POPE: Partial Order Preserving Encoding

Daniel S. Roche* Daniel Apon† Seung Geol Choi*
Arkady Yerukhimovich‡

*U.S. Naval Academy
†University of Maryland
‡MIT Lincoln Laboratory

ACM CCS 2016
Vienna, Austria

Problem: Encrypting a Database Index

Cloud databases are popular for many reasons:

- Low cost
- High availability
- High performance
- ...

But these systems are regularly compromised by attackers.
(Consider just voter databases in the last year!)

Challenge: Securing data without compromising performance (too much)

Tradeoffs and Choices

1. **Features**
   (Query support, multi/single user)
2. **Performance**
   (Server time/memory, client time/memory, transfer size, rounds)
3. **Privacy**
   (What might be leaked? What kind of adversary?)

Our Target Features

This work focuses on a common big data scenario:

- Many insertions (should be as fast as possible)
- Fewer lookups or range queries

Data: Key/value store, i.e. all queries on a single column.

Example Dataset

4 million employees, with lookups by salary.
(California public employees database)

This talk

Focusing on many insertions and fewer range queries:

1. Existing approaches, performance/privacy tradeoffs
2. Our construction: POPE
   Provides a new compromise between performance and privacy
3. Evaluation and experiments

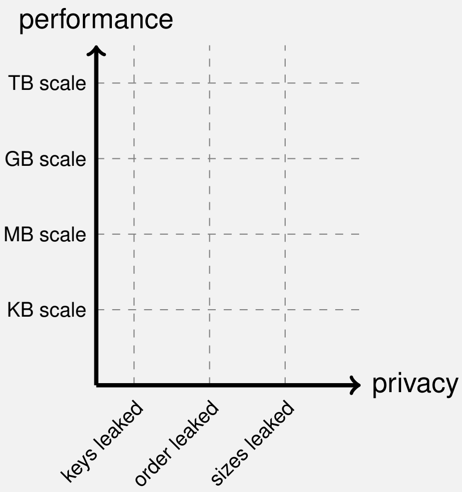
Context of POPE

Our target: Many insertions, few range queries

Current options:

- No encryption
- Traditional OPE
- PPE, ORE, or Interactive OPE
- ORAMs
- Encrypt the entire database

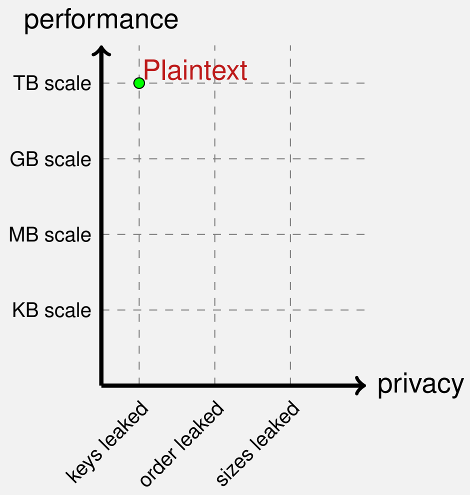
POPE will provide a new compromise in this space



Storing keys in plaintext

Trivial solution:
Store keys in plaintext, encrypt payloads only

Possible with any existing cloud database solution.



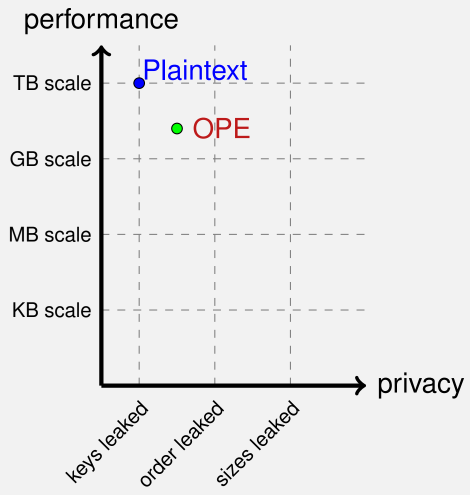
Order-Preserving Encryption (OPE)

Idea:
Can compare keys by comparing ciphertexts.

These schemes are used in industry today!

