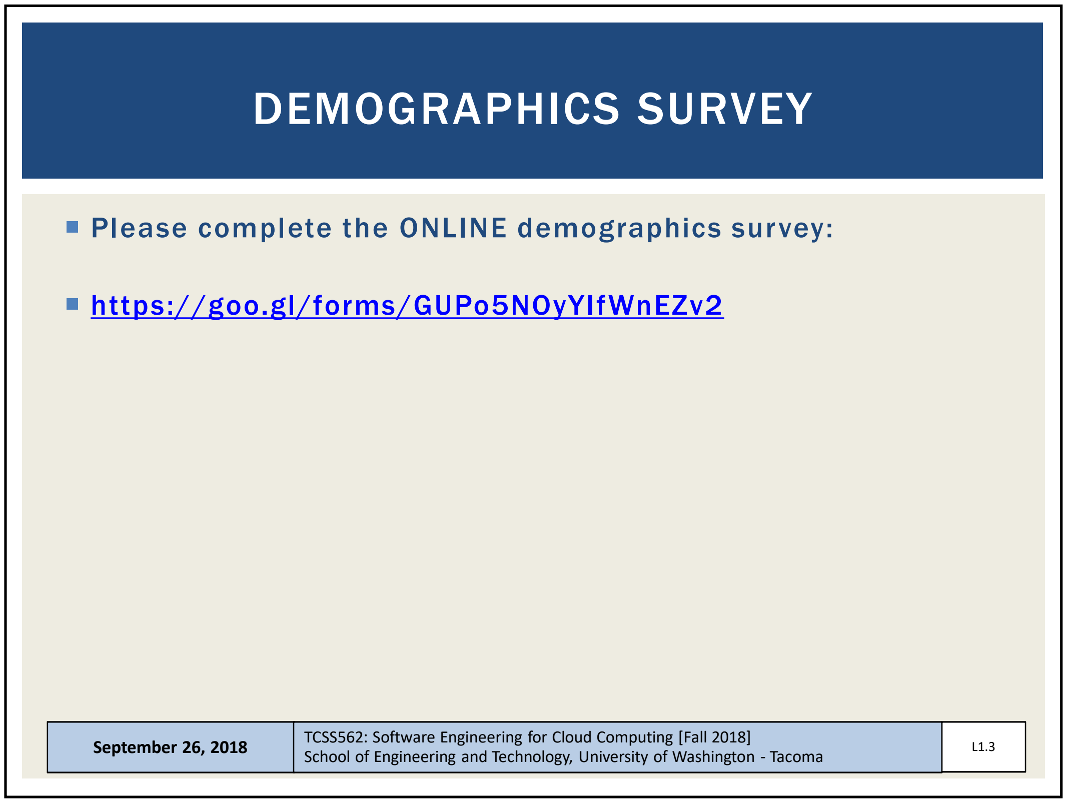
TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Introduction

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

OBJECTIVES

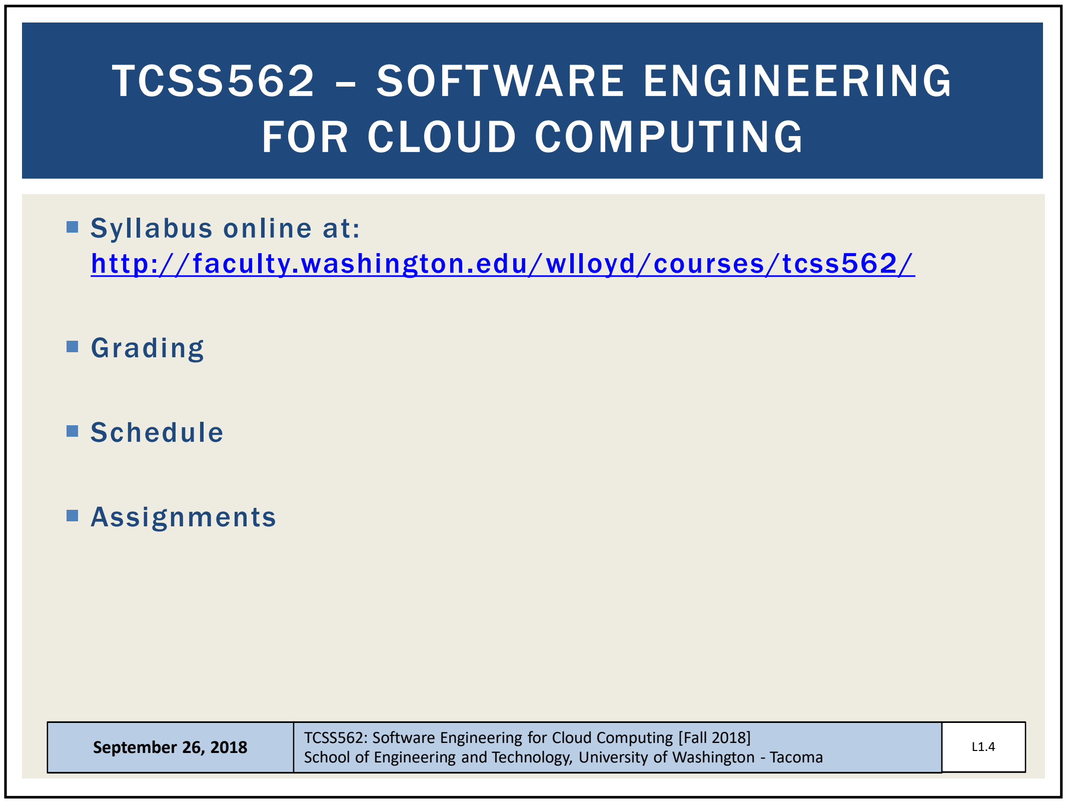
- Syllabus, Course Introduction
- Parallel and distributed systems (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Introduction to Cloud Computing
  - Why study cloud computing?
  - History of cloud computing
  - Business drivers
  - Cloud enabling technologies
  - Terminology
  - Benefits of cloud adoption
  - Risks of cloud adoption



DEMOGRAPHICS SURVEY

- Please complete the ONLINE demographics survey:
- https://goo.gl/forms/GUPo5NOyYIfWnEZv2

September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.3



TCSS562 – SOFTWARE ENGINEERING FOR CLOUD COMPUTING

- Syllabus online at: http://faculty.washington.edu/wlloyd/courses/tcss562/
- Grading
- Schedule
- Assignments

September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.4

REFERENCES

- [1] Cloud Computing: Concepts, Technology and Architecture
- Thomas Erl, Prentice Hall 2013

- [2] Cloud Computing - Theory and Practice
- Dan Marinescu, First Edition 2013, Second Edition 2018

- [3] Cloud Computing: A Hands-On Approach
- Arshdeep Bahga 2013

- Research papers

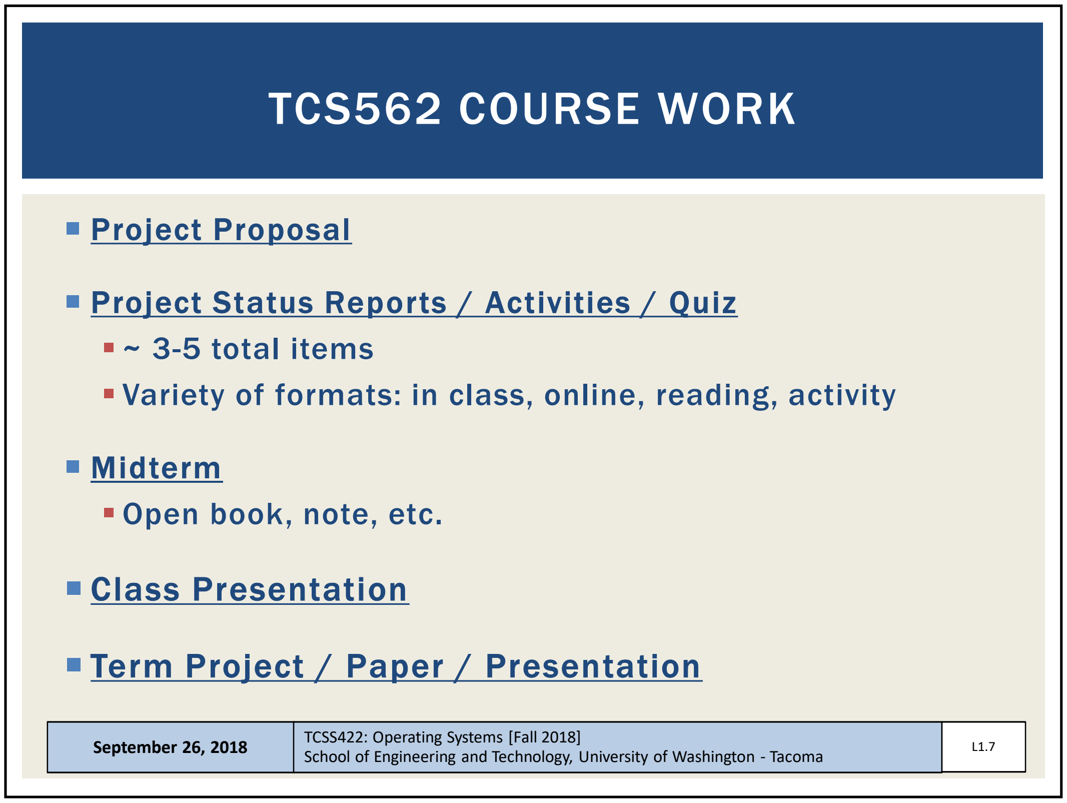
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L1.5

TCSS 562 – Fall 2018

- Mondays/Wednesdays
  - Lecture, midterm, quiz, activities
  - No class:
    Monday Nov 12, Friday Nov 23
  - Key Topics:
  - IaaS, Virtualization, Serverless computing, Containerization
- Fridays
  - Lab Day: Tutorials, group project work
- No Final exam
- Midterm Wednesday November 7th
- Project presentations – Final Exam Week
- Term Project: Build and evaluate alternate implementations of a native cloud serverless application; or group proposed cloud research project

TCSS 562
FALL
2018

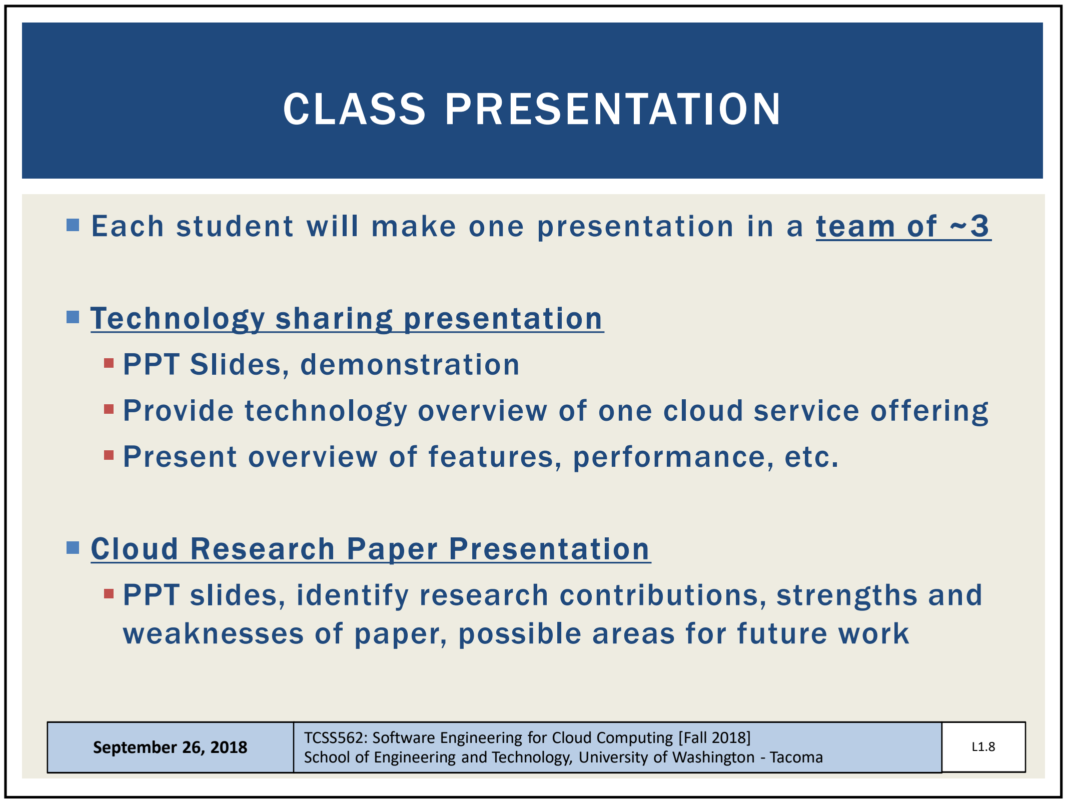
September 26, 2018 TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma L1.6



TCS562 COURSE WORK

- Project Proposal
- Project Status Reports / Activities / Quiz
  - ~ 3-5 total items
  - Variety of formats: in class, online, reading, activity
- Midterm
  - Open book, note, etc.
- Class Presentation
- Term Project / Paper / Presentation

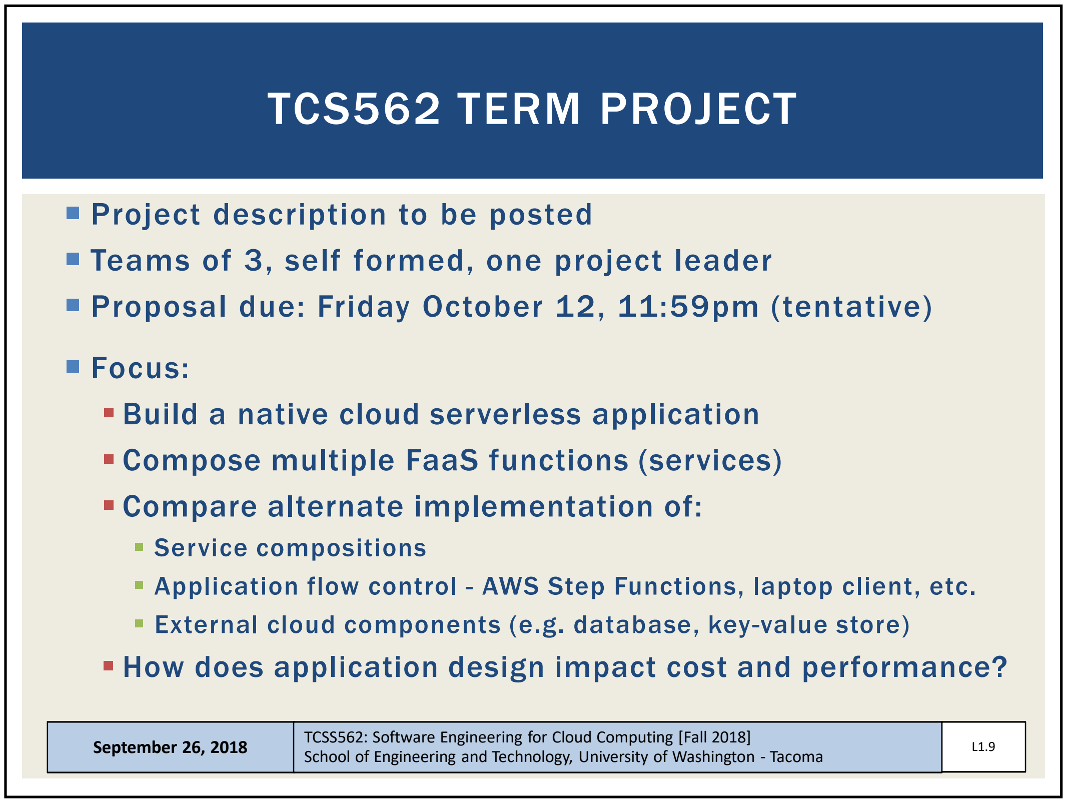
September 26, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.7



CLASS PRESENTATION

- Each student will make one presentation in a team of ~3
- Technology sharing presentation
  - PPT Slides, demonstration
  - Provide technology overview of one cloud service offering
  - Present overview of features, performance, etc.
- Cloud Research Paper Presentation
  - PPT slides, identify research contributions, strengths and weaknesses of paper, possible areas for future work

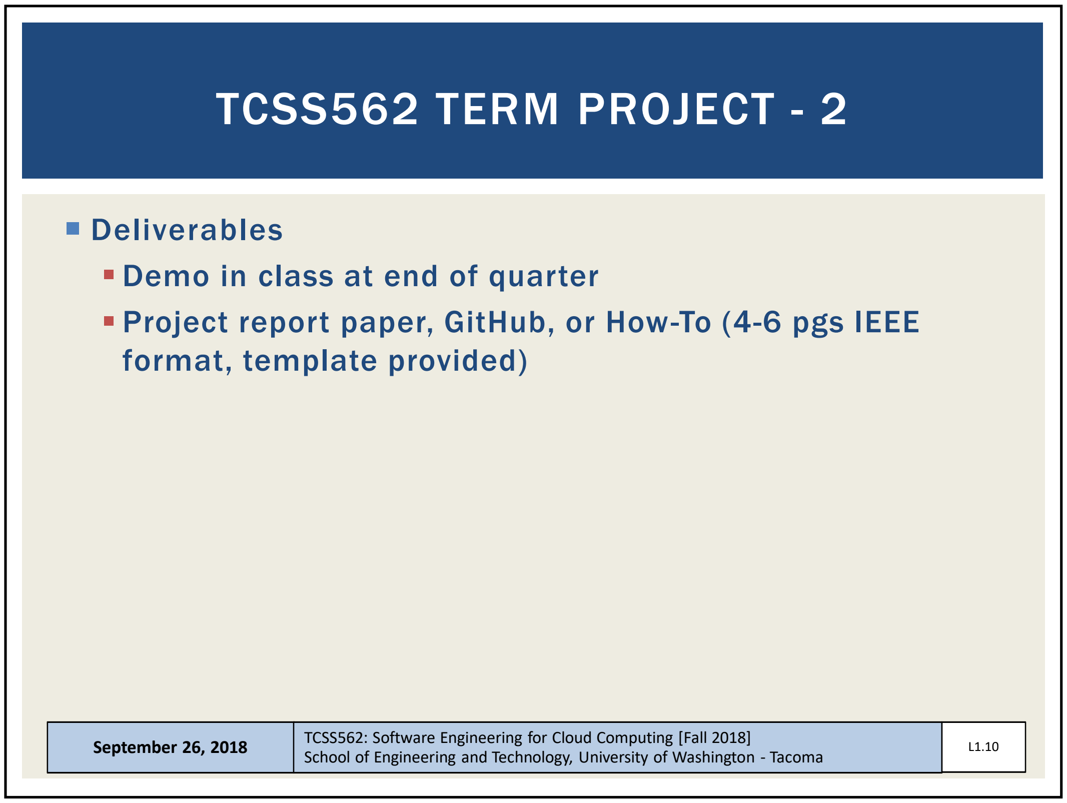
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.8



TCS562 TERM PROJECT

- Project description to be posted
- Teams of 3, self formed, one project leader
- Proposal due: Friday October 12, 11:59pm (tentative)
- Focus:
  - Build a native cloud serverless application
  - Compose multiple FaaS functions (services)
  - Compare alternate implementation of:
    - Service compositions
    - Application flow control - AWS Step Functions, laptop client, etc.
    - External cloud components (e.g. database, key-value store)
  - How does application design impact cost and performance?

