Today

Case Studies of parallel OS

* 

* Corey
* Cellular Disco } Partial reading

Today

Case Studies of parallel OS

* 

midterm: monday

* Corey
* Cellular Disco } partial reading

Today

Case Studies of parallel OS

* 

Midterm: Oct 5
Monday

* Corey
* Cellular Disco

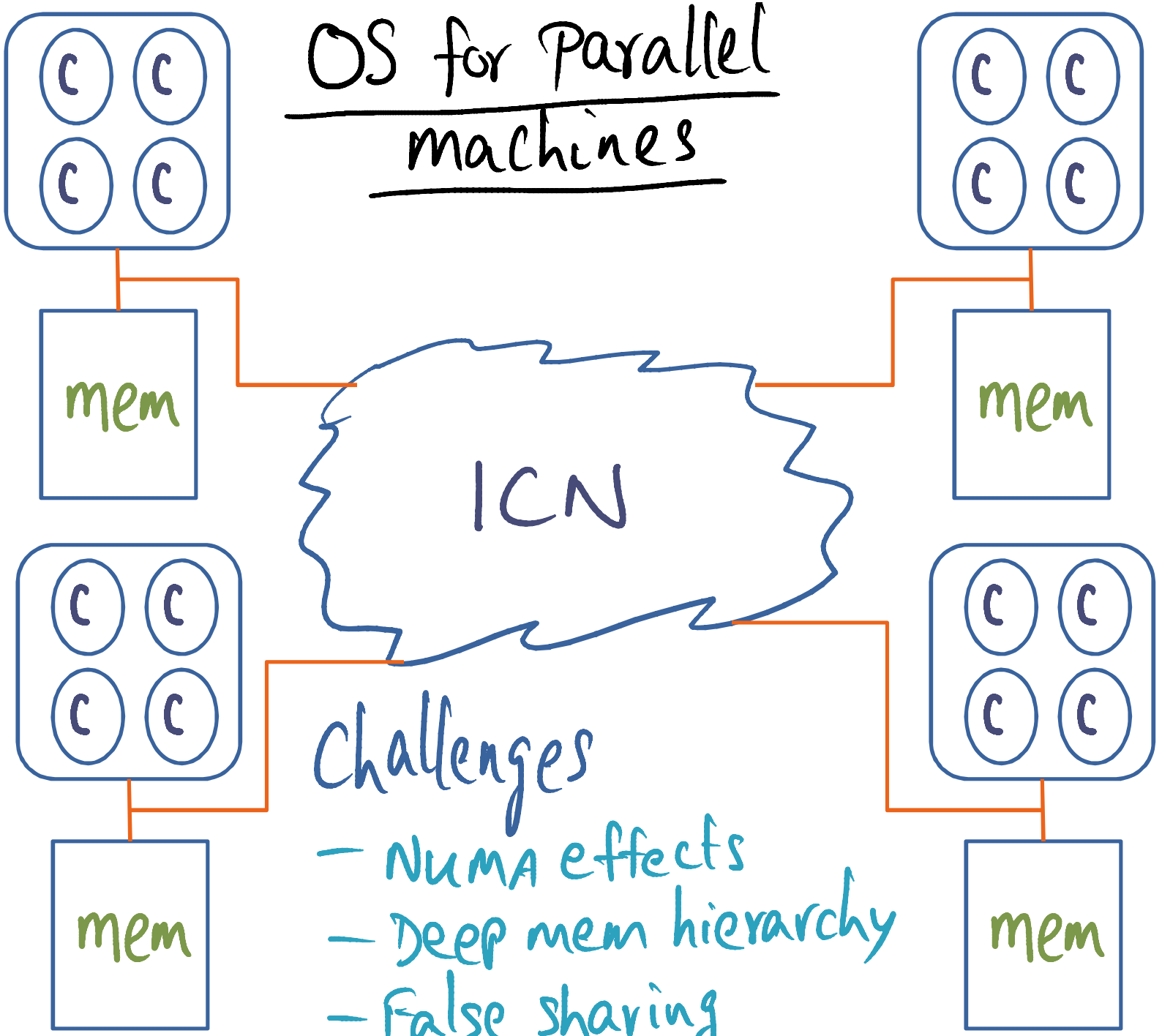
} partial reading

Next week

Distributed systems

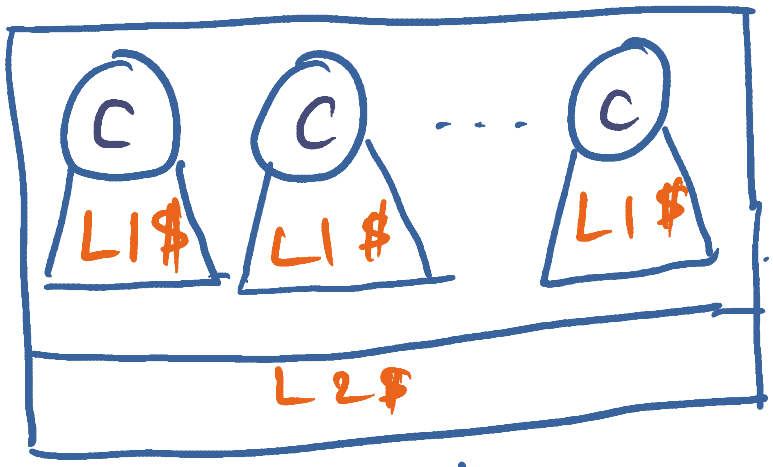
* Watch Lamport clock video
  (First video in Lesson 5 on Udacity)

