Logstor
A log-structured virtual storage

Wuyang Chung
wy-chung@outlook.com

AsiaBSDCon 2026

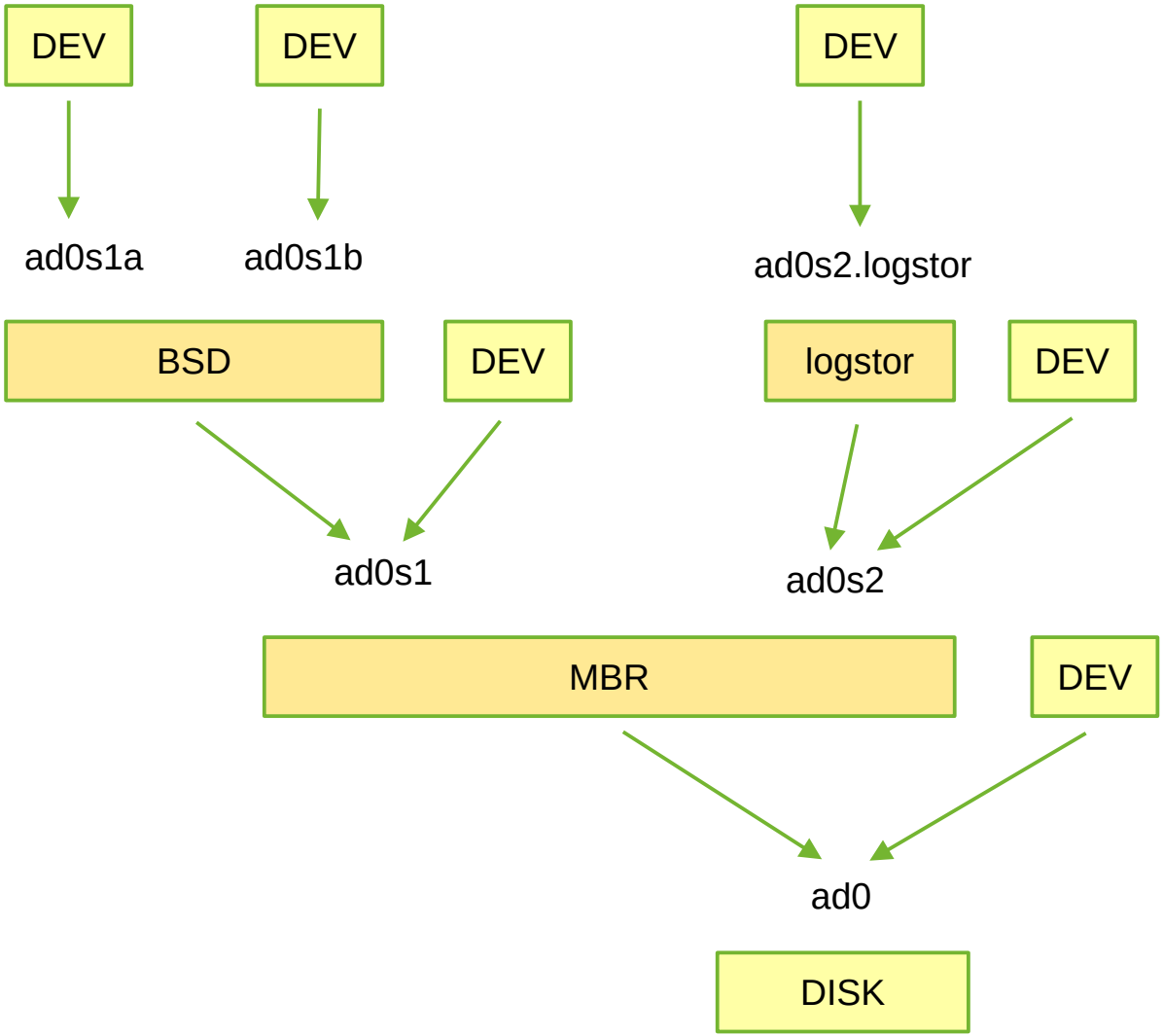
Differences with the previous version

- Previous version runs in user-level using GEOM gate and this version runs in kernel.
- Remove garbage collection and replace it with Stale Sector Recycle (SSR).
- Add new logstor commands *snapshot* and *rollback*.

