

182.690 RECHNERSTRUKTUREN - PERFORMANCE

Thomas Polzer
tpolzer@ecs.tuwien.ac.at
Institut für Technische Informatik

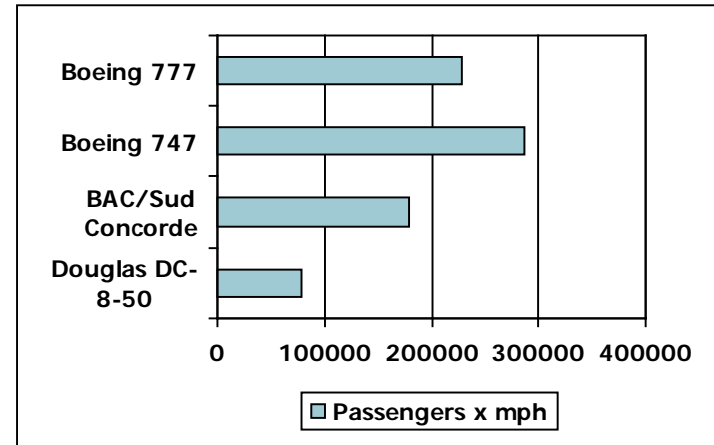
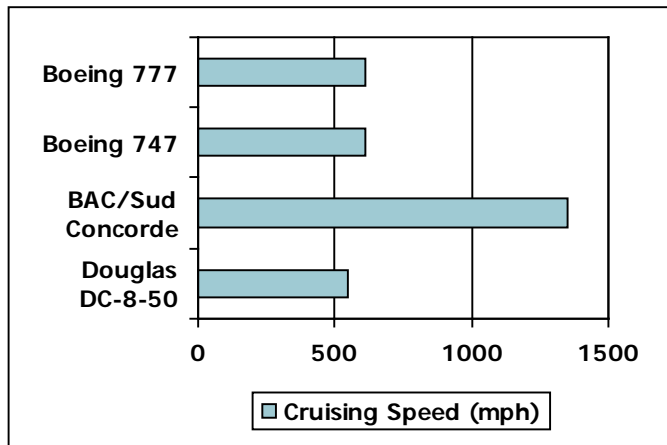
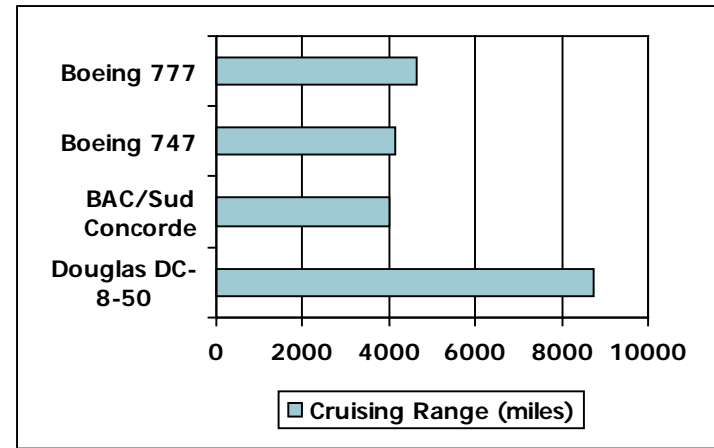
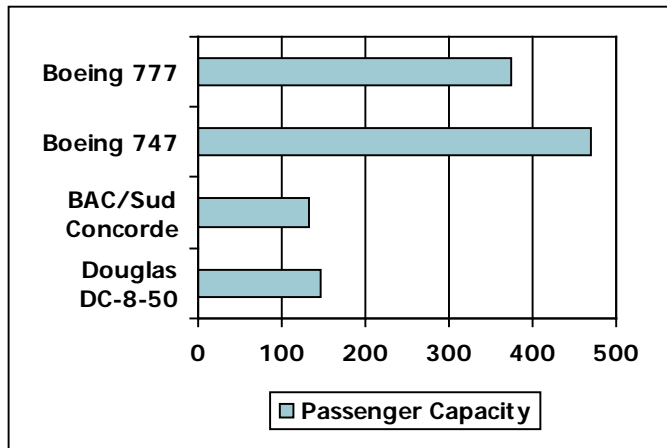


Overview

- We want to be able to
 - Understand
 - Determine and
 - Specifyperformance correctly
- Decision at purchase, design, optimization
- Optimum utilization of HW when programming
- Better understanding of computer architecture

Defining Performance

- Which airplane has the best performance?



