Memory Management in RAMCloud

Steve Rumble
SEDCL Forum
February 2, 2012

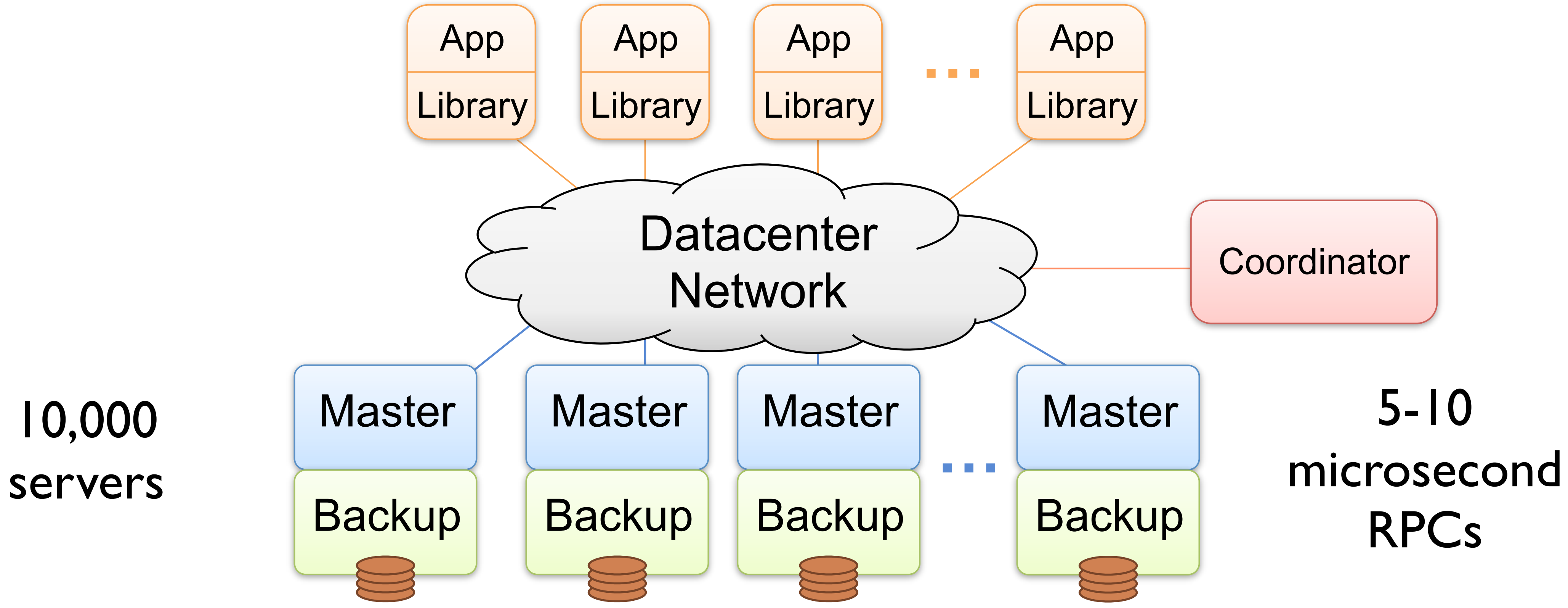
Overview

- RAMCloud stores objects in log-structured memory and uses cleaning to reclaim free space

- Cleaning and writing of new data occur in parallel for maximum performance

- The same log is scattered across the cluster for durability and fast recovery

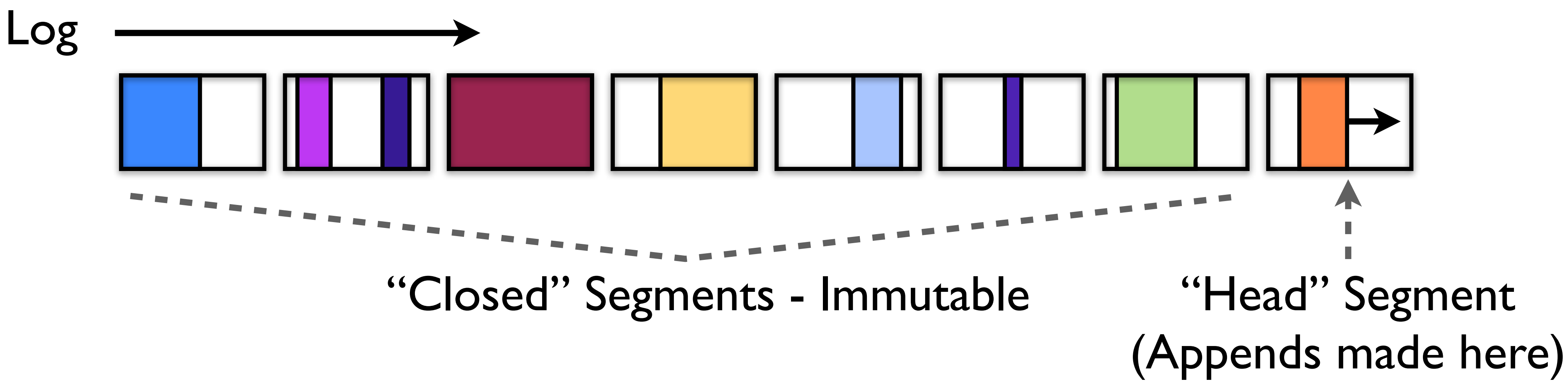
Quick RAMCloud Intro



- Distributed storage system for datacenters
- Design goals are large scale & low latency
- All data in DRAM at all times
- Replicate to remote disks for durability
- Simple key-value data model

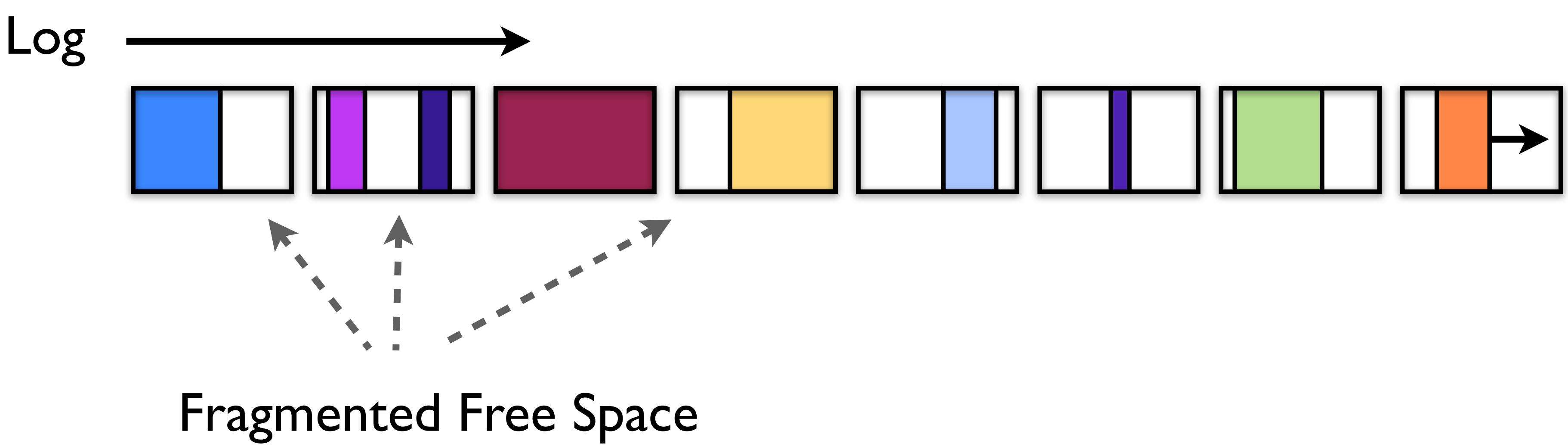
Log-structured Memory

- Each RC server stores objects in its log
  - Logically contiguous, made up of fixed-sized "segments"
  - Append-only. Deleted space reclaimed by "cleaning"

