Date: 04/02/2025 Day: Tuesday.

Parallel & Distributed Computing

yasir.nasir@pucit.edu.pk

Book:- Intro to parallel computing by grama & Gupta 2nd edition.

* PCB (shared memory) Concepts
* Threads, process

Intitution:- Parallel

A = 1, 2, 3, 4, 5, 6

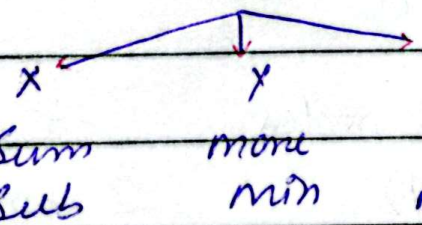
B = 2, 4, 6, 8, 10, 12

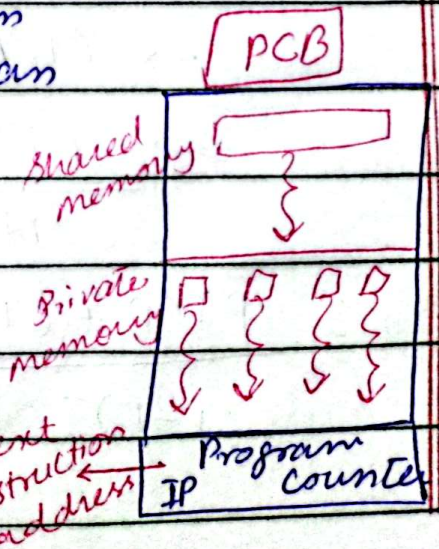
Sum = loop on list

Sum = A[0] + B[0] so we need 6 ms if it takes 1 sec on each.

Better: make 6 threads in a process if you run that all in parallel it only takes 1ms.

Distributed connected all.





Date: / /20 Day:

Lecture #02

Memory unit
Controlled unit
I/P or I/O units.
→ Moore's law.
→ multiple cores
Drawback :- frequency high → High heat.
→ cache
Instruction level
Process level
Memory computation } optimization value for design system
→ Scalability
→ Load Balancing
→ Speed up
→ Display output

parallelized : $PT(1)$

for single node time → $(1-P)+T(1)$

for multiple N nodes

→ $(1-P)T(1)$

→ $\frac{PT(1)}{N}$

$$= \frac{PT(1) + (1-P)T(1)}{\frac{(1-P)T(1) + PT(1)}{N}}$$

Date: 1/20 Day:

$T(1)\,(P+1-P) = 1$

$T(1)\,(1-P+\frac{P}{N}) \quad (1-P)+P/N$

Portion that can be parallelized.

For $N=10$, $P=0.7$

$= \frac{1}{(1-0.7)+\frac{0.7}{10}} = \frac{2.70}{3.257}$

for $N=100$

$\frac{1}{0.3+0.007} = \frac{1}{0.307} = 3.257$

for $N=1000$

$\frac{1}{0.3+0.0007} = \frac{1}{0.3007} = 3.325$

for $N=30$

$\frac{1}{0.3+0.233} = \frac{1}{0.3233} = 3.093$

$P=0$, that means your assumption is true that nothing works in parallel.

$\frac{1}{(1-0)+\frac{0}{N}}$ So no speed up achieved.

ideal case $P=1$

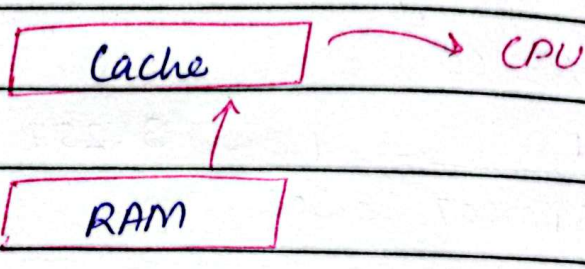
Date: 12/03/2025 Day: Tuesday

Lecture # 03.

Cache : 512 Kb – 128 Mb
← Range →

Data comes from CPU / GPU to cache.
✗ Not from RAM to ~~cache~~ CPU.

