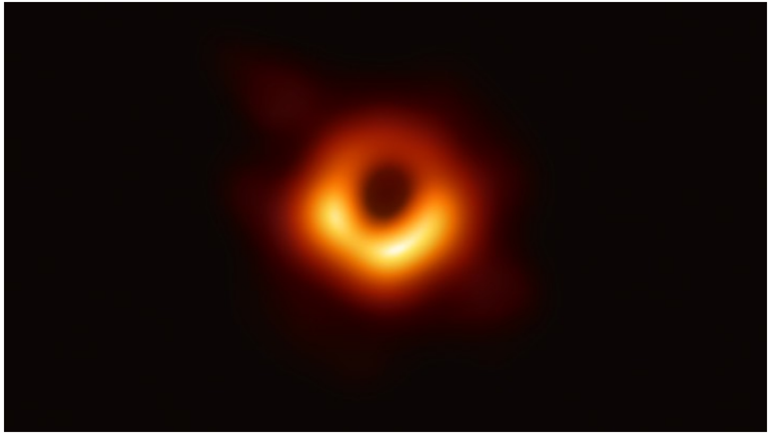
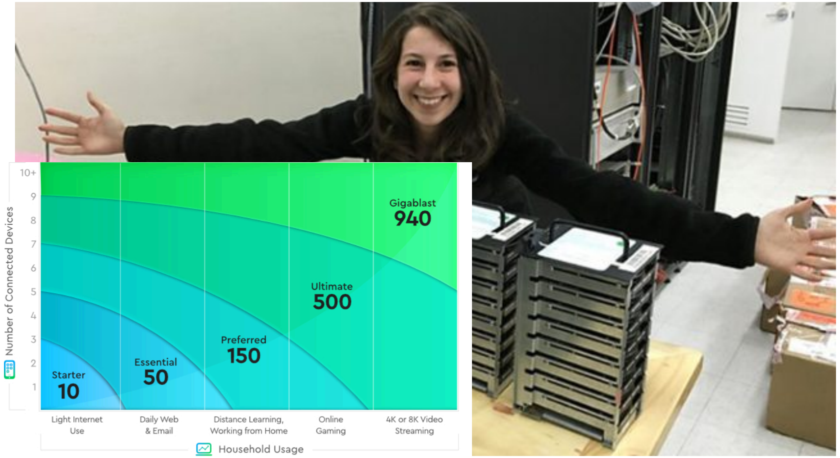
CSE 141: Introduction to Computer Architecture

Performance

Thought Experiment

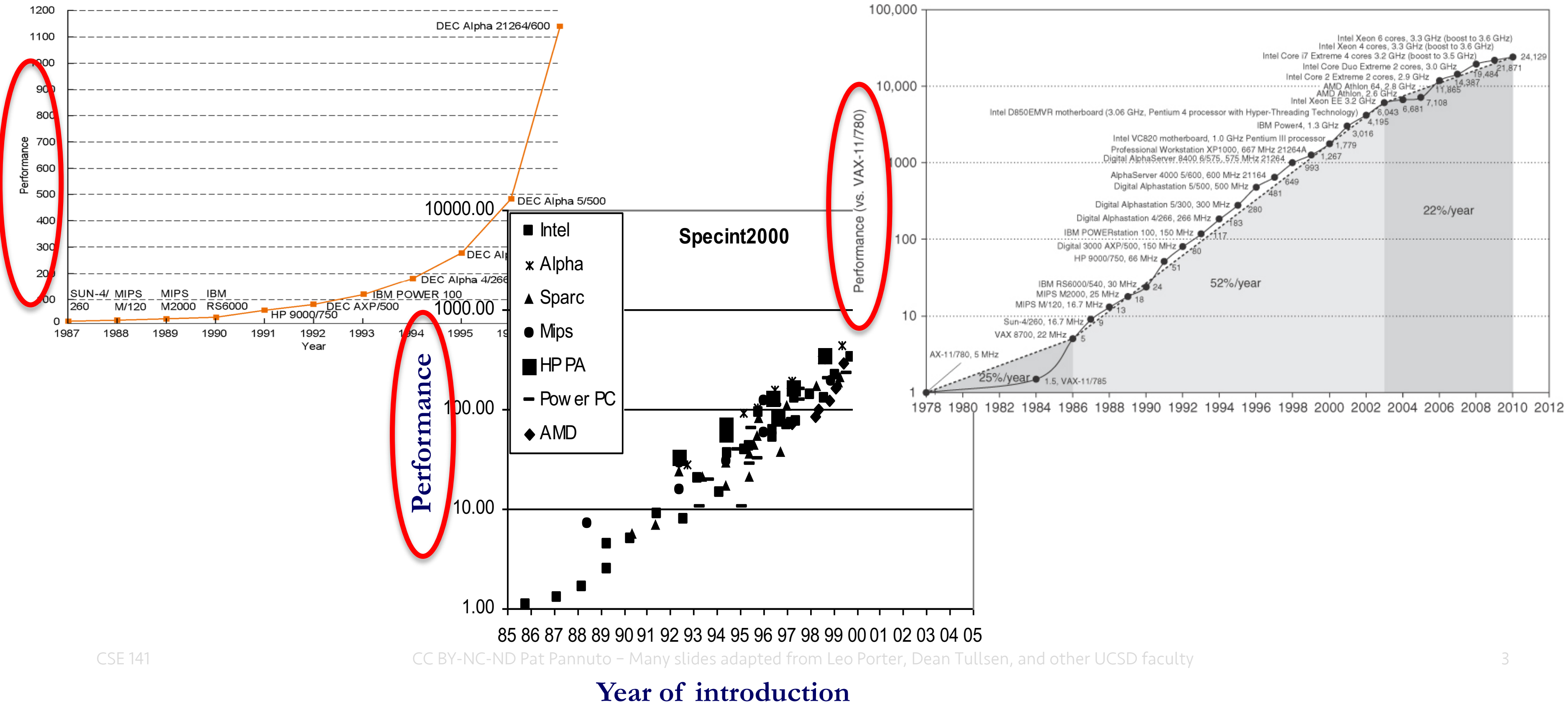
- What is the fastest way to send a picture of a black hole to Boston?
- What is the fastest way to send **5 petabytes** of data to Boston?





