

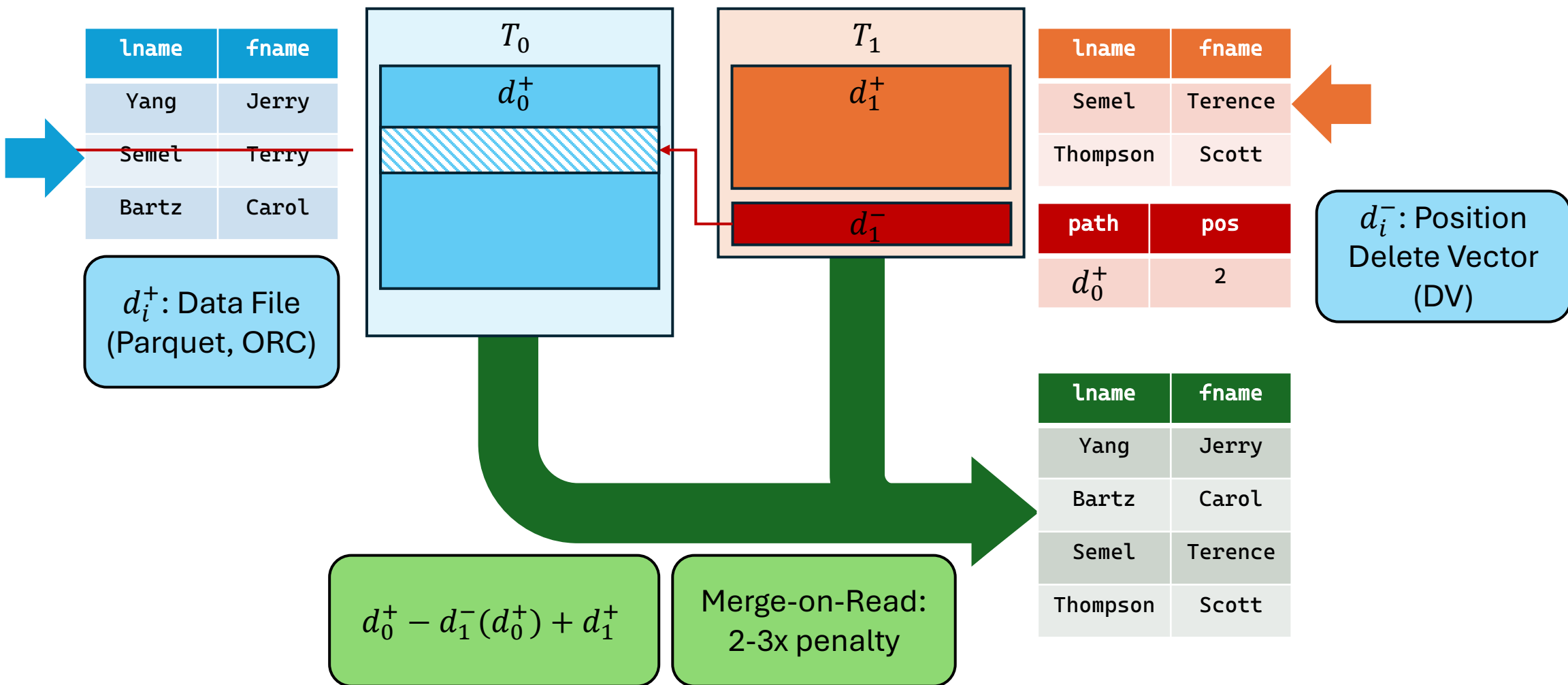


# Commutative Compaction

**Chris Douglas**, Joseph M. Hellerstein

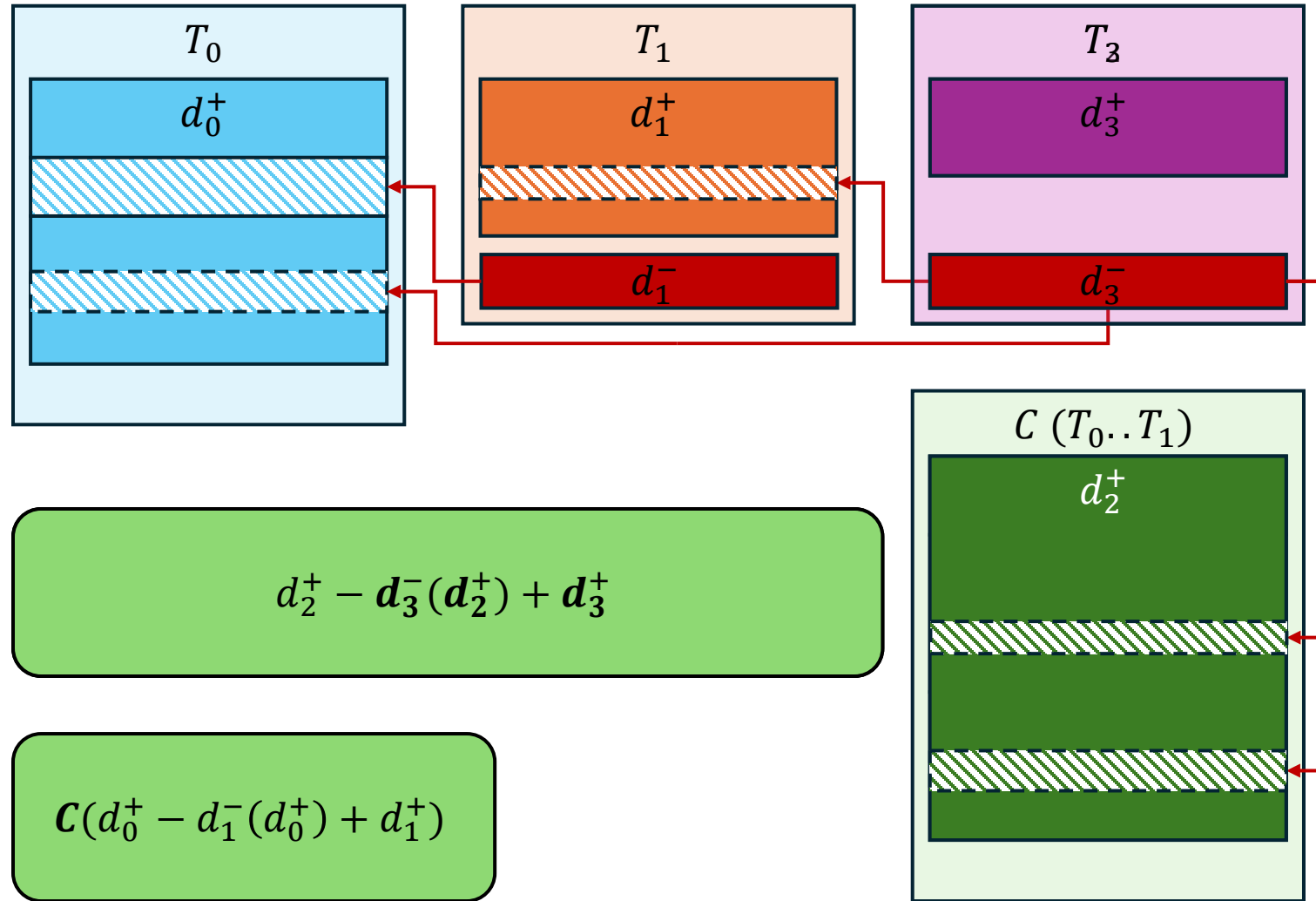
UC Berkeley

# Compaction

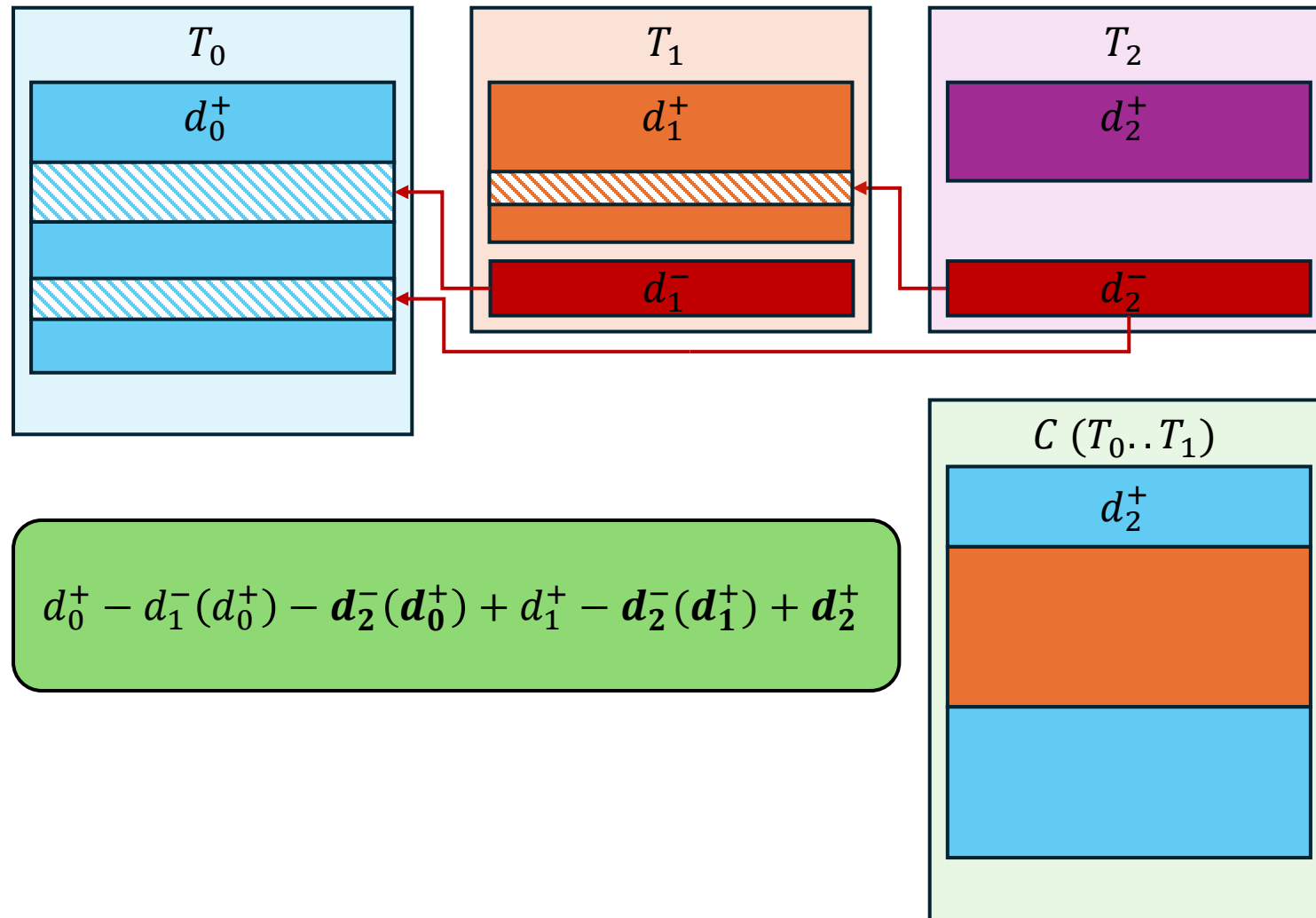




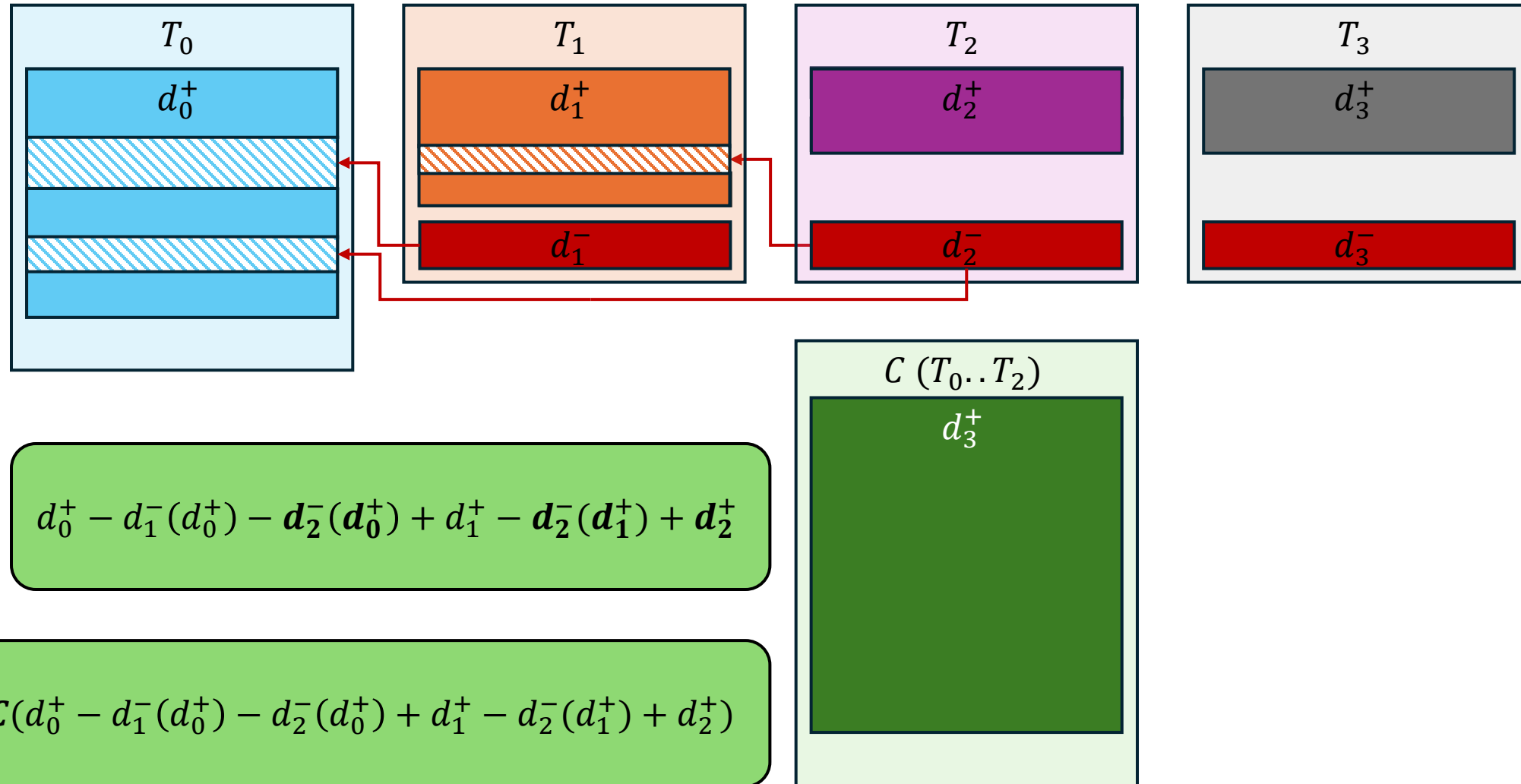
# Compaction



# Compaction



# Compaction

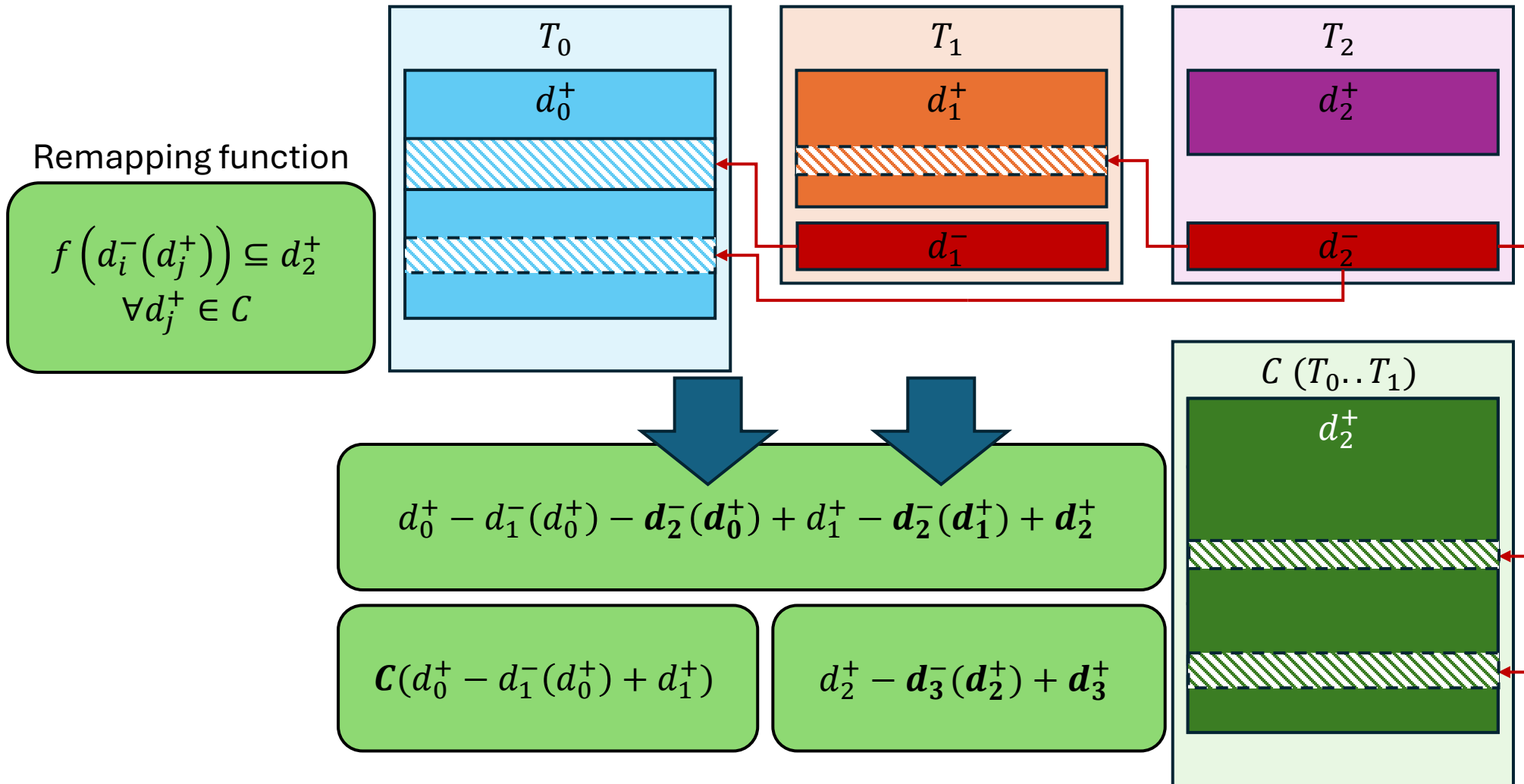


