Performance

- Introduction
- Defining Performance
- The Iron Law of Processor Performance
- Processor performance enhancement
- Performance Evaluation Approaches
- Performance Reporting
- Amdahl's Law

Introduction

- Performance measurement is important:
  - Helps us to determine if one processir or computer works faster than other
  - Helps us to know how much performance improvement has taken place after incorproating some performance enhancement feature
  - Help to see through the marketing hype!
- Provides answer to the following questions:
  - Why is some hardware better than others for different programs?
  - What factors affect system performance?
    - Hardware, OS or compiler?
  - How does the machine’s instruction set affect performance?

Defining Performance in terms of time

- Time is the final measure of computer performance
- A computer exhibits higher performance if it executes program faster
- Response Time (elapsed time, Latency):
    - How long does it take for my job to run?
    - How long does it take to execute (start to finish) my job?
    - How long must/wait for the database query?

Individual user concern

- Throughput:
    - How many jobs can the machine run at once?
    - What is the average execution rate?
    - How much work is getting done?

System Manager concern

Execution Time

- Elapsed time
  - Count everything (disk and memory access, waiting for IO, running other programs, etc) from start to finish
  - A useful number, but often not good for comparison purpose

  Elapsed time = CPU time +wait time (IO, other program, etc.)

- CPU time
  - Doesn't count waiting for IO or time spent running other programs
  - Can be divided into user CPU time and system CPU time(OS calls)

    CPU time = user CPU time + System CPU Time

  Elapsed time = user CPU time + System CPU Time + wait time

- Our focus: User CPU time
  - CPU execution time or simply execution time: time spent executing the lines of code that are in our program

Measuring performance

- for some program running on machine X:

$$\text{Performance} = \frac{1}{execution\ time_x}$$

- X is n times faster than Y means:

$$\frac{Performance_x}{Performance_y} = \text{n}$$

The IRON law of processor performance

$$Processor\ performance = \frac{Time}{Program}$$

$$= \frac{Instruction}{Program} \times \frac{Cycles}{Instruction} \times \frac{Time}{Cycle}$$

(code Size) (CPI) (Cycle time)

Architecture → Implementation → Realization

Compiler Designer Processor designer Chip designer

- **Instructions/Program (Instruction count)**
  - Instructions executed, not static code size
  - Determined by algorithm, compiler, ISA
- **Cycles/Instruction (CPI)**
  - Determined by ISA and CPU organization
  - Overlap among instructions reduces this term
- **Time/cycle (Cycle time)**
  - Determined by technology, organization, clever circuit design

MIPS and MFLOPS

- Used extensively 30 years back.
- **MIPS**: millions of instructions processed per second.
- **MFLOPS:** Millions of Floating-point Operations completed per Second

$$\text{MIPS} = \frac{\text{Instruction Count}}{\text{Exec. Time} \times 10^6} = \frac{\text{Clock Rate}}{\text{CPI} \times 10^6}$$

Problems with MIPS

- Three significant problems with using MIPS:
- So severe, made some one term:
- “Meaningless Information about Processing Speed”
- Problem 1:
- MIPS is instruction set dependent.
- Problem 2:
- MIPS varies between programs on the same computer.
- Problem 3:
  - MIPS can vary inversely to performance!

❑ Consider the following computer:

The machine runs at 100MHz.

Instruction counts (in millions) for each instruction class

Code type-	A (1 cycle)	B (2 cycle)	C (3 cycle)
Compiler 1	5	1	1
Compiler 2	10	1	1

Instruction A requires 1 clock cycle, Instruction B requires 2 clock cycles, Instruction C requires 3 clock cycles.

$$CPI = \frac{\text{CPU Clock Cycles}}{\text{Instruction Count}} = \frac{\sum_{i=1}^{n} CPI_i \times N_i}{\text{Instruction Count}}$$

count cycles

$$CPI_1 = \frac{[(5\times1) + (1\times2) + (1\times3)] \times 10^6}{(5 + 1 + 1) \times 10^6} = 10/7 = 1.43$$

$$MIPS_1 = \frac{100 \text{ MHz}}{1.43} = 69.9$$

$$CPI_2 = \frac{[(10\times1) + (1\times2) + (1\times3)] \times 10^6}{(10 + 1 + 1) \times 10^6} = 15/12 = 1.25$$

$$MIPS_2 = \frac{100 \text{ MHz}}{1.25} = 80.0$$

So, compiler 2 has a higher MIPS rating and should be faster?