Hot topic:

- Agrawal et. al.'04
- Baldyreva et. al., '09, '11
- Mavroforakis et. al., '15
- Lewi & Wu '16 (ORE)



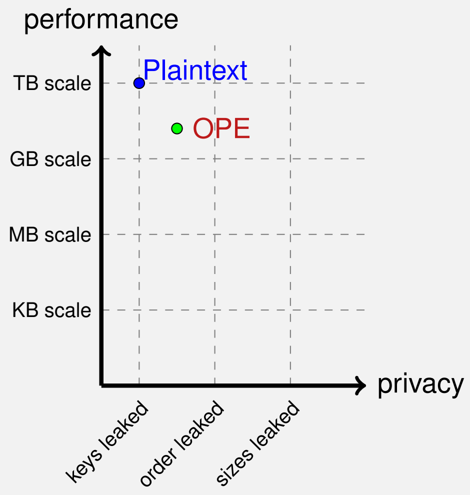
Order-Preserving Encryption (OPE)

Idea:
Can compare keys by comparing ciphertexts.

These schemes are used in industry today!

Hot topic for attacks:

- Baldyreva et. al.'11
- Naveed et. al.'15
- Durak et. al.'16
- Grubbs et. al.'16



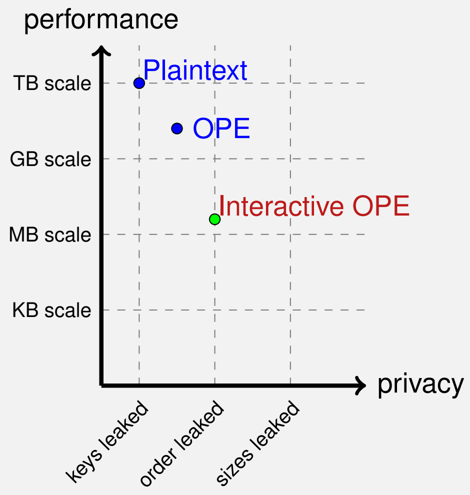
Interactive OPE

Idea:
Use an interactive protocol to compare ciphertexts

Achieves ideal security leaking only the order

- Popa et. al.'13
- Kerschbaum et. al., '14
- Kerschbaum '15
- Boelter '16

(Ideal ORE of Boneh et. al.'15 fits most closely here.)

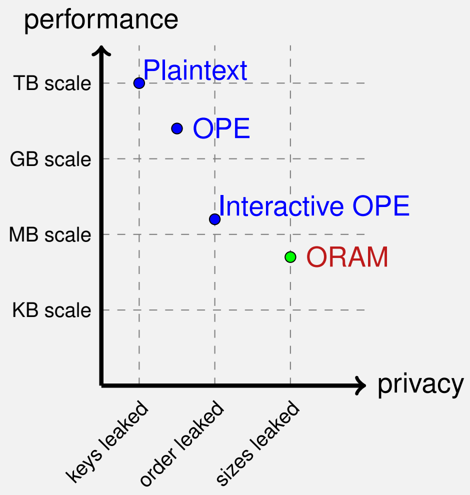


Oblivious RAM (ORAM)

Idea:
Store data structure in an ORAM to hide access patterns

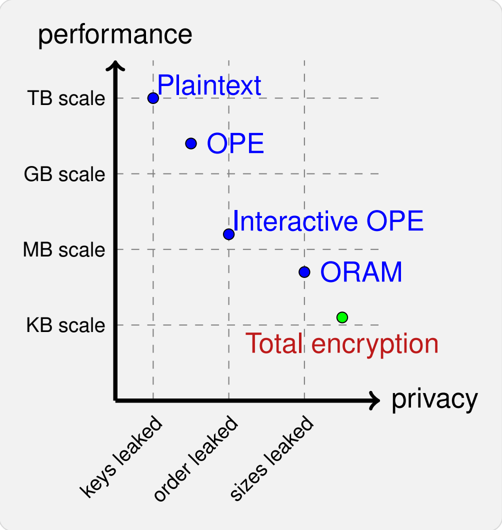
- Goldreich & Ostrovsky '96
- Stefanov et. al. '13
- Wang et. al. '14
- Devadas et. al. '15
- R., Aviv, Choi '16

. . . and many more!



