

Computer Architecture

Lecture 6&7: Understanding Performance



Nguyen Minh Son, Ph.D



Performance

- ❑ Measure, Report, and Summarize
- ❑ Make intelligent choices
- ❑ See through the marketing hype
- ❑ Key to understanding underlying organizational motivation

Why is some hardware better than others for different programs?

*What factors of system performance are hardware related?
(e.g., Do we need a new machine, or a new operating system?)*

How does the machine's instruction set affect performance?

Which of these airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

- How much faster is the Concorde compared to the 747?
- How much bigger is the 747 than the Douglas DC-8?

Computer Performance: TIME, TIME, TIME

□ Response Time (latency)

- Time between start and end of an event
 - How long does it take for my job to run?
 - How long does it take to execute a job?
 - How long must I wait for the database query?

□ Throughput

- Total amount of work (or number of jobs) done
 - How many jobs can the machine run at once?
 - What is the average execution rate?
 - How much work is getting done?

Computer Performance: TIME, TIME, TIME

- *If we upgrade a machine with a new processor what do we improve?*
 - *Response time*

- *If we add a new machine to the lab what do we improve?*
 - *Throughput*

Book's Definition of Performance

- For some program running on machine X,

$$\text{Performance}_X = 1 / \text{Execution time}_X$$

- "X is *N times* faster than Y" means the Speedup *N* is:

$$N = \frac{\text{Performance}(X)}{\text{Performance}(Y)} = \frac{\text{Execution_time}(Y)}{\text{Execution_time}(X)}$$

- Problem:

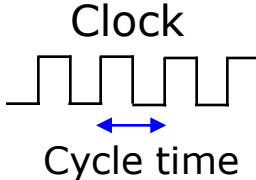
- machine A runs a program in 20 seconds
- machine B runs the same program in 25 seconds

Execution Time

- There are different measures of execution time in computer performance.
 - **Elapsed Time**
 - counts everything (*disk and memory accesses, I/O , etc.*)
 - a useful number, but often not good for comparison purposes
 - **CPU time**
 - doesn't count I/O or time spent running other programs
 - can be broken up into system time, and user time
 - **Our focus: user CPU time**
 - time spent executing the lines of code that are "in" our program
-

Clock Cycles

- Instead of reporting execution time in seconds, we often use **clock cycles** (basic time unit in machine).

$$\text{Execution time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$


- **Cycle time** (or cycle period or clock period) T_{cycle} = time between two consecutive rising edges, measured in seconds.
- **Clock rate** (or clock frequency) $f_{\text{clock}} = 1/\text{cycle-time} =$ number of cycles per second (1 Hz = 1 cycle/second).
 - Example: A 200 MHz clock has cycle time of $1/(200 \times 10^6) = 5 \times 10^{-9}$ seconds = 5 nanoseconds.

