Principles of Computer Architecture

CSE 240A

Fall 2024

Hadi Esmaeilzadeh

hadi@ucsd.edu

University of California, San Diego

How to Summarize Performance

	Computer A	Computer B	Computer C
Program 1	1	10	20
Program 2	1000	100	20
Total time	1001	110	40

Which machine is fastest?

How to Summarize Performance

- Arithmetic Mean $\frac{1}{n}\sum_{i=1}^{n} Time_i$

- Weighted Arithmetic Mean $\sum_{i=1}^{n} Time_i * Weight_i$

  where the sum of the weights is 1.

- Geometric Mean $\sqrt[n]{\prod_{i=1}^{n} ExecutionTimeRatio_i} = \frac{\sqrt[n]{\prod_{i=1}^{n} ExecutionTime_i}}{ExecutionTime_{base}}$

Summarizing Performance

Machines:	A	B
Program 1	1	10
Program 2	1000	100

	Set(1)	Set(2)	Set(3)
W_1	.5	.909	.999
W_2	.5	.091	.001

Arith M/Set(1)	500.5	55
Arith M/Set(2)	91.82	18.18
Arith M/Set(3)	2	10.09
Geo M	31.6	31.6

Arith M: Speedup (A/B) = (10 / 1 + 100 / 1000) / 2 = 5.05
Arith M: Speedup (A/B) = (10 + 100)/(1+1000) = 0.10989

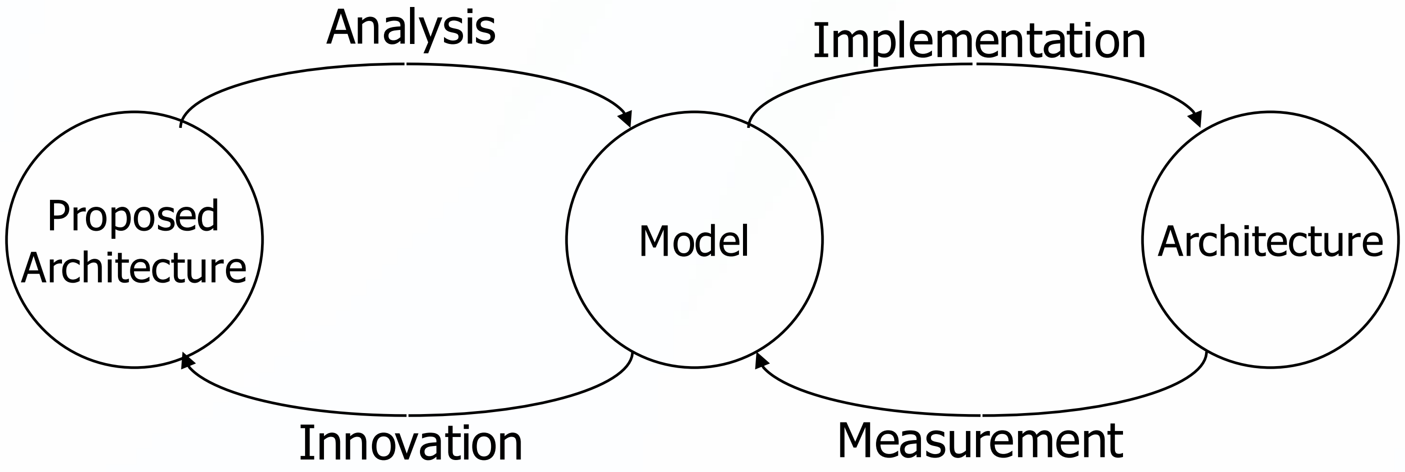
Geo M: Speedup (A/B) = sqrt(sqrt(10 / 1) * sqrt(100 /1000)) = 1
Geo M: Speedup (A/B) = sqrt(10 * 100)/sqrt(1*1000) = 1

Summarizing Performance

- Even the unweighted arithmetic mean implies a weighting
- Ratios of geometric means never change (regardless of which machine is used as the base), and always give equal weight to all benchmarks
- To give unequal weight requires weighted arithmetic mean

Analyzing Performance

- That was all about measuring performance. What tools do we use to analyze (predict) performance in the absence of something to measure?
  - models, equations, queueing theory, mean value analysis, instruction-level simulation, gate-level simulation, ...



