Lec1: 1300

Moores law Express Surf ½ kg
? ½ kg
½ kg.

"Processing speeds, or overall processing power for computers will double every 18 months."
↳ "The number of transistors on an affordable CPU would double every two years [18 months].

Q. why doubling transistor not double speed?
↳ Increase in number of transistor per processor is due to multi-core CPU's.

Q. Moore law not forever ⇒ because transistor ~~would~~ will reach limits of miniaturization at atomic levels.
+ ~~transistor~~ multi processor CPU chips causes heat issues.

Memory/Disk speed Argument:
↑ 40% processors , ↑ 10% DRAM access rate
This mismatch causes performance ~~issue~~ bottlenecks.

Paliel platform ⇒ increased bandwidth.
↳ higher aggregate caches.

Data Communication Argument
- Today data evolve how, internet ki vision emerge hoi as ml large computing platform
- Data volume bhut zyada, in databases, can't be moved.
- Parallel techniques se data process hoye.

Computing vs Systems:

Distributed Systems:
↳ Collection of autonomous computers, connected through a network and distribution middleware.
↳ hardware based acceleration.

$\frac{1-\frac{1}{8}}{1-\frac{1}{2}}$ $\frac{0.375-0.5}{1-0.5}$ $\frac{p(s_0 \| p)}{2}$ $1-f=0.7$

$f=0.3$

$\frac{1}{0.3+\frac{0.7}{8}}=$

Distributed Computing

↳ A specific use of distributed systems, to split large & complex processing into subparts & execute them in parallel, to increase productivity.

↳ Software based acceleration (algos).

Parallel (shared-memory) Computing:

Developing concurrent solutions for following:

1- Multi-core Architecture

2- Many core Architectures (GPUs etc)

Distributed Computing:

- Developing Algos.
- Distributed means geographical distance b/w computers without any shared memory.
- High latency & no shared clock.

Limitations:

- Sync
- Explore proper parallelism
- Low coupling & high cohesion programs multi-kil bnare

Lecture 2

Amdahl law:

⟹ Data Parallelism

1- Data Parallelism: a[i] for i in 0,99. c[i] = b[i]+c[i]

↳ can be done simultaneously

2- Functional Parallelism: → a = 2, b = 3

m = (a+b)/2 s = a^2+b^2 can be performed parallely.

3- Pipelining

→ Inst 1 → output → Input → Inst 2

parallely → Inst 2 Inst 3 → Inst 4 Inst 5

Multiprocessor & Multi Computer

Multiprocessor:

↳ Multiple CPUs with shared memory.

↳ Same address of 2 CPUs refer to shared memory location.

Categories

1- Centralized Multiprocessor (same processor share same memory) UMA SMP

2- Distributed Multiprocessor. ⇒ Dist collection of memories forms one logical address space.

Non uniform memory access ⇒ NUMA

⇒ memory access time varies on physical location of referenced address.

Multicomputer:

↓

Distributed memory, multi CPU.

→ Each processor has direct address to their local memory only.

→ Same address, diff computer ~~[illegible]~~ means shared memory.

Assymetric Multi computer

- Backend processors for "number crunching"
- Frontend saara OS, comp functions chalata
- Backend for ~~multi pro~~ parallel executing of programs.

Symetric Multi computer

- Multiple Users handle krta.

PRAM = Parallel random Access machine

problem

↳ Concurrent read concurrent write (sab kuch allow)

↳ Conc " read Exclusive write

↳ Exclu read conc write

↳ Ex read Exc write (only 1 allowed 1 time)

4 solutions:

↳ Arbitrary

↳ Common

↳ Priority

↳ Sum

$O(mp)$

m → words in memory

p → processor

Communication cost depend on features:

- → Programming Model for communication.
- → Network topology
- → Data handling and routing
- → Associated network Protocols

Total time to transfer message:

- → Startup time (t_s)
- → Per-hop time (t_h) ⇒ node latency
- → Per-word transfer (t_w)
  - ↳ includes overheads that are determined by length of message.

- Store & Forward Routing

$$t_{comm} = t_s + (m t_w + t_h) l$$

in most, t_h is small.

$$t_{com} = t_s + m t_w l$$

- Packet Routing,
  - ↳ breaks in message into packets.

$$t_{comm} = t_s + t_h l + t_w m$$

→ overhead in packet headers.

