Performance Analysis

P. K. Roy
Asst. Professor
Siliguri Institute of Technology

What is it ?

The action or process of performing a task or function.

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

- *Which plane is the best?*

  - which metric do we use?

- *How much faster is the Concorde compared to the 747?*

  Which plane has the longest route?

- *Which plane has the best transporting capacity?*

Performance Analysis

- 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 Instruction Set Architecture (ISA) affect performance?

Computer Performance

- **Response Time or Latency**
  - — 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**
  - — How many jobs can the machine run at once?
  - — What is the average execution rate?
  - — How much work is getting done?

Q1: If we upgrade a machine with a new processor what do we increase?

Q2: If we add a new machine to the lab what do we increase?

Execution Time

Response (or Elapsed) Time:

Total time to complete a task including time spent executing on the CPU, accessing disk & memory, waiting for I/O & other processes, and OS overhead.

CPU Execution Time (simply CPU time):

Total time CPU spends computing on a given task (excluding the time for I/O or running other programs).

CPU time = user CPU time + System CPU execution time

= total time CPU spends only in the program execution + total time OS spends executing tasks for the program

Exercise:

A program have a system CPU time of 20 seconds, a user CPU time of 90 seconds, and a response time of 150 seconds. Calculate the followings –

i) CPU execution time

ii) time for I/O and other processes

Computer Performance

- For some program running on machine A,

  $\textbf{Performance}_X = 1 / \textbf{Execution time}_A$

- “A is n times faster than B“

  $\textbf{Performance}_A \ / \ \textbf{Performance}_B = n$

$$\boxed{\text{Speedup} = n = \frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution Time}_B}{\text{Execution Time}_A}}$$

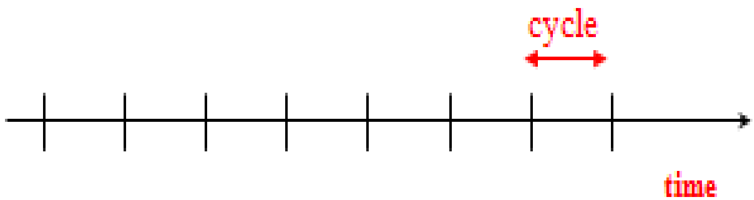
- Problem:
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds
  - How much faster is A compared to B?

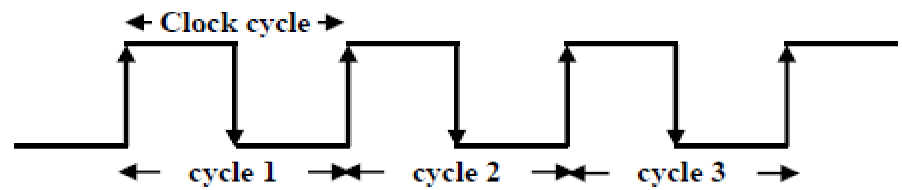
Clock Cycles

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

- Instead of reporting execution time in seconds, we often use cycles
- The CPU clock rate depends on the specific CPU organization (design) and hardware implementation technology (VLSI) used

Clock "ticks" indicate when to start activities (one abstraction):





- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)
- A 200 Mhz. clock has a cycle time $\dfrac{1}{200 \times 10^{6}} \times 10^{9} = 5$ nanoseconds

Cycles/sec = Hertz = Hz
MHz = 10^6 Hz GHz = 10^9 Hz

Nanosecond = nsec =ns = 10^{-9} second
MHz = 10^6 Hz

How to Improve Performance

So, to improve performance (everything else being equal) you can either

__________ the no. of required cycles for a program, or

__________ the clock cycle time or, said another way,

__________ the clock rate.

Fill the blanks with either increase or decrease

How to Improve Performance

- Put it another way:

