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

Cloud Computing – *How did we get here?*

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

1

OBJECTIVES – 10/6

- Questions from 10/4
- Cloud Computing – How did we get here? *(Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)*
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.2

2

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (43 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.16** *(6.15, same lecture Fall 2021)*

- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.35** *(5.19, same lecture Fall 2021)*

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.3

3

FEEDBACK FROM 10/4

- ***Is team project research only for 562 students?***
  I didn't clearly understand what the term project would be.
  - Do you mean, doing a research topic for the term project?
- ***Can you assign a group for us?***
- ***Not directly related, but I am a little concerned about level of knowledge expected for class and if beginners will be able to cope***
  - Bachelors in Computer Science & Systems teaches Java as the primary language at UWT. Most 400-level CSS students have had multiple courses in Java (TCSS 142, 143, 305, 342?, 343?)
  - The amount of programming will depend on the Term Project topic
  - Serverless functions development is not terribly programming intensive

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.4

4

FEEDBACK - 2

- ***Regarding term project, I am not sure about the details of personal choice to do research paper, such as what aspects of cloud computing to search, word limit, and number of resources requirements.***
  - For a gap analysis (literature survey) paper, the idea is to choose a specific cloud computing research problem and go to the literature to assess the current 'state of the art' solutions that are available.
  - The idea is to synthesize what existing research has been done on the problem by summarizing previous efforts, critiquing positives and negatives (i.e. good, bad, and ugly) and then identify future research directions and potential related to the topic
  - Gap analysis paper can help identify an honors/MS/PhD thesis topic
  - Gap analysis will typically review work from 5 full papers, and will cite and mention to a lesser extent 5-10+ related papers

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.5

5

DEMOGRAPHICS SURVEY

- Please complete the ONLINE demographics survey:
- https://forms.gle/XAhBRUR8wsm7CqSs5
- Linked from course webpage in Canvas:
- http://faculty.washington.edu/wlloyd/courses/tcss562/announcements.html
- Completed as of Thursday AM: 53 of 61

October 4, 2022	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L2.6

6

AWS CLOUD CREDITS SURVEY

- Please complete the ONLINE demographics survey:
- https://forms.gle/yz8yrqB7yGD5iHSh9
- Linked from course webpage in Canvas:
- http://faculty.washington.edu/wlloyd/courses/tcss562/announcements.html
- Completed as of Thursday AM: 52 of 61

October 4, 2022	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L2.7

7

TUTORIAL 1

- Introduction to Linux & the Command Line
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_1.pdf
- Tutorial Sections:
  1. The Command Line
  2. Basic Navigation
  3. More About Files
  4. Manual Pages
  5. File Manipulation
  6. VI – Text Editor
  7. Wildcards
  8. Permissions
  9. Filters
  10. Grep and regular expressions
  11. Piping and Redirection
  12. Process Management

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.8

8

TUTORIAL 2

- **Introduction to Bash Scripting**
- https://faculty.washington.edu/wlloyd/courses/tcss562/tutorials/TCSS462_562_f2022_tutorial_2.pdf
- Review tutorial sections:
- Create a BASH webservice client
  1. What is a BASH script?
  2. Variables
  3. Input
  4. Arithmetic
  5. If Statements
  6. Loops
  7. Functions
  8. User Interface
- Call service to obtain IP address & lat/long of computer
- Call service to obtain weather forecast for lat/long

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.9

9

OBJECTIVES – 10/6

- Questions from 10/4
- **Cloud Computing – How did we get here?**
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.10

