Efficient and Oblivious Query Processing for Range and kNN Queries

(Extended Abstract)

Zhao Chang[1], Dong Xie[2], Feifei Li[3], Jeff M. Phillips[4], Rajeev Balasubramonian[4]

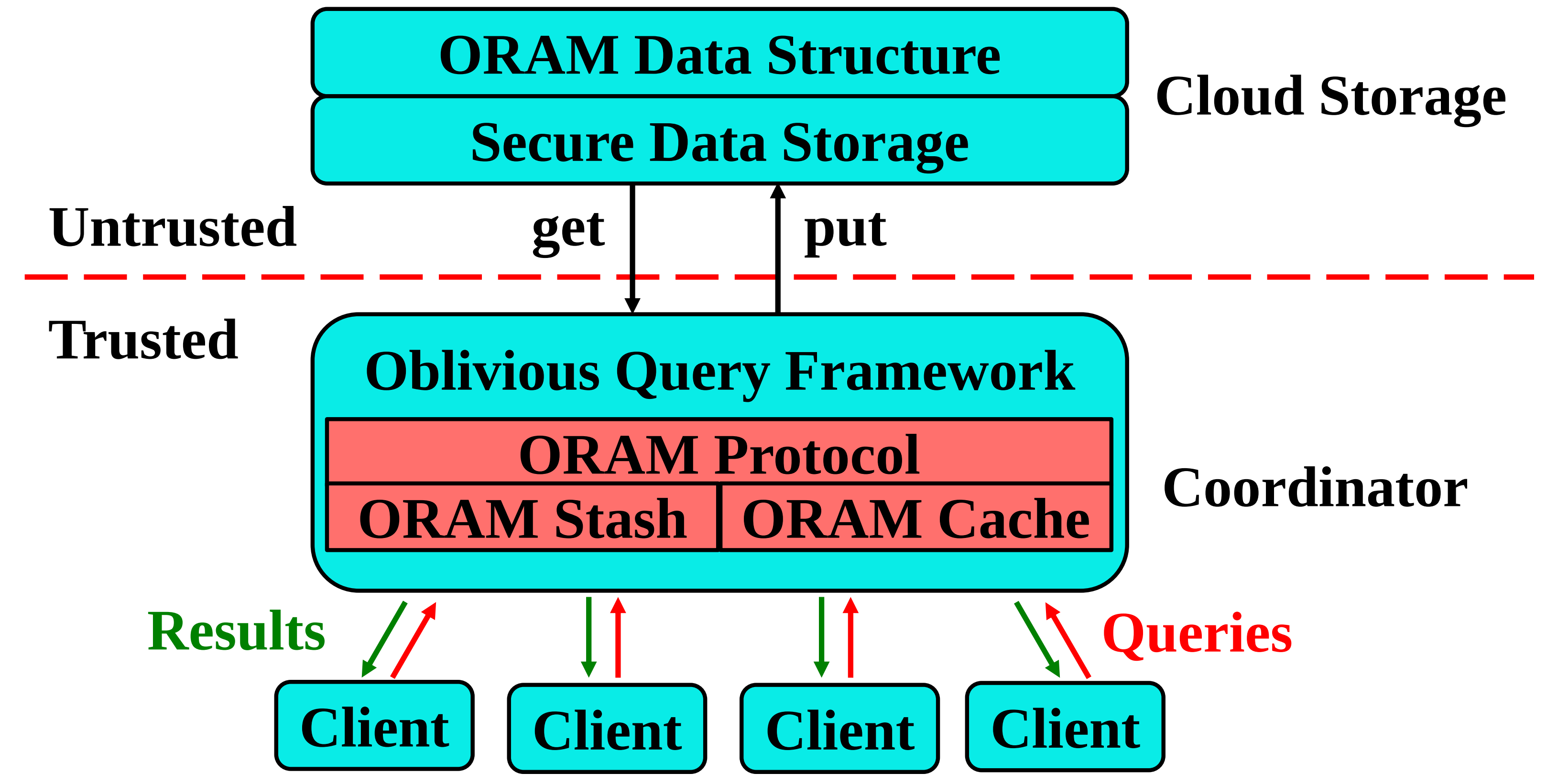
[1]Xidian University, [2]The Pennsylvania State University, [3]Alibaba Group, [4]University of Utah