$$\textit{CPU time} = N_{cycles} * t_{clock} = N_{cycles} / f_{clock}$$

Where

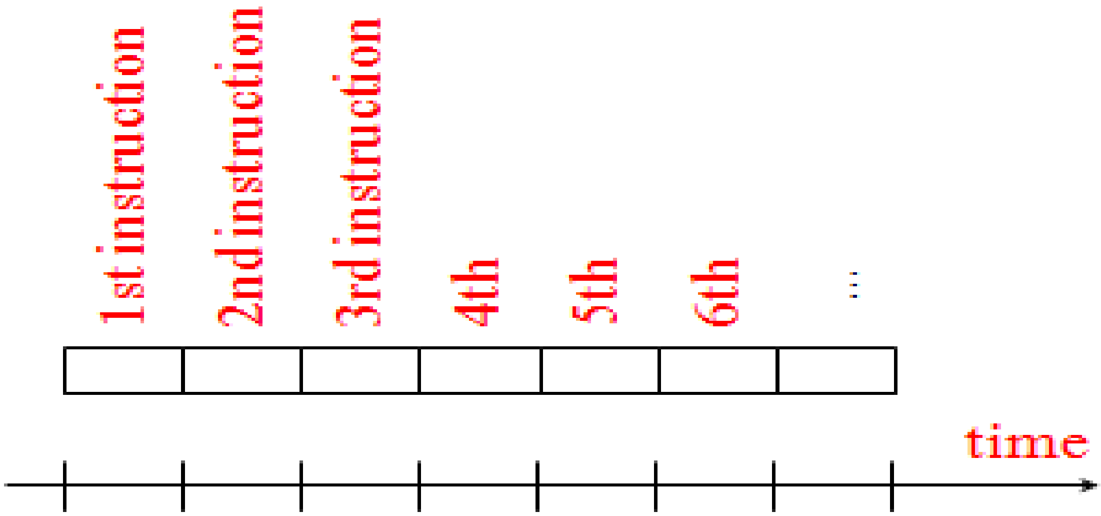
N_{cycles} : *No. of cycles*

t_{clock} : *cycle time or clock time (seconds per cycle)*

f_{clock} : *cycle rate (cycles per second)*

How many cycles are required for a program?

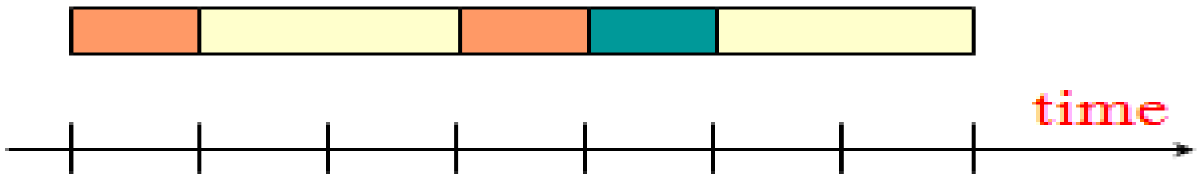
- Could assume that no. of cycles = no. of instructions



This assumption is incorrect,
different instructions take different amounts of time on different machines.

Why?

Different numbers of cycles for different instructions



- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers

- Thus, ***a single machine instruction may take one or more CPU cycles to complete termed as the Cycles Per Instruction (CPI).***

  - **Instructions Per Cycle = IPC = 1/CPI**

- ***Important point:*** *changing the cycle time often changes the number of cycles required for various instructions.*

Understanding 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)

    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?

  - No. of cycles to execute program?
  - No. of instructions in program?
  - No. of cycles per second?
  - Average no. of cycles per instruction?
  - average no. of instructions per second?

- Common pitfall: thinking one of the variables is indicative of performance when it really isn't.

Performance

For a specific program compiled to run on a specific machine (CPU) "A", has the following parameters:

– ***The total executed instruction count of the program.***
– ***The average number of cycles per instruction (average CPI or effective CPI).***
– ***Clock cycle of machine "A"***

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

(This equation is commonly known as the CPU performance equation)