10

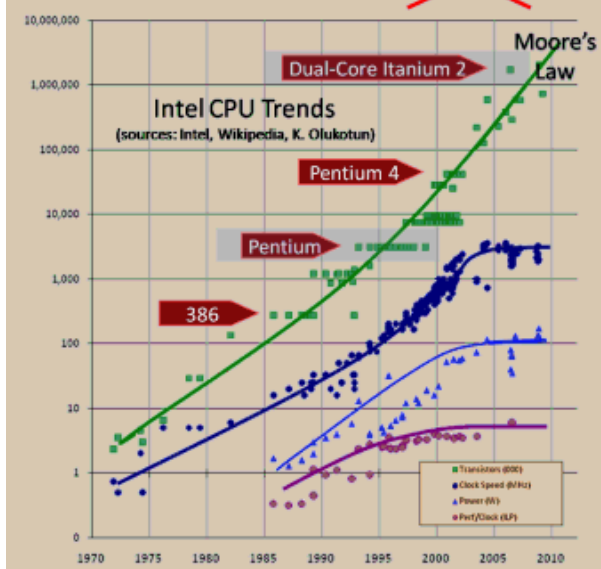
CLOUD COMPUTING: HOW DID WE GET HERE?

- General interest in parallel computing
  - Moore's Law - # of transistors doubles every 18 months
  - Post 2004: heat dissipation challenges: can no longer easily increase cloud speed
  - Overclocking to 7GHz takes more than just liquid nitrogen:
    - https://tinyurl.com/y93s2yz2
- Solutions:
  - Vary CPU clock speed
  - Add CPU cores
  - **Multi-core technology**

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.11

11

Each Year We Get ~~Faster~~ More Processors



Historically:
Boost single-stream performance via more complex chips.

Now:
Deliver more cores per chip (+ GPU, NIC, SoC).

The free lunch is over for today's sequential apps and many concurrent apps. We need killer apps with lots of latent parallelism.

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.12

12

AMD'S 64-CORE 7NM CPUS

- Epyc Rome CPUs
- Announced August 2019
- EPYC 7H12 requires liquid cooling

AMD EPYC 7002 Processors (2P)

	Cores Threads	Frequency (GHz)		L3*	TDP	Price
		Base	Max			
EPYC 7H12	**64 / 128**	**2.60**	**3.30**	**256 MB**	**280 W**	**?**
EPYC 7742	64 / 128	2.25	3.40	256 MB	225 W	$6950
EPYC 7702	64 / 128	2.00	3.35	256 MB	200 W	$6450
EPYC 7642	48 / 96	2.30	3.20	256 MB	225 W	$4775
EPYC 7552	48 / 96	2.20	3.30	192 MB	200 W	$4025

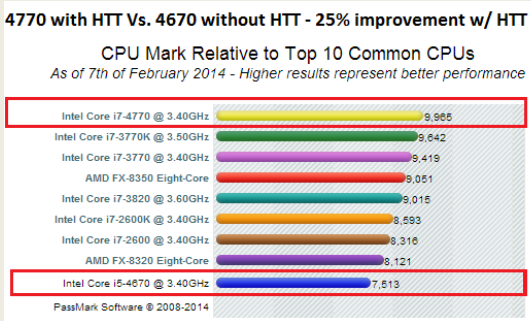
HOST SERVER VCPUS – AMAZON EC2 INFRASTRUCTURE-AS-A-SERVICE CLOUD

- Cloud server virtual CPUs/host
- Growth since 2006 - Amazon Compute Cloud (EC2)

- 1st generation Intel: m1 – 8 vCPUs / host (Aug 2006)
- 2nd generation Intel: m2 – 16 vCPUs / host (Oct 2009)
- 3rd generation Intel: m3 - 32 vCPUs / host (Oct 2012)
- 4th generation Intel: m4 – 48 vCPUs / host (June 2015)
- 5th generation Intel: m5 – 96 vCPUs / host (Nov 2017)
- 6th generation Intel: m6i – 128 vCPUs / host (Aug 2021)
- 6th generation AMD: m6a – 192 vCPUs / host (Nov 2021)

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
- Example: → hyperthreads add +32.9%