(5 petabytes) / (940 Megabits/second) = 1.35 years

Graphs that go up and to the right are good, but what do they mean?



CC BY-NC-ND Pat Pannuto – Many slides adapted from Leo Porter, Dean Tullsen, and other UCSD faculty

The bottom line: Performance

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

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

Measures of "Performance"

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

There are many ways to measure program execution time

```
$ time make  # cargo build
 Compiling hail v0.1.0 (/tock/boards/hail)
    Finished release [optimized + debuginfo] target(s) in 19.96s

real 0m21.146s
user 0m30.388s
sys  0m2.032s
```

- Program-reported time?
- Wall-clock time?
- user CPU time?
- user + kernel CPU time?

Our definition of Performance

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

- 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?

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

Poll Question: What is the speedup?

- Machine A runs program C in 9 seconds.
- Machine B runs the same program in 6 seconds.
- Machine B gets a new compiler, and can now run the program in 3 seconds.
- What is the speedup from the new compiler?

When you have your answer, ***write it down***

Now, ***convince your neighbors of your answer***

A: 0.5 B: 3 C:1.5 D: 0.33 E: None of these

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 (whole number) of clock cycles

Cycle Time

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

X MHz = 1000/X nanoseconds cycle time

Y GHz = 1/Y nanoseconds cycle time

How many clock cycles?

Number of CPU clock cycles =

[Instruction count] * [Average Clock Cycles per Instruction (`CPI`)]

Exercise:

Computer A runs program C in 3.6 billion cycles.

Program C requires 2 billion dynamic instructions.

What is the CPI?

Poll Question: How many clock cycles?

A computer is running a program with CPI = 2.0.

It executes 24 million instructions.

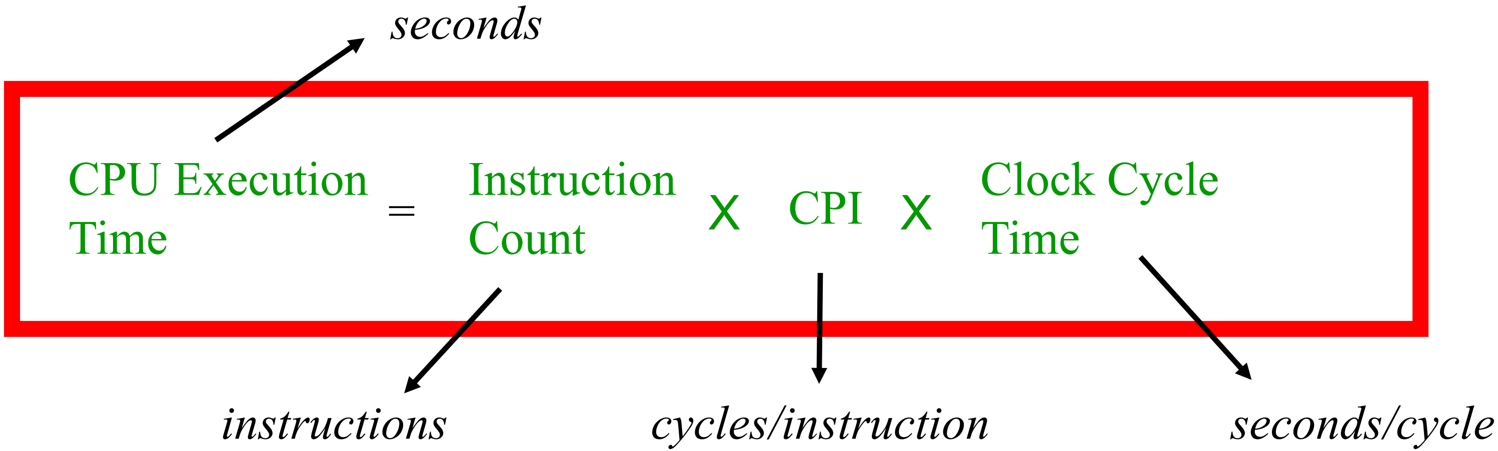
How long will it run?