Response Time and Throughput

- Response time
 - How long it takes to do a task
- Throughput
 - Total work done per unit time
 - e.g., tasks/transactions/... per hour
- How are response time and throughput affected by
 - Replacing the processor with a faster version?
 - Adding more processors?
- We'll focus on response time for now...

Response Time vs. Throughput (Example)

- Example ski-lift:
 - Two chairs every 15 s
 - Time for one ascent: 10 min
- Throughput = 480 Persons/h
- Journey time = time for ascent+ waiting time > 10 min
- Improvement if using four chairs instead of two:
 - Throughput = 960 Persons/h
 - Journey time = time for ascent (still!) + wait time > 10 min

Performance Definition

- Performance = $1 / \text{Execution Time}$
- Relative Performance:
 - “X is n time faster than Y”
 - $\text{Performance}_X / \text{Performance}_Y = \text{Execution Time}_Y / \text{Execution Time}_X = n$

Relative Performance (Example)

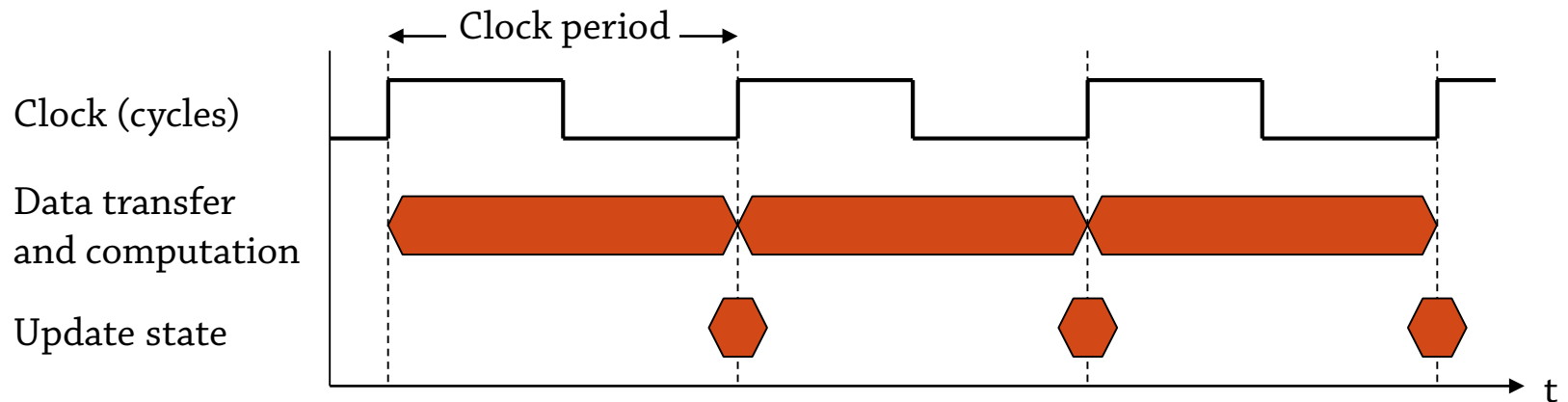
- Example: time taken to run a program
 - 10s on computer A, 15s on computer B
- $\text{Execution Time}_B / \text{Execution Time}_A = 15\text{s} / 10\text{s} = 1.5$
- So A is **1.5 times** faster than B

Measuring Execution Time

- Elapsed time
 - Total response time, including all aspects
 - Processing, I/O, OS overhead, idle time
 - Determines system performance
- CPU time
 - Time spent processing a given job
 - Discounts I/O time, other jobs' shares
 - Comprises **user** CPU time and **system** CPU time
 - Different programs are affected differently by CPU and system performance

Clock

- Operation of digital hardware governed by a constant-rate clock



- Clock period: duration of a clock cycle
 - e.g., $250 \text{ ps} = 0.25 \text{ ns} = 250 \times 10^{-12} \text{ s}$
- Clock frequency (rate): cycles per second
 - e.g., $4.0 \text{ GHz} = 4000 \text{ MHz} = 4.0 \times 10^9 \text{ Hz}$

