

# **CSE 240A:**

# **Principles in Computer Architecture**

## **Performance**

Jishen Zhao (<https://cseweb.ucsd.edu/~jzhao/>)

[Adapted in part from Dean Tullsen, Mary Jane Irwin, Joe Devietti, Onur Mutlu, and others]

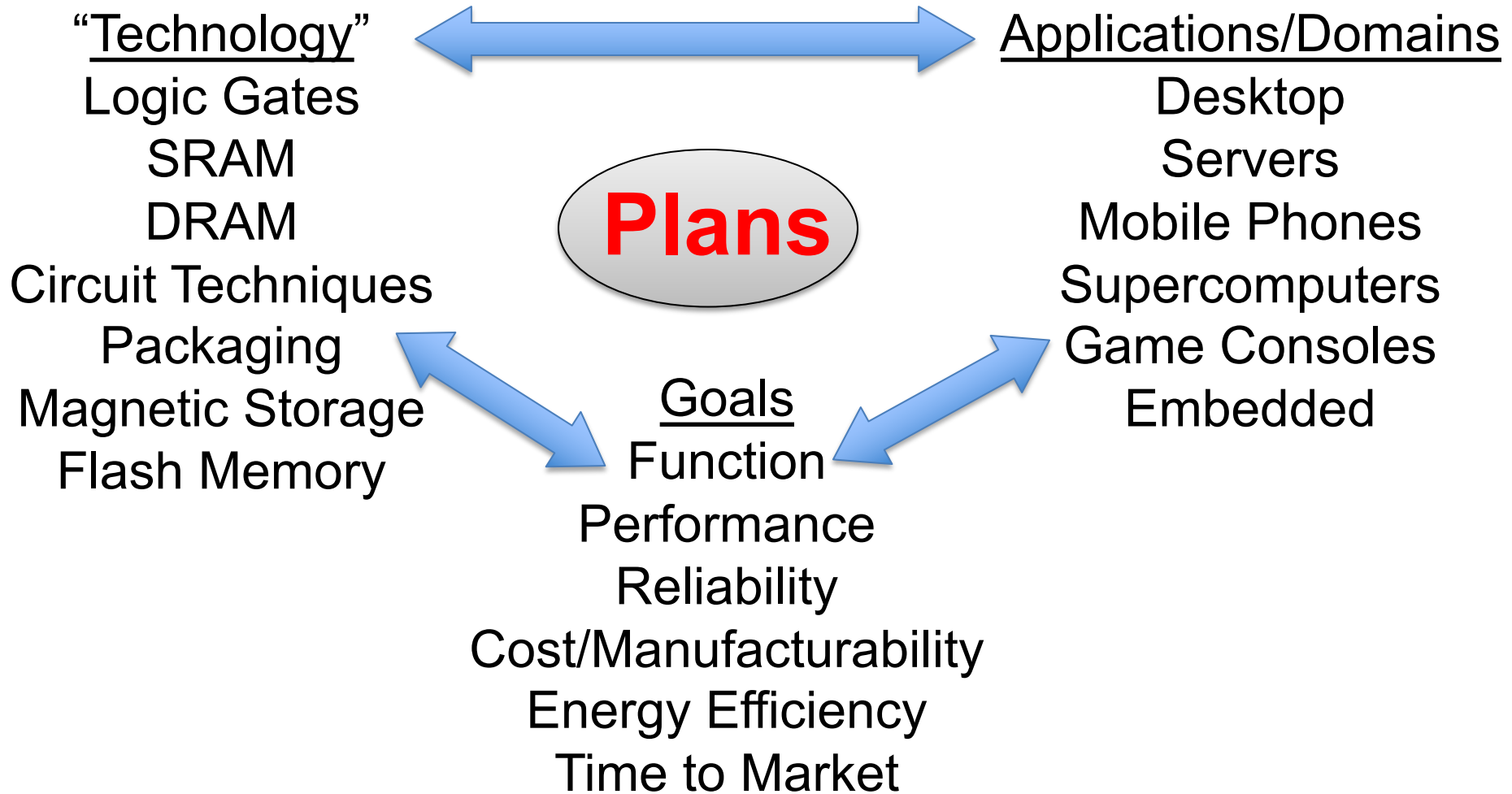
# Reminders

---

- My office hour Thursdays 2:30pm
- Our final exam can count for MS comprehensive exam
- Should have enrolled everyone on the waitlist
- Next week – Experiment with new class format
  - Watch lecture videos that are posted ahead of time
  - Q&A during the class time (attending the class is **not mandatory**)
  - Fill out a survey about class format
- Benefits (some shared with the flipped classroom model)
  - Learn the class materials on your own pace
  - Customized Q&A: Use the class time to provide everyone with more opportunities to discuss with me on any questions – lecture materials, homework assignments, computer architecture research, etc.