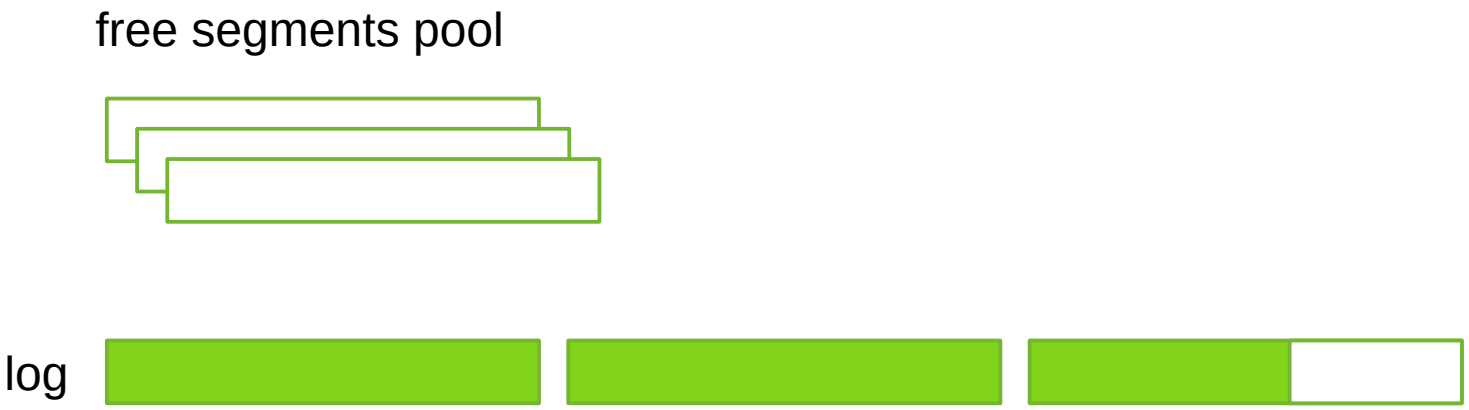
Outline

- Introduction of GEOM
- Introduction of log-structured storage
- Implementation
- Demo

GEOM


DEV
DEV
DEV
ad0s1a
ad0s1b
ad0s2.logstor
BSD
DEV
logstor
DEV
ad0s1
ad0s2
MBR
DEV
ad0
DISK

Log-structured storage (1)



- Blocks are appended to the end of the log.

Log-structured storage (1)

- Upon reaching the end of the log, allocate a segment from the free pool.
- Add it to the end of the log.

Log-structured storage (1)

- When the number of free segments in the pool drops below a threshold, trigger garbage collection.
  - Collect the live sectors and append them to the end of the log.

Stale Sector Recycle (SSR)

- Due to the high overhead of traditional garbage collection, this version implements Stale Sector Recycling (SSR).

Stale Sector Recycle (SSR)