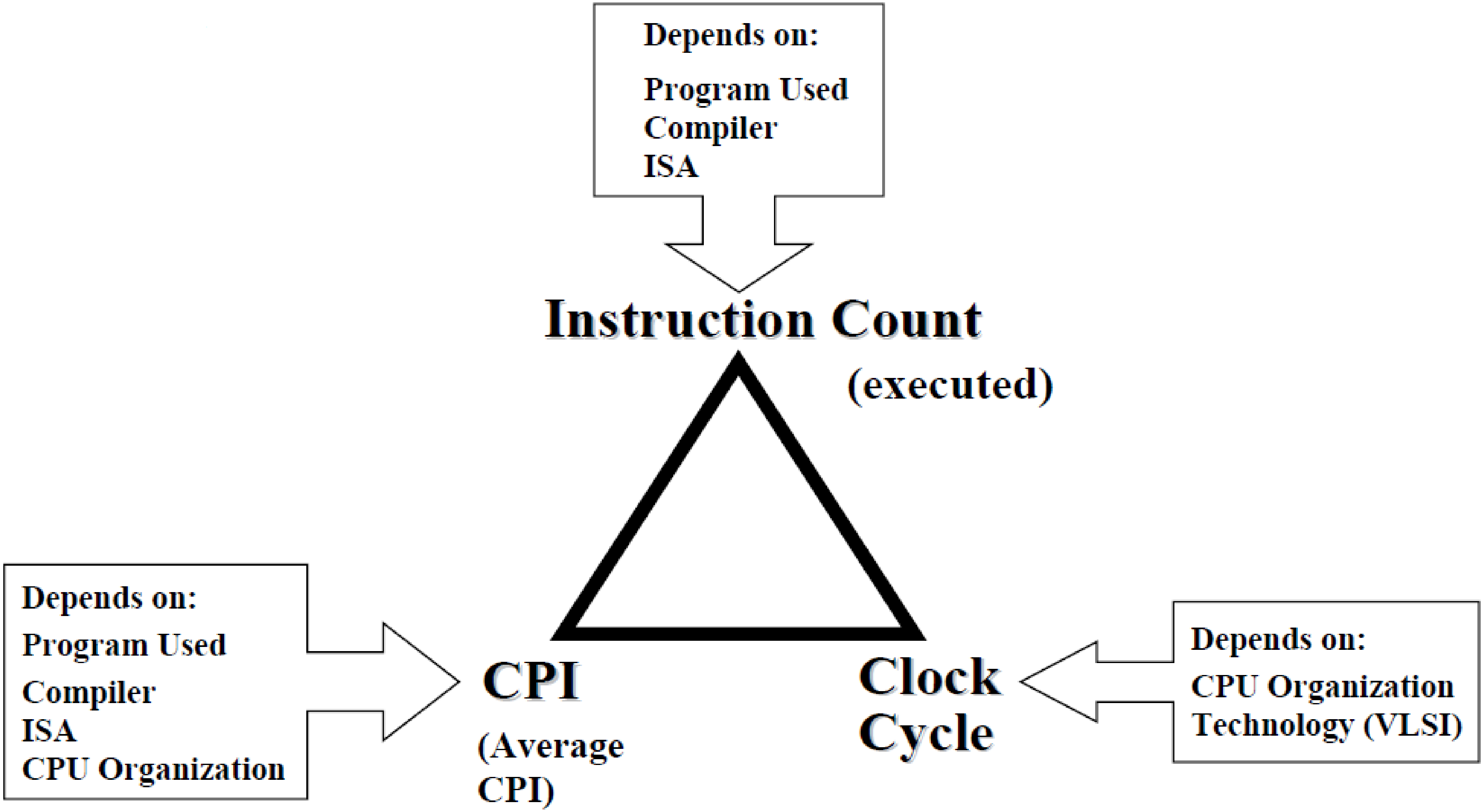
Famous equation:

$$t_{exec} = N_{instructions} \bullet CPI \bullet t_{clock}$$

note that : $N_{cycles} = N_{instructions} \bullet CPI$

CPU Execution Time

$$t_{exec} = N_{instructions} \bullet CPI \bullet t_{clock}$$



CPU Execution Time: Example

- A Program is running on a specific machine (CPU) with the following parameters:
  - Total executed instruction count: 10,000,000 instructions
  - Average CPI for the program: 2.5 cycles/instruction.
  - CPU clock rate: 200 MHz. (clock cycle = 5×10^{-9} seconds) i.e 5 nanoseconds
- What is the execution time for this program:

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

CPU time $t_{exec} = N_{instructions} \bullet CPI \bullet t_{clock}$

$= 10{,}000{,}000 \times 2.5 \times 1 / \text{clock rate}$

$= 10{,}000{,}000 \times 2.5 \times 5 \times 10^{-9}$

$= 0.125$ seconds

Performance Comparison: Example

- From the previous example: A Program is running on a specific machine (CPU) with the following parameters:
  - Total executed instruction count : 10,000,000 instructions
  - Average CPI for the program: 2.5 cycles/instruction.
  - CPU clock rate: 200 MHz.

  Thus: $1/(200 \times 10^6) = 5 \times 10^{-9}$ seconds
- Using the same program with these changes:
  - A new compiler used: New executed instruction count : 9,500,000

    New CPI: 3.0
  - Faster CPU implementation: New clock rate = 300 MHz

  Thus: $1/(300 \times 10^6) = 3.33 \times 10^{-9}$ seconds
- What is the speedup with the changes?

$$\text{Speedup} = \frac{\text{Old Execution Time}}{\text{New Execution Time}}$$

$$\text{Speedup} = (10{,}000{,}000 \times 2.5 \times 5 \times 10^{-9}) / (9{,}500{,}000 \times 3 \times 3.33 \times 10^{-9})$$

$$= .125 / .095 = 1.32$$

or 32 % faster after changes.

