Qus 1 | 21820 6

Amdahl's LAW for fixed workload :-

In many practical applications that demand a real-time response, the computational workload is often fixed with a fixed problem size. As the number of processor increases in a parallel computer, the fixed load is distributed to more processors for parallel execution or, the minimal turnaround time is primary goal

Fixed-Load speedup :- speed obtained for time critical applications is called fixed-load speedup. Tranditional formulations for speedup, including Amdahl's law are all based on a fixed problem size and thus fixed load.

→ Case 1 :- Do $P_i < n$ Assume all processors are used to execute W_i exclusively. The execution time of W_i is [7]

$$t_2(n) = \frac{W_i}{i\Delta}\left\lceil \frac{i}{n} \right\rceil$$

→ THUS response time will be :-

$$T(n) = \sum_{i=1}^{m} \frac{W_i}{i\Delta}\left\lceil \frac{i}{n} \right\rceil$$

case 2 :- If $i > n$ then $t_i(n) = t_i(n) = t_i(\infty) = \frac{W}{i\Delta}$

→ Now we can define fixed-load speedup factor as ratio of $T(1)$ to $T(n)$

$$S_n = \frac{T(1)}{T(n)} = \frac{\sum_{i=1}^{m} W_i}{\sum_{i=1}^{m} \frac{W_i}{i}\left\lceil \frac{i}{n} \right\rceil}$$

→ A number of factors we have ignored may lower the speedup performance. These include communication latencies caused by delay memory access, interprocess communication over a bus or network or OS. overhead and caused delay by interrupts.

→ Let $Q(n)$ be the lumped sum of all system overheads on an n-processor system.

$$S_n = \frac{T(1)}{T(n)+Q(n)} = \frac{\sum_{i=1}^{m} w_i}{\sum_{i=1}^{m} \frac{w_i}{i}\left\lceil \frac{i}{n} \right\rceil + Q(n)}$$

→ The $Q(n)$ is application as well as machine dependent.

→ Amdahl's law implies that the sequential portion of the program W_i does not change with respect to machine size (n). However the parallel portion is evenly executed by n processors resulting in a reduced time.

In Amdahl's law when number of processors increase the load on each processor decr.

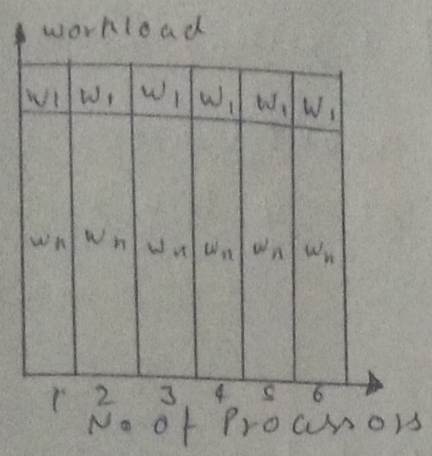


figure: Fixed workload

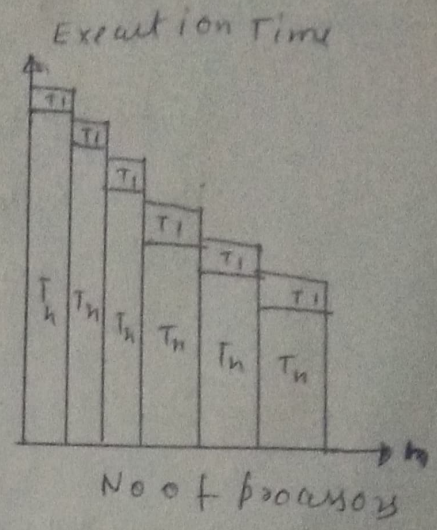
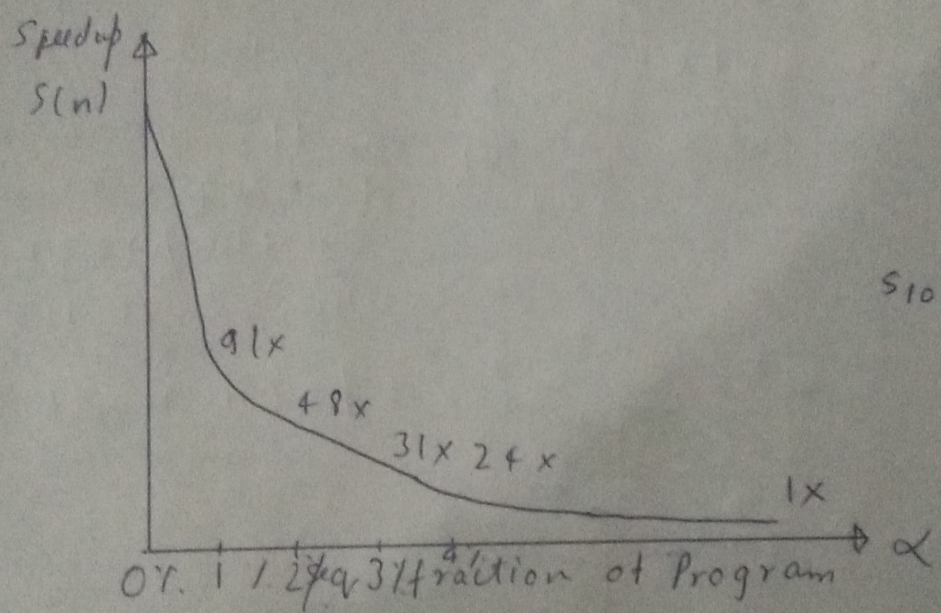


