School of
Information Systems

SMU
SINGAPORE MANAGEMENT UNIVERSITY

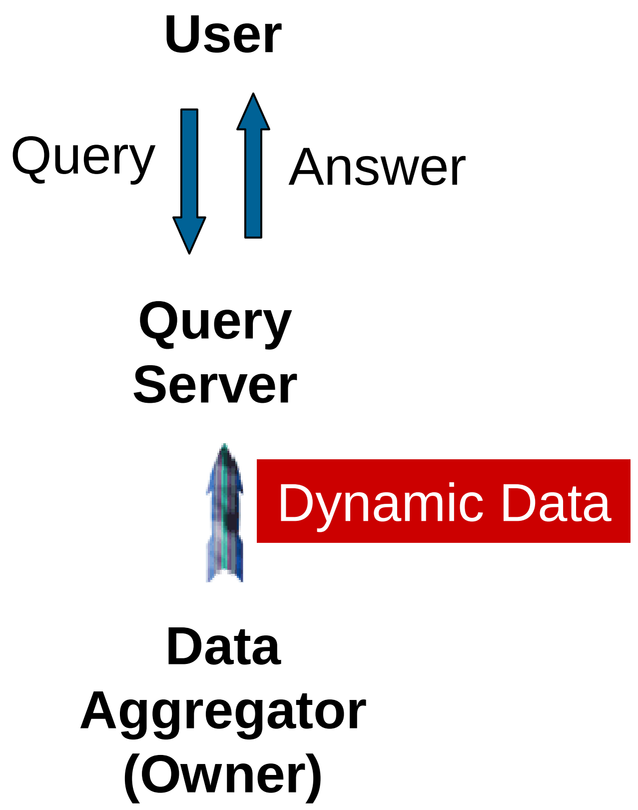
Scalable Verification for Outsourced Dynamic Databases

HweeHwa Pang, Jilian Zhang and Kyriakos Mouratidis

School of Information Systems
Singapore Management University

VLDB 2009

What is the Problem?



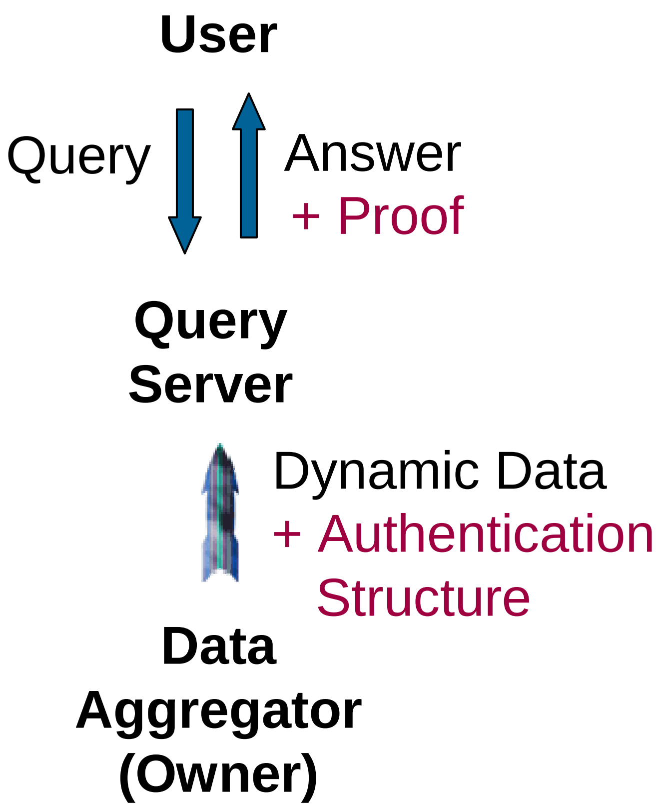
System Model

- Data Aggregator outsources query processing function to the Query Server

Threat Model

- User trusts the Data Aggregator
- Query Server is not untrusted
  - Compromised by hackers
  - Insider attack

Is the Query Answer *"Correct"*?



