Today
- LRVM
- RioVista
- Hg

Next week
- Internet scale computing (Lesson 9)

RVM primitives

Initialization

- initialize(options)
- map (region, options)
- unmap(region)

Body of server code

- begin_xact (tid, restore_mode)
- set_range(tid, addr, size)
- end_xact (tid, commit_mode)
- abort_xact (tid)

GC to reduce log space

- flush ( )
- truncate ( )

} done by LRVM automatically

$\Downarrow$

Provided for app flexibility

Miscellaneous

- query_options (region)
- set_options (options)
- create_log(options, len, mode)

$\Rightarrow$ Simplicity – small set of primitives

How the server uses the primitives

```
Initialize address space from Ext Segs
    ⋮
begin_xact(tid, mode);
    set range (tid, base_addr, #bytes);
        ⋮
    write metadata m1                      ; // contained in range
        ⋮
    write metadata m2                      ; // contained in range
end_xact (tid, mode);                        // or this can be abort
```

How the Server uses the primitives

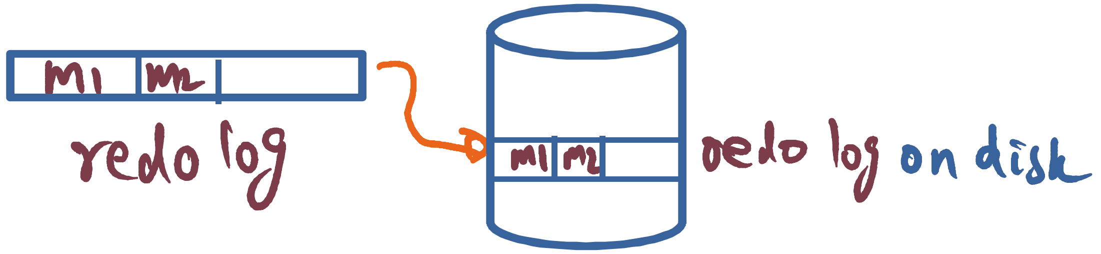
Initialize address space from Ext segs
⋮
begin_xact(tid, mode);
    set range (tid, base_addr, #bytes); → LRVM creates undo record
    ⋮
    write metadata m1 ; // contained in range
    ⋮
    write metadata m2 ; // contained in range
end_xact (tid, mode);   // or this can be abort ⇓ restore from undo record

No action by LRVM

LRVM → end_xact: LRVM gets rid of undo record

⇓

LRVM creates redo log in memory
- flush to disk sync or later depending on mode



Opportunities for Server to optimize transactions

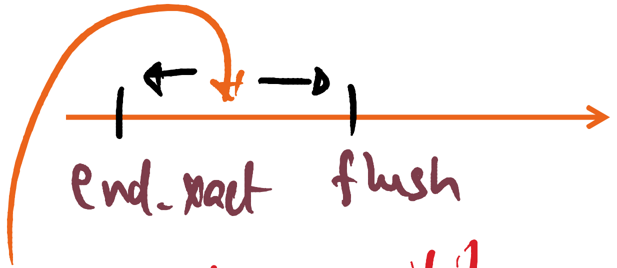
no_restore mode in begin_xact

- no need to create in-mem undo record

no_flush mode in end_xact

- no need to Sync flush redo log to disk

⟹ lazy persistence



⟹ upshot?