Encrypt the whole thing

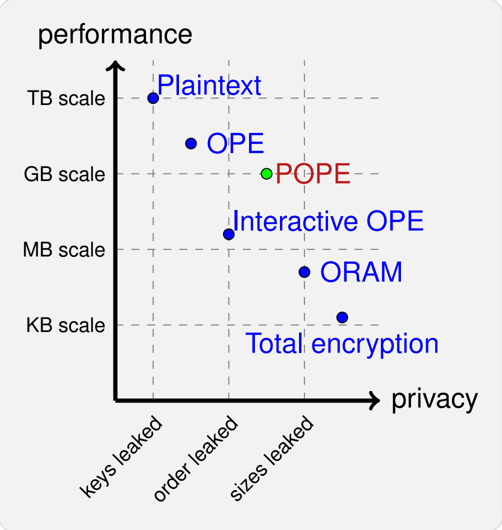
Trivial solution:
Download and re-encrypt the whole database on each access



This talk: *Partial* Order Preserving Encoding

Our idea:
Only perform comparisons necessary to execute the queries.

Improves performance *and* security compared to interative OPE



POPE Data Structure

Main Idea

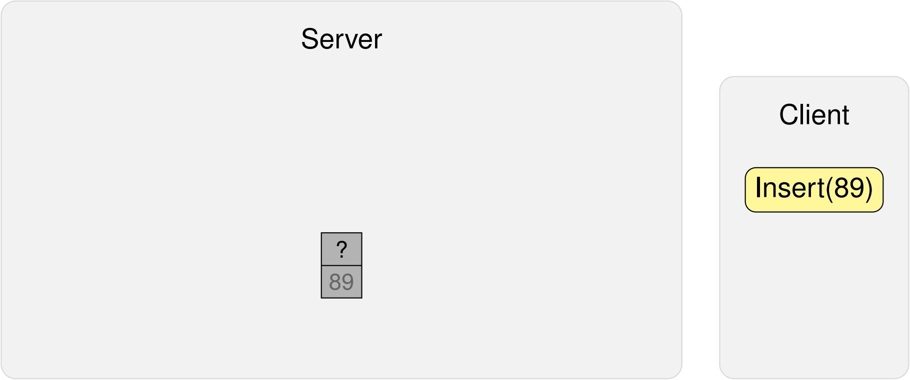
- Server stores a *partially ordered* B-tree
- Every node contains an unordered buffer of key/value pairs
- Non-leaf nodes also have a small ordered list of ciphertexts
- Encryption uses any (randomized) symmetric cipher
- Client performs comparisons at query-time

Influences:

- Buffer trees (Arge '03)
- Mutable OPE (Popa, Li, Zeldovich '13)

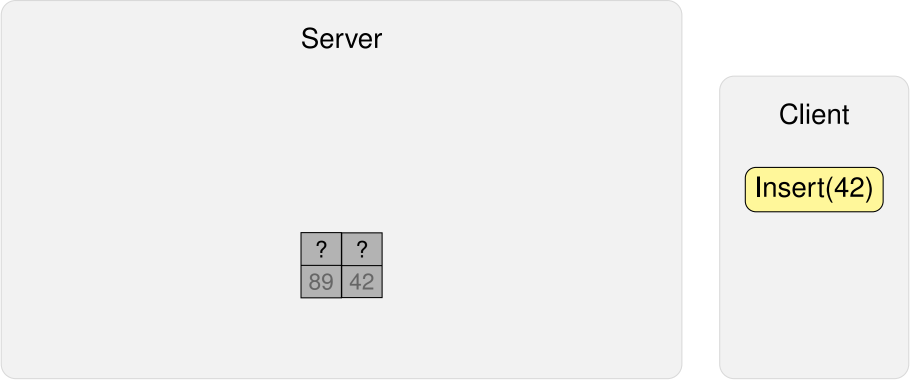
Initial insertions

Inserted ciphertexts are appended (unordered) to the root node.