- Authenticity

  Every value in the answer originated from the Data Aggregator

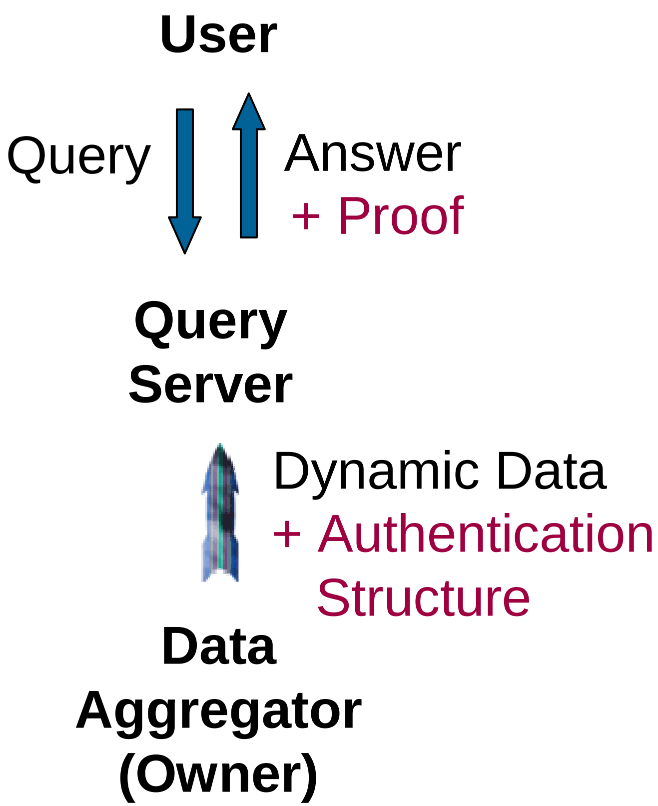
- Completeness

  Every record the satisfies the query condition is in the answer

- ***Freshness***

  The record values in the answer are up-to-date

Authentication Approaches



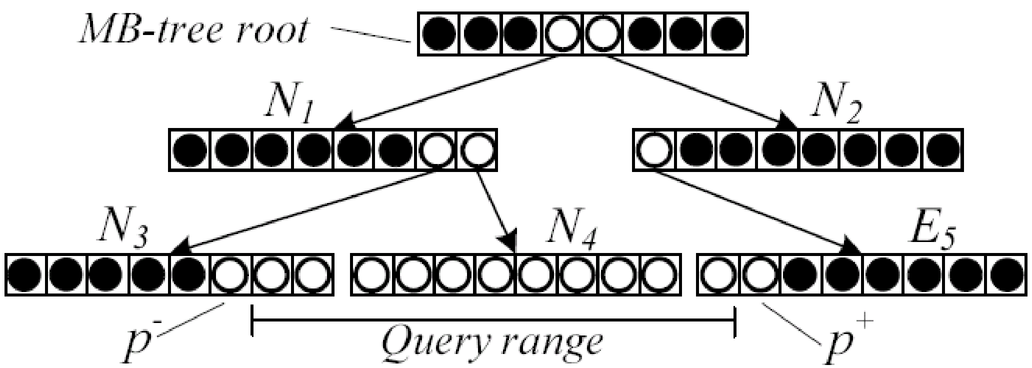
<u>Merkle Hash Tree (MHT)</u>

- Merkle, Crypto 2008
- Devanbu et al, JCS 2003
- Nuckolls et al, DBSec 2005
- Li et al, SIGMOD 2006
- Papadopoulos et al, VLDB 2007
- Goodrich et al, CT-RSA 2008
- …

<u>Signature Aggregation</u>

- Boneh et al, CryptoBytes 2003
- Pang et al, SIGMOD 2005
- Narasimha et al, ACM TOS 2006
- Cheng et al, DBSec 2006
- …

Merkle Hash Tree Approach



(from Li et al, SIGMOD 2006)

Structure

- Embed authentication information in data index
- Node digest is a hash function on node content
- Root digest is certified

Response to query

- Answer = records within query range
- Proof = ● digests along left & right boundaries

User verification

- Compute the ○ digests from records in the answer
- Check ○○○ … + ●●● … = certified root?