deref →

does *not* commute

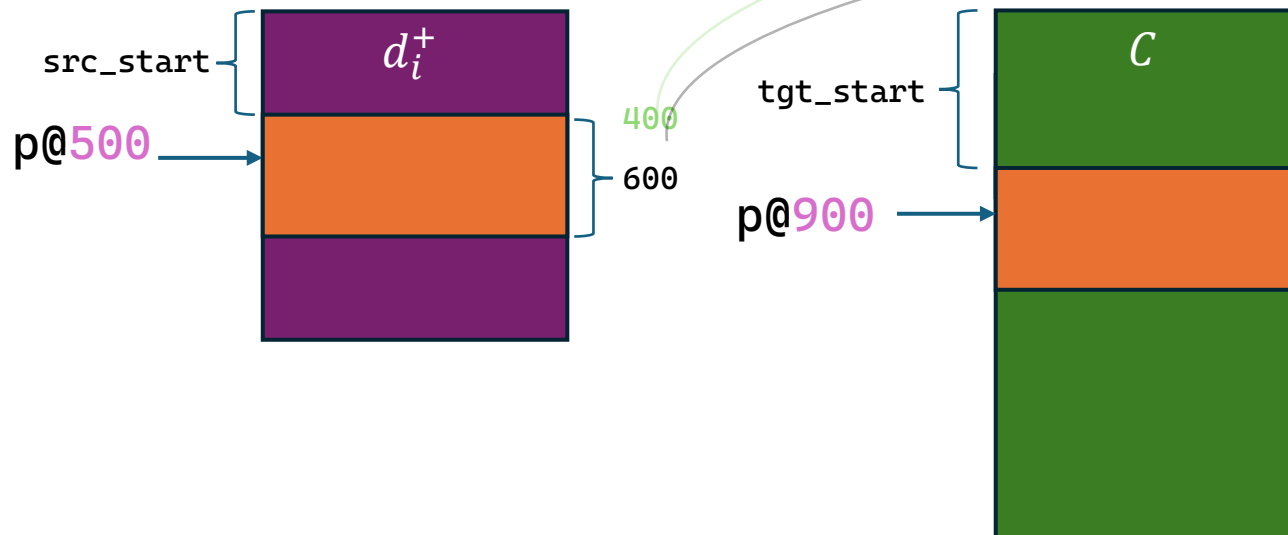
reorder\_rows ⇄

# Compaction: Layout Conflicts



# Compaction Maps

- Definition: set of *remapping functions*
- Intuition: contiguous ranges remapped/deleted
- Compact: 2.6KiB for 10 runs and 8.9KiB for 10k runs

