Online Deduplication for Databases

Lianghong Xu[1], Andy Pavlo[1]
Sudipta Sengupta[2], Greg Ganger[1]
Carnegie Mellon University[1], Microsoft Research[2]

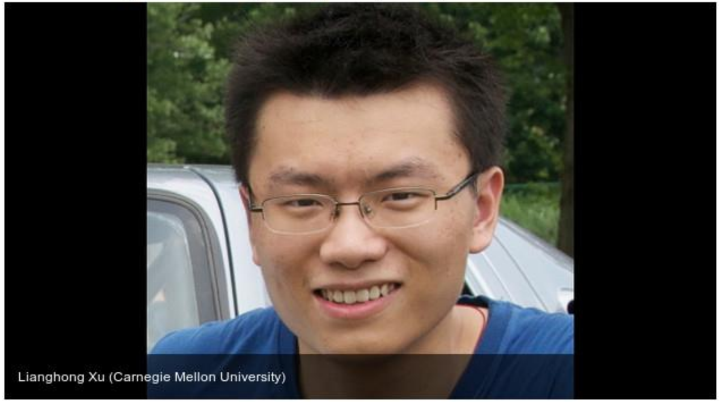
Carnegie Mellon
Parallel Data Laboratory

CMU Database Student Stabbed By Rival DBAs

UPDATED | May 2, 2017 2:00 PM

May 2, 2017 1:54 PM

Filed Under: Carnegie Mellon University, Pittsburgh Police, Stabbing, student stabbed



Lianghong Xu (Carnegie Mellon University)

Follow CBSDFW.COM: Facebook | Twitter

PITTSBURGH (CBSDFW.COM) – Pittsburgh Police say an 26-year-old database PhD student at Carnegie Mellon Universiy was stabbed during a fight with a gang of database administrators (DBAs). This gang has had a long-standing feud with the CMU Database Research group.

POPULAR

President Trump Defends Sharing Information With Russians

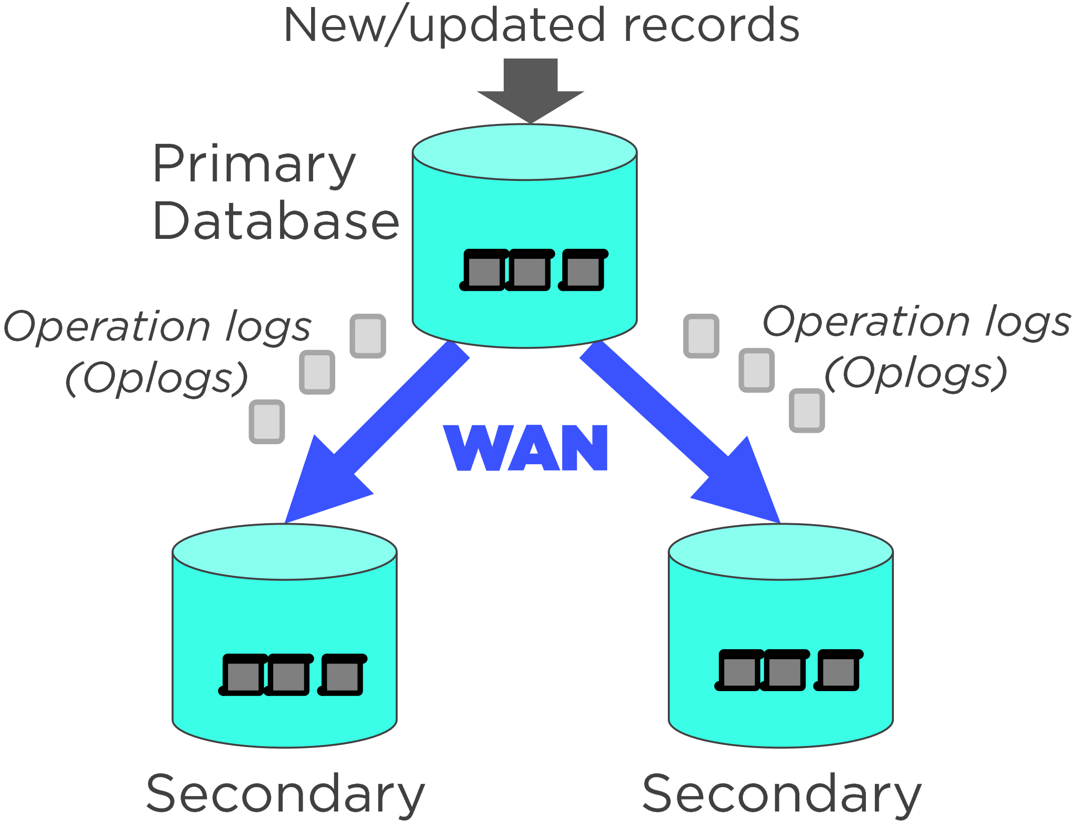
North Texas City Warns of Scam On Social Media

Dallas Police Officer Arrested

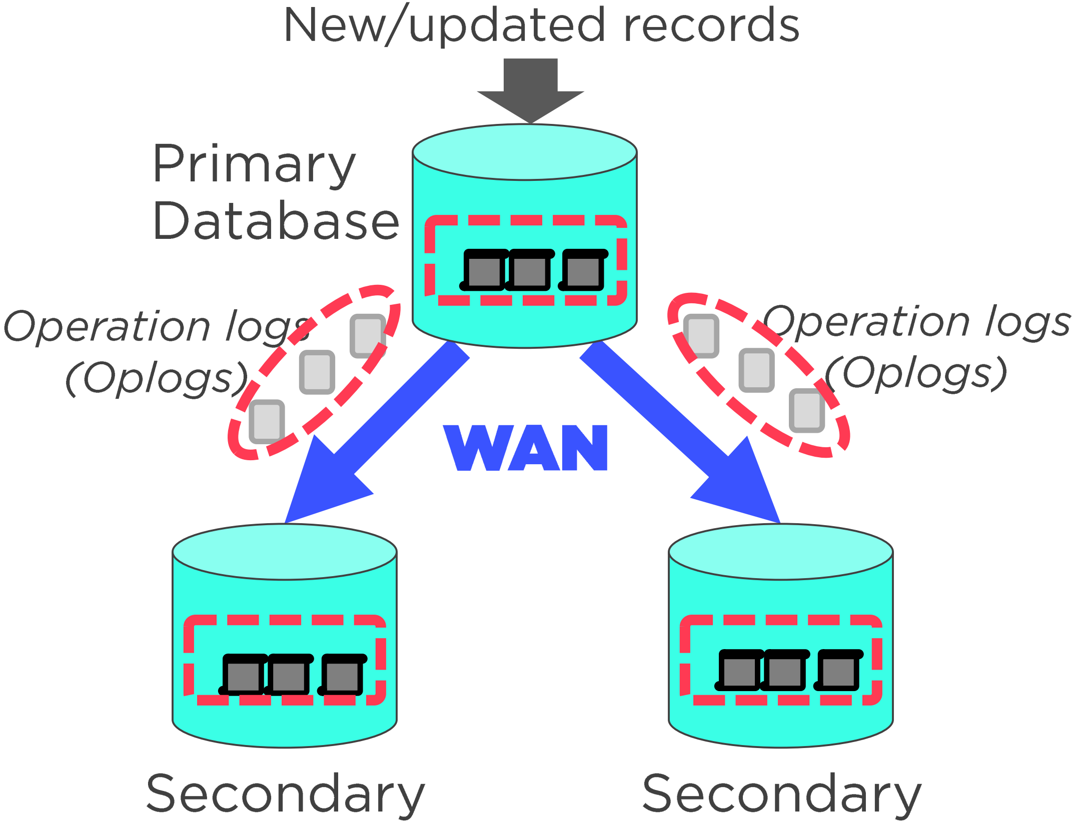
LIKE THIS

Police: Child Sexually

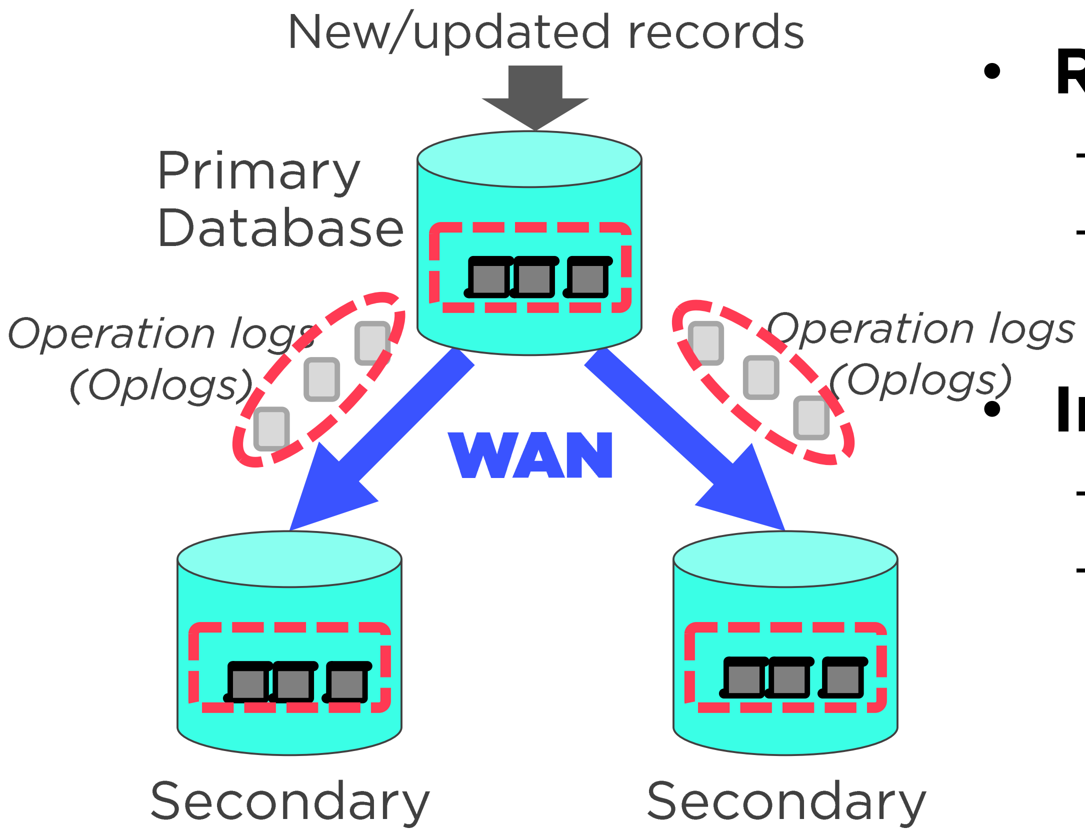
Goal: Data Reduction in DBMSs



Goal: Data Reduction in DBMSs



Goal: Data Reduction in DBMSs



- **Reduce cost**
  - *Storage & network hardware*
  - *Maintenance*
- **Improve Performance**
  - *Reduce disk I/Os*
  - *Lower replication bandwidth*

Current Reduction Approaches

- Block-level compression
  - *Simple, effective*
  - *Fails to address redundancy across blocks*

- Chunk-based Deduplication
  - *Identify global redundancy across data corpus*
  - *Not suitable for database workloads*
    - Records are relatively small
    - Modifications are small and dispersed

Similarity-based Dedup

- sDedup [Xu et al., SoCC'15]
  - *Use byte-level delta compression against similar records to achieve high compression ratio*
  - *Focused on reduction in network bandwidth for replication*
- dbDedup (this work)
  - *Achieve superior compression on both network and storage*
  - *Address challenges with accessing delta-encoded storage*
  - *Allow use of dedup in online DBMSs with low perf. overhead*

Outline

- Goal and Motivation
- **Need For Similarity-based Dedup**
- Our Approach: dbDedup
- Evaluation

Duplication Beyond A Block

- Application-level versioning
  - *Few apps take advantage of sys-level versioning*
  - *Revisions of data items seen as unrelated records*
  - *E.g., WordPress, Wikipedia*

- Partial record copying
  - *Inclusion between records not exposed to the DBMS*
  - *E.g., email reply/forward, quotes in message boards*