OS for Parallel machines



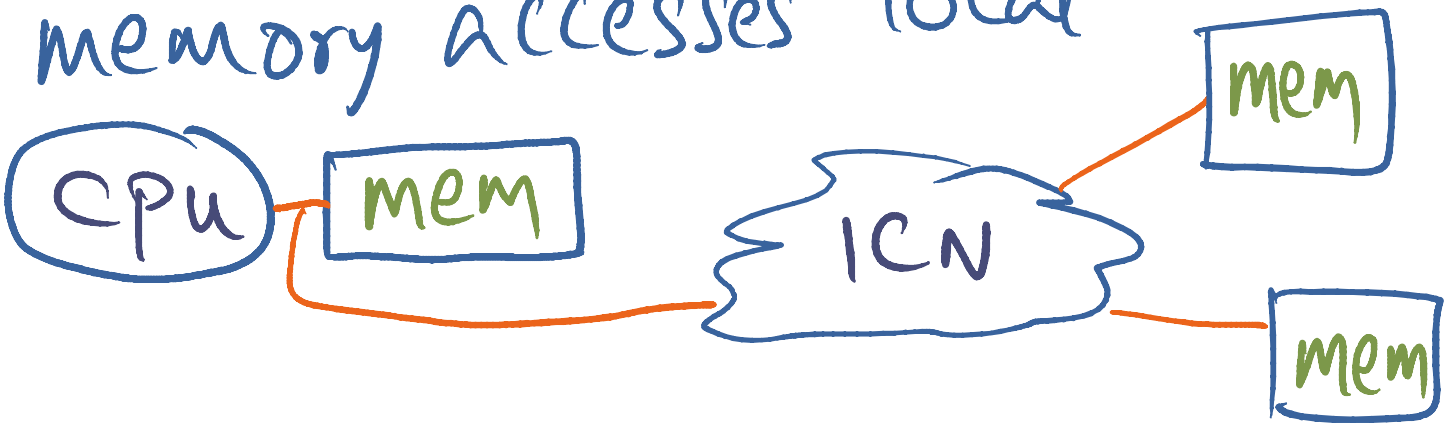
Principles

Cache Conscious decisions

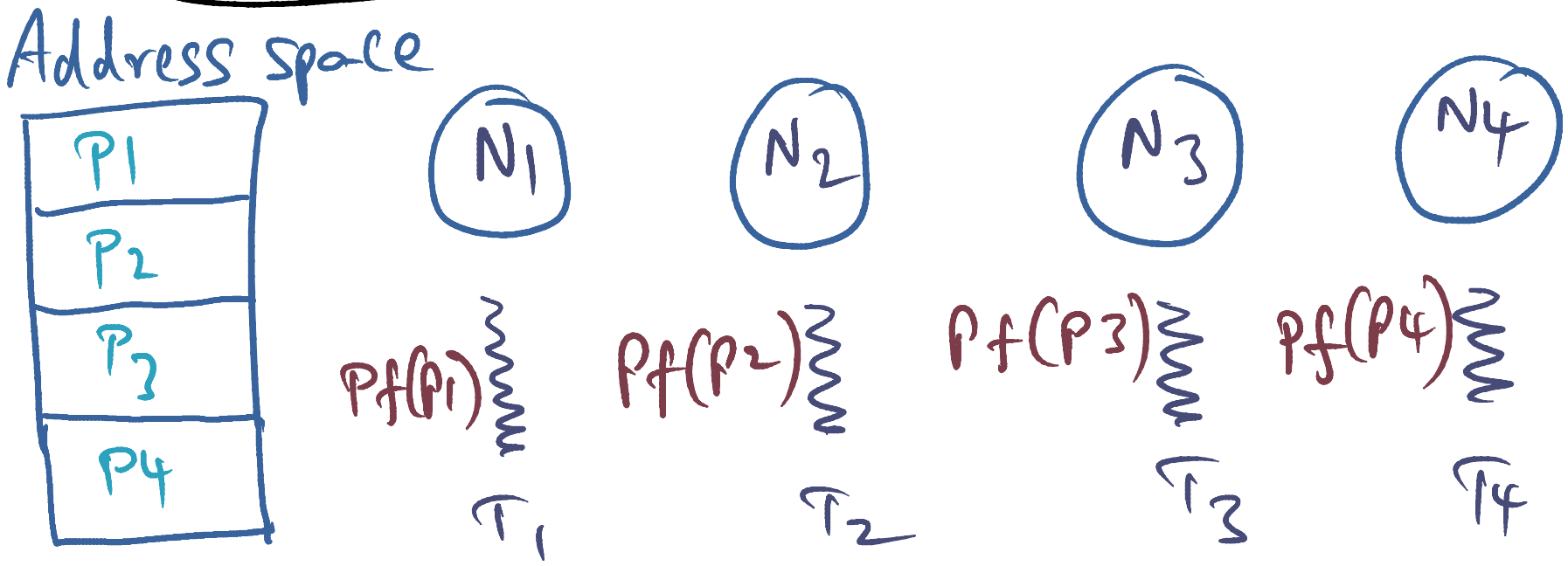


Limit shared System Data Structures

Keep memory accesses local



Concrete Example – P.F. Service

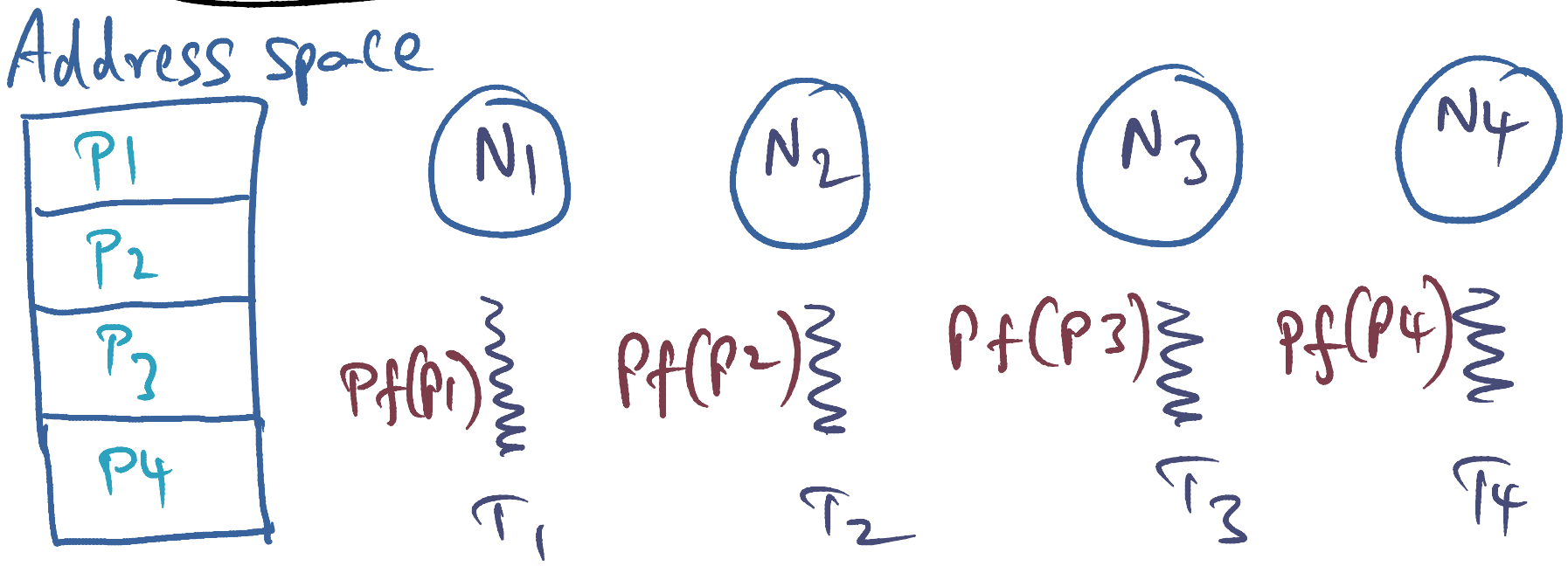


Page table Common

Page fault handler on each node

Page table data structure shared by all handlers

Concrete Example – P.F. Service

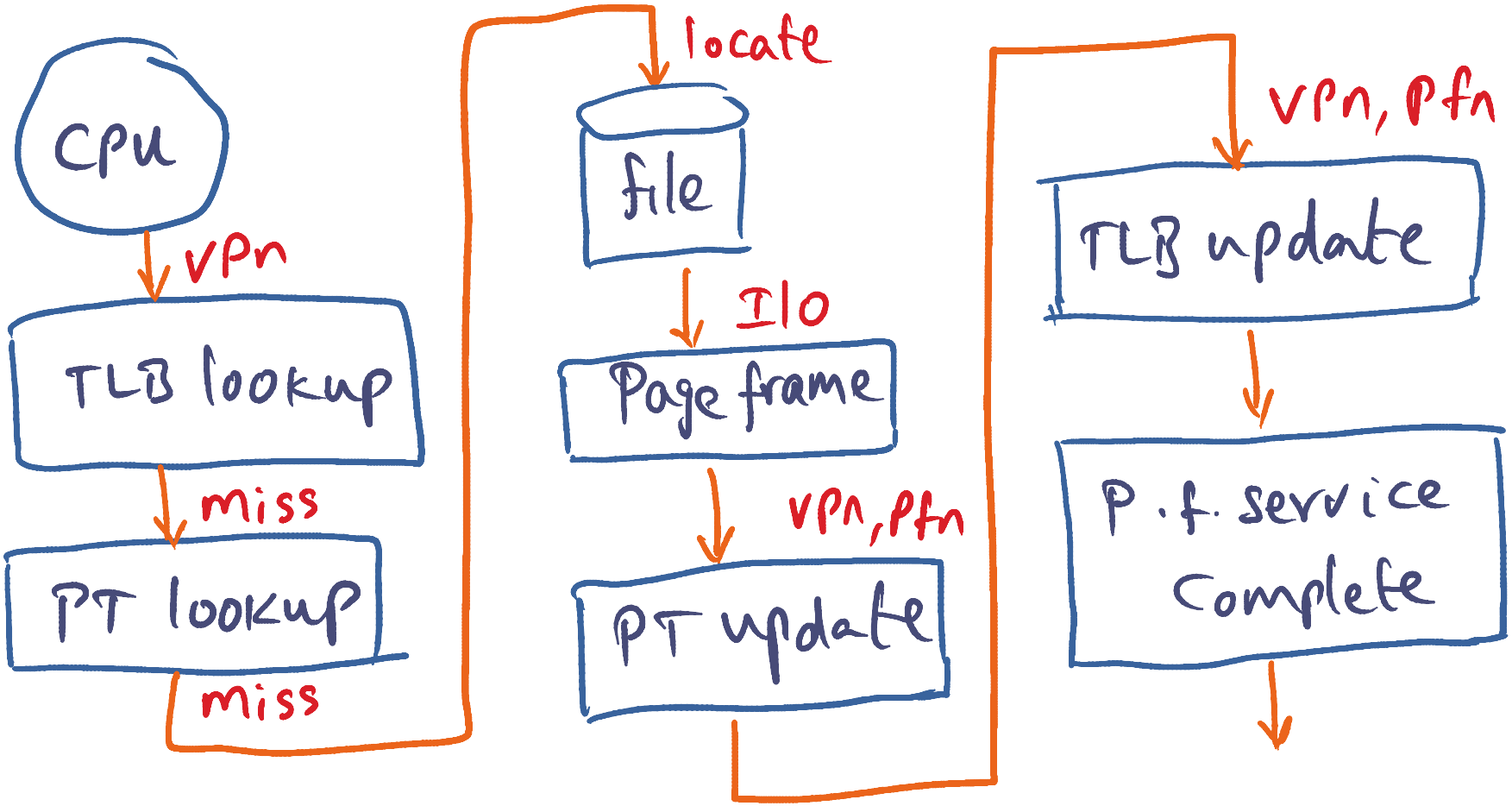


Page table Common

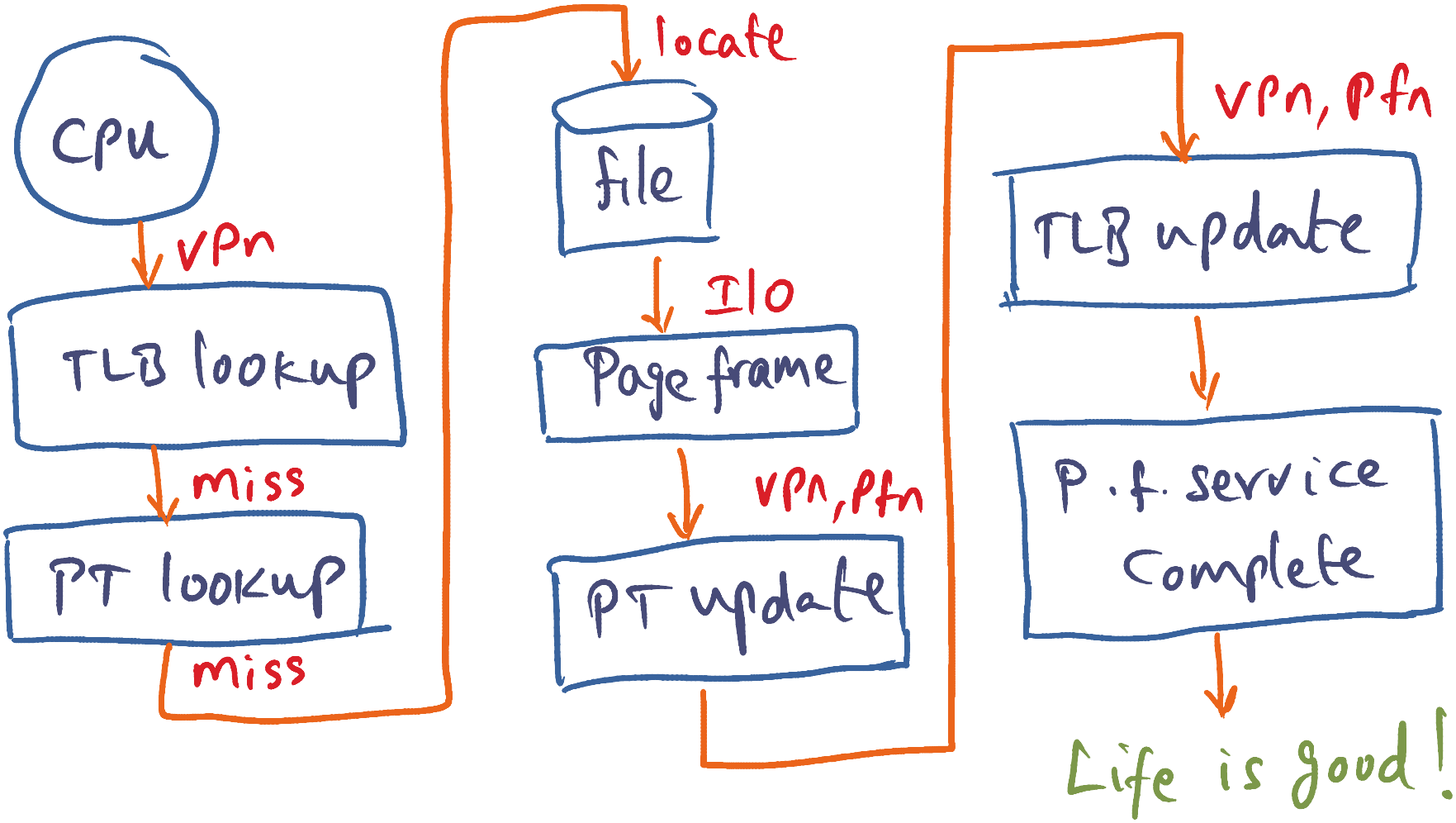
Page fault handler on each node

