



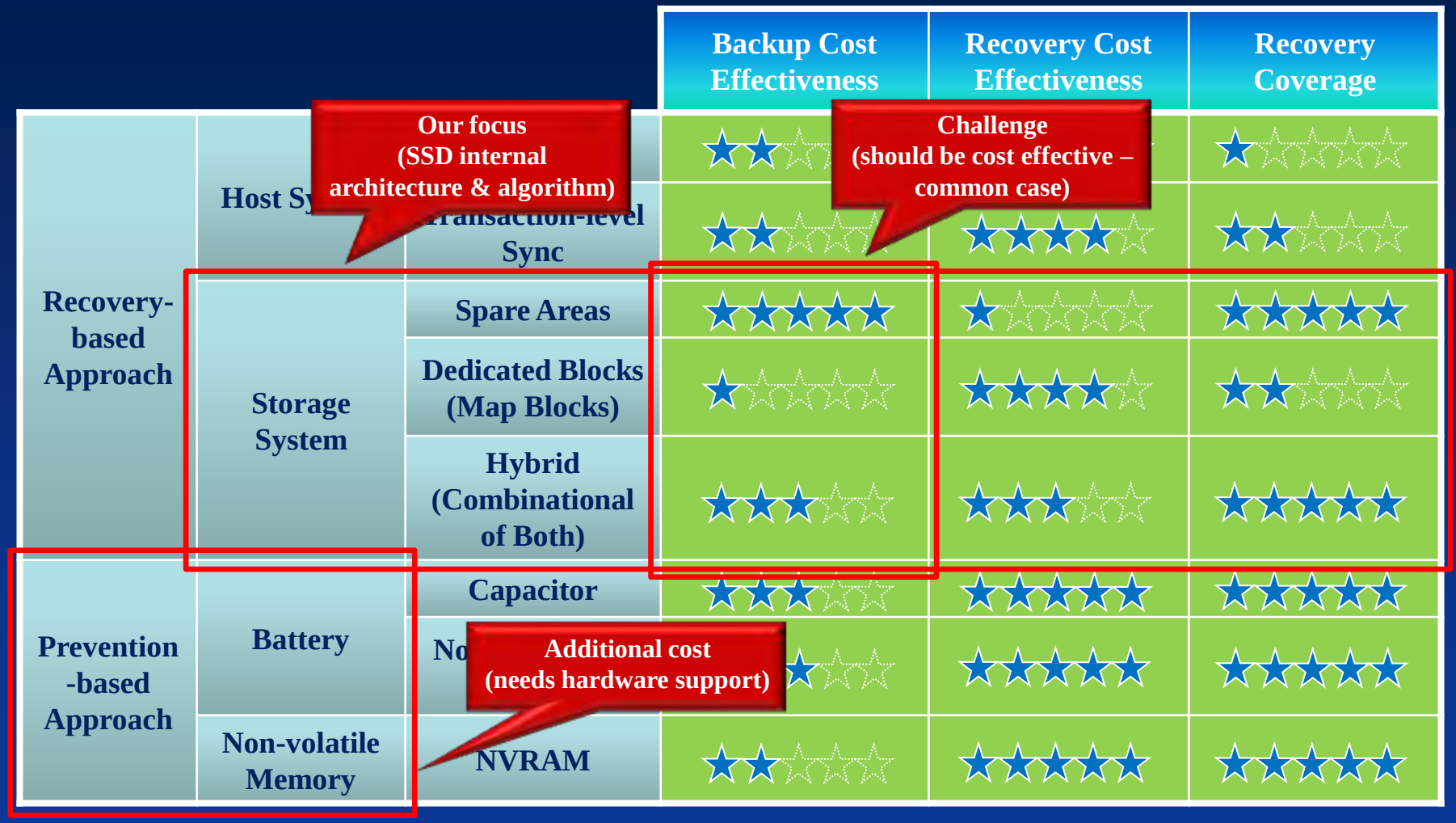
# Power Loss Recovery for NAND Flash Memories

