

**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 – 10/3**

**Questions from 10/1**

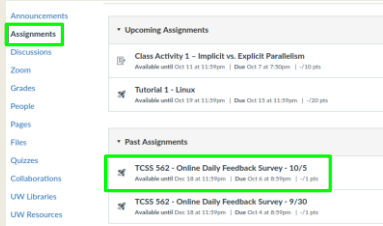
- Tutorial 0, Tutorial 1, Tutorial 2
- Cloud Computing – How did we get here? (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law,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

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.2

2

**ONLINE DAILY FEEDBACK SURVEY**

- Daily Feedback Quiz in Canvas – Take After Each Class
- Extra Credit for completing
- Tuesday class surveys close 11:59pm WED
- Thursday class surveys close 11:59pm MON



October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.3

3

TCSS 562 - Online Daily Feedback Survey - 10/5  
 Started: Oct 7 at 1:18pm  
 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

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.4

4

**MATERIAL / PACE**

- Please classify your perspective on material covered in today's class (40 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- Average – 7.03** (↓ - previous 6.16)
- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- Average – 5.48** (↑ - previous 5.55)
- Response rates:**
- TCSS 462: 27 / 42
- TCSS 562: 13 / 18

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.5

5

**FEEDBACK FROM 10/3**

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

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.6

6

## HYPER-THREADING - 2

- How do I use hyper-threading?
- Hyper-threading is automatic
- Modern CPUs expose each physical CPU core as two CPU cores
- `cat /proc/cpuinfo` command lists individual cores
- Operating system schedules processes & threads to run on a hyper-thread
- On CPUs with hyper-threading, each CPU core has two hyper-threads
- To the operating system they are seen as full-featured independent CPU cores

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.7

7

## CAT /PROC/CPUINFO || LSCPU

```

root@kali:~/# 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    : 0x0
cpu MHz      : 840.021
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_  tm pbe syscall nx pdpe1gb rdtscp ln constant_tsc art arch_perfmon pebs bts rep_good nopl nt
opology monitor tsc_aperflowperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tsc_ssse3 sdbg fma cx16 xtpr pdcm ac
ld sse4_1 sse4_2 xzapic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb
l_1invpcid_single intel_pt sstde tlb_lbb ztbnw kalsoe tpr_shadow vmi flexpriority epb vpid fsgsbase tsc_adjust
bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdseed adx smap clflushopt xsavesopt xsavec xgetbv1 dtherm ida arat
pin pts hwp hwp_notify hwpact_window hwp_opp_mitigation flush_l3d
bugs        : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapsz taa itlb_multihit srbds
bogomips    : 5184.48
clflush size : 64
cache_alignm : 64
address size : 39 bits physical, 48 bits virtual
power management:
    
```

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

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.7

8

## Hyper-Threading (HT) Technology

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

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.10

9

## HYPER-THREADING - 3

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

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.10

10

## FEEDBACK - 2

- It is still not clear to me why a core with two hyperthreads is faster than a core without hyperthreading that is operating at 100%?
- The hyperthreaded core is only faster if running a job that uses multiple threads at the same time (in parallel).
- If the job is sequential, there is likely no difference.
- But for your laptop, the more hyperthreads you have, the more web browser code you can execute in parallel across each tab of the browser
  - Web browsers are multi-process (Chrome) or multi-threaded (Firefox)