RAM : 45b → 128

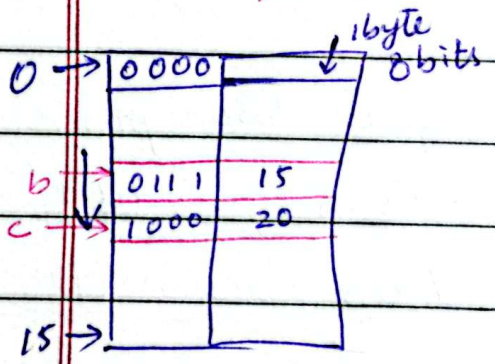


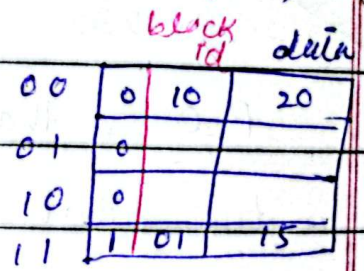
manager
OS + compiler

SSD , I/O operation

→ Direct Mapping

RAM (4 bit address space) → Cache (2 bit addressing space)





4-5 bytes

RAM = 16 bytes size

⇒ example:

b = 15
c = 20
a = b + c

Request :
• fetch data from addr
•
addresses

d = 2b – c

Date: __/__/20 Day: _____

b = [block id][address] = [0][11]

c = [10][00]

- cache miss because in cache we have no values written. It can cause cache overhead.
  ↳ It takes time.

- cache hit — MMU memory control unit map the data on cache and again address Request for data and data will hit this time. (Data Place then hit)

In case of 2b-c the cache will not miss due to we have data in cache.

Offset Byte :- → 32 bit architecture use 32 bit address Bus.

↓ 32 bits

Due to the architecture of computer. 4 bytes

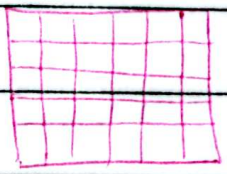
00 01 10 11

1 word

→ cache contain 1 word in memory address, that equals to 4 bytes according to the architecture.

Python Compiler :-

2D Matrix



5 × 6 matrix

(b)

Column major = C1 C2 C3 C4

Row major = R1 , R2 , R3 , R4

↳ C compilers used Row major

Date: 1 /20 Day:

- Assumption = Worst Case

Each row has 1 word.

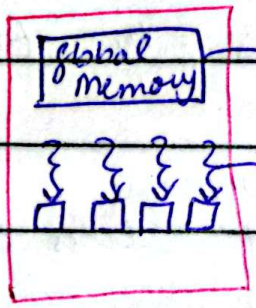
GIL global Interpreter lock.

↳ Interpreter that adds security and limitation → If one thread is in pipeline then other thread can't enter in pipeline.

So no parallelism.

I/O can acheive parallelism due to bound on Data side.

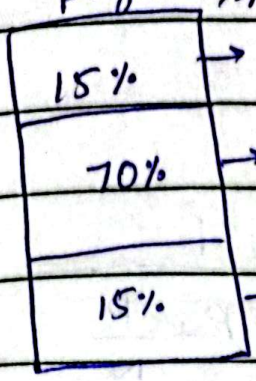
Multiprocessing (CPU Bound.)



• No memory is shared between 2 simultaneous processes that are running.

Date: 1/1/20 -Parallelize- Day:

program space - Amdahl's Law -



15% → input operation serial (sequential)

70% → list concepts (parallelizeable). used here

15% → output operation Serial

* for single thread T(1)

unit Time
1ms/1m/1s

- parallel portion = p
- serial portion = 1-p

time taken by p portion:

$$p.T(1)$$

time taken by $(1-p) = (1-p).T(1)$

* for N nodes / threads T(N)

program will run in single due to its sequential behaviour.

time for serial portion = $(1-p).T(1)$

time for p portion = $\frac{pT(1)}{N}$

$$T(1) = pT(1) + (1-p)T(1)$$

$$T(N) = (1-p)T(1) + \frac{pT(1)}{N}$$

$$\text{Speed up} = \frac{10ms}{2.5} = 4$$

$$= \frac{T(1)}{T(N)}$$

Date: / /20 Day:

$$= \frac{PT(1) + (1-P)\,T(1)}{(1-P)T(1) + \frac{PT(1)}{N}}$$

$$= \frac{T(1)\,[P - P + 1]}{T(1)\,[1 - P + \frac{P}{N}]} = \frac{1}{1 - P + \frac{P}{N}}$$

$N = 10 \Rightarrow 2.70$

$N = 30 \Rightarrow 3.09$ → major diff → 3 times better

$N = 100 \Rightarrow 3.257$ 3.2 times better

$N = 1000 \Rightarrow 3.325$

Date: 24/02/2025 Day: Tuesday

Book: Intro to 11 computing by Grama & Gupta 2nd edition.