figure: Decr execution time



$$S_{1024} = \frac{1024}{1 + 1023\alpha}$$

figure: Fixed-load speed up model Amdahl's law.

Sequential Bottlek

→ The speed curve drops rapidly as α increase. This means with a small percentage of the sequential code, the entire performance cannot go higher than $1/\alpha$.

α → the sequential bottleckneck.

→ The problem of seq bottleneck cannot be solved just by ↑ the number of processors in a system. The real problem then lies in existence of seq trac of code

Gustafson's law for Scaled Problems :-

→ One of the Problems with Amdahl's law is that the workload can not scale match the avalable computing power as the machine size increases.

→ Scaling for High accuracy :-

'Time critical application provided the major motivation leading to the dev. of the fixed-load speed up model and Amdahl's law. Machine size upgraded to obtain more computing power, we may want to ↑ the prob. size in order to create a greater workload, producing more accurate soln.

→ Many advance scinfific modeling simulation application eg the very large scale matrix problem based on Diffential eqn.
With such problem scaling of course we demand more computing power to yield the some execution time.

→ "This problem scaling for accuracy has motivated Gustafson to develop a fixed-time speedup model. The scaled problem keeps all the increased resources busy". Resulting better utilization ratio.

Gustafson's Law :-

→ Fixed time speedup was originally dev by John Gustafson for a special parallelism profile with $w_i = 0$ if $i \neq 1$ and $i \neq n$ similar to Amdahl's law and assuming $Q(n) = 0$

$$S'_n = \frac{\sum_{i=1}^{m'} w_i'}{\sum_{i=1}^{m} w_i} = \frac{w_1' + w_n'}{w_1 + w_n} = \frac{w_1 + n w_n}{w_1 + w_n}$$

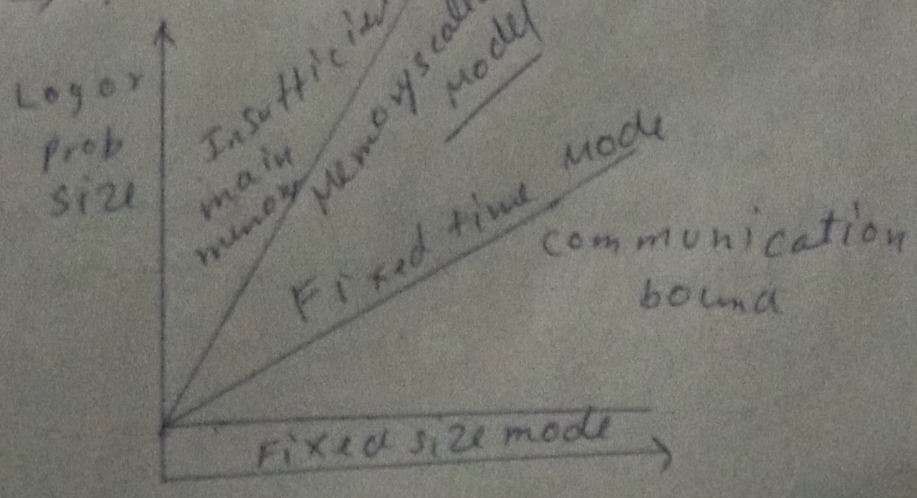
→ $w_i' = n w_n$ & $w_1' + w_n'/n$

→ corresponding to fixed time condn. the parallel work load w'_n has been scaled to n time w_n in a linear fashion