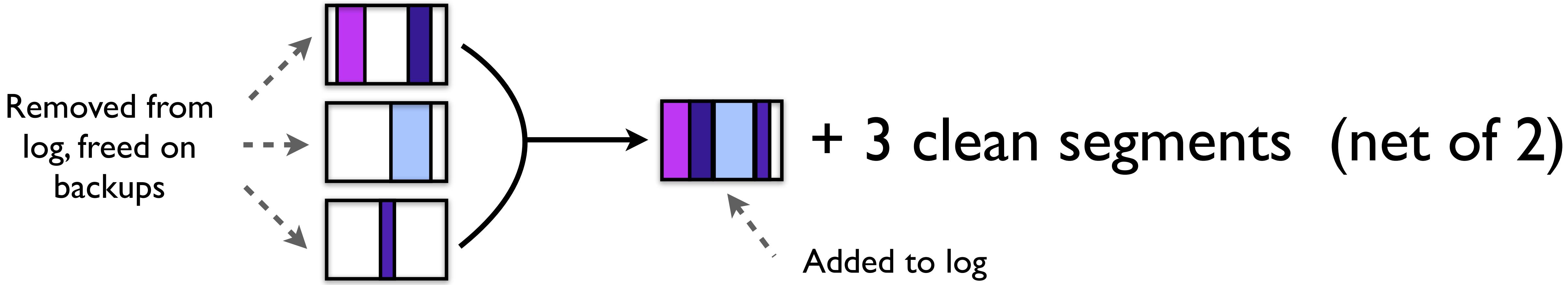


- Format identical in memory & on disks of remote backups

Log Cleaning



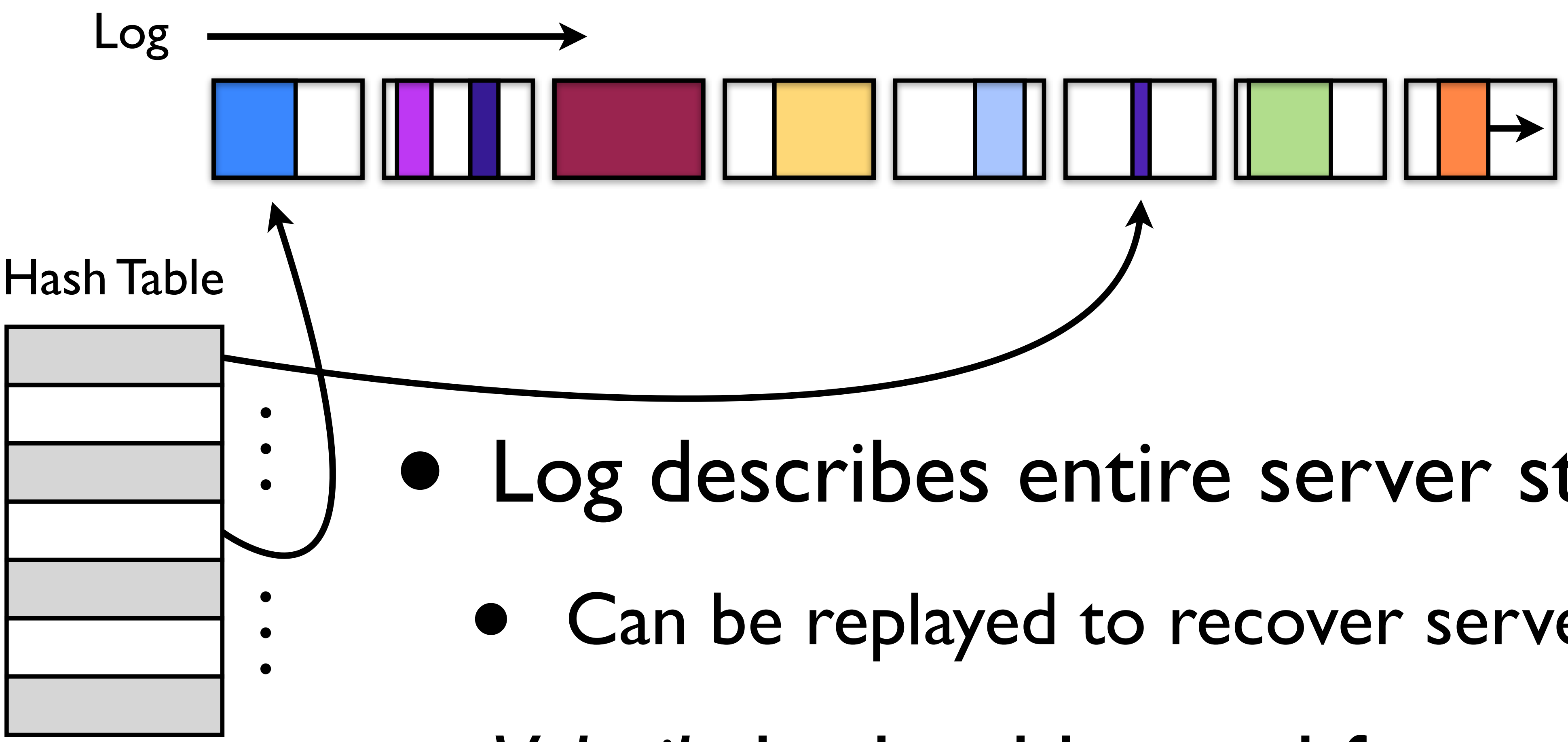
- When objects are deleted, segments become fragmented
- Cleaning reclaims space by writing contents of *N* old segments into < *N* new ones


Removed from log, freed on backups
+ 3 clean segments (net of 2)
Added to log

Why Log + Cleaning?

1. Efficient use of disks on backups
   - High bandwidth for disk writes during normal operation
   - Fast log replay during failure recovery
2. Simplicity: common disk & RAM format
3. Simpler consistency
   - Updates to head of log only
4. Fast allocation with good utilization
   - Fundamental trade-off

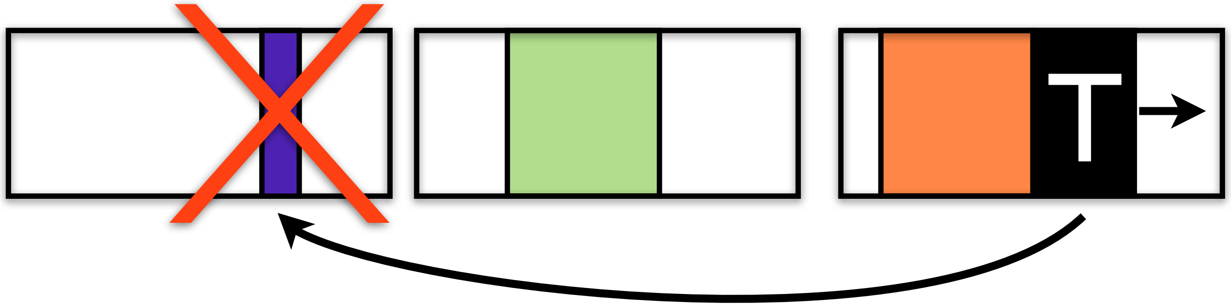
Hash Table



- Log describes entire server state
  - Can be replayed to recover server
- *Volatile* hash table used for:
  - Object lookups   (“read X”)
  - Object liveness checks (“was X deleted?”)
  - Indirection (reorganize memory at will)

Deletions

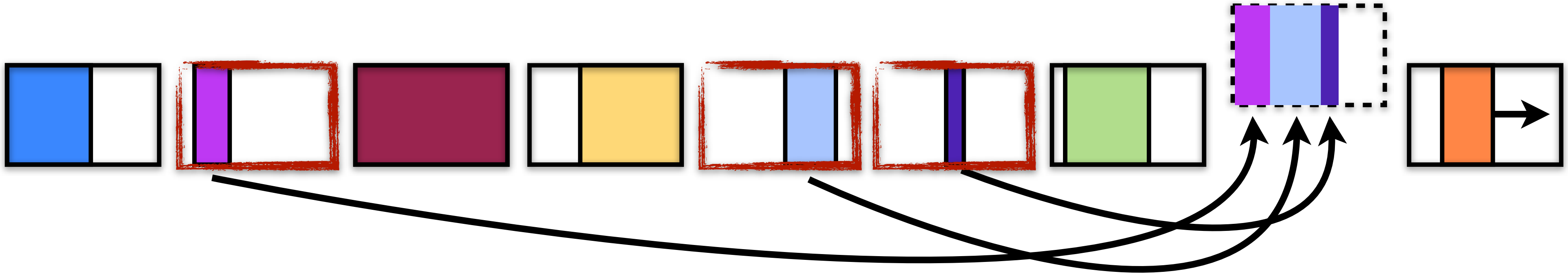
- Deleted objects removed from hash table
  - But remain in log until cleaned
- Must not reincarnate objects after server failure when replaying the log
  - The hash table is not persisted
  - Instead, a delete record (“tombstone”) is written for each deleted object