INTRODUCTION

OQF: Oblivious Query Processing Framework

- Background: protecting data access patterns in cloud
- Problem: range and *k*NN queries
- Objective: maximizing the overall query throughput

FRAMEWORK



ORAM Data Structure

Secure Data Storage

Cloud Storage

Untrusted get put

Trusted

Oblivious Query Framework

ORAM Protocol

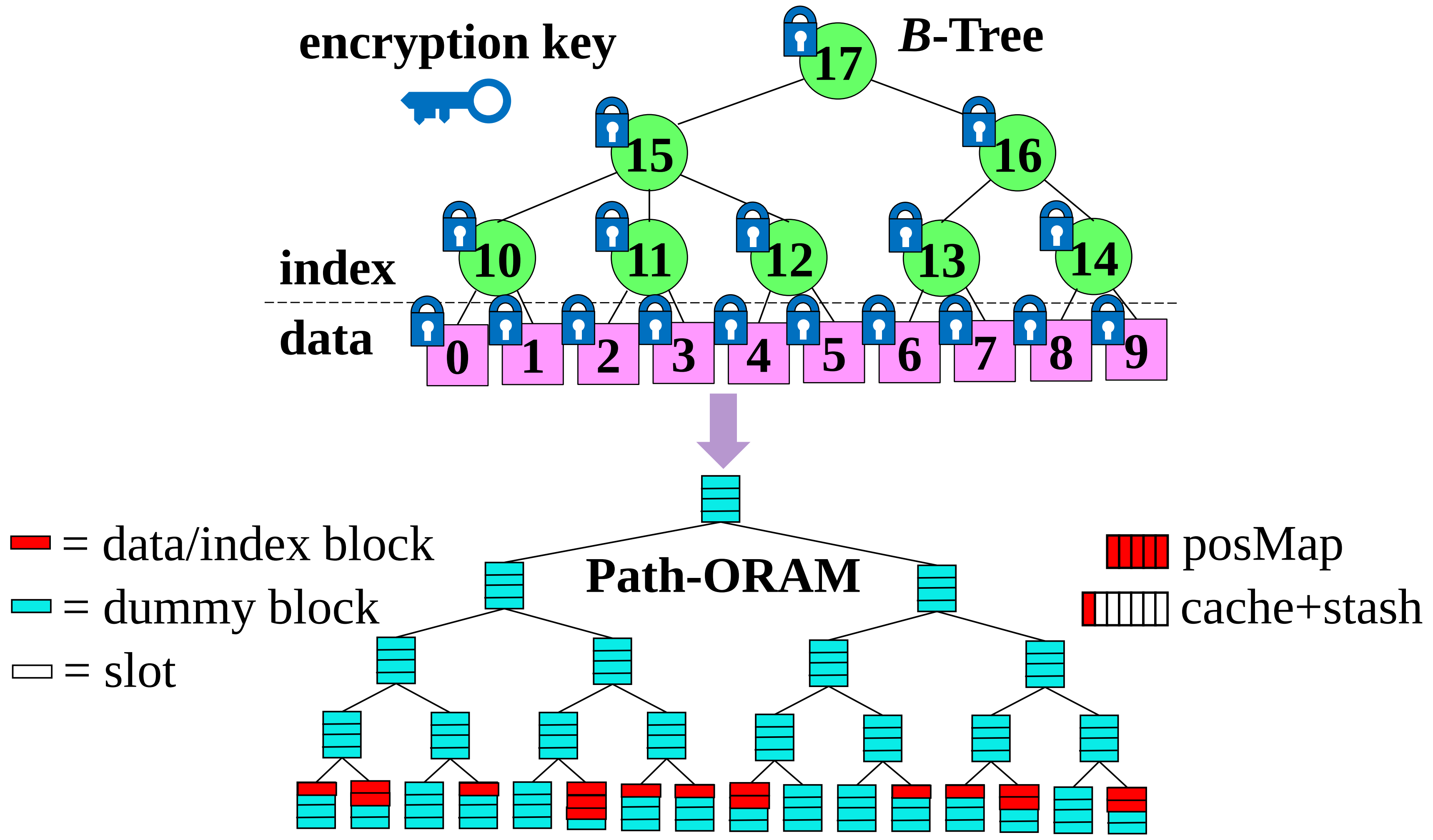
ORAM Stash | ORAM Cache

Coordinator

Results Queries

Client Client Client Client

INTEGRATING INDEX INTO ORAM



encryption key

B-Tree

index

data

Path-ORAM

🟥 = data/index block

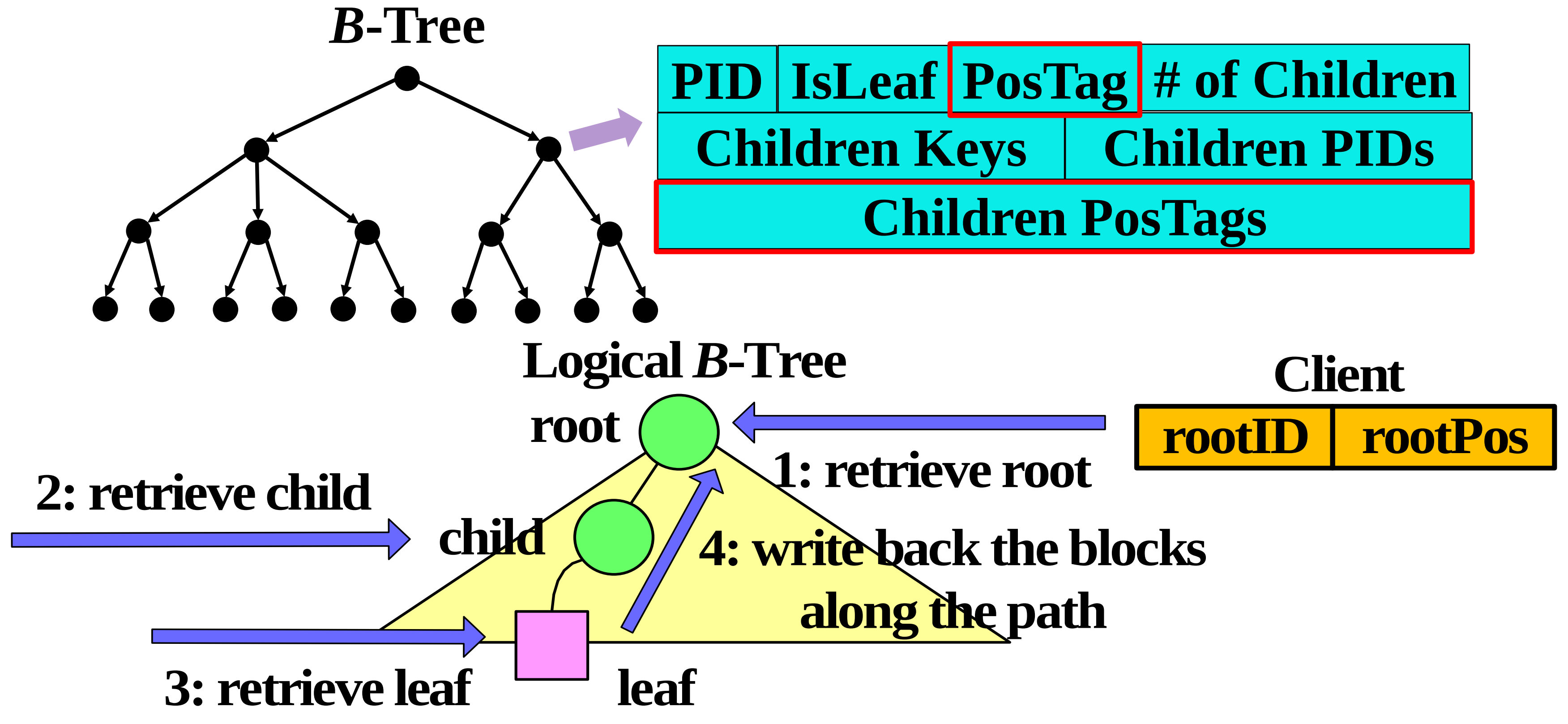
▭ = dummy block

▭ = slot

▥ posMap

▥ cache+stash

OBLIVIOUS INDEX



B-Tree