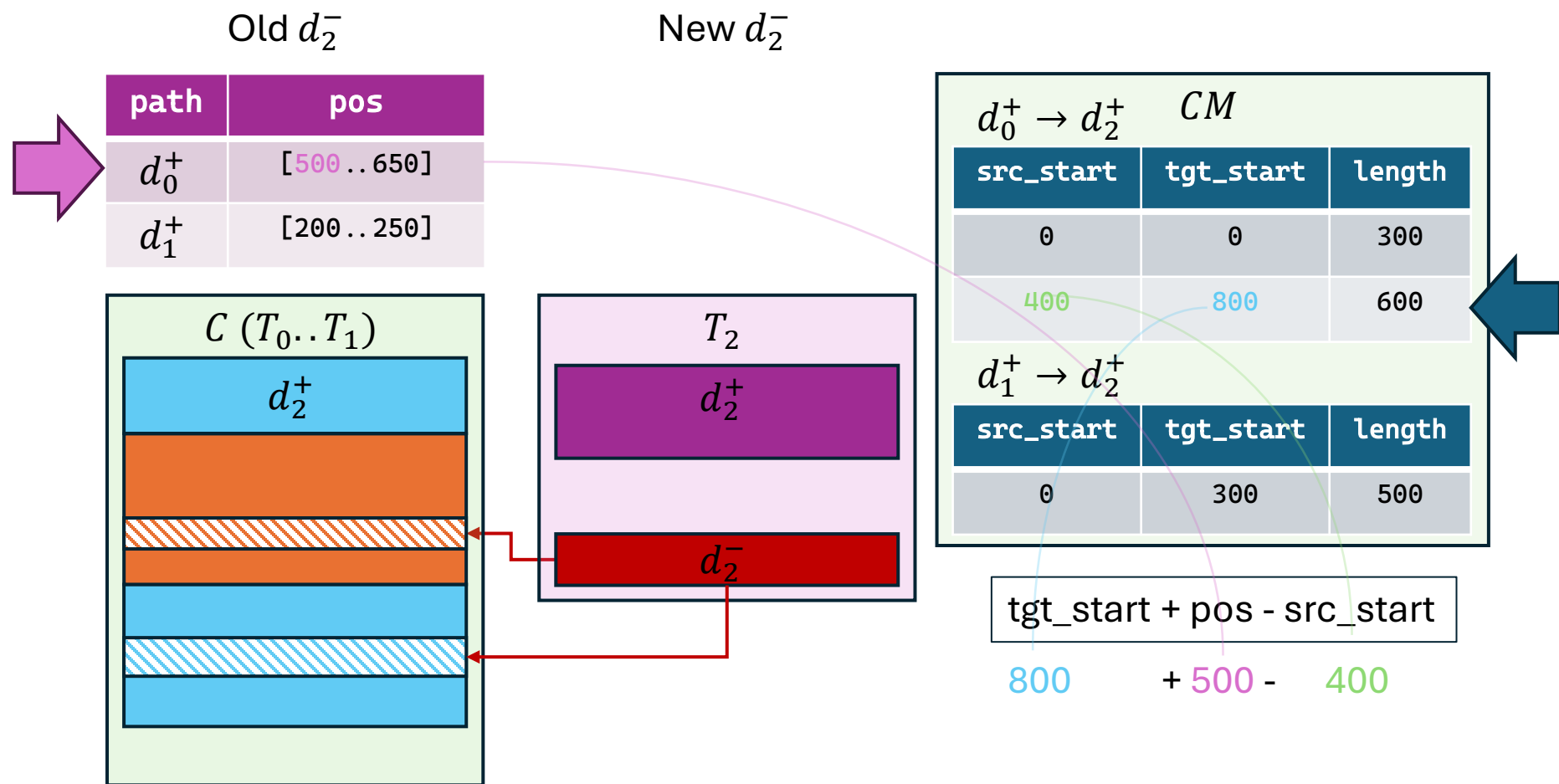


| $d_0^+ \rightarrow d_2^+$ $CM$ |           |        |
|--------------------------------|-----------|--------|
| src_start                      | tgt_start | length |
| 0                              | 0         | 300    |
| 400                            | 800       | 600    |

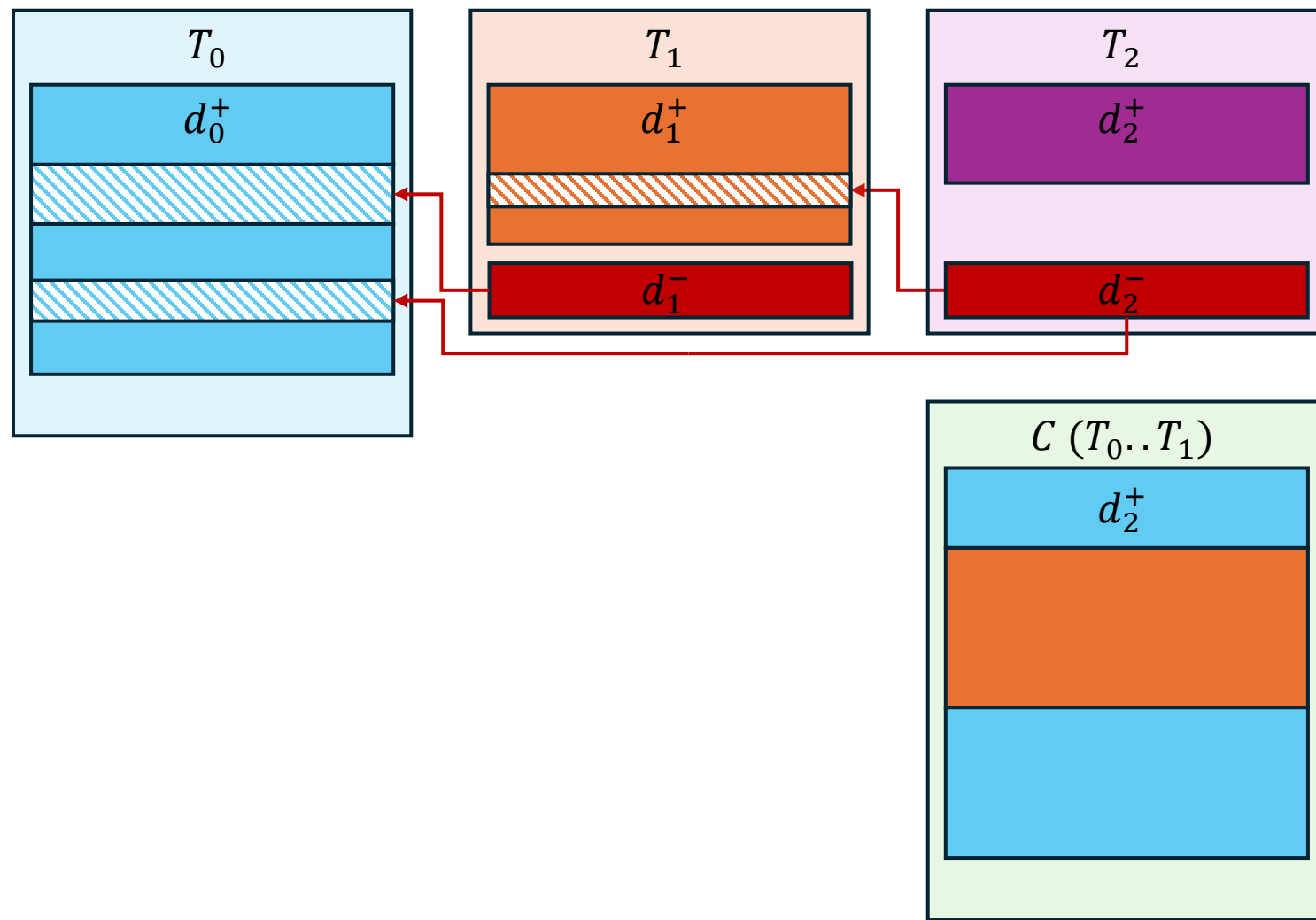
  

| $d_1^+ \rightarrow d_2^+$ |           |        |
|---------------------------|-----------|--------|
| src_start                 | tgt_start | length |
| 0                         | 300       | 500    |