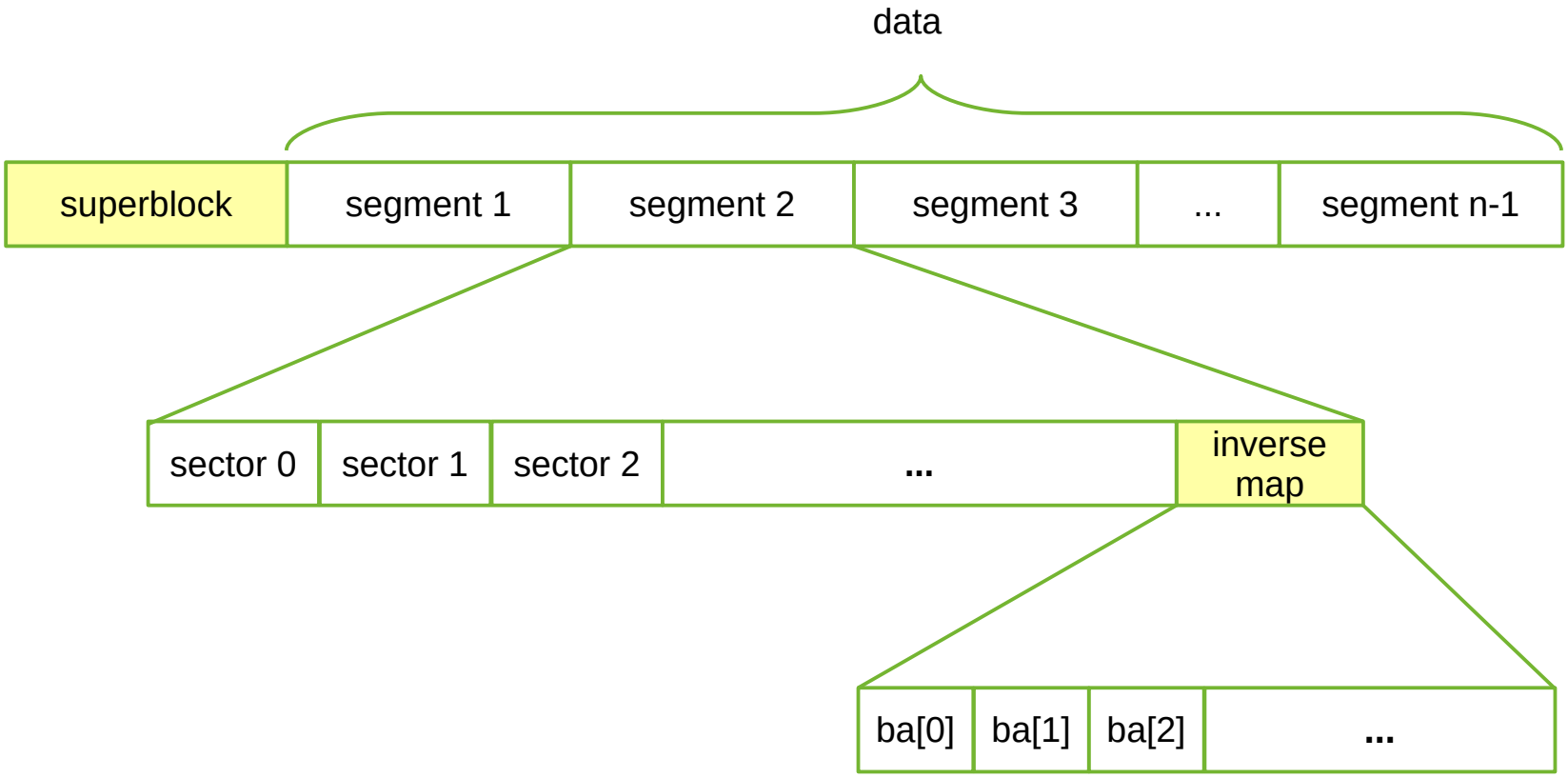
- Append data directly on stale sectors.