  - ↳ can go in different paths.

- Cut through Routing:
  - ↳ splits message into smallest parts called flits,
  - ↳ header info is minimized
  - ↳ all take same route

$$t_{comm} = t_s + t_h l + t_w m$$

Lec 5 PDC

Principles of Parallel Algo design

Steps:

1. Identification
   - ↳ Identify portions of work jo concurrently performhoskty.
   - ↳ work units ko tasks bni rehty
   - ↳ e.g 2 arrays ko initialize krna, are parallel hoskta task.

2. Mapping
   - ↳ Process of mapping concurrent pieces of work or tasks onto multiple processes running parallely.
   - ↳ Multiple processes physically aik processor pr map hoskty.

3. Data Partitioning:
   - ↳ Distributing input/output/intermediate data associated with program.

ways:

- copy whole data at each processing node
  - masla: Memory challenges for large size problem
- give fragments of data to each processing node
  - masla: communication overhead.

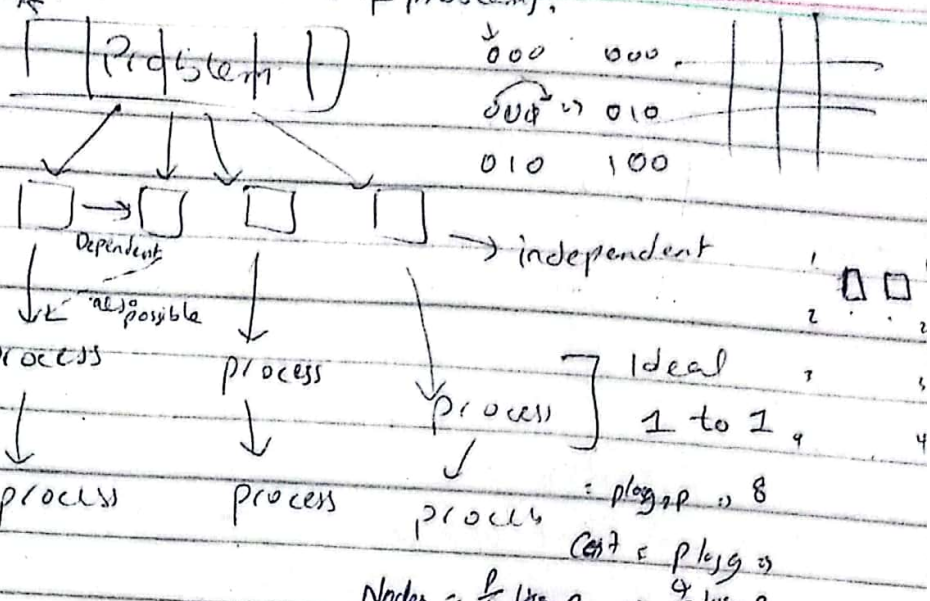
IMDDS

4. Defining Access Protocol:
   - → Managing ~~pro~~ accesses to data shared by multiple processors.

5. Synchronizing:
   - → Processors at various stages of parallel program execution

6. Decomposition:
   - → Process of dividing computation into small parts.

Parts/split (depends on nature of problem).



Ideal 1 to 1

$= p\log_2 p \Rightarrow 8$

$Cost = p\log_2 p \Rightarrow$

$Nodes = \frac{p}{2}\log_2 p \Rightarrow \frac{4}{2}\log_2 p$

$\Rightarrow 4$

Tasks:

- ↳ Programmer defined units of computation.
- ↳ Arbitrary size, but once defined, it is indivisible units of computation.
- ↳ Sab tasks same size K honey zaroori nhi.
- ↳ Simultaneous execution of multiple tasks in key to reduce time.

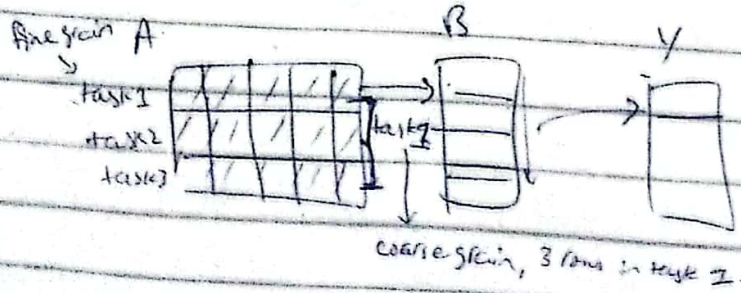
Granularity:

↳ No. of sizes of tasks into which problem is decomposed. determines granularity of decomposition.