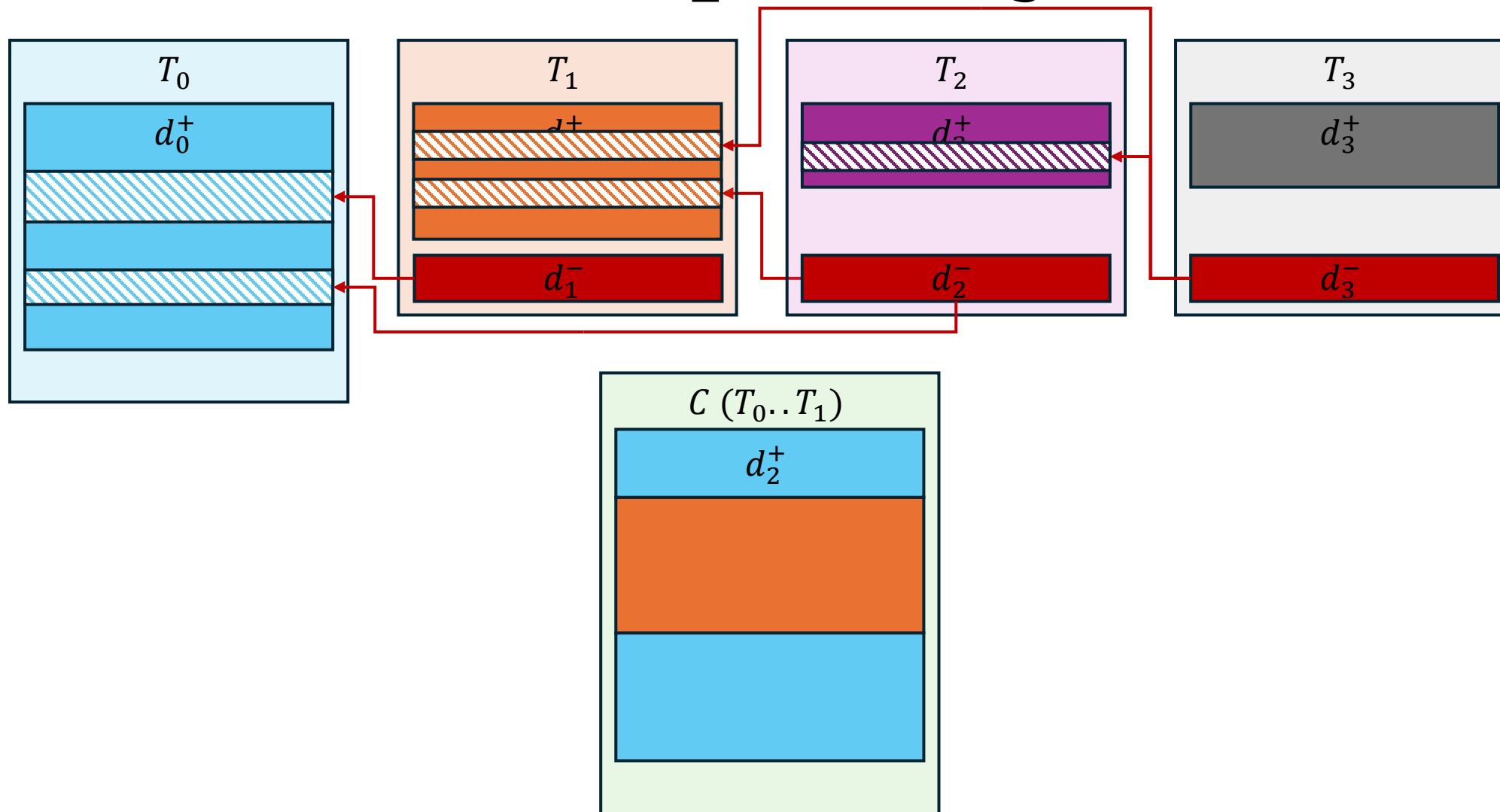
# Commute $T_2$ with $C$



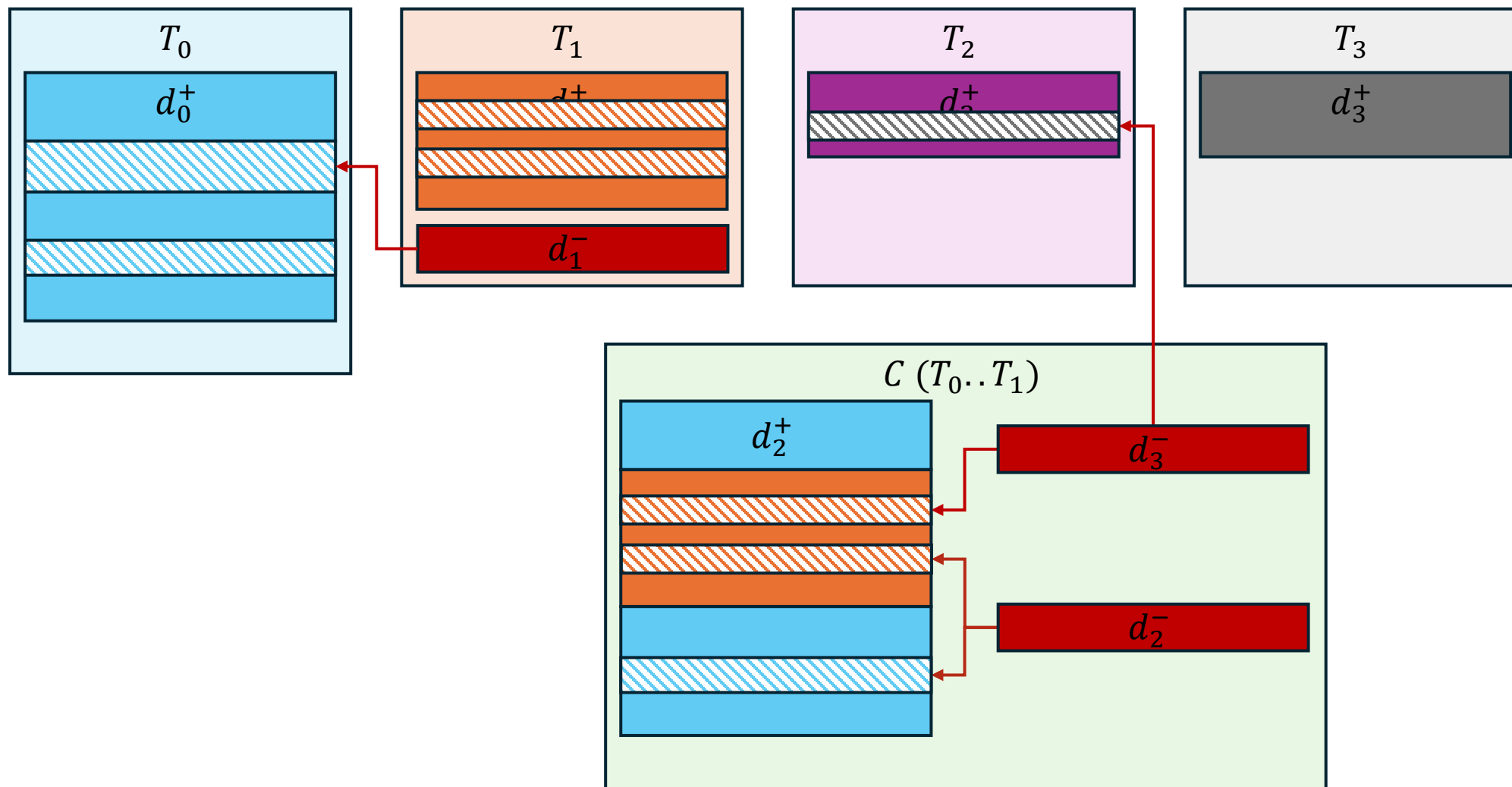
# Commute $\mathcal{C}$ with $T_2$



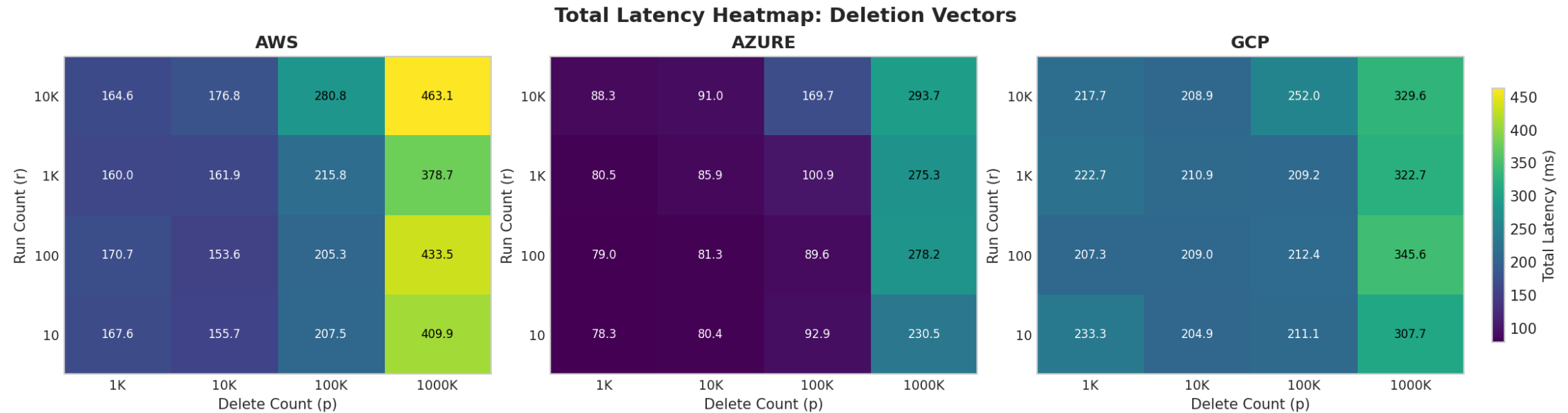
# Commute $C$ with $T_2 \dots$ and $T_3$



# Commute $\mathcal{C}$ with $T_2 \dots$ and $T_3$

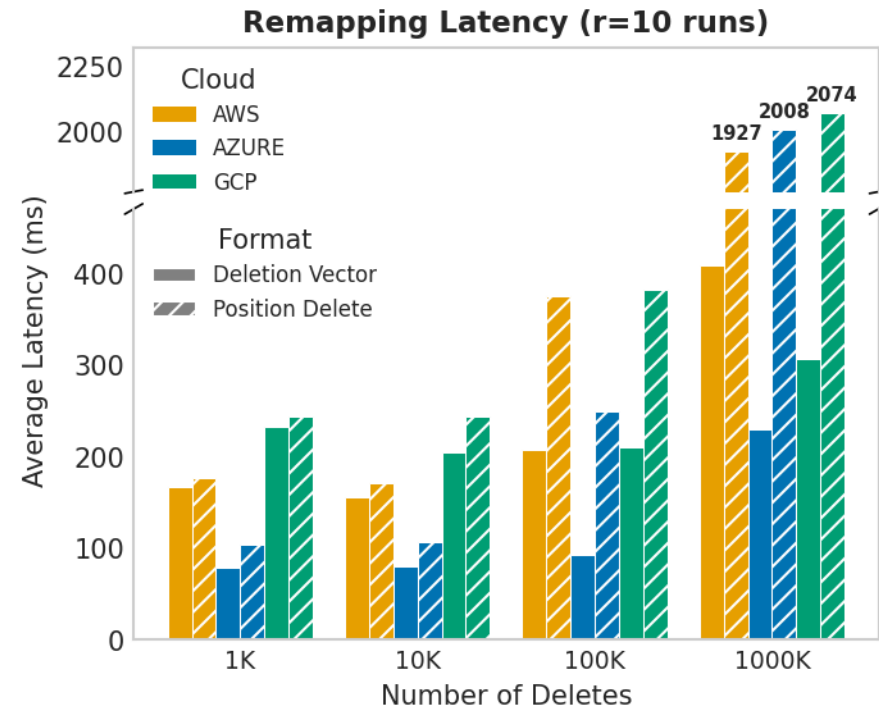
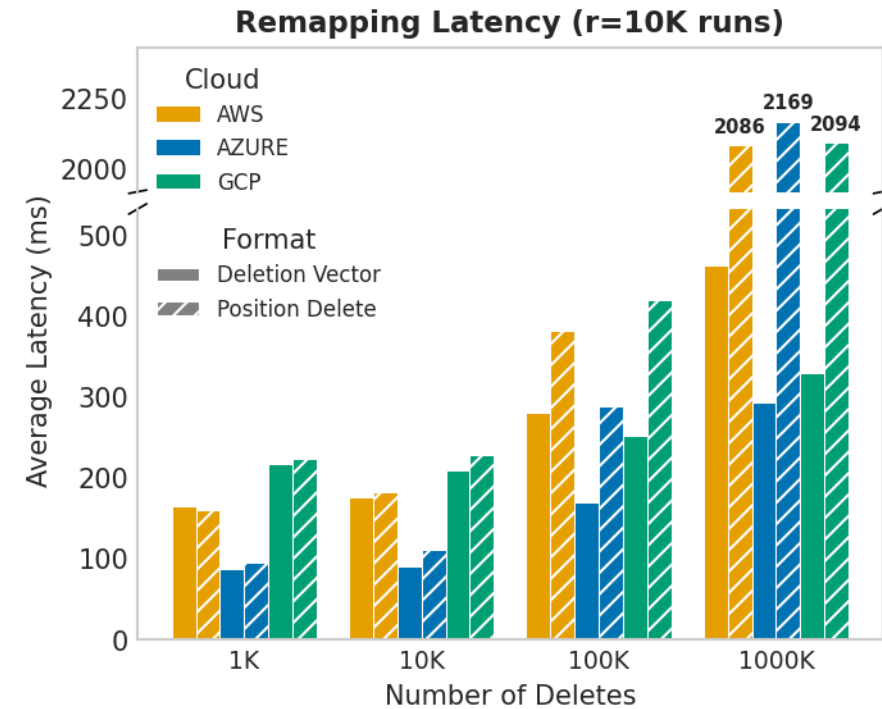


