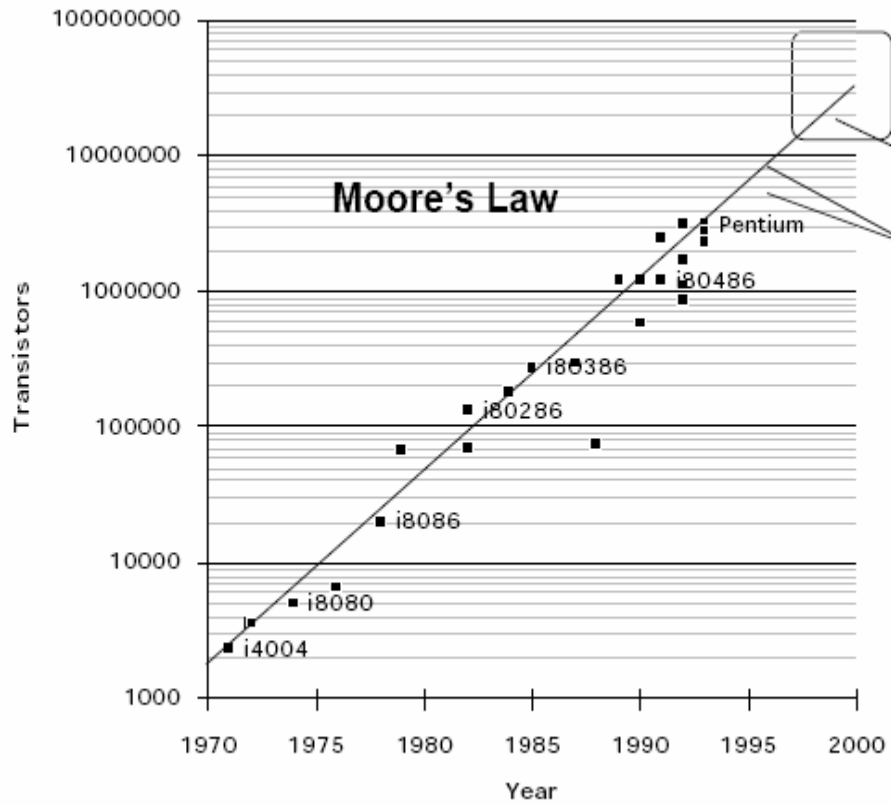


Measuring and Discussing Computer System Performance

or

“My computer is faster than your computer”

Trends



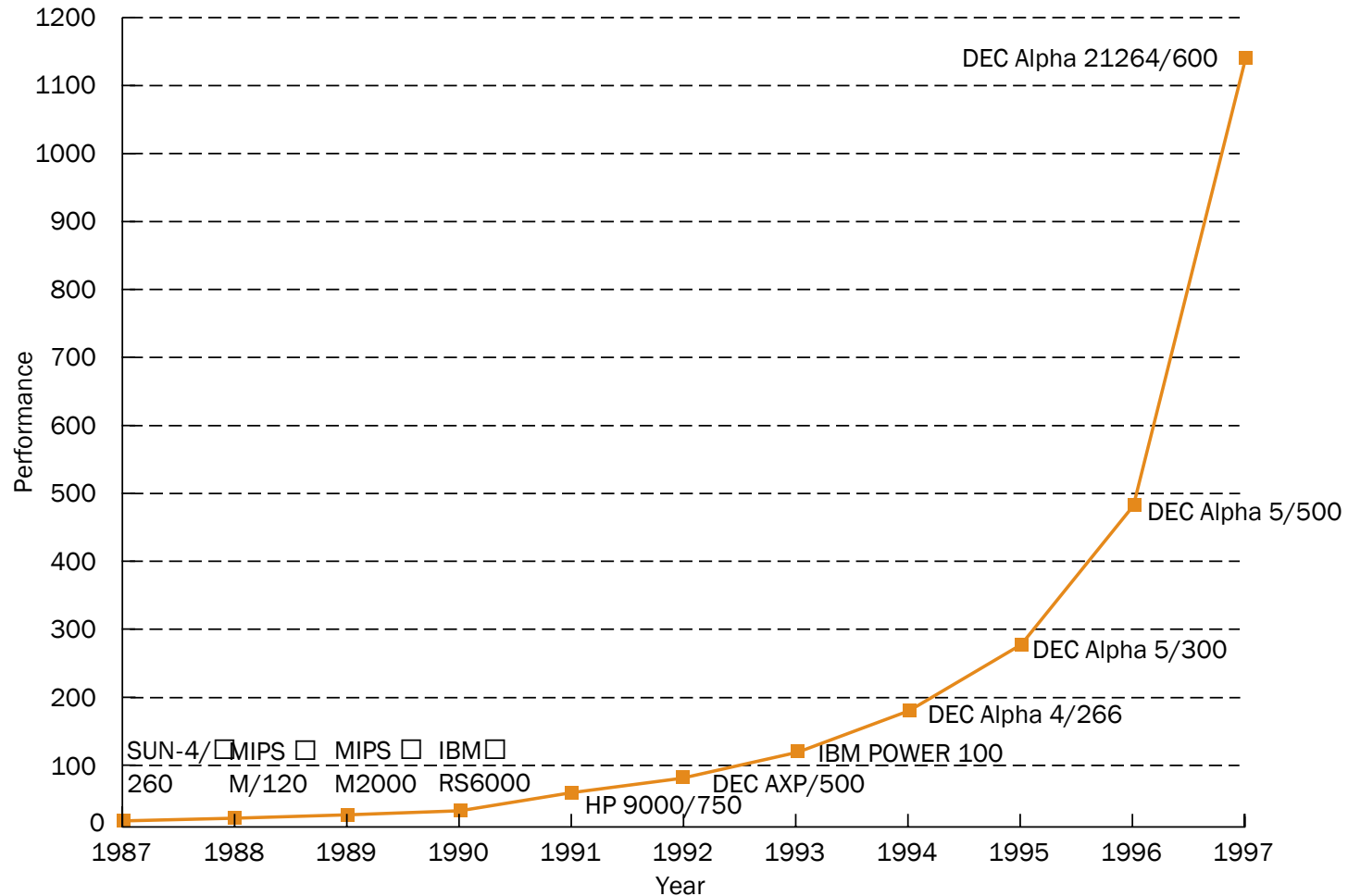
“Graduation Window”