Clock Cycles vs. Real Time

$$\begin{aligned}\text{CPU Time} &= \# \text{CPU Clock Cycles} \times \text{Clock Cycle Time} \\ &= \frac{\# \text{CPU Clock Cycles}}{\text{Clock Rate}}\end{aligned}$$

- Performance improved by
 - Reducing number of clock cycles
 - Increasing clock rate
 - Hardware designer must often trade off clock rate against cycle count

Clock Cycles (Example)

- Computer A has a clock frequency of 2 GHz. Running a specific program requires 10 s. The maximum time allowed to execute the program, however, is 6 s. When analyzing the processor, a designer suggests that, if an 20% clock cycle overhead is introduced, the frequency of the processor can be increased drastically. Which clock frequency would be required to meet the goal?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\begin{aligned}\text{Clock Cycles}_A &= \text{CPU Time}_A \times \text{Clock Rate}_A \\ &= 10\text{s} \times 2 \text{ GHz} = 20 \times 10^9\end{aligned}$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4 \text{ GHz}$$

Clock Cycles for a Program

- How many clock cycles do a specific program require?
- Instruction count determined by program, ISA and compiler
- BUT: not every instruction is finished within one clock cycle!
 - Multiplication, division
 - Floating point arithmetic
 - Memory access
 - Cache misses!

Clocks per Instructions (CPI)

- Average number of clock cycles per instruction (CPI)
 - Depends on CPU hardware
 - Depends on executed program!

Clock Cycles = Instruction Count \times Cycles per Instruction

CPU Time = Instruction Count \times CPI \times Clock Cycle Time

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

Performance Comparison (Example)

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\begin{aligned}\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\ &= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}\end{aligned}$$

$$\begin{aligned}\text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\ &= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}\end{aligned}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

CPI (More Details)

- If different instruction classes take different numbers of cycles:

$$\text{Clock Cycles} = \sum_{i=1}^n (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Instructions class CPI_i have $\text{Instruction Count}_i$ occurrences in the program
- Weighted average CPI

Relative frequency

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^n \left(\text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)$$

CPI Calculation (Example)

- Processor:

Class	A	B	C
CPI for class	1	2	3

- Compiler:

Frequency of instructions	A	B	C	Sum
Compiler 1	2	1	2	5
Compiler 2	4	1	1	6

- Which code is faster ?

CPI Calculation (Example)

- Compiler 1:
 - Instruction Count: 5
 - Clock Cycles: $2 \times 1 + 1 \times 2 + 2 \times 3 = 10$
 - Avg. CPI: $10/5 = 2.0$
- Compiler 2:
 - Instruction Count: 6
 - Clock Cycles: $4 \times 1 + 1 \times 2 + 1 \times 3 = 9$
 - Avg. CPI: $9/6 = 1.5$
- Compiler 2 creates more instruction but a faster program! ➔ Instruction count alone not a useful metric for performance!