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.11

11

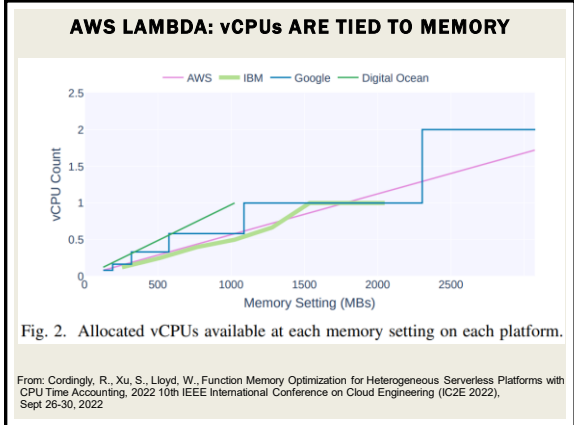
## EFFECTS OF HYPERTHREADING ON AWS LAMBDA

**Figure 1: AWS Lambda Performance Speedup for Sysbench Prime Number Generation vs. Function Memory**

From: Cordingley, R., Heydari, N., Yu, H., Hoang, V., Sadeghi, Z., Lloyd, W., Enhancing Observability of Serverless Computing with the Serverless Application Analytics Framework, Tutorial Paper, 2021 12th ACM/SPEC International Conference on Performance Engineering (ICPE '21), Apr 19-23, 2021.

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.12

12



13

### FEEDBACK - 3

- ***If I use a computer with 8 cores (client) to rent a virtual machine with 128 cores through a cloud provider, the computer with less cores won't decrease the performance of the virtual machine with more cores because they are separate?***
- CORRECT, the performance will not decrease.
- The 8-core (laptop/desktop) is just used to access the remote computer via ssh/graphical desktop
- The laptop/desktop acts as a client computer used to access the powerful remote server
- Any applications / jobs / workloads are run on the remote server, but are launched by the client
  - Through a terminal session (ssh), or remote graphical desktop
  - Or by calling a web service hosted on the powerful server
- You may experience **network latency** between the client and server for large data transfers

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.14

14

### FEEDBACK - 4

- ***Thread level parallelism is not clear***
- Thread level parallelism refers to when parallelism occurs as a result of multiple threads performance 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 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.15

15

### DEMOGRAPHICS SURVEY

- Please complete the ONLINE demographics survey:

We have received 37 responses so far. We are waiting on ~23 responses.

- <https://forms.gle/6ER7PzfP521vdxYW9>
- Linked from course webpage in Canvas:
- <http://faculty.washington.edu/wlloyd/courses/tcss562/announcements.html>

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.16

16

### AWS CLOUD CREDITS SURVEY

- Please complete the AWS Cloud Credits survey:

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

- <https://forms.gle/fmKkLZbxZECbAay16>
- Linked from course webpage in Canvas:
- <http://faculty.washington.edu/wlloyd/courses/tcss562/announcements.html>

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.17

17

### OBJECTIVES - 10/3

- Questions from 10/1
- **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 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.18

18

### OBJECTIVES - 10/3

- Questions from 10/1
- 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 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.19

19

### OBJECTIVES - 10/3

- Questions from 10/1
- 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 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.20

20

### OBJECTIVES - 10/3

- Questions from 10/1
- 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 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.21

21

### CLOUD COMPUTING: HOW DID WE GET HERE? - 5

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

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.22

22

### CLOUD COMPUTING: HOW DID WE GET HERE? - 6


- Compute clouds are large-scale distributed systems
- Infrastructure-as-a-Service (IaaS) Cloud
  - Provide VMs on demand to users
  - **ec2Instances.info** (AWS EC2)
- Clouds can consist of
  - **Homogeneous** hardware (servers, etc.)
  - **Heterogeneous** hardware (servers, etc.)
- Which is preferable?

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.23

23

### HARDWARE HETEROGENEITY

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



October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.24

24

### OBJECTIVES - 10/3

- Questions from 10/1
- 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 3, 2024 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024]<br>School of Engineering and Technology, University of Washington - Tacoma | L3.25 |
|-----------------|--|-------|

25

### CLASS ACTIVITY 1

- Form groups of ~3 - in class or with Zoom breakout rooms
- Each group will complete a MSWORD DOCX worksheet
- Be sure to add names at top of document as they appear in Canvas
- Activity can be completed in class or after class
- The activity can also be completed individually
- When completed, **one person** should submit a PDF of the document to Canvas
- Instructor will score all group members based on the uploaded PDF file
- To get started:
  - Follow the link: (link also available in Canvas)  
[https://faculty.washington.edu/wlloyd/courses/tcss562/assignments/tcss462\\_562\\_f2024\\_tps1.docx](https://faculty.washington.edu/wlloyd/courses/tcss562/assignments/tcss462_562_f2024_tps1.docx)

|                 |  |       |
|-----------------|--|-------|
| October 3, 2024 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024]<br>School of Engineering and Technology, University of Washington - Tacoma | L3.26 |
|-----------------|--|-------|