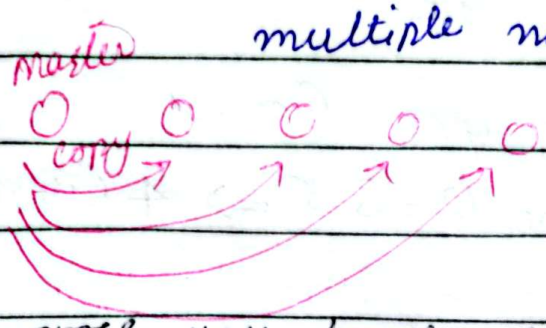
Ch # 04: Basic Communication Operations.

Among processing nodes

↓

Thread / Process

either in one node or accross multiple nodes



Suppose you have 64 GB of Data in memory (1d, linear) Sum it element

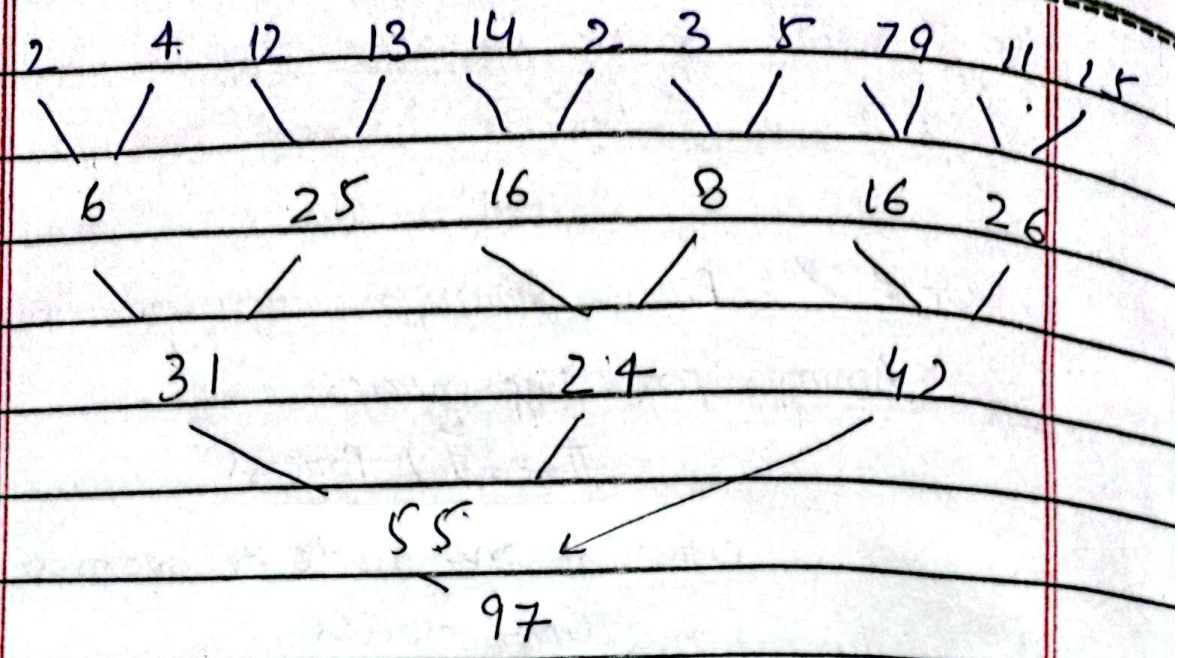
$1G = 2^{30}$, $64 = 2^6 \Rightarrow 2^{36}$

Thread is workflow. flow will stop either when workflow ends or flow joins.

for 'n' processing nodes:-

All to one Reduction

Date: / /20 Day:

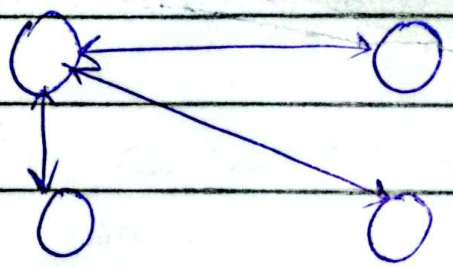


$\lceil \log_2 n \rceil$ => upper Bound

Total $n = 12$ $\log_2 12 = 3.5 \Rightarrow 4$ (upper Bound)

(transfer speed)

6+GB Connection Bandwidth - 2 Gbits/sec



each node can have 2^{20} Threads (nodes.)

- Calculate the time considering each sum can take 1 sec to execute.

1- How much data to send to each node?

2- How much processing of each node?

3- Bottleneck?

4- Send to master Node?

Date: __/__/20 Day: _____

$\frac{64GB}{4} = 16GB$ to each node.

One to all broadcast. As of now we have not studied customized data.

