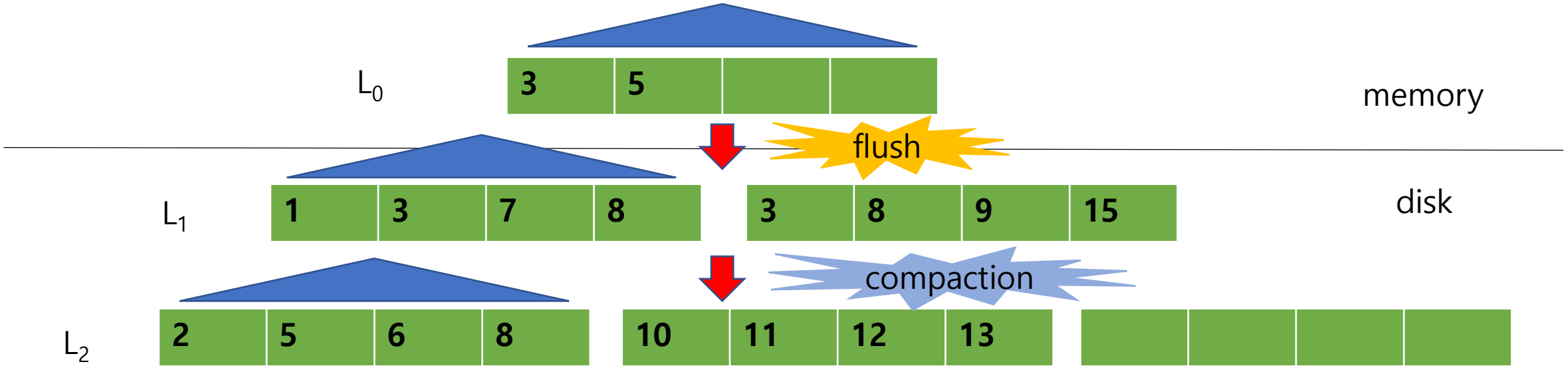
On Log-Structured Merge for Solid-State Drives

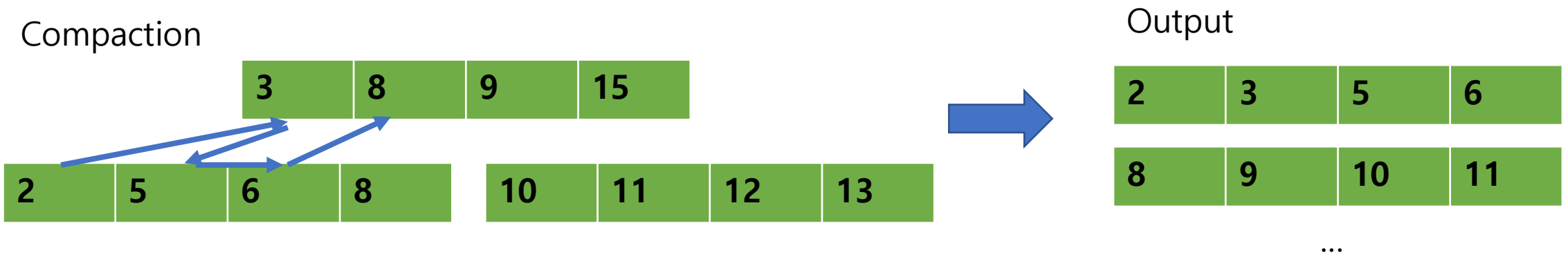
ICDE ‘17, Risi Thonangi, Jun Yang
박사과정 최원기

Log Structure merge tree (LSM tree)