- Alpha 21264: 15 million
- Pentium Pro: 5.5 million
- PowerPC 620: 6.9 million
- Alpha 21164: 9.3 million
- Sparc Ultra: 5.2 million

- CMOS improvements:**
- Die size: 2X every 3 y
 - Line width: halve / 7 y

As of 1992

Performance Marches On ...



- *But what is performance?*

Speed, Cost, Capacity

Component\Year	1991	1993	1995	1997	1999	2001
Microprocessor						
Speed (MHz)	25	33	100	125	350	1000
Cost	180	125	275	250	300	251
Price per unit capacity	7.2	3.787879	2.75	2	0.857143	0.251
RAM Chip						
Megabytes per Chip	1	4	4	16	64	256
Cost	55	140	120	97	125	90
Price per unit capacity	55	35	30	6.0625	1.953125	0.351563
Hard Disk Device						
Megabytes per Disk	105	250	540	2000	8000	40000
Cost	480	375	220	250	220	138
Price per unit capacity	4.571429	1.5	0.407407	0.125	0.0275	0.00345

Plane	DC to Paris time	Speed	Passengers	Throughput (pmp)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

° **Time to do the task**

– *execution time*, response time, latency

° **Tasks per day, hour, week, sec, ns. ..**

– *throughput*, bandwidth

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

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

- only has meaning in the context of a program or workload
- Not very intuitive as an absolute measure

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, the Definition

$$\text{Relative Performance} = \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

