



High Performance Computer Architecture (CS60003)

**Dept. of Computer Science & Engineering
Indian Institute of Technology Kharagpur**

Spring 2020



Benchmarks



Performance Evaluation

- Rely on Queuing theoretical or analytical models: When the underlying system can be described as an abstract model.
 - Example: Assess the throughput of a disk subsystem when the input rate and service time can be expressed as statistical distributions.
 - However, this is not in our scope.
- Measure the execution time of a suite of programs, called benchmarks.
- Simulate the whole system or components.

Benchmarks

- We intend to study the general purpose micro-processor machines.
- The choice of the best machine depends on the workload they perform
 - Single user: That which executes the common programs fastest.
 - Manager of a Webserver: That with least mean response time for queries on the web.
 - Designer of a database: Throughput of transactions.

Benchmarks, are a suite of test programs, available for each category of users.

When we are interested to evaluate the processor and the memory hierarchy, we look forward to those which evaluate in a compute-intensive manner. - neglecting, OS, Networking, IO effects.

Benchmark Classifications

■ Synthetic Benchmarks:

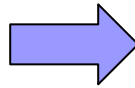
- Artificial programs specially written to measure performance of specific architectural factors.
- Example1, Whetstone: contains a very large proportion of floating point instructions.
- Example1, Dhrystone: contains a very large proportion of string operations.
- Not used for modern processors, but used for embedded processors.

Benchmark Classifications

■ Kernels:

- Small portions of real programs that exercise a certain feature of the system.
- Example1: Livermore. It contains 24 FORTRAN DO loops extracted from real programs.
 - Used for testing potential vectorization of the loops themselves, whether by hardware or by compilers.
 - ***Vectorization is the process of converting an algorithm from operating on a single value at a time to operating on a set of values at one time.***
Modern CPUs provide direct support for vector operations where a single instruction is applied to multiple data (SIMD).

```
for (int i=0; i<16; ++i)  
C[i] = A[i] + B[i];
```



```
for (int i=0; i<16; i+=4)  
C[i:i+3] = A[i:i+3]+B[i:i+3];
```

- Microbenchmarks: Isolate & measure special characteristics of architecture.

Benchmark Classifications

- Complete Programs:
 - Suite of selected programs depends on the intended use of the benchmarks.
 - Nonprofit organization of SPEC (System Performance Evaluation Corporation) tries to “establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers”
 - Publishes benchmarks for variety of applications
 - SPEC CP XXXX: Most relevant to microprocessor vendors and designers (XXXX is the year, like 2000)
 - Olden Suite: 10 programs with extensive linked list processing, thus offering challenges to exploitation of instruction level parallelism and to hide memory latencies.
 - TPC (Transaction Processing Council): Collection of programs for commercial workloads to evaluate servers.

SPEC Benchmarks

- Announced in 1989
- Expressed the performance of a system relative to that of a VAX 11/780, which had a SPEC rating of 1.
- It was modified in 92, 95.
- It was measured relative to a new machine, a 40MHz Sun SPARCstation without a 2nd level cache, and 64MB of memory.
- Revised again in 2000, SPEC2000.
- Normalizing machine was a Sun Ultra5_10 SparcStation, that run at 300 MHz , 2 level cache hierarchy and 256MB of memory. SPEC Rating of 100.
- Revised again in 2006, similar processor but larger cache and main memory. Recent in 2017 (<https://www.spec.org/cpu2017>)

Ingredients

- CPU2000 consists of 12 integer programs (11 written in C, 1 in C++), 14-floating point programs (11 written in FORTRAN77 or 90, and 3 in C)
- Some examples, Compression, FPGA circuit placement and routing, Game playing CHESS, Group Theory interpreter, quantum, water modeling, 3D-graphics, seismic wave, primality testing, finite element car crash, image processing, meteorology, word processing, etc.
- Each program has 3 datasets:
 - test: quick functional test
 - train: medium length runs
 - reference: one used for reporting results
 - Runs for tens or hundreds of billions of instructions.
 - 3 GHz machine with an IPC of 1 it would take 10 seconds to run 30 billion instruction benchmark
 - Compiler optimizations which are specific to any program are disallowed.

Is SPEC representative?

