Dependency Analysis

(1) Dependence: Constraints of Execut^n bcz of each other.

(2) Dependence Analysis: To ~~identy~~ it identify which factors creating ~~obstacle~~ dependency-

(3) Dependency Types: Control Dependency (Graph Based) (Task Based)
Data Dependency: (Interact^n Based)

Dependency Analysis

- Two memory access are Involved.
- Data may refer to a same memory locat^n.
- Ensure data is produced | consumed in right order.

Type of Dependency Analysis

(1) → **Flow Dependency**

- Instruct^n depends on final outcome of previous instrucn.
- also called: Write or write/read dependency.

Add, R_1, R_2
MOV, R_4, R_1

(3) **I/O Dependency**

- If both I/O statement try to use same file for their operation.

- Data flow dependency: Not removed
- Anti dependency: can be removed
- O/P dependency can be removed.

(2) → **Anti Dependency.**

- One Instruct^n is depends upon the data that could be distroyed by another instruct^n.
- Occurance: One ~~register~~ inst. reads a register. & subquent inst. writes value to same locat^n.

Condition: Two inst. have Anti-D only if swapping their order would result in true dependency.

$$\begin{bmatrix} A = B+C \\ C = D+E \end{bmatrix}$$

(4) **O/P dependency**

- Occurs: when two ins both write a result.
- Write write dependency.
- Exist due to limited no. of arch. register.

$$\left.\begin{matrix} A = b+c \\ d = c+d. \end{matrix}\right\} \text{O/P dependency}$$

GB23: HPC: 3:45-4:45 : 15/03/24 : Tuesday

Roll_NO: 18, 19, 20, 21, 23, 24, 25, 26, 27, 30, 31, 32, 33, 34, 35
36, 37, ~~41~~ 38, 41, 42, 44, 45, 46, 47.

PI : 1, 2, 3, 13.

Mapping Parallel Algorithm onto Parallel Architecture.

→ Mapping requirement

Analysis of Parallel Algo. + definition of logical configurat^n of platforms + Mapping of algo to logic. ||alfor.

→ Issues of mapping arises when

↳ No. of process > No. of units processing

↳ communication structure of algo. differs from interconnect structure of parallel machine : Topological Variation

↳ No. of process in PA > No. of units in Parallel M/c : Cardinality Variat^n

Solut^n : Contract^n : Embedding : Multiplexing

Threads Basics. —

Pthread

↳ Creat^n / Terminat^n

↳ Synchronizat^n

↳ controlling Threads & Synchronization Attributes.

Attribute of Objects for Threads.

〃 —— for Mutexes

Composite synchronization Constructs.

OpenMp.

↳ Programming Model.

↳ Reduction Clause

↳ Program : Example

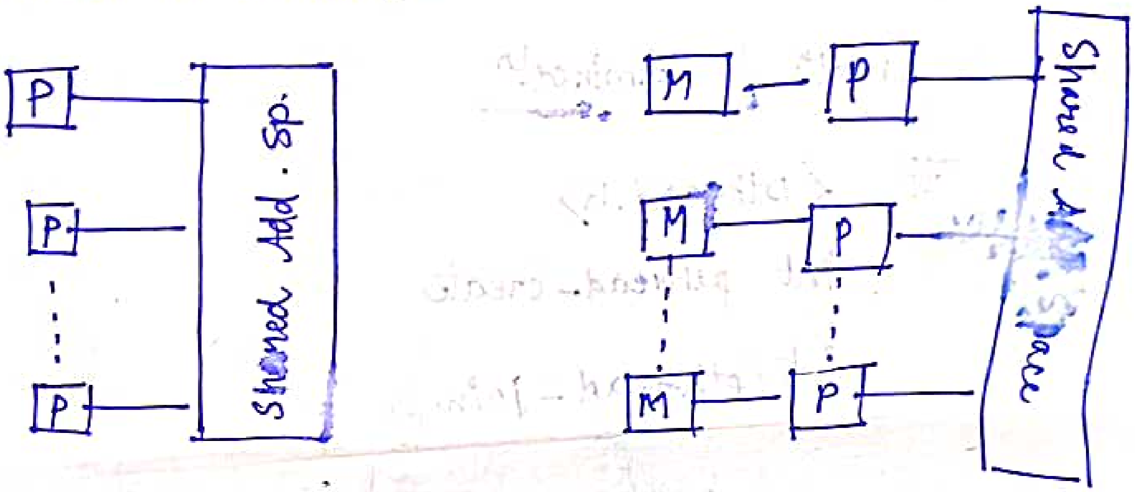
↳ specifying con. Task

Take that after Thread / Pthreads

Thread Basic

↳ Memory in the logical m/c model of thread is globally accessible to every thread.

↳ Subprocess is thread.