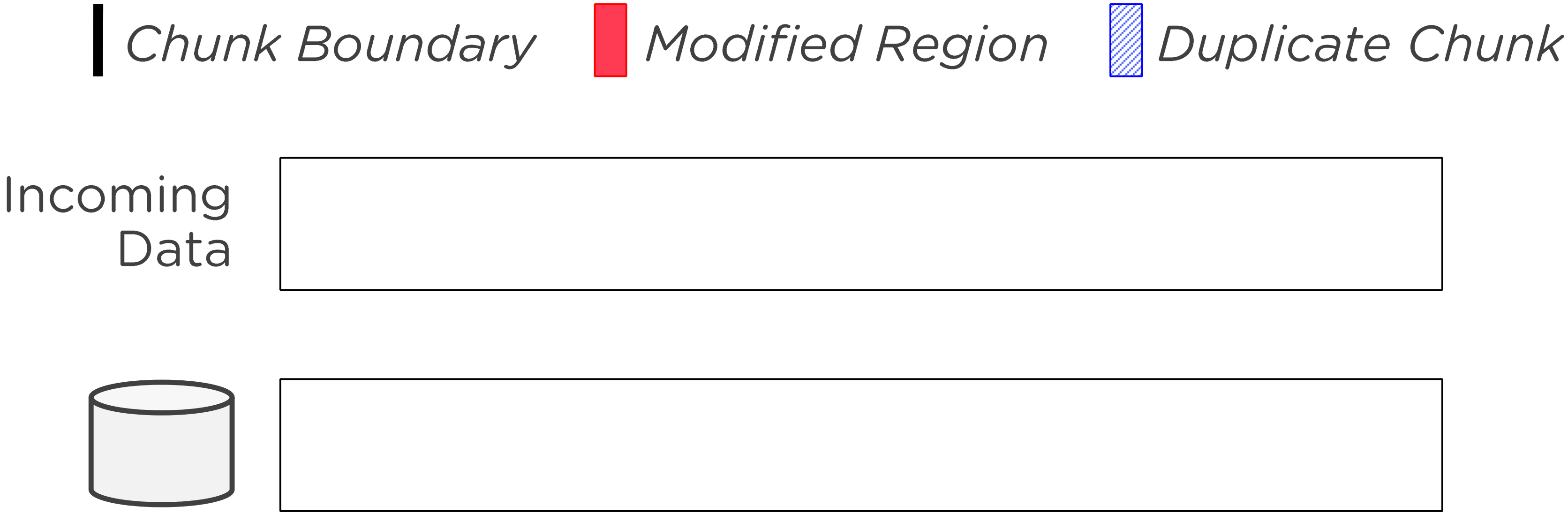
Traditional Dedup: Ideal Case

Chunk Boundary | Modified Region | Duplicate Chunk

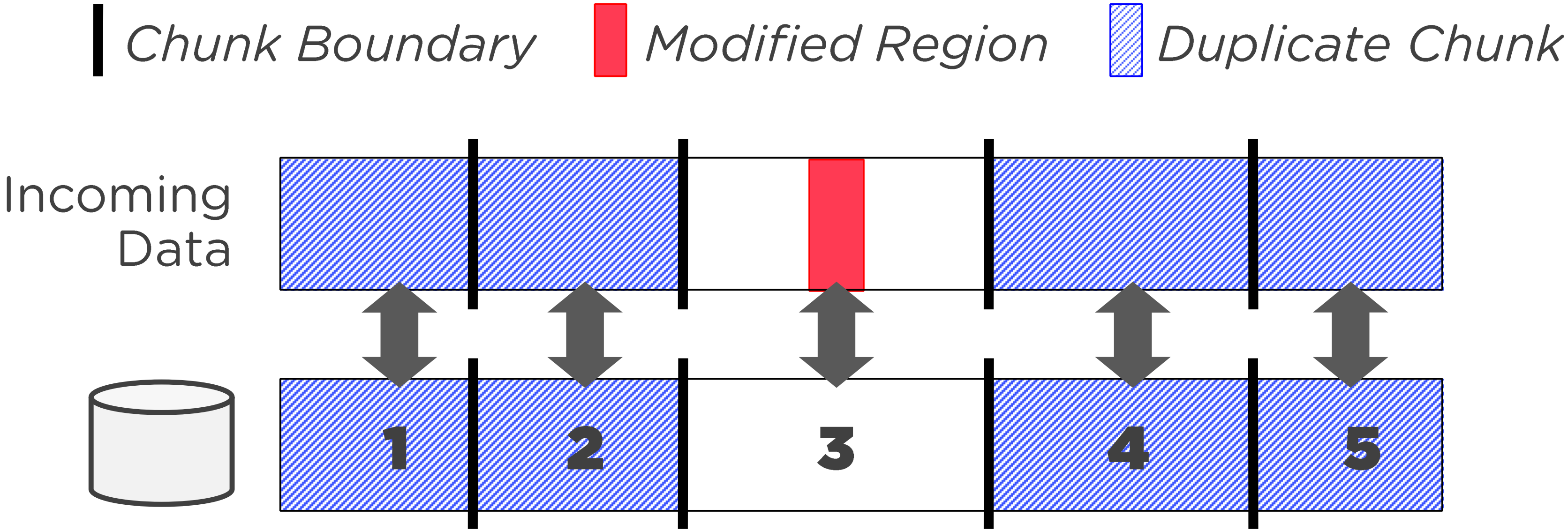
Incoming Data

{BYTE STREAM }

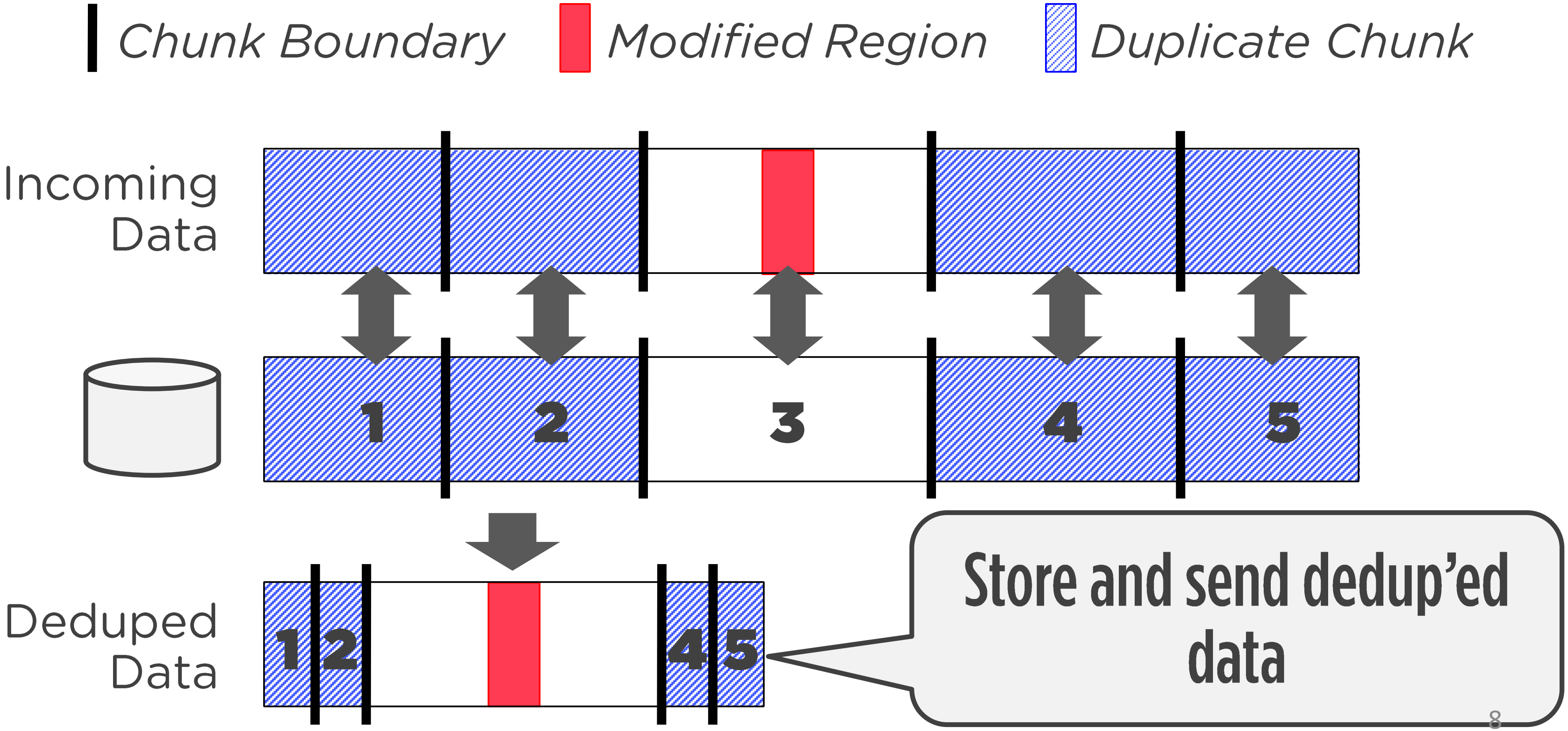
Traditional Dedup: Ideal Case



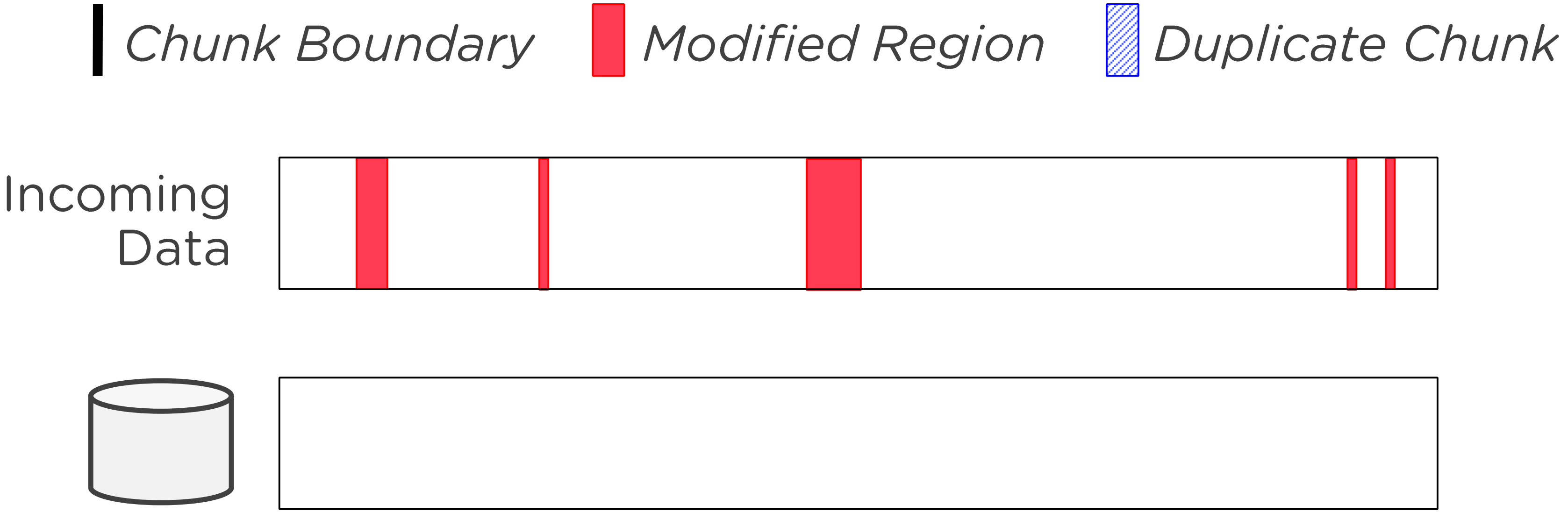
Traditional Dedup: Ideal Case