- LevelDB, HBase, Cassandra, Druid

LSM tree Compaction

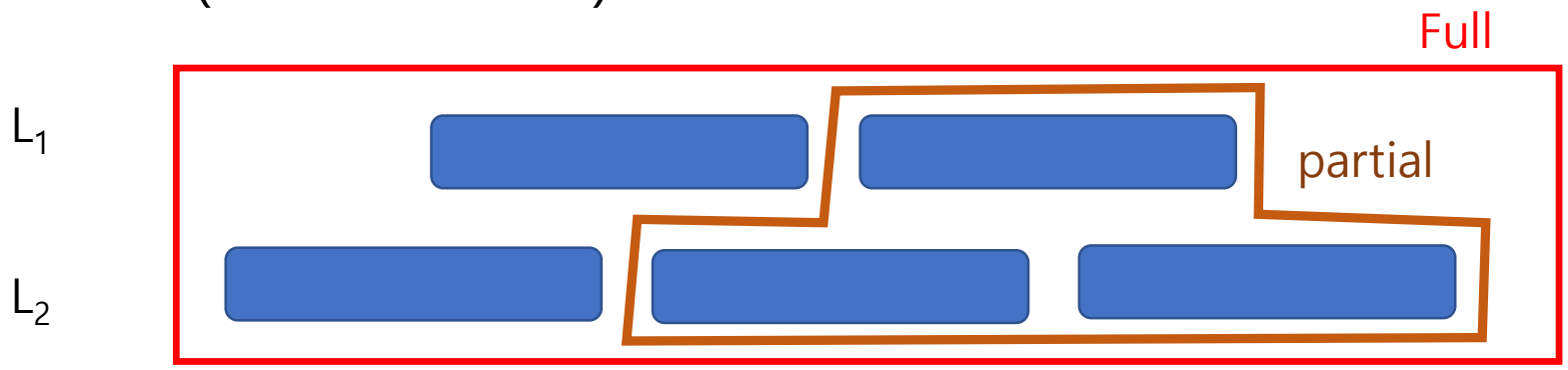


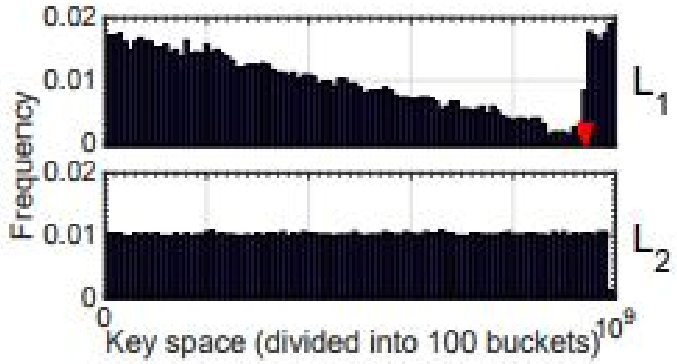
Advantage of Compaction

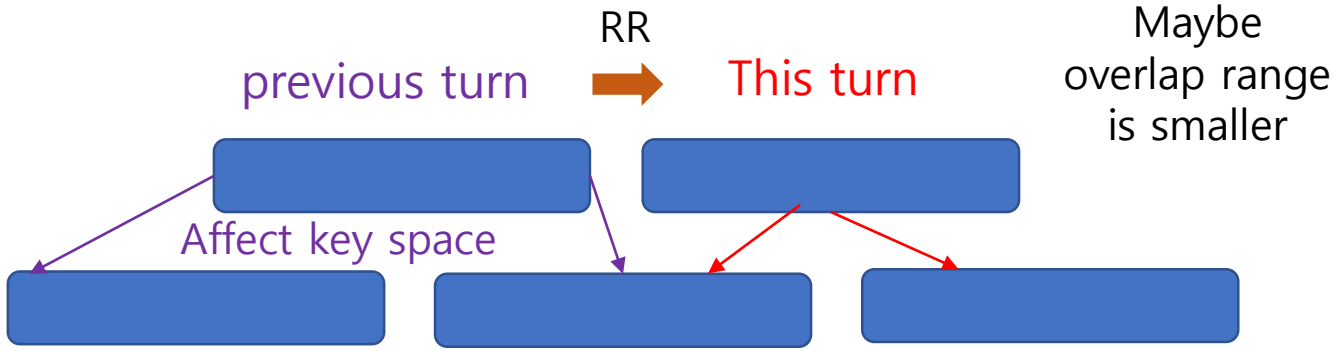
- Utilize Sequential I/O for HDD
- Eliminate in-place update for SSD

Compaction Policy

- Full vs Partial(Round-Robin)







However, in worst case?