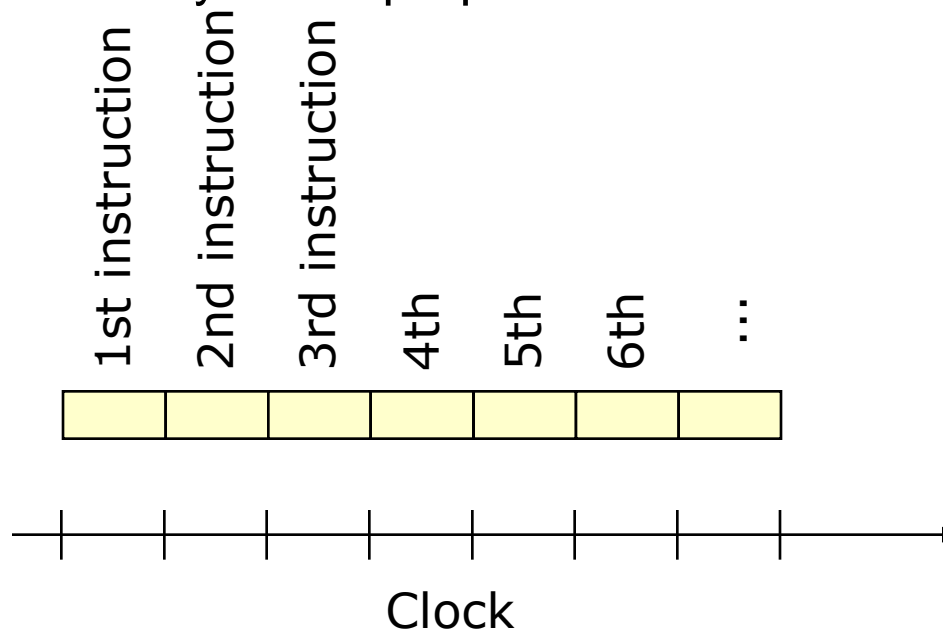
How to Improve Performance

$$\text{Execution time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

- Therefore, to improve performance (everything else being equal), you can do the following:
 - ↓ Reduce the number of cycles for a program, or
 - ↓ Reduce the clock cycle time, or said in another way,
 - ↑ Increase the clock rate.

Instruction Cycles

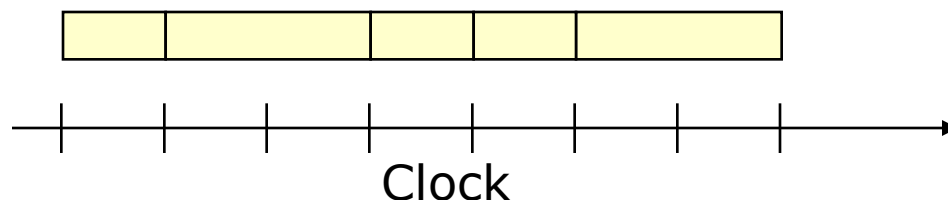
- Can we assume that
 - The number of cycles = number of instructions?
 - The number of cycles is proportional to number of instructions?



- No, the assumptions are incorrect.

Instruction Cycles

- Different instructions take different amount of time to finish.



- For example:
 - Multiply instruction may take more cycles than an Add instruction.
 - Floating-point operations take longer than integer operations.
 - Accessing memory takes more time than accessing registers.

Example 1

- Our favorite program runs in 10 seconds on computer A, which has a 400 MHz clock. We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program. What clock rate should we tell the designer to target at?

■ ANSWER:

Let C be the number of clock cycles required for that program.

For A: Ex. time = 10 sec. = $C \times 1/400\text{MHz}$

For B: Ex. time = 6 sec. = $(1.2 \times C) \times 1/\text{clock_rateB}$

Therefore, $\text{clock_rateB} = ?$

$$= 1.2 \times 400 \times 10 / 6 = 800 \text{ (Mhz)}$$

Cycles Per Instruction

- A given program will require

Some number of instructions (machine instructions)



× **average CPI**

Some number of cycles



× **cycle time**

Some number of seconds

- Recall that different instructions have different number of cycles.

Cycles Per Instruction

□ Average cycle per instruction (CPI)

$$\begin{aligned} \text{CPI} &= (\text{CPU time} \times \text{Clock rate}) / \text{Instruction count} \\ &= \text{Clock cycles} / \text{Instruction count} \end{aligned}$$

CPU time	=	$\frac{\text{Seconds}}{\text{Program}}$	=	$\frac{\text{Instructions}}{\text{Program}}$	x	$\frac{\text{Cycles}}{\text{Instruction}}$	x	$\frac{\text{Seconds}}{\text{Cycle}}$
----------	---	---	---	--	---	--	---	---------------------------------------

$$\text{CPI} = \sum_{k=1}^n \text{CPI}_k \times F_k \quad \textbf{where} \quad F_k = \frac{I_k}{\text{Instruction count}}$$

I_k = instruction frequency

- Invest resources where time is spent!

Example 2

- A compiler designer is deciding between 2 codes for a particular machine. Based on the hardware implementation, there are 3 classes of instructions: Class A, Class B, and Class C, and they require 1, 2, and 3 cycles respectively.
- First code has 5 instructions: 2 of A, 1 of B, and 2 of C.
Second code has 6 instructions: 4 of A, 1 of B, and 1 of C.
- Which code is faster? By how much?
- What is the (average) CPI for each code?