Page table data structure shared by all handlers ⇒ Serialization

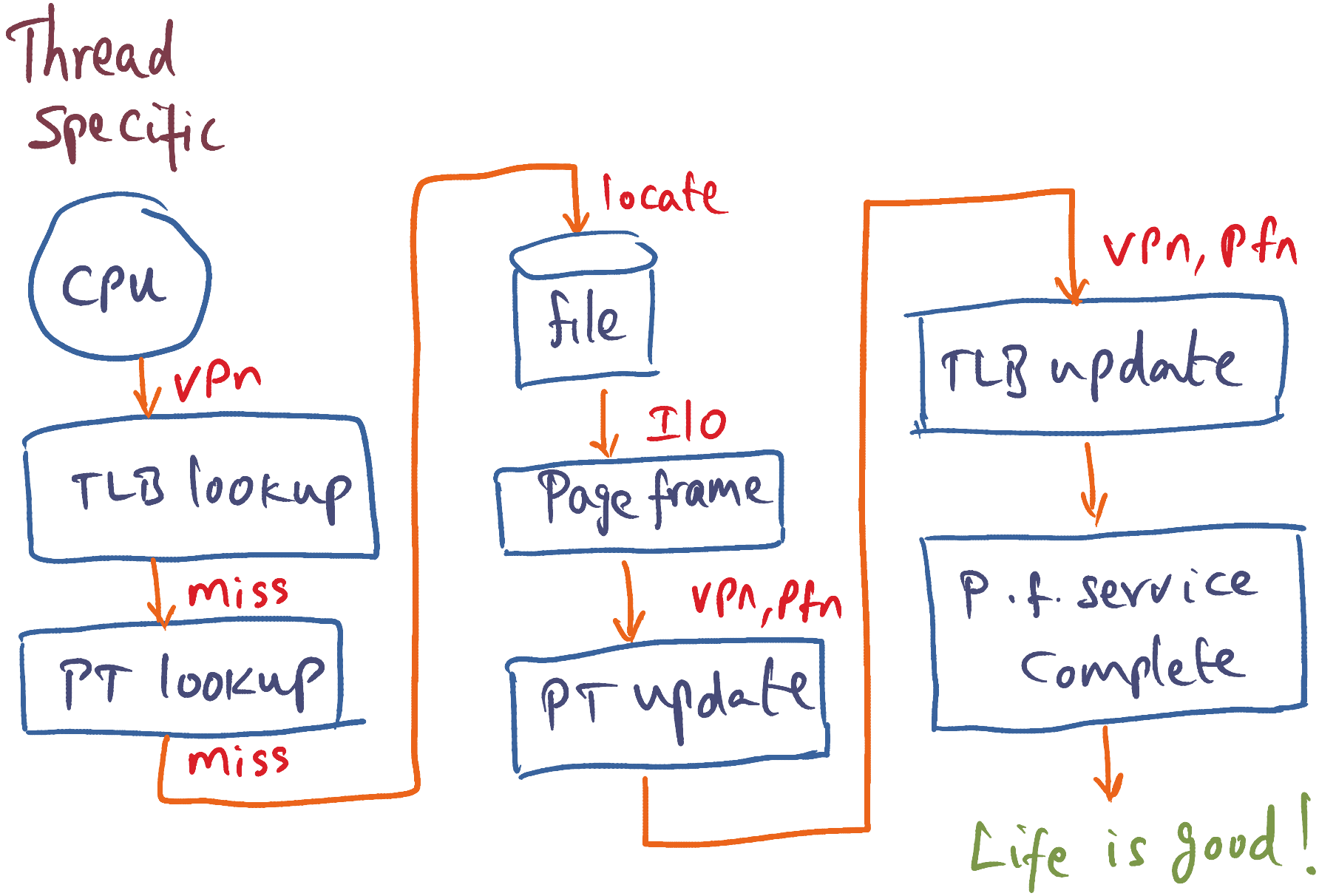
Refresher on page fault service



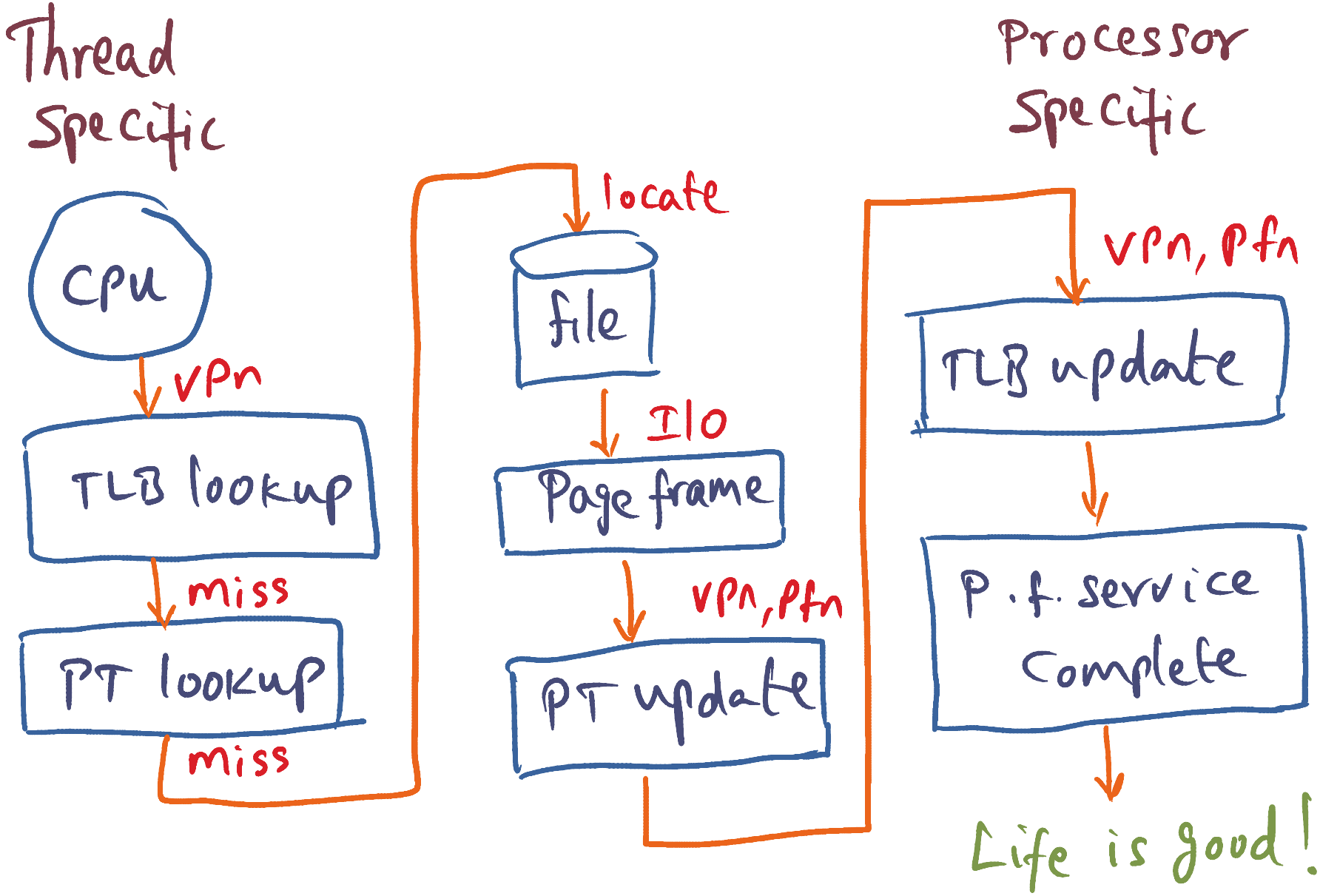
Refresher on page fault service



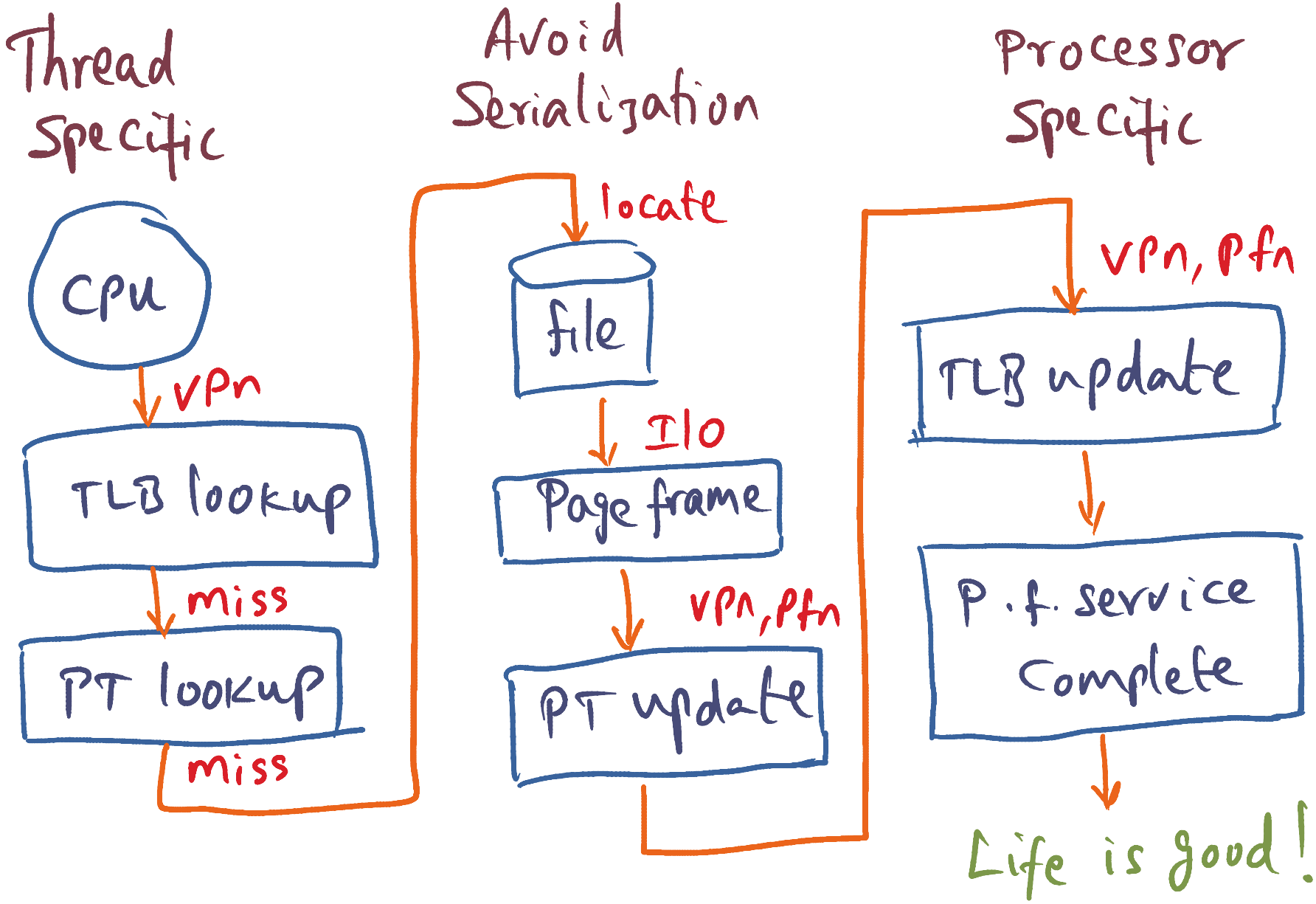
Refresher on page fault service



Refresher on page fault service



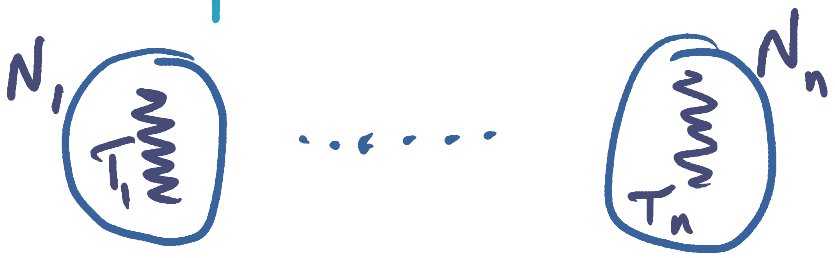
Refresher on page fault service



Parallel OS + page fault service

Easy scenario

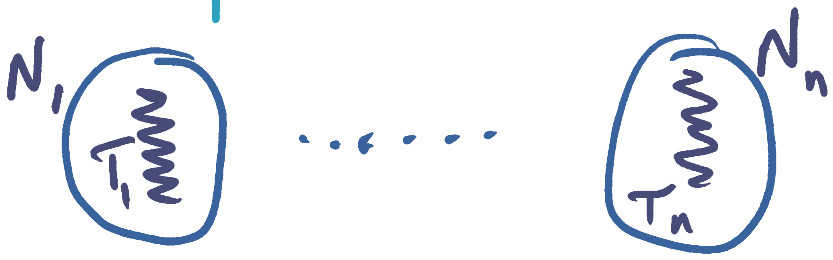
- multiprocess workload



Parallel OS + page fault service