Full vs Partial

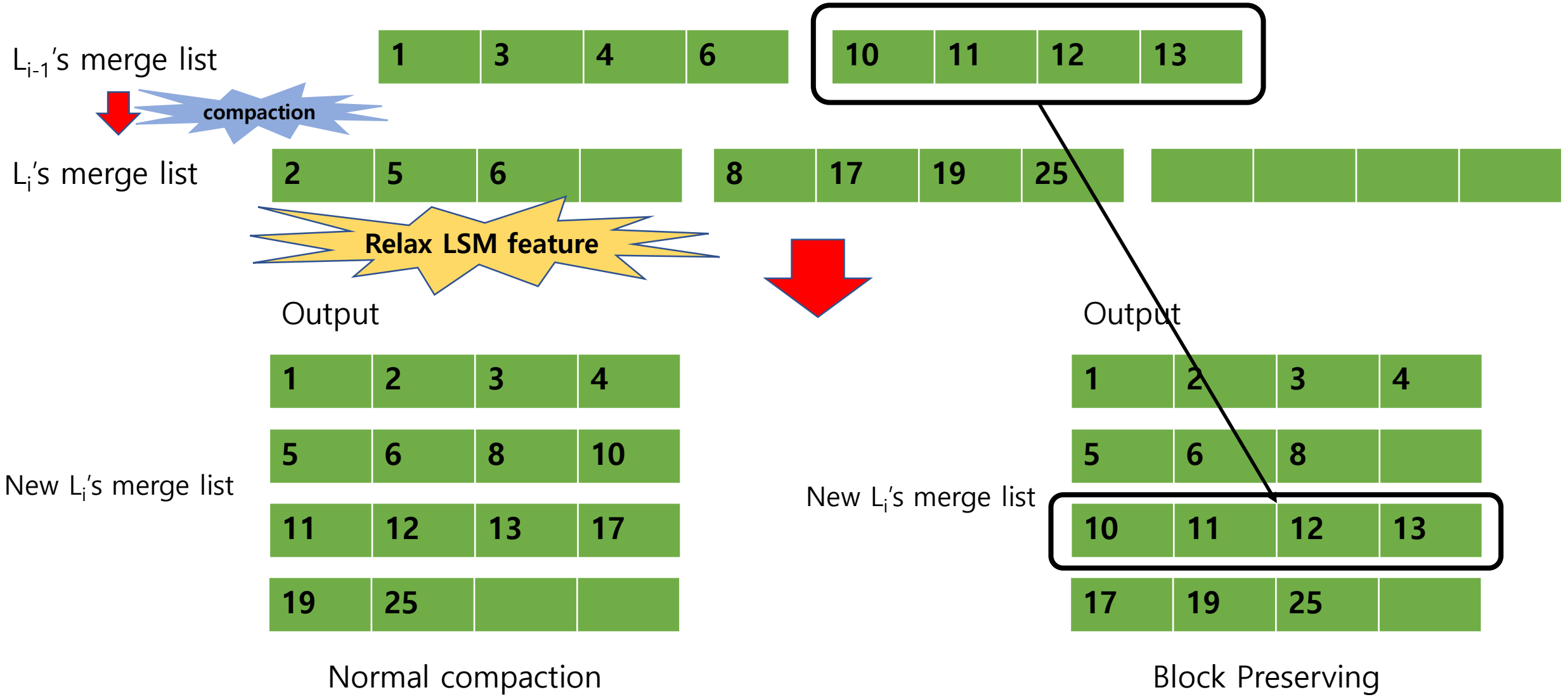
- Full Policy
  - Worst case cost of a merge into $L_i = K_i$
  - $\frac{1}{2}(\mathbb{\Gamma} + 1)$ per block merged

- Partial Policy (Round-Robin)
  - Worst case cost of a merge into $L_i = K_i * \min\{\partial B/\mathbb{\Gamma}, 1 - 1/\mathbb{\Gamma}\}$
  - $(\frac{1}{1-\partial} + o(1))\,\mathbb{\Gamma} + o(1)$ per block merged