PID	IsLeaf	PosTag	# of Children
Children Keys		Children PIDs	
Children PosTags			

Logical *B*-Tree

root

Client

rootID	rootPos

1: retrieve root

2: retrieve child

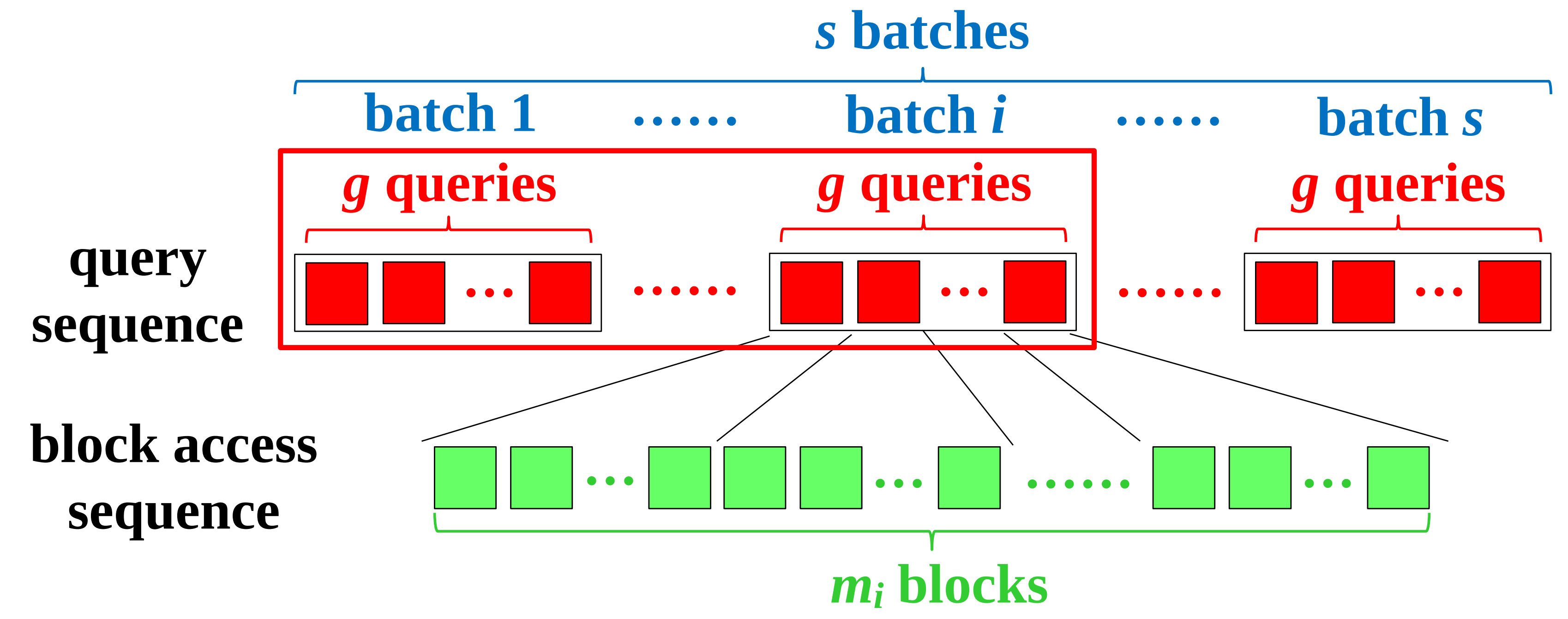
child

4: write back the blocks along the path

3: retrieve leaf

leaf

OPTIMIZATIONS



s batches

batch 1 …… batch *i* …… batch *s*