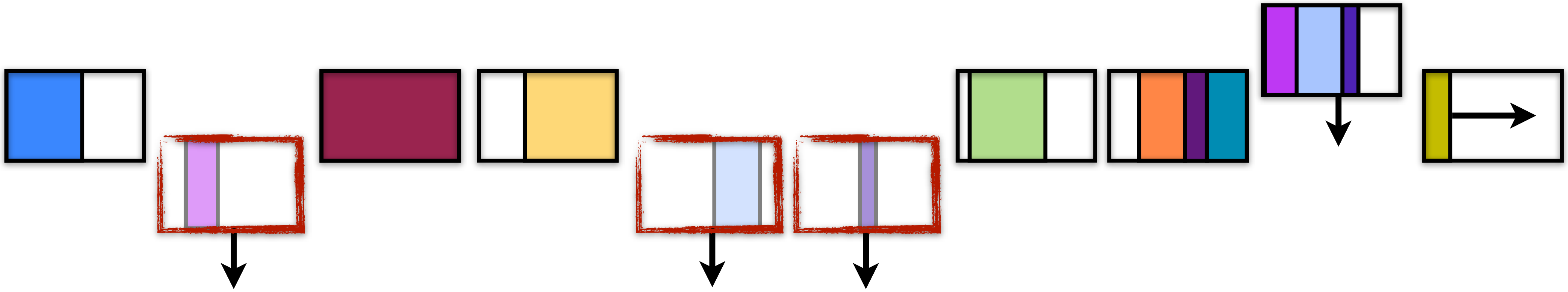
- Tombstones eligible for removal after dead object cleaned

Parallel Cleaning

- Cleaning and regular object writes occur in parallel



- Cleaned segments are freed when new, compacted segments join the log

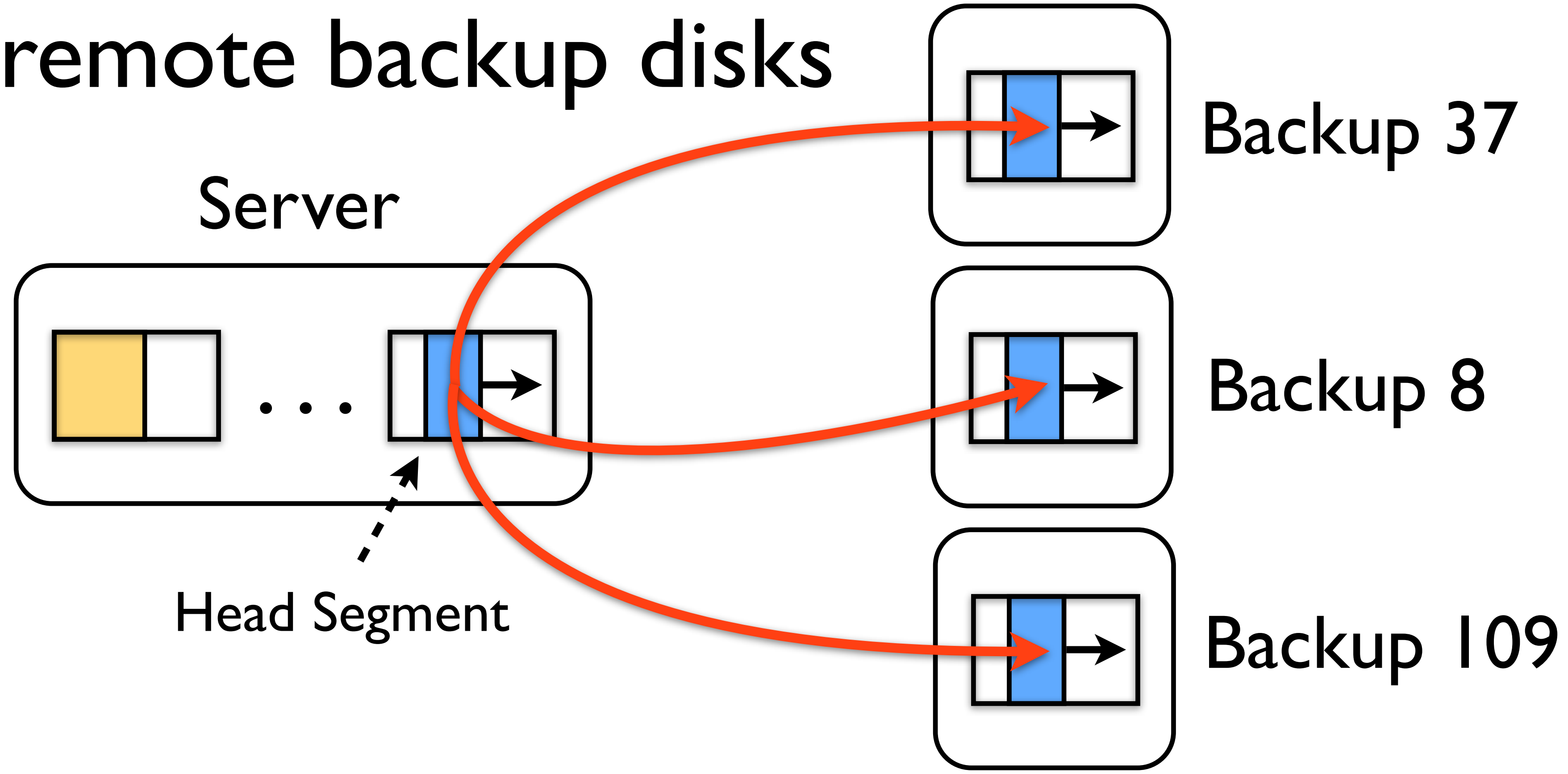


Cleaned segments removed

Copied live data added

Distributed Log

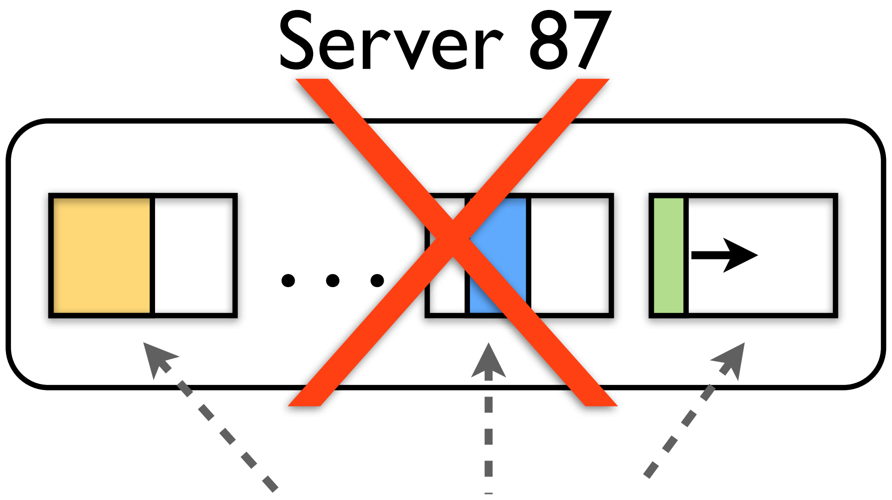
- Each in-memory log is distributed across many remote backup disks



  - Updated synchronously with in-memory copy
  - Segments scattered across different backups for maximum bandwidth during recovery
  - Multiple replicas of each segment for durability

Server Recovery

- When a server fails, its data is spread across the cluster of backup servers

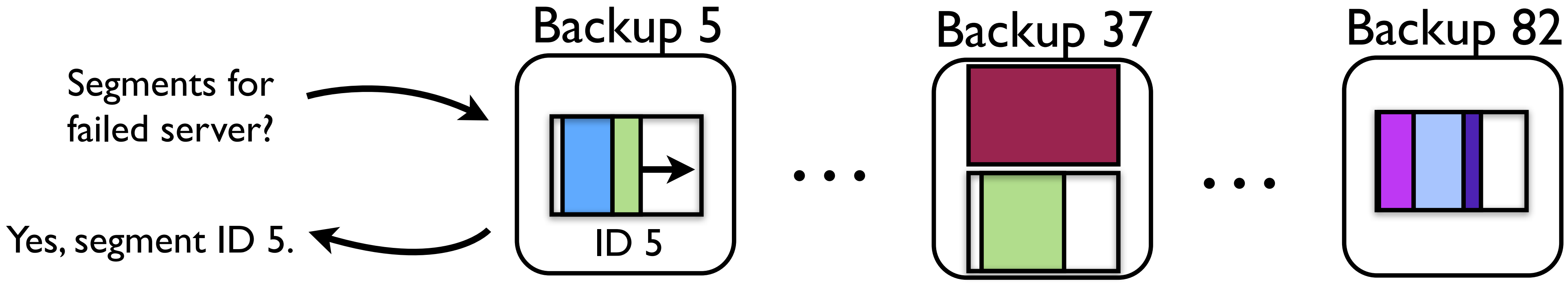


Segment replicas could be anywhere

- Problem:
  - How do we find the log?