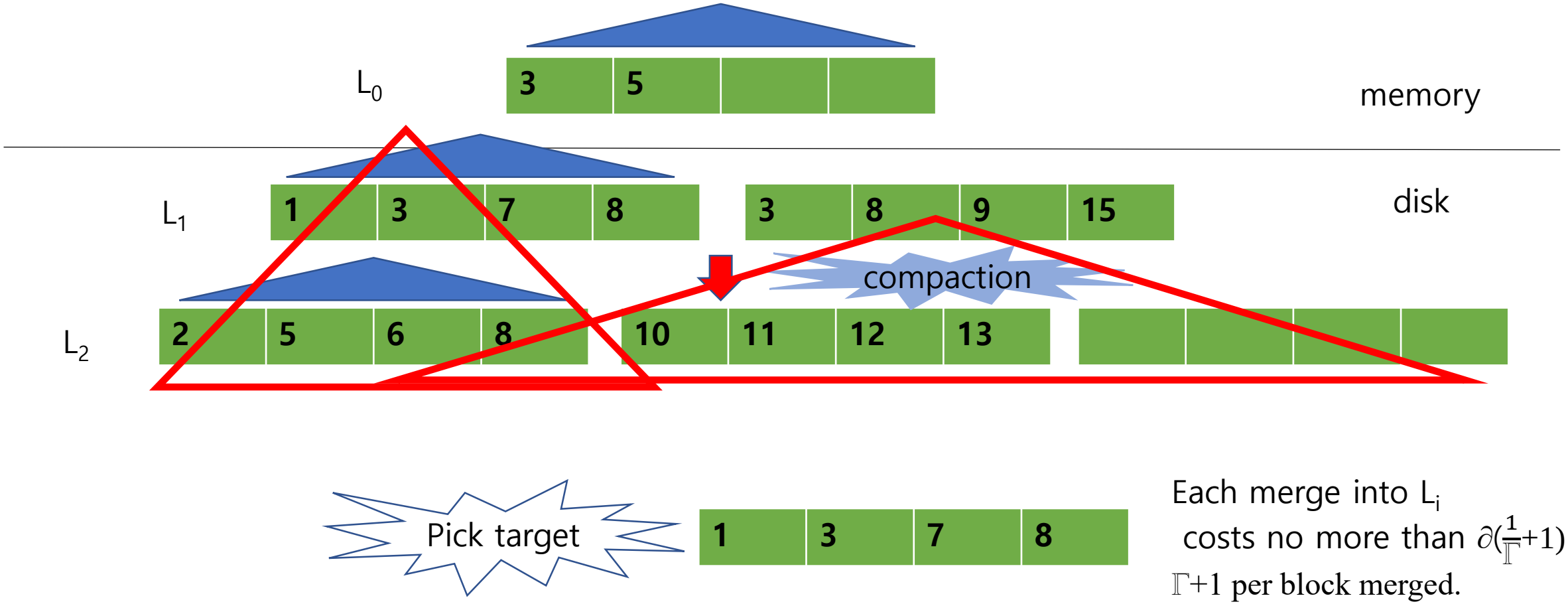
Paper’s proposal

- Implement Block Preserving Technique
  - To reduce writes
- Implement partial merge policy “ChooseBest”
  - With counter-intuitive optimization

Block Preserving Technique

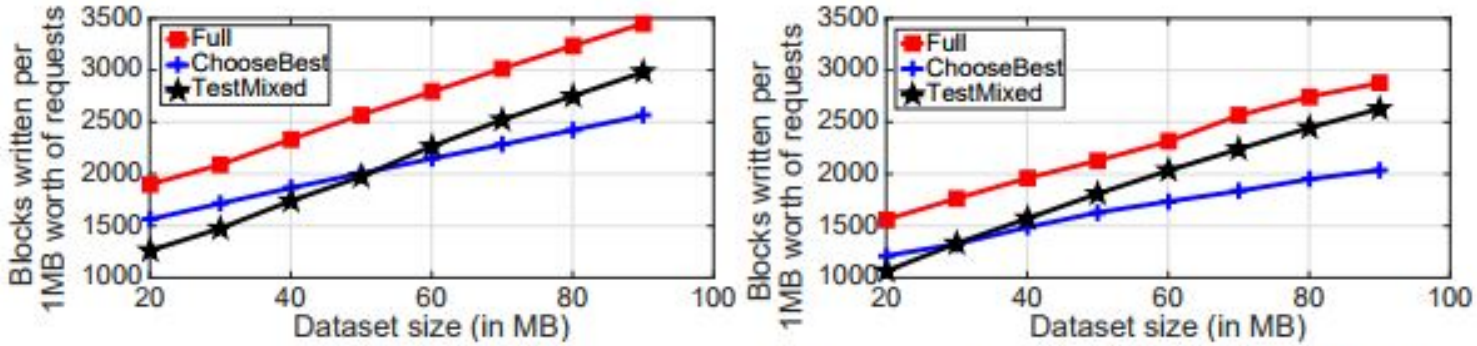


ChooseBest Policy



Each merge into L_i costs no more than $\partial(\frac{1}{\Gamma}+1)$ K_i, $\Gamma+1$ per block merged.