- There seems to be only one program mildly representative of desktop applications (parser)
- Not present in 2006
- There is only one interpreter program-perl
- Some C++ programs in 2006, but no Java programs.
- However, studies of desktop applications and interpreters have shown that:
 - their instruction mixes and,
 - the way they exercise the lower levels of the memory hierarchyare pretty similar to what is happening with the integer programs in SPEC benchmarks.

Reporting Benchmark Results

- Several ways have been advocated.
- Most obvious way is to report the average as arithmetic mean, ie. $T = \frac{1}{n} \sum_i T_i$
- If some programs are more important than others we computed weighted arithmetic means.
- If instead of measuring the rates, like IPC, we use harmonic mean.

□ Thus, $IPC = \frac{n}{\sum_i \frac{1}{IPC_i}}$

Reporting Benchmark Results

- SPEC does not use arithmetic means of execution times
- It normalizes the results of the benchmarks to some reference machine.
- Then the geometric means of the ratios of the execution time of the runs are reported.
 - $G = \sqrt[n]{\prod_i \frac{T_i}{S_i}}$, T_i is the execution time for the i^{th} program, and S_i is the execution time on the reference machine.
 - One advantage, if comparisons between 2 machines be made wrt. an old reference machine, no reevaluation needs to be done wrt. a new reference machine.

Metrics can fool!

	Normalization time on reference machine	Normalized time on M1	Normalized time on M2
Program P1	1	5	10
Program P2	1	5	2
Arithmetic Mean	1	5	6
Geometric Mean	1	5	4.5

Total time for M1=10 time units, for M2=12.

However, Geometric Mean shows M2 is faster!


See how the benchmark can be manipulated.

If the time for M2 for P2 was 3, Geometric Mean for M2 would have been higher.

By “optimizing” the short program by 1 time unit, shows anomaly.

However, opportunities for such optimizations does not exist in SPEC benchmarks, because number of instructions executed in each program is of the same order.

A Sample Report

 SPEC® CPU2017 Floating Point Rate Result <small>Copyright 2017-2018 Standard Performance Evaluation Corporation</small>	
ASUSTeK Computer Inc. ASUS RS700-E9(Z11PP-D24) Server System (2.70 GHz, Intel Xeon Gold 6150)	SPECrate2017_fp_base = 199 SPECrate2017_fp_peak = 201
CPU2017 License: 9016 Test Sponsor: ASUSTeK Computer Inc. Tested by: ASUSTeK Computer Inc.	Test Date: Dec-2017 Hardware Availability: Jul-2017 Software Availability: Sep-2017

Benchmark result graphs are available in the PDF report.

Hardware		Software	
CPU Name:	Intel Xeon Gold 6150	OS:	SUSE Linux Enterprise Server 12 (x86_64) SP2
Max MHz.:	3700		Kernel 4.4.21-69-default
Nominal:	2700	Compiler:	C/C++: Version 18.0.0.128 of Intel C/C++ Compiler for Linux; Fortran: Version 18.0.0.128 of Intel Fortran Compiler for Linux
Enabled:	36 cores, 2 chips, 2 threads/core	Parallel:	No
Orderable:	1, 2 chip(s)	Firmware:	Version 0601 released Oct-2017
Cache L1:	32 KB I + 32 KB D on chip per core	File System:	btfrfs
L2:	1 MB I+D on chip per core	System State:	Run level 3 (multi-user)
L3:	24.75 MB I+D on chip per chip	Base Pointers:	64-bit
Other:	None	Peak Pointers:	64-bit
Memory:	768 GB (24 x 32 GB 2Rx4 PC4-2666V-R)	Other:	None
Storage:	1 x 960 GB SATA SSD		
Other:	None		