Finding the Log

- Ask each backup for list of segments it has for the failed server



- Problem:
  - List of all segments could be too big or small
    - Too big: Old segments that were since cleaned
    - Too small: Lost replicas, cannot recover yet

Exact Segment List

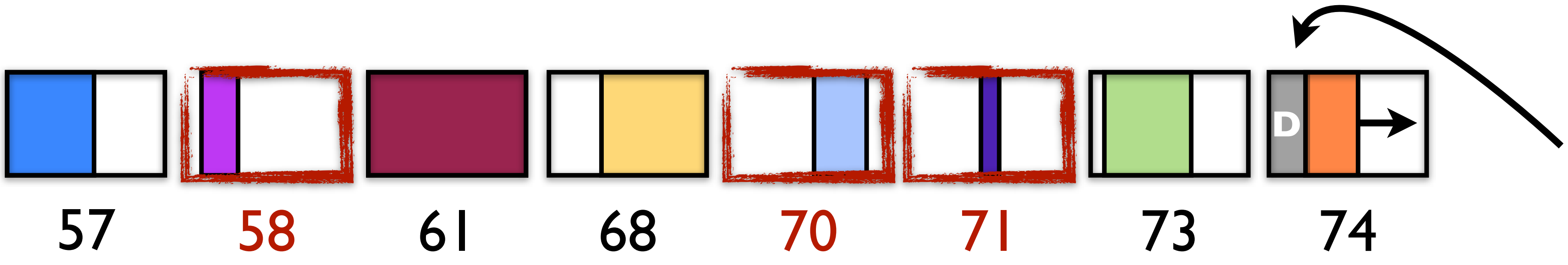
- How do we get exact list of needed segments?
- Two pieces to solution:

1. Make head segments describe the entire log
   - “Log digest”: list of constituent segments

2. Ensure we can always find head of the log
   - Or discover its missing when data loss

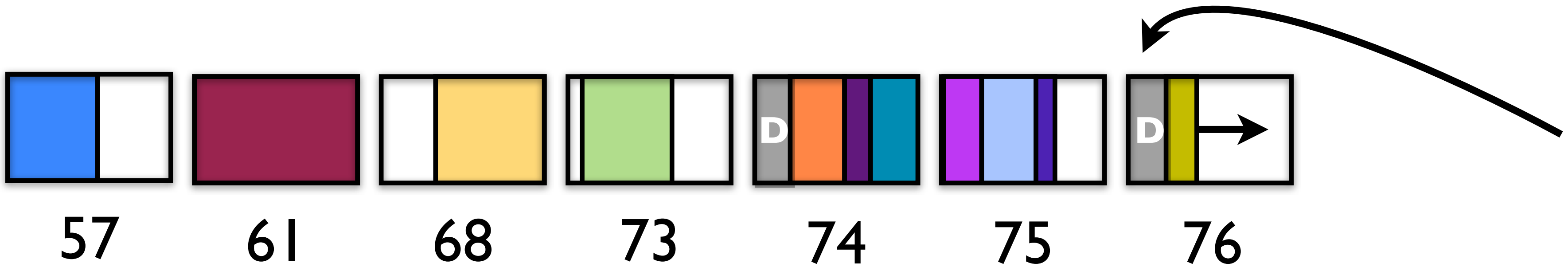
Log Digests

- Head segment enumerates all other segments
  - "Log Digest"
- When new head segment opened, new digest written to new segment



Digest:
(57, 58, 61, 68,
70, 71, 73, 74)

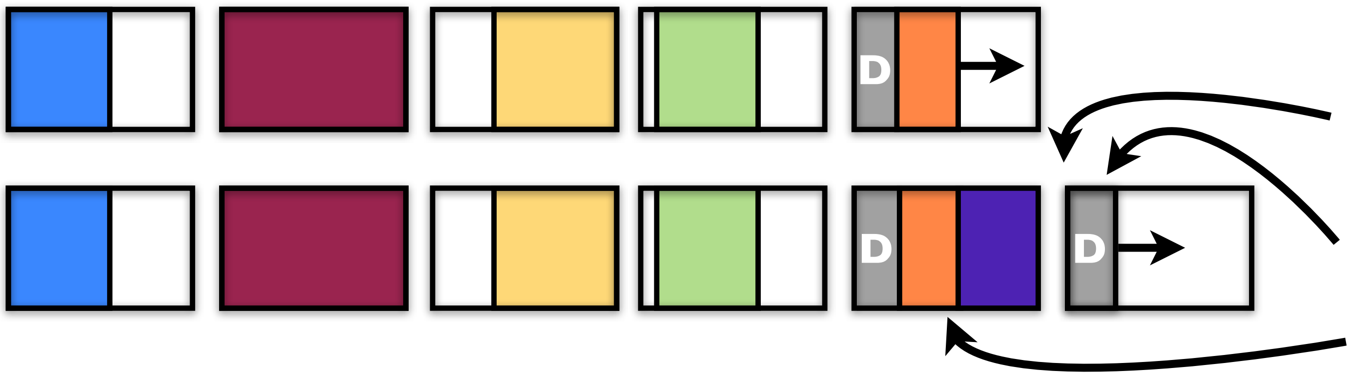
- New digests also used to replace cleaned segments with compacted ones



Digest:
(57, 61, 68, 73,
74, 75, 76)

Open Before Close

- Finding head of the log
  - Segments have two states: open, closed
  - There must always be a head (open) segment
    - If no open segments found, data was lost
- Open before close: To create new head
  - Open new segment, write digest, close old head before handling next write
  - Always 1 or 2 different open head segments