Limitations of MHT Approach

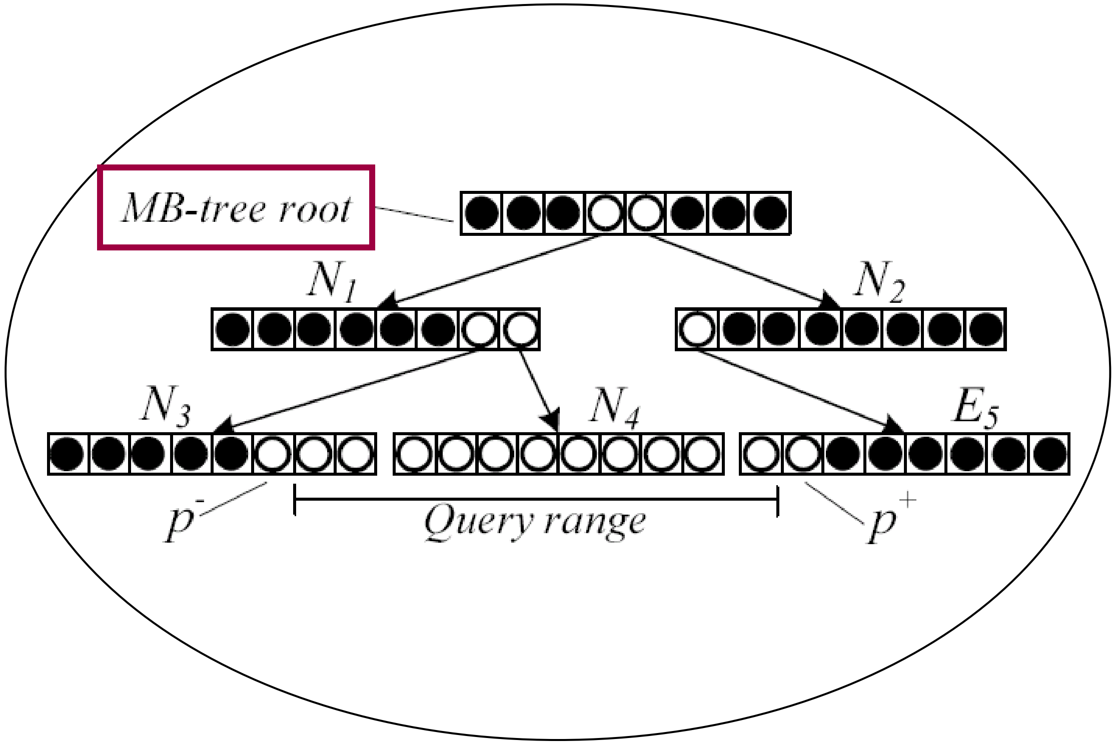
User

Query | Answer + Proof

Query Server

Dynamic Data + Authentication Structure

Data Aggregator (Owner)