Results Table

Benchmark	Base							Peak							
	Copies	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	Copies	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	
503.bwaves_r	72	<u>1529</u>	<u>472</u>	1529	472	1530	472	72	1526	473	1526	473	<u>1526</u>	<u>473</u>	
507.cactuBSSN_r	72	504	181	<u>505</u>	<u>181</u>	505	180	72	<u>505</u>	<u>180</u>	505	180	505	180	
508.namd_r	72	383	179	381	180	<u>383</u>	<u>179</u>	72	382	179	<u>382</u>	<u>179</u>	383	179	
510.parest_r	72	1669	113	<u>1684</u>	<u>112</u>	1700	111	72	1687	112	1699	111	<u>1695</u>	<u>111</u>	
511.povray_r	72	<u>603</u>	<u>279</u>	618	272	596	282	72	521	323	<u>524</u>	<u>321</u>	525	320	
519.lbm_r	72	678	112	677	112	<u>677</u>	<u>112</u>	72	<u>677</u>	<u>112</u>	678	112	677	112	
521.wrf_r	72	772	209	<u>784</u>	<u>206</u>	787	205	72	788	205	785	205	<u>785</u>	<u>205</u>	
526.blender_r	72	<u>495</u>	<u>222</u>	495	222	495	222	72	487	225	<u>487</u>	<u>225</u>	488	225	
527.cam4_r	72	547	230	<u>550</u>	<u>229</u>	550	229	72	543	232	<u>542</u>	<u>232</u>	542	232	
538.imagick_r	72	509	352	<u>509</u>	<u>352</u>	510	351	72	511	351	<u>511</u>	<u>351</u>	511	351	
544.nab_r	72	<u>399</u>	<u>303</u>	398	304	400	303	72	396	306	<u>396</u>	<u>306</u>	395	307	
549.fotonik3d_r	72	1985	141	<u>1986</u>	<u>141</u>	1986	141	72	<u>1985</u>	<u>141</u>	1981	142	1985	141	
554.roms_r	72	1263	90.6	<u>1268</u>	<u>90.2</u>	1269	90.2	72	1271	90.0	1270	90.1	<u>1271</u>	<u>90.0</u>	
SPECrate2017_fp_base		199													
SPECrate2017_fp_peak		201													

Results appear in the order in which they were run. Bold underlined text indicates a median measurement.







Thank You!



High Performance Computer Architecture (CS60003)