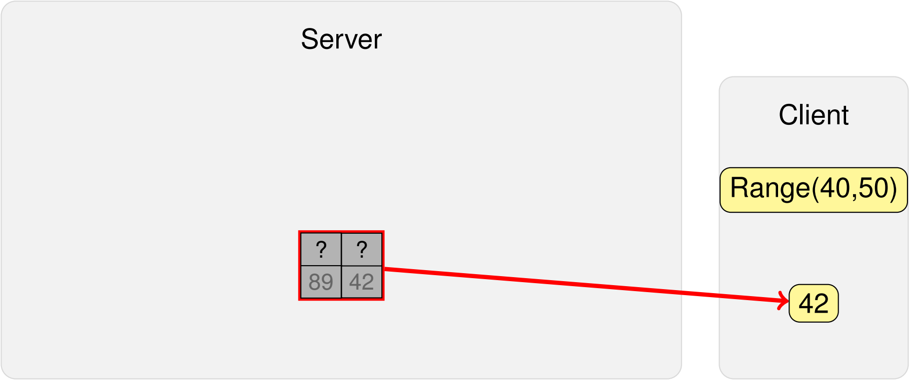
Initial insertions

Inserted ciphertexts are appended (unordered) to the root node.



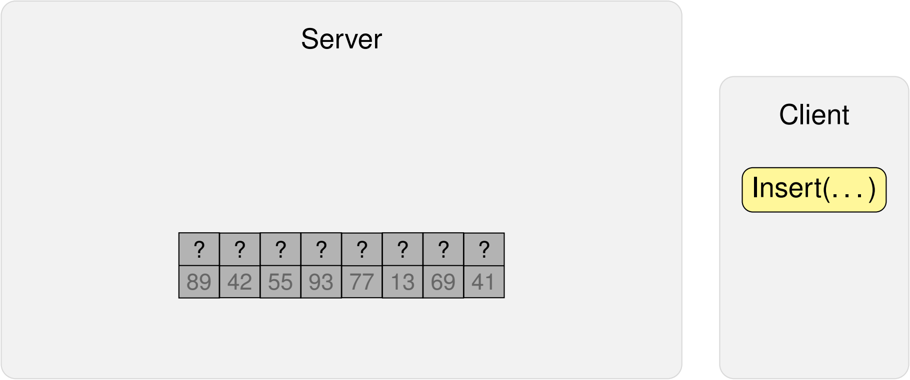
Range search Base case

For a small leaf node, send the entire node to the client.



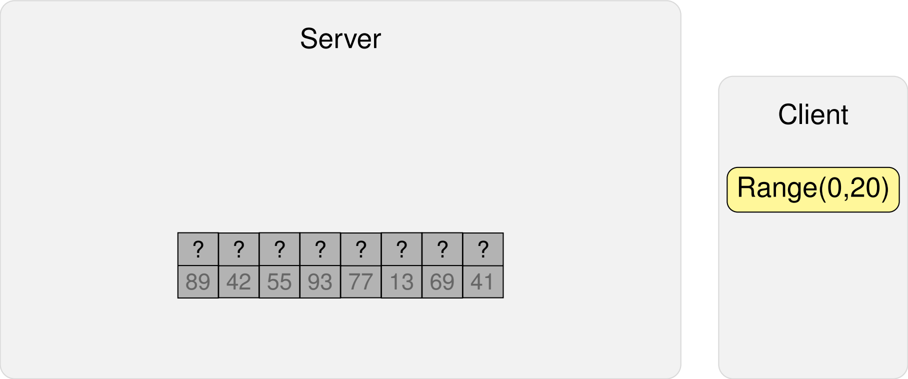
More insertions

Further insertions are appended to the root.



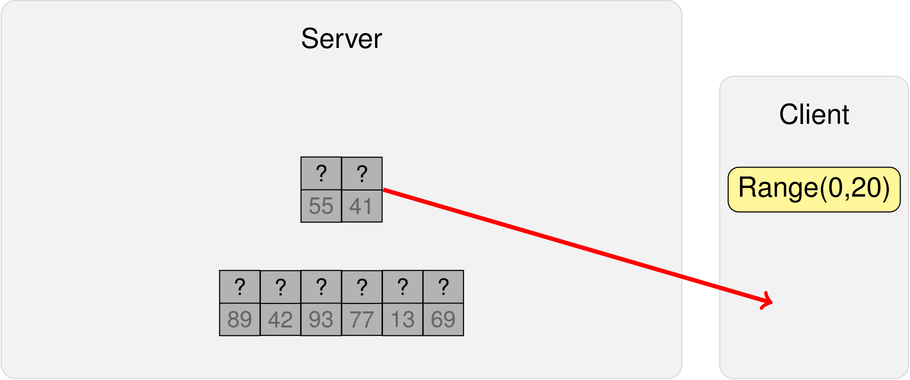
Splitting leaf nodes

Searching a large leaf node requires *splitting*.



