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

Cloud Computing – *How did we get here? - II*

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

1

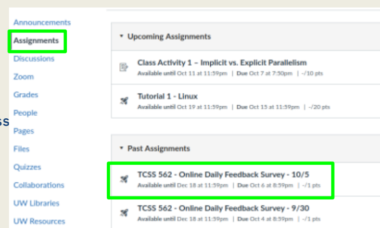
OBJECTIVES – 10/7

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

2

ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Take After Each Class
- **1-point Extra Credit** for completing online
- **2-points Extra Credit** for completing in-person in class
- 36 points possible
- 2.5% added to final course grade for (36/36)

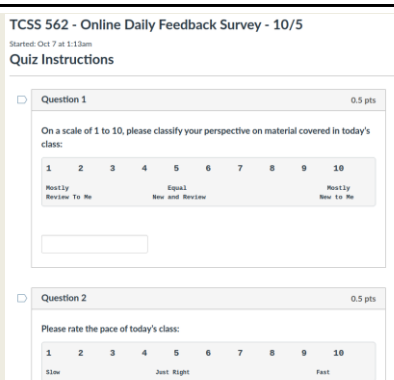


3

WARNING

- DO NOT SUBMIT BOTH A PAPER AND AN ONLINE SURVEY OR YOU WILL LOOSE POINTS
- CANVAS WILL AUTOMATICALLY REPLACE THE PAPER SURVEY SCORE (2 PTS) WITH THE ONLINE SURVEY (1 PT)
- __* COMPLETE ONLY ONE SURVEY FOR EACH CLASS SESSION *__
- WE WILL NOT BE ABLE TO DUPLICATE CHECK SURVEYS FOR EACH CLASS SESSION AND MAKE CORRECTIONS

4



TCSS 562 - Online Daily Feedback Survey - 10/5
Started: Oct 7 at 1:13am
Quiz Instructions

Question 1 — 0.5 pts

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

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

Question 2 — 0.5 pts

Please rate the pace of today's class:

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

5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (46 respondents, 38 in-person, 8 online):
- 1-mostly review, 5-equal new/review, 10-mostly new
- __Average – 7.24__ *(↑ - previous 7.20)*
- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- __**Average – 5.20**__ (__**↑ - previous 5.16**__)

6

FEEDBACK FROM 10/2

- ***Do we need to upload video of doing the assignment (tutorial 1 ?) for quiz or can take quiz directly?***
  - Assuming question is about tutorial 1
  - Quiz can be taken directly, but 50% credit for uploading script of Linux session where you've tested the Linux commands to answer the quiz questions
- ***Are metal "vms" on Amazon a dedicated machine since no hypervisor?***
  - Yes – renting a "metal" instance is one way to reserve an entire physical server, and guarantee there are no other tenants (users)

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.7

7

QUESTIONS - 2

- ***What changed between amd "m6a"-192vCPU/host and "m4a"-192vCPU/host? If # of CPUs did not change what did?***
  - There is no "m4a" ec2 instance family
  - Do you mean what is the difference between m6a.48xlarge and m6a.metal ?
- ***Why would embarrassingly parallel be an issue?***
  - Embarrassingly parallel also is called "pleasingly" parallel
  - This simply means that making the program code parallel was relatively easy – such that the programmer is embarrassed regarding how "easy" their job is
- ***Does data level parallelism require multiple processors ?***
  - On a Von Neumann (control-flow) architecture (e.g. x86/ARM), yes

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.8

8

QUESTIONS - 3

- ***How are the differences between ARM and Intel (x86) processors significant for cloud computing?***
- Papers:
- Characterizing X86 and ARM Serverless Performance Variation: A Natural Language Processing Case Study
- X86 vs. ARM64: An Investigation of Factors Influencing Serverless Performance
- Predicting ARM64 Serverless Function Runtime: Leveraging function profiling for generalized performance models

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.9

9

FEEDBACK - 4

- ***Thread level parallelism is not clear***
- Thread level parallelism refers to when parallelism occurs as a result of multiple threads performing operations in parallel typically on a multi-core computer
- As DevOps engineers, we often are responsible for deploying our applications in the cloud. Therefore, we need to understand the average number and peak number of threads our application requires.
- In class, I demonstrated how this can be observed in Linux using "top" and a multi-threaded prime number generation program

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.10

10

DEMOGRAPHICS SURVEY

- Please complete the ONLINE demographics survey:

We have received 41 responses so far.
We are waiting on ~12 responses.