**Dept. of Computer Science & Engineering
Indian Institute of Technology Kharagpur**

Spring 2020



Measurement of Performance

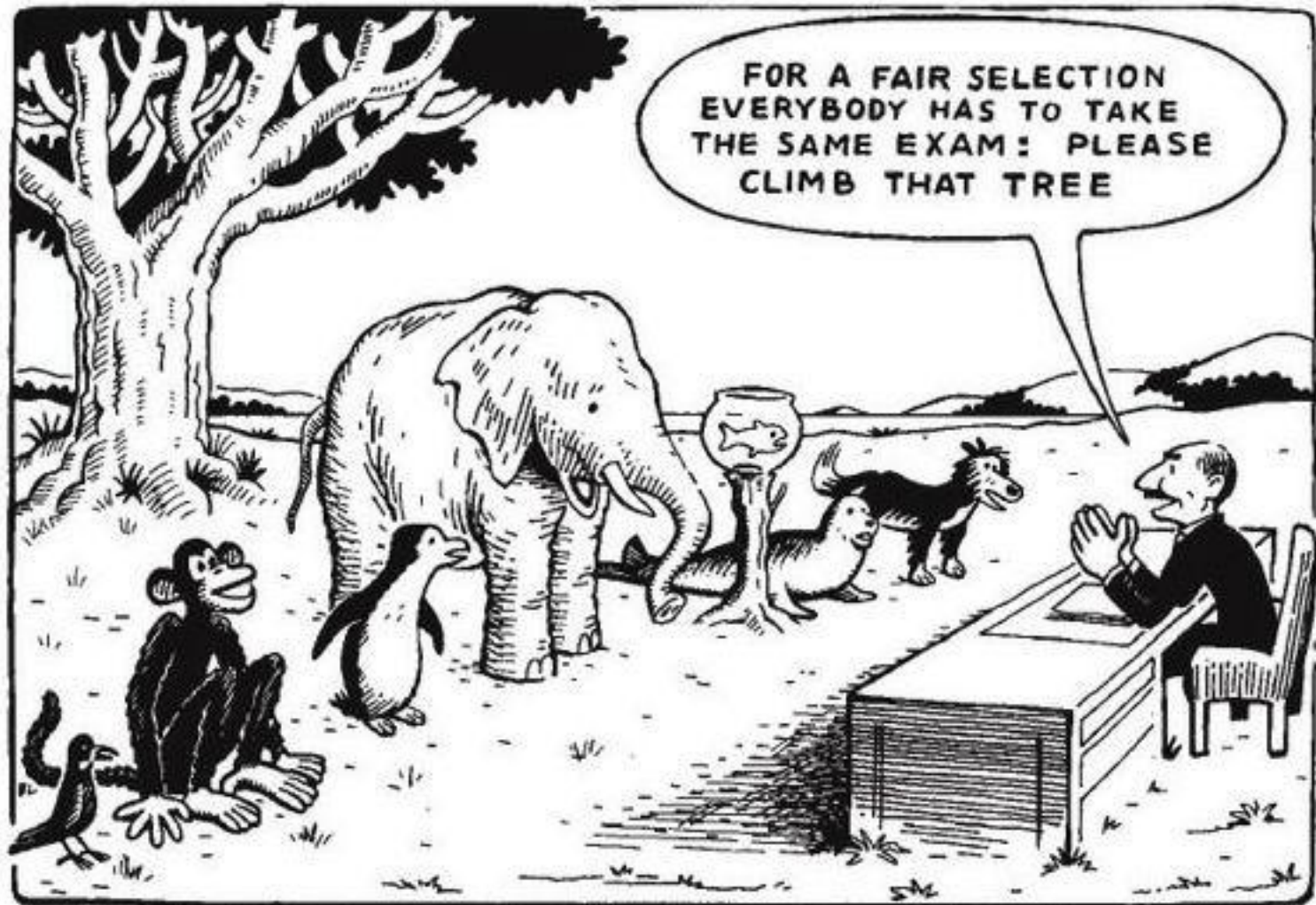
Measurement Techniques

- **Modern computer systems are built from sophisticated microprocessors and extensive memory hierarchies.**
- **Advance in technology have resulted in exponential growth in speed (ie clock frequency) and amount of logic (number of transistors) that a chip can have.**
- **Computer Architects have exploited these factors to enhance performance using architectural techniques:**
 - **Main topic of our course.**

Microprocessor Race

- Over 30 years old.
- Intel 4004 was introduced in 1971.
- The functionality of 4004 wrt. the mainframes of that period was modest.
- 30 years later, workstations powered by engines, like AMD Athlon, IBM PowerPC, Intel Pentium, Sun UltraSPARC can surpass remaining mainframes at a lower cost.
- Raw speed and number of transistors on a chip give a good feel for the progress of processors.
- However, to assess the performance of a computer system we need more precise metrics related to the execution of programs.

What to look for?





Performance Metrics

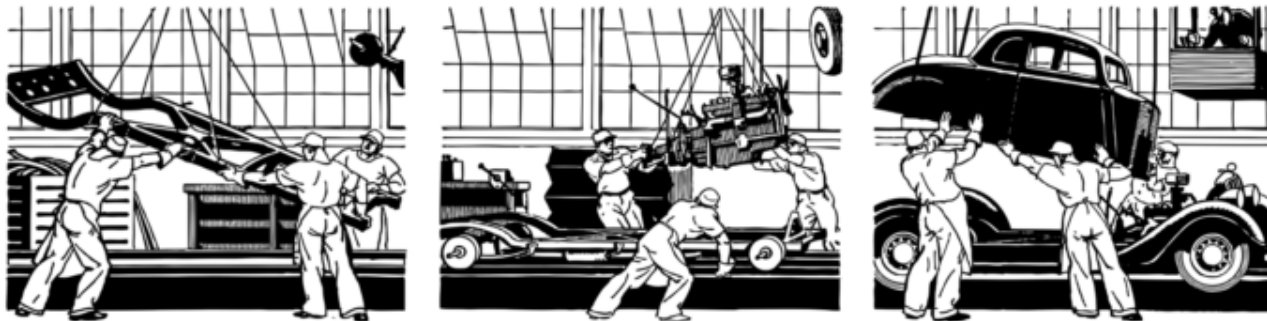
Hence, we need:

- Metrics that reflect the underlying architecture in a program independent fashion.
- A suite of programs, or benchmarks, that are representative of the intended load of a processor.

It is not easy! Lets start with some attempts.

Usual Performance Metrics

- Latency: delay from start to done for one output
- Throughput: no of outputs/second
- Can be confusing, Is $\text{Throughput} = 1/\text{Latency}$? No!
- Consider, a car assembly.



Say, latency is 4 hours, and there are 20 steps.
The throughput is hence 5 cars/hour (if the pipeline is full!)

