

# Lecture 2: Metrics to Evaluate Systems

---

- Topics: Metrics: power, reliability, cost, benchmark suites, performance equation, summarizing performance with AM, GM, HM
- Sign up for the class mailing list!
  - Video 1: Using AM as a performance summary
  - Video 2: GM, Performance Equation
  - Video 3: AM vs. HM vs. GM

# Power Consumption Trends

---

- Dyn power  $\propto$  activity x capacitance x voltage<sup>2</sup> x frequency
- Capacitance per transistor and voltage are decreasing, but number of transistors is increasing at a faster rate; hence clock frequency must be kept steady
- Leakage power is also rising; is a function of transistor count, leakage current, and supply voltage
- Power consumption is already between 100-150W in high-performance processors today
- Energy = power x time = (dynpower + lkgpower) x time

# Problem 1

---

- For a processor running at 100% utilization at 100 W, 20% of the power is attributed to leakage. What is the total power dissipation when the processor is running at 50% utilization?

# Problem 1

---

- For a processor running at 100% utilization at 100 W, 20% of the power is attributed to leakage. What is the total power dissipation when the processor is running at 50% utilization?

$$\begin{aligned}\text{Total power} &= \text{dynamic power} + \text{leakage power} \\ &= 80\text{W} \times 50\% + 20\text{W} \\ &= 60\text{W}\end{aligned}$$

# Power Vs. Energy

---

- Energy is the ultimate metric: it tells us the true “cost” of performing a fixed task
- Power (energy/time) poses constraints; can only work fast enough to max out the power delivery or cooling solution
- If processor A consumes 1.2x the power of processor B, but finishes the task in 30% less time, its relative energy is  $1.2 \times 0.7 = 0.84$ ; Proc-A is better, assuming that 1.2x power can be supported by the system

## Problem 2

---

- If processor A consumes 1.4x the power of processor B, but finishes the task in 20% less time, which processor would you pick:
  - (a) if you were constrained by power delivery constraints?
  - (b) if you were trying to minimize energy per operation?
  - (c) if you were trying to minimize response times?

# Problem 2

---

- If processor A consumes 1.4x the power of processor B, but finishes the task in 20% less time, which processor would you pick:
  - (a) if you were constrained by power delivery constraints?  
Proc-B
  - (b) if you were trying to minimize energy per operation?  
Proc-A is  $1.4 \times 0.8 = 1.12$  times the energy of Proc-B
  - (c) if you were trying to minimize response times?  
Proc-A is faster, but we could scale up the frequency (and power) of Proc-B and match Proc-A's response time (while still doing better in terms of power and energy)

# Reducing Power and Energy

---

- Can gate off transistors that are inactive (reduces leakage)
- Design for typical case and throttle down when activity exceeds a threshold
- DFS: Dynamic frequency scaling -- only reduces frequency and dynamic power, but hurts energy
- DVFS: Dynamic voltage and frequency scaling – can reduce voltage and frequency by (say) 10%; can slow a program by (say) 8%, but reduce dynamic power by 27%, reduce total power by (say) 23%, reduce total energy by 17%  
(Note: voltage drop  $\square$  slow transistor  $\square$  freq drop)



# Problem 3

---

- Processor-A at 3 GHz consumes 80 W of dynamic power and 20 W of static power. It completes a program in 20 seconds.  
What is the energy consumption if I scale frequency down by 20%?  
  
What is the energy consumption if I scale frequency and voltage down by 20%?

# Problem 3

---

- Processor-A at 3 GHz consumes 80 W of dynamic power and 20 W of static power. It completes a program in 20 seconds.

What is the energy consumption if I scale frequency down by 20%?

New dynamic power = 64W; New static power = 20W

New execution time = 25 secs (assuming CPU-bound)

Energy = 84 W x 25 secs = 2100 Joules

What is the energy consumption if I scale frequency and voltage down by 20%?