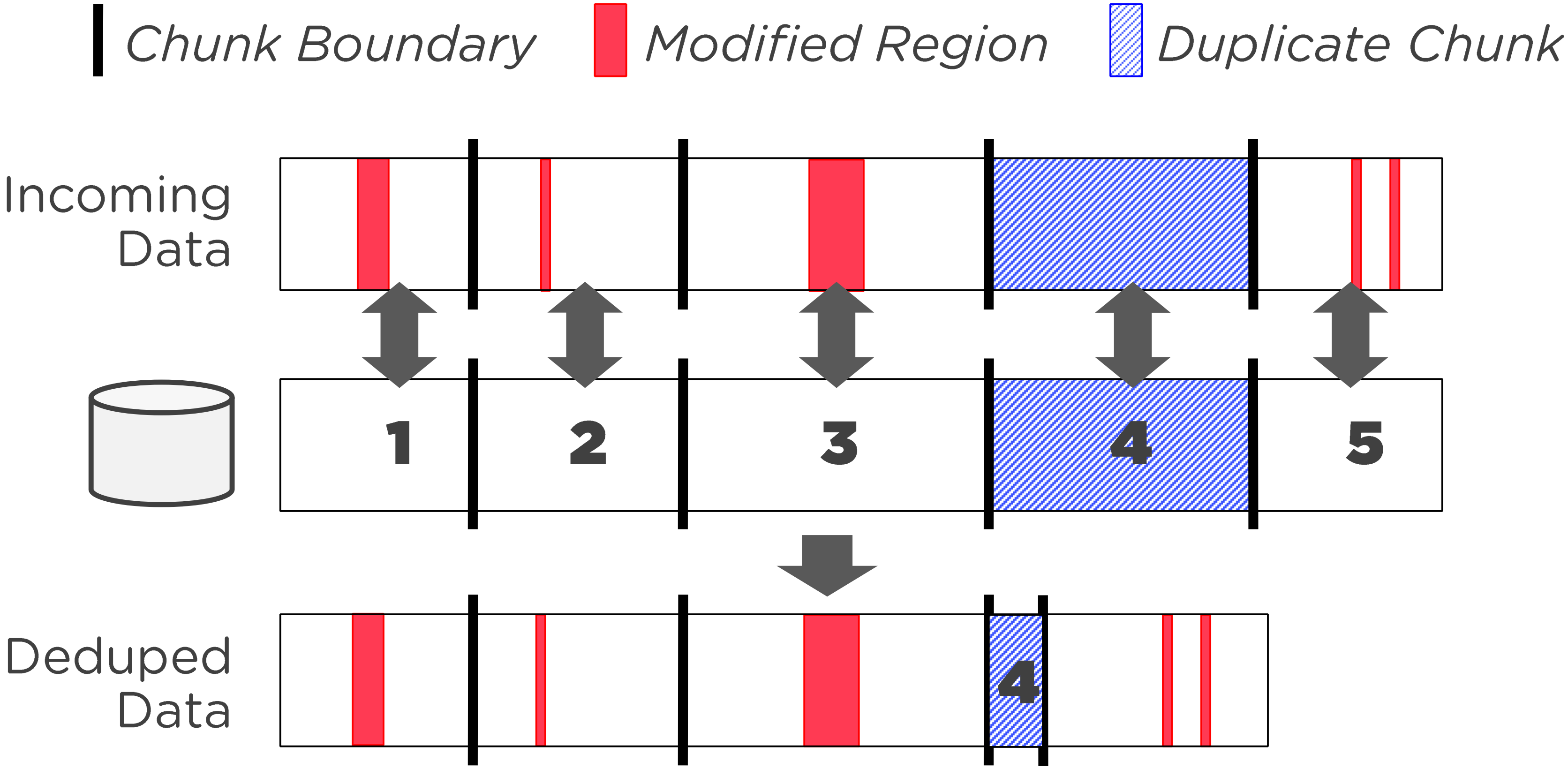
Traditional Dedup: Ideal Case



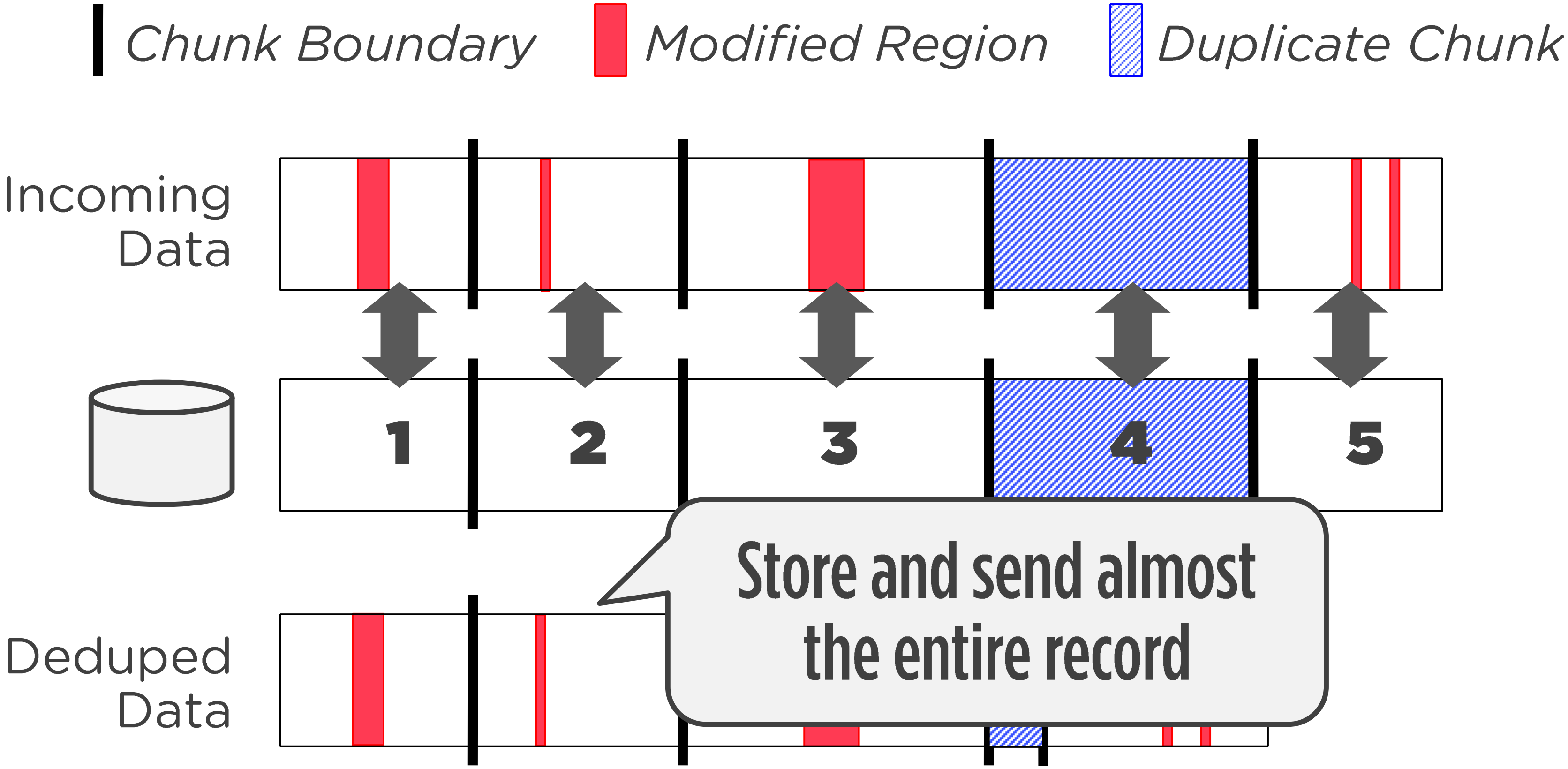
Dedup: Common Case in DBs



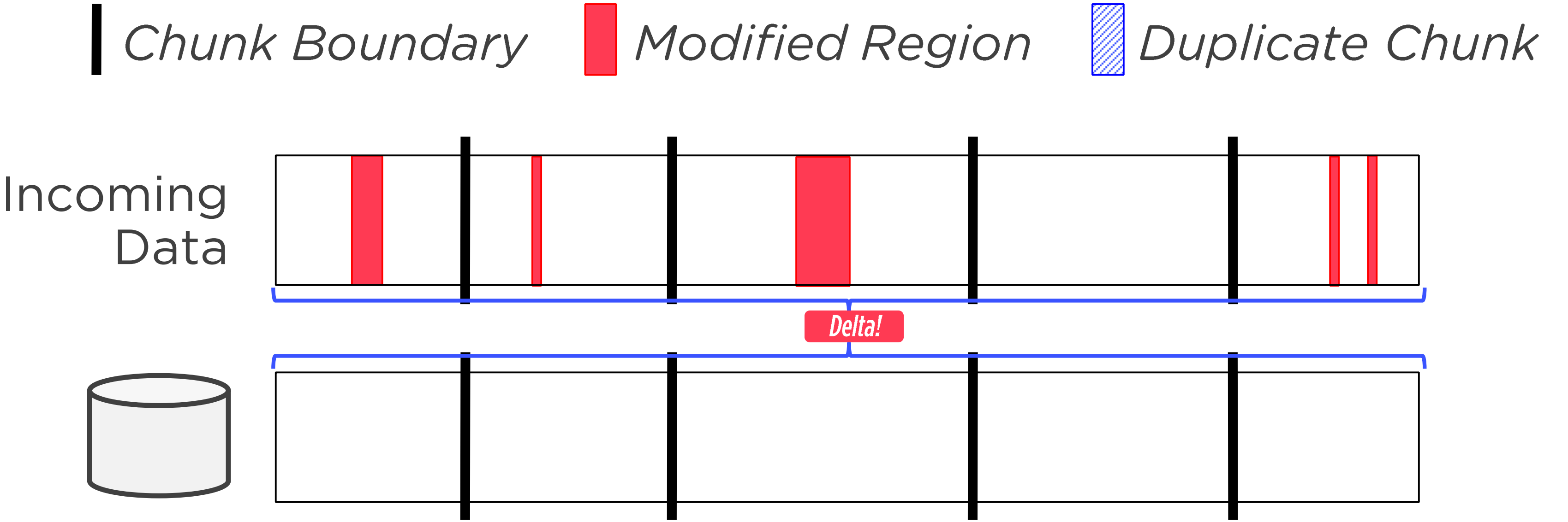
Dedup: Common Case in DBs



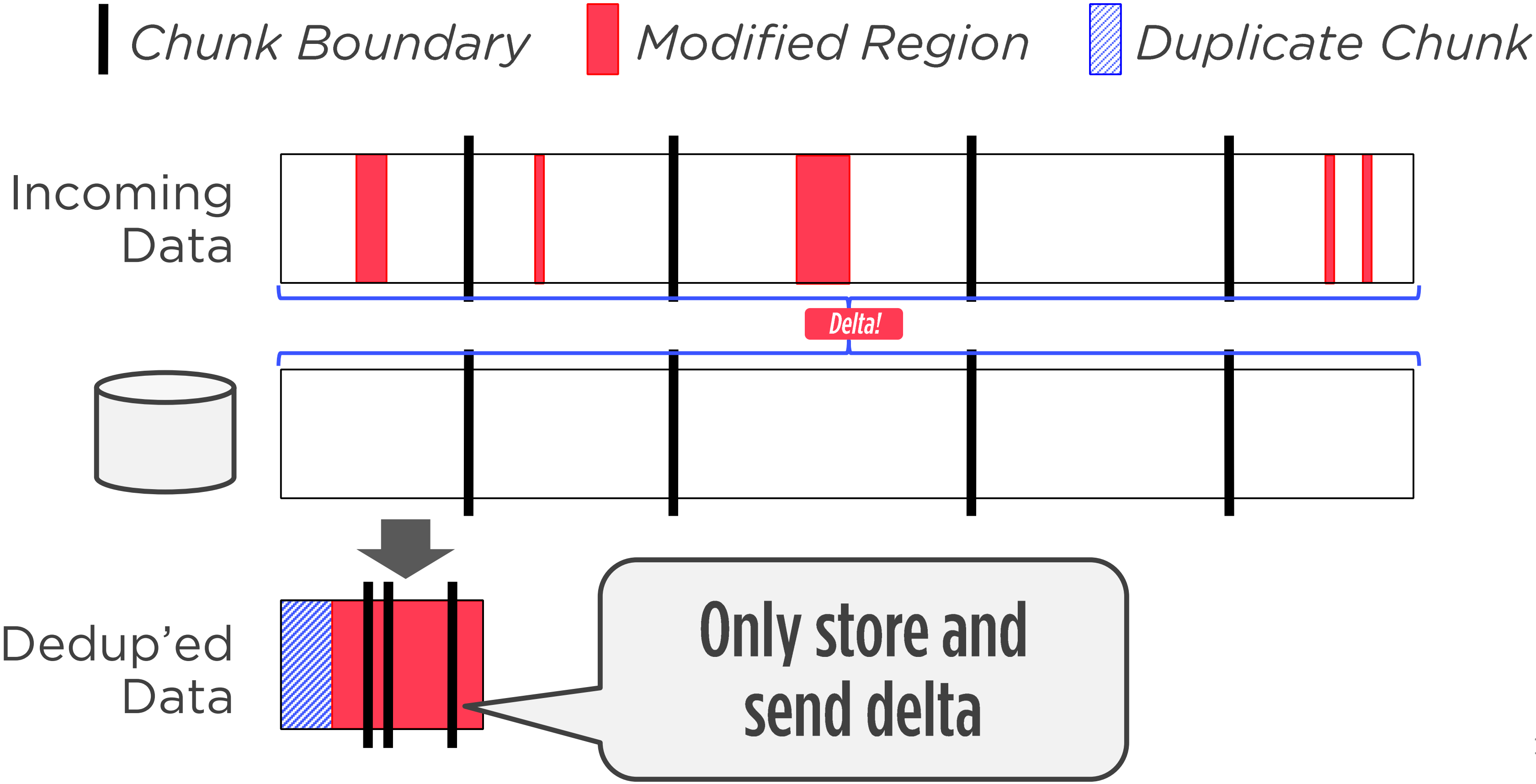
Dedup: Common Case in DBs



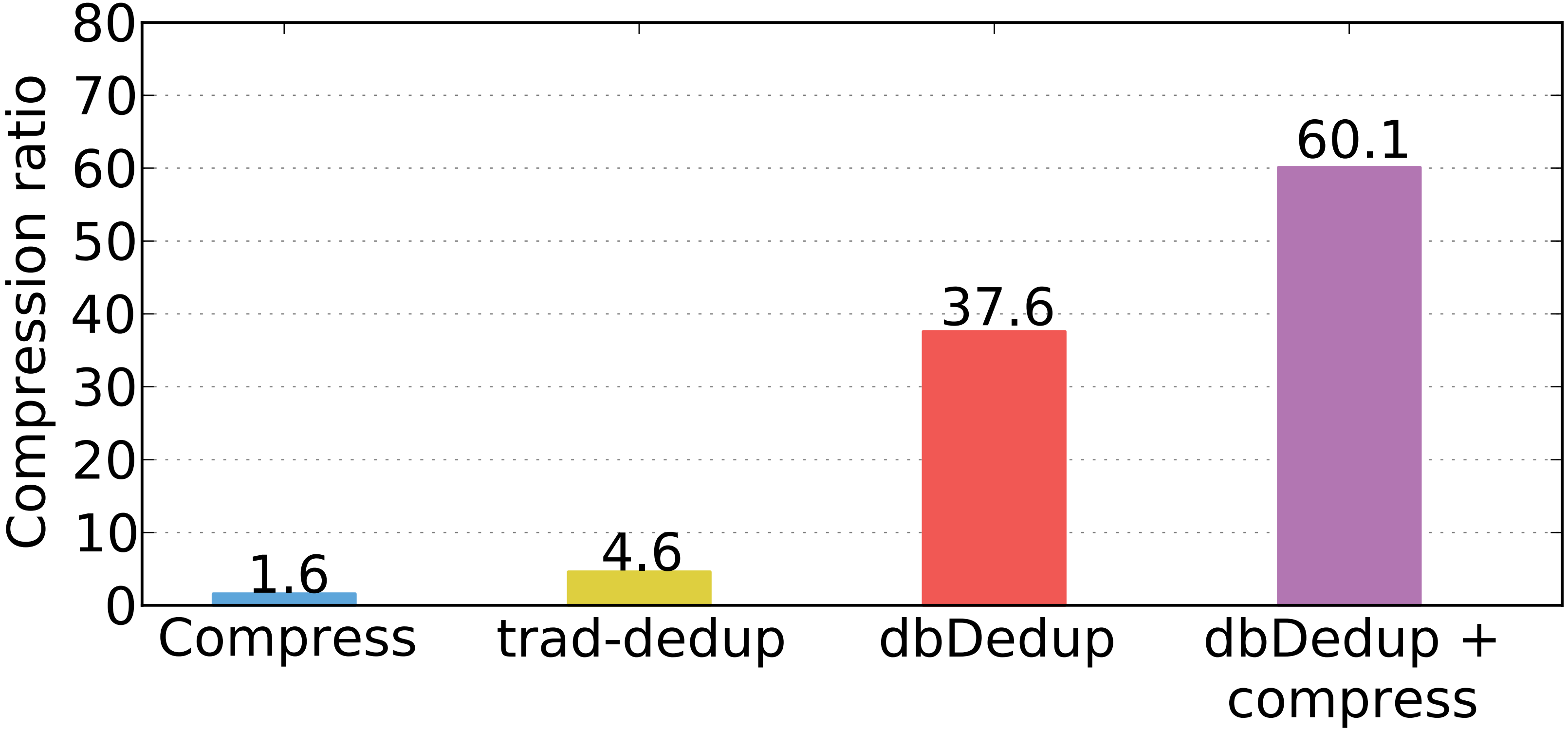
Similarity Dedup (dbDedup)



Similarity Dedup (dbDedup)