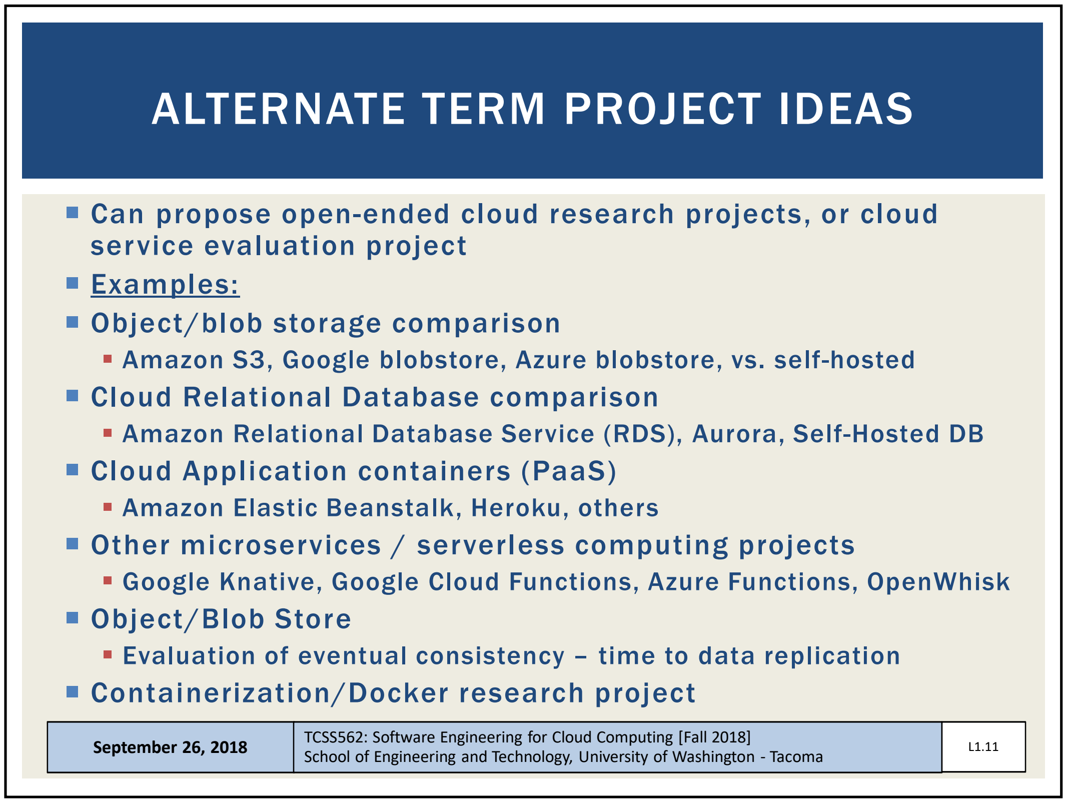
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.9



TCSS562 TERM PROJECT - 2

- Deliverables
  - Demo in class at end of quarter
  - Project report paper, GitHub, or How-To (4-6 pgs IEEE format, template provided)

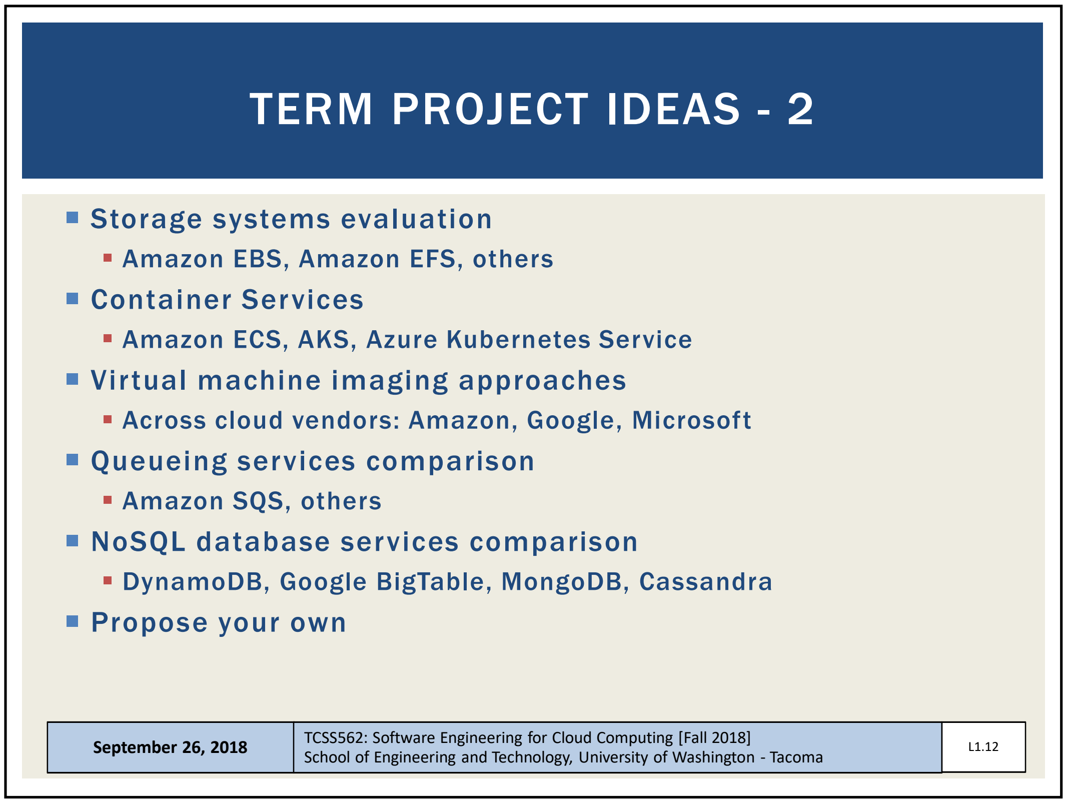
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.10



ALTERNATE TERM PROJECT IDEAS

- Can propose open-ended cloud research projects, or cloud service evaluation project
- Examples:
- Object/blob storage comparison
  - Amazon S3, Google blobstore, Azure blobstore, vs. self-hosted
- Cloud Relational Database comparison
  - Amazon Relational Database Service (RDS), Aurora, Self-Hosted DB
- Cloud Application containers (PaaS)
  - Amazon Elastic Beanstalk, Heroku, others
- Other microservices / serverless computing projects
  - Google Knative, Google Cloud Functions, Azure Functions, OpenWhisk
- Object/Blob Store
  - Evaluation of eventual consistency – time to data replication
- Containerization/Docker research project

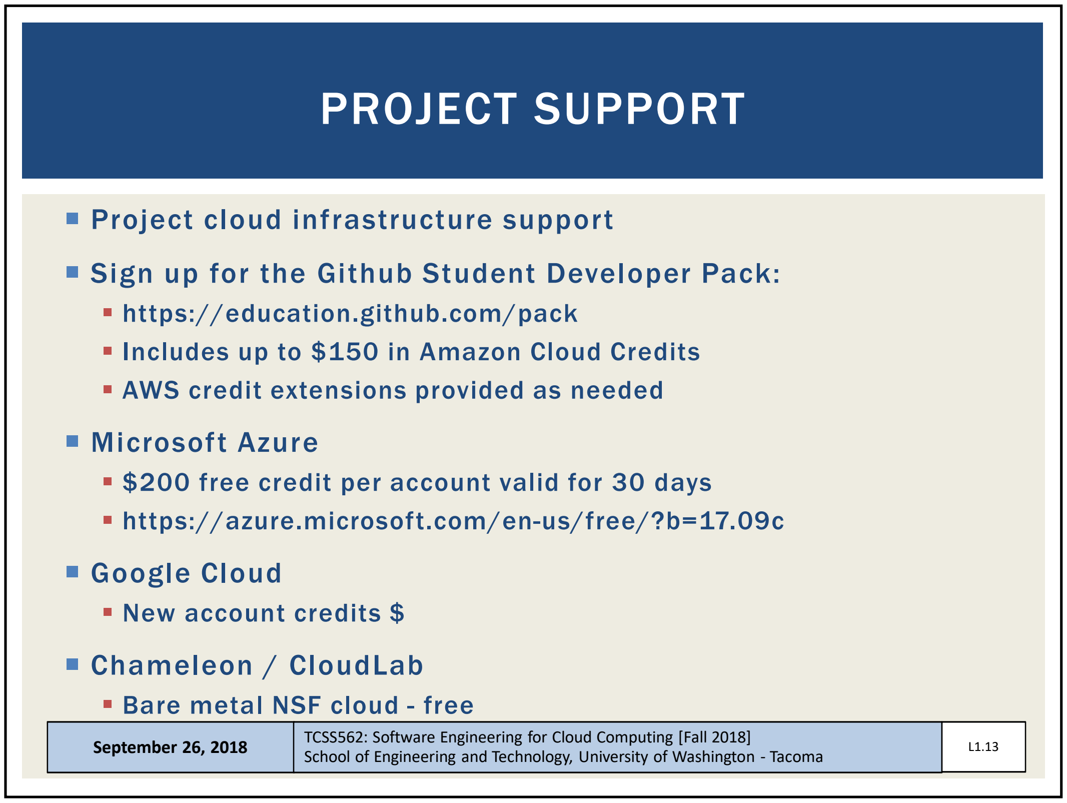
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.11



TERM PROJECT IDEAS - 2

- Storage systems evaluation
  - Amazon EBS, Amazon EFS, others
- Container Services
  - Amazon ECS, AKS, Azure Kubernetes Service
- Virtual machine imaging approaches
  - Across cloud vendors: Amazon, Google, Microsoft
- Queueing services comparison
  - Amazon SQS, others
- NoSQL database services comparison
  - DynamoDB, Google BigTable, MongoDB, Cassandra
- Propose your own

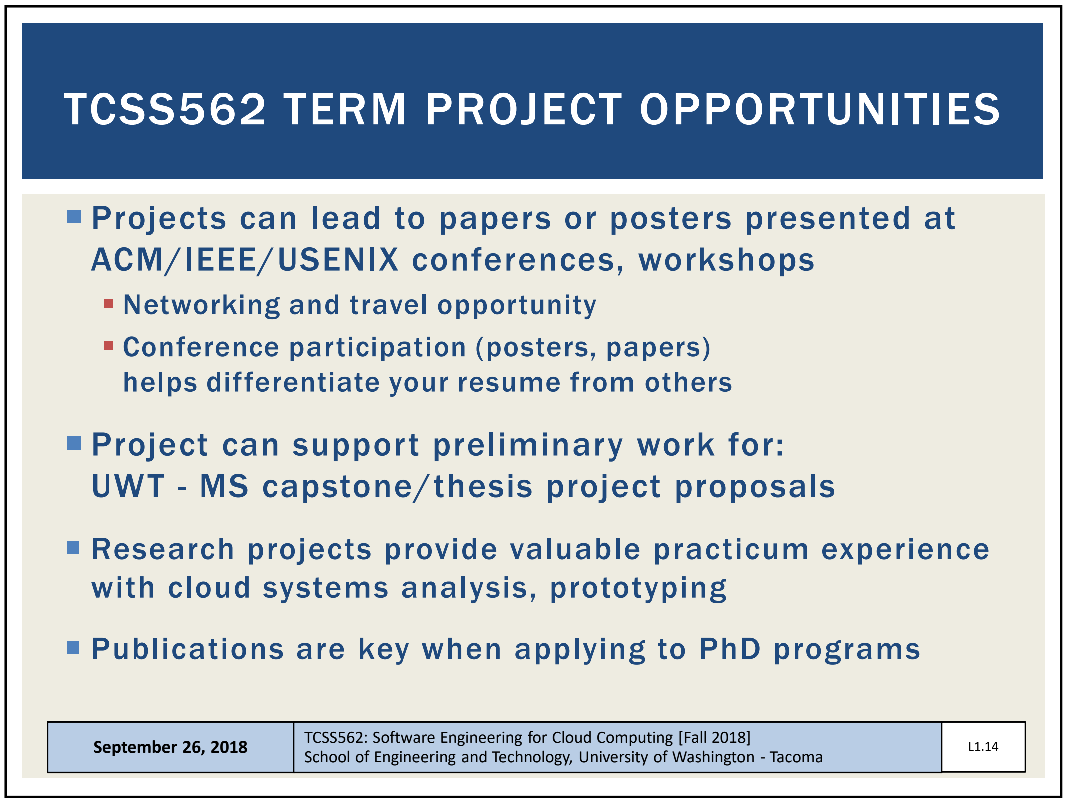
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.12



PROJECT SUPPORT

- Project cloud infrastructure support
- Sign up for the Github Student Developer Pack:
  - https://education.github.com/pack
  - Includes up to $150 in Amazon Cloud Credits
  - AWS credit extensions provided as needed
- Microsoft Azure
  - $200 free credit per account valid for 30 days
  - https://azure.microsoft.com/en-us/free/?b=17.09c
- Google Cloud
  - New account credits $
- Chameleon / CloudLab
  - Bare metal NSF cloud - free

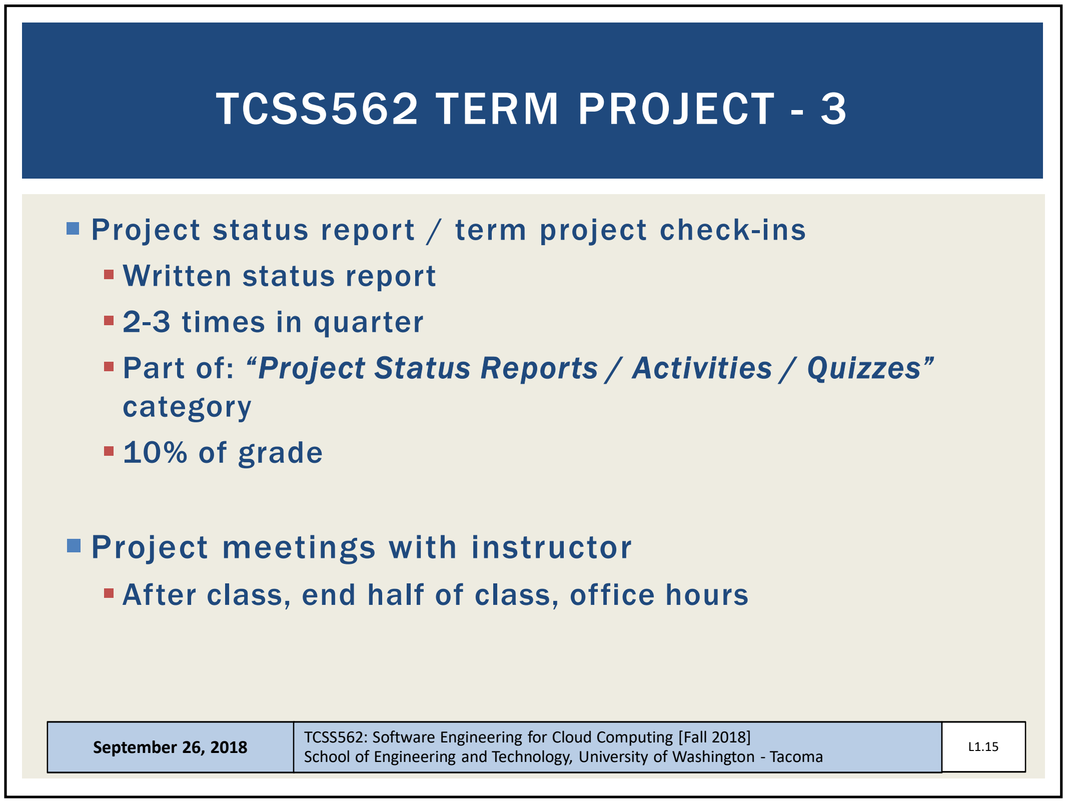
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.13