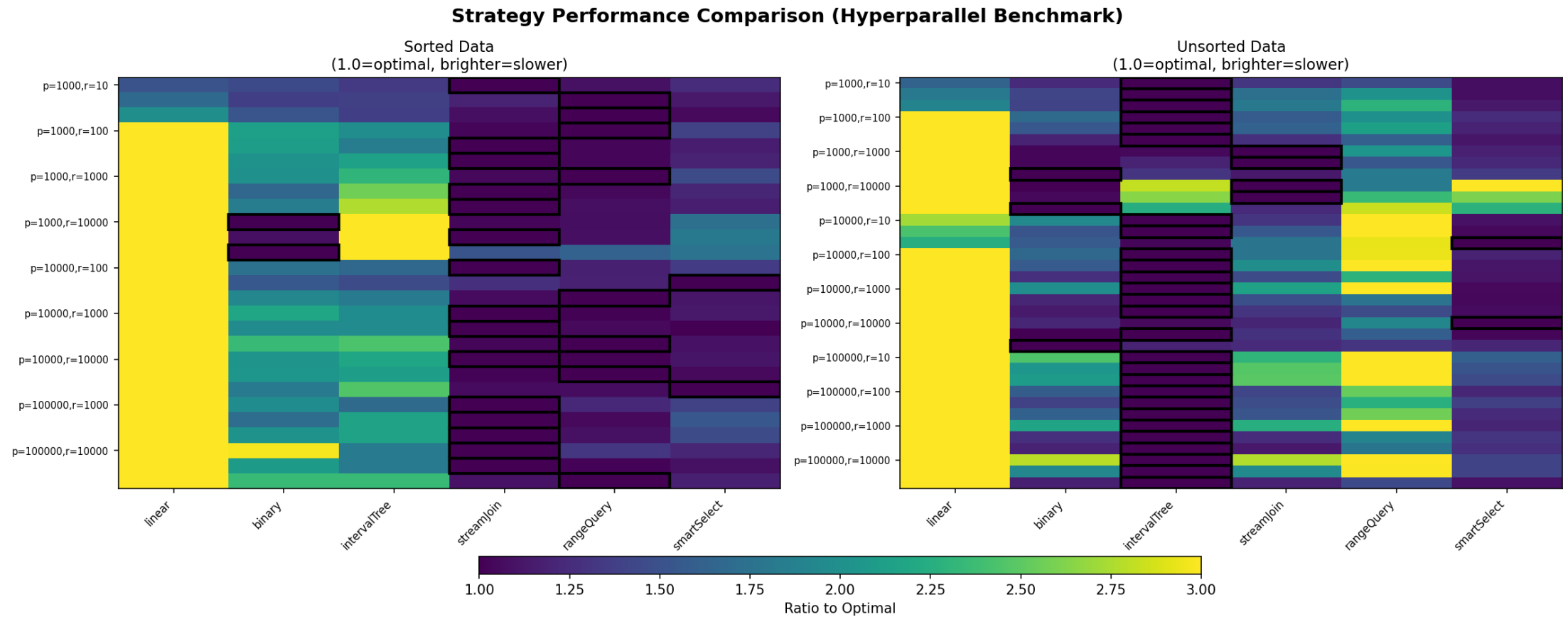
# Remapping Latency: Iceberg v3 (DV)



Re-running compaction: 1-5 minutes  
7200x-360000x speedup

# Remapping Latency: end-to-end



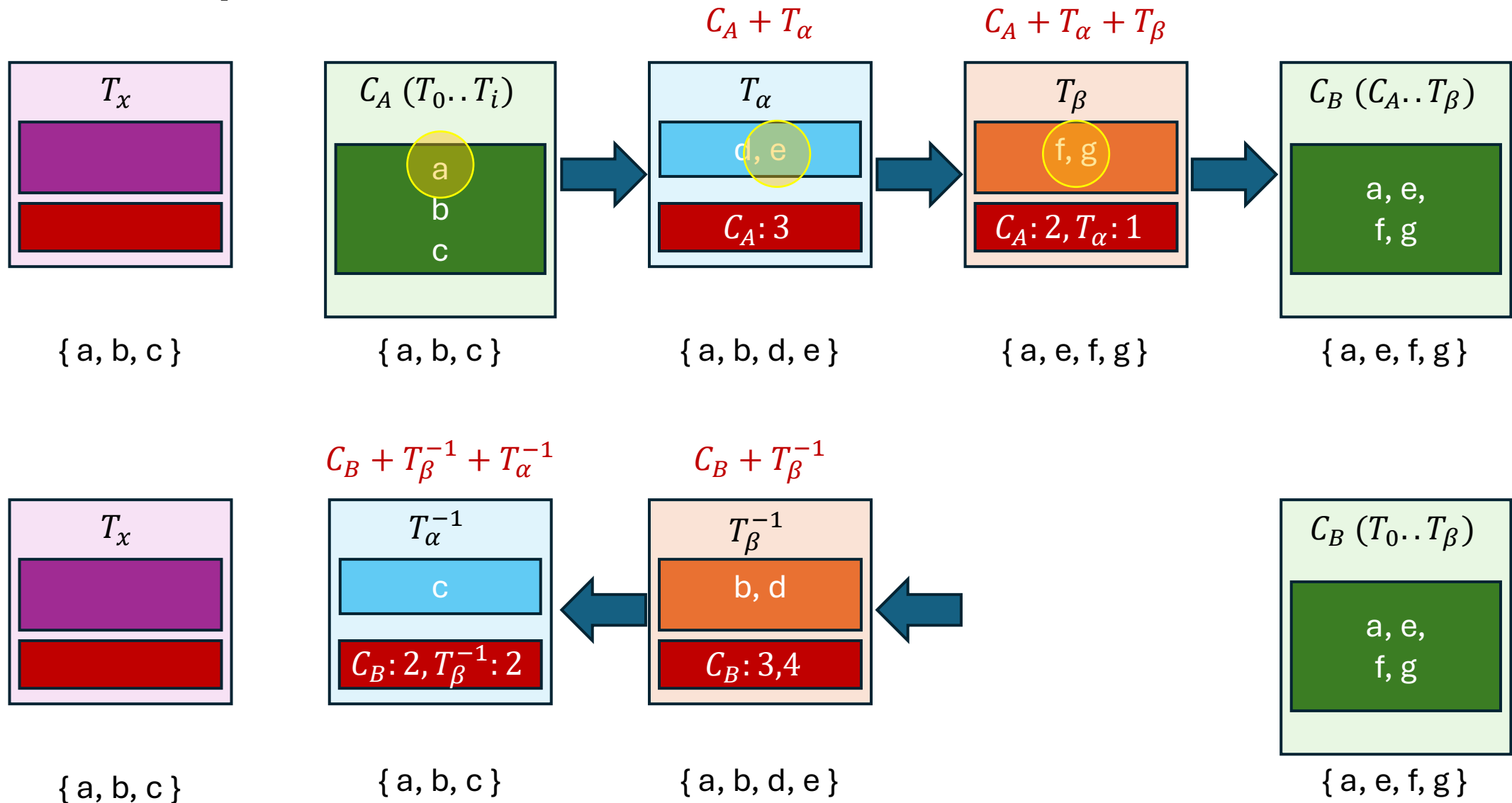




# Future Work

- SERIALIZABLE detects spurious read conflicts
- SORT, Z-ORDER compactions
  - Too large? Change granularity of the map?
- Other maintenance: deduplication, GDPR
  - Rebase applications that do their own dedup?
- Merge index (REMIX, Zhong et al. FAST'21)
  - “Anchor keys” improve lookup + scan without a compaction
- Checkpoint inversion
  - Rewrite transaction snapshots to reference newer checkpoints