Easy scenario

- multiprocess workload

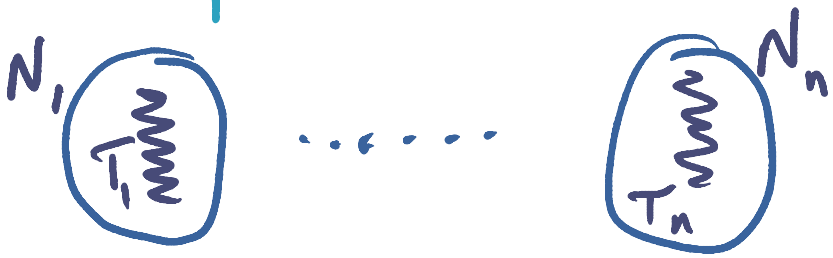


* Threads independent
* page tables distinct
* no serialization

Parallel OS + page fault service

Easy scenario

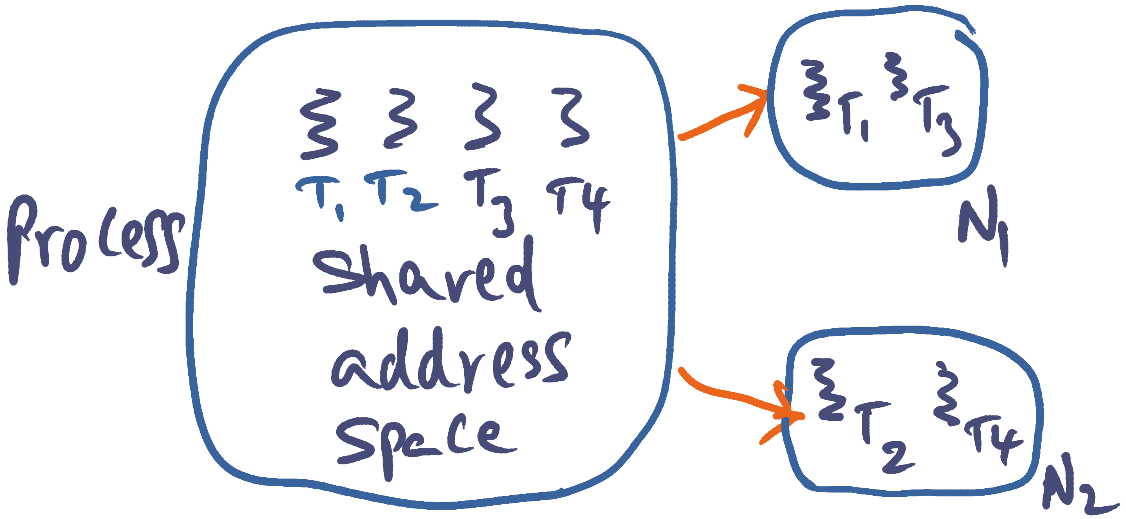
- multiprocess workload



* Threads independent
* page tables distinct
* no serialization

Hard scenario

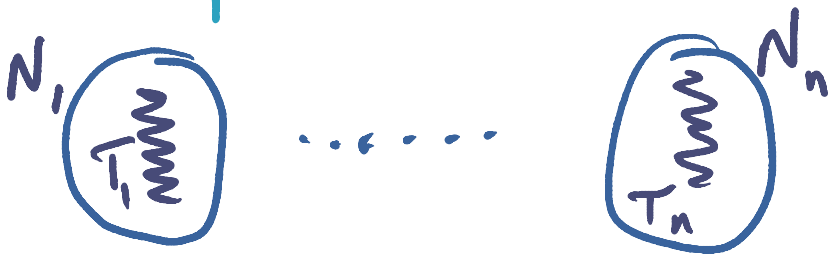
- multi-threaded workload



Parallel OS + page fault service

Easy scenario

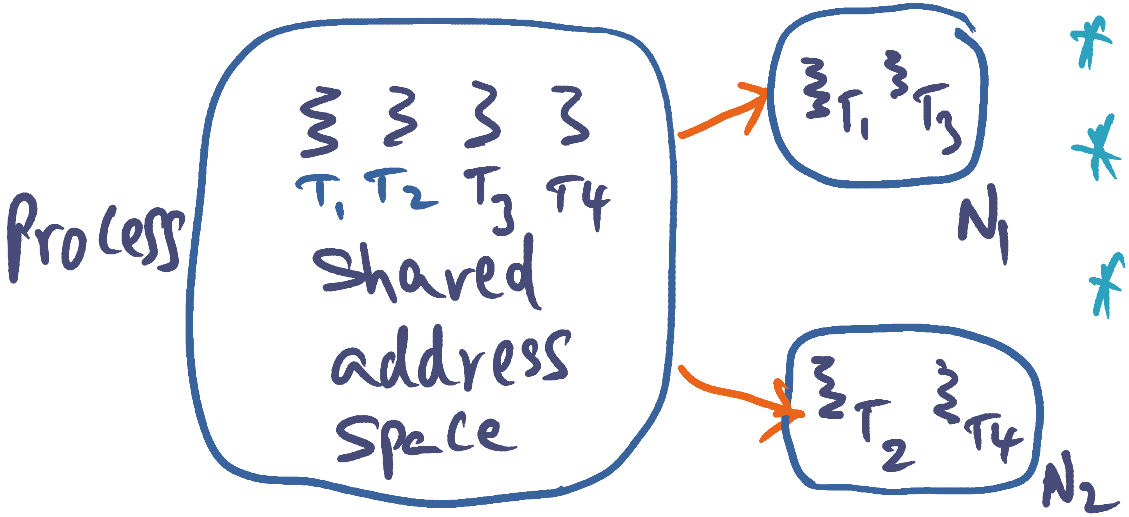
- multiprocess workload



* Threads independent