Compress vs. Dedup



Wikipedia Database (20GB sample)
MongoDB v3.1 w/ Snappy Compression

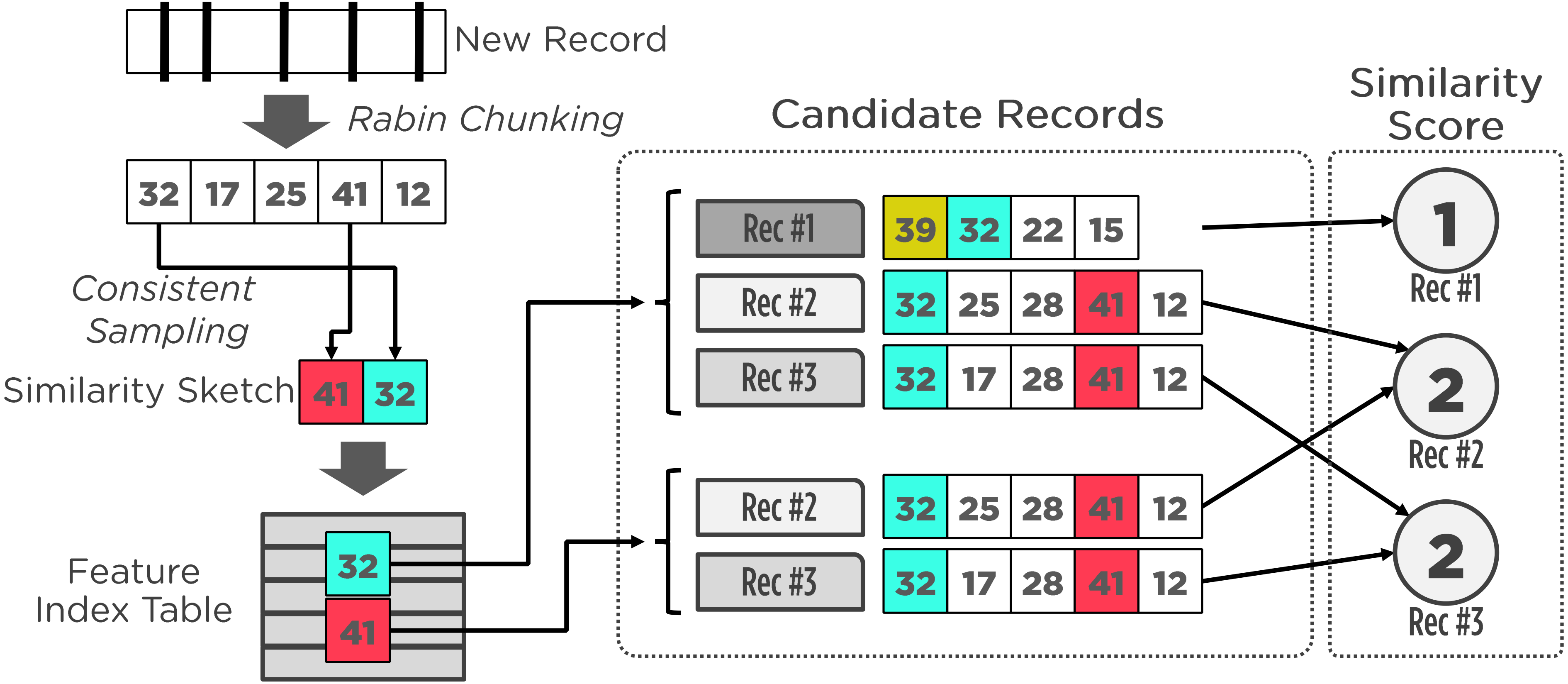
Outline

- Goal and Motivation
- Need for Similarity-based Dedup
- **Our Approach: dbDedup**
- Evaluation

dbDedup Encoding Steps

- For each new record:
  - *Identify Similar Records*
  - *Select the “Best” Match*
  - *Delta Compression*

Identify Similar Records

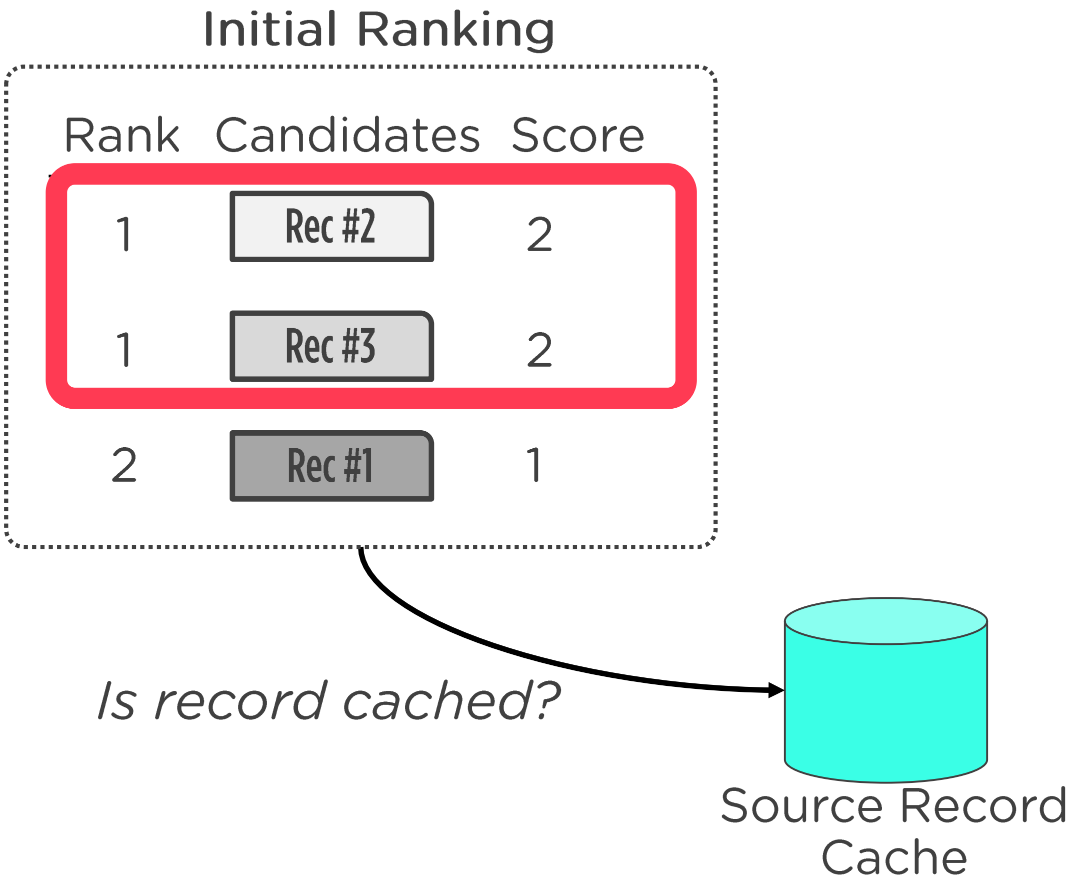


Select the Best Match

Initial Ranking

Rank	Candidates	Score
1	Rec #2	2
1	Rec #3	2
2	Rec #1	1

Select the Best Match



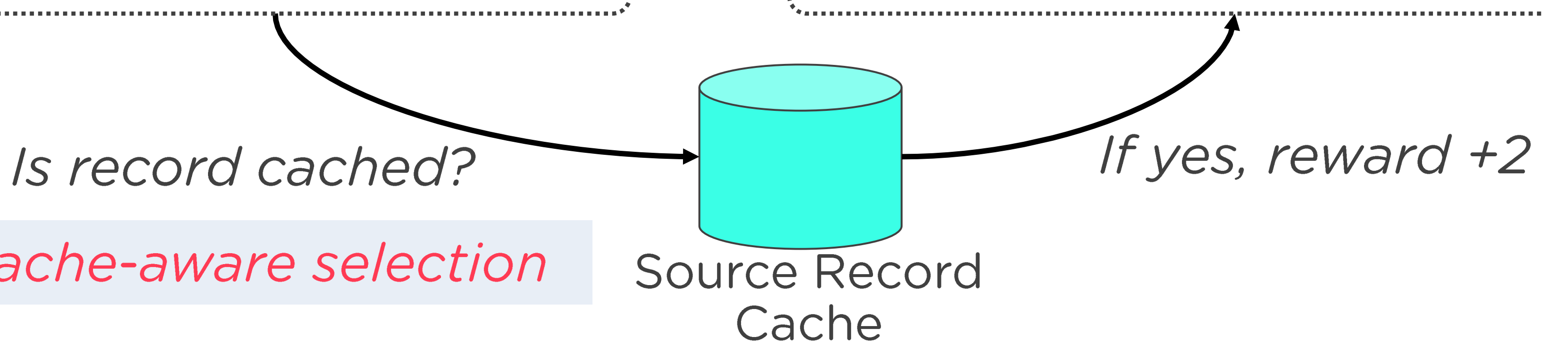
Select the Best Match

Initial Ranking

Rank	Candidates	Score
1	Rec #2	2
1	Rec #3	2
2	Rec #1	1

Final Ranking

Rank	Candidates	Cached?	Score
1	Rec #3	Yes	4
1	Rec #1	Yes	3
2	Rec #2	No	2



Cache-aware selection

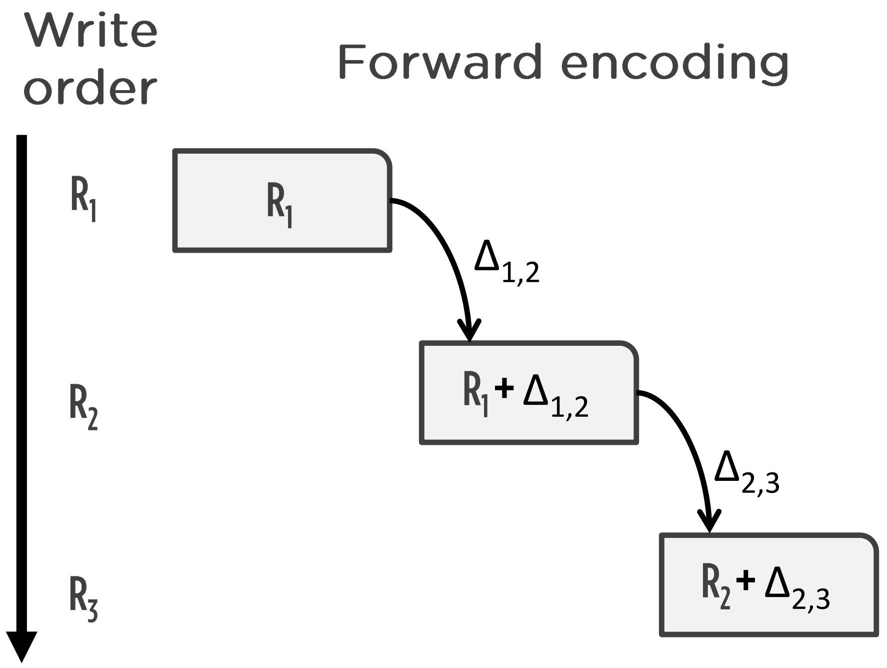
Delta Compression

- Based on the “xDelta” algorithm
  - *Byte-level diff between source and target records*
- Challenge:
  - *Reading encoded records incurs decoding overhead*
- *Solution: new encoding techniques*
  - *Reduce both freq. and cost of decode*

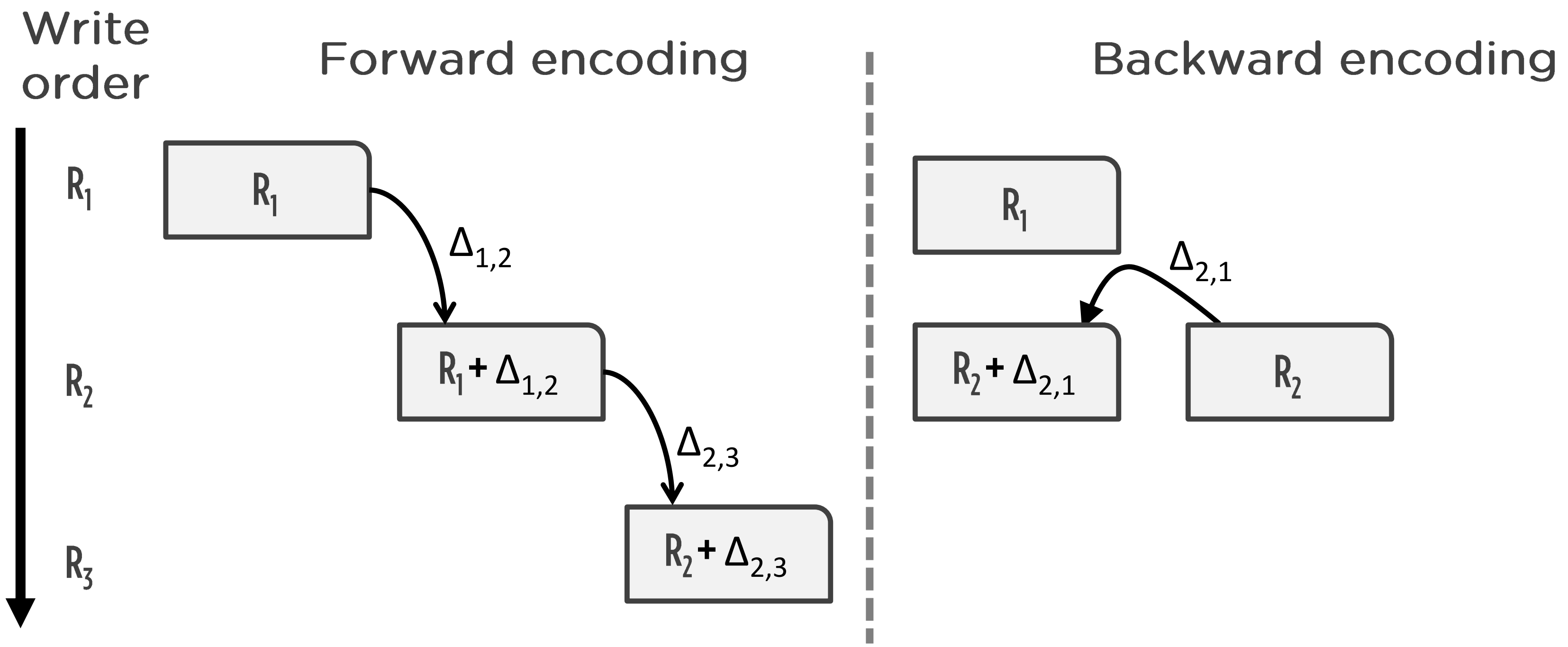
New Encoding Techniques

- Two-way encoding
  - *Forward encoding for network-level dedup*
    - Sending encoding of new records against source
    - Replicas need to use existing record as source
  - *Backward encoding for storage-level dedup*
    - Encode/rewrite source; write unencoded new record
    - Minimize read overhead for the most recent records
- Hop encoding
  - *Reduce worst-case re-construction time*
  - *Maintain compression ratio*