TCSS562 TERM PROJECT OPPORTUNITIES

- Projects can lead to papers or posters presented at ACM/IEEE/USENIX conferences, workshops
  - Networking and travel opportunity
  - Conference participation (posters, papers) helps differentiate your resume from others
- Project can support preliminary work for: UWT - MS capstone/thesis project proposals
- Research projects provide valuable practicum experience with cloud systems analysis, prototyping
- Publications are key when applying to PhD programs

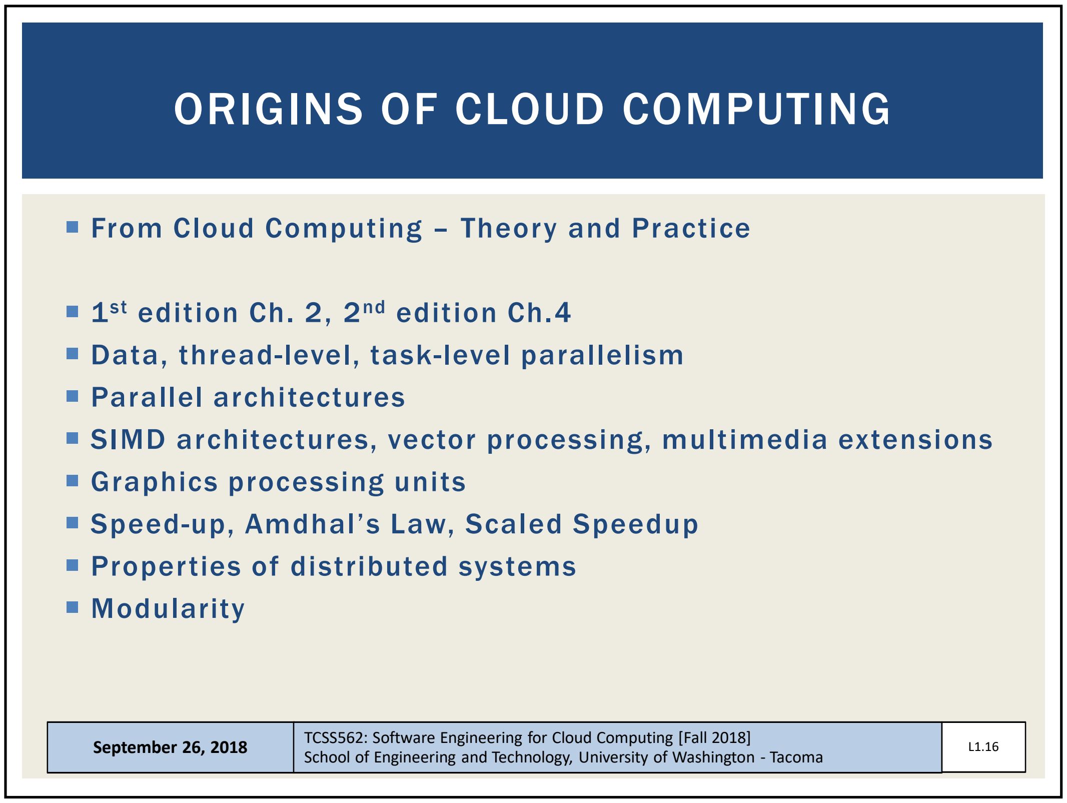
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.14



TCSS562 TERM PROJECT - 3

- Project status report / term project check-ins
  - Written status report
  - 2-3 times in quarter
  - Part of: ***"Project Status Reports / Activities / Quizzes"*** category
  - 10% of grade
- Project meetings with instructor
  - After class, end half of class, office hours

September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.15



ORIGINS OF CLOUD COMPUTING

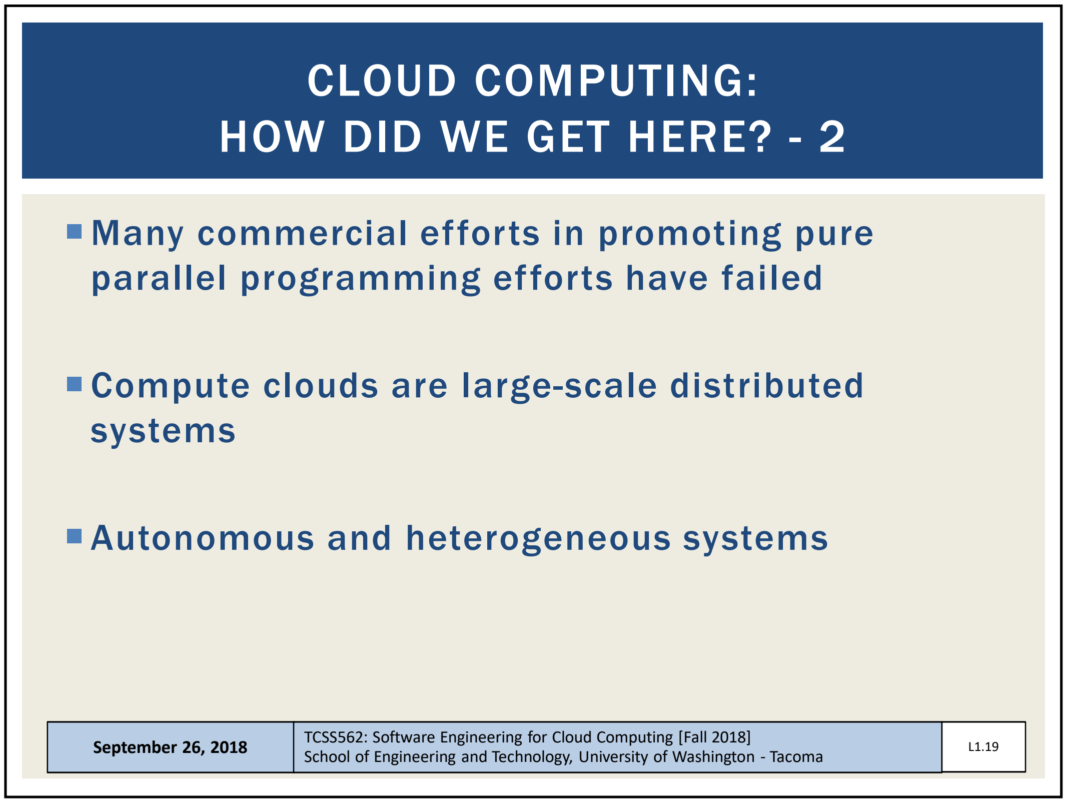
- From Cloud Computing – Theory and Practice
- 1st edition Ch. 2, 2nd edition Ch.4
- Data, thread-level, task-level parallelism
- Parallel architectures
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdhal's Law, Scaled Speedup
- Properties of distributed systems
- Modularity

September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.16

QUESTIONS

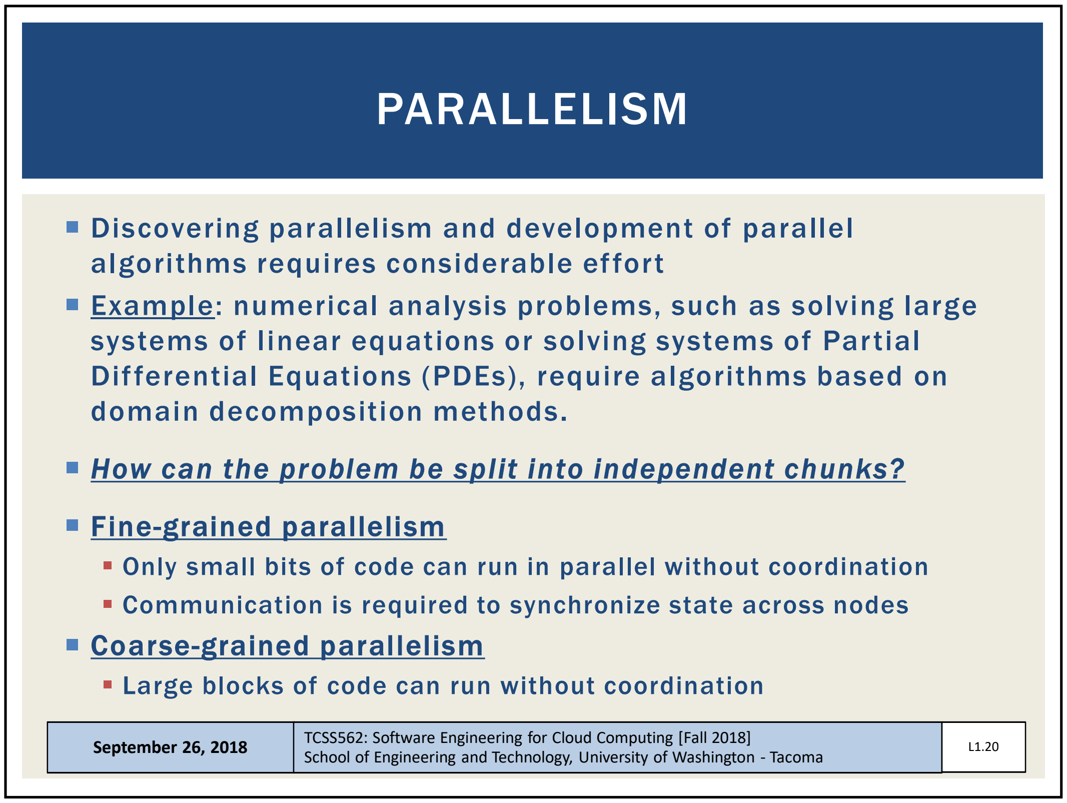
CLOUD COMPUTING: HOW DID WE GET HERE?

- Origins from parallel and distributed systems
- Difficult to expose parallelism in many scientific applications
- Enterprise computing world has been skeptical and less involved in parallel programming
- Multi-core technology has replaced faster CPU clock rates
- Cloud computing enables effortless exploitation of parallelism
- Big Data requires massive amounts of compute resources
- Cloud applications
  - Based on client-server paradigm
  - Thin clients leverage compute hosted on the cloud
  - Applications run many web service instances
  - Employ load balancing



CLOUD COMPUTING: HOW DID WE GET HERE? - 2

- Many commercial efforts in promoting pure parallel programming efforts have failed
- Compute clouds are large-scale distributed systems
- Autonomous and heterogeneous systems



PARALLELISM

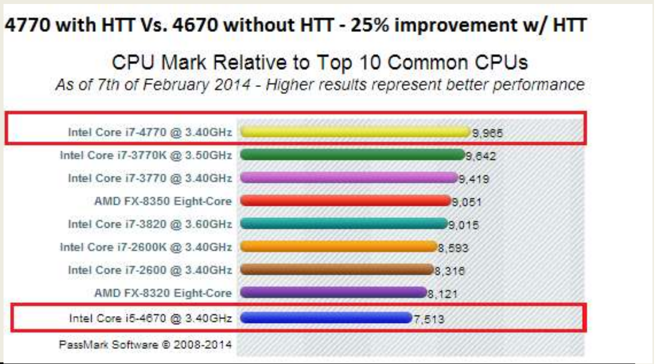
- Discovering parallelism and development of parallel algorithms requires considerable effort
- Example: numerical analysis problems, such as solving large systems of linear equations or solving systems of Partial Differential Equations (PDEs), require algorithms based on domain decomposition methods.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

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
  - Shared memory: interactions can be implicit
  - Processing speed orders of magnitude faster than communication
  - Avoiding coordination achieves the best speed-up
- **Data-level parallelism**
  - Partition data into big chunks, run separate copies of the program on them with little or no communication
  - This is embarrassingly parallel
  - MapReduce programming model is an example

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



THREAD LEVEL PARALLELISM (TLP)

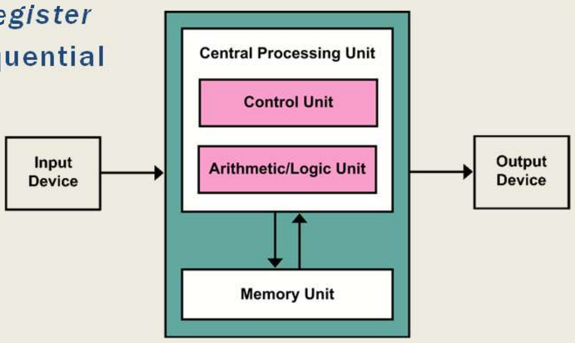
- Number of threads that an application runs at any one time
- Varies throughout program execution
- As a metric:
- <u>Minimum</u>: 1 thread
- Can measure <u>average</u>, <u>maximum</u>

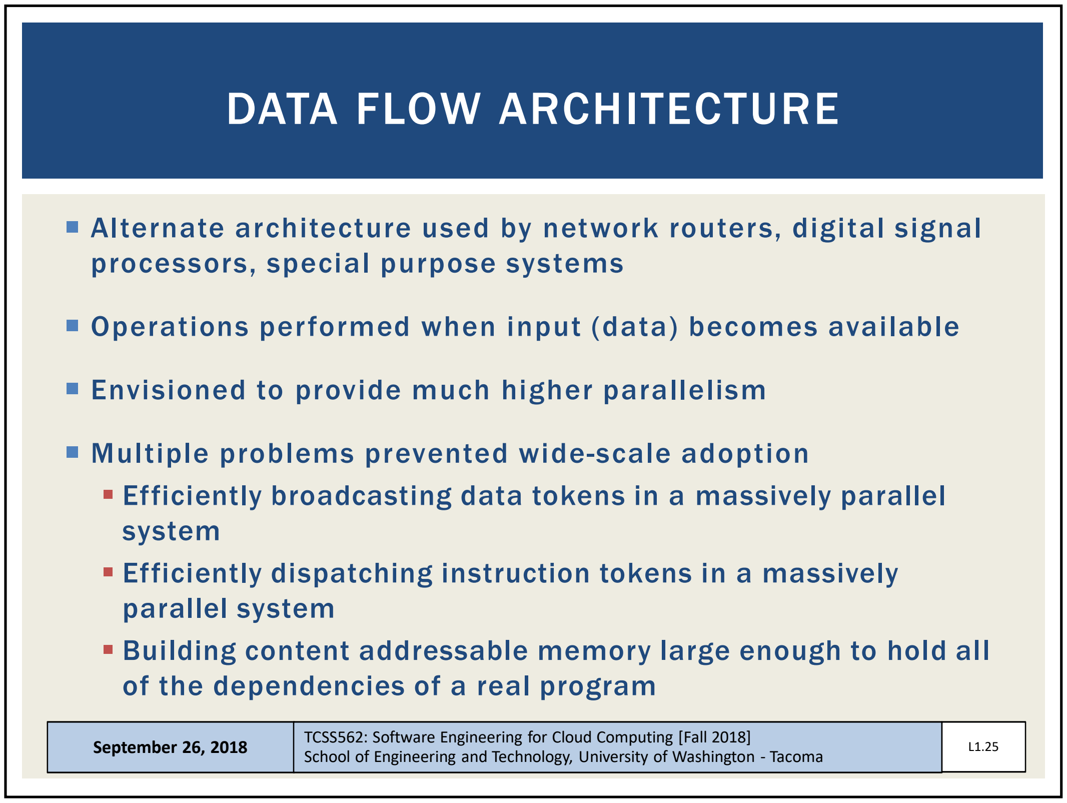
- **What are the consequences of <u>average</u> (TLP) for scheduling an application to run on a computer with a fixed number of CPU cores and hyperthreads?**