New DP = 41W; New static power = 16W;

New exec time = 25 secs; Energy = 1425 Joules

# Other Technology Trends

---

- DRAM density increases by 40-60% per year, latency has reduced by 33% in 10 years (the memory wall!), bandwidth improves twice as fast as latency decreases
- Disk density improves by 100% every year, latency improvement similar to DRAM
- Emergence of NVRAM technologies that can provide a bridge between DRAM and hard disk drives
- Also, growing concerns over reliability (since transistors are smaller, operating at low voltages, and there are so many of them)

# Defining Reliability and Availability

---

- A system toggles between
  - Service accomplishment: service matches specifications
  - Service interruption: services deviates from specs
- The toggle is caused by *failures* and *restorations*
- Reliability measures continuous service accomplishment and is usually expressed as mean time to failure (MTTF)
- Availability measures fraction of time that service matches specifications, expressed as  $MTTF / (MTTF + MTTR)$

# Cost

---

- Cost is determined by many factors: volume, yield, manufacturing maturity, processing steps, etc.
- One important determinant: area of the chip
- Small area  $\square$  more chips per wafer
- Small area  $\square$  one defect leads us to discard a small-area chip, i.e., yield goes up
- Roughly speaking, half the area  $\square$  one-third the cost

# Measuring Performance

---

- Two primary metrics: wall clock time (response time for a program) and throughput (jobs performed in unit time)
- To optimize throughput, must ensure that there is minimal waste of resources

# Benchmark Suites

---

- Performance is measured with benchmark suites: a collection of programs that are likely relevant to the user
  - SPEC CPU 2006: cpu-oriented programs (for desktops)
  - SPECweb, TPC: throughput-oriented (for servers)
  - EEMBC: for embedded processors/workloads

# Summarizing Performance

---

- Consider 25 programs from a benchmark set – how do we capture the behavior of all 25 programs with a single number?

	P1	P2	P3
Sys-A	10	8	25
Sys-B	12	9	20
Sys-C	8	8	30

- Sum of execution times (AM)
- Sum of weighted execution times (AM)
- Geometric mean of execution times (GM)



# Problem 4

---

- Consider 3 programs from a benchmark set. Assume that system-A is the reference machine. How does the performance of system-C compare against that of system-B (for all 3 metrics)?

	P1	P2	P3
Sys-A	5	10	20
Sys-B	6	8	18
Sys-C	7	9	14

- Sum of execution times (AM)
- Sum of weighted execution times (AM)
- Geometric mean of execution times (GM)

# Problem 4

---

- Consider 3 programs from a benchmark set. Assume that system-A is the reference machine. How does the performance of system-C compare against that of system-B (for all 3 metrics)?

	P1	P2	P3	S.E.T	S.W.E.T	GM
Sys-A	5	10	20	35	3	10
Sys-B	6	8	18	32	2.9	9.5
Sys-C	7	9	14	30	3	9.6

- Relative to C, B provides a speedup of 1.03 (S.W.E.T) or 1.01 (GM) or 0.94 (S.E.T)
- Relative to C, B reduces execution time by 3.3% (S.W.E.T) or 1% (GM) or -6.7% (S.E.T)

# Sum of Weighted Exec Times – Example

---

- We fixed a reference machine X and ran 4 programs A, B, C, D on it such that each program ran for 1 second
- The exact same workload (the four programs execute the same number of instructions that they did on machine X) is run on a new machine Y and the execution times for each program are 0.8, 1.1, 0.5, 2
- With AM of normalized execution times, we can conclude that Y is 1.1 times slower than X – perhaps, not for all workloads, but definitely for one specific workload (where all programs run on the ref-machine for an equal #cycles)

# GM Example

---

	Computer-A	Computer-B	Computer-C
P1	1 sec	10 secs	20 secs
P2	1000 secs	100 secs	20 secs

Conclusion with GMs: (i) A=B  
(ii) C is ~1.6 times faster

- For (i) to be true, P1 must occur 100 times for every occurrence of P2
- With the above assumption, (ii) is no longer true

Hence, GM can lead to inconsistencies

# Summarizing Performance

---

- GM: does not require a reference machine, but does not predict performance very well
  - So we multiplied execution times and determined that sys-A is 1.2x faster...but on what workload?
- AM: does predict performance for a specific workload, but that workload was determined by executing programs on a reference machine
  - Every year or so, the reference machine will have to be updated

# CPU Performance Equation

---

- Clock cycle time =  $1 / \text{clock speed}$
- CPU time = clock cycle time x cycles per instruction x number of instructions
- Influencing factors for each:
  - clock cycle time: technology and pipeline
  - CPI: architecture and instruction set design
  - instruction count: instruction set design and compiler
- CPI (cycles per instruction) or IPC (instructions per cycle) can not be accurately estimated analytically

# Problem 5

---

- My new laptop has an IPC that is 20% worse than my old laptop. It has a clock speed that is 30% higher than the old laptop. I'm running the same binaries on both machines. What speedup is my new laptop providing?