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

CAT /PROC/CPUINFO || LSCPU

```
wlloyd@dione:~/Dropbox/courses/tcss562$ cat /proc/cpuinfo | grep -C 20 ht
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 94
model name      : Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
stepping        : 3
microcode       : 0xdc
cpu MHz         : 840.023
cache size      : 6144 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx
fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xt
opology nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pc
id sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb
 invpcid_single intel_pt ssbd ibrs ibpb stibp kaiser tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust
bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdseed adx smap clflushopt xsaveopt xsavec xgetbv1 dtherm ida arat
 pln pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa itlb_multihit srbds
bogomips        : 5184.46
clflush size    : 64
cache_alignment : 64
address sizes   : 39 bits physical, 48 bits virtual
power management:
```

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

17

Hyper-Threading (HT) Technology

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

Figure 2: Processors without Hyper-Threading Tech

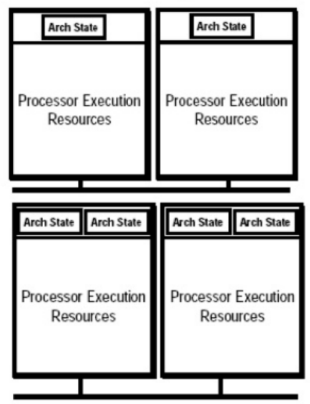
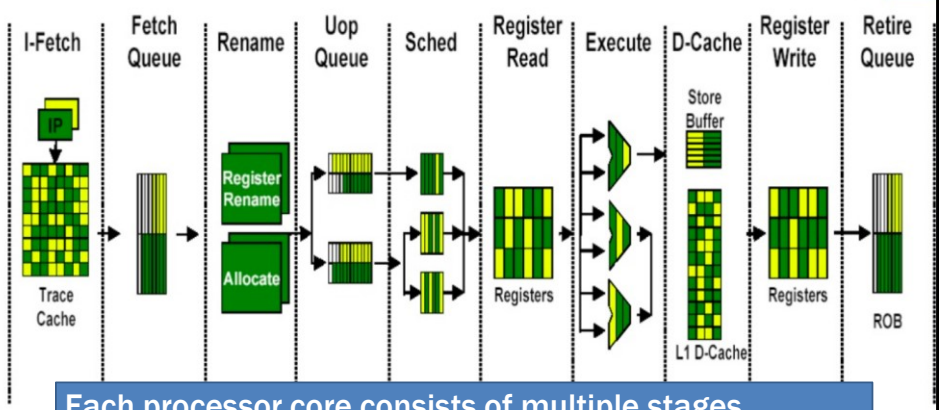


Figure 3: Processors with Hyper-Threading Technology

18

Execution Pipeline



Each processor core consists of multiple stages

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

19

19

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.20

20

CLOUD COMPUTING: HOW DID WE GET HERE? - 2

- To make computing faster, we must go "parallel"
- Difficult to expose parallelism in scientific applications
- Not every problem solution has a parallel algorithm
  - Chicken and egg problem…
- Many commercial efforts promoting pure parallel programming efforts have failed
- Enterprise computing world has been *skeptical* and less involved in parallel programming

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.21

21

CLOUD COMPUTING: HOW DID WE GET HERE? - 3

- **Cloud computing** provides access to "infinite" scalable compute infrastructure on demand
- Infrastructure availability is key to exploiting parallelism
- **Cloud applications**
  - Based on **client-server** paradigm
  - **Thin clients** leverage compute hosted on the cloud
  - Applications run many web service instances
  - Employ load balancing

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.22

22

CLOUD COMPUTING: HOW DID WE GET HERE? - 4

- **Big Data** requires massive amounts of compute resources
- MAP – REDUCE
  - Single instruction, multiple data (SIMD)
  - Exploit data level parallelism
- Bioinformatics example

23

SMITH WATERMAN USE CASE

- Applies dynamic programming to find best local alignment of two protein sequences
  - Embarrassingly parallel, each task can run in isolation
  - Use case for GPU acceleration
- **AWS Lambda Serverless Computing Use Case: Goal:** Pair-wise comparison of all unique human protein sequences (20,336)
  - Python client as scheduler
  - C Striped Smith-Waterman (SSW) execution engine

From: Zhao M, Lee WP, Garrison EP, Marth GT: SSW library: an SIMD Smith-Waterman C/C++ library for use in genomic applications. PLoS One 2013, 8:e82138