- Let's say there are 4 cores, or 8 hyper-threads…

CONTROL-FLOW ARCHITECTURE

- 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

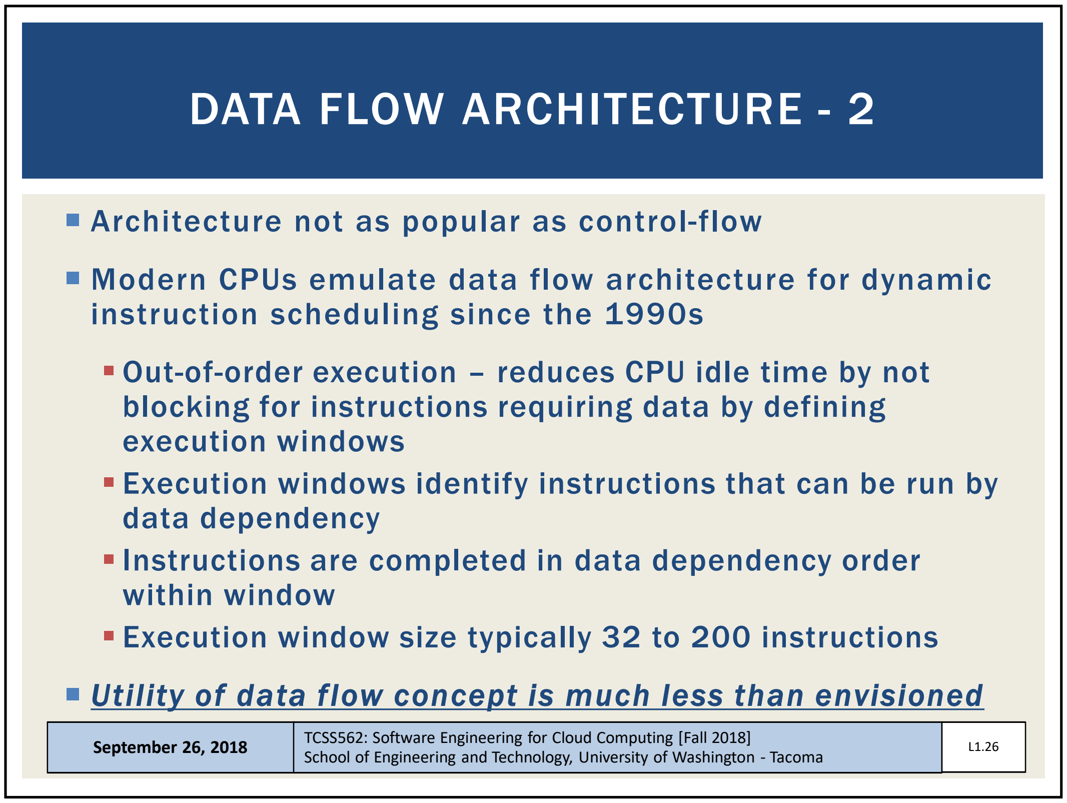




DATA FLOW ARCHITECTURE

- 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 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

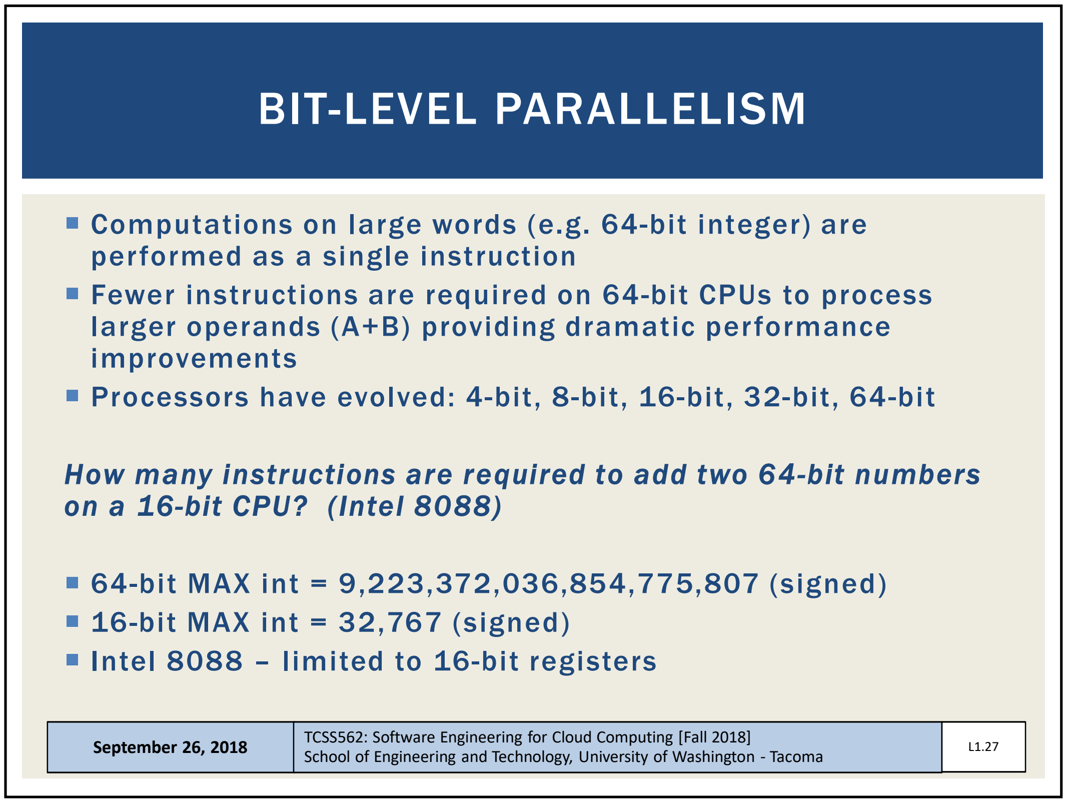
September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.25



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 window
  - Execution window size typically 32 to 200 instructions
- ***Utility of data flow concept is much less than envisioned***

September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.26

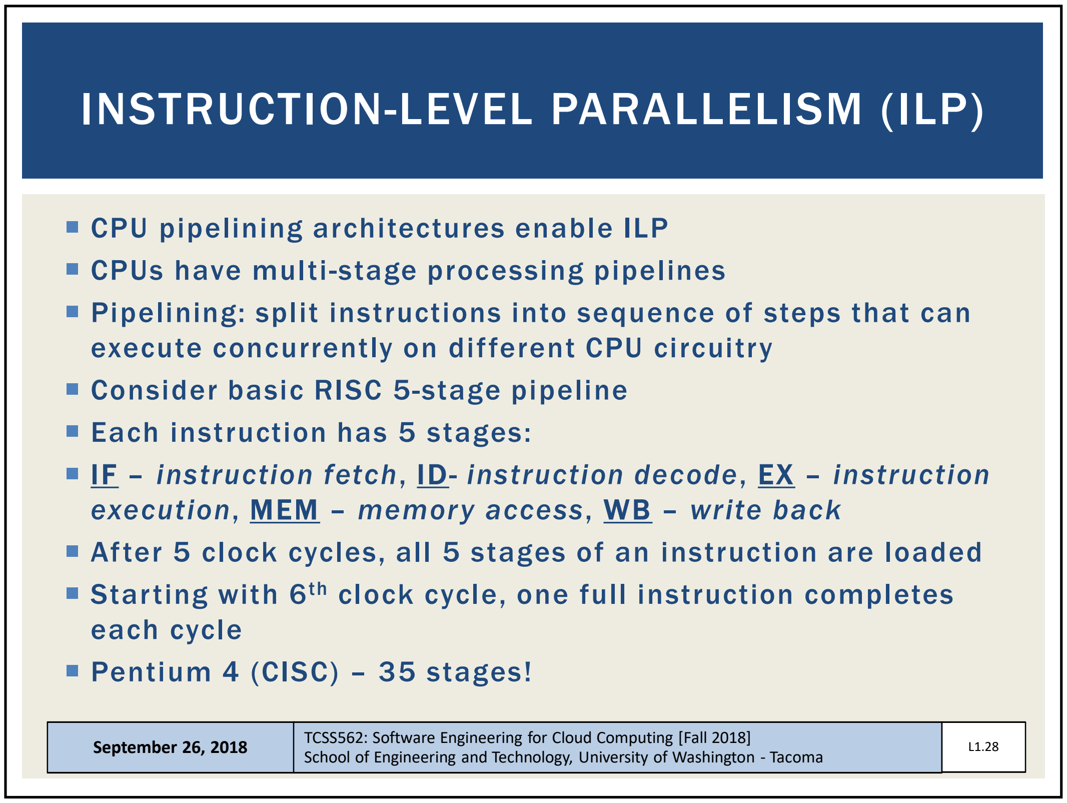


BIT-LEVEL PARALLELISM

- 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

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

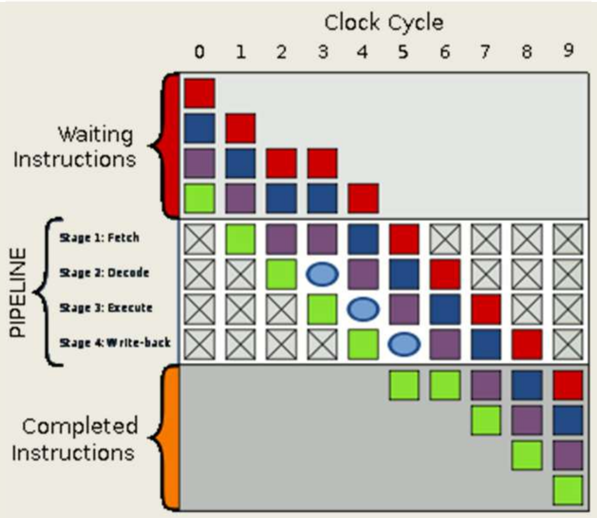
- 64-bit MAX int = 9,223,372,036,854,775,807 (signed)
- 16-bit MAX int = 32,767 (signed)
- Intel 8088 – limited to 16-bit registers



INSTRUCTION-LEVEL PARALLELISM (ILP)

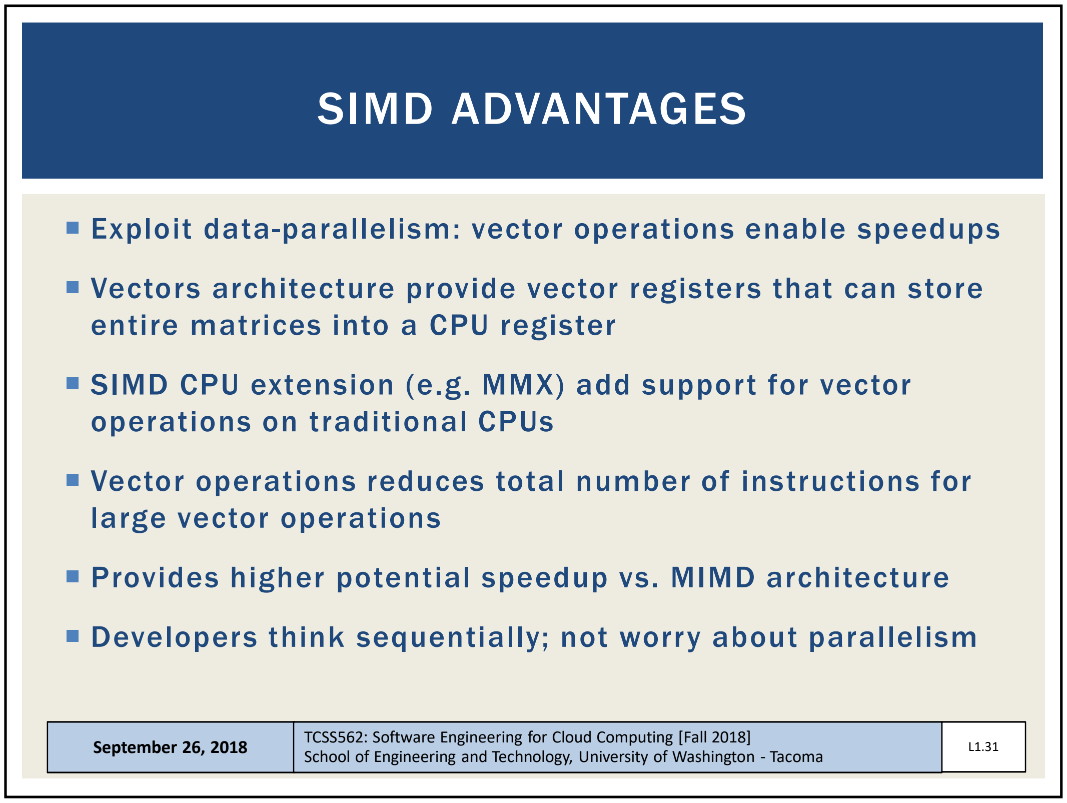
- CPU pipelining architectures enable ILP
- CPUs have multi-stage processing pipelines
- Pipelining: split instructions into sequence of steps that can execute concurrently on different CPU circuitry
- Consider basic RISC 5-stage pipeline
- Each instruction has 5 stages:
- **IF** – *instruction fetch*, **ID**- *instruction decode*, **EX** – *instruction execution*, **MEM** – *memory access*, **WB** – *write back*
- After 5 clock cycles, all 5 stages of an instruction are loaded
- Starting with 6th clock cycle, one full instruction completes each cycle
- Pentium 4 (CISC) – 35 stages!

CPU PIPELINING



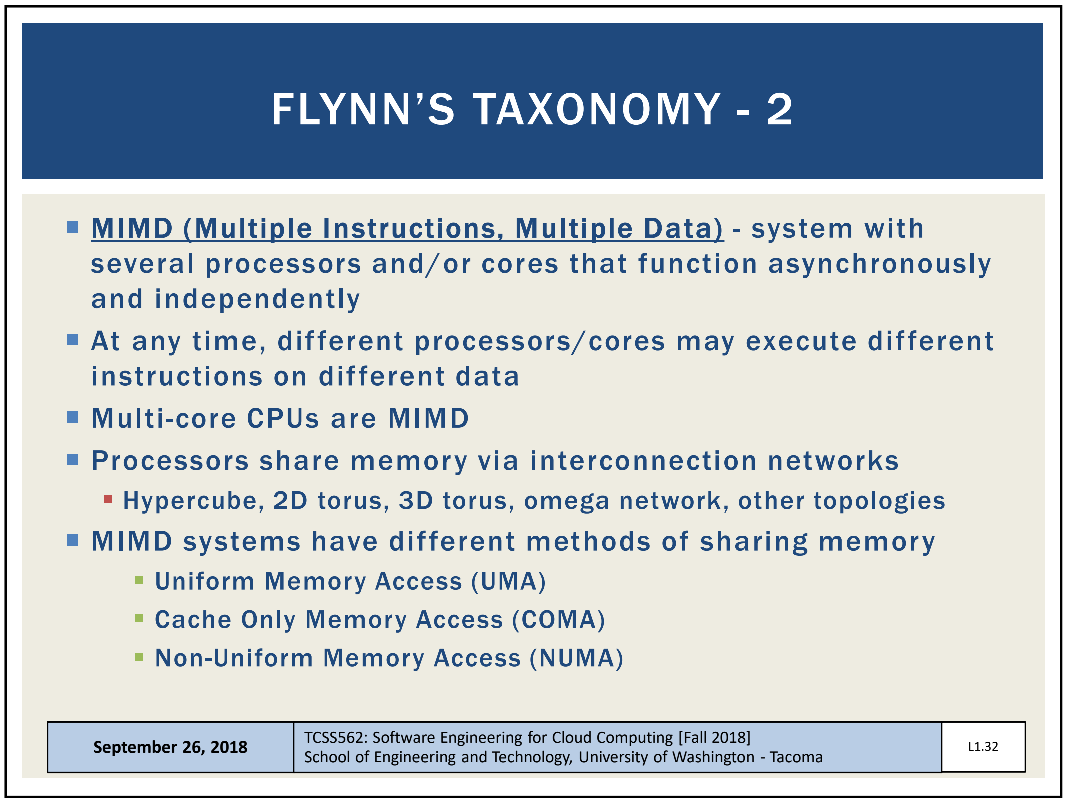
MICHAEL FLYNN'S COMPUTER ARCHITECTURE TAXONOMY

- Michael Flynn's proposed taxonomy of computer architectures based on concurrent instructions and number of data streams (1966)
- SISD (Single Instruction Single Data) – scalar architecture with one processor/core.
  - Individual cores of modern multicore processors are "SISD"
- SIMD (Single Instruction, Multiple Data) - supports vector processing. When a SIMD instruction is issued, the operations on individual vector components are carried out concurrently.
  - Elements of two 64-element vectors can be added in parallel
  - Vector processing instructions have been added to modern CPUs
  - Example: Intel MMX (multimedia) instructions



SIMD 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 reduces total number of instructions for large vector operations
- Provides higher potential speedup vs. MIMD architecture
- Developers think sequentially; not worry about parallelism



FLYNN'S TAXONOMY - 2

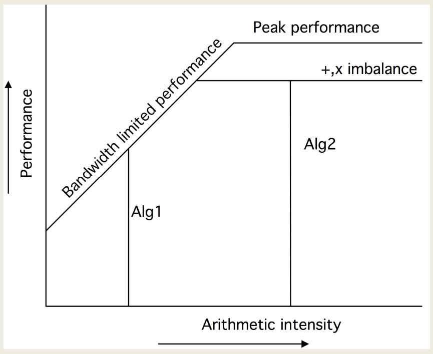
- MIMD (Multiple Instructions, Multiple Data) - system with several processors and/or cores that function asynchronously and independently
- At any time, different processors/cores may execute different instructions on different data
- Multi-core CPUs are MIMD
- Processors share memory via interconnection 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)