Sanghyuk Jung & Yong Ho Song
Hanyang University, Korea

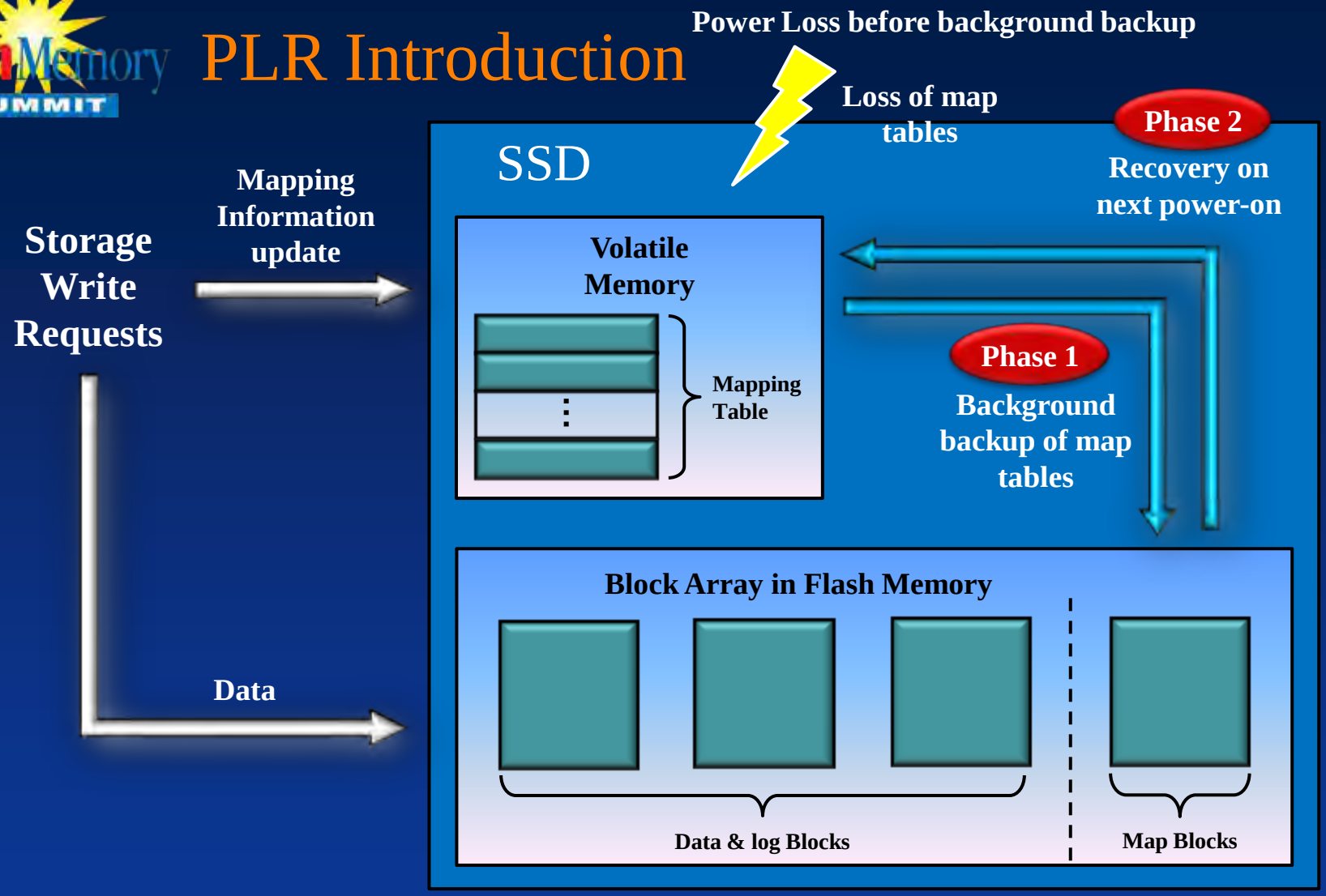

# Agenda

- Data and Meta-Data Synchronization
- *Power Loss Recovery (PLR)* Introduction
- Problem State
- Design Considerations
- *PLR* Schemes using Per-Page Spare Areas with No Synchronization Overhead
- Summary