Does not represent the architecture.

SpeedUp

- Informally we say, “X is N times faster than Y”
- SpeedUp is $N = \frac{Speed(X)}{Speed(Y)}$
 - Using latency or throughput as a measure of speed.
 - $N = \frac{Latency(Y)}{Latency(X)}$
 - $N = \frac{Throughput(X)}{Throughput(Y)}$

Quiz 1

- A laptop takes 4 hours to compress a video.
- A new laptop can do it in 10 mnts.
- $\text{SpeedUp} = 4 \times 60 / 10 = 24$
- $\text{SpeedUp} > 1$, is more intuitive and implies that there is a speedup of performance.

Quiz 2

- Consider the reverse situation.
- A laptop can compress a video in 10 mnts.
- It breaks, and so we now use the old one which compresses in 4 hours.
- $\text{SpeedUp} = 10/4 \times 60 = 1/24 = 0.04$
- $\text{SpeedUp} < 1$, performance has deteriorated.

Average Execution Time

- How to compute the average SpeedUp?

	Processor++	Processor+	SpeedUp
AppA	9s	18s	2
AppB	10s	7s	0.7
AppC	5s	11s	2.2
Average	$24/3=8s$	$36/3=12s$	1.5
Geometric Mean			1.45

Average SpeedUp=1.63

Average using GM, SpeedUp=1.45

Average execution time is computed using Geometric Means.

Quiz 3

- Consider two laptops, which have speedups wrt. the following applications:
 - Music: SpeedUp of 2
 - Games: SpeedUp of 8
- Give an estimate of the overall SpeedUp?

Quiz 3

- Consider two laptops, which have speedups wrt. the following applications:
 - Music: SpeedUp of 2
 - Games: SpeedUp of 8
- Give an estimate of the overall SpeedUp?
- We shall compute the Geometric Mean, thus Overall SpeedUp = $\sqrt[2]{2 \times 8} = 4$.
- Computing, the average speed as arithmetic means would not be correct, as SpeedUp is the ratio of two times.

Execution Time and Instructions Per Cycle (IPC)

- We look forward to metrics to evaluate the microarchitecture and its memory hierarchy.
- The execution time of a program whose code and data reside in the memory hierarchy is denoted by EX_{CPU} .
 - At this point we ignore the effect of I/O.
- EX_{CPU} depends on:
 - the number of instructions executed
 - the time to execute one instruction
- Time to execute one instruction depends on:
 - Cycle time (reciprocal of the clock frequency)
 - Cycles required to execute an instruction
 - CPI: Cycles Per Instruction (Program independent, clock-frequency independent)

Iron Law of Performance

- $EX_{CPU} = \text{Number of Instructions} \times CPI \times \text{cycle time}$
- These 3 components help to factor the contributions of important areas:

- *Number of Instructions:*

- affected by the algorithm and the compiler
- affected by the instruction set architecture

- *CPI:*

- affected by instruction set architecture
- affected by the processor design

- *Clock cycle time:*

- affected by processor design: reduced critical path
- affected by circuit designer, transistor physics

First, approximation assuming that all the instructions take the same time to execute.

Role of the Computer Architect

- Can influence the:
 - Instruction Set: Choosing complex instructions will imply fewer instructions per program, but will take more cycles to execute each instruction.
 - Processor Design: Tradeoff between:
 - A processor that has very short clock cycle at the expense of spending more clock cycles per instruction, or,
 - A processor that has a larger clock cycle duration, but with reduced number of cycles per instruction.
- A Good design would try to balance these choices!

Quiz 4

- Program executes 3 billion instructions.
- Processor spends 2 cycles on each instruction.
- Processor clock is 3GHz.
- What is EX_{CPU} ?

