

Lecture 03

- Topics:
 - Means
 - Amdahl's Law
 - (With lots of examples)
 - Little's Law
 - Cost

Material based on work of many others - Niemier, Katz, Brockman, Kogge, Hu, Sorin, Roth, Hill, Wood, Sohi, Smith, Vijaykumar, Lipasti, Pruvlovic, etc. etc.

Reporting Average Performance

- averages: something architects frequently get wrong
 - (pay attention now and you won't get them wrong on exams)
- important things about averages (i.e., means)
 - ideally proportional to execution time (ultimate metric)
 - Arithmetic Mean (AM) for times
 - Harmonic Mean (HM) for rates (IPCs)
 - Geometric Mean (GM) for ratios (speedups)
 - there is no such thing as the average program
 - use average when absolutely necessary

What Does the Mean Mean?

- arithmetic mean (AM):
 - average execution times of N programs
 - $\sum_{1..N}(\text{time}(i)) / N$
- harmonic mean (HM):
 - average IPCs of N programs
 - arithmetic mean cannot be used for rates (e.g., IPCs)
 - 30 MPH for 1 mile + 90 MPH for 1 mile != avg. 60 MPH
 - $N / \sum_{1..N}(1 / \text{rate}(i))$
- geometric mean (GM):
 - average speedups of N programs
 - $N \sqrt[N]{\prod_{1..N}(\text{speedup}(i))}$

What Does the Mean Mean?

- what if programs run at different frequencies within workload?
 - "weighting"
 - weighted AM = $(\sum_{1..N} w(i) * \text{time}(i)) / N$

GM Weirdness

- what about averaging ratios (speedups)?
 - HM / AM change depending on which machine is the base

	Machine A	Machine B	B/A	A/B
Program 1	1	10	10	0.1
Program 2	1000	100	0.1	10
		AM	$(10+.1)/2=5.05$ B is 5.05x faster!	$(.1+10)/2=5.05$ A is 5.05x faster!
		HM	Like AMs, HMs are the same too!	Like AMs, HMs are the same too!
		GM	$\text{sqrt}(10 \times 0.1) = 1$	$\text{sqrt}(0.1 \times 10) = 1$

Correct behavior, but may need to consider things like execution time too...

- geometric mean of ratios is not proportional to total time!
 - if we take total execution time, B is 9.1 times faster
 - GM says they are equal

Amdahl's Law

- Qualifies performance gain
- Amdahl's Law defined...
 - The performance improvement to be gained from using some faster mode of execution is limited by the amount of time the enhancement is actually used.

- Amdahl's Law defines speedup:

$$\text{Speedup} = \frac{\text{Perf. for entire task using enhancement when possible}}{\text{Perf. For entire task without using enhancement}}$$

Or

$$\text{Speedup} = \frac{\text{Execution time for entire task without enhancement}}{\text{Execution time for entire task using enhancement when possible}}$$

Amdahl's Law and Speedup

- Speedup tells us how much faster the machine will run with an enhancement
- 2 things to consider:
 - 1st...
 - Fraction of the computation time in the original machine that can use the enhancement
 - i.e. if a program executes in 30 seconds and 15 seconds of exec. uses enhancement, fraction = $\frac{1}{2}$ (always < 1)
 - 2nd...
 - Improvement gained by enhancement (i.e. how much faster does the program run overall)
 - i.e. if enhanced task takes 3.5 seconds and original task took 7, we say the speedup is 2 (always > 1)

Deriving the previous formula

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution Time}_{\text{old}}}{\text{Execution Time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

1 ← normalized old execution time

$(1 - \text{Fraction}_{\text{enhanced}})$ + $\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}$

1 - % enhanced (i.e. part of the task will take the same amount of time as before)

$\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}$ → % of task that will run faster how much faster it will run (note: # should be > 1) (otherwise, performance gets worse)

Amdahl's Law

B

Let's learn by example...