- **https://forms.gle/QNUW2hUV7fR7BDmv7**
- Linked from course webpage in Canvas:
- http://faculty.washington.edu/wlloyd/courses/tcss562/announcements.html

October 7, 2025 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.11

11

AWS CLOUD CREDITS SURVEY

- Please complete the AWS Cloud Credits survey:

  Please only complete survey after setting up AWS account or if requiring an IAM user (no-credit card option)

- **https://forms.gle/Y4iWvBRFVLRPnPX37**
- Linked from course webpage in Canvas:
- http://faculty.washington.edu/wlloyd/courses/tcss562/announcements.html

October 7, 2025 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.12

12

OBJECTIVES – 10/7

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

13

OBJECTIVES – 10/7

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

14

OBJECTIVES – 10/7

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

15

OBJECTIVES – 10/7

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

16

CLOUD COMPUTING: HOW DID WE GET HERE? - 5

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

17

CLOUD COMPUTING: HOW DID WE GET HERE? - 6

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

18

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?**

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.19

19

OBJECTIVES – 10/7

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

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.20

20

CLASS ACTIVITY 1

- Form teams of ~3 - in class or with Zoom breakout rooms
- Each team will complete a MSWORD DOCX worksheet
- Be sure to add team member names at top of document as they appear in Canvas
- Activity can be completed in class or after class
- The activity can also be completed individually
- When completed, **one team member** should submit a PDF of the document to Canvas
- All team members will receive a grade based on the uploaded PDF file
- To get started:
  - Follow the link: (link also available in Canvas) https://faculty.washington.edu/wlloyd/courses/tcss562/assignments/tcss462_562_f2025_tps1.docx

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.21

21

CLASS ACTIVITY 1

- Solutions to be discussed..

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.22

22

IMPLICIT PARALLELISM

- Implicit types:
- Why are these methods available automatically without special developer effort?
- Advantages:
- Disadvantages:

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.23

23

EXPLICIT PARALLELISM

- Explicit types:
- Advantages:
- Disadvantages:

October 7, 2025 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2025] School of Engineering and Technology, University of Washington - Tacoma | L3.24

24

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)

25

PARALLELISM QUESTIONS - 2

- 9. An application developer measures the average and peak thread level parallelism (TLP) of an application prior to deployment on the AWS EC2. The developer measures an average TLP of 2.3, and a peak TLP of 7.3. The application is to be deployed using a compute-optimized (c-series) ec2 instance. Using resources online, such as the websites below, , propose a good virtual machine (ec2 type) that satisfies average TLP, and a second for satisfying peak TLP that does not under-provision or over-provision vCPUs for the TLP goal, in order to control costs.
- https://docs.aws.amazon.com/ec2/latest/instancetypes/co.html
- https://instances.vantage.sh/

26

PARALLELISM QUESTIONS - 3

- What is a good ec2 c-series instance for average TLP ?
- Why is this instance good/sufficient for satisfying average TLP?
- What is a good ec2 c-series instance for peak TLP ?
- Why is this instance good/sufficient for satisfying peak TLP ?