Quiz 4

- Program executes 3 billion instructions.
- Processor spends 2 cycles on each instruction.
- Processor clock is 3GHz.
- What is EX_{CPU} ?
- $EX_{CPU} = 3 \times 10^9 \times 2 \times \frac{1}{3 \times 10^{-9}} = 2 \text{ sec.}$

Iron Law of Performance for Unequal Instruction Times

- $EX_{CPU} = (\sum_i IC_i \times CPI_i) \times \text{cycle time}$
 - IC_i : No of instruction for the i th type of instruction
 - CPI_i : Cycles per Instruction for the i th type of instruction

Quiz 5

- Consider the following distribution of total 50 Billion instructions of an application:
 - ☐ 10 Billion instructions, of type Branch, CPI=4
 - ☐ 15 Billion instructions, of type Loads, CPI=2
 - ☐ 5 Billion instructions, of type Stores, CPI=3
 - ☐ Rest are integer Add, CPI=1
 - ☐ The clock is at 4 GHz.
 - ☐ What is the EX_{CPU} ?

Quiz 5

■ Consider the following distribution of instructions:

- ☐ 10 Billion instructions, of type Branch, CPI=4
- ☐ 15 Billion instructions, of type Loads, CPI=2
- ☐ 5 Billion instructions, of type Stores, CPI=3
- ☐ Rest are integer Add, CPI=1
- ☐ The clock is at 4 GHz.
- ☐ $EX_{CPU} = (10 \times 4 + 15 \times 2 + 5 \times 3 + 20 \times 1) / 4 = 105/4 = 26.25 \text{sec.}$

IPC (Instructions Per Cycle)

- IPC is a cosmetic change to CPI in that it is psychologically more appealing to strive for increases in IPC than decrease in CPI.

- Thus,

$$EX_{CPU}$$

$$= (\text{Number of Instructions} \times \text{cycle time}) / \text{IPC}$$

- IPC is a metric which represents the throughput
- EX_{CPU} is a metric which represents the latency.

SpeedUp for Parallel Processor: Amdahl's law

- Let T_i denote the time to execute on i processors $i \in Z$;
- SpeedUp for n processors is:

$$SpeedUp_n = \frac{T_1}{T_n}$$

- Superficially for programs that exhibit lot of parallelism, one would expect $T_n = T_1/n$
- However, even for “embarrassingly parallel” programs, there is always a small amount of sequential execution and synchronization.
- Although, originally stated for parallel processors can be applied for any enhancement on a fraction of the total time.

Amdahl's Law

- $SpeedUp = \frac{1}{(1 - Frac_{Enh}) + \frac{Frac_{Enh}}{SpeedUp_{Enh}}}$
- $Frac_{Enh}$ is the fraction of original execution time that is affected by enhancement.
- **Note that the percentage is wrt. time and not say number of instructions!**

Quiz 6

- Consider a program with 50 Billion instructions.
- Clock frequency is 2GHz.
- Say, an architectural modification enhances the branch instruction so that its CPI goes from 4 to 2.

Instruction Type	% of all instructions in a program	CPI
Arithmetic	40	1
Branch	20	4
Load	30	2
Store	10	3

What is the overall SpeedUp?

Quiz 6

- Consider a program with 50 Billion instructions.
- Clock frequency is 2GHz.
- Say, an architectural modification enhances the branch instruction so that its CPI goes from 4 to 2.

Instruction Type	% of all instructions in a program	CPI
Arithmetic	40	1
Branch	20	4
Load	30	2
Store	10	3

$$\text{SpeedUp} = \frac{1}{0.8 + \frac{0.2}{2}} = \frac{1}{0.9} = 1.111$$

Quiz 6

- Consider a program with 50 Billion instructions.
- Clock frequency is 2GHz.
- Say, an architectural modification enhances the branch instruction so that its CPI goes from 4 to 2.

Instruction Type	% of all instructions in a program	CPI
Arithmetic	40	1
Branch	20	4
Load	30	2
Store	10	3

$$\text{SpeedUp} = \frac{1}{0.8 + \frac{0.2}{2}} = \frac{1}{0.9} = 1.111 \quad \text{Wrong!}$$

Correct Solution

- Using Iron Law before enhancement,

$$\begin{aligned} EX_{CPU} &= (0.4 \times 1 + 0.2 \times 4 + 0.3 \times 2 + 0.1 \times 3) \times 50 \times \frac{1}{2} \\ &= (0.4 + 0.8 + 0.6 + 0.3)25 = 2.1 \times 25 \end{aligned}$$

- After enhancement,

$$\begin{aligned} EX_{CPU} &= (0.4 \times 1 + 0.2 \times 2 + 0.3 \times 2 + 0.1 \times 3) \times 50 \times \frac{1}{2} \\ &= (0.4 + 0.4 + 0.6 + 0.3)25 = 1.7 \times 25 \end{aligned}$$