# Problem 5

---

- My new laptop has an IPC that is 20% worse than my old laptop. It has a clock speed that is 30% higher than the old laptop. I'm running the same binaries on both machines. What speedup is my new laptop providing?

Exec time = cycle time \* CPI \* instrs

Perf = clock speed \* IPC / instrs

Speedup = new perf / old perf

= new clock speed \* new IPC / old clock speed \* old IPC

= 1.3 \* 0.8 = 1.04



# An Alternative Perspective - I

---

- Each program is assumed to run for an equal number of cycles, so we're fair to each program
- The number of instructions executed per cycle is a measure of how well a program is doing on a system
- The appropriate summary measure is sum of IPCs or  
AM of IPCs =  $\frac{1.2 \text{ instr}}{\text{cyc}} + \frac{1.8 \text{ instr}}{\text{cyc}} + \frac{0.5 \text{ instr}}{\text{cyc}}$
- This measure implicitly assumes that 1 instr in prog-A has the same importance as 1 instr in prog-B

# An Alternative Perspective - II

---

- Each program is assumed to run for an equal number of instructions, so we're fair to each program
- The number of cycles required per instruction is a measure of how well a program is doing on a system
- The appropriate summary measure is sum of CPIs or  
AM of CPIs =  $\frac{0.8 \text{ cyc}}{\text{instr}} + \frac{0.6 \text{ cyc}}{\text{instr}} + \frac{2.0 \text{ cyc}}{\text{instr}}$
- This measure implicitly assumes that 1 instr in prog-A has the same importance as 1 instr in prog-B

# AM and HM

---

- Note that  $AM \text{ of IPCs} = 1 / HM \text{ of CPIs}$  and  
 $AM \text{ of CPIs} = 1 / HM \text{ of IPCs}$
- So if the programs in a benchmark suite are weighted such that each runs for an equal number of cycles, then AM of IPCs or HM of CPIs are both appropriate measures
- If the programs in a benchmark suite are weighted such that each runs for an equal number of instructions, then AM of CPIs or HM of IPCs are both appropriate measures

# AM vs. GM

---

- GM of IPCs =  $1 / \text{GM of CPIs}$
- AM of IPCs represents thruput for a workload where each program runs sequentially for 1 cycle each; but high-IPC programs contribute more to the AM
- GM of IPCs does not represent run-time for any real workload (what does it mean to multiply instructions?); but every program's IPC contributes equally to the final measure

# Problem 6

---

- My new laptop has a clock speed that is 30% higher than the old laptop. I'm running the same binaries on both machines. Their IPCs are listed below. I run the binaries such that each binary gets an equal share of CPU time. What speedup is my new laptop providing?

	P1	P2	P3
Old-IPC	1.2	1.6	2.0
New-IPC	1.6	1.6	1.6

# Problem 6

---

- My new laptop has a clock speed that is 30% higher than the old laptop. I'm running the same binaries on both machines. Their IPCs are listed below. I run the binaries such that each binary gets an equal share of CPU time. What speedup is my new laptop providing?

	P1	P2	P3	AM	GM
Old-IPC	1.2	1.6	2.0	1.6	1.57
New-IPC	1.6	1.6	1.6	1.6	1.6

AM of IPCs is the right measure. Could have also used GM. Speedup with AM would be 1.3.

# Speedup Vs. Percentage

---

- “Speedup” is a ratio = old exec time / new exec time
- “Improvement”, “Increase”, “Decrease” usually refer to percentage relative to the baseline  
= (new perf – old perf) / old perf
- A program ran in 100 seconds on my old laptop and in 70 seconds on my new laptop
  - What is the speedup?
  - What is the percentage increase in performance?
  - What is the reduction in execution time?

# Speedup Vs. Percentage

---

- “Speedup” is a ratio = old exec time / new exec time
- “Improvement”, “Increase”, “Decrease” usually refer to percentage relative to the baseline  
= (new perf – old perf) / old perf
- A program ran in 100 seconds on my old laptop and in 70 seconds on my new laptop
  - What is the speedup?  $(1/70) / (1/100) = 1.42$
  - What is the percentage increase in performance?  
 $(1/70 - 1/100) / (1/100) = 42\%$
  - What is the reduction in execution time? 30%



# Title

---

- Bullet