Fine grained → Split into small parts

Coarse grained → Split into big parts.

Task dependency graph:



it is independent task & can be ~~performed~~ performed in any sequence.

→ An abstraction used to express such dependencies among tasks and their relative order of executing is known as "task dependency graph".

→ It is directed acyclic graph

$$\boxed{T1} \rightarrow \boxed{T2}$$

node dependency.

Max deg of concurrency:

↳ Max no. of tasks jo aik time pr execute hoskay - parallely.

↳ Dependency wala ki wajah se less than total task hota.

↳ Rule of thumb:

Max deg of concurrency is always equal to no of leaves in tree.

Avg deg of concurrency:

↳ Avg no. of tasks jo parallely execute hoskay.

$$\frac{\text{total amount of work (all nodes sum)}}{\text{critical path length (sab se lmbay path ki length / nodes sum)}}$$

Critical path: longest directed path b/w any pair of start & finish nodes.

• Shorter critical path favours higher avg deg of concurrency.

→ Jaisay task small hotay, max & avg deg of concurrency increases

2P - 258

Task Interact Graph:

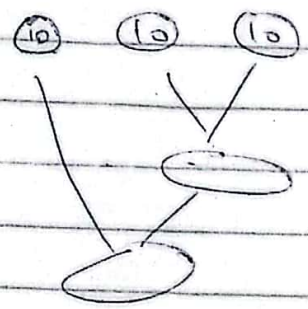
Task kaisay interact krtay, dependency graph sirf bataty k depend ~~kaisay~~ kia task kr rha lekin task interact, distributed data kaisay access krtay means interact graphs.

→ 'Undirected' → ~~di~~ dakin directed sirf flow of data aur Unidirected ho.

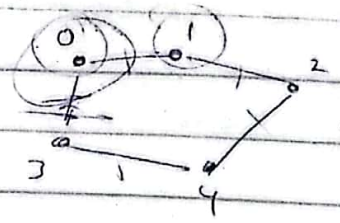
→ Edge set of task interaction graph is 'superset' of edge set of task dependency graph.

→ Task interaction graph is same as task dependency graph in data base query processing.

dependency



Interaction



	0	1	2	3	4
0	•	•		•	
1			•		•
2					•
3	•				•
4			•	•	

- avg deg of concurrency =
- total amount of work =
- critical path
- critical path length.
- max deg of concurrency

Processes & Mapping:

- Process ⇒ logical processing/computation agent jo task perform krta.
- Mapping ⇒ Mechanism by which (jo) task assign krta execution k liye.
- Multiple task should be mapped onto different processes.
- Map task with high mutual interacting onto single process

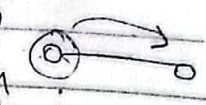
Processors ⇒ hardware units that physically perform computation.

- Depending on problem, multiple proc can be mapped on single processor.
- 1 to 1 correspondance b/w processors & processes.
- We assume many processes as no. of physical CPUs on parallel computer.

Store & Forward

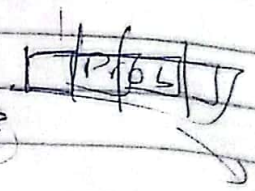
$$t_{comm} = t_s + (t_h + t_w m)\, l$$

message size, links



Packet routing

$$t_{comm} = t_s + t_h l + t_w m$$



Cut.

Select k:

k = 1

minimum distance = 41 = movie 0,

so it is comedy.

- It is better for k = 5, ideally, but we have 4 examples, so atleast taking k = 3.

So,

1st, 3rd, 4th ~~is~~ have smallest distance

So

1st → Action

3rd → comedy hence Barbie = Comedy

4th → comedy

we select Max.

As 2 comedy & 1 action

2 > 1.