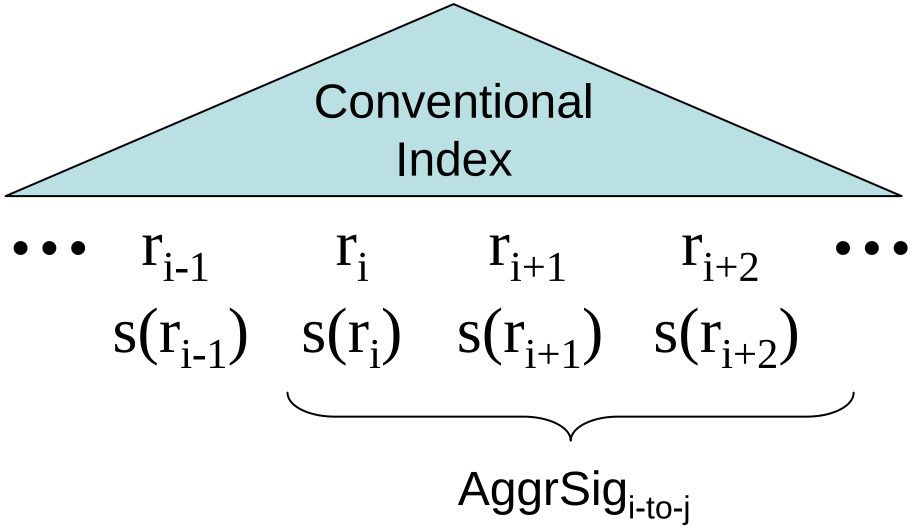


- Queries S-lock the root digest
- Updates X-lock the root digest

⇒ *Locking bottleneck!*

⇒ *Log N update I/Os*

Signature Aggregation Approach

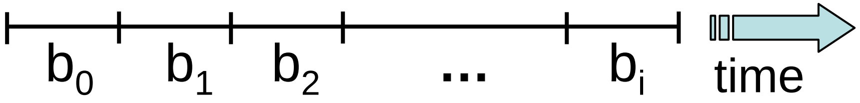


- s(r) = signature of record r
- Suppose answer = $\{r_i, \ldots, r_j\}$
- $AggrSig_{i\text{-to-}j}$ = dynamically computed aggregate on $s(r_i), \ldots, s(r_j)$
- Records can be updated *concurrently*
- Caveat: Signature aggregation operations are much more computationally intensive than hashing function

Challenges with Signature Aggregation

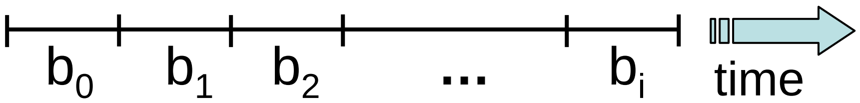
1. How to detect old records and signatures?
   - Not practical to recertify all the records frequently
   - How to invalidate outdated signatures?
2. How to authenticate ad-hoc joins?
3. How to mitigate the much higher computation costs of signature aggregation?

Freshness Protocol



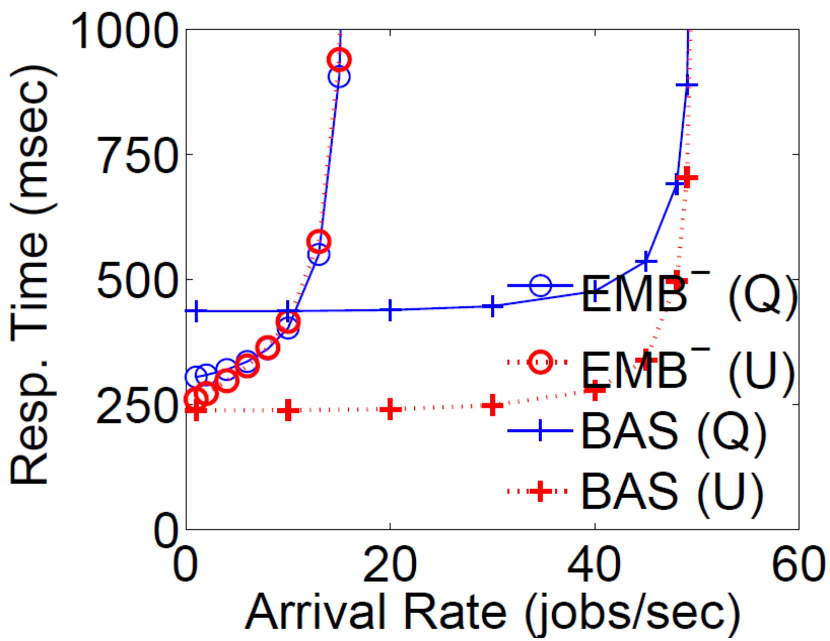
- Data Aggregator releases new record values and signatures to query server and user *immediately*
- Data Aggregator issues a certified bitmap b_i after each time period *i*
- b_i = 0100…110… means that:
  - No change to record 0 in period *i*
  - Record 1 was recertified in period *i*
- Bitmaps are sparse
  - After compression, size is 2 to 3 times the number of 1 bits
  - Proportional to update rate, independent of database size

