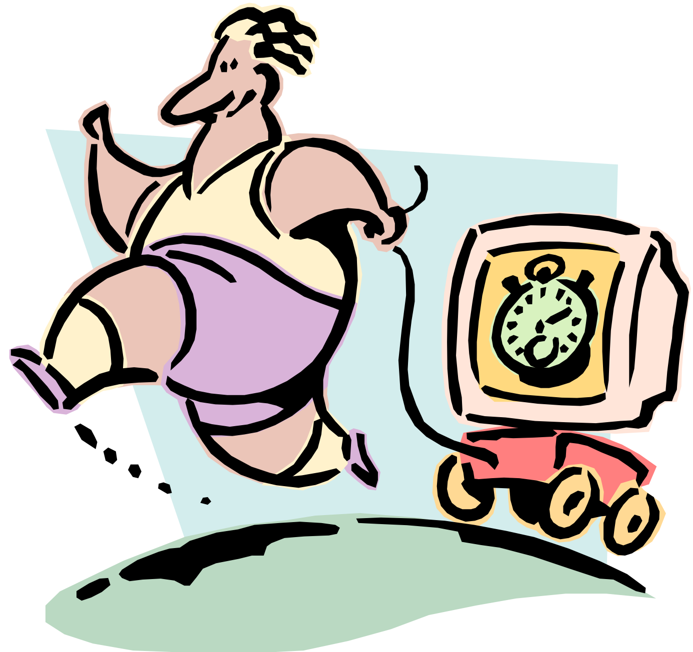


Measuring and Discussing Computer System Performance

or

"My computer is faster than your computer"

The bottom line: Performance

Car	Time to Bay Area	Speed	Passengers	Throughput (pmph)
	3.1 hours	160 mph	2	320
	7.7 hours	65 mph	60	3900

- **Time to do the task**
  – *execution time*, response time, latency
- **Tasks per day, hour, week, sec, ns. ..**
  – *throughput*, bandwidth

Measures of "Performance"

- Execution Time
- Frame Rate
- Throughput (operations/time)
  - Transactions/sec, queries/day, etc.
- Responsiveness
- Performance / Cost
- Performance / Power
- Performance / Energy

How to measure Execution Time?

```
% time program
... program results ...
90.7u 12.9s 2:39 65%
%
```

- Wall-clock time?
- user CPU time?
- user + kernel CPU time?
- Answer:

Our definition of Performance

$$\text{Performance}_X \sim \frac{1}{\text{Execution Time}_X} \quad \text{, for program X}$$

- only has meaning in the context of a program or workload
- Not very intuitive as an absolute measure, but most of the time we're more interested in relative performance.

Relative Performance

- can be confusing

  A runs in 12 seconds

  B runs in 20 seconds

  – A/B = .6 , so A is 40% faster, or 1.4X faster, or B is 40% slower
  – B/A = 1.67, so A is 67% faster, or 1.67X faster, or B is 67% slower

- needs a precise definition

Relative Performance (Speedup), the Definition

$$\text{Speedup (X/Y)} = \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

Example

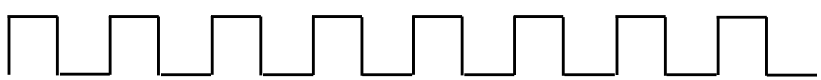
- Machine A runs program C in 9 seconds, Machine B runs the same program in 6 seconds. What is the speedup we see if we move to Machine B from Machine A?

- Machine B gets a new compiler, and can now run the program in 3 seconds. Speedup from the new compiler?

What is Time?

CPU Execution Time = CPU clock cycles * Clock cycle time

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



What is Time?

CPU Execution Time = CPU clock cycles * Clock cycle time

- 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

Cycle Time

GHz = billions of cycles/second, MHz = millions of cycles/second

Y GHz = 1/Y nanoseconds cycle time

How many clock cycles?

Number of CPU clock cycles = Instruction count * Average Clock Cycles per Instruction (CPI)

Computer A runs program C in 3.6 billion cycles. Program C requires 2 billion dynamic instructions. What is the CPI?

How many clock cycles?

Number of CPU clock cycles = Instruction count * Average Clock Cycles per Instruction (CPI)

A computer is running a program with CPI = 2.0, and executes 24 million instructions, how long will it run?

All Together Now

CPU Execution Time = CPU clock cycles * Clock cycle time

CPU clock cycles = Instruction count * Average Clock Cycles per Instruction (CPI)

All Together Now

seconds

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

instructions

cycles/instruction

seconds/cycle

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

- IC = 4 billion, 2 GHz processor, execution time of 3 seconds. What is the CPI for this program?

- Suppose we reduce CPI to 1.2 (through an architectural improvement). What is the new execution time?

An aside…
Cycle Time/Clock Rate

- Increasingly, modern processors can execute at multiple clock rates (cycle times).
- Why?
- However, the granularity at which we can change the cycle time tends to be fairly coarse, so all of these principles and formulas still apply.

Who Affects Performance?