Because we would like to understand the effects of changes we make to the architecture, we mode/measure performance

Speedup (due to architectural change)

- Speedup is just relative performance on the same machine with something changed.
- From before, then:

$$\text{Speedup} = \text{Relative Performance} = \frac{\text{ET for entire task without change}}{\text{ET for entire task with change}}$$

Speedup (due to architectural change)

- Speedup is just relative performance on the same machine with something changed.
- From before, then:

$$\text{Speedup} = \text{Relative Performance} = \frac{\text{ET for entire task without change}}{\text{ET for entire task with change}}$$

Suppose the change only affects part of execution time…

Amdahl's Law

The impact of a performance improvement is limited by the percent of execution time affected by the improvement

$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

Amdahl's Law

The impact of a performance improvement is limited by the percent of execution time affected by the improvement

$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

Make the common case fast!!

TAKE AWAY: Focus your optimizations on where it has the most impact

Attendance

Selection	Answer
Present	A
Present	B
Present	C
Present	B
Present	D

Example

- Program A runs for 20 seconds, but 5 seconds of that time is just waiting for memory. If we double the speed of the memory subsystem, what is the speedup?

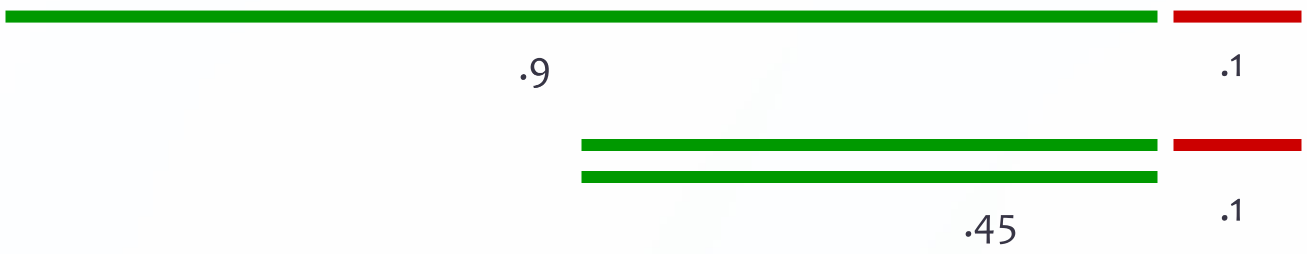
Selection	Speedup
A	2.00
B	1.50
C	1.28
D	1.14
E	None of the above