# Data and Meta-Data Synchronization

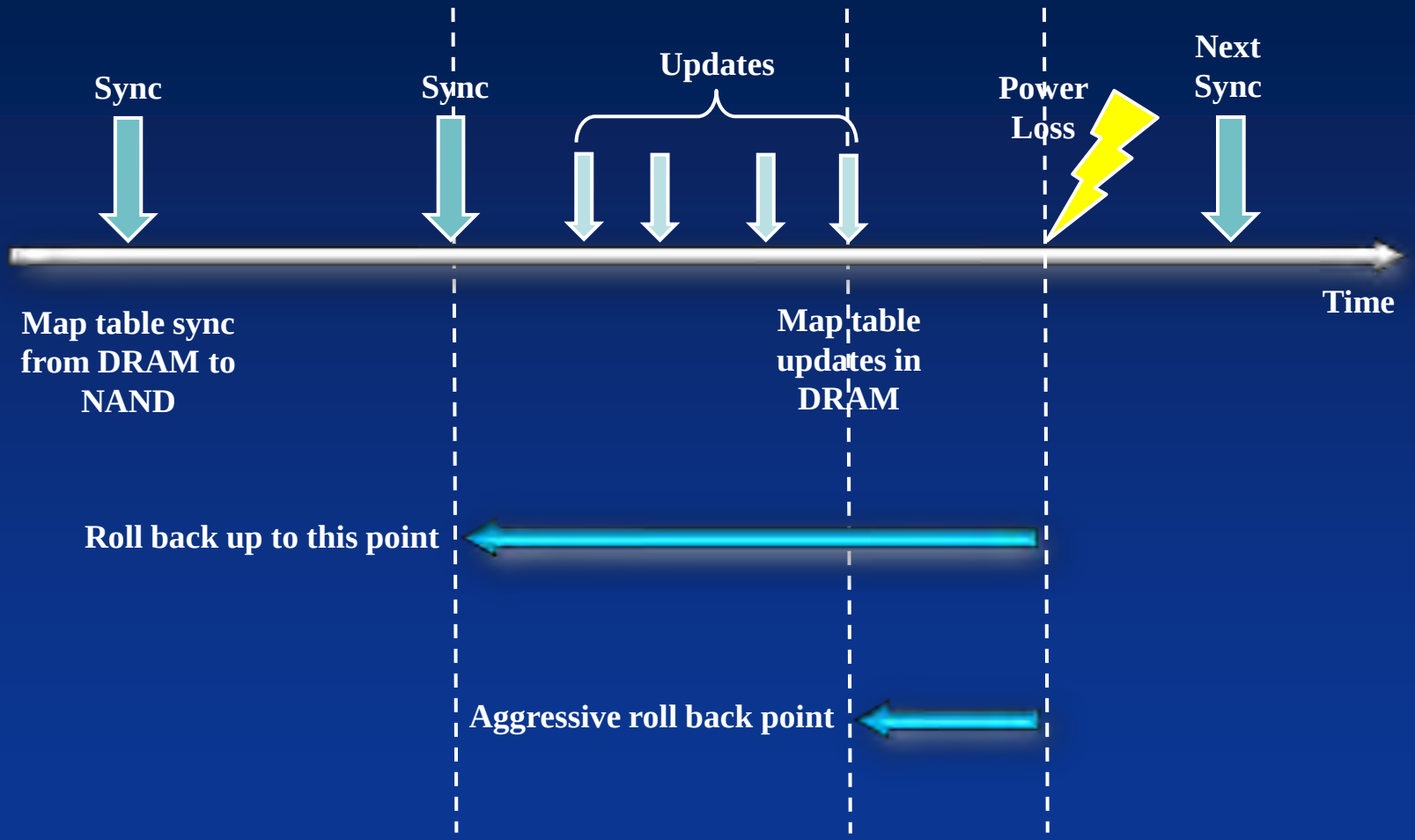


# PLR Introduction

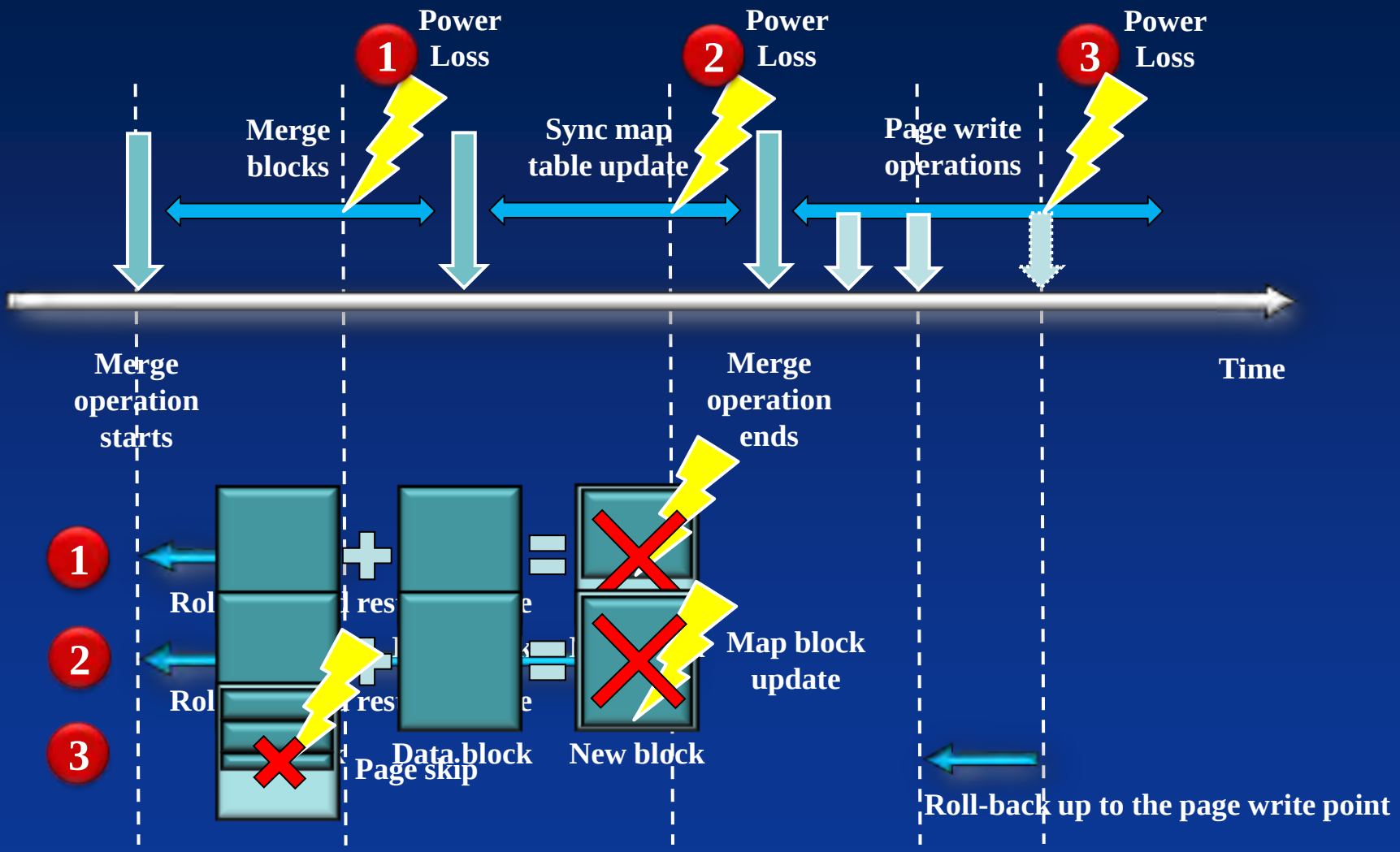



**How to make mapping tables as consistent as possible with backups in NAND Flash?**

# Power Loss Recovery



# Hybrid Mapping: A Case Study




# Problem Statement

- Run-Time Backup Cost Growth
  - Map table backup overhead: approximately 5% of write operation in normal case
- Merge Operations causes Changes in Map Table
  - Dependent upon FTL algorithms and workloads
- A Cost Effective Solution: Mapping Information Backup using Per-Page Spare Areas