26

### CLASS ACTIVITY 1

- Solutions to be discussed..

|                 |  |       |
|-----------------|--|-------|
| October 3, 2024 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024]<br>School of Engineering and Technology, University of Washington - Tacoma | L3.27 |
|-----------------|--|-------|

27

### IMPLICIT PARALLELISM

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

|                 |  |       |
|-----------------|--|-------|
| October 3, 2024 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024]<br>School of Engineering and Technology, University of Washington - Tacoma | L3.28 |
|-----------------|--|-------|

28

### EXPLICIT PARALLELISM

- Explicit types:
- Advantages:
- Disadvantages:

|                 |  |       |
|-----------------|--|-------|
| October 3, 2024 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024]<br>School of Engineering and Technology, University of Washington - Tacoma | L3.29 |
|-----------------|--|-------|

29

### PARALLELISM QUESTIONS

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

|                 |  |       |
|-----------------|--|-------|
| October 3, 2024 | TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024]<br>School of Engineering and Technology, University of Washington - Tacoma | L3.30 |
|-----------------|--|-------|

30

### 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.
- <https://docs.aws.amazon.com/ec2/latest/instancetypes/co.html>
- <https://instances.vantage.sh/>

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.31

31

### 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 ?

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.32

32

### OBJECTIVES – 10/3

- Questions from 10/1
- 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 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.33

33

### 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

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.34

34

### FLYNN'S TAXONOMY

- SISD (Single Instruction Single Data)**  
Scalar architecture with one processor/core.
  - Individual cores of modern multicore processors are "SISD"
- SIMD (Single Instruction, Multiple Data)**  
Supports vector processing
  - When SIMD instructions are issued, operations on individual vector components are carried out concurrently
  - Two 64-element vectors can be added in parallel
  - Vector processing instructions added to modern CPUs
  - Example: Intel MMX (multimedia) instructions

October 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.35

35

### (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 3, 2024 TCSS462/562: (Software Engineering for) Cloud Computing (Fall 2024) School of Engineering and Technology, University of Washington - Tacoma L3.36

36

### FLYNN'S TAXONOMY - 2

- **MIMD (Multiple Instructions, Multiple Data)** - system with several processors and/or cores that function asynchronously and independently
- At any time, different processors/cores may execute different instructions on different data
- Multi-core CPUs are MIMD
- Processors share memory via interconnection networks
  - Hypercube, 2D torus, 3D torus, omega network, other topologies
- MIMD systems have different methods of sharing memory
  - Uniform Memory Access (UMA)
  - Cache Only Memory Access (COMA)
  - Non-Uniform Memory Access (NUMA)

October 3, 2024    TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma    L3.37

37

### ARITHMETIC INTENSITY

$I = \frac{W}{Q}$

- **Arithmetic Intensity:** Ratio of work (W) to memory traffic r/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!)

October 3, 2024    TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma    L3.38

38

### 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??

October 3, 2024    TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma    L3.39

39

### OBJECTIVES - 10/3

- Questions from 10/1
- 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 3, 2024    TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma    L3.40

40

### 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

October 3, 2024    TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma    L3.41

41

### OBJECTIVES - 10/3

- Questions from 10/1
- 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 3, 2024    TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma    L3.42

42

### PARALLEL COMPUTING

- Parallel hardware and software systems allow:
  - Solving problems needing resources not available on a single system.
  - Reduced time required to obtain solution
- The *speed-up* (S) measures effectiveness of parallelization:
 
$$S(N) = T(1) / T(N)$$

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

October 3, 2023    TCSS462/562: Software Engineering for Cloud Computing [Fall 2024]  
 School of Engineering and Technology, University of Washington - Tacoma    L3.43