[Time taken to copy to each node of 64 GB data] = $\frac{64GB \times 2^{30} \text{ bits}}{2G \text{ bits}}$

$\frac{64 \times 8}{2} = 512$

$= 256$ sec

$2 \overline{)512}\ 256$

2

11

10

12

12

0

⇒ Suppose 1B is Representing 1 value of data 16 GB on each node 2^{20} threads of each node $2^4 \times 2^{30} \Rightarrow 2^{34}$

$\frac{2^{34}}{2^{20}} = 2^4 = 16$

Limitation = 2^{20} Threads

Required = $\frac{16G}{2} = 8G$ threads

Chunks ⇒ $\frac{8G}{2^{20}} = \frac{8 \times 2^{30}}{2^{20}} = 8 \times 2^{10} = 8k$

2^{13} chunks

→ Reason why divided by 2

log tree 1st step (n/2)

Date: / /20 Day:

For ease suppose we have 8G threads

$\Rightarrow \log_2 x = \log_2 16G$

$= \log_2 2^4 \times 2^{30}$

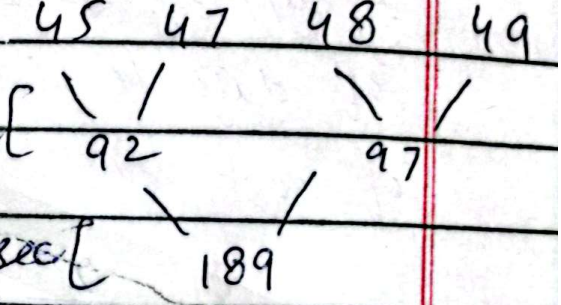
$= \log_2 2^{34}$

$= 34$ units

So now $256s + 34s$

Now single value will be Returned to master in negligible time so last time sum operation takes 2s.

Why 2 sec?



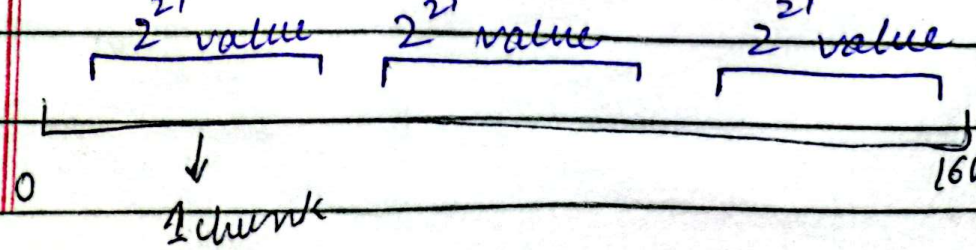
$256s + 34s + 2s = 292s$

so this whole operation will take 292s in whole 8G thread case.

In linear operation then 64Gs be the time.

Now, will come back to 2^{20} threads

We have 16GB of Data at each node.



Date: __/__/20 Day: _____

why 2^{21} value? as 2^{20} threads at each thread process 2 values. So, $2^{20} \times 2 = 2^{21}$ values

$$\frac{16G}{2^{21}} = \frac{16 \times 2^{30}}{2^{21}} = 2^{34-21} = 2^{13}$$

So total no of chunks are 2^{13}.

Let's calculate time for single chunk.

$$n = 2^{21}$$

$\log_2 2^{21}$ = (21s) for each single chunk

⇒ for 2^{13} chunks × 21 seconds.

$= 2^3 \times 2^{10} \times 21s$

$= 8 \times 21 ks = (168 ks)$

So now lastly,

256 + 168k + 2 + 13

256 + 1720 + 2 + 13

172290 s + 13

as 2^{13} trees so also sum them.

Remember:-

each chunk makes it own tree.

so 2^{13} trees

that's why we did $2^{13} \times 21$ sec.

Date: 04/03/2025 Day: Tuesday.

Parallel & Dist Computing -

⇒ Matrix Vector Multiplication.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}_{3\times3} \begin{bmatrix} 4 \\ 2 \\ 3 \end{bmatrix}_{1\times1} = \begin{bmatrix} 4+4+9 \\ 16+10+18 \\ 28+16+27 \end{bmatrix} = \begin{bmatrix} 17 \\ 44 \\ 71 \end{bmatrix}$$

2D or MultiDim Data can be applied Row wise or column wise.