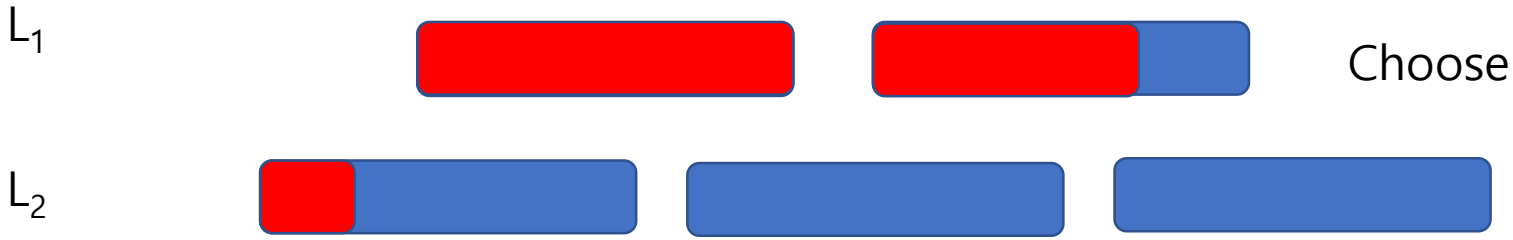
ChooseBest Policy



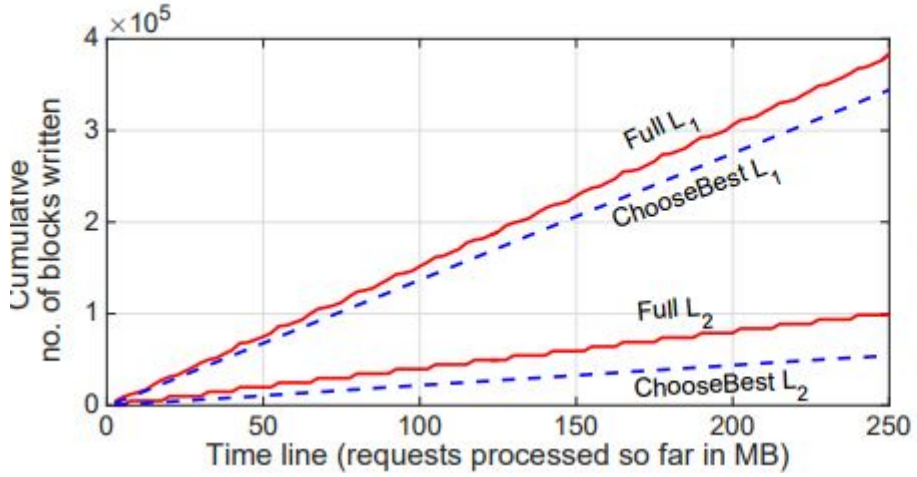
ChooseBest always have better performance.

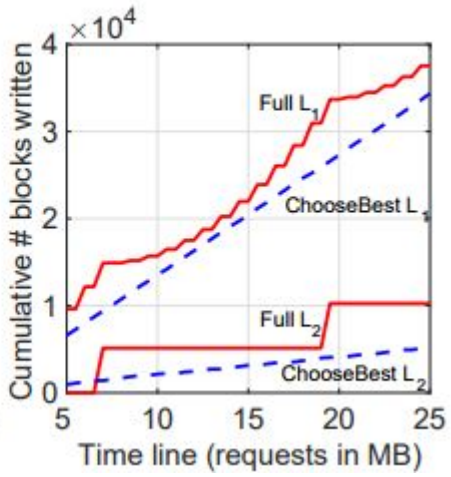
Mixed Policy (Full + ChooseBest)

Why?