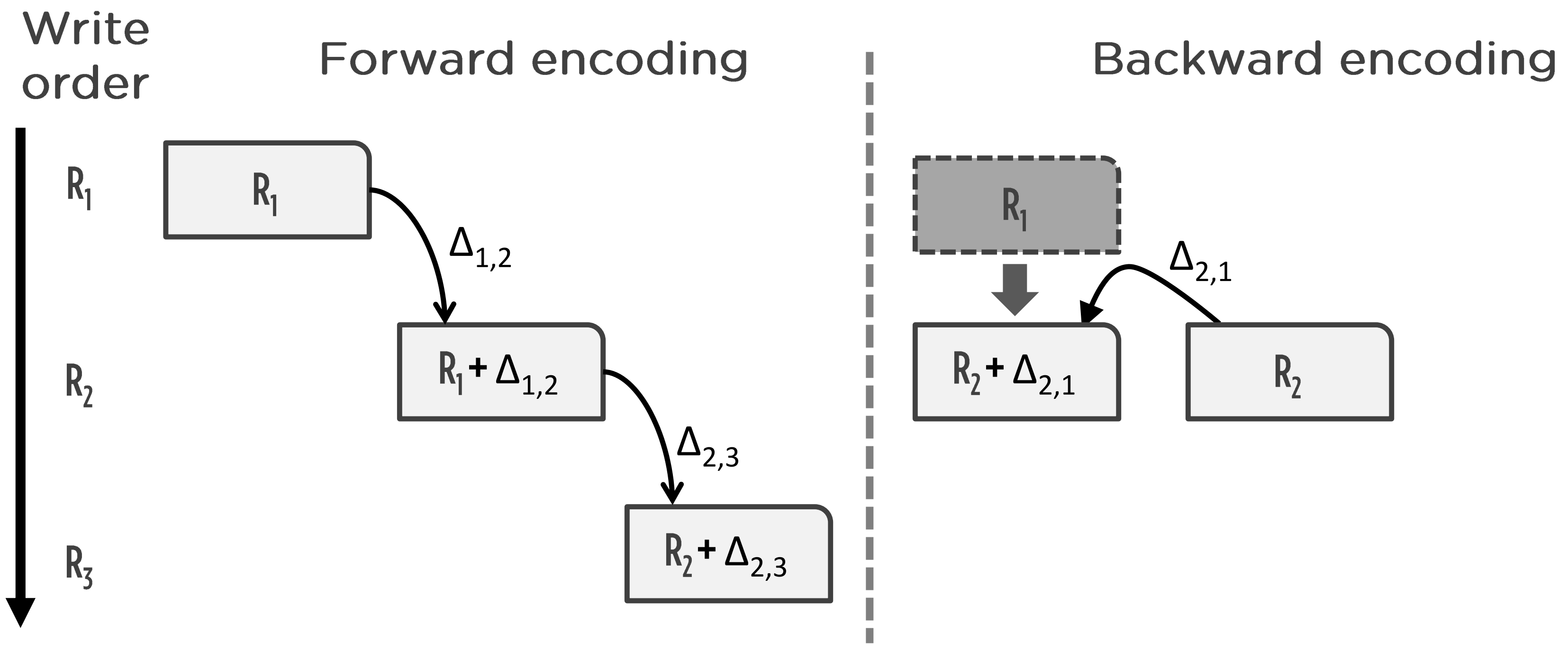
Two-way Encoding



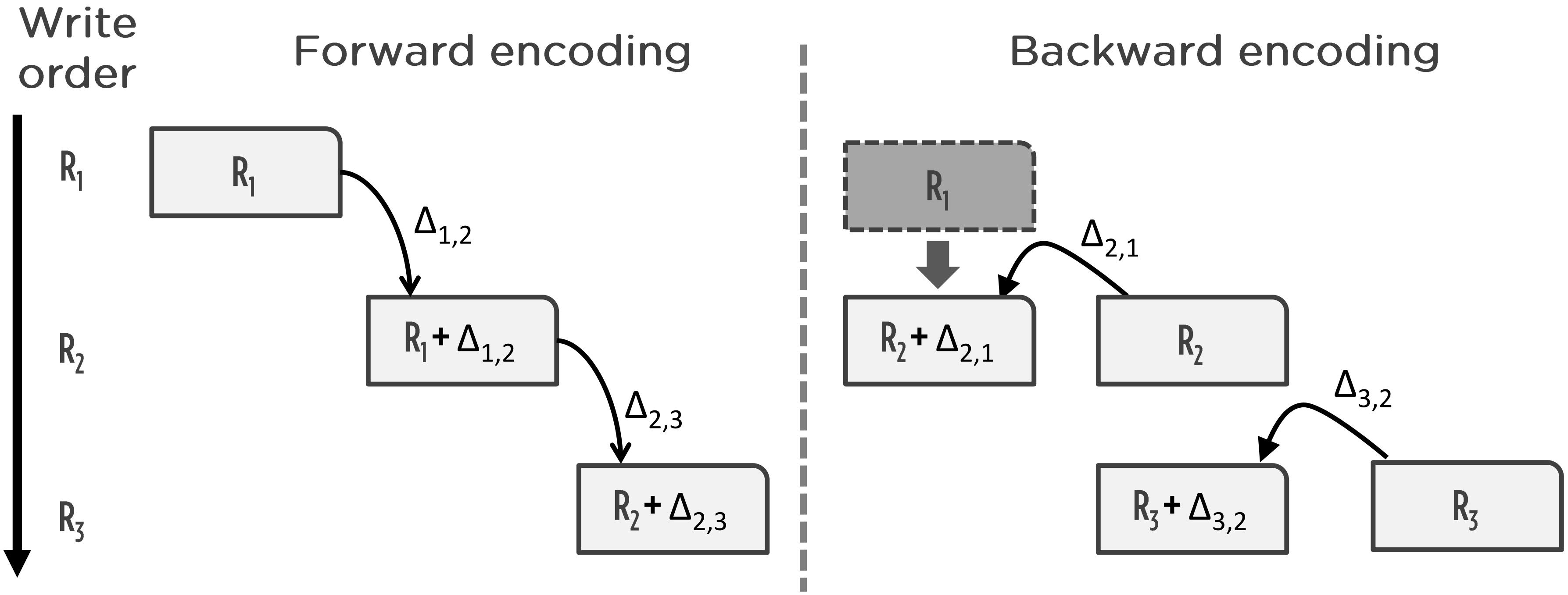
Two-way Encoding



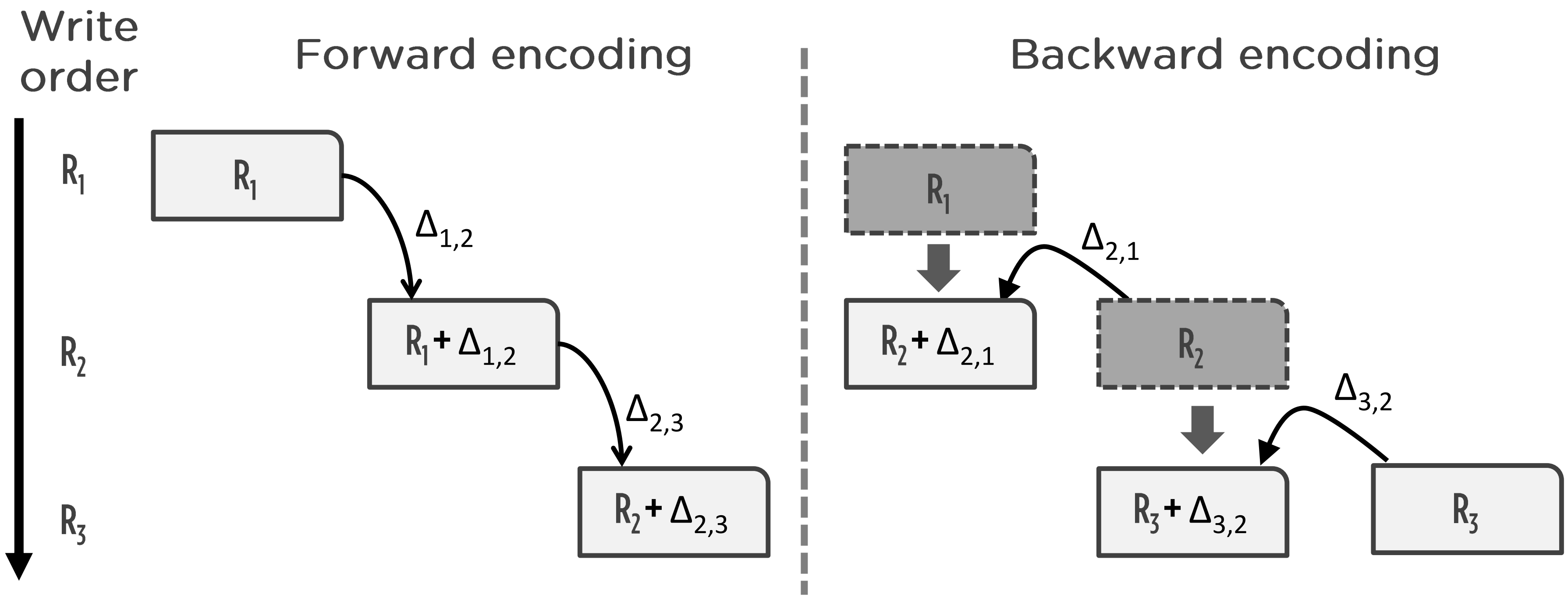
Two-way Encoding



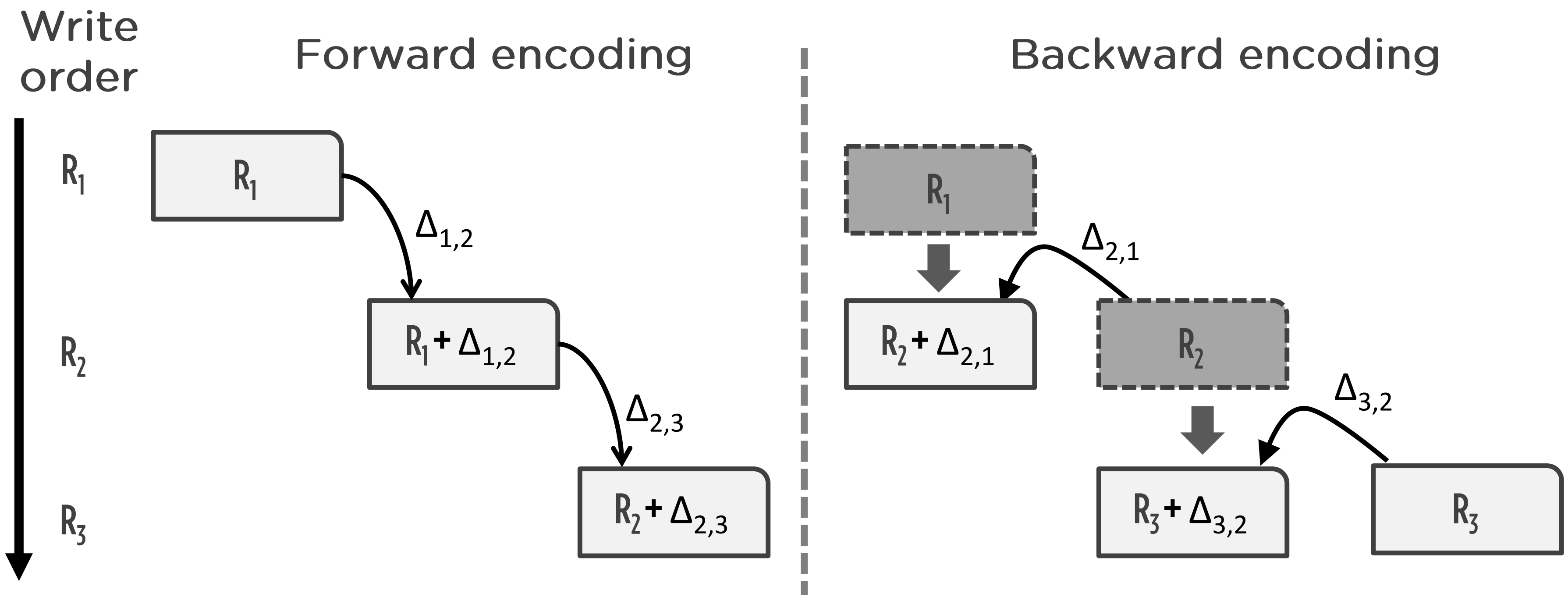
Two-way Encoding



Two-way Encoding

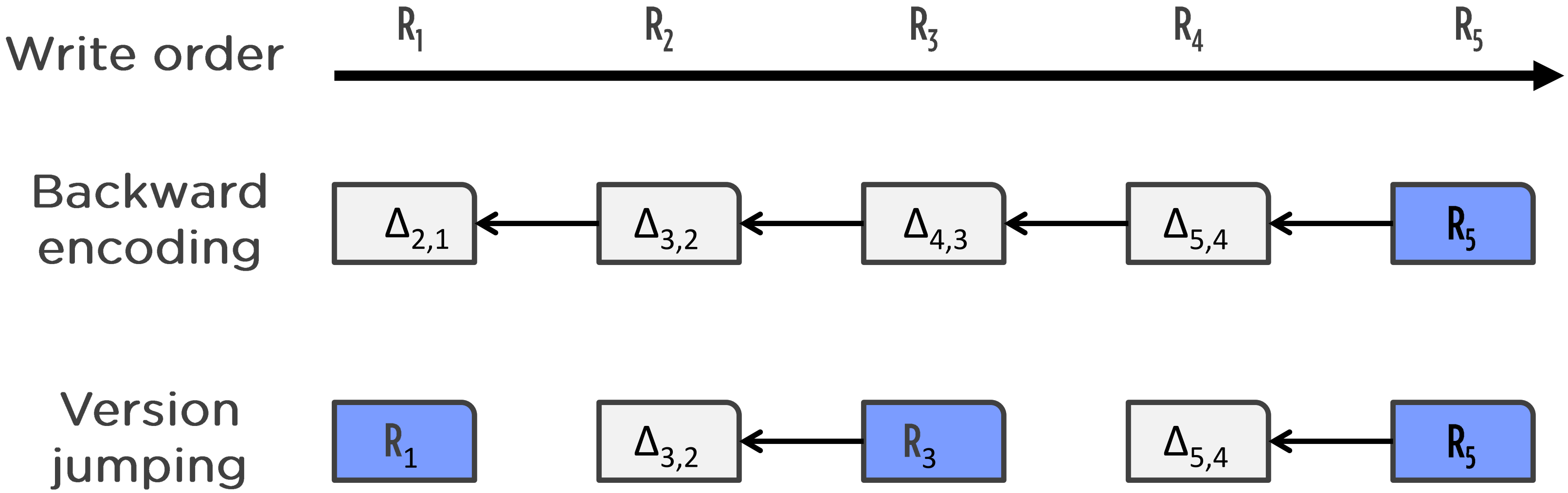


Two-way Encoding

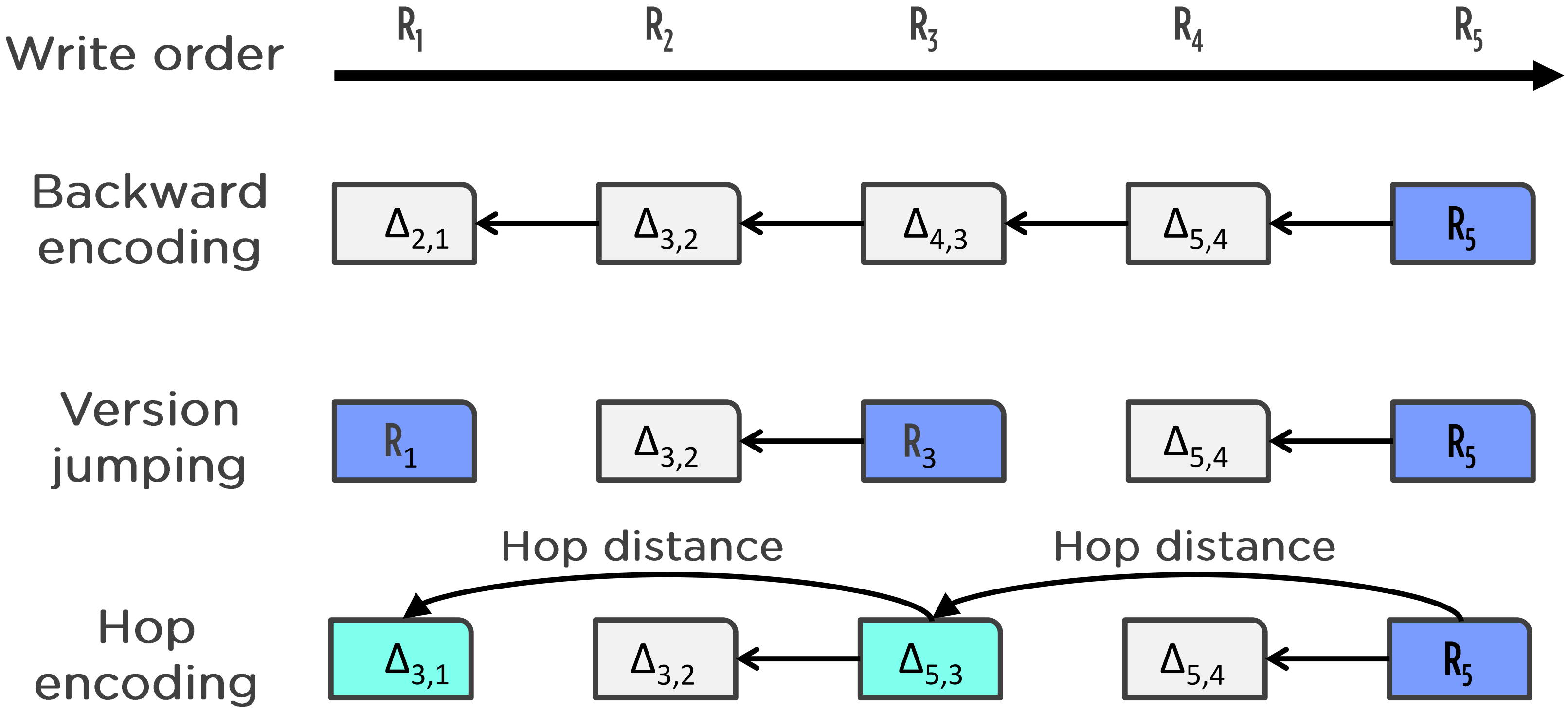


Network-level dedup **Storage-level dedup**

Hop Encoding



Hop Encoding



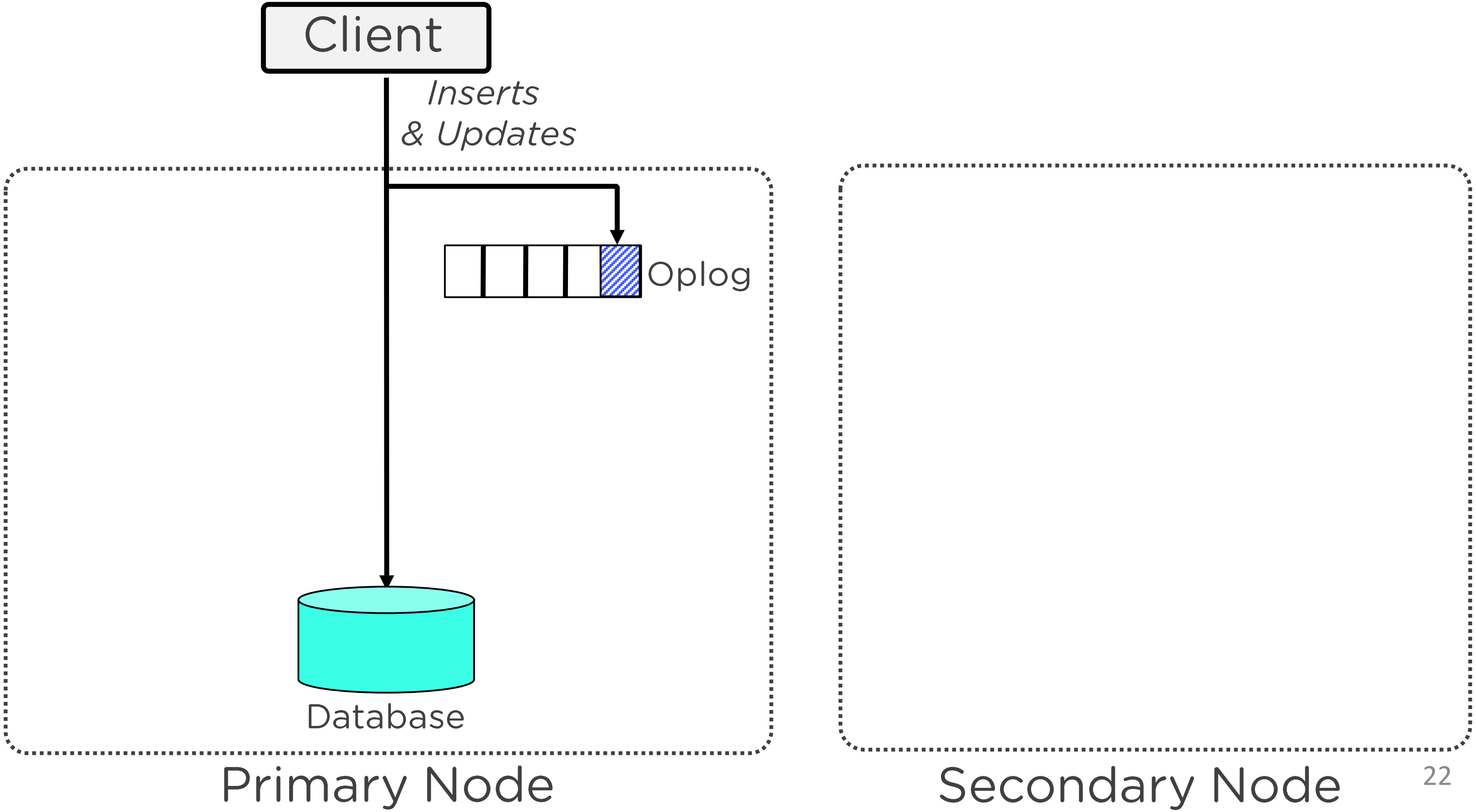
Caching Mechanisms

- Unique challenge for delta compression