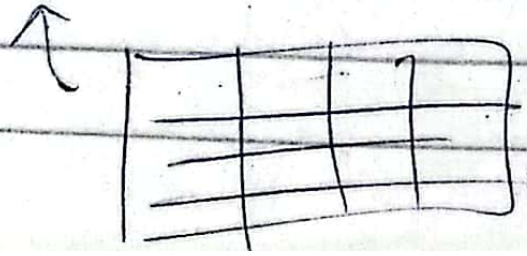
static scheme

2D: we do no. of block = no of process

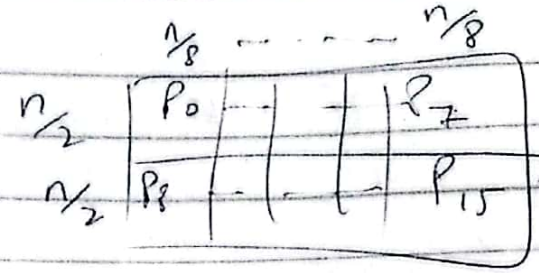
Block size = $\frac{n}{rows} \times \frac{n}{cols} = \frac{n}{4} \times \frac{n}{4}$

Processes = $P_1 \times P_2 = \overset{row}{4} \times \overset{col}{4}$

$= 16$



$\Rightarrow n \times n = 4 \times 4 = 16$

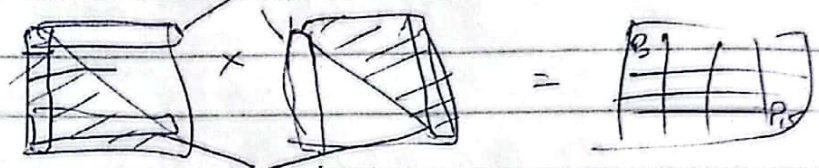


$\Rightarrow n \times n$
$2 \times 8 = 16$

$$BS = \frac{n}{2} \times \frac{n}{8}$$

Drawback:

low balancing = low computation



lower triangular matrix — high balance ↓ high computation — upper triangular matrix

cyclic distribution:

difference b/w 1D static → No of blocks ≥ No of process

• depends on α (alpha) i.e no. of cycles ai k process ko kitni baar repeat krna hoga k sarey process map hojai

$$1 \le \alpha \le \frac{n}{p}$$

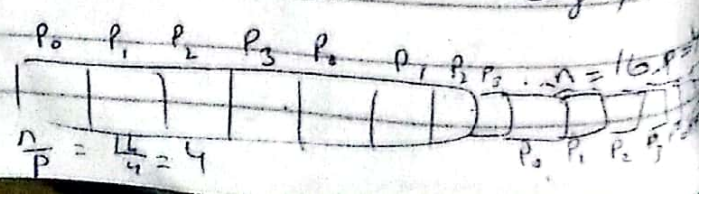
if $\alpha = 1$,

$$BS = \frac{n}{\alpha p} = \frac{n}{p} \Rightarrow$$ Block distribution (works same as previous distributions)

if $\alpha = n/p$

$$BS = \frac{n}{\left(\frac{n}{p}\right) p} = \frac{n}{n} = 1$$ (har block mai one element ho jaa)

e.g



no of cycles $\alpha = \frac{n}{p} = \frac{16}{4} = 4$

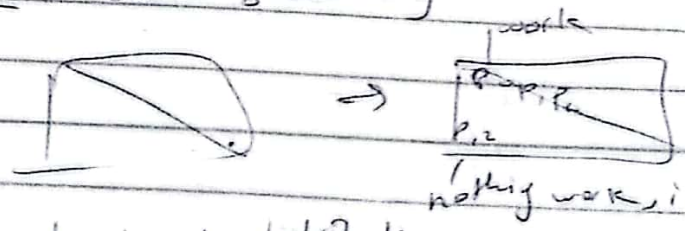
Block Cycle Dist $\frac{n}{\alpha P}$ $\frac{n}{P}$

$n = 16$
$P = 4$
$\alpha = 2$

$\frac{n}{\text{row}} \times \frac{n}{\text{col}}$

$B.S = \frac{n}{\alpha P} = \frac{16}{2 \times 4} = 2$ means 2 elements each in block

Master Load balancing



Randomized distribution

$V = \{1, 2, 3 \ldots n\}$

random kardo
phir map kardo.

5/8

Array 1D
Array 2D

$B.S$ = no. of processors