# Checkpoint Inversion



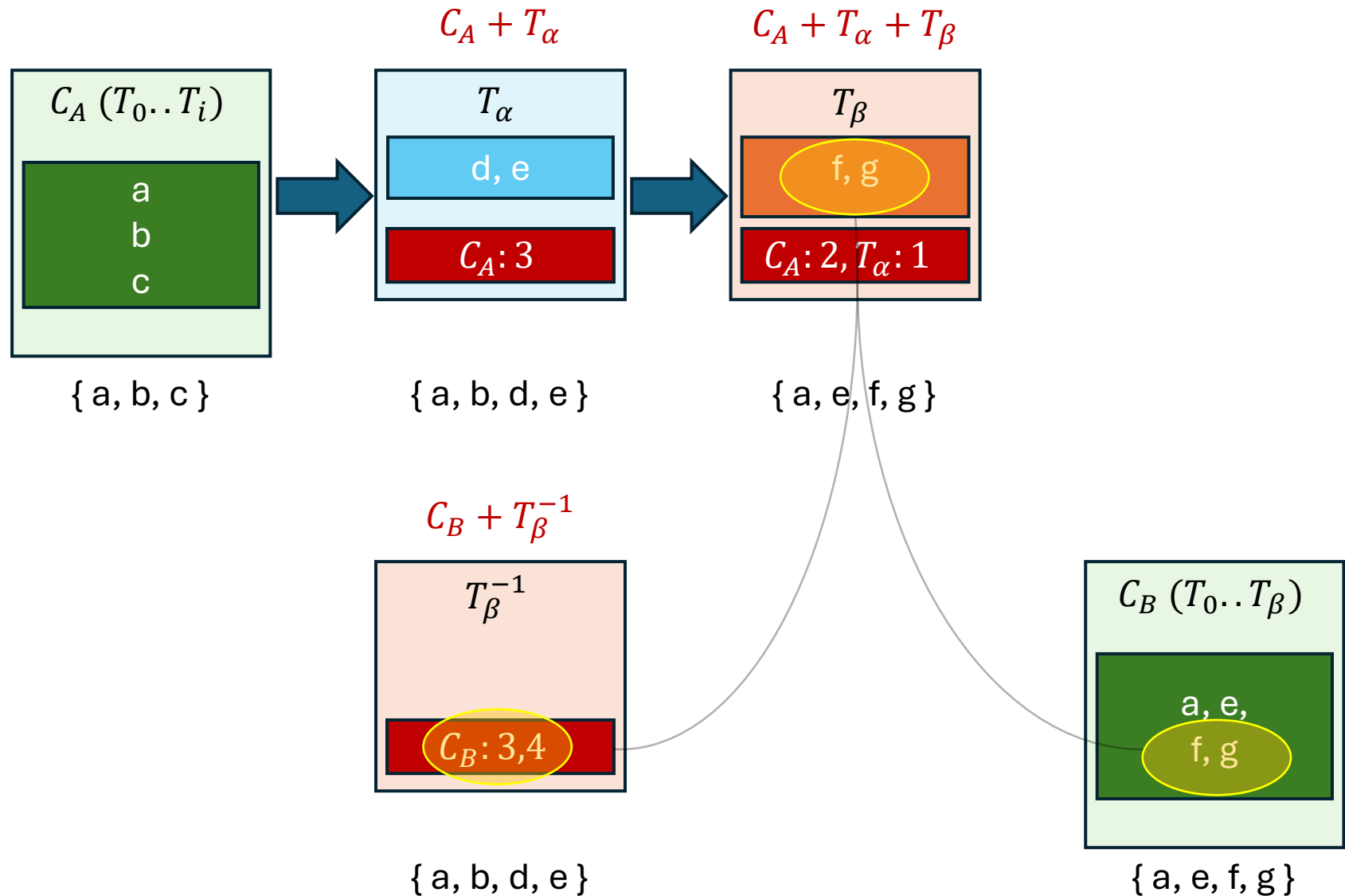
# Checkpoint Inversion

Invert  $T_\beta$

1. Invert  $d_\beta^+$

i) CM:  $d_\beta^+ \rightarrow C_B$

ii) write  $d_{\beta^{-1}}^-$



# Checkpoint Inversion

Invert  $T_\beta$

1. Invert  $d_\beta^+$

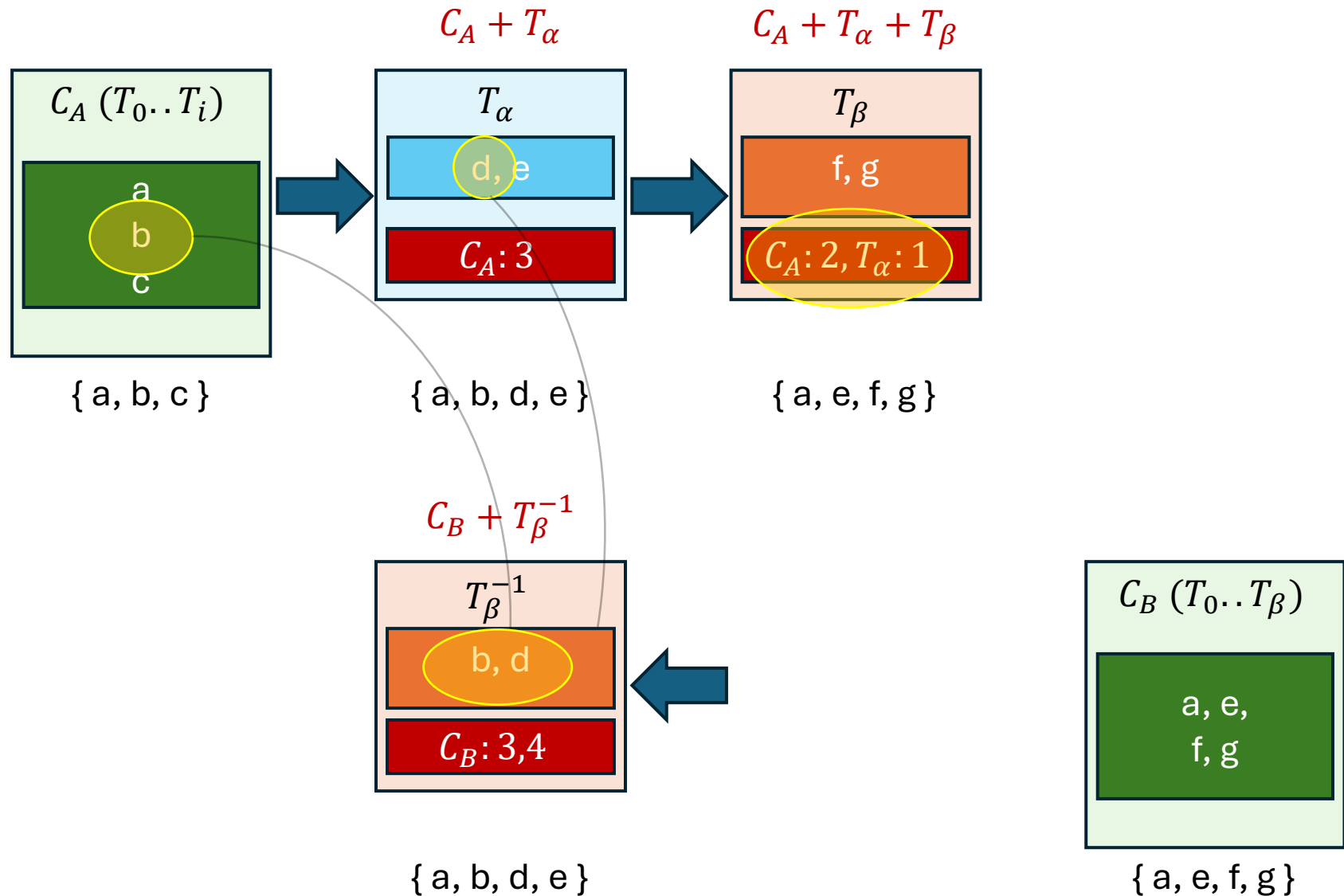
i) CM:  $d_\beta^+ \rightarrow C_B$

ii) write  $d_\beta^{-1}$

2. Invert  $d_\beta^-$

i) Resolve  $d_\beta^-$

ii) Recall  $T_\alpha$  position



# Checkpoint Inversion

Invert  $T_\beta$

1. Invert  $d_\beta^+$

i) CM:  $d_\beta^+ \rightarrow C_B$

ii) write  $d_{\beta-1}^-$

2. Invert  $d_\beta^-$

i) Resolve  $d_\beta^-$

ii) Recall  $T_\alpha$  position

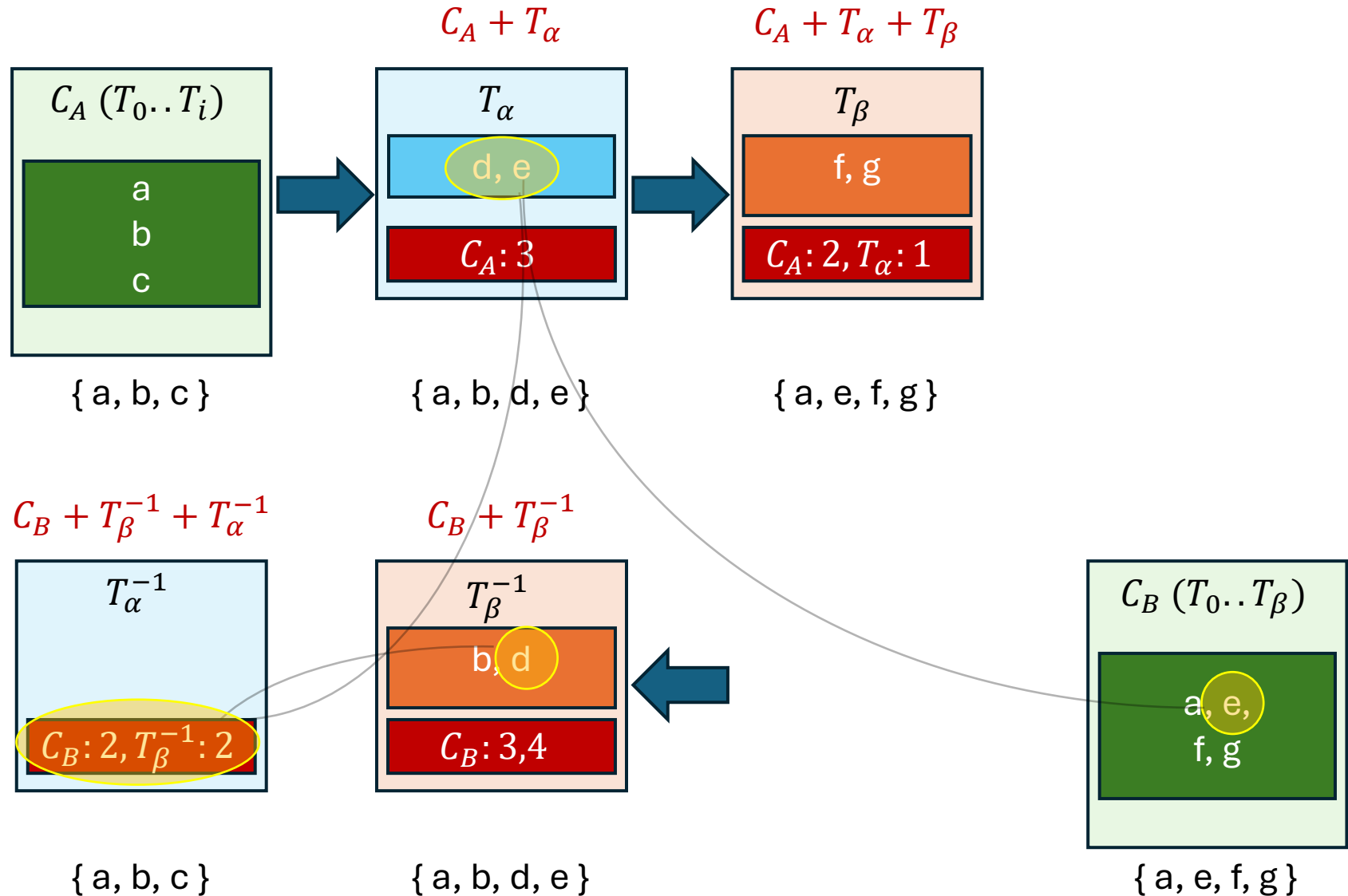
Invert  $T_\alpha$

1. Invert  $d_\alpha^+$

i) CM:  $d_\alpha^+ \rightarrow C_B$

ii)  $d_\alpha^+ \rightarrow d_{\beta-1}^+$

iii) write  $d_{\alpha-1}^-$



# Checkpoint Inversion

Invert  $T_\beta$

1. Invert  $d_\beta^+$

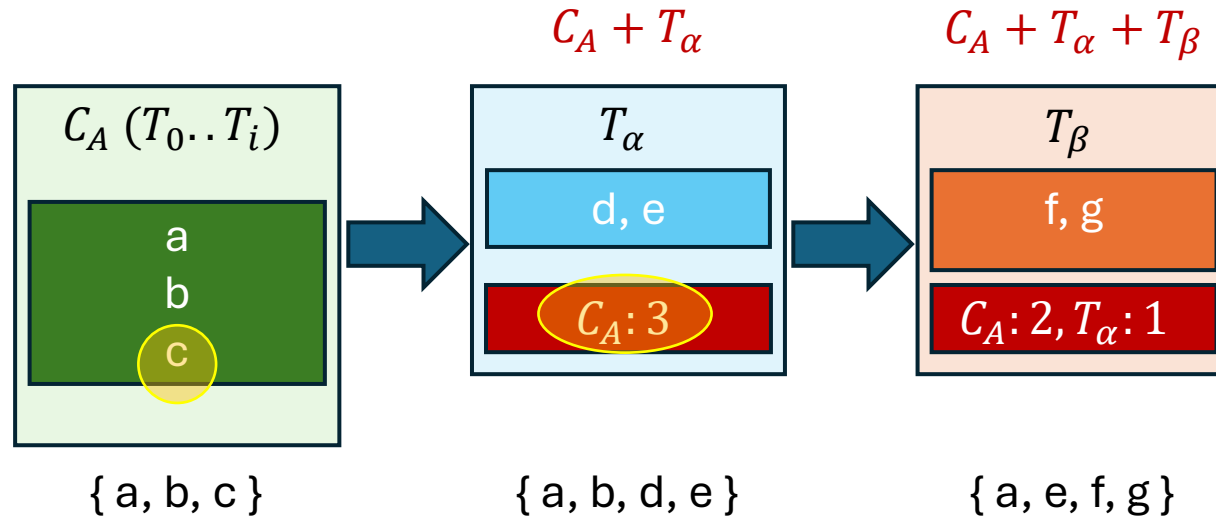
i) CM:  $d_\beta^+ \rightarrow C_B$