① Data Distribution

$$\begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix} * 4 + \begin{bmatrix} 2 \\ 5 \\ 8 \end{bmatrix} * 2 + \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix} * 3$$

② single value Broadcast

$$= \begin{bmatrix} 4 \\ 16 \\ 28 \end{bmatrix} + \begin{bmatrix} 4 \\ 10 \\ 16 \end{bmatrix} + \begin{bmatrix} 9 \\ 18 \\ 27 \end{bmatrix}$$

logarithmic time

③ Reduction

⇒ Prefix - Sum :- exclusive / inclusive

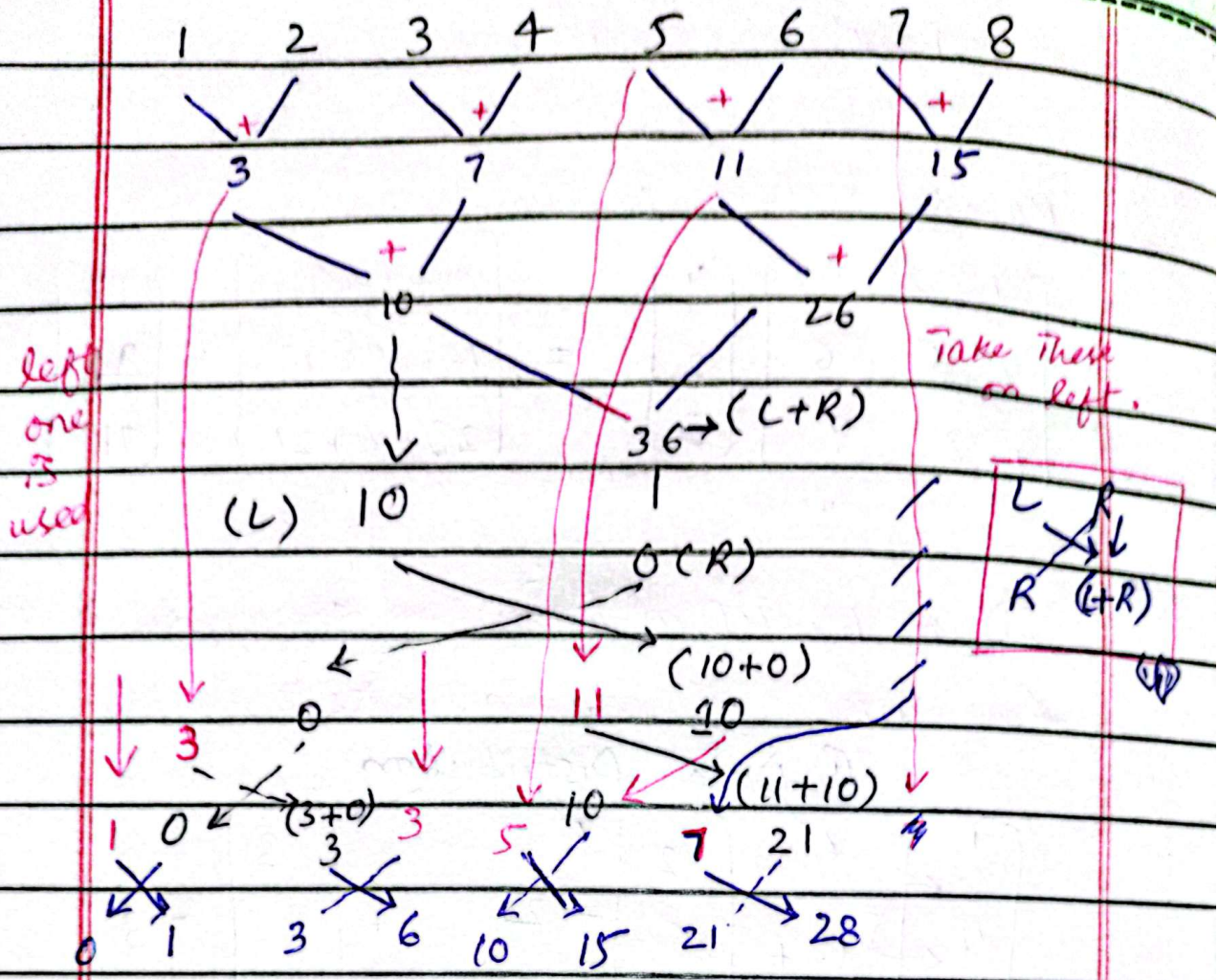
exclusive: 1 2 3 (4) 5 6 7 8

Sum = 6

inclusive: 0 1 3 6 10 15 21 28 = (n)

Time complexity

Date: 1/20 - Brooch's Algorithm - Day:



* (L+R) wali branch ko left pr drag kr lein to tree thik bane ga.
I have made a mistake at 36.

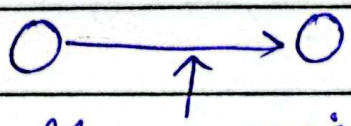
Date: 20/03/2025 Day: Thursday.

Parallel & Dist Comp.