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

- programmer
- compiler
- instruction-set architect
- machine architect
- hardware designer
- materials scientist/physicist/silicon engineer

Performance Variation

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

	Number of instructions	**CPI**	**Clock Cycle Time**
Same machine different programs			
same programs, different machines, same ISA			
Same programs, different machines			

Other Performance Metrics

- MIPS
- MFLOPS, GigaFLOPS, PetaFLOPS (million, billion, quadrillion Floating Point Operations/second)

MIPS

MIPS = Millions of Instructions Per Second

$$= \frac{\text{Instruction Count}}{\text{Execution Time} * 10^6}$$

$$= \frac{\text{Clock rate}}{\text{CPI} * 10^6}$$

- program-independent
- deceptive

Which Programs?

- peak throughput measures (simple programs)?
- synthetic benchmarks (whetstone, dhrystone,...)?
- Real applications

Which Programs?

- peak throughput measures (simple programs)?
- synthetic benchmarks (whetstone, dhrystone,...)?
- Real applications
- SPEC (best of both worlds, but with problems of their own)
  - *System Performance Evaluation Cooperative*
  - Provides a common set of real applications along with strict guidelines for how to run them.
  - provides a relatively unbiased means to compare machines.
- Now, we also have more specialized benchmarks for big data, for datacenter applications, for machine learning, …

Amdahl's Law

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

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

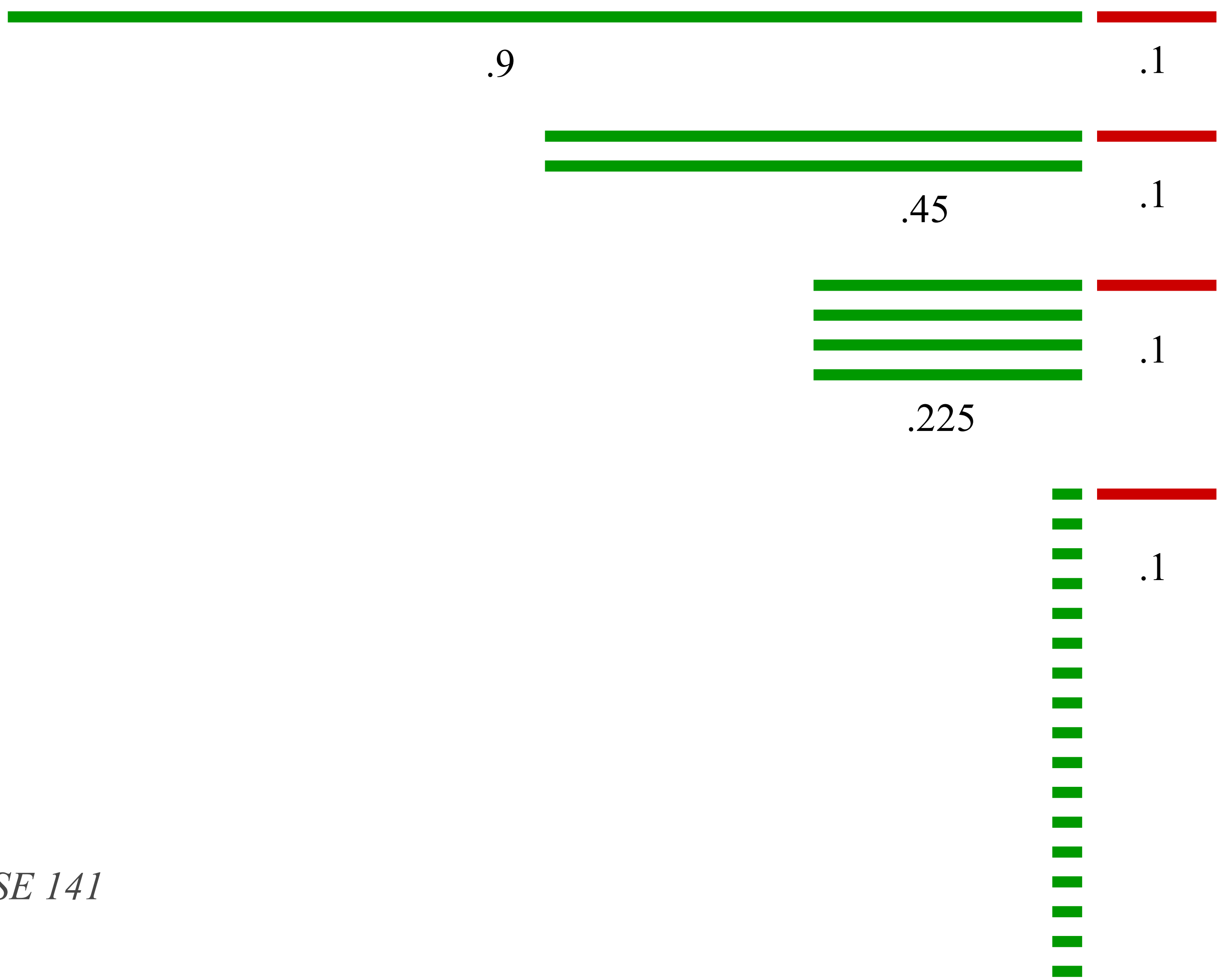
Amdahl's Law

- The impact of a performance improvement is limited by the fraction 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!!   (usually)

Amdahl's Law and Massive Parallelism



Speedup

1.0

1/.55 = **1.82**

1/.325 = **3.07**

< 10

Key Points

- Be careful how you specify performance
- Execution time = instructions * CPI * cycle time
- Use real applications
- Use standards, if possible
- Make the common case fast