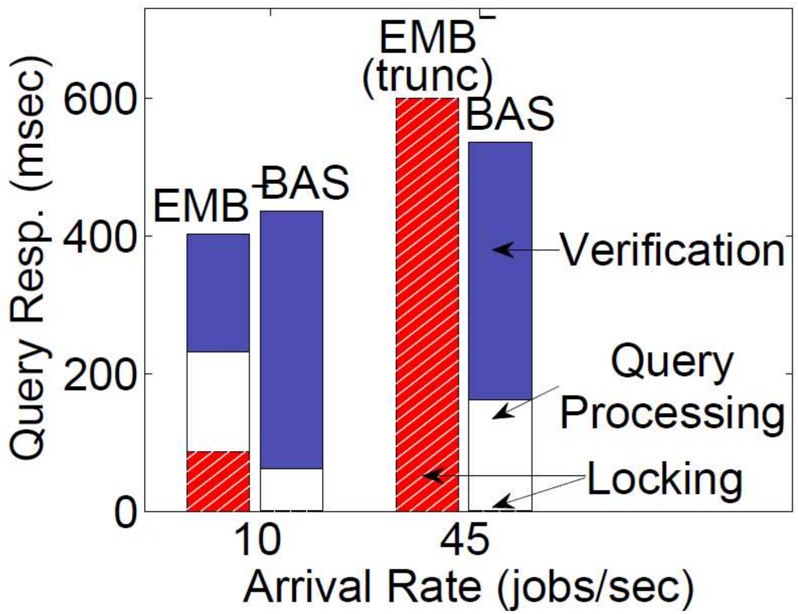
Freshness Check



- Suppose every period lasts ρ seconds, current period is *k*
- A record *j* signed in period *i* is:
  - Current, if $i < k$ and bit *j* is 0 in $b_{i+1}, b_{i+2}, \ldots, b_k$
  - Out of date by at most ρ seconds, if $i = k$
- Details in the paper ...
  - Multiple updates to the same record in a period
  - Background signature renewal to limit the number of bitmaps to be checked
  - Methods for transmitting bitmaps to users, to reduce delays