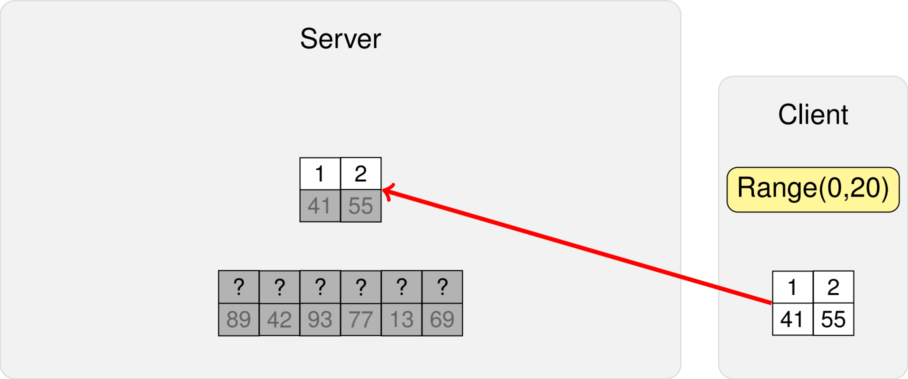
Splitting leaf nodes

1. Server promotes m random items and sends to client.



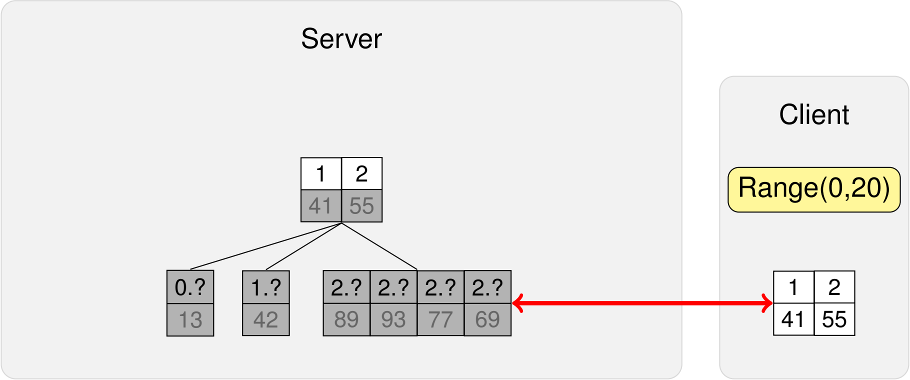
Splitting leaf nodes

2. Client sorts, stores, and remembers the m items.



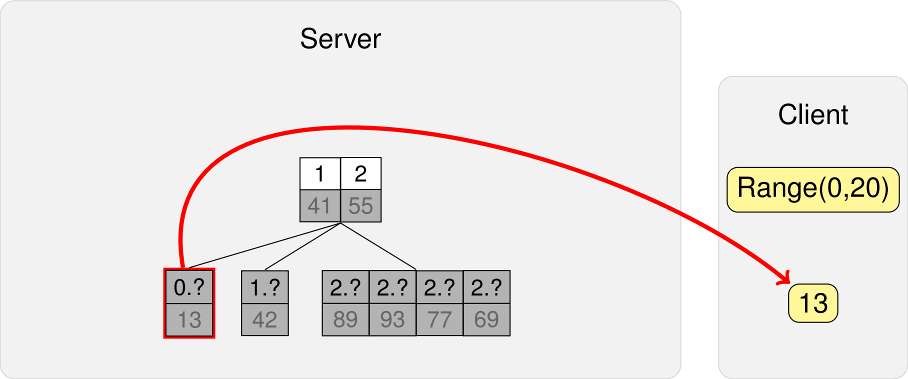
Splitting leaf nodes

3. Client partitions remaining items.



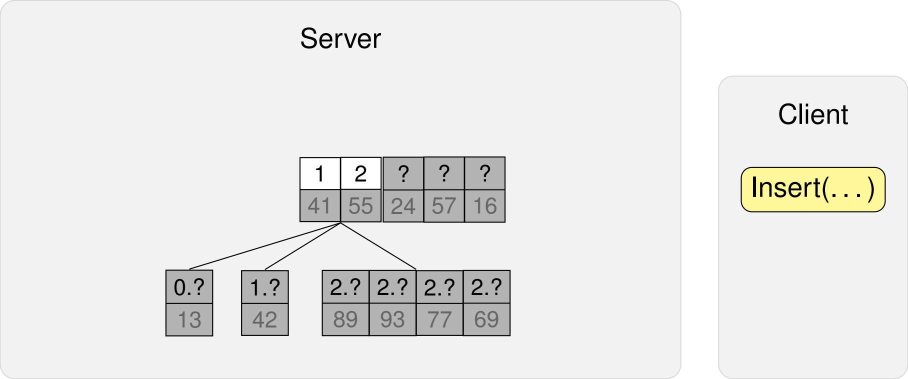
Splitting leaf nodes