g queries *g* queries *g* queries

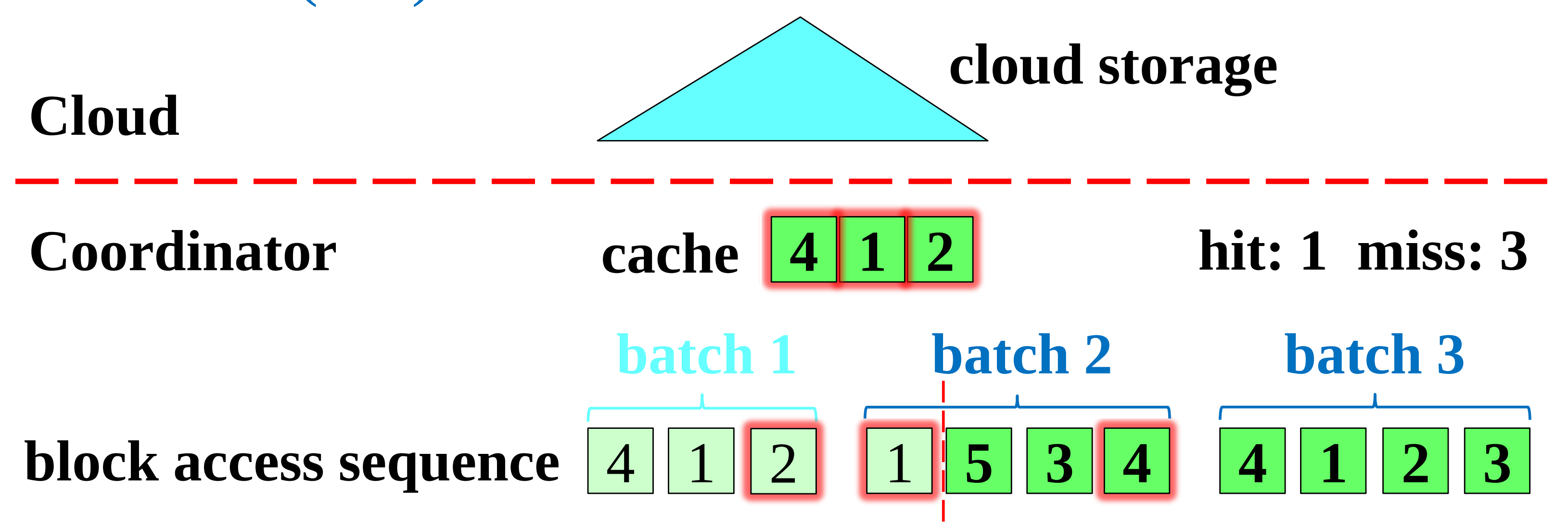
query sequence

block access sequence

m_i blocks

Caching Strategy: batch-FIF (LRU+FIF)

- (1) evict the block that will not be accessed within current batch using LRU strategy
- (2) evict the block that will not be accessed until farthest in future (FIF) within current batch