Implementation

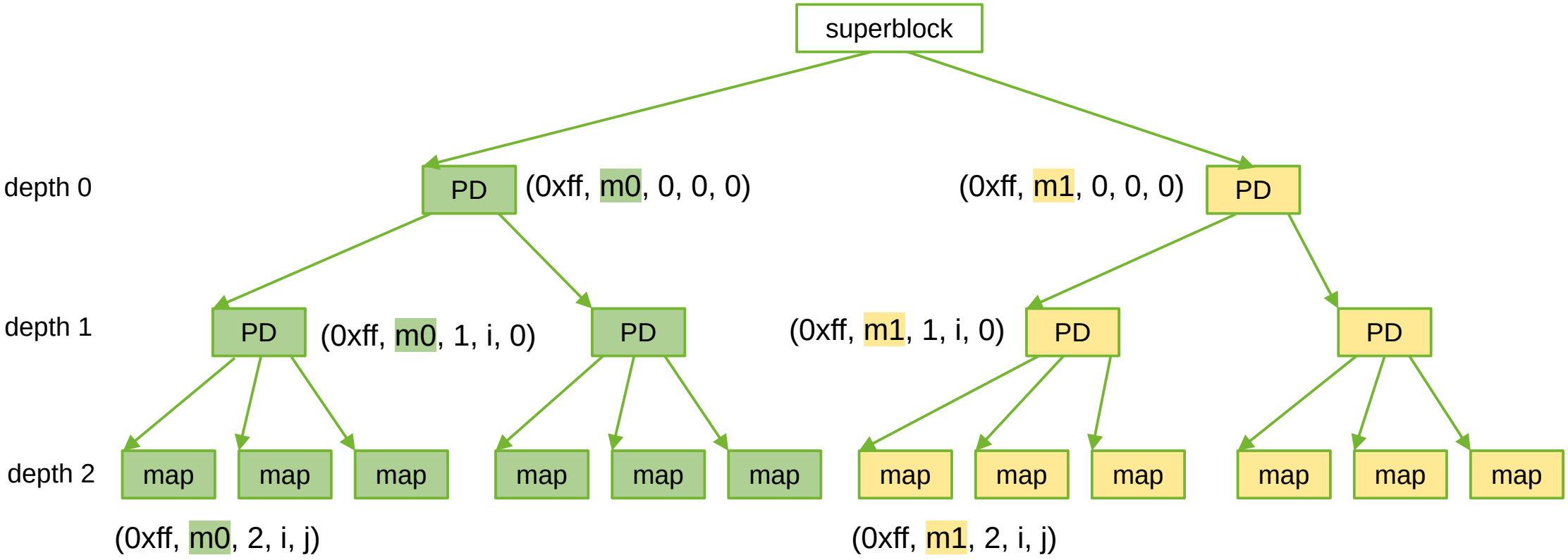
Block : virtual sector to upper layer
Sector : physical sector to lower layer

- Three metadata
  - Forward map
    - From block address to sector address
  - Inverse map
    - From sector address to block address
  - Superblock
- Buffer cache for forward map

Implementation
Superblock and inverse map



Implementation
Forward map (1)



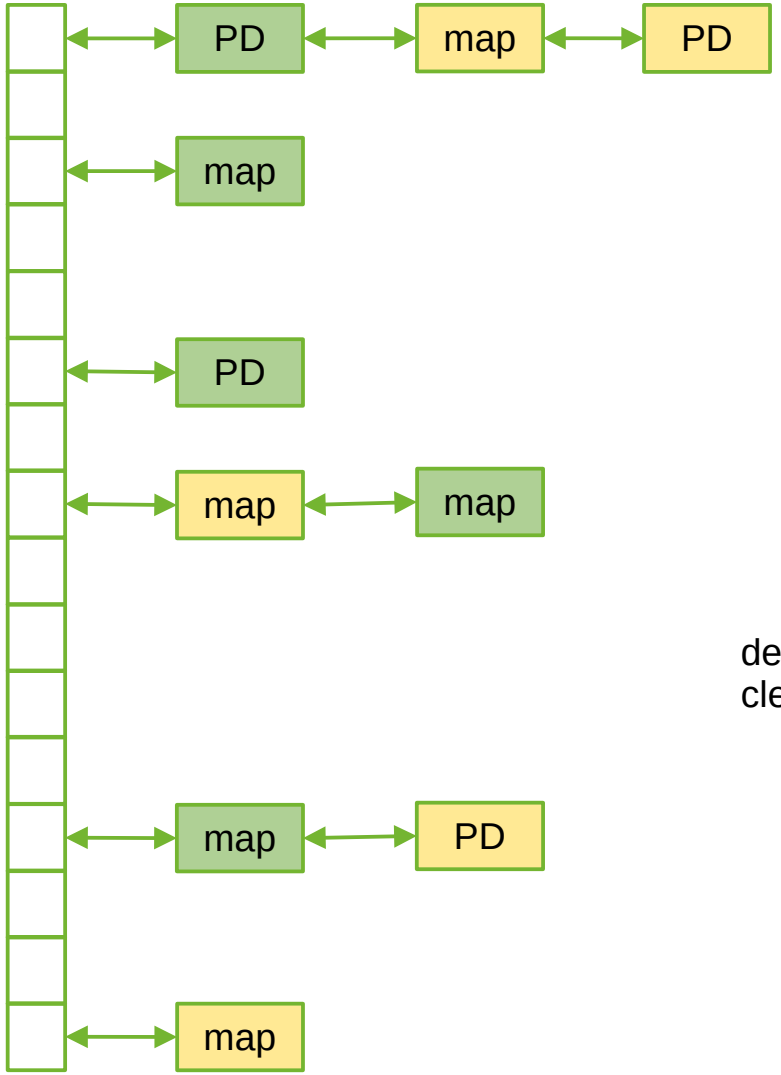

```
bool stale =    forward(inverse(s)) != s;
```