* page tables distinct
* no serialization

Hard scenario

- multi-threaded workload



* Address space shared
* page table Shared
* shared entries in processor TLB's

Recipe for Scalable structure in parallel OS

For every subsystem

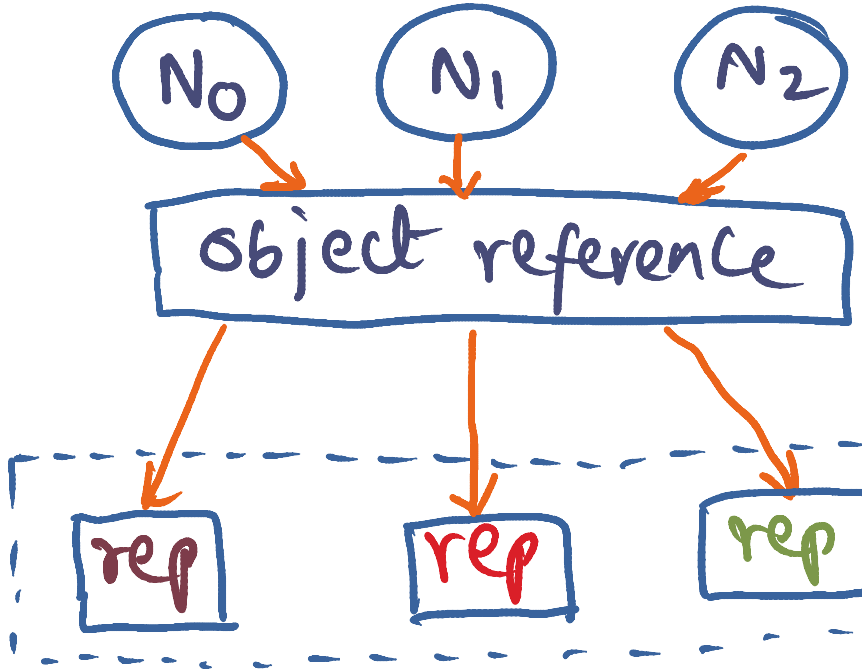
- Determine functionality needs of that service
- To ensure concurrent execution of service
  - * Minimize shared data structures

  Less sharing ⇒ more scalable
- Where possible replicate/partition system data structures
  - ⇒ less locking
  - ⇒ more concurrency

Watch !!

Watch!!
Did you??