- Now let's compare CPU time:

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

$$\text{CPU Time}_1 = \frac{7 \times 10^6 \times 1.43}{100 \times 10^6} = 0.10 \text{ seconds}$$

$$\text{CPU Time}_2 = \frac{12 \times 10^6 \times 1.25}{100 \times 10^6} = 0.15 \text{ seconds}$$

Therefore program 1 is faster despite a lower MIPS!

Computer Performance

"X is N% faster than Y."

$$\frac{\text{Execution Time of Y}}{\text{Execution Time of X}} = 1 + \frac{N}{100}$$

Amdahl's law for overall speedup

$$\text{Overall Speedup} = \frac{1}{(1-F) + \frac{F}{S}}$$

F = The fraction enhanced

S = The speedup of the enhanced fraction

Using Amdahl's law

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

$F = 0.9 \quad S = 10$

$$\text{Overall Speedup} = \frac{1}{(1-0.9)+\frac{0.9}{10}} = \frac{1}{0.1+0.09} = 5.26$$

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

$F = 0.8 \quad S = 1.2$

$$\text{Overall Speedup} = \frac{1}{(1-0.8)+\frac{0.8}{1.2}} = \frac{1}{0.2+0.66} = 1.153$$

Amdahl's Law

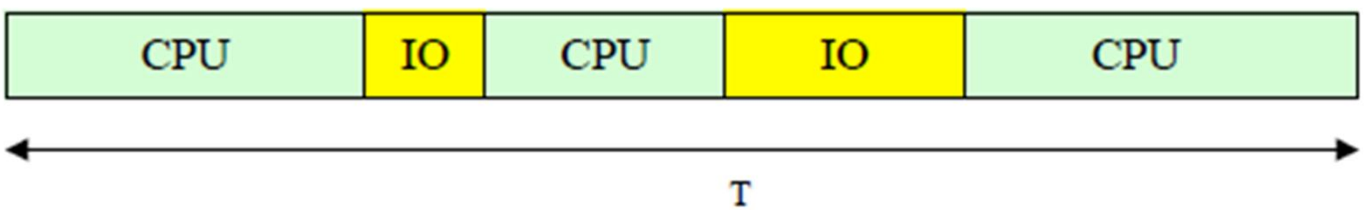
How is system performance altered when some component is changed?

Example 1:

Program execution time is made up of 75% CPU time and 25% I/O time. Which is the better enhancement:

(a) Increasing the CPU speed by 50% or (b) reducing I/O time by half?