Experiment Results



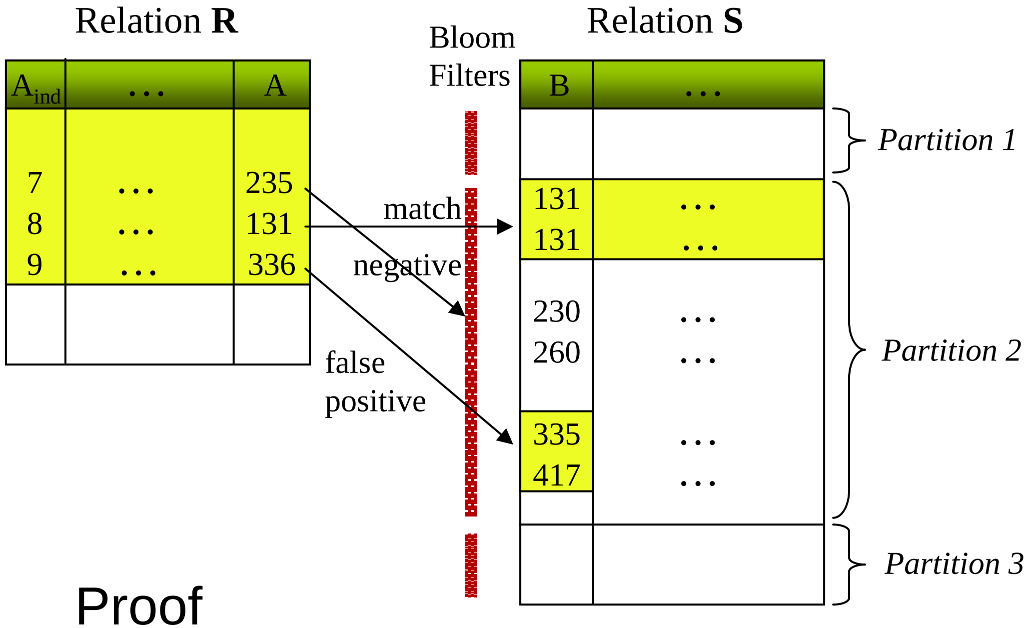


- Database contains 1 million randomly generated records
- Query/update selectivity = 1000 records
- 9 queries to 1 update

Challenges with Signature Aggregation

1. How to detect old records and signatures?
2. How to authenticate ad-hoc join $\mathbf{R} \bowtie_{\mathbf{R}.A=\mathbf{S}.B} \mathbf{S}$?
   - Prove R
   - For record $r \in$ **R** such that $\exists s \in$ **S** with $r.A = s.B$, prove $\sigma_{B=r.A}(\mathbf{S})$
   - How to prove that an *r* record has no match in **S**?
   - Return consecutive $s_1, s_2 \in$ **S** such that $s_1.B < r.A < s_2.B$
   - *Expensive!!*
3. How to mitigate the much higher computation costs of signature aggregation?

Authenticated Ad-Hoc Join $\mathbf{R} \bowtie_{\mathbf{R}.A=\mathbf{S}.B} \mathbf{S}$



- Partition **S** horizontally
- Build a certified Bloom filter $\mathcal{BF}$ on $s.B$ per **S** partition
- $\mathcal{BF}(b)$ = false $\Rightarrow$ b is not in **S**.B (no false negative)
- $\mathcal{BF}(b)$ = true $\Rightarrow$ b is very likely to be in **S**.B (low false positive)

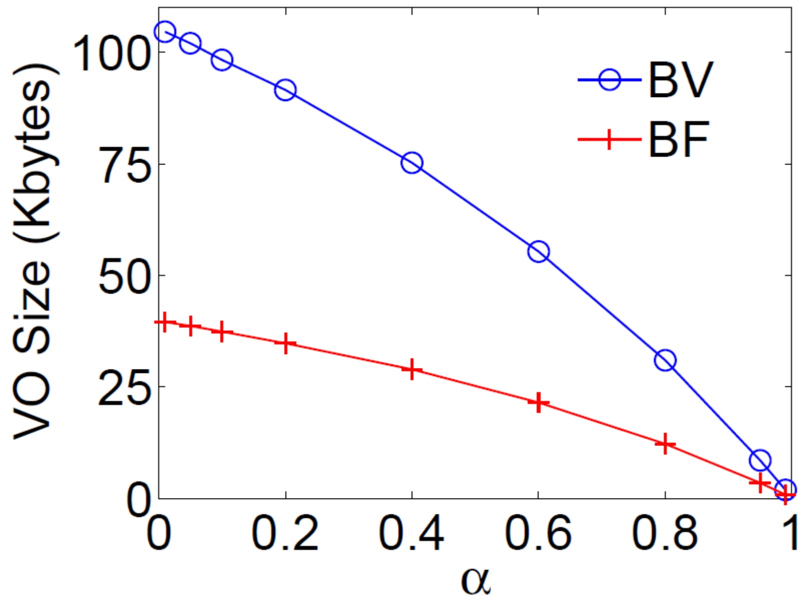
Proof

- Return Bloom filter $\mathcal{BF}$ and partition boundaries if $\mathcal{BF}(r.A)$ = false for some $r \in$ **R**
- Return consecutive $s_1, s_2 \in$ **S** for $r \in$ **R** that has no match in **S**, but $\mathcal{BF}(r.A)$ = true