Tornado's secret sauce: clustered object



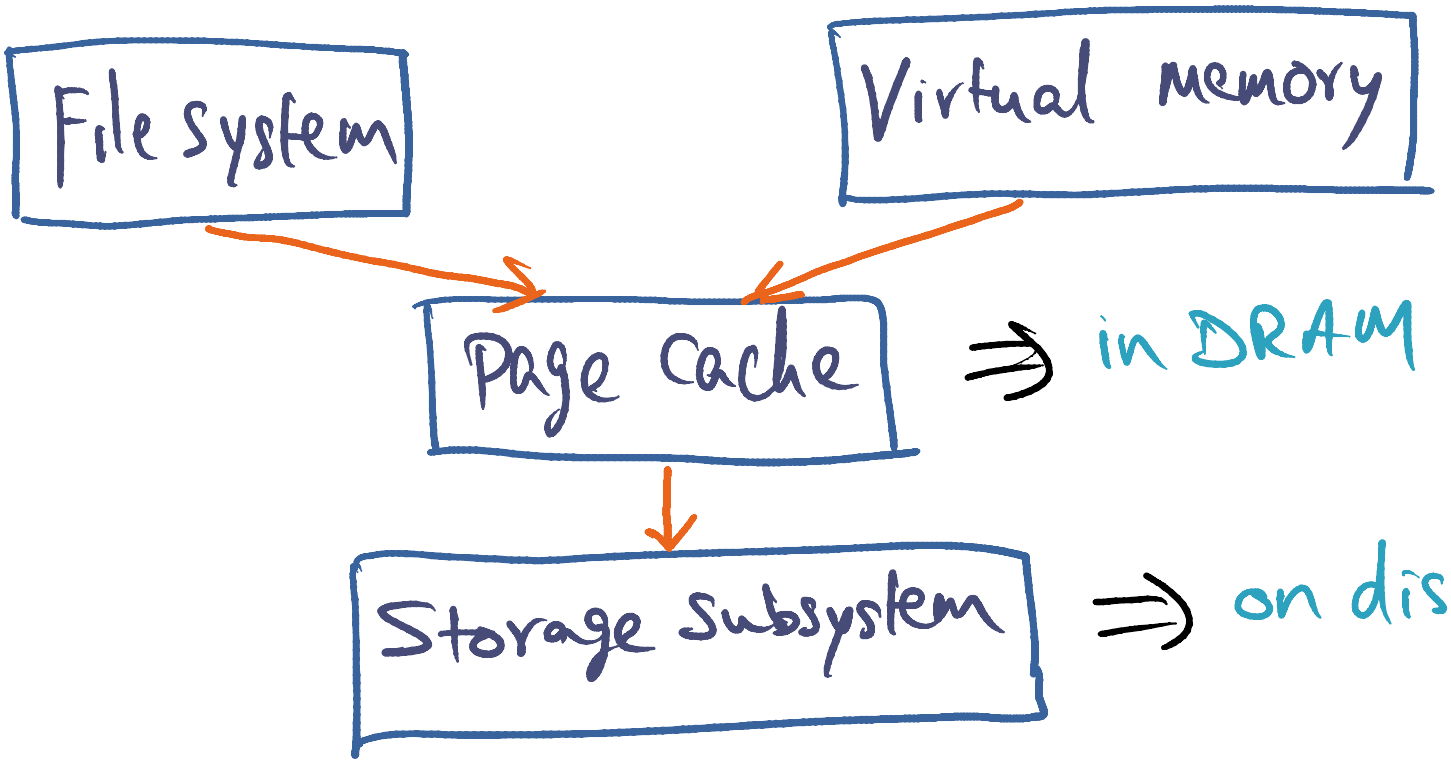
illusion of single object

under the cover: multiple representations

Degree of clustering?

- Choice of implementor of service
  - * Singleton rep, one per core, one per CPU, one per group of CPUs, ...
- PPC for consistency

Traditional Structure

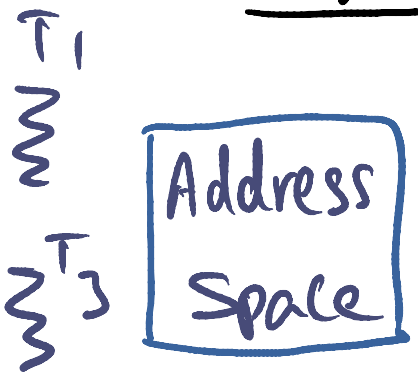


VM data structures

- PCB, TLB, PT
- virtual pages on disk

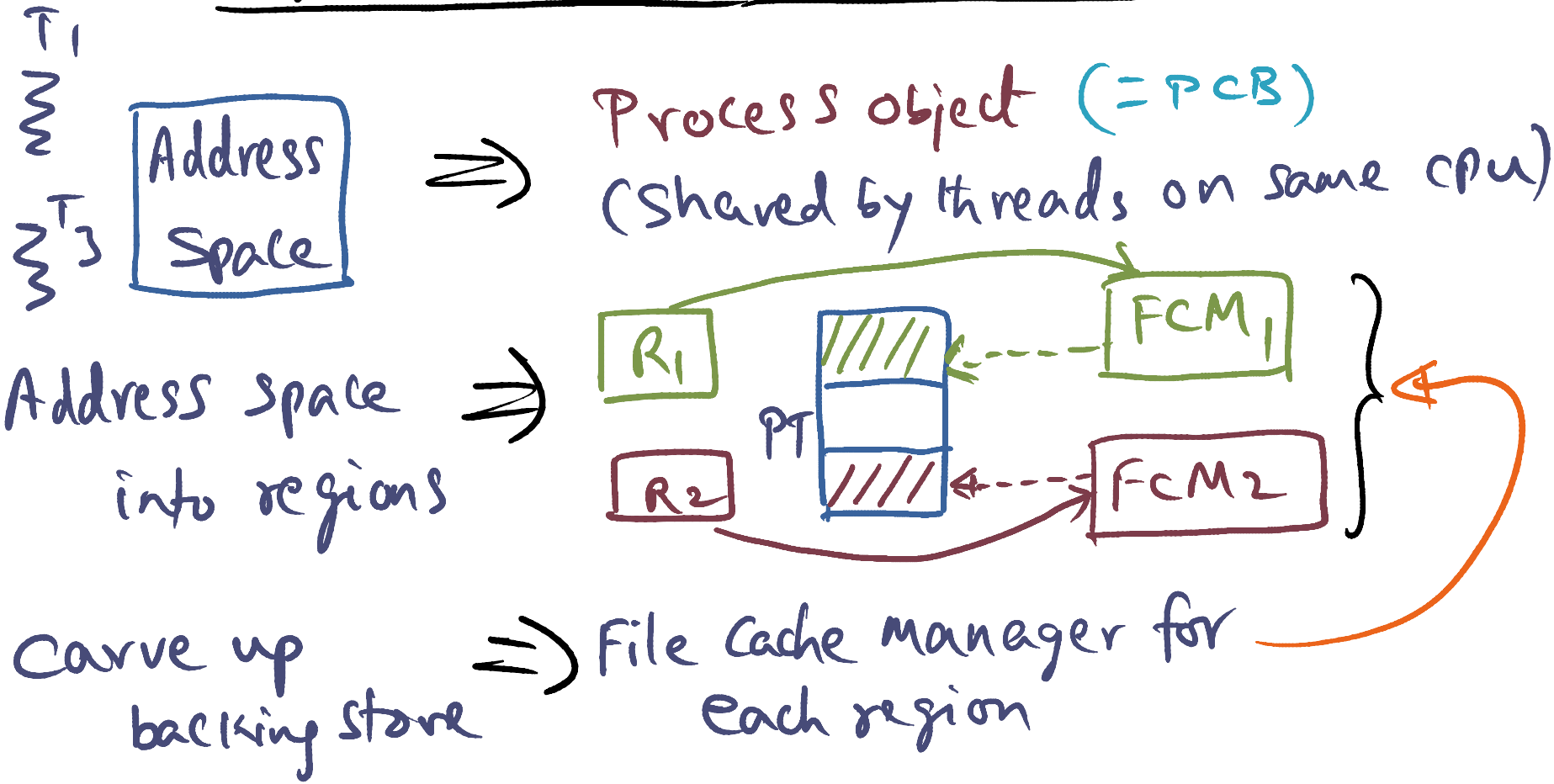
} ⇒ Per process

Objectization of memory management

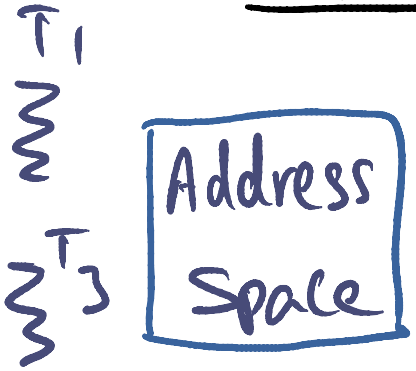


⇒ Process object (= PCB)
(Shared by threads on same CPU)