■ ANSWER:

Let T be the cycle time.

$$\text{Execution time}(\text{code1}) = (2 \times 1 + 1 \times 2 + 2 \times 3) \times T = 10T$$

$$\text{Execution time}(\text{code2}) = (4 \times 1 + 1 \times 2 + 1 \times 3) \times T = 9T$$

$$\text{Execution time}(\text{code1}) / \text{Execution time}(\text{code2}) = 10/9$$

$$\text{CPI}(\text{code1}) = 10/5 = 2$$

$$\text{CPI}(\text{code2}) = 9/6 = 1.5$$

Example 3

- Suppose we have 2 implementations of the same ISA, and a program is run on these 2 machines.
- Machine A has a clock cycle time of 10 ns and a CPI of 2.0.
Machine B has a clock cycle time of 20 ns and a CPI of 1.2.
- Which machine is faster for this program? By how much?

- ANSWER:

Let N be the number of instructions.

Machine A: Execution time = $N \times 2.0 \times 10 \text{ ns}$

Machine B: Execution time = $N \times 1.2 \times 20 \text{ ns}$

$$\begin{aligned} \text{Performance(A)/Performance(B)} &= \text{Execution time(B)/} \\ &\quad \text{Execution time(A)} \\ &= 1.2 \times 20 / 2.0 \times 10 = 1.2 \end{aligned}$$

Example 4

- You are given 2 machine designs M1 and M2 for performance benchmarking. Both M1 and M2 have the same ISA, but different hardware implementations and compilers. Assuming that the clock cycle times for M1 and M2 are the same, performance study gives the following measurements for the 2 designs.

Instruction class	For M1		For M2	
	CPI	No. of instructions executed	CPI	No. of instructions executed
A	1	3,000,000,000,000	2	2,700,000,000,000
B	2	2,000,000,000,000	3	1,800,000,000,000
C	3	2,000,000,000,000	3	1,800,000,000,000
D	4	1,000,000,000,000	2	900,000,000,000

Example 4

a) What is the CPI for each machine?

Let $Y = 1,000,000,000,000$

$$\begin{aligned} \text{CPI}(M1) &= (3Y*1 + 2Y*2 + 2Y*3 + Y*4) / (3Y + 2Y + 2Y + Y) \\ &= 17Y / 8Y = \mathbf{2.125} \end{aligned}$$

$$\begin{aligned} \text{CPI}(M2) &= (2.7Y*2 + 1.8Y*3 + 1.8Y*3 + 0.9Y*2) / \\ &\quad (2.7Y+1.8Y+1.8Y+0.9Y) \\ &= 18Y / 7.2Y = \mathbf{2.5} \end{aligned}$$

b) Which machine is faster? By how much?

Let T be clock cycle.

$$\text{Execution time}(M1) = 2.125 \times (8Y \times T)$$

$$\text{Execution time}(M2) = 2.5 \times (7.2Y \times T)$$

M1 is faster than M2 by 5.8%

Example 4

- c) To further improve the performance of the machines, a new compiler technique is introduced. The compiler can simply eliminate all class D instructions from the benchmark program without any side effects. (That is, there is no change to the number of class A, B and C instructions executed in the 2 machines.) With this new technique, which machine is faster? By how much?

Let $Y = 1,000,000,000,000$; Let C be clock cycle.

$$\begin{aligned} \text{CPI}(M1) &= (3Y*1 + 2Y*2 + 2Y*3) / (3Y + 2Y + 2Y) = 13Y / 7Y \\ &= \mathbf{1.857} \end{aligned}$$

$$\begin{aligned} \text{CPI}(M2) &= (2.7Y*2 + 1.8Y*3 + 1.8Y*3) / (2.7Y+1.8Y+1.8Y) \\ &= 16.2Y / 6.3Y = \mathbf{2.57} \end{aligned}$$

$$\text{Execution time}(M1) = 1.857 \times (7Y \times C)$$

$$\text{Execution time}(M2) = 2.57 \times (6.3Y \times C)$$

M1 is faster than M2 by 24.6%

Example 4

- d) Alternatively, to further improve the performance of the machines, a new hardware technique is introduced. The hardware can simply execute all class D instructions in zero times without any side effects. (There is still execution for class D instructions.) With this new technique, which machine is faster? By how much?