Details in the paper …

- **S** partition size: proof size versus Bloom filter update cost
- Bloom filter size

Experiment Results

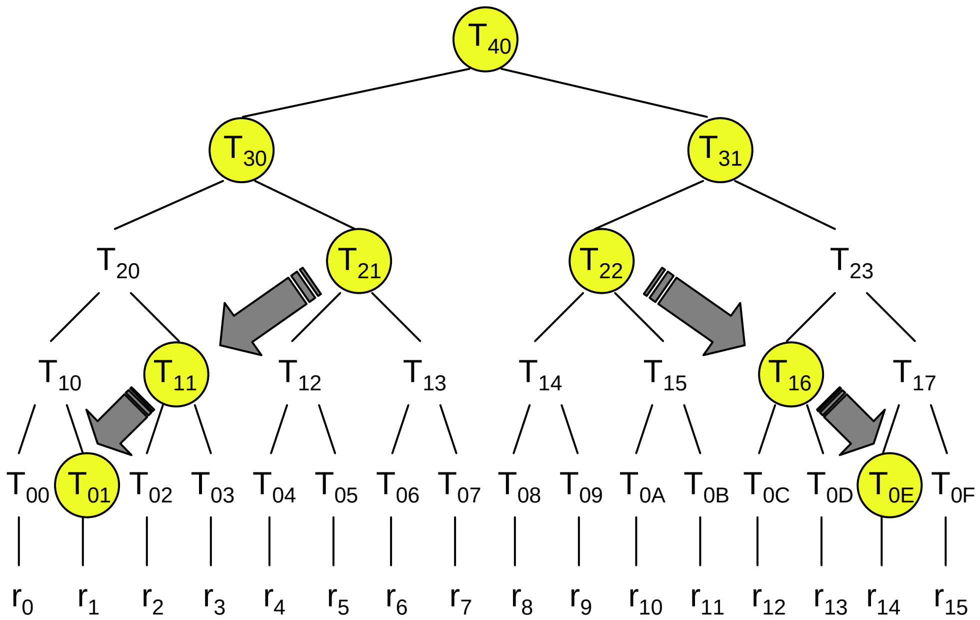


- α = ratio of **R** records with matching **S** records
- BV = always return Boundary Values to prove unmatched **R** records
- BF = use Bloom filters to prove unmatched **R** records

Challenges with Signature Aggregation

1. How to detect old records and signatures?
2. How to authenticate ad-hoc joins?
3. How to mitigate the much higher computation costs of signature aggregation?
    - Aggregating 1000 signatures takes 9 ms (compared to less than 3 μs per hash operation)
    - The Query Server is a shared resource
    - To reduce the number of signature aggregations, can we cache certain “high-utility” aggregates?

Signature Caching



- Imagine a hierarchy of aggregate signatures
- Low-level aggregates are more likely to be reused
- High-level aggregates give more savings when used
- Left-half is symmetrical with right-half