**Power loss is a RARE event**

**Make common case fast!!**

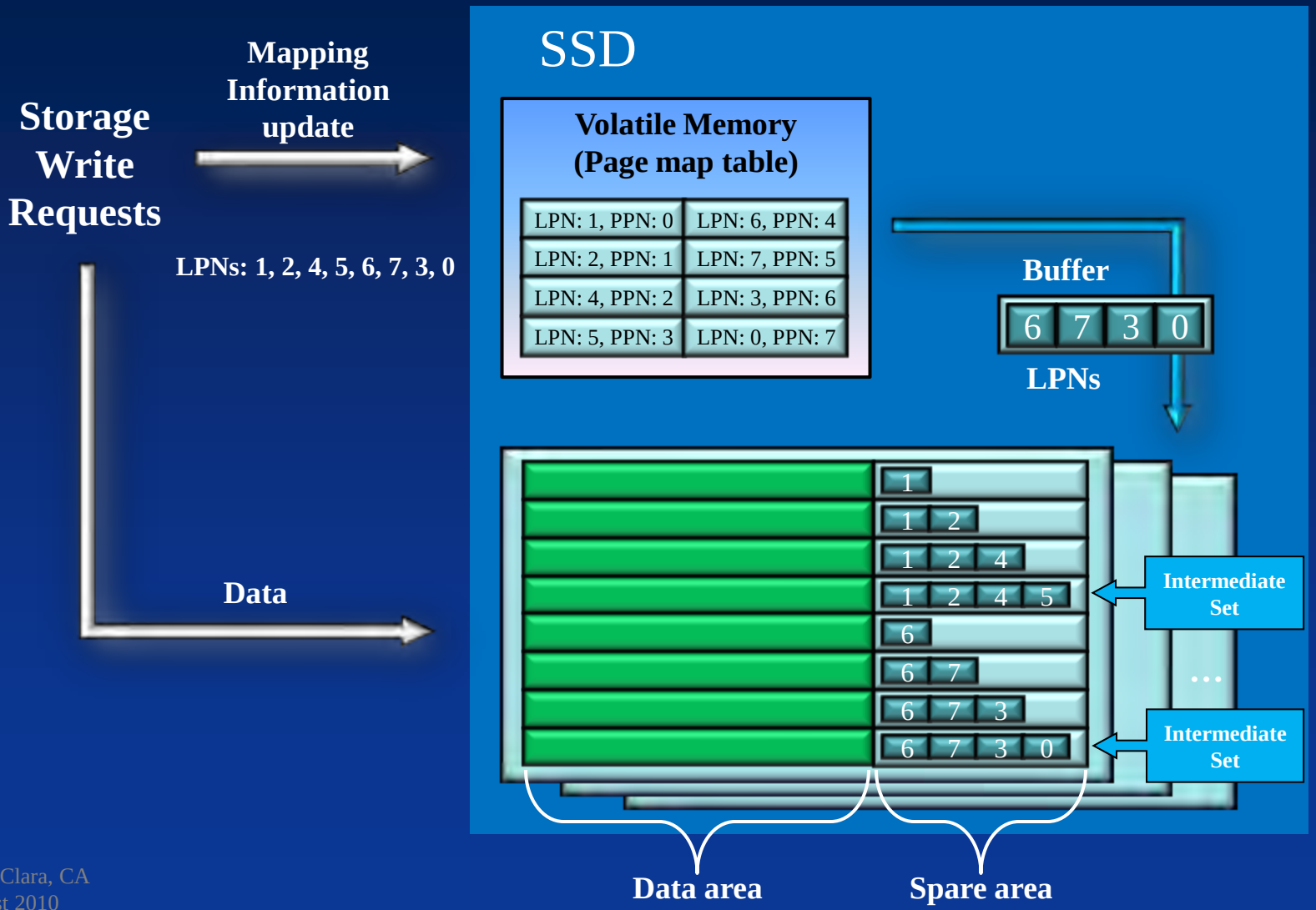

# Design Considerations

- *Run-Time Backup Cost*
  - Map table backup time from DRAM to NAND at run-time
- *Time for Recovery*
  - Map table reconstruction time from NAND to DRAM when a system re-boots after the power loss event
- *Recovery Coverage*
  - Restore data writes as complete as possible