ii) write  $d_{\beta-1}^-$

2. Invert  $d_\beta^-$

i) Resolve  $d_\beta^-$

ii) Recall  $T_\alpha$  position



Invert  $T_\alpha$

1. Invert  $d_\alpha^+$

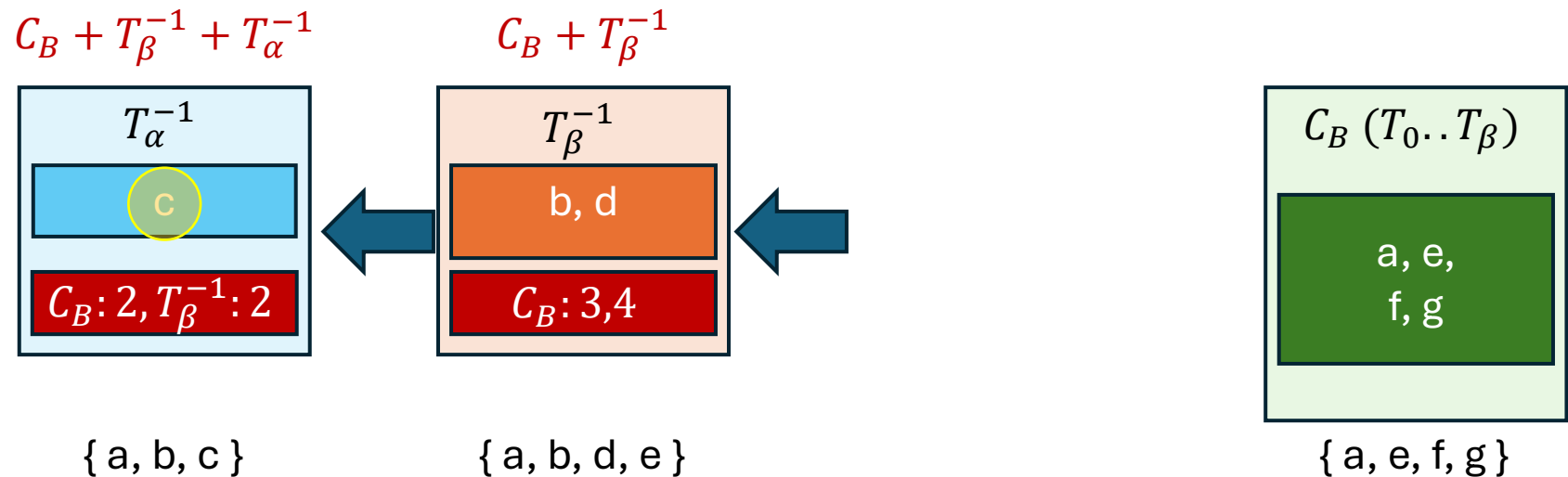
i) CM:  $d_\alpha^+ \rightarrow C_B$

ii)  $d_\alpha^+ \rightarrow d_{\beta-1}^+$

iii) write  $d_{\alpha-1}^-$

2. Invert  $d_\beta^-$

i) Resolve  $d_\alpha^-$



# Checkpoint Inversion

Invert  $T_\beta$

1. Invert  $d_\beta^+$

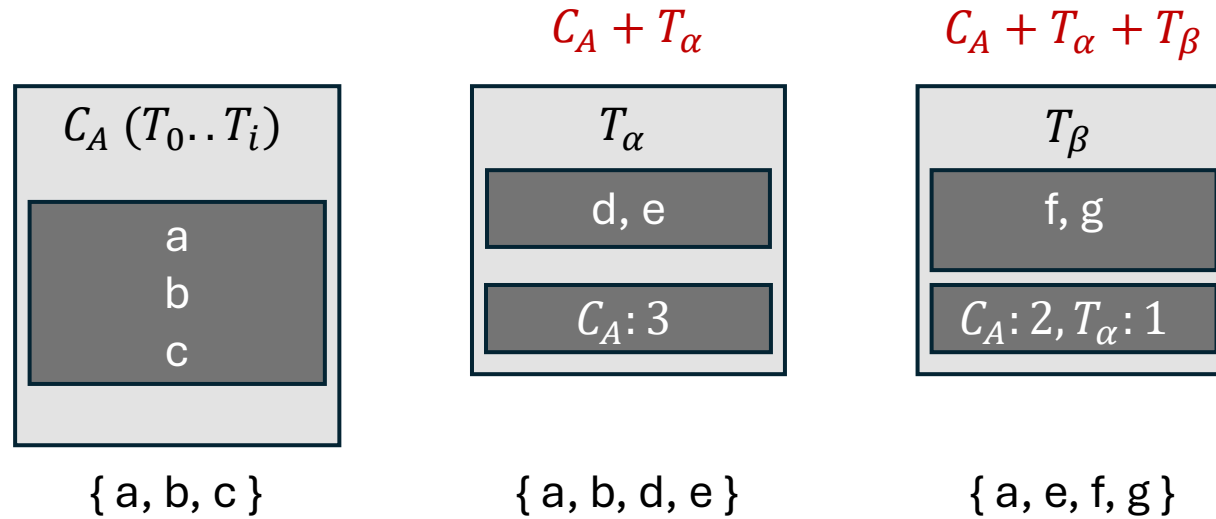
i) CM:  $d_\beta^+ \rightarrow C_B$

ii) write  $d_{\beta-1}^-$

2. Invert  $d_\beta^-$

i) Resolve  $d_\beta^-$

ii) Recall  $T_\alpha$  position



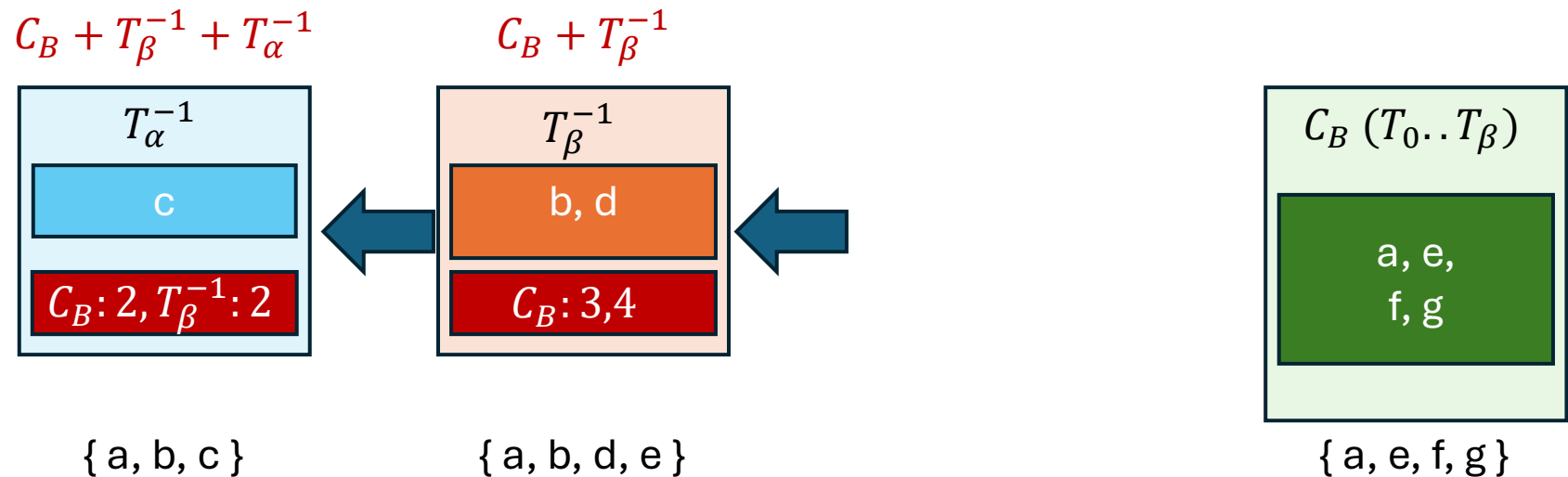
Invert  $T_\alpha$

1. Invert  $d_\alpha^+$

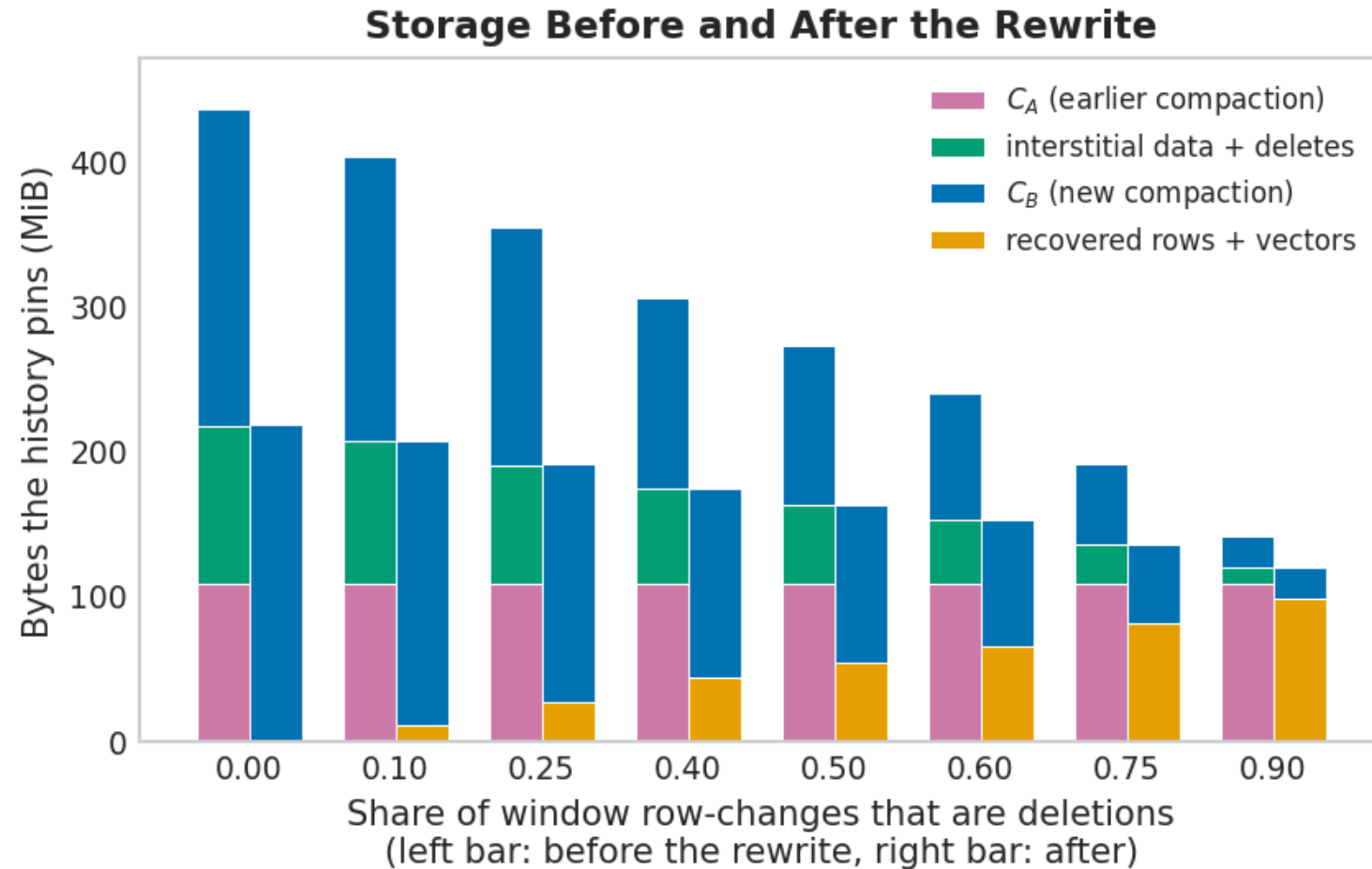
i) CM:  $d_\alpha^+ \rightarrow C_B$

ii)  $d_\alpha^+ \rightarrow d_{\beta-1}^+$

iii) write  $d_{\alpha-1}^-$

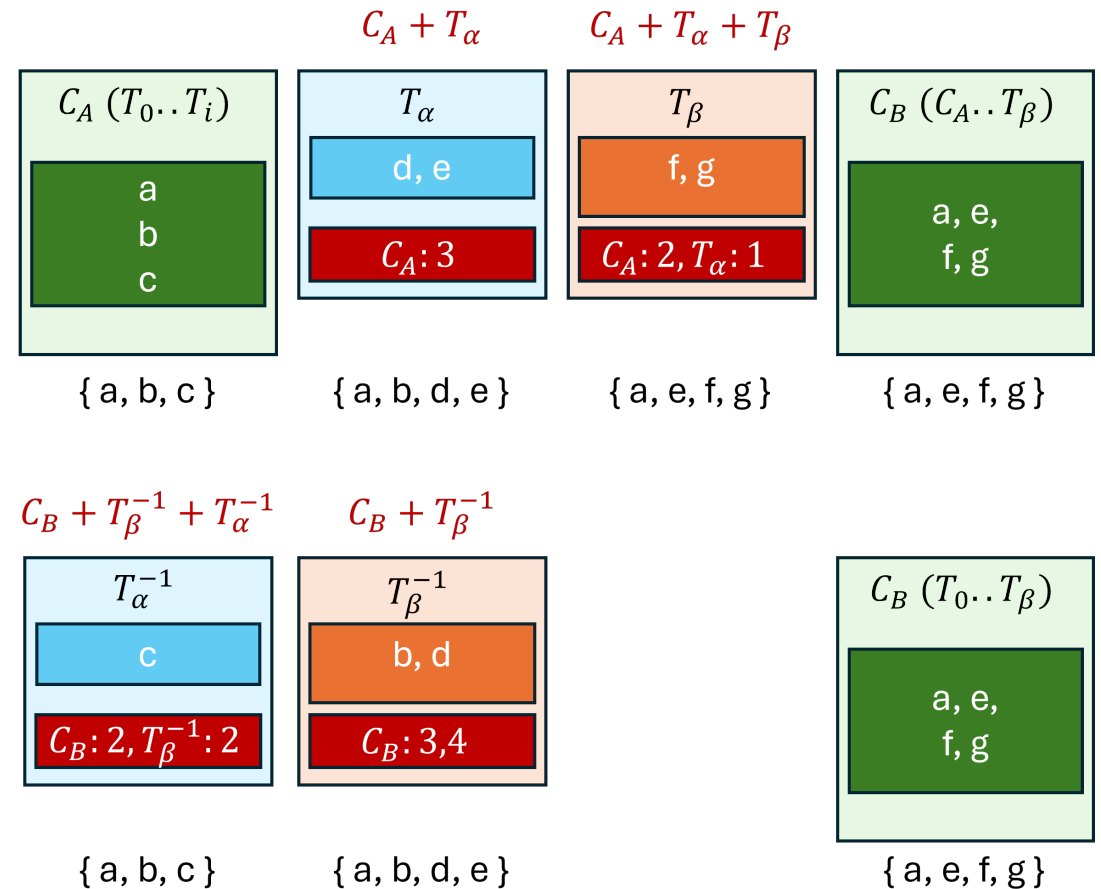


# Checkpoint Inversion: Storage Reclaimed



# Checkpoint Inversion: Breaking Changes

- Scrambles Provenance
