03/04/23

Page No.
Date : / /

Unit-4 → Performance Laws

Amdahl's Law

It is a law governing the speed-up by using parallel processors on a problem, vs using only one serial processor, under the assumption that the problem size remains the same when parallelized

eg – A → Car, B → Bus, C → walk → have to go to Party @ 5pm

could not join

Ily there are some processors which can't run parallel, they have to be sequential

Speedup $= \frac{T(1)}{T(n)}$

$= \frac{\text{Time taken to execute the program using single processor (sequential)}}{\text{Time taken to execute the prog using n no. of processors.}}$

eg – If $T(1) = 1$ sec, $n = 2$, then you take $= 0.5$ sec.

$S = \frac{T(1)}{T(n)} = \frac{1}{0.5} = 2$ (speedup ↑ces by 2 times)

Page No.
Date : / /

Characterizations

① It is based on fixed problem size
(But ideally, if no. of processors ↑, use ↑ the work load - problem size for maintaining efficiency of the system

* Amdahl's law tells that for a given problem size the speed up doesnot ↑se linearly as the no of processors.
The speedup tends to become saturated - doesnot ↑se after a point.

* It states that a small portion of the prog, which cannot be parallelized (serial part) will limit the overall speedup.

Computation problem = Serial part + Parallel part

Let 'f' be the fraction of operations in a computer that must be performed sequentially where $0 \le f \le 1$. The max. speedup achievable by a parallel computer with n processors is :-

Learn

$$S(n) \le \frac{1}{f + \frac{(1-f)}{n}}$$

f → serial part, $\frac{(1-f)}{n}$ → parallel part

$$\frac{n}{1 + (n-1)f}$$

Page No.
Date : / /

Quest 70% of a program's execution time occurs inside a loop that can be executed in parallel and rest 30% in serial. What is the max speedup we should expect from a parallel version of the program executing on 8 CPUs.

Solⁿ

$f = 0.3 =$ serial part
$n = 8$

$$S(8) \le \frac{8}{1 + (0.3 \times 7)}$$

$$S(8) \le 2.58$$

For 80% : 20%

speedup = 3.33

⇒ speedup ↑ses when processors are doing parallel processing

—x—

Ques2 Let a program have 40% of its code enhance to yield a system speedup 2.3 times faster. ~~than~~ What is the factor of improvement of the portion enhanced?

Ans2 40% enhanced → 2.3 times faster
(use formula of next page)

$$S = \frac{1}{(1-0.4) + \left(\frac{0.4}{2.3}\right)} = 1.292$$

Page No.
Date : / /

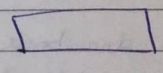
Ques Let a program have 40% of its code enhanced to run 2.5 times faster. What is the overall system speedup?

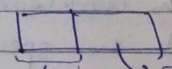
Zahra ppt

T = execution time before improvement
p = fraction of part benifitted from improvement

06/04/24

Let's — workload is fixed in Amdah's

CPU old  1 (takes 1 cycle)

CPU new 

1 − Fenhanced (unhenced) ← → Fenhanced (fraction enhanced portion of data)

takes same time (1 − F)

$\frac{F}{S}$ → at what scale it will be enhanced?

time taken by enhance part

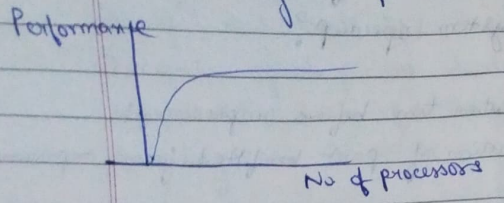
Prachi Prachi Prachi

$$\boxed{S_{all} = \frac{t_{old}}{t_{new}} = \frac{1}{(1-F) + \frac{F}{S}}}$$

unenhanced, enhanced

Page No.
Date : / /

Performance becomes saturated as you keep on increasing the processors, as overhead also ↑s



But ideally, workload also increases as the no. of processors ↑se

This was the drawback of Amdahl's bcz it kept the workload constant and only ↑sed the speedup.
not dynamic workload

—— x ——

Gustafson's Law

- used for scaled programs
- we are scaling for higher accuracy
- Fixed time speedup (speedup doesnot change, ~~workload~~ workload & no. of processor changes)

$$T_p = t_s^* + t_p^*$$

t_s^* → computational time needed for sequential part

t_p^* → computational time needed for parallel part

- workload changes here

$$\boxed{S = f^* + N \cdot (1 - f^*)}$$

S → speedup, N → no. of processors, f^* → dynamic workload

Page No.
Date : / /

Ques 1- 10% of time on a 32 processor machine is spent on one single processor. Compute the scaled speedup using Gustafson's Law.

$f^* = 0.1$, $N = 32$

$S = 0.1 + 32(1 - 0.1)$

$\boxed{S = 28.9}$

—— x ——

Sun & Lee → Nahi karna ✗

—— x ——

Performance Benchmarks

Benchmarking?
It is measuring how the performance of something varies as you change the parameters.

In HPC systems, things that can vary are →

1. no. of parallel processes
2. no. of threads
3. distribution of processors / threads
4. i/p parameters.

→ All these result in performance variations.

Benchmarking - used for scaling
- Dhrystone Benchmark
- Whetstone Benchmark

Page No.
Date : / /

Dhrystone Benchmark

- This benchmark is used to measure and compare the performance of different computers, or the efficiency of the code generated for the same computer by different compilers.

- code gives ~~same~~ approx same performance irrespective of the compiler.

★★ - It uses no floating point operations.

Advantages
1. You can compare results of processors of different architecture.
2. Easy to compile & port (works on micro controller)

Disadv - ~~from PPT~~
1. uses a limited set of features
2. Doesn't allow you to use all the capabilities of the processor.
3. Strange code
4. The results are very dependent on compiler optimizations.

Page No.
Date : / /

Whetstone Benchmark

- It is used for evaluating the performance of both int & floating point nos. in CPU.

——— Unit - 4 done ———

06/04/24

Unit - 5
Memory & Cache Coherency
(Completely theoretical)

Consistency
- needs to be maintained when sharing data.

what?

CPU	Cache	MM
	x=10	x=5

Not consistent X

Memory Consistency Model
A Consistency Model refers to the d. ...

from PPT

CPU	Cache	MM
P1	10	5
	x=5	

P2 → read x=10 — x=5 — inconsistent

⇒ next read should be from where then was last write to avoid inconsistency