

## TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

**Cloud Computing:**  
*How did we get here? – cont'd*

**Introduction to Cloud Computing**

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



## FEEDBACK FROM 10/3

- What is High Availability?
- Gustafson's Law

October 8, 2018

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

L4.2

## SPEED-UP

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

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

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

October 8, 2018

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

L4.3

## 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 8, 2018

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

L4.4

## 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 8, 2018

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

L4.5

## 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 except for 25% that 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 8, 2018

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

L4.6

## 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 8, 2018

TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma

L4.7

## OBJECTIVES

### Cloud Computing: How did we get here?

- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity

### Term Project Description

October 8, 2018

TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma

L4.8

## DISTRIBUTED SYSTEMS

- Collection of autonomous computers, connected through a network with distribution software called "middleware" that enables coordination of activities and sharing of resources

### Key characteristics:

- Users perceive system as a single, integrated computing facility.
- Compute nodes are autonomous
- Scheduling, resource management, and security implemented by every node
- Multiple points of control and failure
- Nodes may not be accessible at all times
- System can be scaled by adding additional nodes
- Availability at low levels of HW/software/network reliability

October 8, 2018

TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma

L4.9

## DISTRIBUTED SYSTEMS - 2

### Key non-functional attributes

- Known as "ilities" in software engineering

- Availability – 24/7 access?
- Reliability – Fault tolerance
- Accessibility – reachable?
- Usability – user friendly
- Understandability – can understand
- Scalability – responds to variable demand
- Extensibility – can be easily modified, extended
- Maintainability – can be easily fixed
- Consistency – data is replicated correctly in timely manner

October 8, 2018

TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma

L4.10

## 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 8, 2018

TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma

L4.11

## 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 8, 2018

TCCS562: Software Engineering for Cloud Computing [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma

L4.12

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

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

October 8, 2018

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

L4.13

TCSS 562 TERM PROJECT


- Term project introduction – via Canvas

October 8, 2018

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

L4.14

QUESTIONS



October 8, 2018

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

L4.15