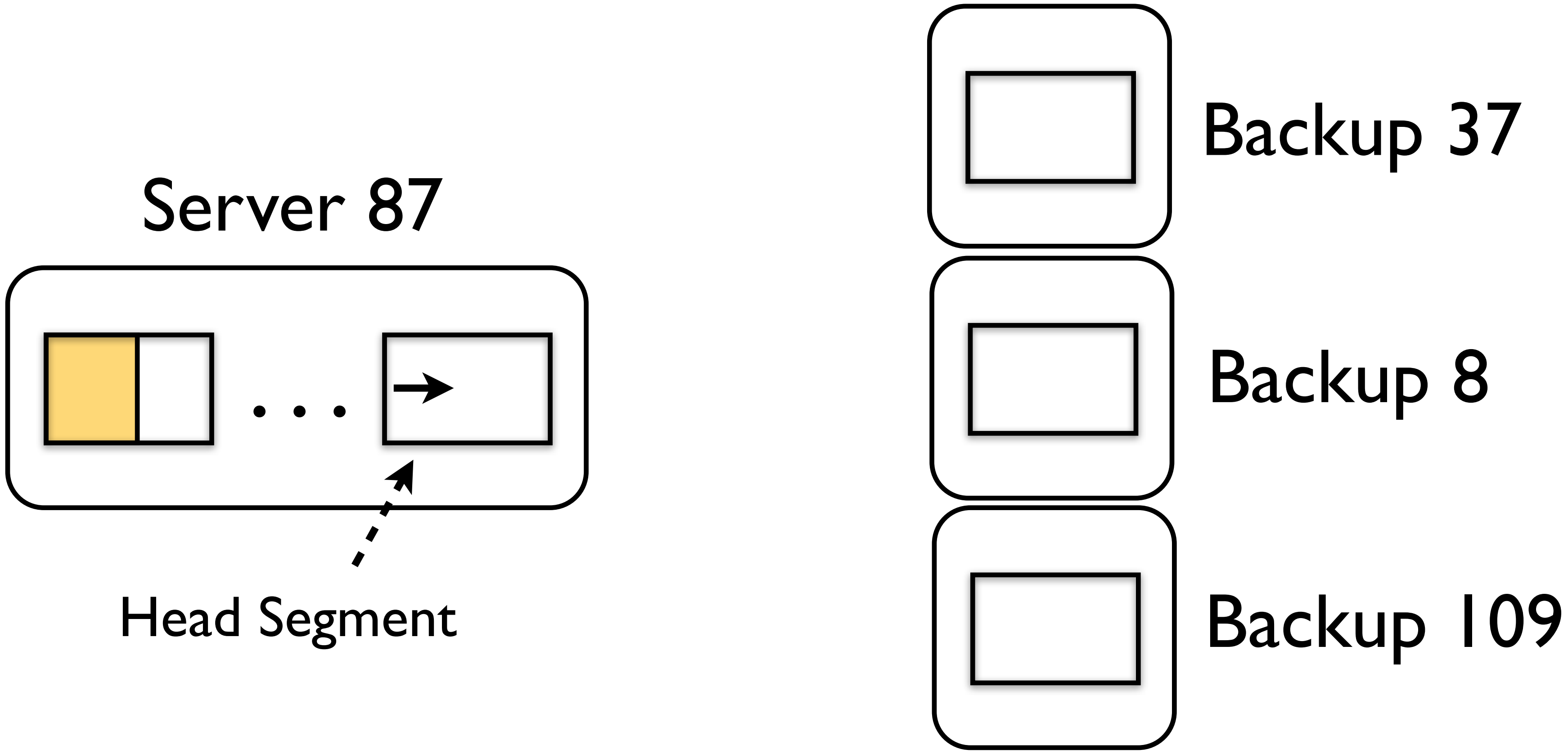
1. Open new head

2. Write digest

3. Close old head

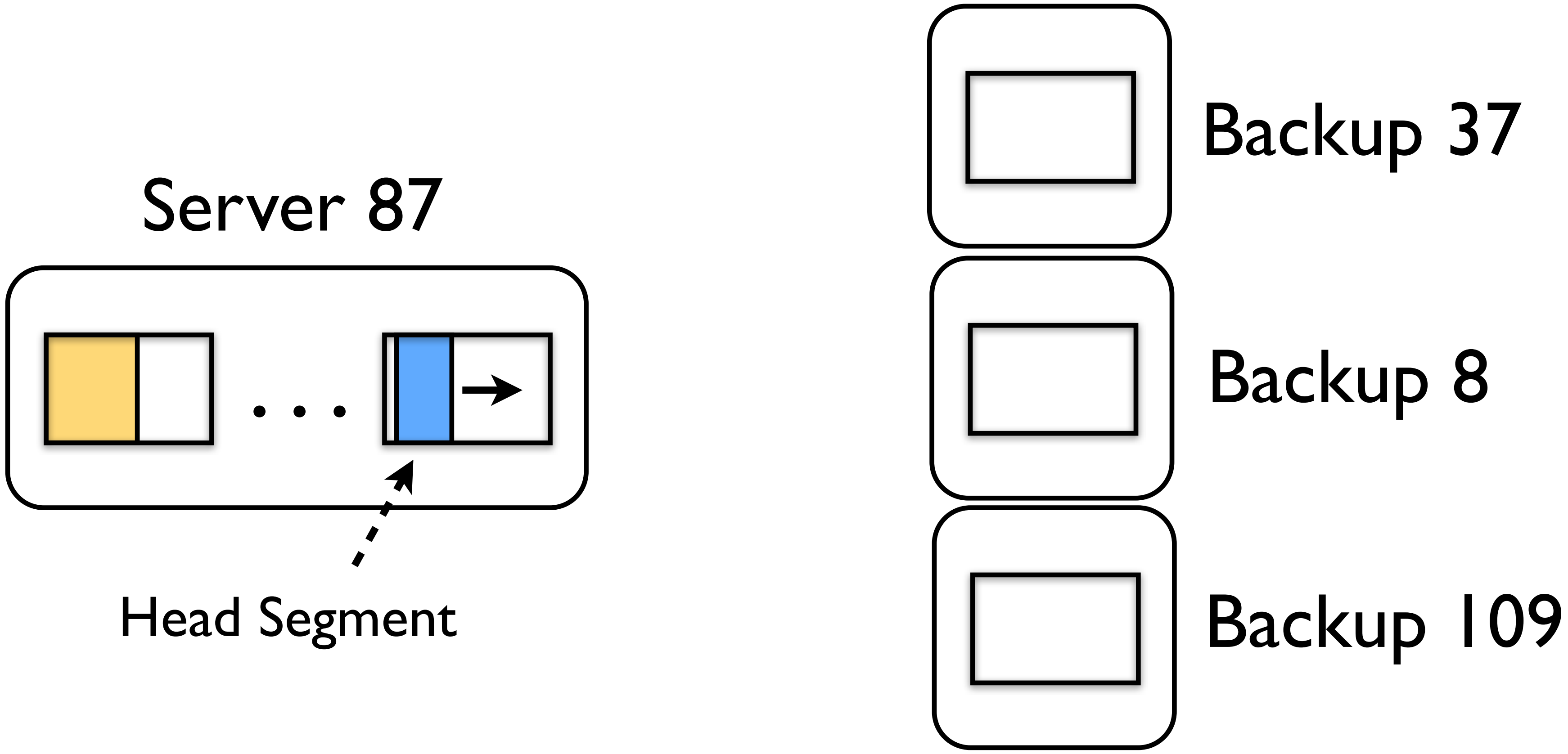
Transient Backup Failures

- When a backup fails, servers lose replicas
  - Must re-replicate segments stored on it
- What if the log head was on that backup?