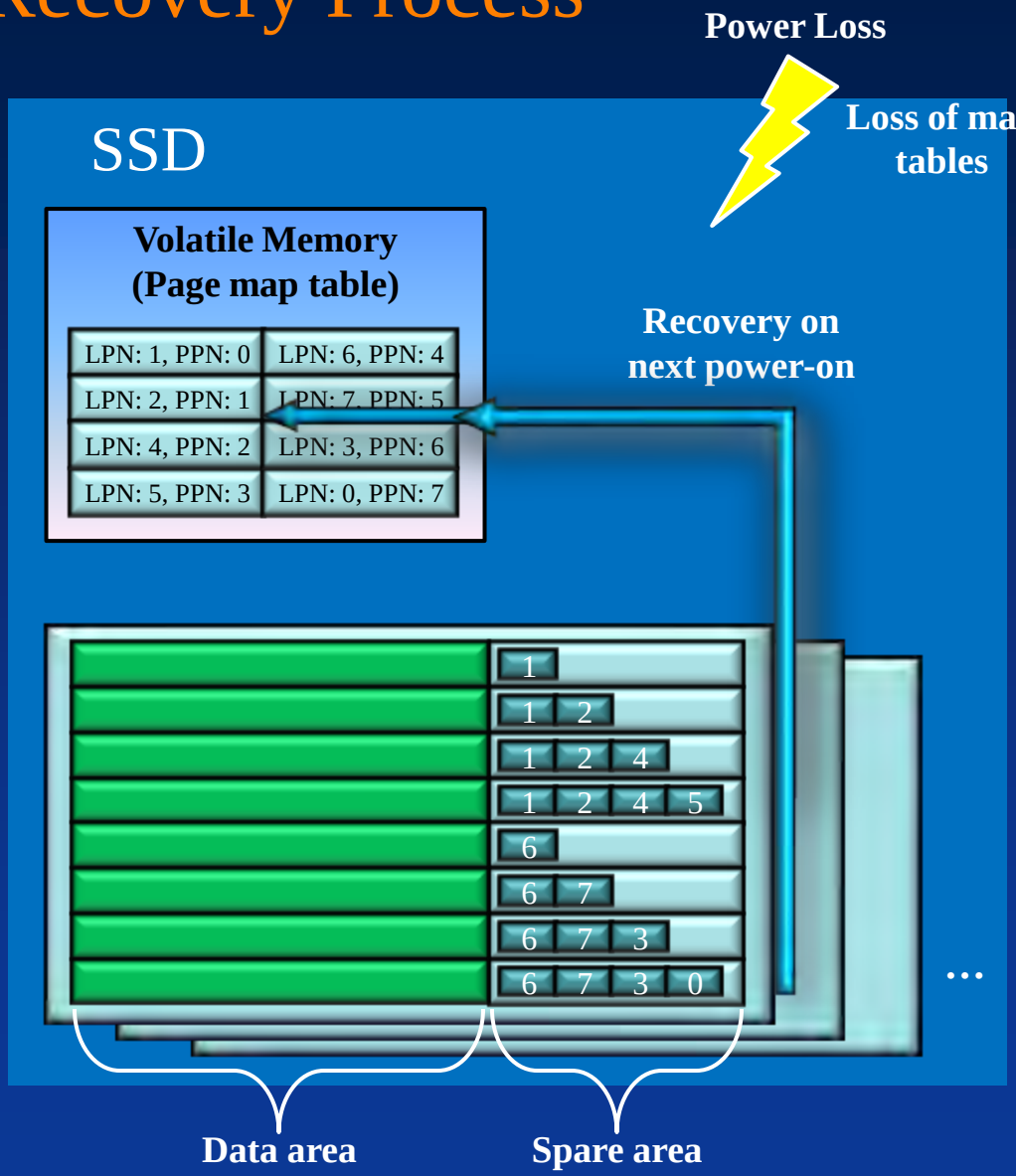
# PLR Schemes using Per-Page Spare Areas with No Synchronization Overhead

- Distribute Mapping Information over Spare Areas
- Different Approach according to Mapping Schemes
  - FTLs with page mapping
    - *Accumulation-based PLR (A-PLR)*
  - FTLs with hybrid mapping
    - *Signature-based PLR (S-PLR)*
- Tradeoff between Backup Cost and Recovery Latency