Let $Y = 1,000,000,000,000$; Let C be clock cycle.

$$\begin{aligned} \text{CPI}(M1) &= (3Y*1 + 2Y*2 + 2Y*3 + Y*0) / (3Y + 2Y + 2Y + Y) \\ &= 13Y / 8Y = \mathbf{1.625} \end{aligned}$$

$$\begin{aligned} \text{CPI}(M2) &= (2.7Y*2 + 1.8Y*3 + 1.8Y*3 + 0.9Y*0) / \\ &(2.7Y+1.8Y+1.8Y+0.9Y) = 16.2Y / 7.2Y = \mathbf{2.25} \end{aligned}$$

$$\text{Execution time}(M1) = 1.625 \times (8Y \times C)$$

$$\text{Execution time}(M2) = 2.25 \times (7.2Y \times C)$$

M1 is faster than M2 by 24.6%

Aspects of CPU Performance

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	instr count	CPI	cycle time
Program,	X		
Algorithms,	X		
Data structure	X		
Compiler	X	X	
Instr. Set	X	X	X
Organization		X	X
Circuit Design, VLSI		X	X
Technology			X

Now that we understand cycles

- A given program will require
 - some number of instructions (machine instructions)
 - some number of cycles
 - some number of seconds
 - We have a vocabulary that relates these quantities:
 - cycle time (seconds per cycle)
 - clock rate (cycles per second)
 - CPI (cycles per instruction)
 - a floating point intensive application might have a higher CPI*
 - MIPS (millions of instructions per second) = $\frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$
 - this would be higher for a program using simple instructions*
-

Performance

- Performance is determined by execution time
- Do any of the other variables equal performance?
 - # of cycles to execute program?
 - # of instructions in program?
 - # of cycles per second?
 - average # of cycles per instruction?
 - average # of instructions per second?
- Common pitfall: thinking one of the variables is indicative of performance when it really isn't.

MIPS (million instructions per second) example

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.

The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

The second compiler's code uses 10 million Class A instructions, 1million Class B instructions, and 1 million Class C instructions.

- Which sequence will be faster according to execution time?
- Which sequence will be faster according to MIPS?

MIPS example

□ Answer:

- CPU clock cycles (Compiler1) = $(5*1 + 1*2 + 1*3) \times 10^6$
- CPU clock cycles (Compiler2) = $(10*1 + 1*2 + 1*3) \times 10^6$
- Ex. time (Compiler1) = CPU clock cycles 1 / clock rate
 $= 10 \times 10^6 / 100 \times 10^6 = 0.10 \text{ sec}$
- Ex. time (Compiler2) = CPU clock cycles 2 / clock rate
 $= 15 \times 10^6 / 100 \times 10^6 = 0.15 \text{ sec}$

$$\text{Performance(Compiler1)} / \text{Performance(Compiler2)} = 1.5$$

The 1st Compiler generates faster program -> run faster

- MIPS1 = $(5 + 1 + 1) \times 10^6 / 0.10 \times 10^6 = \mathbf{70}$
- MIPS2 = $(10 + 1 + 1) \times 10^6 / 0.15 \times 10^6 = \mathbf{80 > MIPS1}$