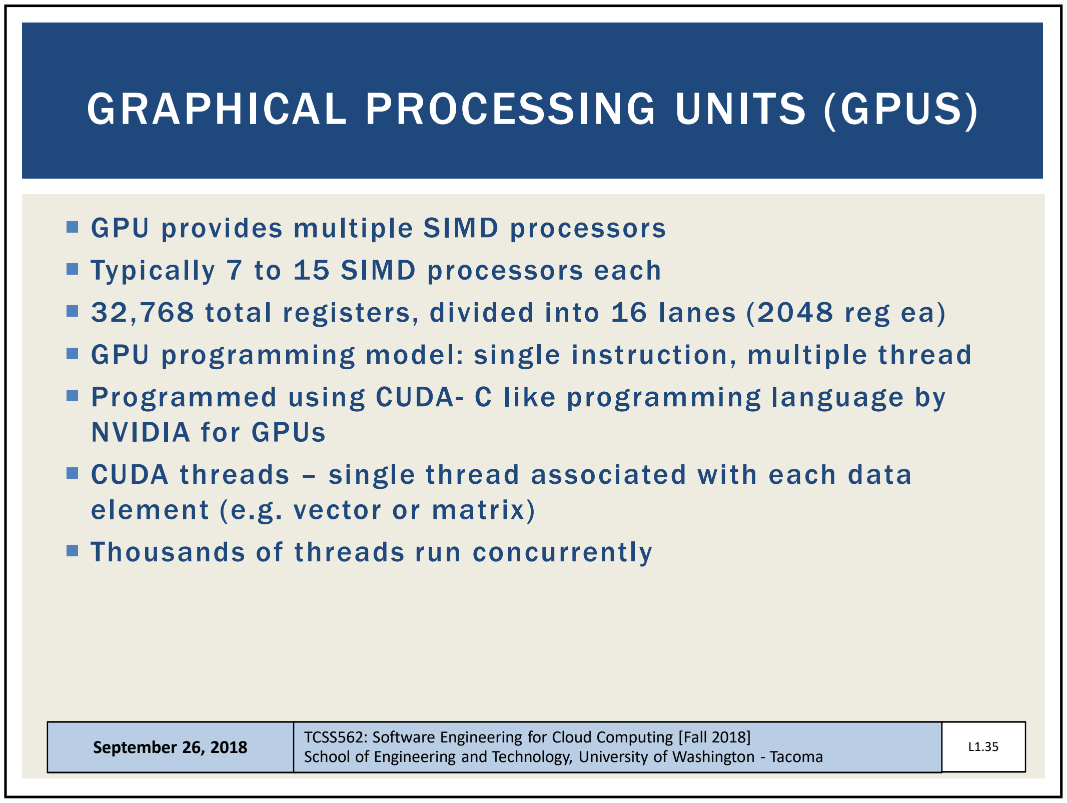
ARITHMETIC INTENSITY

- <u>Arithmetic intensity</u> – number of floating point operations per byte of data read
- SIMD provides fast matrix operations
- Characterizes application scalability based on SIMD support
- ***High arithmetic Intensity:***
  Programs with dense matrix operations scale up nicely
- ***Low arithmetic intensity:***
  Programs with sparse matrix operations do not scale well with problem size

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)

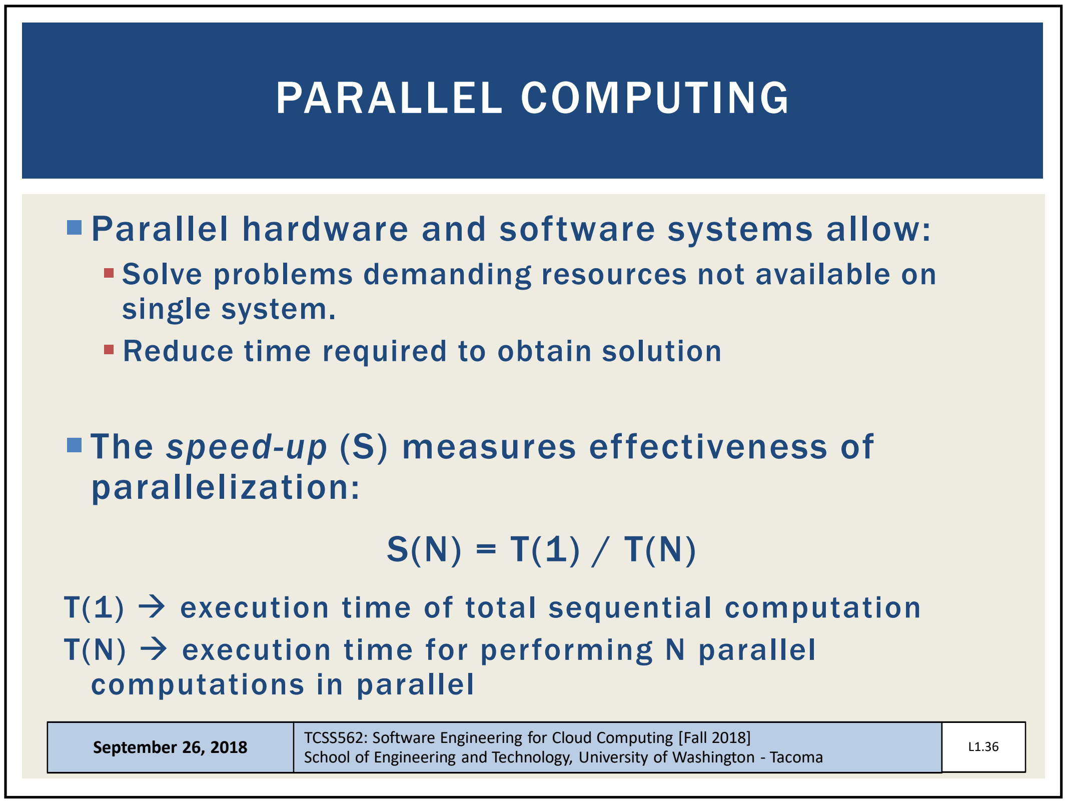




GRAPHICAL PROCESSING UNITS (GPUS)

- GPU provides multiple SIMD processors
- Typically 7 to 15 SIMD processors each
- 32,768 total registers, divided into 16 lanes (2048 reg ea)
- 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

September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.35



PARALLEL COMPUTING

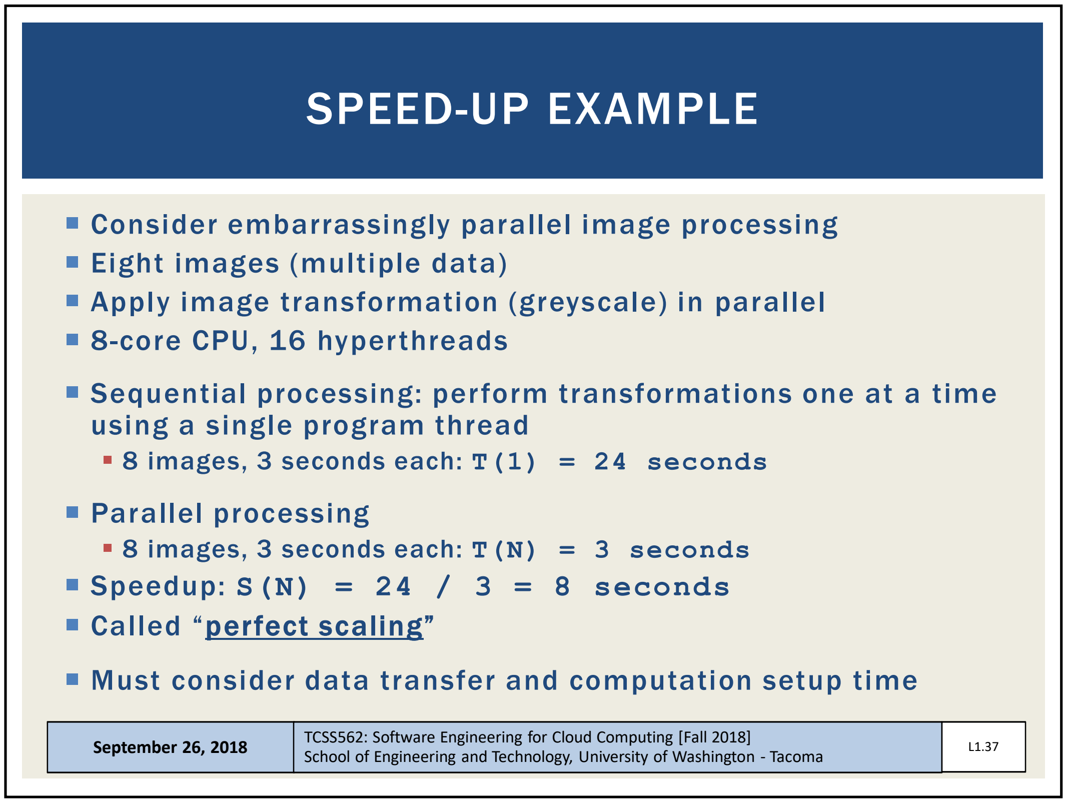
- Parallel hardware and software systems allow:
  - Solve problems demanding resources not available on single system.
  - Reduce time required to obtain solution

- The *speed-up* (S) measures effectiveness of parallelization:

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

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

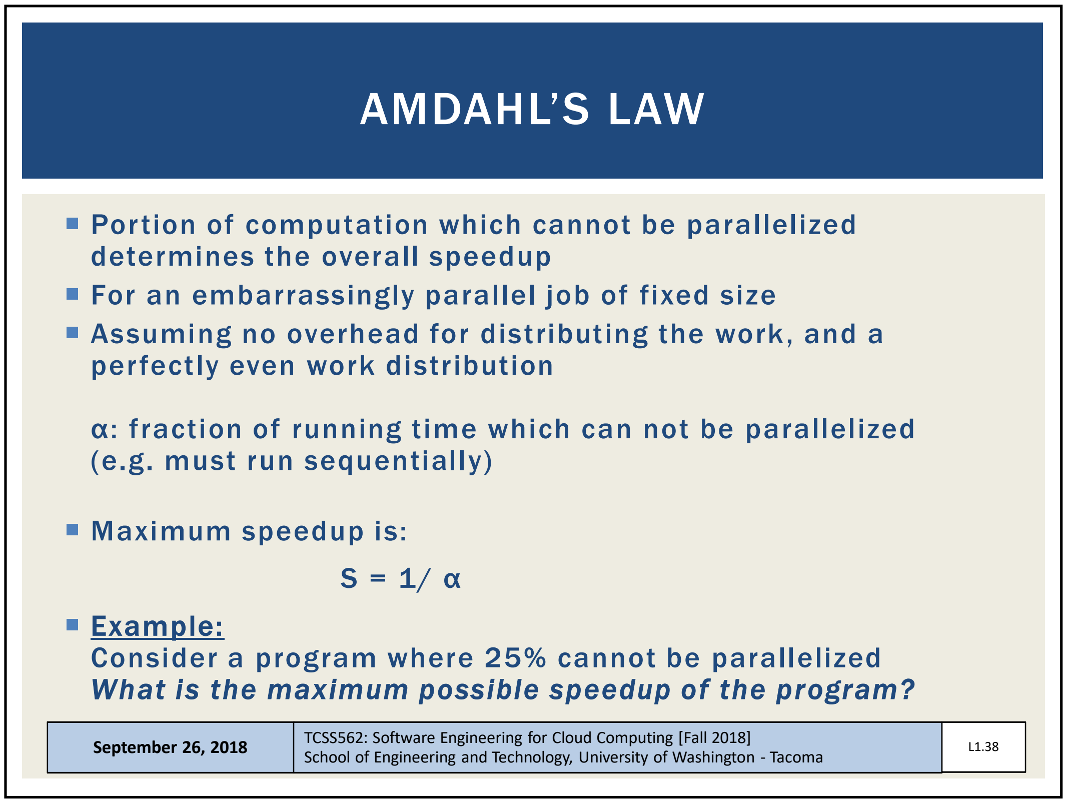
September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.36



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 = 8 seconds`
- Called "perfect scaling"
- Must consider data transfer and computation setup time

September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.37



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 running time which can not be parallelized (e.g. must run sequentially)

- Maximum speedup is:

$$S = 1/\alpha$$

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

September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.38

GUSTAFSON'S LAW

- Calculates the ***scaled speed-up*** with "N" processes

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

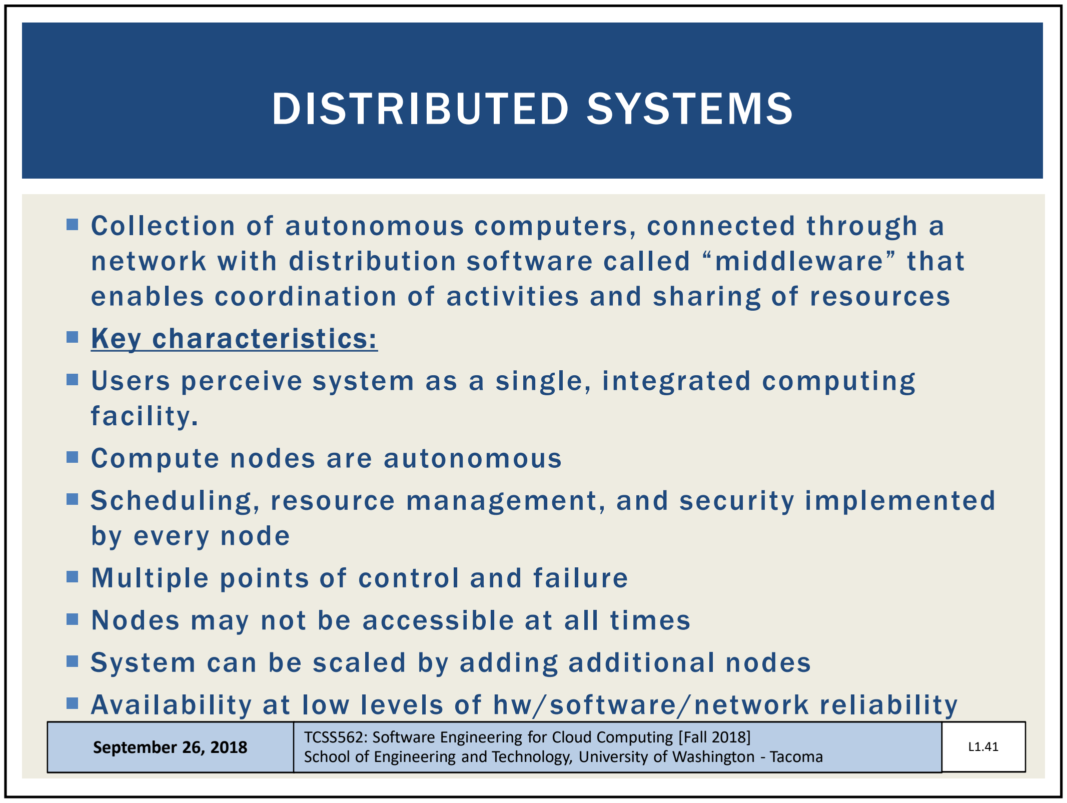
N: Number of processes

α: fraction of running time which can not be parallelized (e.g. must run sequentially)

- Example:
  Consider a program that is embarrassingly parallel but where 25% cannot be parallelized.
  If deploying the job on a 2-core CPU, what scaled speedup is possible assuming the two processes can run in parallel?