Little's Law

- Key Relationship between latency and bandwidth:
 - Average number in system = arrival rate * mean holding time
- Example:
 - How big of a wine cellar should we build?
 - We drink (and buy) an average of 2 bottles per week
 - On average, I want to age my wine 5 years
 - bottles in cellar = 2 bottles/week * 52 weeks/year * 5 years
 - = 520 bottles

Bursty Behavior

- Question:
 - to sustain 2 IPC, how many instructions should processor be able to
 - fetch per cycle?
 - execute per cycle?
 - complete per cycle?
- Answer
 - NOT 2 (more than 2)
 - dependences will cause stalls (under-utilization)
 - if desired performance is X, peak performance must be $> X$
- programs don't always obey "average" behavior
 - can't design processor only to handle average behavior

System Balance

- each system component produces & consumes data
 - make sure data supply and demand is balanced
 - $X \text{ demand} \geq X \text{ supply} \Rightarrow$ computation is "X-bound"
 - e.g., memory bound, CPU-bound, I/O-bound
- X can be bandwidth or latency
 - X is bandwidth \Rightarrow buy more bandwidth
 - X is latency \Rightarrow much tougher problem

Tradeoffs

- "Bandwidth problems can be solved with money. Latency problems are harder, because the speed of light is fixed and you can't bribe God" -David Clark (MIT)
- well...
 - can convert some latency problems to bandwidth problems
 - solve those with money
- We'll get into these issues more later in the semester

Cost

- very important to real designs
 - startup cost
 - one large investment per chip (or family of chips)
 - increases with time
 - per lecture 01, had better give better performance
 - unit cost
 - cost to produce individual copies
 - decreases with time
 - only loose correlation to price and profit
- Moore's corollary: price of high-performance system is constant
 - performance doubles every 18 months
 - cost per function (unit cost) halves every 18 months
 - assumes startup costs are constant (they aren't)

Startup and Unit Costs

- startup cost: manufacturing
 - fabrication plant, clean rooms, lithography, etc. (~\$3B)
 - chip testers/debuggers (~\$5M a piece, typically ~200)
 - few companies can play this game (Intel, IBM, Sun)
 - equipment more expensive as devices shrink
- startup cost: research and development
 - 300-500 person years, mostly spent in verification
 - need more people as designs become more complex
- unit cost: manufacturing
 - raw materials, chemicals, process time (2-5K/wafer)
 - decreased by improved technology & experience

Unit Cost and Die Size (Chip Area)

- unit cost most strongly influenced by physical size of chip (die)
 - semiconductors built on silicon wafers (8")
 - chemical+photolithographic steps create transistor/wire layers
 - In 2004, typical number of metal layers (M) today was 6 ($\alpha = \sim 4$)
 - cost per wafer is roughly constant $C_0 + C_1 * \alpha$ (\$5000)
 - basic cost per chip proportional to chip area (mm^2)
 - typical: 150-200 mm^2 , 50 mm^2 (embedded)-300 mm^2 (Itanium)
 - typical: 300-600 dies per wafer

Unit Cost and Die Size (Chip Area)

- yield (% working chips) inversely proportional to area and α
 - non-zero defect density (manufacturing defect per unit area)
 - $P(\text{working chip}) = (1 + (\text{defect density} * \text{die area}) / \alpha)^{-\alpha}$
- typical defect density: 0.005 per mm^2
- typical yield: $(1 + (0.005 * 200) / 4)^{-4} = 40\%$
- typical cost per chip: $\$5000 / (500 * 40\%) = \25

Unit Cost -> Price

- if chips cost 25\$ to manufacture, why do they cost \$500 to buy?
 - integrated circuit costs 25\$
 - must still be tested, packaged, and tested again
 - testing (time == \$): \$5 per working chip
 - packaging (ceramic+pins): \$30
 - more expensive for more pins or if chip is dissipates a lot of heat
 - packaging yield < 100% (but high)
 - post-packaging test: another \$5
 - total for packaged chip: ~\$65
 - spread startup cost over volume
 - proliferation (i.e., shrinks) is 1,038,378,235th.
- Intel needs to make a profit...and so does Dell

As in the pharmaceutical industry, you pay for the 1st pill, not the 1,038,378,235th.