Parallel Application and performance Aspects :-

- Granularity

  Communication + computation

  * 

  If communication overhead is ↑.

  Computation ↓

  Fine-grained form

  * Vice Versa Coarse-Grained form.

→ less opportunity for performance enhancement

factors to improve :-

↳ Multiple Threading (If possible) ✓

↳ Multiple Processing ✓

↳ Distributed (Communication Cost)

↳ Algorithm / logic

Date: / /20 Day:

Requirement achieved with less processors (efficient) ✓.

→ Automatic Compilation

Automatic Compiler Based
↓
Detect pattern and judge to execute it in parallel

→ Manual C: logic of program driven manually.

↙ Identifying ↓ Implementing ↘
Iterative process

→ In C language (# pragma OpenMP) = Instruction
Compiler Instruction ↑
• work according to that Instruction

```
#pragma omp parallel for
for(        )
{
  ≡
}
```

↓

⇒ Overlap Communication with Computation ↑
• encryption that secures data & Reduce Data

Date: 21/03/2025 Day: Friday.

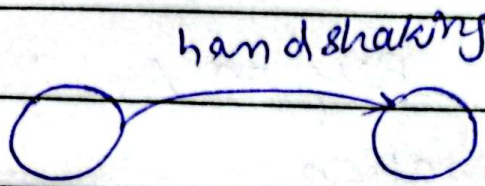
Parallel & Dist Computing.

Latency & Bandwidth → No of Bits Transfer per second.

A ↓ B

O ⟶ O

Msg sharing between two nodes and Time taken by it.



⊟ ⊟ ⊟ ⊟ per chunk at per unit time

➡ Overhead is latency. ↓ Solution:- Reduce time by transferring all data in the form of larger message.

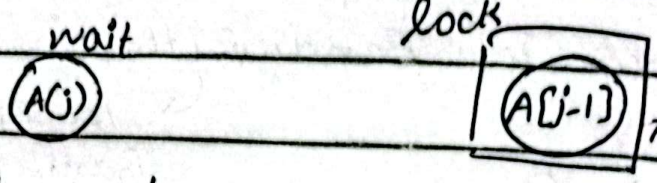
Heat defusion: We need nearby values of pixels → wait untill all the threads come to that barrier and collect data to do further execution (Synchronus) ✓

Non-Blocking State (Pixel Reverse)

↳ (Asynchronus) ✓

Date: 1 /20 Day:

Lock / Semaphores



It can't be solve in parallel.

- Parallel Computers

Instruction: Rely on Architecture

Set of operation

instructions → Add AX, BX (x86)

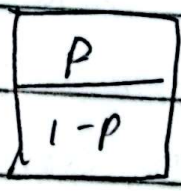
ADD RAX, RAX, RBX (x86-83)

Micro operations:

$r[A] \leftarrow r[B] + r[C]$

This is an operation that can be written in diff architectures.

- Rules.



optimization

Highway Rules-

Amdahl's law - speedup

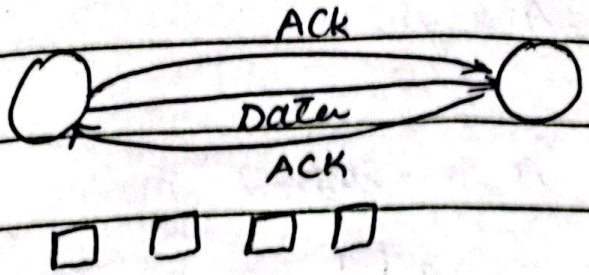
$n = 30 \Rightarrow 3.09$ 3 times better

$n = 100 \Rightarrow 3.21$ 3.2 times better

↳ not utilized.

Date: __/__/20 Day: _____

Plumber's Rule :-



Cache → If Data is to be frequently used place it in cache.

↓

Temporal

Spatial

↓

Take nearby with it, it can be used in future in the same program.

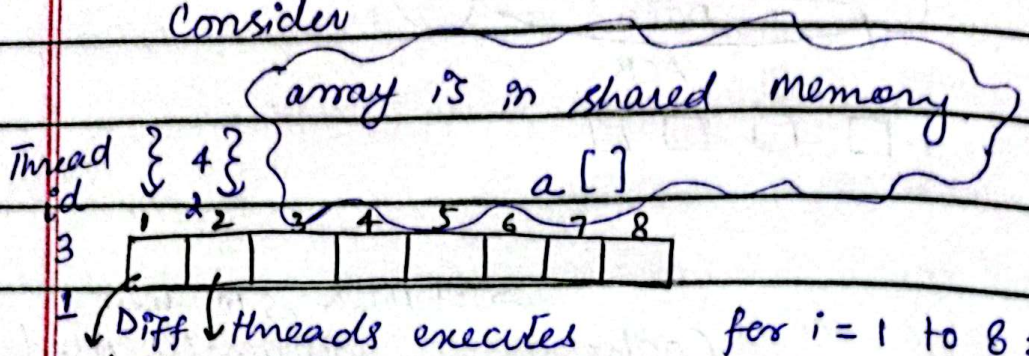
(Take full block of Data to the Cache).