Objectization of memory management

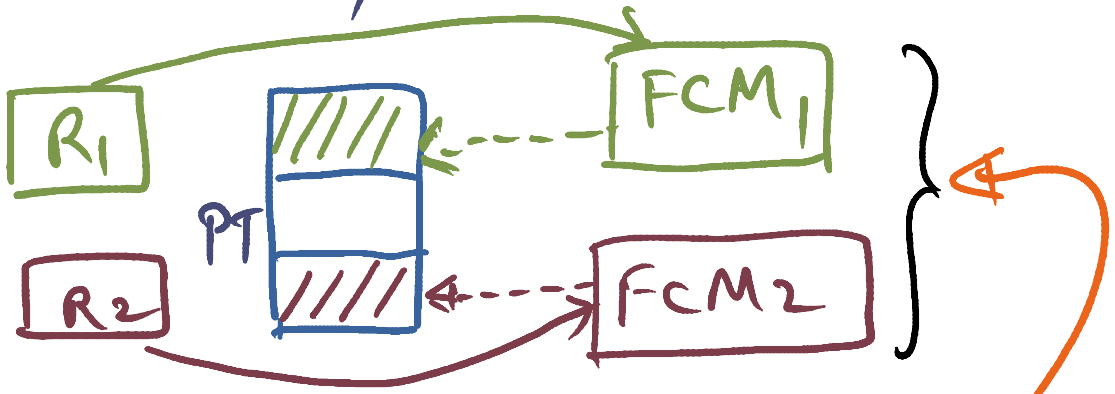


Objectization of memory management



⇒ Process object (= PCB) (Shared by threads on same CPU)

Address space into regions ⇒

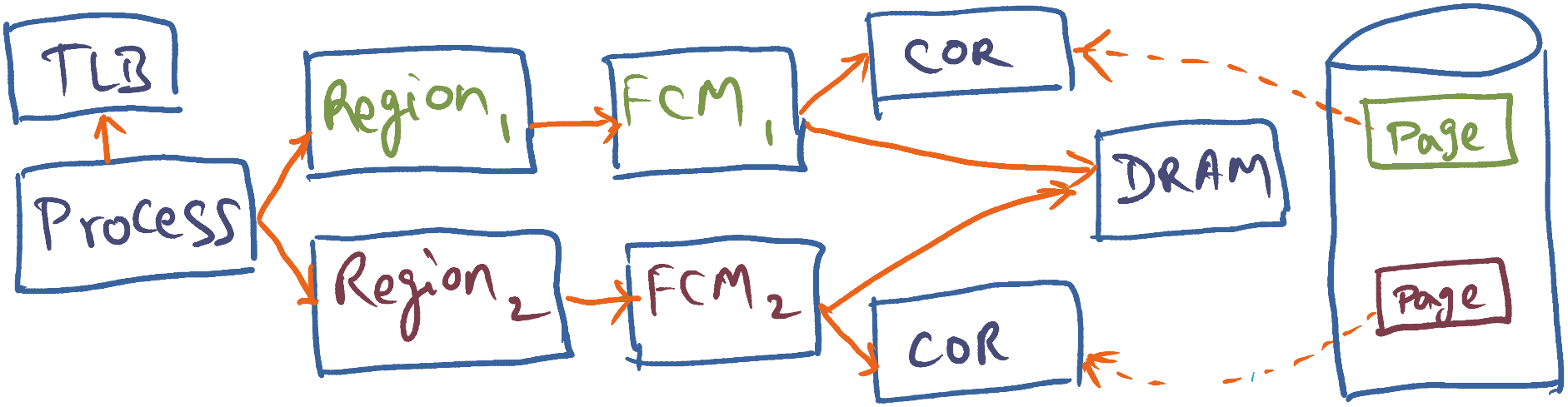


Carve up backing store ⇒ File Cache Manager for each region

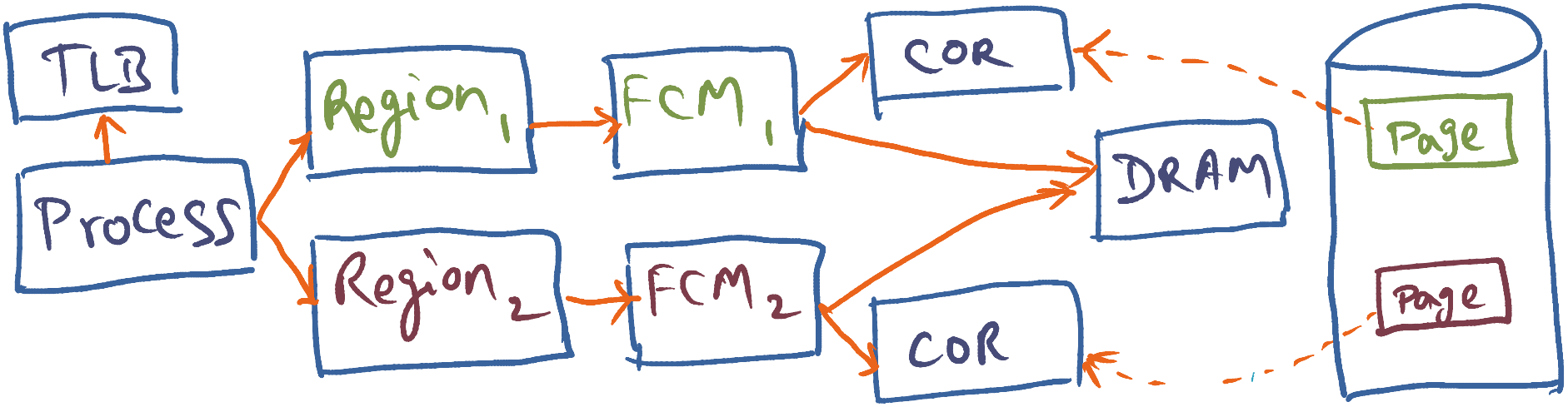
Page frame manager ⇒ DRAM object

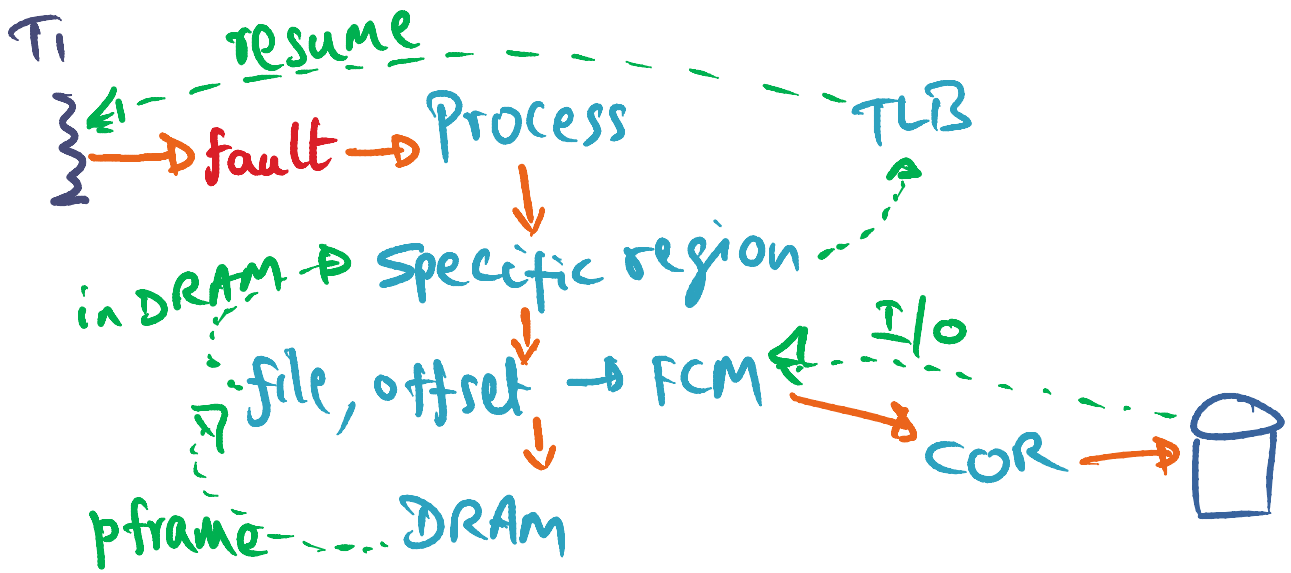
Page I/O ⇒ Cached object rep (COR)