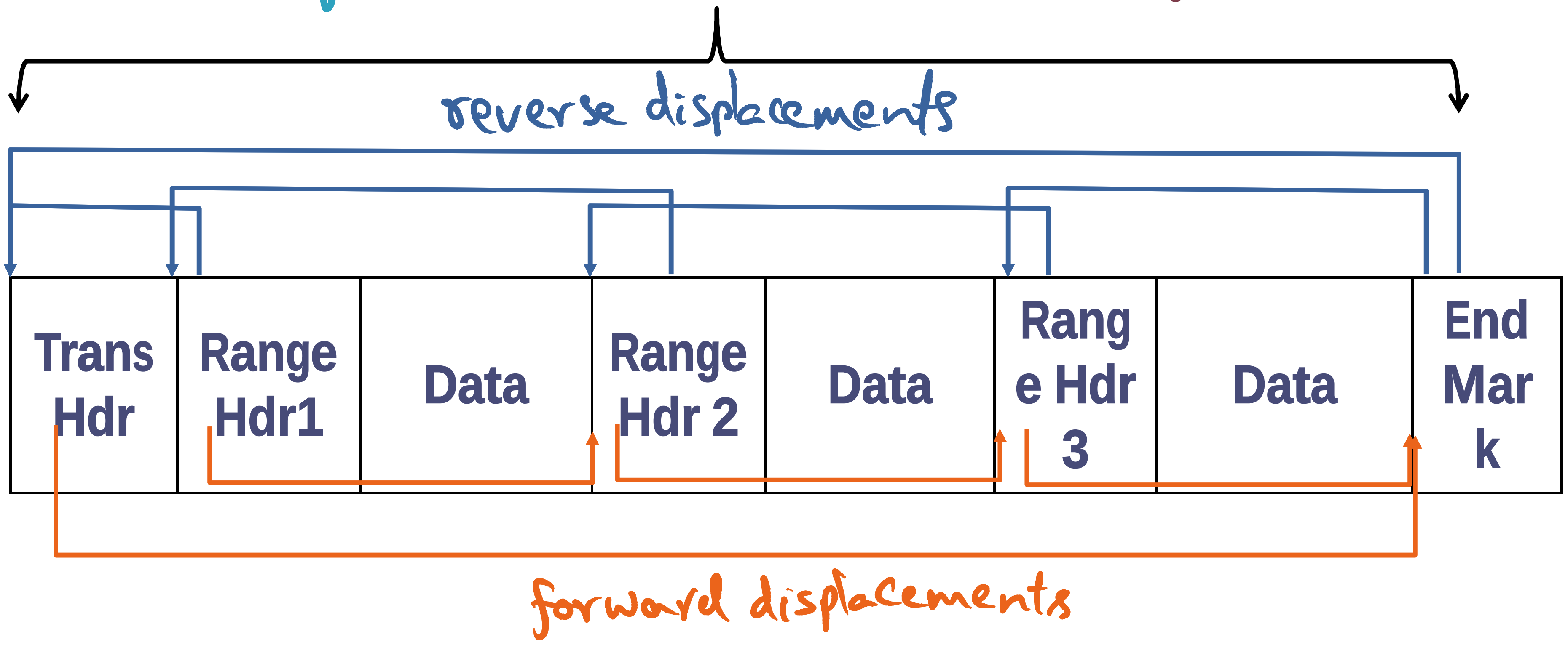
⟹ Window of Vulnerability

use xactions as insurance

Implementation