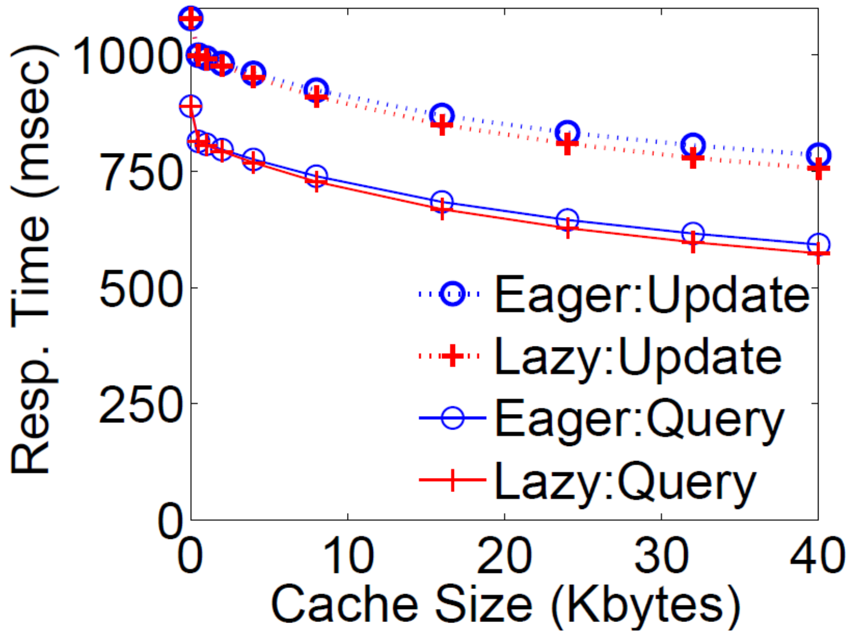
Observations

- Within any level, the 2nd node from the left/right is applicable to the widest range of query cardinalities
- Highest-utility aggregates: 2nd node from level 3 downloads

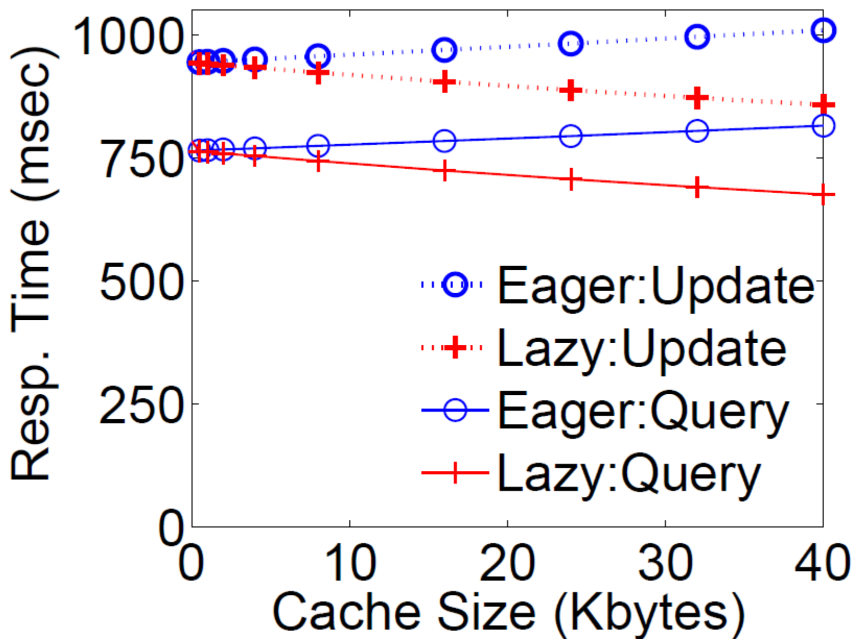
Details in the paper …

- Derivation
- Eager versus Lazy refresh when the underlying signatures change

Experiment Results



(a) 1 update to 9 queries



(b) 4 updates to 6 queries

- Database contains 1 million randomly generated records
- Query selectivity = 1000 records
- Arrival rate = 50 jobs/second

Conclusion

- For outsourced dynamic databases, the criteria for query answer correctness include authenticity, completeness and *freshness*
- The scalability of Merkle hash tree approaches is limited by lock contention
- We introduced a scalable verification protocol, based on signature aggregation approach, that supports freshness guarantee
- We proposed an authentication mechanism for ad-hoc joins
- We demonstrated that caching a small number of aggregate signatures can significantly help the query server