Implementation
Forward map (2)

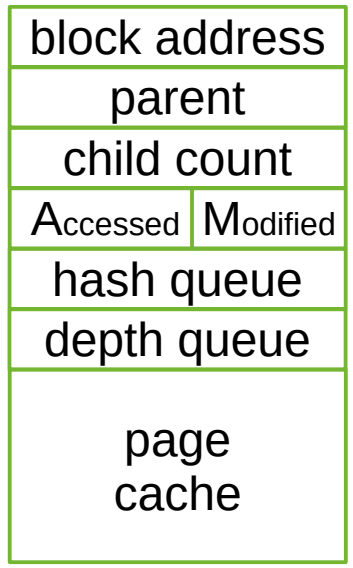
- Assign unique block address for page table pages.
- Block address encoding for page table pages
  - Field 1: A constant prefix (0xFF) identifying the block as a page table page.
  - Field 2: The map number (two bits).
  - Field 3: The depth of the page within the hierarchy.
  - Field 4 & 5: The entry index numbers for the Page Directories (PD).

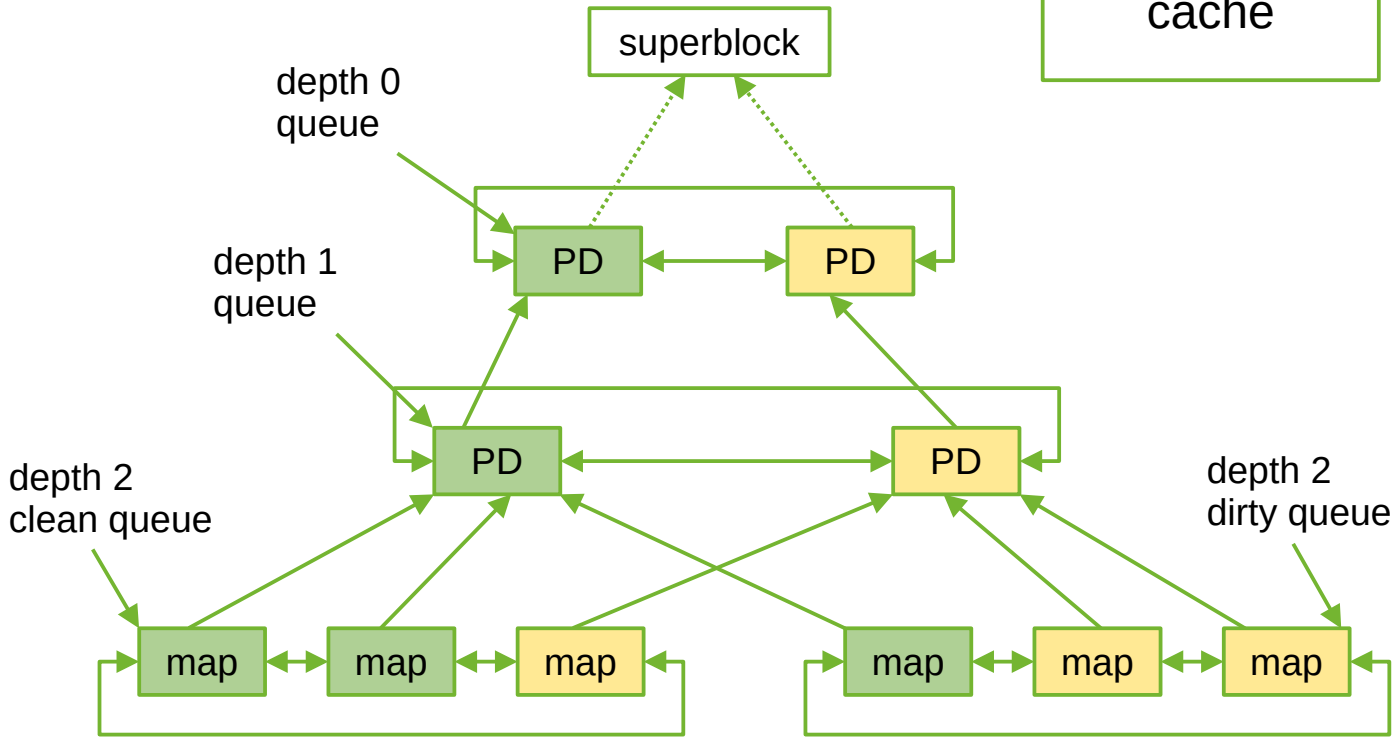
Implementation
Buffer cache

hash table (with chained addressing)

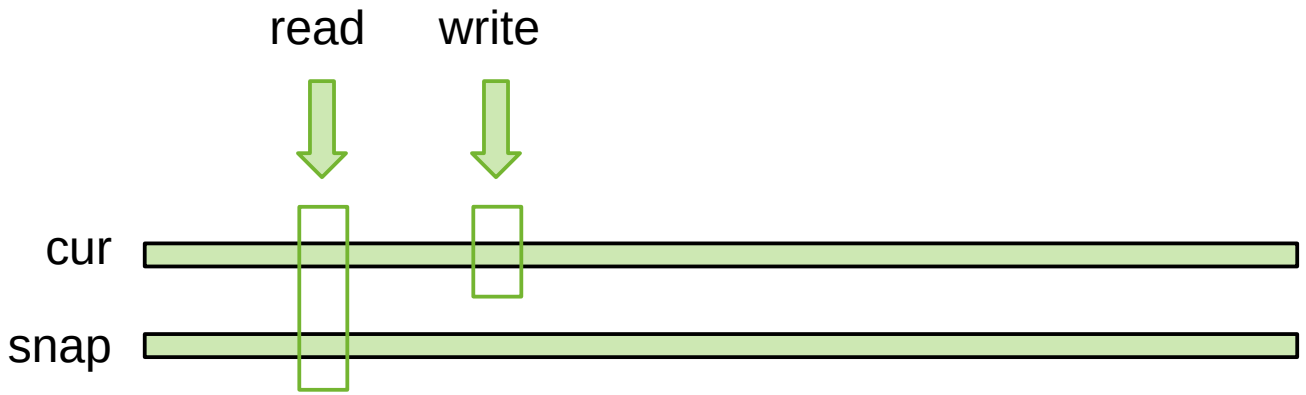


buffer





Read/write operations
(when not doing snapshot)