Performance Summary

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

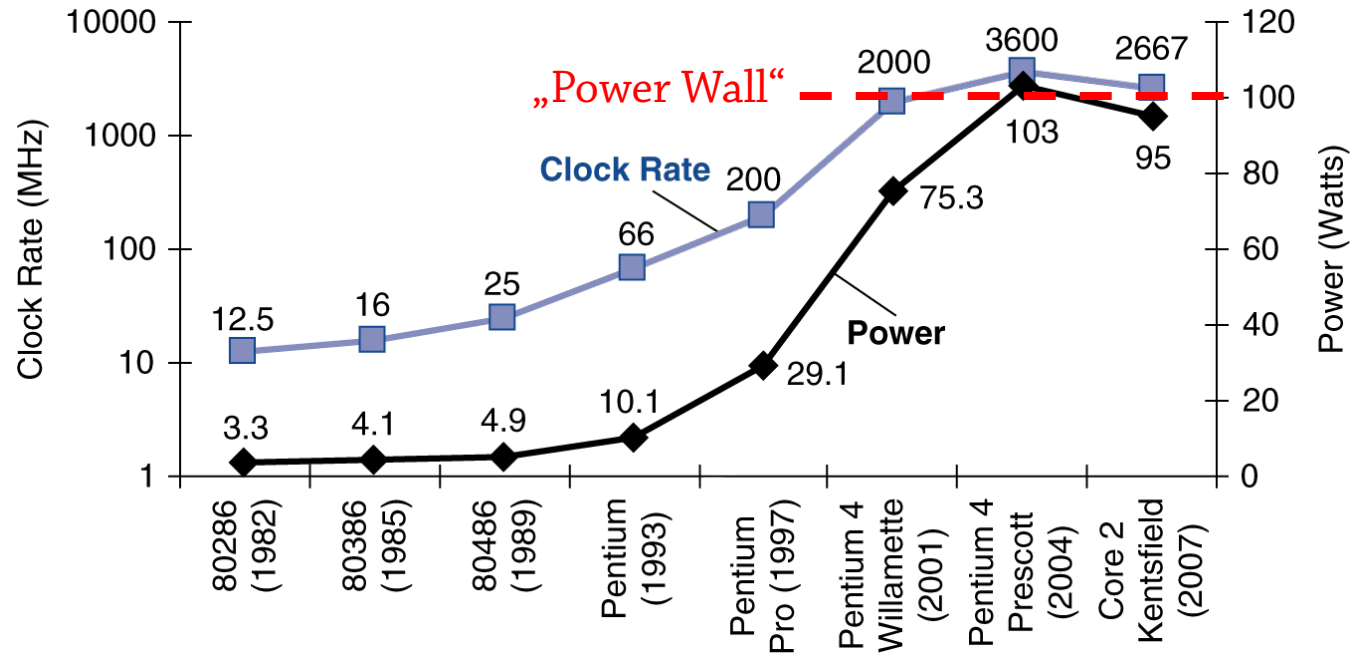
- Performance depends on
 - Algorithm: affects instruction count, possibly CPI
 - Programming language: affects instruction count, CPI
 - Compiler: affects instruction count, CPI
 - Instruction set architecture: affects instruction count, CPI, T_{clk}

Performance Dependencies

	Instruction count	CPI	Clock frequency
Algorithm	X	X	
Programming language	X	X	
Compiler	X	X	
ISA	X	X	X
Core organization		X	X
Technology			X

- In most cases an optimization of all dependencies is necessary to increase the CPU performance and tuning a single factor will not lead to the desired result!

Power Trends (Limits)



- In CMOS IC technology:

$$\text{Power} = \text{Capacitive Load} \times \text{Voltage}^2 \times \text{Switching Frequency}$$