↳ Properties :

↳ Provide Software Portability : through threaded API

↳ Inherent support for latency hiding.

↳ Scheduling & load balancing.

↳ Ease of programming & wide spread use.

The POSIX thread API

↳ Referred to Pthread.

↳ Standard thread APIs.

↳ These can be used for programming with other thread APIs. (NT threads, Solaris. thread Java threads. as well.

↳ Creatn / Terminatn

```
# <pthread.h>
int pthread-create
int pthread-join
```

Time 1- 25 0-255

P-threads (POSIX)

↳ p-threads are standard threads API)

↳ can be used for programming with other thread APIs. like NT thread, solaris threads, Java threads.

↳ include <pthread.h>

↳ Header file contains function declarations & macros for threading interfaces & defines no. of constant used by those functions

↳ Pthread_create (pthread_t* __restrict__ thread, const pthread_attr_t *attr, void *(*start_routi (void * arg), void * __restrict__ arg);

↳ Pthread_create : Creates a new thread within the process with

attr : is the thread attribute object.

→ Pthreat_t : Datatype to uniquely ~~identy~~ identify a thread.

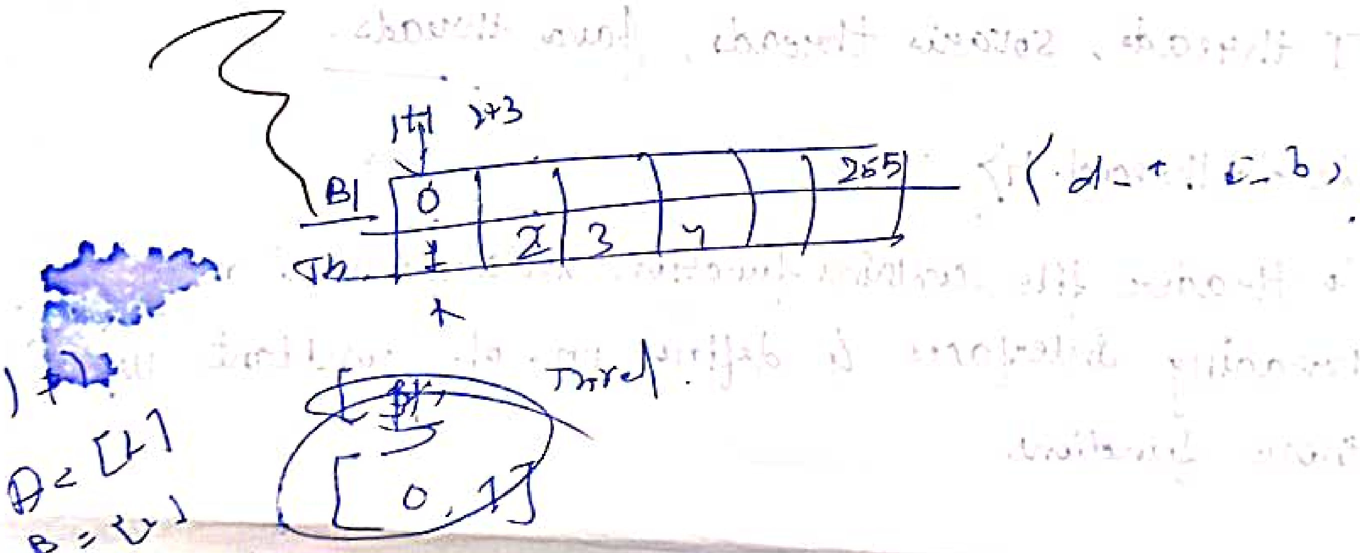
→ start_routine : pointer to sub routine that is executed by thread.

→ Arg : pointer to void that contains the arguement to the function defined in the previous argument.

↳ pthread_join (pthread_t thread , void ** ptr)

: Used to wait for the termination of thread.

thread th : thread id of the thread for which current thread wait.



Synchronization Primitives in P.thread

↳ Multiple threads attempt to manipulate same data item

↳ Results can be ~~inod~~ incoherent

Example :

Each thread tries to update best_cost as follows

```
If ( my-cost < best cost)
        best-cost = my-cost ;
best_cost = 100;
t1 = 50,   t2 = 75.
```

Depend upon the schedule of the threads, the best_cost could be 50 or 75.

↳ The value 75 does not correspond. to any serializatn. of threads.

Controlling Thread & Synchronizatn Attributes

↳ Pthreads allows programmer to change the default attr using attributes objects.

↳ Attribute object: Data structure that describes entity

Entities(thread, mutex, condition variable)

↳ Enhances: Modularity, Realeability

Attributes objects for Thread.

pthread_attr_init : Create attributes objects.

- Setdetachstate :

- setguardsize_np

- setstacksize

- setinheritsched.

- setschedpolicy.

- setschedparam.