4. Finally, the range query results are returned.



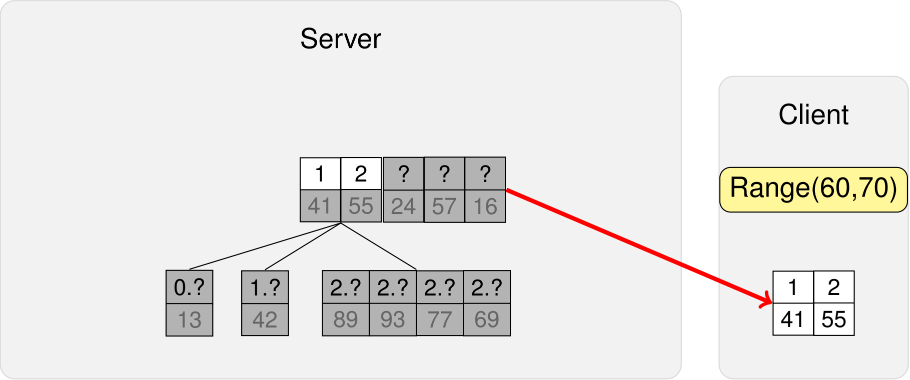
More insertions

Further insertions are appended to the root.



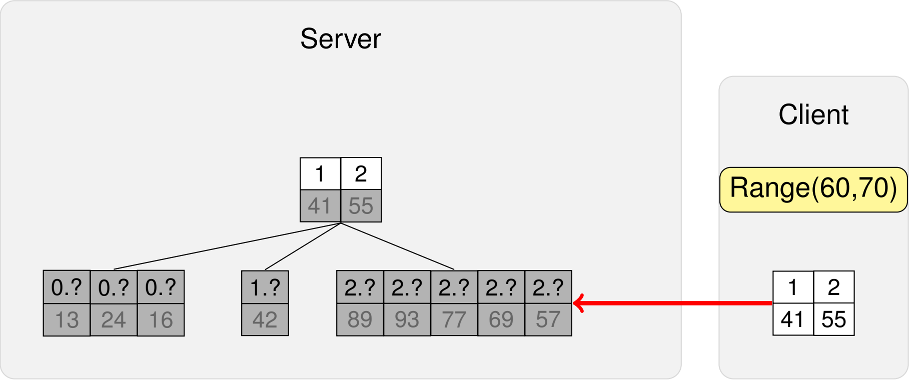
Range query

Queries start by partitioning the root buffer to child nodes.



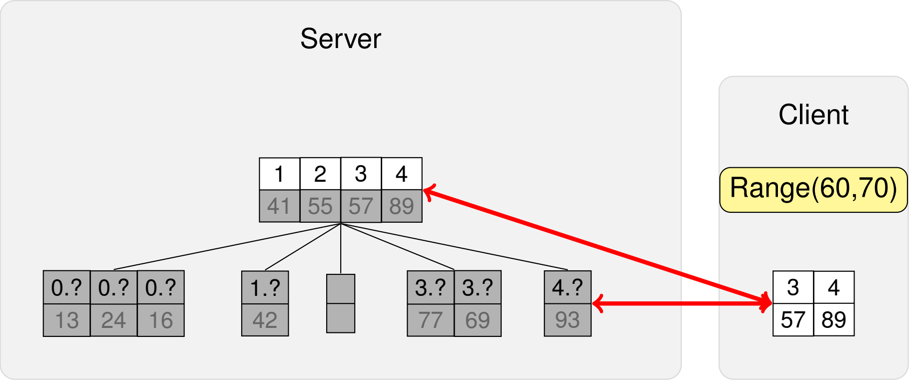
Range query

Queries start by partitioning the root buffer to child nodes.