Execution model: No overlap between CPU and I/O operations



Program execution time $T = T_{cpu} + T_{io}$

$T_{cpu} / T = 0.75$ and $T_{io} / T = 0.25$

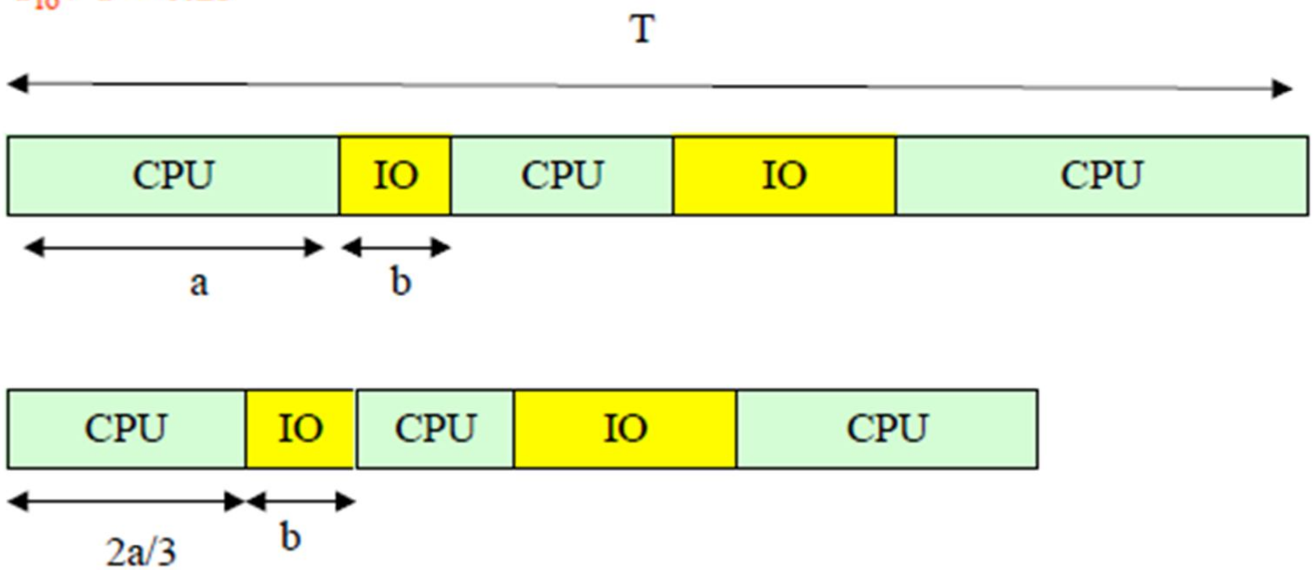
Amdahl's Law

(a) Increasing the CPU speed by 50%

Program execution time $T = T_{cpu} + T_{io}$ $T_{old} = T$

$T_{cpu} / T = 0.75$

$T_{io} / T = 0.25$



Program execution time $T_{new} = T_{cpu} / 1.5 + T_{io}$

$T_{new} = T_{cpu} / 1.5 + T_{io} = 0.75\,T / 1.5 + 0.25T = 0.75T$

For a 50% improvement in CPU speed: Execution time decreases by 25%

Speedup = $T_{old} / T_{new} = T / 0.75T = 1.33$

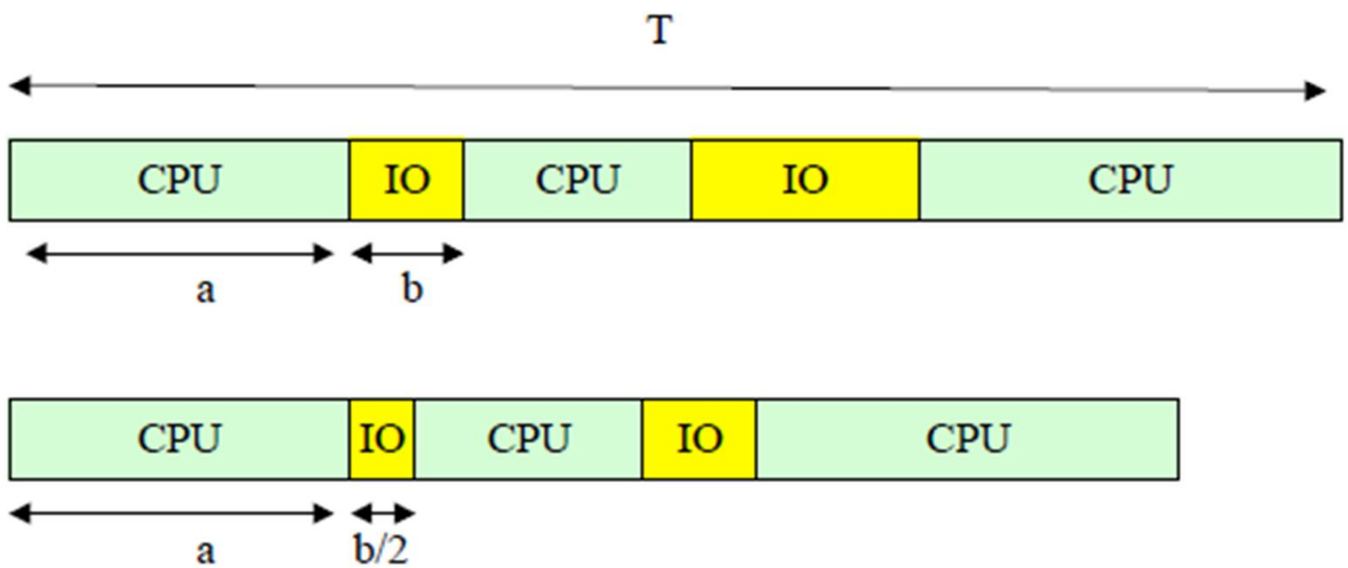
Amdahl's Law

(b) Halve the IO Time

Program execution time $T = T_{cpu} + T_{io}$ $T_{old} = T$

$T_{cpu} / T = 0.75$

$T_{io} / T = 0.25$



Program execution time $T_{new} = T_{cpu} + T_{io} / 2$

$T_{new} = 0.75\,T + 0.25T/2 = 0.875T$

For a 100% improvement in IO speed: Execution time decreases by 12.5%

Speedup $= T_{old} / T_{new} = T/0.875T = 1.14$

Amdahl's Law

Limiting Cases

- CPU speed improved infinitely so T_{CPU} tends to zero
  $T_{new} = T_{IO} = 0.25T$ Speedup limited to 4

- IO speed improved infinitely so T_{IO} tends to zero
  $T_{new} = T_{CPU} = 0.75T$ Speedup limited to 1.33

Problem

- Improving performance
  - Current system
    - * Execution time – 10 sec
    - * Clock speed – 400 MHz
  - New system
    - * Execution time – 6 sec
    - * Clock speed – ?
    - * Number of clock cycles – 1.2 times current system

– Compute the number of clock cycles for current system

*

$$\text{CPU time} = \frac{\text{CPU clock cycles for program}}{\text{Clock rate}}$$

$$10sec = \frac{\text{CPU clock cycles for program}}{400 \times 10^6\text{cps}}$$

* CPU clock cycles for program $= 4000 \times 10^6$

– Compute the clock speed for new system

*

$$\begin{aligned}\text{CPU time} &= \frac{\text{CPU clock cycles for program}}{\text{Clock rate}} \\ 6sec &= \frac{1.2 \times 4000 \times 10^6}{\text{Clock rate}}\end{aligned}$$