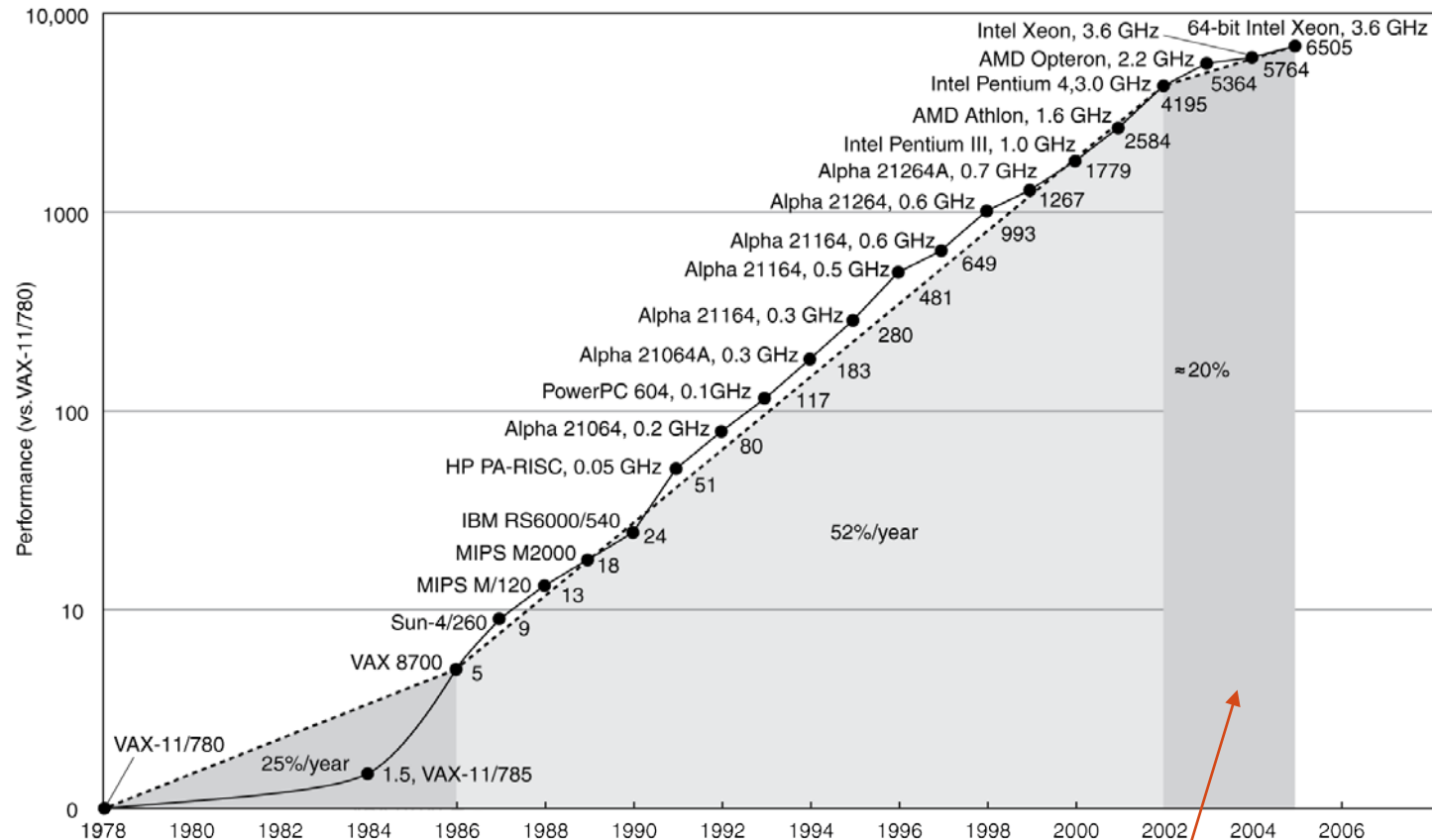
Reducing Power

- Suppose a new CPU has
 - 85% of capacitive load of old CPU
 - 15% voltage and 15% frequency reduction

$$\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{C_{\text{old}} \times 0.85 \times (V_{\text{old}} \times 0.85)^2 \times F_{\text{old}} \times 0.85}{C_{\text{old}} \times V_{\text{old}}^2 \times F_{\text{old}}} = 0.85^4 = 0.52$$

- The power wall
 - We can't reduce voltage further
 - We can't remove more heat
- How else can we improve performance?

Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

Multiprocessors

- Multicore microprocessors
 - More than one processor per chip
- Requires explicitly parallel programming
 - Compare with instruction level parallelism
 - Hardware executes multiple instructions at once
 - Hidden from the programmer
 - Hard to do
 - Programming for performance
 - Load balancing
 - Optimizing communication and synchronization

A Sea Change is at Hand

- The power challenge has forced a change in the design of μ -processors
 - Since 2002 the rate of improvement in the response time of programs on desktop computers has slowed from a factor of 1.5 per year to less than a factor of 1.2 per year
- As of 2006 all desktop and server companies are shipping μ -processors with multiple processors (cores) per chip

A Sea Change is at Hand (2)

Product	AMD Barcelona	Intel IA-64 Nehalem	IBM Power 6	Sun Niagara 2
Cores per chip	4	4	2	8
Clock rate	2.5 GHz	~2.5 GHz?	4.7 GHz	1.4 GHz
Power	120 W	~100 W?	~100 W?	94 W

- Plan of record is to double the number of cores per chip per generation (about every two years)

Benchmarking

- Real application
 - Meaningful results but would be user specific!
- Very simple program
 - Easy to verify but may lead to artificial results
- SPEC (System Performance Evaluation Cooperative)
 - Generally accepted compromise
 - www.spec.org

SPEC CPU Benchmark


- Programs used to measure performance
 - Supposedly typical of actual workload
- SPEC CPU2006
 - Elapsed time to execute a selection of programs
 - Negligible I/O, so focuses on CPU performance
 - Normalize relative to reference machine
 - Summarize as geometric mean of performance ratios
 - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^n \text{Execution time ratio}_i}$$

CINT2006 for Opteron X4 2356

Name	Description	ICx10 ⁹	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							11.7

High cache miss rates



SPEC Power Benchmark

- Power consumption of server at different workload levels
 - Performance: ssj_ops/sec
 - Power: Watts (Joules/sec)

$$\text{Overall ssj_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj_ops}_i \right) / \left(\sum_{i=0}^{10} \text{power}_i \right)$$

SPECpower_ssj2008 for AMD Opteron X4

Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
$\Sigma \text{ssj_ops} / \Sigma \text{power}$		493

Partial Optimization

The execution time of a program is 100 s, 80 s for multiplications and 20 s for all other instructions.