# *A-PLR*: Backup Process

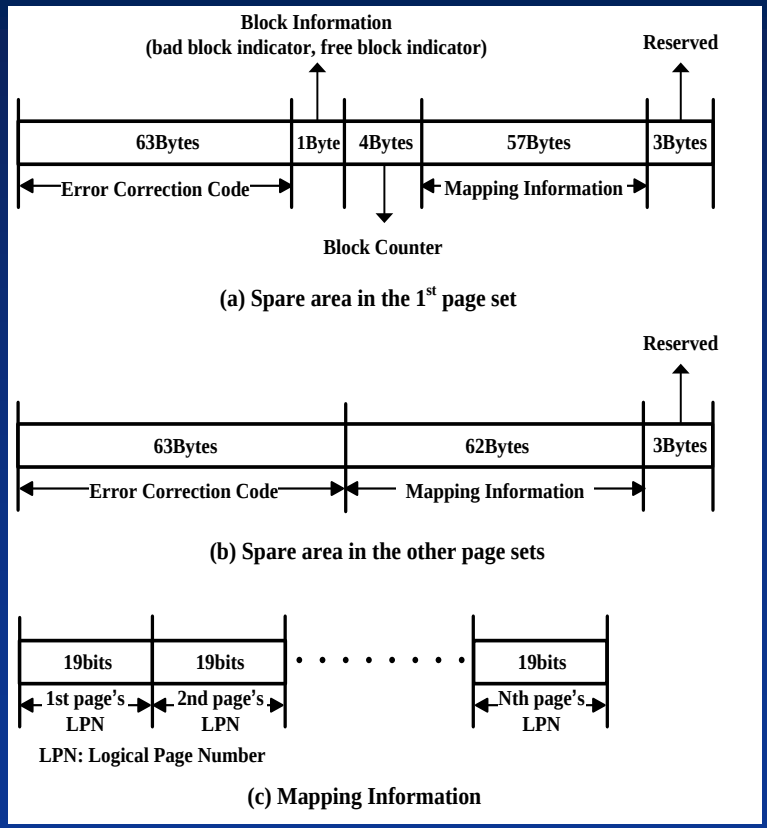


# *A-PLR*: Recovery Process

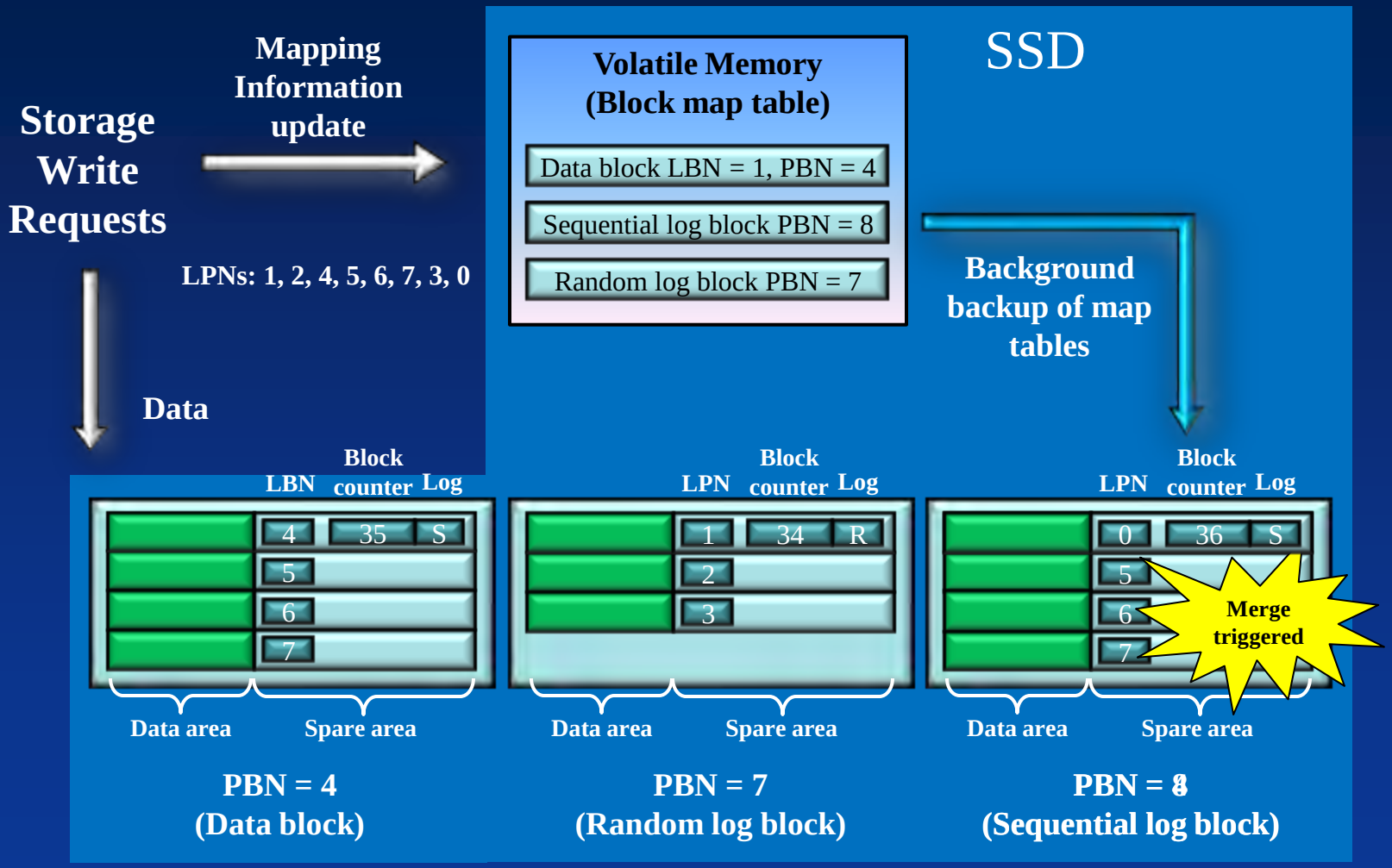


# *A-PLR*: Data Structure

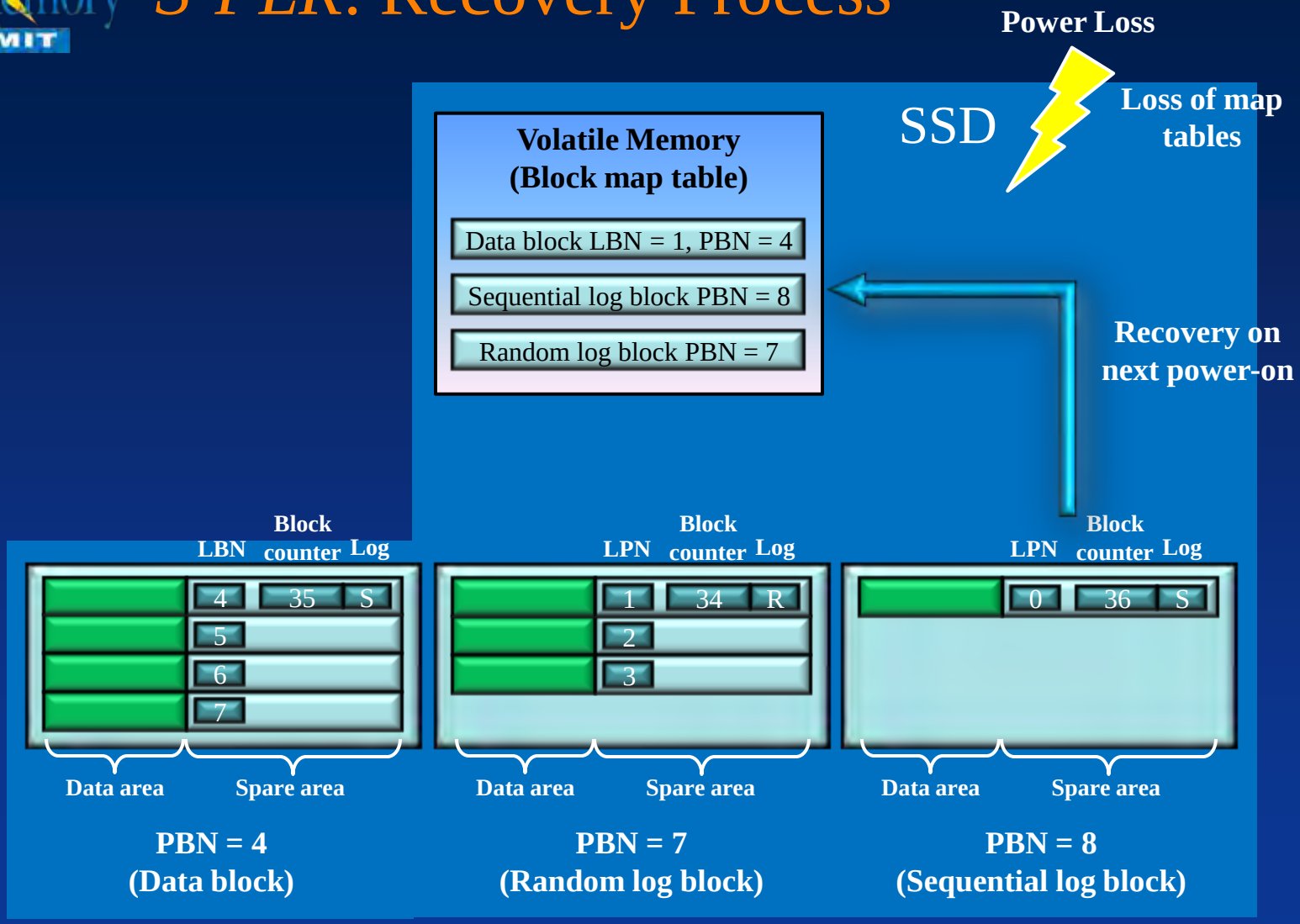
- Mapping Info Accumulation in Spare Areas
  - No additional overhead due to spare program in large block NAND flash
  - Power loss event before completing a new intermediate set
    - Searching with *b-tree algorithm* (log-scale search)
- Handling Duplicated LPNs
  - Storing block information and block counter in the first intermediate set
  - Power loss event during *GC and merge operations*
    - The same LPNs may be found in multiple blocks
    - Using the block counter as a recency indicator