  - *Extra disk I/Os to read/update records*
- Solution: specialized caches
  - *Source Record cache*
    - Reduce disk reads to source records
    - Keep only the latest version of a record
  - *Lossy write-back cache*

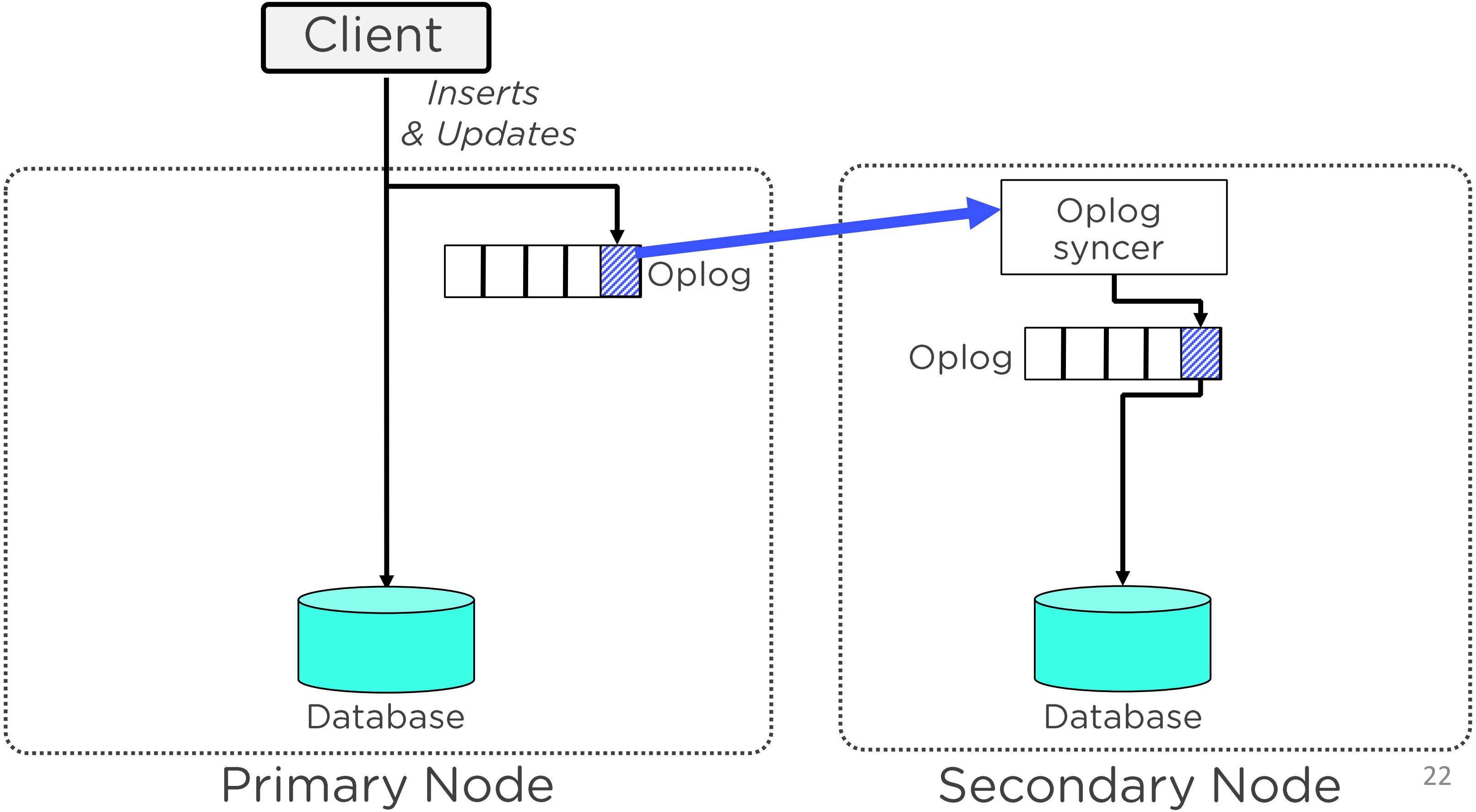
Lossy Write-back Cache

- Reduce overhead of writing backward-encoded sources
  - *Delay updates until system is idle*
  - *Prioritize updates based on space savings*
- “Lossy” property
  - *Unapplied updates can be safely discarded*
  - *Records remain un-encoded in such case*