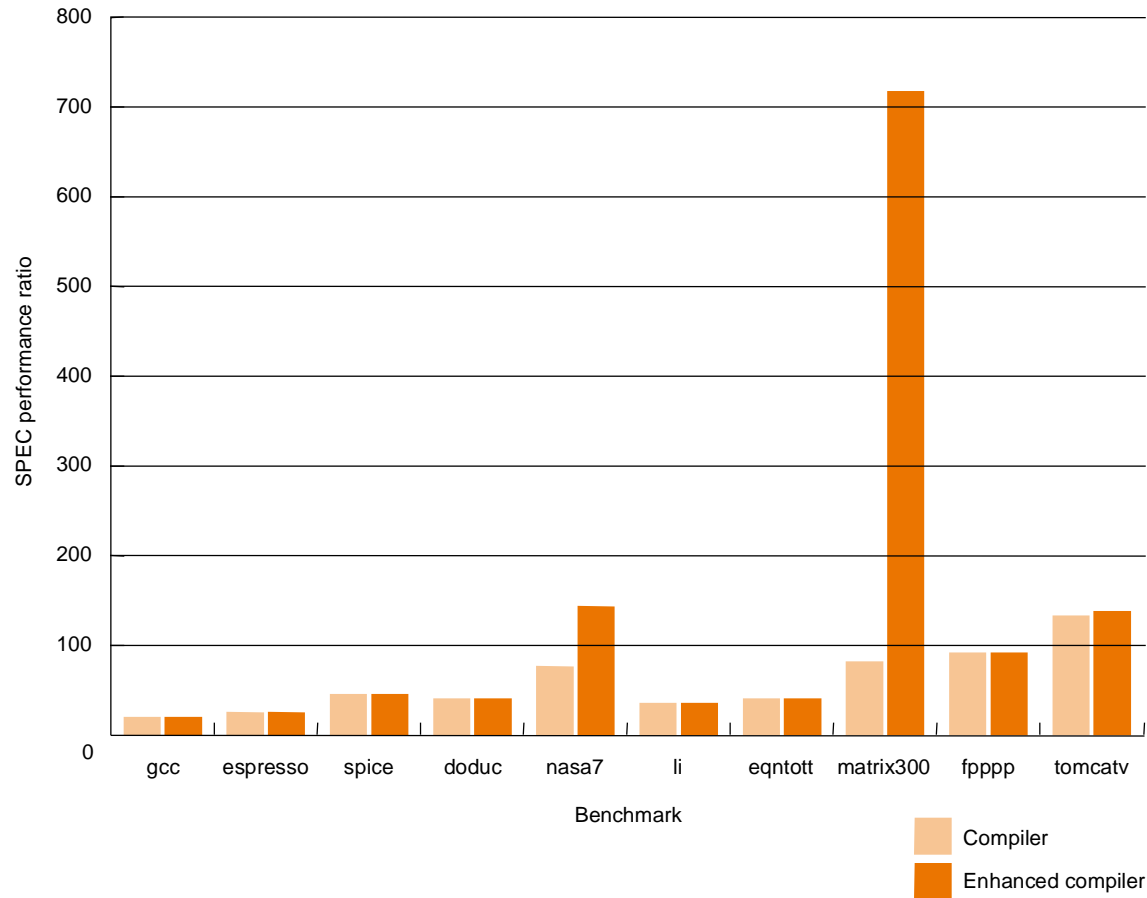
The 2nd Compiler has a higher MIPS rating ! (Pitfall)

Benchmarks

- Performance best determined by running a real application
 - Use programs typical of expected workload
 - Or, typical of expected class of applications
e.g., compilers/editors, scientific applications, graphics, etc.
 - Small benchmarks
 - nice for architects and designers
 - easy to standardize
 - can be abused
 - SPEC (System Performance Evaluation Cooperative)
 - companies have agreed on a set of real program and inputs
 - can still be abused (Intel's "other" bug)
 - valuable indicator of performance (and compiler technology)
-