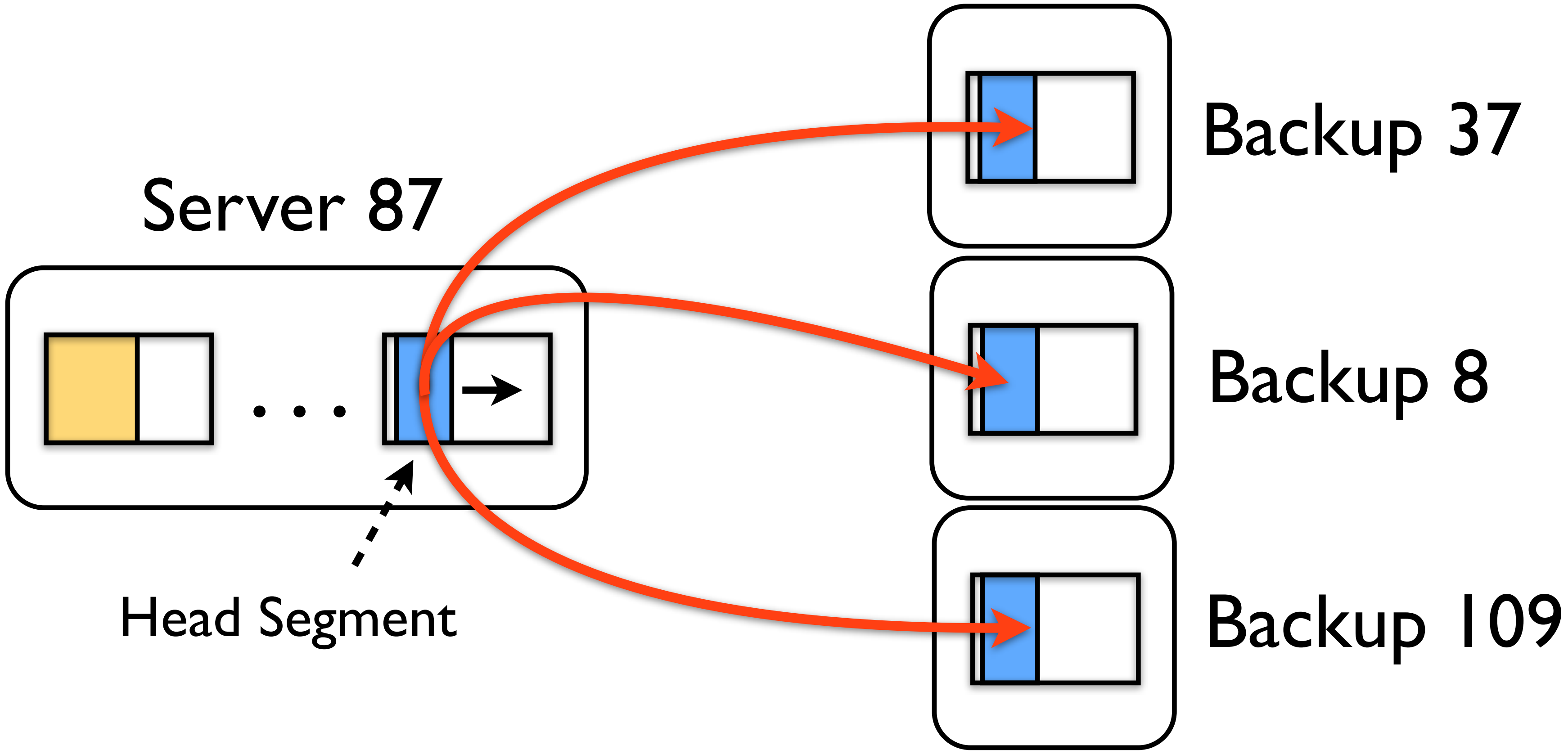
Transient Backup Failures

- When a backup fails, servers lose replicas
  - Must re-replicate segments stored on it
- What if the log head was on that backup?



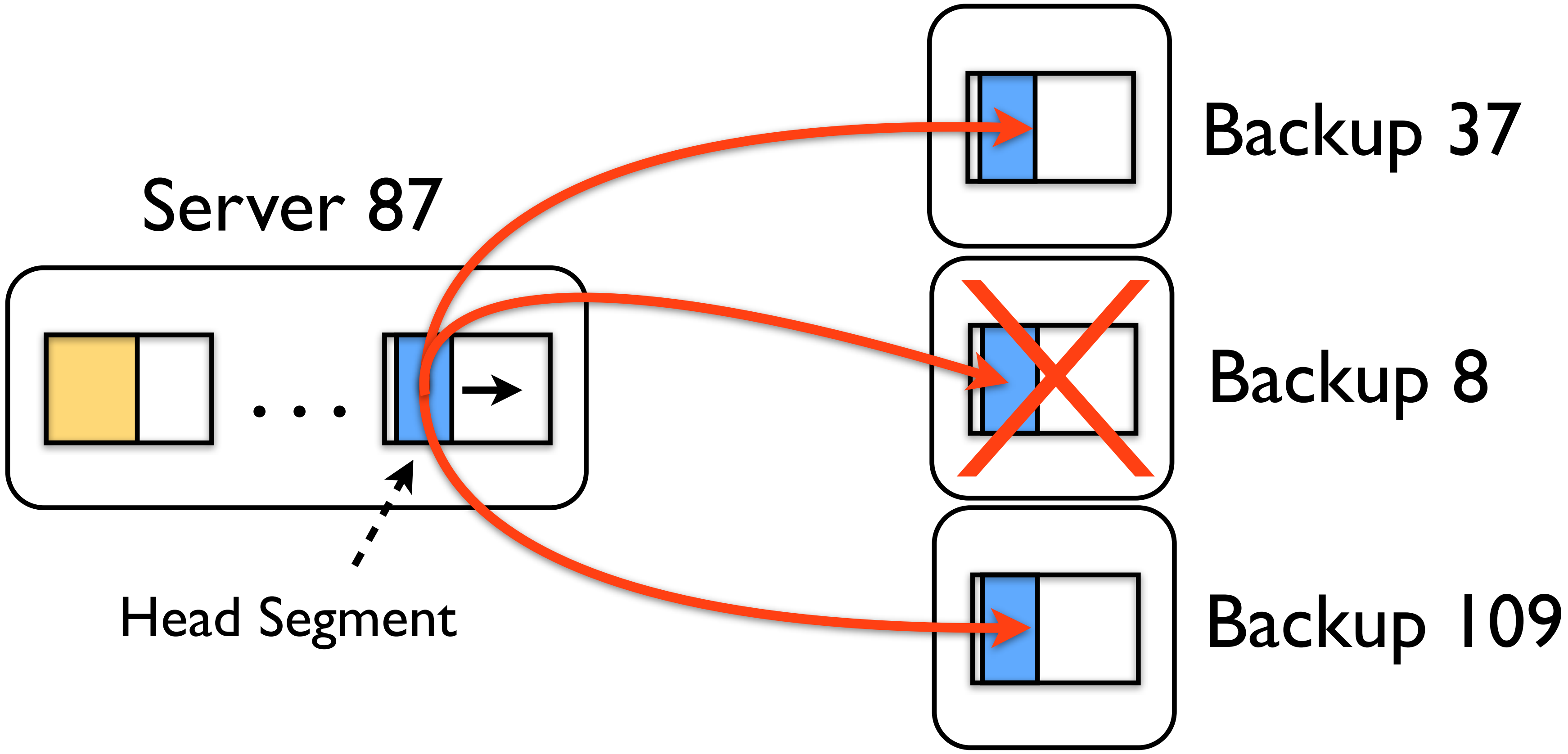
Transient Backup Failures

- When a backup fails, servers lose replicas
  - Must re-replicate segments stored on it
- What if the log head was on that backup?