Amdahl’s Law in the Era of Multicores

Thread Level Parallelism

Amdahl’s Law in the Era of Multicores

Speedup



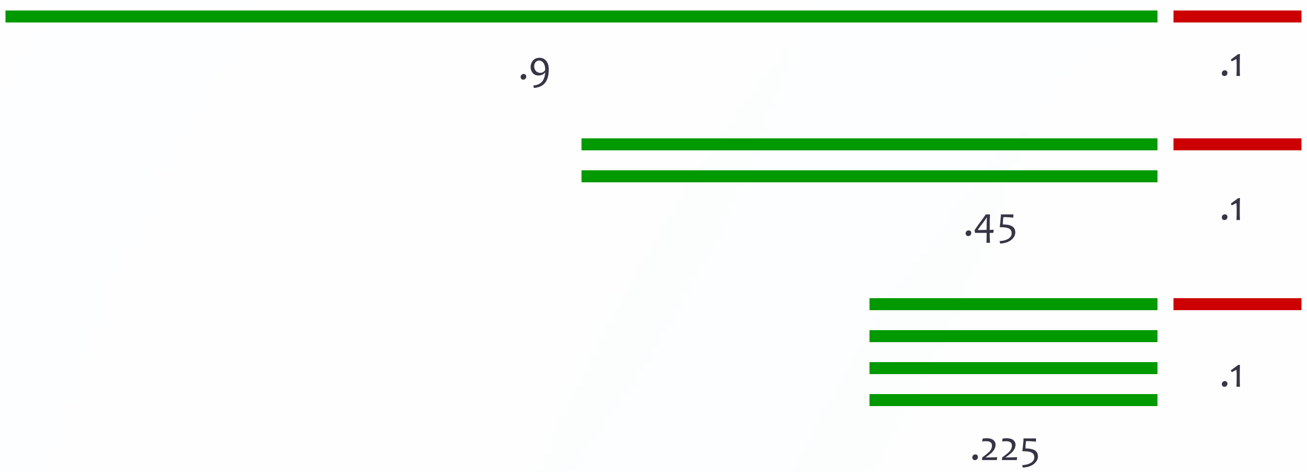
1.0

1/.55 = **1.82**

Thread Level Parallelism

Amdahl’s Law in the Era of Multicores

Speedup



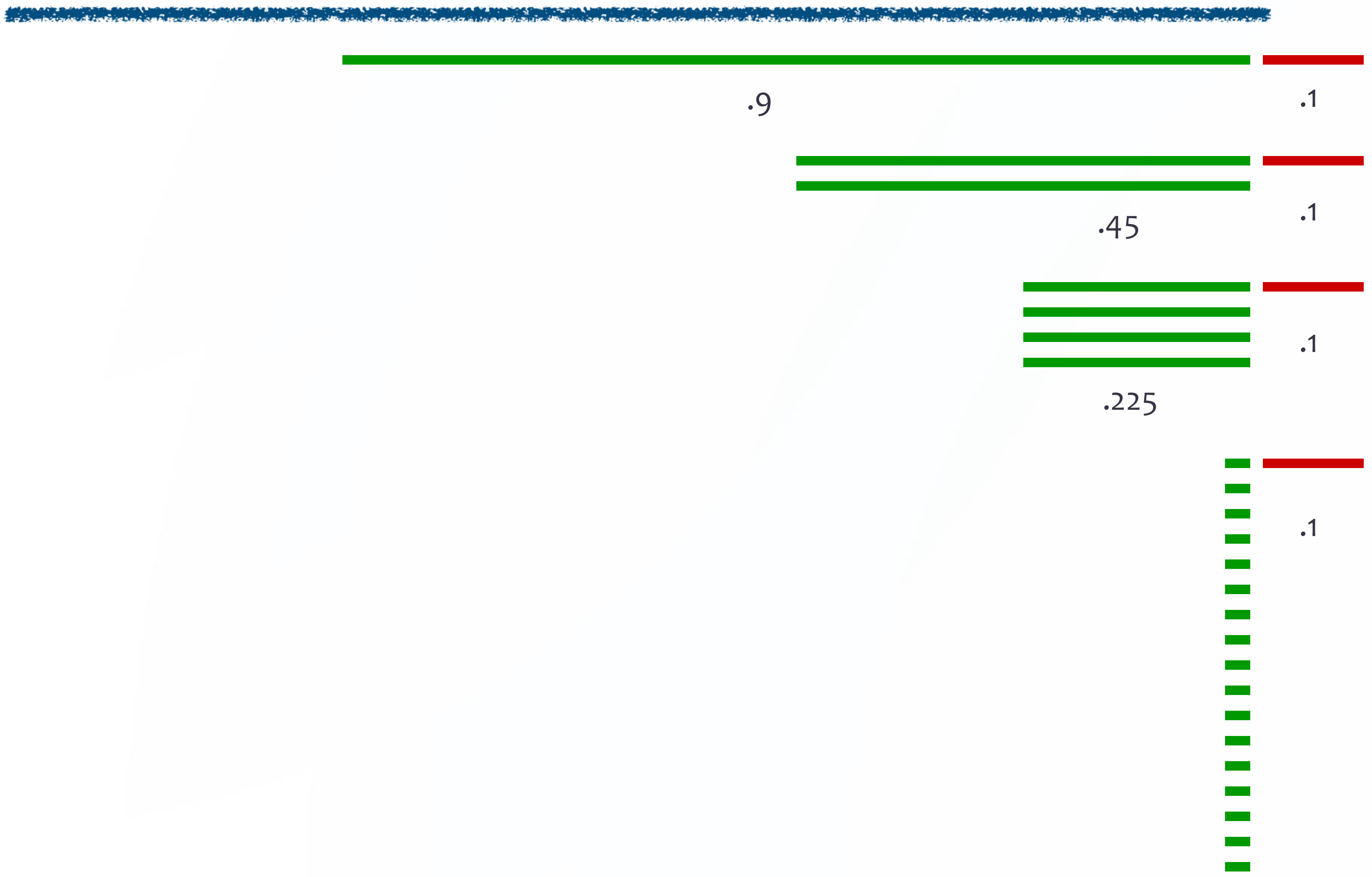
1.0

1/.55 = **1.82**

1/.325 = **3.07**

Thread Level Parallelism

Amdahl's Law in the Era of Multicores



Speedup

1.0

1/.55 = **1.82**

1/.325 = **3.07**

< 10

Is there any other way besides TLP?

- Speedup a single application through TLP runs into problems when using multicore because of diminishing returns and dominance of serial execution.

- What if there was a time constraint?

Human perception (graphics)

Earthquake prediction

Weather prediction

Scale Up (Data Level Parallelism instead of TLP)

Beyond Time…

- Energy = Power * Time
- Joules = Watts * Seconds

Beyond Time…

- Energy = power * time
- Joules = Watts * seconds

- If you can halve execution time of a program but to do so, you raise the average power by 50%. Did you save energy?

Selection	Answer
A	Yes
B	No

Beyond Time...

- Energy = power * time
- Joules = Watts * seconds

- If you can halve execution time of a program but to do so, you raise the average power by 25%. Did you save energy?

- We'll touch on energy concerns in 240A a bit, but 240C (and 240B, to some extent) will discuss this.

Back to CPUs, what is Time?

What is Time?

CPU Execution Time = CPU clock cycles * Clock cycle time

= CPU clock cycles / Clock rate

Every conventional processor has a clock with an associated clock cycle time or clock rate.

Every program runs in an integral number of clock cycles.

GHz = billions of cycles/second

X GHz = 1/X nanoseconds cycle time

How many clock cycles?

Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)

or

CPI = CPU clock cycles / Instruction count

All Together Now

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

All Together Now

seconds

cycles/instruction

CPU Execution Time = Instruction Count × CPI × Clock Cycle Time

instructions

seconds/cycle

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

IC = 40 billion, 4 GHz processor, execution time of 30 seconds.

What is the CPI for this program?

Selection	CPI
A	3
B	30
C	1.5
D	15*10^9
E	None of the above

Examples

- 4 GHz processor, program runs in 30 seconds, executing 40 billion instructions with CPI above
- New compiler reduces IC to 32 billion, but changes CPI to 3.3: good or bad?

Putting it together

- Suppose you have a 2 GHz Core i7 called Machine A.
- You also have a 4 GHz Core i7 called Machine B.
- Let's say they have the same underlying architecture (same pipelines, caches, etc.), just one is running at a faster clock rate.
- Running program X (same binary, etc.), could Machine A have an average CPI of 0.9 for program X and Machine B have an average CPI of 1.1 for program X?

Selection	Answer
A	Yes, this is possible but unlikely
B	Yes, this is possible and likely
C	No, this is impossible

Who Affects Performance?

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

- programmer
- compiler
- instruction-set architect
- machine architect
- hardware designer

What Affects Performance?

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

- pipelining
- superpipelining
- cache
- from CISC to RISC
- superscalar

Key Points

- We need to be precise about how to specify performance.
- Performance is only meaningful in the context of a workload.
- Be careful how you summarize performance.
- Amdahl’s law
- ET = IC * CPI * CT