24

SMITH WATERMAN RUNTIME

- Laptop server and client (2-core, 4-HT): 8.7 hours
- AWS Lambda FaaS, laptop as client: 2.2 minutes
  - Partitions 20,336 sequences into 41 sets
  - Execution cost: ~ 82¢ (~237x speed-up)
- AWS Lambda server, EC2 instance as client: 1.28 minutes
  - Execution cost: ~ 87¢ (~408x speed-up)
- Hardware
  - Laptop client: Intel i5-7200U 2.5 GHz :4 HT, 2 CPU
  - Cloud client: EC2 Virtual Machine - m5.24xlarge: 96 vCPUs
  - Cloud server: Lambda ~1000 x Intel E5-2666v3 2.9GHz CPUs

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.25

25

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.26

26

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.27

27

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.28

28

OBJECTIVES – 10/6

- **Questions from 10/4**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.29

29

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 problems 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

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.30

30

PARALLELISM - 2

- **Coordination of nodes**
- Requires ***message passing*** or ***shared memory***
- Debugging parallel ***message passing*** code is easier than parallel ***shared memory*** code
- ***Message passing***: all of the interactions are clear
  - Coordination via specific programming API (MPI)
- ***Shared memory***: interactions can be implicit – *must read the code!!*
- Processing speed is orders of magnitude faster than communication speed (CPU > memory bus speed)
- Avoiding coordination achieves the best speed-up

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.31

31

TYPES OF PARALLELISM

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

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.32

32

THREAD LEVEL PARALLELISM (TLP)

- Number of threads an application runs at any one time
- Varies throughout program execution
- As a metric:
- **Minimum**: 1 thread
- Can measure **average**, **maximum (peak)**
- **QUESTION: What are the consequences of average (TLP) for scheduling an application to run on a computer with a fixed number of CPU cores and hyperthreads?**
- Let's say there are 4 cores, or 8 hyper-threads…
- ***Key to avoiding waste of computing resources is knowing your application's TLP…***

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.33

33

TLP – PRIMES EXAMPLE

- Multi-threaded prime number generation
- Compute-bound workload
- Can use variable # of threads
- Generates n prime numbers
- **Runtimes: 100,000 primes**
- 1 thread: 59.15 s
- 2 threads: 30.957 s
- 4 threads: 15.539 s
- 8 threads: 12.112 s
- Observe TLP with top

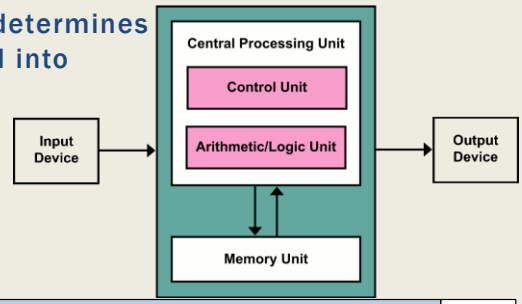
```
time ./primes8 30000 >/dev/null
```

October 6, 2022	TCSS462/562: (Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.34

34

CONTROL-FLOW ARCHITECTURE

- Typical architecture used today – w/ multiple threads
- By John von Neumann (1945)
- Also called the Von Neumann architecture
- Dominant computer system architecture
- Program counter (PC) determines next instruction to load into *instruction register*
- Program execution is sequential



October 6, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L3.35

35

DATA-LEVEL PARALLELISM

- Partition data into big chunks, run separate copies of the program on them with little or no communication
- Problems are considered to be ***embarrassingly parallel***
- Also perfectly parallel or pleasingly parallel…
- Little or no effort needed to separate problem into a number of parallel tasks
- MapReduce programming model is an example

October 6, 2022 | TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma | L3.36

36

DATA FLOW ARCHITECTURE

- ***Alternate architecture*** used by network routers, digital signal processors, special purpose systems
- Operations performed when input (data) becomes available
- Envisioned to provide much higher parallelism
- Multiple problems has prevented wide-scale adoption
  - Efficiently broadcasting data tokens in a massively parallel system
  - Efficiently dispatching instruction tokens in a massively parallel system
  - Building content addressable memory large enough to hold all of the dependencies of a real program