→ The relation of a scaled workload to Gustafson's scaled speedup. In fact Gustafson's law can be restate as follow in terms of $\alpha = w_1$ and $1 - \alpha = w_n$

$$S_n' = \frac{\alpha + n(1-\alpha)}{\alpha + (1-\alpha)} = n - \alpha(n-1)$$

→



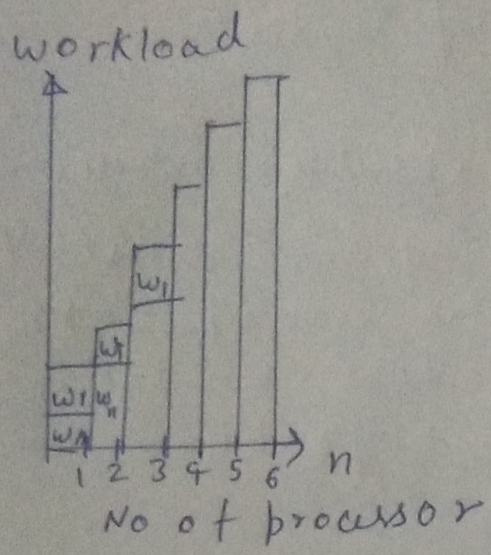
Memory Bounded speed up Model:-

→ THE memory bounded speed up model generalizes Amdahl's law and Gustafson's law to maximize the use of both CPU & memory capacities. The idea is to solve largest problem, limited by memory space

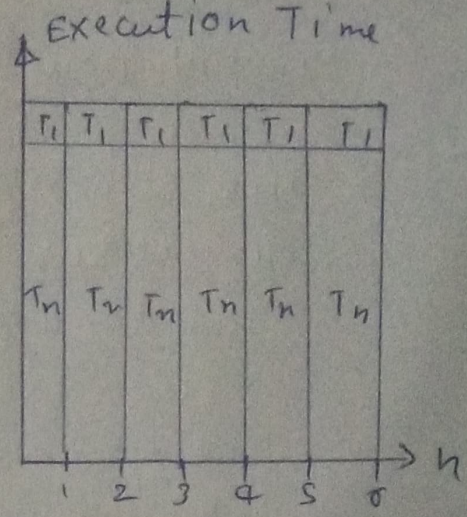
→ Memory bound Problems:-
Large scale satic or engineering computations often require large memory space. In fact many applications of parallel computers are memory-bound rather than CPU bound or I/O bound.

→ This is specially true in multicomputer system using distributed memory. THE local memory attached to each node is relatively small.

→ A memory bound model was dev under the idea that "solve the largest ~~problem~~ possible problem limited only by the available memory capacity. This Model may result in an incr in execution time to achieve scalable performance.



(a) scaled workload



(b) fixed execution Time

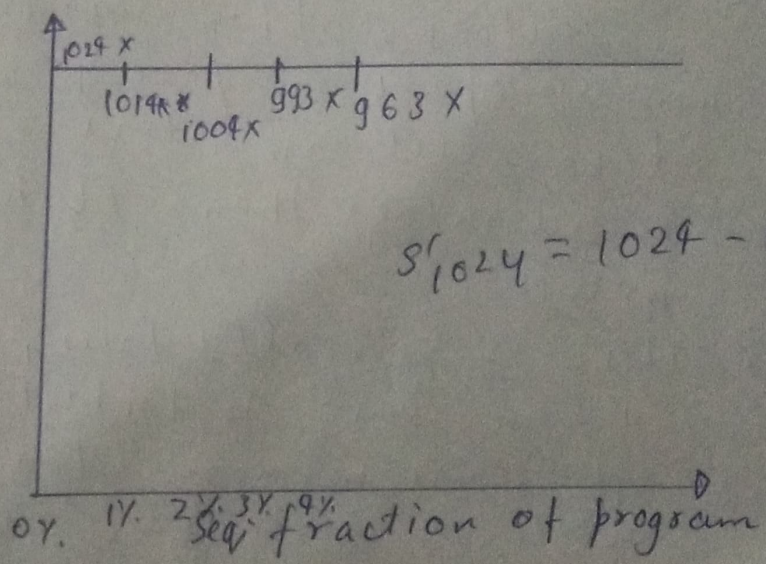


figure: Fixed time speed up model and Gustafson's la

Fixed Memory speedup:-

M → memory sep of a given problem

W → computational workload.