How much faster must the multiplication be such that the program now runs

(a) 4 times faster

New execution time: 25s, 20s for other instructions is unchanged → 5s for multiplication → $80 \text{ s} / 5 \text{ s} = 16$ times faster

(b) 5 times faster?

New execution time: 20s, 20s for other instructions is unchanged → **No more time for multiplications!**

Amdahl's Law

- Pitfall: Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- Corollary: make the common case fast

Amdahl's Law (Example)

A computer uses 30% of the execution time in a particular application for floating point multiplications and 8% for floating point divisions. When doing a redesign of the computer two options are available:

- Speed up FP divisions by a factor of eight
- Speed up FP multiplications by a factor of two

Which of the two options is better?

- Option 1: improved time is $0,08 * T * 1 / 8 + 0,92 * T = 0.93 * T$
- Option 2: improved time is $0,3 * T * 1/2 + 0,7 * T = 0,85 * T$
- Option 2 is better!

Fallacy: Expecting Low Power at Idle

- Look back at AMD Opteron X4 power benchmark
 - At 100% load: 295W
 - At 50% load: 246W (83%)
 - At 10% load: 180W (61%)
- Google data centers
 - Mostly operates at 10% – 50% load
 - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

Pitfall: MIPS as Performance Metric

- MIPS: Millions of Instructions Per Second
 - Doesn't account for
 - Differences in ISAs between computers
 - Differences in complexity between instructions

$$\begin{aligned}\text{MIPS} &= \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} \\ &= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}\end{aligned}$$

- CPI varies between programs on a given CPU

MIPS as Performance Metric (Example)

Two different compiler generate the code sequences with the following frequency of instructions for the same user code:

Compiler	CPI = 1	CPI = 2	CPI = 3
Compiler 1	$5 * 10^9$	$1 * 10^9$	$1 * 10^9$
Compiler 2	$10 * 10^9$	$1 * 10^9$	$1 * 10^9$

Which code is the faster one, how many MIPS does a processor with 2GHz clock frequency achieve when executing the different code sequences?

Compiler 1: Execution time: $(5*1+1*2+1*3) * 10^9 * 500\text{ps} = 5\text{s}$
→ $7000*10^6$ instr. in 5s = $7000*10^6 / (5*10^6) = 1400$ MIPS

Compiler 2: Execution time: $(10*1+1*2+1*3) * 10^9 * 500\text{ps} = 7.5\text{s}$
→ $12000*10^6$ instr. in 7.5s = $12000*10^6 / (5*10^6) = 2400$ MIPS

EXAMPLES



Ex 1: Performance-ISA

An instruction set has two implementations M1 and M2. Each implementation has four classes of instructions (A, B, C and D). The implementation M1 can be clocked with 1.5 GHz, while the implementation M2 achieves 2 GHz.

Class	CPI (M1)	CPI (M2)	Frequency
A	1.4	2	40%
B	2	1.7	43%
C	5.2	4	11%
D	1	8	6%

Ex 1: Performance-ISA

Tasks:

- Calculate the peak MIPS value for both implementations.
- Which clock frequency must be used for implementation M1 such that it has the same Peak MIPS value as M2?
- Calculate the average MIPS value for the given mix of instructions for both implementations.