Computer Performance Measures : MIPS (Million Instructions Per Second) Rating

For a specific program running on a specific CPU the MIPS rating is a measure of how many millions of instructions are executed per second:

$$\text{MIPS Rating} = \text{Instruction count} \;/\; (\text{Execution Time} \times 10^6)$$
$$= \text{Instruction count} \;/\; (\text{CPU clocks} \times \text{Cycle time} \times 10^6)$$
$$= (\text{Instruction count} \times \text{Clock rate}) \;/\; (\text{Instruction count} \times \text{CPI} \times 10^6)$$
$$= \text{Clock rate} \;/\; (\text{CPI} \times 10^6)$$

Major problem with MIPS rating:

- MIPS rating does not account for the count of instructions executed.
- A higher MIPS rating in many cases may not mean higher performance or better execution time. i.e. due to compiler design variations.

In addition the MIPS rating:

- Does not account for the instruction set architecture (ISA) used.
- Thus it cannot be used to compare computers/CPUs with different instruction sets.

Under what conditions can the MIPS rating be used to compare performance of different CPUs?

- The MIPS rating is only valid to compare the performance of different CPUs provided that the following conditions are satisfied:

- **The same program is used**

  (actually this applies to all performance metrics)

- **The same ISA is used**

- **The same compiler is used**

⇒ (Thus the resulting programs used to run on the CPUs and obtain the MIPS rating are identical at the machine code level including the same instruction count)

MFLOPS (Million FLOating-Point Operations Per Second)

- A floating-point operation is an addition, subtraction, multiplication, or division operation applied to numbers represented by a single or a double precision floating-point representation.
- MFLOPS, for a specific program running on a specific computer, is a measure of millions of floating point-operation (megaflops) per second:

$$\text{MFLOPS} = \text{Number of floating-point operations} \ / \ (\text{Execution time} \times 10^6)$$

- MFLOPS rating is a better comparison measure between different machines (applies even if ISAs are different) than the MIPS rating.
  - Applicable even if ISAs are different
- Program-dependent: Different programs have different percentages of floating-point operations present. i.e compilers have no floating- point operations and yield a MFLOPS rating of zero.
- Dependent on the type of floating-point operations present in the program.
  - Peak MFLOPS rating for a CPU: Obtained using a program comprised entirely of the simplest floating point instructions (with the lowest CPI) for the given CPU design which does not represent real floating point programs.

Exercise

- For the multi-cycle MIPS

  Load 5 cycles
  Store 4 cycles
  R-type 4 cycles
  Branch 3 cycles
  Jump 3 cycles

- If a program has

  50% R-type instructions
  10% load instructions
  20% store instructions
  8% branch instructions
  2% jump instructions

What is the CPI?

Ans: 3.6

Exercise

In a simple m/c with load-store architecture having clock rate 50 MHz, let the instruction frequency be as follows for a program –

Operations	Frequency	No. of clock cycles
ALU	40	1
Load	20	2
Store	10	2
Branch	30	2

Calculate MIPS value for the m/c.

Ans: 31.25

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	Clock rate
Program	X		
Compiler	X	X	
Instr. Set	X	X	
Organization	X		X
Technology		X	

Performance Enhancement Calculations: Amdahl's Law

- The performance enhancement possible due to a given design improvement is limited by the amount that the improved feature is used
- Amdahl's Law:

Performance improvement or speedup due to enhancement E:

$$\text{Speedup(E)} = \frac{\text{Execution Time without E}}{\text{Execution Time with E}} = \frac{\text{Performance with E}}{\text{Performance without E}}$$

original

– Suppose that enhancement E accelerates a fraction F of the execution time by a factor S and the remainder of the time is unaffected then:

$$\text{Execution Time with E} = ((1\text{-}F) + F/S) \text{ X } \text{Execution Time without E}$$

Hence speedup is given by:

$$\text{Speedup(E)} = \frac{\cancel{\text{Execution Time without E}}}{((1 - F) + F/S) \text{ X } \cancel{\text{Execution Time without E}}} = \frac{1}{(1 - F) + F/S}$$