- Hence, SpeedUp = $2.1/1.7 = 1.24$

Implication of Amdahl's Law

$$\blacksquare \text{ SpeedUp} = \frac{1}{(1 - \text{Frac}_{Enh}) + \frac{\text{Frac}_{Enh}}{\text{SpeedUp}_{Enh}}}$$

- Enhancement 1: SpeedUp of 20 on component which takes 10% of time

$$\square \text{ SpeedUp} = \frac{1}{0.9 + \frac{0.1}{20}} = 1.105$$

- Enhancement 2: SpeedUp of 2 on component which takes 80% of time

$$\square \text{ SpeedUp} = \frac{1}{0.2 + \frac{0.8}{2}} = 1.67$$

- Even with an infinite SpeedUp on 10% of time, $\text{SpeedUp} = \frac{1}{0.9 + 0} = 1.111$

Implication of Amdahl's Law

$$\blacksquare \text{ SpeedUp} = \frac{1}{(1 - \text{Frac}_{Enh}) + \frac{\text{Frac}_{Enh}}{\text{SpeedUp}_{Enh}}}$$

- Enhancement 1: SpeedUp of 20 on component which takes 10% of time

$$\square \text{ SpeedUp} = \frac{1}{0.9 + \frac{0.1}{20}} = 1.105$$

- Enhancement 2: SpeedUp of 2 on component which takes 80% of time

$$\square \text{ SpeedUp} = \frac{1}{0.2 + \frac{0.8}{2}} = 1.67$$

- Even with an infinite SpeedUp on 10% of time, $\text{SpeedUp} = \frac{1}{0.9 + 0} = 1.111$

Make Common Case Fast!

Quiz 7

- Consider the following statistics:

Inst Type	% of Time	CPI
Arithmetic	40	1
Branch	20	4
Load	30	2
Store	10	3

$$\text{SpeedUp}_1 = \frac{1}{0.8 + \frac{0.2}{4/3}} = 1.05$$

$$\text{SpeedUp}_2 = \frac{1}{0 + \frac{1}{1.15}} = 1.15$$

$$\text{SpeedUp}_3 = \frac{1}{0.9 + \frac{0.1}{3/2}} = 1.034$$

Which is the best.

Improvement among:

1. Branch: CPI 4 \rightarrow 3
2. Increase Clock Frequency: 2 \rightarrow 2.3GHz
3. Store: CPI 3 \rightarrow 2?

LHADMA's Law

- Cautions that in pursuit of optimizing the common case, the uncommon case shouldn't be slowed down too much.

- Example: Consider the following two options

- ☐ Improvement of 2x on components which take 90% time
- ☐ But slow down the rest by 10x

$$SpeedUp = \frac{1}{\frac{0.1}{0.1} + \frac{0.9}{2}} = \frac{1}{1.45} = 0.7$$

- Even with an infinite speedup on the 90% component: $SpeedUp = \frac{1}{\frac{0.1}{0.1} + \frac{0.9}{\infty}} = 1$

Law of Diminishing Returns

- Diminishing returns with respect to Amdahl's Law refers to the fact that continuing to optimize a specific part of the execution eventually provides less and less gains in speedup
 - This is so as the optimized portion grows smaller and smaller, accounting for less and less of execution time.
- Intuitively, it can be considered that the “low hanging fruit” of optimizations will be gone after a while, making it harder to optimize.
- The practical implication of this is that after optimizing a portion of the execution, one must re-evaluate which portion is now dominant.

Example

- Consider a program execution which has two parts in the execution time, p and q.
- Initially, p and q are both 50% of the time.
- Due to an optimization, say there is a SpeedUp on q by 50%.

□ Thus, $SpeedUp = \frac{1}{0.5 + \frac{0.5}{2}} = 1.33$

- Say, the architect optimizes the 2nd part again with a further SpeedUp of 2.
- Note, now p takes 0.67 of the time, and q (due to the first optimization) takes 0.33 of the time.

□ Thus, $SpeedUp = \frac{1}{0.67 + \frac{0.33}{2}} = 1.2$

- Thus, the speedup diminishes; eventually we will get almost no speedup, as we are speeding up less and less of the execution time!



Thank You!