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.37

37

DATA FLOW ARCHITECTURE - 2

- Architecture not as popular as control-flow
- Modern CPUs emulate data flow architecture for dynamic instruction scheduling since the 1990s
  - Out-of-order execution – reduces CPU idle time by not blocking for instructions requiring data by defining execution windows
  - Execution windows: identify instructions that can be run by data dependency
  - Instructions are completed in data dependency order within execution window
    - Execution window size typically 32 to 200 instructions

Utility of data flow architectures has been much less than envisioned

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.38

38

BIT-LEVEL PARALLELISM

- Computations on large words (e.g. 64-bit integer) are performed as a single instruction
- Fewer instructions are required on 64-bit CPUs to process larger operands (A+B) providing dramatic performance improvements
- Processors have evolved: 4-bit, 8-bit, 16-bit, 32-bit, 64-bit

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

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

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.39

39

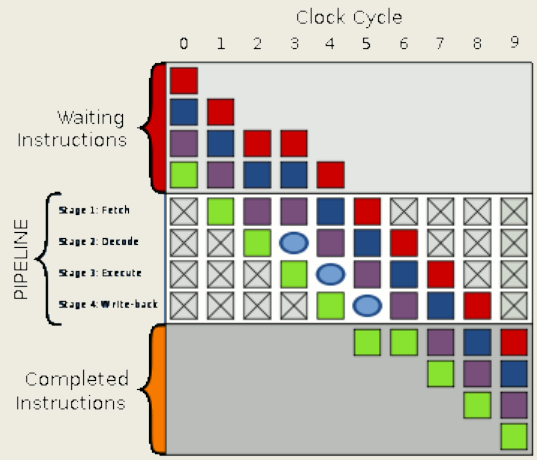
INSTRUCTION-LEVEL PARALLELISM (ILP)

- CPU pipelining architectures enable ILP
- CPUs have multi-stage processing pipelines
- Pipelining: split instructions into sequence of steps that can execute concurrently on different CPU circuitry
- Basic RISC CPU - Each instruction has 5 pipeline stages:
- **IF** – *instruction fetch*
- **ID**- *instruction decode*
- **EX** – *instruction execution*
- **MEM** – *memory access*
- **WB** – *write back*

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.40

40

CPU PIPELINING



41

INSTRUCTION LEVEL PARALLELISM - 2

- RISC CPU:
- After 5 clock cycles, all 5 stages of an instruction are loaded
- Starting with 6th clock cycle, one full instruction completes each cycle
- The CPU performs 5 tasks per clock cycle!
  Fetch, decode, execute, memory read, memory write back
- Pentium 4 (CISC CPU) – processing pipeline w/ 35 stages!

42

OBJECTIVES – 10/6

- **Questions from 10/4**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.43

43

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 Google Doc to Canvas
- Instructor will score all group members based on the uploaded PDF file
- To get started:
  - Log into your UW Google Account (https://drive.google.com) using you UW NET ID
  - Follow the link:
    https://tinyurl.com/tcss462-562-a1

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.44

44

ACTIVITY 1

- Solutions to be discussed..

45

IMPLICIT PARALLELISM

- Applies to:
- Advantages:
- Disadvantages:

46

EXPLICIT PARALLELISM

- Applies to:
- Advantages:
- Disadvantages:

PARALLELISM QUESTIONS

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

PARALLELISM QUESTIONS - 2

- 9. For thread level parallelism (TLP) where a programmer has spent considerable effort to parallelize their code and algorithms, what consequences result when this code is deployed on a virtual machine with too few virtual CPU processing cores?
- What happens when this code is deployed on a virtual machine with too many virtual CPU processing cores?