43

### SPEED-UP EXAMPLE

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

October 3, 2023    TCSS462/562: Software Engineering for Cloud Computing [Fall 2024]  
 School of Engineering and Technology, University of Washington - Tacoma    L3.44

44

### AMDAHL'S LAW

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

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

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

October 3, 2023    TCSS462/562: Software Engineering for Cloud Computing [Fall 2024]  
 School of Engineering and Technology, University of Washington - Tacoma    L3.45

45

### AMDAHL'S LAW

Speed-up formula → 
$$S = \frac{1}{(1-f) + \frac{f}{N}}$$


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

October 3, 2023    TCSS462/562: Software Engineering for Cloud Computing [Fall 2024]  
 School of Engineering and Technology, University of Washington - Tacoma    L3.46

46

### AMDAHL'S LAW EXAMPLE

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



October 3, 2023    TCSS462/562: Software Engineering for Cloud Computing [Fall 2024]  
 School of Engineering and Technology, University of Washington - Tacoma    L3.47

47

### 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)
- Can be used to estimate runtime of parallel portion of program

October 3, 2023    TCSS462/562: Software Engineering for Cloud Computing [Fall 2024]  
 School of Engineering and Technology, University of Washington - Tacoma    L3.48

48



### GUSTAFSON'S LAW

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

$$S(N) = N + (1 - N) \alpha$$
- N: Number of processors
- $\alpha$ : 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  $\alpha = \sigma / (\pi + \sigma)$
- Where  $\sigma$  = sequential time,  $\pi$  = parallel time
- Our Amdahl's example:  $\sigma = 3s$ ,  $\pi = 1s$ ,  $\alpha = .75$

October 3, 2023 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.49

49

### GUSTAFSON'S LAW

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

$$S(N) = N + (1 - N) \alpha$$
- N: Number of processors
- $\alpha$ : 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.  $\alpha = .75$   
**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?**

October 3, 2023 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.50

50

### GUSTAFSON'S EXAMPLE

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

October 3, 2023 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.51

51

### GUSTAFSON'S EXAMPLE

- QUESTION:**  
 What is the maximum theoretical speed-up on a **2-core CPU**?  
 $S(N) = N + (1 - N) \alpha$   
 $N = 2$ ,  $\alpha = .75$   
 $S(N) = 2 + (1 - 2) .75$   
 $S(N) = ?$   
 For 2 CPUs, speed up is 1.25x
- What is the maximum theoretical speed-up on a **16-core CPU**?  
 $S(N) = N + (1 - N) \alpha$   
 $N = 16$ ,  $\alpha = .75$   
 $S(N) = 16 + (1 - 16) .75$   
 $S(N) = ?$   
 For 16 CPUs, speed up is 4.75x

October 3, 2023 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.52

52

### MOORE'S LAW

- Transistors on a chip doubles approximately every 1.5 years
- CPUs now have 100s of cores
- Power dissipation is a major concern for heat removal  
 What kind of processor are modern Intel CPUs?  
 Transistors are packed more densely on modern 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 3, 2023 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.53

53

### OBJECTIVES - 10/3

- Questions from 10/1
- 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 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.54

54

## 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 3, 2024 TCCS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.55

55

## 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 3, 2024 TCCS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.56

56

## 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 3, 2024 TCCS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.57

57

## OBJECTIVES – 10/3

- Questions from 10/1
- 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 3, 2024 TCCS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.58

58

## 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 3, 2024 TCCS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.59

59

## 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 3, 2024 TCCS462/562: Software Engineering for Cloud Computing [Fall 2024] School of Engineering and Technology, University of Washington - Tacoma L3.60

60

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

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

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma L3.61

61


### 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:**  
 $S(N) = N - \alpha(N-1)$
- Moore's Law
- Symmetric core, Asymmetric core, Dynamic core CPU
- Distributed Systems Non-function quality attributes
- Distributed Systems - Types of Transparency
- Types of modularity- Soft, Enforced

October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma L3.62

62

# QUESTIONS



October 3, 2024 TCSS462/562: Software Engineering for Cloud Computing [Fall 2024]  
School of Engineering and Technology, University of Washington - Tacoma L3.63

63