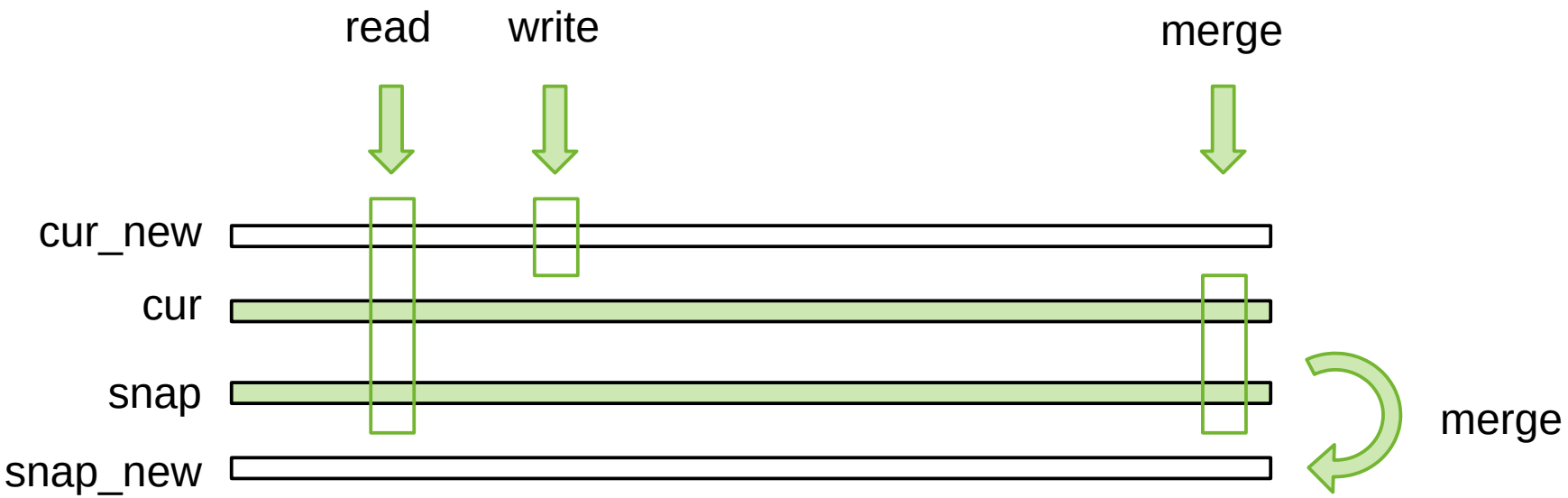

```
uint32_t block_to_sector(b)
{
    If (cur[b] != null)
        return cur[b];
    if (snap[b] != null)
        return snap[b];
    return null
}
```

```
update(b, s)
{
    cur[b] = s;
}
```

Rollback

- Clear cur map

Snapshot begin



- Create cur_new and snap_new
- Merge cur and snap to snap_new

Snapshot end

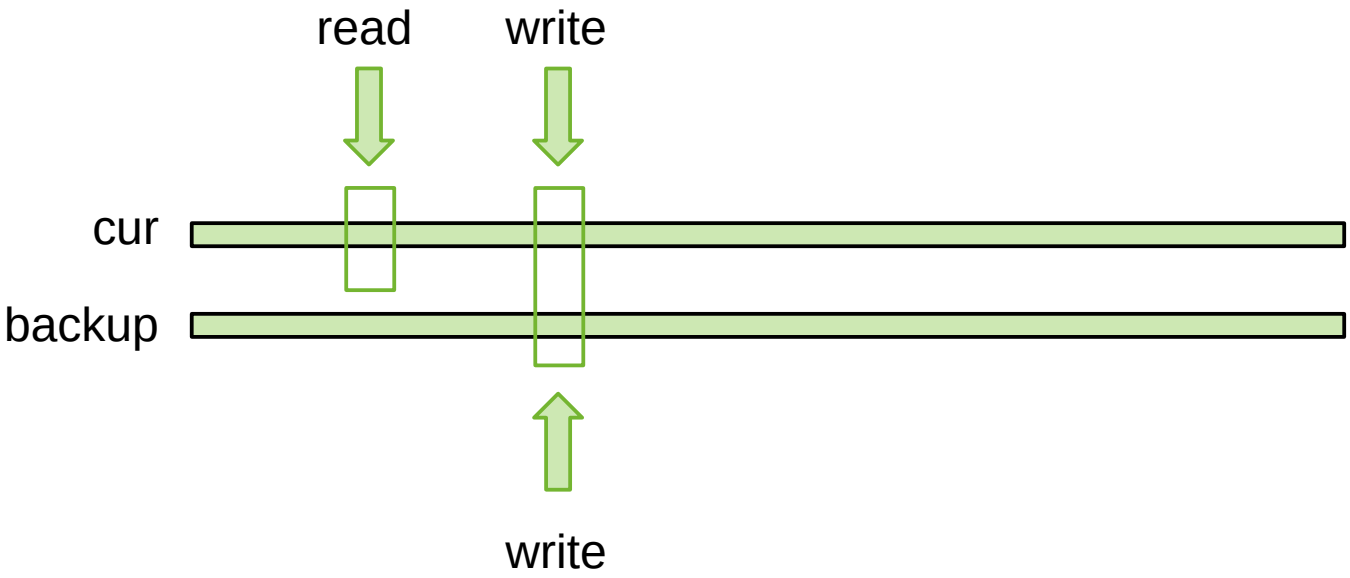
- Rename cur_new to cur
- Rename snap_new to snap
- Delete previous cur and snap