Selection	Answer
A	2.4 seconds
B	12 million cycles
C	48 million seconds
D	48 million cycles
E	None of the above

Putting it all together

CPU Execution Time = [CPU clock cycles] * [Clock cycle time]

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



seconds

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

instructions

cycles/instruction

seconds/cycle

Poll Question: All Together Now

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

- Instruction Count = 4 billion
- 2 GHz processor
- Execution time of 3 seconds

What is the CPI for this program?

When you have your answer, ***write it down***

A: 0.375

B: 0.67

C: 0.375 * 10^{-18}

D: 1.5

E: None of these

Cycle Time/Clock Rate is no longer fixed

- 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? How?

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: What affects what?

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

	Number of Instructions	CPI	Clock Cycle Time
Same machine, different programs			
Sam programs, different machine, same ISA			
Same programs, different machines			

MIPS

(the performance measure, not the architecture...)

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!

Some also discuss [M]FLOPS
“Floating point operations per second”

Which programs are best, are “most fair”, to run when measuring performance?

- 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.

Amdahl’s Law

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

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

- Make the common case fast!!

Amdahl's Law and Massive Parallelism

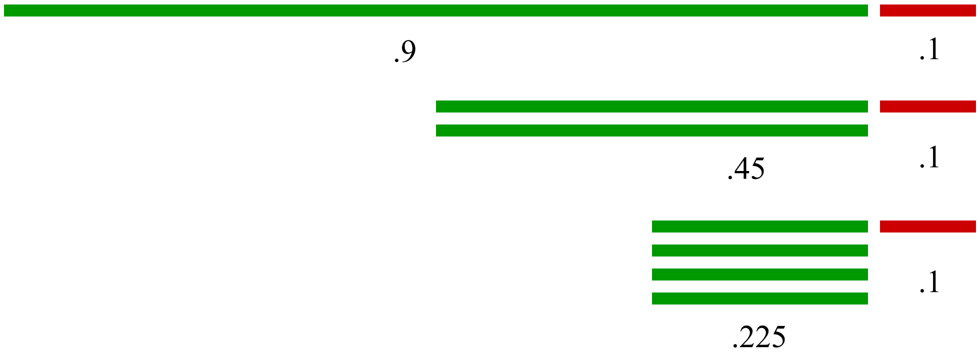
Amdahl’s Law and Massive Parallelism

Speedup

1.0

1/.55 = **1.82**

Amdahl’s Law and Massive Parallelism



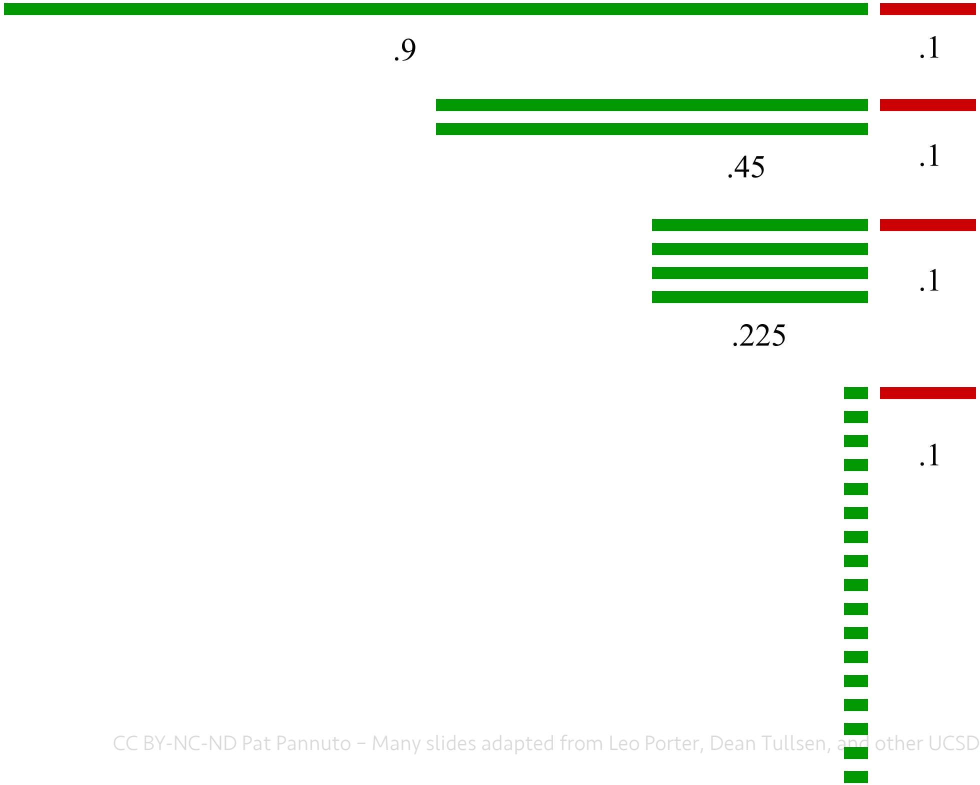
Speedup

1.0

1/.55 = **1.82**

1/.325 = **3.07**

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