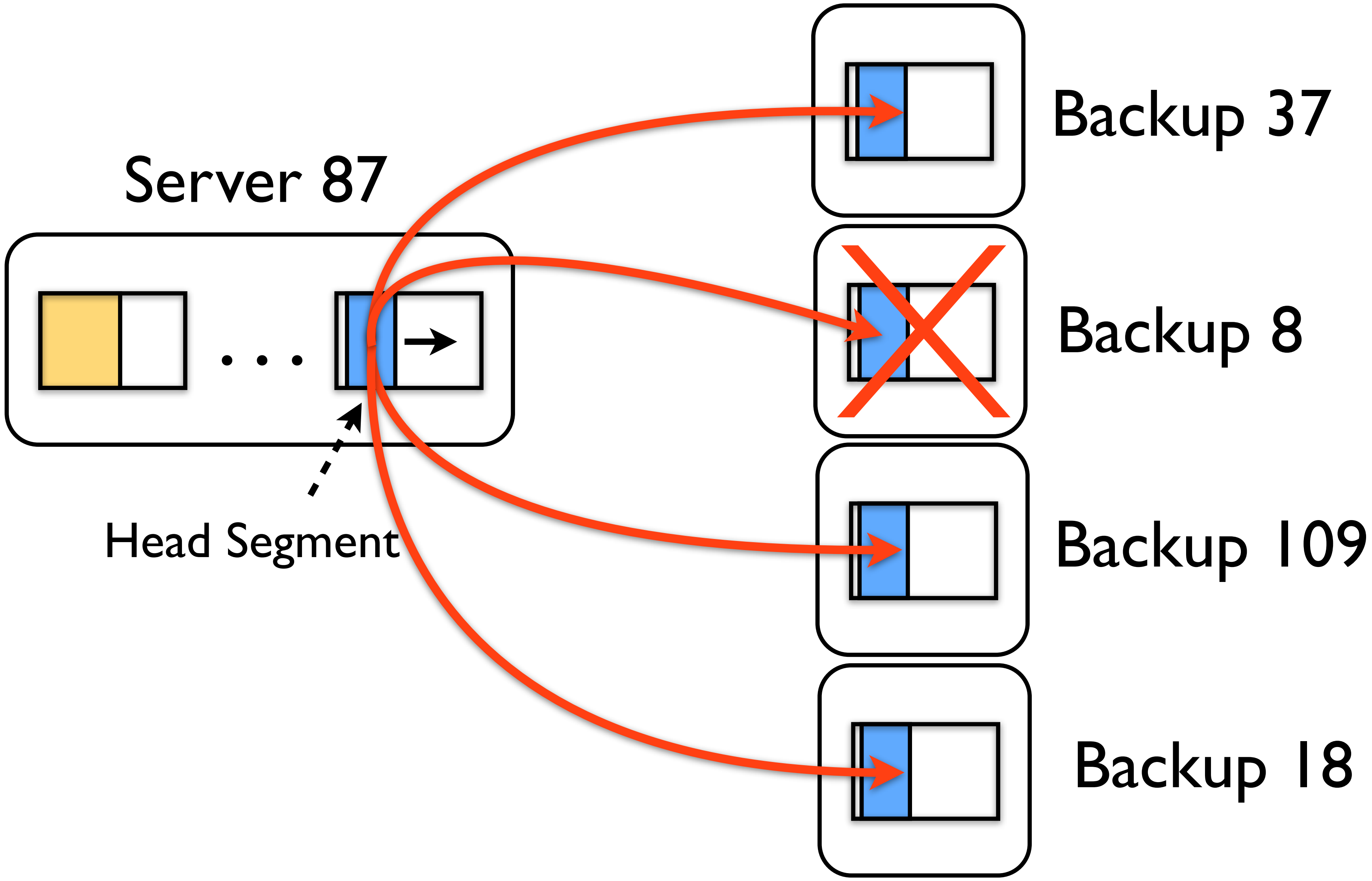
Transient Backup Failures

- When a backup fails, servers lose replicas
  - Must re-replicate segments stored on it
- What if the log head was on that backup?



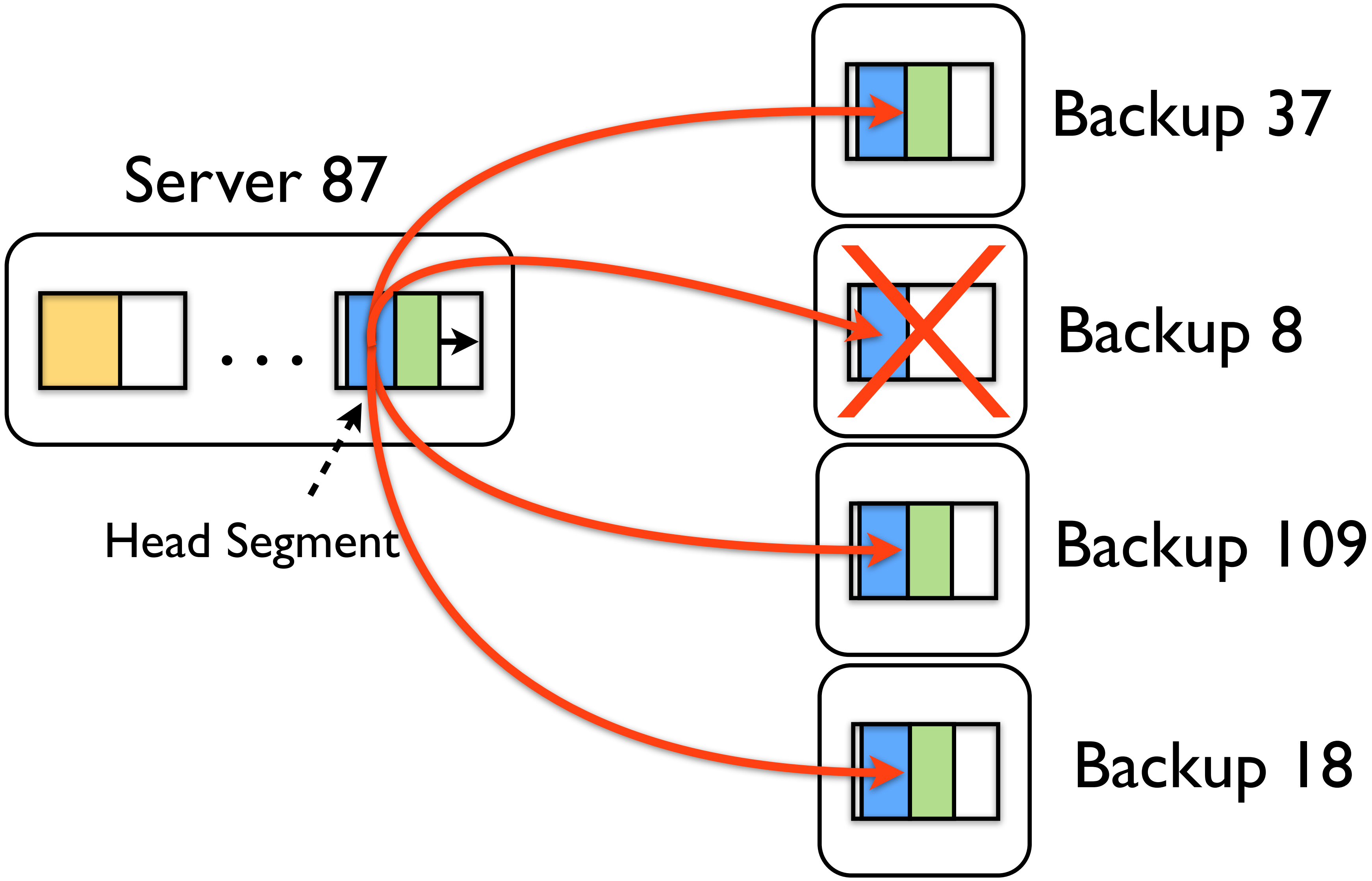
Transient Backup Failures

- When a backup fails, servers lose replicas
  - Must re-replicate segments stored on it
- What if the log head was on that backup?