# *S-PLR*: Backup Process

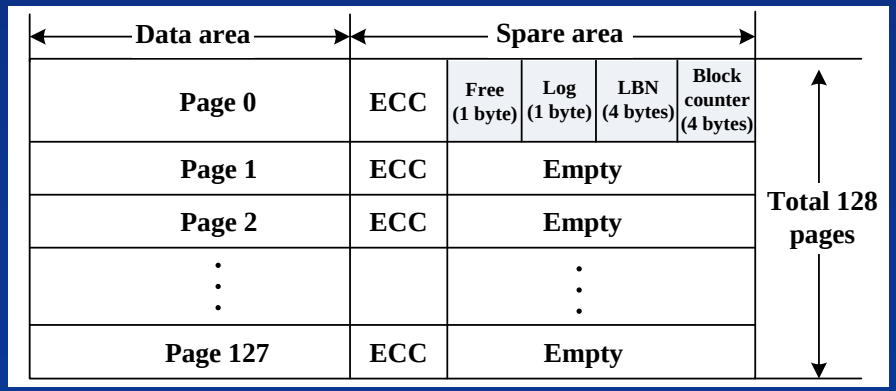


# *S-PLR*: Recovery Process



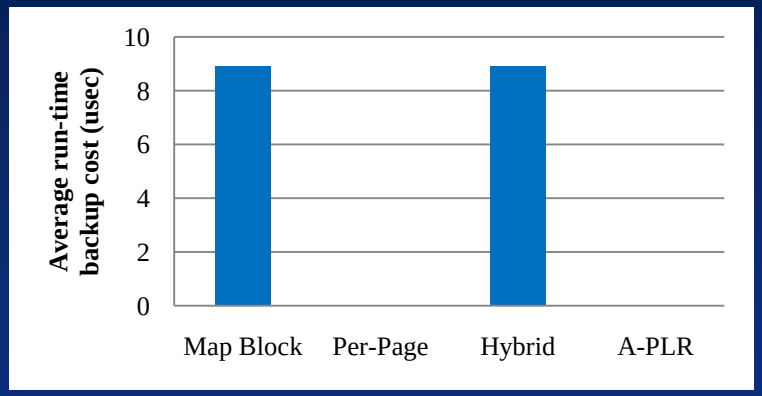
# *S-PLR*: Data Structure

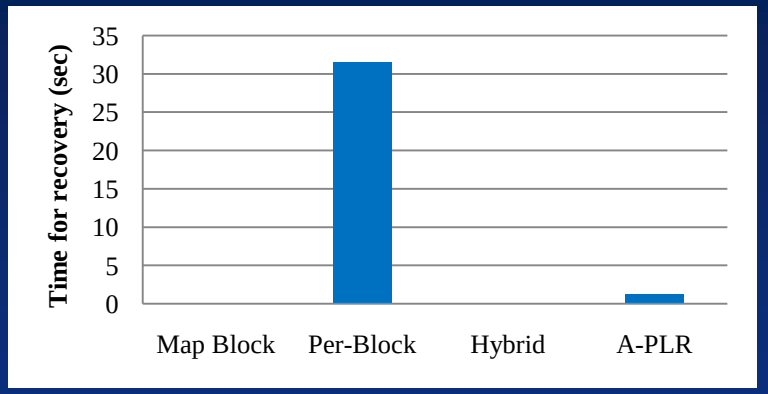
- Signature
  - *Free Flag*: whether the block is free or not
  - *Log Flag*: *S (Sequential log), R (Random log), or D (Data block)*
  - *LBN*: *Logical Block Number*
  - *Block Counter*: recency indicator
- Recovery Mechanism
  - Step 1: Reading the first pages of all blocks in NANDs
  - Step 2: Sorting blocks on *Log Flags* and then *Block Counters*
  - Step 3: Validating each block based on block use scenario



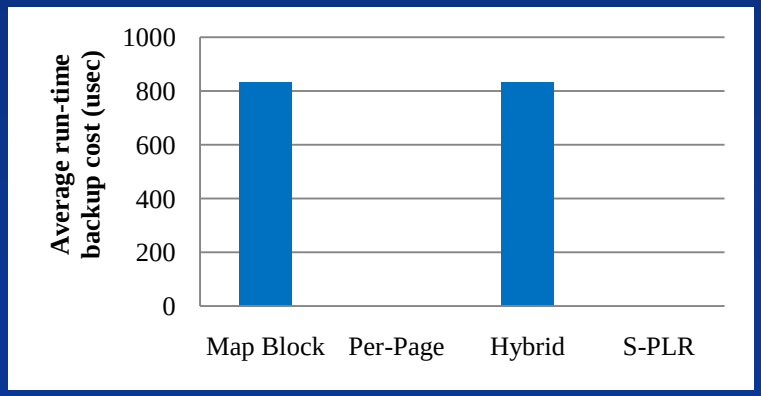
# Performance

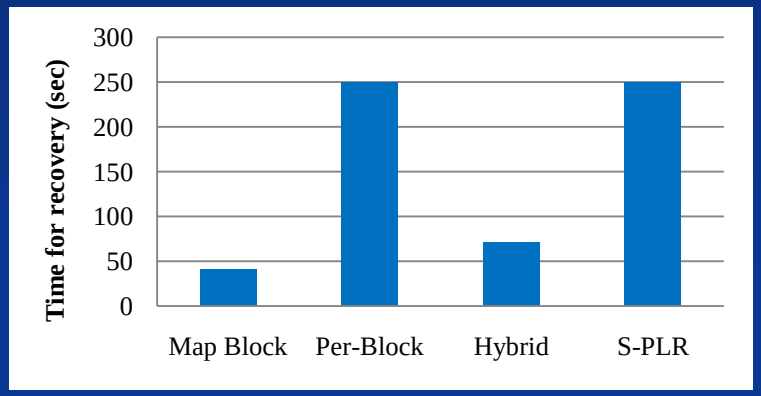
- *Page-Level Mapping (A-PLR)*




- *Block-Level Mapping (S-PLR)*




# Summary

- PLR Schemes using Spare Areas
  - Efficient use of spare area
    - Mapping info accumulation *(A-PLR)* / signature *(S-PLR)* in spare areas in order to eliminate map blocks
  - Removal of space overhead and time latency
- Small Additional Delays for Multi-Chip SSDs
  - Recovery parallelism among channels and ways
- Future Work
  - PLR for FTLs with *data compression*


# Thank you