Date: 27/03/2025 Day: Thursdays

- Parallel & Distributed Computing -

→ Dependence Analysis -

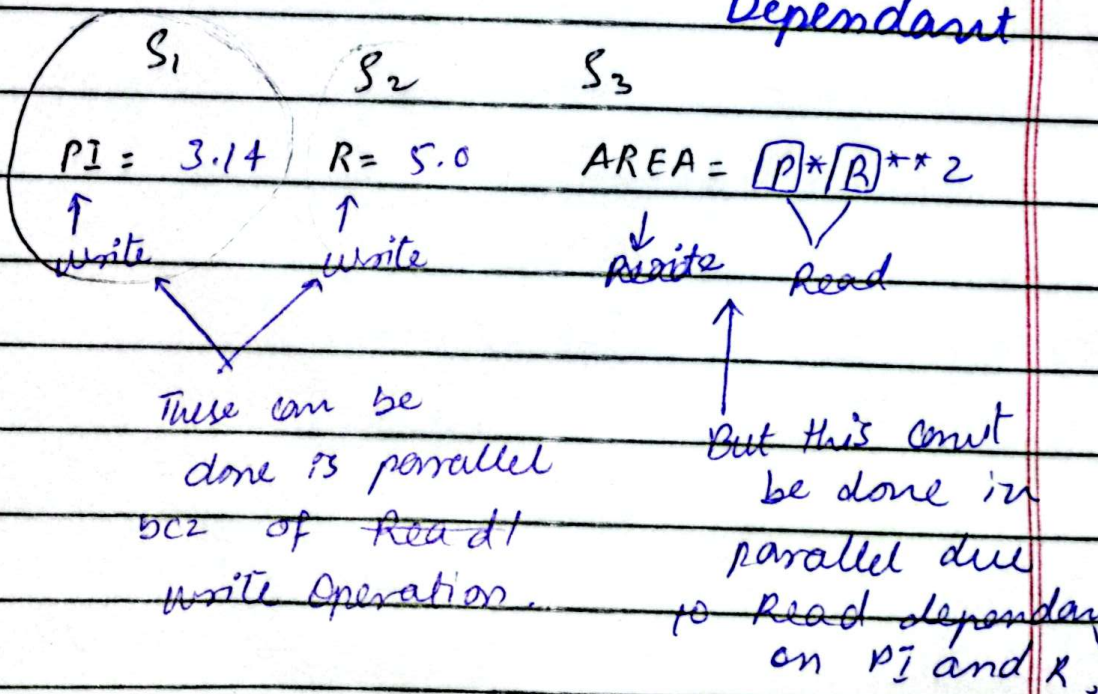
Consider



then

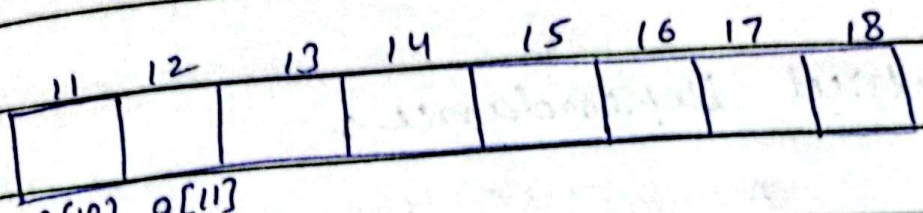
→ fully independant so parallel execution is possible utilize all the resources that are available.

Not parallel Dependant



Date: / /20 Day:

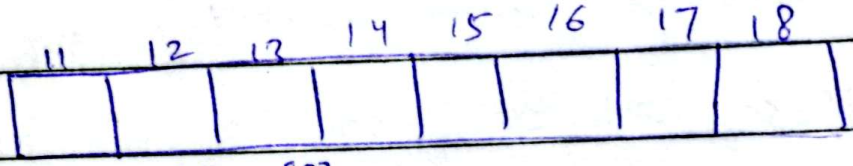
for i = 11, 20 Not parallel
a[i] = a[i-1] + 3



read a[10] a[11]
write a[11] a[12]

Dependancy
You don't know that which operation done earlier read or write.

for i = 11, 20
a[i] = a[i-10] + 3 Parallel



read a[1] a[2] a[3]
write a[11] a[12] a[13]

- It can be executed parallel bcz the index read can't be in write and vice versa.