$$W = g(M) \;\&\; M = g^{-1}(W)$$

↳ inverse of g

→ In a multicomputer the total memory capacity increases linearly with number of nodes available.

we can write

$$W \doteq \sum_{i=1}^{m^*} W_i$$

m^* → is max of scaled problem. D.O.P. as the workload for seq^n execution of the program in single node.

$$W^* = \sum_{i=1}^{m^*} W_i^* \quad \text{(for n nodes)}$$

$$\longrightarrow S_n^* = \frac{\sum_{i=1}^{m^*} W_i^*}{\sum_{i=1}^{m^*} \frac{W_i^*}{i}\left\lceil \frac{i}{n} \right\rceil + Q(n)}$$

→ we can assume $g^*(nM)$ where nM is increased memory capacity for n node multicomputer.

→ The factor G(n) reflects incr^n in workload as memory increases n times

$$S^*_W = \frac{W_1^* + W_n^*}{W_1^* + W_n/n}$$

we assume $g^*(nM) = G(n)g(M) = G(n)W_m$

as $W_n = g(M)$ & g^* is a homogenous func n

$$\Rightarrow \quad S^*_n = \frac{W_1^* + W_n^*}{W_1^* + W_n^*/n} = \frac{W_1 + G(n)W_n}{W_1 + G(n)W_n/n}$$

There are 3 special cases

<u>Case 1</u> $G(n) = 1$

This corresponds when problem size is fixed THUS speed up becomes equal to Amdahl's law.

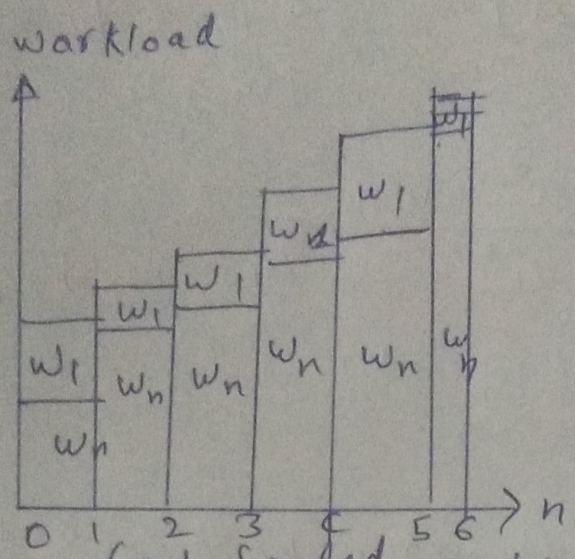
<u>Case 2</u> $G(n) = n$

This applies to the case where workload incr n times.

Gustatson's law

<u>Case 3:-</u> $G(n) > n$

Here computational workload incr faster that the memory req^n. THUS the fixed memory model will likely give a Higher speedup than fixed time speedup.



(a) Scaled workload

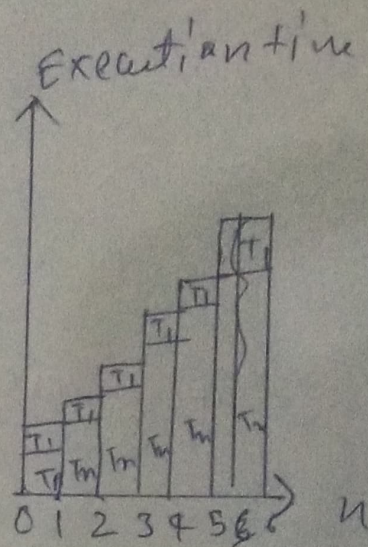


figure speed up model using fixed memory

Ques 2

→ A linear pipeline processor is a cascade of processing stages which are linearly connected to perform a fixed funcn over a stream of data flowing from one end to another.

→ A linear pipeline is constructed with k processing stages.

→ Depnting on the control of data flow along the pipeline we mode linear pipeline in 2 categories:-

① Asyncronous Model:-

→ Data flow b/w adjacent stages in an asynchronous pipeline is controlled by a Handshaking protocol.