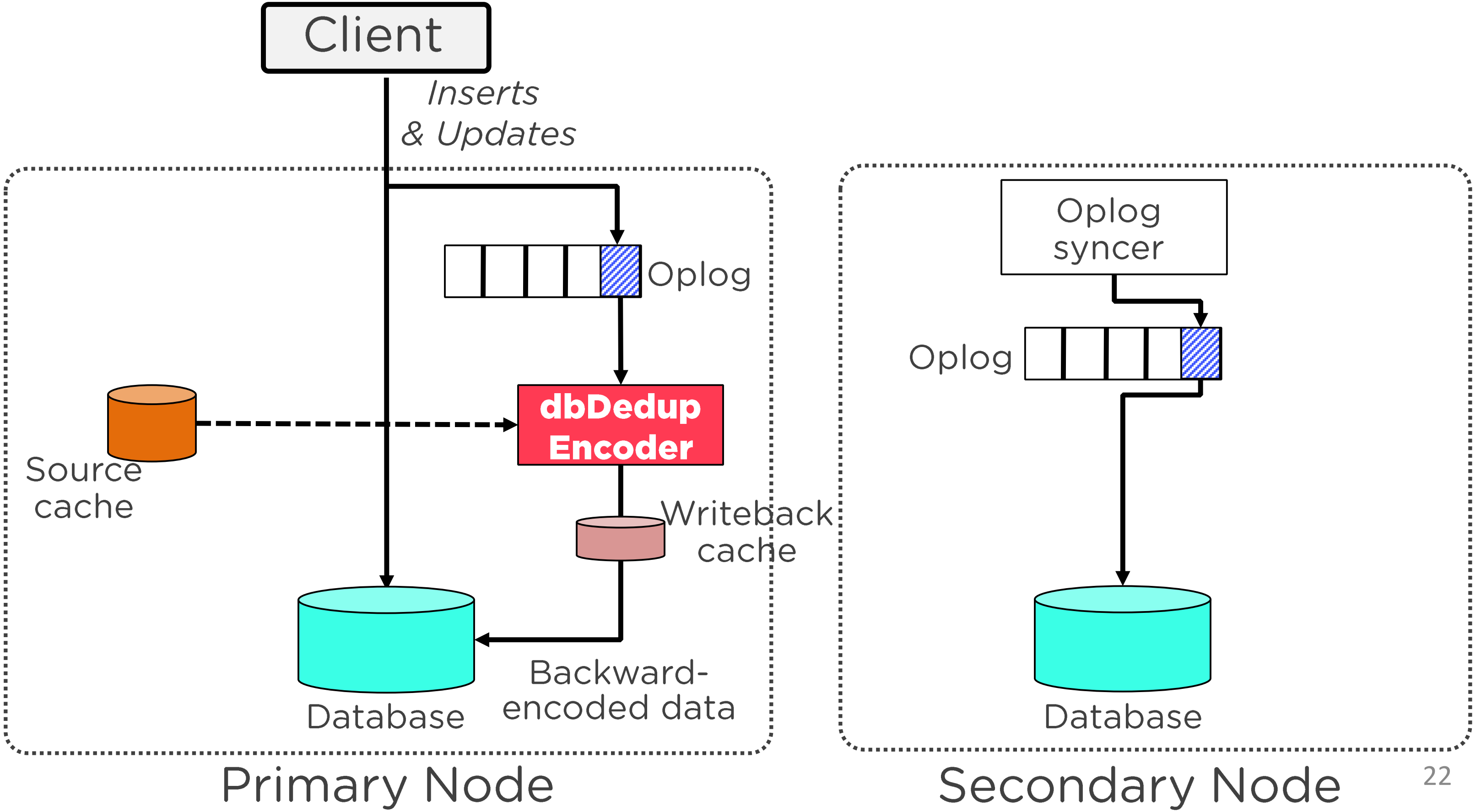
Integration into DBMS



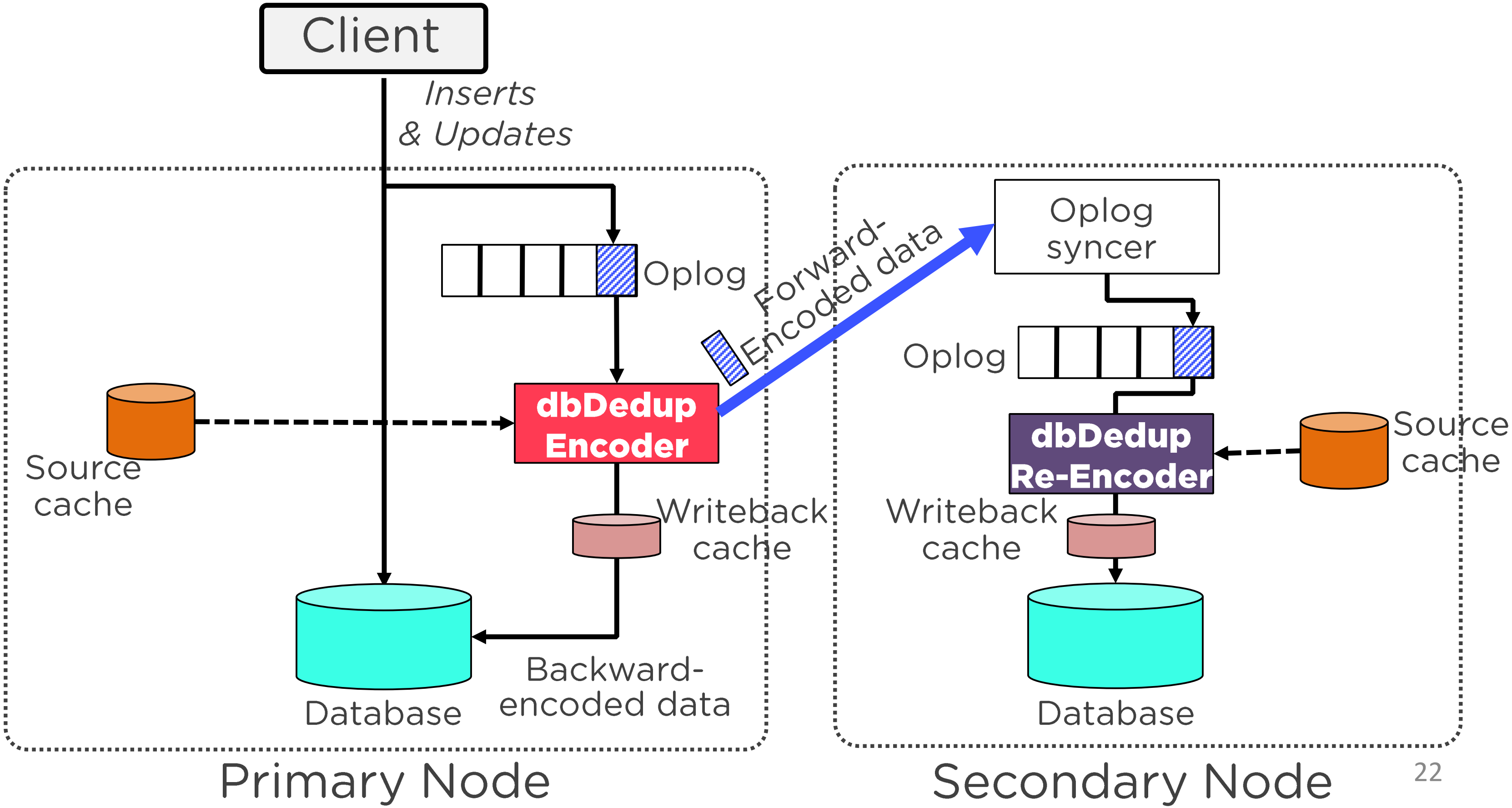
Integration into DBMS



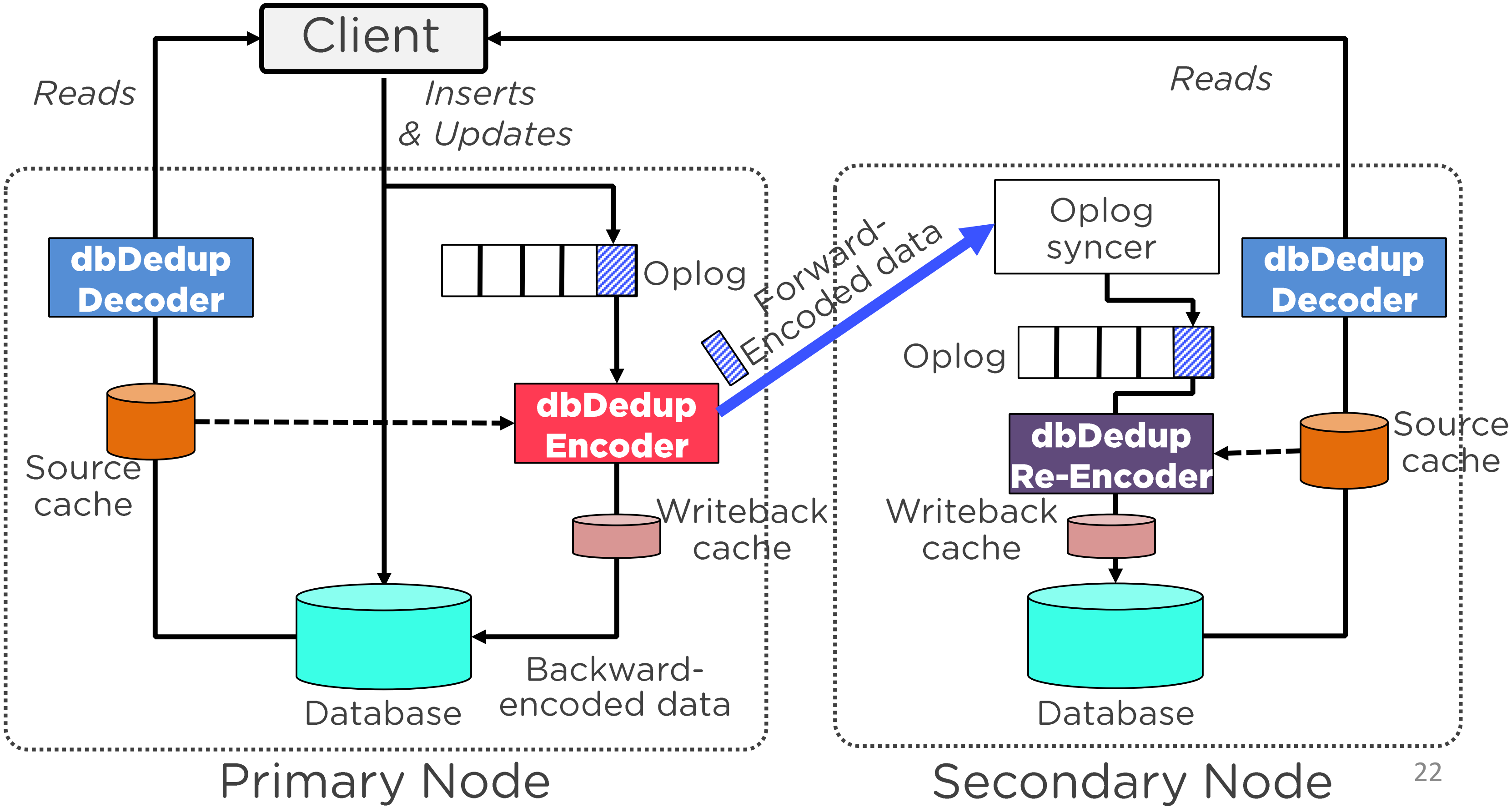
Integration into DBMS



Integration into DBMS



Integration into DBMS



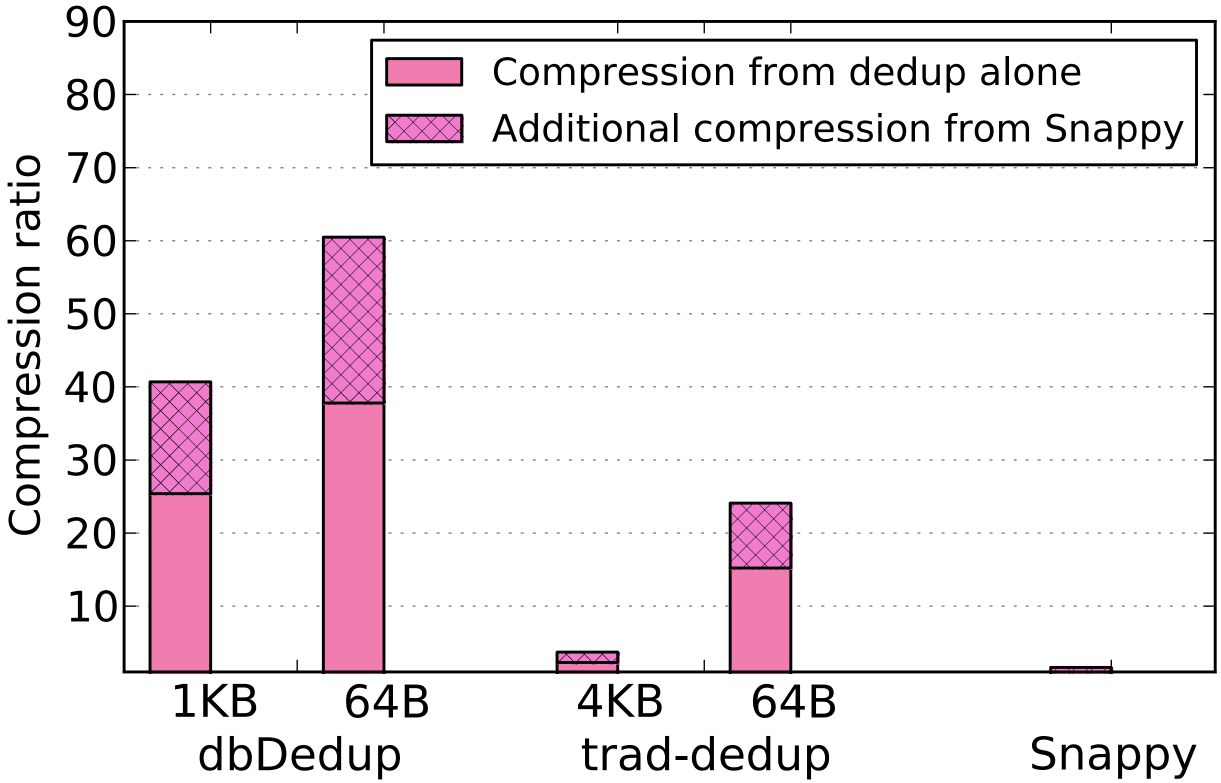
Outline

- Goal and Motivation
- Need for Similarity-based Dedup
- Our Approach: dbDedup
- **Evaluation**