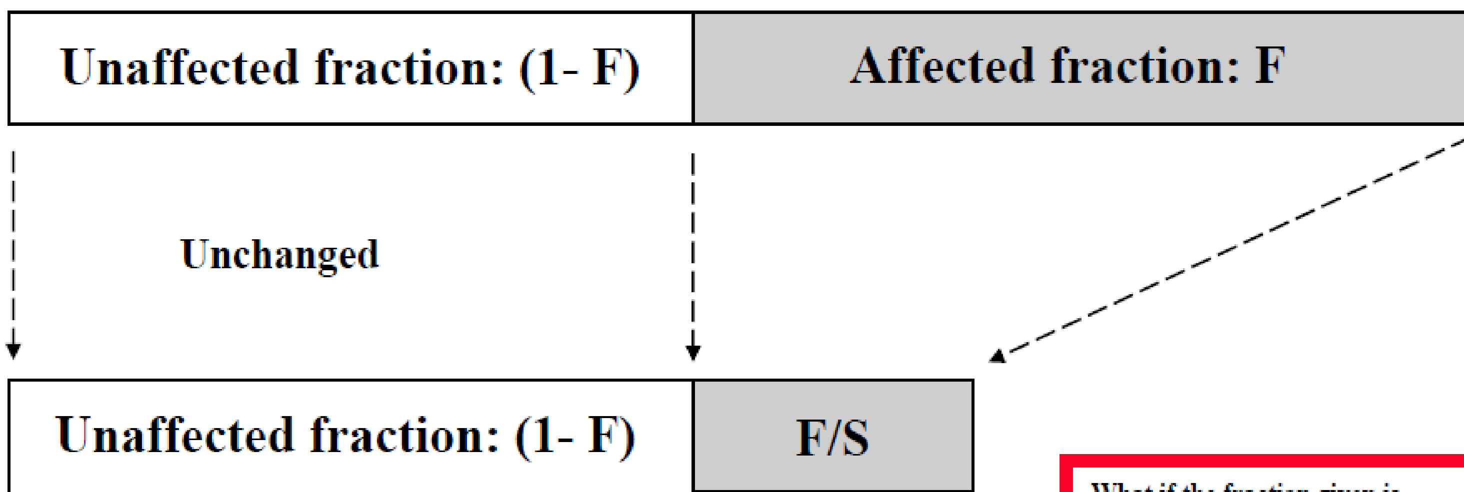
F (Fraction of execution time enhanced) refers to original execution time before the enhancement is applied

Pictorial Depiction of Amdahl's Law

Enhancement E accelerates fraction F of original execution time by a factor of S

Before:
Execution Time without enhancement E: (Before enhancement is applied)

- shown normalized to 1 = (1-F) + F =1



What if the fraction given is
after the enhancement has been applied?
How would you solve the problem?
(i.e find expression for speedup)

After:
Execution Time with enhancement E:

$$\text{Speedup(E)} = \frac{\text{Execution Time without enhancement E}}{\text{Execution Time with enhancement E}} = \frac{1}{(1 - F) + F/S}$$

Amdahl's Law

PARALLEL COMPUTING

Amdahl's Law: The speed-up of a program is given by

$$S_n = \frac{1}{\alpha + (1-\alpha)/n} \leq \frac{1}{\alpha} \text{ when } n = \infty$$

Where, n = number of processors and α = sequential fraction of the program

- If $\alpha = 0$, the maximum speed-up is n. However, the actual speed-up will be much less due to fixed memory size, interprocessor communication and synchronization delays.

Example

if $\alpha = 0.1$, n= 10

$$\text{Speedup} = \frac{1}{0.1 + \frac{0.9}{10}} \approx 5$$

As $n \longrightarrow \infty$ $\text{Speedup} \longrightarrow 10$

Observations of the Amdahl's law

- Small number of sequential operations can significantly limit speedup achievable by a parallel computer

  – This is one of the stronger arguments against parallel computers

- Amdahl's arguments serves as a way of determining whether an algorithm is a good candidate for parallelization

  – This argument doesn't take into account the problem size
  – In most applications as data size increases, the sequential part diminishes.

References

1. Advanced Computer Architecture – Kai Hwang
2. Advanced Computer Architectures – Dezso Sima, Peter Karsuk
3. Computer Architecture & Organization – John P. Hayes
4. Computer System Architecture – M. Morris Mano
5. Computer Organization & Architecture – T. K. Ghosh
6. Computer Organization & Architecture – Xpress Learning