Stale Sector Recycle (SSR)

```
bool stale(s)
{
    b = inverse(s);
    for (i = 0; i < n; ++i) {
        s1 = forward(i, b);
        if (s == s1)
            return false;
    }
    return true;
}
```

```
bool stale =    forward(inverse(s)) != s;
```

Alternative approach
Read/write



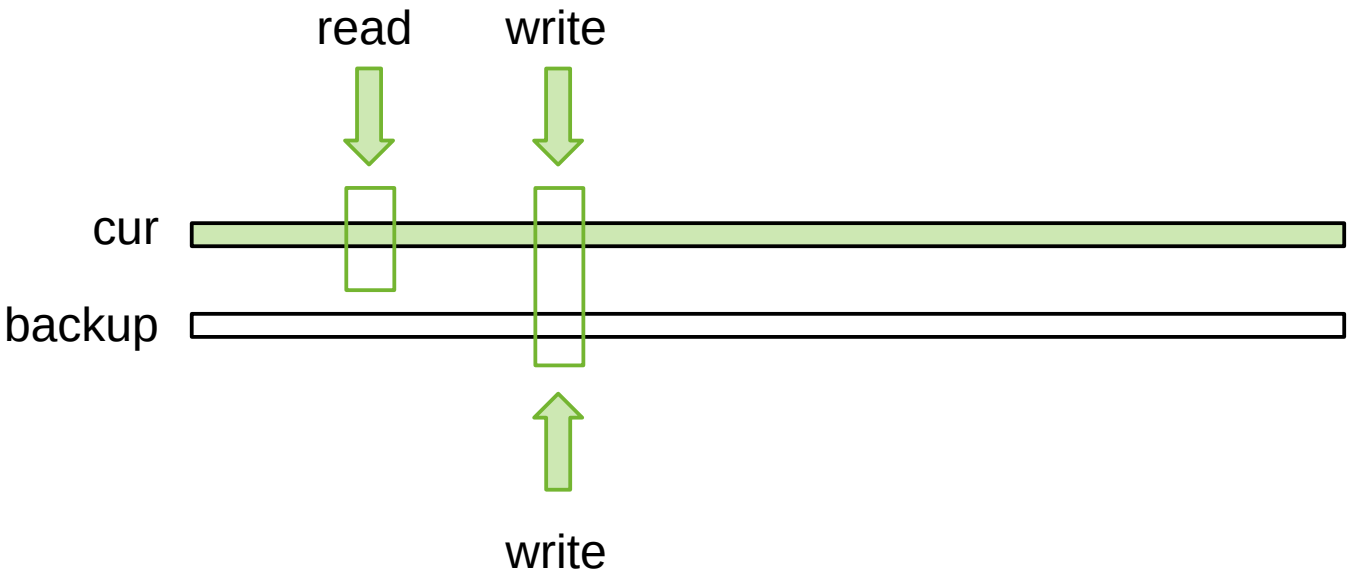

```
update(b, s) // update block to sector mapping
{
      If (cur[b] != null && backup[b] == null)
            backup[b] = cur[b];

      cur[b] = s;
}
```

Alternative approach
Rollback


```
rollback()
{
      for (i = 0; i < block_max; ++i)
            If (backup[i] != null)
                  cur[i] = backup[i];
      clear_map(backup);
}
```

Alternative approach
Snapshot



- Clear backup map

Limitations

block size	segment size	max virtual storage size
512	2^7 * 2^9 = 64K	2^7 * 2^7 * 2^7 * 2^9 = 1G
1K	2^8 * 2^10 = 256K	2^8 * 2^8 * 2^8 * 2^10 = 16G
2K	2^9 * 2^11 = 1M	2^9 * 2^9 * 2^9 * 2^11 = 256G
4K	2^10 * 2^12 = 4M	2^10 * 2^10 * 2^10 * 2^12 = 4T

- Fixed with three-level page table and 4K block size
- The size of the virtual disk reduces dramatically with smaller block size.

Demo