Choose Best always full

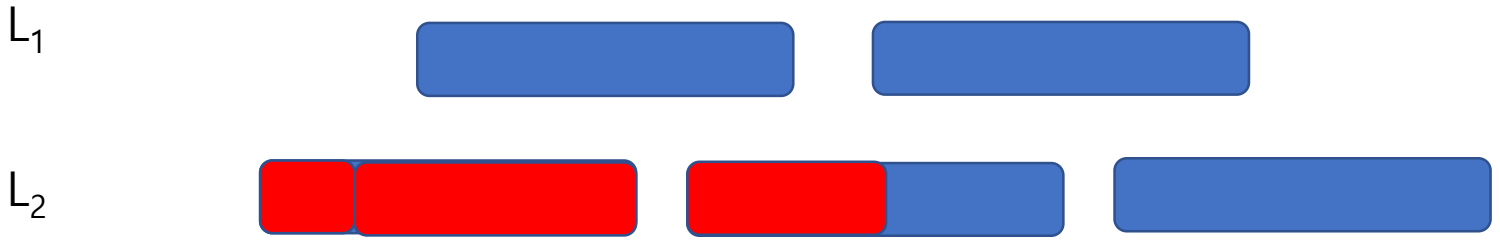


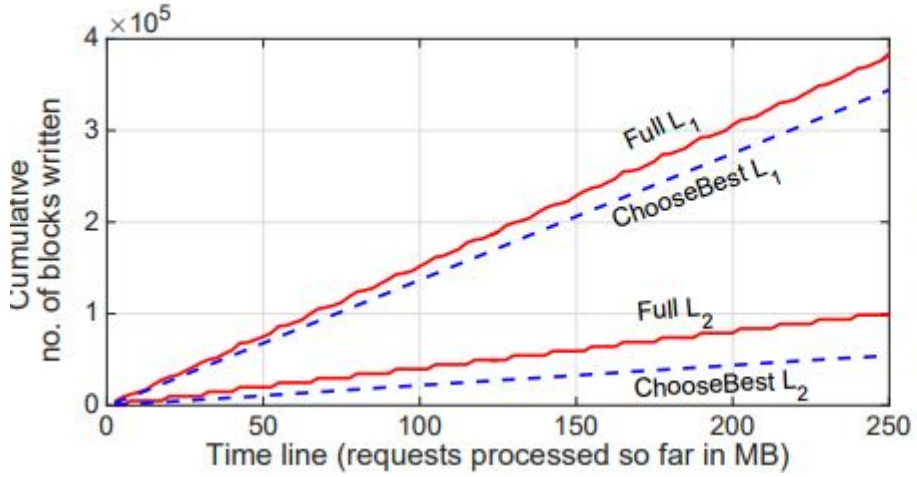


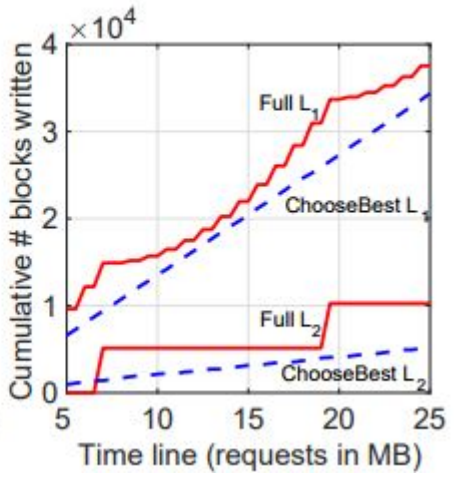
Mixed Policy (Full + ChooseBest)

Why?

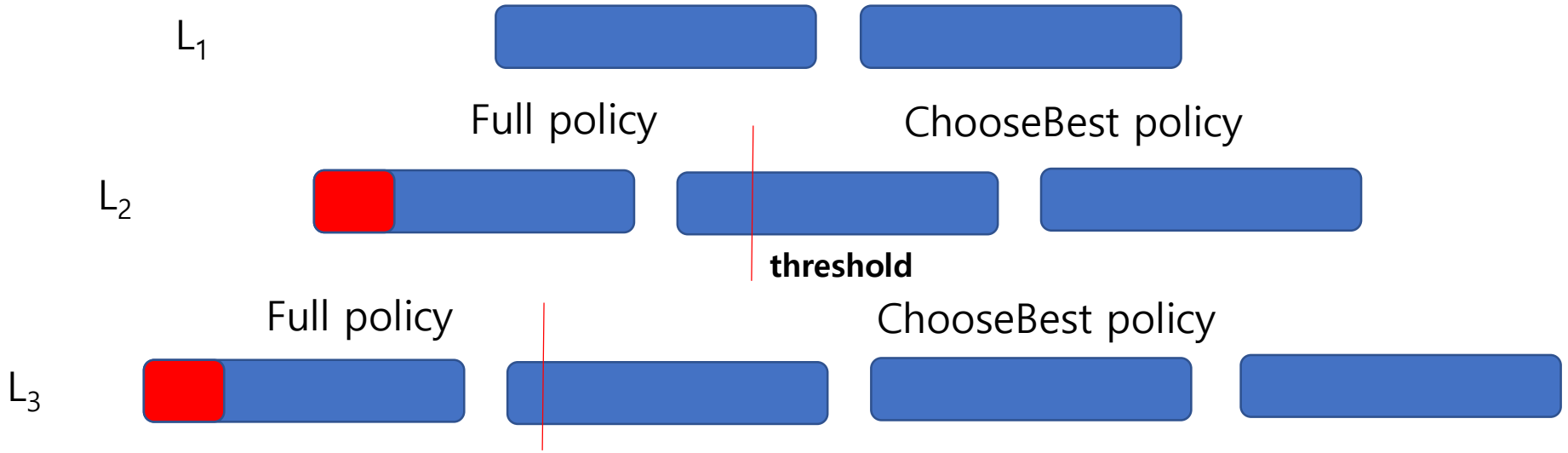
- When L_2 is affordable, Full policy can be applied.
- Then cost of L_1 is lower