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.49

49

OBJECTIVES – 10/6

- **Questions from 10/4**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.50

50

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.51

51

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.52

52

(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

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

- **Arithmetic intensity**: Ratio of work (W) to memory traffic r/w (Q)

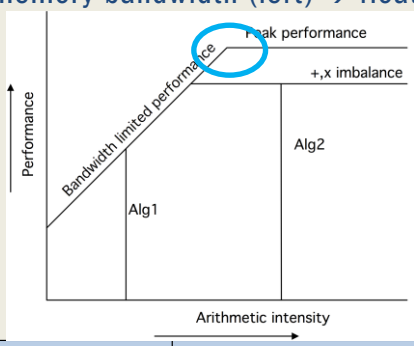
$$I = \frac{W}{Q}$$

  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!)

55

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??*

56

OBJECTIVES – 10/6

- **Questions from 10/4**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.57

57

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.58

58

OBJECTIVES – 10/6

- **Questions from 10/4**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.59

59

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

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.60

60

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.61

61

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

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.62

62

AMDAHL'S LAW

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

- 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))

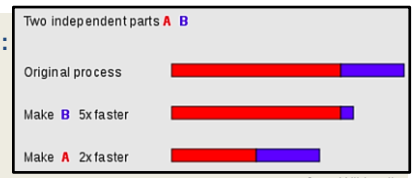
- **Using Amdahl's law, what is the maximum possible speed-up?**

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.63

63

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



from Wikipedia

- 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
- % improvement = 100 * (1 – 1/1.25) = **20%**

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.64

64

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*

65

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*
- Where $\alpha = \sigma / (\pi + \sigma)$
- Where σ= sequential time, π =parallel time
- Our Amdahl's example: σ= 3s, π =1s, α =.75

66

GUSTAFSON'S LAW

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

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

N: Number of processors

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

- Example:
  Consider a program that is embarrassingly parallel, but 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?***

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.67

67

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.68

68

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) = ?

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.69

69

MOORE'S LAW

- Transistors on a chip doubles approximately every 1.5 years
- CPUs now have billions of transistors
- Power dissipation issues at faster clock rates leads to heat removal challenges
  - Transition from: increasing clock rates → to adding CPU cores
- ***Symmetric core processor*** –multi-core CPU, all cores have the same computational resources and speed
- ***Asymmetric core processor*** – on a multi-core CPU, some cores have more resources and speed
- ***Dynamic core processor*** – processing resources and speed can be dynamically configured among cores
- ***Observation: asymmetric processors offer a higher speedup***

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.70

70

OBJECTIVES – 10/6

- **Questions from 10/4**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.71

71

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.72

72

DISTRIBUTED SYSTEMS - 2

- Key non-functional attributes
  - Known as "ilities" in software engineering
- Availability – 24/7 access?
- Reliability - Fault tolerance
- Accessibility – reachable?
- Usability – user friendly
- Understandability – can under
- Scalability – responds to variable demand
- Extensibility – can be easily modified, extended
- Maintainability – can be easily fixed
- Consistency – data is replicated correctly in timely manner

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.73

73

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.74

74

OBJECTIVES – 10/6

- **Questions from 10/4**
- Cloud Computing – How did we get here?
  (Marinescu Ch. 2 - 1st edition, Ch. 4 - 2nd edition)
- Data, thread-level, task-level parallelism & Parallel architectures
- Class Activity 1 – Implicit vs Explicit Parallelism
- SIMD architectures, vector processing, multimedia extensions
- Graphics processing units
- Speed-up, Amdahl's Law, Scaled Speedup
- Properties of distributed systems
- Modularity
- Introduction to Cloud Computing – loosely based on book #1: Cloud Computing Concepts, Technology & Architecture

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.75

75

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.76

76

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.77

77

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 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.78

78

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

October 6, 2022	TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022] School of Engineering and Technology, University of Washington - Tacoma	L3.79

79

QUESTIONS

October 6, 2022

TCSS462/562:(Software Engineering for) Cloud Computing [Fall 2022]
School of Engineering and Technology, University of Washington - Tacoma

L3.130

130

WE WILL RETURN AT 7:00PM

131

TCSS 562 OFFICE HOURS

PLEASE SAY HELLO

L3.132

132