Solution:

- $\text{Peak MIPS}(M1) = f / (\min(\text{CPI}) * 10^6) = 1500 * 10^6 / (1 * 10^6) = 1500$
 $\text{Peak MIPS}(M2) = 2000 * 10^6 / (1,7 * 10^6) = 1176.47$
- $f = \text{MIPS} * \min(\text{CPI}) * 10^6 = 1176.47 * 1 * 10^6 = 1.176 \text{ GHz}$
- $\text{MIPS}(M1) = f / (\text{CPI}_{\text{avg}} * 10^6) =$
 $1500 / (1,4 * 0,4 + 2 * 0,43 + 5,2 * 0,11 + 1 * 0,06) = 1500 / 2.052 = 731$
 $\text{MIPS}(M2) = f / (\text{CPI}_{\text{avg}} * 10^6) =$
 $2000 / (2 * 0,4 + 1,7 * 0,43 + 4 * 0,11 + 8 * 0,06) = 2000 / 2.451 = 816$

Ex 2: Performance

There are three options for the redesign of an existing processor. To benchmark the processor a program existing of the following components is developed:

Part 1	Calculate_Float()	20%
Part 2	Calculate_ALU()	18%
Part 3	Control()	20%
Part 4	Save()	20%
Part 5	Interrupts, OS service	Rest

Ex 2: Performance

Part	Option A	Option B	Option C
1	Speedup by 2	Slow down by 1.3	Speed up by 4
2	Slow down by 1.2	Speed up by 4	Slow down by 1.2
3	Unchanged	Speedup by 2	Speedup by 1.3
4	Speedup by 1.2	Unchanged	Slow down by 1.25
5	Slow down by 2.3	Speedup by 3	Slow down by 1.2

- What is the speedup of the three options
- Assume you could combine the best two options to a forth one. What would be its speedup?

Ex 2: Performance

- New relative execution times:

Part	Option A	Option B	Option C
1	$0.2/2 = 0.1$	$0.2*1.3 = 0.26$	$0.2/4 = 0.05$
2	$0.18*1.2 = 0.216$	$0.18 / 4 = 0.045$	$0.18*1.2 = 0.216$
3	$0.2*1 = 0.2$	$0.2/2 = 0.1$	$0.2/1.3 = 0.154$
4	$0.2/1.2 = 0.167$	$0.2*1 = 0.2$	$0.2*1.25 = 0.25$
5	$0.22*2.3 = 0.506$	$0.22/3 = 0.073$	$0.22*1.2 = 0.264$
S	1.189	0.678	0.934

- Speedup:

- Speedup A = $T_{\text{orig}}/T_A = 1 / 1.189 = 0.841 \rightarrow \text{Slower!}$
- Speedup B = $T_{\text{orig}}/T_B = 1 / 0.678 = 1.475$
- Speedup C = $T_{\text{orig}}/T_C = 1 / 0.934 = 1.071$

Ex 2: Performance

■ Combination of B and C:

Part	Option B	Option C	Option D
1	$0.2 * 1.3 = 0.26$	$0.2 / 4 = 0.05$	$0.2 / 4 = 0.05$
2	$0.18 / 4 = 0.045$	$0.18 * 1.2 = 0.216$	$0.18 / 4 = 0.045$
3	$0.2 / 2 = 0.1$	$0.2 / 1.3 = 0.154$	$0.2 / 2 = 0.1$
4	$0.2 * 1 = 0.2$	$0.2 * 1.25 = 0.25$	$0.2 * 1 = 0.2$
5	$0.22 / 3 = 0.073$	$0.22 * 1.2 = 0.264$	$0.22 / 3 = 0.073$
S	0.678	0.934	0.468

■ Speedup:

- Speedup A = $T_{\text{orig}} / T_A = 1 / 1.189 = 0.841 \rightarrow \text{Slower!}$
- Speedup B = $T_{\text{orig}} / T_B = 1 / 0.678 = 1.475$
- Speedup C = $T_{\text{orig}} / T_C = 1 / 0.934 = 1.071$
- Speedup D = $T_{\text{orig}} / T_D = 1 / 0.468 = 2.137$

Conclusion

- Determining the performance is important
- Execution time: the best performance measure
- Performance depends on the selected program
 - Real applications are the best benchmarks
 - Be careful when using performance metrics (MIPS vs. execution time)
- Performance optimizations are a tradeoff between
 - Clock frequency
 - CPI
 - Instruction count
- Optimize the common case!
- Power is a limiting factor
 - Use parallelism to improve performance
- Keep idle power in mind!

182.690 RECHNERSTRUKTUREN - PERFORMANCE

Thomas Polzer
tpolzer@ecs.tuwien.ac.at
Institut für Technische Informatik

