

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

CLOUD: VENDOR LOCK-IN


Cloud A (Cloud Provider X)
supports message encryption and digital signatures
cloud consumer
requires encryption and digital signing of messages
Cloud B (Cloud Provider Y)
supports message encryption only

QUESTIONS

WE WILL RETURN AT 7:03PM

TCSS 562
OFFICE HOURS

PLEASE SAY HELLO

L2.113