SPEC '89

□ Compiler “enhancements” and performance



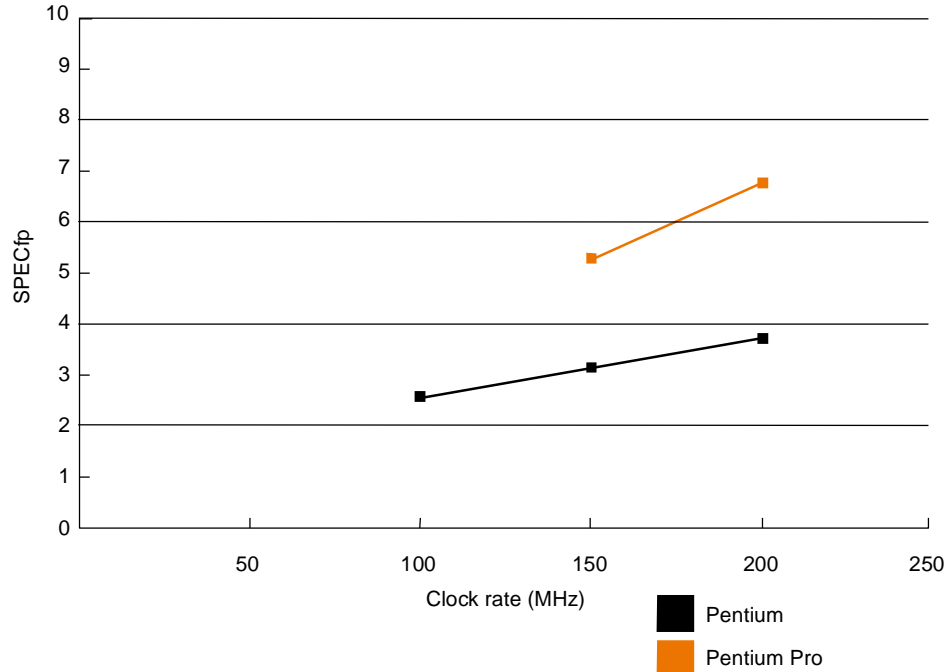
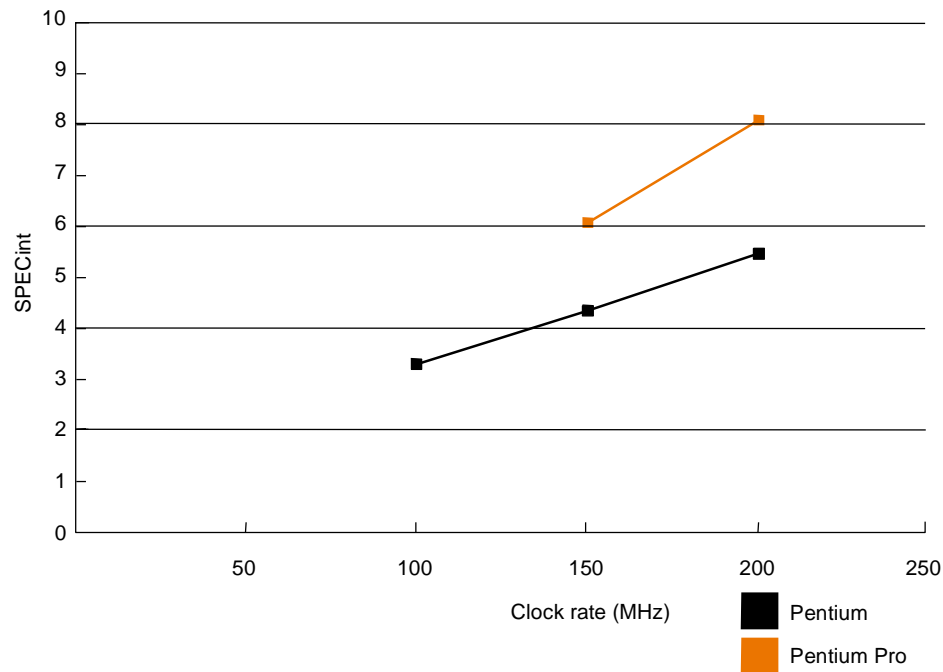
SPEC '95

Benchmark	Description
go	Artificial intelligence; plays the game of Go
m88ksim	Motorola 88k chip simulator; runs test program
gcc	The Gnu C compiler generating SPARC code
compress	Compresses and decompresses file in memory
li	Lisp interpreter
ljpeg	Graphic compression and decompression
perl	Manipulates strings and prime numbers in the special-purpose programming language Perl
vortex	A database program
tomcatv	A mesh generation program
swim	Shallow water model with 513 x 513 grid
su2cor	quantum physics; Monte Carlo simulation
hydro2d	Astrophysics; Hydrodynamic Naiver Stokes equations
mgrid	Multigrid solver in 3-D potential field
applu	Parabolic/elliptic partial differential equations
trub3d	Simulates isotropic, homogeneous turbulence in a cube
apsi	Solves problems regarding temperature, wind velocity, and distribution of pollutant
fpppp	Quantum chemistry
wave5	Plasma physics; electromagnetic particle simulation

SPEC '95

Does doubling the clock rate double the performance?

Can a machine with a slower clock rate have better performance?

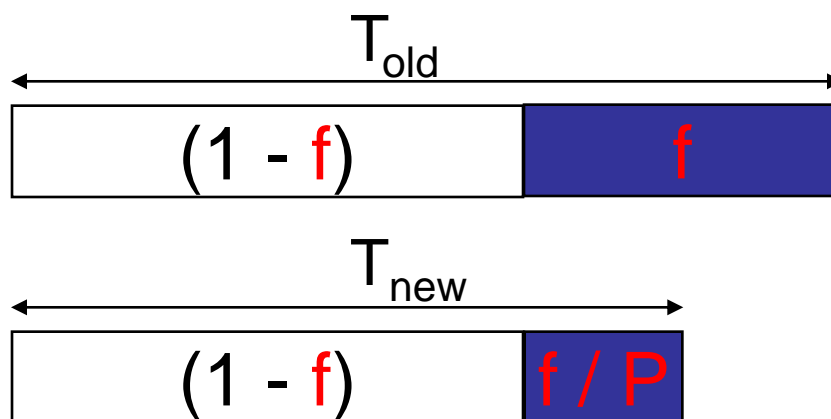


Amdahl's Law

□ Make the common case faster

□ $\text{Speedup} = \text{Perf}_{\text{new}} / \text{Perf}_{\text{old}} = T_{\text{old}} / T_{\text{new}} = \frac{1}{(1-f) + \frac{f}{P}}$

□ Performance improvement from using faster mode is limited by the fraction the faster mode can be applied.