27

OBJECTIVES – 10/7

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

28

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

29

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

30

(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

31

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)

32

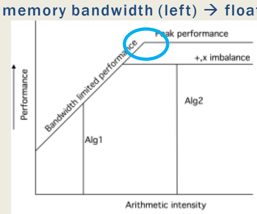
ARITHMETIC INTENSITY

- **Arithmetic intensity**: Ratio of work (W) to memory traffic r/w (Q) $I = \frac{W}{Q}$
  Example: # of floating point ops per byte of data read
- Characterizes application scalability with SIMD support
  - *SIMD can perform many fast matrix operations in parallel*
- ***High arithmetic Intensity:***
  Programs with dense matrix operations scale up nicely (many calcs vs memory RW, supports lots of parallelism)
- ***Low arithmetic intensity:***
  Programs with sparse matrix operations do not scale well with problem size
  (memory RW becomes bottleneck, not enough ops!)

33

ROOFLINE MODEL

- When program reaches a given arithmetic intensity performance of code running on CPU hits a "roof"
- CPU performance bottleneck changes from: memory bandwidth (left) → floating point performance (right)



Key take-aways:
When a program's has **low** Arithmetic Intensity, memory bandwidth limits performance..

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

34

OBJECTIVES – 10/7

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

35

GRAPHICAL PROCESSING UNITS (GPUs)

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

36

OBJECTIVES – 10/7

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

PARALLEL COMPUTING

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

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

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

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

AMDAHL'S LAW

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

1. Divide job into two parts
2. Part A that will still be sequential
3. Part B that will be sped-up with parallel computing

- Portion of computation which cannot be parallelized will determine (i.e. limit) the overall speedup
- Amdahl's law assumes jobs are of a fixed size
- Also, Amdahl's assumes no overhead for distributing the work, and a perfectly even work distribution

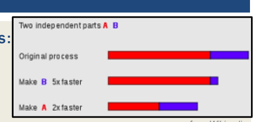
AMDAHL'S LAW

Speed-up formula →

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

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

AMDAHL'S LAW EXAMPLE



from Wikipedia

- Program with two independent parts:
  - Part A is 75% of the execution time
  - Part B is 25% of the execution time
- Part B is made 5 times faster with parallel computing
- Estimate the percent improvement of task execution
- Original Part A is 3 seconds, Part B is 1 second
- N=5 (speedup of part B)
- f=.25 (only 25% of the whole job (A+B) will be sped-up)
- S=1 / ((1-f) + f/S)
- S=1 / ((.75) + .25/5)
- S=1.25 (speed up is 1.25x faster)
- % improvement = 100 * (1 – 1/1.25) = **20%**

GUSTAFSON'S LAW

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

N: Number of processors

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

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

43

GUSTAFSON'S LAW

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

N: Number of processors

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

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

44

GUSTAFSON'S LAW

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

N: Number of processors

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

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

45

GUSTAFSON'S EXAMPLE

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

46

GUSTAFSON'S EXAMPLE

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

For 2 CPUs, speed up is 1.25x

For 16 CPUs, speed up is 4.75x

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

47

MOORE'S LAW

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

What kind of processor are modern Intel CPUs ?

- ***Symmetric core processor*** –multi-core CPU, all cores have the same computational resources and speed
- ***Asymmetric core processor*** – on a multi-core CPU, some cores have more resources and speed
- ***Dynamic core processor*** – processing resources and speed can be dynamically configured among cores
- ***Observation: asymmetric processors offer a higher speedup***

48

OBJECTIVES – 10/7

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

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

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

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

OBJECTIVES – 10/7

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

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!

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

55

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

- **Bit-level parallelism**
- **Instruction-level parallelism** (CPU pipelining)
- **Flynn's taxonomy**: computer system architecture classification
  - **SISD** – Single Instruction, Single Data (modern core of a CPU)
  - **SIMD** – Single Instruction, Multiple Data (Data parallelism)
  - **MIMD** – Multiple Instruction, Multiple Data
  - MISD is RARE; application for fault tolerance…
- **Arithmetic intensity**: ratio of calculations vs memory RW
- **Roofline model:**
  Memory bottleneck with low arithmetic intensity
- **GPUs**: ideal for programs with high arithmetic intensity
  - SIMD and Vector processing supported by many large registers

56

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

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

57

QUESTIONS

58