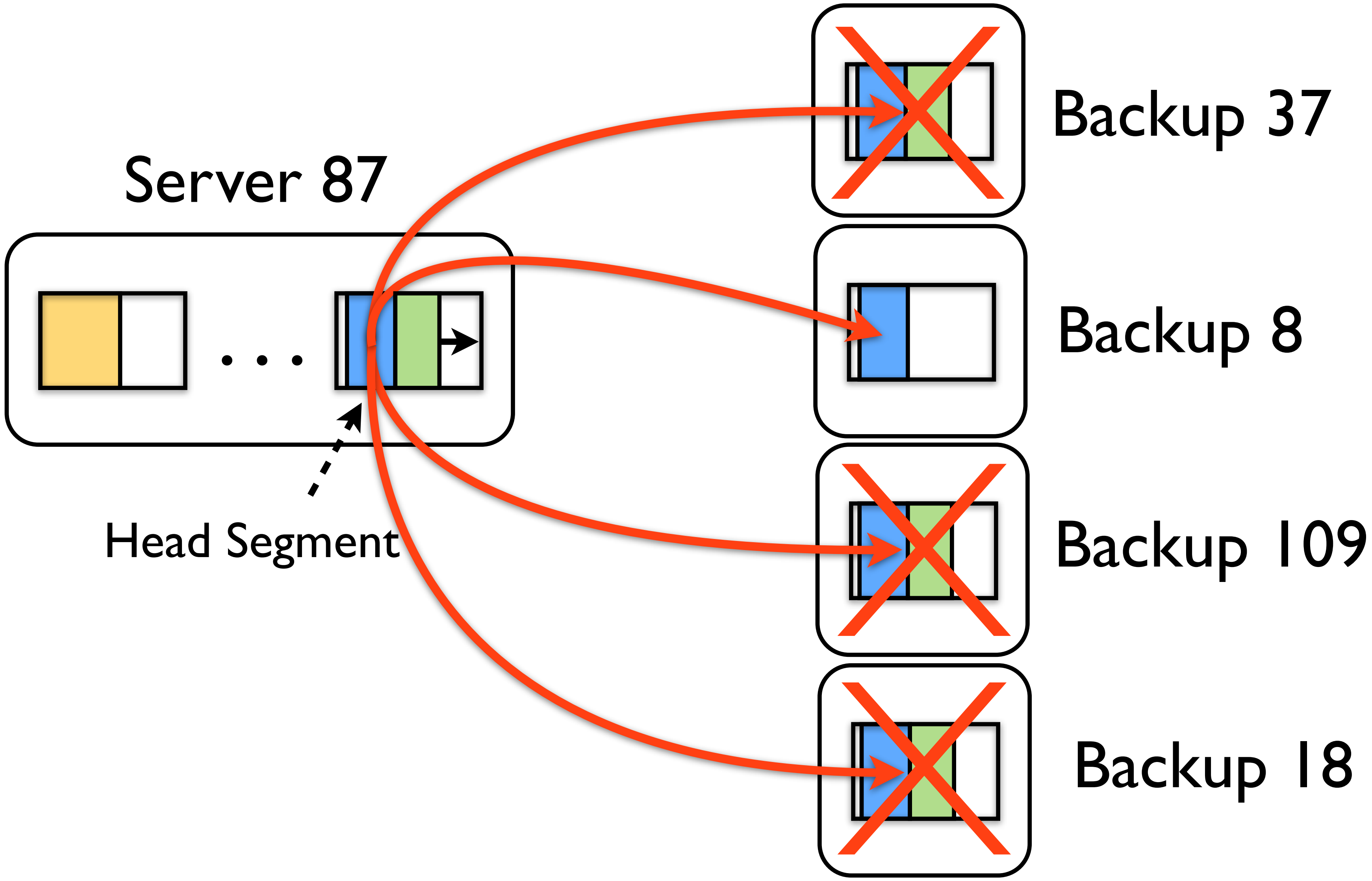
Transient Backup Failures

- When a backup fails, servers lose replicas
  - Must re-replicate segments stored on it
- What if the log head was on that backup?



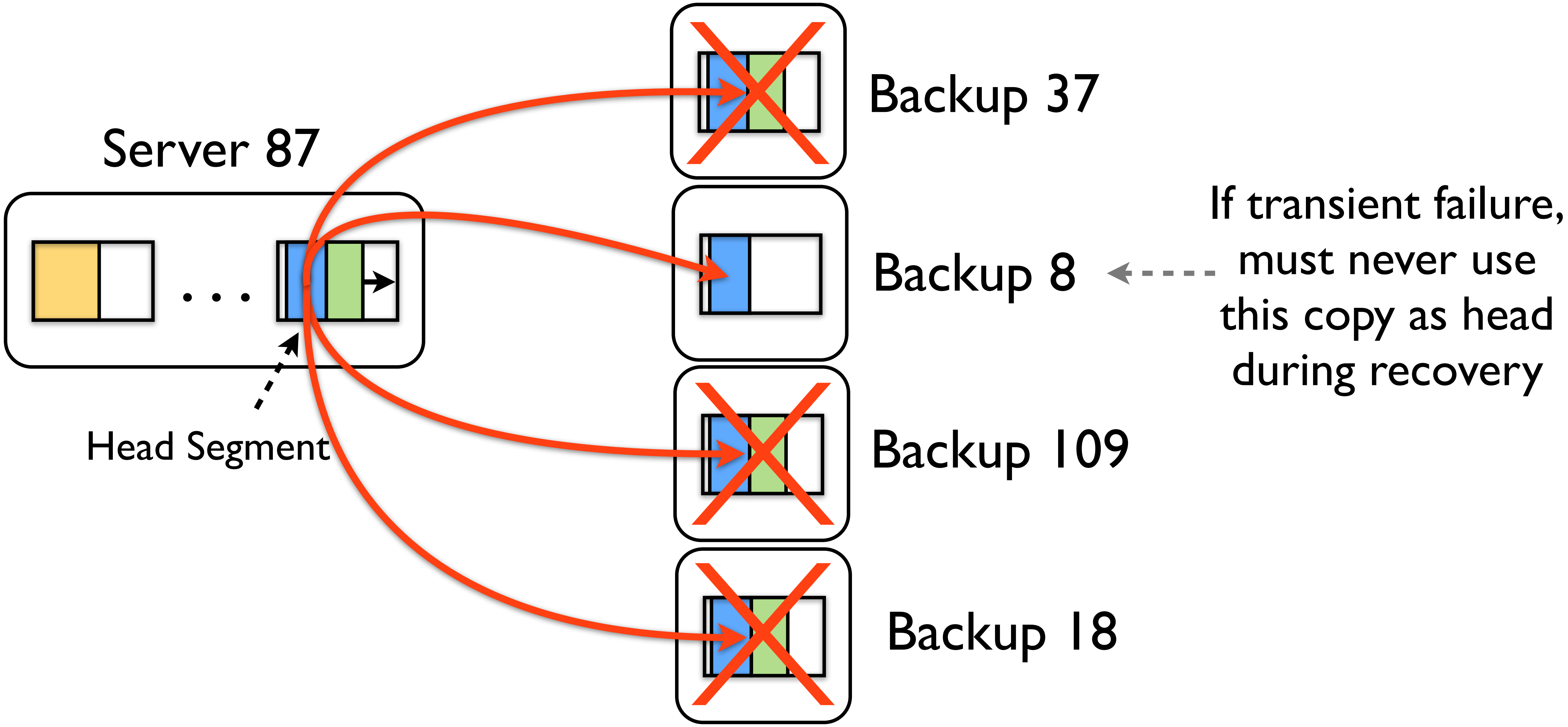
Transient Backup Failures

- When a backup fails, servers lose replicas
  - Must re-replicate segments stored on it
- What if the log head was on that backup?



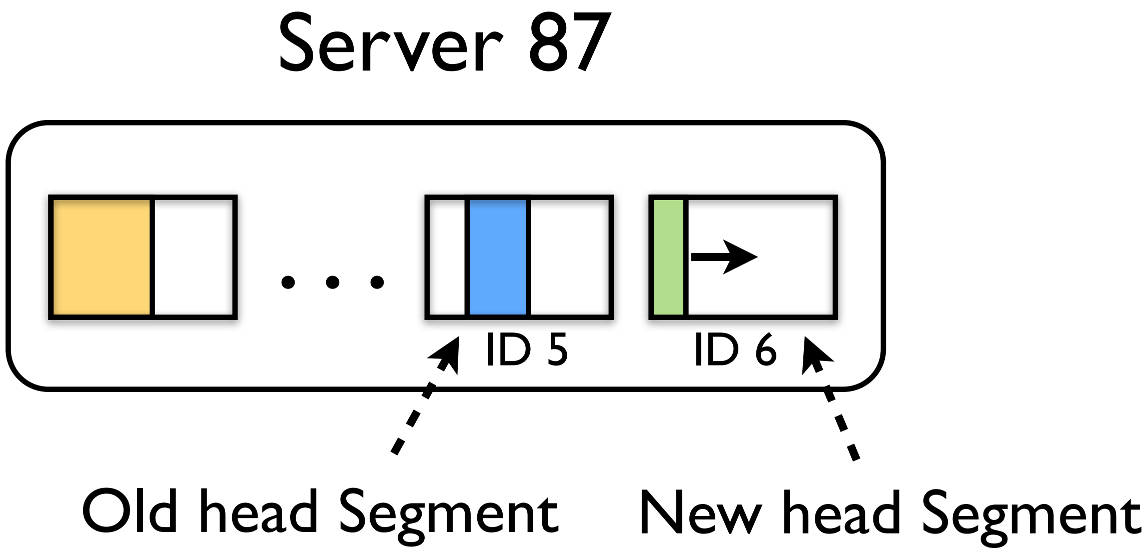
Transient Backup Failures

- When a backup fails, servers lose replicas
  - Must re-replicate segments stored on it
- What if the log head was on that backup?


Server 87
. . .
Head Segment
Backup 37
Backup 8
Backup 109
Backup 18
If transient failure, must never use this copy as head during recovery

Min Open Segment ID

- Solution: If backup of head segment fails
  - Immediately allocate new head segment
  - Close previous head
  - Tell coordinator to never use open segments of a smaller ID than new head
  - Re-replicate old, closed head segment



Coordinator

Server	Min ID
87	6
• • •	• • •

Future Work

- Cleaner measurement, optimization
    - Very little tweaking done so far
- Comparison with other schemes and systems
    - Different backup and in-memory structure
    - Efficiency compared to other allocators
- Cluster-wide memory management
    - Migrating objects to evenly distribute load

Conclusions

- Log-structure + cleaning used in memory and on disk
- Parallel cleaning looks promising
- Logs scattered across backup disks

Questions?