Cloud

cloud storage

Coordinator

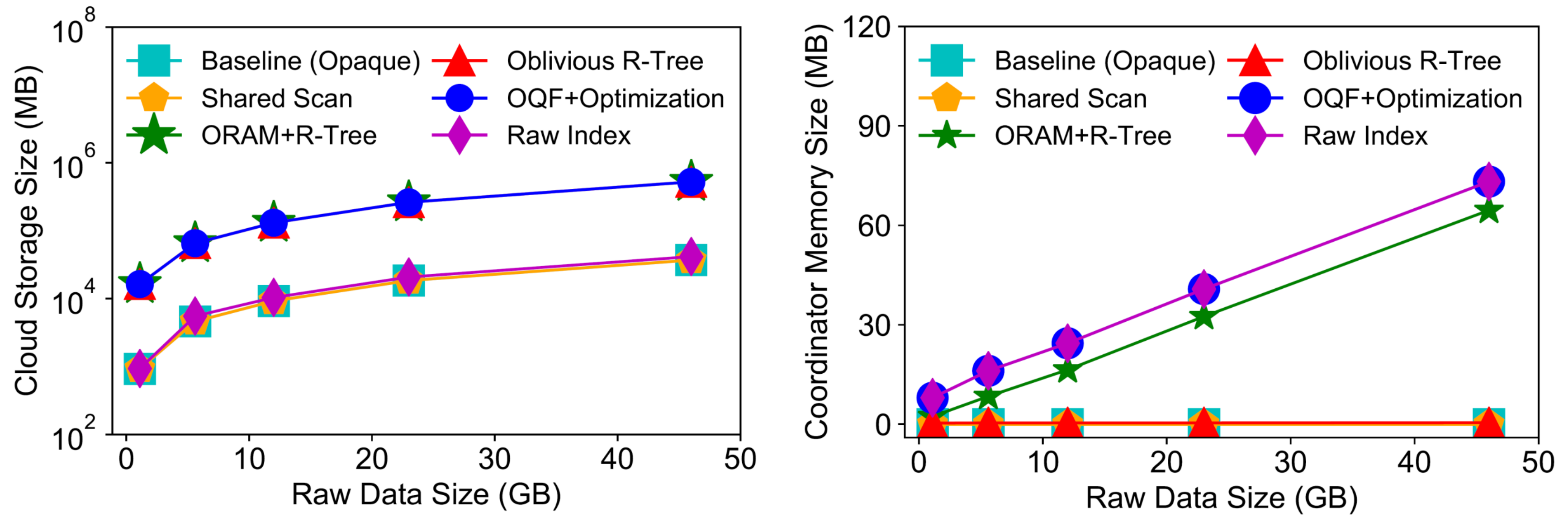
cache 4 1 2

hit: 1 miss: 3

batch 1 batch 2 batch 3

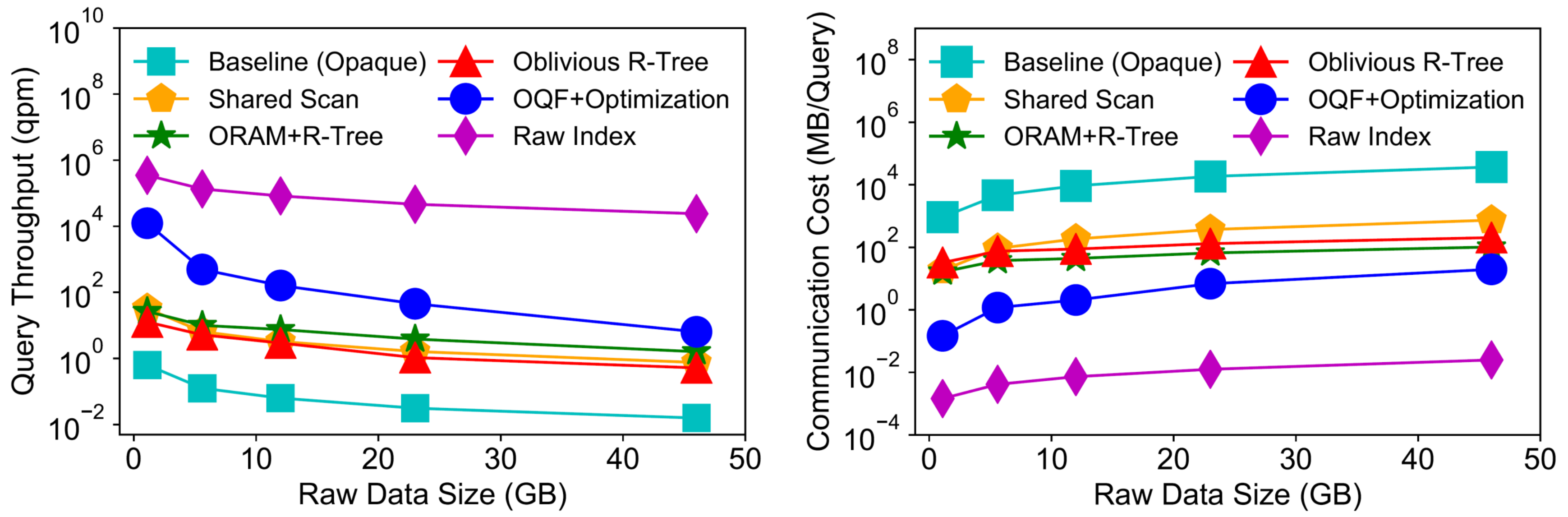
block access sequence 4 1 2 | 1 5 3 4 | 4 1 2 3

EXPERIMENTAL EVALUATION



(a) cloud storage size. (b) coordinator memory size.

Figure 1: Storage cost against raw data size.



(a) query throughput. (b) communication cost.

Figure 2: Performance of *R*-tree range query against raw data size.

CONCLUSION

1. We propose an oblivious query framework (OQF).
2. Our objective is to process range and *k*NN queries with high throughput.
3. The main idea is to integrate indices into ORAM and leverage batch processing and ORAM caching.
4. Extensive experimental evaluation has demonstrated the effectiveness and efficiency of our method.