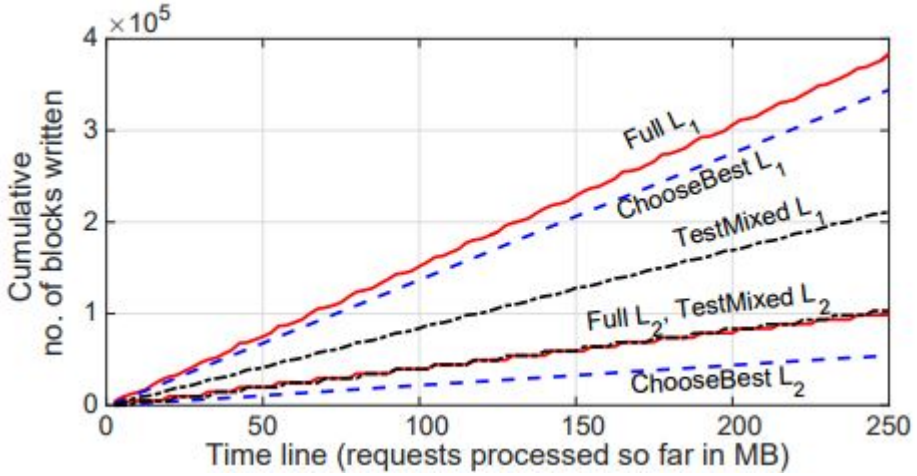




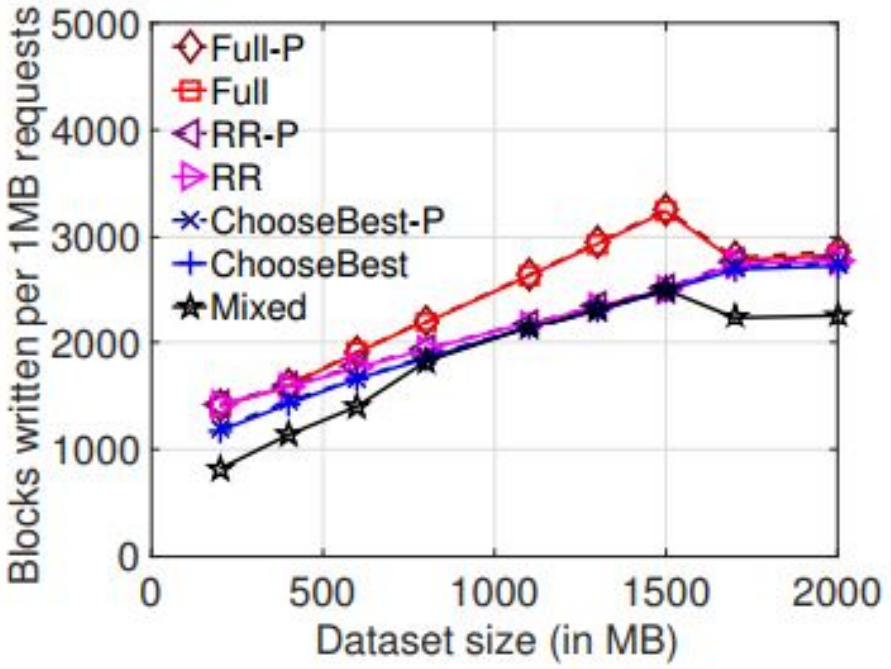


Mixed Policy





Experiment Result



Thank you