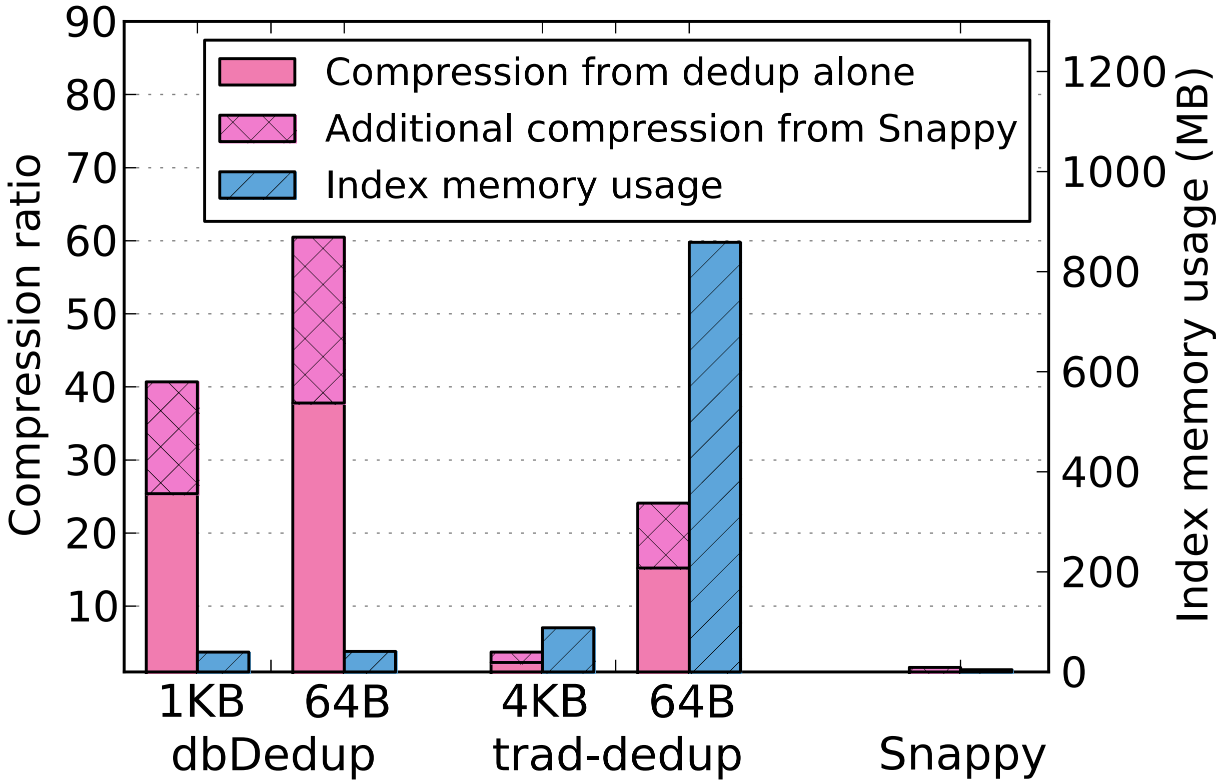
Evaluation

- MongoDB (v3.1)
  - *1 primary, 1 secondary node, 1 client*
  - *Node Config: 4 cores, 8GB RAM, 100GB HDD storage*
- Datasets:
  - *Wikipedia dump (20GB out of ~12TB)*
  - *Additional datasets evaluated in the dissertation*

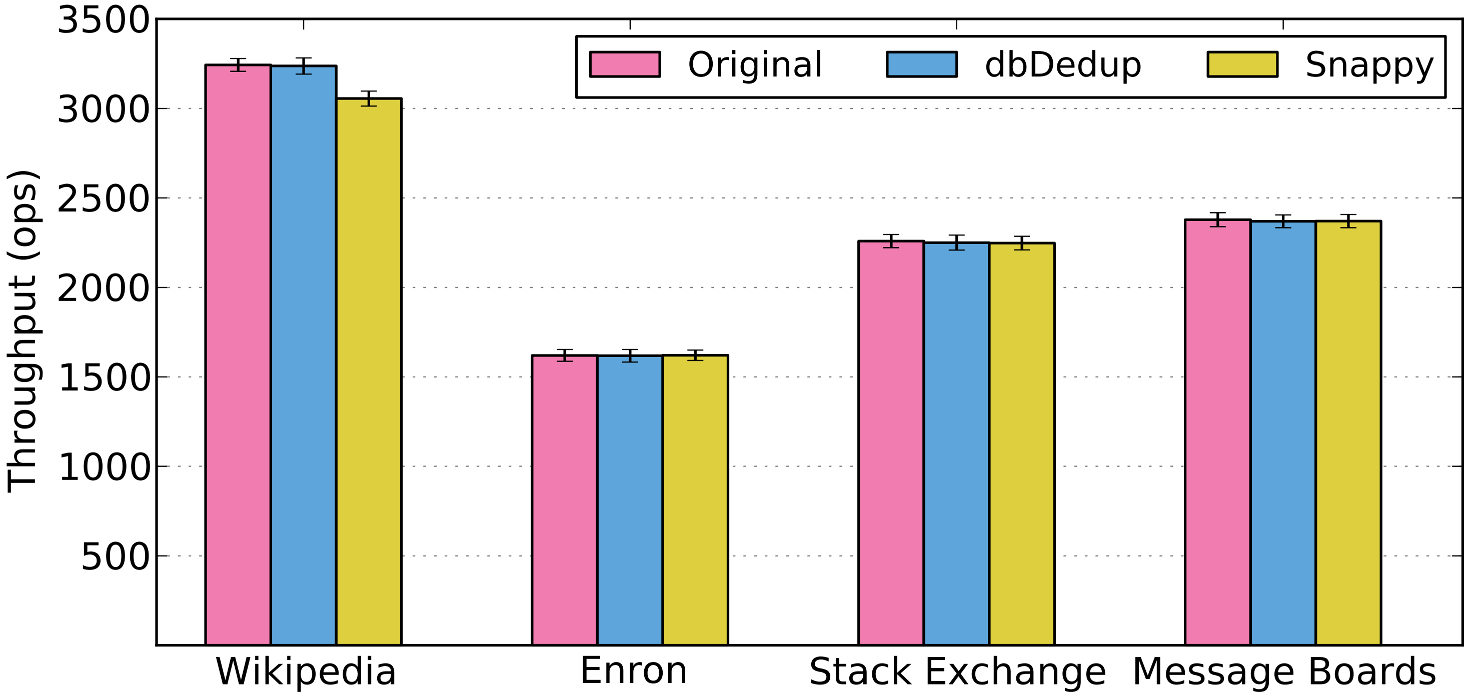
Compression Ratio



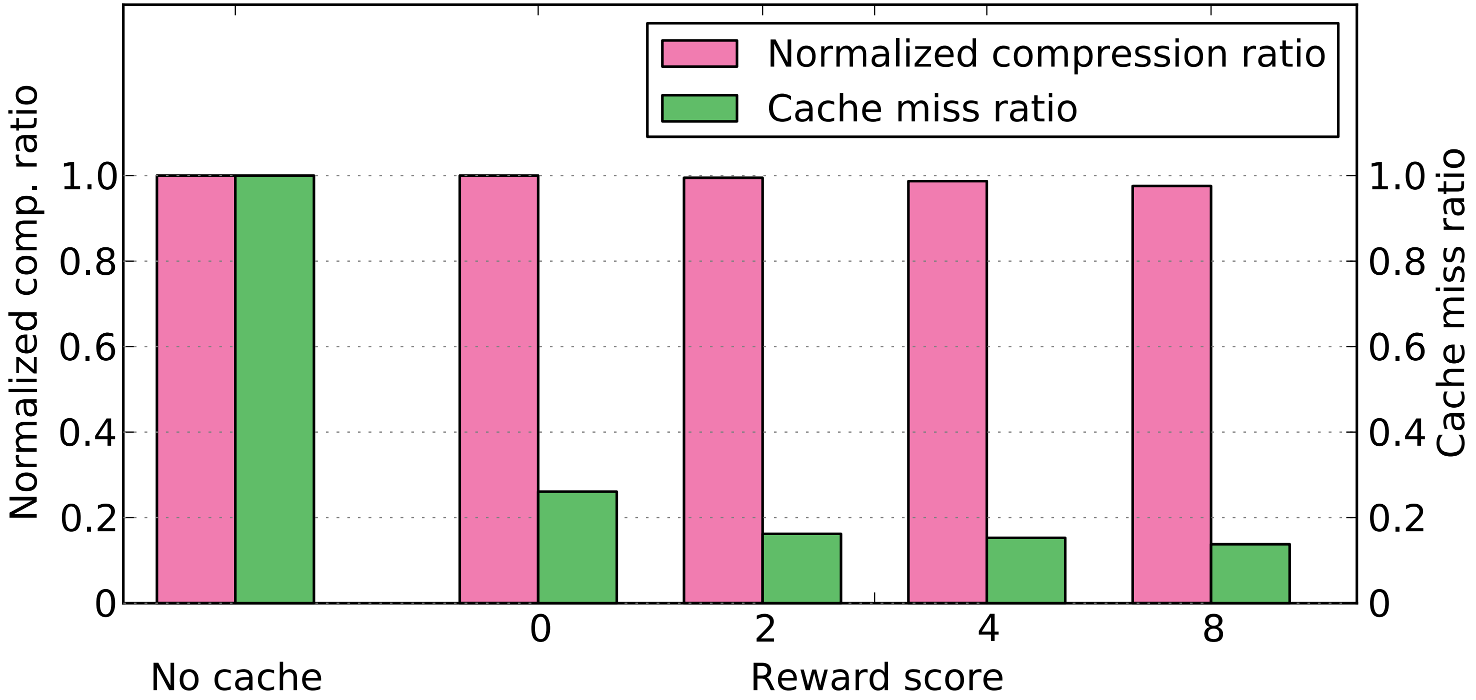
Comp. Ratio & Index Memory



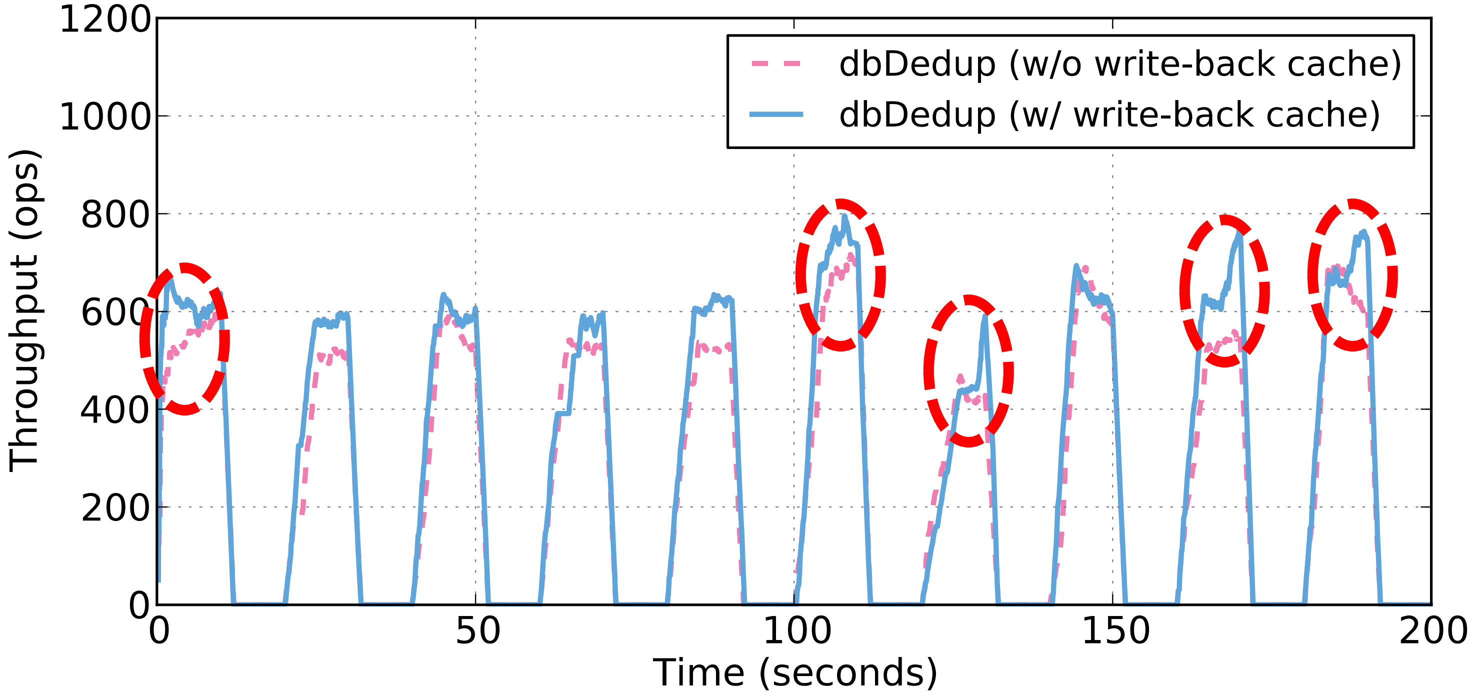
Throughput Overhead



Effect of Source Cache



Effect of Lossy Write-back



Conclusion

- **dbDedup**: Similarity-based dedup for DBMSs
  - *First DB dedup system for both storage and network layers*
  - *Much greater data reduction than traditional dedup*
  - *Up to 38x compression ratio for Wikipedia*
  - *Resource-efficient design with negligible overhead*