# Review: What is computer architecture?

---



# Today: Performance

---

- Performance metrics
  - Latency and throughput
  - Speedup
  - Averaging
- CPU Performance



**Lots of in-class exercises/examples**

# Performance Metrics

# Performance: Latency vs. Throughput

---

- **Latency (execution time)**: time to finish a fixed task
- **Throughput (bandwidth)**: number of tasks per unit time
  - Different: exploit parallelism for throughput, not latency



- Choose definition of performance that matches your goals
  - Scientific program? latency.
  - Web server? throughput.

# Examples

---

- How to measure the performance of moving people for 10 miles **round trip**?
  - Car: capacity = 5, speed = 60 miles/hour
  - Bus: capacity = 60, speed = 20 miles/hour
  - **Latency**: how long does ~ take to move one person for 20 miles?
  - **Throughput**: how many people can ~ move per hour?



Answer:

- **Latency**: **car = 20 min**, bus = 60 min
- **Throughput**: car = 15 PPH, **bus = 60 PPH**

# Examples

---

- Fastest way to send 10TB of data from US to UK?  
FTP, SMB, Rsync / Robocopy, other?



Used FedEx overnight to deliver the drive



Even 1 Gbps data transfer takes days!





# Amazon Does This...

| Available Internet Connection | Theoretical Min. Number of Days to Transfer 1TB at 80% Network Utilization | When to Consider AWS Import/Export? |
|-------------------------------|--|-------------------------------------|
| T1 (1.544Mbps)                | 82 days  | 100GB or more                       |
| 10Mbps                        | 13 days  | 600GB or more                       |
| T3 (44.736Mbps)               | 3 days   | 2TB or more                         |
| 100Mbps                       | 1 to 2 days  | 5TB or more                         |
| 1000Mbps                      | Less than 1 day  | 60TB or more                        |



[Amazon Web Services](#) » [AWS Import/Export](#) » AWS Import/Export Calculator

|                             |                          |                           |
|-----------------------------|--------------------------|---------------------------|
| Operation Type              |                          | Import to S3 ▾            |
| Location                    | AWS Region               | US Standard Region ▾      |
| AWS Import/Export Data Load | Total Terabytes to Load  | 1 <input type="text"/> TB |
|                             | Number of Devices        | 1 <input type="text"/>    |
|                             | Wipe Device After Import | No ▾                      |
| Estimated Transfer Speed    | Average File Size*       | 1 <input type="text"/> MB |
|                             | Interface Type           | eSATA ▾                   |
|                             | Transfer Speed**         | 22.51 MB/sec              |

# What we learned

---

Measuring performance

## **Latency & throughput**



# Comparing Performance - Speedup

---

- Speedup of A over B
  - $X = \text{Latency}(B) / \text{Latency}(A)$  (divide by the faster)
  - $X = \text{Throughput}(A) / \text{Throughput}(B)$  (divide by the slower)
- A is X% faster than B if
  - $X = ((\text{Latency}(B) / \text{Latency}(A)) - 1) * 100$
  - $X = ((\text{Throughput}(A) / \text{Throughput}(B)) - 1) * 100$
  - $\text{Latency}(A) = \text{Latency}(B) / (1 + (X/100))$
  - $\text{Throughput}(A) = \text{Throughput}(B) * (1 + (X/100))$
- Car/bus example
  - Latency?
  - Throughput?
  - See next slide...

# Car/bus example

---

- **Latency:** car = **20 min**, bus = 60 min
- **Throughput:** car = 15 PPH, **bus = 60 PPH**

## Speedup?

- Latency:
  - Speedup of car over bus is 3
  - Car is 200% faster than bus
- Throughput:
  - Speedup of bus over car is 4
  - Bus is 300% faster than car

# Comparing Performance - Speedup

---

- Program A runs for 200 cycles
- Program B runs for 350 cycles

What is “cycle”?  
Execution time \* clock frequency  
i.e., second \* Hz

- Speedup of A over B?
  - $\text{Speedup} = 350/200 = 1.75$
  - As a percentage:  $(1.75 - 1) * 100 = 75\%$  (Program A runs 75% faster than program B)
- If program C is 50% faster than A, how many cycles does C run for?
  - 133 cycles



# Note

---

- Speedup of A over B
  - $X = \text{Latency}(B) / \text{Latency}(A)$
  - $X = \text{Throughput}(A) / \text{Throughput}(B)$

What if  $X < 1$ ?