MOORE'S LAW

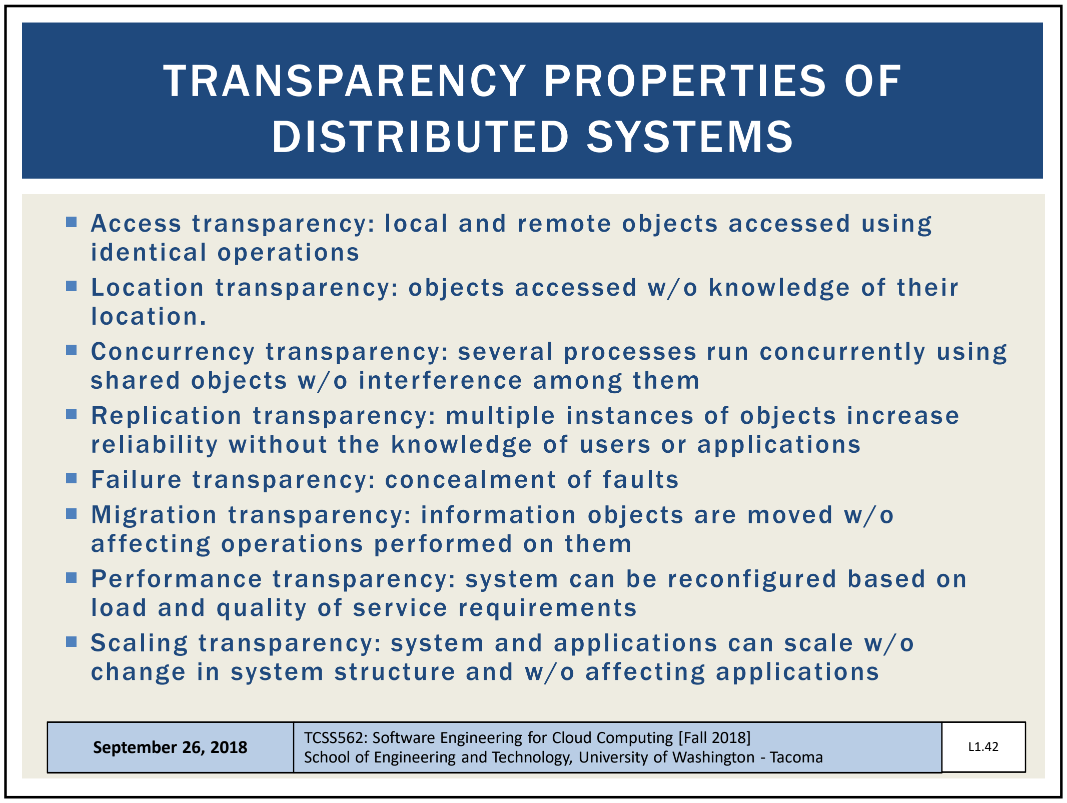
- Transistors on a chip doubles approximately every 1.5 years
- CPUs now have billions of transistors
- Power dissipation issues at faster clock rates leads to heat removal challenges
  - Transition from: increasing clock rates → to adding CPU cores
- ***Symmetric core processor*** – multi-core CPU, 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***



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

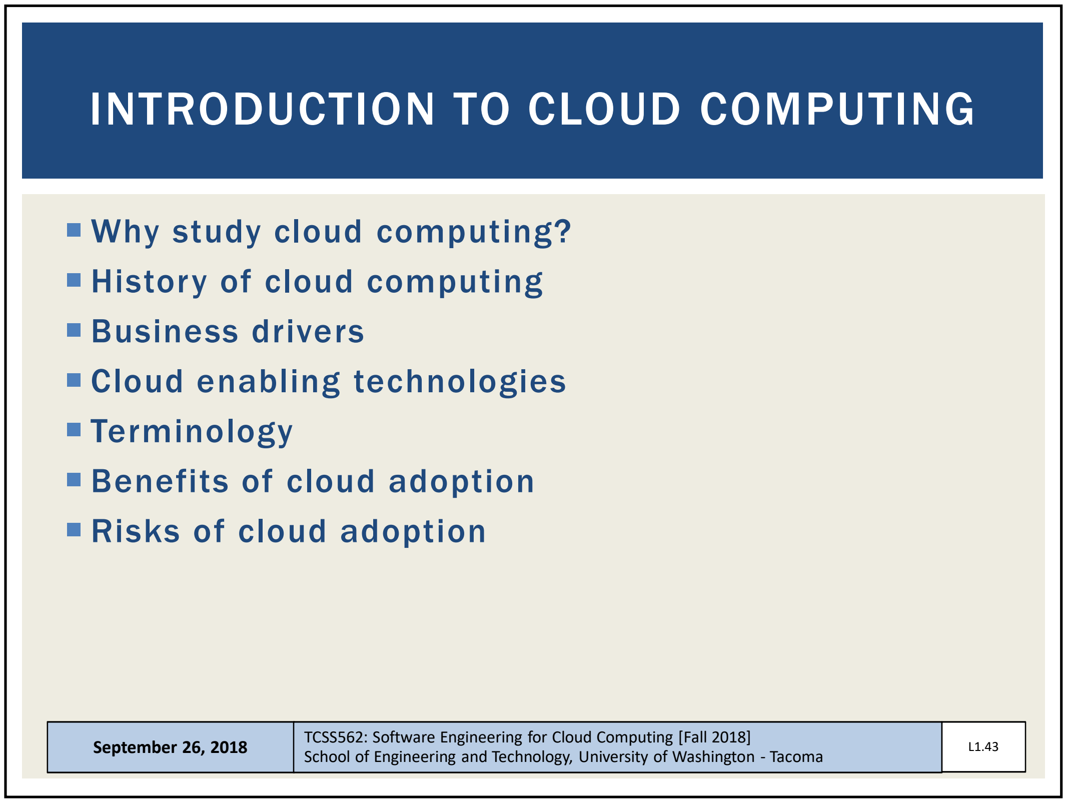
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.41



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 increase reliability without the knowledge of users or applications
- Failure transparency: concealment of faults
- Migration transparency: information 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

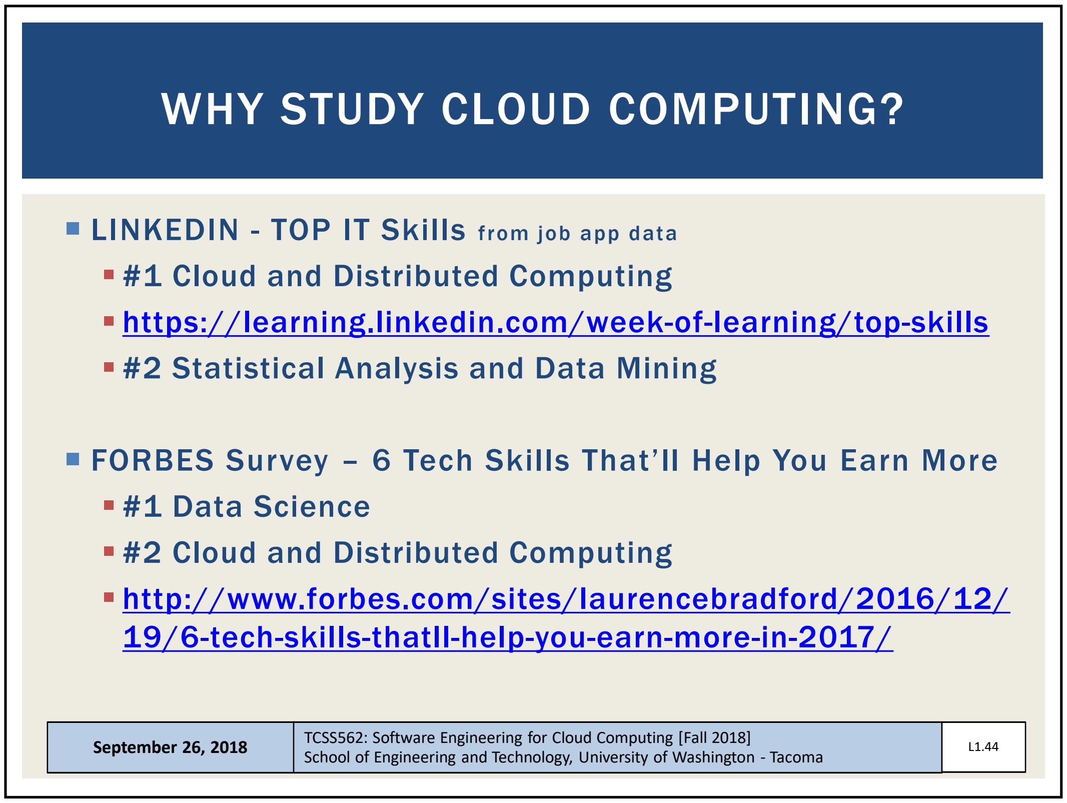
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.42



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 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.43



WHY STUDY CLOUD COMPUTING?

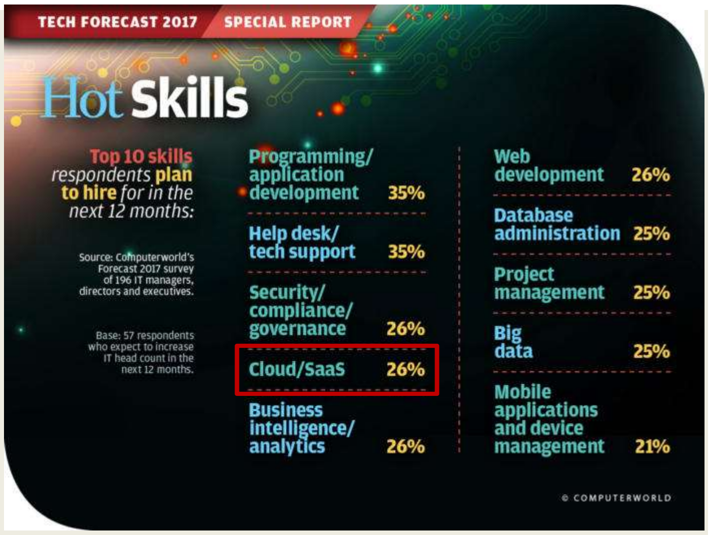
- LINKEDIN - TOP IT Skills from job app data
  - #1 Cloud and Distributed Computing
  - https://learning.linkedin.com/week-of-learning/top-skills
  - #2 Statistical Analysis and Data Mining

- 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 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.44

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 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.45

A BRIEF HISTORY OF CLOUD COMPUTING

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

September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.46

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

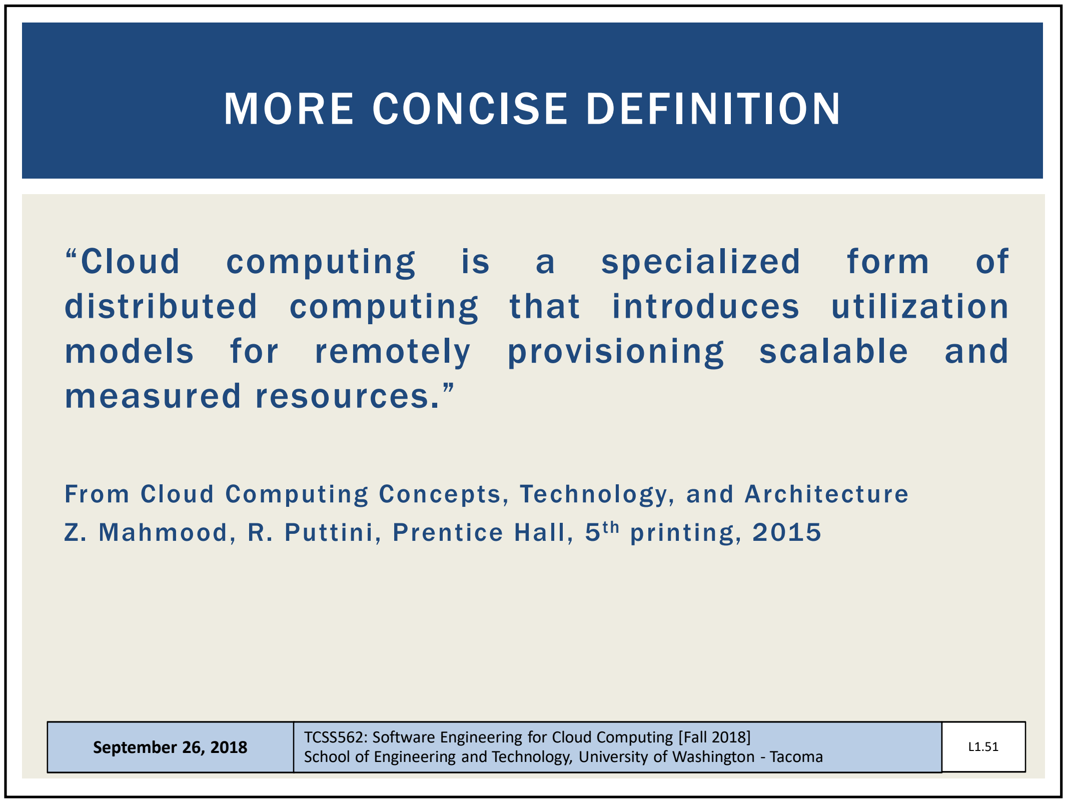
- 2006 – Infrastructure-as-a-Service (IaaS)
  - Amazon launches Elastic Compute Cloud (EC2) service
  - Organization can “lease” computing capacity and processing power to host enterprise applications
  - Infrastructure

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

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"…

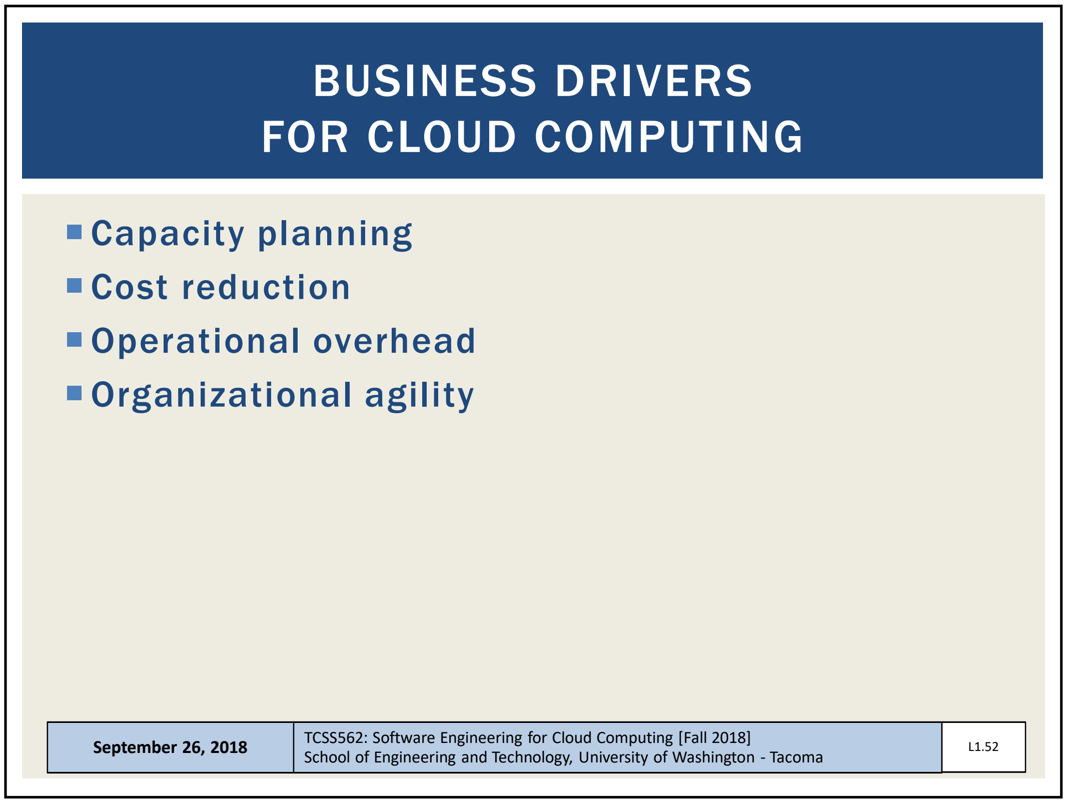


MORE CONCISE DEFINITION

"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

September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.51



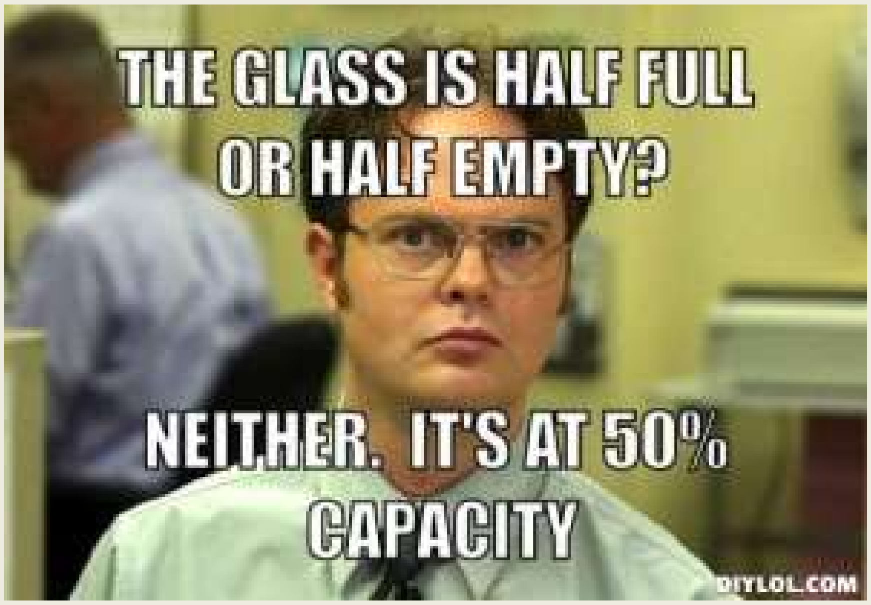
BUSINESS DRIVERS FOR CLOUD COMPUTING

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

September 26, 2018	TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L1.52

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

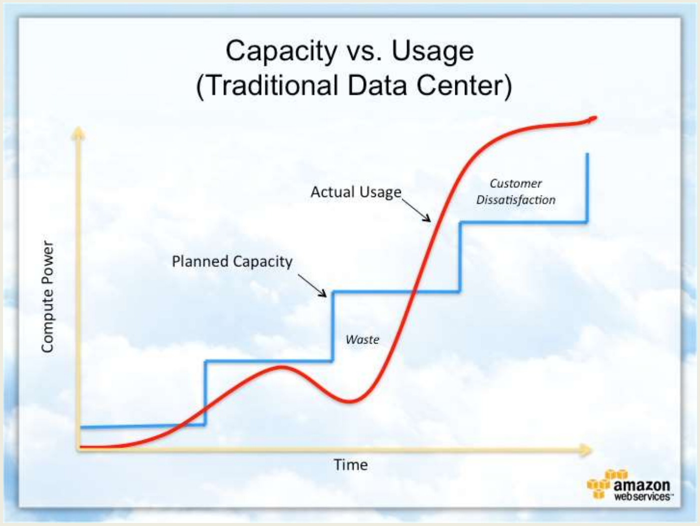

THE GLASS IS HALF FULL OR HALF EMPTY?
NEITHER. IT'S AT 50% CAPACITY
DIYLOL.COM

Dwight, The Office TV sitcom

BUSINESS DRIVERS FOR CLOUD - 2

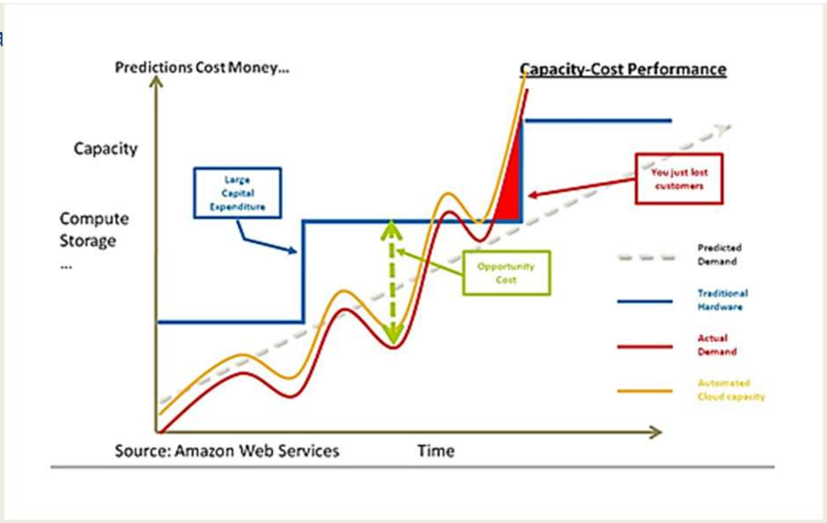
- Capacity planning
  - Over-provisioning: is costly due to too much infrastructure
  - Under-provisioning: is costly due to potential for business loss from poor quality of service
- Capacity planning strategies
  - Lead strategy: add capacity in anticipation of demand (pre-provisioning)
  - Lag strategy: 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