→ Control Dependancy:-

If-else structure that (If) is T or F Then we Go further.

→ True flow dependancy:-

write → x = y + z ↓ first write then Read.
w = x + 3
↑ Read

Date: 1/20 Day:

Anti Dependance :

first Read then write

Output Dependance :

A variable changes 2 time.

✓ $W = \pi r^2$

✓ $W = \pi(r-1)^2$ (update here)

(Not use in parallel due to conflict).

Input Dependance :

Read a single value 2 time in first equation then in 2nd equation.

Solution to all 3:- { These can be handled or eliminate by changing variable names }.

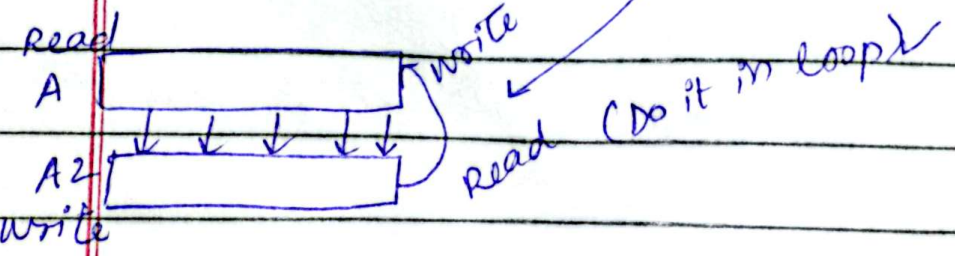
Eliminate Anti Dependency:-

$i = 1, 10$

$A[i] = A[i+1]$

$i = 1, 10$ $\quad$ $i = 1, 10$

$A2[i] = A[i]$ $\quad$ $A[i] = A2[i+1]$

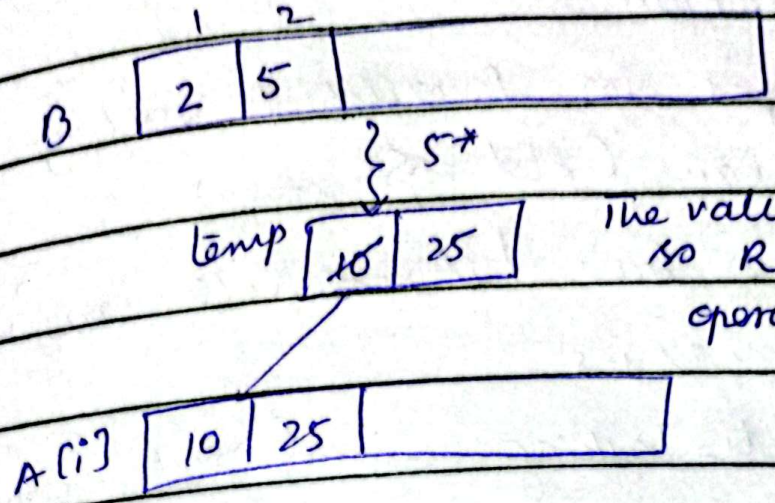


Day:

⇒ for i = 1 to n

temp = 5 * B[i]

A[i] = temp



The value changes so Read Write operation Conflicts.

⇒ i = 3, 11, 4

A[i+2] = A[i+1] + 1

3 A[5] = A[2] + 1

7 A[9] = A[8] + 1

11 A[13] = A[10] + 1

you

Date: 10/04/2025 Day: Thursday.

- Parallel & Dist Comp -

- Mid Term Syllabus -

- Topics -

Slides

- Introduction
- Challenges in Parallel Computing
- Moore's law (old & new)
- Amdal's law | {Amdahl's law}
- Preliminaries
  - { Multi-Tasking (Don't cram definition only concepts)
  - multi-programing
  - multi-processing
  - Parallel programming }
- Speed up
- Parallel efficiency
- Basic Communication Pattern (Book Gramma Chapter 4)
  - (Broadcast
  - Reduce (only linear)
  - Scatter
  - Gather

Note : Only for one-Dimensional structure

- Parallelizing Applications and performance aspects (slides)

Date: / /20 Day:

(Granularity, Bandwidth, latency, Speedup, Efficiency).

- Designing parallel programs.
  Automatic vs manual Syncronization,
  Dependancy Analysis (till detection of dependency in loops).
- Cache
  (Cache locality → Principle, Spatial locality → Slide
  Miss / Hit Rate → Book
  Cache Coherence → Slide)

Numericals → imp