Amdahl's Law

Useful for evaluating the impact of a change. (A general observation.)

Insight:

- Improving a feature cannot improve performance beyond the use of the feature

Suppose we introduce a particular enhancement that improves fraction f of execution time by factor S. Then:

$$\text{Execution time}_{new} = ((1 - f) + f/S) \times \text{Execution time}$$

Example

Task: complete EE/CS 314 project

- Walk to lab: 15 minutes
- Work in lab: 600 minutes
- Food: 45 minutes
- Walk home: 10 minutes
- Total time = 670 minutes.

Skip meals! And sleep in the lab! :-)

$\Rightarrow$ 10.4% improvement

Observation: doesn't help with the time spent working.

Other Performance Metrics

Commonly used: MIPS

(Millions of instructions per second)

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

Problems with MIPS as a metric:

- Instructions with different capabilities?
- Programs with different instruction mixes?
- Might not predict which machine is faster!

MIPS As A Performance Metric

Consider an optimized and unoptimized version of a program:

	Memory Instrs.	ALU Instrs.	Branch Instrs.	Total Instrs.
Unopt	200 M	150 M	50 M	400 M
Optimized	100 M	100 M	50 M	250 M

	Memory cycles (2x)	ALU cycles (1x)	Branch cycles (3x)	CPI
Unopt	400 M	150 M	150 M	1.75
Optimized	200 M	100 M	150 M	1.8

MIPS As A Performance Metric

Assume a 750MHz Clock:

- MIPS (Unopt): $750/1.75 = 428.6$
- MIPS (opt): $750/1.8 = 416.7$

Execution time:

- Unopt: $CPI \times IC \times 1/CR$
  $= 1.75 \times 400/750 = 0.93$ *secs*
- Opt: $CPI \times IC \times 1/CR$
  $= 1.8 \times 250/750 = 0.6$ *secs*

MIPS doesn't predict which program is faster! (why?)

What Is "Typical"?

- Source of serious debate
- Instruction mix can significantly impact performance
- Benchmarks for performance
  - Started with simple “toy” programs (quicksort, sieve, etc)
  - Moved to synthetic benchmarks (dhrystone, whetstone)
  - Kernels (time critical parts of real programs)
  - Today: SPEC (collection of standard programs)

Summarizing Performance

Given the execution time of a number of programs, can we come up with one figure of merit?

Arithmetic mean:

$$\text{Average time} = \frac{1}{n}\sum_{i=1}^{n} \text{Execution time}_i$$

Assumes each benchmark is equally important; long-running benchmarks dominate.

If we know workload, we can weight execution times:

$$\text{Weighted average time} = \frac{\sum_{i=1}^{n} W_i \times \text{Execution time}_i}{\sum_{i=1}^{n} W_i}$$

Normalized Performance

Pick weights to be 1/(time on a reference machine)

What if we attempt a summary using normalized performance? Consider two machines A and B:

	P1 secs	P2 secs	Relative to A			Relative to B		
			P1	P2	AM	P1	P2	AM
A	10	50	1	1	1	0.1	50	25
B	100	1	10	0.02	5	1	1	1

Total execution time on A: 60 secs
Total execution time on B: 101 secs

If equal runs of programs P1 and P2, A is 1.7 times faster than B.

Geometric Mean

Can be used to combine relative measures.

$$\text{Geometric Mean} = \sqrt[n]{\prod_{i=1}^{n} R_i}$$

	P1 secs	P2 secs	Relative to A P1	Relative to A P2	Relative to A GM	Relative to B P1	Relative to B P2	Relative to B GM
A	10	50	1	1	1	0.1	50	2.2
B	100	1	10	0.02	0.45	1	1	1

Some unfortunate properties:

- does not track execution time!
- all improvements are equal

Harmonic Mean

Used to average rates like MIPS or MFLOPS.

$$\text{Harmonic Mean} = \frac{n}{\sum_{i=1}^{n}(1/R_i)}$$

Another way to look at it:

$$\frac{1}{\text{Harmonic Mean}} = \frac{\sum_{i=1}^{n}(1/R_i)}{n}$$

i.e. convert rate to time/task, average, convert back to rate. Also can be weighted like arithmetic mean.

Performance Estimation

Time is the measure of performance!

- Recall how we derived the CPU performance equation.

Typically, tradeoff between performance and cost.

- cost: parts, labor for assembly, etc.

New metric: power.

- portable devices
- want to extend battery life

Power

Where does power dissipation come from?

- Underlying implementation: transistor networks
- We can approximate these as RC circuits
- Power dissipated: CV^2, C capacitance, V voltage. (Resistors are dissipative)
- Power dissipated when the voltage on a wire changes.

(Quick recap: $I = \frac{V}{R}e^{-t/RC}$, $E = CV^2$)

$\Rightarrow$ time $\propto 1/V$, energy $\propto V^2$

$\Rightarrow$ power $\propto V^3$

Power

Total power:

- Alpha 21164: 50W
- Alpha 21264: 72W
- 550 MHz Pentium III: 30.8W
- 300 MHz Mobile Pentium II: 9W
- 160 MHz StrongARM: 0.5W

Sources of power consumption:

- Inputs to combinational logic, flip-flops
- Signals within combinational logic
- Clock keeps changing!

Power

Minimize power consumption:

- Avoid "spurious" changes of signal values
- Turn off clock when part of the circuit not in use
  clock gating
- "Modes" of operation: turn off large parts of the chip
- Slow down clock in idle mode
- Reduce voltage in idle mode
- Only activate part of circuit used for computation?
- Eliminate clocks?