  - “What changed?”
  - CDC logs, replication, audit
- Logical states still available
  - Materialized differences lost
- Row identity
  - `_row_id` saved for restore
  - need “inverse map” for position





# Thank you



- Avoid re-execution of compactions and transactions
- Writer-centric state to *repair* conflicts
- Physical layout impacts logical functions
  - Cooperative concurrency control



**Chris Douglas**  
[chris\\_douglas@berkeley.edu](mailto:chris_douglas@berkeley.edu)

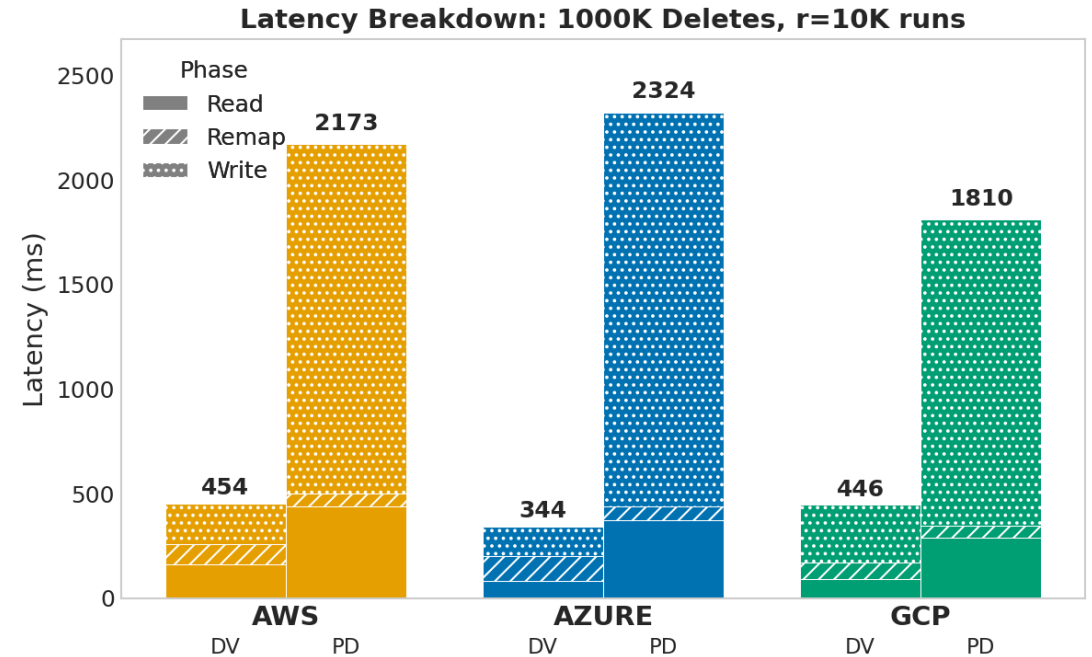
Joseph M. Hellerstein  
[hellerstein@berkeley.edu](mailto:hellerstein@berkeley.edu)

## Thank you

Ashvin Agrawal, Carlo Curino, Owen O'Malley, Tiemo  
Bang, Gopal Vijayaraghavan, Natacha Crooks, Micah  
Murray, Pavan Lanka, and Sumedh Sakdeo

# Remapping Latency

- Not I/O but encode/decode
- v2: Parquet encoding cost dominates
- v3: PD files are rewritten as DV in Puffin files



# Encoding Cost

## Insert-Only Window: Reclaim vs Window Depth