→ when stage S_i is ready to be ~~submit~~ transmit, it sends a ready signal to stage S_{i+1}.

→ After stage S_{i+1} xcn the incoming data it returns an acknoledge signal to S_i.

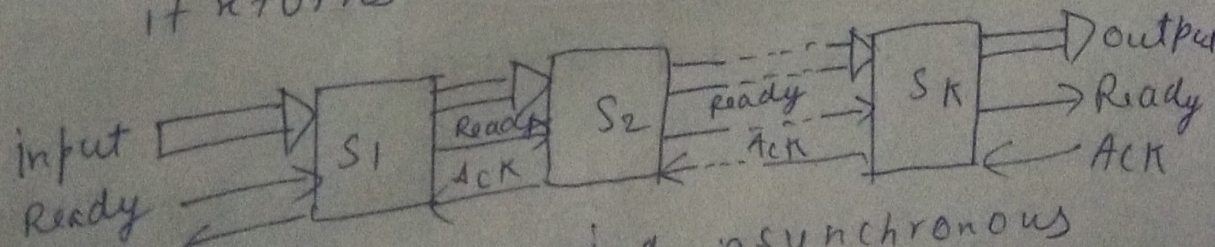


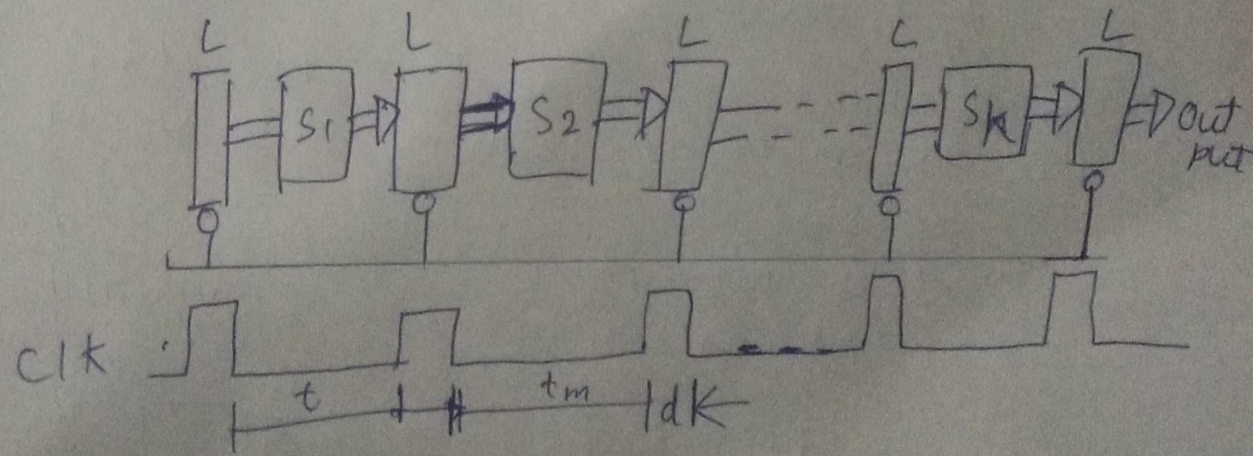
figure: showing asynchronous pipeline model.

Syncronous Model ⇒

→ Synchronous pipelines are illustrated in figure below. Clocked latches are used to interfacing between stages.

→ The latches are made with Master slave flip flops, which can isolate inputs from outputs. upon arrival of a clock pulse, all latches transfer data to next stage simultaneously.

→ The pipeline stages are combinational logical ckt. It is desired to have appⁿ equal delays in all stages. These ~~latche~~ delays determine the clock period and thus the speed of the pipeline



(b) Syncronous pipeline mode

S_i → stage i

L → Latch

t → ~~time~~ clock period

d → latch delay

Stages	1	2	3	4
S_1	X			
S_2		X		
S_3			X	
S_4				X

Rep o table o f 4 stage linear pipeline

Ack → Ackⁿ signal

Syncronous Model.-

→ The pipeline stages are combinational logic ckt. It is desired to have approx equal delay in all stages. These delays determine the clk period & thus speed of pipeline.

successive tasks or operations are initiated one per cycle to enter the pipeline. One the pipeline is filled up one result emerges for each additional cycle.