-- means A is slower than B

# Speedup and % Increase and Decrease

---

- Program A runs for 200 cycles
- Program B runs for 350 cycles
- Percent increase and decrease are **not the same**.
  - % increase of cycles:  $((350 - 200)/200) * 100 = 75\%$
  - % decrease of cycles:  $((350 - 200)/350) * 100 = 42.3\%$

# What we learned

---

Comparing performance

## **Speedup**

Performance metrics

**Latency, throughput, speedup**



# Averaging performance

# Mean (Average) Performance Numbers

---

- **Arithmetic:**  $(1/N) * \sum_{P=1..N} \text{Latency}(P)$ 
    - For units that are proportional to time (e.g., latency)
  - **Harmonic:**  $N / \sum_{P=1..N} 1/\text{Throughput}(P)$ 
    - For units that are inversely proportional to time (e.g., throughput)
- 
- You can add latencies, but not throughputs
    - $\text{Latency}(P1+P2, A) = \text{Latency}(P1, A) + \text{Latency}(P2, A)$
    - $\text{Throughput}(P1+P2, A) \neq \text{Throughput}(P1, A) + \text{Throughput}(P2, A)$
  - **Geometric:**  $N\sqrt[N]{\prod_{P=1..N} \text{Speedup}(P)}$ 
    - For unitless quantities (e.g., speedup ratios)

# For Example...

---

1 mile @ 30 miles/hour + 1 mile @ 90 miles/hour

- You drive two miles
  - 30 miles per hour for the first mile
  - 90 miles per hour for the second mile
- Question: what was your average speed?
  - Hint: the answer is not 60 miles per hour
  - Why?



# Answer: 45 miles/hour

---

- You drive two miles
  - 30 miles per hour for the first mile
  - 90 miles per hour for the second mile
- Question: what was your average speed?
  - Hint: the answer is not 60 miles per hour
  - 0.03333 hours per mile for 1 mile
  - 0.01111 hours per mile for 1 mile
  - 0.04444 hours for 2 miles
  - = 45 miles per hour
  - $\neq (30 + 90) / 2$

# What we learned

---

Averaging performance

**Arithmetic mean for latency**

**Harmonic mean for throughput**

**Geometric mean for speedup**



# CPU Performance

How to evaluate

Latency, throughput, and speedup

# CPU Performance Equation

---

- Latency = seconds / program =
  - $(\text{insns} / \text{program}) * (\text{cycles} / \text{insns}) * (\text{seconds} / \text{cycle})$
- **Insns / program**: insn count
  - Impacted by program, compiler, ISA
- **Cycles / insn**: **CPI**
  - Impacted by program, compiler, ISA, **micro-arch**
- **Seconds / cycle**: **clock period (Hz)**
  - Impacted by micro-arch, technology
- For low latency (better performance) minimize all three
  - Difficult: often pull against one another
  - Example we have seen: RISC vs. CISC ISAs
    - ±RISC: low CPI/clock period, high insn count
    - ±CISC: low insn count, high CPI/clock period

# Cycles per Instruction (CPI)

---

- **CPI**: Cycles/instruction
  - **IPC** =  $1/\text{CPI}$ 
    - Used more frequently than CPI
    - Favored because “bigger is better”, but harder to compute with
  - Different instructions have different cycle costs
    - E.g., “add” typically takes 1 cycle, “divide” takes >10 cycles
  - Depends on relative instruction frequencies (what if idle)
- CPI example
  - A program executes equal: integer, floating point (FP), memory ops
  - Cycles per instruction type: integer = 1, memory = 2, FP = 3
  - What is the CPI?  $(1 \text{ cycle} + 2 \text{ cycles} + 3 \text{ cycles}) / 3 \text{ instrs} = 2$   
Calculated in another way:  $33\% \times 1 + 33\% \times 2 + 33\% \times 3 = 2$



# CPI Example

---

- Assume a processor with instruction frequencies and costs
  - Integer ALU: 50%, 1 cycle
  - Load: 20%, 5 cycle
  - Store: 10%, 1 cycle
  - Branch: 20%, 2 cycle
- Which change would improve performance more?
  - A. "Branch prediction" to reduce branch cost to 1 cycle?
  - B. Faster data memory to reduce load cost to 3 cycles?
- Compute CPI
  - Base =  $0.5*1 + 0.2*5 + 0.1*1 + 0.2*2 = 2$  CPI
  - A =  $0.5*1 + 0.2*5 + 0.1*1 + 0.2*1 = 1.8$  CPI (1.11x or 11% faster)
  - B =  $0.5*1 + 0.2*3 + 0.1*1 + 0.2*2 = 1.6$  CPI (1.25x or 25% faster)
    - **B is the winner**



# Measuring CPI

---

- How are CPI and execution-time actually measured?
  - Execution time? stopwatch timer (Unix "time" command)
  - Cycles = execution time \* clock frequency
  - How is instruction count measured? → truly executed insns
- Hardware event counters
  - Available in most processors today
  - One way to measure instruction count
- Cycle-level micro-architecture simulation
  - + Measure exactly what you want ... and impact of potential fixes!
  - Method of choice for many micro-architects
- More useful is CPI breakdown ( $CPI_{CPU}$ ,  $CPI_{MEM}$ , etc.)
  - So we know what performance problems are and what to fix

# Summary: Performance Metrics

---

- **Performance metrics:** Latency & throughput
- **Comparing performance:** Speedup
- **Averaging performance:**
  - Arithmetic mean
  - Harmonic mean
  - Geometric mean
- **Measuring CPU performance:** CPI (IPC)