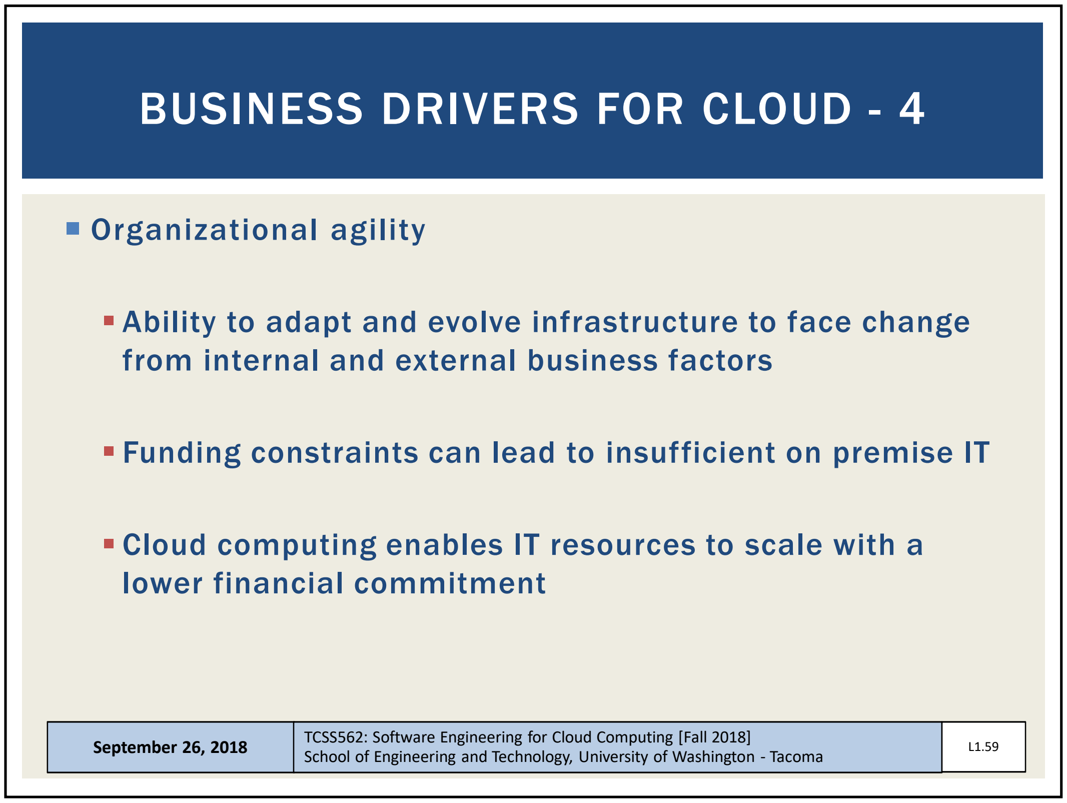


September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L1.57

BUSINESS DRIVERS FOR CLOUD - 3

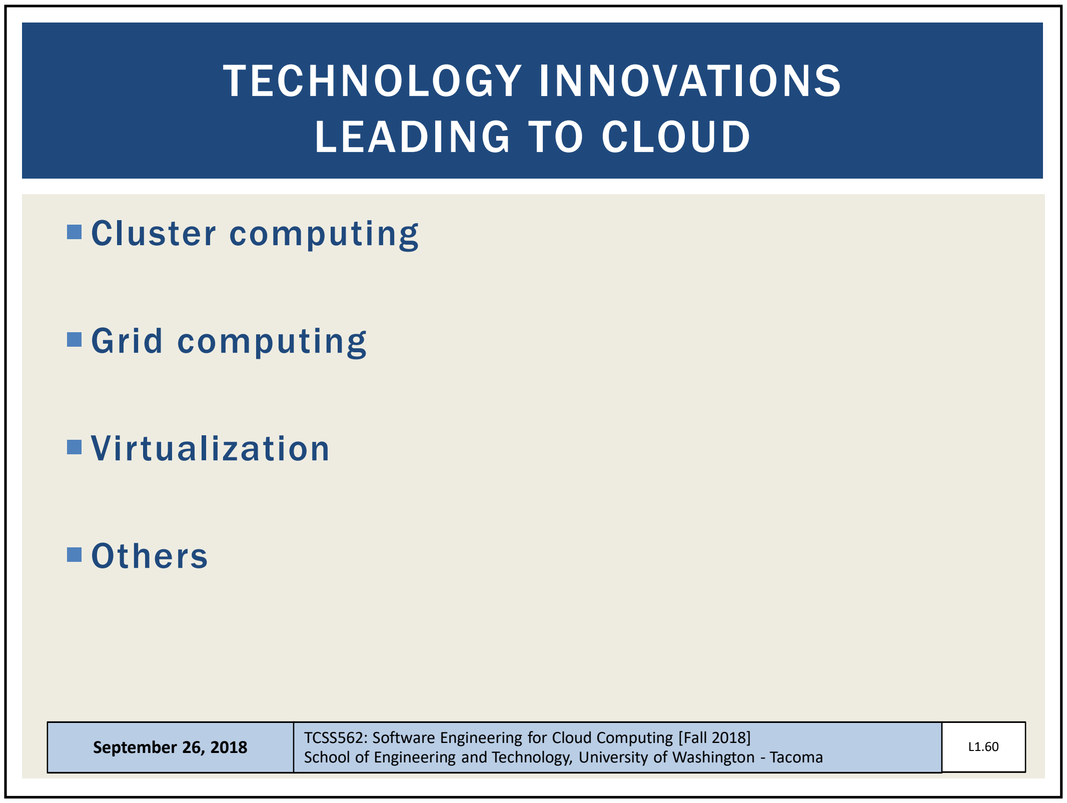
- Cost reduction
  - IT Infrastructure acquisition
  - IT Infrastructure maintenance
- Operational overhead
  - Technical personnel to maintain physical IT infrastructure
  - System upgrades, patches that add testing to deployment cycles
  - Utility bills, capital investments for power and cooling
 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

September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L1.58



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



TECHNOLOGY INNOVATIONS LEADING TO CLOUD

- Cluster computing
- Grid computing
- Virtualization
- Others

CLUSTER COMPUTING

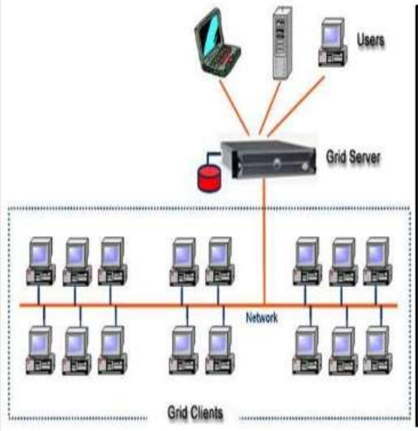
- Cluster computing (clustering)
  - Cluster is a group of independent IT resources interconnected as a single system
  - Servers configured with homogeneous hardware and software
    - Identical or similar RAM, CPU, HDDs
  - Design emphasizes redundancy as server components are easily interchanged to 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

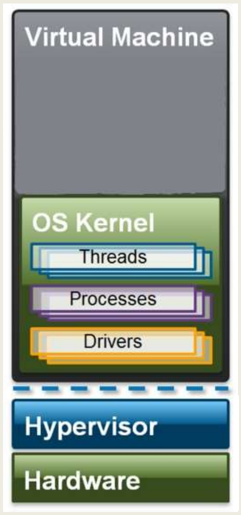
How Grid computing works ?



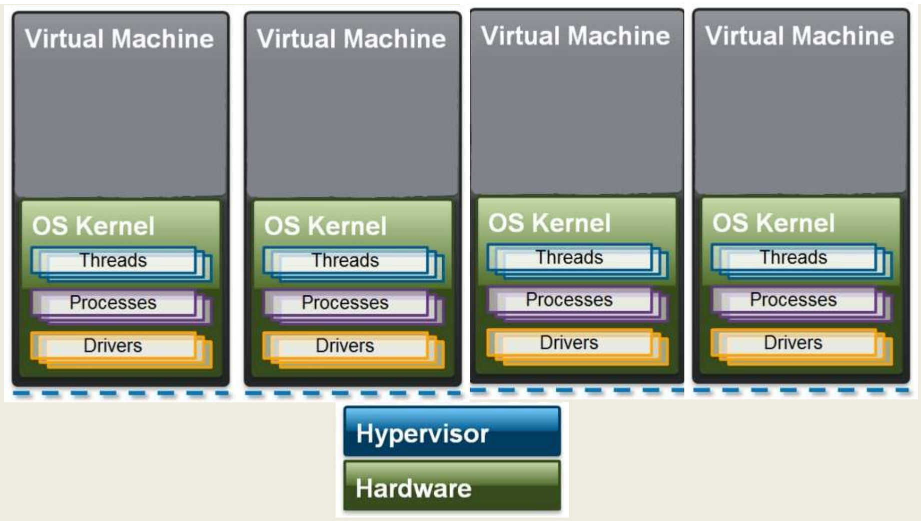
In general, a grid computing system requires:

- **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

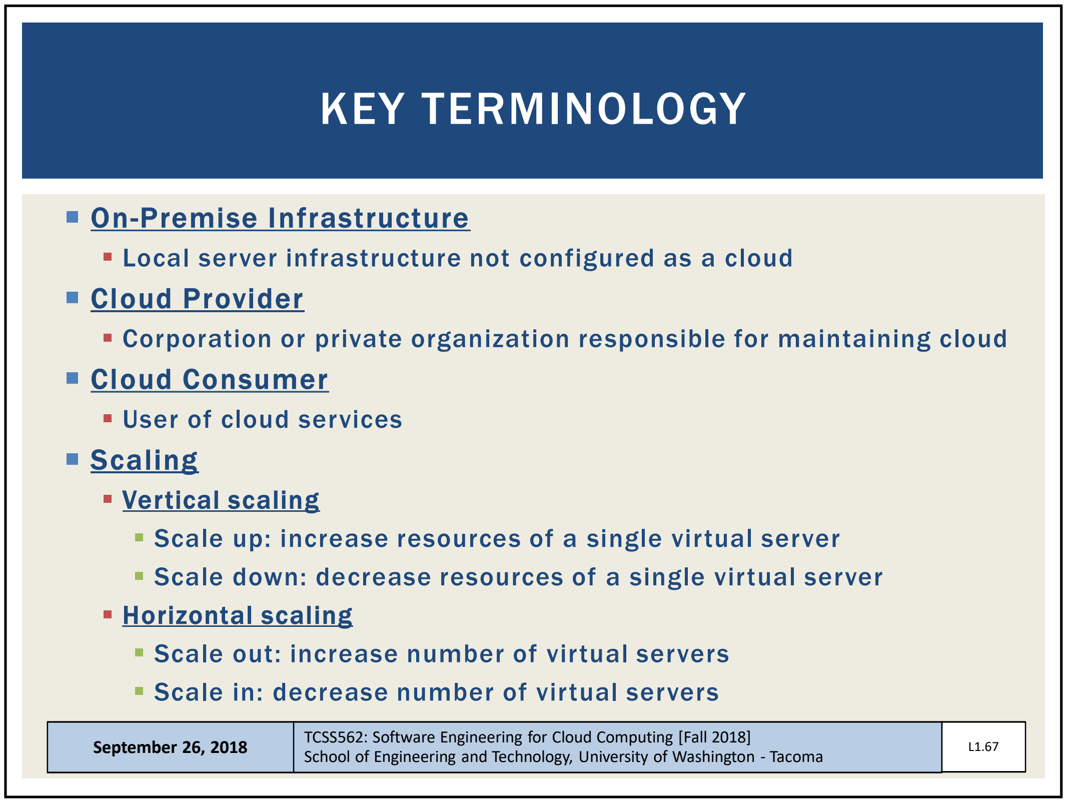


VIRTUALIZATION

- 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

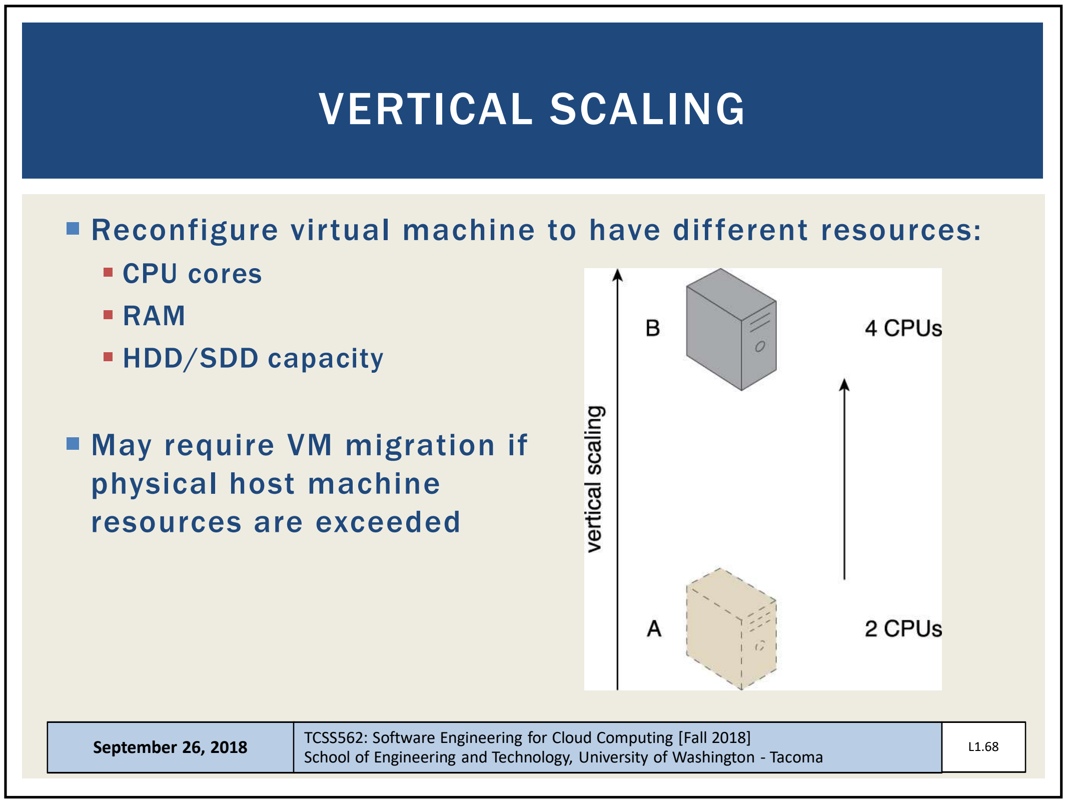
- CPU and other HW enhancements have minimized performance GAPs