Redo log

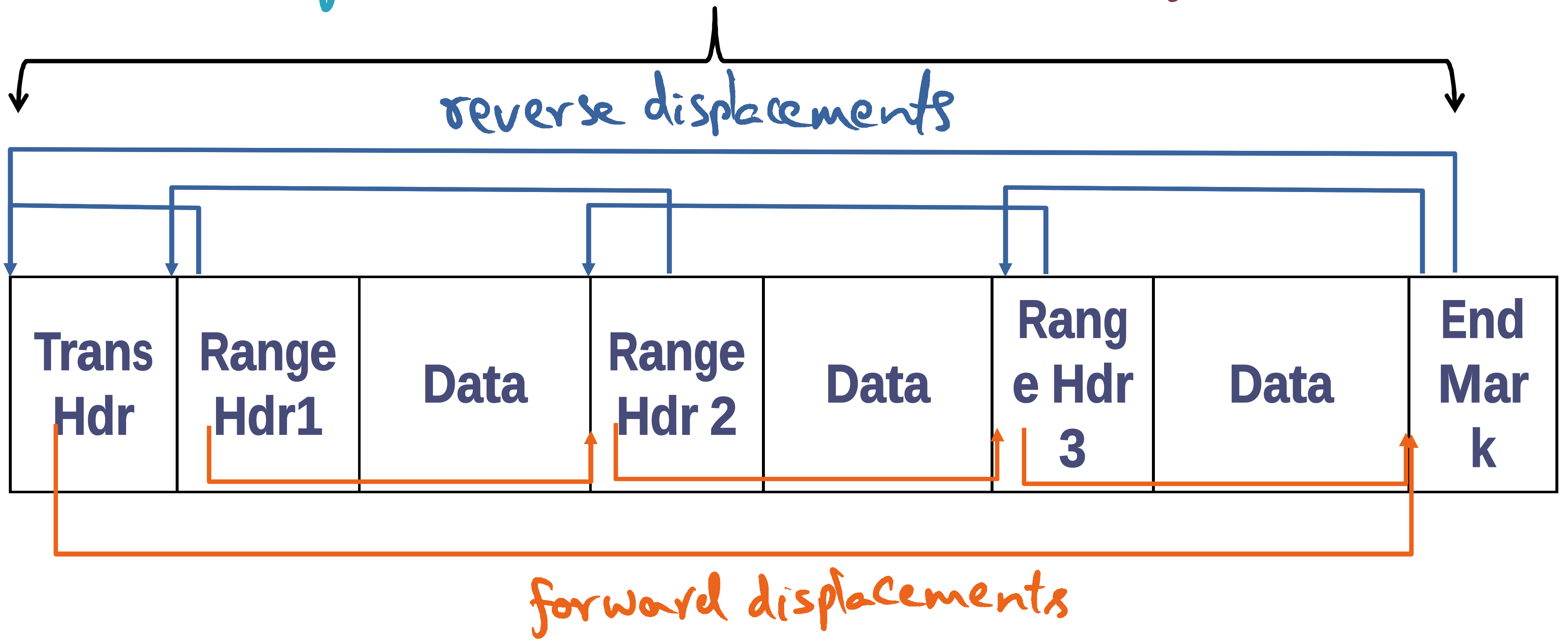
- All changes to different regions between begin and end xact