Objectized Structure of VM Manager

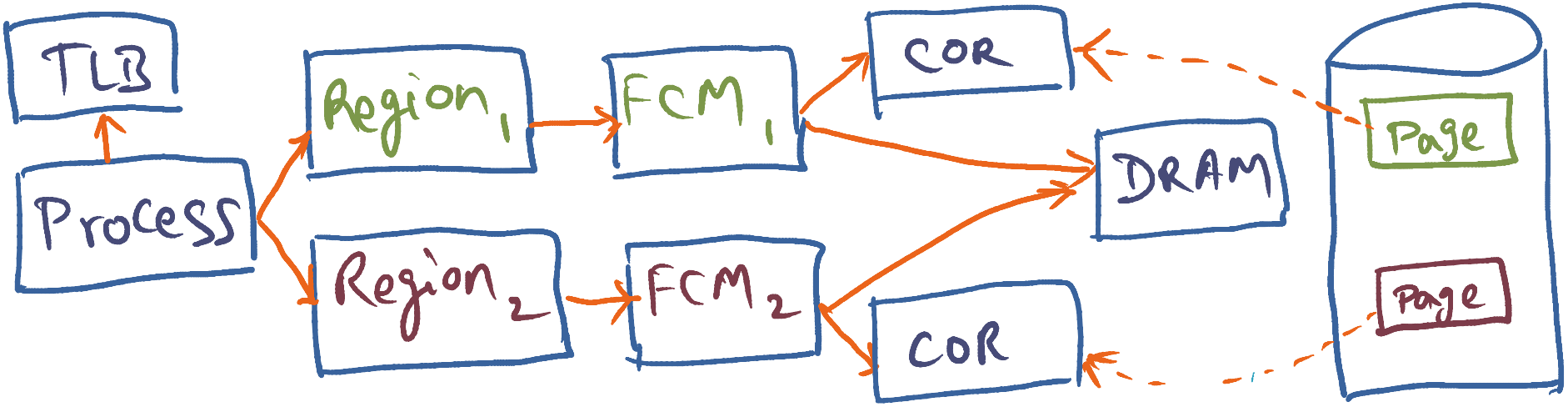


Objectized structure of VM manager



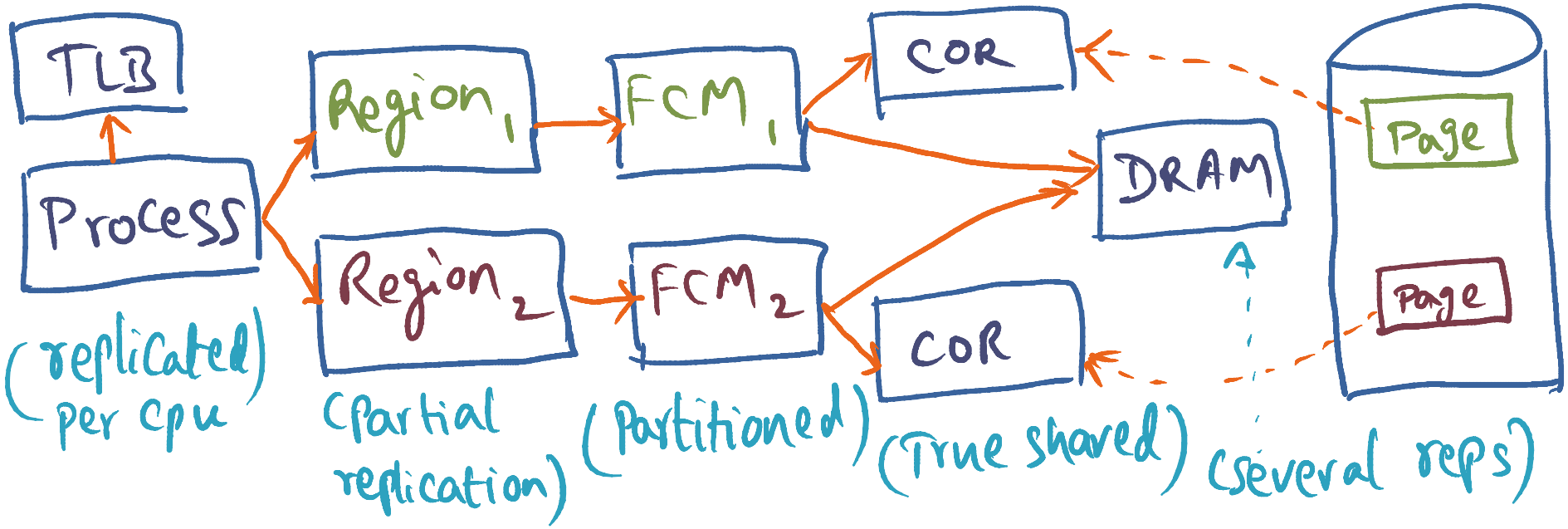


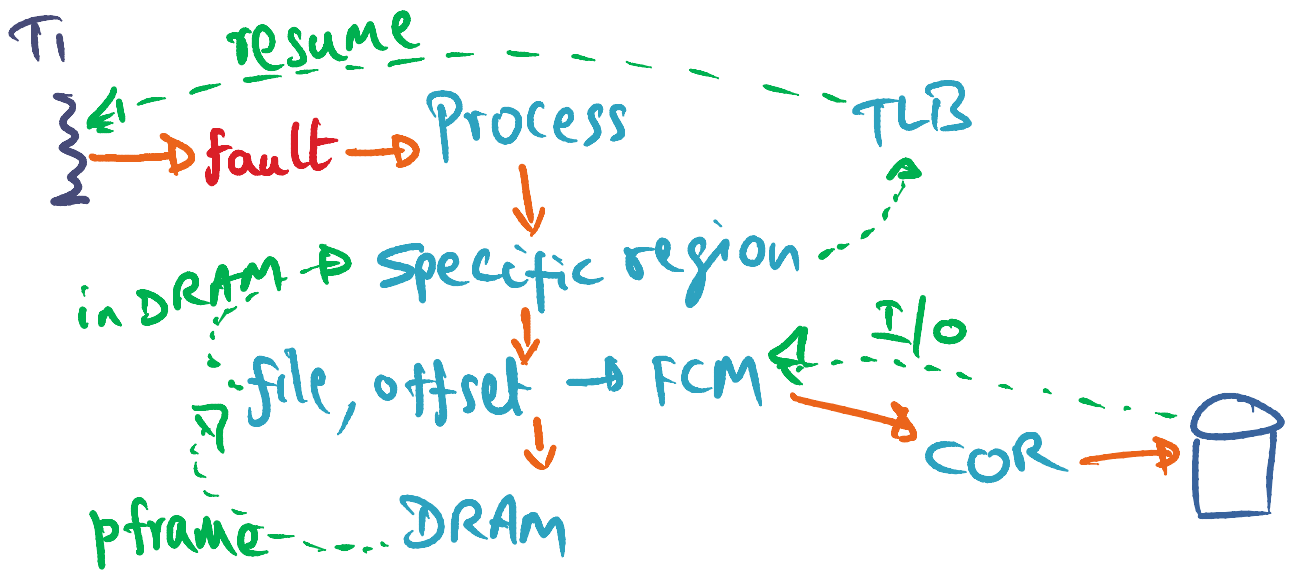
Group activity



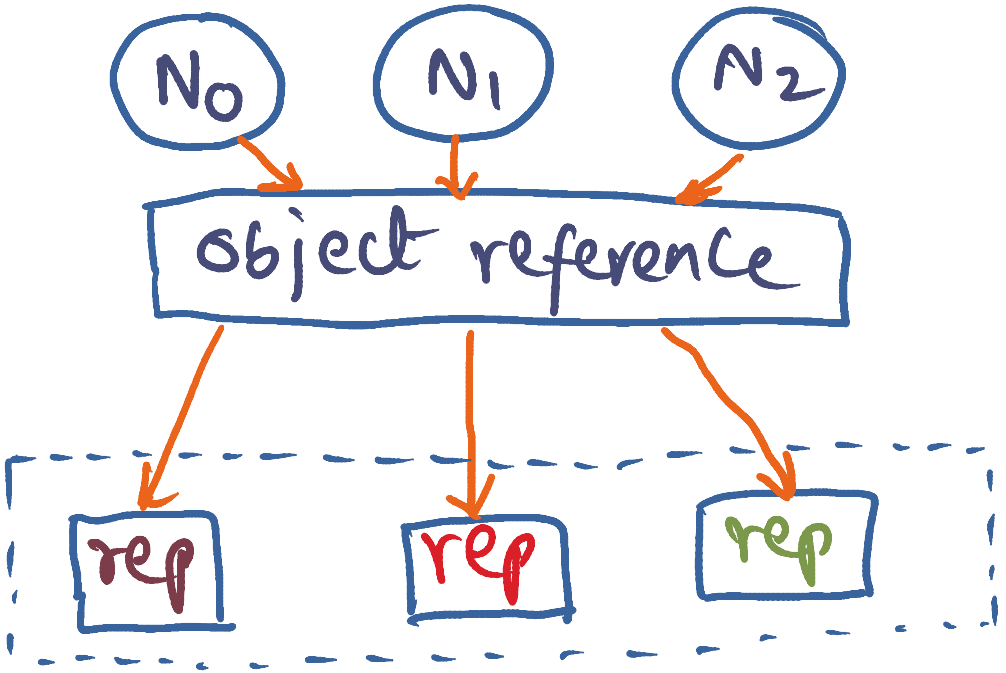
Starting with the above figure, discuss for each object in the path of the memory manager which clustering strategy (a single shared object, partial replication, full replication, partitioned reps) is most appropriate.

Objectized structure of VM manager





Advantages of clustered object



same object reference on all nodes

Allows incremental optimization

* usage pattern determines level of replication