KEY TERMINOLOGY

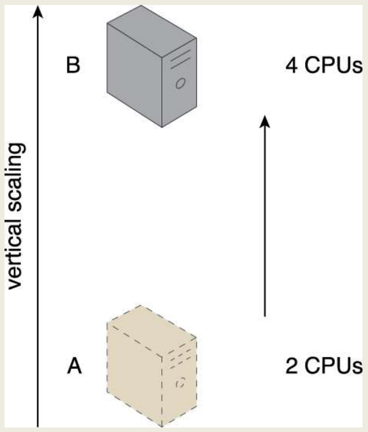
- **On-Premise Infrastructure**
  - Local server infrastructure not configured as a cloud
- **Cloud Provider**
  - Corporation or private organization responsible for maintaining cloud
- **Cloud Consumer**
  - User of cloud services
- **Scaling**
  - **Vertical scaling**
    - Scale up: increase resources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

September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.67



VERTICAL SCALING

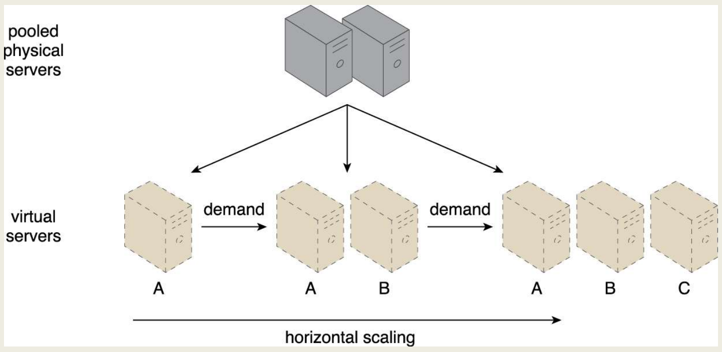
- Reconfigure virtual machine to have different resources:
  - CPU cores
  - RAM
  - HDD/SDD capacity
- May require VM migration if physical host machine resources are exceeded



September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.68

HORIZONTAL SCALING

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



HORIZONTAL VS VERTICAL SCALING

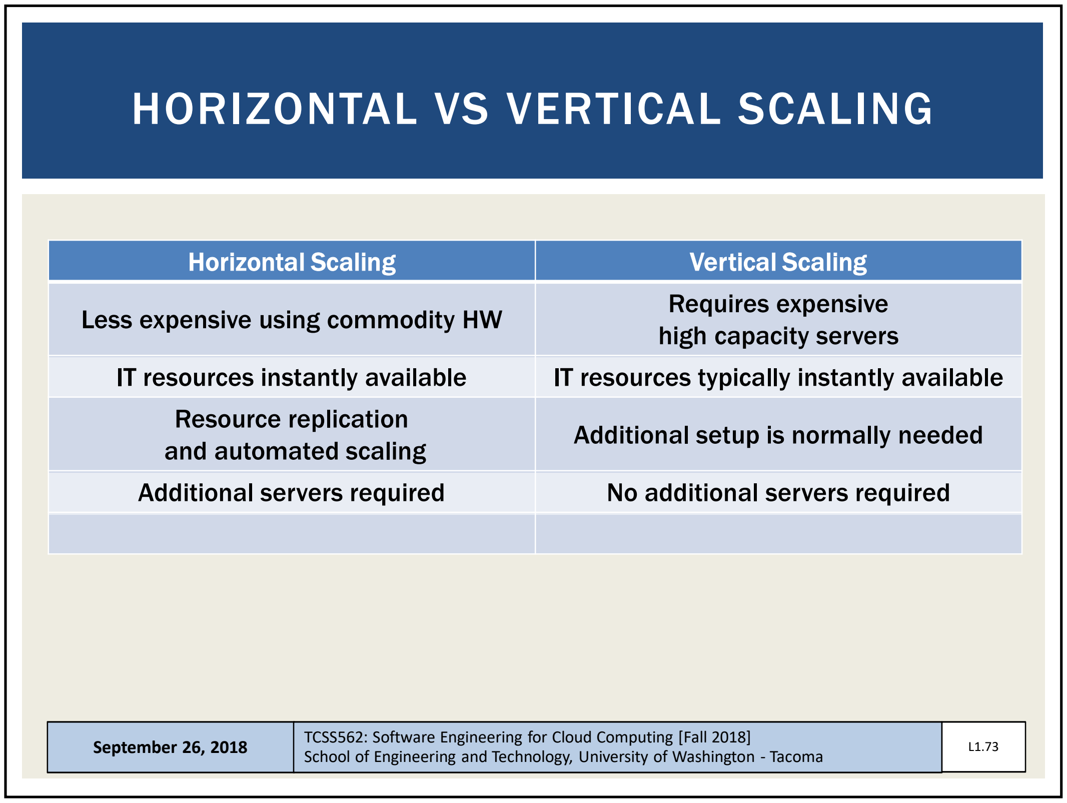
Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers

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

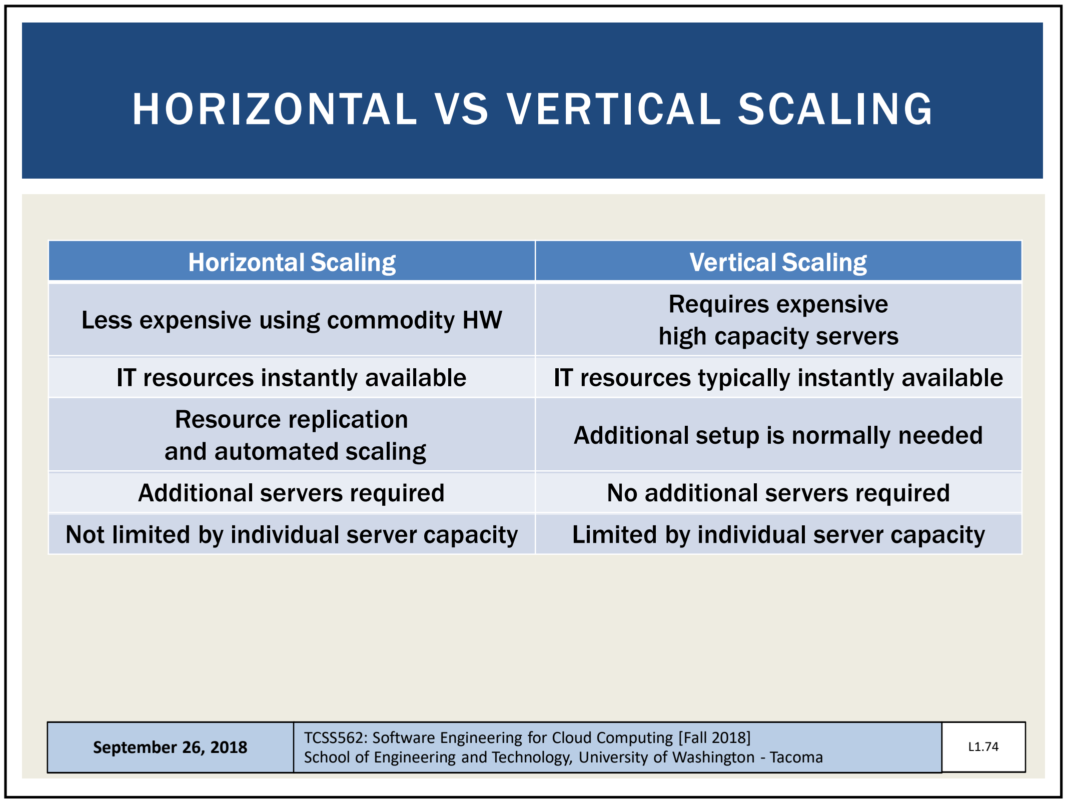
Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available
Resource replication and automated scaling	Additional setup is normally needed



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

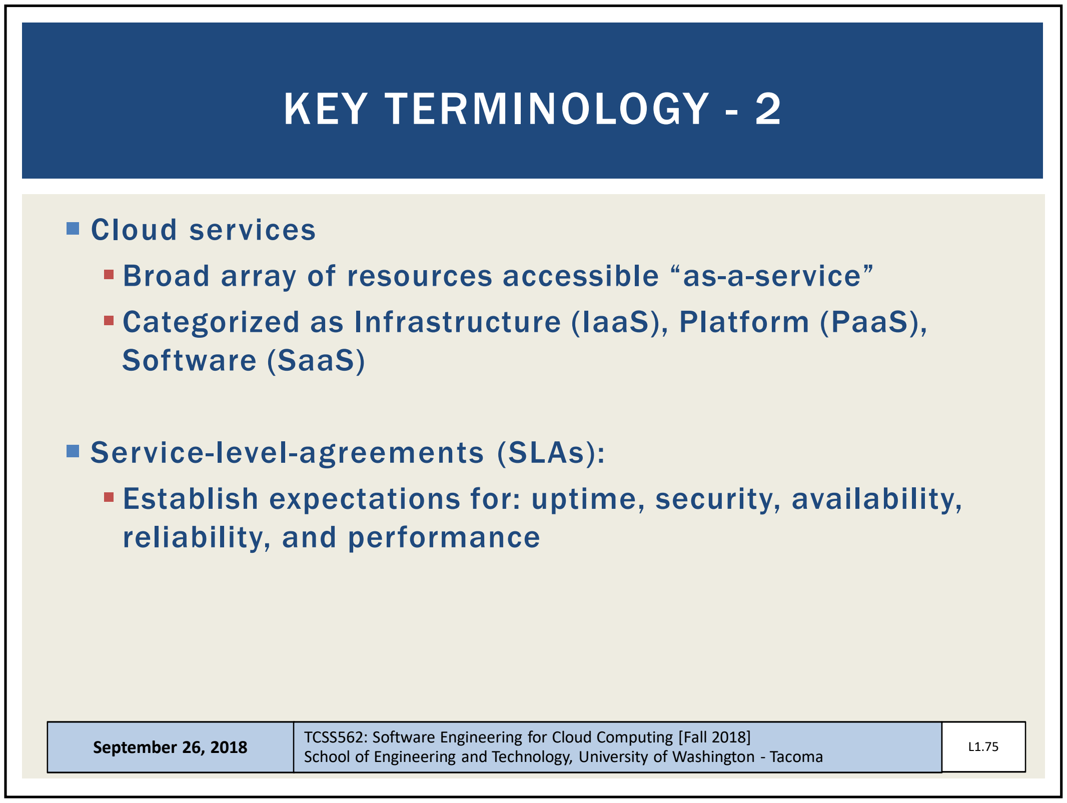
September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.73



HORIZONTAL VS VERTICAL SCALING

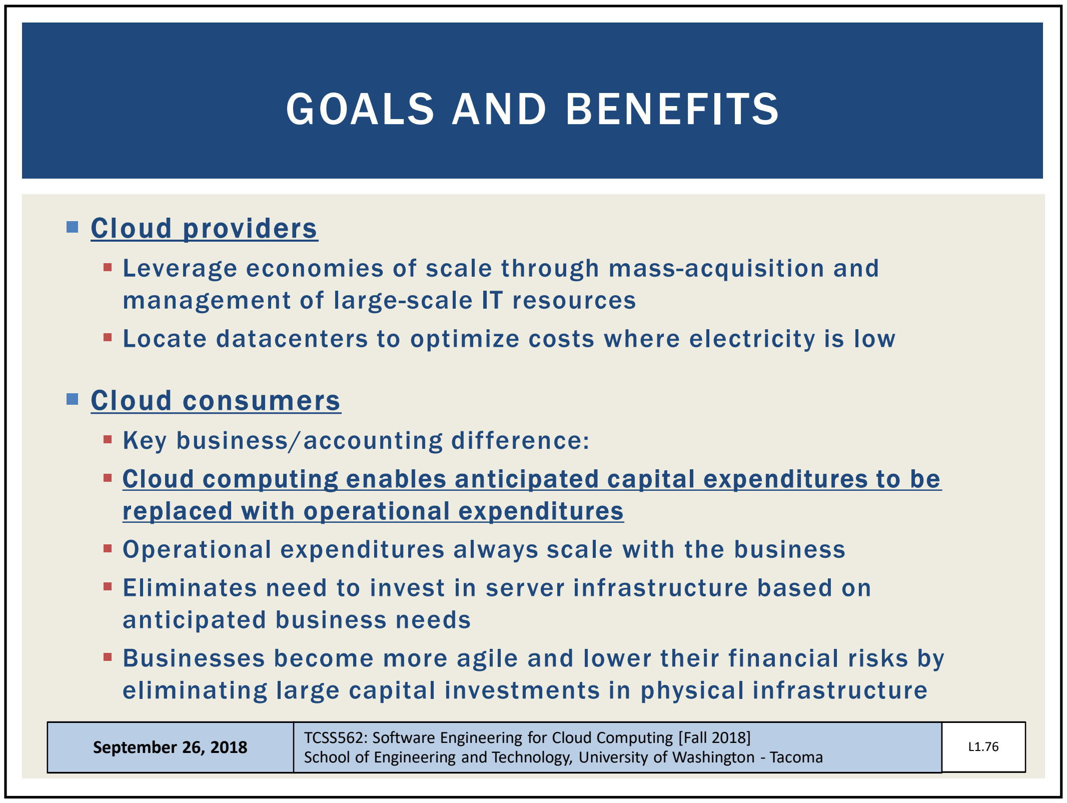
Horizontal Scaling	Vertical Scaling
Less expensive using commodity HW	Requires expensive high capacity servers
IT resources instantly available	IT resources typically instantly available
Resource replication and automated scaling	Additional setup is normally needed
Additional servers required	No additional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

September 26, 2018 | TCSS562: Software Engineering for Cloud Computing [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L1.74



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



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

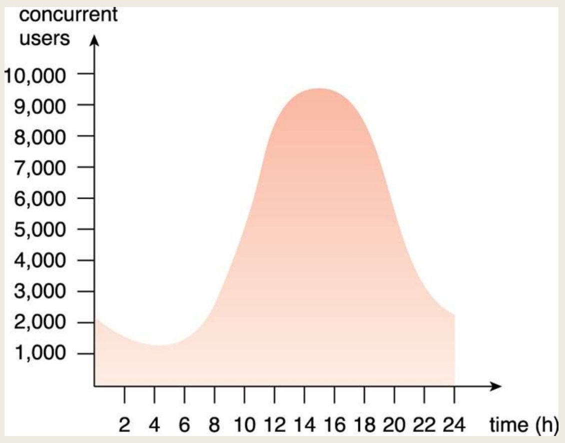
- Example: Using 100 servers for 1 hour costs the same as using 1 server for 100 hours
- Rosetta Protein Folding: Working with a UW-Tacoma graduate student, we recently deployed this science model across 5,900 compute cores on Amazon for 2-days...
- ***What is the cost to purchase 5,900 compute cores?***
- 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

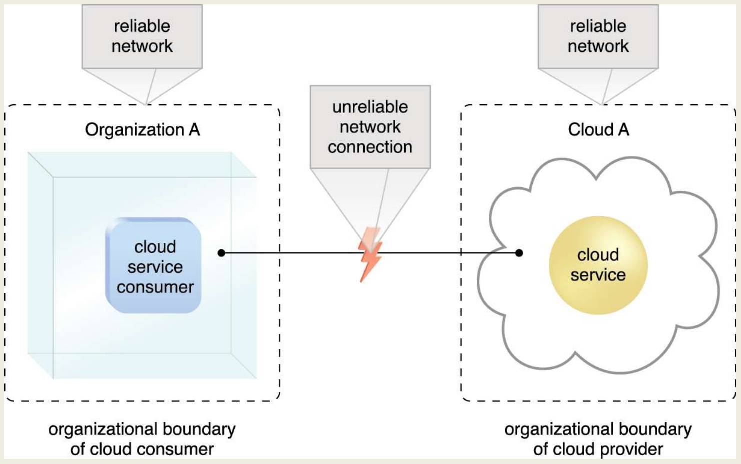
- Increased scalability
  - Example demand over a 24-hour day →
- Increased availability
- Increased reliability



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

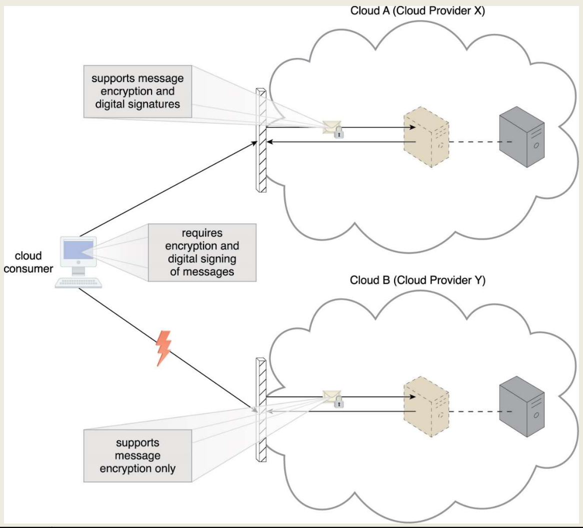
NETWORK LATENCY COSTS


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