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

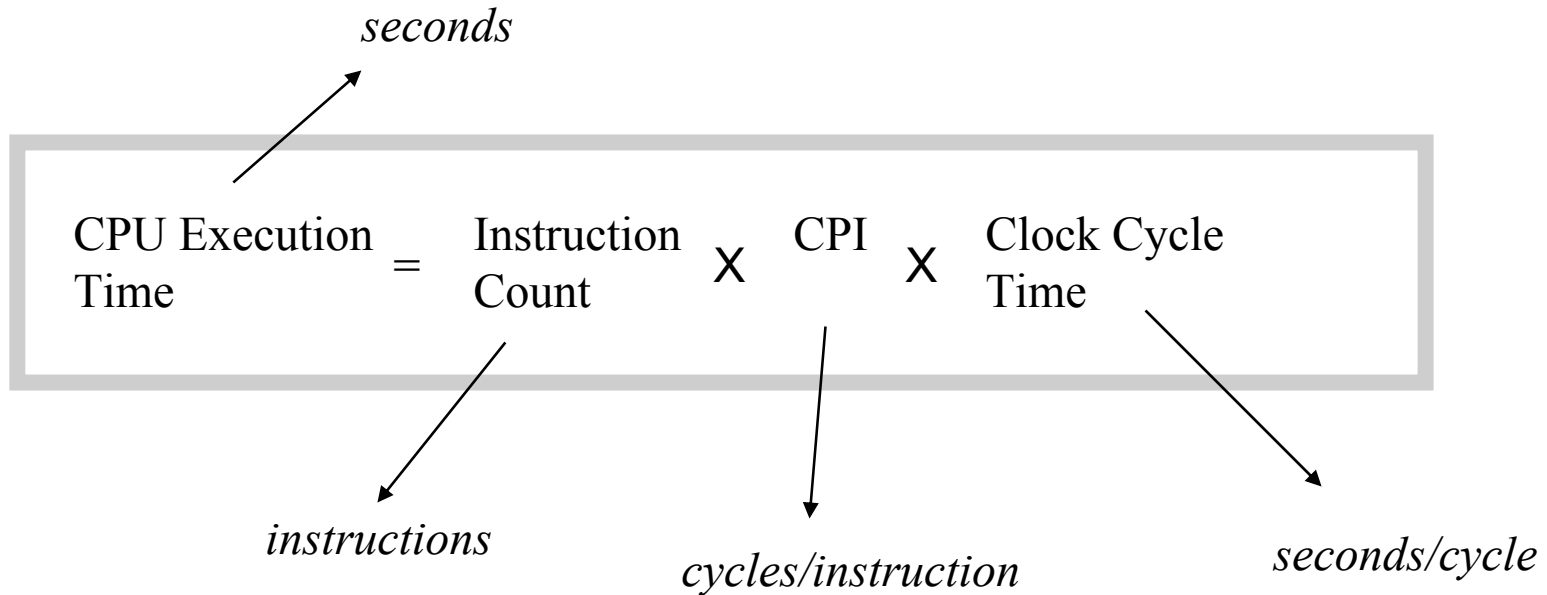
MHz = millions of cycles/second

GHz = billions of cycles/second

How many clock cycles?

Number of CPU cycles = Instructions executed * Average
Clock Cycles per Instruction (CPI)

All Together Now



Who Affects Performance?

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

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

Performance Variation

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

	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
- GFLOPS

MIPS

MIPS = Millions of Instructions Per Second

= Instruction Count

Execution Time * 10^6

= Clock rate

CPI * 10^6

- program-independent
- deceptive

Which Programs?

- peak throughput measures (simple programs)
- synthetic benchmarks (whetstone, dhrystone, NAS Parallel...)
- 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.



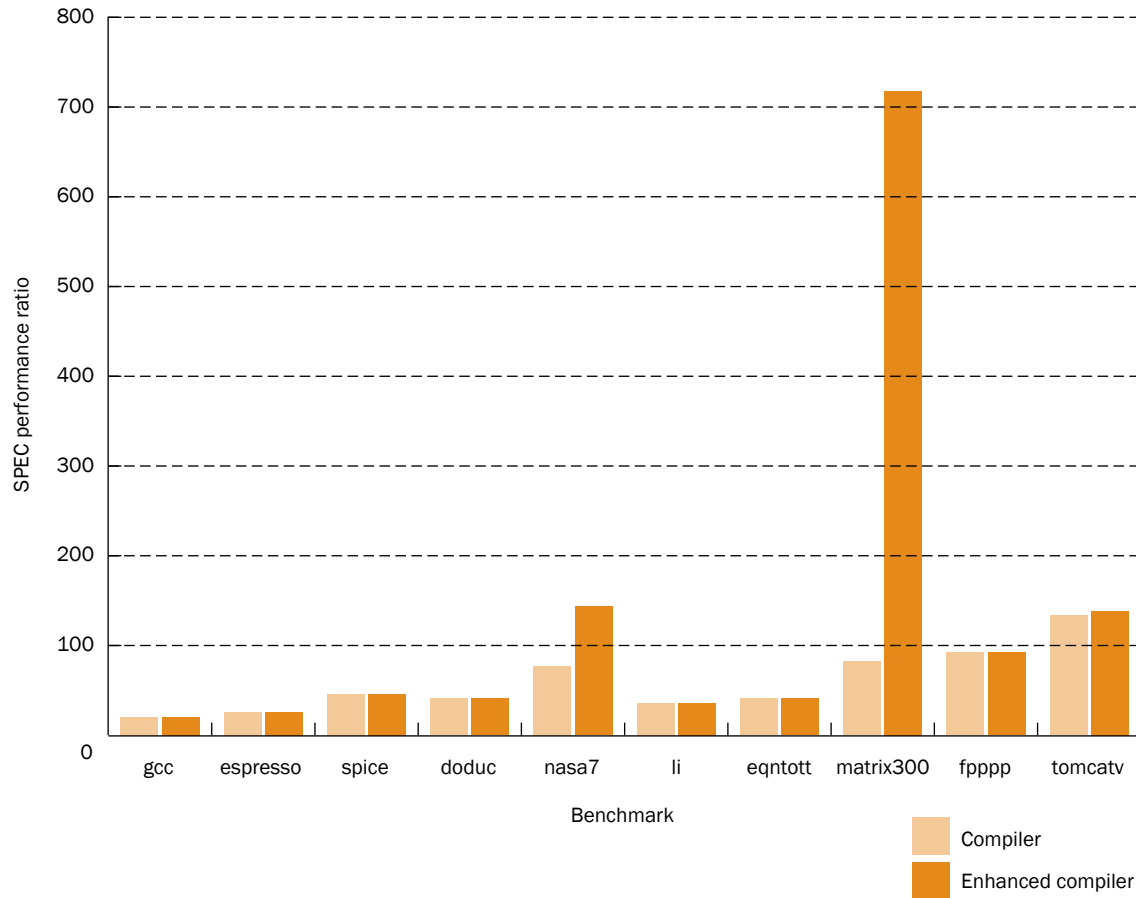
<http://www.top500.org/>

Ranking of worlds most powerful supercomputers based on
Linpack – a benchmark that correlates very closely to peak

PMaC <http://www.sdsc.edu/pmac/pmac.html>

PMaC
Performance Modeling and Characterization

Danger in Benchmark-Specific Performance Measures



- measures compiler as much as architecture!

Amdahl's Law

- The impact of a performance improvement is limited by the percent 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!!

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