*

$$\begin{aligned}\text{Clock rate} &= \frac{1.2 \times 4000 \times 10^6}{6} \\ &= 800 \times 10^6 \\ &= 800\text{MHz}\end{aligned}$$

- Basic performance equation

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

or

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times C_i)$$

  * C_i is the number of instructions of class i
  * CPI_i is the average number of cycles per instruction for class i
  * n is the number of instruction classes

Code sequence	Number of instructions		
	A	B	C
c_1	2	1	2
c_2	4	1	1

Find out the number of instructions for each code sequence, the faster code sequence, and CPI for each code sequence

Number of instructions in sequence $c_1 = 2 + 1 + 2 = 5$

Number of instructions in sequence $c_2 = 4 + 1 + 1 = 6$

Obviously, sequence c_1 executes fewer instructions

$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10$

$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9$

Code sequence c_2 is faster

$\text{CPI} = \dfrac{\text{CPU clock cycles}}{\text{Instruction count}}$

$\text{CPI}_1 = \frac{10}{5} = 2$

$\text{CPI}_2 = \frac{9}{6} = 1.5$

Bench mark sample with problem

- A program runs in 100 sec on a machine, with multiply operations taking up 80 seconds of this time. How much does the speed of multiplication need to improve to get a five-fold increase in code execution?

$$
\begin{aligned}
\text{Execution time after improvement} &= \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected} \\
\frac{100}{5} &= \frac{80}{n} + (100 - 80) \\
20 &= \frac{80}{n} + 20 \\
0 &= \frac{80}{n}
\end{aligned}
$$

There is no amount by which we can improve the performance of multiply to realize a five-fold increase in overall performance

- This is *Amdahl's Law* in computing, or the law of diminishing returns in everyday life
- Opportunity of improvement is affected by how many time the event occurs