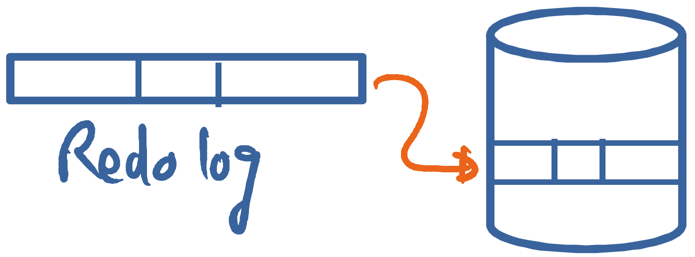
Implementation

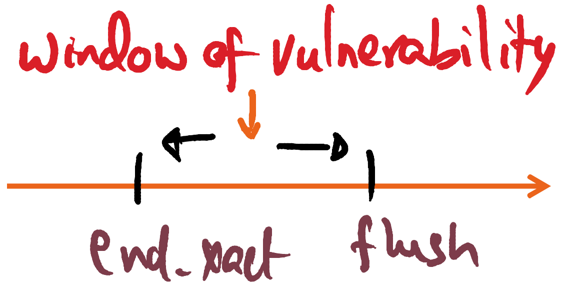
Redo log

- All changes to different regions between begin and end xact

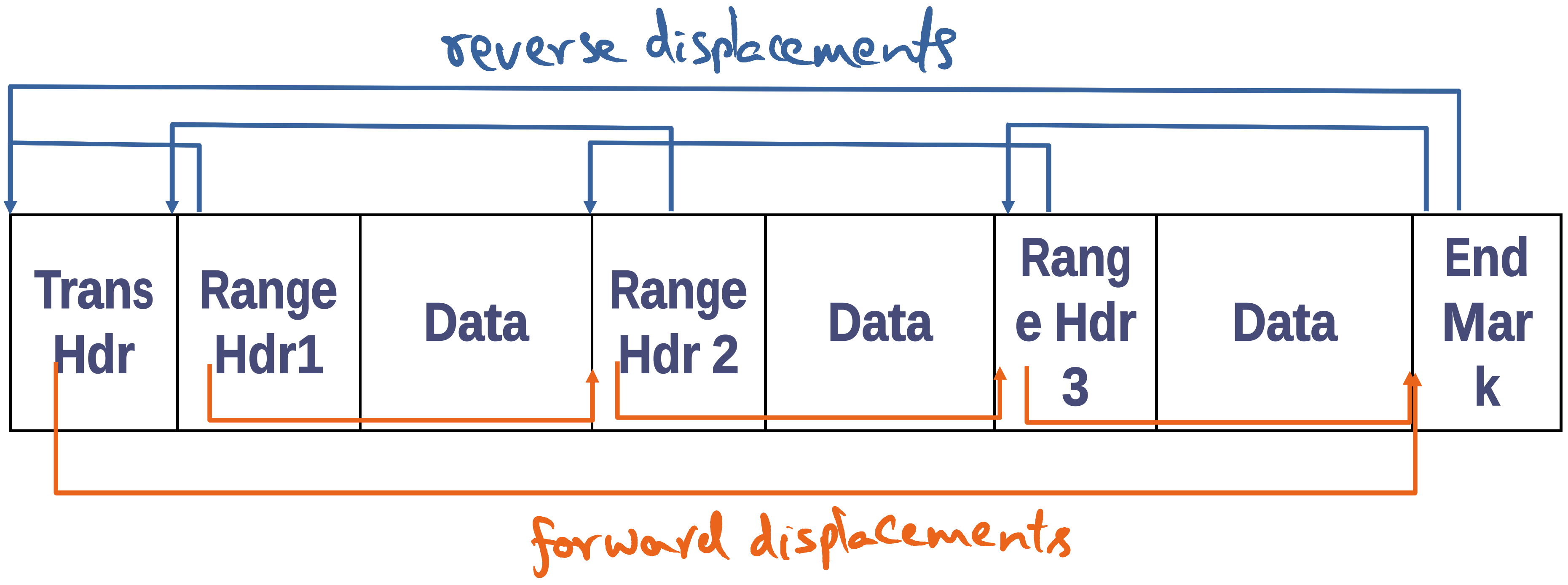


Commit

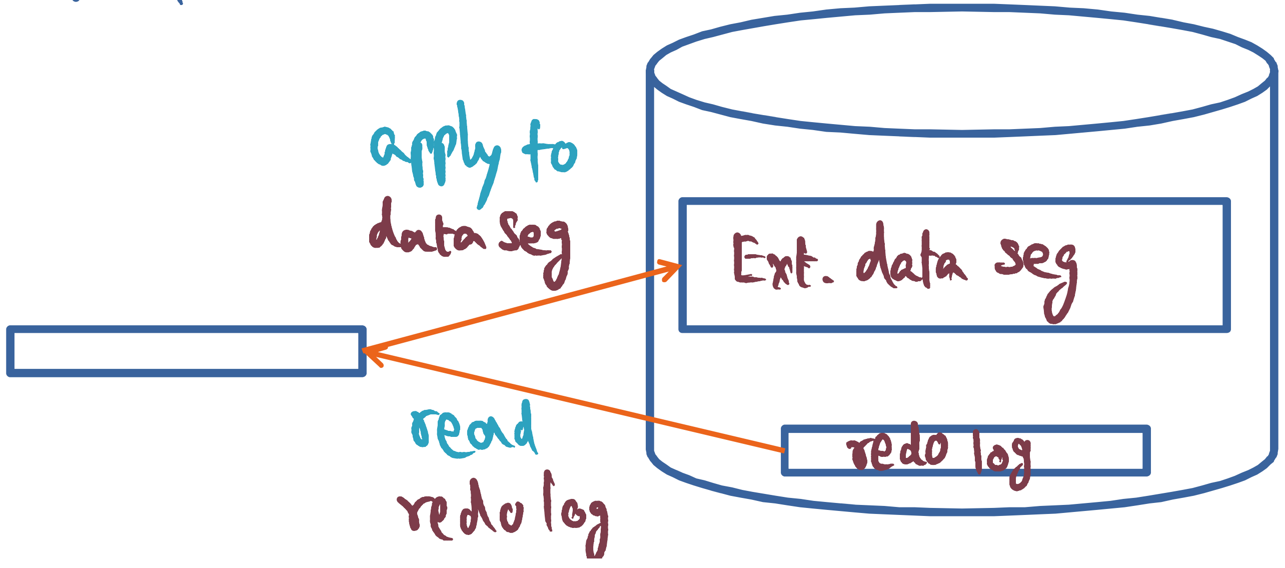