Amdahl's Law Example

- New CPU 10X faster
- I/O bound server, so 40% time waiting for I/O

$$\begin{aligned} \text{Speedup}_{\text{overall}} &= \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}} \\ &= \frac{1}{(1 - 0.4) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56 \end{aligned}$$

- Apparently, its human nature to be attracted by 10X faster, vs. keeping in perspective its just 1.6X faster

Amdahl's Law Example

- Overall speedup if we make 90% of a program run 10 times faster.

$$\begin{aligned} \text{Speedup}_{\text{overall}} &= \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}} \\ &= \frac{1}{(1 - 0.9) + \frac{0.9}{10}} = \frac{1}{0.19} = 5.26 \end{aligned}$$

- Overall speedup if we make 80% of a program run 20% faster.

$$\begin{aligned} \text{Speedup}_{\text{overall}} &= \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}} \\ &= \frac{1}{(1 - 0.8) + \frac{0.8}{1.2}} = \frac{1}{0.86} = 1.153 \end{aligned}$$

Amdahl's Law

- Pitfall: Expecting the improvement of one aspect of a machine to increase performance by an amount proportional to the size of the improvement.
- Example:
 - Suppose a program runs in 100 seconds on a machine, with multiply operations responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?

100 (total time) = 80 (for multiply) + UA (unaffected)

→ Speedup = 4 = $1 / [(1 - 0.8) + 0.8/N]$ → $0.05 = 0.8/N$

→ $N = 8 / 0.05 = 16$

Or: $100/4$ (new total time) = 5 (for multiply) + UA (unaffected)

Amdahl's Law

- Example (continued):

- How about making it 5 times faster?

100 (total time) = 80 (for multiply) + UA (unaffected)

→ Speedup = 5 = $1 / [(1 - 0.8) + 0.8/N]$ → $0 = 0.8/N$

→ N ??? !

100/5 (new total time) = **0** + UA (unaffected): !

Amdahl's Law

- This concept is the **Amdahl's law**. Performance is limited to the non-speedup portion of the program.
- Execution time after improvement = Execution time of unaffected part + (execution time of affected part / speedup)
- Corollary of Amdahl's law: Make the common case fast.

Example 6

- Suppose we enhance a machine making all floating-point instructions run **five times** faster. If the execution time of some benchmark before the floating-point enhancement is **12 seconds**, what will the speedup be if **half of the 12 seconds** is spent executing floating-point instructions?

Ex. Time (enhance machine) = $6 + 6/5 = 7.2$

→ Speedup = $12/7.2 = 1.67$

Amdahl' law: Speedup = $1/[(1-0.5)+0.5/5] = 1.67$

Example 7

- We are looking for a benchmark to show off the **new floating-point unit** described in the previous example, and we want the overall benchmark to show a **speedup of 3**. One benchmark we are considering **runs for 100 seconds** with the old floating-point hardware. How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?

Ex. Time (old) = 100ns, Speedup = 3

⇒ Ex. Time (new) 33.33ns

Amdahl's law: $\text{Speedup} = 3 = 1/[(1-x) + x/5] \Rightarrow x = 5/6 \Rightarrow \text{Ex. time (new)} = 100/6 + 100 \cdot x/5 = 100/3 = 33.33 \text{ sec}$

Conclusion

- Performance is specific to a particular program/s
 - Total execution time is a consistent summary of performance
 - For a given architecture performance increases come from:
 - increases in clock rate (without adverse CPI affects)
 - improvements in processor organization that lower CPI
 - compiler enhancements that lower CPI and/or instruction count
 - Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance
 - You should not always believe everything you read! Read carefully!
-

Enjoy !!!

Q&A