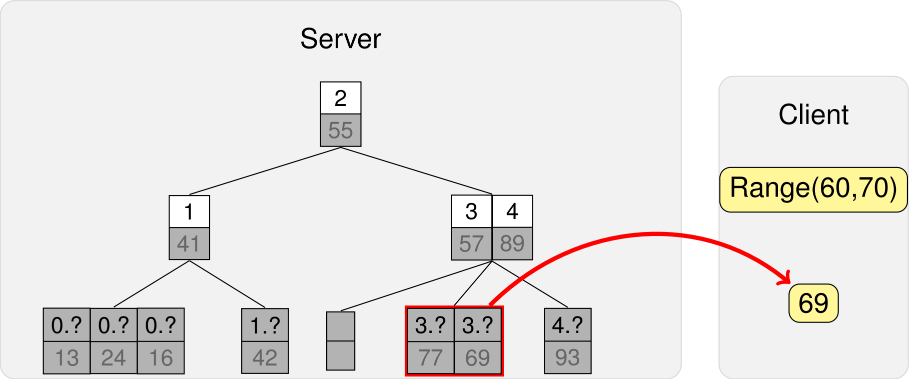
Range query

This may result in further leaf node splits.



Range query

The sorted parts of nodes are not allowed to get too large.



Performance

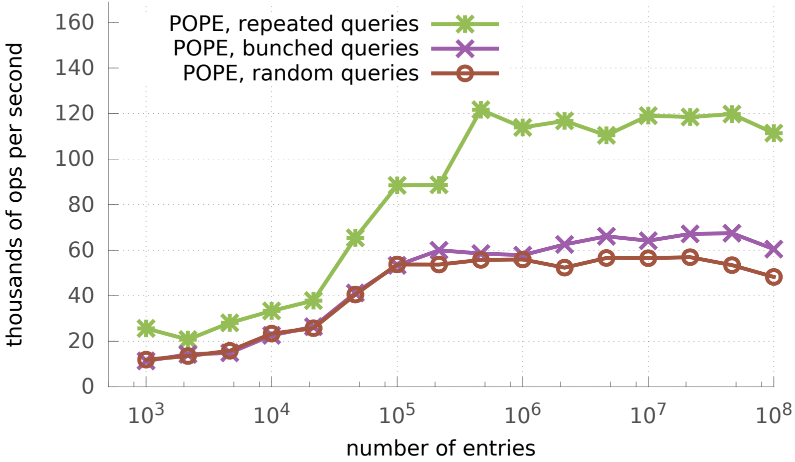
While *some* queries may be costly due to interactive partitioning, the *average* cost per operation is optimal:

Amortized Analysis

The average cost per operation is $O(1)$, and
the worst-case round complexity per operation is $O(1)$, assuming:

- n insertions
- Reasonable client-side temporary storage ($L \in \Omega(n^{O(1)})$)
- Not too many range queries ($m \leq \frac{n}{L}$)

Experimental Performance



Note: Number of queries was $\sqrt{n}$ in all cases.

POPE Security

Server cannot learn more than the order of the keys.
(IND-OCPA notion of Boldyreva et. al. '11, achieved by Popa et. al. '13)

Tie-breaking randomness hides key frequencies also.
(IND-FAOCPA of Kerschbaum '15)

Only a partial order is leaked.
Under previous assumptions of n insertions, m queries and client storage L, the relative order between at least

$$\Omega\left(\frac{n^2}{mL} - n\right)$$

pairs of elements is not revealed.

Thanks!

The Paper

Daniel S. Roche, Daniel Apon, Seung Geol Choi,
and Arkady Yerukhimovich
"POPE: Partial Order Preserving Encoding"
`https://arxiv.org/abs/1610.04025`

Code: `https://github.com/dsroche/pope`

Thanks to: National Science Foundation, Office of Naval Research