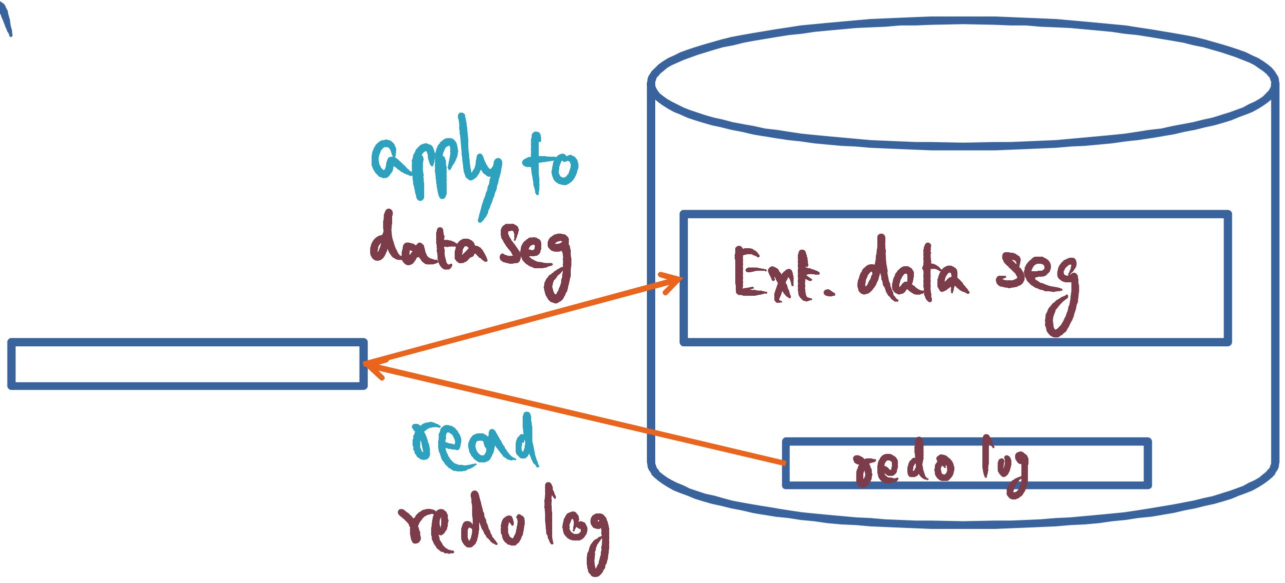
Crash Recovery



Resume from Crash

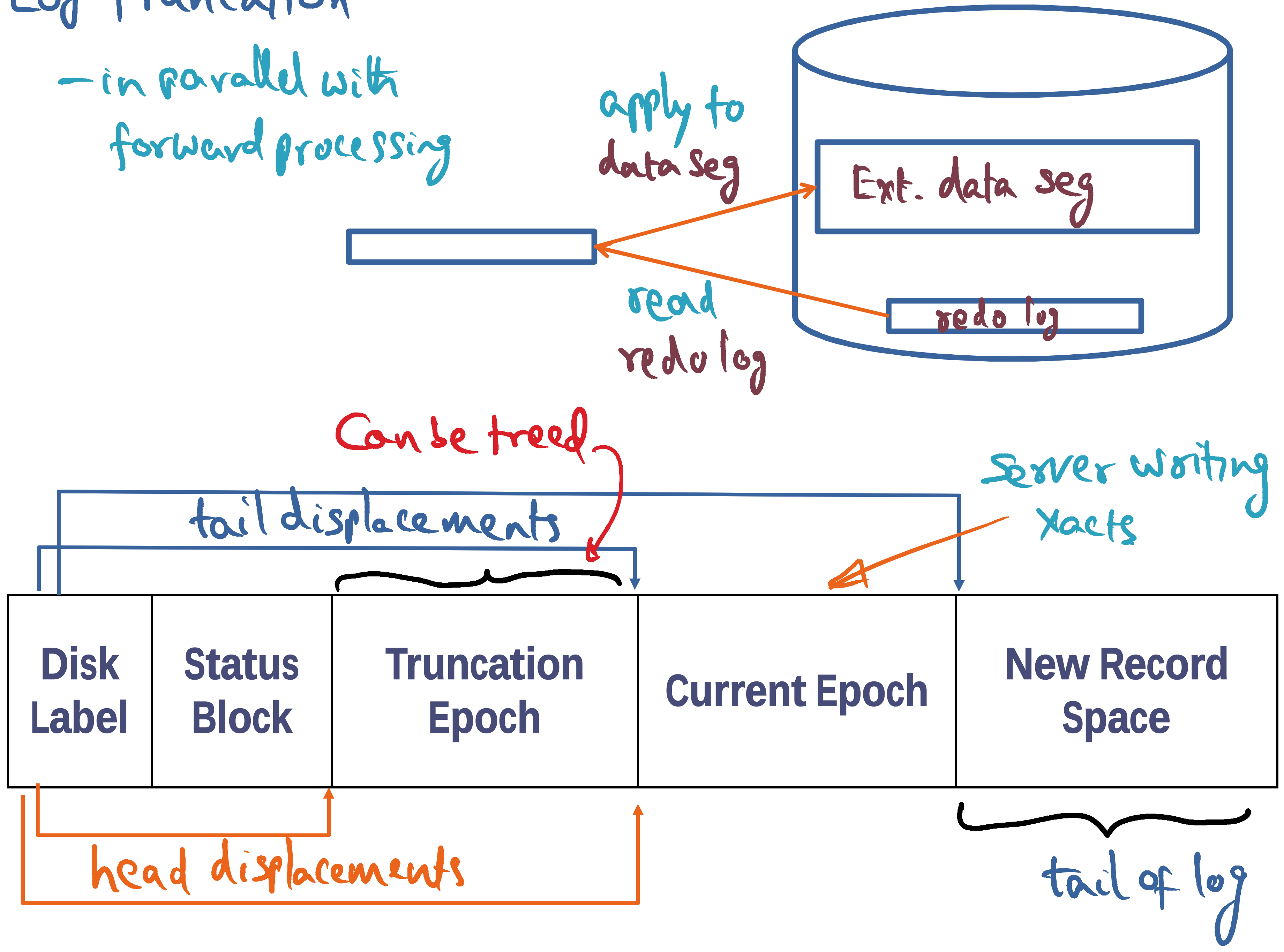


Log truncation



Log truncation

- in parallel with forward processing



1) Difference between

Traditional transaction

Vs

LRVM transaction

2) Why the difference

Key Takeaways

- Classic systems research.
  - Understand pain point for developers
  - Solution to solve that pain point
- Managing persistence for critical data structures is the pain point identified by LRVM.
  - LRVM proposes using “light weight” transactions (without all the ACID properties)