Attributes Objects for Mutexes.

↳ Mutexes (Mutual Exclusion)

: to save the processes from being in deadlock

: If thread is using resources, the value is set = 0;

If thread is available, then value = 1

functions.

↳ Pthread_Mutex_init :

↳ for the initialization of attribute object:

Types of Mutexes.

→ Pthread_ Mutex _ Normal . NP .

→ Pthread. Mutex _ RECURSIVE _ NP .

→ Pthread. MUTEX _ ERRORCHECK _NP .

Threads

Directive Based

↓

① ROPEN MP.

② for : C, C++, fortan.

Standard thread.

① POSIX - Pthreads

② can be used with NTF threads solaris threads, Java threads.

openMP programming model

① Conditional Parallelization

② Degree of concurrency

③ Data Handling

① Uses if clause

② Scalar expression

③ Determines whether the parallel construct results in creat^n of threads.

① Clause used

↳ private (variable list)

↳ firstprivate (vario. list)

↳ shared (variable list)

→ Degree of concurrency ←

① Uses num_threads

② Integer expression

③ Specifies the no. of threads created.

Private: Indicates variable local to the each thread

firstprivate: Similar to private Only difference is values of variables are initialized already.

Shared: Indicates that variables are shared across the threads

Sample program

```
#pragma omp parallel if (is_parallel == 1) num_threads(8)
~~(a) private (a)~~   ~~(b) fi~~
      (a) private
      (b) shared.
      (c) first private.
```

98

OPen MP

↳ Directive based t[illegible] -API.

↳ Used with: C, C++ & FORTAN.

↳ for programming shared address space.

↳ Provides: ① concurrency
② synchronization.
③ Data Handling

with the need to use Mutexes explicitly.

↳ Uses #pragma compiler. directives.

↳ ① #pragma omp directive. [clause.]
② # pragma omp parallel. [clause]

① OpenMP program executes serially until they are encounter the parallel directives which creates a group of threads.

② The main thread that encounters with parallel directiv is set as 0. Within the group. and called as master

If is-parallel = = 1

then .

1. 8 threads are created.
2. Each thread gets private copies of variable (a) and (c).
3. Share a single value of (b).
4. The value of each copy of (c) is intialized to the value of c before the parallel directive.
5. Tho default. state of variable in specified by the clause. default (shared) or default (none

Reduction clause

- How multiple local copies of a variable at different threads are combined into a single copy at the master when ~~the~~ threads exit.

- The usage of the reduction clause is reduction (operator : variable list)

- These ~~ke~~ variables are private to threads

- operator: +, *, -, &, |,

#pragma omp parallel reduction ~~(+:sum)num_~~
(+ :sum) num_threads (8)

{
 * compute local sums here */
}

Specifying concurrent Task

- Used with conjuction with other directives to specify concurrency across interactions / task.

- Two ~~direction~~ directives of OPEN MP.

↙ for

↓ section

58, 63, 65, 67. 70.

Parallel Processing fundamental Degisn Issue.

① Synchronizatn ② Scheduling. ③ Job allocation.

~~④ Job allocation.~~ ⑤ P. Job partioning. ⑥ Dependency anelysis

⑤ Mapping Parallel algo to Parallel Archi.

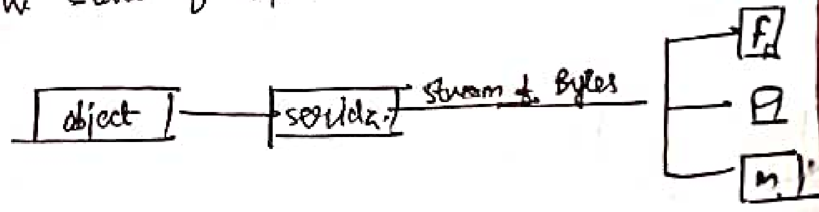
⑥ Performance Analysis of Parallel Algo.

Synchronization.

- Seq. of work & task.
- Performance factor
- Requires serielizatn.

~~Serielizat~~

It is a process of converting a data object : code / data represented in a storage into a series of bytes that save the state of object into transmissible form.



- Types
  - Barrier
  - Lock / Semaphore
  - Synchronous com. Operatn.
    - ↓ Acknowledgment Based
    - To look up the coordinatn b/w the s-process & r-process.

se to protect access to global data.

) one data can use one k at a time.

Job scheduling # Policies.

↓ FCFS

↓ Look ahead Optimizing Scheduler

- Backtracking Algo.
- Branches the variable to evaluate one of its value.

→ Gang scheduling

→ Task syn.

- Concurrent Gang
  - Multiple gangs run for t time
  - One stops then another gang executes
- SHARE scheduling.
  - ↓ Gangs with same resource utiliz

- Used to ensure that if two or more process / threads will be ready to communicate at the same time.

→